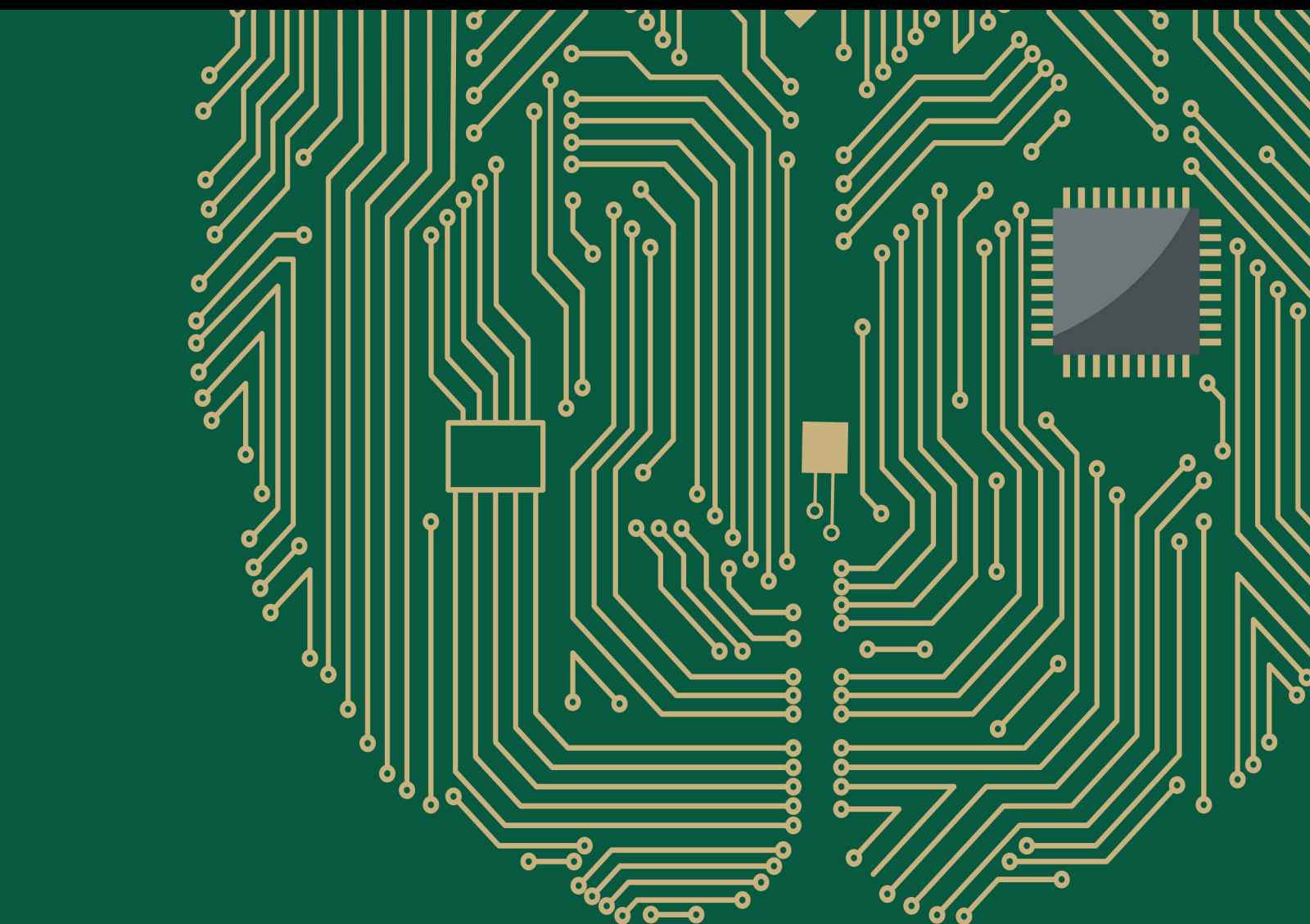
Computational Intelligence and Neuroscience

Recent Developments in Deep Learning for Engineering Applications

Lead Guest Editor: Athanasios Voulodimos
Guest Editors: Nikolaos Doulamis, George Bebis, and Tania Stathaki



Recent Developments in Deep Learning for Engineering Applications

Computational Intelligence and Neuroscience

Recent Developments in Deep Learning for Engineering Applications

Lead Guest Editor: Athanasios Voulodimos
Guest Editors: Nikolaos Doulamis, George Bebis, and Tania Stathaki

Editorial Board

Contents

Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 8141259, 2 pages
https://doi.org/10.1155/2018/8141259

Editorial

Recent Developments in Deep Learning for Engineering Applications

Athanasios Voulodimos,[1] **Nikolaos Doulamis**,[2] **George Bebis**,[3] **and Tania Stathaki**[4]

[1] *University of West Attica, Athens, Greece*
[2] *National Technical University of Athens, Athens, Greece*
[3] *University of Nevada, Reno, NV, USA*
[4] *Imperial College London, London, UK*

Correspondence should be addressed to Athanasios Voulodimos; thanosv@mail.ntua.gr

Received 8 April 2018; Accepted 8 April 2018; Published 10 May 2018

Deep learning is among the most rapidly growing fields enabling computational models of multiple processing layers to learn and represent data with multiple levels of abstraction thus implicitly capturing intricate structures of large-scale data. The recent surge of interest in deep learning methods is mainly due to the abundance of complex data from different sources (visual, medical, social, and sensor) and in a variety of application domains, but also due to the fact that deep learning approaches have been shown to significantly outperform previous state-of-the-art techniques in several research problems.

The aim of this Special Issue is to present new academic research advances and industrial developments of machine learning with emphasis on deep learning for engineering applications. We received twenty-four (24) submissions and each paper was reviewed by at least two experts. Nine (9) papers were accepted for publication in this Special Issue, covering a variety of topics.

In their paper, J. Kim et al. propose a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) based deep learning framework for energy disaggregation. The conducted experiments on two public datasets (UK-DALE and REDD) indicate that the proposed framework achieves higher performance rates compared to popular Nonintrusive Load Monitoring (NILM) techniques (including Hidden Markov Model based frameworks).

Q. Lan et al. present an optimized GPU implementation of 3D Convolutional Neural Networks and apply it in a video classification scenario. Comparisons of their approach with cuDNN suggest a significant speedup achieved by the former approach.

Focusing on potential users of brain-computer interfaces (BCI), E. Carabez et al. propose a Convolutional Neural Network (CNN) based architecture for identification and classification of brain activity elicited as P300 waves. The CNN models used are given a novel single trial three-dimensional (3D) representation of the electroencephalogram (EEG) data as input, maintaining temporal and spatial information close to the experimental setup. The derived results show increased accuracy for the proposed architecture and 3D input in single trial P300 classification compared to other approaches in the literature.

In another paper of this collection, Y. Feng et al. present a combined deep learning and reinforcement learning framework for entity and relation extraction from unstructured text. The proposed approach combines bidirectional LSTM, tree-LSTM, and Q-learning technique in a two-step decision process and attains higher recall and accuracy rates compared to state-of-the-art methods on publicly available textual datasets for automatic content extraction.

The application of deep learning for disease estimation on plant images is addressed in the paper of G. Wang et al. The authors propose a series of Convolutional Neural Networks fine-tuned via transfer learning techniques, trained to recognize the severity level of disease on plant images, thus forming a framework that yields promising results.

Convolutional Neural Networks are also an integral part of the framework proposed by J. Feng et al. for geographical scene classification. The described model aims at optimizing the feature extractor, so that it can learn more suitable visual vocabularies from geotagging images, resulting in higher recognition rates than other approaches in the literature.

A brief overview of the most important techniques and applications of deep learning in the field of computer vision is provided in the review paper of A. Voulodimos et al. The authors provide a short presentation and comparative analysis of three major deep learning schemes used in computer vision, that is, Convolutional Neural Networks (CNNs), Deep Belief Networks (DBNs), and Deep Boltzmann Machines (DBMs) and Stacked Autoencoders, and their applications on major computer vision problems including object detection, action and activity recognition, face recognition, and human pose estimation.

Another two papers of the Special Issue also present interesting and innovative achievements in machine learning for engineering, though not strictly following a deep learning approach. X. Zhao et al. propose a strategy for achieving high accuracy in synthetic aperture radar (SAR) automatic target recognition (ATR) tasks. Their approach includes novel pose rectification and image normalization to produce images with less variations, followed by feature extraction using wavelet coefficients. A group of discrimination trees are learned and combined into a final classifier in the framework of Real-AdaBoost. The efficacy of the proposed framework is demonstrated through extensive experiments with the public release database MSTAR.

Finally, X. Nie et al. propose the Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm as an improvement of the Cat Swarm Optimization (CSO) algorithm. The enhanced version addresses one of the problems of CSO, that is, the fact that it is often trapped in local optima in nonlinear optimization problems with a large number of local extreme values. The authors apply their proposed algorithm in a photovoltaic MPPT application scenario.

Athanasios Voulodimos
Nikolaos Doulamis
George Bebis
Tania Stathaki

Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 7068349, 13 pages
https://doi.org/10.1155/2018/7068349

Review Article

Deep Learning for Computer Vision: A Brief Review

Athanasios Voulodimos,[1,2] **Nikolaos Doulamis**,[2] **Anastasios Doulamis**,[2] **and Eftychios Protopapadakis**[2]

[1]*Department of Informatics, Technological Educational Institute of Athens, 12210 Athens, Greece*
[2]*National Technical University of Athens, 15780 Athens, Greece*

Correspondence should be addressed to Athanasios Voulodimos; thanosv@mail.ntua.gr

Received 17 June 2017; Accepted 27 November 2017; Published 1 February 2018

Academic Editor: Diego Andina

Over the last years deep learning methods have been shown to outperform previous state-of-the-art machine learning techniques in several fields, with computer vision being one of the most prominent cases. This review paper provides a brief overview of some of the most significant deep learning schemes used in computer vision problems, that is, Convolutional Neural Networks, Deep Boltzmann Machines and Deep Belief Networks, and Stacked Denoising Autoencoders. A brief account of their history, structure, advantages, and limitations is given, followed by a description of their applications in various computer vision tasks, such as object detection, face recognition, action and activity recognition, and human pose estimation. Finally, a brief overview is given of future directions in designing deep learning schemes for computer vision problems and the challenges involved therein.

1. Introduction

Deep learning allows computational models of multiple processing layers to learn and represent data with multiple levels of abstraction mimicking how the brain perceives and understands multimodal information, thus implicitly capturing intricate structures of large-scale data. Deep learning is a rich family of methods, encompassing neural networks, hierarchical probabilistic models, and a variety of unsupervised and supervised feature learning algorithms. The recent surge of interest in deep learning methods is due to the fact that they have been shown to outperform previous state-of-the-art techniques in several tasks, as well as the abundance of complex data from different sources (e.g., visual, audio, medical, social, and sensor).

The ambition to create a system that simulates the human brain fueled the initial development of neural networks. In 1943, McCulloch and Pitts [1] tried to understand how the brain could produce highly complex patterns by using interconnected basic cells, called neurons. The McCulloch and Pitts model of a neuron, called a MCP model, has made an important contribution to the development of artificial neural networks. A series of major contributions in the field is presented in Table 1, including LeNet [2] and Long Short-Term Memory [3], leading up to today's "era of deep learning." One of the most substantial breakthroughs in deep learning came in 2006, when Hinton et al. [4] introduced the Deep Belief Network, with multiple layers of Restricted Boltzmann Machines, greedily training one layer at a time in an unsupervised way. Guiding the training of intermediate levels of representation using unsupervised learning, performed locally at each level, was the main principle behind a series of developments that brought about the last decade's surge in deep architectures and deep learning algorithms.

Among the most prominent factors that contributed to the huge boost of deep learning are the appearance of large, high-quality, publicly available labelled datasets, along with the empowerment of parallel GPU computing, which enabled the transition from CPU-based to GPU-based training thus allowing for significant acceleration in deep models' training. Additional factors may have played a lesser role as well, such as the alleviation of the vanishing gradient problem owing to the disengagement from saturating activation functions (such as hyperbolic tangent and the logistic function), the proposal

Table 1: Important milestones in the history of neural networks and machine learning, leading up to the era of deep learning.

Milestone/contribution	Contributor, year
MCP model, regarded as the ancestor of the Artificial Neural Network	McCulloch & Pitts, 1943
Hebbian learning rule	Hebb, 1949
First perceptron	Rosenblatt, 1958
Backpropagation	Werbos, 1974
Neocognitron, regarded as the ancestor of the Convolutional Neural Network	Fukushima, 1980
Boltzmann Machine	Ackley, Hinton & Sejnowski, 1985
Restricted Boltzmann Machine (initially known as Harmonium)	Smolensky, 1986
Recurrent Neural Network	Jordan, 1986
Autoencoders	Rumelhart, Hinton & Williams, 1986 Ballard, 1987
LeNet, starting the era of Convolutional Neural Networks	LeCun, 1990
LSTM	Hochreiter & Schmidhuber, 1997
Deep Belief Network, ushering the "age of deep learning"	Hinton, 2006
Deep Boltzmann Machine	Salakhutdinov & Hinton, 2009
AlexNet, starting the age of CNN used for ImageNet classification	Krizhevsky, Sutskever, & Hinton, 2012

of new regularization techniques (e.g., dropout, batch normalization, and data augmentation), and the appearance of powerful frameworks like TensorFlow [5], theano [6], and mxnet [7], which allow for faster prototyping.

Deep learning has fueled great strides in a variety of computer vision problems, such as object detection (e.g., [8, 9]), motion tracking (e.g., [10, 11]), action recognition (e.g., [12, 13]), human pose estimation (e.g., [14, 15]), and semantic segmentation (e.g., [16, 17]). In this overview, we will concisely review the main developments in deep learning architectures and algorithms for computer vision applications. In this context, we will focus on three of the most important types of deep learning models with respect to their applicability in visual understanding, that is, Convolutional Neural Networks (CNNs), the "Boltzmann family" including Deep Belief Networks (DBNs) and Deep Boltzmann Machines (DBMs) and Stacked (Denoising) Autoencoders. Needless to say, the current coverage is by no means exhaustive; for example, Long Short-Term Memory (LSTM), in the category of Recurrent Neural Networks, although of great significance as a deep learning scheme, is not presented in this review, since it is predominantly applied in problems such as language modeling, text classification, handwriting recognition, machine translation, speech/music recognition, and less so in computer vision problems. The overview is intended to be useful to computer vision and multimedia analysis researchers, as well as to general machine learning researchers, who are interested in the state of the art in deep learning for computer vision tasks, such as object detection and recognition, face recognition, action/activity recognition, and human pose estimation.

The remainder of this paper is organized as follows. In Section 2, the three aforementioned groups of deep learning model are reviewed: Convolutional Neural Networks, Deep Belief Networks and Deep Boltzmann Machines, and Stacked Autoencoders. The basic architectures, training processes, recent developments, advantages, and limitations of each group are presented. In Section 3, we describe the contribution of deep learning algorithms to key computer vision tasks, such as object detection and recognition, face recognition, action/activity recognition, and human pose estimation; we also provide a list of important datasets and resources for benchmarking and validation of deep learning algorithms. Finally, Section 4 concludes the paper with a summary of findings.

2. Deep Learning Methods and Developments

2.1. Convolutional Neural Networks. Convolutional Neural Networks (CNNs) were inspired by the visual system's structure, and in particular by the models of it proposed in [18]. The first computational models based on these local connectivities between neurons and on hierarchically organized transformations of the image are found in Neocognitron [19], which describes that when neurons with the same parameters are applied on patches of the previous layer at different locations, a form of translational invariance is acquired. Yann LeCun and his collaborators later designed Convolutional Neural Networks employing the error gradient and attaining very good results in a variety of pattern recognition tasks [20–22].

A CNN comprises three main types of neural layers, namely, (i) convolutional layers, (ii) pooling layers, and (iii) fully connected layers. Each type of layer plays a different role. Figure 1 shows a CNN architecture for an object detection in image task. Every layer of a CNN transforms the input volume to an output volume of neuron activation, eventually leading to the final fully connected layers, resulting in a mapping of the input data to a 1D feature vector. CNNs have been extremely successful in computer vision applications, such as face recognition, object detection, powering vision in robotics, and self-driving cars.

(i) Convolutional Layers. In the convolutional layers, a CNN utilizes various kernels to convolve the whole image as

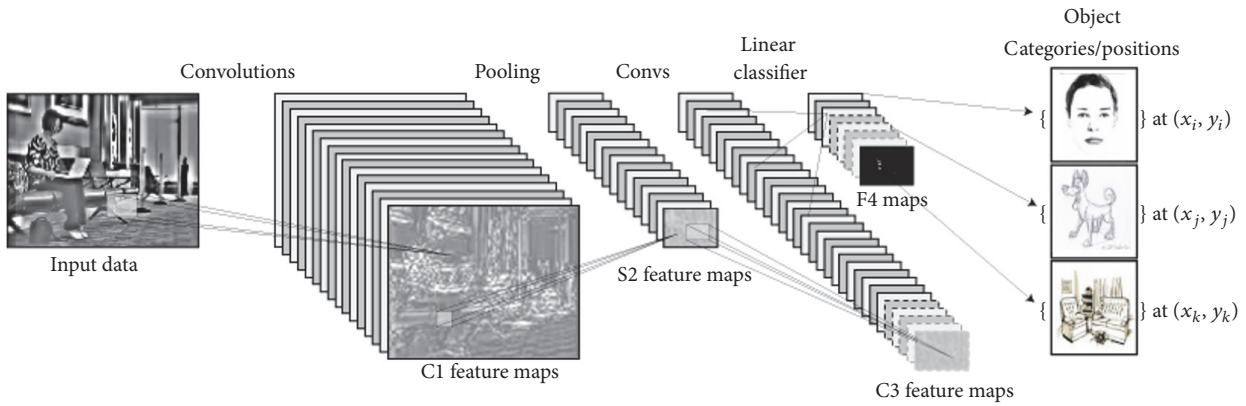


Figure 1: Example architecture of a CNN for a computer vision task (object detection).

well as the intermediate feature maps, generating various feature maps. Because of the advantages of the convolution operation, several works (e.g., [23, 24]) have proposed it as a substitute for fully connected layers with a view to attaining faster learning times.

(ii) Pooling Layers. Pooling layers are in charge of reducing the spatial dimensions (width × height) of the input volume for the next convolutional layer. The pooling layer does not affect the depth dimension of the volume. The operation performed by this layer is also called subsampling or downsampling, as the reduction of size leads to a simultaneous loss of information. However, such a loss is beneficial for the network because the decrease in size leads to less computational overhead for the upcoming layers of the network, and also it works against overfitting. Average pooling and max pooling are the most commonly used strategies. In [25] a detailed theoretical analysis of max pooling and average pooling performances is given, whereas in [26] it was shown that max pooling can lead to faster convergence, select superior invariant features, and improve generalization. Also there are a number of other variations of the pooling layer in the literature, each inspired by different motivations and serving distinct needs, for example, stochastic pooling [27], spatial pyramid pooling [28, 29], and def-pooling [30].

(iii) Fully Connected Layers. Following several convolutional and pooling layers, the high-level reasoning in the neural network is performed via fully connected layers. Neurons in a fully connected layer have full connections to all activation in the previous layer, as their name implies. Their activation can hence be computed with a matrix multiplication followed by a bias offset. Fully connected layers eventually convert the 2D feature maps into a 1D feature vector. The derived vector either could be fed forward into a certain number of categories for classification [31] or could be considered as a feature vector for further processing [32].

The architecture of CNNs employs three concrete ideas: (a) local receptive fields, (b) tied weights, and (c) spatial subsampling. Based on local receptive field, each unit in a convolutional layer receives inputs from a set of neighboring units belonging to the previous layer. This way neurons are capable of extracting elementary visual features such as edges or corners. These features are then combined by the subsequent convolutional layers in order to detect higher order features. Furthermore, the idea that elementary feature detectors, which are useful on a part of an image, are likely to be useful across the entire image is implemented by the concept of tied weights. The concept of tied weights constraints a set of units to have identical weights. Concretely, the units of a convolutional layer are organized in planes. All units of a plane share the same set of weights. Thus, each plane is responsible for constructing a specific feature. The outputs of planes are called feature maps. Each convolutional layer consists of several planes, so that multiple feature maps can be constructed at each location.

During the construction of a feature map, the entire image is scanned by a unit whose states are stored at corresponding locations in the feature map. This construction is equivalent to a convolution operation, followed by an additive bias term and sigmoid function:

$$\mathbf{y}^{(d)} = \sigma\left(\mathbf{W}\mathbf{y}^{(d-1)} + \mathbf{b}\right), \tag{1}$$

where d stands for the depth of the convolutional layer, $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias term. For fully connected neural networks, the weight matrix is full, that is, connects every input to every unit with different weights. For CNNs, the weight matrix $\mathbf{W}$ is very sparse due to the concept of tied weights. Thus, $\mathbf{W}$ has the form of

$$\begin{bmatrix} \mathbf{w} & 0 & \cdots & 0 \\ 0 & \mathbf{w} & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{w} \end{bmatrix}, \tag{2}$$

where $\mathbf{w}$ are matrices having the same dimensions with the units' receptive fields. Employing a sparse weight matrix reduces the number of network's tunable parameters and thus increases its generalization ability. Multiplying $\mathbf{W}$ with layer inputs is like convolving the input with $\mathbf{w}$, which can be seen as a trainable filter. If the input to $d-1$ convolutional layer is of

dimension $N \times N$ and the receptive field of units at a specific plane of convolutional layer d is of dimension $m \times m$, then the constructed feature map will be a matrix of dimensions $(N - m + 1) \times (N - m + 1)$. Specifically, the element of feature map at (i, j) location will be

$$\mathbf{y}_{ij}^{(d)} = \sigma\left(x_{ij}^{(d)} + b\right) \tag{3}$$

with

$$x_{ij}^{(d)} = \sum_{\alpha=0}^{m-1} \sum_{b=0}^{m-1} w_{\alpha b} \mathbf{y}_{(i+\alpha)(j+b)}^{(d-1)}, \tag{4}$$

where the bias term b is scalar. Using (4) and (3) sequentially for all (i, j) positions of input, the feature map for the corresponding plane is constructed.

One of the difficulties that may arise with training of CNNs has to do with the large number of parameters that have to be learned, which may lead to the problem of overfitting. To this end, techniques such as stochastic pooling, dropout, and data augmentation have been proposed. Furthermore, CNNs are often subjected to pretraining, that is, to a process that initializes the network with pretrained parameters instead of randomly set ones. Pretraining can accelerate the learning process and also enhance the generalization capability of the network.

Overall, CNNs were shown to significantly outperform traditional machine learning approaches in a wide range of computer vision and pattern recognition tasks [33], examples of which will be presented in Section 3. Their exceptional performance combined with the relative easiness in training are the main reasons that explain the great surge in their popularity over the last few years.

2.2. Deep Belief Networks and Deep Boltzmann Machines. Deep Belief Networks and Deep Boltzmann Machines are deep learning models that belong in the "Boltzmann family," in the sense that they utilize the Restricted Boltzmann Machine (RBM) as learning module. The Restricted Boltzmann Machine (RBM) is a generative stochastic neural network. DBNs have undirected connections at the top two layers which form an RBM and directed connections to the lower layers. DBMs have undirected connections between all layers of the network. A graphic depiction of DBNs and DBMs can be found in Figure 2. In the following subsections, we will describe the basic characteristics of DBNs and DBMs, after presenting their basic building block, the RBM.

2.2.1. Restricted Boltzmann Machines. A Restricted Boltzmann Machine ([34, 35]) is an undirected graphical model with stochastic visible variables $\mathbf{v} \in \{0, 1\}^D$ and stochastic hidden variables $\mathbf{h} \in \{0, 1\}^F$, where each visible variable is connected to each hidden variable. An RBM is a variant of the Boltzmann Machine, with the restriction that the visible units and hidden units must form a bipartite graph. This restriction allows for more efficient training algorithms, in particular the gradient-based contrastive divergence algorithm [36].

The model defines the energy function E: $\{0, 1\}^D \times \{0, 1\}^F \to \mathbb{R}$:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{D} \sum_{j=1}^{F} W_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F} \alpha_j h_j, \tag{5}$$

where $\theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}\}$ are the model parameters; that is, W_{ij} represents the symmetric interaction term between visible unit i and hidden unit j, and b_i, a_j are bias terms.

The joint distribution over the visible and hidden units is given by

$$P(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \qquad \mathcal{Z}(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \tag{6}$$

where $\mathcal{Z}(\theta)$ is the normalizing constant. The conditional distributions over hidden $\mathbf{h}$ and visible $\mathbf{v}$ vectors can be derived by (5) and (6) as

$$P(\mathbf{h} \mid \mathbf{v}; \theta) = \prod_{j=1}^{F} p\left(h_j \mid \mathbf{v}\right), \qquad P(\mathbf{v} \mid \mathbf{h}; \theta) = \prod_{i=1}^{D} p\left(v_i \mid \mathbf{h}\right). \tag{7}$$

Given a set of observations $\{\mathbf{v}_n\}_{n=1}^N$ the derivative of the log-likelihood with respect to the model parameters can be derived by (6) as

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{v}_n; \theta)}{\partial W_{ij}} = \mathbb{E}_{P_{\text{data}}}\left[v_i h_j\right] - \mathbb{E}_{P_{\text{model}}}\left[v_i h_j\right], \tag{8}$$

where $\mathbb{E}_{P_{\text{data}}}$ denotes an expectation with respect to the data distribution $P_{\text{data}}(\mathbf{h}, \mathbf{v}; \theta) = P(\mathbf{h} \mid \mathbf{v}; \theta) P_{\text{data}}(\mathbf{v})$, with $P_{\text{data}}(\mathbf{v}) = (1/N) \sum_n \delta(\mathbf{v} - \mathbf{v}_\mathbf{n})$ representing the empirical distribution and $\mathbb{E}_{P\text{model}}$ is an expectation with respect to the distribution defined by the model, as in (6).

A detailed explanation along with the description of a practical way to train RBMs was given in [37], whereas [38] discusses the main difficulties of training RBMs and their underlying reasons and proposes a new algorithm with an adaptive learning rate and an enhanced gradient, so as to address the aforementioned difficulties.

2.2.2. Deep Belief Networks. Deep Belief Networks (DBNs) are probabilistic generative models which provide a joint probability distribution over observable data and labels. They are formed by stacking RBMs and training them in a greedy manner, as was proposed in [39]. A DBN initially employs an efficient layer-by-layer greedy learning strategy to initialize the deep network, and, in the sequel, fine-tunes all weights jointly with the desired outputs. DBNs are graphical models which learn to extract a deep hierarchical representation of

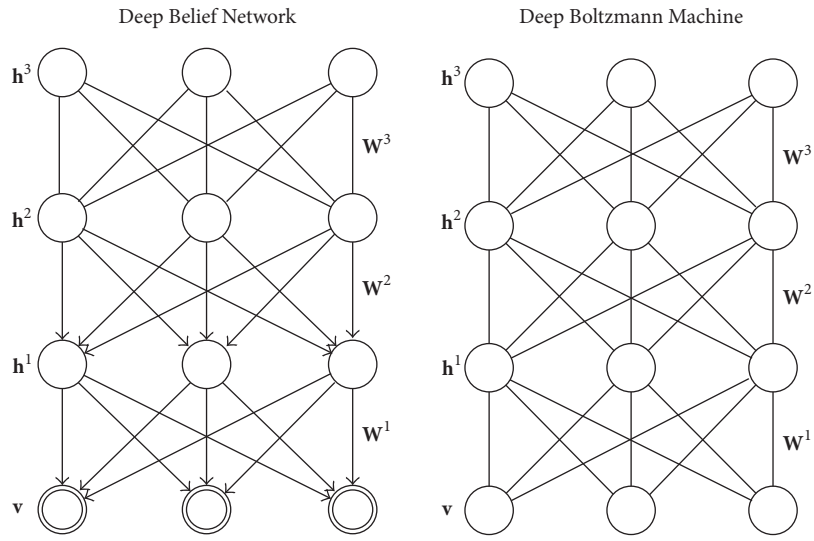


Figure 2: Deep Belief Network (DBN) and Deep Boltzmann Machine (DBM). The top two layers of a DBN form an undirected graph and the remaining layers form a belief network with directed, top-down connections. In a DBM, all connections are undirected.

the training data. They model the joint distribution between observed vector $\mathbf{x}$ and the l hidden layers $\mathbf{h}^k$ as follows:

$$P\left(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^l\right) = \left(\prod_{k=0}^{l-2} P\left(\mathbf{h}^k \mid \mathbf{h}^{k+1}\right)\right) P\left(\mathbf{h}^{l-1}, \mathbf{h}^l\right), \quad (9)$$

where $\mathbf{x} = \mathbf{h}^0$, $P(\mathbf{h}^k \mid \mathbf{h}^{k+1})$ is a conditional distribution for the visible units at level k conditioned on the hidden units of the RBM at level $k + 1$, and $P(\mathbf{h}^{l-1} \mid \mathbf{h}^l)$ is the visible-hidden joint distribution in the top-level RBM.

The principle of greedy layer-wise unsupervised training can be applied to DBNs with RBMs as the building blocks for each layer [33, 39]. A brief description of the process follows:

(1) Train the first layer as an RBM that models the raw input $\mathbf{x} = \mathbf{h}^0$ as its visible layer.

(2) Use that first layer to obtain a representation of the input that will be used as data for the second layer. Two common solutions exist. This representation can be chosen as being the mean activation $P(\mathbf{h}^1 = 1 \mid \mathbf{h}^0)$ or samples of $P(\mathbf{h}^1 \mid \mathbf{h}^0)$.

(3) Train the second layer as an RBM, taking the transformed data (samples or mean activation) as training examples (for the visible layer of that RBM).

(4) Iterate steps ((2) and (3)) for the desired number of layers, each time propagating upward either samples or mean values.

(5) Fine-tune all the parameters of this deep architecture with respect to a proxy for the DBN log- likelihood, or with respect to a supervised training criterion (after adding extra learning machinery to convert the learned representation into supervised predictions, e.g., a linear classifier).

There are two main advantages in the above-described greedy learning process of the DBNs [40]. First, it tackles the challenge of appropriate selection of parameters, which in some cases can lead to poor local optima, thereby ensuring that the network is appropriately initialized. Second, there is no requirement for labelled data since the process is unsupervised. Nevertheless, DBNs are also plagued by a number of shortcomings, such as the computational cost associated with training a DBN and the fact that the steps towards further optimization of the network based on maximum likelihood training approximation are unclear [41]. Furthermore, a significant disadvantage of DBNs is that they do not account for the two-dimensional structure of an input image, which may significantly affect their performance and applicability in computer vision and multimedia analysis problems. However, a later variation of the DBN, the Convolutional Deep Belief Network (CDBN) ([42, 43]), uses the spatial information of neighboring pixels by introducing convolutional RBMs, thus producing a translation invariant generative model that successfully scales when it comes to high dimensional images, as is evidenced in [44].

2.2.3. Deep Boltzmann Machines. Deep Boltzmann Machines (DBMs) [45] are another type of deep model using RBM as their building block. The difference in architecture of DBNs is that, in the latter, the top two layers form an undirected graphical model and the lower layers form a directed generative model, whereas in the DBM all the connections are undirected. DBMs have multiple layers of hidden units, where units in odd-numbered layers are conditionally independent of even-numbered layers, and vice versa. As a result, inference in the DBM is generally intractable. Nonetheless, an appropriate selection of interactions between visible and hidden units can lead to more tractable versions of the model. During network training, a DBM jointly trains all layers of a specific unsupervised model, and instead of maximizing the likelihood directly, the DBM uses a stochastic maximum likelihood (SML) [46] based algorithm to maximize the lower

bound on the likelihood. Such a process would seem vulnerable to falling in poor local minima [45], leaving several units effectively dead. Instead, a greedy layer-wise training strategy was proposed [47], which essentially consists in pretraining the layers of the DBM, similarly to DBN, namely, by stacking RBMs and training each layer to independently model the output of the previous layer, followed by a final joint fine-tuning.

Regarding the advantages of DBMs, they can capture many layers of complex representations of input data and they are appropriate for unsupervised learning since they can be trained on unlabeled data, but they can also be fine-tuned for a particular task in a supervised fashion. One of the attributes that sets DBMs apart from other deep models is that the approximate inference process of DBMs includes, apart from the usual bottom-up process, a top-down feedback, thus incorporating uncertainty about inputs in a more effective manner. Furthermore, in DBMs, by following the approximate gradient of a variational lower bound on the likelihood objective, one can jointly optimize the parameters of all layers, which is very beneficial especially in cases of learning models from heterogeneous data originating from different modalities [48].

As far as the drawbacks of DBMs are concerned, one of the most important ones is, as mentioned above, the high computational cost of inference, which is almost prohibitive when it comes to joint optimization in sizeable datasets. Several methods have been proposed to improve the effectiveness of DBMs. These include accelerating inference by using separate models to initialize the values of the hidden units in all layers [47, 49], or other improvements at the pretraining stage [50, 51] or at the training stage [52, 53].

2.3. Stacked (Denoising) Autoencoders. Stacked Autoencoders use the autoencoder as their main building block, similarly to the way that Deep Belief Networks use Restricted Boltzmann Machines as component. It is therefore important to briefly present the basics of the autoencoder and its denoising version, before describing the deep learning architecture of Stacked (Denoising) Autoencoders.

2.3.1. Autoencoders. An autoencoder is trained to encode the input $\mathbf{x}$ into a representation $\mathbf{r}(\mathbf{x})$ in a way that input can be reconstructed from $\mathbf{r}(\mathbf{x})$ [33]. The target output of the autoencoder is thus the autoencoder input itself. Hence, the output vectors have the same dimensionality as the input vector. In the course of this process, the reconstruction error is being minimized, and the corresponding code is the learned feature. If there is one linear hidden layer and the mean squared error criterion is used to train the network, then the k hidden units learn to project the input in the span of the first k principal components of the data [54]. If the hidden layer is nonlinear, the autoencoder behaves differently from PCA, with the ability to capture multimodal aspects of the input distribution [55]. The parameters of the model are optimized so that the average reconstruction error is minimized. There are many alternatives to measure the reconstruction error, including the traditional squared error:

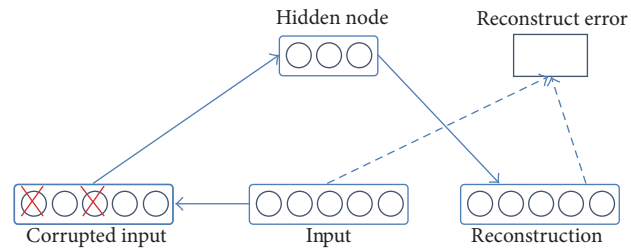


Figure 3: Denoising autoencoder [56].

$$L = \|\mathbf{x} - \mathbf{f}(\mathbf{r}(\mathbf{x}))\|^2, \tag{10}$$

where function $\mathbf{f}$ is the *decoder* and $\mathbf{f}(\mathbf{r}(\mathbf{x}))$ is the reconstruction produced by the model.

If the input is interpreted as bit vectors or vectors of bit probabilities, then the loss function of the reconstruction could be represented by cross-entropy; that is,

$$L = -\sum_i \mathbf{x}_i \log \mathbf{f}_i(\mathbf{r}(\mathbf{x})) + (1 - \mathbf{x}_i) \log(1 - \mathbf{f}_i(\mathbf{r}(\mathbf{x}))). \tag{11}$$

The goal is for the representation (or *code*) $\mathbf{r}(\mathbf{x})$ to be a distributed representation that manages to capture the coordinates along the main variations of the data, similarly to the principle of Principal Components Analysis (PCA). Given that $\mathbf{r}(\mathbf{x})$ is not lossless, it is impossible for it to constitute a successful compression for all input $\mathbf{x}$. The aforementioned optimization process results in low reconstruction error on test examples from the same distribution as the training examples but generally high reconstruction error on samples arbitrarily chosen from the input space.

2.3.2. Denoising Autoencoders. The denoising autoencoder [56] is a stochastic version of the autoencoder where the input is stochastically corrupted, but the uncorrupted input is still used as target for the reconstruction. In simple terms, there are two main aspects in the function of a denoising autoencoder: first it tries to encode the input (namely, preserve the information about the input), and second it tries to undo the effect of a corruption process stochastically applied to the input of the autoencoder (see Figure 3). The latter can only be done by capturing the statistical dependencies between the inputs. It can be shown that the denoising autoencoder maximizes a lower bound on the log-likelihood of a generative model.

In [56], the stochastic corruption process arbitrarily sets a number of inputs to zero. Then the denoising autoencoder is trying to predict the corrupted values from the uncorrupted ones, for randomly selected subsets of missing patterns. In essence, the ability to predict any subset of variables from the remaining ones is a sufficient condition for completely capturing the joint distribution between a set of variables. It should be mentioned that using autoencoders for denoising was introduced in earlier works (e.g., [57]), but the substantial contribution of [56] lies in the demonstration of the successful use of the method for unsupervised pretraining of a deep architecture and in linking the denoising autoencoder to a generative model.

2.3.3. Stacked (Denoising) Autoencoders. It is possible to stack denoising autoencoders in order to form a deep network by feeding the latent representation (output code) of the denoising autoencoder of the layer below as input to the current layer. The unsupervised pretraining of such an architecture is done one layer at a time. Each layer is trained as a denoising autoencoder by minimizing the error in reconstructing its input (which is the output code of the previous layer). When the first k layers are trained, we can train the $(k + 1)$th layer since it will then be possible compute the latent representation from the layer underneath.

When pretraining of all layers is completed, the network goes through a second stage of training called fine-tuning. Here supervised fine-tuning is considered when the goal is to optimize prediction error on a supervised task. To this end, a logistic regression layer is added on the output code of the output layer of the network. The derived network is then trained like a multilayer perceptron, considering only the encoding parts of each autoencoder at this point. This stage is supervised, since the target class is taken into account during training.

As is easily seen, the principle for training stacked autoencoders is the same as the one previously described for Deep Belief Networks, but using autoencoders instead of Restricted Boltzmann Machines. A number of comparative experimental studies show that Deep Belief Networks tend to outperform stacked autoencoders ([58, 59]), but this is not always the case, especially when DBNs are compared to Stacked Denoising Autoencoders [56].

One strength of autoencoders as the basic unsupervised component of a deep architecture is that, unlike with RBMs, they allow almost any parametrization of the layers, on condition that the training criterion is continuous in the parameters. In contrast, one of the shortcomings of SAs is that they do not correspond to a generative model, when with generative models like RBMs and DBNs, samples can be drawn to check the outputs of the learning process.

2.4. Discussion. Some of the strengths and limitations of the presented deep learning models were already discussed in the respective subsections. In an attempt to compare these models (for a summary see Table 2), we can say that CNNs have generally performed better than DBNs in current literature on benchmark computer vision datasets such as MNIST. In cases where the input is nonvisual, DBNs often outperform other models, but the difficulty in accurately estimating joint probabilities as well as the computational cost in creating a DBN constitutes drawbacks. A major positive aspect of CNNs is "feature learning," that is, the bypassing of handcrafted features, which are necessary for other types of networks; however, in CNNs features are automatically learned. On the other hand, CNNs rely on the availability of ground truth, that is, labelled training data, whereas DBNs/DBMs and SAs do not have this limitation and can work in an unsupervised manner. On a different note, one of the disadvantages of autoencoders lies in the fact that they could become ineffective if errors are present in the first layers. Such errors may cause the network to learn to reconstruct the average of the training data. Denoising autoencoders [56], however, can retrieve the correct input from a corrupted version, thus leading the network to grasp the structure of the input distribution. In terms of the efficiency of the training process, only in the case of SAs is real-time training possible, whereas CNNs and DBNs/DBMs training processes are time-consuming. Finally, one of the strengths of CNNs is the fact that they can be invariant to transformations such as translation, scale, and rotation. Invariance to translation, rotation, and scale is one of the most important assets of CNNs, especially in computer vision problems, such as object detection, because it allows abstracting an object's identity or category from the specifics of the visual input (e.g., relative positions/orientation of the camera and the object), thus enabling the network to effectively recognize a given object in cases where the actual pixel values on the image can significantly differ.

Table 2: Comparison of CNNs, DBNs/DBMs, and SdAs with respect to a number of properties. + denotes a good performance in the property and − denotes bad performance or complete lack thereof.

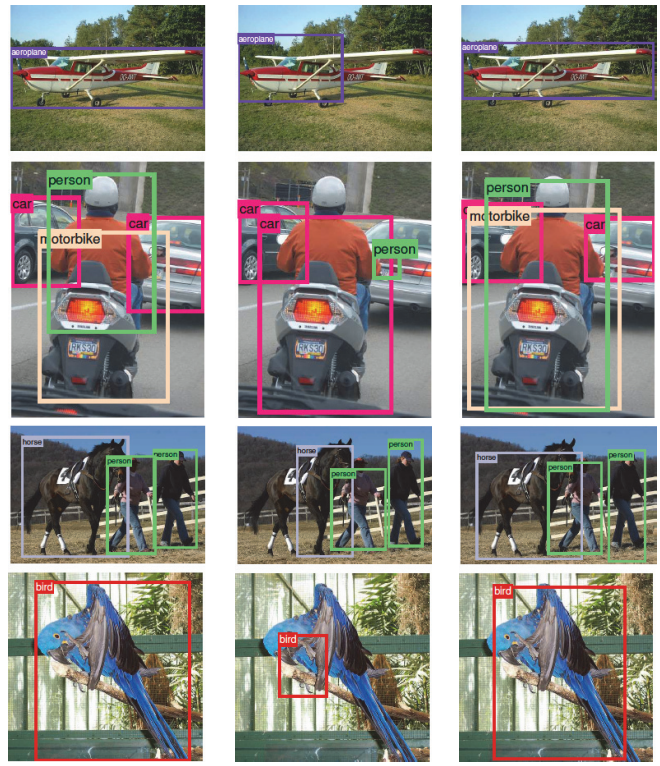
Model properties	CNNs	DBNs/DBMs	SdAs
Unsupervised learning	−	+	+
Training efficiency	−	−	+
Feature learning	+	−	−
Scale/rotation/translation invariance	+	−	−
Generalization	+	+	+

3. Applications in Computer Vision

In this section, we survey works that have leveraged deep learning methods to address key tasks in computer vision, such as object detection, face recognition, action and activity recognition, and human pose estimation.

3.1. Object Detection. Object detection is the process of detecting instances of semantic objects of a certain class (such as humans, airplanes, or birds) in digital images and video (Figure 4). A common approach for object detection frameworks includes the creation of a large set of candidate windows that are in the sequel classified using CNN features. For example, the method described in [32] employs selective search [60] to derive object proposals, extracts CNN features for each proposal, and then feeds the features to an SVM classifier to decide whether the windows include the object or not. A large number of works is based on the concept of Regions with CNN features proposed in [32]. Approaches following the Regions with CNN paradigm usually have good detection accuracies (e.g., [61, 62]); however, there is a significant number of methods trying to further improve the performance of Regions with CNN approaches, some of which succeed in finding approximate object positions but often cannot precisely determine the exact position of the object [63]. To this end, such methods often follow a joint object detection—semantic segmentation approach [64–66], usually attaining good results.

A vast majority of works on object detection using deep learning apply a variation of CNNs, for example, [8, 67, 68]



(a) (b) (c)

Figure 4: Object detection results comparison from [66]. (a) Ground truth; (b) bounding boxes obtained with [32]; (c) bounding boxes obtained with [66].

(in which a new def-pooling layer and new learning strategy are proposed), [9] (weakly supervised cascaded CNNs), and [69] (subcategory-aware CNNs). However, there does exist a relatively small number of object detection attempts using other deep models. For example, [70] proposes a coarse object locating method based on a saliency mechanism in conjunction with a DBN for object detection in remote sensing images; [71] presents a new DBN for 3D object recognition, in which the top-level model is a third-order Boltzmann machine, trained using a hybrid algorithm that combines both generative and discriminative gradients; [72] employs a fused deep learning approach, while [73] explores the representation capabilities of a deep model in a semisupervised paradigm. Finally, [74] leverages stacked autoencoders for multiple organ detection in medical images, while [75] exploits saliency-guided stacked autoencoders for video-based salient object detection.

3.2. Face Recognition. Face recognition is one of the hottest computer vision applications with great commercial interest as well. A variety of face recognition systems based on the extraction of handcrafted features have been proposed [76–79]; in such cases, a feature extractor extracts features from an aligned face to obtain a low-dimensional representation, based on which a classifier makes predictions. CNNs brought about a change in the face recognition field, thanks to their feature learning and transformation invariance properties. The first work employing CNNs for face recognition was [80]; today light CNNs [81] and VGG Face Descriptor [82] are among the state of the art. In [44] a Convolutional DBN achieved a great performance in face verification.

Moreover, Google's FaceNet [83] and Facebook's DeepFace [84] are both based on CNNs. DeepFace [84] models a face in 3D and aligns it to appear as a frontal face. Then, the normalized input is fed to a single convolution-pooling-convolution filter, followed by three locally connected layers and two fully connected layers used to make final predictions. Although DeepFace attains great performance rates, its representation is not easy to interpret because the faces of the same person are not necessarily clustered during the training process. On the other hand, FaceNet defines a triplet loss function on the representation, which makes the training process learn to cluster the face representation of the same person. Furthermore, CNNs constitute the core of OpenFace [85], an open-source face recognition tool, which is of comparable (albeit a little lower) accuracy, is open-source, and is suitable for mobile computing, because of its smaller size and fast execution time.

3.3. Action and Activity Recognition. Human action and activity recognition is a research issue that has received a lot of attention from researchers [86, 87]. Many works on human activity recognition based on deep learning techniques have been proposed in the literature in the last few years [88]. In [89] deep learning was used for complex event detection and recognition in video sequences: first, saliency maps were used

for detecting and localizing events, and then deep learning was applied to the pretrained features for identifying the most important frames that correspond to the underlying event. In [90] the authors successfully employ a CNN-based approach for activity recognition in beach volleyball, similarly to the approach of [91] for event classification from large-scale video datasets; in [92], a CNN model is used for activity recognition based on smartphone sensor data. The authors of [12] incorporate a radius–margin bound as a regularization term into the deep CNN model, which effectively improves the generalization performance of the CNN for activity classification. In [13], the authors scrutinize the applicability of CNN as joint feature extraction and classification model for fine-grained activities; they find that due to the challenges of large intraclass variances, small interclass variances, and limited training samples per activity, an approach that directly uses deep features learned from ImageNet in an SVM classifier is preferable.

Driven by the adaptability of the models and by the availability of a variety of different sensors, an increasingly popular strategy for human activity recognition consists in fusing multimodal features and/or data. In [93], the authors mixed appearance and motion features for recognizing group activities in crowded scenes collected from the web. For the combination of the different modalities, the authors applied multitask deep learning. The work of [94] explores combination of heterogeneous features for complex event recognition. The problem is viewed as two different tasks: first, the most informative features for recognizing events are estimated, and then the different features are combined using an AND/OR graph structure. There is also a number of works combining more than one type of model, apart from several data modalities. In [95], the authors propose a multimodal multistream deep learning framework to tackle the egocentric activity recognition problem, using both the video and sensor data and employing a dual CNNs and Long Short-Term Memory architecture. Multimodal fusion with a combined CNN and LSTM architecture is also proposed in [96]. Finally, [97] uses DBNs for activity recognition using input video sequences that also include depth information.

3.4. Human Pose Estimation. The goal of human pose estimation is to determine the position of human joints from images, image sequences, depth images, or skeleton data as provided by motion capturing hardware [98]. Human pose estimation is a very challenging task owing to the vast range of human silhouettes and appearances, difficult illumination, and cluttered background. Before the era of deep learning, pose estimation was based on detection of body parts, for example, through pictorial structures [99].

Moving on to deep learning methods in human pose estimation, we can group them into holistic and part-based methods, depending on the way the input images are processed. The holistic processing methods tend to accomplish their task in a global fashion and do not explicitly define a model for each individual part and their spatial relationships. DeepPose [14] is a holistic model that formulates the human pose estimation method as a joint regression problem and does not explicitly define the graphical model or part detectors for the human pose estimation. Nevertheless, holistic-based methods tend to be plagued by inaccuracy in the high-precision region due to the difficulty in learning direct regression of complex pose vectors from images.

On the other hand, the part-based processing methods focus on detecting the human body parts individually, followed by a graphic model to incorporate the spatial information. In [15], the authors, instead of training the network using the whole image, use the local part patches and background patches to train a CNN, in order to learn conditional probabilities of the part presence and spatial relationships. In [100] the approach trains multiple smaller CNNs to perform independent binary body-part classification, followed with a higher-level weak spatial model to remove strong outliers and to enforce global pose consistency. Finally, in [101], a multi-resolution CNN is designed to perform heat-map likelihood regression for each body part, followed with an implicit graphic model to further promote joint consistency.

3.5. Datasets. The applicability of deep learning approaches has been evaluated on numerous datasets, whose content varied greatly, according the application scenario. Regardless of the investigated case, the main application domain is (natural) images. A brief description of utilized datasets (traditional and new ones) for benchmarking purposes is provided below.

(1) Grayscale Images. The most used grayscale images dataset is MNIST [20] and its variations, that is, NIST and perturbed NIST. The application scenario is the recognition of handwritten digits.

(2) RGB Natural Images. Caltech RGB image datasets [102], for example, Caltech 101/Caltech 256 and the Caltech Silhouettes, contain pictures of objects belonging to 101/256 categories. CIFAR datasets [103] consist of thousands of 32×32 color images in various classes. COIL datasets [104] consist of different objects imaged at every angle in a 360 rotation.

(3) Hyperspectral Images. SCIEN hyperspectral image data [105] and AVIRIS sensor based datasets [106], for example, contain hyperspectral images.

(4) Facial Characteristics Images. Adience benchmark dataset [107] can be used for facial attributes identification, that is, age and gender, from images of faces. Face recognition in unconstrained environments [108] is another commonly used dataset.

(5) Medical Images. Chest X-ray dataset [109] comprises 112120 frontal-view X-ray images of 30805 unique patients with the text-mined fourteen disease image labels (where each image can have multilabels). Lymph Node Detection and Segmentation datasets [110] consist of Computed Tomography images of the mediastinum and abdomen.

(6) Video Streams. The WR datasets [111, 112] can be used for video-based activity recognition in assembly lines [113], containing sequences of 7 categories of industrial tasks. YouTube-8M [114] is a dataset of 8 million YouTube video URLs, along with video-level labels from a diverse set of 4800 Knowledge Graph entities.

4. Conclusions

The surge of deep learning over the last years is to a great extent due to the strides it has enabled in the field of computer vision. The three key categories of deep learning for computer vision that have been reviewed in this paper, namely, CNNs, the "Boltzmann family" including DBNs and DBMs, and SdAs, have been employed to achieve significant performance rates in a variety of visual understanding tasks, such as object detection, face recognition, action and activity recognition, human pose estimation, image retrieval, and semantic segmentation. However, each category has distinct advantages and disadvantages. CNNs have the unique capability of feature learning, that is, of automatically learning features based on the given dataset. CNNs are also invariant to transformations, which is a great asset for certain computer vision applications. On the other hand, they heavily rely on the existence of labelled data, in contrast to DBNs/DBMs and SdAs, which can work in an unsupervised fashion. Of the models investigated, both CNNs and DBNs/DBMs are computationally demanding when it comes to training, whereas SdAs can be trained in real time under certain circumstances.

As a closing note, in spite of the promising—in some cases impressive—results that have been documented in the literature, significant challenges do remain, especially as far as the theoretical groundwork that would clearly explain the ways to define the optimal selection of model type and structure for a given task or to profoundly comprehend the reasons for which a specific architecture or algorithm is effective in a given task or not. These are among the most important issues that will continue to attract the interest of the machine learning research community in the years to come.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research is implemented through IKY scholarships programme and cofinanced by the European Union (European Social Fund—ESF) and Greek national funds through the action titled "Reinforcement of Postdoctoral Researchers," in the framework of the Operational Programme "Human Resources Development Program, Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) 2014–2020.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115–133, 1943.

[2] Y. LeCun, B. Boser, J. Denker et al., "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2 (NIPS*89)*, D. Touretzky, Ed., Denver, CO, USA, 1990.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[5] TensorFlow, Available online: https://www.tensorflow.org.

[6] B. Frederic, P. Lamblin, R. Pascanu et al., "Theano: new features and speed improvements," in *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*, 2012, http://deeplearning.net/software/theano/.

[7] Mxnet, Available online: http://mxnet.io.

[8] W. Ouyang, X. Zeng, X. Wang et al., "DeepID-Net: Object Detection with Deformable Part Based Convolutional Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1320–1334, 2017.

[9] A. Diba, V. Sharma, A. Pazandeh, H. Pirsiavash, and L. V. Gool, "Weakly Supervised Cascaded Convolutional Networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5131–5139, Honolulu, HI, July 2017.

[10] N. Doulamis and A. Voulodimos, "FAST-MDL: Fast Adaptive Supervised Training of multi-layered deep learning models for consistent object tracking and classification," in *Proceedings of the 2016 IEEE International Conference on Imaging Systems and Techniques, IST 2016*, pp. 318–323, October 2016.

[11] N. Doulamis, "Adaptable deep learning structures for object labeling/tracking under dynamic visual environments," *Multimedia Tools and Applications*, pp. 1–39, 2017.

[12] L. Lin, K. Wang, W. Zuo, M. Wang, J. Luo, and L. Zhang, "A deep structured model with radius-margin bound for 3D human activity recognition," *International Journal of Computer Vision*, vol. 118, no. 2, pp. 256–273, 2016.

[13] S. Cao and R. Nevatia, "Exploring deep learning based solutions in fine grained activity recognition in the wild," in *Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 384–389, Cancun, December 2016.

[14] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, pp. 1653–1660, USA, June 2014.

[15] X. Chen and A. L. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in *Proceedings of the NIPS*, 2014.

[16] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 1520–1528, Santiago, Chile, December 2015.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 3431–3440, IEEE, Boston, Mass, USA, June 2015.

[18] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[19] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[21] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[22] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam, "A mathematical motivation for complex-valued convolutional networks," *Neural Computation*, vol. 28, no. 5, pp. 815–825, 2016.

[23] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - Weakly-supervised learning with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 685–694, June 2015.

[24] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[25] Y. L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the ICML*, 2010.

[26] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 6354, no. 3, pp. 92–101, 2010.

[27] H. Wu and X. Gu, "Max-Pooling Dropout for Regularization of Convolutional Neural Networks," in *Neural Information Processing*, vol. 9489 of *Lecture Notes in Computer Science*, pp. 46–54, Springer International Publishing, Cham, 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in *Computer Vision – ECCV 2014*, vol. 8691 of *Lecture Notes in Computer Science*, pp. 346–361, Springer International Publishing, Cham, 2014.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[30] W. Ouyang, X. Wang, X. Zeng et al., "DeepID-Net: Deformable deep convolutional neural networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 2403–2412, USA, June 2015.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 580–587, Columbus, Ohio, USA, June 2014.

[33] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.

[34] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *In Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 194–281, MIT Press, Cambridge, MA, USA, 1986.

[35] G. E. Hinton and T. J. Sejnowski, "Learning and Relearning in Boltzmann Machines," vol. 1, p. 4.2, MIT Press, Cambridge, MA, 1986.

[36] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," in *Proceedings of the tenth international workshop on artificial intelligence and statistics., NP: Society for Artificial Intelligence and Statistics*, pp. 33–40, 2005.

[37] G. Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 9, p. 926, 2010.

[38] K. Cho, T. Raiko, and A. Ilin, "Enhanced gradient for training restricted Boltzmann machines," *Neural Computation*, vol. 25, no. 3, pp. 805–831, 2013.

[39] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *American Association for the Advancement of Science: Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[40] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning—a new frontier in artificial intelligence research," *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13–18, 2010.

[41] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[42] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference (ICML '09)*, pp. 609–616, ACM, Montreal, Canada, June 2009.

[43] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Unsupervised learning of hierarchical representations with convolutional deep belief networks," *Communications of the ACM*, vol. 54, no. 10, pp. 95–103, 2011.

[44] G. B. Huang, H. Lee, and E. Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 2518–2525, June 2012.

[45] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 24, pp. 448–455, 2009.

[46] L. Younes, "On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates," *Stochastics and Stochastics Reports*, vol. 65, no. 3-4, pp. 177–228, 1999.

[47] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep Boltzmann machines," in *Proceedings of the AISTATS*, 2010.

[48] N. Srivastava and R. Salakhutdinov, "Multimodal learning with deep Boltzmann machines," *Journal of Machine Learning Research*, vol. 15, pp. 2949–2980, 2014.

[49] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep Boltzmann machines," *Neural Computation*, vol. 24, no. 8, pp. 1967–2006, 2012.

[50] R. Salakhutdinov and G. Hinton, "A better way to pretrain Deep Boltzmann Machines," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012*, pp. 2447–2455, usa, December 2012.

[51] K. Cho, T. Raiko, A. Ilin, and J. Karhunen, "A two-stage pretraining algorithm for deep boltzmann machines," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8131, pp. 106–113, 2013.

[52] G. Montavon and K. Müller, "Deep Boltzmann Machines and the Centering Trick," in *Neural Networks: Tricks of the Trade*, vol. 7700 of *Lecture Notes in Computer Science*, pp. 621–637, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[53] I. Goodfellow, M. Mirza, A. Courville et al., "Multi-prediction deep Boltzmann machines," in *Proceedings of the NIPS*, 2013.

[54] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[55] N. Japkowicz, S. J. Hanson, and M. A. Gluck, "Nonlinear autoassociation is not equivalent to PCA," *Neural Computation*, vol. 12, no. 3, pp. 531–545, 2000.

[56] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *in Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, W. W. Cohen, A. McCallum, and S. T. Roweis, Eds., pp. 1096–1103, ACM, 2008.

[57] P. Gallinari, Y. LeCun, S. Thiria, and F. Fogelman-Soulie, "Memoires associatives distribuees," in *Proceedings of the in Proceedings of COGNITIVA 87*, Paris, La Villette, 1987.

[58] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 473–480, Corvallis, Ore, UA, June 2007.

[59] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems (NIPS06)*, B. Sch, J. Platt, and., T. Hoffman, and B. Schölkopf, Eds., vol. 19, pp. 153–160, MIT Press, 2007.

[60] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[61] R. Girshick, "Fast R-CNN," in *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV '15)*, pp. 1440–1448, December 2015.

[62] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[63] J. Hosang, R. Benenson, and B. Schiele, "How good are detection proposals, really?" in *Proceedings of the 25th British Machine Vision Conference, BMVC 2014*, gbr, September 2014.

[64] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Computer Vision—ECCV 2014*, vol. 8695 of *Lecture Notes in Computer Science*, pp. 297–312, Springer, 2014.

[65] J. Dong, Q. Chen, S. Yan, and A. Yuille, "Towards unified object detection and semantic segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8693, no. 5, pp. 299–314, 2014.

[66] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler, "SegDeepM: Exploiting segmentation and context in deep neural networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 4703–4711, USA, June 2015.

[67] J. Liu, N. Lay, Z. Wei et al., "Colitis detection on abdominal CT scans by rich feature hierarchies," in *Proceedings of the Medical Imaging 2016: Computer-Aided Diagnosis*, vol. 9785 of *Proceedings of SPIE*, San Diego, Calif, USA, February 2016.

[68] G. Luo, R. An, K. Wang, S. Dong, and H. Zhang, "A Deep Learning Network for Right Ventricle Segmentation in Short:Axis MRI," in *Proceedings of the 2016 Computing in Cardiology Conference*.

[69] T. Chen, S. Lu, and J. Fan, "S-CNN: Subcategory-aware convolutional networks for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[70] W. Diao, X. Sun, X. Zheng, F. Dou, H. Wang, and K. Fu, "Efficient Saliency-Based Object Detection in Remote Sensing Images Using Deep Belief Networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 2, pp. 137–141, 2016.

[71] V. Nair and G. E. Hinton, "3D object recognition with deep belief nets," in *Proceedings of the NIPS*, 2009.

[72] N. Doulamis and A. Doulamis, "Fast and adaptive deep fusion learning for detecting visual objects," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7585, no. 3, pp. 345–354, 2012.

[73] N. Doulamis and A. Doulamis, "Semi-supervised deep learning for object tracking and classification," pp. 848–852.

[74] H.-C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1930–1943, 2013.

[75] J. Li, C. Xia, and X. Chen, "A benchmark dataset and saliency-guided stacked autoencoders for video-based salient object detection," *IEEE Transactions on Image Processing*, 2017.

[76] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: high-dimensional feature and its efficient compression for face verification," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, pp. 3025–3032, June 2013.

[77] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in *Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV '13)*, pp. 3208–3215, December 2013.

[78] T. Berg and P. N. Belhumeur, "Tom-vs-Pete classifiers and identity-preserving alignment for face verification," in *Proceedings of the 23rd British Machine Vision Conference (BMVC '12)*, pp. 1–11, September 2012.

[79] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: a joint formulation," in *Computer Vision—ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Proceedings, Part III*, vol. 7574 of *Lecture Notes in Computer Science*, pp. 566–579, Springer, Berlin, Germany, 2012.

[80] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 8, no. 1, pp. 98–113, 1997.

[81] X. Wu, R. He, Z. Sun, and T. Tan, A light CNN for deep face representation with noisy labels, https://arxiv.org/abs/1511.02683.

[82] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," in *Proceedings of the British Machine Vision Conference 2015*, pp. 41.1-41.12, Swansea.

[83] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 815–823, IEEE, Boston, Mass, USA, June 2015.

[84] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 1701–1708, Columbus, Ohio, USA, June 2014.

[85] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: a general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, 2016.

[86] A. S. Voulodimos, D. I. Kosmopoulos, N. D. Doulamis, and T. A. Varvarigou, "A top-down event-driven approach for concurrent activity recognition," *Multimedia Tools and Applications*, vol. 69, no. 2, pp. 293–311, 2014.

[87] A. S. Voulodimos, N. D. Doulamis, D. I. Kosmopoulos, and T. A. Varvarigou, "Improving multi-camera activity recognition by employing neural network based readjustment," *Applied Artificial Intelligence*, vol. 26, no. 1-2, pp. 97–118, 2012.

[88] K. Makantasis, A. Doulamis, N. Doulamis, and K. Psychas, "Deep learning based human behavior recognition in industrial workflows," in *Proceedings of the 23rd IEEE International Conference on Image Processing, ICIP 2016*, pp. 1609–1613, September 2016.

[89] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann, "DevNet: A Deep Event Network for multimedia event detection and evidence recounting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 2568–2577, USA, June 2015.

[90] T. Kautz, B. H. Groh, J. Hannink, U. Jensen, H. Strubberg, and B. M. Eskofier, "Activity recognition in beach volleyball using a DEEp Convolutional Neural NETwork: leveraging the potential of DEEp Learning in sports," *Data Mining and Knowledge Discovery*, vol. 31, no. 6, pp. 1678–1705, 2017.

[91] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition, (CVPR '14)*, pp. 1725–1732, Columbus, OH, USA, June 2014.

[92] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016.

[93] J. Shao, C. C. Loy, K. Kang, and X. Wang, "Crowded Scene Understanding by Deeply Learned Volumetric Slices," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 3, pp. 613–623, 2017.

[94] K. Tang, B. Yao, L. Fei-Fei, and D. Koller, "Combining the right features for complex event recognition," in *Proceedings of the 2013 14th IEEE International Conference on Computer Vision, ICCV 2013*, pp. 2696–2703, Australia, December 2013.

[95] S. Song, V. Chandrasekhar, B. Mandal et al., "Multimodal Multi-Stream Deep Learning for Egocentric Activity Recognition," in *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2016*, pp. 378–385, USA, July 2016.

[96] R. Kavi, V. Kulathumani, F. Rohit, and V. Kecojevic, "Multiview fusion for activity recognition using deep neural networks," *Journal of Electronic Imaging*, vol. 25, no. 4, Article ID 043010, 2016.

[97] H. Yalcin, "Human activity recognition using deep belief networks," in *Proceedings of the 24th Signal Processing and Communication Application Conference, SIU 2016*, pp. 1649–1652, tur, May 2016.

[98] A. Kitsikidis, K. Dimitropoulos, S. Douka, and N. Grammalidis, "Dance analysis using multiple kinect sensors," in *Proceedings of the 9th International Conference on Computer Vision Theory and Applications, VISAPP 2014*, pp. 789–795, prt, January 2014.

[99] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[100] A. Jain, J. Tompson, and M. Andriluka, "Learning human pose estimation features with convolutional networks," in *Proceedings of the ICLR*, 2014.

[101] J. J. Tompson, A. Jain, Y. LeCun et al., "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proceedings of the NIPS*, 2014.

[102] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[103] A. Krizhevsky and G. Hinton, Learning multiple layers of features from tiny images, 2009.

