Complexity

Complex Deep Learning and Evolutionary Computing Models in Computer Vision

Lead Guest Editor: Li Zhang
Guest Editors: Chee Peng Lim and Jungong Han

Complex Deep Learning and Evolutionary Computing Models in Computer Vision

Complexity

Complex Deep Learning and Evolutionary Computing Models in Computer Vision

Lead Guest Editor: Li Zhang
Guest Editors: Chee Peng Lim and Jungong Han

Editorial Board

Contents

Hindawi
Complexity
Volume 2019, Article ID 1671340, 2 pages
https://doi.org/10.1155/2019/1671340

Editorial

Complex Deep Learning and Evolutionary Computing Models in Computer Vision

Li Zhang,[1] **Chee Peng Lim**,[2] **and Jungong Han**[3]

[1]*Northumbria University, Newcastle, UK*
[2]*Deakin University, Geelong, Australia*
[3]*Lancaster University, Lancaster, UK*

Correspondence should be addressed to Li Zhang; li.zhang@northumbria.ac.uk

Received 15 April 2019; Accepted 15 April 2019; Published 1 July 2019

Computer vision, which deals with how computers can be used to gain high-level understanding pertaining to information contained in digital images or videos, is an important, yet challenging, technology. The advent of deep learning and associated paradigms such as evolutionary computing models has propelled computer vision to the next level, solving a variety of complex problems in diverse applications, such as object detection, motion tracking, semantic segmentation, and emotion recognition. In this special issue, recent advances with respect to mathematical modeling, simulation, and/or analysis of deep learning and evolutionary computing models for undertaking complex phenomena in computer vision, are presented. A variety of problems, which include automatic object tracking, understanding image content, optical character recognition, personalized recommendation and movie summarization systems, and underwater video streaming, are covered. A description of each article is presented, as follows.

The paper titled "Multimodal Deep Feature Fusion (MMDFF) for RGB-D Tracking" addresses the limitation of existing tracking methods in processing geometrical features extracted from depth images by using a multimodal deep feature fusion model. The proposed model consists of four deep convolutional neural networks (CNNs). It extracts RGB (red, green, blue) and depth features from images using RGB-specific and depth-specific CNN models and exploits their correlated relationship using an RGB-depth correlated CNN model. In addition, a motion-specific CNN is used to provide high-level motion information for object tracking. Empirical evaluation with two RGB-depth datasets demonstrates that the proposed method achieves better performances, especially in situations where occlusion occurs, the movement is fast, and the target size is small, as compared with those from other state-of-the-art trackers.

Semantic image segmentation is useful for understanding the semantic information contained in an image. To improve the time-consuming pixel-level annotation process, a weakly supervised semantic segmentation method using the CNN and extreme learning machine is proposed in the paper titled "Weakly Supervised Deep Semantic Segmentation Using CNN and ELM with Semantic Candidate Regions". Semantic inference of candidate regions is realized based on the relationship and neighborhood rough set associated with semantic labels. After completing the inference step of all semantic labels, the extreme learning machine is used to learn the inferred candidate regions. The experimental results of two benchmark datasets show the proposed method is able to outperform several state-of-the-art alternatives for deep semantic segmentation.

To undertake imbalanced datasets, a deep learning model for unbalanced distribution character recognition based on a focal connectionist temporal classification (CTC) loss function is proposed in the paper titled "Focal CTC Loss for Chinese Optical Character Recognition on Unbalanced Datasets". The proposed model consists of three main components: convolutional, recurrent, and transcription layers. The residual network is used as the convolutional layers, which extract a feature sequence from the input image. The bidirectional long short-term memory is used as the recurrent layers, which predicts a label distribution for each frame. The focal CTC function is used in the transcription layer, which translates the per-frame predictions into the final

label sequence. A series of experimental studies using both synthetic and real image sequence datasets indicates that the proposed model is able to achieve better performance as compared with those from traditional CTC on imbalanced datasets.

The paper titled "A New Type of Eye Movement Model Based on Recurrent Neural Networks for Simulating the Gaze Behavior of Human Reading" tackles gaze behavior of human reading as a word-based sequence labeling task in natural language processing. The eye movement data are used to train a model that can predict the eye movements of the same reader reading a previously unseen text. Based on a combination of CNN models, bidirectional long short-term memory networks, and conditional random fields, a recurrent neural network is used to generate a gaze point prediction sequence. The empirical results indicate that the recurrent neural network-based model is able to achieve similar accuracy in predicting a user's fixation points during reading, with the advantages of less reliance on data features and less preprocessing than some existing machine learning models.

In e-commerce services, it is important for a personalized recommendation system to learn the latent user and item representations from implicit interactions between users and items. A neural personalized ranking model for collaborative filtering with implicit frequency feedback is introduced in the paper titled "Neural Personalized Ranking via Poisson Factor Model for Item Recommendation". A ranking-based Poisson factor model is developed, which adopts a pair-wise learning method to learn the rankings of preferences between items. A multilayer perceptron is used to learn the nonlinear user-item interaction relationships. The personalized ranking model is able to capture the complex structure of user-item interactions. The empirical results indicate the superiority of the proposed method over state-of-the-art recommendation algorithms.

To allow viewers to have an idea about the semantics of a movie in a short time, a movie summarization system produces a short video sequence that contains the most important scenes from the movie. In the paper titled "Personalized Movie Summarization using Deep CNN-Assisted Facial Expression Recognition", a user preference-based movie summarization technique is developed using a deep CNN model to analyze the emotional state of the characters through facial expression recognition. Segmentation of movie shots with faces using an entropy method is conducted. Then, a summary with respect to user preference from seven basic emotion classes is produced. A subjective evaluation using five Hollywood movies shows the effectiveness of the proposed scheme in terms of user satisfaction, while an objective evaluation indicates its superiority over state-of-the-art movie summarization methods.

Traffic loads and congestion management are important issues in wireless sensor networks. A proactive caching strategy based on a stacked sparse autoencoder to predict data package content popularity is devised in the paper titled "Deep Learning Based Proactive Caching for Effective WSN-Enabled Vision Applications". A simple frame structure of software defined network and network function virtualization technologies, coupled with the autoencoder in the sink and control nodes of the wireless sensor network, is constructed. The structure and model parameters associated with the stacked sparse autoencoder are optimized through training. A series of simulation studies to compare the proposed method with traditional classical caching strategies indicate that the stacked sparse autoencoder is able to improve the prediction accuracy for enhancing performance of wireless sensor networks.

Most of the classical monocular visual simultaneous localization and mapping (SLAM) methods do not consider the motion characteristics of the platform during the initialization phase. As such, a motion hypothesis to transform the solution of camera motion into an error elimination problem during the initialization process is introduced in the paper titled "Passive Initialization Method Based on Motion Characteristics for Monocular SLAM". The error is reduced by using a multiframe optimization method based on Bundle Adjustment, thus improving the accuracy of the initialization process. A hardware-in-the-loop simulation system on a fixed-wing aircraft is established as the test platform. The results indicate that the success rate of monocular SLAM initialization can be greatly improved, as compared with that of existing methods. However, this method cannot be used indiscriminately on platforms characterized by randomized motions.

In the paper titled "Design and Implementation of an Assistive Real-time Red Lionfish Detection System for AUV/ROVs", a remotely operated underwater vehicle with a robotic system for divers to locate red lionfish through real-time object recognition with a CNN-based model is designed and implemented. The assistive robot is able to identify red lionfish such that the divers can maximize their catch before each dive. The underwater vehicle is equipped with a camera to collect live videos underwater, and the video streams are processed in real-time to detect red lionfish. The developed system has been evaluated in areas currently invaded by red lionfish in the Gulf of Mexico. The outcome indicates the usefulness of the system for detecting red lionfish with high confidence in real-time.

It is hoped that this special issue serves as a cornerstone to further stimulate and promote research studies related to theory and application of deep learning and evolutionary computing models for advancing computer vision technology and delivering benefits to our society.

Conflicts of Interest

The editors declare no conflicts of interest.

Acknowledgments

The guest editors would like to thank the authors for contributing their articles and the reviewers for improving the quality of the articles through constructive comments and suggestions.

Li Zhang
Chee Peng Lim
Jungong Han

Hindawi
Complexity
Volume 2019, Article ID 3581419, 10 pages
https://doi.org/10.1155/2019/3581419

Research Article

Personalized Movie Summarization Using Deep CNN-Assisted Facial Expression Recognition

Ijaz Ul Haq,[1] **Amin Ullah**,[1] **Khan Muhammad**,[2] **Mi Young Lee**,[1] **and Sung Wook Baik**[1]

[1]*Intelligent Media Laboratory, Digital Contents Research Institute, Sejong University, Seoul 143-747, Republic of Korea*
[2]*Department of Software, Sejong University, Seoul 143-747, Republic of Korea*

Correspondence should be addressed to Sung Wook Baik; sbaik@sejong.ac.kr

Received 16 November 2018; Revised 22 January 2019; Accepted 2 April 2019; Published 5 May 2019

Guest Editor: Li Zhang

Personalized movie summarization is demand of the current era due to an exponential growth in movies production. The employed methods for movies summarization fail to satisfy the user's requirements due to the subjective nature of movies data. Therefore, in this paper, we present a user-preference based movie summarization scheme. First, we segmented movie into shots using a novel entropy-based shots segmentation mechanism. Next, temporal saliency of shots is computed, resulting in highly salient shots in which character faces are detected. The resultant shots are then forward propagated to our trained deep CNN model for facial expression recognition (FER) to analyze the emotional state of the characters. The final summary is generated based on user-preferred emotional moments from the seven emotions, i.e., afraid, angry, disgust, happy, neutral, sad, and surprise. The subjective evaluation over five Hollywood movies proves the effectiveness of our proposed scheme in terms of user satisfaction. Furthermore, the objective evaluation verifies the superiority of the proposed scheme over state-of-the-art movie summarization methods.

1. Introduction

The video data is exponentially increasing over the Internet and personal storage devices including social networks, surveillance, and movies data due to advances and easy access to capturing technologies. Movies data specifically has become one of the most entertaining sources for viewers. However, browsing a movie in enormous collections and searching for a desired scene within a complete movie is a tedious and time-consuming task. Movie summarization (MS) techniques have tried to tackle this problem by producing a short video sequence from the movie, which contains the most important events or scenes. Hence, the viewers may have an idea about the context and the semantics of the movie by watching only the important scenes.

In recent years, many MS techniques have been presented by researchers that can be broadly categorized into automatic MS techniques [1–10] and user-preference based MS techniques [11–15]. In automatic MS techniques, there is no direct preference from the users to generate a summary. These techniques rely on multiple clues such as scripts, subtitles, and movies structure in combination with visual and aural features. For instance, Ngo et al. [1] utilized concept-expansion trees to construct a relational graph for characterizing the semantic concepts of documentary videos. You et al. [2] proposed a summarization method, where four perceptive models are fused according to different cues including motion, contrast, statistical rhythm, and special scenes using a linear combination to generate the summary. In contrast, Weng et al. [3] analyzed the movie from the perspective of relationships between the characters of a movie rather than audio-visual features. They constructed a role-network and then identified the leading roles. The role-network is then used to segment the movie into several substories. Similarly, Salamin et al. [4] used an audio segmentation method along with maximum a posteriori probability (MAP) approach to determine main characters in a movie, which can be used for indexing and retrieval of specific character's shots as well as for summary generation. Sang and Xu [5] used face clustering to get main characters and,

based on their appearance, they generated movie summary which contained only main characters. Evangelopoulos et al. [6] proposed a multimodal saliency based summarization scheme for movies. They extracted features from three different modalities and integrated them to form a multimodal saliency curve. They used spatiotemporal saliency model, an AM-FM speech model, and Part of Speech (POS) tagging to extract features from visual, aural, and textual module, respectively. Aparicio et al. [8] summarized movies and documentaries of different genres by analyzing six different text summarization algorithms on movie scripts and subtitles. The key contribution of their work is selecting a method that best fits among the six techniques for a movie or documentary of a particular genre. Another text based movie summarization scheme presented by Hesham et al. [9] generated a short summary as a trailer using subtitles of the movies. Hang Do et al. [10] summarized movies based on developing characters network. The relationships between characters are based on their appearance, which is used to segment the full-length movie into scenes. Finally, the storyline of the movie is generated as a summary by measuring the social strength of each character in the social network.

Due to diverse nature of movies and contradiction between user's preferences, the generated summary using automatic MS techniques may be felicitous for one user, but it may be infelicitous to others. Such MS schemes do not have strength to generate a summary that can fulfill the diverse subjective requirements of users. Therefore, user's preferred shots selection for movie summary is still a challenging problem, which is well addressed by user-preference based MS techniques. For example, Li et al. [11] suggested that substories from movies can be detected using short- and long-term audio-visual temporal features analysis. The length of generated summary is controlled by user input. Similarly, Ellouze et al. [12] used audio-visual features for personalized movie summary, where user can choose contents and type of various shots along with the duration of the summary. However, the comparison of user-preferred contents and movie contents is performed at feature level rather than semantic level. Peng et al. [13] utilized the emotion and attention of a viewer to generate summary according to user's mood by analyzing his facial expression, eye movement, blink, and head motion while watching a video. A movie summarization scheme based on user generated data is presented by Sun et al. [15]. They used real-time comments given by audience on the timestamp of a movie. The contents of the comments show the concepts of the ongoing scene, while the number of the comments indicates excitement level of the audience. Concluding the MS literature, movies are richer sources, providing semantics of complex ideas through audio-visual data. Therefore, segmenting a movie based on semantic level features gives best baseline for MS. Human emotions are one of the semantic level information, which can be extracted from video contents. FER has various applications in different domains such as medical [21], content recommendation [22], surveillance [23], safety [24], and robotics [25]. Similarly, in movies data, emotions of characters are the prominent element that directly gets audience's attention, which can be exploited to generate a meaningful summary.

Recently, a lot of research has been done for human emotion recognition using facial expression analysis. A comprehensive survey on FER is presented by Corneanu et al. [26] for RGB, 3D, thermal, and multimodal schemes. Traditional facial feature extraction schemes such as Gabor wavelet transform [27], optical flow [28], local binary patterns [29], and model-based methods [30] have many limitations like high computation and low performance due to diverse environments, i.e., light changes, pose, and clutter background. Moreover, these schemes are restricted to frontal faces and uniform skin colors. Recently, deep learning technologies [31] have shown tremendous results in the field of computer vision compared to traditional approaches. For instance, Kim et al. [32] proposed a hierarchical committee of deep CNNs by combining the decisions of multiple models trained on public FER databases. A feature redundancy-reduced (FRR-CNN) is proposed by Xie et al. [33] for FER to generate less redundant features and compact representation of the image. Uddin et al. [34] extracted local directional position patterns from depth video data and fed them into a deep belief network (DBN) for FER. Inspired from CNN approaches, in this paper, we proposed a user-preference based MS scheme, based on FER to determine the emotional state of the characters using CNN model. The user should choose the kind of emotional states that he prefers to be part of the final summary. The main contributions of this work are summarized as follows:

(1) Emotions of characters in a movie are the prominent elements that directly get audience's attention. Therefore, users always show interest in certain kind of emotional scenes in a movie. In this paper, we present a framework for generating summary based on user's preferred emotional scenes via an input query during summary generation.

(2) The hierarchical structure of a movie assists generating a sensible summary in which shots segmentation plays an important role. Therefore, we propose an entropy-based shots segmentation mechanism, which segments shots based on visual information. This strategy helps categorize shots into informative and noninformative.