[104] S. A. Nene, S. K. Nayar, and H. Murase, Columbia object image library (coil-20), 1996.

[105] T. Skauli and J. Farrell, "A collection of hyperspectral images for imaging systems research," in *Proceedings of the Digital Photography IX*, USA, February 2013.

[106] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3," *Datasets*, 2015.

[107] E. Eidinger, R. Enbar, and T. Hassner, "Age and gender estimation of unfiltered faces," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2170–2179, 2014.

[108] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep., University of Massachusetts, Amherst, 2007.

[109] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-Ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3462–3471, Honolulu, HI, May 2017.

[110] A. Seff, L. Lu, A. Barbu, H. Roth, H.-C. Shin, and R. M. Summers, "Leveraging mid-level semantic boundary cues for automated lymph node detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9350, pp. 53–61, 2015.

[111] A. Voulodimos, D. Kosmopoulos, G. Vasileiou et al., "A dataset for workflow recognition in industrial scenes," in *Proceedings of the 2011 18th IEEE International Conference on Image Processing, ICIP 2011*, pp. 3249–3252, Belgium, September 2011.

[112] A. Voulodimos, D. Kosmopoulos, G. Vasileiou et al., "A threefold dataset for activity and workflow recognition in complex industrial environments," *IEEE MultiMedia*, vol. 19, no. 3, pp. 42–52, 2012.

[113] D. I. Kosmopoulos, A. S. Voulodimos, and A. D. Doulamis, "A system for multicamera task recognition and summarization for structured environments," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 161–171, 2013.

[114] S. Abu-El-Haija et al., "YouTube-8M: A large-scale video classification benchmark," Tech. Rep., 2016, https://arxiv.org/abs/1609.08675.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 7186120, 18 pages
https://doi.org/10.1155/2017/7186120

Research Article

Automatic Target Recognition Strategy for Synthetic Aperture Radar Images Based on Combined Discrimination Trees

Xiaohui Zhao,[1] Yicheng Jiang,[1] and Tania Stathaki[2]

[1]*Research Institute of Electronic Engineering Technology, Harbin Institute of Technology, Harbin, Heilongjiang, China*
[2]*Department of Electrical and Electronic Engineering, Imperial College, London, UK*

Correspondence should be addressed to Xiaohui Zhao; xh.zhao@outlook.com

Received 13 March 2017; Revised 21 September 2017; Accepted 8 October 2017; Published 29 November 2017

Academic Editor: Elio Masciari

A strategy is introduced for achieving high accuracy in synthetic aperture radar (SAR) automatic target recognition (ATR) tasks. Initially, a novel pose rectification process and an image normalization process are sequentially introduced to produce images with less variations prior to the feature processing stage. Then, feature sets that have a wealth of texture and edge information are extracted with the utilization of wavelet coefficients, where more effective and compact feature sets are acquired by reducing the redundancy and dimensionality of the extracted feature set. Finally, a group of discrimination trees are learned and combined into a final classifier in the framework of Real-AdaBoost. The proposed method is evaluated with the public release database for moving and stationary target acquisition and recognition (MSTAR). Several comparative studies are conducted to evaluate the effectiveness of the proposed algorithm. Experimental results show the distinctive superiority of the proposed method under both standard operating conditions (SOCs) and extended operating conditions (EOCs). Moreover, our additional tests suggest that good recognition accuracy can be achieved even with limited number of training images as long as these are captured with appropriately incremental sample step in target poses.

1. Introduction

Synthetic aperture radar (SAR) is a valuable technique for remote sensing and monitoring applications. Automatic target recognition (ATR) of SAR images is one of the most challenging SAR applications [1]. A typical SAR ATR system recognizes tactical ground targets of interests, that is, tanks, howitzers, and armoured vehicles, which is essential for identifying friends and foes and prerequisite for precision strikes.

SAR ATR involves a sequence of processes, such as some type of preprocessing, feature extraction, classifier construction, and finally target classification. The preprocessing stage may involve multiple types of processing that aims at facilitating the efficiency of image interpretation and analysis in the subsequent stages, for example, by suppressing the clutter reflections that obscure the contrast between the target of interest and the clutter. Moreover, SAR images are resized, shifted, and rotated to predefined standards. The so-called resizing is normally implemented by cropping out part of the image. The shifting and rotating processes are also known as image registration and pose rectification, respectively [2, 3].

Feature extraction is another essential stage which extracts effective discriminant features for improving recognition accuracy. Several features have already been exploited in SAR ATR [4–10]. Based on the consideration that tactical ground targets usually have a rectangular shape with different widths and lengths, geometric features are commonly used in SAR ATR. Zernike moments (ZMs) are employed in [6], taking advantage of their linear transformation invariance properties and robustness to the presence of noise. In [7], features are extracted based on pseudo-Zernike moments (pZm), which have merits such as the invariance properties, the independent property, and much lower sensitivity to noise in comparison with the ZMs. In [11], multiple geometric features are produced from calculating the axis projection of a target shape blob rotated clockwise with certain increment about the centre of the target. Then, the redundancy of the learned feature set is eliminated by keeping the rank of the

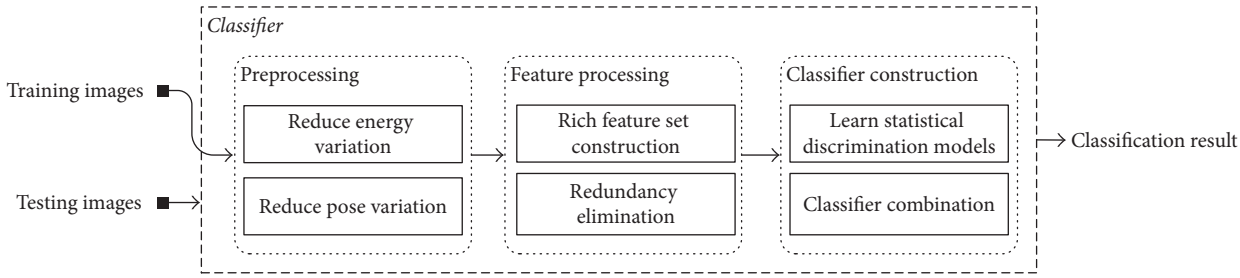


Figure 1: The SAR ATR scheme.

covariance matrix of the new feature set the same as that of the entire data set. However, the geometric features of the target of interest in SAR images are difficult to measure precisely due to the cluttered background and variations in poses and depression angles. Therefore, the recognition accuracy is not guaranteed. The polar mapping method, which is frequently used in ISAR image classification, is modified and used in [3] to address the SAR ATR problem. The original images are converted from the original 2D spatial domain (range and cross-range) to images in the polar coordinate domain (radius and angle) to produce polar-mapped images. The polar-mapped images are similar to the images that are mapped from the same target even in different poses. For that reason, the commonly used pose estimator is not necessarily needed for polar-mapped images. However, the performance of the polar mapping method depends highly on the determination of the reference central point for coordinate transformation, which is not a simple task especially for SAR images captured under various clutter environments.

Certain features are not feasible to be directly applied to classification due to their high dimensionality [12–19]. In [12], a compact representation feature, the monogenic signal, is employed for SAR ATR, where the high dimensional problem is circumvented by uniform downsampling, normalization, and concatenation of the monogenic components. Feature dimensionality reduction methods for SAR ATR based on manifold learning theory are also studied in recent years [13–17]. In [16], each sample is given a weight, which is called the sample discriminant coefficient (SDC), relating to its similarity to neighbouring samples, and then the SDC is combined with the Local Discriminant Embedding (LDE) method for producing redundancy-reduced features. Similarly, in [17], the so-called neighbourhood geometric centre scaling embedding (NGCSE) method is proposed, where geometric centre scaling is introduced into the neighbourhoods such that the samples are provided with clear clustering directions. However, the performance of most of the nonlinear dimensionality reduction methods relies heavily on the parameter selection of the neighbourhood, which is still an open problem.

The nearest neighbour classifier is one of the most used classifiers, where the extracted features are directly fed into the classifier to achieve the classification results [16]. Sparse representation based classification (SRC) is recently developed and exploited in SAR ATR, where the feature vectors of the testing samples are coded as sparse linear combinations of the feature vectors of the training samples, and the target with the minimum residual energy is recognized [12, 20]. Methods such as Support Vector Machines (SVM), Neural Networks (NN), and adaptive boosting (AdaBoost) are all vastly exploited in SAR ATR [2, 5, 21–23]. Various choices of base learners can be combined with the AdaBoost algorithm to solve the SAR ATR problem [2]. As explained in the Hughes phenomenon (also known as the curse of dimensionality), the difficulty of constructing classifier models becomes more prominent especially when the feature set is high in dimensionality while the number of the training data is limited (a fact in SAR ATR). However, the combination of the AdaBoost and graphical models is empirically proven in [24] to demonstrate good performance even when the training data is limited in number.

The SAR images are known for their indistinct appearances, variations in target appearances, and small number of available training samples. These problems must be properly addressed to achieve good recognition results for ATR tasks. To this end, a SAR ATR scheme is introduced as illustrated in Figure 1. Firstly, an initial processing stage is applied to facilitate the efficiency of feature extraction in the subsequent stages. More specifically, aiming at reducing the impact of variations in SAR images caused by variational echo energy and target poses, an image energy normalization process and a pose rectification process are applied sequentially. The construction of effective feature sets for ATR tasks is of crucial importance for achieving reliable recognition results. Therefore, it is suggested to extract a rich feature set that is formed by combining various types of discrimination features and then construct a more compact feature set by eliminating the redundancy of the rich feature set. We have decided to employ wavelet-based features. A rich feature set is firstly formed by combining the decomposed wavelet subband features, for example, the low-frequency information in LL subband coefficients and the high-frequency information in both LH and HL subband coefficients, where the HH subband is not involved since it is not stable feature in SAR images [25]. The involved coefficients actually depict the combination of texture features and horizontal and vertical edge features. After this, a compact low dimensional feature set which comprises features which retain most of the variance is constructed by employing the Principle Component Analysis (PCA) technique [26]. The relationship among features is statistically learned in a discriminative fashion rather than a generative fashion. Specifically, instead of using the true distribution,

which is usually unknown for most of the time, the empirical estimates are learned in a discriminative fashion by maximizing the J-divergence. Therefore, although the learned models may have low consistency with the real model of target classes due to limited amount of training data, high discrimination ability can still be achieved. Then, a final classifier is constructed by combining several discriminative tree based classifiers with the Real-AdaBoost framework [27]. To evaluate the performance of the proposed method, the moving and stationary target acquisition and recognition (MSTAR) public release data set is involved. Experimental results demonstrate that the proposed method outperforms several widely cited methods under both standard operating conditions (SOCs) and extended operating conditions (EOCs).

Variation reduction techniques that facilitate the efficiency of feature extraction are introduced in Section 2. The feature extraction and processing techniques are introduced in Section 3. The recognition scheme is detailed in Section 4. Experimental results using the MSTAR public database are shown in Section 5, followed by our conclusions in Section 6.

2. Variation Reduction Techniques

2.1. Image Energy Normalization. The echo strength of SAR is strongly affected by, for example, the range distance between the imaging target and its corresponding radar and several other reasons; therefore the average amplitude of image pixels in different image chips may be different even for the same target [28]. To mitigate the potential influence of amplitude variations in subsequent features extraction, the image energy normalization process needs to be applied. Let M and N denote the number of pixels in range and cross-range dimension for a given SAR image chip. The SAR image chip can be denoted as $X(m,n)$, where $m = 1,\ldots,M$ and $n = 1,\ldots,N$ are the dimension of range and cross-range, respectively. The energy normalized image pixel $X''(m,n)$ can be described as

$$X''(m,n) = \frac{X'(m,n) - X'_{\min}(m,n)}{X'_{\max}(m,n) - X'_{\min}(m,n)}, \quad (1)$$

where $X'_{\min}(m,n)$ and $X'_{\max}(m,n)$ is the minimum and maximum value among all pixels of $X'(m,n)$, respectively, and $X'(m,n)$ is calculated as

$$X'(m,n) = \frac{X(m,n)}{\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N} X^2(m,n)}}. \quad (2)$$

The benefit of employing the image energy normalization process is provided in Section 5.1.

2.2. Pose Rectification. Pose rectification is beneficial for improving the accuracy of SAR ATR and can be achieved by rotating the given images according to the pose of target of interests. However, targets with partial defected contour shapes that are caused by the shadow effect may suffer from poor pose estimation accuracies. This section introduces a pose estimation method that is based on the exploration of targets' geometrical information for achieving higher estimation accuracy.

Several methods have been proposed for achieving higher accuracy in pose estimation. The methods proposed in [29, 30] are based on maximizing the mutual information with multilayer perceptron (MLP). Although a low estimation error is achieved, these methods are computationally expensive and require a long training time. The method proposed in [31] is based on the 2D continuous wavelet transform (CWT), where the orientation that maximizes the angular energy is considered as the estimated pose. However, this method is based on the assumption that the target of interest is already placed in the image centre, which is difficult to achieve especially for SAR images with indistinct targets.

In fact, the tactical ground targets show rectangular shaped boundaries, which can be used for pose estimation. Therefore, methods based on the analysis of the geometrical information of target of interests have been proposed. The methods proposed in [2, 32] are based on finding the encapsulating box of the target of interest, where the basic assumption is that the edges of the estimated box should be tangent to the rectangular shaped target boundaries. However, this is not always true with incomplete target shape boundaries due to the shadow effects in SAR images. Moreover, the least squares linear fit based methods estimate the centreline of the target of interest, where the slope of the centreline is considered as the target pose. However, for similar reasons, the shadow effect in SAR may produce images with defected target, which can affect the corresponding pose estimation results. As discussed, the encapsulating box based methods have failed to achieve the optimum estimation result due to the defect targets in SAR images. However, as will be introduced, the Radon transform based method can achieve better estimation result in such scenarios [33]. Therefore, better estimation accuracy can be achieved by employing these two methods in a well-designed fashion. Firstly, the target of interest is segmented from the SAR image, and the rectangle that has the minimum perimeter around the segmented target is considered as the minimum bounding rectangle (MBR) [34]. Then, the completeness of the target of interest can be evaluated. In the case of a target with complete contour shape in the SAR image, the MBR estimated result is considered as the final result. Otherwise, the Radon transform is conducted and its estimation is used as the final result.

2.2.1. Estimation for Targets with Complete Contour Shapes. Tactical targets in SAR images have randomly distributed poses ranging from 0° to 360° (the target pose is defined as the angle between the target's longer edge and the horizontal image axis). Tactical targets in SAR images show rectangular-like shapes. Figure 2 shows the segmented SAR chips, where the target poses can be estimated according to the inclination angle of its MBR. As introduced in [34], the rectangle that has the shortest perimeter enclosing a convex polygon has at least one side collinear with one of the convex edges. The MBR can be efficiently calculated as follows:

Step 1. Estimate the centroid of the target of interest.

Step 2. Compute the convex polygon of the target of interest.

(a) (b) (c) (d) (e) (f)

Figure 2: Illustration of targets with complete contour shapes.

(a) (b) (c) (d) (e) (f)

Figure 3: Illustration of targets with defected contour shapes.

Step 3. Compute and store the edge orientations of the convex polygon.

Step 4. Rotate a bounding rectangle according to the stored edge orientations until a full rotation is done.

Step 4.1. Find a fitted rectangle.

Step 4.2. Store the perimeter of the fitted rectangle.

Step 4.3. Rotate the rectangle.

Step 5. Return the rectangle corresponding to the minimum perimeter.

2.2.2. Estimation for Targets with Incomplete Contour Shapes. Due to the imaging principle of SAR, partial part of the target of interest is not radiated by radar beam, and therefore the imaged target shows incomplete boundary shape. However, the long edge of the target of interest is always well imaged, as shown in Figure 3. In fact, the Radon transform (RT) can be used for long edge detection. Therefore, for SAR images with targets that show incomplete contour shapes, the RT based estimation can achieve higher accuracy. The application of the RT on a target image $\mathbf{I}(x, y)$ limited by a set of angles can be considered as calculating the projection of the target along given angles. The calculated projection result is the sum of pixel numbers in each single direction, where a line can be found in the corresponding target image according to the peak of the projection result [33]. Define $\mathbf{G}(\rho, \theta)$ as the projection at angle θ with distant ρ to the image centroid, and the RT is implemented as follows:

$$\mathbf{G}(\rho, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{I}(x, y)\, \delta(\rho - x\cos(\theta) - y\sin\theta)\, dx\, dy, \tag{3}$$

where $\delta(\cdot)$ is the Dirac delta function. The parameters ρ and θ determine the projection direction, where the projection is repeated from $\theta \in [0^\circ : 180^\circ)$. Note that a pixel in the RT transform is divided into four subpixels such that accurate projection result can be achieved, where the projection contribution is calculated according to the position of the subpixel that hits the projection bin.

2.2.3. Degree of Overlapping Rectangle. In fact, for any given image, the completeness of the target in SAR images can be automatically calculated. As introduced in Section 2.2.1, the calculated MBR has at least one edge overlap with the target boundary. Therefore, in the case of a complete target, one long edge of the target of interest will overlap with that of its corresponding MBR. In the case of a target with partial defect, the diagonal line of the target of interest may overlap with a long edge of its corresponding MBR with few pixels. Let N_l denote number of pixels of the two MBR long edges, and let N_t denote the number of target pixels that overlap with the two MBR long edges. The completeness of the target in SAR images can be evaluated as follows: the target is firstly dilated, and then the degree of overlapping rectangle is calculated as N_t/N_l, and finally the completeness of the target boundary is evaluated according to the calculated degree of overlapping rectangle. After dilation, since the difference between the complete contour shape and defected contour shape is large, the proposed method is not sensitive to the selected threshold employed for evaluating the degree of overlapping. Overall, as shown in Figure 4, the target pose is estimated using the MBR based method or the RT based method depending on the evaluation result of the degree of overlapping rectangle, and several estimation results are shown in Figure 5.

3. Feature Extraction and Processing Techniques

3.1. Rich Feature Set Extraction. Feature extraction is of crucial importance to the overall performance of the entire ATR system. It is ideally preferable to extract features that have characteristics of high discrimination ability (or, in other words, high interclass variation) and high tolerance to target translation. These feature characteristics can be achieved by efficiently employing the wavelet decomposition technique. As depicted in Figure 6, the texture features are reflected in LL and the horizontal and vertical edge feature are reflected in LH and HL, respectively. HH is actually a combination of features reflected in LH and HL. Furthermore, the translation invariant features can be extracted by sequentially further decomposing the previously decomposed image to a much

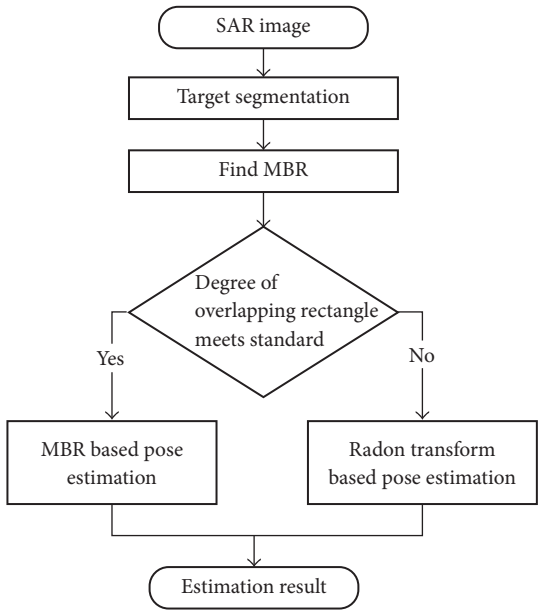


Figure 4: Illustration of the proposed pose estimation method.

coarser resolution. The idea behind the translation invariant features is that each decomposition process throws away the exact positional information of certain feature that exists in a specific area. More specifically, as illustrated in Figure 7, a pixel point in a newly decomposed image implicitly reflects the presence of certain feature(s) in a corresponding entire local region in the original image.

Several wavelet families have been proposed with the shape and duration of the mother wavelets being the main differences among them. The number of vanishing moments (order number) is used as an indication of the wavelets' smoothness and the frequency response flatness of the wavelet filters. It is suggested that we employ one fixed mother wavelet for the entire recognition scheme. A wavelets' comparison test is conducted in [5], where 7 mother wavelets with variations in order numbers are compared, according to minimum distance. To determine the most appropriate mother wavelet, in this paper, we compare 7 mother wavelets with more variations in order numbers with the maximum margin criterion (MMC) [35]. Specifically, we compare discrete Meyer wavelet, Biorthogonal wavelets (orders 1.1, 1.3, 1.5, 2.2, 2.4, 2.6, 2.8, 3.1, 3.3, 3.5, 3.7, 3.9, 4.4, 5.5, and 6.8), Coiflets (orders 1, 2, 3, 4, and 5), Haar wavelet, Daubechies wavelets (orders 2, 3, 4, 7, 10, 25, and 45), Reverse biorthogonal wavelets (orders 1.1, 1.3, 1.5, 2.2, 2.4, 2.6, 2.8, 3.1, 3.5, 3.7, 3.9, 4.4, 5.5, and 6.8), and Symlets (orders 2, 4, 8, and 16).

MMC finds the mother wavelet that maximizes the average margin between classes. This is achieved by comparing the difference between the average within-class distance and the average between-class distance $d_w - d_b$. The mother wavelet that achieves the maximum difference is the best selection. Suppose we have c classes $C_1, C_2, \ldots, C_c$, each class with n_i samples and therefore, $n = \sum_{i=1}^{c} n_i$ samples in total. Let x_j^i denote the jth sample in the ith class, let m_i be the centroid of the ith class, and let m be the centroid of the training set. The average within-class distance d_w and the average between-class distance d_b can be denoted as

$$d_w = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n_i}\left\|x_j^i - m_i\right\|_F^2$$
$$d_b = \frac{1}{n}\sum_{i=1}^{c} n_i \left\|m_i - m\right\|_F^2. \quad (4)$$

The comparison of the discrimination performance of the mentioned wavelets is illustrated in Figure 8. It is noted that the Reverse biorthogonal wavelet 3.1 achieves the highest value, an observation which indicates that it has the highest discrimination ability among these wavelets. Therefore, the Reverse biorthogonal wavelet 3.1 is selected as the default mother wavelet for feature extraction in SAR images.

The above process yields large sets of features which exhibit a high variability as far as the quality of the discriminative information that they convey is concerned. To achieve the truly effective features, the PCA is used, which achieves comparable result to both 2D-PCA and two-stage 2D-PCA when they are employed for SAR feature compression purposes, as analysed in [26]. Moreover, the PCA is much more efficient as far as both computation time and storage space are concerned. The implementation of the PCA is introduced as follows:

Given. Data $X = [x_1, \ldots, x_L]$. Number of principal components k.

Step 1. Subtract the mean of variables from X.

Step 2. Solve the Singular Value Decomposition (SVD) of $X = \mathrm{USV}^T$.

Step 3. The dimensionality reduced feature set is calculated with the first k column of V as XV_k.

3.2. Learn Statistical Relationship among Features. Since access to data arising from true distributions is often not available, the learned models based separately on positive/negative samples are usually not accurate enough for classification. In fact, the discriminative methods construct models from both the positively and negatively labelled samples in a discriminative fashion. Since the final objective is classification, even if the learned distributions may not converge to the true distributions, the constructed discriminative models tend to have better discrimination performance than the generative models [36, 37].

In binary classification case, which can be naturally extended to the more general M-ary classification case, for a given labelled training set $\mathcal{S} := \{(x^{(1)}, y^{(1)}), \ldots, (x^{(L)}, y^{(L)})\}$, where $y^{(l)}$ represents the sample label, each pair $(x^{(l)}, y^{(l)}) \in \mathcal{X}^n \times \{+1, -1\}$ ($\mathcal{X}$ is normally a finite set of integer values as $\mathcal{X} = \{0, \ldots, 255\}$). Supposing we have two models $p(x) := P_{X|Y}(x \mid y = 1)$ and $q(x) := P_{X|Y}(x \mid y = -1)$ that can describe the true distribution of p and q, the log-likelihood

Figure 5: Illustration of the proposed pose estimation method, where the red rectangle is the MBR and the green line is the estimation result of the Radon transform. Note that the Radon transform is not used when the degree of overlapping rectangle meets the specified standard.

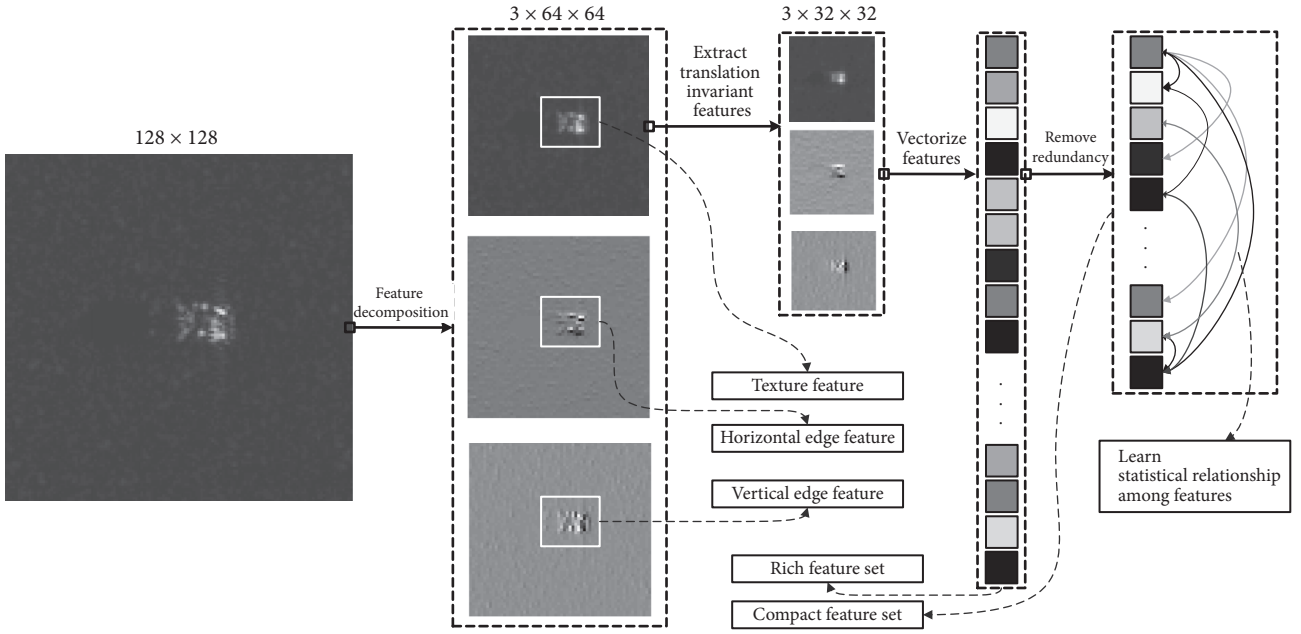


Figure 6: Illustration of the proposed feature extraction and processing technique.

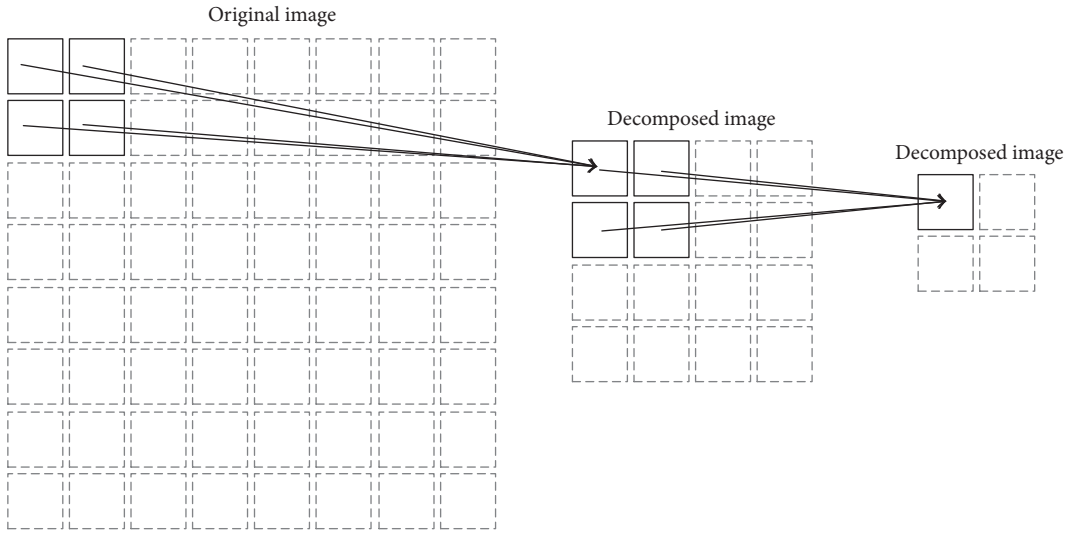


Figure 7: A single pixel point in the decomposed image depicts the existence of features in a corresponding entire local region of the original image.

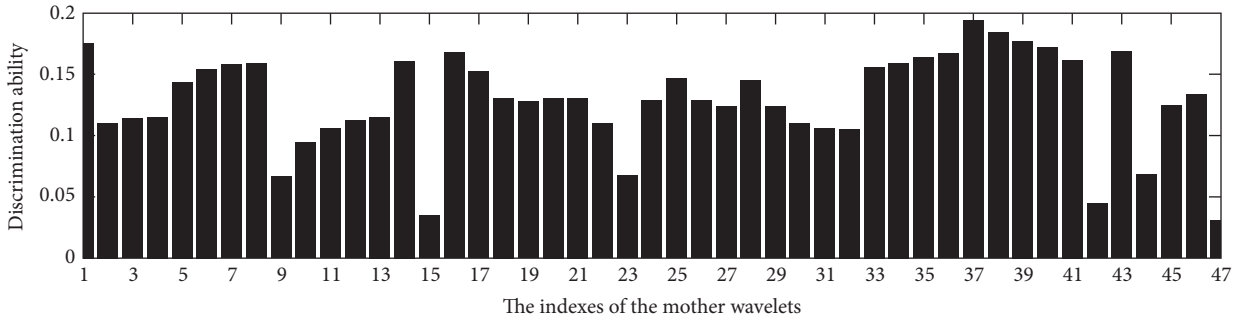


Figure 8: Discrimination ability of various types of mother wavelets. The involved 47 mother wavelets are sequentially discrete Meyer wavelet, Biorthogonal wavelets (orders 1.1, 1.3, 1.5, 2.2, 2.4, 2.6, 2.8, 3.1, 3.3, 3.5, 3.7, 3.9, 4.4, 5.5, and 6.8), Coiflets (orders 1, 2, 3, 4, and 5), Haar wavelet, Daubechies wavelets (orders 2, 3, 4, 7, 10, 25, and 45), Reverse biorthogonal wavelets (orders 1.1, 1.3, 1.5, 2.2, 2.4, 2.6, 2.8, 3.1, 3.5, 3.7, 3.9, 4.4, 5.5, and 6.8), and Symlets (orders 2, 4, 8, and 16). The 37th mother wavelet is the Reverse biorthogonal wavelet 3.1.

ratio test is known to be the optimal test (under both the Neyman-Pearson and Bayesian settings [38])

$$\log \frac{p(x)}{q(x)} \underset{\hat{y}=-1}{\overset{\hat{y}=+1}{\gtrless}} \eta, \tag{5}$$

where η is the threshold [38].

In most cases, it is impossible to have access to the true conditional distributions p and q. Approximations $\widehat{p}$ and $\widehat{q}$ are normally built to learn the unknown distribution from the labelled training set $\mathcal{S}$. Therefore, the log-likelihood ratio test can be rewritten as

$$\log \frac{\widehat{p}(x)}{\widehat{q}(x)} \underset{\hat{y}=-1}{\overset{\hat{y}=+1}{\gtrless}} \eta. \tag{6}$$

The recently proposed method named discriminative tree estimates the multivariate distributions $\widehat{p}$ and $\widehat{q}$ jointly from both the positively and negatively labelled samples in the training set S of Tan et al. [37]. This method is based on the assumption that the learned distribution $\widehat{p}(x)$ is Markov with respect to an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ represents the vertex set and $\mathcal{E} \subset \binom{\mathcal{V}}{2}$ represents the set of all unordered pairs of vertexes. The mentioned Markov conforms to the local Markov property

$$p\left(x_i, x_{\mathcal{V}\setminus i}\right) = p\left(x_i, x_{\mathcal{N}(i)}\right), \quad \forall i \in \mathcal{V}, \tag{7}$$

where $\mathcal{N}(i) \coloneqq \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ represents the set of neighbour nodes of i and $x_{\mathcal{A}} = \{x_i : i \in \mathcal{A}\}$ for any set $\mathcal{A} \subset \mathcal{V}$.

A tree structured distribution $\widehat{p}$ that is Markov with respect to an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be factorized as follows [39]:

$$\widehat{p}(x) = \prod_{i \in \mathcal{V}} \widehat{p}_i(x_i) \prod_{(i,j) \in \mathcal{E}} \frac{\widehat{p}_{i,j}\left(x_i, x_j\right)}{\widehat{p}_i(x_i)\,\widehat{p}_j\left(x_j\right)}, \tag{8}$$

where $\widehat{p}_i(x_i)$ represents the marginal of the random variable x_i and $\widehat{p}_{i,j}(x_i, x_j)$ represents the pairwise marginal of the pair (x_i, x_j).

Based on this, for a given distribution p, the projection of p onto some tree distribution $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as follows:

$$\widehat{p}(x) \coloneqq \prod_{i \in \mathcal{V}} p_i(x_i) \prod_{(i,j) \in \mathcal{E}} \frac{p_{i,j}\left(x_i, x_j\right)}{p_i(x_i)\,p_j\left(x_j\right)}. \tag{9}$$

We digress here to introduce the method for constructing models in generative fashion and then provide the method for constructing models in discriminative fashion. The generative methods attempt to construct a model that is the same as the underling model of the classification target. The widely researched generative method, namely, the Chow-Liu algorithm [40], employs the KL-divergence as the measure of the differences between two probability distributions p and $\widehat{p}$. The optimization in the Chow-Liu algorithm is therefore defined as

$$\min_{\widehat{p} \in \mathcal{T}} D\left(p \parallel \widehat{p}\right) \coloneqq \min_{\widehat{p} \in \mathcal{T}} E_p \log\left(\frac{p}{\widehat{p}}\right), \tag{10}$$

where $\widehat{p} \in \mathcal{T}$ states that $\widehat{p}$ is a tree structured distribution over the same alphabet as $\mathcal{T}$. It is shown by Chow and Liu that this optimization problem can be solved by using a maximum weight spanning tree (MWST) algorithm (e.g., Kruskal's [41]) where the mutual information is used to represent the edge weights between pairs of variables.

In contrast, the recently proposed discriminative method employs the J-divergence as the measure of the separation between two probability distributions p and q. The J-divergence is defined as follows [42]:

$$J(p, q) \coloneqq D\left(p \parallel q\right) + D\left(q \parallel p\right). \tag{11}$$

The optimization problem reduces to two tractable MWST problems for maximizing the tree approximate J-divergence over the two tree structured-distributions $\widehat{p}$ and $\widehat{q}$ for known empirical distributions $\widetilde{p}$ and $\widetilde{q}$, which is defined as

$$\left(\widehat{p}, \widehat{q}\right) = \underset{\widehat{p} \in \mathcal{T}_{\widetilde{p}}, \widehat{q} \in \mathcal{T}_{\widetilde{q}}}{\operatorname{argmax}} \widehat{J}\left(\widehat{p}, \widehat{q}; \widetilde{p}, \widetilde{q}\right), \tag{12}$$

where

$$\widehat{J}\left(\widehat{p},\widehat{q};\widetilde{p},\widetilde{q}\right) \coloneqq \sum_{x\in\mathcal{X}^n}\left(\widetilde{p}\left(x\right)-\widetilde{q}\left(x\right)\right)\log\left[\frac{\widehat{p}\left(x\right)}{\widehat{q}\left(x\right)}\right]. \tag{13}$$

It is noted that, as described in [37], (13) can be decoupled into two independent optimization problems:

$$\begin{aligned}\widehat{p} &= \underset{p\in\mathcal{T}_{\widetilde{p}}}{\operatorname{argmin}} D\left(\widetilde{p}\parallel p\right)-D\left(\widetilde{q}\parallel p\right)\\ \widehat{q} &= \underset{q\in\mathcal{T}_{\widetilde{q}}}{\operatorname{argmin}} D\left(\widetilde{q}\parallel q\right)-D\left(\widetilde{p}\parallel q\right).\end{aligned} \tag{14}$$

These can be solved by the MWST algorithm

$$\psi_{i,j}^{(+)} \coloneqq E_{\widetilde{p}_{i,j}}\left[\log\frac{\widetilde{p}_{i,j}}{\widetilde{p}_i\widetilde{p}_j}\right]-E_{\widetilde{q}_{i,j}}\left[\log\frac{\widetilde{p}_{i,j}}{\widetilde{p}_i\widetilde{p}_j}\right]. \tag{15}$$

Overall, the procedure of the learning of the discriminative tree is summarized in the following steps [37]:

Given. Training set $\mathcal{S}$.

Step 1. Estimate the pairwise statistics $\widetilde{p}_{i,j}(x_i,x_j)$ and $\widetilde{q}_{i,j}(x_i,x_j)$ for all edges (i,j).

Step 2. Calculate edge weights $\{\psi_{i,j}^{(+)}\}$ and $\{\psi_{i,j}^{(-)}\}$ for all edges (i,j).

Step 3. Find the optimal tree structures with the given edge weights.

Step 4. Set $\widehat{p}$ and $\widehat{q}$ to be the projection of $\widetilde{p}$ onto $\mathscr{C}_{\widehat{p}}$ and $\widetilde{q}$ onto $\mathscr{C}_{\widehat{p}}$, respectively.

Step 5. Classify the test sample x using the learned distributions $\widehat{p}$ and $\widehat{q}$ in a likelihood ratio test $h(x) = \operatorname{sgn}[\log(\widehat{p}(x)/\widehat{q}(x))]$.

Since the classification result is finally determined by the numerical result of the log-likelihood ratio test, we choose to employ one fixed threshold 0 for the entire training process. This is because likelihoods larger than 0 indicate higher probability of belonging to $p(x)$. Similarly, likelihoods smaller than 0 indicate high probability of belonging to $q(x)$.

4. Recognition Scheme

The main aim of classifier construction in ATR is to convert a wealth of training data into useful knowledge for classification by learning. However, a classifier learned from massive amounts of high varying data is not guaranteed to achieve good performance in classification and may yield large feature dimensions. Therefore, it is of great importance to find effective representations for the targets of interest to be used for constructing the classifiers.

Extracted features might comprise large sets of features which at a glance might be worth of exploiting but turn out to be too "messy" and high in redundancy. In fact, the learning process of the classifiers could be enormously benefited from a feature dimensionality reduction process after the acquisition of the extracted features as previously discussed. The redundancy-reduced features can be used for learning classifiers, where efficient classifiers and better classification accuracy results can be achieved. Therefore, it is suggested to enlarge the quantity of the potential features but then eliminate the existing redundancy, reduce the dimensionality of the enlarged feature set, and finally exploit the preserved features for classification, which comprise the characteristics of proper combination of both quality and quantity. In the proposed recognition scheme, features are extracted with wavelet decomposition, but then the dimension of the feature set is reduced to provide a feature set rich in discriminative information but with limited dimensionality and less redundancy. To make the most of the extracted features, tree structured classifiers are learned in discriminative fashion based on the statistical information provided by the training data of the target classes. In the learned classifiers, the feature nodes are connected as a spanning tree, where each node is connected to another node which has the maximum relevance. Moreover, the relevance between feature nodes can be accordingly calculated. Finally, classifiers are combined using the Real-AdaBoost algorithm to construct the final classifier that has high classification accuracy and is less prone to overfitting, where the recently proposed discriminative trees are involved as the base classifiers. A generic sequence of steps of the proposed scheme is illustrated in Figure 9.

4.1. Construct a Strong Classifier. Efforts have been constantly made to construct a classifier with high classification accuracy and strong generalization ability (the later meaning that performance of the classifier learned from a given training dataset will still be good when the classifier is exposed to unseen data) [43]. Employing ensemble learning methods is one of the solutions. Ensemble learning methods construct and combine a set of base classifiers instead of constructing and using one single classifier learned from the training dataset. Base classifiers can be generated from a training dataset with the use of any learning algorithm (e.g., decision tree, graphical models, and neural networks).

AdaBoost is one of the ensemble methods that have achieved great success in diverse domains [27, 43–47]. The general idea of the AdaBoost is to constantly update the distribution of the training data such that the learning of the base classifiers in each iteration focuses more on the wrongly labelled samples by the previous learned base classifiers. Real-Adaboost is a variant of the AdaBoost which has been empirically proved to have better performance than ordinary AdaBoost (Discrete-AdaBoost) [27, 37, 44]. Specifically, for a given training dataset $\mathcal{S}$, each sample is assigned with an initial weight $\omega_0^{(l)} = 1/L$, where L is the number of training samples. A base classifier is learned in each iteration t such that $h_t : \mathcal{X}^n \to \mathbb{R}$, where a larger absolute value in $h_t(x)$ indicates higher confidence. Then, the samples wrongly labelled by $h_t(x)$ are increased in weights such that the constructed classifiers in the following iterations can focus on the misclassified samples. Finally, the combined classifier resulting after T iterations is

$$H_T\left(x\right) = \operatorname{sgn}\left[\sum_{t=1}^{T}\alpha_t h_t\left(x\right)\right], \tag{16}$$

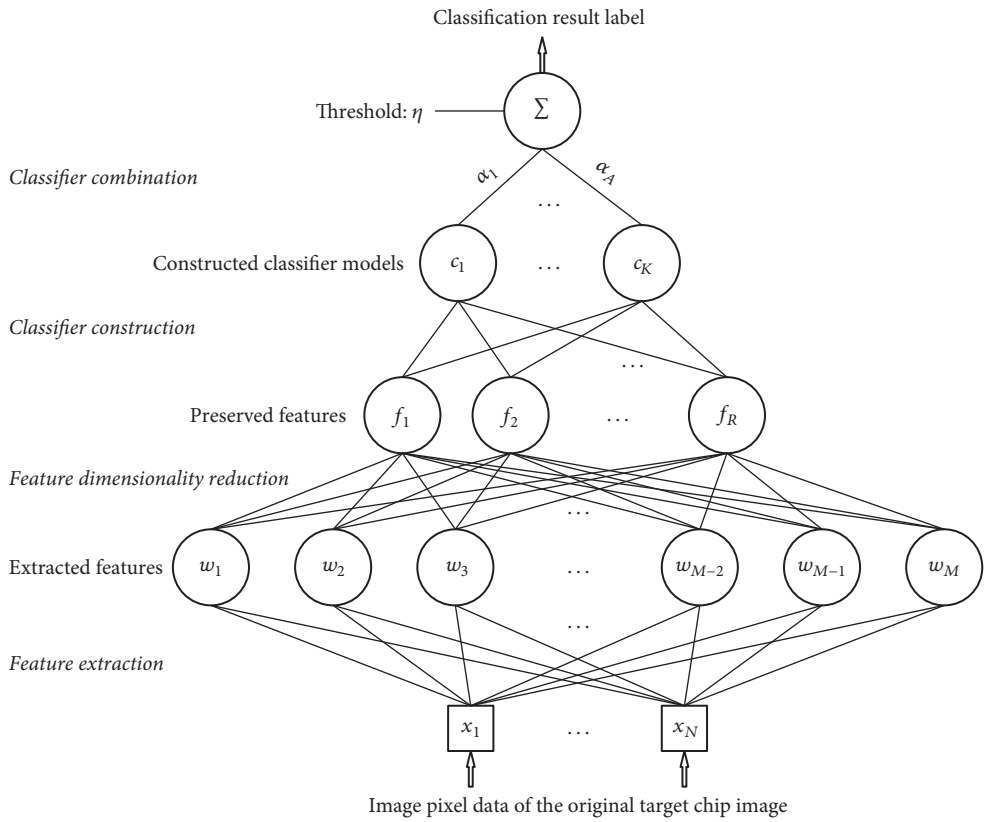


Figure 9: A generic diagram that depicts the various steps of the proposed SAR ATR scheme.

where sgn is the sign function that $\mathrm{sgn}(a) = 1$ if $a \geq 0$ and -1 otherwise and α_t is the coefficient calculated in each iteration for minimizing the weighted training error. Overall, the Real-AdaBoost algorithm trains a set of base classifiers sequentially and combines them to a strong classifier, where the current learned base classifiers focus more on the wrongly labelled samples by the previous base classifiers.

The ensemble process of the Real-AdaBoost is iterated with the rearrangement of the training set distribution while the learning method of the base classifiers is not changed. For the learning of the base classifier in each iteration t, the group of redundancy and dimensionality reduced wavelet features are employed and fed to learn the discriminative trees for classifier construction. By employing the learning method introduced in Section 3.2, a pair of discriminative trees is constructed to provide an estimation of the classification result. Specifically, the pair of discriminative trees constitutes a base classifier for the Real-AdaBoost $h_t : \mathcal{X}^n \to \mathbb{R}$, where $h_t(x) = \log[\widehat{p}_t(x)/\widehat{q}_t(x)]$, and $\widehat{p}_t$ and $\widehat{q}_t$ denotes the learned discriminative tree models at the tth iteration of the Real-AdaBoost. After T iterations, T pairs of discriminative trees are learned and combined to construct a stronger classifier with better approximation of the classification result which can be written as Viola and Jones [45]

$$H_T(x) = \mathrm{sgn}\left[\sum_{t=1}^{T} \alpha_t \log\left(\frac{\widehat{p}_t(x)}{\widehat{q}_t(x)}\right)\right]$$

$$= \mathrm{sgn}\left[\log\left(\frac{\prod_{t=1}^{T} \widehat{p}_t(x)^{\alpha_t}}{\prod_{t=1}^{T} \widehat{q}_t(x)^{\alpha_t}}\right)\right]$$

$$= \mathrm{sgn}\left[\left(\frac{\widehat{p}^*(x)}{\widehat{q}^*(x)}\right)\right], \tag{17}$$

where $\widehat{q}^*(x) = \prod_{t=1}^{T} \widehat{p}_t(x)^{\alpha_t}$ and $\widehat{q}^*(x) = \prod_{t=1}^{T} \widehat{q}_t(x)^{\alpha_t}$.

For the iterative updating of the training set distribution, the misclassified samples are reassigned with larger weights and the correctly classified samples are reassigned with smaller weights compared to their previous weights. Regarding the weight distribution updating problem, simply reduplicating the samples with higher weights is time and computation inefficient. This is because as the number of iterations increases, the wrongly labelled samples would be much less in number but have much larger weights. Therefore, the final training set is fixed in size and constructed in random sampling fashion, where samples of the original training set are chosen according to the updated distribution weights. The entire classifier construction scheme is summarized as below:

Given. Training dataset S. Number of iterations T.

Step 1. Wavelet feature extraction from the given training dataset S.

Step 2. Redundancy and dimensionality reduction for the extracted features.

Step 3. Initialization of the distribution weights, $w_0^{(l)} = 1/L$ for all $1 \le l \le L$.

Step 4. Classifier construction

(1) *for* $t = 1 : T$ *do*

(2) Learn the pair of discriminative trees $\widehat{p}_t$, $\widehat{q}_t$ from the weighted empirical distributions $\widetilde{p}_w$ and $\widetilde{q}_w$.

(3) Get the base classifier $h_t(x) := \log[\widehat{p}_t(x)/\widehat{q}_t(x)]$.

(4) Calculate the coefficient α_t

$$\alpha_t = \frac{1}{2}\log\frac{1-\sum_{i=1}^{L} w_t^{(l)} y^{(l)} \operatorname{sgn}\left(h_t\left(x^{(l)}\right)\right)}{\sum_{i=1}^{L} w_t^{(l)} y^{(l)} \operatorname{sgn}\left(h_t\left(x^{(l)}\right)\right)}. \quad (18)$$

(5) Update the weighted empirical distribution:

$$w_{t+1}^{(l)}(i) = \frac{w_t^{(l)} \exp\left[-\alpha_t y^{(l)} h_t\left(x^{(l)}\right)\right]}{\zeta_t}, \quad \forall l = 1, \ldots, L, \quad (19)$$

where ζ_t is the normalization factor (to ensure that $w_{t+1}^{(l)}$ will be a distribution).

(6) *end for*

Step 5. Output the final classifier $h_t(x) = \log[\widehat{p}_t(x)/\widehat{q}_t(x)]$ with coefficients $\{\alpha_t\}_{t=1}^{T}$.

4.2. Multiclass Classification. The One-vs.-One (OvO) and One-vs.-All (OvA) are the two most popular strategies for the extension of a two-class classification (binary classification) problem to a multiclass classification (multinomial classification) [48]. For a K class problem, the OvO strategy trains $K(K-1)/2$ binary classifiers, each of which classifies a pair of classes selected from the original training set. For the classification of the unseen samples, the samples are fed and tested in all $K(K-1)/2$ classifiers by employing a voting scheme where the class which achieves the highest number of positive predictions would be considered as the final prediction. The OvA strategy trains one classifier for every class where the samples of the target class are considered as positive samples and all of the rest of the samples as negative samples. At predication stage, the unseen sample is assigned with the label of class k if its corresponding classifier produces the highest likelihood score.

5. Experimental Results

In this section, the performance of the proposed scheme is evaluated and compared with several established methods. The widely used SAR ATR experimental validation and comparison benchmark moving and stationary target acquisition and recognition (MSTAR) public release database is employed for performance evaluation [49–51]. The MSTAR database consists of 10 vehicle classes, which are collected by X-band SAR with 1-ft by 1-ft resolution, including BMP2, BTR70, T72, BTR60, 2S1, BRDM2, D7, T62, ZILI131, ZSU234, and SLICY. The collection of target images is captured under various depression angles and aspect angles, which are suitable for testing the SAR ATR methods with targets under various operating conditions.

There are two categories of operating conditions in the MSTAR database: the standard operating conditions (SOCs) and the extended operating conditions (EOCs) [50]. The targets captured under SOCs are listed in Table 1 including information about vehicle types, number of chip images, serial numbers, and depression angles. It is worth noting that the EOCs are much more difficult for SAR ATR than the SOCs. In EOC-1, the depression angles are larger in variation where the training images are captured under 15° and the testing images are captured under 30°, as shown in Table 2.

Table 1: 10 classes of targets of the MSTAR dataset under SOCs.

Training set			
Vehicle	Number of images	Serial number	Depression angle
BMP2	699	9563, 9566, C21	17°
BTR70	233	C71	17°
T72	699	132, 812, S7	17°
BTR60	256	k10yt7532	17°
2S1	299	B01	17°
BRDM2	299	E71	17°
D7	299	92v13015	17°
T62	299	A51	17°
ZILI131	299	E12	17°
ZSU234	299	D08	17°
Testing set			
Vehicle	Number of images	Serial number	Depression angle
BMP2	587	9563, 9566, C21	15°
BTR70	196	C71	15°
T72	588	132, 812, S7	15°
BTR60	196	k10yt7532	15°
2S1	274	B01	15°
BRDM2	274	E71	15°
D7	274	92v13015	15°
T62	274	A51	15°
ZILI131	274	E12	15°
ZSU234	274	D08	15°

Table 2: 4 classes of targets of the MSTAR dataset under EOC-1.

Training set			
Vehicle	Number of images	Serial number	Depression angle
2S1	274	B01	15°
BRDM2	274	E71	15°
ZSU234	274	D08	15°
T72	274	A64	15°
Testing set			
Vehicle	Number of images	Serial number	Depression angle
2S1	288	B01	30°
BRDM2	288	E71	30°
ZSU234	288	D08	30°
T72	288	A64	30°

Table 3: 5 variants of T72 with different serial numbers of the MSTAR dataset under EOC-2.

Training set			
Vehicle	Number of images	Serial number	Depression angle
BMP2	233	C21	17°
BRDM2	298	E71	17°
BTR70	233	C71	17°
T72	233	132	17°
Testing set			
Vehicle	Number of images	Serial number	Depression angles
T72	419	S7	15° and 17°
T72	572	A32	15° and 17°
T72	573	A62	15° and 17°
T72	573	A63	15° and 17°
T72	573	A64	15° and 17°

In EOC-2, the training and testing set have various versions of T72 with different serial numbers, as shown in Table 3.

We experiment with both two-level and three-level two-dimensional wavelet decomposition with respect to the Reverse biorthogonal wavelet (the selected mother wavelet as introduced in Section 3.1). In the following, wavelet 768 ($256 \times 3 = 768$) and wavelet 192 ($64 \times 3 = 192$) are used to denote the two-level and three-level wavelet decomposition, respectively. The stopping criterion of the Real-AdaBoost is set to 400 iterations. The segmentation of the target of interest is implemented with the MRF model based method, where the potential class number is 2, the expectation is 0.4, and the maximum iteration number is 50. The segmented target is dilated with a disk-shaped template with radius 3. The degree of overlapping rectangle is 0.5 indicating that an appropriate MBR must have more than 50% long edge overlapping pixels in terms of the target of interest. Moreover, the proposed method is implemented using Matlab R2013a and tested on a computer with 1.8 GHz CPU and 4 GB RAM. Regarding the computation complexity, for a classifier trained for classifying 10 targets in OvO fashion, the processing time for one single sample takes less than 0.02 s, including the processes of extraction and compressing of features and recognition of targets.

Before applying the proposed method to SAR ATR and comparing with other methods, it is necessary to test the proposed method in conjunction with several important processes, including image energy normalization, feature extraction, extension of two-class to multiclass classification, and pose rectification. These four tests are conducted in Sections 5.1 to 5.4, and the comparisons of recognition accuracy performance with other methods are provided in Section 5.5.

5.1. Image Energy Normalization. The significance of image energy normalization in SAR ATR is tested in this section, where the performance of the proposed scheme is tested with or without image energy normalization processing. The dataset includes all 10 classes captured under SOCs as listed in Table 1. The wavelet 192 is used for feature extraction.

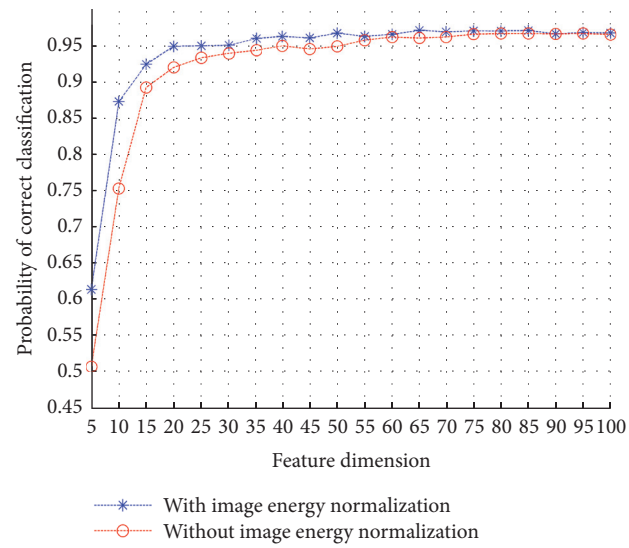


Figure 10: Performance comparison among the two cases which refer to the employment or not of image energy normalization.

It is noticed in Figure 10 that the involvement of normalization before feature extraction is beneficial for improving classification accuracy. In fact, as the dimension of feature vectors employed for classification grows, the advantage of image energy normalization diminishes. This is because a larger training feature set provides more information for classification, where the classifier is empowered with more discrimination ability by exploiting the provided information. However, the classification with normalization achieves good classification accuracy (around 96%) even when the feature vector dimension is much lower, yielding an accuracy which is almost the same as the accuracy achieved with higher feature dimensions. Therefore, it is still suggested to employ image energy normalization for preprocessing, especially for classifiers constructed from training feature sets of lower dimensionality. In the following, the image energy normalization process is employed as a standard default processing step.

5.2. Extension to Multiclass. We compare the OvO and OvA strategies on the same training set (all 10 classes under SOCs) to test their performance on the SAR ATR problem. It is noted in Figure 11 that the OvO strategy appears to be outperforming the OvA strategy marginally in the SAR ATR problem. The marginal differences in recognition accuracy lie in the unbalance of the training set, where the OvA strategy employs the positive sample classes that are much less in quantity than the negative sample classes. In fact, the advantage of the OvA strategy is that it is less in computation and time complexity, where the OvO constructs 45 classifiers and the OvA constructs 10 classifiers for a 10-class problem, respectively. Since the aim of this paper is to provide a SAR ATR scheme with high recognition accuracy, OvO is employed as the default strategy for solving the multiclass problem.

5.3. Feature Extraction. In this section, we compare the performance of feature extraction using the wavelet 192

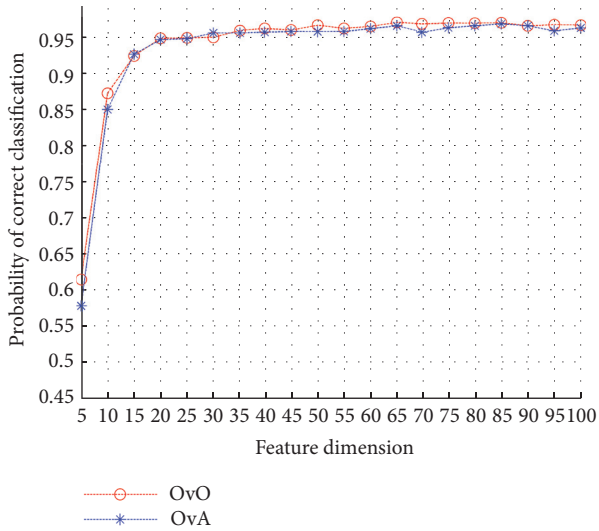


Figure 11: Performance comparison between OvO and OvA strategies.

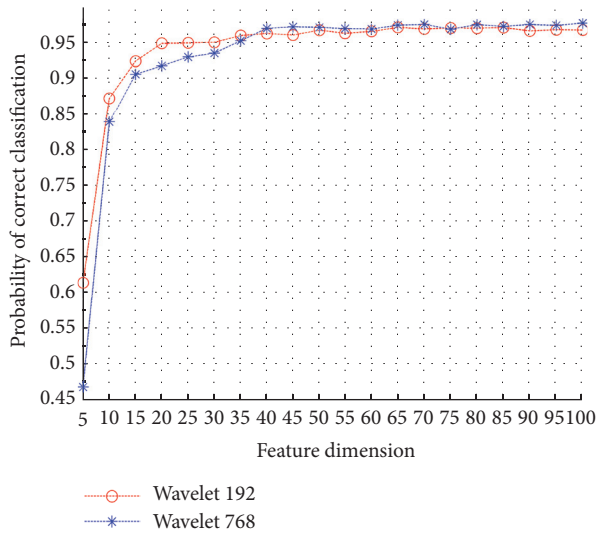


Figure 12: Performance comparison between wavelet 192 and wavelet 768.

(three-level wavelet decomposition) and the wavelet 768 (two-level wavelet decomposition). All 10 classes captured under SOCs are employed for both training and testing. As illustrated in Figure 12, these two curves coupled with each other. The wavelet 192 outperforms the wavelet 768 when feature vectors possess lower dimensions. However, this situation changes as the dimension of feature vectors grows to 40. Moreover, the best classification result (97.46%) is achieved by the use of wavelet 768 with feature dimension of 70. This is because the wavelet 768 provides more features for discrimination and the proposed ATR scheme constructs and combines several discriminative tree classifiers that make the most of the discriminative information inherent to the extracted features.

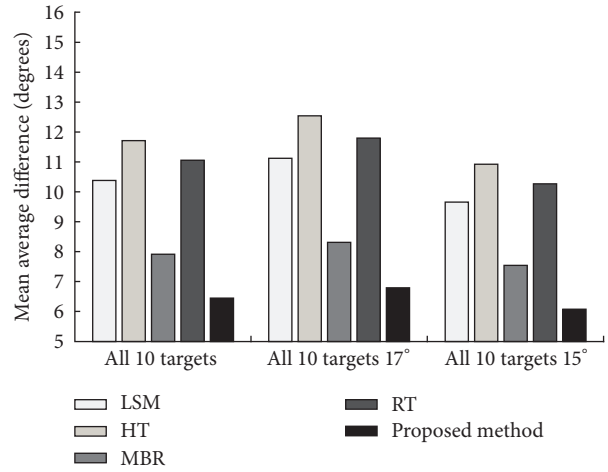


Figure 13: Performance comparison for all 10 targets at both depression angles of 17° and 15°.

5.4. Pose Rectification

5.4.1. Pose Estimation. To test the performance of the proposed pose estimation method, the estimation results of the proposed methods are compared to the ground truth of target poses (the azimuth information provided in the MSTAR database). The correctness of the estimation results is evaluated with the so-called mean absolute difference (MAD), which is calculated as $\text{Err} = 1/n \sum_{i=1}^{n} |E_t(i) - E_e(i)|$. The MAD reveals the actual estimation error about the ground truth in comparison to the mean error (ME), since it prevents the offset of the positive and negative errors. Moreover, the performance of the proposed method is evaluated and compared with several widely cited methods, such as the least square method (LSM) based estimation, the Hough transform (HT) based method, the MBR based method, and the Radon transform (RT) based method.

All 10 targets captured with different depression angles and target poses are involved to test the robustness of the proposed method over depression angle variations. The evaluation results for the 10 targets at depression angles 17° and 15° are listed in Tables 4 and 5, respectively. All serial number variants of BMP2 and T72 in MSTAR dataset are involved to test the robustness over variation in serial numbers. The evaluation results of the data captured at depression angles 17° and 15° are listed in Tables 6 and 7, respectively. It is noted that the MBR based method has achieved much lower estimation error in comparison to other methods. However, the performance of the MBR based method has estimation error higher than 10° in several tests. More specifically, the MBR based method achieves the highest estimation error 15.32° for the BRDM2 captured at depression angle of 15°. In comparison with these methods, the proposed method achieves the lowest estimation error in all tests (lower than 10°).

Furthermore, the average estimation error of the above tests is illustrated as bar figure in Figures 13 and 14, such that a much more distinct comparison can be observed. Similarly, the proposed method is compared to the least square method (LSM) based estimation, the HT based method, the MBR

Table 4: MAD evaluation of the proposed method for all 10 targets at depression angle 17° (degrees).

Vehicle	BTR60	2S1	BRDM2	D7	T62	ZIL131	ZSU234	BMP2	BTR70	T72
LSM	7.05	11.05	15.48	11.93	10.47	10.79	15.42	9.39	9.33	10.22
HT	9.17	11.26	12.14	18.86	14.69	12.34	15.16	10.72	8.09	12.64
MBR	6.10	7.45	14.66	6.27	7.18	11.87	8.23	6.79	9.38	4.99
RT	7.23	9.55	10.38	19.32	13.32	9.27	18.40	9.54	7.49	13.73
Proposed method	**4.85**	**5.84**	**8.70**	**6.27**	**9.39**	**8.02**	**7.83**	**5.43**	**6.67**	**4.51**

Table 5: MAD evaluation of the proposed method for all 10 targets at depression angle 15° (degrees).

Vehicle	BTR60	2S1	BRDM2	D7	T62	ZIL131	ZSU234	BMP2	BTR70	T72
LSM	5.52	10.63	15.21	10.09	9.20	8.67	14.07	7.56	7.00	6.87
HT	7.00	10.16	10.96	16.72	13.49	10.13	14.32	9.02	6.87	10.38
MBR	5.11	6.64	15.32	6.40	6.93	10.23	8.81	5.12	7.01	**3.53**
RT	4.76	9.63	9.38	17.36	12.34	7.08	17.31	8.27	5.85	10.61
Proposed method	**3.81**	**5.42**	**9.04**	**6.32**	**6.10**	**5.89**	**7.67**	**4.45**	**5.13**	3.38

Table 6: MAD evaluation of the proposed method for all serial number variants of BMP2 and T72 at depression angle 17° (degrees).

Vehicle type	BMP2			T72		
	132	812	s7	9563	9566	c21
LSM	17.15	9.58	13.12	13.53	13.22	12.06
HT	18.87	14.81	16.23	12.96	13.04	13.76
MBR	7.97	5.50	6.40	11.90	9.96	8.71
RT	21.46	15.15	17.62	11.47	13.65	12.25
Proposed method	**7.93**	**5.42**	**5.79**	**8.47**	**7.94**	**6.96**

Table 7: MAD evaluation of the proposed method for all serial number variants of BMP2 and T72 at depression angle 15° (degrees).