(3) Deep learning approaches need huge data for better learning of parameters, whereas existing FER CNN models are not trained on such huge data. Therefore, for precise training of FER first we trained the model from the scratch over VGG face dataset to learn the structure of face and then we fine-tuned this model for FER on KEDF dataset. Our strategy gives prominent results over state-of-the-art techniques.

The rest of paper is arranged as follows: Section 2 discusses the proposed methodology for movie summarization in detail. Experimental results and discussion are presented in Section 3, followed by the conclusion and future work in Section 4.

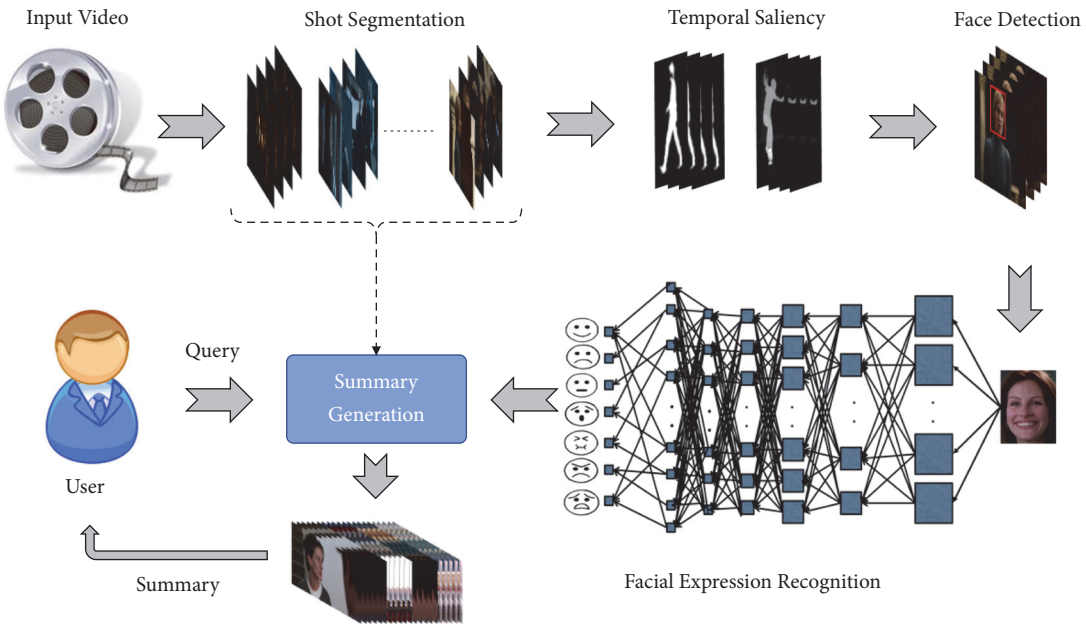


Figure 1: Overall framework of the proposed movie summarization scheme.

2. Proposed Methodology

Our proposed MS scheme is four folded: (1) entropy-based shots segmentation, (2) saliency extraction and face detection, (3) FER based on deep CNN model, and (4) summary generation. All the steps are discussed in subsequent sections in detail. The proposed system can generate summaries based on user-preference for any genre of movie. The overall framework of the proposed scheme is given in Figure 1.

2.1. Entropy-Based Shots Segmentation. Movies are also called structured videos because they include hierarchical structure of scenes and shots. A single shot is an uninterrupted segment of a movie that consists of sequential frames with static or continuous camera motion, while a scene consists of one or more shots of the same place or activity captured from different angles [1]. This structure assists in MS at initial stage by segmenting the full-length movie into shots and scenes. Shots segmentation is a key step for any summarization technique, especially, when dealing with entertainment videos. Recently, numerous domain specific shots segmentation techniques have been proposed such as color histogram based [15], deep feature based [35], person appearance based [31], and spare coding based methods [36]. In this paper, we proposed entropy-based shots segmentation technique, which analyzes frame sequences and selects the frame with sharp change in visual contents. Entropy E_N for a single frame can be calculated using (1) and (2).

$$p_i = \frac{n_i}{N_c} \tag{1}$$

$$E_N = -\sum_{i=1}^{N_c} p_i \log\left(p_i\right) \tag{2}$$

Herein, p_i is the probability of pixel, $q_i \in N_C$, N_C is the number of pixels in the neighborhood of pixel q_i, and n_i is the number of pixels having same intensities. Entropy of a frame represents the amount of information and semantics of the visual contents. Thus, it helps categorize the shots into informative and noninformative. Furthermore, the generated summary represents the informative shots only by excluding the shots with no or less information.

2.2. Saliency Extraction and Face Detection. Generally, saliency of an image is used to extract the foreground information, which can also be used for predicting the amount of information in it [37, 38]. To select the most informative shots, we calculated the average saliency score of a single shot using a saliency optimization technique [39]. Firstly, saliency map of a frame is obtained and the sum of all nonzero pixels is divided by the total number of image pixels. Secondly, average saliency score for a single shot is calculated by dividing the total sum of individual frame by total number of frames in a shot. Finally, the average saliency score is compared with a predefined threshold to select the most salient shots. In this way, all the nonsalient or noninformative shots are discarded, and the salient shots become the part of generated summary. The salient shots are further analyzed to detect characters' faces. For face detection, a multitask cascaded network [40] is used with additional constraint of size. The size constraint for a face is applied due to variation found in scale and poses to remove unwanted faces. Therefore, we selected only those faces, which are 15% of the frame size because main characters are filmed focused and closed. Also, FER is not working perfectly for small-sized faces [41]. Figure 2 represents some sample movie frames with detected faces and their corresponding saliency maps.

(a)

(b)

Figure 2: Sample frames from test movies: (a) highly salient shots with detected actor's faces and saliency maps and (b) low salient shots with detected actor's faces and saliency maps.

2.3. Facial Expression Recognition Using Transfer Learning. Training a deep CNN model requires a huge amount of data for learning its parameters from the scratch. However, transfer learning becomes a key concept in deep learning since it effectively deals with the problems having small datasets [42, 43]. Recently, CNN has beaten human error on image classification, when trained on a dataset with millions of data samples. However, some tasks such as FER are still facing the lack of data. Therefore, to tackle this problem, we used the concept of transfer learning using ResNet [20] CNN model for FER. ResNet CNN model has many versions including ResNet-34, ResNet-50, ResNet-101, and ResNet-152 layers networks. We have utilized ResNet-50 to balance the accuracy and time complexity of the system. Originally, it is trained on 224×224 images of ImageNet [44] dataset, which contains millions of samples for 1000 categories. We did not achieve good results when using weights of the pretrained ResNet-50 model for fine-tuning it on KDEF FER dataset [45]. The reason is that we have only face images that represent emotion of a human with very little changes on face in KDEF dataset and the model, which is pretrained on general categories data, is not effective for it. For this purpose, we introduced two step learning procedure where the first step includes training a CNN model from the scratch for face identification and second is transfer learning of the same model for FER. For face identification, we used large-scale VGG face [46] dataset to train weights of ResNet-50 for face data. The ResNet is primarily inspired by the structure of VGG [47] model, where both models use small size kernels for convolutional features extraction. The small-sized filters help learn all kind of tiny patterns in data, which are very common in FER [48]. ResNet utilized multiple consecutive branches of convolutional layers stacked on each other and performed down sampling with stride of 2. The network is ended with a global average pooling layer and one fully connected layer, presenting the number of classes for classification. The architecture of 50 weighted layers is given in Figure 3. The implementation details about transfer learning are discussed in the experimental section.

2.4. Summary Generation. The shots with high saliency and faces are forward propagated to the proposed trained CNN model for FER. In our experiments, we observed that FER from a single frame of a shot is not effective in representing the emotional state of the entire shot. Also, there is a possibility that a shot may contain multiple faces. Hence, the maximum detected emotion is selected as emotional state for the entire shot. Finally, the summary is generated according to the user's query specifying emotional state from the predefined seven classes of emotions. The flow of summary generation is visualized in Figure 4.

3. Experimental Evaluation

In this section, we have discussed the experimental evaluation of the proposed MS scheme. We performed two sets

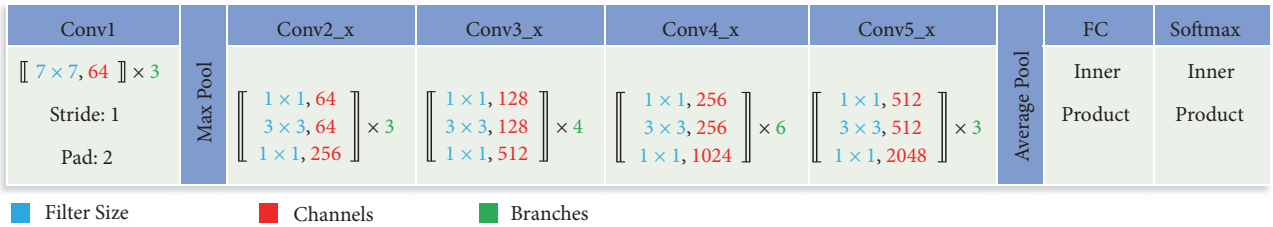


Figure 3: Architecture of 50-layer residual CNN model, which is changed for 128×128 face images. Conv3_1, Conv4_1, and Conv5_1 are down sampled using max pooling with stride of 2.

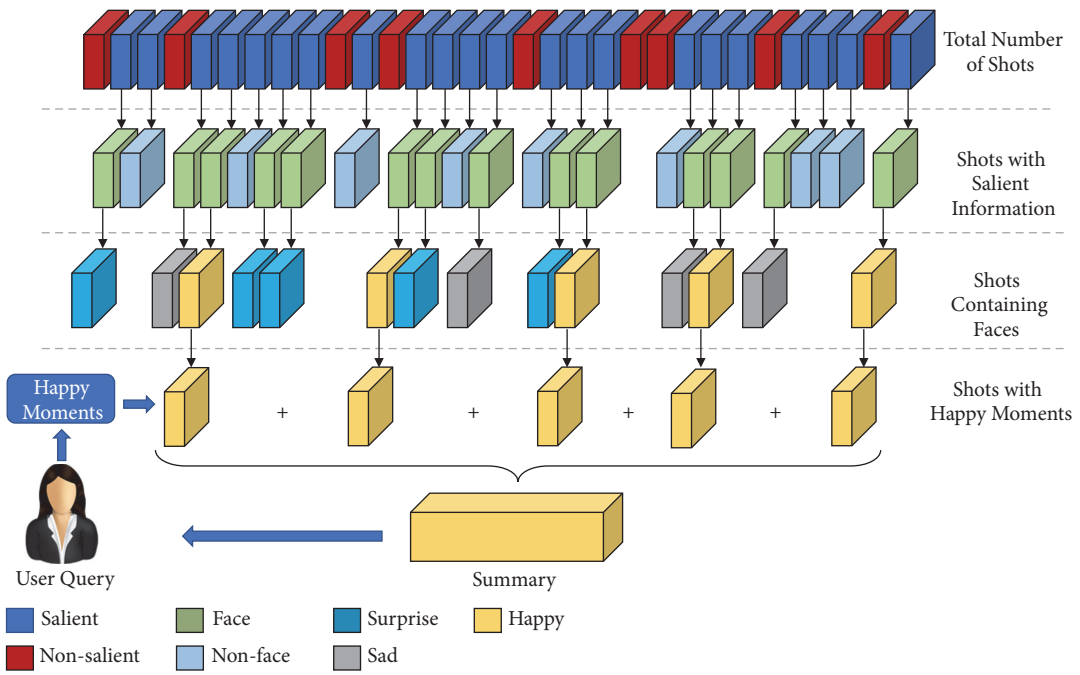


Figure 4: Flow diagram of summary generation based on user-preferences.

of experiments: (1) an evaluation of the trained model on KDEF dataset and its comparison with other models and (2) a subjective evaluation on five Hollywood movies of different genres. The experiments are performed using a deep learning framework known as Caffe [49] that is installed over Ubuntu16.04 operating system and equipped with NVIDIA TITAN X GPU having 12 GB dedicated memory running over a hardware of Intel™ Core i5 CPU with 64 GB RAM.

3.1. Datasets. We have used two datasets: VGG face [46] to train ResNet CNN model from scratch and KDEF dataset [45] for transfer learning. VGG face dataset contains 2.6 M images of 2.6 K celebrities around the world. KDEF dataset contains 4900 images of 70 subject including 35 males and 35 females. This dataset contains 7 classes, i.e., afraid, angry, disgust, happy, neutral, sad, and surprise. For each class, image samples are taken from five different angles in two sessions. This dataset best fits for our problem because, in movies, characters' faces are also found in a variety of poses. Figure 5 represents some sample images from each class of KDEF dataset. For subjective evaluation, five Hollywood movies are used. The detailed description of test movies is given in Table 1.

3.2. Objective Evaluation of Facial Expression Recognition. The ResNet CNN model is first trained on VGG face [46] dataset having 2597 classes. We resized all the images to 128×128 face regions and each pixel of the image is subtracted from the mean image to normalize the intensities. The full VGG face dataset is filtered out and some images were discarded in training. The dataset used in our experiments contains 0.42 million training and 0.14 million validation images. The original ResNet-50 is trained on 224×224 images; therefore, if we use pretrained ResNet-50, then its kernel size, stride, and padding information is not fitting for 128×128 image classification problem. In our method, the weights of ResNet for 128×128 face images are initialized from the scratch. The detailed description can be seen in the original ResNet article [20]. The model is trained for 50 epochs with 64 batch size and the learning rate is initialized with 0.01, which is decreased after every 10 epochs by the factor of learning rate ratio 10. The reason behind decreasing the

Figure 5: Sample images from each class of KDEF dataset.

Table 1: Description of the test movies data.

Movie ID	Movie title	Genre	Length (min)	Number of shots	Number of scenes
TLH	The Lake House	Fantasy/Drama	105	865	76
MBBN	My Blueberry Nights	Romance/Drama	90	1009	73
YHGM	You've Got Mail	Comedy/Romance	119	1049	62
SA	Salt	Action/Adventure	100	2339	78
NH	Notting Hill	Comedy/Romance	124	1623	42

Table 2: Transfer learning results of different CNN model for KDEF dataset.

CNN models	Overall accuracy (%)
MobileNet [16]	40.65
SqueezNet [17]	45.37
AlexNet [18]	46.81
GoogleNet [19]	52.63
ResNet-50 (224×224) [20]	64.83
ResNet-50 (128×128)	*93.65*

learning rate is to prevent the model from overfitting problem during the training. We achieved precise results on VGG face dataset using ResNet-50 layers network, where the accuracy after 50 epochs reached 96.82% and the loss decreased up to 7×10^{-5}. Results of different CNN models fine-tuned on KDEF datasets are given in Table 2. It is clear from Table 2 that all the pretrained models have achieved very less accuracy when fine-tuned on the original weights. The reason of low accuracy is that these models are trained on general categories dataset and we need a model whose parameters are previously trained on face data. Therefore, first we trained ResNet-50 using large-scale VGG face dataset to learn face structure and then fine-tuned the trained model on KDEF dataset for FER.

After training ResNet with face recognition dataset, we claim that its weights can now learn face features and its structure effectively. Therefore, we have used the parameters of the trained model for transfer learning of FER using KDEF dataset. For fine-tuning process, all images of the KDEF dataset are resized to 128×128 face regions and each pixel of image is subtracted from mean image to normalize the intensities. In transfer learning process, we have initialized the learning rate from 0.001 and decreased it after each 10 epochs by the factor of learning rate ratio 10. The model is fine-tuned for 30 epochs, achieving 92.08% validation accuracy with loss of 0.192 at final epoch. Confusion matrix and overall accuracy for the test set of KDEF dataset are given in Table 3. All categories are not much confused with each other, i.e., afraid, angry, and sad classes achieved per-class accuracy under 90% while the rest of all classes have accuracy above 90%. The results for this dataset are very convincing, making our trained model capable of FER in the heterogamous movie data. The KDEF dataset has various categories of face poses and viewpoint variations, which help to easily analyze the character's facial expressions in the movie.

3.3. Subjective Evaluation of Generated Summary. One of the challenging steps in movie summarization is the evaluation of the generated summary due to the lack of standards. Generally, there are two types of assessment used in video summarization literature that can be categorized into intrinsic and extrinsic techniques. In intrinsic evaluation, the generated summary is directly analyzed from its contents. For instance, fluency in generated summary, coverage of the main theme of original video, and similarity with referenced summary generated by movies expert are checked. In extrinsic evaluation, the performance is evaluated as information retrieval problem using a multichoice questionnaire. The excellence of summary is then measured by the increase in quiz scores. In this paper, we followed the second technique because it generates summary based on user's query to select emotional shots of a specific class. In our experiments, a total of ten subjects participated in the subjective evaluation, in which six students are selected from graduate and four from undergraduate program having age in the range from 20 to 25 years. All the participants were instructed to watch the selected movies before the evaluation and asked to rate the following three questions between 1 and 10 after watching the summary generated by their desired query. Table 4 represents the statistics of all the detected emotions in the informative shots of the test movies.

Table 3: Confusion matrix and overall accuracy for the test set of KDEF dataset.

	Afraid	Angry	Disgust	Happy	Neutral	Sad	Surprise	Per-class accuracy (%)
Afraid	*51*	1	1	1	0	1	7	82.26
Angry	1	*62*	0	0	1	6	0	88.57
Disgust	0	1	*73*	0	0	2	0	96.05
Happy	0	0	2	*71*	0	0	0	98.03
Neutral	1	1	0	0	*70*	1	2	93.33
Sad	3	1	1	1	3	*64*	0	87.67
Surprise	1	0	0	0	0	0	*43*	97.73
				Overall accuracy				**93.65**

Table 4: Statistics of all the detected emotions in the informative shots of the test movies.

Movie ID	Number of shots detected in each emotion category						
	Afraid	Angry	Disgust	Happy	Neutral	Sad	Surprise
TLH	12	17	0	24	113	87	0
MBBN	19	13	1	18	211	48	6
YHGM	9	16	2	46	186	38	4
SA	13	20	0	4	102	72	2
NH	128	101	3	157	389	203	12

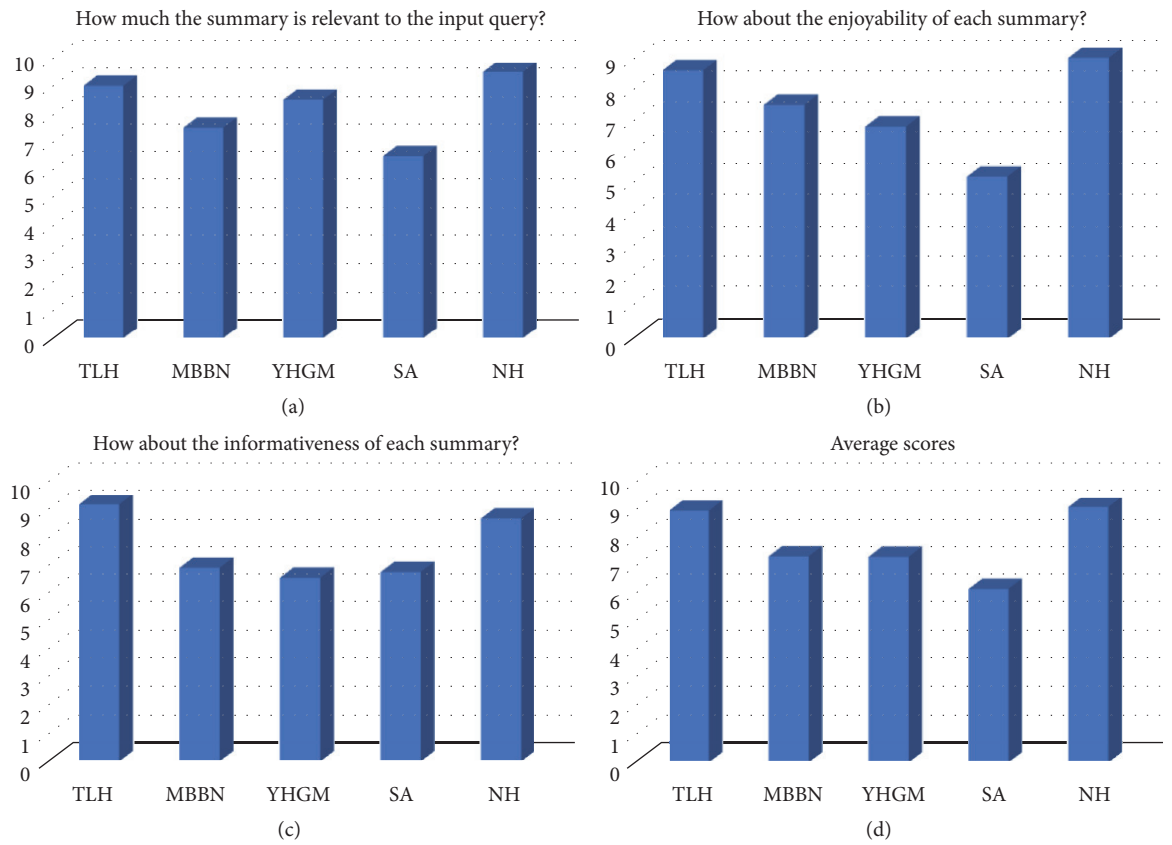


Figure 6: Subjective evaluation of the generated summaries: (a), (b), and (c) represent average scores for Q 1, Q 2, and Q 3, respectively, and (d) shows the average scores of all the three questions.

Figure 7: Sample shots from movie "*Notting Hill*" with emotional states predicted by our proposed scheme.

Q 1: How much the summary is relevant to the input query?

Q 2: How about the enjoyability of each summary?

Q 3: How about the informativeness of each summary?

In Figure 6, the average scores of all the participants for each test movie are calculated and represented in the form of graphs. It is clear from Figure 6 that all the test movies give good results except "*Salt*", which is an action movie. In action movies, the human movements are fast, making the task of FER very challenging due to blur effects. Our proposed scheme achieved best results for movie "*Notting Hill*", which is a romantic movie, containing very rich emotions. Figure 7 represents some shots from movie "*Notting Hill*" with corresponding emotional state of the shots. Concluding the overall evaluation and discussion, we claim that the performance of our proposed scheme is best on movies of genre drama, comedy, romance, and fantasy compared to action and adventures.

4. Conclusion

In this article, we presented a user-preference based movie summarization scheme. First, we segmented the movie into shots using a novel entropy-based shots segmentation mechanism. Secondly, we computed temporal saliency for each shot to discard nonsalient shots. Next, character's faces are detected in the salient shots and fed into a deep CNN model for FER. Finally, the summary is generated according to the user's query of any emotion state from the predefined seven classes. We evaluated our trained model for FER and the overall proposed scheme of movie summarization using objective and subjective analysis. We found that our proposed scheme demonstrates better performance compared to other movies summarization techniques. In future, we aim to conduct experiments on animated movies and fuse both aural and visual features for movies summary generation.

Data Availability

All the datasets and testing movies are publicly available. We provided the citation of each dataset and the movies are downloaded from online databases, i.e., YouTube and Crackle. The python code for analysis and the trained model will be made available to readers upon request.

Conflicts of Interest

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

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07043302).

References

[1] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang, "Video summarization and scene detection by graph modeling," *IEEE Transactions on*

Circuits and Systems for Video Technology, vol. 15, no. 2, pp. 296–304, 2005.

[2] J. You, G. Liu, L. Sun, and H. Li, "A multiple visual models based perceptive analysis framework for multilevel video summarization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 273–285, 2007.

[3] C.-Y. Weng, W.-T. Chu, and J.-L. Wu, "RoleNet: Movie analysis from the perspective of social networks," *IEEE Transactions on Multimedia*, vol. 11, no. 2, pp. 256–271, 2009.

[4] H. Salamin, S. Favre, and A. Vinciarelli, "Automatic role recognition in multiparty recordings: Using social affiliation networks for feature extraction," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1373–1380, 2009.

[5] J. Sang and C. Xu, "Character-based movie summarization," in *Proceedings of the 18th ACM International Conference on Multimedia ACM Multimedia 2010, MM'10*, pp. 855–858, October 2010.

[6] G. Evangelopoulos, A. Zlatintsi, A. Potamianos et al., "Multimodal saliency and fusion for movie summarization based on aural, visual, and textual attention," *IEEE Transactions on Multimedia*, vol. 15, no. 7, pp. 1553–1568, 2013.

[7] C.-M. Tsai, L.-W. Kang, C.-W. Lin, and W. Lin, "Scene-based movie summarization via role-community networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 11, pp. 1927–1940, 2013.

[8] M. Aparício, P. Figueiredo, F. Raposo, D. Martins De Matos, R. Ribeiro, and L. Marujo, "Summarization of films and documentaries based on subtitles and scripts," *Pattern Recognition Letters*, vol. 73, pp. 7–12, 2016.

[9] M. Hesham, B. Hani, N. Fouad, and E. Amer, "Smart trailer: Automatic generation of movie trailer using only subtitles," in *Proceedings of the 1st International Workshop on Deep and Representation Learning, IWDRL 2018*, pp. 26–30, 2018.

[10] T. T. Do, Q. H. Tran, and Q. D. Tran, "Movie indexing and summarization using social network techniques," *Vietnam Journal of Computer Science*, vol. 5, no. 2, pp. 157–164, 2018.

[11] Y. Li, S.-H. Lee, C.-H. Yeh, and C.-C. J. Kuo, "Techniques for Movie Content Analysis and Skimming Tutorial and overview on video abstraction techniques," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 79–89, 2006.

[12] M. Ellouze, N. Boujemaa, and A. M. Alimi, "IM(S)2: Interactive movie summarization system," *Journal of Visual Communication and Image Representation*, vol. 21, no. 4, pp. 283–294, 2010.

[13] W.-T. Peng, W.-T. Chu, C.-H. Chang et al., "Editing by viewing: Automatic home video summarization by viewing behavior analysis," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 539–550, 2011.

[14] R. Kannan, G. Ghinea, and S. Swaminathan, "What do you wish to see? A summarization system for movies based on user preferences," *Information Processing & Management*, vol. 51, no. 3, pp. 286–305, 2015.

[15] Z. Li, X. Liu, and S. Zhang, "Shot Boundary Detection based on Multilevel Difference of Colour Histograms," in *Proceedings of the 1st International Conference on Multimedia and Image Processing, ICMIP 2016*, pp. 15–22, June 2016.

[16] A. G. Howard, M. Zhu, B. Chen et al., *Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, 2017, https://arxiv.org/abs/1704.04861.

[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, *SqueezeNet: Alexnet-Level Accuracy with 50x Fewer Parameters and < 0.5 MB Model Size*, 2016, https://arxiv.org/abs/1602.07360.

[18] 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, 2012.

[19] 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, June 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 770–778, July 2016.

[21] L. Hazelhoff, J. Han, S. Bambang-Oetomo, and P. H. de With, "Behavioral state detection of newborns based on facial expression analysis," in *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 698–709, 2009.

[22] L. Canini, S. Benini, and R. Leonardi, "Affective recommendation of movies based on selected connotative features," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 4, pp. 636–647, 2013.

[23] M. Sajjad, M. Nasir, F. U. M. Ullah, K. Muhammad, A. K. Sangaiah, and S. W. Baik, "Raspberry Pi assisted facial expression recognition framework for smart security in law-enforcement services," *Information Sciences*, 2018.

[24] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlett, and J. Movellan, "Drowsy driver detection through facial movement analysis," in *Proceedings of the International Workshop on Human-Computer Interaction*, pp. 6–18, 2007.

[25] L. Zhang, M. Jiang, D. Farid, and M. A. Hossain, "Intelligent facial emotion recognition and semantic-based topic detection for a humanoid robot," *Expert Systems with Applications*, vol. 40, no. 13, pp. 5160–5168, 2013.

[26] C. A. Corneanu, M. O. Simón, J. F. Cohn, and S. E. Guerrero, "Survey on RGB, 3D, thermal, and multimodal approaches for facial expression recognition: history, trends, and affect-related applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1548–1568, 2016.

[27] M. Tkalcic, A. Odic, and A. Kosir, "The impact of weak ground truth and facial expressiveness on affect detection accuracy from time-continuous videos of facial expressions," *Information Sciences*, vol. 249, pp. 13–23, 2013.

[28] C.-K. Hsieh, S.-H. Lai, and Y.-C. Chen, "An optical flow-based approach to robust face recognition under expression variations," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 233–240, 2010.

[29] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: a comprehensive study," *Image and Vision Computing*, vol. 27, no. 6, pp. 803–816, 2009.

[30] M. A. A. Dewan, E. Granger, G.-L. Marcialis, R. Sabourin, and F. Roli, "Adaptive appearance model tracking for still-to-video face recognition," *Pattern Recognition*, vol. 49, pp. 129–151, 2016.

[31] A. Ullah, K. Muhammad, J. Del Ser, S. W. Baik, and V. Albuquerque, "Activity recognition using temporal optical flow convolutional features and multi-layer LSTM," *IEEE Transactions on Industrial Electronics*, 2018.

[32] B.-K. Kim, J. Roh, S.-Y. Dong, and S.-Y. Lee, "Hierarchical committee of deep convolutional neural networks for robust facial expression recognition," *Journal on Multimodal User Interfaces*, vol. 10, no. 2, pp. 173–189, 2016.

[33] S. Xie and H. Hu, "Facial expression recognition with FRR-CNN," *IEEE Electronics Letters*, vol. 53, no. 4, pp. 235–237, 2017.

[34] M. Z. Uddin, M. M. Hassan, A. Almogren, A. Alamri, M. Alrubaian, and G. Fortino, "Facial expression recognition utilizing local direction-based robust features and deep belief network," *IEEE Access*, vol. 5, pp. 4525–4536, 2017.

[35] K. Muhammad, T. Hussain, and S. W. Baik, "Efficient CNN based summarization of surveillance videos for resource-constrained devices," *Pattern Recognition Letters*, 2018.

[36] J. Li, T. Yao, Q. Ling, and T. Mei, "Detecting shot boundary with sparse coding for video summarization," *Neurocomputing*, vol. 266, pp. 66–78, 2017.

[37] D. Zhang, J. Han, L. Jiang, S. Ye, and X. Chang, "Revealing event saliency in unconstrained video collection," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1746–1758, 2017.

[38] D. Zhang, H. Fu, J. Han, A. Borji, and X. Li, "A review of co-saliency detection algorithms," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, p. 38, 2018.

[39] W. Zhu, S. Liang, Y. Wei, and J. Sun, "Saliency optimization from robust background detection," in *Proceedings of the27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 2814–2821, June 2014.

[40] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[41] I. U. Haq, K. Muhammad, A. Ullah, and S. W. Baik, "DeepStar: detecting starring characters in movies," *IEEE Access*, vol. 7, pp. 9265–9272, 2019.

[42] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2017.

[43] M. Seera and C. P. Lim, "Transfer learning using the online fuzzy min–max neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 469–480, 2014.

[44] J. Deng, W. Dong, R. Socher et al., "ImageNet: a large-scale hierarchical image database," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 248–255, 2009.