Vehicle type	BMP2			T72		
	132	812	s7	9563	9566	c21
LSM	12.58	9.30	10.74	11.26	10.96	9.71
HT	15.98	12.36	13.32	9.44	9.63	11.58
MBR	7.75	4.85	4.53	8.57	6.93	6.57
RT	17.16	11.31	13.62	9.38	10.31	10.61
Proposed method	**7.21**	**4.60**	**4.34**	**6.96**	**5.63**	**5.71**

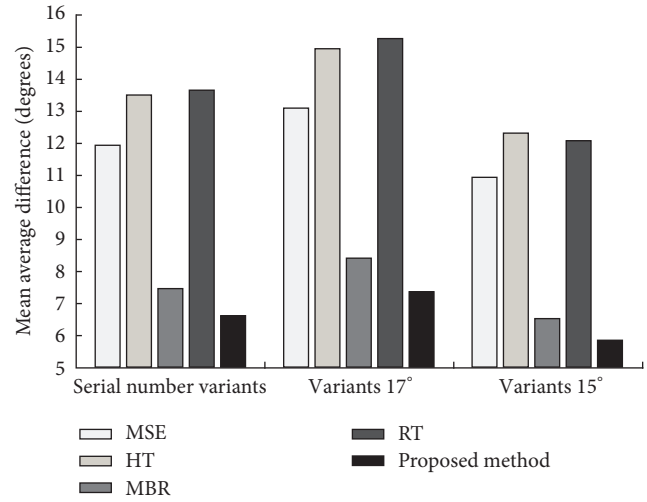


Figure 14: Performance comparison for serial number variants of BMP2 and T72 at both depression angles of 17° and 15°.

based method, and the RT based method. It is noted that, for most of these methods, higher estimation error is achieved in the serial number variation test. Compared with these methods, the proposed method achieves robust and accurate estimation results in both tests. Specifically, the proposed method achieves the lowest average estimation error in all tests (lower than 8°).

5.4.2. Pose Rectification. The various target poses introduce great variations into the SAR images. It has been experimentally proven in several researches that rotating images in certain directions or introducing rotationally invariant features is beneficial for improving classification accuracy [2, 3]. To this end, we rotate the image according to the target poses in the SAR images, which is named as pose rectification. In this section, we test the performance of the proposed scheme with or without pose rectification using the same training and testing set (all 10 classes under SOC). The SAR images are rotated anticlockwise according to their poses. As can be seen from Figure 15, the classification with pose rectification universally outperforms the classification without pose rectification. It is also noted that the best classification accuracy (99.3%) is achieved by the wavelet 768 with feature dimension of 75. These results meet our expectation that the classification can benefit from eliminating the pose variations in SAR images. Specifically, the rectification of poses provides target images for classification with fewer variations.

5.4.3. Outlier Rejection Performance. To evaluate the outlier rejection performance of the proposed method, a varying threshold for the log-likelihood test, which is introduced in Section 3.2, was incorporated to provide the ROC curve. BTR70, BMP2, and T72 listed in Table 1 are involved for classifier training and the SLICY set is involved as confusers

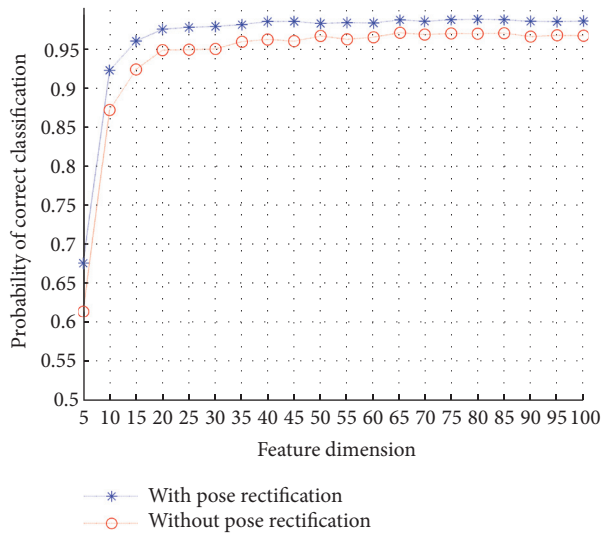


Figure 15: Performance comparison among the two cases which refer to the employment or not of pose rectification.

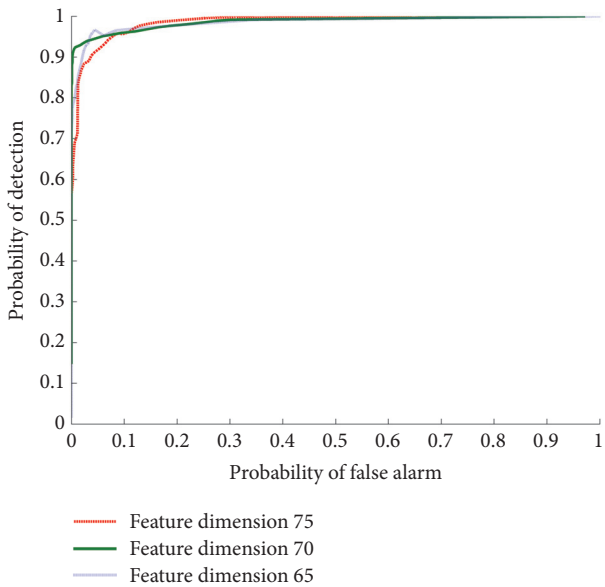


Figure 16: The outlier rejection performance of the proposed method with wavelet 768 compressed with different dimensions.

with 1168 image chips, that is, 210 chips captured at 15°, 298 chips captured at 16°, 386 chips captured at 17°, and 274 chips captured at 29°. As shown in Figure 16, at the probability of detection $P_d = 90\%$, the probability of false alarm for feature dimension of 75 is $P_{\text{fa}} = 2.34\%$. It is clear that the proposed method is robust at rejecting confuser targets.

5.5. ATR Performance Comparisons. The effectiveness of the proposed ATR scheme is tested in this section. Several widely cited methods are involved for performance comparison, for example, the Extended Maximum Average Correlation Height Filter (EMACH) [53], the Support Vector Machine (SVM) classifier with Gaussian kernel [52], feature fusion via AdaBoost with neural networks as the base classifiers [2], and the Iterative Graph Thickening (IGT) approach [24]. The best result obtained from the proposed method is used to compare with other methods.

Table 8 lists the performance comparison of the mentioned methods under SOCs. It is noted that the proposed method has achieved significant improvements in classification accuracy in comparison with other methods. A majority of the classes are correctly classified with 100% accuracy and the rest have P_{cc} higher than 98%, which is also much higher than other methods. Moreover, the superiority of the proposed method is strengthened by the fact that the average P_{cc} (99.3%) is much higher than the second highest average P_{cc} (84.8%).

Four distinct target classes are involved in the following test: EOC-1, including 2S1, BRDM2, T72, and ZSU234, as listed in Table 2. All of these four classes are involved in training and testing stages. The only difference is that the training and testing set are captured under depression angles 15° and 30°, respectively. The increase in depression angle variations introduces a bigger challenge to the classification problem. It is noted in Table 9 that the classification accuracy of the most of the mentioned methods is lower than 88% under EOC-1, where the superiority of the proposed method (higher than 96%) is obvious. Furthermore, the average classification accuracy of the proposed method is 97.5% which is much higher than the other listed methods.

The training dataset under EOC-2 is composed of four different target classes, BMP2, BRDM2, BTR70, and T72, as summarized in Table 2. This test aims at testing the performance of the SAR ATR algorithms with significant different in serial numbers and configurations. The testing set has only the T72 family with five different serial numbers and the training set is composed of all these four mentioned classes. In addition, the training set was obtained at 17° while the testing set was obtained at depression angles of both 15° and 17° as shown in Table 3. Table 10 lists the performance comparison of the mentioned methods under EOC-2. The improvement in classification accuracy is substantial since the average P_{cc} of the proposed method is 96.9% which is much higher than the second highest 84.8%.

5.6. Performance Comparison of Variations in Target Poses. As analysed in Section 5.4, the involvement of pose rectification is beneficial for improving the classification performance. In fact, a single target will exhibit different appearances when it is captured under various poses. In this section, we conduct an experiment to test the influence of the appearance differences introduced by the pose variations. The experimental database is almost the same as the data listed in Table 1 except that only one single serial number of each target is involved for training and testing (C21 for BMP2 and S7 for T72). The images for training are selected from the training database with different sample steps of target poses (varied from 1° to 7°), where 51 images are selected for the training of each target. For example, the poses for Step 1 are $1°, 2°, \ldots, 51°$, the poses for Step 2 are $1°, 3°, \ldots, 101°$, and the poses for Step 7 are $1°, 8°, \ldots, 351°$. Additionally, we have also investigated the possibility of training classifiers using training databases with different sizes, for example,

Table 8: Confusion matrix of EMACH, method proposed in [52], method proposed in [2], IGT, and the proposed method tested under SOCs (P_{cc} (%)).

Confusion matrix of EMACH, the method proposed in [52], and the proposed method										
Vehicle	BMP2	BTR70	T72	BTR60	2S1	BRDM2	D7	T62	ZILI131	ZSU234
BMP2	90/90/**100**	2/2/0	4/3/0	1/1/0	1/1/0	0/2/0	0/0/0	1/0/0	0/1/0	1/0/0
BTR70	2/3/0	93/90/**100**	1/3/0	0/0/0	1/0/0	1/2/0	0/0/0	2/0/0	0/0/0	0/2/0
T72	2/2/0	0/1/0	96/93/**100**	0/3/0	1/0/0	0/0/0	0/0/0	0/0/0	1/1/0	0/0/0
BTR60	0/2/0	1/2/0	0/1/2	95/92/**98**	1/0/0	0/0/0	3/3/0	0/0/0	0/0/0	0/0/0
2S1	5/5/0	6/3/0	4/2/1	2/0/0	74/81/**99**	3/3/0	1/2/0	2/3/0	1/0/0	2/1/0
BRDM2	3/6/1	6/8/0	3/2/0	0/1/0	1/0/0	84/79/**99**	2/0/0	0/3/0	0/0/0	1/1/0
D7	2/0/1	3/0/0	2/0/0	1/0/0	0/1/0	0/0/0	85/98/**99**	3/0/0	2/0/0	2/1/0
T62	1/1/0	1/0/0	1/0/0	1/1/0	4/0/1	0/0/0	0/0/0	86/91/**99**	4/4/0	2/3/0
ZILI131	2/2/0	0/1/0	1/0/0	2/0/0	0/0/0	0/0/0	0/0/0	4/0/0	88/95/**100**	3/2/0
ZSU234	1/0/0	0/1/0	4/0/1	2/3/0	0/0/0	0/0/0	0/1/0	1/0/0	0/3/0	92/92/**99**
Confusion matrix of the method proposed in [2], IGT, and the proposed method										
Vehicle	BMP2	BTR70	T72	BTR60	2S1	BRDM2	D7	T62	ZILI131	ZSU234
BMP2	92/95/**100**	2/1/0	2/1/0	0/0/0	1/1/0	2/1/0	0/0/0	0/0/0	1/1/0	0/0/0
BTR70	3/2/0	93/94/**100**	0/0/0	0/0/0	0/00/	2/2/0	0/0/0	0/0/0	0/0/0	2/2/0
T72	2/2/0	1/1/0	96/96/**100**	1/1/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0
BTR60	2/1/0	0/0/0	2/1/2	93/97/**98**	0/0/0	0/0/0	3/1/0	0/0/0	0/0/0	0/0/0
2S1	3/3/0	4/4/0	1/1/1	0/0/0	87/89/**99**	2/0/0	0/0/0	2/1/0	0/0/0	1/2/0
BRDM2	5/2/1	3/1/0	2/4/0	0/0/0	0/0/0	85/90/**99**	5/2/0	0/0/0	0/0/0	0/1/0
D7	0/0/1	0/0/0	0/0/0	0/0/0	1/1/0	0/0/0	98/**99**/**99**	0/0/0	0/0/0	1/0/0
T62	1/1/0	0/0/0	0/0/0	0/0/0	0/0/1	0/0/0	0/0/0	93/95/**99**	3/3/0	3/1/0
ZILI131	2/2/0	2/2/0	0/1/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	94/95/**100**	2/0/0
ZSU234	1/1/0	0/0/0	0/0/1	1/1/0	0/0/0	0/0/0	0/0/0	0/0/0	2/2/0	96/96/**99**

The average P_{cc} of these 5 methods are sequentially 88.3%, 90.1%, 92.7%, 94.6%, and **99.3**%.

Table 9: Confusion matrix of EMACH, method proposed in [52], method proposed in [2], IGT, and the proposed method tested under EOC-1 (P_{cc} (%)).

Vehicle	2S1	BRDM2	T72	ZSU234
2S1	67/74/77/78/**99**	15/8/5/6/0	12/9/11/9/1	6/9/7/7/0
BRDM2	17/12/15/15/2	57/66/73/76/**97**	19/9/5/6/1	7/13/7/3/0
T72	7/17/11/10/0	9/6/9/9/0	66/73/75/78/**96**	18/4/5/3/4
ZSU234	10/7/4/5/1	7/5/6/5/0	2/3/2/2/1	81/85/88/88/**98**

The average P_{cc} of these 5 methods are sequentially 67.75%, 74.5%, 78.25%, 80.0%, and **97.5**%.

Table 10: Confusion matrix of EMACH, method proposed in [52], method proposed in [2], IGT, and the proposed method tested under EOC-2 (P_{cc} (%)).

Vehicle	BMP2	BRDM2	BTR70	T72
T72_S7	4/5/4/5/0	8/4/6/4/2	6/4/2/3/1	82/87/88/88/**97**
T72_A32	9/7/5/6/0	5/5/8/3/0	5/2/3/2/0	81/86/84/89/**99**
T72_A62	8/7/6/5/0	6/5/5/4/1	3/6/4/4/3	83/84/85/87/**97**
T72_A63	13/15/11/7/0	6/6/9/5/1	11/3/4/7/2	70/76/76/81/**97**
T72_A64	16/9/10/11/0	4/5/5/4/2	12/13/9/6/3	68/73/76/79/**94**

The average P_{cc} of these 5 methods are sequentially 76.8%, 81.2%, 81.8%, 84.8%, and **96.9**%.

51 training images, 60 training images, 71 training images, and 85 training images. Features are extracted with wavelet 768 and reduced to dimensionality of 55. The results are illustrated in Figure 17.

It is noted in Figure 17 that as the incremental step of poses increases, the achieved classification accuracy grows too. More specifically, a much higher P_{cc} of 90.3% is achieved by employing 51 training images with incremental Step 7 in pose, in comparison with a P_{cc} of 42.9% achieved by employing 85 training images with incremental Step 1 in pose. The principle behind this observation is that the training datasets formed with small pose variation steps can provide less target signal information and thus, their content is not sufficient enough to cover the different appearances of the targets captured under various poses. In contrast, a much more complete training dataset can be formed when the involved images are captured with larger pose variations. The experimental results in Figure 17 show that the best classification performance

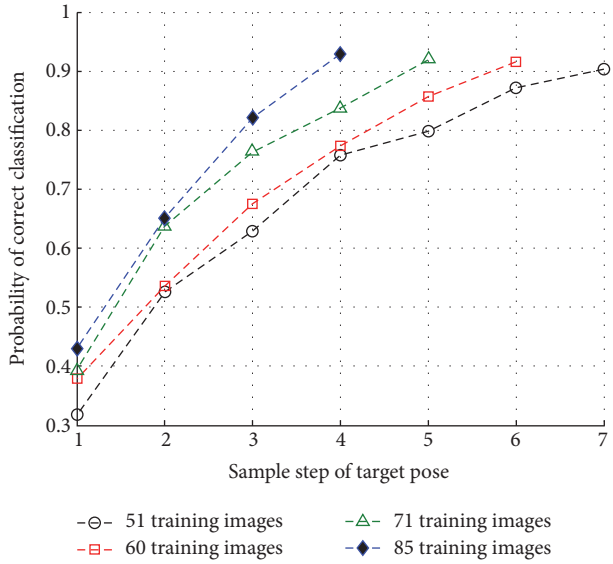


Figure 17: Performance comparison of training data selected with different incremental steps of target poses.

90.3% is achieved when training with 51 images captured using incremental Step 7 and 93.0% is achieved when training with 85 images captured using incremental Step 4. Moreover, it is quite straightforward to find that better classification performance is always achieved when training with relatively larger number of training images. It is worth pointing out that only a small number of images are involved in the training stage rather than several hundreds of them as used in the previous tests. This is a promising result which implies that a good classification result can be achieved even with much less number of training images, as long as they are captured with appropriate incremental step.

6. Conclusion

In this paper, we presented a systematic scheme for the SAR ATR task. The proposed scheme involves three main stages: preprocessing, feature extraction and processing, and classifier construction. The effectiveness of involving several preprocessing approaches (e.g., the image energy normalization and the pose rectification processes) is analysed and empirically verified. The results suggest that the involvement of these preprocessing steps is beneficial for improving the classification accuracy. Moreover, we proposed to expand the feature set to provide more information for discrimination and then eliminate the redundancy and dimensionality of the extended feature set to form a more compact and efficient feature set. Finally, the discriminative trees are learned as the base classifiers and combined to construct a strong classifier by using the Real-AdaBoost algorithm. The proposed method is evaluated with the MSTAR dataset under various operating conditions. Experimental results demonstrate that the proposed method outperforms traditional methods, for example, EMACH, SVM, NN, and IGT. The advantages of the proposed method give credit to the reduction of variations in target images, the improvement of feature efficiency, the elimination of redundancy in feature sets, and the excellent generalization capability of the combined strong classifier. Moreover, we have tested the classification performance of the classifiers trained with different combinations of target poses. Experimental results show that a classifier trained with training images covering large variations of target poses can produce good classification result even with limited number of training images.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the EU H2020 TERPSICHORE project "Transforming Intangible Folkloric Performing Arts into Tangible Choreographic Digital Objects" under the Grant Agreement 691218.

References

[1] S. Ochilov and D. A. Clausi, "Operational SAR sea-ice image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 11, pp. 4397–4408, 2012.

[2] Y. Sun, Z. P. Liu, S. Todorovic, and J. N. Li, "Adaptive boosting for SAR automatic target recognition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 1, pp. 112–125, 2007.

[3] J.-I. Park and K.-T. Kim, "Modified polar mapping classifier for SAR automatic target recognition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 2, pp. 1092–1107, 2014.

[4] Ö. Aytekin, M. Koc, and I. Ulusoy, "Local primitive pattern for the classification of SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 4, pp. 2431–2441, 2013.

[5] N. M. Sandirasegaram, "Spot SAR ATR using wavelet features and neural network classifier," Tech. Rep., Report. DTIC Document, 2005.

[6] M. Amoon and G.-A. Rezai-rad, "Automatic target recognition of synthetic aperture radar (SAR) images based on optimal selection of Zernike moments features," *IET Computer Vision*, vol. 8, no. 2, pp. 77–85, 2014.

[7] C. Clemente, L. Pallotta, I. Proudler, A. de Maio, J. J. Soraghan, and A. Farina, "Pseudo-Zernike-based multi-pass automatic target recognition from multi-channel synthetic aperture radar," *IET Radar, Sonar & Navigation*, vol. 9, no. 4, pp. 457–466, 2015.

[8] C. Zhu, H. Zhou, R. Wang, and J. Guo, "A novel hierarchical method of ship detection from spaceborne optical image based on shape and texture features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 9, pp. 3446–3456, 2010.

[9] Z. Jianxiong, S. Zhiguang, C. Xiao, and F. Qiang, "Automatic target recognition of SAR images based on global scattering center model," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3713–3729, 2011.

[10] E. Giusti, M. Martorella, and A. Capria, "Polarimetrically-persistent-scatterer-based automatic target recognition," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 11, pp. 4588–4599, 2011.

[11] J.-I. Park, S.-H. Park, and K.-T. Kim, "New discrimination features for SAR automatic target recognition," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 3, pp. 476–480, 2013.

[12] G. Dong, N. Wang, and G. Kuang, "Sparse Representation of Monogenic Signal: With Application to Target Recognition in SAR Images," *IEEE Signal Processing Letters*, vol. 21, no. 8, pp. 952–956, 2014.

[13] H. Chen, H. Chang, and T. Liu, "Local discriminant embedding and its variants," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 2, pp. 846–853, June 2005.

[14] F. Dornaika and A. Bosaghzadeh, "Exponential local discriminant embedding and its application to face recognition," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 921–934, 2013.

[15] X. Fang, Y. Xu, X. Li, Z. Fan, H. Liu, and Y. Chen, "Locality and similarity preserving embedding for feature selection," *Neurocomputing*, vol. 128, pp. 304–315, 2014.

[16] X. Liu, Y. L. Huang, J. F. Pei, and J. Y. Yang, "Sample discriminant analysis for SAR ATR," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 12, pp. 2120–2124, 2014.

[17] Y. Huang, J. Peia, J. Yanga, B. Wang, and X. Liu, "Neighborhood geometric center scaling embedding for SAR ATR," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 180–192, 2014.

[18] X. Zhao, Y. Jiang, T. Stathaki, and H. Zhang, "Gait recognition method for arbitrary straight walking paths using appearance conversion machine," *Neurocomputing*, vol. 173, pp. 530–540, 2016.

[19] X. Zhao, Y. Jiang, and W.-Q. Wang, "Efficient Clutter Suppression in SAR Images with Shedding Irrelevant Patterns," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1828–1832, 2015.

[20] X. Xing, K. Ji, H. Zou, and J. Sun, "Sparse representation based sar vehicle recognition along with aspect angle," *The Scientific World Journal*, vol. 2014, Article ID 834140, pp. 174-175, 2014.

[21] G. Akbarizadeh, "A new statistical-based kurtosis wavelet energy feature for texture recognition of SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 11, pp. 4358–4368, 2012.

[22] L. Bruzzone, M. Marconcini, U. Wegmüller, and A. Wiesmann, "An advanced system for the automatic classification of multitemporal SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 6, pp. 1321–1334, 2004.

[23] Y. Wang, P. Han, X. Lu, R. Wu, and J. Huang, "The performance comparison of Adaboost and SVM applied to SAR ATR," in *Proceedings of the CIE International Conference on Radar (ICR '06)*, pp. 1–4, October 2006.

[24] U. Srinivas, V. Monga, and R. G. Raj, "SAR automatic target recognition using discriminative graphical models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 591–606, 2014.

[25] S. G. Mallat, "Theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[26] C. Qiu, H. Ren, H. Zou, and S. Zhou, "Performance comparison of target classification in SAR images based on PCA and 2D-PCA features," in *Proceedings of the 2009 Asia-Pacific Conference on Synthetic Aperture Radar, APSAR 2009*, pp. 868–871, China, October 2009.

[27] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[28] H. Zhang, H. Lin, and Y. Li, "Impacts of feature normalization on optical and SAR data fusion for land use/land cover classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 5, pp. 1061–1065, 2015.

[29] Q. Zhao, D. Xu, and J. C. Principe, "Pose Estimation for SAR Automatic Target Recognition," in *Proceedings of Image Understanding Workshop*, pp. 827–832, 1999.

[30] J. C. Principe, D. Xu, and J. W. Fisher III, "Pose estimation in SAR using an information theoretic criterion," in *Proceedings of the Algorithms for Synthetic Aperture Radar Imagery V*, pp. 218–229, usa, April 1998.

[31] L. M. Kaplan and R. Murenzi, "Pose estimation of SAR imagery using the two dimensional continuous wavelet transform," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2269–2280, 2003.

[32] L. Voicu, R. Patton, and H. Myler, "Multi-criterion vehicle pose estimation for SAR-ATR," in *In Proceedings of SPIE - The International Society for Optical Engineering*, vol. 372110.

[33] S. Helgason, "The Radon transform," *Progress in Mathematics*, vol. 13, no. 89, pp. 81–133, 2009.

[34] G. Toussaint, "Solving geometric problems with the rotating calipers," in *Proceedings of the MELECON '83, Mediterranean Electrotechnical Conference*.

[35] J. Liu, S. Chen, X. Tan, and D. Zhang, "Comments on 'Efficient and robust feature extraction by maximum margin criterion'," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 18, no. 6, pp. 1862–1864, 2007.

[36] S. Sanghavi, V. Tan, and A. Willsky, "Learning graphical models for hypothesis testing," in *Proceedings of the 2007 IEEE/SP 14th WorkShoP on Statistical Signal Processing, SSP 2007*, pp. 69–73, USA, August 2007.

[37] V. Y. Tan, S. Sanghavi, I. Fisher, and A. S. Willsky, "Learning graphical models for hypothesis testing and classification," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, pp. 5481–5495, 2010.

[38] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience Publication, New York, USA, 2nd edition, 2012.

[39] S. Lauritzen, *Graphical Models*, Oxford University Press, Oxford,England, 1996.

[40] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[41] J. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[42] S. Kullback, *Information Theory and Statistics*, Wiley-Interscience Publication, New York, USA, 1959.

[43] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[44] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[45] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 4, pp. 51-52.

[46] A. Vezhnevets and V. Vezhnevets, "Modest AdaBoost-teaching AdaBoost to generalize better," in *Graphicon*, pp. 12987–12997, 2005.

[47] S. Wu and H. Nagahashi, "A new method for solving overfitting problem of gentle AdaBoost," in *Proceedings of the 5th International Conference on Graphic and Image Processing, ICGIP 2013*, China, October 2013.

[48] J. Milgram, M. Cheriet, and R. Sabourin, "Speeding Up the Decision Making of Support Vector Classifiers," in *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pp. 57–62, Tokyo, Japan.

[49] J. C. Mossing and T. D. Ross, "An evaluation of SAR ATR algorithm performance sensitivity to MSTAR extended operating conditions," in *Proceedings of the Algorithms for Synthetic Aperture Radar Imagery V*, pp. 554–565, USA, April 1998.

[50] T. Ross, S. Worrell, V. Velten, J. Mossing, and M. Bryant, "Standard SAR ATR evaluation experiments using the MSTAR public release data set," in *Proceedings of the Algorithms for Synthetic Aperture Radar Imagery V*, pp. 566–573, USA, April 1998.

[51] T. D. Ross and C. J. Mossing, "MSTAR evaluation methodology," in *AeroSense'99*, pp. 705–713, International Society for Optics and Photonics, 1999.

[52] Q. Zhao and J. C. Principe, "Support vector machines for SAR automatic target recognition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 2, pp. 643–654, 2001.

[53] R. Singh and B. V. K. Vijaya Kumar, "Performance of the extended maximum average correlation height (EMACH) filter and the polynomial distance classifier correlation filter (PDCCF) for multi-class SAR detection and classification," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4727, pp. 265–276, 2002.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 8348671, 8 pages
https://doi.org/10.1155/2017/8348671

Research Article

High Performance Implementation of 3D Convolutional Neural Networks on a GPU

Qiang Lan,[1,2] Zelong Wang,[1,2] Mei Wen,[1,2] Chunyuan Zhang,[1,2] and Yijie Wang[1,2]

[1]*College of Computer, National University of Defense Technology, Changsha 410073, China*
[2]*National Key Laboratory of Parallel and Distributed Processing, Changsha 410073, China*

Correspondence should be addressed to Qiang Lan; lanqiang_nudt@163.com

Received 16 April 2017; Revised 19 July 2017; Accepted 6 August 2017; Published 8 November 2017

Academic Editor: Athanasios Voulodimos

Convolutional neural networks have proven to be highly successful in applications such as image classification, object tracking, and many other tasks based on 2D inputs. Recently, researchers have started to apply convolutional neural networks to video classification, which constitutes a 3D input and requires far larger amounts of memory and much more computation. FFT based methods can reduce the amount of computation, but this generally comes at the cost of an increased memory requirement. On the other hand, the Winograd Minimal Filtering Algorithm (WMFA) can reduce the number of operations required and thus can speed up the computation, without increasing the required memory. This strategy was shown to be successful for 2D neural networks. We implement the algorithm for 3D convolutional neural networks and apply it to a popular 3D convolutional neural network which is used to classify videos and compare it to cuDNN. For our highly optimized implementation of the algorithm, we observe a twofold speedup for most of the 3D convolution layers of our test network compared to the cuDNN version.

1. Introduction

Convolutional neural networks have proven advantages over traditional machine learning methods on applications such as image classification [1–4], tracking [5, 6], detection [7–11]. However, the primary downside of convolutional neural networks is the increased computational cost. This becomes especially challenging for 3D convolution where handling even the smallest instances requires substantial resources.

3D convolutional neural networks have recently come to the attention of the scientific community. In [12], a database for 3D object recognition named ObjectNet3D is presented. The database focuses on the problem of recognizing the 3D pose and the shape of objects from 2D images. Another repository of 3D CAD models of objects is ShapeNet [13]. In [14], the authors propose VoxNet, a 3D convolutional neural network, to solve the robust object recognition task with the help of 3D information, while the authors of [15] propose a 3D convolutional neural networks for human-action recognition.

In the light of these successful applications, it is worthwhile to explore new ways of speeding up the 3D convolution operation. In this paper we do so by deriving the 3D convolution forms of the minimal filtering algorithms invented by Toom and Cook [16] and generalized by Winograd [17]. Our experiments show this algorithm to be very efficient in accelerating 3D convolutional neural network in video classification applications.

2. Related Work

Many approaches aim to directly reduce the computational cost within CNN. In [18], the authors analyse the algebraic properties of CNNs and propose an algorithmic improvement to reduce the computational workload. They achieve a 47% reduction in computation without affecting the accuracy. In [19], convolution operations are replaced with pointwise products in the Fourier domain, which can reduce the amount of computation significantly. Reference [20] evaluates two fast Fourier transform (FFT) convolution implementations, one based on Nvidia cuFFT [21] and the other based on Facebook's FFT implementation. The FFT method can achieve an obvious speeding up of performance when the filter size is large, and the disadvantage of the FFT

method is that it consumes much more memory than the standard method.

In [22], the authors use WMFA (Winograd Minimal Filter Algorithm) [17] to implement the convolution operation. In theory, fewer multiplications are needed in the WMFA, while not much extra memory is needed. WMFA is easy to parallelize; Lavin and Gray [22] implemented the algorithm on GPU, and they achieved better performance than the fastest cuDNN library. In [23], the authors show a novel architecture implemented in OpenCL on an FPGA platform; the algorithm they use to do the convolution is WMFA, which significantly boosts the performance of the FPGA. However, both works implemented 2D convolutional neural networks.

In this paper, we make four main contributions. Firstly, we derive the 3D forms of WMFA and design detailed algorithm to implement 3D convolution operation based on 3D WMFA. Secondly, we analyse the arithmetic complexity of 3D WMFA and prove 3D WMFA method can reduce computation in theory. Thirdly, we implement 3D WMFA for GPU platform and propose several optimization techniques to improve the performance of 3D WMFA. Finally, we evaluate the performance of 3D convolutional neural networks based on several implementations and prove the advantage of our proposed 3D WMFA method.

3. Fast 3D Convolution Algorithm

3.1. Preliminary: 3D Convolutional Neural Networks. For the 2D convolution, kernels have fixed width and height, and they are slid along the width and height of the input feature maps. For the 3D convolution, both feature maps and kernels have depth dimension, and the convolution also needs to slide along the depth direction. We can compute the output of a 3D convolutional layer using the following formula:

$$Y_{i,k,x,y,z} = \sum_{c=0}^{C-1} \sum_{t=0}^{T-1} \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} I_{i,c,x+t,y+r,z+s} F_{k,t,r,s,c}, \quad (1)$$

where $Y_{i,k,x,y,z}$ represents the result of a convolution operation at the kth channel feature and $I_{i,c,x+t,y+r,z+s}$ is one of the input features, while $F_{k,t,r,s,c}$ is one of the filters. Equation (1) represents a direct convolution method, which requires intensive computability. The detailed arithmetic complexity of this method is shown in Section 3.3.

3.2. 3D WMFA. We introduce a new, fast algorithm to compute a 3D convolutional layer. The algorithm is based on WMFA. In order to introduce the 3D WMFA, firstly, we will give a simple introduction to the 1D WMFA. WMFA computes output with a tile size of m each time; we use $F(m, r)$ to represent the output tile and r is the filter size. According to the definition of convolution, $2 \times 3 = 6$ multiplications are required to compute $F(2, 3)$, but we can reduce the number of multiplications to do the convolution if we use the following WMFA:

$$F(2,3) = \begin{bmatrix} i_0 & i_1 & i_2 \\ i_1 & i_2 & i_3 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}, \quad (2)$$

where

$$\begin{aligned} m_1 &= (i_0 - i_2) f_0, \\ m_2 &= (i_1 + i_2) \frac{f_0 + f_1 + f_2}{2}, \\ m_4 &= (i_1 - i_3) f_2, \\ m_3 &= (i_2 - i_1) \frac{f_0 - f_1 + f_2}{2}. \end{aligned} \quad (3)$$

The number of multiplications needed is $\mu(F(2,3)) = 2 + 3 - 1 = 4$; however, four additions are needed to transform the input image, three additions to transform the filter, and four additions to transform the result of the dot product. We can use a matrix form to represent the computation:

$$Y = A^T \left[\left(B^T i \right) \odot (Gf) \right]. \quad (4)$$

We call the A^T, G, and B^T transform matrices, and the values of the transforming matrices are

$$\begin{aligned} B^T &= \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \\ G &= \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}, \\ A^T &= \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}. \end{aligned} \quad (5)$$

In (4), $i = \begin{bmatrix} i_0 & i_1 & i_2 & i_3 \end{bmatrix}^T$ and $f = \begin{bmatrix} f_0 & f_1 & f_2 \end{bmatrix}^T$ represent the input tile and filter tile, respectively. As described in [22], the format of the 2D WMFA is as follows:

$$Y = A^T \left[\left[GfG^T \right] \odot \left[B^T i B \right] \right] A, \quad (6)$$

where f is the filter with size $r \times r$ and i is the image with size $(m + r - 1) \times (m + r - 1)$. To compute $F(2 \times 2, 3 \times 3)$, we need $4 \times 4 = 16$ multiplications; however, $4 \times 9 = 36$ multiplications are needed according to the convolution definition. Therefore, 2D WMFA can reduce the number of multiplications by a factor of $36/16 = 2.25$ at the cost of increasing 32 additions in the data transformation stage, 28 floating point instructions at the filter transformation stage, and 24 additions at the inverse transformation stage. For a convolutional layer, the number of input channels and number of output channels are large, which means the input channels need to convolve different filters, so the transformed input tile can be reused as many times as the number of output channels. Each filter needs to be slid in x

Input: $I_0[in\_size][in\_size][in\_size]$
Temp array: $I_1[in\_size][out\_size][in\_size]$, $I_2[in\_size][out\_size][out\_size]$
Output: $I_3[out\_size][out\_size][out\_size]$
for $i = 0$ to in_size **do**
  for $j = 0$ to in_size **do**
    $I_1[i][0 : out\_size][j] = T_m I_0[i][0 : in\_size][j]$
  end for
end for
for $i = 0$ to in_size **do**
  for $j = 0$ to out_size **do**
    $I_2[i][j][0 : out\_size] = T_m I_1[i][j][0 : in\_size]$
  end for
end for
for $i = 0$ to out_size **do**
  for $j = 0$ to out_size **do**
    $I_3[0 : out\_size][i][j] = T_m I_2[0 : in\_size][i][j]$
  end for
end for

Algorithm 1: 3D winograd transformation.

and y direction of input channel during convolution, so each transformed filter is reused as many times as the number of subtiles of input channel. And since the output tile is reduced along the input channels, the inverse transformation is done after reduction; then the number of inverse transformation is determined by the number of output channels. Therefore, the cost of data transformation stage, filter transformation stage, and the inverse transformation stage keep low in real convolutional layer implementation.

We can also apply the 3D WMFA to 3D convolution. To compute $F(2 \times 2 \times 2, 3 \times 3 \times 3)$, we apply the 3D Winograd transformation to the input tile and filter tile and apply 3D Winograd inverse transformation to the dot product of the transformed input image tile and the transformed filter tile. Algorithm 1 is a general form of the 3D Winograd transformation. In the algorithm, T_m is the transformation matrix; the transformation matrix can be G applied to transform the filter tile or B^T applied to transform the input image tile. The dot product of the transformed input image tile and transformed filter tile will be accumulated along the C channels, which can be converted to a matrix multiplication similar to the description in [22]

$$\begin{aligned} Y_{i,x,y,z,k} &= \sum_{c=0}^{C-1} I_{i,c,x,y,z} * F_{k,c} = \sum_{c=0}^{C-1} A^T \left[U_{k,c} \odot V_{c,i,x,y,z}\right] A \\ &= A^T \left[\sum_{c=0}^{C-1} U_{k,c} \odot V_{c,i,x,y,z}\right] A. \end{aligned} \tag{7}$$

Consider the sum

$$M_{k,i,x,y,z} = \sum_{c=0}^{C-1} U_{k,c} \odot V_{c,i,x,y,z}. \tag{8}$$

The previous equation can be divided into several submatrix multiplications; assume the output tile size is (ε, η, ν), using new coordinates $(i, \tilde{x}, \tilde{y}, \tilde{z})$ to replace (i, x, y, z), yielding

$$M^{\varepsilon,\eta,\nu}_{k,i,\tilde{x},\tilde{y},\tilde{z}} = \sum_{c=0}^{C-1} U^{(\varepsilon,\eta,\nu)}_{k,c} V^{(\varepsilon,\eta,\nu)}_{c,i,\tilde{x},\tilde{y},\tilde{z}}. \tag{9}$$

This equation represents the matrix multiplication, and it can be simplified as follows:

$$M^{(\varepsilon,\eta,\nu)} = U^{(\varepsilon,\eta,\nu)} V^{(\varepsilon,\eta,\nu)}. \tag{10}$$

Algorithm 2 gives the overview of the 3D WMFA. The algorithm mainly consists of four stages, which are Winograd transformation of the input feature tile; Winograd transformation of the filter tile; the matrix multiplication, which is converted from the dot product of the transformed input tile and the transformed filter tile; and the inverse Winograd transformation of the result of the matrix multiplication.

3.3. Arithmetic Complexity Analysis. For input feature maps with size $N \times C \times D \times H \times W$, filters with size $K \times C \times k \times k \times k$, and the output features with size $N \times K \times M \times P \times Q$, the total number of float operations in the multiplication stage can be represented as follows:

$$L_1 = 2N \left\lceil \frac{M}{m} \right\rceil \left\lceil \frac{P}{m} \right\rceil \left\lceil \frac{Q}{m} \right\rceil CK (m + r - 1)^3, \tag{11}$$

where r is the filter size and m is the size of the output subtile. However, if we use the direct convolution method, which is computed according to the definition of convolution, the total number of float operations is computed as follows:

$$L_2 = 2NMPQCKr^3. \tag{12}$$

Dividing L_1 by L_2 yields

$$\frac{L_2}{L_1} = \frac{m^3 * r^3}{(m + r - 1)^3}. \tag{13}$$

$P = N\lceil M/m\rceil\lceil P/m\rceil\lceil Q/m\rceil$ is the number of image tiles.
$\alpha = m + r - 1$ is the input tile size.
Neighbouring tiles overlap by $r - 1$.
$d_{c,b} \in R^{\alpha\times\alpha\times\alpha}$ is input tile b in channel c.
$g_{k,c} \in R^{r\times r\times r}$ is filter k in channel c.
$Y_{k,b} \in R^{m\times m\times m}$ is output tile b in filter k.
for $k = 0$ to K **do**
  for $c = 0$ to C **do**
    $u = T_k(g_{k,c}) \in R^{\alpha\times\alpha\times\alpha}$
    Scatter u to matrices U: $U_{k,c}^{(i,j,k)} = u_{i,j,k}$
  end for
end for
for $b = 0$ to P **do**
  for $c = 0$ to C **do**
    $v = T_d(d_{c,b}) \in R^{\alpha\times\alpha\times\alpha}$
    Scatter v to matrices V: $V_{c,b}^{(i,j,k)} = v_{i,j,k}$
  end for
end for
for $i = 0$ to α **do**
  for $j = 0$ to α **do**
    for $k = 0$ to α **do**
      $M^{(i,j,k)} = U^{(i,j,k)}V^{(i,j,k)}$
    end for
  end for
end for
for $k = 0$ to K **do**
  for $b = 0$ to P **do**
    Gather m from matrices M: $m_{i,j,k} = M_{k,b}$
    $Y_{k,b} = T_m(m)$
  end for
end for

Algorithm 2: 3D Convolutional layer implemented with WMFA $F(m \times m \times m, r \times r \times r)$.

Assuming $m = 2$ and $r = 3$, there is an arithmetic complexity reduction of $(2 * 2 * 2 * 3 * 3 * 3)/(4 * 4 * 4) = 216/64 = 3.375$. However, there are some extra computations in Winograd transformation stage, so we cannot achieve so much complexity reduction in reality. The detailed performance improvements are shown in Section 5.2.

4. Implementation and Optimizations

4.1. Implementation. We have three implementation versions for the 3D WMFA on a GPU. In our implementations, cuBLAS is called to do the multiplication. Furthermore, we manually implement six kernels according to Algorithm 2. Figure 1 shows the flow of our baseline implementation. The *imageTransform* kernel transforms all the image subtiles, the *filterTransform* kernel transforms all the filter tiles, and *ouputTransform* kernel inversely transforms the result of the multiplication. For the baseline implementation, we also have two additional kernels to reorganize the transformed image data and transformed filter data and one kernel to reorganize the result of the multiplication before the results go to the *outputTransform* kernel.

Winograd transformation algorithm is suitable for parallelization on GPU. Taking the *imageTransform* kernel as an example, the input to the *imageTransform* kernel is the input feature map. We assume the input feature maps have a size of $N \times C \times D \times H \times W$, where N is the batch size, C is the number of input channels, and $D \times H \times W$ is the size of a single channel. As is described in Algorithm 2, the number of image tiles for each channel is P, so the total number of image tiles is $P \times C$; all those image tiles can be transformed independently. For the baseline of the GPU implementation, the image data is stored in *NCDHW* order, and we set the number of threads in one block to be 32, each thread is responsible for processing Winograd transformation of one input subtile, and the number of blocks in the grid is set to $(\lceil N/32\rceil, \lceil M/m\rceil\lceil P/m\rceil\lceil Q/m\rceil, C)$. We can make full use of the large-scale parallel processing units of a GPU when the number of blocks is large.

For the *filterTransform* kernel, there are $K \times C$ filter tiles; we still set the number of threads of one block to be 32 and the number of blocks to $(\lceil K/32\rceil, C, 1)$. Before we call cuBLAS to implement the matrix multiplication, we need to reorganize the transformed filter and transformed image data. For transformed filter, there are $K \times C$ tiles in total, and each tile has size of $\alpha \times \alpha \times \alpha$, each time we gather one value from one tile to generate a submatrix of size $K \times C$. Figure 2 shows how the transformed data are reorganized in new layout on GPU; they are implemented by the kernels *reshapeTransformedImage* and *reshapeTransformedFilter* in

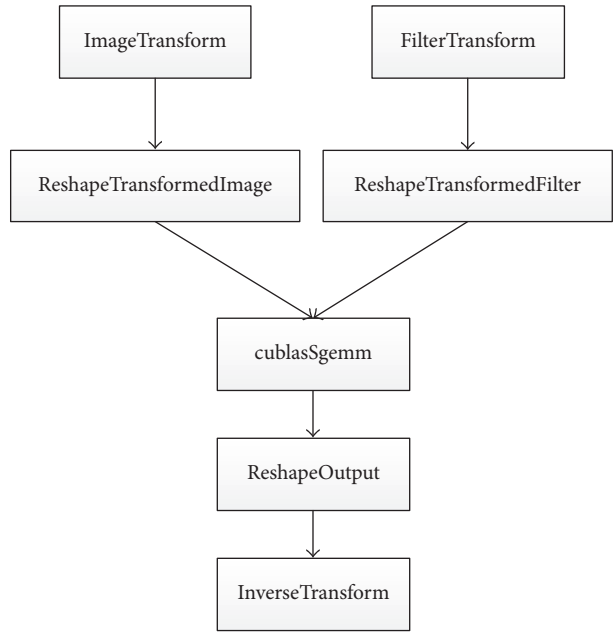


Figure 1: The computing flow of 3D WMFA.

our baseline implementation. They gather correlated and transformed filters and transformed image data to form two submatrices, and *SGEMM* from the cuBLAS library is called to do the multiplication. The result of the multiplication also needs to be reshaped before the inverse transformation.

4.2. Optimizations. We make two optimizations to achieve higher performance. The first optimization is to align memory access, which can make memory access more efficient and increase the cache hit rate. The second optimization is to combine the transformation kernel with the reshape kernel to reduce global memory access.

The storing order of data and how data is accessed by a thread can significantly affect the performance. For the baseline implementation, the image data is stored in *NCDHW* order, and threads in the same block access data along the N-dimension, which means data accessed by threads keeps long distances. However, the size of the cache on a GPU is limited; therefore, if the distance between the data items accessed by threads is larger than the size of the cache line, then each thread needs to access global memory separately, causing lots of memory accesses. In our first optimization version, we change the storage order of image data to *CDHWN*. Since the image data is stored starting from the N-dimension, data accessed by all threads in the same block is continually stored, and all data items loaded from the global memory are useful, which means the bandwidth is fully used. The same optimization method is applied to the filter transformation and inverse transformation kernel.

Based on the first optimization, we apply our second optimization to improve performance further. The second optimization is to reduce the number of global memory accesses. For the baseline implementation, after the filter transformation or image transformation kernel is executed, the transformed filter or transformed image data need to be stored in a new layout using *reshapeTransformedFilter* and *reshapeTransformedImage* kernel and using the *ReshapeOutput* kernel before the inverse transformation, while on our second optimized version, we move the work of the reshape kernel to the transformation kernel, so in the optimized transform kernel, after the inputs are transformed, the result will be stored directly back in the expected layout. In the optimized inverse transformation, before inverse transformation, the required data is gathered directly from the global memory.

Table 1: Properties of the GeForce GTX 1080.

Parameters	Values
CUDA capability major/minor version number	6.1
Total amount of global memory	8 GB
CUDA cores	2560
L2 cache Size	2 MB
Warp size	32
Total number of registers available per block	64 KB

5. Experiments

5.1. Experimental Setup. All experiments are evaluated on a GeForce GTX 1080 GPU, which has a total amount of 8 GBytes global memory and has 20 multiprocessors. Detailed parameters of the GTX 1080 are shown in Table 1.

5.2. Performance Evaluation. We apply our 3D WMFA to a widely used 3D neural network called v3d [9], which is used to classify videos. The 3D neural network has five convolutional layers; Table 2 shows the information about these 3D convolutional layers.

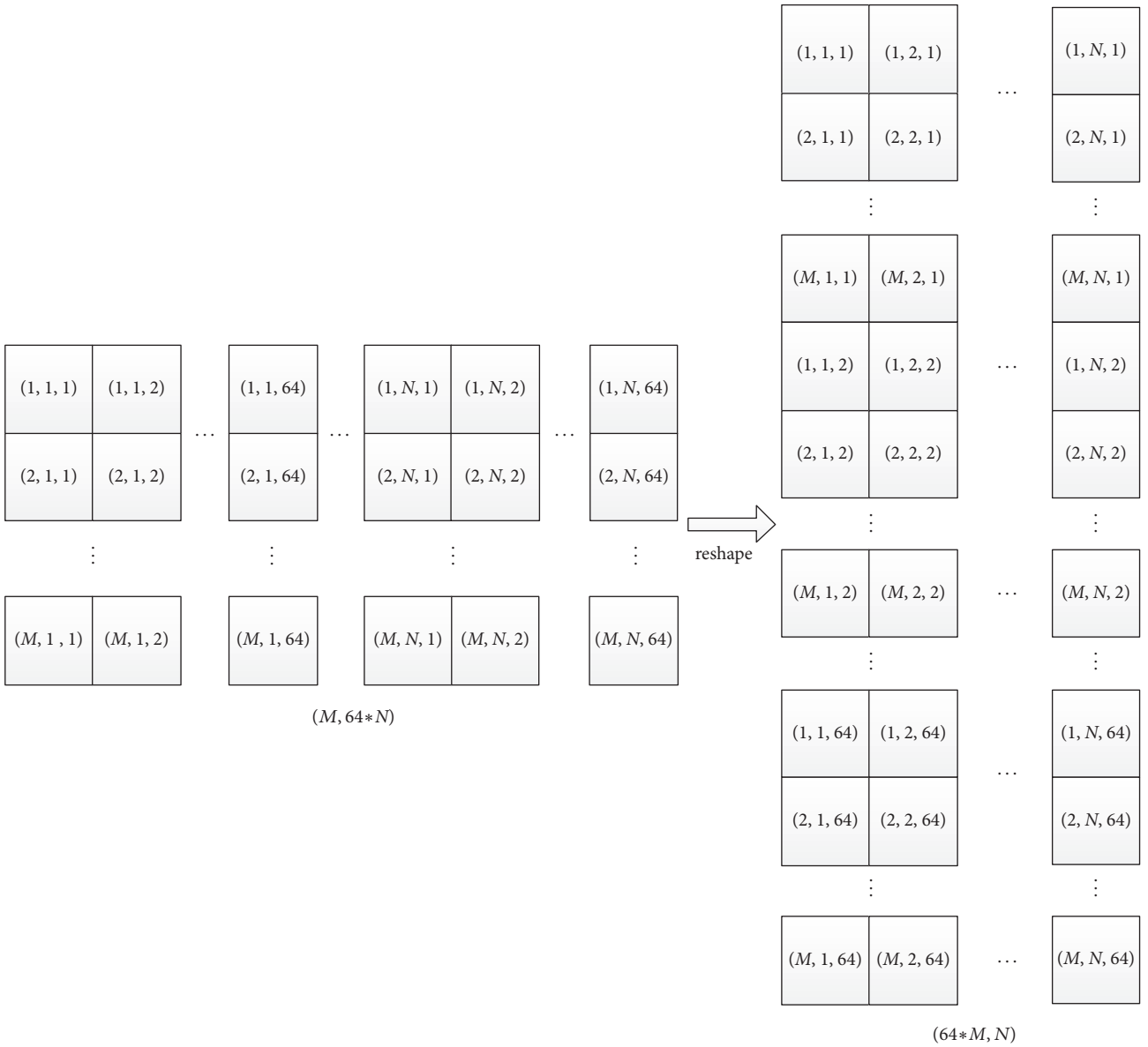


Figure 2: For an input matrix, its size is $(M, N * \alpha * \alpha * \alpha)$; α is the tile size, here equal to 4. After the reshape kernel is applied, lots of small submatrices with new layouts are generated.

Table 2: Convolution layers of a 3D network; the filter size in all layers is $3 \times 3 \times 3$, and the GFLOPS columns calculate the number of flops operations in each convolutional layer. Assume the batch size is 32.

Layer	$C \times D \times H \times W \times N$	K	GFLOPS
conv1	$3 \times 16 \times 112 \times 112 \times 32$	32	16.65
conv2	$32 \times 16 \times 56 \times 56 \times 32$	64	88.8
conv3	$64 \times 8 \times 28 \times 28 \times 32$	256	88.8
conv4	$256 \times 4 \times 14 \times 14 \times 32$	256	44.4
conv5	$256 \times 2 \times 7 \times 7 \times 32$	256	5.55

Firstly, we evaluate the performance of our three implementations on the 3D convolutional layers, except for the first convolutional layer, which has only three input channels and is not yet supported in the algorithm. Figure 3 shows the increase in speed we achieved after we used two optimizations. For the first optimization, we observe a 3 to 4 times speeding up for all these test convolution layers compared to the baseline implementation. However, for the second optimization, the maximum speeding up can be close to 42 for the third convolution; even the minimum speeding up is about 13 for the last convolution layer. The first optimization makes memory access more efficient, and the second optimization reduces lots of unnecessary global memory accesses. Since the latency of global memory access on a GPU is large, we achieve a good performance improvement in the second optimization.

We explore the detailed performance for one specific convolution layer to see how these two optimizations improve

Table 3: Performance of cuDNN SGEMM versus that of the 3D WMFA on 3D convolution layers. Performance is measured in effective TFLOPS.

Layer	$C \times D \times H \times W \times N$	K	TFLOPS		Speedup
			cuDNN SGEMM	3D WMFA	
conv2	$32 \times 16 \times 56 \times 56 \times 32$	64	1.21	1.28	1.05
conv3	$64 \times 8 \times 28 \times 28 \times 32$	256	2.38	3.31	1.39
conv4	$256 \times 4 \times 14 \times 14 \times 32$	256	2.4	4.72	1.96
conv5	$256 \times 2 \times 7 \times 7 \times 32$	256	1.46	2.1	1.44

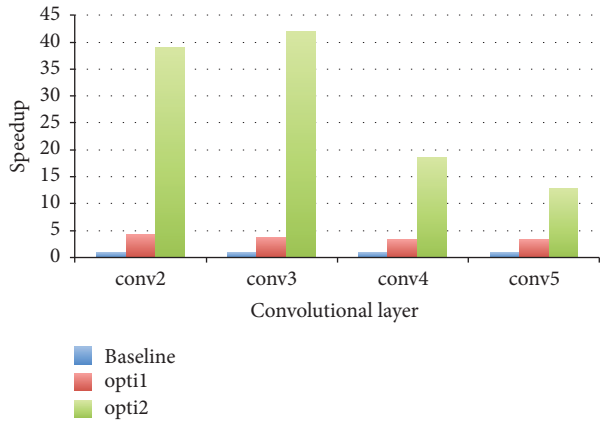


Figure 3: Speedup with different optimizations on 3D convolution layers.

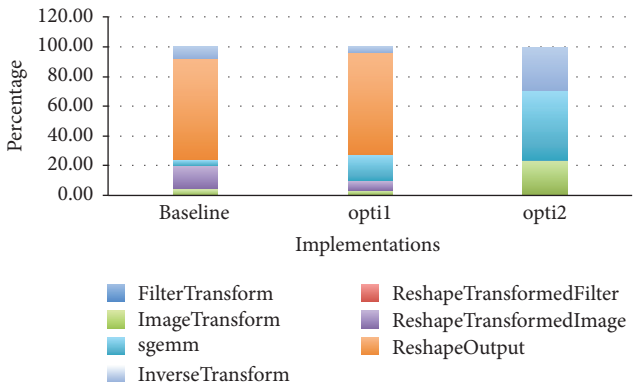


Figure 4: Time percentage distribution of each kernel in each implementation version for a specific convolution layer.

each kernel separately. We profile the time percentage of each kernel in each implementation version for conv3, which takes most of the computation time among all these convolution layers. Figure 4 shows the profiling results; the kernel *ReshapeOutput* in both baseline and the first optimization takes up the most time, since the kernel contains lots of global memory accesses. However, in the second optimization version, there are no *reshape* kernels and the kernel *sgemm* takes most of the execution time. In all three implementations, the kernel *filterTransform* takes only about 0.1% of the total time.

Finally, we compare our best optimized implementation with the cuDNN library. The cuDNN library is the fastest

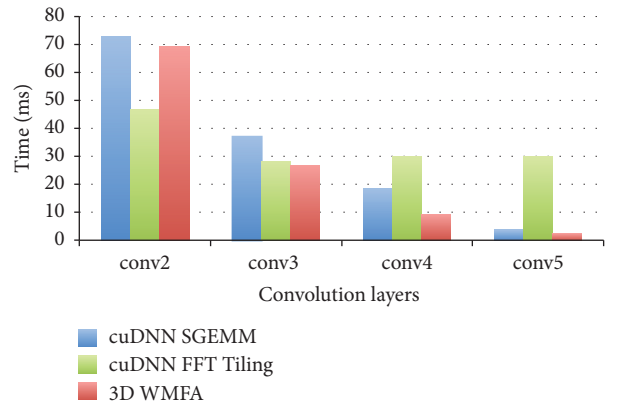


Figure 5: Execution time of different methods on 3D convolution layers.

deep-learning library. There are two algorithms available to implement a 3D convolution layer: one converts the convolution to a matrix multiplication and the other exploits the FFT tiling method to implement the convolution. We use *cuDNN SGEMM* and *cuDNN FFT Tiling* to represent the two methods called in the cuDNN library, and we use *3D WMFA* to represent our algorithm. Figure 5 shows the execution time of these three methods on four convolution layers. The *3D WMFA* method is about 30% slower than the *cuDNN FFT Tiling* method on the conv2 layer; however, it is a bit faster than *cuDNN SGEMM* method. The *cuDNN FFT Tiling* method achieves a fast speed at the cost of consuming a large amount of memory. Since parameters C and K are not large in the conv2 layer, this makes the matrix multiplication in *3D WMFA* on small scale, which affects the performance. However, with parameters C and K increased on layers conv3, conv4, and conv5, the *3D WMFA* method achieves better performance than the other two methods. We added the execution time of each layer for each method, the total time of all layers is 132.6 ms for cuDNN SGEMM method, 135.4 ms for cuDNN FFT Tiling method, and 108.2 ms for 3D WMFA which is better than the other two methods.

We can also calculate the performance of these two methods in *TFLOPS*. Table 3 shows the effective *TFLOPS* of *cuDNN SGEMM* and *3D WMFA* method. We achieve a maximum speedup of 1.96 compared to *cuDNN SGEMM*.

6. Conclusions

A 3D convolution layer requires a high computational cost and consumes lots of memory. We designed a 3D WMFA

to implement 3D convolution operation. Compared to traditional convolution methods, such as SGEMM or FFT, the 3D WMFA can reduce computation, in theory. When we implemented the algorithm on a GPU, we observed the expected performance of the algorithm in the experiments. For some 3D convolution layers, we even achieve close to 2 times speedup compared to cuDNN library.

However, the computation and memory requirements of 3D convolution obviously increase with more complex 3D neural networks. In our future work, we will implement $F(4\times 4\times 4, 3\times 3\times 3)$ to reduce the computation further to ease the intensive computation problem and adopt a FP16 data type to compute, which can save half of the memory usage directly. It is also necessary to parallel the convolution computation among multi-GPUs or multinodes.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors gratefully acknowledge support from the National Key Research and Development Program under no. 2016YFB1000401; the National Nature Science Foundation of China under NSFC nos. 61502509, 61402504, and 61272145; the National High Technology Research and Development Program of China under no. 2012AA012706; and the Research Fund for the Doctoral Program of Higher Education of China under SRFDP no. 20124307130004.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[2] M. Lin, Q. Chen, and S. Yan, Network in network, CoRR abs/1312.4400,.

[3] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[4] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, abs/1409.1556, CoRR, 1409.

[5] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1610–1623, 2010.

[6] S. J. Nowlan and J. C. Platt, "A convolutional neural network hand tracker," *Advances in Neural Information Processing Systems*, pp. 901–908, 1995.

[7] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata, "Pedestrian detection with convolutional neural networks," in *Proceedings of the 2005 IEEE Intelligent Vehicles Symposium*, pp. 224–229, Las Vegas, Nev, USA, June 2005.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, Nev, USA, June 2016.

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition, (CVPR '14)*, pp. 1725–1732, Columbus, Ohio, USA, June 2014.

[10] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[11] "Batch size for training convolutional neural networks for sentence classification," *Journal of Advances in Technology and Engineering Research*, vol. 2, no. 5, 2016.

[12] Y. Xiang, W. Kim, W. Chen et al., "Objectnet3D: A large scale database for 3D object recognition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9912, pp. 160–176, 2016.

[13] A. X. Chang, T. A. Funkhouser, L. J. Guibas et al., *Shapenet: An information-rich 3d model repository., CoRR abs/1512.03012*, An information-rich 3d model repository, Shapenet.

[14] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pp. 922–928, Hamburg, Germany, October 2015.

[15] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[16] D. E. Knuth, *The Art of Computer Programming*, vol. 2, Seminumerical Algorithms, Addison-Wesley Longman Publishing Co, Boston, Mass, USA, 3rd edition, 1997.

[17] S. Winograd, *Arithmetic complexity of computations*, vol. 33 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 1980.

[18] J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8681, pp. 281–290, 2014.

[19] M. Mathieu, M. Henaff, and Y. LeCun, Fast training of convolutional networks through ffts, CoRR abs/1312.5851,.

[20] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, *Fast convolutional nets with fbfft: A GPU performance evaluation, CoRR abs/1412.7580*, Fast convolutional nets with fbfft, A GPU performance evaluation.

[21] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[22] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 4013–4021, July 2016.

[23] U. Aydonat, S. O'Connell, D. Capalija, A. C. Ling, and G. R. Chiu, "An OpenCL™ deep learning accelerator on Arria 10," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2017*, pp. 55–64, Monterey, Calif, USA, February 2017.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 8163949, 9 pages
https://doi.org/10.1155/2017/8163949

Research Article

Convolutional Neural Networks with 3D Input for P300 Identification in Auditory Brain-Computer Interfaces

Eduardo Carabez, Miho Sugi, Isao Nambu, and Yasuhiro Wada

Department of Electrical Engineering, Nagaoka University of Technology, 1603-1 Kamitomioka, Nagaoka, Niigata 940-2188, Japan

Correspondence should be addressed to Eduardo Carabez; eduardo@stn.nagaokaut.ac.jp

Received 3 May 2017; Revised 31 August 2017; Accepted 10 September 2017; Published 7 November 2017

Academic Editor: Athanasios Voulodimos

From allowing basic communication to move through an environment, several attempts are being made in the field of brain-computer interfaces (BCI) to assist people that somehow find it difficult or impossible to perform certain activities. Focusing on these people as potential users of BCI, we obtained electroencephalogram (EEG) readings from nine healthy subjects who were presented with auditory stimuli via earphones from six different virtual directions. We presented the stimuli following the oddball paradigm to elicit P300 waves within the subject's brain activity for later identification and classification using convolutional neural networks (CNN). The CNN models are given a novel single trial three-dimensional (3D) representation of the EEG data as an input, maintaining temporal and spatial information as close to the experimental setup as possible, a relevant characteristic as eliciting P300 has been shown to cause stronger activity in certain brain regions. Here, we present the results of CNN models using the proposed 3D input for three different stimuli presentation time intervals (500, 400, and 300 ms) and compare them to previous studies and other common classifiers. Our results show >80% accuracy for all the CNN models using the proposed 3D input in single trial P300 classification.

1. Introduction

Brain-computer interfaces (BCI) offer a way for people to communicate with devices using their brain. Although the applications and environments in which BCI have been explored are numerous, here we focus on their potential supporting role for people with muscle movement limitations.

Some BCI use event-related potentials (ERP) to link a person's brain to the actuator or device the person intends to interact with. ERP are brain activity patterns that can be measured by electroencephalography (EEG). Among the many ERP, we used P300 for this study. P300 is the positive deflection expected between 250 and 700 ms after the BCI user identifies an irregular (expected) cue among regular ones in an experimental setup. This way of presenting stimuli to the BCI user is known as the oddball paradigm. P300 can be elicited through the oddball paradigm using different stimuli (e.g., sound or image). BCI applications and experiments involving EEG, P300, and image stimuli that focus on people with motor disadvantages have been widely explored and successfully developed in the past [1–3].

For this study, we used sound stimuli to elicit P300 through the oddball paradigm. Although images have been successfully used for such tasks, their use requires that the subjects (who might have physical disabilities) retain control of their eyes and some face and head muscles as well. However, that is not the case for blind people who have lost their ability to see or were never sighted, or for patients with complete locked-in syndrome, who are not in control of their eye movements. By using sound stimuli, we believe that a more portable BCI can be developed, which is suitable for those who cannot receive visual stimuli or simply prefer to dedicate their vision to other tasks.

Once P300 is elicited, the BCI should be able to recognize it and classify it as such. For this purpose, we used several convolutional neural network (CNN) structures. CNN represent a specific topology of a multilayer perceptron (part

of the artificial neural network (ANN) family). Like many other machine learning models, CNN have been used for classification purposes with satisfactory results in different applications [4–7].

Unlike many types of ANN, CNN can handle two- or three-dimensional (2D or 3D) inputs without mapping data onto a one-dimensional (1D) vector, which can be a cause of information loss depending on the nature of the data. Data mapping is common in BCI applications, but as studies show that eliciting P300 causes stronger brain activity in certain brain regions, maintaining both spatial and temporal EEG information when making the CNN input might be key to achieving higher accuracy in P300 classification. With this in mind, we propose a novel 3D input for the CNN. Our approach avoids the information loss that comes with data mapping and allows main CNN operations (convolution and pooling) to take place without the limitations described in other studies [8].

We use our proposed 3D input to test 30 different CNN structures for P300 classification. The CNN structures varied from each other by the kernels (patches) used during the convolution or pooling processes. We also used different pool strides to cause or avoid overlapping, depending on the case, as this has been reported to improve the CNN performance in some applications [9].

The following sections of this work are organized as follows: in Section 2, we explain in detail the experimental setup used to produce and process the dataset used. Further, the general CNN structure and details regarding the shape of the proposed 3D input are presented in Section 3. Finally, in Sections 4 and 5, we discuss our results, comparing them to those obtained in other similar studies and also presenting the performance of other common classifiers used in this context.

2. Dataset

2.1. Experimental Setup. The dataset used for this study corresponds to evoked P300 waves from nine healthy subjects (8 men, 1 women) obtained using an auditory BCI paradigm. A digital electroencephalogram system (Active Two, BioSemi, Amsterdam, Netherlands) was used to record brain activity at 256 Hz. The device consists of 64 electrodes distributed over the head of the subjects by a cap, using the configuration shown in Figure 1(a). This study was approved by the ethics boards of the Nagaoka University of Technology. All subjects signed consent forms that contained detailed information about the experiment and all methods complied with the Declaration of Helsinki.

The subjects were presented with auditory stimuli (100 ms of white noise), similar to that performed in [10], using the out-of-head sound localization method presented in [11], so that subjects could hear the stimuli coming from one of six virtual directions via earphones (see Figure 2(a)). Stimuli were followed by a silent interval of time. One stimulus and one corresponding silent interval were referred to as a trial. Three different trial lengths (500, 400, and 300 ms) were used to analyze the impact of the speed of stimuli presentation on the identification of the P300 wave. Each subject completed 12 experimental sessions, each consisting of around 180 trials for a given trial length. On each session, the subjects were asked to focus on only the sound perceived to be coming from one of the six virtual directions, which was called the target direction. The subjects counted in silence with their eyes closed every time they perceived sound being produced from the target direction and ignored the rest. Ideally, this should elicit P300. The target direction rotated from directions 1 to 6, one by one, for sessions 1 to 6 and then repeated in the same order for sessions 7 to 12. The direction in which stimuli were presented was pseudorandomized; therefore for every six trials, sound from each direction was produced at least once and stimuli coming from the target direction were never produced sequentially to avoid overlapping of P300.

2.2. Preprocessing and Data Accommodation. Before sorting into training and test sets, EEG data were baseline corrected using a Savitzky-Golay filter from −100 ms before stimulus onset until the end of the trial (i.e., end of the silent period after stimulus offset).

A filtering process was also conducted along all EEG channels using Butterworth coefficients for a bandpass filter with cutoff frequencies of 0.1 and 8 Hz. Next data were downsampled to 25 Hz (approximately a tenth of the original size). Data were downsampled as the original size would result in longer processing and training/testing times. Similar downsampling can be found in [10]. Nonaveraged trials were used for this study.

As each subject performed 12 experimental sessions (see Figure 2(b)), with around 180 trials in each of them, data collection for each subject consists of approximately 2160 trials for a given trial length for each subject. Given the pseudorandomized nature of the stimuli production, for each six produced stimuli, one was from the target direction. That stimuli were labeled as the target trial and the rest as nontarget trials. Consequently, of nearly 2160 trials, each subject was expected to produce around 360 target trials as a result of 12 sessions (i.e., a sixth of them), while the remaining are nontarget trials. In this case, the target direction is not particularly relevant, as independently of where the target direction is located, perceiving stimuli correctly from that direction should elicit P300. What is important is to determine is whether the user can differentiate among the six virtual directions and that focusing on one of them and perceiving sound from it are possible with the proposed experimental setup.

Training and test sets were generated for each subject on a given trial length using only that subject's data. To generate the training and test sets for each subject, first we shuffled the target trials with the same happening to the nontarget trials. Next, we distributed half of the target trials in each set with the same applying for nontarget trials. This resulted in training and test sets for each subject containing around 1100 trials each, with approximately 180 target trials and 900 nontarget trials in each set.

As can be seen in Figure 3(c), regardless of the trial length, the proposed input consisted of 1100 ms of recorded brain activity after stimulus onset. We consider the same amount of information to fairly evaluate all trial lengths and compare our results to previous work in Section 4.

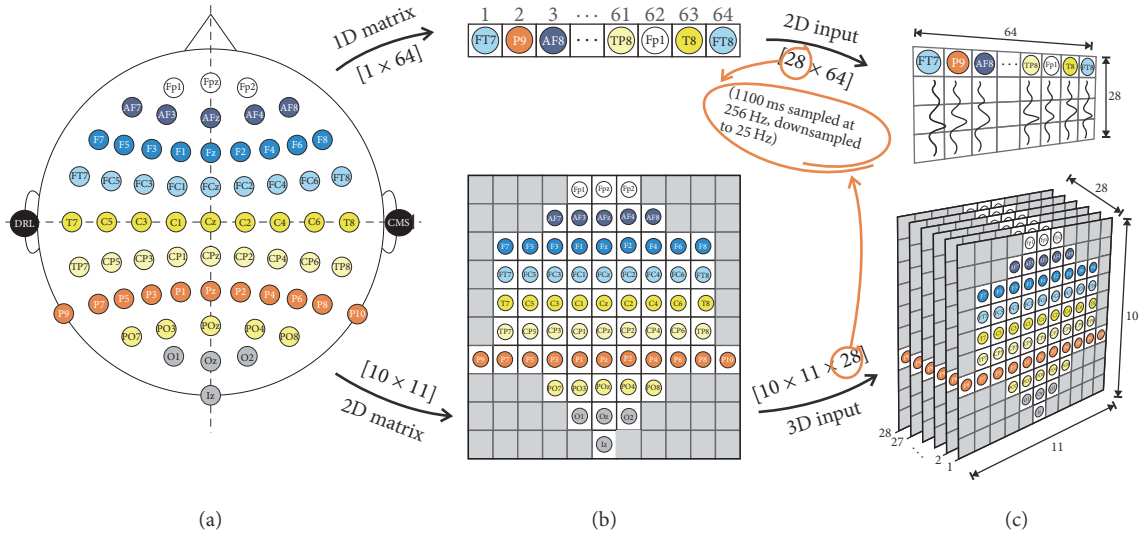


Figure 1: The different steps of input construction: (a) The experimental EEG channel layout. (b) EEG channel matrix disposition to form 2D and 3D inputs (upper and lower images, resp.). Gray cells contain no information. (c) Usual 2D input shape and proposed 3D input shape following our considerations (upper and lower images, resp.).

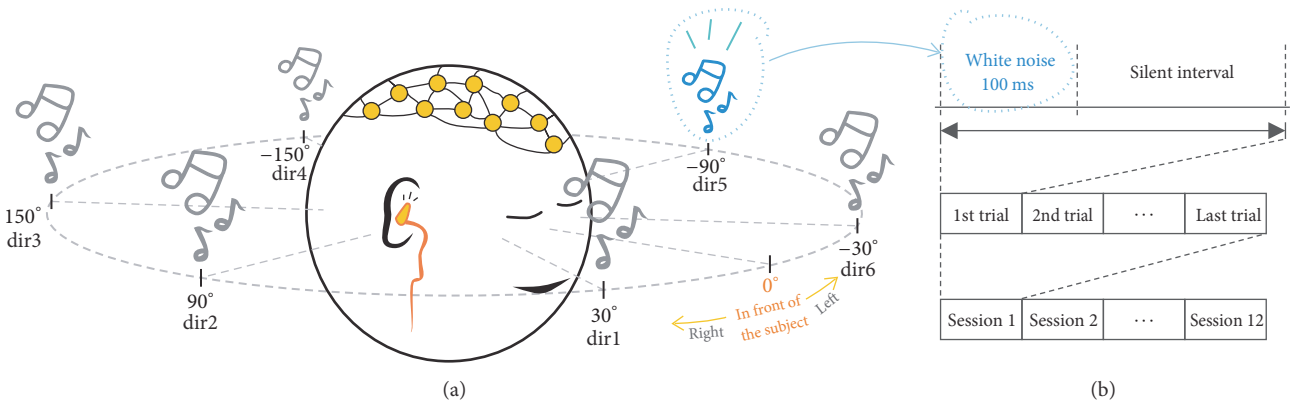


Figure 2: (a) Position representation for the six virtual directions with respect to the subject. (b) Conformation of the 12 sessions all 9 subjects took part of.

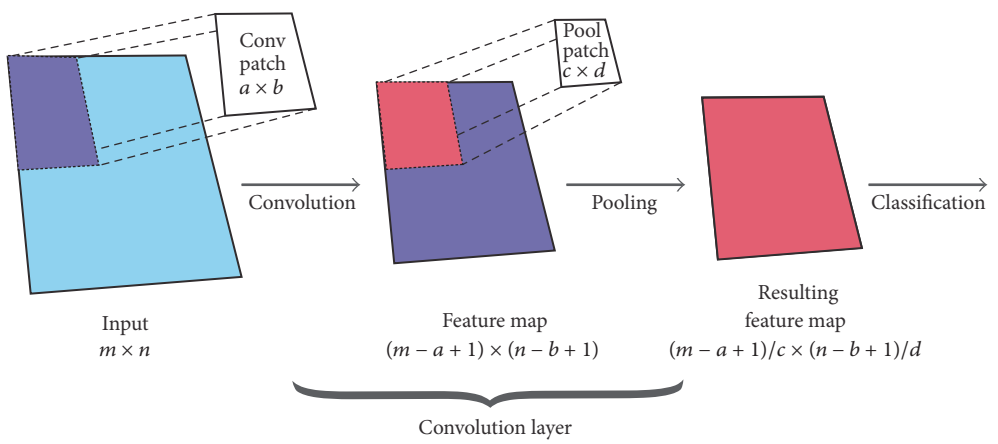


Figure 3: General structure of a CNN. Overlapping is not shown in this figure. Default pooling stride is being applied.

3. Input Shape and CNN Model

3.1. 3D Input. For the detection of P300 using EEG, the electrode position is relevant as there are areas where the potential is experienced more strongly [10]. This, however, has not been addressed in previous research, instead mapping the 3D data (position of electrodes and time) into a 2D vector that contains all EEG channel activity during the experiment. This not only causes information loss, but also prevents classifiers such as CNN to be used without special consideration (as observed in [8]).

To avoid information loss and limitations of CNN operations, positions of the 64 electrodes were mapped onto a 10 × 11 matrix (see Figure 1(b)), maintaining their position as close as possible to their real arrangement in the experimental setup. Time information is presented through an extra axis, so the 3D input has the shape shown in Figure 1(c). Cells that do not correspond to an electrode (gray ones) are set to zero in all instances.

In Section 4, we presented a 2D input for performance comparison purposes. In that case, the input has the shape depicted in Figure 1 (upper flow). The preprocessing, data accommodation (train and test set size), and any other considerations made for the 3D input in Section 2.2 also apply for the 2D one.

3.2. CNN Model. This particular neural network architecture is a type of multilayer perceptron with feature-generation and a dimension-reduction oriented layer, which together compose a convolutional layer. Unlike other layered-based neural networks, CNN can receive a multidimensional input in its original form, process it, and successfully classify it without a previous feature extraction step. The general structure of the CNN is presented in Figure 3. For our study, we used a 3D input and produce 28 feature maps (one for each time sample). While CNN with layers lacking the pooling process are also possible, the pooling process offers scale invariance for the resulting feature maps. It also helps preventing overfitting and allows reduction of computational complexity of the model by reducing the size of the resulting feature maps, thereby shortening training/test times.

Here, we proposed 30 different CNN models to investigate the impact that different convolution and pool patches have on model performance. The proposed models varied from each other in terms of convolution or pool patch size. The CNN models were implemented using a GeForce GTX TITAN X GPU by NVIDIA in Python 2.7 using the work developed by [12].

Additionally, fixed pooling strides were used as an alternative to the default value, which had the same size as the pool patch, with the purpose of forcing pool patches to overlap (or not) during the pooling process, as this has been reported to improve the CNN performance [9]. For this purpose, we applied fixed pooling strides with the values [1 × 1], [1 × 2], [1 × 3], [2 × 2], and [2 × 3]. While normally the pooling stride is given as an integer value, in the work of [12], used in this study, the pooling stride must be defined as an array of two values, with the first one corresponding to the step(s) taken along the x-axis and the second one of those taken along the y-axis. The whole input is spanned using this approach, with only the pooling process affected. For the convolution process, the stride is 1. When a pooling stride different than the default one is used, areas where the pooling patch is applied to the feature map can overlap from one application to another, or contrarily certain areas can be skipped depending on the size of the stride and the pool patch. With our proposed pooling strides, we intended to cause overlapping in the application areas to show whether this impacts the CNN performance (as in [9]). We believe this approach could benefit CNN models as spanning the same area more than once with the max pooling approach could pick up the features corresponding to the P300 production as this wave causes stronger activity in specific brain areas. This should create a resulting feature map containing multiple times this part of the feature map, making classification easier.

Table 1: Proposed convolution, pool patches, and pool stride for the current study.

Patch number	Convolution patch	Pool patch	Pool stride
(0)	[3 × 3]	[2 × 2]	Default
(1)	[2 × 2]	[3 × 3]	[1 × 1]
(2)	[3 × 2]	[1 × 2]	[1 × 2]
(3)	[2 × 3]	[1 × 3]	[1 × 3]
(4)	[2 × 4]	[2 × 3]	[2 × 2]
(5)	[1 × 4]		[2 × 3]

For a given trial length and pool stride value, 30 CNN models were trained for each subject. As there are nine subjects, three trial lengths, and six pool stride values, a total of 4860 CNN were trained for this research. However, only results showing the average performance of the nine subjects will be presented. Tested convolution and pool patches are summarized in Table 1, as well as their patch number, which will be used to present results in the next section.

Each patch is referenced by a number, starting from 0. All possible patch combinations were tested with the resulting model using particular convolution and pool patches, with a patch code consisting of two digits being presented. The first digit corresponds to the convolution patch and the second one to the pool patch. Therefore, for patch code 24, we are referring to the CNN model that used the [3 × 2] convolution patch and the [2 × 3] pool patch. Given that the tested CNN are numerous, we present a statistical analysis in Section 4.1 implementing ANOVA between the models and the proposed pool strides.

As for the learning rate of CNN, it was set at 0.008 based on preliminary tests. The optimization method we used is the stochastic gradient descent as it has been demonstrated [13] to be beneficial for training neural networks on datasets with large examples, using the mini batch approach (batch size of 100). Classification at the output layer is performed using the softmax function, which produces a label based on the probability of a given example to belong to one dataset class.

To calculate classification accuracy we have to consider that the proportion of target and nontarget trials in the

	00	01	02	03	04	10	11	12	13	14
D	0.863	0.858	0.862	0.860	0.859	0.859	0.858	0.862	0.862	0.861
1×2	**0.865**	0.861	0.862	0.862	0.862	0.862	0.861	0.862	0.861	0.862
2×2	0.863	0.861	0.841	0.860	0.859	0.858	0.858	0.861	0.862	0.860
1×3	0.861	0.860	0.862	0.860	0.862	0.862	0.862	0.863	0.862	0.860
2×3	0.860	0.859	0.856	0.858	0.859	0.862	0.859	0.864	0.860	0.861
1×1	0.865	0.858	0.862	0.864	0.862	0.860	0.859	0.863	0.861	0.861
	20	21	22	23	24	30	31	32	33	34
D	0.863	0.859	0.862	0.863	0.861	0.862	0.857	0.862	0.861	0.857
1×2	0.862	0.862	0.862	0.860	0.862	0.862	0.861	0.862	0.862	0.865
2×2	0.863	0.859	0.859	0.836	0.860	0.862	0.860	0.864	0.862	0.862
1×3	0.861	0.859	0.861	0.863	0.861	0.861	0.862	0.859	0.861	0.863
2×3	0.862	0.857	0.840	0.845	0.856	0.838	0.862	0.861	0.860	0.859
1×1	0.863	0.854	0.863	0.861	0.862	0.863	0.858	0.863	0.862	0.864
	40	41	42	43	44	50	51	52	53	54
D	0.862	0.860	0.863	0.861	0.862	0.864	0.861	0.865	0.862	0.861
1×2	0.863	0.861	0.863	0.861	0.862	0.863	0.859	0.865	0.865	0.862
2×2	0.862	0.863	0.862	0.863	0.862	0.864	0.861	0.861	0.837	0.862
1×3	0.863	0.862	0.860	0.845	0.864	0.862	0.862	0.863	0.862	0.863
2×3	0.863	0.862	0.861	0.861	0.862	0.862	0.860	0.859	0.837	0.860
1×1	0.863	0.859	0.862	0.862	0.861	0.862	0.854	0.863	0.863	0.862

Figure 4: Summary of results from nine subjects in the 500 ms trial interval.

	00	01	02	03	04	10	11	12	13	14
D	0.861	0.856	0.861	0.858	0.856	0.859	0.856	0.861	0.860	0.857
1×2	0.861	0.859	0.861	0.860	0.859	0.861	0.858	0.861	0.860	0.859
2×2	0.861	0.856	0.837	0.857	0.858	0.859	0.857	0.859	0.860	0.857
1×3	0.861	0.858	0.861	0.857	0.858	0.858	0.855	0.861	0.860	0.857
2×3	0.858	0.857	0.861	0.856	0.856	0.858	0.858	0.858	0.858	0.857
1×1	0.859	0.855	0.862	0.861	0.858	0.859	0.853	0.860	0.861	0.856
	20	21	22	23	24	30	31	32	33	34
D	0.861	0.857	0.862	0.859	0.856	0.860	0.856	0.860	0.860	0.858
1×2	0.860	0.858	0.862	0.861	0.859	0.861	0.855	0.861	0.861	0.859
2×2	0.861	0.855	0.861	0.837	0.856	0.860	0.859	0.860	0.860	0.857
1×3	0.859	0.859	0.863	0.859	0.860	0.859	0.858	0.858	0.860	0.860
2×3	0.858	0.857	0.838	0.839	0.856	0.833	0.857	0.859	0.858	0.857
1×1	0.859	0.854	0.862	0.858	0.857	0.858	0.853	0.862	0.858	0.855
	40	41	42	43	44	50	51	52	53	54
D	0.862	0.856	0.863	0.859	0.859	0.856	0.853	0.861	0.860	0.858
1×2	0.862	0.855	0.863	0.863	0.861	0.859	0.856	**0.863**	0.860	0.858
2×2	0.862	0.858	0.861	0.860	0.859	0.856	0.858	0.859	0.834	0.857
1×3	0.859	0.857	0.860	0.859	0.859	0.857	0.859	0.860	0.860	0.857
2×3	0.859	0.857	0.860	0.859	0.859	0.856	0.855	0.858	0.835	0.858
1×1	0.857	0.856	0.860	0.862	0.857	0.857	0.854	0.860	0.859	0.857

Figure 5: Summary of results from nine subjects in the 400 ms trial interval.

training and test sets was not even. Thus, we used the expression

$$\text{accuracy} = \sqrt{\frac{\text{TP}}{P} \times \frac{\text{TN}}{N}}, \tag{1}$$

where TP stands for true positives and reflects the number of correctly classified target examples, and TN stands for true negatives and reflects the number of correctly classified nontarget examples. P and N represent the total number of examples of target and nontarget classes, respectively, for this case. This expression heavily penalizes poor individual classification in binary classification tasks.

4. P300 Identification: Results and Discussion

The results presented next correspond to the average accuracy obtained for the nine subjects in testing of CNN models. The highest and lowest accuracy rates are highlighted in bold and red fonts, respectively. By analyzing the performance obtained using different pooling strategies in the form of different fixed pooling strides (presented in the first column from left to right), it is often observed that some pooling strategies do not offer relevant differences at first glance.

By analyzing the summarized results for the three trial intervals, we found no clear tendency for which model and pool stride offer the highest or lowest accuracies. For instance, in the 500 ms trial interval models (Figure 4), the lowest accuracy was obtained from the model with patch code 23 and $[2 \times 2]$ pool stride, while in both the 400 and 300 ms cases, these results were obtained using the model with patch code 30 and $[2 \times 3]$ pool stride, which is similar to the 500 ms case.

As for the highest accuracy results, there are some similarities in the 400 and 300 ms trial intervals (Figures 5 and 6, resp.). In these cases, the implemented models used

	00	01	02	03	04	10	11	12	13	14
D	0.848	0.847	0.851	0.850	0.849	0.847	0.845	0.850	0.850	0.848
1×2	0.848	0.849	0.851	0.851	0.849	0.849	0.849	0.850	0.850	0.849
2×2	0.848	0.848	0.835	0.848	0.849	0.847	0.847	0.851	0.850	0.846
1×3	0.849	0.848	**0.852**	0.850	0.849	0.848	0.846	0.851	0.850	0.848
2×3	0.850	0.847	0.850	0.850	0.849	0.849	0.848	0.848	0.848	0.848
1×1	0.848	0.848	0.851	0.850	0.849	0.847	0.847	0.849	0.850	0.849
	20	21	22	23	24	30	31	32	33	34
D	0.848	0.846	0.851	0.849	0.847	0.850	0.844	0.850	0.850	0.845
1×2	0.849	0.849	0.851	0.848	0.849	0.850	0.847	0.850	0.850	0.848
2×2	0.848	0.847	0.851	0.833	0.849	0.850	0.848	0.850	0.848	0.847
1×3	0.849	0.847	0.851	0.849	0.849	0.848	0.848	0.849	0.850	0.847
2×3	0.849	0.846	0.835	0.836	0.847	0.832	0.846	0.848	0.847	0.845
1×1	0.848	0.847	0.849	0.848	0.849	0.850	0.848	0.850	0.850	0.848
	40	41	42	43	44	50	51	52	53	54
D	0.849	0.846	0.848	0.850	0.848	0.847	0.844	0.848	0.850	0.845
1×2	0.849	0.848	0.848	0.848	0.848	0.851	0.845	0.848	0.848	0.845
2×2	0.848	0.847	0.847	0.850	0.848	0.847	0.845	0.846	0.833	0.846
1×3	0.849	0.848	0.851	0.850	0.850	0.846	0.846	0.849	0.850	0.848
2×3	0.847	0.848	0.850	0.848	0.848	0.847	0.845	0.847	0.834	0.845
1×1	0.849	0.846	0.850	0.849	0.848	0.846	0.843	0.851	0.846	0.847

Figure 6: Summary of results from nine subjects in the 300 ms trial interval.

pool patch (2) under the $[1 \times 3]$ or $[1 \times 2]$ pooling strides, which prevent the models from overlapping and are very similar one to each other. For the 500 ms case, pooling stride was also $[1 \times 2]$, the same as in the 400 ms trial interval, while the pool patch was different. If we look back at Table 1, we can see that these convolution patches are quite different from each other.

By analyzing Figure 8, it can be noted that even if the results do not vary strongly one from another, there is a clear pattern of improved performance using data from the 500 ms trial length, followed by the 400 and 300 ms ones. This behavior is expected, as faster production of stimuli can cause subjects to fail to identify stimuli coming from the target direction and therefore incorrectly produce the P300 wave.

Table 2: Summary of the highest, lowest, and average accuracies obtained for the CNN models using the 3D input.

	Highest	Lowest	Average
500 ms	0.865	0.836	0.86
400 ms	0.863	0.833	0.858
300 ms	0.852	0.832	0.848

By summarizing the results in this way, we also observed that the [1 × 2] pooling stride offers the best results, at least in the 500 and 400 ms trial length, while the [1 × 3] pool stride is optimal on the 300 ms trial length.

On the other hand, the [2 × 3] pool stride produces the lowest results, without being detrimental. Differences between the highest and lowest pool strides rely on how much overlapping the strides provide. While the [2 × 3] pooling stride prevents some pool patches from overlapping at all or even skipping some areas of the input, the [1 × 3] pooling stride forces most pool patches to overlap. In the study by [13], no differences were reported between performance for approaches with or without overlapping, contrary to the report by [9], where better CNN model performance was achieved using overlapping pool strategies. In our case, we found little to no change between different pooling strategies tested. In the above cited research, it is mentioned that success in applying pooling strategies might depend completely on the nature, shape, and conditions of the used data.

In Table 2, the average values considering all models for each of the three trial lengths are presented next to their corresponding lowest and highest values.

By comparing these numbers, apparently there are no big differences between trial lengths and their highest/lowest values. We believe this lack of variation between the many tested models is the result of the implementation of the 3D input, which, regardless of the speed of the stimuli presentation used in this study, can present the necessary information for correct classification. To support this idea, we tested 4 additional CNN models using the commonly 2D input approach with convolution patches [1 × 4] and [3 × 3]. As for the pool patches, we also tested two, with sizes [1 × 2] and [2 × 2], both with a default pooling stride as our results so far indicate the pooling strategies do not offer significant CNN performance differences. The patches were chosen as they are the same as those used by the CNN models with the best results using the 3D input. Besides the input shape and the convolution and pool patches, the parameters of the CNN models using the 2D input do not differ from the ones presented so far in Section 3.2. The results from the models using the 2D input can be seen in Table 3.

In the case of the results for the models using the 2D input, the difference between models and trial lengths is more easily noticed. While in the results involving the 3D input the difference of the overall highest and lowest accuracy is of about 3%; in the case of the 2D input results, the difference is around 10%. Also, we can see that the convolution and pool patches that consider information from only one channel at a time offer better results than those in which information from multiple channels is considered.

Table 3: Proposed convolution and pool patches for the CNN models in which a 2D input was implemented. CP and PP stand for *convolution patch* and *pool patch*, respectively.

CP	PP	Accuracy		
		500 ms	400 ms	300 ms
[1 × 4]	[1 × 2]	**0.781**	**0.768**	**0.734**
	[2 × 2]	0.753	0.716	0.727
[3 × 3]	[1 × 2]	0.766	0.725	0.732
	[2 × 2]	0.724	0.707	0.698
Average		0.756	0.729	0.722

Table 4: Results for each subject in those models with the highest accuracy for each trial length considering the CNN models with both the 3D and 2D input approach. The subject's number appears on the first column to the left.

	500 ms		400 ms		300 ms	
	3D	2D	3D	2D	3D	2D
(1)	0.880	0.72	0.877	0.736	0.859	0.685
(2)	0.873	0.823	0.866	0.78	0.878	0.744
(3)	**0.896**	**0.774**	**0.908**	**0.794**	0.859	0.774
(4)	0.828	0.812	0.83	0.766	0.831	0.785
(5)	0.864	0.78	0.851	0.792	0.826	0.697
(6)	0.828	0.808	0.828	0.763	0.825	0.685
(7)	0.881	0.788	0.858	0.752	**0.886**	**0.796**
(8)	0.879	0.792	0.904	0.77	0.847	0.721
(9)	0.863	0.736	0.847	0.76	0.853	0.722

In Table 4 the individual results for each subject considering the models with the highest accuracy for each trial length using both 3D and 2D approaches are shown. The models with highest accuracy are those presented with bold font in Figures 4–6 and Table 3.

The results obtained show individual performance patterns appearing in both approaches in a similar way. For a given trial length, the subject with the highest individual accuracy is the same regardless of the approach (3D or 2D). Although in some cases the accuracy difference between both approaches for a single subject is minimal, all the results from the CNN models using 3D input offered better accuracies.

Besides the difference in the obtained accuracies, the train/test of the 3D input models was also faster than that of the models using the 2D input. The average time of the 3D input models for training/testing a single subject was around 8 minutes, while for the case of the models using the 2D input, around 18 minutes were necessary.

4.1. Statistical Analysis. Given that the results obtained so far do not show big differences of the performance of the CNN models whether we consider the models themselves or the trial length in which they were tested, we conducted an analysis of variance (ANOVA) to further examine the results.

First, we checked if applying the different tested models (variation of convolution and pool patch sizes) had a significant impact on the performance of the CNN models. The

Table 5: Results for the ANOVA between the 30 tested models. The critical F value is 1.54.

	F	p	Significant differences
500 ms	1.29	0.159	No
400 ms	1.44	0.08	No
300 ms	1.55	0.047	Yes

Table 6: Results for the ANOVA between the 6 implemented pool strides. The critical F value is 2.26.

	F	p	Significant differences
500 ms	4.21	0.001	Yes
400 ms	4.72	0.0004	Yes
300 ms	5.015	0.0002	Yes

results are shown in Table 5 for the three trial lengths that were considered for this study.

We found that, for the models using examples from the 500 and 400 ms trial lengths, there were no significant differences, but there were ones for the case of the 300 ms trial length. As the trial length becomes shorter, it becomes harder for users to correctly identify the sound coming from the target direction and in this case the differences between models become clearer.

Next, we present the results for the ANOVA between the tested pool strides in Table 6. In this case, there are significant differences among the implemented pool strides regardless of the trial length. We proposed applying several pooling strides to explore whether by causing or avoiding overlapping during the pooling process the performance of the models improved. In Figure 5 we have the average accuracy for all the models under each pool stride displayed, but the small differences between the results made it difficult to state if they were different enough to make an assessment. Now, the results in Table 6 show that varying the pooling stride to cause overlapping or avoid it significantly impacts the performance of the tested CNN models.

4.2. Comparison with Previous Work and Other Classifiers. The current results show an improvement of around 15% over the work of [10], from where the experimental setup for this research was borrowed (see Figure 7). The EEG data was obtained also in a similar fashion, enabling the current comparison. Also, this study shares some similarities with that done by [14] which is why we also include it in the comparison. In most cases, it is difficult to make an appropriate comparison due to the differences in the nature of the experiments, subjects, and technologies used and for such reasons, the comparison is only demonstrative. All the results used for comparison in Figure 7 are the ones corresponding to the single trial (also noted as nonaveraged) case. The highest results obtained for the implementation of the 2D input are also included.

We now compare the results from the models using the proposed 3D input with those obtained by using support vector machines (SVM) and Fisher's discriminant analysis (FDA). We chose to use these two classifiers as they are

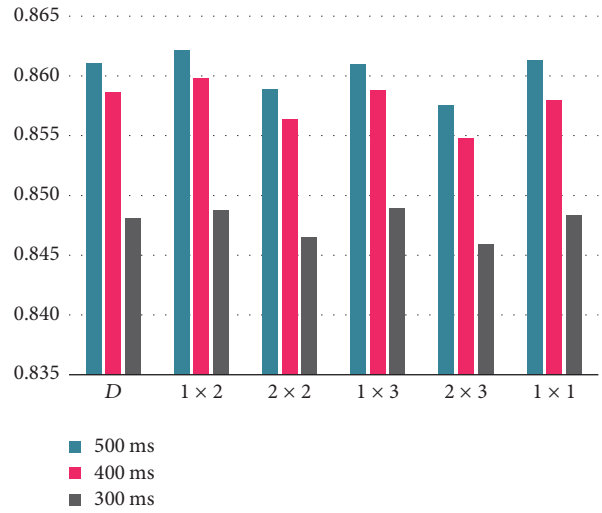


Figure 7: Averaged accuracy rate by pooling stride for three proposed trial intervals.

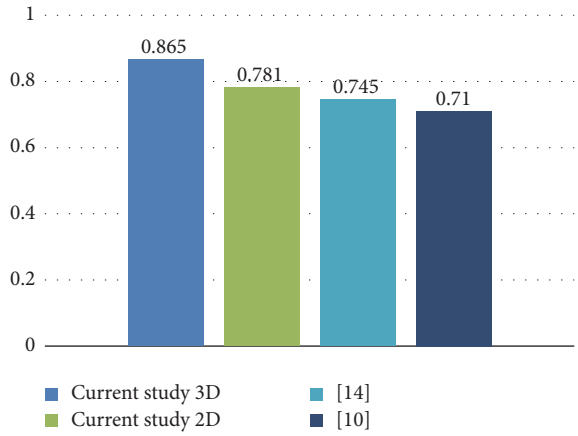


Figure 8: Average accuracy rate for the single trial case. Conditions of experiment and subjects might differ between the studies.

Table 7: Comparison between the highest accuracies obtained using the proposed 3D input for CNN models, a SVM, and a FDA.

	CNN 3D	SVM	FDA
500 ms	0.865	0.709	0.745
400 ms	0.863	0.711	0.731
300 ms	0.852	0.691	0.707

common in this context and the SVM was used in [10]. Table 7 shows the results for comparison of the highest accuracies obtained in the different trial lengths. Details about the SVM and FDA can be found in the appendix.

Both the FDA and the SVM offer accuracies below those from the model using the proposed input.

5. Final Comments and Future Work

Through this research we found that it is possible to implement a 3D input shape using EEG data with success for

different CNN models that exhibit different pooling strategies based on proposed fixed pooling strides that might cause the models to overlap during the pooling process or avoid it. We hypothesize that using this approach might yield better results compared to the most common approaches which use 2D mapped version of the data. The basis of such thinking lies in the nature of the convolution and pooling processes, which highly depend on the relation between a data point and its surroundings. This lack of variation might point to a better representation of the information given by the proposed 3D input. Also, we found that, for the current study, causing overlapping with fixed pooling stride significantly impacts the performance of the tested CNN models.

The obtained results showed improvement over others seen in similar studies using nonaveraged data. Also, when compared to other classifiers commonly used in this context, the CNN models using the proposed input performed better.

While we believe this was a successful application of a novel input structure, we consider that such construction will perform particularly well when the nature of the data is such that mapping it to simpler representations comes with information loss. For other BCI approaches as well as for other kinds of brain activity readings, this approach might not be the best fit and its application might require a case by case analysis.

With the proposed 3D input we were able to find also a faster way to train/test, as this approach showed taking less time for such tasks than the models using a 2D input. We expect to keep using this input representation to test its limitations and possible new applications in future studies.

Appendix

SVM. Analysis was performed using LIBSVM software [15] and implemented in MATLAB (Mathworks, Natick, USA). We used a weighted linear SVM [16] to compensate for imbalance in the target and nontarget examples. Thus, we used a penalty parameter of C+ for the target and C− for the nontarget examples. The penalty parameter for each class was searched in the range of 10^{-6} to 10^{-1} ($10^{-6} \leq 10^{m} \leq 10^{-1}$; m: $-6:0.5:-1$) within the training. We determined the best parameters as those obtaining the highest accuracy using 10-fold cross-validation for the training. Using the best penalty parameters, we constructed the SVM classifier using all training data and applied it to the test data.

FDA. We used a variant of the regularized Fisher discriminant analysis (FDA) as the classification algorithm [14]. In this algorithm, a regularized parameter for FDA is searched for by particle swarm optimization (for details, see [14]) within the training. In this study, we used all EEG channels without selection.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partly supported by JSPS Kakenhi Grant nos. 2430051 and 16K00182 and Nagaoka University of Technology Presidential Research Grant.

References

[1] B. Rebsamen, E. Burdet, C. Guan et al., "Controlling a wheelchair indoors using thought," *IEEE Intelligent Systems*, vol. 22, no. 2, pp. 18–24, March 2007.

[2] E. W. Sellers and E. Donchin, "A P300-based brain-computer interface: initial tests by ALS patients," *Clinical Neurophysiology*, vol. 117, no. 3, pp. 538–548, 2006.

[3] M. Chang, N. Nishikawa, Z. R. Struzik et al., Comparison of P300 Responses in Auditory, Visual and Audiovisual Spatial Speller BCI Paradigms, ArXiv e-prints, Jan. 2013.

[4] H. Cecotti and A. Gräser, "Time Delay Neural Network with Fourier transform for multiple channel detection of Steady-State Visual Evoked Potentials for Brain-Computer Interfaces," in *Proceedings of the 16th European Signal Processing Conference (EUSIPCO '08)*, pp. 1–5, August 2008.

[5] I. Güler and E. D. Übeyli, "Multiclass support vector machines for EEG-signals classification," *IEEE Transactions on Information Technology in Biomedicine*, vol. 11, no. 2, pp. 117–126, 2007.

[6] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association, INTERSPEECH 2013*, pp. 3366–3370, fra, August 2013.

[7] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR, vol. abs/1409.1556, 2014, http://arxiv.org/abs/1409.1556.

[8] H. Cecotti and A. Gräser, "Convolutional neural networks for P300 detection with application to brain-computer interfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 433–445, 2011.

[9] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[10] I. Nambu, M. Ebisawa, M. Kogure, S. Yano, H. Hokari, and Y. Wada, "Estimating the Intended Sound Direction of the User: Toward an Auditory Brain-Computer Interface Using Out-of-Head Sound Localization," *PLoS ONE*, vol. 8, no. 2, Article ID e57174, 2013.

[11] S. Yano, H. Hokari, and S. Shimada, "A study on personal difference in the transfer functions of sound localization using stereo earphones," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 83, no. 5, pp. 877–887, 2000.

[12] I. J. Goodfellow, D. Warde-Farley, P. Lamblin et al., Pylearn2: a machine learning research library, ArXiv e-prints, Aug. 2013.

[13] T. N. Sainath, B. Kingsbury, G. Saon et al., "Deep Convolutional Neural Networks for Large-scale Speech Tasks," *Neural Networks*, vol. 64, pp. 39–48, 2015.

[14] A. Gonzalez, I. Nambu, H. Hokari, and Y. Wada, "EEG channel selection using particle swarm optimization for the classification of auditory event-related potentials," *The Scientific World Journal*, vol. 2014, Article ID 350270, 11 pages, 2014.

[15] C. Chang and C. Lin, "LIBSVM: a Library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.

[16] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proceedings of the 7th IEEE Workshop on Neural Networks for Signal Processing (NNSP '97)*, pp. 276–285, September 1997.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 1583847, 11 pages
https://doi.org/10.1155/2017/1583847

Research Article

Chaos Quantum-Behaved Cat Swarm Optimization Algorithm and Its Application in the PV MPPT

Xiaohua Nie,[1] Wei Wang,[1] and Haoyao Nie[2]

[1]*Information Engineering School, Nanchang University, Nanchang, Jiangxi Province 330031, China*
[2]*Economics Management School, Nanchang University, Nanchang, Jiangxi Province 330031, China*

Correspondence should be addressed to Xiaohua Nie; niexiaoh@163.com

Received 14 January 2017; Revised 23 March 2017; Accepted 11 September 2017; Published 17 October 2017

Academic Editor: Athanasios Voulodimos

Cat Swarm Optimization (CSO) algorithm was put forward in 2006. Despite a faster convergence speed compared with Particle Swarm Optimization (PSO) algorithm, the application of CSO is greatly limited by the drawback of "premature convergence," that is, the possibility of trapping in local optimum when dealing with nonlinear optimization problem with a large number of local extreme values. In order to surmount the shortcomings of CSO, Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm is proposed in this paper. Firstly, Quantum-behaved Cat Swarm Optimization (QCSO) algorithm improves the accuracy of the CSO algorithm, because it is easy to fall into the local optimum in the later stage. Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm is proposed by introducing tent map for jumping out of local optimum in this paper. Secondly, CQCSO has been applied in the simulation of five different test functions, showing higher accuracy and less time consumption than CSO and QCSO. Finally, photovoltaic MPPT model and experimental platform are established and global maximum power point tracking control strategy is achieved by CQCSO algorithm, the effectiveness and efficiency of which have been verified by both simulation and experiment.

1. Introduction

Solar energy is widely used due to the advantages of renewable and nonpolluting. Global maximum power point tracking (GMPPT) is one of the basic means to improve the overall efficiency of photovoltaic power generation system. However, the traditional maximum power point tracking (MPPT) algorithm is often ineffective because of the output power curve with multipeak problem in complex shade conditions. The problem results in efficiency reducing of photovoltaic power generation [1, 2]. Many scholars have carried out massive researches to solve these questions. Many optimization algorithms are used to realize global maximum power point tracking and achieve good results, like Particle Swarm Optimization (PSO) algorithm, evolutionary algorithm, fuzzy logic control algorithm, neural network control algorithm, chaos search algorithm, and so on [2–9].

The Cat Swarm Optimization (CSO) algorithm was proposed by Chu et al. in 2006 [10]. The experimental results showed that the CSO algorithm could find the global optimal solution in a short time, and the CSO algorithm is better than the PSO algorithm in the convergence speed and nonlinear optimization of the local extremum [11–18]. However, the CSO algorithm has the shortcoming of "premature convergence" in the practical application. If the number of iteration is increased continuously, it will cause the convergence time to multiply while the accuracy of the optimal value is not significantly improved. The same problems also occur in the PSO algorithm. A novel chaotic quantum-behaved PSO algorithm is proposed for solving nonlinear system of equations in [19]. Different chaotic maps are introduced to enhance the effectiveness and robustness of the algorithm. The comparison of results reveals that the proposed algorithm can cope with the highly nonlinear problems and outperform other algorithms. The Hybrid Chaotic Quantum-behaved Particle Swarm Optimization (HCQPSO) algorithm is used for thermal design of plate fin heat exchangers in [20]. The HCPQSO algorithm successfully combines a

variant of Quantum-behaved Particle Swarm Optimization with efficient local search mechanisms to yield better results in terms of solution accuracy and convergence rate. It is also observed that the proposed algorithm successfully converges to optimum configuration with a higher accuracy. A chaotic improved Particle Swarm Optimization algorithm is proposed for photovoltaic MPPT in [21]. An improved cuckoo search (ICS) algorithm is proposed to establish the parameters of chaotic systems in [22]. The numerical results demonstrate that the algorithm can estimate parameters with high accuracy and reliability. An evolutionary approach of parking space guidance is proposed based upon a novel Chaotic Particle Swarm Optimization (CPSO) algorithm in [23]. The Chaotic Firefly algorithm using tent maps is proposed for optimally coordinating the relays in [24]. Chaos theory has been incorporated to prevent the search process from being trapped in local minima by modifying the concept of random movement factor variable.

In this paper, a Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm is proposed. The cat with the quantum behavior has no definite trajectory in tracking and has the possibility of getting rid of the local optimal point with large disturbance. Since there is only position vector in the control parameter, with no velocity vector, and the convergence time is shortened, then, the tent chaotic map is introduced to change the cat position information in the iteration process, the "premature convergence" problem of the CSO algorithm is avoided, and the precision of searching is further improved. In Section 2, the CQCSO algorithm is tested in five nonlinear function curves with multiple local extreme points. The results show that the proposed CQCSO algorithm has high tracking precision and fast convergence speed. Moreover, the proposed CQCSO algorithm has the ability to adapt to complex and different curves. In Section 3, the proposed CQCSO algorithm is applied to PV MPPT control, and a multipeak MPPT control strategy based on CQCSO algorithm is proposed. The simulation and experimental results show that the proposed control strategy more efficiently tracks the global maximum power points.

2. Chaos Quantum-Behaved Cat Swarm Optimization (CQCSO) Algorithm

2.1. QCSO Algorithm. The CSO algorithm is a kind of swarm intelligence algorithms based on the cat's living habits and foraging. It is superior to the PSO in searching accuracy and convergence time [12–19]. It is widely used to solve the optimization problem. In the CSO algorithm, the cat's position is regarded as a feasible solution to the optimization problem, and then the cat swarms are divided into two groups according to the mixture ratio (MR), which are the searching cat group and the tracking cat group. The searching cat group refers to the genetic algorithm to complete the cat's location update. The cats with the highest fitness value are selected to replace the current cat position by copying and mutating individuals. The tracking cat groups are similar to the PSO algorithm and use the cat's own speed and the current position information to update the position of the cat continuously, so that each individual can move closer to the global optimal solution. Although the search of the global optimal solution can be realized, there exists shortcoming of "premature convergence" problem like other swarm intelligent algorithms.

Quantum-behaved Cat Swarm Optimization (QCSO) algorithm is a combination of CSO algorithms and quantum mechanics. In the evolutionary process, each tracing cat has a B_i-centered DELTA potential well, which makes each tracing cat converge to an attractor B_i. It continuously updates the location of the cat by tracking individual extremes and global extremes, so that the cat's speed and location are uncertain. So, it can be distributed in a certain probability to search space at any position. It is possible to get rid of the local optimal points in disturbing environment. As a result, the QCSO algorithm can jump out of the local optimum and improve the accuracy of CSO algorithm.

The updated expression of individual position in quantum space is

$$X_i^{k+1} = B_i^k + ba_k \left| C^k - X_i^k \right| \ln\left(r_1^{-1}\right), \tag{1}$$

where

$$\begin{aligned} B_i^k &= r_2 D_i^k + \left(1 - r_2\right) G^k, \\ b &= \begin{cases} -1, & \left(r_3 \le 0.5\right) \\ 1, & \left(r_3 > 0.5\right), \end{cases} \\ a_k &= a_1 - \frac{\left(a_1 - a_2\right) k'}{\overline{k'}}, \\ \left(C_1^k, C_2^k, \ldots, C_d^k\right) &= \frac{1}{m}\left(\sum_{i=1}^{m} D_{i1}^k, \sum_{i=1}^{m} D_{i2}^k, \ldots, \sum_{i=1}^{m} D_{id}^k\right), \end{aligned} \tag{2}$$

where m is population size, $i = 1, 2, \ldots, m$, d is dimension, k is maximum number of iterations, $k = \underline{1}, 2, \ldots, \overline{k}$, k' is maximum number of traces, $k' = 1, 2, \ldots, \overline{k'}$, C^k is the cat group optimal position center of the kth iteration, X_i^k is the optimal position of the ith cat for the kth iteration, D_i^k is the optimal position of the ith cat for the kth iteration, G^k is the global optimal position of the population at the kth iteration, a_k is the kth iteration expansion contraction factor, a_1, a_2 are the initial compression factor and the termination value, respectively, $a_1 = 1.0, a_2 = 0.5$, and r_1, r_2, r_3 are random numbers with uniform distribution in the $[0, 1]$ interval.

2.2. CQCSO Algorithm. The QCSO algorithm improves the accuracy of the CSO algorithm, because the cat in the evolutionary process continues to move closer to the optimal position of the population, the diversity of the population gradually decreases, and it is easy to fall into the local optimum in the later stage. Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm is proposed by introducing tent map for jumping out of local optimum in this paper.

The individual location update expression of tent map is as follows:

$$x_i^{k+1} = \begin{cases} 2x_i^k, & 0 \le x_i^k \le 0.5 \\ 2\left(1 - x_i^k\right), & 0.5 < x_i^k \le 1, \end{cases} \tag{3}$$

where $x_i^k \in [0\ \ 1]$, $i = 1, 2, \ldots, m$, $k = 1, 2, \ldots, \overline{k}$. $x_i^k \in [0\ \ 1]$ can be mutually mapped transformation with the chaotic variables $Y_i^k \in [a\ \ b]$ through

$$x_i^k = \frac{\left(Y_i^k - a\right)}{(b-a)}, \tag{4}$$

$$Y_i^k = a + x_i^k (b-a). \tag{5}$$

Detailed CQCSO algorithm steps are as follows.

Step 1. Initialize the cat swarm, set the population size m, the maximum iteration number $\overline{k}$, and the mixture ratio (MR) of the cat optimization algorithm, and randomly initialize position of the cat population between $[a\ b]$; it is expressed with the row vector Y.

Step 2. The fitness value of all cats in the population was calculated, and the cat with the greatest fitness was selected and recorded.

Step 3. According to MR, cats swarms are randomly grouped. MR represents the proportion of the number of cats in the tracking group in the entire cat population. MR is generally a smaller value to ensure that most of the cats in the swarm are in search mode and a few cats are in tracking mode.

Step 4. When the cat is in the search group, it replicates its position according to the size of seeking memory pool (SMP), executes the selection operator, updates SMP, and replaces the position of the current cat with the candidate point with the highest fitness value. In the end, the optimal value updating is completed. The cat of tracking group updates its own position information according to formula (1).

Step 5. The cat with the best fitness is recorded in the reserved populations.

Step 6. It is judged whether the termination condition is satisfied, and if so, the program ends and the optimal solution is output. If it is not satisfied, it is judged whether the position of the global optimal cat is the same after the kth and $k-1$ iterations, and if not, Steps 3–6 are repeated. If it is the same, it means that it has fallen into local optimum and needs to deal with chaos. The mapping of the normal variable Y_i^k is performed by using formula (4). The chaotic variables x_i^k after mapping are in the range [0 1]. Chaos variable x_i^k is mapped to get x_i^{k+1} using formula (3). And then through formula (5), chaos variable x_i^{k+1} mapping transformation, the next iteration of the conventional variables Y_i^{k+1} is obtained. Steps 2–6 are repeated to optimize iteration.

2.3. Verification for CQCSO Algorithm. In order to verify the superiority of the CQCSO algorithm, five kinds of nonlinear functions with multiple local extremum peaks such as Schaffer, Shubert, Griewank, Rastrigrin, and Rosenbrock are compared for seeking optimization. In simulation, the total number of cat groups is set to 20; each function program runs 50 times.

(1) Schaffer Function

$$\min f(x_1, x_2) = 0.5 + \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}, \tag{6}$$

where $x_1, x_2 \in [-10, 10]$; Schaffer function is a two-dimensional complex function with numerous small points. The minimum value 0 is obtained at (0, 0). Because this function has strong concussion, it is hard to find the global optimal value. The seeking optimization result is obtained and shown in Figure 1(a).

(2) Shubert Function

$$\min f(x, y) = \left\{\sum_{i=1}^{5} i\cos\left[(i+1)x + i\right]\right\} \times \left\{\sum_{i=1}^{5} i\cos\left[(i+1)y + i\right]\right\}, \tag{7}$$

where $x, y \in [-10, 10]$; Shubert function is a two-dimensional complex function with 760 local extrema points. The global minimum value −186.7309 is obtained at (−1.42513, 0.80032). The seeking optimization result is obtained and shown in Figure 1(b).

(3) Griewank Function

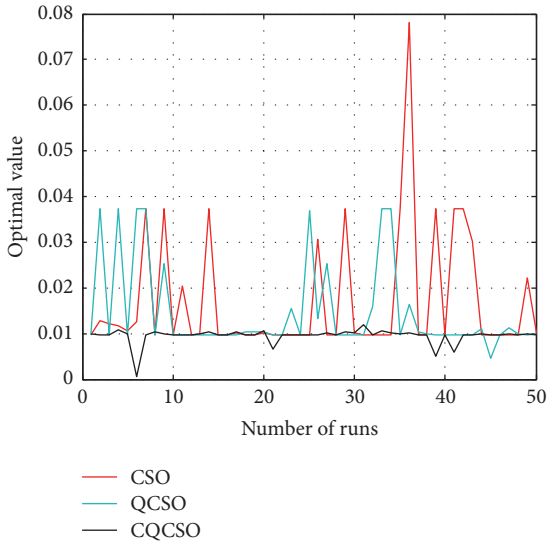
$$\min f(x_i) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{8}$$

where $x_i \in [-600, 600]$; Griewank function has many local minimums whose numbers are related to the dimension. The global minimum value 0 is obtained at $(x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$. Griewank function is a typical nonlinear multimodal function. It is usually considered to be a complex multimodal problem that is difficult to handle by the optimization algorithm. The function dimension D is set to 3. The seeking optimization result is obtained and shown in Figure 1(c).

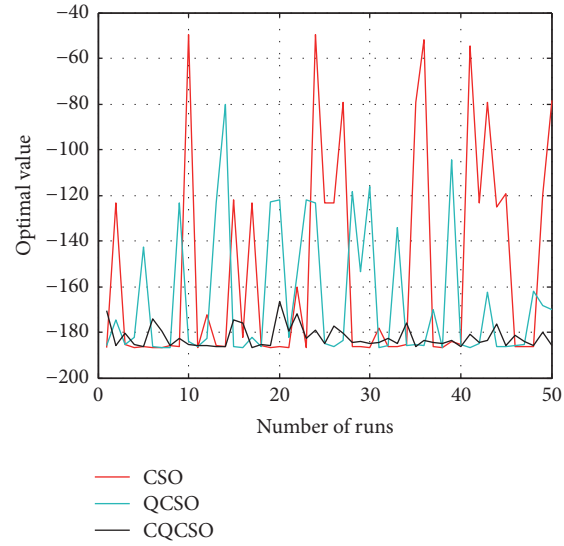
(4) Rastrigrin Function

$$\min f(x_i) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right], \tag{9}$$

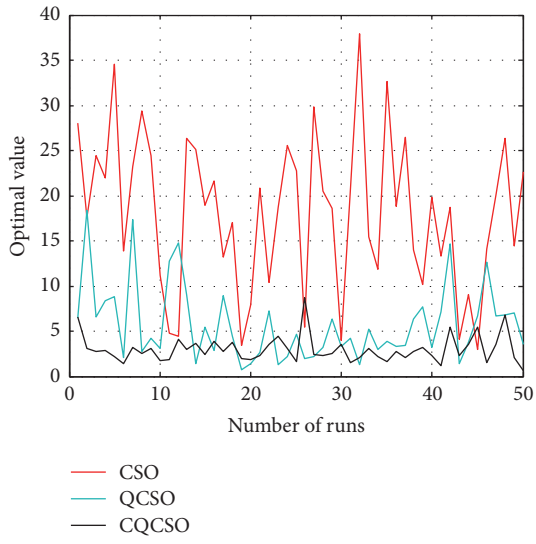
where $x_i \in [-5.12, 5.12]$; Rastrigrin function is a multimodal function with about 10D local minima. The global minimum value 0 is obtained at $(x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$. Since its



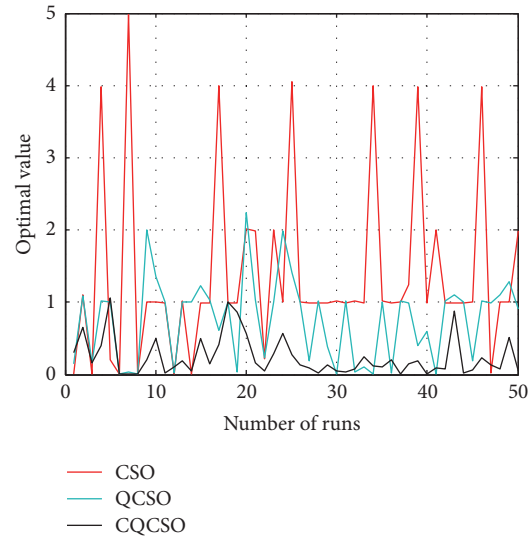
(a) Seeking optimization result of Schaffer function



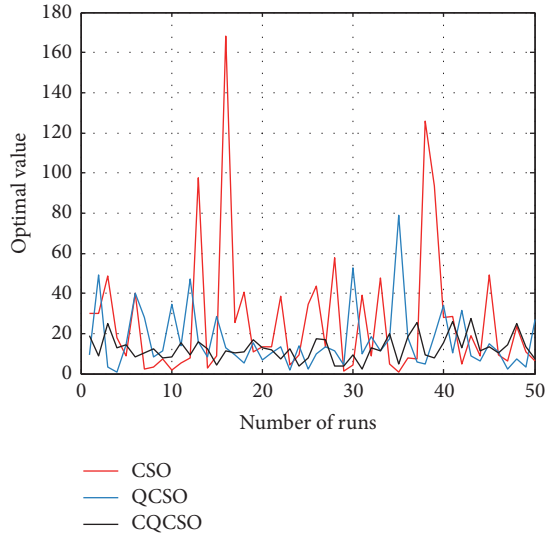
(b) Seeking optimization result of Shubert function



(c) Seeking optimization result of Griewank function



(d) Seeking optimization result of Rastrigrin function



(e) Seeking optimization result of Rosenbrock function

Figure 1: Seeking optimization results of five function.

Table 1: Function simulation data.

Algorithm		CSO	QCSO	CQCSO
Schaffer	TIME/s	0.0193	0.0161	0.0156
	BEST	0.0171	0.0147	0.0095
	STD	0.0137	0.0098	0.0017
Shubert	TIME/s	0.2077	0.1776	0.1756
	BEST	−154.4	−166.2	−182.2
	STD	46.350	28.700	4.721
Griewank	TIME/s	0.0317	0.0254	0.0242
	BEST	18.040	5.729	2.987
	STD	8.6060	4.2070	1.5040
Rastrigrin	TIME/s	0.7721	0.6363	0.6343
	BEST	1.3750	0.7618	0.2455
	STD	1.2450	0.5724	0.2701
Rosenbrock	TIME/s	0.0393	0.0338	0.0331
	BEST	26.450	16.710	12.620
	STD	33.250	15.480	6.048

peak shape appears to fluctuate undulatingly, it is difficult to find the global optimal value. The function dimension D is set to 3. The seeking optimization result is obtained and shown in Figure 1(d).

(5) Rosenbrock Function

$$\min f(x_i) = \sum_{i=1}^{D-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right], \quad (10)$$

where $x_i \in [-2.048, 2.048]$; the global optimal point of Rosenbrock function is located in a smooth, narrow parabolic valley. It is difficult to distinguish the search direction. The minimum value 0 is obtained at $(x_1, x_2, \ldots, x_n) = (1, 1, \ldots, 1)$. The function dimension D is set to 4. The seeking optimization result is obtained and shown in Figure 1(e).

The seeking optimization of five functions is simulated 50 times. The simulation results are summed up in Table 1, where TIME is the average convergence time, BEST is the average optimum value, and STD is standard deviation of the optimal values in Figures 1(a)–1(e).

The Shubert function is used as an example. From Figure 1(b) and Table 1, the optimal value error obtained by CSO, QCSO, and CQCSO algorithm is 17.31%, 10.99%, and 2.43%, respectively, compared with the true value −186.7309. the precision of CQCSO algorithm is highest. In the convergence time, the QCSO algorithm shortens 0.0301 seconds compared with the CSO algorithm; then the CQCSO algorithm shortens 0.002 seconds compared with the QCSO algorithm. Therefore, simulation results show that the CQCSO algorithm can not only effectively solve the "premature convergence" problem of the CSO algorithm, but also improve the optimal solution accuracy and shorten the convergence time.

3. CQCSO Algorithm Application in PV MPPT

3.1. MPPT Flowchart Based on CQCSO Algorithm. The *P-V* characteristic curve function for photovoltaic cell can be gotten as follows [2–9]:

$$P_L = V_L \times I_{sc} \times \left[1 - C_1 \exp\left(\frac{V_L}{C_2 \times V_{oc}}\right)\right], \quad (11)$$

where I_{sc}, V_{oc}, C_1, and C_2 are the manufacturer-given parameters of photovoltaic cells. The output powers P_L change while output voltages V_L are adjusted. In this paper, the output voltages V_L are proportional to the duty cycles which are output by controller.

The *P-V* characteristic curve's fitness function (12) for photovoltaic array can be gotten from (11).

$$P_{array} = N_s \times N_p \times P_L, \quad (12)$$

where N_s are the numbers of photovoltaic cell in series and N_p are the numbers of parallel photovoltaic cell strings.

While the cells in array are partially shaded, the photovoltaic array's *P-V* characteristic curve can be gotten as shown in Figure 2. The *P-V* curve shows the multipeak characteristic with measurement interference. It is one nonlinear function seeking optimization with multiple local extreme points. In order to track the maximum power point better, the recursive least squares method is used to previously filter in real time the *P-V* characteristic curve under the local shading condition before searching optimization.

PV MPPT control strategy flowchart based on CQCSO algorithm is shown in Figure 3. The position of each cat X_i is defined as the array output voltage value U_i, and the fitness value is the array output power value.

3.2. Simulation Results. In this paper, the MPPT control system is consisting of PV array, Boost circuit, and MPPT

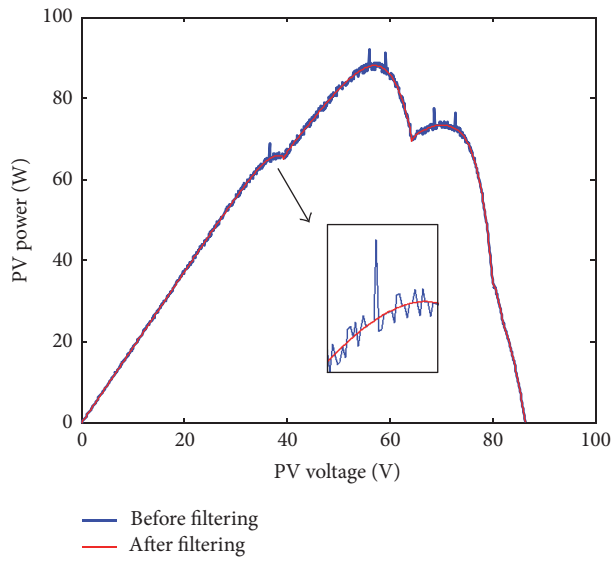


Figure 2: *P-V* characteristics of PV array under complex application environments.

Table 2: Parameters of PV modules.

Maximum power value	10 W ±3%	Pressure value of system	1000 VDC
Maximum power voltage	17.5 V	Maximum power current	0.57 A
Open circuit voltage	21.6 V	Short circuit current	0.62 A

controller. PV array is composed of 3 × 4 PV cells. The array structure is shown in Figure 4(a). The detailed parameters of PV modules are given in Table 2. In Figure 4(a), the light irradiation of 1C and 1D PV cell is 700 W/m^2, the light irradiation of the 2D PV cell is 100 W/m^2, the light irradiation of the rest of the PV cells is 1000 W/m^2, and the reference temperature of all PV cells is 25°C. The Boost circuits for MPPT are shown as Figure 4(b).

The total number of cat groups is set to 20. The maximum number of iterations is 100. The PV global maximum power point tracking *P-N* (maximum power value power-number of iterations) curve and the STD-*N* (standard deviation-number of iterations) curve are obtained as shown in Figure 5. The simulation results show that the CQCSO algorithm can be applied to the maximum power point tracking of PV multipeak curve, and the global maximum power point is the best among the three algorithms (CSO, QCSO, and CQCSO).

Tent chaos is introduced into PSO algorithm to obtain CPSO algorithm [21]. CPSO algorithm is applied to maximum power point tracking in solar photovoltaic system. We compare the performances of the CSO, QCSO, CQCSO, PSO, and CPSO algorithms as shown in Table 3. The total number of cat groups is set to 10. The packet rate MR is 0.2. The maximum number of iterations is 100. The mean convergence time (Time), power minimum (Min), power maximum (Max), power mean (Mean), and standard deviation (STD) of 100 runs are summarized.

From Table 3, it is not difficult to find that CSO algorithm has an average convergence time of 0.00291 s, the average value obtained is 87.72 W, while the average run time of the PSO algorithm is 0.00431 s, and the average optimal value is 87.66 W. CSO and PSO algorithm are of premature convergence, and the convergence time is longer. These would lead to a significant reduction in the efficiency of photovoltaic power generation.

CPSO algorithm can effectively solve the PSO algorithm premature convergence problem, and the convergence time becomes shorter. The convergence time of QCSO algorithm is shortened from 0.00291 s to 0.00282 s compared with the convergence time of CSO algorithm, and the average power value increased from 87.72 W to 87.89 W. The CQCSO algorithm presented in this paper is the maximum power average of 88.02 using the shortest convergence time of 0.00268 s and the minimum STD value. Therefore, the simulation results show that the proposed CQCSO algorithm has high efficiency in the photovoltaic MPPT.

3.3. Experiment Results. In order to verify the validity of the proposed CQCSO algorithm in the MPPT control and that the CQCSO algorithm is more efficient than QCSO algorithm and CSO algorithm, the small-scale MPPT control system based on the K60 microcontroller is built and shown in Figure 6, which is established in accordance with the configuration shown in Figure 4.

The MPPT controller adopts MK60-DN512VLL10 chip, the voltage and current are real-timely sampled by Hall sensor and then are sent to the K60 microcontroller, and the data will be calculated to get the right duty cycle. The parameters of the Boost circuit are set as follows: L = 4 mH, C_2 = 470 uF, f_2 = 30 kHz, and the load is a resistor. The duty cycle is used to control the Boost circuit through K60 computing for switch tube to adjust the PV array output voltage. The duty cycles are proportional to the output voltages. The PV powers are

Table 3: Result of experimental data.