[45] M. G. Calvo and D. Lundqvist, "Facial expressions of emotion (KDEF): Identification under different display-duration conditions," *Behavior Research Methods*, vol. 40, no. 1, pp. 109–115, 2008.

[46] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference 2015*, p. 6, 2015.

[47] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014, https://arxiv.org/abs/1409.1556.

[48] K. Muhammad, J. Ahmad, and S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management," *Neurocomputing*, vol. 288, pp. 30–42, 2018.

[49] Y. Jia, E. Shelhamer, J. Donahue et al., "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the ACM Conference on Multimedia (MM '14)*, pp. 675–678, 2014.

Hindawi
Complexity
Volume 2019, Article ID 5498606, 12 pages
https://doi.org/10.1155/2019/5498606

Research Article

Deep Learning Based Proactive Caching for Effective WSN-Enabled Vision Applications

Fangyuan Lei,[1,2] **Jun Cai**,[1] **Qingyun Dai**,[1,2] **and Huimin Zhao**[3]

[1]*School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou 510640, China*
[2]*School of Information Engineering, Guangdong University of Technology, Guangzhou, China*
[3]*School of Computer Sciences, Guangdong Polytechnic Normal University, Guangzhou 510640, China*

Correspondence should be addressed to Jun Cai; 597069873@qq.com and Huimin Zhao; zhaohuimin@gpnu.edu.cn

Received 15 November 2018; Revised 28 February 2019; Accepted 14 March 2019; Published 2 May 2019

Guest Editor: Jungong Han

Wireless Sensor Networks (WSNs) have a wide range of applications scenarios in computer vision, from pedestrian detection to robotic visual navigation. In response to the growing visual data services in WSNs, we propose a proactive caching strategy based on Stacked Sparse Autoencoder (SSAE) to predict content popularity (PCDS2AW). Firstly, based on Software Defined Network (SDN) and Network Function Virtualization (NFV) technologies, a distributed deep learning network SSAE is constructed in the sink nodes and control nodes of the WSN network. Then, the SSAE network structure parameters and network model parameters are optimized through training. The proactive cache strategy implementation procedure is divided into four steps. (1) The SDN controller is responsible for dynamically collecting user request data package information in the WSNs network. (2) The SSAEs predicts the packet popularity based on the SDN controller obtaining user request data. (3) The SDN controller generates a corresponding proactive cache strategy according to the popularity prediction result. (4) Implement the proactive caching strategy at the WSNs cache node. In the simulation, we compare the influence of spatiotemporal data on the SSAE network structure. Compared with the classic caching strategy Hash + LRU, Betw + LRU, and classic prediction algorithms SVM and BPNN, the proposed PCDS2AW proactive caching strategy can significantly improve WSN performance.

1. Introduction

According to the latest Cisco release of the Visual Network Index (VNI) [1] forecast, the number of devices connected to the Internet of Things (IoT) is projected to expand to somewhere between 20 and 46 billion by 2021. Thanks to the simple deployment and various practical applications, the potential benefits of wireless sensor networks (WSNs) concern a wide range of application scenarios, from pedestrian detection to robot vision navigation, from industrial systems to home appliances. While billions of new devices connect to the network in a short period of time in WSNs, there will have huge data traffic. Therefore, congestion control and data share should be considered. As Figure 1 shows, the WSN network is deployed for traffic monitor, which composed of different network protocols that cannot be directly connected, and it is difficult to achieve fast sharing of sensing data to meet the performance requirements of the system.

For specific applications, once the distributed WSN is deployed for traffic monitor, the sensor node handles not only the sensing tasks but also maintenance of the route status. With the solidified resource management mode, when the upper-layer application requirements change, it is very difficult to apply flexible changes according to the new requirements. Therefore, it is seriously wasting resources without realizing dynamic perception.

Software-defined network (SDN) [2] as a new type of network architecture attracting attention in recent years is applied to future networks. The core idea of SDN is the separation of control plane and forwarding plane. The control plane is aware of network status and network resources, and then the central controller flexibly and dynamically configures the logic control functions and high-level policies of the network. On the data plane, this configuration can be performed without affecting the normal network traffic. Network Function Virtualization (NFV) [3] applies the goal

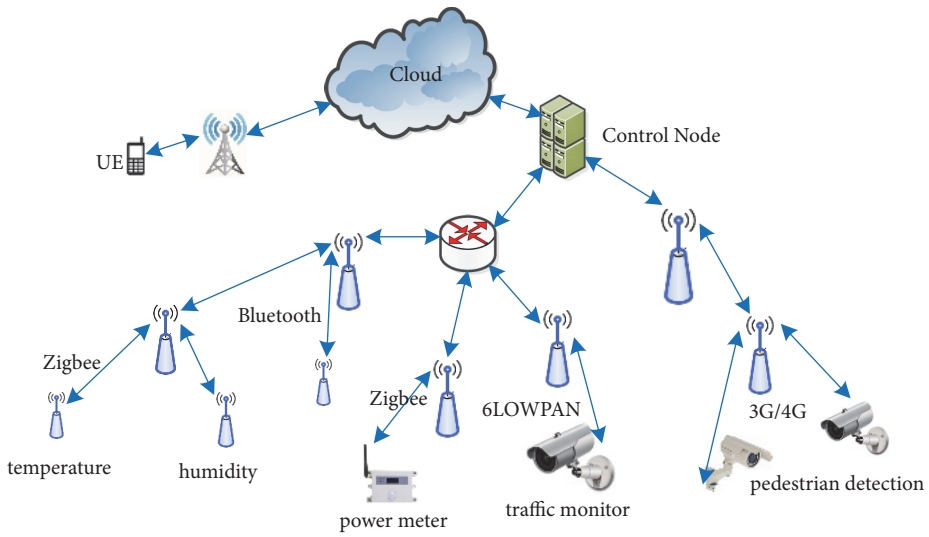


Figure 1: WSNs application in traffic monitor.

of automated management and distributed deployment by introducing virtualization technology. SDN and NFV are introduced to improve the performance in WSNs [4].

To address the congestion issue, proactive caching [5, 6], which wildly used in ICN, is introduced into the WSNs [7]. Data package caching will improve the performance of WSNs, including reducing packet latency, network traffic, and BER of transmission. Recently, proactive caching based on content popularity prediction of Deep Learning (DL), has attracted widespread attention in academia and industry [8–13], which will significantly benefit cache efficiency. Content popularity prediction based on DL can automatically find the rule from the data and use this rule to predict the unknown data. Therefore, in WSN data, caching would significantly improve resource utilization and network performance.

Therefore, we propose an efficient proactive cache strategy based on distributed stacked sparse autoencoder in WSNs (PCDS2AW) for data package content popularity prediction. Firstly, NFV functions are used to virtual part of hardware resources in the control node and sink nodes and routers of general hardware. Then, based on those virtual hardware resources, a distributed deep learning network, SSAEs, is constructed. Next, the SDN controller works in conjunction with global cache resources and SSAEs. At the WSNs network level, the SDN controller can dynamically sense the cache utilization information and periodically collect user data packet request information. These historical requests and real-time requested packet information are input into the distributed SSAE for content popularity prediction. Finally, based on the data package, popularity prediction the SDN controller generates the proactive cache strategy and synchronizes it to the WSNs cache nodes through the SDN flow table to implement content caching and replacement.

As a virtual technology, NFV can provide support for reference machine learning and can achieve seamless cooperation with SDN [4]. SDN uses stream-based data forwarding, with a focus on data exchange and forwarding, while WSN is constrained by the specific type of sensor deployed, which is essentially data-oriented. Therefore, based on NFV, machine learning algorithms are used to predict network traffic, and SDN realizes fast forwarding and complements WSN's efficient perception.

In order to implement cache strategy based on the global data package content popularity prediction, the SDN controller should know the popularity of all requested data package information in WSNs. Therefore, the SDN/NFV technology is used in upper-layer include sink nodes, control nodes, and router to construct some virtual content request and statistics servers. These statistics servers are configured to collect the request data package information from the WSN nodes. The content request information is aggregated to a virtual global content request statistics server. Therefore, the control node collects a large amount of global data package request information and uses the deep learning algorithm to build a prediction content popularity model and then uses the prediction model to guide the WSNs for efficient caching to reduce traffic.

The main contributions of this paper are as follows: (1) in the WSNs, we propose a method for constructing a distributed stack sparse autoencoder deep learning network; (2) SSAEs use the spatiotemporal information of user request data packets to predict the data packet popularity; (3) under the cooperation of SDN controller, cache nodes implement the proactive cache and replacement of the data packet content of the whole network, which makes the utilization of cache resources more reasonable; (4) simulation results show that, compared with Betw [14], Hash [15], and Opportunistic [16], the proposed proactive caching strategy could improve the performance of WSNs.

The remainder of this article is structured as follows. In Section 2, related works are described. In Section 3, the system model is proposed. In Section 4, some evaluation metrics and the numerical simulation results are discussed. Finally, we conclude in Section 5.

2. Related Works

In order to meet the needs of the interconnection of all kinds of objects, the distributed WSN for application design is beginning to transform to support heterogeneous interconnection, which is one of the root causes of SDN introduction of WSN.

In 2012, Luo [17]and Mahmud [18] discussed the integration of SDN and WSN almost simultaneously and made important contributions to the birth of SDWSN. Zeng et al. [19] proposed a software-defined sensor networks (SDSN) architecture that supports "sensing as a service" combined with cloud computing, in which architecture and the controllers are wirelessly used for different sensing tasks. In [20], the author proposed Elon system which implements the function of dynamically modifying the sensing node code by means of replaceable components. Subsequently, the author's team further implemented the transmission on the SDSN. In [21], based on a new event-triggered strategy, the author proposed an event-triggered distributed multisensor data fusion algorithm for wireless sensor networks (WSNs). In [22], the authors deployed multiple controllers to build a more flexible control channel, which greatly reduced the average network control delay. The dynamic redefinition function of the node function [23] could effectively extend the network lifetime. In [24], the authors proposed a general architecture for implementing the controller on the WSN base station to enhance the intelligent management of the network. In [25], the authors defined the cluster head as a switch in the wired network, and the only controller is deployed in the WSN base station to save the sensor node energy. 0The author [26] proposed UbiFlow, which was a software-defined IoT system for ubiquitous flow control and mobility management in a large number of heterogeneous WSNs.

SDN and NFV are not combined standard, and they provide different aspects of network-based services. Innovative network paradigms SDN and NFV have attracted the interest of IoT network operators and service providers. Some researchers have introduced SDN into the IOT network architecture, such as SDN-WISE [27], SDWN [28], TinySDN [22], and UbiFlow [26]. However, many of these architectures lacked virtualization methods. In [4], in order to quickly respond to the challenges brought by flexible deployment of the Internet of Things, the author combined NFV technology with SDN technology and proposed a general SDN-IoT architecture. In [29], to verify and test the 6LoWPAN testbed, a customized Software Defined–Network Functioning Virtualisation (SD–NFV) is proposed.

As a kind of technology to improve network performance, cache technology is widely used in future Internet and communication. Recently, caching has also been a concern in the Internet of things. In [30], the author introduced the in-network caching to the IOT. Firstly, the author defined the lifetime of IOT transmission data, which was determined by application time and location tags, then a trade-off model between multi-hop cost and data freshness was proposed and applied to the content router of the Internet of things. In [31], to adapt highly dynamic environments with a coarse knowledge of car trajectories, the author proposed a Mobility-Aware Probabilistic (MAP) edge caching strategy. In [32], to enhance the service in mobile ad hoc networks (MANNET), the authors proposed a comprehensive strategy which includes cache placement, cache discovery, cache consistency, and cache replacement algorithms. In [33], the author developed a dynamic source rate control algorithm based on cache awareness. Depending on the network traffic, the rate control algorithm used a cache management policy (such as a cache elimination policy and a cache size allocation) to move the transmission window so that packet loss may be mitigated during high speed. In [34], the caching mechanism which was applied to the transport protocol would effectively reduce the number of end-to-end retransmissions, node power consumption, and packet delay to improve network performance. In [35], the author had comprehensive evaluation of cache utilization characteristics with the cache replacement techniques. When caching techniques were applied in the WSNs, the simulation results showed that the performance of packet loss rate, power consumption, and packet transmission delay improved.

Content popularity prediction is one of the key points for caching in which content should be storages; there is no literature to mention in WSN fields. The author [36] used the Markov model to predict sensor operation and proposed a smart cache model based on sensor power to improve cache hit ratios in the ICN-based IoT sensor networks. In [15], based on content popularity, the author proposed a caching decision policy, which allowed a single ICN router to cache content more or less according to the popularity characteristics of the content. In [37], the author proposed a regression method that supported vector regression and Gaussian radial basis functions to predict the popularity of YouTube and Facebook online videos. Mao [38] proposed a multitasking learning (MTL) module and a relational network (RN). The module's general prediction model used to predict TV drama views. In [39], a bi-directional long-term short-term memory neural network (BiLSTM) was proposed to predict the prevalence of online content. Studied in video and news texts, data sets have shown that deep network performance is greatly improved. In [40], the authors proposed that content popularity predictions could translate as classification problems and then made the end-to-end multimodal prediction based on the deep neural networks. In the experiment, text information and visual information were used to verify the validity of the models. In [41], to reduce congestion of backhaul network traffic, the authors proposed to predict user request based on file popularity and user and file patterns, during off-peak request the system proactive cached files. Through the acquisition of mobile subscriber service data of multiple base stations within a few hours interval, analysis was conducted on the big data platform to study the extent of evaluation of the content popularity, and proactive caching was performed [42]. The result showed that the level of satisfaction with user requests could be increased, while the backhaul network traffic could be reduced.

Deep learning has raised concerns in WSN and IOT. In SDN-IoT network [43], to forecast the congestion and traffic load, the author proposed a deep learning algorithm

to predict the future traffic load. In [44], in order to predict the evolution of traffic in the global network, the author proposed a hybrid convolution long-term memory neural network (CRS-ConvLSTM NN) model based on critical path.

Our research shows an effort to be combined with SDN, NFV, DL, and WSN. The proposed proactive cache strategy provides a reliable and efficient method to share the network resource of WSN cache devices. We propose a simple structure of SDN / NFV combined with SSAEs to solve the challenges of WSN, especially in traffic load and congestion management issues. This will improve the network efficiency and flexibility of WSN applications.

3. Methods

3.1. Distribute Deep Learning Networks Architecture on the WSN. The distributed deep learning network system architecture is shown in Figure 2. In WSN, the upper-layer transmission hardware consists of sink nodes, control nodes, and routers. All of the elements are SDN-enabled architecture, which realizes the separation of the control plane and the forwarding plane. With calculation and caching function, these network elements cooperate through the SDN controller. Therefore, the NFV/SDN technology is utilized to construct virtual distributed deep learning network architecture. In the WSN, those devices which have cache resources, such as sink node, router, sub-control node, and main control node, will share partial hardware by NFV to construct the deep learning network. The sink node/router resource will be the input part, and the main control node is the major calculation resource. Then the SDN controller is constructed on the main controller node. With SDN enabled in the WSN, SDN community is with sink node and router through the flow table. Therefore, the main control node may be dynamically aware of the status of the WSN.

On the distributed deep learning network, we construct the SSAEs. The hidden layers and output layer are on the main controller. In our model, SSAEs will predict the data package popularity in the future through historical user data. Therefore, a statistics server will be a virtual construct on the main control node. Those sink nodes will collect the local data package request information and send it to the statistics server. During the training stage, the statistics server will provide the historical data to the input layers. And the prediction stage, the statistics service summarizes the several timeslot request data for the input layers. Then the SSAEs make a prediction for data package in the future. The SDN controller generates a cache strategy based on the prediction. And the cache strategy is synchronized to those cache nodes to implement proactive caching in the period.

3.2. Deep Learning Network

3.2.1. Sparse Auto-Encoder. As an unsupervised feature learning method is widely studied in the field of deep learning, Sparse Auto-Encoder (SAE) has the capability to find a concise and efficient representation of complex data.

The SAE network has three different layers: input layer, hidden layer, and output layer. In order to obtain the optimal hidden layer parameters, that is, to minimize the SAE reconstruction error, the SAE network requires that the output layer be equal to the input layer. The data procedure of SAE is divided into encoding and decoding.

Encoding: The encoding process is to obtain feature y of the input layer value x. The original user requirement data is denoted as vector $x = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]^T$; parameter n means the total number of the input nodes.

$$y = f(x) = \delta(Wx + b) \tag{1}$$

where vector $y = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]^T$ indicated the feature expression of the hidden layer, and parameter m denotes the nodes of the hidden layers. Parameter b denotes the bias vectors, and W represents the weight matrix from input layer to hidden layer. $\delta(x)$ represents the activation function.

Decoding: The decoding process is to obtain the reconstructed vector z of the output layer from the hidden layer value y.

$$z = g(y) = \delta\left(W^T y + b'\right) \tag{2}$$

where $z = [z^{(1)}, z^{(2)}, \ldots, z^{(n)}]^T$, y denotes the feature expression of output layer, and b' denotes the bias vector.

In the SAE feature learning process, in order to minimize the loss caused by coding, the basic requirement is that the reconstructed output is close to the input. In the process of constructing the loss function, in order to learn more optimized sparse features, the sparse penalty term is added to the objective function of the encoder. The feature learned in this way is not simply repeated input. In fact, the role of the sparse penalty term in the hidden layer is to control the number of activated neurons. When the output of the neuron is 0, the neuron is the inactive state. While its output is 1, the neuron is active. Another goal in the feature learning process is to reconstruct the input with fewer active nerves. The reconstruction loss function of SAE is as follows:

$$\begin{aligned} L = {} & \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\left(x^{(i)} - z^{(i)}\right)^2\right] + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}^2 \\ & + \beta\sum_{j=1}^{m}\left[\rho\log\frac{\rho}{\rho_j} + (1-\rho)\log\frac{1-\rho}{1-\rho_j}\right] \end{aligned} \tag{3}$$

The loss function consists of three parts; the first parts denotes mean square error between the reconstructed output and the input. And the second part is $L2$ regularization which used to control the over-fitting issue, and w_{ij} denotes the weight. The last part is the sparsity regularization, where ρ denotes the desired target value, and $\rho_j = (1/m)\sum_{i=1}^{m}[y_j(x(i))]$, which mean the average activation output. In the loss function, parameter λ is to prevent overfitting, and β is to control the sparsity penalty.

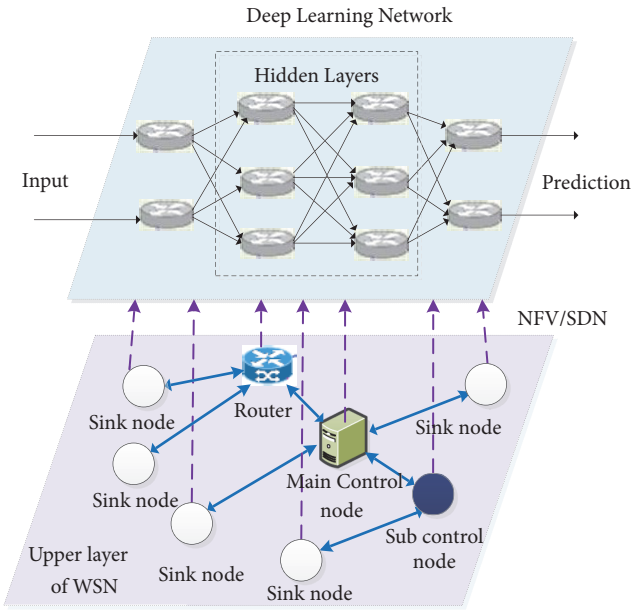


Figure 2: Deep learning network structure in WSNs.

With the greedy layer-wise training and backpropagation algorithm being used to obtain the parameters W and b, the detail iteration process is as follows:

$$w_{ij} = w_{ij} - \varepsilon \frac{\partial}{\partial w_{ij}} L \tag{4}$$

$$b_i = b_i - \varepsilon \frac{\partial}{\partial b_i} L \tag{5}$$

The hyperparameter learning rate ε is used to control the decreasing function of time. After training, SAE would learn the effective sparse feature representations.

3.2.2. Stacked Sparse Auto-Encoder Plus Softmax Classifier. In the WSN system, we will use the SSAE (Stacked Sparse Auto-Encoder) deep learning network due to the limitation of resources. The SSAEs are a deep neural network which consists of multiple layers of basic SAE. The outputs of each SSAEs layer are connected to the inputs of the successive layer. As shown in Figure 3, the SSAEs are composed of two layers SAE, where the first SAE is treated as an encoder, and the second SAE is a decoder. The output y_1 of the first SAE will be the input value of the input layer of the second SAE. When all the SSAE training finished, the model got the features parameter. And all the parameters of SSAEs are utilized to initialize the Softmax classifier to classify the objects.

3.3. Proactive Cache Strategy

3.3.1. Network Parameters of SSAEs. When we construct the SSAEs, some system parameter should be decided in advance, such as the dimension of the input layer, the number of the layers of the hidden layer, the number of neurons in each hidden layer, and the activation function of each layer.

In the SSAEs model, we could use some data groups to make a prediction. It is assumed that there has t width window slide along the time axis. The basic unit widow is one timeslot. In unit windows, all the sink nodes will report corresponding measure data, and those data contain spatial distribute of the WSN sink nodes. If few sink nodes' data missing, we consider the corresponding data is zero. Therefore, if there is more than one timeslot, the measure data contain the spatial-temporal information. If the timeslot window is t, there are $p \times t$ measure data which will provide for the input layers of SSAEs. And the $p \times t$ measure will concatenate as a vector.

Especially, when the window $t > 1$, the prediction of SSAEs will contain the spatial-temporal correlation of data package popularity. The measured data could present as $X^{i-1}, X^{i-2}, \ldots, X^{i-t}$, where i denotes the timeslot and t means the past time. Therefore, we could use the past measure to predict future data package popularity. The input data of the SSAE contain time dependence of content popularity. And the $p \times t$ measure data represent the traffic information of sink nodes in the WNSs.

In this article, it assumes that the content popularity has q levels; therefore the output of the Softmax classifier is also q levels. The output is present as the one-hot vector, and the corresponding level is the level of data package popularity.

Based on the previous description, we do not discuss the changes in the number of sensors in the WSN network. In the SSAEs network model construction, information collection is implemented at the sink nodes rather than directly on the sensor because the sensor nodes may change. In the first case, the sensor node is reduced, which is manifested by the fact that the sensor node loses connection with the WSN sink node for various reasons. This situation is similar to the fact

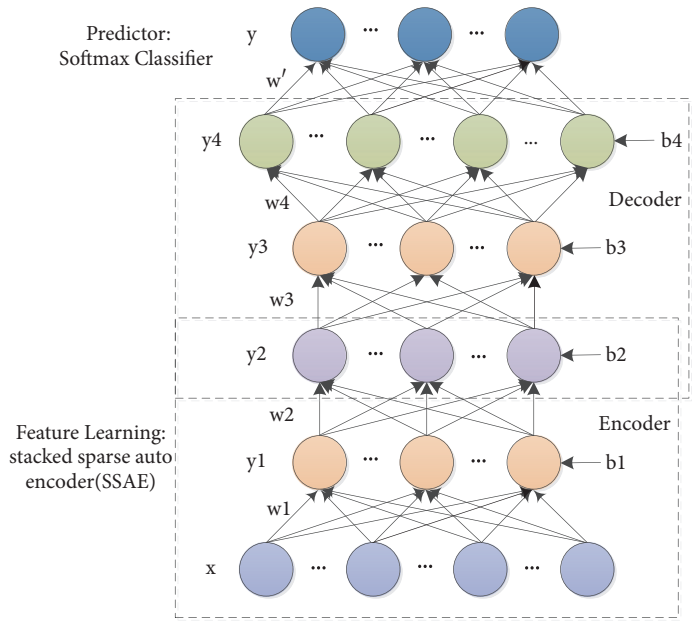


Figure 3: SSAEs structure.

that the data obtained by the sink node from the sensor node is zero, which does not affect the structure of the SSAE network and does not affect the prediction results. In the second case, new sensor nodes are added to the WSN network due to the need to acquire new data. In this case, we can consider a certain amount of node redundancy when initially building the network model. The measured data of these redundant nodes may take the average of the measured values of the WSNs sensors in the corresponding time slots as their values.

3.3.2. WSNs Data Package Popularity Prediction. Briefly, the popularity prediction procedure of SSAEs consist of initialization, training, and running stages.

(i) Initialization Stage. In this stage, the greedy layer-wise training and backpropagation algorithm is used to train the SSAEs model. Therefore, the initialization stage should get the labeling data, including the user request data vector and the corresponding classified information. As mentioned earlier, the user data package of the WSN network is collected by the SDN controller. And the output value is a multi-dimension one-hot vector which denotes the data package popularity in the appointed timeslot in the future.

(ii) Training Stage. In this stage, the pre-training obtains the general parameters, and fine-tuning will optimize those parameters. The greedy layer-wise unsupervised learning algorithm is executed from down to top. Then with the gradient-based optimization technique the BP algorithm is executed from top to down to finetune the SSAEs parameters.

When training is finished, the optimized parameters of SSAEs are fixed.

(iii) Running Stage. In this stage, the SDN controller will dynamically input the user request data information; then the SSAEs will make a prediction. And the data package popularity prediction will send to the SDN controller to generate cache strategy for the WSNs cache nodes.

3.3.3. Proactive Cache Strategy in WSNs. Based on the data package popularity prediction and the WSNs status, the SDN controller will generate the cache strategy.

In the SDN-enabled WSN, SDN controller reserves the topology information of all the WSN nodes and the routing information and is dynamically aware of the cache status of the cache node. The cache status information contains how many cache spaces are available and what is the priority level of the cache of each node. The basic principle of cache strategy is to reserve the high-level content and replace the low-level one. And those contents will be popular in the future time period and not be replaced. The SDN controller will periodically update the strategy when the SSAEs dynamically adjust the data package popularity prediction.

SDN controller generates the cache policies, and cache nodes are only responsible for the execution of the caching strategy. Due to the SDN controller taking the whole topology of WSN network and the cache status of all cache nodes into consideration, the cache strategy will be efficient to avoid the waste of caching resources and reduce the communication overhead between those cached nodes.

In the SDN-enabled WSN, SDN controller reserves the topology information of all the WSN node and the routing information and is dynamically aware of the cache status of the cache node. The cache status information contains how many cache spaces are available and what is priority level of the cache of each node. The basic principle of cache strategy is to reserve the high-level SDN controller

and is not only responsible for the routine work but also responsible for the deployment and maintenance PCDS2AW. When the cache strategy is updated, the SDN controller will update the routing forwarding table for content in the WSNs node and synchronize it to WSN node. When WSN node receives the routing forwarding table, that routing information will actively be inserted into the normal switch routing table before the data packet arrives. The data package will be cached to the assigned space when the high-level data packet arrives. Meanwhile, the low-level data packet will be replaced if the cache space is not enough.

WSN cache node addressing can work normally even if the routing and forwarding tables are missing or incomplete. The input information of the neuron in the PCDS2AW network is incomplete or some nodes are input timeout or the data noise is large. The PCDS2AW can still work properly and can predict the data package popularity of the input user request information. Therefore, PCDS2AW could show good performance of robustness.

Algorithm 1 (the implementation of PCDS2AW for proactive caching for WSN).

Initial: parameters of PCDS2AW

L = 3: Number of hidden layers

Ni = 1200: Number of input dimension

Nh = [300,200,100]: Number of hidden layer dimension

Ts=6: Number of timeslots

Sp = 0.1: Number of sparse parameters

α *= [1.2, 1.5]*: the Zipf law parameter

X_i: generate the data package request data based on $P(X_i = i) = i^{-\alpha} / \sum_{j=1}^{M} j^{-\alpha}$.

W_{li}: generate the weight parameter for each layer of SSAE.

b_i: generate the bias parameter for each layer of SSAE.

Pn = 40000: the number of pre-training epochs;

Fn = 10000; the number of fine-training epochs;

Training:

P = [Pn, Fn]

Repeat:

For Do *L*=1:3

(1) Input X_i,

(2) Calculate y_i based on Formula (1); $X_i = y_i$;

(3) Decode z_i based on Formula (2);

(4) Calculate cost function J based on Formula (3)

(5) Update $W_{li}$and b_i based on Formula (4) and (5).

Pi = Pi -1;

Until (*Pi=0*)

Proactive Cache:

Input X^t: the used request data information form slot *t-i* to *t*.

Calculate the data package popularity *Cp* based on Formula (1) ~ (3).

Sort $C_m = \min[C_{ij}]$, for all cache node N_k;

Replace the $C_m$with *Cp*.

4. Results

4.1. Experimental Environments. In our simulation, we use the TensorFlow framework to construct the SSAEs deep learning network. And the SDN is built on the OpenFlow protocol which separates the control plane of the WSN nodes from the data plane. Our simulation platform runs on Ubuntu 14.04 with 4 GPU cards, NVIDIA GeForce TitanX 12G GDDR5, and 64G RAM memories.

To simulate packet behavior in a WSN network, we assume that the set of packet content throughout the network is $F = f_1, f_2, \ldots, f_R$. These packets are generated by the content server in the WSN network and they are represented by the set $S = s_1, s_2, \ldots, s_p$. For ease of research, we assume that each content packet is randomly generated by only one content server, and each content server is only connected to one network node. At the same time, assuming that all content servers have the same size content unit, the cache space of each cache node of the WSN is the same size, and the cache slot in the cache memory can only accommodate one content unit. In addition, it is assumed that the timing of user packet content requests in the WSN system is subject to the Poisson distribution process. Assuming the user sends a packet request from a fixed virtual node, the overall user packet request conforms to Zipf's law. The frequency at which the user requests the content popularity i $(1 \le i \le M)$ of the data packet is as follows:

$$P(x = i) = \frac{(i^{-\alpha})}{C}, \qquad C = \sum_{j=1}^{M} j^{-\alpha} \tag{6}$$

where M is the total category of the data package content.

In the experiment, the user content request parameters in the WSN network collected by the SDN controller are first constructed as a one-dimensional sequence and then subjected to $[0, 1]$ normalization processing, and, finally, these measurement parameters are input to the input layer of the SSAEs network.

In the active cache emulation, when the user requests the content of the packet, the content matching is first performed in the cache in the corresponding node of the WSN. If the content is found, it indicates a cache hit; otherwise, the cache is not hit. When the user requests that the data packet is missed in the cache, the requested data packet content is traversed throughout the content distribution path of the content server. When the packet requested by the user is grouped, the SSAEs predict the content popularity level based on the measured values in the set slot segment. At the same time, the content popularity prediction result is sent to the SDN controller to generate a new cache policy. The SDN controller synchronizes the cache policy with the WSN cache node data plane through the flow table.

4.2. Evaluation Metrics. In this article, the research goal is full use of the cache resources in the WSN network by reducing the WSN cache node and increasing the storage

Table 1: Architecture structure for SSAEs. Table 1 is reproduced from F. Lei et al. (2018) (under the Creative Commons Attribution License/public domain).