Algorithm	CSO	QCSO	CQCSO	PSO	CPSO
Time/s	0.00291	0.00282	0.00268	0.00431	0.00402
Min/W	79.02	87.38	87.46	73.21	83.1
Max/W	87.95	87.95	88.06	87.94	87.94
Mean/W	87.72	87.89	88.02	86.91	87.66
STD	1.036	0.08009	0.07054	3.03	0.7371

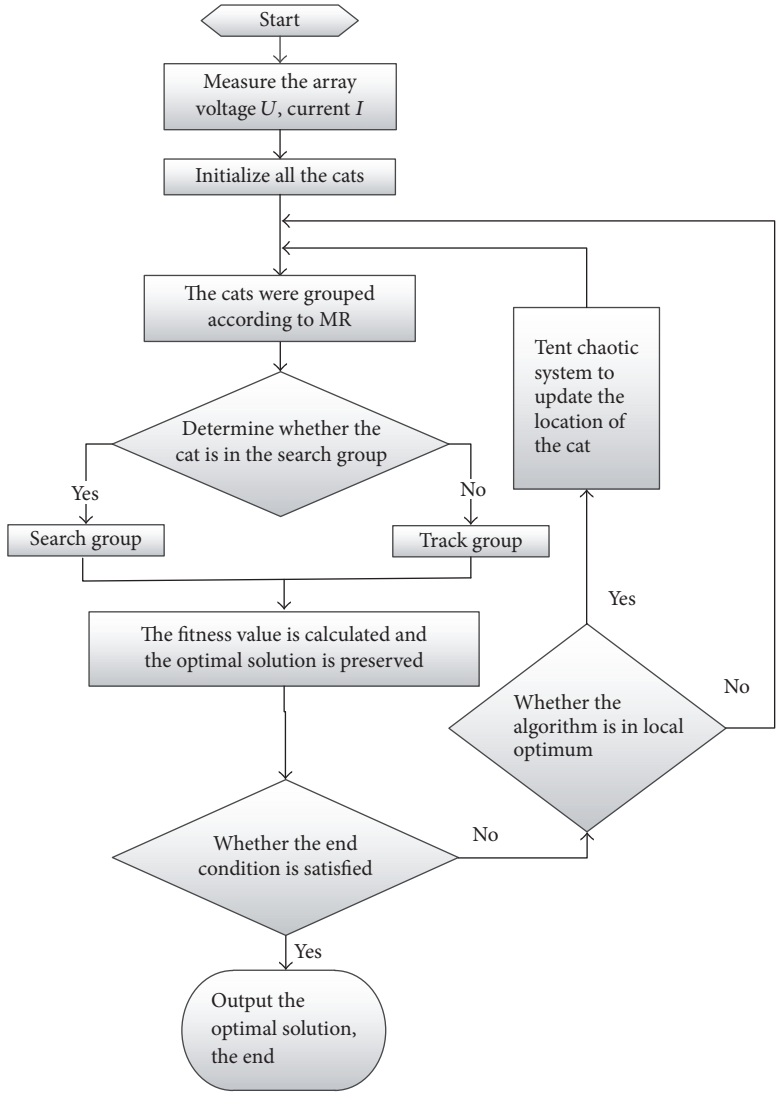
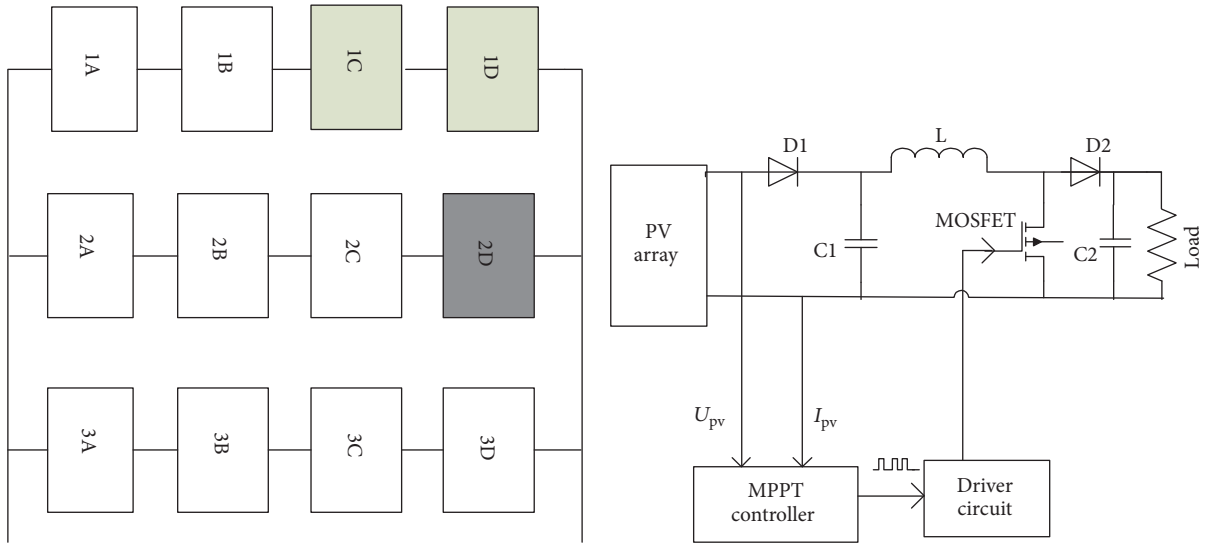


Figure 3: PV MPPT flowchart based on CQCSO algorithm.

changed as the output voltages are adjusted. A large number of experiments show that the duty cycle of the maximum power point is in the range of 15% to 85%.

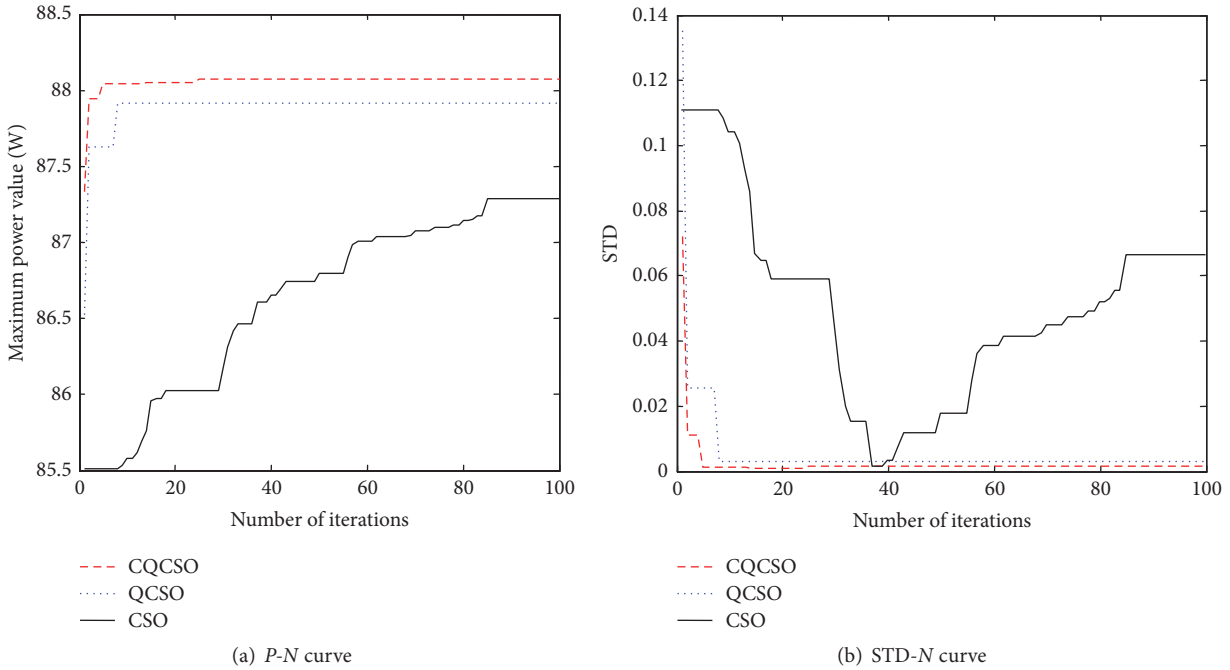
Experimental data is shown in Figure 7. The experimental time is from 6:00 to 18:00, the PV data of power, voltage, and current were scanned and sampled in 1-minute intervals, and 360 static data curves over time were obtained. The time-voltage-power 3D curve is shown in Figure 7(a). The time-power dynamic curve is shown in Figure 7(b).

The parameters of the CSO, QCSO, and CQCSO algorithm are as follows: the total number of cats is 10, the mixture ratio (MR) is 0.2, the size of seeking memory pool



(a) PV array (b) The Boost circuit for MPPT

Figure 4: Configuration of PV array and circuit for MPPT.



(a) *P-N* curve (b) STD-*N* curve

Figure 5: PV MPPT simulation curve.

(SMP) in the search group is 3, and the number of tracking loops in the tracking group is 3. Three kinds of algorithms are used to track the 360 data curves, and 360 dynamic maximum power points with time change are obtained as shown in Figure 8. It can be seen that the three algorithms can be very good in tracking the maximum power point in accordance with the environment changes. Between 13:00 and 14:00 on the test day, due to the shelter of the clouds, light radiation decreased and the maximum power values tracked were reduced. Comparing Figures 8(a), 8(b), and 8(c), we find that the QCSO algorithm is the best, the CQCSO algorithm is the second best, and the problem of "premature convergence" in CSO algorithm is serious.

In Figure 8, the *P-V* curves are single-peak state under uniform illumination, while the previous simulation and function tests are performed in the multipeak state. Therefore, we simulate complex conditions for the PV system by artificially shading the PV cells. The parameters of the QCSO

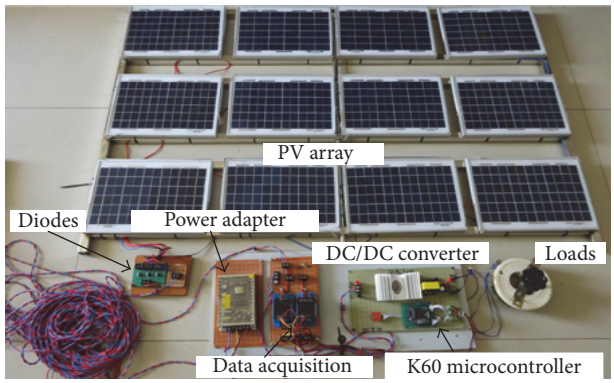
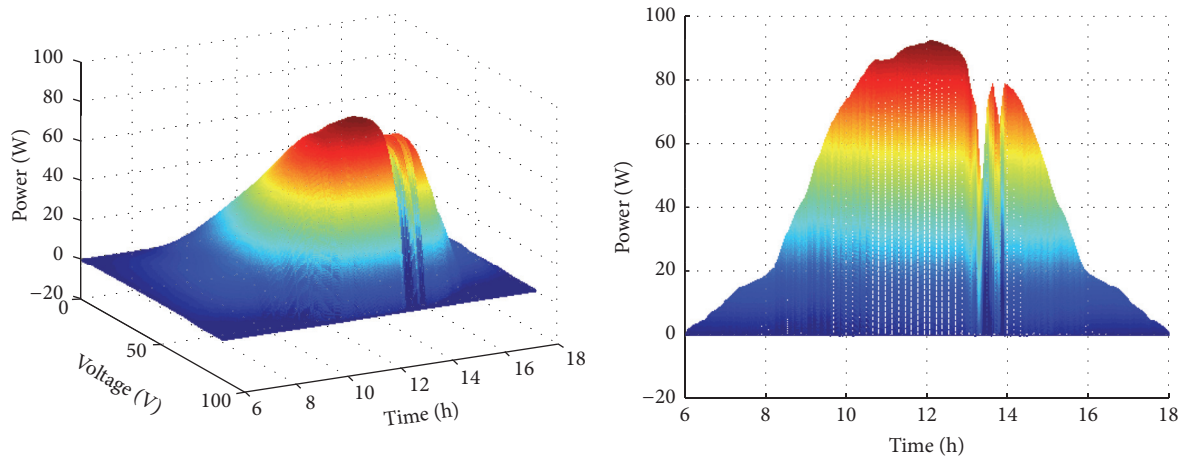


Figure 6: Experimental test platform.



(a) Time-voltage-power 3D curve

(b) Time-power dynamic curve

Figure 7: Experimental data.

and CQCSO algorithms are consistent with Figure 8. The experimental test time is 5 minutes. The result is shown in Figure 9.

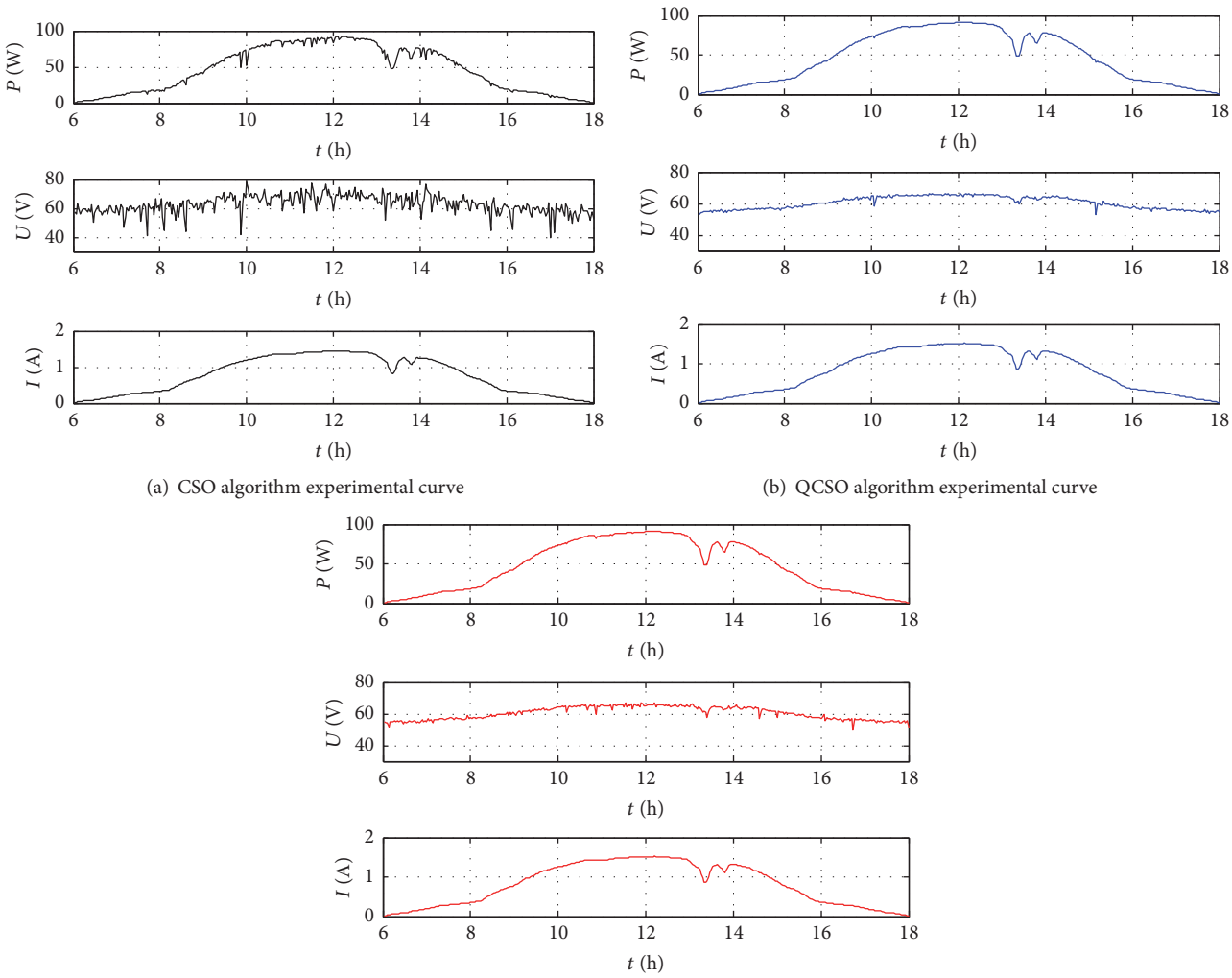
The experiment results show that the QCSO algorithm has obvious advantages over the CSO algorithm both in the case of a single peak and in the case of multiple peaks. The CQCSO algorithm solves the problem of "premature convergence" in CSO algorithm, but if the PV array is in the uniform illumination condition, the *P-V* curve shows a single-peak state. If the chaos is introduced at this time, although the complexity of the algorithm is increasing, the tracking accuracy is not improved. The QCSO algorithm appears to be better than the CQCSO algorithm, as shown in Figure 8. However, the *P-V* curve is in a multipeak state; it can be seen from Figure 9 which average maximum power value of the CQCSO algorithm is better than the QCSO algorithm in 5 minutes. Therefore, it is concluded that the CQCSO algorithm is more accurate than the QCSO algorithm in the complex case; that is, the QCSO algorithm improved by the tent map has more advantages in solving the multiextremum problem.

4. Conclusion

In multilocal extremum optimization, the traditional Cat Swarm Optimization (CSO) algorithm has problems such as "precocity convergence," slow tracking speed, and poor tracking accuracy. In this paper, we can get the following through the simulation and the experiment.

(1) Firstly, the quantum Cat Swarm Optimization (QCSO) algorithm is proposed. Secondly, Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm is proposed by introducing tent map. Finally, the CQCSO algorithm is verified by five nonlinear test functions. The simulation results show that the CQCSO algorithm can jump out of the local extreme points and improve the tracking precision and convergence speed.

(2) The CQCSO algorithm is applied to the multipeak maximum power point tracking for photovoltaic array under complex conditions. Both the simulation and experiment results show that the proposed CQCSO algorithm has higher tracking efficiency than the QCSO, CSO, PSO, and CPSO algorithm. In the maximum power point tracking system for



(a) CSO algorithm experimental curve

(b) QCSO algorithm experimental curve

(c) CQCSO algorithm experimental curve

Figure 8: The experiment curve.

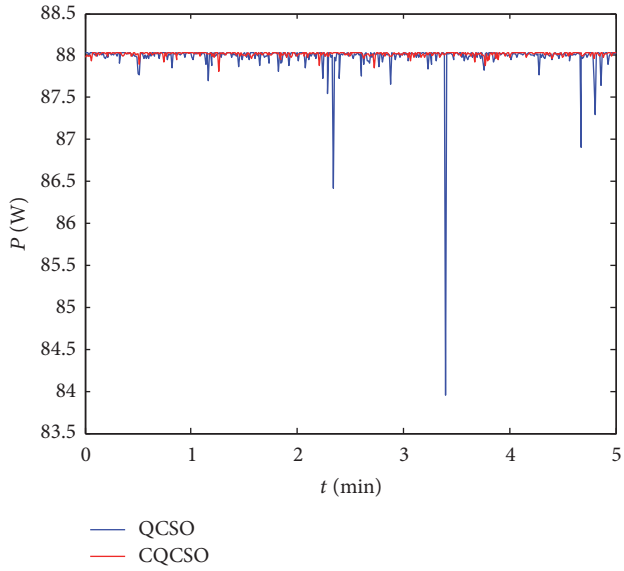


Figure 9: The experiment curve in a complex situation.

photovoltaic power generation, it is certain that it will obtain a larger power value in a shorter period of time, to improve the photovoltaic power generation efficiency.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation of China (no. 51467013).

References

[1] A. M. Latham, R. Pilawa-Podgurski, K. M. Odame, and C. R. Sullivan, "Analysis and optimization of maximum power point tracking algorithms in the presence of noise," *IEEE Transactions on Power Electronics*, vol. 28, no. 7, pp. 3479–3494, 2013.

[2] A. R. Jordehi, "Maximum power point tracking in photovoltaic (PV) systems: a review of different approaches," *Renewable & Sustainable Energy Reviews*, vol. 65, pp. 1127–1138, 2016.

[3] M. Miyatake, M. Veerachary, F. Toriumi, N. Fujii, and H. Ko, "Maximum power point tracking of multiple photovoltaic arrays: a PSO approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 367–380, 2011.

[4] Z. Salam, J. Ahmed, and B. S. Merugu, "The application of soft computing methods for MPPT of PV system: a technological and status review," *Applied Energy*, vol. 107, pp. 135–148, 2013.

[5] K. Ishaque and Z. Salam, "A review of maximum power point tracking techniques of PV system for uniform insolation and partial shading condition," *Renewable & Sustainable Energy Reviews*, vol. 19, pp. 475–488, 2013.

[6] M. A. Eltawil and Z. Zhao, "MPPT techniques for photovoltaic applications," *Renewable & Sustainable Energy Reviews*, vol. 25, pp. 793–813, 2013.

[7] M. A. G. de Brito, L. Galotto, L. P. Sampaio, G. de Azevedo Melo, and C. A. Canesin, "Evaluation of the main MPPT techniques for photovoltaic applications," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 3, pp. 1156–1167, 2013.

[8] B. Subudhi and R. Pradhan, "A comparative study on maximum power point tracking techniques for photovoltaic power systems," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 89–98, 2013.

[9] A. Reza Reisi, M. Hassan Moradi, and S. Jamasb, "Classification and comparison of maximum power point tracking techniques for photovoltaic system: a review," *Renewable & Sustainable Energy Reviews*, vol. 19, pp. 433–443, 2013.

[10] S. C. Chu, P. W. Tsai, and J. S. Pan, "Cat swarm optimizatio," in *Pacific Rim International Conference on Artificial Intelligence*, pp. 854–858, Springer, Berlin, Germany, 2006.

[11] G. Naveen Kumar and M. Surya Kalavathi, "Cat Swarm Optimization for optimal placement of multiple UPFC's in voltage stability enhancement under contingency," *International Journal of Electrical Power & Energy Systems*, vol. 57, pp. 97–104, 2014.

[12] L. Pappula and D. Ghosh, "Linear antenna array synthesis using cat swarm optimization," *International Journal of Electronics and Communications*, vol. 68, no. 6, pp. 540–549, 2014.

[13] F. Yang, M. Ding, X. Zhang, W. Hou, and C. Zhong, "Non-rigid multi-modal medical image registration by combining L-BFGS-B with cat swarm optimization," *Information Sciences*, vol. 316, pp. 440–456, 2015.

[14] Z. Wang, C. Chang, and M. Li, "Optimizing least-significant-bit substitution using cat swarm optimization strategy," *Information Sciences*, vol. 192, pp. 98–108, 2012.

[15] P. M. Pradhan and G. Panda, "Solving multiobjective problems using cat swarm optimization," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2956–2964, 2012.

[16] S. K. Saha, S. P. Ghoshal, R. Kar, and D. Mandal, "Cat Swarm Optimization algorithm for optimal linear phase FIR filter design," *ISA Transactions*, vol. 52, pp. 781–794, 2013.

[17] P. Tsai, J. Pan, S. Chen, and B. Liao, "Enhanced parallel cat swarm optimization based on the Taguchi method," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6309–6319, 2012.

[18] L. Guo, Z. Meng, Y. Sun, and L. Wang, "Parameter identification and sensitivity analysis of solar cell models with cat swarm optimization algorithm," *Energy Conversion and Management*, vol. 108, pp. 520–528, 2016.

[19] O. E. Turgut, M. S. Turgut, and M. Turhan Coban, "Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system of equations," *Computers and Mathematics with Applications*, vol. 68, no. 4, pp. 508–530, 2014.

[20] O. E. Turgut, "Hybrid chaotic quantum behaved particle swarm optimization algorithm for thermal design of plate fin heat exchangers," *Applied Mathematical Modelling*, vol. 40, no. 1, pp. 50–69, 2016.

[21] Y.-Y. Hong, A. A. Beltran, and A. C. Paglinawan, "A chaos-enhanced particle swarm optimization with adaptive parameters and its application in maximum power point tracking," *Mathematical Problems in Engineering*, vol. 2016, Article ID 6519678, 19 pages, 2016.

[22] J. Wang, B. Zhou, and S. Zhou, "An improved cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 2959370, 8 pages, 2016.

[23] N. Dong, X. Fang, and A.-g. Wu, "A novel chaotic particle swarm optimization algorithm for parking space guidance," *Mathematical Problems in Engineering*, vol. 2016, Article ID 5126808, 14 pages, 2016.

[24] S. S. Gokhale and V. S. Kale, "An application of a tent map initiated Chaotic Firefly algorithm for optimal overcurrent relay coordination," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 336–342, 2016.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 4216281, 22 pages
https://doi.org/10.1155/2017/4216281

Research Article

Nonintrusive Load Monitoring Based on Advanced Deep Learning and Novel Signature

Jihyun Kim,[1] Thi-Thu-Huong Le,[2] and Howon Kim[2]

[1]*IoT Research Center, PNU, Busan, Republic of Korea*
[2]*Pusan National University, Busan, Republic of Korea*

Correspondence should be addressed to Jihyun Kim; kjhps000@gmail.com

Received 5 May 2017; Revised 2 August 2017; Accepted 21 August 2017; Published 2 October 2017

Academic Editor: Nikolaos Doulamis

Monitoring electricity consumption in the home is an important way to help reduce energy usage. Nonintrusive Load Monitoring (NILM) is existing technique which helps us monitor electricity consumption effectively and costly. NILM is a promising approach to obtain estimates of the electrical power consumption of individual appliances from aggregate measurements of voltage and/or current in the distribution system. Among the previous studies, Hidden Markov Model (HMM) based models have been studied very much. However, increasing appliances, multistate of appliances, and similar power consumption of appliances are three big issues in NILM recently. In this paper, we address these problems through providing our contributions as follows. First, we proposed state-of-the-art energy disaggregation based on Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) model and additional advanced deep learning. Second, we proposed a novel signature to improve classification performance of the proposed model in multistate appliance case. We applied the proposed model on two datasets such as UK-DALE and REDD. Via our experimental results, we have confirmed that our model outperforms the advanced model. Thus, we show that our combination between advanced deep learning and novel signature can be a robust solution to overcome NILM's issues and improve the performance of load identification.

1. Introduction

Demand for energy is growing rapidly worldwide, and the demand for electric energy is growing even more rapidly [1]. Electric energy demand will be expected to double between 2010 and 2050 [2]. However, as the global population continues to grow globally and fossil fuels are becoming increasingly depleted, current electricity production methods are not sustainable. The International Electrotechnical Commission (IEC) has stated that the intelligent and economic use of electricity, as the primary energy source, will be the most important factor in solving energy problems [3]. As a result, research is being conducted on ways to efficiently utilize energy in factories, buildings, and homes.

In the field of factories, researches are being carried out on Factory Energy Management Systems (FEMS) for efficient electric energy use. Recently, it has been linked to the Cyber Physical Systems (CPS) of Industry 4.0, and related research will be more active in this area [4]. In the field of buildings, research is underway on Building Energy Management System (BEMS) to reduce unnecessary energy consumption [5]. Depending on the external environment, air conditioners can be appropriately controlled to reduce the electric energy use. Unnecessary energy use can also be reduced through monitoring. In homes, the smart grid will be able to efficiently monitor and manage energy as it becomes increasingly realistic [6].

One approach to increasing the efficiency of domestic electricity use is to inspire positive behavioral change in consumers [7]. This can be achieved by analyzing energy use. Nonintrusive Load Monitoring (NILM) is one alternative to do so. NILM is a process for analyzing changes in the voltage and current going into a house and deducing which appliances are being used in the house along with their individual energy consumption. The initial concept of disaggregating residential power load information was proposed by Hart [8]. He demonstrated how different electrical appliances generate distinct power signatures, including their active power,

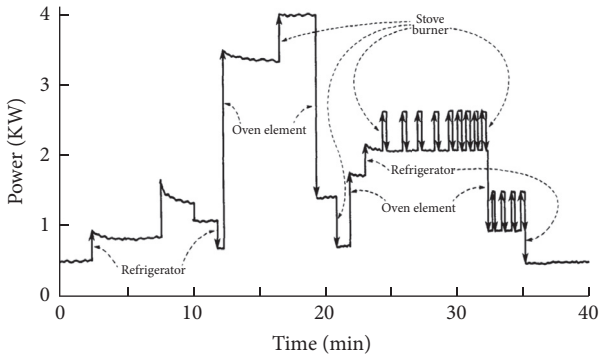


Figure 1: Basic concept of NILM [8].

current, and voltage (Figure 1). He showed how on-off events were sufficient to characterize the use of some appliances. Many researchers have studied this concept and improved the NILM model.

In order to disaggregate a power consumption based on NILM, we have to understand features of appliances. Hart defined three types of appliance model according to the features. Three types are *On/Off*, *Finite State Machine (FSM)*, and *Continuously Variable* [8, 9].

The first type *On/Off* has binary states of power such as a light and a toaster. Figure 2(a) shows the power pattern of light. When it is turned on, the power level is increased from 10 to 12 watts. The appliances of type 1 can be easily classified because of a clear feature. However, if two appliances which have similar power consumption are operating, they could not be distinguished. In this case, a classification model does not know which appliance is generating the current power. Therefore, we need an additional factor to distinguish them.

The second type *FSM* has multiple states of power such as a lamp and a fan. The power pattern of lamp is shown in Figure 2(b). Three operation states are in lamp according to brightness. This kind of appliances can be modeled by FSM. So it could be simple to classify a single appliance of type 2. However, when the multiple power consumption is summed, the classification is become complicated, because it is hard to figure out that the current power is originated in the variable pattern. Therefore, we need to observe the pattern for a period of time in order to classify the appliances.

The third type has a continuous changeable power pattern such as a washing machine and a light dimmer. In Figure 2(c), the power consumption is fluctuated while heating/washing or rinsing/drying a laundry. Strictly speaking, it is a multiple state appliance but it cannot be modeled due to infinite middle states. To classify type 3 appliances, we have to understand the feature and observe the long-range pattern.

Lastly, we define an additional type *Always On*. The appliances of type 4 are always operating except a special case. For example, a refrigerator in Figure 2(d) is operating consistently. The type 4 appliances could have a periodic pattern or a single pattern. According to circumstances, we can reduce the number of appliances when we train the classification model.

The main contributions in this work consist of two points. First one is *a construction state-of-the-art NILM model*. The proposed model is robust against an increase of appliance due to the reduced time complexity. Although we use the low sample rate data, the multistate appliances and the similar power appliances can be classified efficiently. As learning a long-range power pattern result, we can solve the previous problems, which are mentioned in Section 3. We show that the proposed model outperforms via the experimental results with UK-DALE and REDD. Second one is *a discovery of the novel signature*. The proposed signature raises the performance of classification for the multistate appliance. By emphasizing the power variation when we train the model, the variation could be clearly trained. We show the efficiency via the validation experiments in Section 5.3.

This paper is organized the structure as follows. We give problem statement from related works and proposed solutions in Section 2. Section 3 describes two NILM datasets to be used in our experiments. In Section 5, we conduct two main experiments. The first experiment is about validating the expected effects of the novel signature and learning architecture. The second experiment is about measuring the overall performance and comparing with the existing models and state of the art. Final section, we summarize our works and provide the conclusion.

2. Related Work

2.1. Signature Based Approach. There are two different kinds of signature. The first signature is the steady state. It makes use of steady-state features that are derived under the steady-state operation of the appliances. Hart proposed the concept of NILM and power change method at the same time in his paper [8]. In this method, real power (P) and reactive power (Q) are used as the input signature. By computing changes of P and Q, the appliances are classified. Hart demonstrated how different electrical appliances generated distinct power consumption signatures. He showed how on/off events were sufficient to characterize the use of some appliances. The advantage of this method is that we can use a low sampling dataset and it is easy to identify the appliances having high power consumption. However, it is hard to classify the appliances having low power consumption and the multistate appliances which are types 2 to 4 in Figure 2. In addition to that, the appliances having similar power consumption cannot be classified. To improve NILM, other researches have attempted and proposed alternative signature identification techniques. Najmeddine et al. and Figueiredo et al. used the current I and the voltage V as the signature [11, 12]. They extracted the features such as Root Mean Square (RMS) of the current and peak. These features are well suited for classifying the appliances in the kitchen. However, the multistate appliances cannot be classified as ever. One of the signatures extracted from high sampling rate data is the harmonic of current. There are several researches acquiring the harmonics via Fourier series [13–16]. The appliances of types 1 and 4 are well classified by current harmonics. But it requires the harmonics set of all combinations of the appliances. As increasing the number of appliances, the harmonics for classification increase exponentially. This is not a practical approach and the memory problem could occur. There are

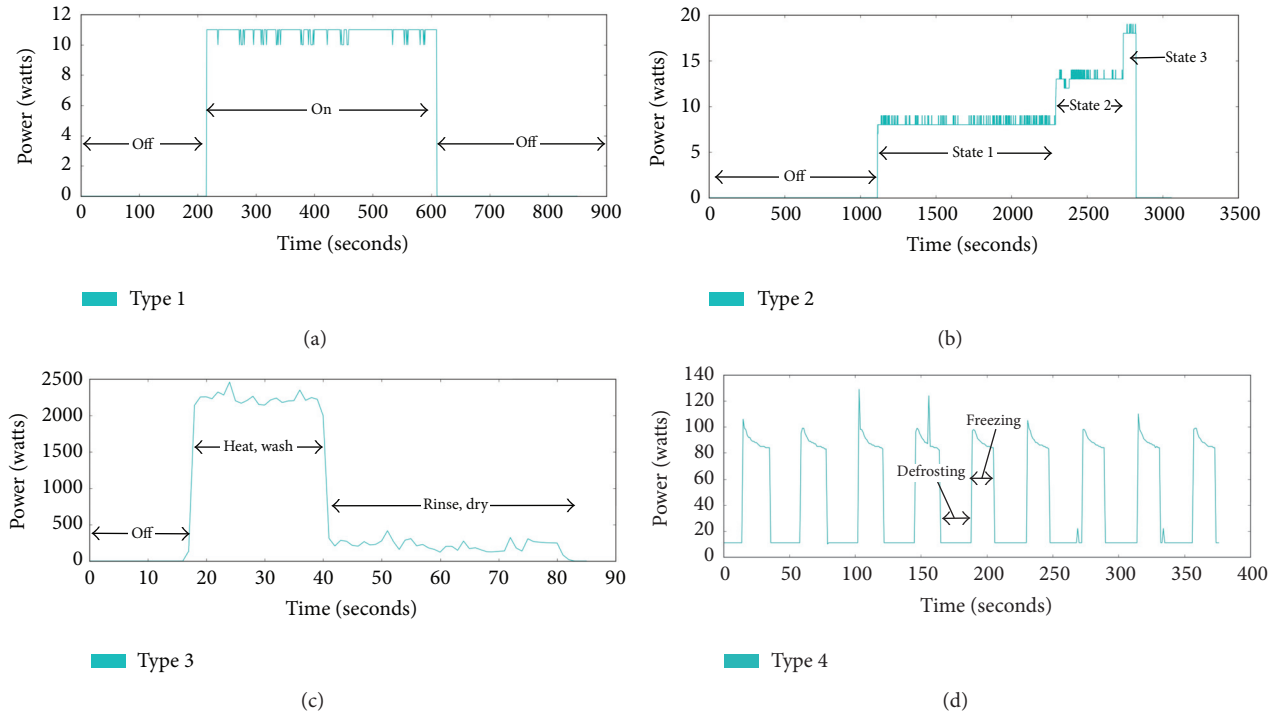


Figure 2: Four types of appliances. (a) Type 1 (On/Off): light; (b) type 2 (FSM): lamp; (c) type 3 (Continuously Variable): washing machine; (d) type 4 (Always On): refrigerator.

fancy approaches for NILM research. Lam et al. proposed the appliance categorization method based on the shape of $V - I$ trajectory [17]. They categorized the appliances to eight groups with high accuracy. Gupta et al. tried to identify the appliances based on the noise generated from the operation of appliance [18]. However, the noise is easy to be affected by the environment. So it cannot be high efficient signature. In addition, we need to equip the device for measuring the noise. Therefore, it cannot be practical method.

The second signature is the transient state. This signature is less overlapping in comparison with steady-state signatures. However, high sampling rate requirement in order to capture the transients is the major disadvantage. Chang et al. showed that the power consumption while an appliance is turning on can be calculated as a signature [19]. Five years later, they identified the transient physical behavior of power by using wavelet transform [20]. Another way for classification, Leeb et al. proposed the method analyzing the spectral envelope via Short Time Fourier Transform (STFT) [21]. In this way, the method of using transient power as a signature is suitable for classification of types 1 to 3. But it cannot catch type 4. Norford and Leeb showed that the shape of transient data can be a feature [22]. Cole and Albicki used power spikes generated from transition states [23]. This signature is efficient for classification but it is applied to several specific appliances. Like this, the method of using the current transient when the appliances are turning on is suitable for classification of types 1 and 2. But it cannot catch types 3 and 4. Patel et al. sampled the voltage noise extracted from the transient event [24]. They defined three types of noise: on/off transient noise, steady-state line voltage noise, and steady-state continuous noise. However, to use this method, we need to have knowledge about the power flow such as the reactive power, the active power, and phase of voltage relative to current. The method using the voltage noise is suitable for classification of the multistate appliances.

2.2. Learning Model Based Approach. Some useful temporal graphical models, the variants HMM, are used for NILM. Zoha et al. proposed a solution using FHMM to recognize appliance specific load patterns from the aggregated power measurements [25]. They defined five features that are combinations of the power measurement such as the real power, reactive power, and voltage waveforms. They achieved the f-measure of 0.614 for five appliances when they use multistate FHMM. Kolter and Johnson also used FHMM for NILM [26]. They had collected their own dataset called REDD which will be explained in the Section 4. The average accuracy was 47.7%. To improve the researches using FHMM, a number of researchers have extended FHMM. Kim et al. developed probabilistic models of appliance behavior using variants of FHMM [27]. The variants are Factorial Hidden Semi-Markov Model (FHSMM), Conditional FHHH (CFHMM), and Conditional FHSMM (CFHSMM). Figure 3 shows the relationships between the variant FHMMs and their performance comparison.

Kolter and Jaakkola used a combination of additive FHMM and difference FHMM [28]. For inferencing, they proposed an Additive Fractional Approximate MAP (AFAMAP) algorithm. They achieved the recall of 0.603 for 7

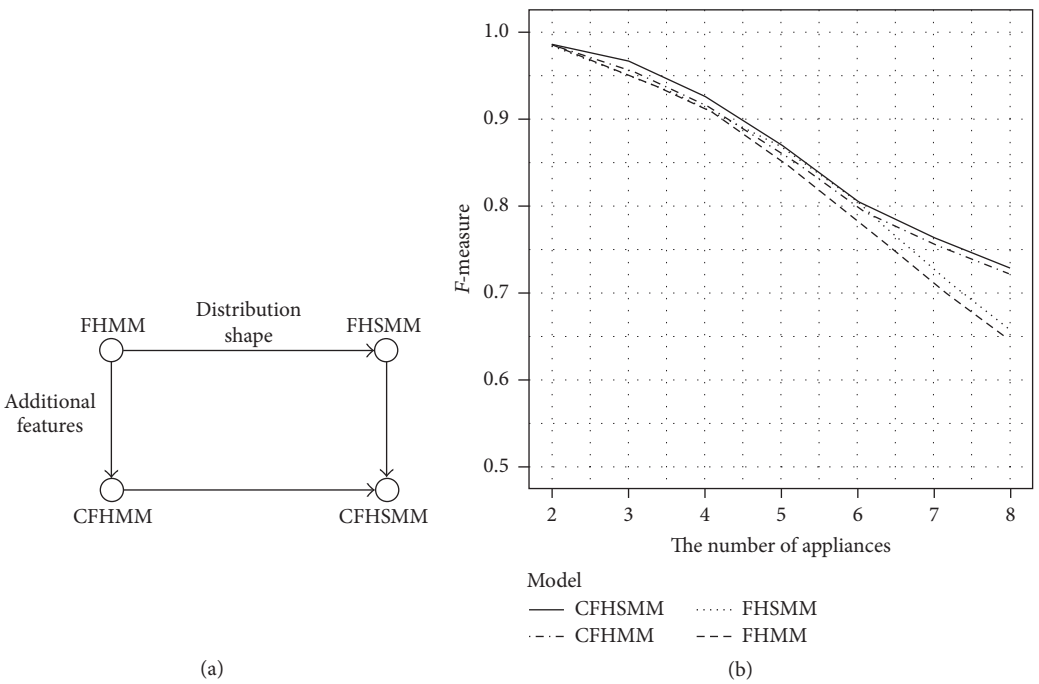


Figure 3: Variants of FHMM and performance results [10]. (a) Relationships between the variant FHMMs; (b) performance comparison.

appliances. Parson el al. proposed the variant of the difference HMM and the extended Viterbi algorithm [29]. Zeifman proposed a Viterbi algorithm with Sparse Transitions (VAST) [30]. And he used Markov chain (MC) for NILM.

3. Problem Statement and Proposed Solution

3.1. Problem Statement. The data for NILM is collected in a timed sequence. Therefore, the models dealing with a sequential data are used as an appliance classification model. The models which are usually used are Factorial Hidden Markov Model (FHMM) based on the Markov process and its variants [32]. However, these models have three problems to classify the appliances. The problems are as follows:

(i) The first problem is about the time complexity. In case of FHMM, the time complexity is $O(Mn^{M+1}T)$ where M is the number of appliances and n is the possible states of hidden unit and T is the length of observation sequence. As the number of appliances is increased, the time complexity is increased exponentially as well. This leads to reduction of the classification performance of the model.

(ii) The second problem is about the difficulty of classifying the multistate appliance. The multistate appliance is the appliance which has the multiple state of power consumption. To distinguish them, a long-range pattern should be trained. However, the existing models are based on the first-order Markov process. Therefore, the pattern could not be trained efficiently. In addition, a low sample rate data (e.g., a collected data by 1~6 Hz sample rate) is not proper for classifying the multistate appliances. If we use a high sample rate data for NILM, this problem might be solved due to catching a momentary change of power consumption. But there is a limitation for the real life because a high cost device is needed for collecting the high sample rate data.

(iii) The third problem is a difficulty of classifying the appliances having similar power consumption. If we use only power consumption as an observation of FHMM, the model could not distinguish the similar power appliances.

We surveyed the basic models and the related works of NILM so far. However, there are some problems from the original basics of NILM. In this section, we summarized them into three.

Problem 1. The researchers have used FHMM and the FHMM variants for NILM [25–27, 29]. As we know, the time complexity of FHMM is $O(Mn^{M+1}T)$ where M is the number of appliances and n is the possible states of hidden unit and T is the length of observation sequence. Therefore, as increasing the number of appliances, the time complexity also increases exponentially. As a result, the performance of classification model is dropped. We can easily notice that in Figure 3(b).

Problem 2. The second problem is that FHMM and the FHMM variants are hard to classify type 2 to 4 appliances. A long-term pattern is needed to learn but most models are based on the first-order Markov chain [35]. So these models predict the current state through the previous state. As a result, FHMM and the FHMM variants have a limitation for classifying the multistate appliances. In addition, there is another problem based on the signature perspective. In the

signature based approach, we can separate the low sampling rate signature and the high sampling rate signature. The advantage of first signature is that we can easily collect the data from a simple sensor. However, the low sampling rate signature was unsuitable for classifying types 2 to 4 in the most researches [8, 11, 12]. To solve this problem, we need more precise data and high sampling rate signature. By this signature, the researchers classified all types of appliances. However, a specific device should be equipped in order to get the signature. It means that the high cost is required for application of the real life. This is not a good way for popularizing NILM.

Problem 3. The last problem is the difficulty of classifying the appliances which have similar power consumption. In the previous models, the observation was only a series of power consumption. HSMM deals with a duration of operation but how can we know the durations for each appliance? The durations are not stable and it is a very tiresome business to extract the durations for each appliance. For example, in Figure 4, we extracted the patterns of air-conditioner from one house. We can notice that the duration is different. It depends on a user's behavior.

Through summarizing the problems, we concluded that the problems from FHMM and the FHMM variants are the fundamental issues from the architecture of FHMM. Therefore, we propose a deep learning based NILM model. Also, we propose the novel low sampling signature in order to apply to the real life. We have three challenges for successful application of deep learning.

Challenge 1. Our model must be robust even though the number of appliances is increased.

The significant problem of NILM models based on FHMM is that the performance is decreased as increasing the number of appliances. This will be the biggest obstacle for applying NILM to the real life. Therefore, the proposed model should be robust even though the number of appliances is increased.

Challenge 2. The multistate appliance should be classified.

The problem when we use a low sample rate data is that it is hard to classify the multistate appliances. This is caused by the fact that various patterns of appliance cannot be reflected efficiently to the main signal (total power consumption) due to a wide sampling gap. Although it is type 1 appliance, the main signal could be a multistate signal because of overlapping signal for each appliance. So we must classify the multistate appliances.

Challenge 3. Although there are the appliances having the similar power consumption, they should be classified.

If power consumption is the same between two appliances, they would be the same appliances. Except this case, the appliances having the similar power consumption can slightly be different. For the accurate performance, we need to distinguish the similar appliances.

3.2. The Proposed Solution

3.2.1. Proposed Novel Signature. Sequence data selected from real life might be power consumption sampled at 1 to 6 Hz. Because of the cost of a sensor, there is a limitation to collecting high sample rate data. To popularize NILM, we have to use low sample rate data from such a sensor. In Section 3, we acknowledge that it is hard to classify multistate appliances (type 2, type 3, and type 4) when using low sample rate data. To solve the problem, we propose a novel signature, which is a key input feature. The main idea of the signature is to separate the original signal, which is low sample rate data (power consumption), using a reflection rate and to subtract one variant power signal from the other variant power signal. We denote the difference as Δ_p which represents variation of the original signal. The purpose of Δ_p is to emphasize the variation of multistate appliances by applying the same importance with the original signal to a training model. The result is that performance of the model for an appliance with many variations may be better than when using only the original signal.

To calculate Δ_p, we need to generate the variant power signal. The purpose of using the variant power signal is to reduce noise. An unintended noise can occur and it can have a negative influence on learning the signal pattern. We can regulate noise with the reflection rate. The reflection rate ranges from 0 to 0.99. Power signals that increase or decrease gradually are reflected to the variant power signal, while noise, which occurs in a moment, is not reflected well. This lower reflection rate reduces the negative effect of momentary noise. However, if the reflection rate is too low, the original power pattern may not be reflected. Thus, it is vital to set the reflection rate properly. Through the experiment explained in Section 5.3.1, we empirically found that 0.1 or 0.01 is the proper reflection rate.

Algorithm 1 shows how to generate the variant power signal. There are two inputs for the algorithm: P is the series of the original signal with the time length T and α is the reflection rate, which is a ratio for reflecting variation between the original signal and the variant power signal. The output VP is the variant power signal generated by the algorithm. In line (1), the variant power signal is initialized to zero. The generation process is given in lines (2) to (5). After the variation at time i is calculated in line (3), it is reflected to the variant signal at reflection rate α in line (4). Figure 5 is an example of the variant power signal. Figure 5(a) is the series of the original signal with the time length 5000. Figure 5(b) is the variant power signal when the reflection rate is 0.1. Note that the pattern is smoother than the original. This indicates that momentary noise has been reduced while long-range variations remain. Figure 5(c) is the variant power signal when the reflection rate is 0.01. In this case, the original signal changes are more slowly reflected to the variant power signal. With two different variant power signals, we can calculate the variations in the original signal.

The algorithm for the novel signature is in Algorithm 2. Two variant power signals are the input and the novel signature representing the series of variation is the output. Δ_p is calculated by subtracting the inputs. In the scale of the

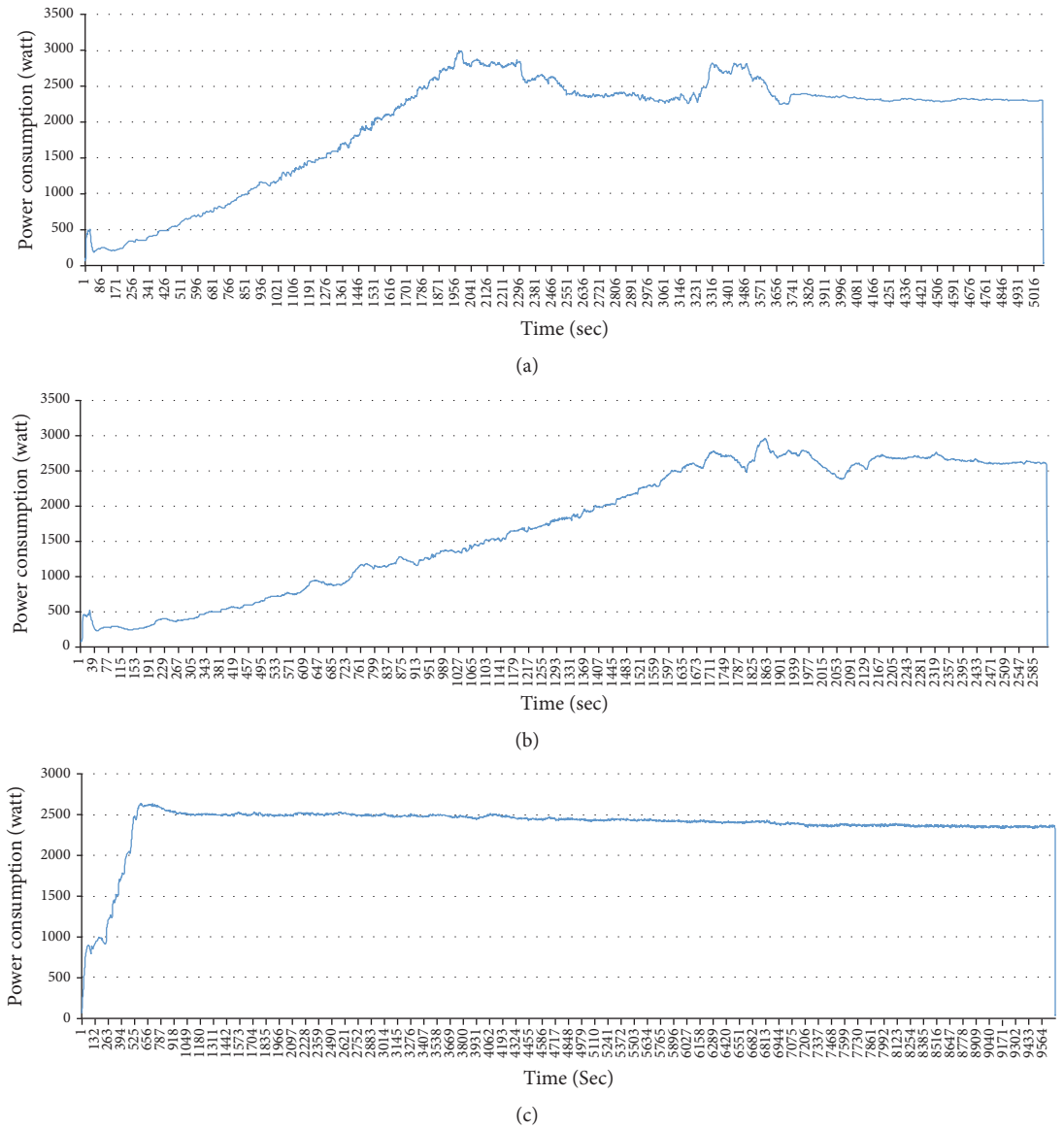


Figure 4: Different durations of air-conditioner: (a) duration: 5016 sec; (b) duration: 2585 sec; (c) duration: 9564 sec.

original signal, small variations are easily ignored. However, if we give Δ_p the same weight as the original signal, these small variations become more apparent than before. This effect of the novel signature will be validated in Section 5.3.2.

Figure 6 represents a computation of Δ_p. By Algorithm 1, the variant power signals (Figure 6(b)) are generated from the original signal (Figure 6(a)). We can see that the patterns of the variant power signal differ according to the reflection rates. The variant power signals in Figure 6(b) can be the inputs for Algorithm 2. Finally, the series of variations, Δ_p, is generated by Algorithm 2.

3.2.2. Learning Architecture. In this section, we introduce how to apply deep learning to NILM. We choose RNN as the learning algorithm. RNN can learn the sequential data such as the power consumption of household. And we apply LSTM to hidden layer in order to prohibit the vanishing gradient problem. We call this model LSTM-RNN. There are two reasons for applying deep learning to NILM. The first reason is that the time complexity was getting higher as the number of appliances was increased in the FHMM variants. The time complexity of FHMM is $O(Mn^{M+1}T)$ where M is the number of appliances and n is the possible states of hidden unit and T is the length of observation sequence. As we can see, the number of appliances has a close relation with the time complexity. As a result, the performance was dropped as the number of appliances was increased. In deep learning, the major time complexity is originated from backpropagation. Stochastic Gradient Decent (SGD) is usually used for backpropagation [36]. As a result, the time complexity of SGD is $O(md)$. In our model, the output dimension is the number of appliances. So the time complexity is not increased exponentially in contrast with FHMM. If we use the deep learning based NILM model, the model could be

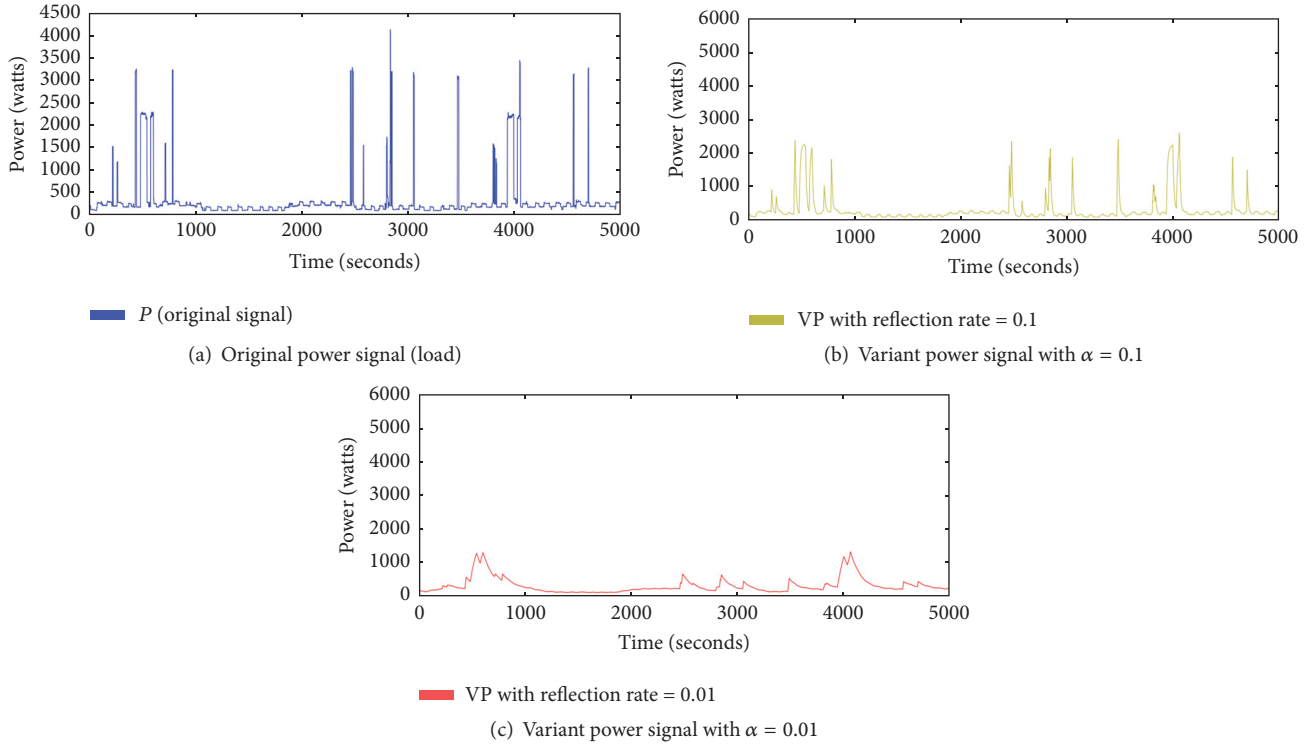


Figure 5: Patterns of the variant power signals.

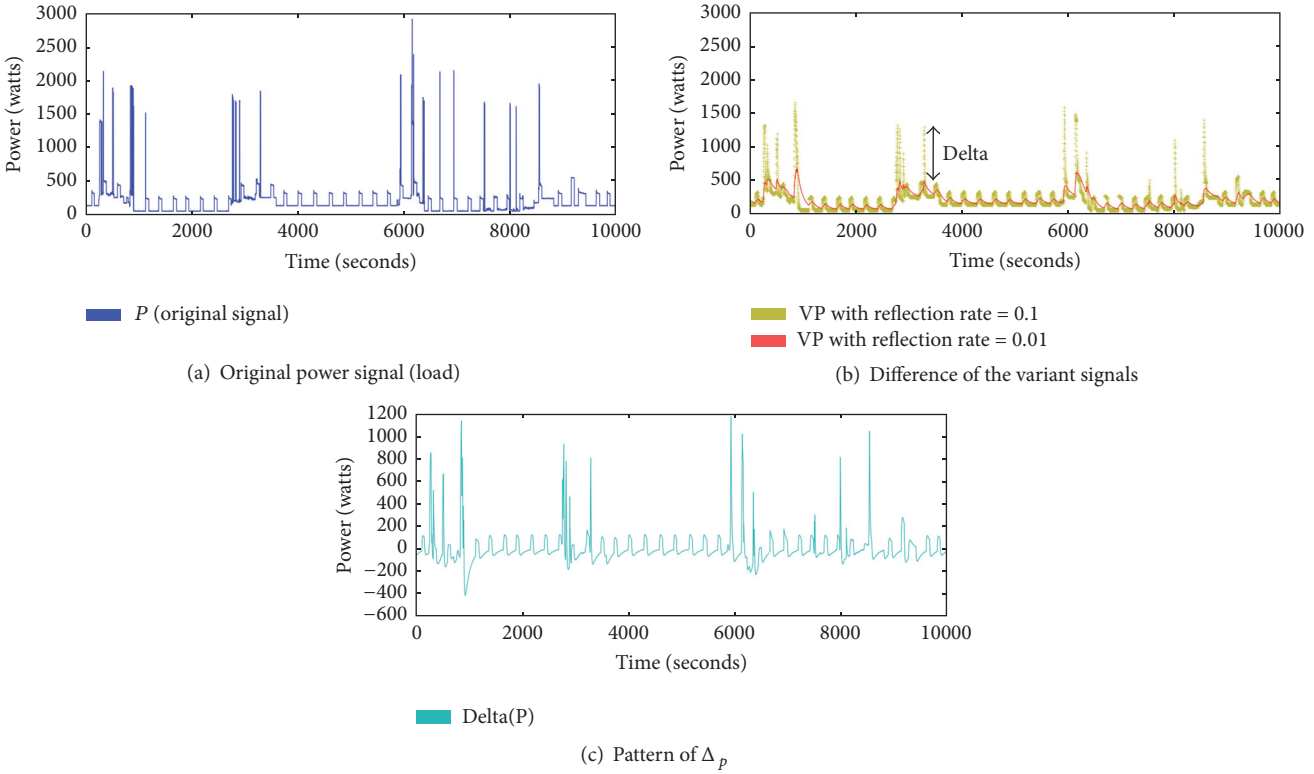


Figure 6: Computing Δ_p from the variant signals.

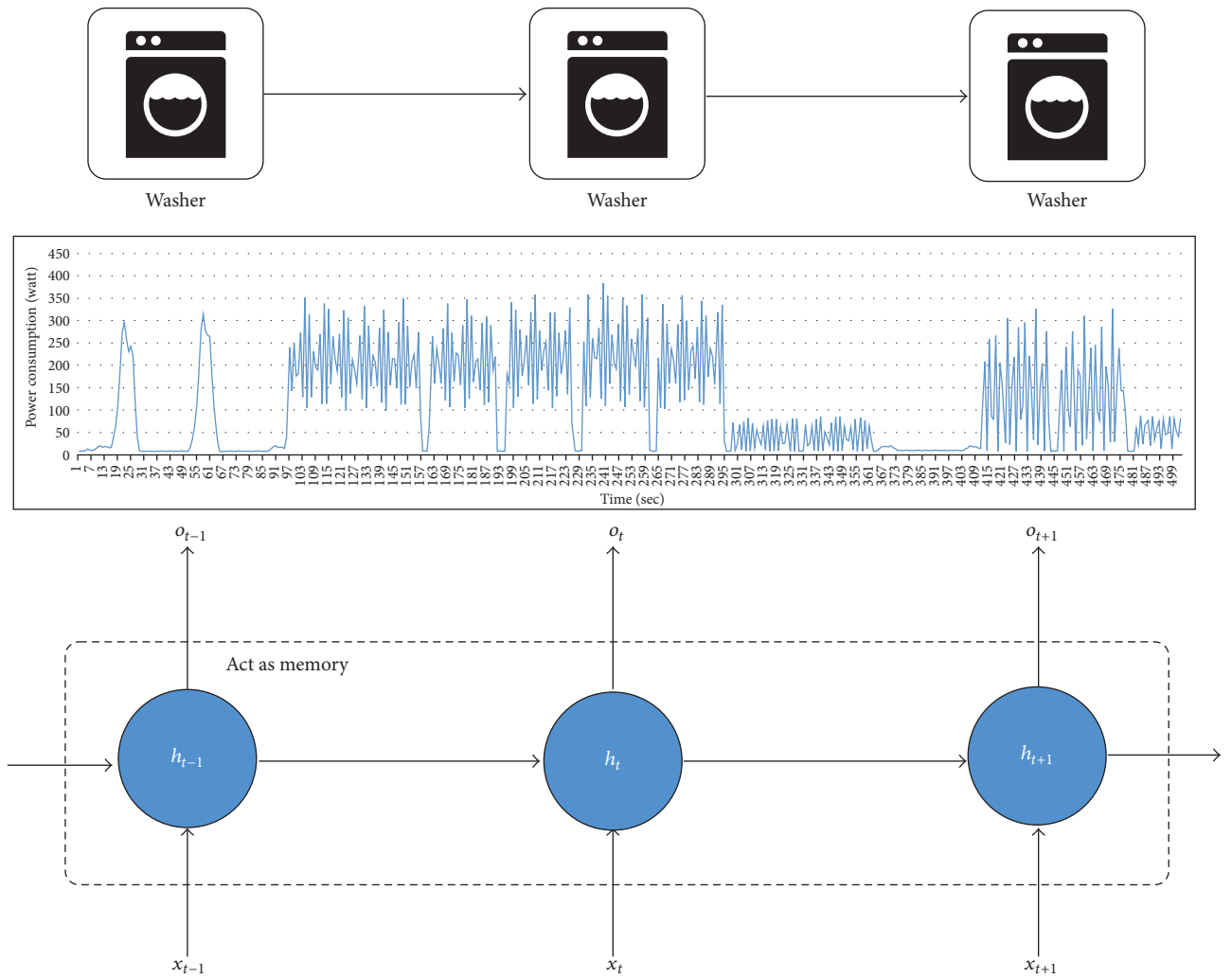


Figure 7: Long-range pattern learning.

Require: $P = p_1, p_2, \ldots, p_T$, Reflection rate α
Ensure: $\mathrm{VP} = \mathrm{vp}_1, \mathrm{vp}_2, \ldots, \mathrm{vp}_T$
(1) $\mathrm{VP} \leftarrow 0$
(2) **for** $i = 2$ to T **do**
(3) $\quad d \leftarrow p_i - \mathrm{vp}_{i-1}$
(4) $\quad \mathrm{vp}_i \leftarrow \mathrm{vp}_{i-1} + \alpha \cdot d$
(5) **end for**
(6) **return** VP

Algorithm 1: Generation algorithm for the variant power signal.

Require: $\mathrm{VP}^1 = \mathrm{vp}_1^1, \mathrm{vp}_2^1, \ldots, \mathrm{vp}_T^1$, $\mathrm{VP}^2 = \mathrm{vp}_1^2, \mathrm{vp}_2^2, \ldots, \mathrm{vp}_T^2$
Ensure: $\Delta_p = \delta_1, \delta_2, \ldots, \delta_T$
(1) **for** $i = 2$ to T **do**
(2) $\quad \delta_i \leftarrow \mathrm{vp}_i^1 - \mathrm{vp}_i^2$
(3) **end for**
(4) **return** Δ_p

Algorithm 2: Computing algorithm for the novel signature.

robust relatively. Additionally, deep learning uses the parallel computing via GPU. It means that even though the time complexity is high, GPU can reduce it. This is one of the major reasons why many researchers have been fascinated by deep learning. For the record, the latest GPU, Titan X with Pascal Architecture, has 3584 cores, 12 GB memory, and 11 GFLOP.

The second reason is that the long-term pattern cannot be learned in the FHMM variants (see Figure 7), because they are based on the first-order Markov chain. On the other hand, the hidden layer of LSTM-RNN is a kind of memory. A washer is the representative of the multistate appliance. Although the pattern is long and changeable, it could be learned when we set a length of LSTM-RNN similarly. This feature could classify the multistate appliances and the appliances having the similar power consumption.

We realized that the NILM model based on deep learning has a high possibility of achieving three challenges. One

important thing when using LSTM-RNN model is how to choose the suitable factors of this model. Here we discover four factors, which are considered much effective in our model.

Preprocessing Method. Generally, Z-Score method and Min–Max Scaling are considered as a normalization method for deep learning. In deep learning, a model has better performance when a distribution of input data is close to Gaussian distribution. Min–Max Scaling is the most simple way to normalize the input data. It converts the scale in range from 0 to 1 by using the minimum and maximum value. This method holds the original distribution but it cannot be the Gaussian distribution. Z-Score method normalizes the input data by using the mean and standard deviation. This method cannot hold the original distribution when the input distribution is not the Gaussian. Nevertheless, it is important that the mean of input data is close to zero, because if the input is all positive or negative, the weight could be updated in one way. In case of NILM, one of inputs, power consumption, is all positive value. Therefore, we use Z-Score as the preprocessing method. We are going to take an experiment about the preprocessing method in Section 5.4.1.

Weight Initialization Method. Asymmetry is the most important thing for the weight initialization. We can simply think that if the weights are initialized to zero, the update could be clear. But soon we realize that this method is wrong. There are only bias values in forward propagation. The simplest way to initialize the weights with asymmetry is selecting values from normal distribution or uniform distribution. However, this method has the problem that a variance of output increases with an input dimension. To solve this problem, the variance of output needs to be divided by the square root of input dimension. By this way, Glorot initialization and He initialization are commonly used for the weight initialization. The simple equations of them are as follows:

$$W_{\text{glorot}} = \frac{\text{random}\left(\text{fan}_{\text{in}}, \text{fan}_{\text{out}}\right)}{\sqrt{\text{fan}_{\text{in}}}},$$
$$W_{\text{he}} = \frac{\text{random}\left(\text{fan}_{\text{in}}, \text{fan}_{\text{out}}\right)}{\sqrt{\text{fan}_{\text{in}}/2}}, \quad (1)$$

where fan_{in} and fan_{out} are the number of input and output dimensions and *random* is a function of random selection in range from fan_{in} to fan_{out} from normal distribution or uniform distribution. Both equations are similar but He initialization selects the weight values in wide range relatively. In deep neural network like RNN, the weight can easily be shrunk to nothing when the values are close to zero. This leads to vanishing gradient problem. Therefore, we use He initialization due to the wide range value. A related experiment will be conducted in Section 5.4.2.