Timeslot	Dimension of input layer	Hidden Layers	Hidden Layers units	MAPE (%)	MAE	RMSE
2	400	3	[300 200 100]	25.73	13.92	24.79
4	800	3	[300 200 100]	23.68	12.89	21.91
6	1200	3	[300 200 100]	22.11	13.95	20.24
8	1600	3	[300 200 100]	24.64	15.94	23.56
10	2000	4	[300 300 200 100]	26.32	16.28	25.73

content difference rate. Therefore, the evaluation criteria of the proactive cache are the cache hit rate, cache route hop count reduction rate. In addition, we also consider the impact of the data packet content popularity Zipf parameters on these evaluation metrics.

The results of the SSAEs prediction model will directly affect the generation of the caching strategy. Therefore, it is necessary to evaluate the SSAEs model performance indicators. We use three performance metrics: root mean square error (RMSE), mean absolute error (MAE), and mean absolute error (MAPE). The specific calculation formula for these indicators is as follows:

$$REMS = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(o_i - p_i)^2} \tag{7}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|o_i - p_i| \tag{8}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{n}\frac{|o_i - p_i|}{o_i} \tag{9}$$

where o_i is the observed number of data package being requested, p_i is the predicted number of data package content being requested, and N denotes the total number of evaluation samples. The RMSE measures the extremum effect and the error range of the predicted values, and the MAE measures the specificity of the average predicted value. Both of them evaluate the absolute error. MAPE reflects the relative error. When the MAPE is minimized, we consider the model as the optimal structure.

The proactive caching performance metrics is as follows.

CHR (Cache Hit Rate) is a traditional measure of caching performance. The CHR is defined as the ratio of the number of hits requested by the user in the intermediate node to the total number of packets requested by the user.

$$CHR = \frac{CacheHit}{TotalRequest} \tag{10}$$

where *TotalRequest* is the total data package content requests in all WSN nodes and *CacheHit* is total number of data package content request hit the cache nodes. It means the higher the *CHR*, the higher the cache efficiency.

HRR (Hop Reduction Ratio) is the ratio of the number of hops when requesting a data packet hit the cache node to the number of hops from the request data packet to the data source node.

$$HRR(t) = \frac{\sum_{r=1}^{R} h_r(t)}{\sum_{r=1}^{R} H_r(t)} \tag{11}$$

where $H_r(t)$ and $h_r(t)$ mean the hops that user get the request data package from source nodes and cache node in timeslot $[t, t+k]$, respectively. And if there are not cache resources in WSNs, $h_r(t) = 0$, then $HRR = 0$.

4.3. SSAEs Architecture Structure. Before performing content prediction, it is necessary to determine the appropriate SSAE architecture structure parameters, including the input layer dimension, the number of hidden layers, and the number of neurons in each hidden layer. We assume that cache nodes (including sink nodes, router, and control node) are 100 in the WSN network, and sink node is connected to 200 sensors, so 200 measurement data can be collected in each time slot. Therefore, the dimension of the appropriate input data can be determined by studying the number of slots. In the WSNs network, the SDN controller collects the WSN network node to input the user request data to the SSAE network every k time slot. As reported in [6, 10], we set the SSAEs deep learning with 3 hidden layers. We compare the time slot k from the set $\{2, 4, 6, 8, 10\}$, which means that the input dimensions range from 400 to 2000. When the MAPE is minimum, we obtain the optimal SSAEs structure for our prediction system model.

The SSAE network architecture test results are shown in Table 1. As the time slot increases from 2 to 10, the dimension of the input parameters is gradually increased. When the time slot is increased from 2 to 4, that is, the dimension of the input parameter is increased from 400 to 800, the MAPE reduction is significant. When the time slot is 6, the input dimension is 1200, and the MAPE reaches the minimum value, which is about 22.11%. Subsequently, as the time slot increases, that is, the input dimension increases, the MAPE increases. In subsequent experiments, the input dimension of the SSAEs network was set to 1200 and the three-layer hidden layer unit was [300 200 100]. In this process, the number of time slots represents the time-dependent characteristics of the number of inputs. If the number of time slots is too small, the correlation of the data cannot be reflected. If the time slot is too large, additional potential irrelevant inputs will be introduced, making it more difficult for the network architecture to learn a good representation.

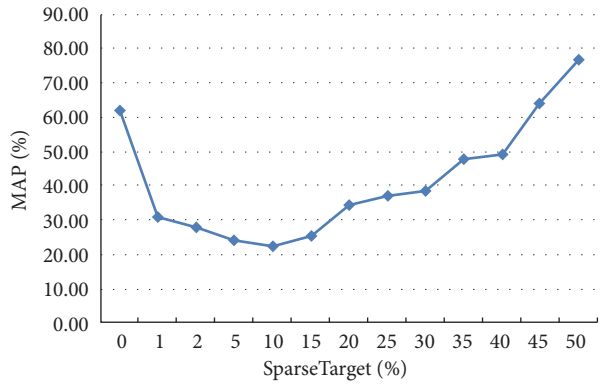


Figure 4: Sparse parameter of SSAEs.

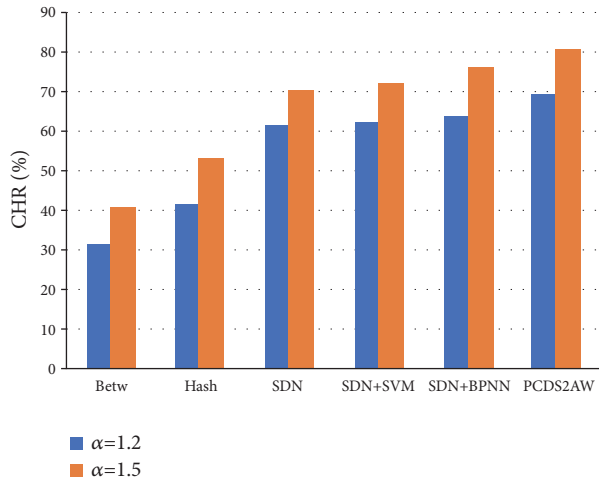


Figure 5: The impact of content popularity on caching.

The effect of sparse parameter of the SSAEs is compared, as shown in Figure 4. When the spare target is 0, which means no sparse target for the SSAEs, the MAP is about 62.18%. With the sparse target increase from 0 to 10%, the MAP is the minimum, 22.11%. With the sparse target increase from 10% to 50%, the MAP also increases to 78.86%. In the SSAEs, the sparse keep as 10%.

4.4. Experimental Results. In the simulation, we also assume that there are 100 cache nodes in the WSN network, 200 measurement data can be collected in each time slot. The other design parameters are as follows: the quantity of users' data package request is set to 120000, the average data package content size is the 1k bit, and the unit size of node cache is the 1k bit. All the following results are the average of 10 rounds simulation.

We study changes in network performance due to PCDS2AW prediction and generalization capabilities. Firstly, we compare PCDS2AW with traditional classic caching strategies that do not support SDN, such as Betw + LRU, Hash + LRU, and Opportunistic. At the same time, we also compare PCDS2AW with caching strategies that support SDN, such as SDN + BPNN (Back Propagation Network), SDN + SVM (Support Vector Machine). BPNN is a classical neural network that learns features through hidden layers. SVM is a widely used classic prediction model. In these comparative experiments, we used the same training set and test set as PCDS2AW to train and test these models.

Figure 5 shows the CHR (cache hit ratio) comparison of Betw, Hash, SDN, SDN+SVM, SDN+BPNN and PCDS2AW schemes when alpha =1.2 and alpha =1.5 are used. As the alpha value increases, the data distribution requested by the user is more concentrated, and the probability of the data packet hitting the cache is also improved, which shows that the CHR in all schemes is correspondingly improved. For Hash-LRU, all the nodes in the domain collaborate and realize the cooperative caching of the response content. Although the content cannot be optimized, the hit rate of the cache content is enhanced, and the cache hit rate increases by 27.5%. As for the Betw-LRU, the cache hit rate increases by 29.1% with the user's content request more centralized. In the SDN, because SDN's network perception function further makes full use of the storage space of all caching nodes, increasing the probability and utilization of caching content, CHR reaches 61.4% and 70.3%, respectively. And with the SVM to predict the popularity, the CHR of SDN+SVM is higher 1.1% and 2.1%

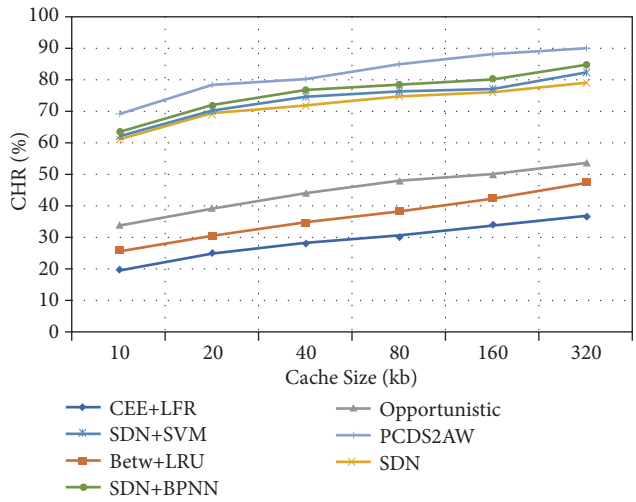


Figure 6: Effect of cache size.

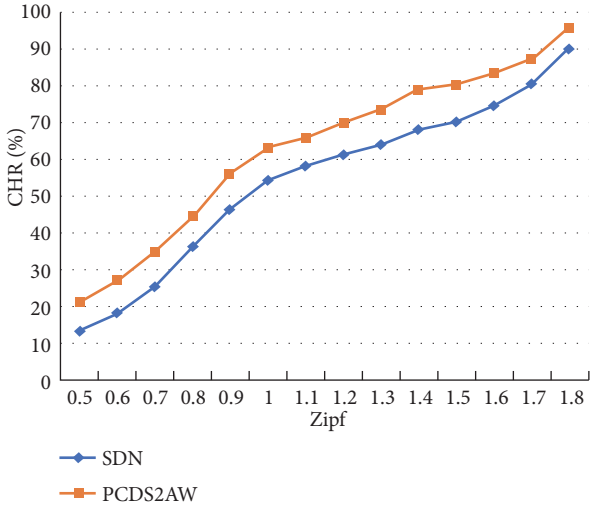


Figure 7: Effect of Zipf parameter α.

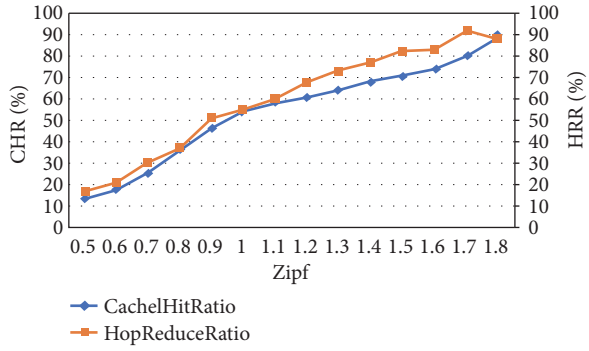


Figure 8: Effect of Zipf with CHR and HRR.

than SDN. The SDN+BPNN cache hit rate reach 64.2% and 76.4%. The PCDS2AW achieves the best than others; CHR reaches 69.6% and 80.3%, respectively.

Figure 6 shows the effect of cache size on the cache hit ratio. As the size of the cache space of the cache node increases, the cache hit ratio in all cache schemes also increases. This is because as the cache storage space increases, the cache node can cache more data content packets, so there is a higher probability of buffering the data packets requested by the user. As the cache space increased from 10kb to 320kb, the cache hit rate for CEE increased from 19.9% to 32.5%; the cache hit rate of the Betw scheme increased from 25.9% to 47.5%; Opportunistic increased from 34.1% to 53.4%; the cache hit rate of SDN increased from 61.4% to 79.1%; CHR of SDN+SVM increased from 62.3% to 82.4%, the SDN+BPNN increased from 63.7% to 84.7%, and the PCDS2AW increased from 69.3% to 90.1%.

Figure 7 shows the influence of the content popularity parameter α on the cache hit rate in the Zipf distribution. With the increase of content popularity, the concentration of requests for user content is increasing. In the case of cache space, the probability of the content in the cache is increased and the cache hit rate is increased. In Figure 6, cache popularity alpha increased by 1.8 from 0.5; the cache hit rate increased from 13.30% to 89.98% for SDN cache scheme and from 21.5% to 95.7% for PCDS2AW.

Figure 8 shows the CHR and HRR relation of PCDS2AW with the Zipf distribution. With Zipf increase, the CHR and HRR also increase. When cache popularity alpha equals 1.7, the HRR reaches the maximum about 91.88%.

Computational complexity: The PCDS2AW algorithm is mainly divided into two stages: offline feature model training and online content popularity prediction. In the offline phase, SSAEs feature training is mainly used to obtain model parameters. The online stage directly predicts the input value input into the model for content popularity prediction. The prediction model SSAE in this paper is three layers [300, 200, 100], and the parameter dimension of the input layer is 1200. The sparse parameter in our model is 0.10. Because multiplication is the most time-consuming, we only estimate the number of the multiplications of the online prediction in our model.

The number of multiplications in the prediction process is (1200∗300+300∗200+200∗100)∗0.10=44000. In our simulation, the predicted time was 0.0435 s. Therefore, the model can meet the needs of online prediction.

5. Conclusions

This paper presents a simple frame structure of SDN/NFV coupled with SSAEs to address the challenges of WSNs particularly in the issues of traffic loads and congestion management. We construct the distributed deep learning network, SSAEs by SDN/NFV technical. After detail analyzes the architecture parameters, the unsupervised training method is used to obtain the prediction model. The experiments show that the SSAEs can greatly improve the prediction accuracy, and then the performance of WSN can be improved based on proactive caching.

In the future, we plan to combine the SSAEs with Incremental Extreme Learning Machine (IELM) to online training and update the SSAE network parameter, while the user request dynamic changes to get data package prediction and continues to optimize our model to get better performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This research was funded by the National Natural Science Foundation of China [U170120078, 61571141, 61702120, and 61672008], the Scientific and Technological Projects of Guangdong Province [2017A050501039], the Guangdong Provincial Key Laboratory Project [2018B030322016], the Guangdong Provincial Application-Oriented Technical Research and Development Special Fund Project [2015B010131017, 2015B090923001, 2016B010127006, and 2017B010125003], the Guangdong Province General Colleges and Universities Featured Innovation [2015GXJK080], and the Qingyuan Science and Technology Plan Project [170809111721249 and 170802171710591].

References

[1] Cisco VNI, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021*, 2017.

[2] B. Raghavan, T. Koponen, A. Ghodsi, M. Casado, S. Ratnasamy, and S. Shenker, "Software-defined internet architecture:decoupling architecture from infrastructure," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets '12)*, 2012.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[4] M. Ojo, D. Adami, and S. Giordano, "A SDN-IoT architecture with NFV implementation," in *Proceedings of the GLOBECOM Workshops*, 2016.

[5] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016.

[6] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for ICN: a deep learning approach with SDN," *IEEE Access*, vol. 99, p. 1, 2017.

[7] N. Abani, T. Braun, and M. Gerla, "Proactive caching with mobility prediction under uncertainty in information-centric networks," in *Proceedings of the 4th ACM Conference on Information-Centric Networking, ICN '17*, ACM, 2017.

[8] Y. L. Cun, B. Boser, J. S. Denker et al., "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems*, vol. 2, no. 2, pp. 396–404, 1990.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2012.

[10] F. Lei, Q. Dai, J. Cai et al., "A proactive caching strategy based on deep Learning in EPC of 5G," in *Proceedings of the International Conference on Brain Inspired Cognitive Systems*, Springer, 2018.

[11] G. Wu, J. Han, and Y. Guo, "Unsupervised deep video hashing via balanced code for large-scale video retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2019.

[12] G. Wu, J. Han, and Z. Lin, "Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning," *IEEE Transactions on Industrial Electronics*, 2018.

[13] G. Ding, Y. Guo, K. Chen, C. Chu, J. Han, and Q. Dai, "DECODE: deep confidence network for robust image classification," *IEEE Transactions on Image Processing*, 2019.

[14] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[15] K. Suksomboon, S. Tarnoi, J. Yusheng et al., "PopCache: Cache more or less based on content popularity for information-centric networking," in *Proceedings of the 38th Annual IEEE Conference on Local Computer Networks (LCN '13)*, pp. 236–243, 2013.

[16] X. Hu and J. Gong, "Opportunistic on-path caching for named data networking," *IEICE Transactions on Communications*, vol. E97B, no. 11, pp. 2360–2367, 2014.

[17] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor openflow: enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[18] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Proceedings of the International Conference on Computer Science and Network Technology*, 2012.

[19] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[20] W. Dong, Y. Liu, C. Chen, L. Gu, and X. Wu, "Elon: enabling efficient and long-term reprogramming for wireless sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 4, pp. 1–27, 2014.

[21] L. Jiang, L. Yan, Y. Xia, Q. Guo, M. Fu, and L. Li, "Distributed fusion in wireless sensor networks based on a novel event-triggered strategy," *Journal of The Franklin Institute*, 2018.

[22] B. T. De Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Proceedings of the Communications*, 2014.

[23] T. Miyazaki, S. Yamaguchi, K. Kobayashi et al., "A software defined wireless sensor network," in *Proceedings of the 2014 International Conference on Computing, Networking and Communications (ICNC '14)*, 2014.

[24] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Proceedings of the Communications*, 2014.

[25] P. Jayashree and F. Infant Princy, "Leveraging SDN to conserve energy in WSN-An analysis," in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking, ICSCN '15*, 2015.

[26] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, "UbiFlow: Mobility management in urban-scale software defined IoT," in *Proceedings of the IEEE Conference on Computer Communications*, 2015.

[27] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM '15)*, pp. 513–521, 2015.

[28] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: unbridling SDNs," in *Proceedings of the 2012 European Workshop on Software Defined Networking*, pp. 1–6, 2012.

[29] B. R. Al-Kaseem and H. S. Al-Raweshidyhamed, "SD-NFV as an energy efficient approach for M2M networks using cloud-based 6LoWPAN Testbed," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1787–1797, 2017.

[30] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," in *Proceedings of the IEEE International Conference on Communications*, 2014.

[31] A. Mahmood, C. Casetti, C. F. Chiasserini et al., "Mobility-aware edge caching for connected cars," in *Proceedings of the Conference on Wireless On-Demand Network Systems and Services*, 2015.

[32] C. Jayapal, S. Jayavel, and V. P. Sumathi, "Enhanced service discovery protocol for MANET by effective cache management," *Wireless Personal Communications*, pp. 1–17, 2018.

[33] M. I. Alipio and N. M. C. Tiglao, "Dynamic source rate control for cache-based transport protocol in wireless sensor networks," *Computer Communications*, vol. 113, pp. 14–24, 2017.

[34] M. Alipio, N. M. Tiglao, A. Grilo, F. Bokhari, U. Chaudhry, and S. Qureshi, "Cache-based transport protocols in wireless sensor networks: a survey and future directions," *Journal of Network and Computer Applications*, vol. 88, pp. 29–49, 2017.

[35] C. Panagiotou, C. Antonopoulos, and S. Koubias, "A comprehensive evaluation of cache utilization characteristics in large scale WSN considering network driven cache replacement techniques," in *Proceedings of the MATEC Web of Conferences*, EDP Sciences, 2018.

[36] R. Sukjaimuk, Q. N. Nguyen, and T. Sato, "A smart congestion control mechanism for the green IoT sensor-enabled information-centric networking," *Sensors*, 2018.

[37] T. Trzcinski and P. Rokita, "Predicting popularity of online videos using support vector regression," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2561–2570, 2017.

[38] Y. Mao, Y. Shen, G. Qin, and L. Cai, "Predicting the Popularity of Online Videos via Deep Neural Networks," https://arxiv.org/abs/1711.10718, CoRR, 2017.

[39] W. Stokowiec, T. Trzciński, K. Wołk, K. Marasek, and P. Rokita, "Shallow reading with deep learning: predicting popularity of online content using only its title," in *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, 2017.

[40] H. Cai, Y. Zhang, Y. Wang et al., "Predicting relative popularity via an end-to-end multi-modality model," in *Proceedings of the International Forum on Digital TV and Wireless Multimedia Communications*, 2017.

[41] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: the role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[42] E. Bastug, M. Bennis, E. Zeydan et al., "Big data meets telcos: A proactive caching perspective," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 549–557, 2015.

[43] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An intelligent traffic load prediction based adaptive channel assignment algorithm in SDN-IoT: a deep learning approach," *IEEE Internet of Things Journal*, p. 1, 2018.

[44] G. Yang, Y. Wang, H. Yu, Y. Ren, and J. Xie, "Short-term traffic state prediction based on the spatiotemporal features of critical road sections," *Sensors*, vol. 18, no. 7, p. 2287, 2018.

Hindawi
Complexity
Volume 2019, Article ID 8641074, 12 pages
https://doi.org/10.1155/2019/8641074

Research Article

A New Type of Eye Movement Model Based on Recurrent Neural Networks for Simulating the Gaze Behavior of Human Reading

Xiaoming Wang,[1] **Xinbo Zhao**,[1] **and Jinchang Ren**[2]

[1]*National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology, School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China*
[2]*Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow G1 1XW, UK*

Correspondence should be addressed to Xinbo Zhao; xbozhao@nwpu.edu.cn

Received 6 November 2018; Revised 11 February 2019; Accepted 27 February 2019; Published 24 March 2019

Guest Editor: Jungong Han

Traditional eye movement models are based on psychological assumptions and empirical data that are not able to simulate eye movement on previously unseen text data. To address this problem, a new type of eye movement model is presented and tested in this paper. In contrast to conventional psychology-based eye movement models, ours is based on a recurrent neural network (RNN) to generate a gaze point prediction sequence, by using the combination of convolutional neural networks (CNN), bidirectional long short-term memory networks (LSTM), and conditional random fields (CRF). The model uses the eye movement data of a reader reading some texts as training data to predict the eye movements of the same reader reading a previously unseen text. A theoretical analysis of the model is presented to show its excellent convergence performance. Experimental results are then presented to demonstrate that the proposed model can achieve similar prediction accuracy while requiring fewer features than current machine learning models.

1. Introduction

Using computers to simulate humans or to reproduce certain intelligent behaviors related to human vision is a typical computer vision task [1], such as simulating eye movements in reading. However, reading is complex cognitive behavior and the underlying cognitive process occurs only in the brain [2]. Modeling such behavior requires obtaining some explicit indicators via such methods as eye tracking.

When reading a text, the eyes of a skilled reader do not move continuously over the lines of text. Instead, reading proceeds by alternating between fixations and rapid eye movements called *saccades* [3]. This behavior is determined by the physiological structure of the human retina. Most of the optic nerve cells are concentrated in the fovea and only when the visual image falls in this area can it be "seen" clearly. Unfortunately, the fovea only provides about a 5-degree field of view [4]. Therefore, the reader needs to change the fixation point through successive saccades so that the next content falls on the fovea region of the retina. By analyzing eye movements during reading, we can quantify the reader's actions and model for reading. Eye tracking helps researchers to determine where and how many times subjects focus on a certain word, along with their eye movement sequences from one word to another [5]. Figure 1 shows an example eye movement trajectory from an adult reader.

Models of eye movement control have been studied in cognitive psychology [6–9]. Researchers integrated a large amount of experimental data and proposed a variety of eye movement models such as easy-rider (E-Z) reader [10] and saccade generation with inhibition by foveal targets (SWIFT) [11]. Although these eye movement models typically have parameters that are fit to empirical data, their predictions are rarely tested on unseen data [12]. Moreover, their predictions are usually averaged over a group of readers, while eye movement patterns vary significantly between individuals [13]. Predicting the actual eye movements that an individual will make while reading a new text is arguably a challenging problem.

Some recent work has studied eye movement patterns from a machine learning perspective [14–17]. These studies were inspired by recent work in natural language processing

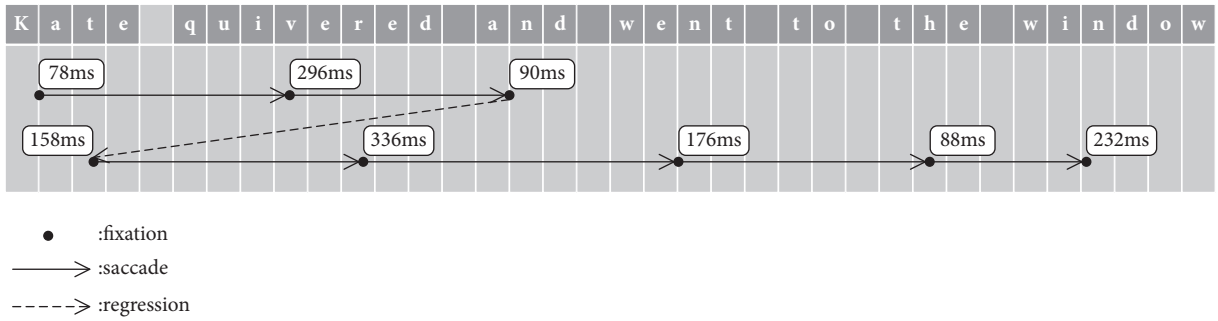


Figure 1: Eye track of an adult reader. Black dots indicate the position of the gaze point, the number next to each dot indicates the duration of each gaze point (in ms), and the arrow indicates the direction of the saccade.

(NLP) and are less tied to psychophysiological assumptions about the mechanisms that drive eye movements. The work presented in [14] was the first to apply machine learning methods to simulate human eye movements. The authors used a transformation-based model to predict word-based fixation of unseen text. Reference [17] applied a conditional random field (CRF) model to predict which words in a text are fixated by a reader. However, traditional supervised learning requires more features and preprocessing of data, which may lead to high latency in human-computer interaction applications.

Aided by their parallel distributed processing paradigm, neural networks have been widely used in pattern recognition and language processing because of their parallel distribution [18]. In 2010, Jiang [19] studied how to apply neural networks to multiple fields and provided a review of the key literature on the development of neural networks in computer-aided diagnosis. In 2011, Ren [20] proposed an improved neural network classifier that introduced balanced learning and optimization decisions, enabling efficient learning from unbalanced samples. In 2012, Ren [21] proposed a new balanced learning strategy with optimal decision making that enables effective learning from unbalanced samples and is further used to evaluate the performance of neural networks and support vector machines (SVMs).

In recent years, deep neural networks (DNN) have become a popular topic in the field of machine learning. DNN has successfully improved the recognition rate and some excellent optimization algorithms and frameworks have been proposed and applied. Guo (2018) proposed a novel robust and general vector quantization (VQ) framework to enhance both robustness and generalization of VQ approaches [22]. Liu (2018) presented an efficient bidirectional Gated Recurrent Unit (GRU) network to explore the feasibility and the potential of mid-air dynamic gesture based user identification [23]. Liu (2019) presented an end-to-end multiscale deep encoder (convolution) network, which took both the reconstruction of image pixels' intensities and the recovery of multiscale edge details into consideration under the same framework [24]. Wu (2018) proposed an unsupervised deep hashing framework and adopted an efficient alternating approach to optimize the objective function [25]. Wu (2019) proposed a Self-Supervised Deep Multimodal Hashing (SSDMH) method and demonstrated the superiority of SSDMH over state-of-the-art cross-media hashing approaches [26]. Luan (2018) developed a new type of deep convolutional neural networks (DCNNs) to reinforce the robustness of learned features against the orientation and scale changes [27]. Besides, some methods based on recurrent networks have been proposed, developed, and studied for natural language processing [28–30].

In this paper, we formalize the problem of simulating the gaze behavior of human reading as a word-based sequence labeling task (which is a classic NLP application). In the proposed method, the eye movement data of a reader reading some texts is used as training data and a bidirectional Long Short-Term Memory-Conditional Random Field (bi-LSTM-CRF) neural network architecture is used to predict the eye movement of the same reader reading a previously unseen text. The model is focused on achieving similar prediction accuracy while requiring fewer features than existing methods. However, it is worth emphasizing that in this study we focus only on models of where the eyes move during reading, and we will not be concerned with the temporal aspect of how long the eyes remain stationary at fixated words.

The remainder of this paper is organized into the following sections. Section 2 introduces the problem formulation. Section 3 proves the convergence of the model. Section 4 describes the layers of our neural network architecture. Section 5 discusses the training procedure and parameter estimation. Section 6 demonstrates the superiority of the proposed method with verification experiments. Section 7 concludes the paper with final remarks. Before ending the current section, it is worth pointing out the main contributions of the paper as follows.

(i) This paper proposes and tests a new type of eye movement model based on recurrent neural networks, which is quite different from previous research on eye movement model.

(ii) The convergence of the RNN-based model for predicting eye movement of human reading is proved in this paper.

(iii) An experiment of foveated rendering further demonstrates the novelty and effectiveness of recurrent

neural networks for solving the problem of simulating the gaze behavior of human reading.

2. Problem Formulation

Experimental findings in eye movement and reading research suggest that eye movements in reading are both goal-directed and discrete [6]. This means that the saccadic system selects visual targets on a nonrandom basis and that saccades are directed towards particular words rather than being sent a particular distance. Under this view, there are a number of candidate words during any fixation, with each having a certain probability of being selected as the target for the subsequent saccade. For our purposes we will assume that a probabilistic saccade model assigns a probability to fixation sequences resulting from saccade generation over the words in a text. Let us use the following simple representations of a text and fixation sequence.

Let $\boldsymbol{R}$ denote a set of readers, and text $\boldsymbol{T}$ represent a sequence of word tokens (text) $(\mathrm{w}_1, \ldots, \mathrm{w}_\mathrm{n})$. Let $\boldsymbol{F}$ denote a sequence of fixation token positions in $\boldsymbol{T}$, generated by a reader $\boldsymbol{r}$ ($\boldsymbol{r} \in \boldsymbol{R}$). The fixation token positions set $\boldsymbol{S}(\boldsymbol{F})$ is the set of token positions that removes repetitive elements from $\boldsymbol{F}$, where $\boldsymbol{S} = \{\mathrm{S}_1, \ldots, \mathrm{S}_\mathrm{m}\}(1 \le \mathrm{S}_\mathrm{i} \le \mathrm{n})$.

$$S(F) = \arg\max p(S \mid T, r) \tag{1}$$

$p(\mathrm{S} \mid \mathrm{T}, \mathrm{r})$ is a reader-specific distribution of eye movement patterns given a text $\boldsymbol{T}$. For example, the text "*Kate quivered and went to the window*" is represented by $\boldsymbol{T}$ = (Kate, quivered, and, went, to, the, window). A sequence of fixations in a reader's reading of the text which is *Kate-quivered-and-went-to-the-window* is represented by $\boldsymbol{F} = (1, 2, 3, 1, 2, 4, 6, 7)$; and the corresponding fixation token positions set is $\boldsymbol{S}(\boldsymbol{F}) = \{1, 2, 3, 4, 6, 7\}$.