Activation Function. In general, the softmax is used as an activation function for multiclass classification [37]. It can select a one class among all classes. However, in NILM, many appliances could have been operated in the same time. It means that we need to classify the multiple classes. Therefore, we set the boundary with assuming that the appliance is turned on when the output unit value is bigger than the boundary. To do this, we change the softmax to the hyperbolic tangent (tanh) as the activation function. By this way, we can classify the multiple appliances at the same time. The sigmoid can also be the activation function. However, while the output of sigmoid is not zero-centered, the tanh has the zero-centered output. As a result, we could set the static boundary when we use tanh as the activation function.

Optimizer. There are three optimizers for SGD. One of them is Adagrad. It decreases a learning rate when a signal is frequent or the weight has a high gradient. In the opposite case, it increases the learning rate. However, this method has a low performance due to decreasing the learning rate fast. RMSProb solves this problem by an exponential moving average. Adam is a combination of RMSProb and Momentum. It is known as the most efficient method for SGD. Therefore we use Adam as an optimizer.

The inputs are the original signal (power consumption) and Δ_p. The output unit represents the on/off state of each appliance. By using the novel signature Δ_p, we can expect the better performance for classification of multistate appliances. We apply dropout between the input layer and the hidden layer in order to prevent overfitting. This is the simplest way to regularize the weights and is suitable for deep neural networks like an RNN. We set the number of hidden units to be double the number of output units. A small number of hidden units require more training epochs, whereas a larger number of hidden units require more computation time per epoch. We found that double the number of output units is a proper number for the NILM model through heuristic experiments. Based on these methods, we construct the architecture of an LSTM-RNN for NILM (Figure 8).

4. Dataset Description

There are several public datasets for NILM. Table 1 shows descriptions for them. Among them, we concentrated on two datasets such as REDD and UK-DALE because of proper for NILM. Beside the public dataset, we had collected 6 appliances and aggregate data during 5 months. We are going to use our private dataset as well.

The UK-DALE was released in 2014 at first and it has been updated every year [38]. There are power data for 5 households and each house has the different first measurement date. House 1 includes 3.5-year power data for 52 appliances. The 16 kHz data will not be used in this paper. And houses 1, 2, and 5 include 1-second data but there are no labeled data. Therefore we will not use 1-second data as well. We give an explanation for the 6-second data. Table 2 shows the detailed description of UK-DALE. House 1 has the large number of appliances. As the number of appliances is increased, a NILM model is hard to distinguish the appliances. But the disadvantage can be overcome because of much data. On the other hand, houses 3 and 4 have the small number of meters. The difference between of them is that a meter of house 4 is shared. For example, "tv_dvd_digibox_lamp" means that

Table 1: Public datasets for NILM [31].

Dataset	Number of houses	Duration per house	Appliance sample frequency	Aggregate sample frequency
REDD (2011)	6	3–19 days	3 sec	1 sec, 15 kHz
BLUED (2012)	1	8 days	N/A	12 kHz
Smart (2012)	3	3 months	1 sec	1 sec
Tracebase(2012)	N/A	N/A	1–10 sec	N/A
Sample (2013)	10	7 days	1 min	1 min
HES (2013)	251	1 or 12 months	2 or 10 min	2 or 10 min
AMPds (2013)	1	1 year	1 min	1 min
iAWE (2013)	1	73 days	1 or 6 sec	1 sec
UK-DALE (2016)	5	1–3.5 years	6 sec or 16 kHz	1 or 6 sec, 16 kHz

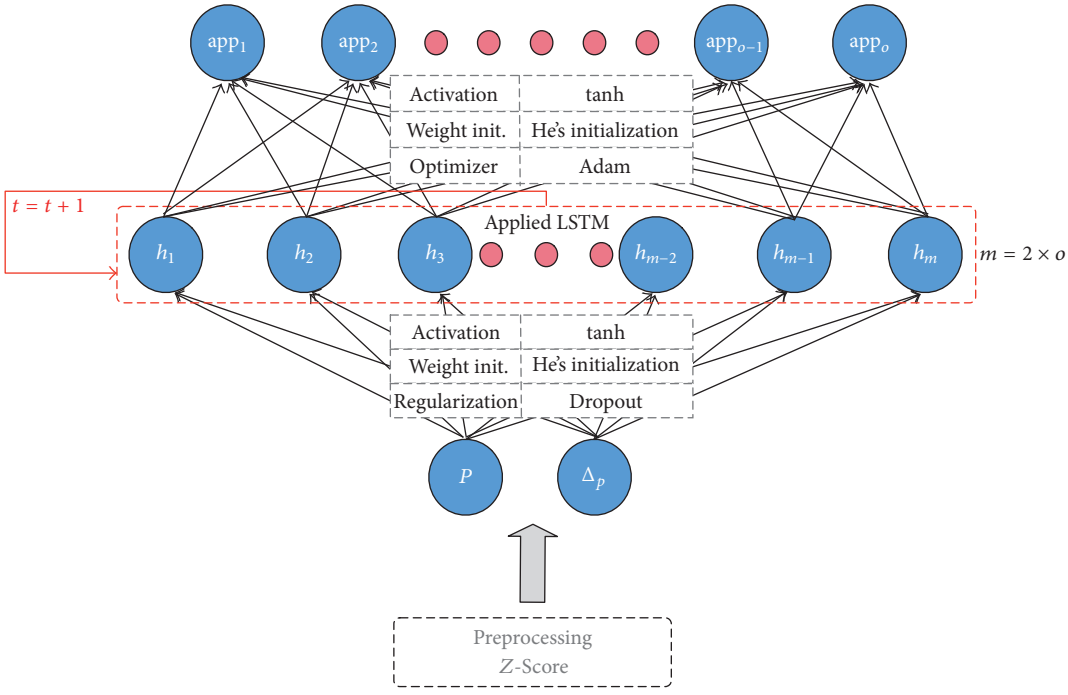


Figure 8: LSTM-RNN for NILM.

the 4 appliances are using the same meter. When the NILM model is trained by a dataset, the shared meter may cause a confusion or make a distinguishable pattern. In Section 5, we can confirm the effect of the shared meter.

The REDD is the first public dataset for NILM [26]. The major purpose of REDD is the standard dataset for benchmarking the NILM algorithms. In REDD, there are AC waveform data with sampling rate of 15 kHz. Therefore, REDD can be used for each approach using the high or low sampling data. It is sampled from six different houses in Massachusetts of the United States. There are three categorized data, low_freq, high_freq, and high_freq_raw. We are going to use low_freq data in this paper. Each house's dataset is composed of the main phases sampled 1 Hz and the individual data of each appliance sampled 3 or 4 Hz. Table 3 shows the detailed description of REDD. Each house has the appliance in range from eleven to twenty-six. We renamed the same appliances like kitchen_outlets1 and kitchen_outlets2.

The problem of UK-DALE and REDD is the asynchronization between a total load and each appliance data due to the different sample rate. So we have to synthesize each appliance data for generating an output dataset. When we collect the data from the real household, noise can be included in the total load. However, the synthesized data is not affected on the noise. To solve the problem, we collect the data of six appliances for five months. The description of the private data is in Table 4.

5. Experiment

5.1. Experiment Description. The purpose of the experiment is to satisfy three challenges defined. The challenges are as follows.

Challenge 1. Our model must be robust even though the number of appliances is increasing.

Table 2: The detailed description of UK-DALE.

House	Date of first measurement	Submeters	Appliances
1	2012-11-19	53	Boiler, solar_thermal_pump, laptop, washing_machine, dishwasher, tv, kitchen_lights, htpc, kettle, toaster, fridge, microwave, lcd_office, hifi_office, breadmaker, amp_livingroom, adsl_router, livingroom_s_lamp, soldering_iron, gigE_&_USBhub, hoover, kitchen_dt_lamp, bedroom_ds_lamp, lighting_circuit, iPad_charger, subwoofer_livingroom, livingroom_lamp_tv, and so on
2	2013-02-17	20	Laptop, monitor, speakers, server, router, server_hdd, kettle, rice_cooker, running_machine, laptop2, washing_machine, dish_washer, fridge, microwave, toaster, playstation, modem, cooker
3	2013-02-27	5	Kettle, electric_heater, laptop, projector
4	2013-03-09	6	Tv_dvd_digibox_lamp, kettle_radio, gas_boiler, freezer, washing_machine_microwave_breadmaker
5	2014-06-29	26	Stereo_speakers, desktop, hairdryer, primary_tv, 24_inch_lcd, treadmill, network_attached_storage, server, 24_inch_lcd_bedroom, PS4, steam_iron, nespresso_pixie, atom_pc, toaster, home_theatre_amp, sky_hd_box, kettle, fridge_freezer, oven, electric_hob, dishwasher, microwave, washer_dryer, vacuum_cleaner

Table 3: The detailed description of REDD.

House	Submeters	Appliances
1	20	Oven1, oven2, refrigerator, washer_drye1, dishwasher, kitchen_outlets1, kitchen_outlets2, lighting, lighting1 washer_dryer2, microwave, bathroom_gfi, electric_heat, stove, kitchen_outlets3, washer_dryer3kitchen_outlets4, lighting2
2	11	Kitchen_outlets1, lighting, dishwasherstove, microwave, washer_dryer, kitchen_outlets2, refrigerator, disposal
3	22	Outlets_unknown1, kitchen_outlets1outlets_unknown2, lighting 1, microwave, electronics, refrigerator, disposal, dishwasher, furance, lighting2, lighting3, outlets_unknown3, washer_dryer1, washer_dryer2, lighting4, smoke_alarms, lighting5, bathroom_gfi, kitchen_outlets2
4	20	Lighting1, furance, kitchen_outlets1, stove, outlets_unknown, washer_dryer, air_conditioning1, air_conditioning2, miscellaeneous, smoke_alarms, lighting2, kitchen_outlets2, dishwaser, bathroom_gfi1, bathroom_gfi2, lighting3, lighting4, air_conditioning3
5	26	Microwave, lighting1, outlets_unknown1, furance, outlets_unknown2, washer_dryer1, washer_dryer2, subpanel1, subpanel2, electric_heat1, electric_heat2, lighting2, outlets_unknown3, bathroom_gfi, lighting3, refrigerator, lighting4, dishwaser, disposal, electronics, lighting5, kitchen_outlets1, kitchen_outlets2, outdoor_outlets
6	17	Kitchen_outlets1, washer_dryer, stove, electronics,bathroom_gfi, refrigerator, dishwasher, outlets_unknown1, outlets_unknown2, electric_heat, kitchen_outlets2, lighting, air_conditioning1, air_conditioning 2, air_conditioning3

Challenge 2. The multistate appliance should be classified as well.

Challenge 3. The similar power consumption in appliances should be classified.

In this section, we take two validation experiments for the proposed signature and learning method and one performance measurement experiment. The first validation experiment consists of two subtests. The first one is a heuristic approach test for the optimized reflection rate. In this test, we compute three different Δ_p by three different reflection rates. By using Δ_ps, we train the models with the same condition. The second subtest is related to challenge 2. In Section 3.2.1, we expected that the novel signature could classify the multistate appliances having many variations. To validate this, we train two models having $\{P\}$ and $\{P, \Delta_p\}$ as inputs for each. And then we analyze the results.

The second validation experiment is related to challenges 1 and 3. This experiment consists of four subtests. In the first subtest, we are going to find an optimized preprocessing method. Z-Score normalization and Min–Max Scaling are popular methods for preprocessing. There is no rule which one is proper for a specific situation. So we experiment them for choosing the optimized method. The second subtest is about choice of the weight initialization methods. Among the initialization methods, the Glorot initialization and the

Table 4: Private dataset.

House	Date of first measurement	Submeters	Appliances
1	2016-03-11	6	Air-con., dehumidifier, TV, washer, toaster, oven

	Positive (predicted)	Negative (predicted)
Positive (actual)	TP	FN
Negative (actual)	FP	TN

Figure 9: Confusion matrix.

He initialization are well known for providing good performance. But it is not revealed that which one is better or not. Therefore, we train two models with only changing the initialization method and compare the result. In the third subtest, we compare the performance as the number of appliances is increased. We could confirm that our model is robust even though the number of appliances is increased. In the last subtest, we synthesize the sampled data of appliances having the similar power consumption. Then, we confirm that the appliances are well classified or not.

The last experiment is about measuring the overall performance. We are going to use the UK-DALE and REDD as the training/test data and measure the performance for each house in the dataset. Also, we take the same experiment by FHMM and compare the result. All models used in the experiment are implemented with Python programming and Theano library.

5.2. Performance Metrics. Up to now we have generally assumed that the best way of measuring the performance of a classifier is by its predictive accuracy, that is, the proportion of unseen instances it correctly classifies. However, predictive accuracy on its own is not a reliable indicator of a classifier's effectiveness. As well as the overall predictive accuracy on unseen instances it is often helpful to see a breakdown of the classifier's performance; it is a *confusion matrix* which is proposed by Kohavi and Provost, in 1998 [39].

Confusion matrix is described in Figure 9, including four categories. True positive (TP) is examples correctly labeled as positive. False positive (FP) refers to negative examples incorrectly labeled as positive. True negative (TN) corresponds to negatives correctly labeled as negative. Finally, false negative (FN) refers to positive examples incorrectly labeled as negative.

We assume that the positive state means that an appliance is turned on. When the appliance is turned off, we regard it as a negative state. Based on the confusion matrix, we can calculate *Recall*, *Precision*, *Accuracy*, and *F*1-*score* for evaluating the NILM model. The formulas to compute the value of them are given by

$$\begin{aligned} \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Accuracy} &= \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{FP}) + (\text{FN} + \text{TN})}, \\ F1\text{-score} &= 2 * \frac{\text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}. \end{aligned} \tag{2}$$

Recall is a ratio of the number of correct classifications to the total number of actual positive instances. A meaning of that *Recall* is high and the appliances are well classified by the NILM model when the actual instances are positive. *Precision* is a ratio of the number of correct classifications to the total number of predicted positive instances. A meaning of that *Precision* is high and a probability of well classification is high when the NILM model predicts the positive instances. *Accuracy* is a ratio of correct classification to the total test data. *F*1-*score* is the harmonic average of *Recall* and *Precision*.

5.3. Proposed Signature Validation Experiment

5.3.1. Optimum Reflection Rate. To find an optimum reflection rate, we set three different reflection rates, 0.1, 0.01, and 0.001, and compute three variant signals. For experiment, we randomly select the 15 appliances from UK-DALE. By using the variant signals, we calculate three Δ_p. $\Delta_p 1$ is a result of subtraction of the first signal and the second signal. $\Delta_p 2$ is a result of subtraction of the first signal and the third signal. $\Delta_p 3$ is a result of subtraction of the second signal and the third signal. Figure 10 shows the pattern of each Δ_p. Intuitionally, $\Delta_p 2$ and $\Delta_p 3$ are biased. The values of each Δ_p are detailed in Table 5. We can notice that the mean of $\Delta_p 1$ is close to zero unlike the others. A variance and standard deviation of $\Delta_p 1$ are almost in middle of the others. And the scale of range of $\Delta_p 1$ and $\Delta_p 2$ is similar. We could know which one is proper to NILM by a comparison experiment.

Table 6 shows performance results. All performances are similar except that $\Delta_p 1$ is better than the others. Therefore, we will use (0.1, 0.01) as a reflection rate.

5.3.2. Effect of the Novel Signature. In this section, we train the two models having $\{P\}$ and $\{P, \Delta_p\}$ as an inputs for each. We extracted 72-day data as a training dataset and 30-day data as a test dataset. Each model was trained 2000 epochs. As we explained in Section 4, the private dataset has six appliances. Actually, these appliances are not operating in the same time. However, in our work, we assumed that multiple appliances could have been operated in the same time. Therefore, $2^6 = 64$ which is the number of combinations of all appliances. We represented the on/off state by the binary representation. And we change the output unit values to one binary string. After that, the string is converted to a decimal number that stands for the combination. A sequence of appliances is {Air-conditioner, Washer, Dehumidifier, Oven, TV, Toaster}.

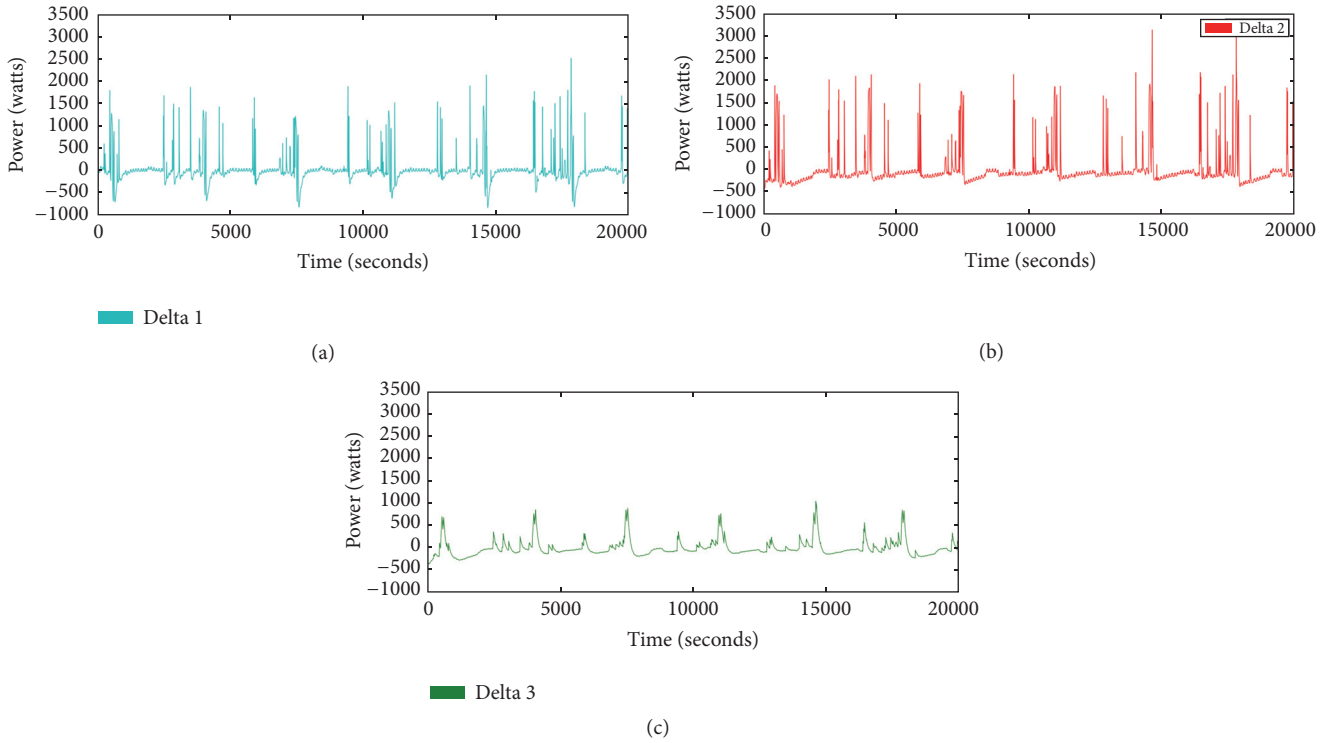


Figure 10: Comparison of Δ_p. (a) Pattern of $\Delta_p 1$; (b) pattern of $\Delta_p 2$; (c) pattern of $\Delta_p 3$.

Table 5: Statistic of Δ_p.

Δ_p	Range	Mean	Variance	Standard deviation
$\Delta_p 1$	(−841~2510)	1.047	83116.404	288.299
$\Delta_p 2$	(−411~3149)	−13.361	140077.047	374.268
$\Delta_p 3$	(−366~1040)	−14.358	34845.009	186.668

For example, if the washer and toaster are operating, the binary string would be {010001} and the combination number would be 17.

Figure 11 shows the classification ratio comparison between the models. The x-axis is a combination and the y-axis is a classification ratio. We can easily notice that if we use the proposed signature additionally, the more combinations could be classified. 30 combinations were classified when we use the proposed signature whereas 11 combinations were classified when we use only power consumption (not all combinations are included in the dataset). Figure 12 shows the performance of each appliance. The second model using $\{P, \Delta_p\}$ has the better performance in all appliances except the dehumidifier. Particularly, there are large gaps in the washer, oven, and toaster. The operation time for the oven and toaster is only 1.5 hours and 2.8 hours during 30 days. Even though the sample number is small, the second model has the better performance relatively.

However, the dehumidifier performance of first model is higher than that of the second one. We see this performance in Figure 13. The pattern of dehumidifier is simple. A fluctuating Δ_p could be efficient for learning the pattern but the other case is not. Because the importance of Δ_p is the same with the power consumption, this kind of simple pattern could be learned slowly. To confirm this, we trained 2000 epochs more for the second model. In Table 7, the performance of dehumidifier is improved from 0.65 to 0.82 whereas the other performances are similar with before. Because the washer has fluctuating Δ_p, the performance of the second model is about double. As a result, the proposed signature could be very efficient factor for classifying the multistate appliances.

5.4. Deep Learning Model Validation

5.4.1. Optimum Preprocessing Method. Z-Score normalization and Min–Max Scaling are popular methods for preprocessing. However, it has not revealed which one is better. To find an optimum method for NILM, we train the two different models. The datasets for all models are the same. We extracted 23-day data from house 5 of UK-DALE. 17-day data is used as a training dataset and 6-day data is used as a test dataset. Each model trains 1000 epochs.

Figure 14 shows the performance results of the models. We can notice that Z-Score method has the better performance than Min–Max Scaling. In NILM, one of the inputs is the power consumption. As we know, all power consumption

Table 6: Performance measurement with different reflection rates.

Δ_p (reflection rate)	Precision	Recall	Accuracy	$F1$-Score
$\Delta_p 1$ (0.1, 0.01)	0.753	0.843	0.901	0.795
$\Delta_p 2$ (0.1, 0.001)	0.714	0.836	0.886	0.771
$\Delta_p 3$ (0.01, 0.001)	0.730	0.812	0.888	0.769

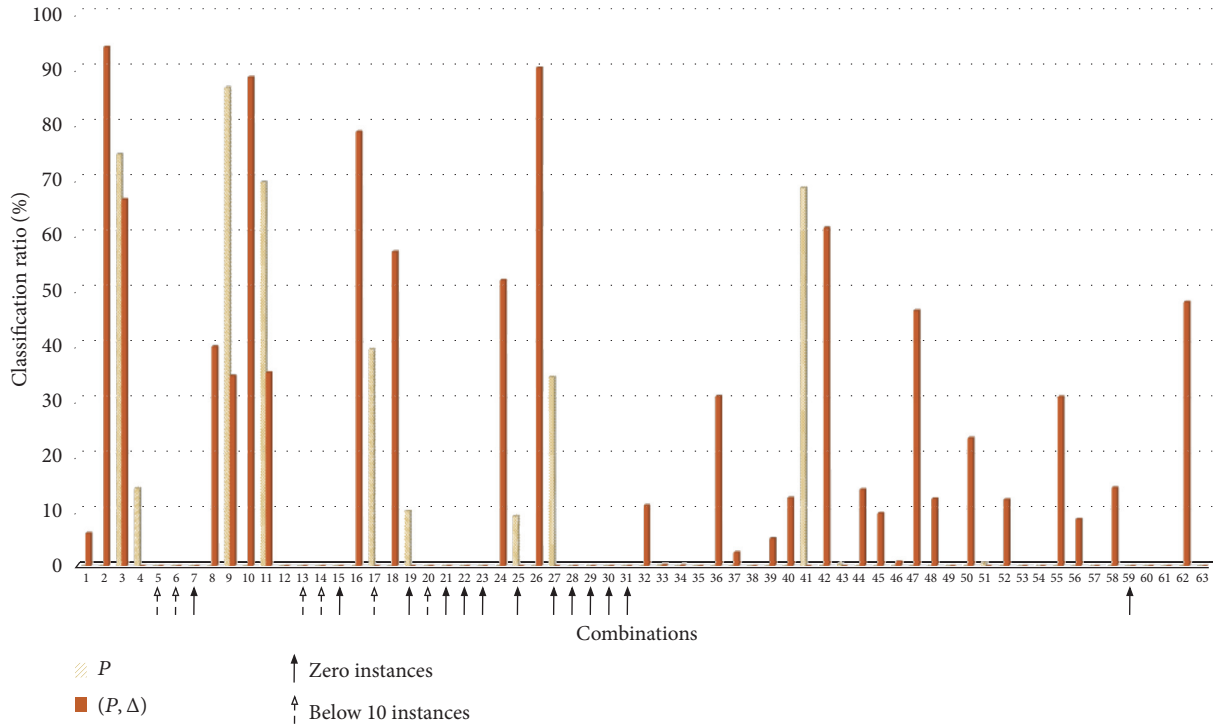


Figure 11: Classification ratio comparison $\{P\}$, $\{P, \Delta_p\}$.

Table 7: The performance of second model (4000 epochs).

Appliance	$F1$-Score
Air-conditioner	0.89
Washer	0.67
Dehumidifier	0.82
Oven	0.55
TV	0.70
Toaster	0.49

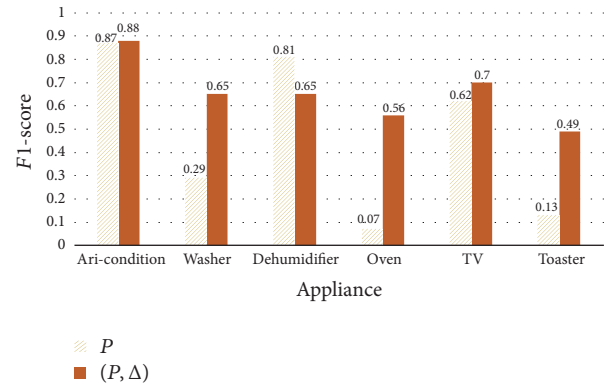
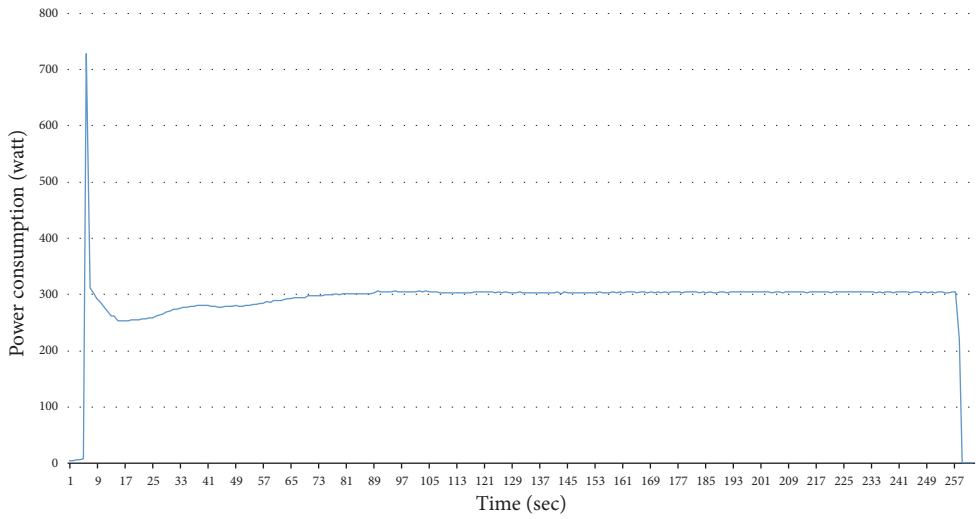


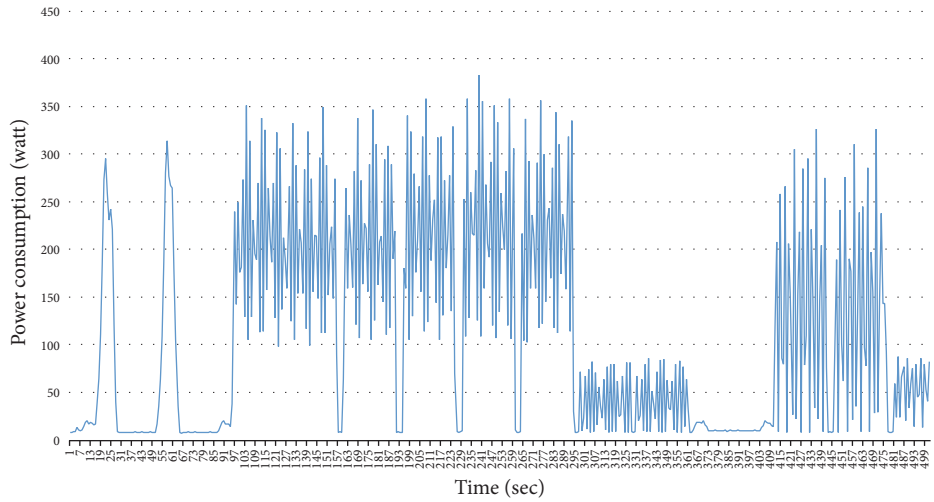
Figure 12: Classification performance of the appliances.

is positive. If all input data are the positive value, all weights could be increased or decreased when the backpropagation is processed. As a result, the dataset for NILM should be standardized. As expected in Section 3.2.2, Z-Score method is proper for preprocessing.

5.4.2. Optimum Weight Initialization Method. Among the weight initialization methods introduced, the researchers usually use Glorot initialization and He initialization. But they still have argued about the better method between two methods. The methods use a similar theoretical analysis. They found a good variance for the distribution from which the initial parameters are drawn. This variance is adapted to the activation function used and is derived without explicitly considering the type of the distribution. As such, their theoretical conclusions hold for any type of distribution of the determined variance. To compare the weight initialization methods, we applied the same distribution (uniform distribution) to the methods. We extracted 71-day data from the private dataset. 50-day data are used as a training dataset and 21-day data are used as a test dataset. We train two models



(a)



(b)

Figure 13: The patterns of dehumidifier and washer: (a) dehumidifier; (b) washer.

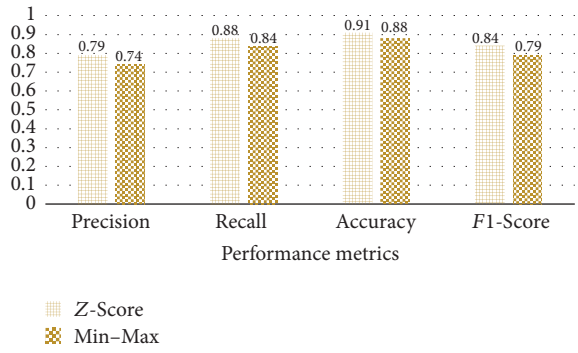


Figure 14: Comparison of the preprocessing methods.

during 1000 epochs for each and set a time step to be 500, a loss function to be MSE, and optimization to be Adam.

Table 8 shows the result of experiment. The precision of Glorot initialization is much higher than the precision of He initialization. In case of the recall, He's method is higher than Glorot's method. The model which used Glorot initialization has a high hit rate but a number of classified samples are low. The model which used He initialization has a low hit rate but a number of classified samples are high. The accuracy and $F1$-Score are similar. Before we select the method, we trained the models during 5000 epochs for each. Table 9 shows the results. After training more than 4000 epochs, the precision of He's method is much improved than the previous result even though the recall is dropped. However, the model which used Glorot initialization is not much improved in all metrics. He initialization was proposed in case the input data is positive. The power consumption is all positive values but Δ_p is not. We could not convince He initialization. But guessing from the result, we can know that He initialization

Table 8: Performance comparison of initialization methods (1000 epochs).

Metric	He initialization	Glorot initialization
Precision	0.529	0.962
Recall	0.775	0.456
Accuracy	0.934	0.959
$F1$-Score	0.629	0.619

Table 9: Performance comparison of initialization methods (5000 epochs).

Metric	He initialization	Glorot initialization
Precision	0.954	0.954
Recall	0.535	0.466
Accuracy	0.964	0.960
$F1$-Score	0.685	0.626

Table 10: Control parameters for experiment.

Control parameter	Method
Input	P, Δ_p
Num. of training samples	1,500,000
Num. of test samples	500,000
Time step	500
Num. of epochs	1000
Cost function	Mean Squared Error
Optimization	Adam
Preprocessing	Z-Score
Weight init.	He initialization
Regularization	Dropout

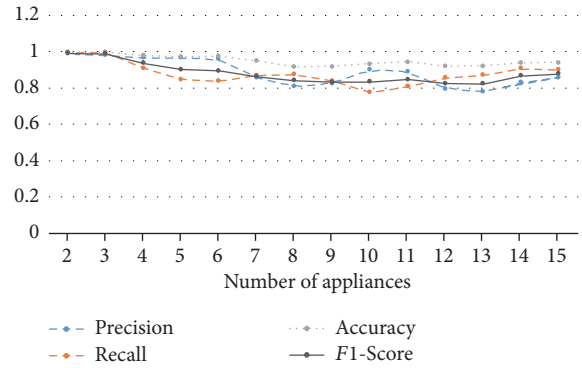


Figure 15: The result of performance measurement model in increasing appliances.

is efficient for NILM. As a result, we will use He initialization as the weight initialization method.

5.4.3. Performance Measurement with Increasing the Number of Appliances. In this section, we are going to take an experiment for confirming the robustness of proposed model even though the number of appliances is increased. In the experiment, we hold all parameters except the number of appliances. The control parameters are in Table 10. The fifteen appliances for measuring the performance are selected from house 5 of UK-DALE (Table 20). Table 11 shows the appliances used for each experiment. To observe the performance precisely, we synthesize the data of appliances with adding one appliance to the previous ones.

Figure 15 shows the result of experiment. The x-axis is the number of appliances. The $F1$-Score representing the overall performance of the model is decreased from 2 to 9. However, after 9 appliances, the $F1$-Score pattern walks up and down in range from 0.8 to 0.9. Due to the high performance, the $F1$-Score is going down in the early phase. However, it is going up and down in the late phase. Therefore, we cannot conclude that the model is influenced by the number of appliances. We compare the two models; those numbers of appliances are 13 and 14 for each. The $F1$-Score of first model is 0.822 but it is increased to 0.866 when the 14 appliances are trained. The added appliance is home_theatre_amp. The total operation time is about 68 hours and we can notice that the appliance has many variations in Figure 16. On the other hand, the $F1$-Score is going down when the number of appliances is from 7 to 8. The added appliance is core2_server. We can notice that the power pattern is monotonous in Figure 17. Therefore, we can think that the major factor affecting on the performance is the data. As a result, we can conclude that the proposed model for NILM is robust against the number of appliances.

5.4.4. Performance Measurement of Appliances Having the Similar Power Consumption. In this section, we are going to take an experiment for confirming that the appliances having a similar power consumption are well classified by the NILM model. If the similar power appliances are operating in a different time, the model could be hard to classify them. However, the power consumption and the operation time are slightly different even though the appliances are the similar power appliance. The power gap would be emphasized by the proposed signature and the operation time could be memorized in the hidden neurons of the proposed model. To validate our assumption, we collected the 11 numbers of the similar power appliances from UK-DALE and synthesized their data in Table 11. And the control parameters are the same with the previous experiment (Table 10). Table 12 shows the result of performance measurement. We can see that they

Table 11: A List of the similar power appliances.

Number	Appliance	Power range (watt)
(1)	adsl_router	6~7
(2)	Data_logger_pc	12~13
(3)	hifi_office	12~14
(4)	Livingroom_lamp_tv	11~14
(5)	Livingroom_s_lamp2	7~9
(6)	Modem	9~10
(7)	Office_lamp1	14
(8)	Office_lamp2	9~10
(9)	Server-hdd	10~13
(10)	Speaker	10~11
(11)	Subwoofer_livingroom	15~16

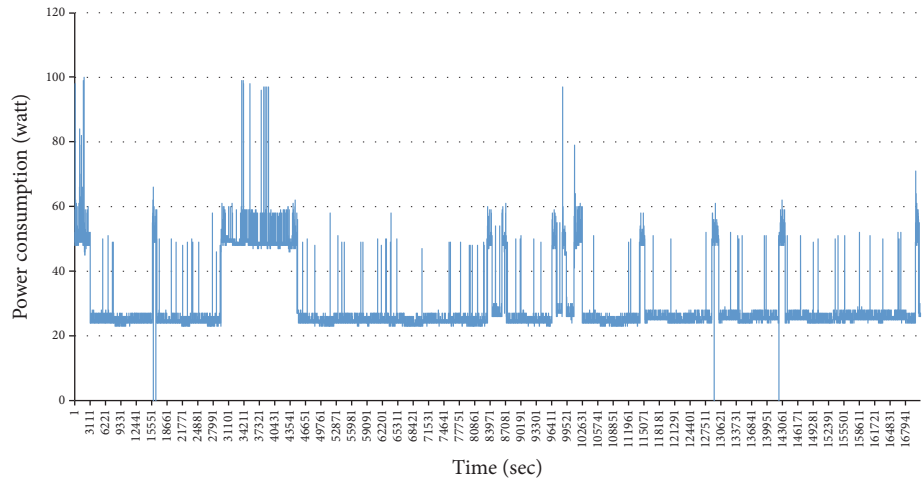


Figure 16: Power pattern of home_theatre_amp.

Table 12: Result of performance measurement for classification of similar power appliance.

Metric	Similar power appliance based model
Precision	0.955
Recall	0.948
Accuracy	0.957
*F*1-Score	0.951

are well classified even though they are the similar power appliances.

To validate our assumption, we collected the 11 numbers of the similar power appliances and 3 numbers of the multistate appliances having a high power consumption from UK-DALE and synthesized their data in Table 13. And the control parameters are the same with the previous experiment (Table 10). We train two models. The first one is the model using only similar power appliances. In the second model, the multistate appliances are included additionally. As we expected, we can observe that the performance is going down from the first model to the second model in Table 14. Although the performance is decreased, we can realize that.

5.5. Performance Comparison with the Existing Models. We could confirm that our proposed signature and model are efficient for NILM through the validation experiments. In this section, we would like to compare the performance with the existing models. We take an experiment for all houses in UK-DALE and REDD. Firstly, we train all houses of UK-DALE and the result will be compared with FHMM. The parameters for UK-DALE experiments are in Table 15.

Table 16 shows the overall performance of UK-DALE houses. The house having the lowest performance is the first house. We proved that our model is affected by the data. There are 53 appliances in the first house. The possibility of being the appliance data decreasing the performance is higher than the other houses. For example, the usage of battery_charger and breadmaker is very rare. This kind of appliance is the major reason for dropping the performance. We compare the performance with FHMM by implementing it. The result is shown in Table 17. We can notice that our model is outperforming FHMM. The performance is almost double in most houses.

Secondly, we train all houses of REDD and the result will be compared with the existing models. The results are good in all houses (see Table 18). We compare our model with

Table 13: A List of the similar power appliances with adding three multistate appliances.

Number	Appliance	Power range (watt)
(1)	adsl_router	6~7
(2)	Data_logger_pc	12~13
(3)	hifi_office	12~14
(4)	Livingroom_lamp_tv	11~14
(5)	Livingroom_s_lamp2	7~9
(6)	Modem	9~10
(7)	Office_lamp1	14
(8)	Office_lamp2	9~10
(9)	Server-hdd	10~13
(10)	Speaker	10~11
(11)	Subwoofer_livingroom	15~16
(12)	Fridge	85~252
(13)	Gas_oven	14~52
(14)	Hoover	500~2021

Table 14: Result of performance measurement for classification of similar power appliances with adding three multistate appliances.

Metric	Similar power appliance based model	All appliance based model
Precision	0.955	0.896
Recall	0.948	0.863
Accuracy	0.957	0.911
$F1$-Score	0.951	0.879

Table 15: Parameters for UK-DALE experiment.

Parameter	Method
Input	P, Δ_P
Time step	500
Num. of epochs	3000
Cost function	Mean Squared Error
Optimization	Adam
Preprocessing	Z-Score
Weight init.	He initialization
Regularization	Dropout

three existing models such as FHMM, Additive FHMM, and HieFHMM [28, 32]. Among them, HieFHMM is the state-of-the-art model [34]. Due to few numbers of appliances, the existing models did not test house 2. Similar to UK-DALE result, our model has the highest performance in all houses. In case of house 4, our performance is double compared to that of state of the art. We can confirm this in Table 19.

6. Conclusions

In this paper, we considered advanced deep learning is new approach to build state-of-the-art NILM tool. Besides, we proposed a novel signature to overcome multistate appliance issues in NILM. Furthermore, we give three-problem statement and then address them as efficiently. By this way, we perform many experiments for validating the signature and the architecture of the proposed model. Through our experimental results, we realized that our model overcomes the existing problem in energy disaggregation. Via measurement the overall performance, we confirmed that our model outperformed the previous models. In particular, the performance of house 4 in REDD dataset is double result compared to that of state of the art.

In future work, we will focus on a prediction of the power consumption of each appliance, because this problem is different in classification task. We cannot apply our currently model to prediction. Hence, we need to modify our model to generative model. If we can do this, our model could be applied to gas or water disaggregation. In other words, we construct the model which is used for the comprehensive energy management at home.

Abbreviations

NILM: Nonintrusive Load Monitoring
FHMM: Factorial Hidden Markov Model
GPU: Graphics Processing Unit
UK-DALE: UK Domestic Appliance Level Electricity
REDD: Reference Energy Disaggregation Data
HMM: Hidden Markov Model
FSM: Finite State Machine
MP: Markov process
EM: Expectation Maximization
HSMM: Hidden Semi-Markov Model
DHMM: Difference Hidden Markov Model
DBN: Deep Belief Network
RBM: Restricted Boltzmann Machine
AI: Artificial Intelligence

Table 16: Overall performance of UK-DALE.

House	Precision	Recall	Accuracy	*F*1-Score
1	0.778	0.752	0.917	0.764
2	0.872	0.756	0.905	0.810
3	0.950	0.856	0.981	0.900
4	0.964	0.972	0.969	0.968
5	0.816	0.878	0.914	0.846

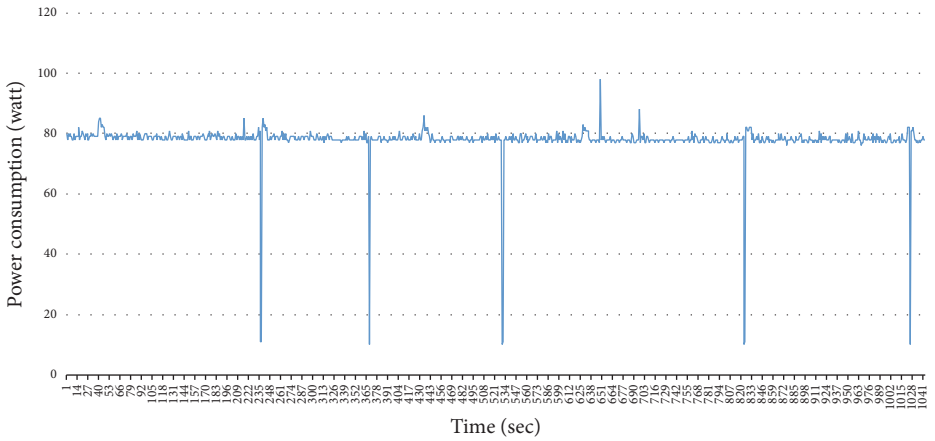


Figure 17: Power pattern of core2_server.

Table 17: Performance comparison with FHMM (UK-DALE).

House	FHMM	Our model
1	0.304	**0.764**
2	0.423	**0.810**
3	0.495	**0.900**
4	0.651	**0.968**
5	0.424	**0.846**

Table 18: Overall performance of REDD.

House	Precision	Recall	Accuracy	*F*1-Score
1	0.942	0.959	0.968	0.950
2	0.961	0.943	0.964	0.952
3	0.829	0.840	0.920	0.835
4	0.846	0.867	0.909	0.856
5	0.930	0.893	0.953	0.911
6	0.898	0.954	0.943	0.925

AAT: Automatic Alternative Text
ANN: Artificial Neural Network
tanh: Hyperbolic tangent
MSE: Mean Squared Error
CE: Cross Entropy
GD: Gradient Descent
RNN: Recurrent Neural Network
BPTT: Backpropagation through Time
RTRL: Real Time Recurrent Learning
FFNN: Feed-Forward Neural Network
LSTM: Long Short-Term Memory
PCA: Principal Component Analysis
ReLU: Rectifier Linear Unit
SGD: Stochastic Gradient Decent
RMS: Root Mean Square
STFT: Short Time Fourier Transform
FHSMM: Factorial Hidden Semi-Markov Model
CFHMM: Conditional Factorial Hidden Markov Model
CFHSMM: Conditional Factorial Hidden Semi-Markov Model
AFAMAP: Additive Fractional Approximate MAP
VAST: Viterbi Algorithm with Sparse Transitions
MC: Markov chain
CT: Current Transformer
BLUED: Building Level Fully Labeled Data Set for Electricity Disaggregation
CMU: Carnegie Mellon University
UMASS: University of Massachusetts
HES: Household Electricity Use Study
AMPds: Almanac of Minutely Power Dataset
IIIT: Indraprastha Institute of Information Technology
iAWE: International Associations for Wind Engineering
TP: True positive
FP: False positive
TN: True negative
FN: False negative
HieFHMM: Hierarchical Factorial Hidden Markov Model

Table 19: Performance comparison with the existing models (REDD).

House	FHMM [33]	Additive FHMM [34]	HieFHMM [34]	Our model
1	0.450	0.749	0.854	**0.950**
3	0.590	0.619	0.834	**0.835**
4	0.430	0.417	0.424	**0.856**
5	0.500	0.795	0.796	**0.911**
6	0.440	0.391	0.820	**0.925**

Table 20: Appliances for each experiment.

Number of appliances	Included appliance
2	Stereo_speakers_bedroom, i7_desktop
3	Stereo_speakers_bedroom, i7_desktop, hairdryer
4	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv
5	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom
6	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill
7	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage
8	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server
9	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd
10	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd, steam_iron
11	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd, steam_iron, nespresso_pixie
12	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd, steam_iron, nespresso_pixie, atom_pc
13	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd, steam_iron, nespresso_pixie, atom_pc, toaster
14	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd, steam_iron,nespresso_pixie, atom_pc, toaster, home_theatre_amp
15	Stereo_speakers_bedroom, i7_desktop, hairdryer, primary_tv, 24_inch_lcd_bedroom, treadmill, network_attached_storage, core2_server, 24_inch_lcd, steam_iron, nespresso_pixie, atom_pc, toaster, home_theatre_amp, sky_hd_box

HW: Hamming weight
WISA: International Workshop on Information Security Applications
MBC: Mask based Block Comb
NCC: Normalized Cross Correlation.

Disclosure

The current address of Jihyun Kim is Room 203, Science Technology Research Bldg. (709), Pusan National University, Jangjeon-dong, Geumjeong-gu, Busan 609-735, Republic of Korea. The current address of Thi-Thu-Huong Le and Howon Kim is Pusan National University, Information Security & IoT Lab, Room 6405-1, 6th Eng. Bldg. (A06), Pusan National University, Jangjeon-Dong, Geumjeong-Gu, Busan 609-735, Republic of Korea. The funding sponsor had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; and in the decision to publish the findings.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Jihyun Kim defined three NILM problems as well as three challenges of NILM. Besides, he proposed novel signature and its algorithm. He developed the data generation process and made datasets for the experiment. Thi-Thu-Huong Le developed LSTM-RNN model on NILM and performed the experiments. Professor Howon Kim managed this project and reviewed the manuscript.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Industry 4.0s Research and Development Program (S0604-17-1002)

supervised by the NIPA (National IT Industry Promotion Agency).

References

[1] International Energy Agency, World Energy Outlook 2015. White Paper, IEA, 2015.

[2] D. Larcher and J.-M. Tarascon, "Towards greener and more sustainable batteries for electrical energy storage," *Nature Chemistry*, vol. 7, no. 1, pp. 19–29, 2015.

[3] International Electrotechnical Commission, Report to WTO TBT Committee. White Paper, IEC, 2010.

[4] J. Lee, B. Bagheri, and H.-A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[5] P. Zhao, S. Suryanarayanan, and M. G. Simoes, "An energy management system for building structures using a multi-agent decision-making control methodology," *IEEE Transactions on Industry Applications*, vol. 49, no. 1, pp. 322–330, 2013.

[6] D. Niyato, L. Xiao, and P. Wang, "Machine-to-machine communications for home energy management system in smart grid," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 53–59, 2011.

[7] R. Ford and C. Church, *Reducing domestic energy consumption through behaviour modification [Phd. thesis]*, 2009.

[8] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[9] S. Lee, B. Song, Y. Kwon, and J. Kim, "Non-intrusive Load Monitoring for Home Energy Usage with Multiple Power States Recognition," in *Proceedings of the Computer and Computing Science 2015*, pp. 282–289.

[10] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *Proceedings of the 11th SIAM International Conference on Data Mining (SDM '11)*, pp. 747–758, April 2011.

[11] H. Najmeddine, K. El Khamlichi Drissi, C. Pasquier et al., "State of art on load monitoring methods," in *Proceedings of the 2008 IEEE 2nd International Power and Energy Conference, PECon 2008*, pp. 1256–1258, mys, December 2008.

[12] M. B. Figueiredo, A. De Almeida, and B. Ribeiro, "An experimental study on electrical signature identification of non-intrusive load monitoring (nilm) systems," *International Conference on Adaptive and Natural Computing Algorithms*, pp. 31–40, 2011.

[13] K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura, and K. Ito, "Nonintrusive appliance load monitoring based on integer programming," in *Proceedings of the SICE Annual Conference 2008 - International Conference on Instrumentation, Control and Information Technology*, pp. 2742–2747, August 2008.

[14] A. Cole and A. Albicki, "Nonintrusive identification of electrical loads in a three-phase environment based on harmonic content," in *Proceedings of the IMTC/2000 - 17th IEEE Instrumentation and Measurement Technology Conference 'Smart Connectivity: Integrating Measurement and Control'*, pp. 24–29, May 2000.

[15] C. Laughman, K. Lee, R. Cox et al., "Power signature analysis," *IEEE Power and Energy Magazine*, vol. 1, no. 2, pp. 56–63, 2003.

[16] J. Li, S. West, and G. Platt, "Power decomposition based on SVM regression," in *Proceedings of the 2012 International Conference on Modelling, Identification and Control (ICMIC '12)*, pp. 1195–1199, June 2012.

[17] H. Y. Lam, G. S. K. Fung, and W. K. Lee, "A novel method to construct taxonomy electrical appliances based on load signatures," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 2, pp. 653–660, 2007.

[18] S. Gupta, M. S. Reynolds, and S. N. Patel, "ElectriSense: Single-point sensing using EMI for electrical event detection and classification in the home," in *Proceedings of the 12th International Conference on Ubiquitous Computing, UbiComp 2010*, pp. 139–148, dnk, September 2010.

[19] H. H. Chang, H. T. Yang, and C.-L. Lin, "Load identification in neural networks for a non-intrusive monitoring of industrial electrical loads," in *Proceedings of the International Conference on Computer Supported Cooperative Work in Design*, pp. 664–674, 2007.

[20] H.-H. Chang, "Non-intrusive demand monitoring and load identification for energy management systems based on transient feature analyses," *Energies*, vol. 5, no. 11, pp. 4569–4589, 2012.

[21] S. B. Leeb, S. R. Shaw, and J. L. Kirtley, "Transient event detection in spectral envelope estimates for nonintrusive load monitoring," *IEEE Transactions on Power Delivery*, vol. 10, no. 3, pp. 1200–1210, 1995.

[22] L. K. Norford and S. B. Leeb, "Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms," *Energy and Buildings*, vol. 24, no. 1, pp. 51–64, 1996.

[23] A. I. Cole and A. Albicki, "Data extraction for effective non-intrusive identification of residential power loads," in *Proceedings of the 1998 IEEE Instrumentation and Measurement Technology Conference (IMTC '98)*, pp. 812–815, May 1998.

[24] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd, "At the flick of a switch: detecting and classifying unique electrical events on the residential power line (nominated for the best paper award)," in *In International Conference on Ubiquitous Computing*, pp. 271–288, 2007.

[25] A. Zoha, A. Gluhak, M. Nati, and M. A. Imran, "Low-power appliance monitoring using Factorial Hidden Markov Models," in *Proceedings of the 2013 IEEE 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing: Sensing the Future (ISSNIP '13)*, pp. 527–532, April 2013.

[26] Z. J. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *Proceedings of the In Workshop on Data Mining Applications in Sustainability (SIGKDD)*, pp. 59–62, San Diego, CA, USA, 2007.

[27] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *SDM*, vol. 11, pp. 747–758, 2011.

[28] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial HMMs with application to energy disaggregation," *Journal of Machine Learning Research*, vol. 22, pp. 1472–1482, 2012.

[29] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence and the 24th Innovative Applications of Artificial Intelligence Conference*, pp. 356–362, July 2012.

[30] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 23–31, 2012.

[31] N. Batra, J. Kelly, O. Parson et al., "NILMTK: an open source toolkit for non-intrusive load monitoring," in *Proceedings of the 5th ACM International Conference on Future Energy Systems (e-Energy '14)*, pp. 265–276, ACM, Cambridge, UK, June 2014.

[32] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Machine Learning*, vol. 29, no. 2-3, pp. 245–273, 1997.

[33] A. F. Arguimbau, *Algorithms for energy disaggregation [Msc. thesis]*, Universitat Politecnica de Catalunya, 2016.

[34] J. Huang, Z. Zhang, Y. Li, Z. Peng, and J. H. Son, "Energy disaggregation via hierarchical factorial hmm," in *Proceeding of The Second International Workshop on NILM*, 2014.

[35] J. R. Norris, *Markov Chains*, Cambridge University Press, Cambridge, UK, 1998.

[36] M. A. Zinkevich, M. Weimer, A. Smola, and L. Li, "Parallelized stochastic gradient descent," in *Proceedings of the 24th Annual Conference on Neural Information Processing Systems 2010 (NIPS '10)*, pp. 2595–2603, December 2010.

[37] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, USA, 2006.

[38] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," *Scientific Data*, vol. 2, Article ID 150007, 2015.

[39] R. Kohavi and F. Provost, "Guest editors' introduction: on applied research in machine learning," *Machine Learning*, vol. 30, no. 2-3, pp. 111-112, 1998.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 7643065, 11 pages
https://doi.org/10.1155/2017/7643065

Research Article

Joint Extraction of Entities and Relations Using Reinforcement Learning and Deep Learning

Yuntian Feng, Hongjun Zhang, Wenning Hao, and Gang Chen

Institute of Command Information System, PLA University of Science and Technology, Nanjing, Jiangsu 210007, China

Correspondence should be addressed to Yuntian Feng; fengyuntian2009@live.cn

Received 23 January 2017; Revised 10 April 2017; Accepted 21 May 2017; Published 14 August 2017

Academic Editor: Athanasios Voulodimos

We use both reinforcement learning and deep learning to simultaneously extract entities and relations from unstructured texts. For reinforcement learning, we model the task as a two-step decision process. Deep learning is used to automatically capture the most important information from unstructured texts, which represent the state in the decision process. By designing the reward function per step, our proposed method can pass the information of entity extraction to relation extraction and obtain feedback in order to extract entities and relations simultaneously. Firstly, we use bidirectional LSTM to model the context information, which realizes preliminary entity extraction. On the basis of the extraction results, attention based method can represent the sentences that include target entity pair to generate the initial state in the decision process. Then we use Tree-LSTM to represent relation mentions to generate the transition state in the decision process. Finally, we employ Q-Learning algorithm to get control policy π in the two-step decision process. Experiments on ACE2005 demonstrate that our method attains better performance than the state-of-the-art method and gets a 2.4% increase in recall-score.

1. Introduction

Information extraction [1] is the task of automatically extracting entities, relations, and events from unstructured texts. Researchers usually do research on entity extraction, relation extraction, and event extraction as separated tasks, but in fact there are important dependencies among tasks. For instance, entity information can further help relation extraction, so relation extraction takes the results of entity extraction as input. If just using a pipelined approach to tackle the above problem, information from each task cannot interact and get any feedback. Therefore, we make a detailed study of joint extraction of entities and relations from unstructured texts, which can pass the information of entity extraction to relation extraction and obtain feedback in order to improve the performance of entity extraction and relation extraction simultaneously.

In recent years, more and more researchers have applied deep learning to entity extraction and relation extraction. Huang et al. [2] proposed a bidirectional LSTM with a CRF layer (BILSTM-CRF) for sequence tagging, which included part-of-speech tagging (POS), chunking, and named entity recognition (NER). Nguyen and Grishman [3] proposed to combine the traditional feature-based method and the convolutional and recurrent neural networks for relation extraction. Deep learning can automatically extract features of entities and relations between entities to replace the method of designing features manually. It reduces the dependence of external resources and achieves good performance.

But how to pass entity information to relation extraction and obtain feedback is the research focus to the task of joint extraction of entities and relations, which means that we need an effective combination of different deep learning methods. To tackle the problem, we use reinforcement learning to model the task as a two-step decision process. Because it is difficult to find some measures to directly represent the state from unstructured texts, we use some deep learning methods to extract the state in the process. Firstly, we regard entity extraction as a sequence tagging task and use bidirectional LSTM to capture the context information, which preliminarily realizes the tagging of entity state. On the basis of preliminary results, we use attention based method to represent the sentences that include target entity pair and generate the initial state s_1 in the decision process, where the

Table 1: A sentence in ACE2005 dataset.

Sentence	While either divesting or inviting third parties to take a minority stake in the remaining Entertainment assets.	
Entity ID = "AFP_ENG_20030319.0879-E24"	Type = "ORG" Subtype = "Commercial"	third parties
Entity ID = "AFP_ENG_20030319.0879-E25"	Type = "ORG" Subtype = "Entertainment"	Entertainment
Relation ID = "AFP_ENG_20030319.0879-R2"	Type = "ORG-AFF" Subtype = "Investor-Shareholder"	RefID = "AFP_ENG_20030319.0879-E24" Role = "Arg-1" RefID = "AFP_ENG_20030319.0879-E25" Role = "Arg-2"

first decision is made. Then we use Tree-LSTM to capture the most important information of relation mentions and generate the transition state s_2, where the second decision is made. The meaning of the two-step decision is as follows: the first decision is to judge if a sentence that includes target entity pair is a relation mention according to the preliminary results of entity extraction; the second decision is to classify the relation mention into a certain targeted type. By designing the reward function per step, entity information and relation information can interact. Finally, we use Q-Learning to get control policy π by maximizing cumulative rewards through a sequence of actions, which is essentially the mapping from state to action. In the training process of Q-Learning, all the parameters are jointly updated, which helps to realize the joint extraction of entities and relations. We conduct experiments on ACE2005 dataset and achieve better recall-score of both entity mentions and relation mentions than the state-of-the-art method. In the following, we define the task in Section 2 and present our method in Section 3. Then we detail an extensive evaluation in Section 4 and finally conclude in Section 5.

2. Task Definition

Our task is to extract all the entities and relations from unstructured texts simultaneously. In the section we randomly pick a sentence from ACE2005 dataset to analyze. The entity mentions and relation mention in the sentence are shown in Table 1, where Entity ID, Relation ID, and RefID are the identifications of mentions.

Entity Extraction. It can be taken as a sequence tagging task, which assigns a tag to each word s_t in the input sequence $S = [s_1, s_2, \ldots, s_n]$. The tag of a word means a combination of the entity type it belongs to and the boundary type it locates within. The boundary types are the Beginning, Inside, Last, Outside, and Unit of an entity (BILOU scheme). Table 1 shows two entity mentions in the sentence. The first entity mention is "third parties," and its entity type is "ORG." The second entity mention is "Entertainment," and its entity type is "ORG." ACE2005 dataset defines 7 coarse-grained entity types, which are "PER" (Person), "ORG" (Organization), "LOC" (Location), "GPE" (Geo-Political Entities), "FAC" (Facility), "VEH" (Vehicle), and "WEA" (Weapon). The types all have their own different subtypes.

Relation Extraction. It is to extract semantic relations of the targeted types between a pair of entities. Table 1 shows one relation mention in the sentence, of which the relation type is "ORG-AFF." The first entity argument is "third parties," and the second entity argument is "Entertainment." The order of the arguments cannot be changed, which means the relation type is with direction. ACE2005 dataset defines 7 coarse-grained relation types between entities, which are "PHYS" (Physical), "PART-WHOLE" (Part-Whole), "PER-SOC" (Person-Social), "ORG-AFF" (Org-Affiliation), "ART" (Artifact), "GEN-AFF" (Gen-Affiliation), and "METONYMY" (Metonymy). Similarly, the types all have their own different subtypes.

Joint Extraction. It is to extract entities and relations in a sentence simultaneously. In the process of extraction, entity information and relation information can interact and get feedback information. Therefore, the joint extraction is more practical and different than separated entity extraction and separated relation extraction. We define and conduct research on the joint extraction task and present to use both reinforcement learning and deep learning for the task in the following section.

3. Our Method

The section combines three deep learning methods in the decision process of reinforcement learning for the joint extraction task. Firstly, we describe the two-step decision process; then we expound three deep learning methods used in this paper, that are bidirectional LSTM, attention mechanism, and Tree-LSTM; finally, we introduce Q-Learning algorithm that can get control policy π.

3.1. Reinforcement Learning. In general, entity extraction is performed before relation extraction, and its results can also be taken as the input of relation extraction. Relation extraction is fundamentally divided into two stages: judge if a sentence that includes target entity pair is a relation mention; classify the relation mention into a targeted type. According to the thoughts, we model the joint extraction task as a two-step decision process by reinforcement learning. The two steps correspond to entity extraction and relation extraction roughly, and the specific flow is shown in Figure 1.

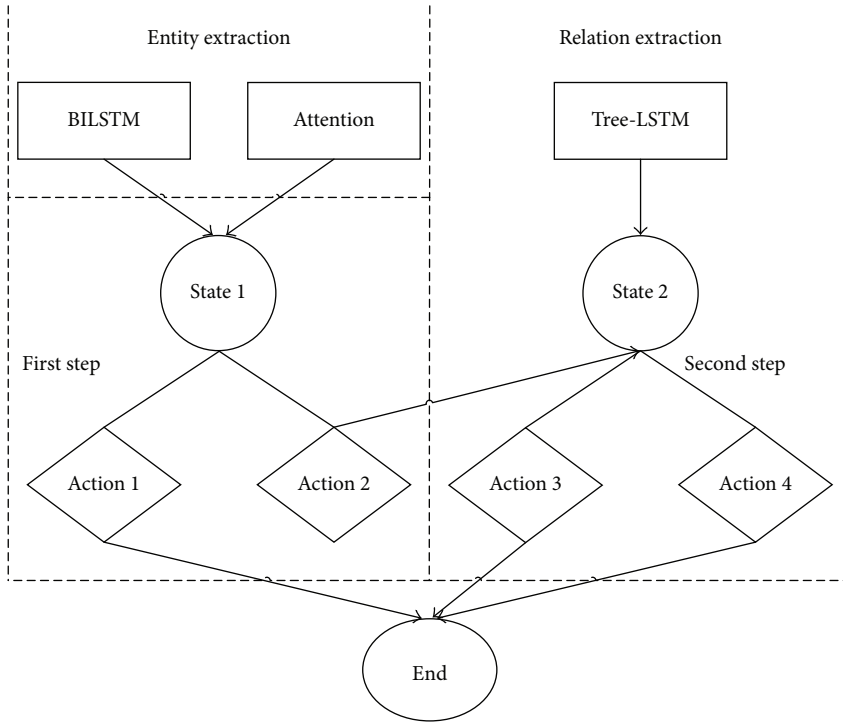


Figure 1: Two-step decision process.

Reinforcement Learning (RL). It [4] is a commonly used framework for learning control policies by the agent, through interacting with its environment.

State. The internal state S in the environment consists of the initial state s_1, the transition state s_2, and the end state s_e. Because it is difficult to find some appropriate measures to directly represent the state from unstructured texts, we use some deep learning methods to automatically extract features of texts, which can represent the state in the decision process. To be specific, we use bidirectional LSTM (Section 3.2) to realize preliminary entity extraction and use attention based method (Section 3.3) to generate the initial state $s_1 = \text{Att}(X; \boldsymbol{\theta}_1)$. In addition, we use Tree-LSTM (Section 3.4) to generate the transition state $s_2 = \text{Tree}(X; \boldsymbol{\theta}_2)$. The action taken at s_2 realizes preliminary relation extraction. X is the features of the input sentence; $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are parameters in the above models.

Action. There are a set of predefined actions A in the environment: Action 1 a_1, Action 2 a_2, Action 3 a_3, Action 4 a_4, and so forth. The first decision judges to take a_1 or a_2. a_1 is to judge that a sentence that includes target entity pair is not a relation mention, and a_2 is to judge that a sentence that includes target entity pair is a relation mention. The second decision judges to take a_3 or $a_4 \ldots$. a_3 is to classify the relation mention into a targeted type, and a_4 is to classify the relation mention into another targeted type. $R = r_1, r_2, r_3, r_4, \ldots$ denotes the reward obtained for each action. The agent takes an action a in state s and receives a reward r from the environment. (s_1, a_1, r_1, s_e), (s_1, a_2, r_2, s_2), (s_2, a_3, r_3, s_e), and (s_2, a_4, r_4, s_e) denote the transitions of the decision process.

Transition and Reward Function. A state transition tuple (s_1, a_1, r_1, s_e) means that the agent takes a_1 at s_1 and then transits to s_e. If the judgement of a_1 is right, then the agent receives a reward $r_1 = 10$; if the judgement of a_1 is wrong, then set $r_1 = -20$ to punish the wrong judgement of the first decision. A state transition tuple (s_1, a_2, r_2, s_2) means that the agent takes a_2 at s_1, then transits to s_2, and receives a reward $r_2 = 5$. A state transition tuple (s_2, a_3, r_3, s_e) means that the agent takes a_3 at s_2 and then transits to s_e. If the judgement of a_3 is right, then the agent receives a reward $r_3 = 10$; if the judgement on type is wrong, then set $r_3 = -10$ to punish the wrong judgement of the second decision; if it is not a relation mention, then set $r_3 = -20$. The meaning of other state transition tuple (s_2, a_4, r_4, s_e) and the definition of its reward function are similar to those of (s_2, a_3, r_3, s_e).

3.2. BILSTM. Long Short-Term Memory (LSTM) [5] is a variant of recurrent neural networks (RNN) designed to cope with the gradient vanishing problem, and LSTM is very useful to find and exploit long range dependencies in the data. Now lots of LSTM variants have been proposed and applied to natural language processing tasks, such as sentiment analysis, relation classification, and question answering system. We use bidirectional LSTM (BILSTM) to model word sequence, which can efficiently make use of past features and future features. BILSTM finds the right representation of each word and assigns a tag of entity state to each word in the input sequence to realize preliminary entity extraction. BILSTM mainly consists of three representation layers: embedding layer, BILSTM layer, and output layer. Figure 2 gives the basic structure of the BILSTM.

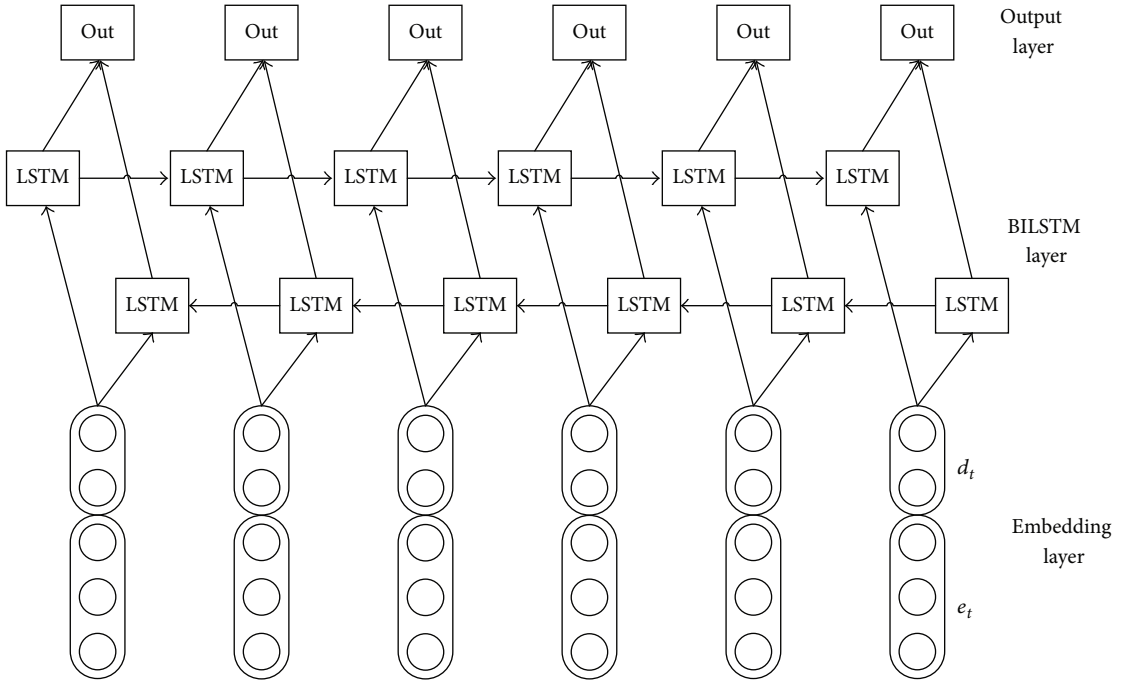


Figure 2: Basic structure of BILSTM.

3.2.1. Embedding Layer. The embedding layer converts discrete features of each word into continuous features as input of the BILSTM layer. We do forward and backward for input sentence, so we need a special treatment at the beginning and the end of the sequence.

Part-of-speech feature can further help entity extraction, so we only use word embedding e_t and part-of-speech embedding d_t to represent each word w_t in the input sentence, which replace the method of designing features manually. After passing through the lookup table, the lowercased word is mapped to its corresponding embedding. For word feature, the lookup table is initialized by the publicly available word embeddings. For part-of-speech feature, the lookup table is randomly initialized with values drawn from a uniform distribution. The word embeddings and the part-of-speech embeddings are allowed to be modified during training.

We concatenate the word embedding e_t and the part-of-speech embedding d_t of each word w_t to generate input feature vector $x_t = [e_t, d_t]$. The matrix $X = [x_1, x_2, \ldots, x_n]$ represents the features of the whole sentence, and is passed to the BILSTM layer, where n is the length of the input sentence.

3.2.2. BILSTM Layer. Basically, each LSTM unit in the BILSTM layer is composed of three multiplicative gates: an input gate i_t, a forget gate f_t, and an output gate o_t. The gates can control the proportions of information to forget and to pass on to the next time step. In addition, there is a memory cell c_t in each LSTM unit, which can keep the previous state and memorize the features of the current input word. Therefore, the data sources of each LSTM unit are as follows: the feature vector $x_t = [e_t, d_t]$ at time t, the hidden state vector h_{t-1} before time t or h_{t+1} after time t, and the cell vector c_{t-1}. The forward passes are implemented as follows:

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right), \\ g_t &= \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + W_{cc}c_{t-1} + b_c\right), \\ c_t &= i_t g_t + f_t c_{t-1}, \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right), \\ h_t &= o_t \tanh\left(c_t\right), \end{aligned} \tag{1}$$

where W are weight matrices, b are bias vectors, and their subscripts have the meaning as the name suggests. σ denotes the logistic function.

The backward passes over time are carried out in a similar way to forward passes. The hidden state vectors of two directions h_t and h'_t are simultaneously computed at time t in the BILSTM layer, so we can efficiently make use of past features and future features for a specific time frame.

3.2.3. Output Layer. We treat entity extraction as a sequence labeling task. By assigning an entity tag to each word, we realize preliminary entity extraction on top of the BILSTM layer. At time t, we pass the hidden state vectors of two directions h_t and h'_t to a softmax layer.

$$y_t = \text{softmax}\left(W_{hy}h_t + W_{h'y}h'_t + b_y\right). \tag{2}$$

Here, W are weight matrices and b is bias vector.

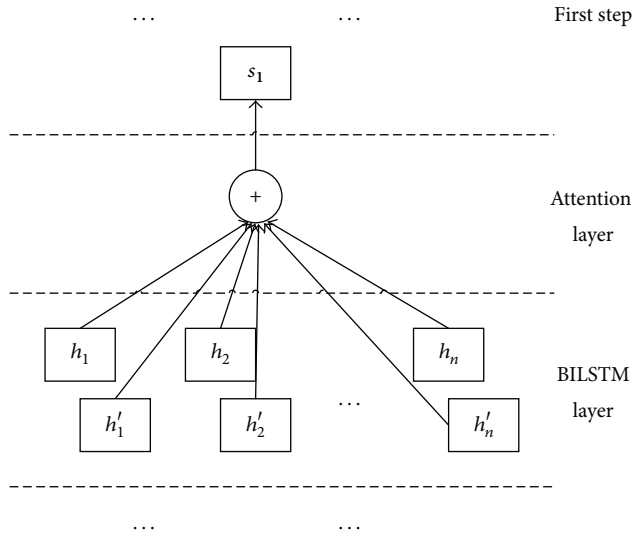


Figure 3: Attention layer.

3.2.4. Objective Function. We employ the Viterbi algorithm to inference the tag sequence $T = [t_1, t_2, \ldots, t_n]$ for a given input sentence $W = [w_1, w_2, \ldots, w_n]$. To model the tag dependency, we use the transition score $\mathbf{A}_{ij}$ for measuring the probability of the transformation from tag i to tag j. Thus, the sentence-level score can be formulated as follows:

$$s(W, T, \boldsymbol{\theta}_0) = \sum_{i=1}^{n} \left(\mathbf{A}_{t_{i-1}t_i} + y_i(t_i)\right). \tag{3}$$

Here, $y_i(t_i)$ is the score for choosing tag t_i for the ith word in the input sentence. $\boldsymbol{\theta}_0$ is the parameter set of BILSTM.

For a given training instance (W_i, T_i), W_i is a given sentence and the correct tag sequence for W_i is T_i. We search for the tag sequence with the highest score:

$$T^* = \underset{\widehat{T}}{\operatorname{argmax}}\, s\left(W_i, \widehat{T}, \boldsymbol{\theta}_0\right). \tag{4}$$

Here, $\widehat{T}$ is a predicted tag sequence.

The regularized objective function for m training instances is the loss function $J(\boldsymbol{\theta}_0)$ including a l_2-norm term:

$$\begin{aligned} J(\boldsymbol{\theta}_0) &= \frac{1}{m}\sum_{i=1}^{m} l_i(\boldsymbol{\theta}_0) + \frac{\lambda}{2}\left\|\boldsymbol{\theta}_0\right\|_2^2, \\ l_i(\boldsymbol{\theta}_0) &= \max\left(0, s\left(W_i, \widehat{T}_i, \boldsymbol{\theta}_0\right) + \Delta\left(T_i, \widehat{T}_i\right) - s\left(W_i, T_i, \boldsymbol{\theta}_0\right)\right). \end{aligned} \tag{5}$$

Here, $\Delta(T_i, \widehat{T}_i)$ is a structured margin loss for predicted tag sequence $\widehat{T}$. λ is an $L2$ regularization hyperparameter.

To minimize $J(\boldsymbol{\theta}_0)$, we use a generalization of gradient descent called subgradient method [6] which computes a gradient-like direction.

3.3. Attention Mechanism. Recently, attention mechanisms have successfully been applied to machine translation [7], text summarization [8], text comprehension [9], syntactic constituency parsing [10], relation classification [11], and text classification [12]. Inspired by those studies, we introduce attention based method to compute the hidden state vectors h_t and h'_t in the BILSTM layer and generate the initial state s_1 in the decision process. The method can obtain the information of entity extraction and represent the sentences that include target entity pair. After the first decision on s_1, we realize preliminary entity extraction and get ready to perform relation extraction. In essence, attention based method can pass entity information to relation extraction and obtain feedback information of relation extraction by jointly updating all the parameters. Attention based method better integrates entity extraction and relation extraction.

After realizing preliminary entity extraction, we choose two entities as target entity pair in the sentence $W = [w_1, w_2, \ldots, w_n]$. The attention layer is depicted in Figure 3. Let H be a matrix consisting of the hidden state vectors $[h_1, h'_1, h_2, h'_2, \ldots, h_n, h'_n]$ in the BILSTM layer, and H is the input of the attention layer. Then attention based method represents the sentence that includes target entity pair as a weighted sum of these hidden state vectors.

$$\begin{aligned} A &= \tanh(H), \\ \alpha &= \operatorname{softmax}\left(\omega^T A\right), \\ s_1 &= \tanh\left(H\alpha^T\right). \end{aligned} \tag{6}$$

Here, α is the normalized weight vector and ω is a parameter vector. s_1 is the initial state, in which we denote by $s_1 = \operatorname{Att}(X; \boldsymbol{\theta}_1)$, and $\boldsymbol{\theta}_1$ represents all the parameters in this method.

After generating the initial state s_1, the first decision will be made to judge if a sentence that includes target entity pair is a relation mention. We pass s_1 to a softmax output layer to get y_a, which is the probability of relation mention and

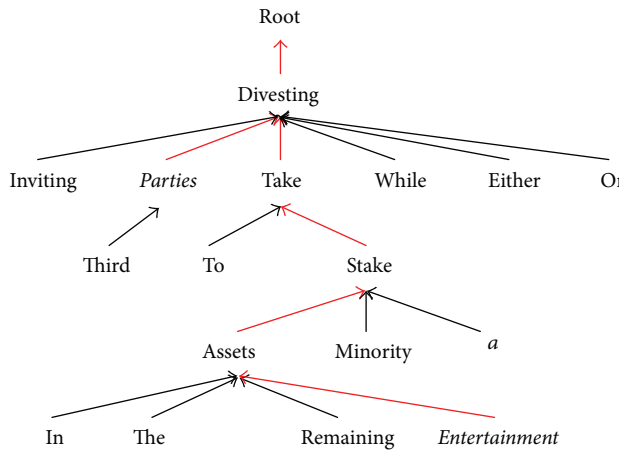


Figure 4: Dependency tree of a relation mention.

nonrelation for a sentence. Finally, we can determine to take a_1 or a_2.

$$y_a = \text{softmax}\left(W_{sy}s_1 + b_y\right). \tag{7}$$

Here, W is weight matric and b is bias vector.

The objective function for m training instances is the negative log-likelihood:

$$J\left(\boldsymbol{\theta}_1\right) = -\frac{1}{2m}\sum_{i=1}^{m} t_0^{(i)} \log\left(y_a^{(i)}(0)\right) + t_1^{(i)} \log\left(y_a^{(i)}(1)\right) + \frac{\lambda}{2}\left\|\boldsymbol{\theta}_1\right\|_2^2 \tag{8}$$

Here, $t_0^{(i)}$ and $t_1^{(i)}$ are the one-hot represented ground truth. $y_a^{(i)}(0)$ and $y_a^{(i)}(1)$ are the estimated probability for relation mention and nonrelation, respectively. λ is an $L2$ regularization hyperparameter.

To minimize $J(\boldsymbol{\theta}_1)$, we use a simple optimization technique called stochastic gradient descent (SGD).

3.4. Tree-LSTM. Unlike traditional sequence LSTM, Tree-LSTM [13] is constructed over a tree structure. As is known to all, the dependency tree is very useful for analyzing the relations between words. Two words may be far apart in the linear structure and separated by many unrelated words or preposition structure, but they are in hyponymy for the dependency tree. Therefore, we construct the Tree-LSTM over the dependency tree to represent relation mentions in a bottom-up way. Tree-LSTM can extract the core dependency relation between target entity pair and generate the transition state s_2 in the decision process. The second decision on s_2 performs preliminary relation extraction.

We take the relation mention "AFP_ENG_20030319.0879-R2" in Table 1 as an example to illustrate, and the two entity arguments are "third parties" and "Entertainment." Firstly, we perform dependency parsing on the relation mention and generate the dependency tree, as shown in Figure 4. Instead of using the full mention boundary, we use head spans for entities directly. The entity head of "third parties" is "parties," and the entity head of "Entertainment" is "Entertainment." The core dependency relation between target entity pair is shown by red lines in Figure 4. So we use dependency tree as a backbone to construct Tree-LSTM. Moreover, for the convenience of implementation, we prune or pad dependency trees to keep the same depth and width.

Like BILSTM, each LSTM unit of Tree-LSTM takes continuous feature vector of a word as input. In addition to word embedding e_t and part-of-speech embedding d_t, we use entity type embedding t_t and entity position embedding l_t, to which entity type feature and entity position feature are mapped. We can get the entity type features from the preliminary results of entity extraction and get the entity position features by computing the relative distances of the current word to the two entity arguments. Unlike BILSTM, the LSTM unit does not accept hidden state vectors of the adjacent words and accept the hidden state vectors of all children nodes h_{tk} as input. The Tree-LSTM is developed from its leaf node in a recursive way up to the root, which is the common ancestor ("divesting" in Figure 4) of all the words. Then we carry out nonlinear transformation on the hidden state vector of the ancestor to generate s_2, which is the final representation of relation mentions and serves as the transition state in the decision process. We denote s_2 by $s_2 = \text{Tree}(X; \boldsymbol{\theta}_2)$, and $\boldsymbol{\theta}_2$ represents all the parameters in the Tree-LSTM.

After generating the transition state s_2, the second decision will be made to classify the relation mention into a targeted type. Then s_2 is passed to a softmax output layer to get y_r, which is the probability of different types for a relation mention. Finally, we choose a type with the maximum probability, which determines to take a_3 or $a_4 \ldots$.

$$y_r = \text{softmax}\left(W_{sy}s_2 + b_y\right). \tag{9}$$

Here, W is weight matric and b is bias vector. At each dependency tree, we use a softmax layer to predict the type for the root node given the inputs X observed at its children nodes.

Initialize BILSTM, the Attention Layer, and Tree-LSTM with random parameters
$\boldsymbol{\eta} = \mathbf{0}$
Pre-train BILSTM, the Attention Layer, and Tree-LSTM respectively
for epoch = 1, 2, ... **do**
  for each input sentence X **do**
    Use the deep learning models above to automatically extract features of X, and generate s_1 and s_2.
    for $t = 1, 2$ **do**
      r, s' = the reward and state after taking the action $\pi(s)$
      $a' = \pi(s')$
      Perform gradient descent step:
      $$-\frac{\partial E_{\boldsymbol{\eta}}}{\partial \boldsymbol{\eta}} = E\left[2\left(Q_{\pi}(s,a) - Q_{\boldsymbol{\eta}}(s,a)\right)\frac{\partial Q_{\boldsymbol{\eta}}(s,a)}{\partial \boldsymbol{\eta}}\right]$$
      $$Q_{\pi}(s,a) = \frac{1}{t}r + \frac{t-1}{t}Q^{\pi}(s',a')$$
      The update rule is
      $$\boldsymbol{\eta} = \boldsymbol{\eta} + \alpha\left(\frac{1}{t}r + \frac{t-1}{t}Q_{\boldsymbol{\eta}}(s',a') - Q_{\boldsymbol{\eta}}(s,a)\right)\frac{\partial Q_{\boldsymbol{\eta}}(s,a)}{\partial \boldsymbol{\eta}}$$
      Where α is update step, and r is the reward function (Section 3.1), and (s', a') is the state-action pair of next time.
      $\pi(s) = \underset{a''}{\operatorname{argmax}}\, Q_{\boldsymbol{\eta}}(s, a'')$
      $s = s', a = a'$
    end for
  end for
end for

Algorithm 1: Training procedure for Q-Learning.

The objective function for m training instances is the negative log-likelihood:

$$J(\boldsymbol{\theta}_2) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(y_r^{(i)}\right) + \frac{\lambda}{2}\left\|\boldsymbol{\theta}_2\right\|_2^2. \tag{10}$$

Here, $y_r^{(i)}$ is the estimated probability for the true type at each root node. The root node of Tree-LSTM is able to selectively incorporate information from each child. λ is an $L2$ regularization hyperparameter.

To minimize $J(\boldsymbol{\theta}_2)$, we use AdaGrad [14].

3.5. Q-Learning. Q-Learning algorithm [15] is a popular form of reinforcement learning and can be used to learn an optimal state-action value function $Q(s, a)$ for the agent. The agent takes an action a in state s by consulting $Q(s, a)$, which is a measure of the action's expected long-term reward. The aim is to maximize some cumulative rewards through a sequence of actions. As the state space is infinite in the decision process, it is impractical to obtain $Q(s, a)$ for all possible state-action pairs.

For the above challenge, we approximate $Q(s, a)$ using a neural network, which can represent $Q(s, a)$ as a parameterized function $Q_{\boldsymbol{\eta}}(s, a) = \mathrm{MLP}(\phi(X; \boldsymbol{\theta}), a; \boldsymbol{\eta})$. $\phi(X; \boldsymbol{\theta})$ refers to $s_1 = \mathrm{Att}(X; \boldsymbol{\theta}_1)$ and $s_2 = \mathrm{Tree}(X; \boldsymbol{\theta}_2)$ above, where $\boldsymbol{\theta}$ can be obtained by pretraining the deep learning models above and $\boldsymbol{\eta}$ represents the parameters in the neural network, which are learnt by performing stochastic gradient descent step with RMSprop [16].

To approximate the real value function Q^{π} as closely as possible, we measure the degree of approximation with the least squares error:

$$E_{\boldsymbol{\eta}} = E\left[\left(Q_{\pi}(s,a) - Q_{\boldsymbol{\eta}}(s,a)\right)^2\right]. \tag{11}$$

In Q-Learning, we use the estimated value function $Q_{\boldsymbol{\eta}}(s, a)$ instead of the real value function $Q^{\pi}(s, a)$. During each epoch, the updates of parameters aim to reduce the discrepancy between the estimation $Q_{\boldsymbol{\eta}}(s, a)$ and the expectation $Q^{\pi}(s, a)$. The agent starts from a random $Q_{\boldsymbol{\eta}}(s, a)$ and continuously updates its values by making the decisions and obtaining rewards. Then the agent can maximize its expected future rewards by choosing the action with the highest $Q_{\boldsymbol{\eta}}(s, a'')$. Finally, Q-Learning algorithm gets control policy π in the two-step decision process. Algorithm 1 details the Q-Learning training procedure.

During the training procedure we pretrain BILSTM, the attention layer, and Tree-LSTM, respectively. The training parameters mainly include all the parameters $\boldsymbol{\theta}_0$ in BLSTM, all the parameters $\boldsymbol{\theta}_1$ in the attention layer, and all the parameters $\boldsymbol{\theta}_2$ in Tree-LSTM.

The functionality of the attention model in our RL method is very similar to that of a separate relation mention classification part in a pipeline. We use deep learning methods to represent words and sentences in the text and use RL to combine three tasks in the decision process, that are entity extraction, relation mention classification, and relation classification. The pipeline architecture just passes the information of entity extraction to relation extraction and does not enable information to flow in the global architecture.

However, our RL method not only combines the above tasks sequentially but also globally makes decisions. At the beginning, the decisions have close to a random chance. After several epochs, they will be stabilizing. Meanwhile, the parameters in our architecture are globally updated and eventually converge. Therefore, our RL method can obtain feedback from decision-making and state changes and enable information to flow in the global architecture. The attention model connects entity extraction task with relation extraction task, thus helping us to realize the joint extraction of entities and relations. Experimental results demonstrate that our RL method performs slightly better than the pipeline method for both entity extraction and relation extraction, which shows that we are on the right track.

4. Experiments

4.1. Data. Most previous work has reported results on ACE2005 data set, so we evaluate our method on ACE2005 for joint extraction of entities and relations. We use three common metrics to evaluate the performance: microprecision (P), recall (R), and primary micro $F1$-scores ($F1$). An entity mention is correct when its entity type and the region of its head are correct, and a relation mention is correct when its relation type and both entity arguments are correct.

Data source for English in ACE2005 is as follows: 20% Newswire (NW), 20% Broadcast News (BN), 15% Broadcast Conversation (BC), 15% Weblog (WL), 15% Usenet Newsgroups/Discussion Forum (UN), and 15% Conversational Telephone Speech (CTS). The two small subsets UN and CTS are informal, so we remove them. In addition, in order to compare with state of the art, we employ the same method as previous work [17] to split and preprocess the data. Training set contains 351 documents, development set contains 80 documents, and testing set contains 80 documents.

4.2. Hyperparameters. We set up Python2.7 + Theano + Cuda7.5 environments to implement our method. We use the publicly available word embedding Glove [18] to initialize the word embedding table, and its dimension n_e is 300. We fix the dimension of part-of-speech embedding n_d and the dimension of entity type embedding n_t to 50 and fix the dimension of entity position embedding n_l to 5. Those feature embeddings are randomly initialized and allowed to be modified during training. In addition, we fix the state size of all the LSTM units to 200 and fix the dimensions of other hidden layers to 100. We use tanh for the nonlinear function.

We tune hyperparameters using development set to achieve high $F1$. The best hyperparameters are as follows. Dropout rate [19] is 0.5, minibatch size is 30, the constraint of max-norm regularization is equal to 3, and initial learning rate is 0.0005. The reward after each action is described in the Section 3.1. Therefore, for all the experiments below, we will directly employ the best hyperparameters.

4.3. Overall Performance. We run experiments to analyze the effectiveness of the various components of our joint extraction method.

Firstly, we compare the performance of BILSTM with a baseline system, LSTM for entity extraction task. We train models using training set and report models' performance on development set in Table 2. The result shows that BILSTM obtains better performance than LSTM on all evaluation metrics. Bidirectional model can actually improve the performance of sequence tagging task. Therefore, throughout the experiment, we will use BILSTM to extract entities.

Table 2: Performance for entity extraction task.

Method	Entity		
Score	P (%)	R (%)	$F1$ (%)
LSTM	81.0	78.1	79.5
BILSTM	82.5	79.8	81.1

Table 3: Performance for relation extraction task.

Method	Relation		
Score	P (%)	R (%)	$F1$ (%)
CNN	63.1	52.9	57.6
Tree-LSTM	63.9	54.1	58.6
RL	63.6	59.4	61.4

Then, to demonstrate the effectiveness of the relation extraction component of our method, we carry out experiments on relation extraction when entities are known. We build a baseline system, CNN. In addition, we parse relation mentions using the Stanford neural dependency parser [20] and directly use Tree-LSTM extract relations. On the basis of Tree-LSTM, we use reinforcement learning method to control the process of relation extraction. We compare the performance of the above three methods on development set in Table 3. The result demonstrates that Tree-LSTM is better suited to extract relations than CNN, and reinforcement learning method obtains a substantial gain in recall-score over Tree-LSTM with 3.7%. Therefore, in the rest of the experiment, we will use reinforcement learning method based on Tree-LSTM to extract relations.

Finally, we demonstrate the effectiveness of our joint extraction method. We build a pipelined system, which directly connects the entity extraction component and the relation extraction component above. To be specific, the pipelined system first trains the entity extraction model and then builds a separate relation extraction model using the detected entities. Our joint system is based on the pipelined system. The joint system uses attention based method to pass entity information to relation extraction and updates the parameters in all the components simultaneously during the training procedure for Q-Learning, which realizes the joint extraction of entities and relations. We compare the performance of the two systems on development set in Table 4. The result demonstrates that our joint system slightly improves the performance of entity extraction and significantly improves the performance of relation extraction. Therefore, the experiments show that our method is effective and practical.

We will clearly show the process of the above experiments. Figure 5 shows the average reward after each training epoch. At the beginning of training, the reward is negative, because the agent takes actions randomly. But with the increase

Table 4: Performance of two extraction systems.

Method	Entity			Relation		
Score	P (%)	R (%)	$F1$ (%)	P (%)	R (%)	$F1$ (%)
Pipeline	82.5	79.8	81.1	60.2	43.9	50.8
Joint	83.6	80.4	82.0	60.6	45.9	52.2

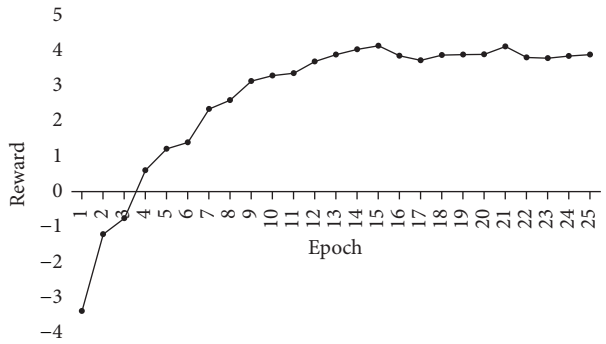


Figure 5: Learning curve of average reward.

of epoch number, the reward improves gradually. Figure 6 shows the learning curves of the performance for entity extraction and relation extraction. The $F1$-score in both (a) and (b) increases simultaneously. From the two figures, we can clearly see that all the metrics significantly improve and then stabilize after 13 epochs of training. So we set the number of training epochs as 13.

4.4. Comparison with State of the Art. Now deep learning methods achieve state-of-the-art performance in end-to-end relation extraction task. Miwa and Bansal [21] stacked bidirectional tree-structured LSTM-RNNs on bidirectional sequential LSTM-RNNs to extract entities and relations between them, which could capture both word sequence and dependency tree substructure information. The method is denoted by SPTree. Table 5 compares our joint extraction method with SPTree on the testing set and shows that our method performs slightly better than SPTree for both entity mentions and relation mentions. Although our method is not comparable with SPTree in precision-score, our method outperforms the best results of SPTree in recall-score. The main reason is that the reward after each action in reinforcement learning may play an important role.

4.5. Analysis. We pretrain the attention model which is used for relation mention classification. Relation mention classification is always processed in a very unbalanced corpus, where most sentences are not a relation mention. From Figure 7, we see that the SGD algorithm gets to the minimum objective fast, but the objective function's value is a bit high. That means that during the pretraining of the attention model there would be a huge loss. The parameters in the attention layer are updated to accepted values, which are prepared for Q-Learning. When we do Q-Learning, we learn a stacked MLP on top of the attention model (without softmax output layer). From Figure 7, we see that Q-Learning takes more epochs to converge but reduces the value of the objective function in the first stage of the MDP. That means that our reinforcement learning method is effective despite the huge loss and poor initialization in the pretraining of the attention model. Moreover, Figure 8 shows the learning curves of the performance for relation mention classification. We can see that our reinforcement learning method gets good performance in the $F1$-score, which is also a proof of our effectiveness.

Table 5: Comparison with state of the art.

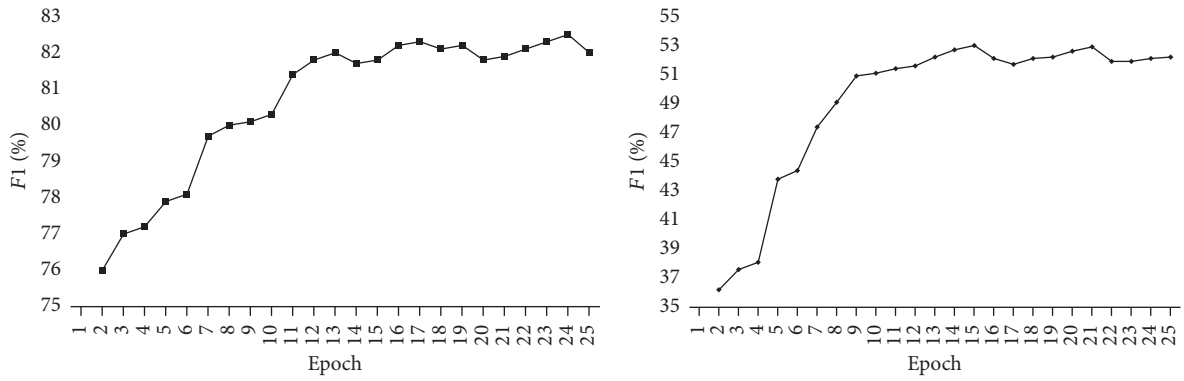
Method	Entity			Relation		
Score	P (%)	R (%)	$F1$ (%)	P (%)	R (%)	$F1$ (%)
SPTree	85.5	81.2	83.3	65.8	42.9	51.9
Joint	85.0	**82.4**	83.7	65.9	**45.3**	53.7

5. Related Work

As for joint extraction of entities and relations, the research has been dominated by four methods. The first one is structured prediction. Li and Ji [17] presented an incremental joint framework to simultaneously extract entity mentions and relations using structured perceptron with efficient beam-search. The second one is integer linear programming. Dan and Yih [22] studied global inference for entity and relation identification via a linear programming formulation. The third one is card-pyramid parsing. Kate and Mooney [23] presented a new method for joint entity and relation extraction using card-pyramid parsing. The last one is global probabilistic graphical models. Yu and Lam [24] jointly identified entities and extracted relations in encyclopedia text via a graphical model approach.

Recently, deep learning methods have been widely used in many research areas with the aim of reducing the number of handcrafted features. However, the only work of end-to-end (joint) extraction of relations between entities with deep learning methods is due to Miwa and Bansal [21], and most researchers simply solve entity extraction, relation classification, or relation extraction separately. Chiu and Nichols [25] presented a novel neural network architecture for named entity recognition, which automatically detected word- and character-level features using a hybrid bidirectional LSTM and CNN architecture. Zhang et al. [26] proposed bidirectional long short-term memory networks (BLSTM) to model the sentence with complete, sequential information about all words for relation classification. Nguyen and Grishman [27] departed from these traditional approaches with complicated feature engineering by introducing a convolutional neural network for relation extraction.

At present, the research of reinforcement learning has risen. El-Laithy and Bogdan [28] presented a reinforcement learning framework for spiking networks with dynamic synapses. Mousavi et al. [29] discussed the notion of context transfer in reinforcement learning tasks. However, few researchers apply reinforcement learning in text processing tasks. We use both reinforcement learning and deep learning to simultaneously extract entities and relations from unstructured texts. To the best of our knowledge, there has been no



(a) Performance of entity extraction

(b) Performance of relation extraction

Figure 6: Learning curves of the performance.

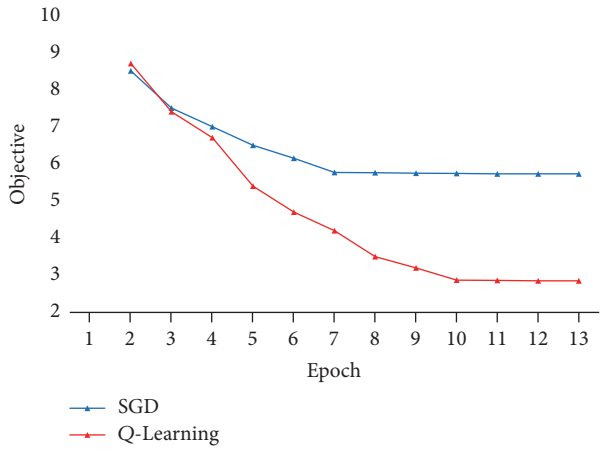


Figure 7: Objective values in the attention model.

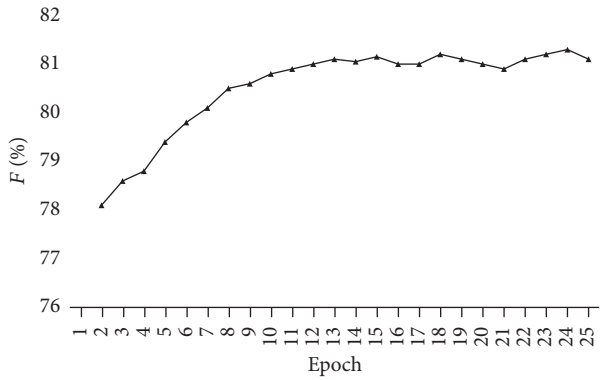


Figure 8: Performance of relation mention classification.

work on employing reinforcement learning for information extraction so far. This paper is the first attempt to fill in that gap and provides a good thinking way for future research in this area.

6. Conclusions

In this paper we define and research the joint extraction of entities and relations. We model the task as a two-step decision process in reinforcement learning. In addition, we use deep learning methods to represent the state in the decision process. Attention based method can pass entity information to relation extraction task. During the training procedure for Q-Learning, all the parameters are updated simultaneously to realize the interaction and feedback of entity information and relation information. The reward after each action in reinforcement learning apparently helps to improve the recall-score. Under the same experimental conditions, our method outperforms the state-of-the-art method in $F1$-score of entity mentions and relation mentions. In future work, we plan to perfect the model of the two-step decision process and optimize the Q-Learning algorithm.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] "Information Extraction: a multidisciplinary approach to an emerging information technology," Springer, Heidelberg, Germany, 1997.

[2] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, https://arxiv.org/abs/1508.01991v1.

[3] T. Nguyen H and R. Grishman, "Combining Neural Networks and Log-linear Models to Improve Relation Extraction," 2015, https://arxiv.org/abs/1511.05926.

[4] C. Amato and G. Shani, "High-level reinforcement learning in strategy games," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 75–82, International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "subgradient methods for structured prediction," *Journal of Machine Learning Research*, vol. 2, pp. 380–387, 2007.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, https://arxiv.org/abs/1409.0473.

[8] A. M. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization," 2015, https://arxiv.org/abs/1509.00685.

[9] A. M. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 379–389, Lisbon, Portugal, September 2015.

[10] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," *Advances in Neural Information Processing Systems*, pp. 2773–2781, 2015.

[11] L. Wang, Z. Cao, G. de Melo, and Z. Liu, "Relation classification via multi-level attention CNNs," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1298–1307, Berlin, Germany, August 2016.

[12] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 1480–1489, Human Language Technologies, June 2016.

[13] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, Beijing, China, July 2015.

[14] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2121–2159, 2011.

[15] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude," *Coursera: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.

[17] Q. Li and H. Ji, "Incremental Joint Extraction of Entity Mentions and Relations," 2014.

[18] J. Pennington, R. Socher, and C. D. Manning, "GloVe: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 1532–1543, October 2014.

[19] G. E. Hinton, N. Srivastava, and A. Krizhevsky, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, https://arxiv.org/abs/1207.0580.

[20] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 740–750, October 2014.

[21] M. Miwa and M. Bansal, "End-to-end Relation Extraction using LSTMs on Sequences and Tree Structures," 2016, https://arxiv.org/abs/1601.00770.

[22] R. Dan and W. T. Yih, Global Inference for Entity and Relation Identification via a Linear Programming Formulation, 2007.

[23] J. R. Kate and R. J. Mooney, "Joint entity and relation extraction using card-pyramid parsing," in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pp. 203–212, Association for Computational Linguistics, 2010.

[24] X. Yu and W. Lam, "Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach," in *Proceedings of the International Conference on Computational Linguistics*, vol. 23-27, pp. 1399–1407, Beijing , China, August 2010.

[25] P. C. J. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs," 2015, https://arxiv.org/abs/1511.08308.

[26] S. Zhang, D. Zheng, and X. Hu, Bidirectional Long Short-Term Memory Networks for Relation Classification, 2015.

[27] T. H. Nguyen and R. Grishman, "Relation extraction: perspective from convolutional neural networks," *The Workshop on Vector Space Modeling for Natural Language Processing*, pp. 39–48, 2015.

[28] K. El-Laithy and M. Bogdan, "A reinforcement learning framework for spiking networks with dynamic synapses," *Computational Intelligence and Neuroscience*, vol. 2011, Article ID 869348, 12 pages, 2011.

[29] A. Mousavi, B. N. Araabi, and M. N. Ahmadabadi, "Context transfer in reinforcement learning using action-value functions," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 428567, 2014.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 2917536, 8 pages
https://doi.org/10.1155/2017/2917536

Research Article

Automatic Image-Based Plant Disease Severity Estimation Using Deep Learning

Guan Wang, Yu Sun, and Jianxin Wang

School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

Correspondence should be addressed to Yu Sun; sunyv@bjfu.edu.cn and Jianxin Wang; wangjx@bjfu.edu.cn

Received 8 March 2017; Revised 9 May 2017; Accepted 4 June 2017; Published 5 July 2017

Academic Editor: Athanasios Voulodimos

Automatic and accurate estimation of disease severity is essential for food security, disease management, and yield loss prediction. Deep learning, the latest breakthrough in computer vision, is promising for fine-grained disease severity classification, as the method avoids the labor-intensive feature engineering and threshold-based segmentation. Using the apple black rot images in the PlantVillage dataset, which are further annotated by botanists with four severity stages as ground truth, a series of deep convolutional neural networks are trained to diagnose the severity of the disease. The performances of shallow networks trained from scratch and deep models fine-tuned by transfer learning are evaluated systemically in this paper. The best model is the deep VGG16 model trained with transfer learning, which yields an overall accuracy of 90.4% on the hold-out test set. The proposed deep learning model may have great potential in disease control for modern agriculture.

1. Introduction

The plant diseases are a major thread to losses of modern agricultural production. Plant disease severity is an important parameter to measure disease level and thus can be used to predict yield and recommend treatment. The rapid, accurate diagnosis of disease severity will help to reduce yield losses [1]. Traditionally, plant disease severity is scored with visual inspection of plant tissue by trained experts. The expensive cost and low efficiency of human disease assessment hinder the rapid development of modern agriculture [2]. With the population of digital cameras and the advances in computer vision, the automated disease diagnosis models are highly demanded by precision agriculture, high-throughput plant phenotype, smart green house, and so forth.

Inspired by the deep learning breakthrough in image-based plant disease recognition, this work proposes deep learning models for image-based automatic diagnosis of plant disease severity. We further annotate the apple healthy and black rot images in the public PlantVillage dataset [3] with severity labels. To explore the best network architecture and training mechanism, we train shallow networks of different depth from scratch and fine-tune the pretrained state-of-the-art deep networks. The models' capabilities of correctly predicting the disease severity stage are compared. The best model achieves an accuracy of 90.4% on the hold-out test set. Our results are a first step towards the automatic plant disease severity diagnosis.

An overview of the rest of the paper is as follows: Section 2 reviews the literature in this area, Section 3 presents the deep learning proposal, Section 4 describes the methodology, Section 5 presents achieved results and related discussions, and, finally, Section 6 holds our conclusions.

2. Related Work

Various studies have found that image-based assessment approaches produce more accurate and reproducible results than those obtained by human visual assessments. Stewart and McDonald [4] used an automated image analysis method to analyze disease symptoms of infected wheat leaves caused by *Zymoseptoria tritici*. This method enabled the quantification of pycnidia size and density, along with other traits and their correlation, which provided greater accuracy and precision compared with human visual estimates of virulence. Barbedo [5] designed an image segmentation method to measure disease severity in white/black background, which eliminated the possibility of human error and reduced time

taken to measure disease severity. Atoum et al. [6] proposed a novel computer vision system, Cercospora Leaf Spot (CLS) Rater, to accurately rate plant images in the real field to the United States Department of Agriculture (USDA) scale. The CLS Rater achieved a much higher consistency than the rating standard deviation of human experts. Many of these image-based assessment approaches for plant diseases share the same basic procedure [5–13]. Firstly, preprocessing techniques are employed to remove the background and segment the lesion tissue of infected plants. After that, discriminative features are extracted for further analysis. At last, supervised classification algorithms or unsupervised cluster algorithms are used to classify features according to the specific task. Along with advances in computer science, many interactive tools are developed. The Assess [14] is the most commonly used and also the discipline-standard program to estimate disease severity. The Leaf Doctor app [15], developed as an interactive smartphone application, can be used on color images to distinguish lesion areas from healthy tissues and calculate percentage of disease severity. The application achieved even higher accuracy than the Assess.

But these aforementioned plant disease severity estimation approaches are semiautomatic because they depend heavily on series of image-processing technologies, such as the threshold-based segmentation of the lesion area and hand-engineered features extraction. There is usually great variance in color both between lesions of different diseases and between lesions from the same disease at different stages. Therefore, it is very difficult to determine the appropriate segmentation threshold for plant disease images without human assistance. What is more, the time consuming hand-crafted feature extraction should be performed again for new style images. To the best of our knowledge, completely automatic image-based plant disease severity estimation method using computer vision has not yet been reported.

The deep learning approach leads a revolution in speech recognition [16], visual object recognition [17], object detection [18, 19], and many other domains such as drug discovery [20], genomics [21], and building reorganization [22]. Deep learning is very promising for automatically grading plant disease severity. Recently, there have been some works using deep learning method for plant species identification and plant disease identification. The recent years of the well-known annual plant species identification campaigns PlantCLEF [23] were performance-wise dominated by deep learning methods. Choi [24] won the PlantCLEF 2015 by using the deep learning model GoogleNet [25] to classify 1000 species. Mehdipour Ghazi et al. [26] combined the outputs of GoogleNet and VGGNet [27] and surpassed the overall validation accuracy of [24]. Hang et al. [28] won the PlantCLEF 2016 by the enhanced VGGNet model. For plant disease identification, Sladojevic et al. [29] created a dataset with more than 3,000 images collected from the Internet and trained a deep convolutional network to recognize 13 different types of plant diseases out of healthy leaves. Mohanty et al. [30] used a public dataset PlantVillage [3] consisting of 54,306 images of diseased and healthy plant leaves collected under controlled conditions and trained a deep convolutional neural network to identify 14 crop species and 26 diseases. In comparison with classification among different diseases, the fine-grained disease severity classification is much more challenging, as there exist large intraclass similarity and small interclass variance [31]. Deep learning avoids the labor-intensive feature engineering and threshold-based segmentation [32], which is promising for fine-grained disease severity classification.

3. Deep Learning Proposal

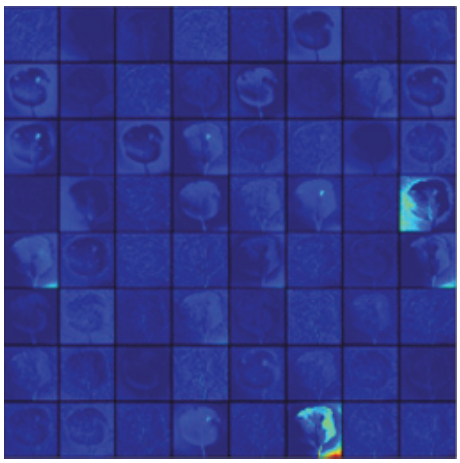
3.1. Deep Convolutional Neural Network. To explore the best convolutional neural network architecture for the fine-grained disease severity classification problem with few training data, we compare two architectures, namely, building a shallow network from scratch and transfer learning by fine-tuning the top layers of a pretrained deep network.

The shallow networks consist of only few convolutional layers with few filters per layer, followed by two fully connected layers, and end with a softmax normalization. We train shallow networks of 2, 4, 6, 8, and 10 convolutional layers. Each convolutional layer has 32 filters of size 3×3, a Rectified Linear Units (ReLU) activation, and all layers are followed by a 2×2 max-pooling layer, except for the last convolutional layer, which has 64 filters. The first fully connected layer has 64 units with a ReLU activation and is followed by a dropout layer with a dropout ratio of 50%. The last fully connected layer has 4 outputs, corresponding with the 4 classes, which feed into the softmax layer to calculate the probability output.

3.2. Transfer Learning. It is notable that the amount of images we can learn from is quite limited. Transfer learning is a useful approach to build powerful classification network using few data, by fine-tuning the parameters of a network pretrained on a large dataset, such as ImageNet [26]. Although the disease severity classification is targeted for finer grained image category classification problem compared to the ImageNet, the lower layers only encode simple features, which can be generalized to most computer vision tasks. For example, the first layer only represents direction and color, and the visualization of activations in the first layer of VGG16 model is shown in Figure 1. Though not trained on the plant disease dataset, the model can be activated against the diseased spots, the leaf, and the background.

For transfer learning, we compare the VGGNet [27], Inception-v3 [33], and ResNet50 [17] architectures. VGGNet and the original Inception architecture GoogleNet yielded similar high performance in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), and ResNet won the first place of the challenge in 2016. The VGGNet involves 16 (VGG16) and 19 (VGG19) weight layers and shows a significant improvement on prior configurations by using an architecture with very small convolution filters. The original Inception architecture GoogleNet combines the network-in-network approach and the strategy of using a series of filters of different sizes to handle multiple scales. The Inception-v3 is an improved Inception architecture which can be scaled up with high computational efficiency and low parameter count. ResNet is built up by stacking residual building blocks. Each building block is composed of several convolutional

(a)



(b)

Figure 1: Visualization of activations for an input image in the first convolutional layer of the pretrained VGG16 model: (a) original image; (b) the first convolutional layer output.

layers with a skip connection. It lets each stacked layer fit a residual mapping, while skip connections carry out identity mapping. It is easier to optimize the residual mapping than to optimize the original mapping. The architecture solves the degeneration problem: as stacking more layers, the accuracy gets saturated and then degrades rapidly. ResNet50 is the 50-layer version of the network.

4. Material and Experiment

4.1. Data Material. The PlantVillage is an open access database of more than 50,000 images of healthy and diseased crops, which have a spread of 38 class labels. We select the images of healthy apple leaves and images of apple leaf black rot caused by the fungus *Botryosphaeria obtusa*. Each image is assessed into one class by botanists: healthy stage, early stage, middle stage, or end stage. The healthy-stage leaves are free of spots. The early-stage leaves have small circular spots with diameters less than 5 mm. The middle-stage leaves have more than 3 spots with at least one frog-eye spot enlarging to irregular or lobed shape. The end-stage leaves are so heavily infected that will drop from the tree. Each image is examined by agricultural experts and labeled with appropriate disease severity. 179 images which are inconsistent among experts are abandoned. Figure 2 shows some examples of every stage. Finally, we get 1644 images of healthy leaves, 137 early-stage, 180 middle-stage, and 125 end-stage disease images.

Table 1: The number of samples in training and test sets.

Class	Number of images for training	Number of images for testing
Healthy stage	110×12	27×12
Early stage	108	29
Middle stage	144	36
End stage	102	23

As healthy leaves are much more than the diseased leaves, there is much difference in the number of samples per class. The number of samples per class should be balanced to reduce the bias the network may have towards the healthy-stage class with more samples. Our strategy of balancing is as follows: for early stage, middle stage, and end stage, about 80% of the images are used as the training set and the left 20% are the hold-out test set. For healthy-stage leaves, the images are divided into 12 clusters, with 110 images in each cluster on average for training. 27 images are left for testing. The final accuracy is estimated by averaging over 12 runs on the clusters. As the PlantVillage dataset has multiple images of the same leaf taken from different orientations, all the images of the same leaf should be either in the training set or in the test set. Table 1 shows the number of images used as training and test sets for each class.

4.2. Image Preprocessing. The samples in the PlantVillage dataset are arbitrarily sized RGB images. Thanks to the powerful end-to-end learning, deep learning models only need 4 basic image preprocessing steps. Images are processed according to the following stages: firstly, we resize all the images to 256×256 pixels for shallow networks, 224×224 for VGG16, VGG19, and ResNet50, and 299×299 for Inception-V3. We perform both the model optimization and prediction on these rescaled images. Secondly, all pixel values are divided by 255 to be compatible with the network's initial values. Thirdly, sample-wise normalization is performed. Normalization can significantly improve the efficiency of end-to-end training. The normalization is performed as follows: for each input x, we calculate the mean value m_x and standard deviation s_x and then transform the input to $x' = (x - m_x)/s_x$, so that the individual features more or less look like standard normally distributed data with zero mean and unit variance. Finally, several random augmentations including random rotation, shearing, zooming, and flipping are applied to the training images. The augmentation prevents overfitting and makes the model generalize better.

4.3. Neural Network Training Algorithm. The basic architecture in the convolutional neural network begins with several convolutional layers and pooling layers, followed by fully

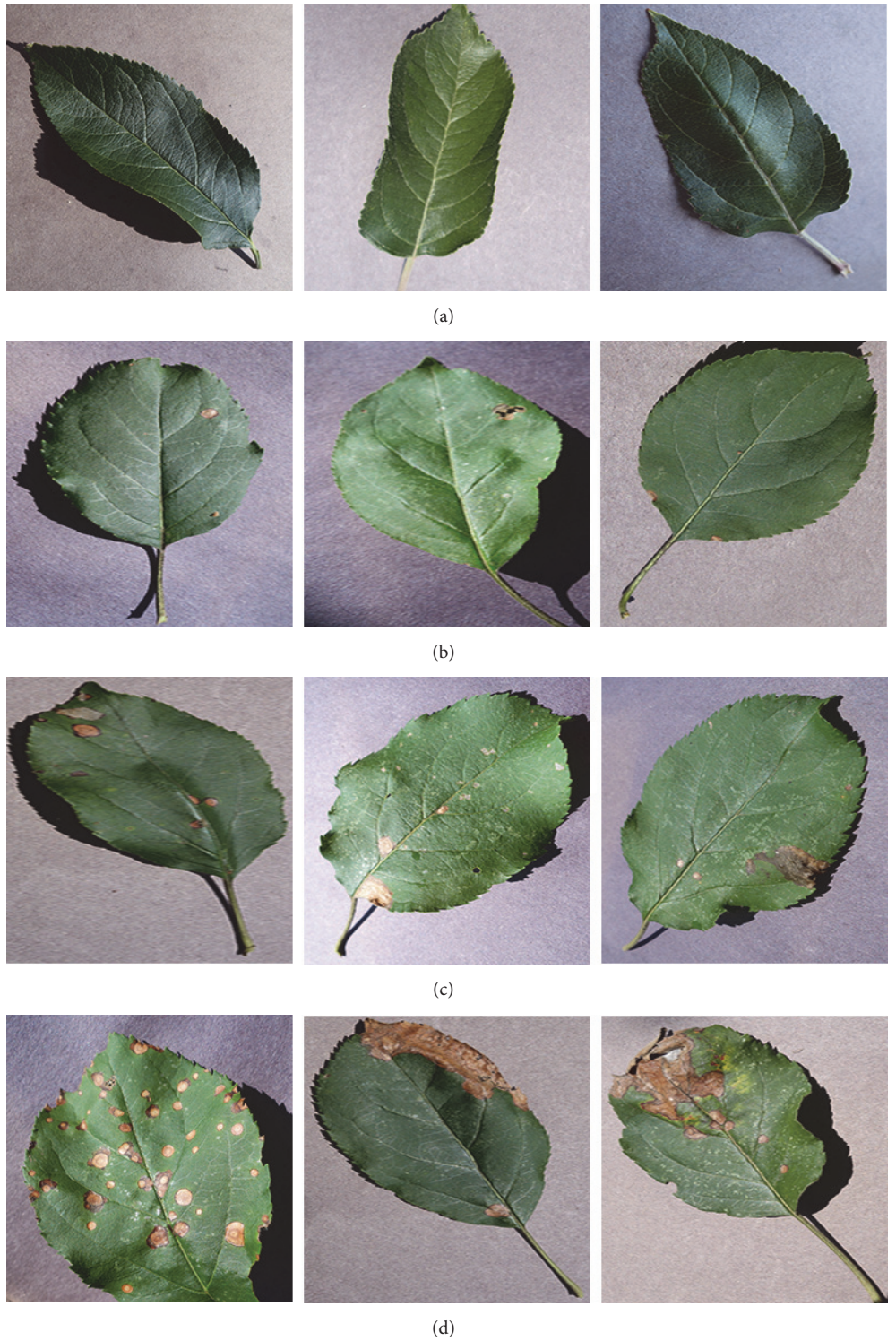


Figure 2: Sample leaf images of the four stages of apple black rot: (a) healthy stage, (b) early stage, (c) middle stage, and (d) end stage.

connected layers. For an input x of the ith convolutional layer, it computes

$$x_{ic} = \mathrm{ReLU}\left(W_i * x\right), \tag{1}$$

where $*$ represents the convolution operation and W_i represents the convolution kernels of the layer. $W_i = [W_i^1, W_i^2, \ldots, W_i^K]$, and K is the number of convolution kernels of the layer. Each kernel W_i^K is an $M \times M \times N$ weight matrix with

M being the window size and N being the number of input channels.

ReLU represents the rectified linear function $\mathrm{ReLU}(x) = \max(0, x)$, which is used as the activation function in our models, as deep convolutional neural networks with ReLUs train several times faster than their equivalents with saturating nonlinearities.

A max-pooling layer computes the maximum value over nonoverlapping rectangular regions of the outputs of each convolution kernel. The pooling operation enables position invariance over larger local regions and reduces the output size.

Fully connected layers are added on top of the final convolutional layer. Each fully connected layer computes $\mathrm{ReLU}(W_{\mathrm{fc}}X)$, where X is the input and W_{fc} is the weight matrix for the fully connected layer.

The loss function measures the discrepancy between the predicted result and the label of the input, which is defined as the sum of cross entropy:

$$E(W) = -\frac{1}{n}\sum_{x_i=1}^{n}\sum_{k=1}^{K}\left[y_{ik}\log P\left(x_i = k\right) + \left(1 - y_{ik}\right)\log\left(1 - P\left(x_i = k\right)\right)\right], \tag{2}$$

where W indicates the weight matrixes of convolutional and fully connected layers, n indicates the number of training samples, i is the index of training samples, and k is the index of classes. $y_{ik} = 1$ if the ith sample belongs to the kth class; else $y_{ik} = 0$. $P(x_i = k)$ is the probability of input x_i belonging to the kth class that the model predicts, which is a function of parameters W. So the loss function takes W as its parameters.

Network training aims to find the value of W that minimizes the loss function E. We use gradient descent algorithm where W is iteratively updated as

$$W_k = W_{k-1} - \alpha\frac{\partial E(W)}{\partial W}, \tag{3}$$

where α is the learning rate, which is a very important parameter that determines the step size of the learning. The value of learning rate should be carefully evaluated.

We use early stopping as the training stop strategy to stop training when the network begins to overfit the data. The performance of the network is evaluated at the end of each epoch using the test set. If the loss value of the test set stops improving, the network will stop training.

To prevent overfitting, the transfer learning is conducted as follows: fully connected layers are replaced with a new one and only fine-tune the top convolutional block for VGG16 and VGG19, the top two inception blocks for Inception-v3, and the top residual block for ResNet50, along with the new fully connected layers. To avoid triggering large gradient updates to destroy the pretrained weights, the new fully connected network should be initialized with proper values rather than with random values. So firstly we freeze all layers except the new fully connected network. The new fully connected network is trained on the output features of the final convolutional layer. The weights learned from training are initial values for fine-tuning. After that, the top convolutional block for VGG16 and VGG19, the top two inception blocks for Inception-v3, and the top residual block for ResNet50 are unfreezed and then trained along with the new fully connected network with a small learning rate.

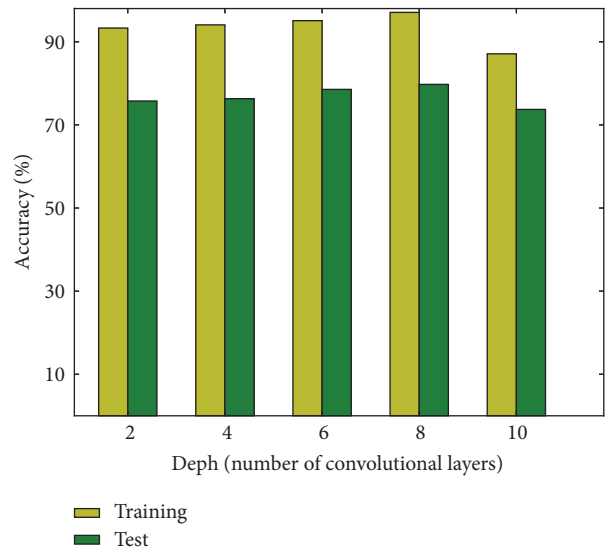


Figure 3: Accuracies of shallow networks.

The parameters for training shallow networks and fine-tuning pretrained models are presented in Table 2. Besides, a learning rate schedule is employed. The initial learning rate is dropped by a factor of 10 every 50 epochs for training shallow networks with less than 6 convolutional layers and fine-tuning deep networks. And it dropped by 10 every 100 epochs for shallow networks with 6 or more convolutional layers. Because the network goes deeper, it needs more training steps to converge.

4.4. Implementation. The experiment is performed on an Ubuntu workstation equipped with one Intel Core i5 6500 CPU (16 GB RAM), accelerated by one GeForce GTX TITAN X GPU (12 GB memory). The model implementation is powered by the Keras deep learning framework with the Theano backend.

5. Result and Discussion

Figure 3 shows the training and testing accuracies of shallow networks trained from scratch. Each bar represents the average result of 12 runs. Both training and test accuracies improve slightly with the depth of the model at first. The best performance, that is, a test accuracy of 79.3%, is achieved by the network with 8 convolutional layers. But the accuracies fall when the network's depth exceeds 8, as there are insufficient training data for models with too many parameters. To circumvent this problem, transfer learning is applied to the state-of-the-art deep models.

The results of fine-tuning the ImageNet pretrained models are reported in Figure 4. Each bar represents the average result of 12 runs. The overall accuracy on the test set we

Table 2: The hyperparameters of training.

Parameters	Learning from scratch	Transfer learning	
		Training fully connected layers	Fine-tuning
Training algorithm	SGD	RMSP	SGD
Learning rate	0.01	0.01	0.0001
Batch size		32	
Early stopping		10 epochs	

Table 3: Confusion matrix for the prediction of VGG16 model trained with transfer learning.

		Predicted			
		Healthy stage	Early stage	Middle stage	End stage
Ground truth	Healthy stage	27	0	0	0
	Early stage	0	27	2	0
	Middle stage	0	5	30	1
	End stage	0	0	3	20

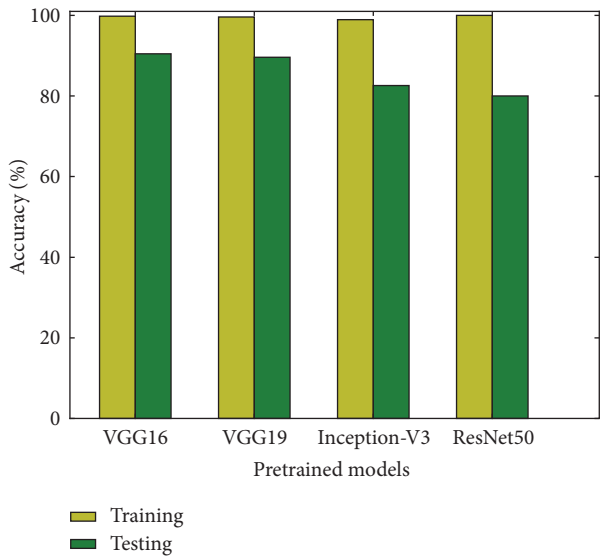


Figure 4: Accuracies of the state-of-the-art extreme deep models trained with transfer learning.

obtained varies from 80.0% to 90.4%. The performance of fine-tuned models is superior to that of models trained from scratch. The best result is achieved by the VGG16 model, with an accuracy of 90.4%. The results indicate that transfer learning alleviates the problem of insufficient training data.

For comparison, an ANN model is trained by SGD optimizer end-to-end on the training set. A test accuracy of 31% is achieved, which is basically random guessing. Without the convolutional feature extractor, the ANN cannot extract local correlations and learn discriminative features from the images.

The confusion matrix of the VGG16 model on the hold-out test set is shown in Table 3. The fraction of accurately predicted images for each of the four stages is displayed in detail. All of the healthy-stage leaves are correctly classified. The accuracies of early stage and end stage are 93.1% and 87.0%, respectively. Middle stage is prone to be misclassified, with an accuracy of 83.3%. However, the misclassified stages are only confused with their adjacent stages. For example, the early stage is only confused with the middle stage, and none of early-stage is classified as end stage.

From the results displayed in Figure 4, it is notable that the training accuracies of deep networks are close to 100% and trigger the early stopping. Since deep learning is data-driven, training on more data will further increase the test accuracy. It is also important to note that the best performance is achieved by the VGGNet. The result is consistent with that of [26, 28], where the VGGNet showed better performance in the PlantCLEF plant identification task. Though ResNet achieved state-of-the-art result on the ImageNet dataset, it performs poorer than VGGNet on fine-grained classification tasks. The SGD optimizer might put the residual mapping in building blocks of ResNet to zero too early, which leads to a local optimization and results in the poor generalization in fine-grained classification.

6. Conclusion

This work proposes a deep learning approach to automatically discover the discriminative features for fine-grained classification, which enables the end-to-end pipeline for diagnosing plant disease severity. Based on few training samples, we trained small convolutional neural networks of different depth from scratch and fine-tuned four state-of-the-art deep models: VGG16, VGG19, Inception-v3, and ResNet50. Comparison of these networks reveals that fine-tuning on pretrained deep models can significantly improve the performance on few data. The fine-tuned VGG16 model performs best, achieving an accuracy of 90.4% on the test set, demonstrating that deep learning is the new promising technology for fully automatic plant disease severity classification.

In future work, more data at different stages of different diseases will be collected with versatile sensors, like infrared

camera and multispectral camera. The deep learning model can be associated with treatment recommendation, yield prediction, and so on.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Yu Sun and Guan Wang contributed equally to this work.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities: 2017JC02 and TD2014-01.

References

[1] C. H. Bock, G. H. Poole, P. E. Parker, and T. R. Gottwald, "Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging," *Critical Reviews in Plant Sciences*, vol. 29, no. 2, pp. 59–107, 2010.

[2] A. M. Mutka and R. S. Bart, "Image-based phenotyping of plant disease symptoms," *Frontiers in Plant Science*, vol. 5, article no. 734, 2015.

[3] "Plant Polyphenols, Prenatal Development and Health Outcomes," *Biological Systems: Open Access*, vol. 03, no. 01, 2014.

[4] E. L. Stewart and B. A. McDonald, "Measuring quantitative virulence in the wheat pathogen zymoseptoria tritici using high-throughput automated image analysis," *Phytopathology*, vol. 104, no. 9, pp. 985–992, 2014.

[5] J. G. A. Barbedo, "An automatic method to detect and measure leaf disease symptoms using digital image processing," *Plant Disease*, vol. 98, no. 12, pp. 1709–1716, 2014.

[6] Y. Atoum, M. J. Afridi, X. Liu, J. M. McGrath, and L. E. Hanson, "On developing and enhancing plant-level disease rating systems in real fields," *Pattern Recognition*, vol. 53, pp. 287–299, 2016.

[7] F. Qin, D. Liu, B. Sun, L. Ruan, Z. Ma, and H. Wang, "Identification of alfalfa leaf diseases using image recognition technology," *PLoS ONE*, vol. 11, no. 12, Article ID e0168274, 2016.

[8] H. Al Hiary, S. Bani Ahmad, M. Reyalat, M. Braik, and Z. ALRahamneh, "Fast and Accurate Detection and Classification of Plant Diseases," *International Journal of Computer Applications*, vol. 17, no. 1, pp. 31–38, 2011.

[9] E. Omrani, B. Khoshnevisan, S. Shamshirband, H. Saboohi, N. B. Anuar, and M. H. N. M. Nasir, "Potential of radial basis function-based support vector regression for apple disease detection," *Measurement: Journal of the International Measurement Confederation*, vol. 55, pp. 512–519, 2014.

[10] D. L. Hernández-Rabadán, F. Ramos-Quintana, and J. Guerrero Juk, "Integrating SOMs and a Bayesian Classifier for Segmenting Diseased Plants in Uncontrolled Environments," *Scientific World Journal*, vol. 2014, Article ID 214674, 2014.

[11] F. E. Correa-Tome, "Comparison of perceptual color spaces for natural image segmentation tasks," *Optical Engineering*, vol. 50, no. 11, p. 117203, 2011.

[12] M. Schikora, A. Schikora, K. H. Kogel, and D. Cremers, "Probabilistic classification of disease symptoms caused by Salmonella on Arabidopsis plants, presented at the GI Jahrestagung," *Probabilistic classification of disease symptoms caused by Salmonella on Arabidopsis plants, presented at the GI Jahrestagung*, 2010.

[13] J. G. A. Barbedo, "A new automatic method for disease symptom segmentation in digital photographs of plant leaves," *European Journal of Plant Pathology*, vol. 147, no. 2, pp. 349–364, 2016.

[14] L. Lamari, *Assess: Image Analysis software helpdesk, Version 2*, vol. 1, APS Press, 2008.

[15] S. J. Pethybridge and S. C. Nelson, "Leaf doctor: A new portable application for quantifying plant disease severity," *Plant Disease*, vol. 99, no. 10, pp. 1310–1316, 2015.

[16] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, pp. 770–778, Las Vegas, Nev, USA, June 2016.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[19] N. Doulamis and A. Voulodimos, "FAST-MDL: Fast Adaptive Supervised Training of multi-layered deep learning models for consistent object tracking and classification," in *Proceedings of the 2016 IEEE International Conference on Imaging Systems and Techniques, IST 2016*, pp. 318–323, October 2016.

[20] E. Gawehn, J. A. Hiss, and G. Schneider, "Deep Learning in Drug Discovery," *Molecular Informatics*, vol. 35, no. 1, pp. 3–14, 2016.

[21] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning," *Nature Biotechnology*, vol. 33, no. 8, pp. 831–838, 2015.

[22] K. Makantasis, N. Doulamis, and A. Voulodimos, "Recognizing Buildings through Deep Learning: A Case Study on Half-timbered Framed Buildings in Calw City," in *Proceedings of the Special Session on Computer Vision, Imaging and Computer Graphics for Cultural Applications*, pp. 444–450, Porto, Portugal, Feburary 2017.

[23] H. Goëau, P. Bonnet, and A. Joly, "LifeCLEF plant identification task 2015," in *Proceedings of the Conference and Labs of the Evaluation Forum (CLEF '15)*, 2015.

[24] S. Choi, "Plant identification with deep convolutional neural network: SNUMedinfo at LifeCLEF plant identification task 2015," in *Proceedings of the 16th Conference and Labs of the Evaluation Forum, CLEF 2015*, September 2015.

[25] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[26] M. Mehdipour Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, pp. 228–235, 2017.

[27] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, https://arxiv.org/abs/1409.1556.

[28] S. T. Hang and M. Aono, "Open world plant image identification based on convolutional neural network," in *Proceedings of the*

2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), pp. 1–4, Jeju, South Korea, December 2016.

[29] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 3289801, 2016.

[30] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, article 1419, 2016.

[31] S. Xie, T. Yang, X. Wang, and Y. Lin, "Hyper-class augmented and regularized deep learning for fine-grained image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 2645–2654, June 2015.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 2818–2826, July 2016.

Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 5169675, 14 pages
https://doi.org/10.1155/2017/5169675

Research Article

Bag of Visual Words Model with Deep Spatial Features for Geographical Scene Classification

Jiangfan Feng, Yuanyuan Liu, and Lin Wu

College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Correspondence should be addressed to Yuanyuan Liu; s140231038@stu.cqupt.edu.cn

Received 13 February 2017; Revised 31 March 2017; Accepted 18 May 2017; Published 19 June 2017

Academic Editor: Athanasios Voulodimos

With the popular use of geotagging images, more and more research efforts have been placed on geographical scene classification. In geographical scene classification, valid spatial feature selection can significantly boost the final performance. Bag of visual words (BoVW) can do well in selecting feature in geographical scene classification; nevertheless, it works effectively only if the provided feature extractor is well-matched. In this paper, we use convolutional neural networks (CNNs) for optimizing proposed feature extractor, so that it can learn more suitable visual vocabularies from the geotagging images. Our approach achieves better performance than BoVW as a tool for geographical scene classification, respectively, in three datasets which contain a variety of scene categories.

1. Introduction

Geographical scene classification can effectively facilitate the environmental management, indoor and outdoor mapping, and other analysis in dealing with spatial data. Spatial features have been proven to be useful in improving the representation of the image and increasing classification accuracies. In practice, a widely used method named bag of visual words (BoVW) finds the collection of local spatial features in the images and combining appearance and spatial information of images. The traditional BoVW model tends to require the identification of the spatial features of each pixel with a small number of training samples. However, despite the fact that their advantages have been proven in small data sets, their performance is variable and less satisfactory while dealing with large datasets due to the complexity and diversity of landscape and land cover pattern. Therefore, it is a challenge to obtain higher interpretation accuracies when dealing with increased geographical imagery.

To overcome these drawbacks, we combined the CNN-based spatial features and the BoVW-based image interpretation into geographical scene classification. The convolutional neural network (CNN) is a biologically inspired trainable architecture that can learn multilevel hierarchies of features, which perfectly meet the demand of feature learning tasks. With multiple layers, the input images' geographical features can be extracted and generated hierarchically and then expressed as geotags. Owing to the special architecture of CNN, the features learned by the networks are invariant to translation, scaling, tilting, and other forms of distortion of input images, reducing the influence of the deformation of the same geologic structure.

In detail, we utilize the invariance property of BoVW model. At the same time, the methods of preprocessing the geographic scene were taken, and the adaptability of the network structure was modified, so as to improve the generalization ability of the classification model. We combine the CNN-based feature extractor with the BoVW-based scene classification method in this study. The use of the CNN here helps us to look for the appropriate spatial feature extractors for BoVW more adaptively and methodically. The spatial features are obtained by training CNN of the original data set. The strategy of combining the BoVW with CNN-based feature extractor can well reveal the intrinsic properties of original data. And the experimental results show that the proposed method is more invariant to various transformations, which produces a much higher level representation of richer information and achieves superior performance at the geographic scene classification.

2. Related Works

Local feature descriptor has been widely used in the scene classification task, as it is a fundamental element for the low-level image representation. Over the past decade, considerable efforts have been devoted to designing appropriate feature descriptors for scene classification. As early as 2004, Csurka and other scholars [1] first formally proposed a visual word packet model algorithm for image scene classification. Good performance is achieved by using the local features in scene classification algorithm [2, 3]. Later Lazebnik et al. [4] proposed SPM (spatial pyramid matching) based on SIFT features, and the SIFT features in the scene classification had achieved a good performance. The GIST features proposed by Oliva and Tarralba [5] are to capture the spatial structure characteristics of the scene and ignore the subtle texture information of the objects or backgrounds contained in the image. In addition to using the label category level, the algorithm based on local feature can also use the bounding box information of the image [6]. These partial image features can be viewed as an expansion of image features from the lower to mid-level semantic features; what is more, these mid-level features can be obtained with more information of visual elements [7].

However, when the image type reaches over thousands of categories and the database capacity exceeds one million, the huge amount of data usually becomes difficult for the traditional method based on low-level features and high-level semantics; fortunately, the deep learning method based on large data has a better performance. In particular, the deep convolutional neural network has made a new breakthrough in scene classification tasks. CNNs are multilayer classes of deep learning models that use a single neural network to train the end-to-end pixel values from the original image to the output of classifier. Compared with the traditional neural network, CNN is more suitable for displaying the spatial structure of the image. Convolution corresponds to the local features of the image and pooling makes the feature obtained by convolution with translation invariance. The idea of CNNs was firstly introduced in [8], improved in [9], and refined and simplified in [10, 11]. The latest state of the art in image classification is the GoogLeNet, a deep CNN which has 22 layers [12]. Convolutional neural networks are very effective at learning image representations with shift-invariant features, directly from an original input image without any handmade feature extraction [13]. Thus, the application of convolutional neural network is more extensive, and its powerful ability of feature recognition has been fully reflected. The pretraining of CNNs in large databases (such as ImageNet) can be used as a universal feature extractor, which can be applied to other visual recognition tasks, and is far better than the method of manual feature design [14–18]. For larger datasets with very high spatial and spectral resolution, the deep learning frameworks seem to be suitable and more efficient in solving hyperspectral image classification [19–21]. In order to reduce the gap between naive feature extraction and methods relying on domain expertise, Vakalopoulou et al. [22] present an automated feature extraction scheme based on deep learning. And by integrating additionally spectral information during the training procedure, the calculated deep features can account for the building to not-building object discrimination [23]. Furthermore, Zuo et al. [18] propose a deep learning based approach that encodes pixels spectral and spatial information for hyperspectral data classification.

Although the chain structure of CNN had been used to solve the general rough classification problem, it did not do well in the geographic scene classification. The global representation of the coded representation is invariant to various transformations, but it has lost the spatial information. The representation of convolutional feature is more discriminative than the fully connected feature [24], and it is more sensitive to translation, rotation, and scalation [25]. In recent years, some traditional methods have been introduced into deep learning models, such as SVM, which is applied to CNN, and the CNN-SVM method [26] is proposed. Compared with the original model, the classification performance is improved, and it is proved that SVM as a classifier of depth model is very feasible. He et al. [27] found that in depth training convolutional neural network needs a uniform size and uniform size of image zoom will cause image or cutting deformation; they made the space of pyramid (Spatial pyramid Pooling, SPP polymerization). The aggregation layer can unify the output characteristics of different sizes to a uniform size, which is convenient to connect to the final fully connection layer. Cimpoi et al. [24] used the pretrained CNN on the PLACES database to extract the MIT-67 database of the convolutional feature and the whole connection feature and then used the linear SVM for classification. It turned out that the BoVW, VLAD, and Fisher Vectors coding method based on the convolutional feature can effectively improve the global representation of the image. However, in the study of An et al. [10], BoVW is on the invariance to image translation, rotation, and zoom, but the dictionary in the BoVW model is unsupervised learning (such as k-means) and it is unable to obtain the optimal dictionary adaptive classification task. BoVW model of unsupervised learning (such as k-means clustering) is relatively independent with convolutional features of supervised learning, and it could not obtain the optimal dictionary for adaptive classification tasks. Generally speaking, our method develops the hierarchical feature learning structure, which gets superior performance at the geographic scene classification.

3. Proposed Approach

As discussed above, geographical scene classification aims at categorizing images according to spatial semantic property. In the view of geographical scene classification, the semantics of images are relationships and difference among different geographical features. Generally speaking, our method develops the multilevel hierarchical feature learning structure, which gets superior performance at the geographical scene classification.

3.1. Overview. As is shown in Figure 1, we firstly use convolutional neural networks (CNNs) for optimizing proposed feature extractor to get more information from the geotagging images and then choose an appropriate layer to be embedded

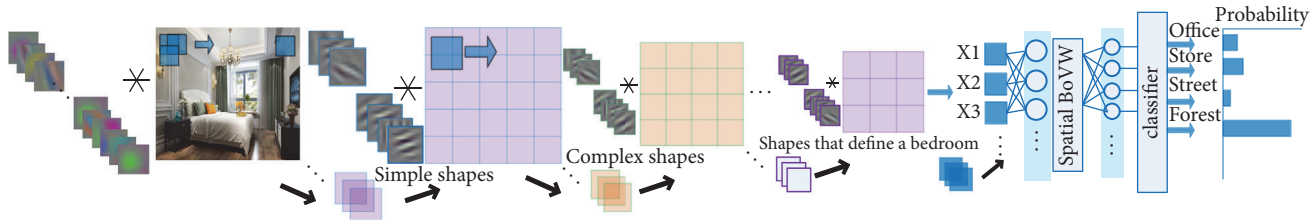


Figure 1: The architecture of our model.

in BoVW model to learn more suitable visual vocabularies, finally getting result by training a classifier.

The most straightforward way of increasing the performance of a neural network is to increase the depth and width of the network. However, many drawbacks are produced out with this method. For example, the larger the network is, the more parameters it contains, which make the network easier to overfit. Also, the use of computational resources is expanded dramatically. As a result, the fundamental approach of solving both issues would be by ultimately moving from fully connected to sparsely connected architectures, even inside the convolutions. As described in the next section, we train a network model that is suitable for extracting features.

3.2. Feature Extraction. The network structure of convolutional neural networks compared with the traditional network has three main points, which are locally connected, shared weights, and subsampling. The receptive field, the kernel, the number of layers, and the number of feature maps of each layer are primary variables affecting the quality of deep features. In this paper, we study how to extract the geographic features with various spatial correlations automatically and find out the visual structure information with automatic recognition in the image and focus on the feature extraction of the image content according to the identified region. We input the characteristics into the visual word bag model to achieve efficient identification of geographical images.

In the convolutional layer, we assume that the given large original image $l \times h$ is defined as x. At first, we obtain the small size image $a \times b$ from the large-size image by training the sparse coding. And we obtain k characteristics by calculating $f = \sigma(wx_s + b)$, and σ is an activation function; w and b are the weights and deviations between the visual layer unit and the implicit unit. For each small image, we get the corresponding value $f_s = \sigma(w^1 x_s + b^1)$, the convolution values of these f_s, and the matrix of convolution of the characteristics.

After obtaining the characteristics by convolution, these characteristics need to be classified. In theory, people can use all the extracted features to train the classifier, but too many parameters will make the training of the classifier difficult and prone to overfitting. Selecting the characteristics of different locations for polymerization statistics is called pooling. So we select the maximum pool or the average pool.

3.3. The Combination of BoVW Model and CNN. In order to learn the representation end-to-end, we design a CNN architecture that simulates this standard classification channel in a unified and principled manner with different modules. Given a geographical image, we crop the CNN at the last convolutional layer and view it as a dense descriptor extractor. That is to say, the output of the last convolutional layer is a $H \times W \times F$ map which can be considered as a set of F-dimensional descriptors extracted at $H \times W$ spatial locations. Among them, W and H represent the width and height of the feature map, respectively. F represents the number of channels, that is, the number of feature maps. Since each image contains a set of low-dimensional feature vectors, which has similar structure as dense SIFT, we propose to encode these feature vectors into a single feature vector using standard BoVW encoding. Namely, we design a new pooling layer inspired by the spatial bag of visual words that pools extracted descriptors into a fixed image representation and its parameters are learnable via back-propagation.

In this paper, we use the dictionary as a convolution dictionary D. Whether the convolutional dictionary $D = [d_1, d_2, \ldots, d_k]$can be supervised or not depends on the encoding mode of the convolutional vectors. Each encoding coefficient in the coding vector $C_I = [C_{i1}, C_{i2}, \ldots, C_{ik}]$ must be a compound function about the convolutional words d_k and the convolutional vector F_i, and the F_i model parameters can be guided. Here we use the soft distribution coding method; formula (1) is as follows:

$$c_{ik} = \frac{\exp\left(-\beta \left\| F_i - d_k \right\|_2^2\right)}{\sum_{k=1}^{K} \exp\left(-\beta \left\| F_i - d_k \right\|_2^2\right)}. \tag{1}$$

The encoding coefficients c_{ik} represent the degree of membership of the convolutional vector and the convolutional words. During the model training, we found that the soft distribution coding coefficients c_{ik} tend to zero or saturation, and it leads to the gradient disappearance of the model and makes the model unable to train. Therefore, in this paper, a method based on weighted direction similarity is used in the experiment. It uses a positive value of the vector dot product to indicate that it can be viewed as a weighted directional similarity, such as formula (2):

$$c_{ik} = \left[\left\langle F_i, d_k \right\rangle\right]_+. \tag{2}$$

Among them, $[\cdot]_+$ is said to take a positive part, and the negative part returns to zero. From the point of view of the activation function, $[\cdot]_+$ is similar as ReLU function, which is a kind of nonsaturation nonlinear transformation function. After the end of encoding direction similarity, all encoding vector volume laminated polymerized (i.e., the global summation of the spatial dimension and the pooling)

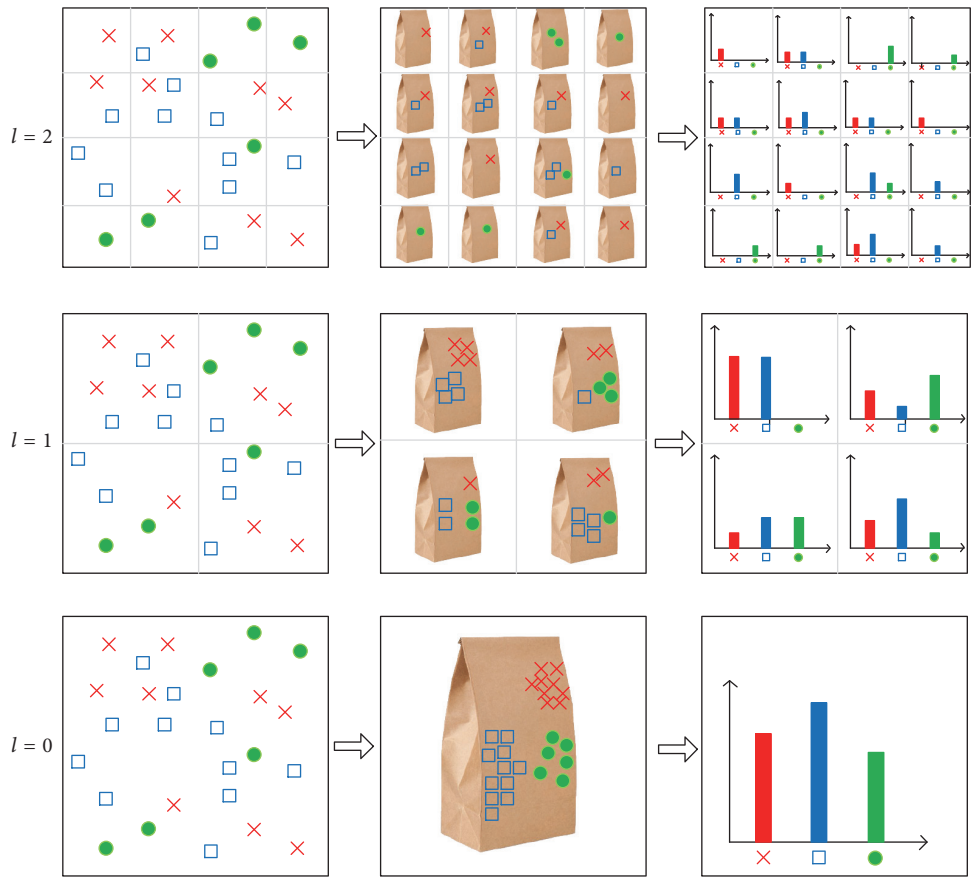


Figure 2: Space pyramid visual word bag model.

image global representation. The maxout activation function is used to carry out nonlinear transformation to the adjacent dimension of the representation, and the response is selected to form the final BoVW representation. In this paper, the BoVW representation is used to replace the full link representation $P = [p_1, p_2, \ldots, p_{k/2}]$, and the enhancement model is invariant to various image transformations. As illustrated in Figure 1 the layer can be visualized as a metalayer that is further decomposed into basic CNN layers connected up in a graph.

3.4. Image Representation Based on Spatial Vision. Csurka et al. [1] first proposed the BoVW model of Natural Language Processing field into image classification. Similar to the document consisting of a number of words, each image is represented as a visual word frequency histogram, in order to achieve better classification results. Bag of visual words model requires an unsupervised algorithm to extract low-level features from the images of the training image set, such as K-means algorithm, and clusters these low-level features according to the number of given cluster centers.

Because the traditional bag of words model does not consider the visual spatial information between local features, and it ignores the position order information of the image features, which is not conducive to the image feature extraction space. So Lazebnik et al. [4] put forward visual spatial pyramid bag of words model to make up for deficiencies. The idea of the spatial pyramid is to divide the images in different scales on each scale. Each image subregion is represented as a histogram vector, and the vectors of all subregions on each scale are combined to form a whole. The spatial pyramid histogram vector gives the image of the space of the gold tower vector representation. It repeats the provision of increasingly fine mesh sequences in the feature space to form a multiresolution histogram pyramid. Then the similarity between the two feature sets is defined as the weighted sum of the feature matching numbers at each level of the pyramid. And the visual spatial pyramid bag of words model is shown in Figure 2.

Scene classification is to extract the image features of all kinds of scenes from the scene image database and then classify all kinds of scenes. The basic content of the scene image classification includes data representation, classification model learning, and judgment. And it has a branch in the middle added semantic analysis topics. Overview of spatial BoVW model in scene classification block diagram as shown in Figure 3.

4. Experiments and Results

In this section, we first describe the experimental data sets and image preprocessing method. Afterwards, we describe

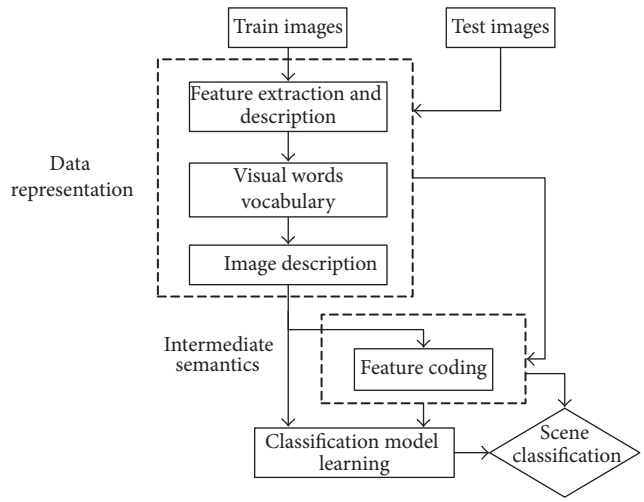


Figure 3: Overview of spatial BoVW model in scene classification.

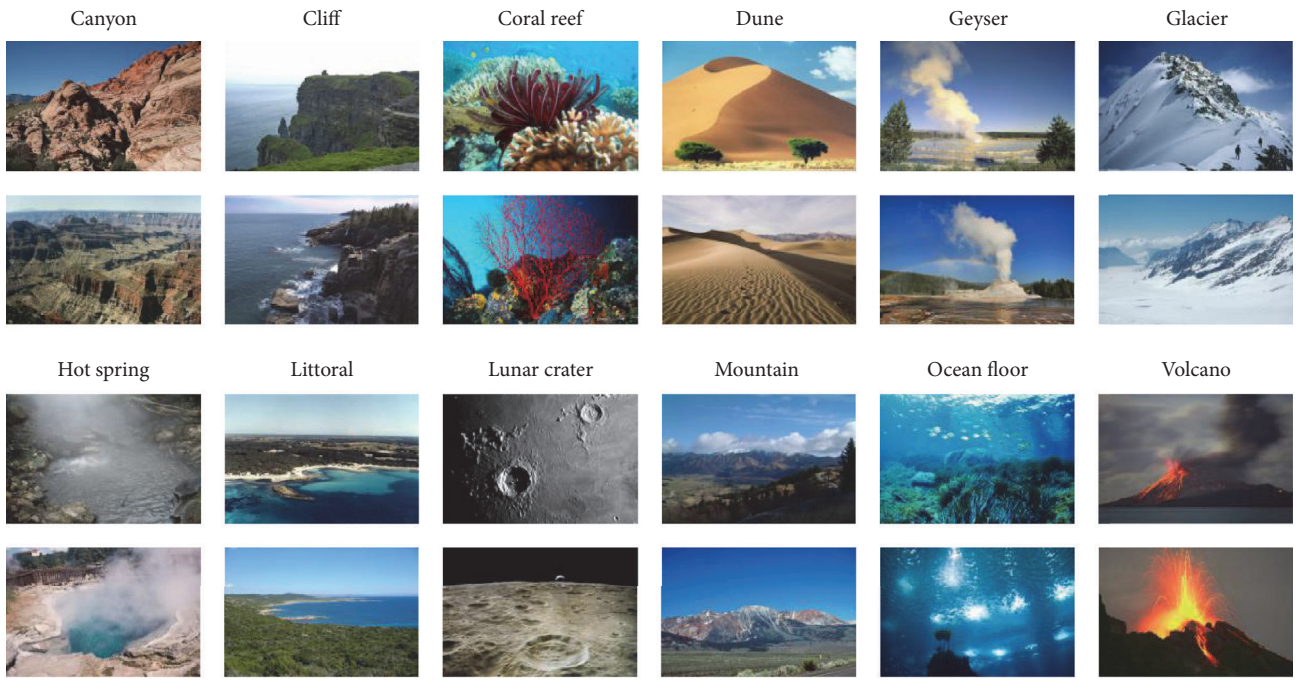


Figure 4: A few example in twelve-scene categories set.

CNN configuration and the parameter settings of the proposed method. The results obtained for the scene classification are last discussed.

4.1. Description of Data Sets. Here, we examine the proposed approach with two popular datasets and one our own dataset. Figure 4 shows a few example images representing various scenes that are included in the dataset we made, and the contents of the three data sets we choose are summarized here.

15-scene data set is composed of fifteen-scene categories: thirteen were provided by Li and Perona [28] (eight of these were originally collected by Oliva and Torralba [5]), and two (industrial and store) were collected by Lazebnik et al. [4]. This scene database is the most widely used database in the scene classification task, and it is appropriate enough (15 classes and approximately 10K images) to enable running a large number of experiments in a reasonable amount of time. Within each scenario are challenges relevant to complex topographies, unique spatial pattern, and different structures inside the region, and so forth.

Most geographical scene classification methods worked well for outdoor scenes but perform poorly in the indoor domain. The MIT indoor 67 [29] contains 67 scene categories and a total of 15,620 images, such as bakery, library, gym, and meeting room, and so forth. And the subtle classification of

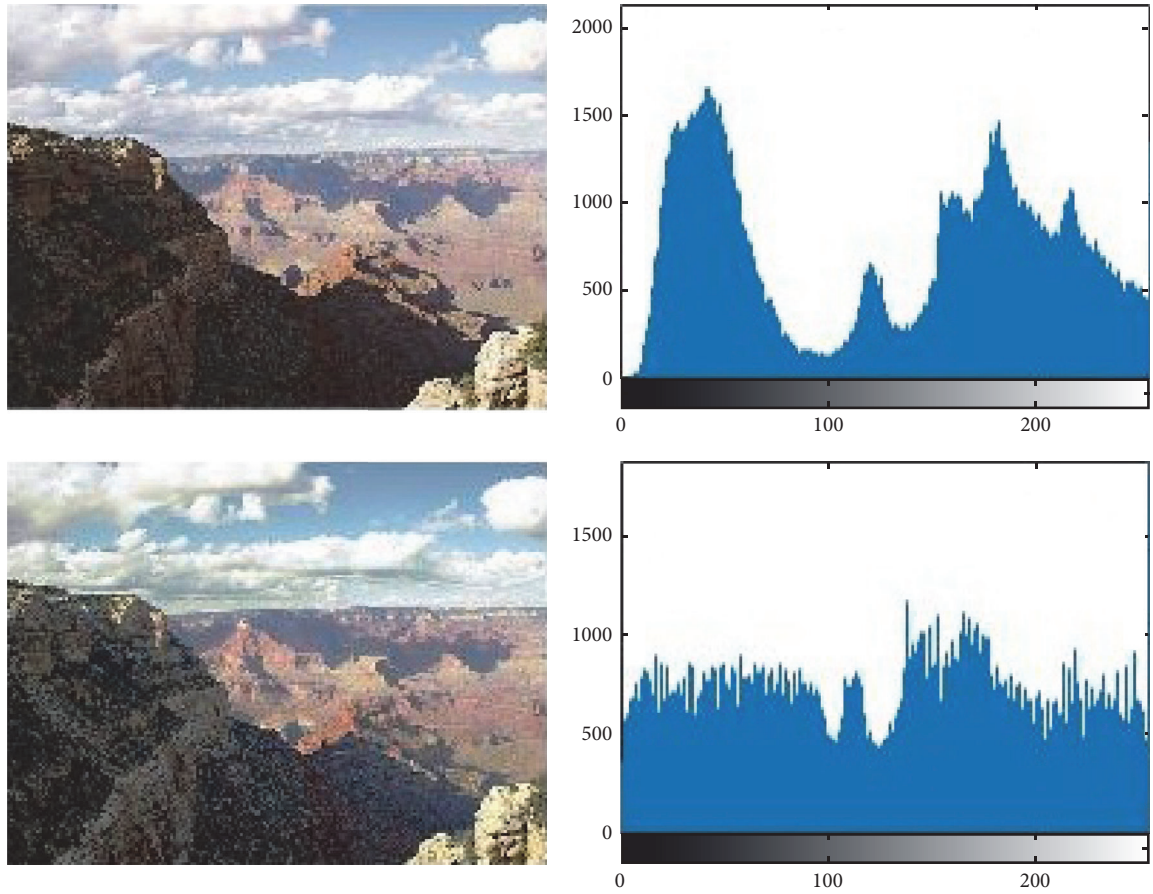


Figure 5: An example image before and after being processed with the average filter.

these interior layouts poses a great challenge to the sensitivity of the method in the geospatial pattern.

And the data set we made is based on ImageNet dataset, including twelve classes. ImageNet is an image database organized according to the WordNet hierarchy, where each node hierarchy is depicted by thousands of images. These images are collected from the web and manually annotated by artificial use of Amazon's Turkish robots. Geographic images are one of ImageNet's 12 semantic branches. There are 175 subclasses under the branch, some of which contain more than 500 images, and some do not contain any image. For a better training network, we select 12 categories for experimentation, each of which contains more than 1000 images. This dataset contains a changing geographical appearance and is realistic about the needs of the geographical scene classification. A few examples in twelve-scene categories set are as shown in Figure 4.

In order to generate a large amount of training data which are similar to the original data in real time, we use the random combination of affine transformations (translation, rotation, scaling, and cropping), horizontal inversion, and elastic cutting to amplify the training data. These methods can reduce the overfitting of model training and make the model get better generalization performance.

4.2. Preprocessing. For the input image, we adopt two preprocessing methods, including histogram averaging and Laplacian sharpening. Histogram averaging can increase the smoothness of the image, so that the image will not appear bright or dark situation. Since Laplacian is a differential operator, its application enhances the gray-scale region of the image and attenuates the slowly changing region of gray.

Figures 5 and 6, respectively, intuitively display that the contrast of the picture changes. Through the contrast figures above, we can intuitively see that the image enhancement based on histogram equalization and specification can reduce light noise of the captured image and improve the contrast details and the dynamic range of gray level of the image.

In order to verify the effect of image preprocessing on image feature representation, we randomly select a picture and enter a model that has been trained. The second convolutional layer feature graph is visualized after the input network of the preprocessed picture, and the second layer feature map after the input network of the unprocessed picture is also visualized. And we compare these two visual images in Figure 7.

As shown in Figure 7, we explain a little bit; (a) is the original picture; (b) is the visualization of the feature map of the second layer of the original image; (c) shows the picture

Figure 6: An example image before and after applying Laplacian operator.

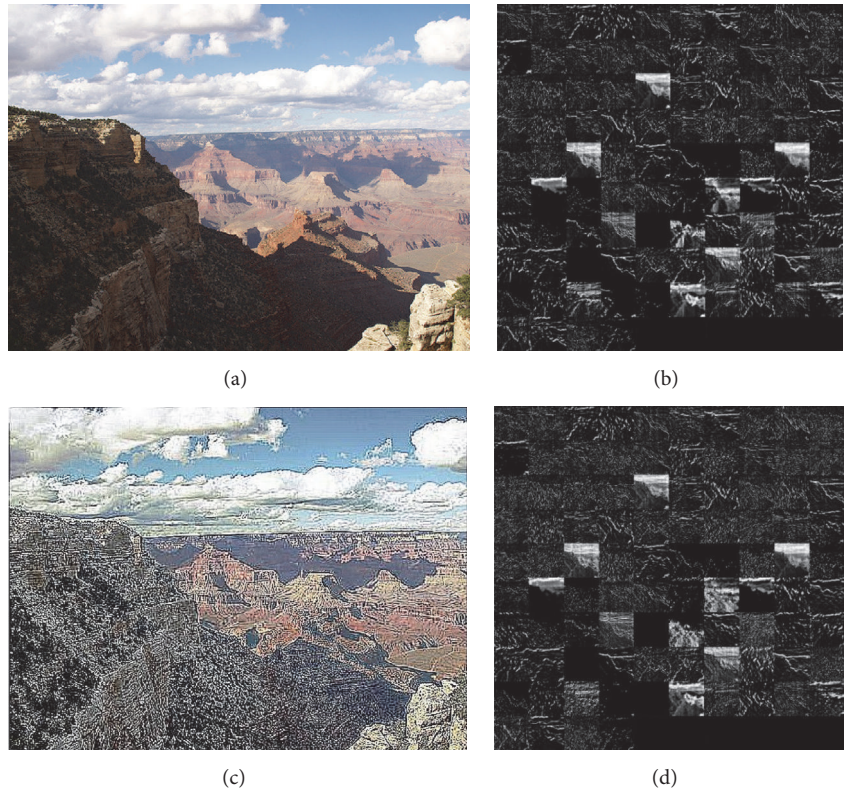


Figure 7: A comparison of the visualization of features before and after pretreatment.

after the preprocessing operation, and we can observe that the spatial structure of the picture is clearer; (d) graph is the visualization of the feature of the second layer after the image input network is preprocessed. By comparison we empirically found that taking the following two steps as a preprocession helps to improve the image regularization without much trade-off for the feature reconstruction error.

4.3. CNN Configuration

4.3.1. Network Architecture. In this paper, we choose deep learning frameworks named TensorFlow [35] to create and train our CNN model to extract the feature. During the deep feature extraction process, it is important to address the configuration of the deep learning framework. We will draw a vivid image to show the architecture of the our proposed CNN and introduce the network in detail.

Figure 8 shows our network architecture that can be summarized as 198 × 198-94 × 94 × 20-22 × 22 × 20-9 × 9 × 50-1000-1000-M, where M is the number of classes. The input is a down sampled and normalized image of size 198 × 198. The network model of feature extraction we used has eight layers; three layers are convolutional layers, with each layer of laminated roll being pool layer and last two layers are fully connected layers. We empirically set the size of the receptive field to 11 × 11, which offered enough contextual information. Two fully connected layers of 1000 nodes each follow the convolutional and pooling layers. The last layer is

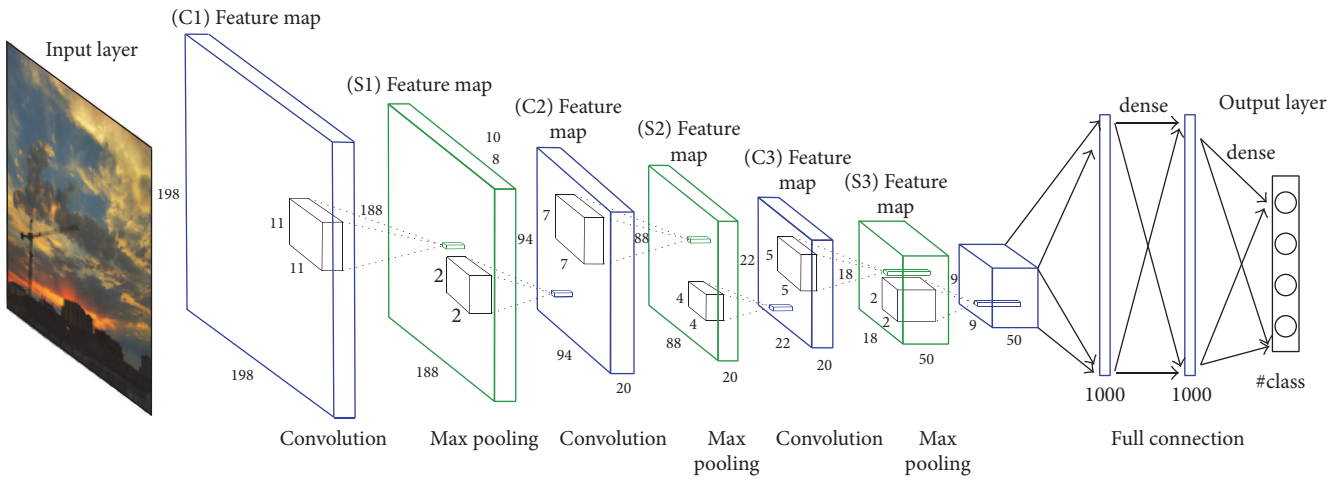


Figure 8: The architecture of the proposed CNN.

a logistic regression with softmax that outputs the probability on each class, as defined in the following equation:

$$P\left(y = i \mid x, W_1, \ldots, W_M, b_1, \ldots, b_M\right) = \frac{e^{W_i x + b_i}}{\sum_{j=1}^{M} e^{W_j x + b_j}}, \quad (3)$$

where x is the output of the second fully connected layer, W_i and b_i are the weights and biases of the neuron in this layer, and M is the number of classes. All weights are initialized to 10^{-2} and biases are set to 0. The class that outputs the max probability is taken as the predicted class, which can be described in the following equation ($\hat{y}$ denotes the predicted class):

$$\hat{y} = \arg\max_i P\left(y = i \mid x, W_1, \ldots, W_M, b_1, \ldots, b_M\right). \quad (4)$$

The receptive field is a term originally to describe an area of the body surface where a stimulus could elicit a reflex [36]. From the CNN visualization point of view, the receptive field is the output feature map node response corresponding to the input image area. In the training process, the cross entropy loss value unifies the normalized prediction value and the normalized coding of the label. TensorFlow provides a visualization tool TensorBoard which can be used to visualize changes in cross entropy during training.

In order to obtain the suitable scale characteristics in the case of a given database, it is necessary to find a suitable granularity. To further verify the effect of receptive field transforms, we conducted the experiments in 15-scene. We constantly adjust the size of the field according to the value of loss and classification results. We find that the transformation of the receptive field, in order to get more accurately modeling of complex invariances, does lead to a nontrivial increase in accuracy gain. One might speculate that using larger receptive fields could replace our receptive field transforms. The reasoning would be that larger receptive fields, hence larger neighborhoods, could be sufficient in capturing the change in position of features under complex view conditions. We repeated the experiment with a setup in which the receptive fields of the pooling grids in the final layer and the receptive field transform functions are set to be an identity function. The result was a decrease in accuracy over the base system which confirms our earlier claim that traditional pooling fails to meet requirement of complex invariance at object feature level and that the problem is not solvable by choice of receptive field size.

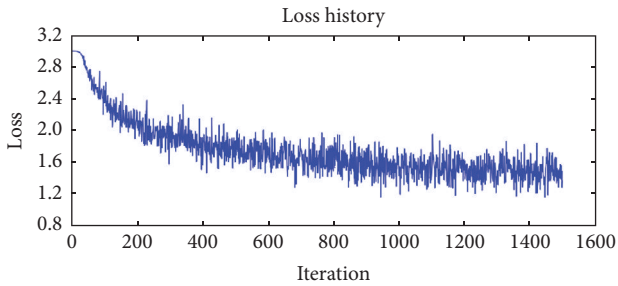


Figure 9: Training loss of our baseline network.

Figure 9 illustrates regression loss decrease against number of iterations for our network. Our model loss drops sharply at first and then reaches a plateau around iteration 600 and then begins to decrease linearly and converges to around 1.2. The loss of the model still decreased linearly when we cut off the training process.

4.3.2. Selectivity Analysis of Convolutional Feature. The purpose of this paper is to try to combine the strong feature learning ability of CNN and the invariant nature of BoVW coding model. This section focuses on solving the problem of embedding BoW model in CNN structure and selects the hierarchical feature of CNN by reconstructing the image.

Reconstructing images using features from deeper layers of the network tends to decide which layer of the convolutional layer will be embedded in the BoVW model. As Figure 10 shows, we reconstruct the features using the provided optimization parameters from the third convolutional layer of the pretrained model. In the cell below, we try to

Figure 10: Reconstruct features from the third layer of the pretrained model.

reconstruct the best image you can by inverting the features from the fifth layers in Figure 11. We will need to play with the hyperparameters to try and get a good result.

From the paper by Mahendran et al. [34, 37], we will see that reconstructions from deep features tend not to look much like the original image, so we should not expect the results to look like the reconstruction above. But we should be able to get an image that shows some discernable structure within 1000 iterations.

4.4. Parameter Settings. After obtaining the convolutional feature, we make full use of the invariance property of the BoVW model to obtain a more discriminative high-level distributed representation. This can not only avoid the gradient disappear in training and increase the identification of the characteristics of the middle convolution, but also increase the spatial structure information. After embedding into the BoVW model, the experiment encountered the following parameters.

4.4.1. The Selection of K Values. An important parameter in the BoVW model is dictionary size K, which needs to be tuned. In order to determine their values, Figure 12 shows the classification accuracy versus dictionary size K. From this figure, we can find that when $K = 396$, our approach obtains the best performance (90.14%).

4.4.2. Results of Different Kernel Functions. There are various options for this classifier, including GMM-type probabilistic models, SVMs, and neural networks. In this section, a simple SVM classifier [38] is employed in our method. The selection of kernel function is very important in the training process, because it determines the classification effect of the trained model. Next, we discuss the most suitable kernel function by experiment. This paper extracts six categories of images from the MIT indoor 67 data sets (bakery, classroom, elevator, museum, restaurant, and train station) to carry out classification experiments, and the experiment was divided into three parts: two classes (bakery, classroom), four-class classification (bakery, classroom, elevator, and museum), and six-class classification (bakery, classroom, classification elevator, museum, restaurant, and train station). Firstly, according to the previous chapter, the visual feature extraction, clustering, and expression are carried out, and the experimental results

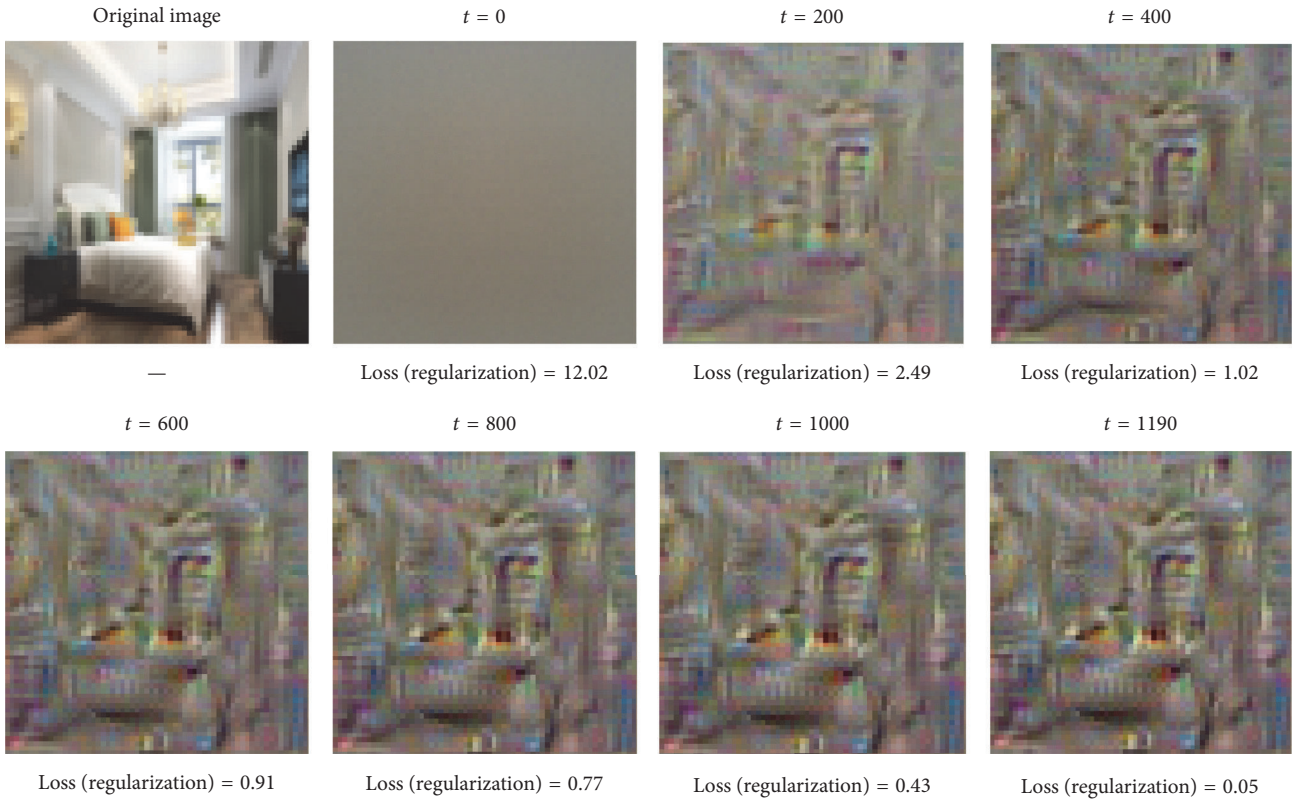


Figure 11: Reconstruct features from the fifth layer of the pretrained model.

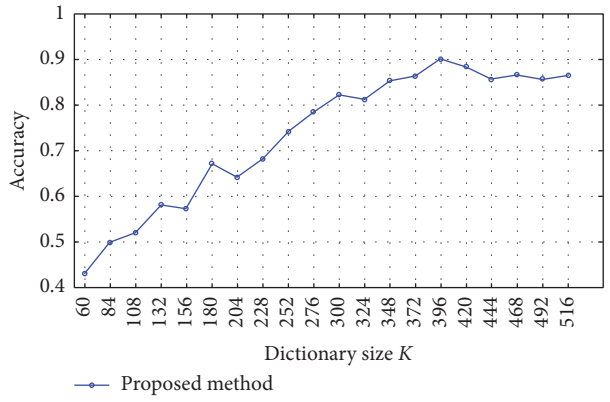


Figure 12: Comparison of different size K in our method.

Table 1: Classification accuracy of different kernel functions.

Name	2-class classification	4-class classification	6-class classification
Ploy	82.24%	79.20%	71.18%
RBF	87.57%	80.13%	73.68%
HIK	88.60%	84.41%	77.59%

are the average of 10 experiments. Table 1 gives the statistical results of SVM classification corresponding to different kernel functions.

Table 2: Time cost under different strides.

Stride	32	64	96	128
Time (sec)	0.102	0.036	0.023	0.018

Different kernel functions will be used to form different algorithms. The accuracy of these algorithms is greatly affected by the kernel function. Through the experimental comparison, we can see that the kernel function is the best choice and finally use the histogram intersection kernel (HIK) function for SVM classification.

4.5. Results and Discussion

4.5.1. Computational Cost. Our CNN is implemented using TensorFlow [35]. With TensorFlow we can easily run our algorithm on a GPU to speed up the process without much optimization. Our experiments are performed on a PC with 3.4 GHz CPU and GeForce 980Ti GPU. We measure the processing time on images using our model of 50 kernels with 198×198 input size and test the model using the part of those strides that gives the state-of-the-art performance in the experiments on 15-scene dataset. Table 2 shows the average processing time per image under different strides. Note that our implementation is not fully optimized. For example, the normalization process for each image is performed on the CPU in about 0.031 sec, which represents a significant portion of the total time. From Table 2 we can see that, with a

Table 3: The classification accuracy of different methods on 15-scene and MIT indoor datasets.

Number	Method	MIT indoor (%)	15-scene (%)
1	SPM [4]	34.40	81.4
2	OTC [30]	47.33	—
3	LPR [31]	44.8	85.8
4	Discriminative patches ++ [32]	49.40	—
5	FV + bag of parts [33]	63.18	—
6	MOP-CNN [25]	68.88	—
7	Our approach	66.09	90.1

Table 4: The classification accuracy of different fine-tuning model on 12-scene datasets.

Number	Method	12-scene (%)	Time (h)
1	*K*Means + SVM	59.80	0.5
2	Sift + BoVW	61.02	0.5
3	Local–global feature BoVW [34]	60.23	1.2
4	Fine-tuning Cifar + BoVW	51.12	12
5	Fine-tuning Alexnet + BoVW	67.01 ± 1.22	28
6	Fine-tuning GoogLeNet + BoVW	68.21 ± 0.61	36
7	Our approach	75.12	23

sparser sampling pattern (stride greater than 64), real time processing can be achieved while maintaining state-of-the-art performance.

At the same time, as shown in Table 4, we also recorded the fine-tuning of different models in the same visual database running time. The experiments show that our method under the condition of consumption in a certain time achieved good classification effect.

4.5.2. Performance on Different Datasets. After training our net for 100000 iterations, each iteration was composed by a random proportional subsample of 64 images across the entire dataset. In this section, we report results the performance on different datasets compared with previous studies. As shown in Table 3, in order to verify the classification performance of our method be more comprehensively, we compare our best results with the newly improved CNN method and other various state-of-the-art methods that have reported classification accuracy on 15-scene and MIT indoor datasets.

On the 15-scene dataset, our method achieves 90.1% accuracy, which exceeds the majority of existing methods. On the indoor-67 dataset, the two best reported accuracies are 63.18 [33] and 68.88 [25]. One used Fisher vector model and another extracted CNN activation for local patches at multiple scale levels. The accuracy of the activation features is rather close to the MOP-CNN [25]. However, these methods result in signature dimensionalities in the order of O (10^6), whose dimensionality can be almost one order of magnitude higher than our characteristic dimension.

In addition, in order to further refine the performance evaluation results, confusion matrix method is usually used to evaluate the performance of classification model in different categories of scene image content. Among them, the element values on the diagonal of the confusion matrix reflect the accuracy of the visual word packet model for each scene image classification; the higher the accuracy, the better the performance. Based on the predicted category for each test case, we will now construct a confusion matrix. Entry (i, j) in this matrix would be the proportion of times a test image of ground truth category i was predicted to be category j. An identity matrix is the ideal case. The visual predicted matrixes for two different datasets are shown in Figure 13. Through this color confusion matrix, we can intuitively know the scope of the overall accuracy, as well as what kind of classification effect is better. For example, from the visual confusion matrix, through color changes and contrast, we easily recognize that suburb is very easy to recognize.

In order to evaluate our model training results, we choose another six methods to do a comparative experiment on the similar data set. We used the traditional two methods and also reproduced the method which uses BoVW model [39]. Simultaneously, we fine-tune the three famous and well-trained CNN models to make the model adaptable to the new classified data set.

Since each neural network model has the desired image input size, we adjust the image size in the same way so that it can be used in this network model and output the corresponding classification and annotation. To make the comparison process more fair, all the parameters of each super-model, including the right to reinitialize, gradient descent optimization algorithm, the weights learning rate decay and decline algorithm, and training time, test time is set to the same value. We also modeled the last layer of the output layer into the same BoVW model to compare the classification results. All experiments are repeated ten times with different randomly selected training and test images, and the average of per-class recognition rates is recorded for each run. The final result is reported as the mean and standard deviation of the results from the individual runs, as shown in Table 4.

4.5.3. Visualized Table. We create convenient and intuitive interface to visualize the classification results table. This table is with one row per category, with 3 columns, training examples, false positives, and false negatives. False positives are instances claimed as that category but belonging to another category; for example, in the "Kitchen" row an image was classified as "Kitchen" but is actually "Store." This same image would be considered a false negative in the "Store" row, because it should have been claimed by the "Store" classifier but was not. False negatives are instances belonging to true category but with wrong predicted label. As shown in Figure 14, we show the first four lines of the table. From this

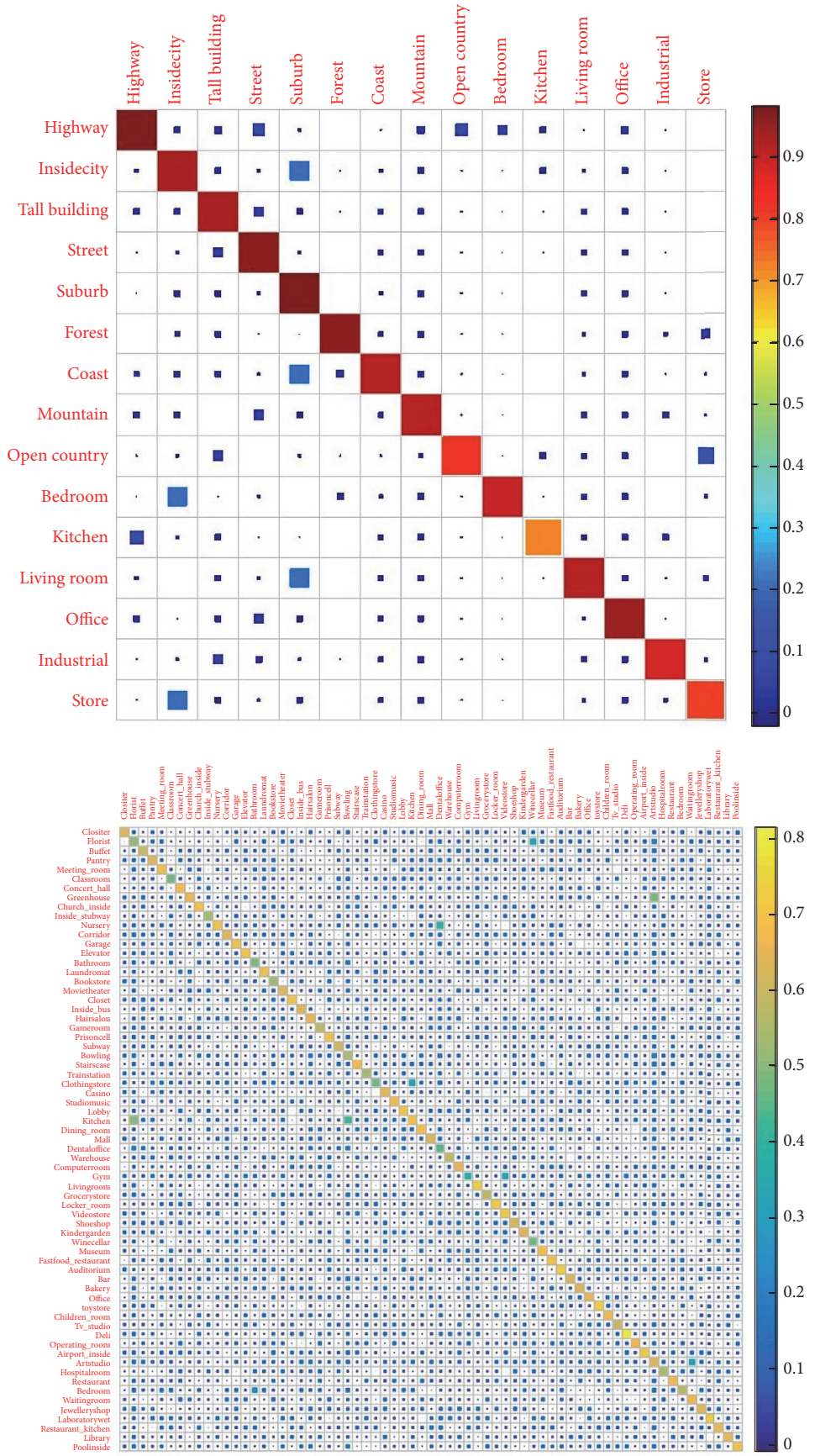


Figure 13: The visual predicted matrix of 15-scene and MIT indoor-67 datasets.

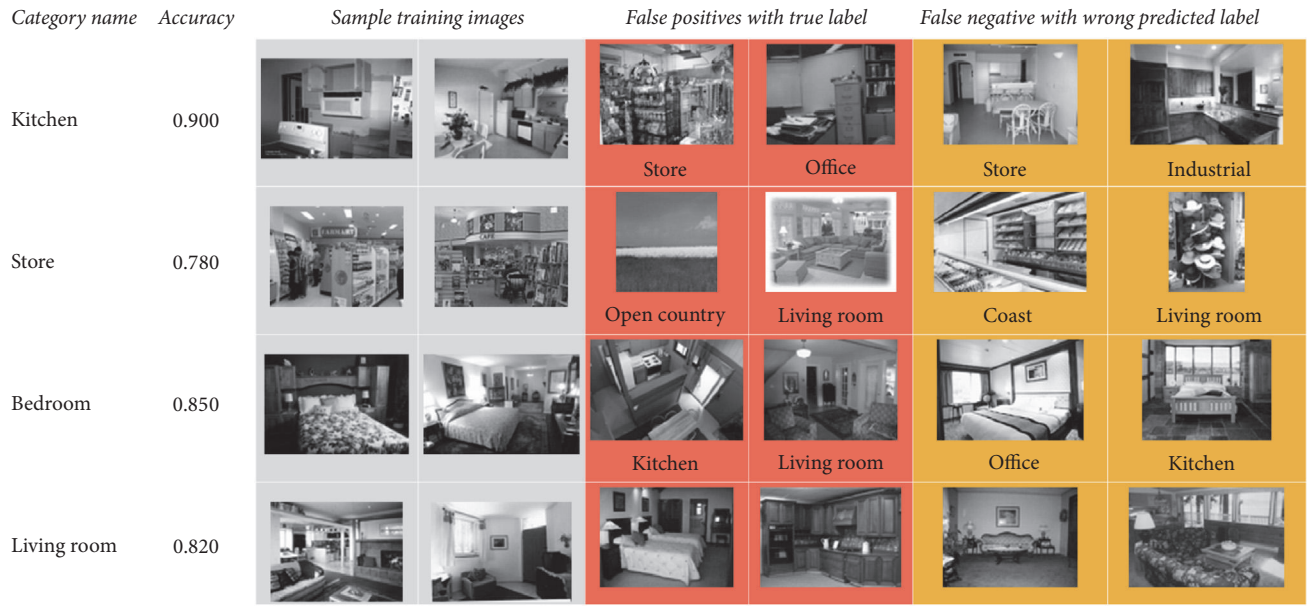


Figure 14: A visual table in fifteen-scene categories.

table, we can intuitively see which scene images are marked with wrong labels, and which scene images have not been identified.

5. Conclusions

In this paper, we present a method for geographical scene classification, which is a generalized version of convolutional neural networks based on higher level of hierarchy. As objects are the parts that compose a scene, detectors tuned to the objects that are discriminant between scenes are learned in the inner layers of the network. The learned parts have been shown to perform well on the task of scene classification, where they improved a very solid bag of words. According to simulation results applied to fifteen-scene categories, MIT indoor-67 dataset, and 12-scene dataset, even with less number of trainable parameters in less time the proposed hierarchical structure could classify unseen samples with higher accuracy. In the future we will explore how to improve encoding technique to ensure that the CNN descriptors are embedded into a finite set of prototypical elements in order to classify geotagging images.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work is supported by the National Nature Science Foundation of China (41571401), the Program for Innovation Team Building of Mobile Internet and Big Data at Institutions of Higher Education in Chongqing (CXTDX201601021), and the Natural Science Foundation Project of Chongqing (cstc2014jcyjA00027).

References

[1] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," *Workshop on Statistical Learning in Computer Vision Eccv*, pp. 1–22, 2004.

[2] D. Song and D. Tao, "Biologically inspired feature manifold for scene classification," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 174–184, 2010.

[3] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. Van Gool, "Modeling scenes with local descriptors and latent aspects," in *Proceedings of the 10th IEEE International Conference on Computer Vision, ICCV 2005*, pp. 883–890, October 2005.

[4] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 2169–2178, June 2006.

[5] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[6] Y. Zheng, Y. Jiang, and X. Xue, "Learning hybrid part filters for scene recognition," in *Computer Vision—ECCV 2012*, vol. 7576 of *Lecture Notes in Computer Science*, pp. 172–185, Springer, Berlin, Germany, 2012.

[7] J. Sun and J. Ponce, "Learning discriminative part detectors for image classification and cosegmentation," in *Proceedings of the 2013 14th IEEE International Conference on Computer Vision, ICCV 2013*, pp. 3400–3407, December 2013.

[8] K. Fukushima, "Neocognitron: a hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[10] D. C. An, U. Meier, and J. Masci, "Flexible, high performance convolutional neural networks for image classification[C]//

IJCAI," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1237–1242, Barcelona, Spain, 2011.

[11] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proceedings of the 7th International Conference on Document Analysis and Recognition*, vol. 2, pp. 958–963, IEEE Computer Society, Edinburgh, UK, August 2003.

[12] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, IEEE, Boston, MA, USA, June 2015.

[13] G. Hinton E, N. Srivastava, and A. Krizhevsky, "Improving neural networks by preventing co-adaptation of feature detectors," *Computer Science*, vol. 3, no. 4, pp. 212–223, 2012.

[14] B. Zhou, A. Garcia L, J. Xiao et al., "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 1, pp. 487–495, 2015.

[15] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '14)*, pp. 512–519, IEEE, Columbus, OH, USA, June 2014.

[16] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, pp. 14680–14707, 2015.

[17] Y. Xu, T. Mo, Q. Feng, P. Zhong, M. Lai, and E. I.-C. Chang, "Deep learning of feature representation with multiple instance learning for medical image analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '14)*, pp. 1626–1630, Florence, France, May 2014.

[18] Z. Zuo, G. Wang, B. Shuai, L. Zhao, Q. Yang, and X. Jiang, "Learning discriminative and shareable features for scene classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689, no. 1, pp. 552–568, 2014.

[19] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2015*, pp. 4959–4962, ita, July 2015.

[20] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, no. 2, Article ID 258619, 12 pages, 2015.

[21] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[22] M. Vakalopoulou, K. Karantzalos, N. Komodakis, and N. Paragios, "Building detection in very high resolution multispectral data with deep learning features," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2015*, pp. 1873–1876, July 2015.

[23] A. Merentitis and C. Debes, "Automatic fusion and classification using random forests and features extracted with deep learning," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2015*, pp. 2943–2946, July 2015.

[24] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 3828–3836, June 2015.

[25] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8695, no. 7, pp. 392–407, 2014.

[26] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[28] F. F. Li and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 524–531, IEEE Computer Society, 2005.

[29] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops '09)*, pp. 413–420, June 2009.

[30] R. Margolin, L. Zelnik-Manor, and A. Tal, "OTC: A novel local descriptor for scene classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8695, no. 7, pp. 377–391, 2014.

[31] F. Sadeghi and M. F. Tappen, "Latent pyramidal regions for recognizing scenes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7576, no. 5, pp. 228–241, 2012.

[32] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7573, no. 2, pp. 73–86, 2012.

[33] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2013*, pp. 923–930, June 2013.

[34] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 427–436, June 2015.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, and Z. Chen, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, https://www.tensorflow.org/.

[36] J. Alonso and Y. Chen, "Receptive field," *Scholarpedia*, vol. 4, no. 1, p. 5393, 2009.

[37] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 5188–5196, June 2015.

[38] C. Chang, J. Lin C, and LIBSVM., "A library for support vector machines," in *Acm Transactions on Intelligent Systems & Technology*, vol. 2, pp. 389–396, 3, article 27 edition, 2007.

[39] Q. Zhu, Y. Zhong, B. Zhao, G.-S. Xia, and L. Zhang, "Bag-of-Visual-Words Scene Classifier with Local and Global Features for High Spatial Resolution Remote Sensing Imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 6, pp. 747–751, 2016.