The next object of study is the prediction of the fixation point sequence $\boldsymbol{F}$ based on a specific reading event $\boldsymbol{E}$ involving the reader $\boldsymbol{R}$ reading the text $\boldsymbol{T}$. The training data consist of words and a Boolean value indicating whether they are fixation words, in the form of $((\mathrm{w}_1, \mathrm{boolFixation}_1), \ldots, (\mathrm{w}_\mathrm{n}, \mathrm{boolFixation}_\mathrm{n}))$.

$\boldsymbol{M}$ is a recurrent neural networks model. Given some text as input, the purpose of this model is to generate a sequence of fixation points that approximate human reading behavior. We evaluate the performance of model $\boldsymbol{M}$ by comparing the predicted fixation sequence set $\boldsymbol{S}_{\boldsymbol{M}}$ with the actual fixation sequence set $\boldsymbol{S}_{\boldsymbol{O}}$ observed in a reading experiment involving $\boldsymbol{R}$ and $\boldsymbol{T}$. To do this, we train $\boldsymbol{M}$ on a novel set of texts $\boldsymbol{X} = \{\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{m}\}$ generated by a reader $\boldsymbol{r} \in \boldsymbol{R}$. The goal is to infer

$$s^* = \arg\max_{s \in S(F)} p(S \mid M, X, r) \tag{2}$$

3. Convergence Analysis

For neural networks with $\boldsymbol{m}$ hidden nodes and $\boldsymbol{n}$ training data samples, if the ReLU activation function is used, a previous study in [31] has shown that, as long as $\boldsymbol{m}$ is sufficiently enough, the randomly initialized gradient descent algorithm converges to the global optimal solution. In this section, by following the gradient descent provably optimized method, the convergence performance of our RNN-based model is presented below.

First, we consider a recurrent neural network of the following form:

$$z^{(t)} = Ux^{(t)} + Wh^{(t-1)} + b \tag{3}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{h}$ is a hidden layer, $\boldsymbol{z}$ is an inactive hidden layer, $\boldsymbol{U}$ is the matrix of $\boldsymbol{x}^{(\mathrm{t})}$ to $\boldsymbol{h}^{(\mathrm{t})}$, $\boldsymbol{W}$ is the matrix of $\boldsymbol{h}^{(\mathrm{t}-1)}$ to $\boldsymbol{h}^{(\mathrm{t})}$, and $\boldsymbol{b}$ is the bias of the hidden layer.

Use the tanh activation function to activate all nodes of the layer $\boldsymbol{z}$ to the layer $\boldsymbol{h}$:

$$h^{(t)} = \phi\left(z^{(t)}\right) = \tanh\left(z^{(t)}\right) \tag{4}$$

Map the layer $\boldsymbol{h}^{(\mathrm{t})}$ to the layer $\boldsymbol{o}^{(\mathrm{t})}$:

$$o^{(t)} = Vh^{(t)} + c \tag{5}$$

where $\boldsymbol{V}$ is the matrix of $\boldsymbol{h}^{(\mathrm{t})}$ to $\boldsymbol{o}^{(\mathrm{t})}$. Use $\boldsymbol{o}^{(\mathrm{t})}$ to derive the predicted value $\hat{y}$ and loss function of the model.

$$\hat{y}^{(t)} = \sigma\left(o^{(t)}\right) = crf\left(\mathrm{o}^{(t)}\right) \tag{6}$$

$$L = -\sum_{t=1}^{n}\sum_{k=1}^{C} y_k^{(t)} \ln\left(\hat{y}_k^{(t)}\right) \tag{7}$$

where $\boldsymbol{crf}$ is an activation function, $\boldsymbol{T}$ is the transform matrix of $\boldsymbol{y}_{\mathrm{t}-1}$ to $\boldsymbol{y}_\mathrm{t}$, $\boldsymbol{S}$ is the state matrix of $\boldsymbol{y}_\mathrm{t}$, Z is a scaling factor, and λ and μ are weights, which is defined as

$$crf(o(t)) = \frac{1}{Z\left(o^{(t)}\right)} \exp\left(\lambda T_{y_{t-1}, y_t} o^{(t)} + \mu S_{y_t} o^{(t)}\right) \tag{8}$$

There are 3 parameters that need to be optimized, namely, U, V, and W. Among them, the optimization process of the two parameters of W and U needs to trace the historical data, the parameter V is relatively simple and only needs the current data, and then we will solve the partial derivative of the parameter V first.

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{n} \frac{\partial L^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V^{(t)}} \tag{9}$$

The solution of the partial and partial derivatives of W and U is relatively complicated because of the need to involve historical data. Let us assume that there are only three moments, and then the partial derivative of L to W at the third moment is

$$\begin{aligned}\frac{\partial L^{(3)}}{\partial W} = {} & \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial W} + \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial W} \\ & + \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial W}\end{aligned} \tag{10}$$

It can be observed that, at some point, the partial derivative of L to W needs to trace back all the information before this

moment. We can write the general formula of L to the partial derivative of W at time t according to formula (10):

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial W} \tag{11}$$

$$\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} = \prod_{j=k+1}^{t} crf' \cdot W_s \tag{12}$$

To prove that the RNN-based model will converge, it is only necessary to prove that there is an upper limit on the number of incorrect training examples on the training data set. The following lemmas are presented before the proof.

Lemma 1. *If the training data set is linearly separable, then there is a shortest distance from all training data points to the distance separating the hyperplanes, denoted as γ. Prove that* $\|\widehat{y}^{(k)} - y\|_2^2 \le (1 - \eta\gamma/2)^k \|\widehat{y}^{(0)} - y\|_2^2$ *is established.*

Proof. According to Cauchy inequality $\widehat{w} \cdot \widehat{w}_{opt} \le \|\widehat{w}\| \cdot \|\widehat{w}_{opt}\|$.

$$\begin{aligned} &\because \widehat{w}_k = \widehat{w}_{k-1} + \widehat{x}_t y_t \\ &\therefore \widehat{w}_k \cdot \widehat{w}_{opt} = \left(\widehat{w}_{k-1} + \widehat{x}_t y_t\right) \widehat{w}_{opt} \ge \widehat{w}_{k-1} \widehat{w}_{opt} + \eta\gamma \\ &\qquad \ge \widehat{w}_{k-2} \widehat{w}_{opt} + 2\eta\gamma \ge \cdots \ge k\gamma \\ &\therefore \left\| \widehat{y}^{(k)} - y \right\|_2^2 \le \left(1 - \eta\gamma/2\right)^k \left\| \widehat{y}^{(0)} - y \right\|_2^2 \end{aligned} \tag{13}$$

The proof is thus completed. □

Lemma 2. *Let* $R = 4\sqrt{n}\|y - \widehat{y}^{(0)}\|_2 / \sqrt{m}\lambda_0$*, then the number of misclassifications k of the algorithm on the training data set satisfies the inequality* $\|y - \widehat{y}^{(k+1)}\|_2^2 \le (1 - kR/2)\|y - \widehat{y}^{(k)}\|_2^2$.

Proof. At the k-th iteration, we have $\|\widehat{y}^{(k)} - y\|_2^2 \le (1 - \eta\gamma/2)^k \|\widehat{y}^{(0)} - y\|_2^2$. If k'=0,. . .,k, then we have for every $\boldsymbol{r} \in$ [m]

$$\left\| w_r^{(k+1)} - w_r^{(0)} \right\|_2 \le \frac{4\sqrt{n} \left\| y - \widehat{y}^{(0)} \right\|_2}{\sqrt{m}\lambda_0} = R \tag{14}$$

Next, we calculate the difference in prediction between two consecutive iterations.

$$\begin{aligned} \widehat{y}^{(k+1)} - \widehat{y}^{(k)} = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r (&\sigma\left(w_r^{(k+1)^T} x_i \right) \\ &- \sigma\left(w_r^{(k)^T} x_i \right) \end{aligned} \tag{15}$$

As in the classical analysis of gradient descent, we also need to bound the quadratic term.

$$\begin{aligned} \left\| \widehat{y}^{(k+1)} - \widehat{y}^{(k)} \right\|_2^2 &\le \eta^2 \frac{1}{m} \left(\sum_{r=1}^{m} \left\| \frac{\partial L\left(W^{(k)}\right)}{\partial w_r^{(k)}} \right\|_2 \right)^2 \\ &\le \eta^2 n^2 \left\| y - \widehat{y}^{(k)} \right\|_2^2 \end{aligned} \tag{16}$$

With these estimates, we are ready to prove the theorem.

$$\begin{aligned} \left\| y - \widehat{y}^{(k+1)} \right\|_2^2 &= \left\| y - \widehat{y}^{(k)} - \left(\widehat{y}^{(k+1)} - \widehat{y}^{(k)} \right) \right\|_2^2 \\ &\le \left(1 - \eta\lambda_0 + 2\eta n^2 R + 2\eta n^{3/2} R + \eta^2 n^2\right) \left\| y - \widehat{y}^{(k)} \right\|_2^2 \\ &\le \left(1 - \frac{kR}{2}\right) \left\| y - \widehat{y}^{(k)} \right\|_2^2 \end{aligned} \tag{17}$$

□

This indicates that there is an upper limit on the number of misclassifications, and the algorithm will converge when the final number of misclassifications reaches the upper limit. So if the data is linearly separable, then the model does converge.

4. Network Architecture

In this section, we describe a CNN-LSTM-CRF-based network architecture for reading eye movement prediction, consisting of a word embedding layer, a CNN layer, a bidirectional LSTM layer, and a CRF layer from bottom to top.

4.1. Word Embedding and CNN Layer. It was shown in [28] that word embedding plays a crucial role in improving the performance of sequence labeling. We vectorize the text corpus by converting each text into a sequence of integers (each integer is an index of the mark in the dictionary), where the coefficients of each mark can be based on the number of words.

Previous studies, e.g., [32, 33], showed that convolutional neural networks (CNNs) are effective methods for extracting word form information from characters in a word and encoding it into a neural representation. Our CNN is similar to that used in [33], except that we use not only word embedding as CNN input, but also the linguistic features of the word.

4.2. RNN Layer. One of the key features of recurrent neural networks (RNNs) is that they can be used to connect past information to current tasks, such as inferring the meaning of a current word from previous words. However, a RNN cannot connect past information when the gap between the relevant information and the current position increases.

Long Short-Term Memory (LSTM) network is a special RNN that can learn long-term dependencies through specialized design. A LSTM is also a multilayered type of the neural network, in which the single neural network layer contains four interactive sublayers, as shown in Figure 2.

In many sequence labeling tasks, it is best to have access to past (left) and future (right) contexts, while LSTM's hidden state does not get information from the future. An excellent solution is the bidirectional LSTM (bi-LSTM) [34], whose validity has been proven in previous studies. Therefore, we can effectively use past features (through the forward state) and future features (through the backward state). We use

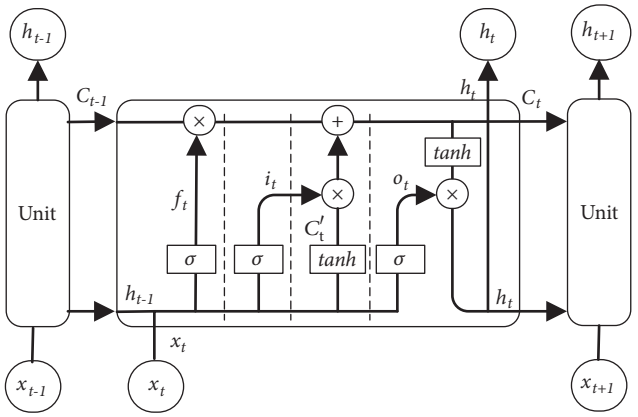


Figure 2: The repeating unit in LSTM networks.

backpropagation through time (BPTT) [35] to train bi-LSTM networks, which help to process multiple sentences at the same time.

4.3. CRF Layer. A conditional Random Fields (CRF) is a conditional probability distribution model whose nodes can be divided exactly into two disjoint sets X and Y, which are the observed and output variables, respectively. Under CRF, the inference problem for CRF is essentially the same as for a Markova Random Field (MRF) where an input random variable X is given, and the output Y is observed. The output variable Y represents a saccade target (fixation) label sequence, and the input variable X represents the word sequence that needs to be marked or labeled. The reading saccade target prediction problem to be solved is transformed into a sequence labeling problem in this paper. Under the condition that the random variable X is x, the conditional probability for the random variable Y to be valued as y is

$$P(y \mid x) = \frac{1}{Z(x)} \cdot \exp\left[\sum_{i,k} \lambda_k t_k (y_{i-1}, y_i, x, i) + \sum_{i,l} \mu_l s_l (y_i, x, i)\right] \quad (18)$$

In this expression, Z (x) is a scaling factor; t_k is a transfer feature function that depends on the current landing position and the previous position. s_l is a state feature function that also depends on the current landing position. The value of the feature function t_k and s_l is 1 or 0, which means when it meets the feature condition, the value is 1; otherwise it is 0. λ_k and μ_l are weights, which are obtained by training the model. Parameter training is achieved based on a maximum likelihood criterion and maximum a posteriori criterion, whose goal is to maximize the probability of correctly marking the target sequence in the training set.

Whether the current word (x) is a fixation word (y_i) depends not only on the feature value of the current word (x), but also on whether the previous word is a fixation word (y_{i-1}). This coincides with the characteristics of the linear chain CRF.

4.4. CNN-LSTM-CRF Architecture. In the final stage, we built our neural network model by feeding the output vector of the bi-LSTM into the CRF layer. Figure 3 details our network architecture.

5. Model Training

In this section, we provide detailed information on training neural networks. We use the *keras-contrib* library [36] to implement a neural network that contains useful extensions to the official *Keras* package [37]. Model training is run on the GeForce GTX 1070 graphical processing unit. It takes approximately 20 min to complete the model training using the setup discussed in this section.

5.1. Datasets. There are several eye-tracking corpora in existence. Among these, the Dundee corpus [38] is notable. However, the Dundee corpus is not publicly available due to licensing restrictions. Our experiments used data from the Provo corpus [39], which is publicly accessible and may be downloaded from the Open Science Framework at https://osf.io/sjefs. The eye-tracking corpus has eye movement data from 84 native English-speaking participants, all of whom read the complete 55 texts for comprehension, including online news articles, popular science magazine articles, and public domain works of fiction. Eye movements were documented using a SR Research EyeLink 1000 Plus eye-tracker with a spatial resolution of 0.01° and a sampling rate of 1000 Hz (see [39] for details).

For the experiments reported here, the corpus was randomly divided into three data sets in the following proportions: 60% texts for training, 20% texts for development and validation, and the last 20% texts for testing.

5.2. Features. Evidence from the psychological literature indicates that the selection mechanism of fixation and saccade is determined by the combination of low-level visual cues (such as word length) and language cognitive factors of the text [7, 40]. The linguistic factors known to influence whether or not to skip words are word length, frequency, and predictability; that is, shorter words in the text are easier to skip than longer words. Predictability involves high-level cognitive factors that are difficult to quantify. Thus, we use parts of speech (POS) as a proxy to represent the high-level cognitive factors. Words in the corpus are tagged for parts of speech using the Constituent Likelihood Automatic Word-Tagging System (CLAWS) [41]. Using the CLAWS tags, words are divided into nine separate classes. The passages contain a total of 682 nouns, 502 verbs, 364 determiners, 287 prepositions, 227 adjectives, 196 conjunctions, 169 adverbs, 109 pronouns, and 153 other words and symbols.

Ultimately, the features used by the neural network consist of a tokenized word, word length (for low-level visual features), and parts of speech (for high-level cognitive features).

5.3. Training Procedure. The model used in this paper was trained using general stochastic gradient descent (SGD),

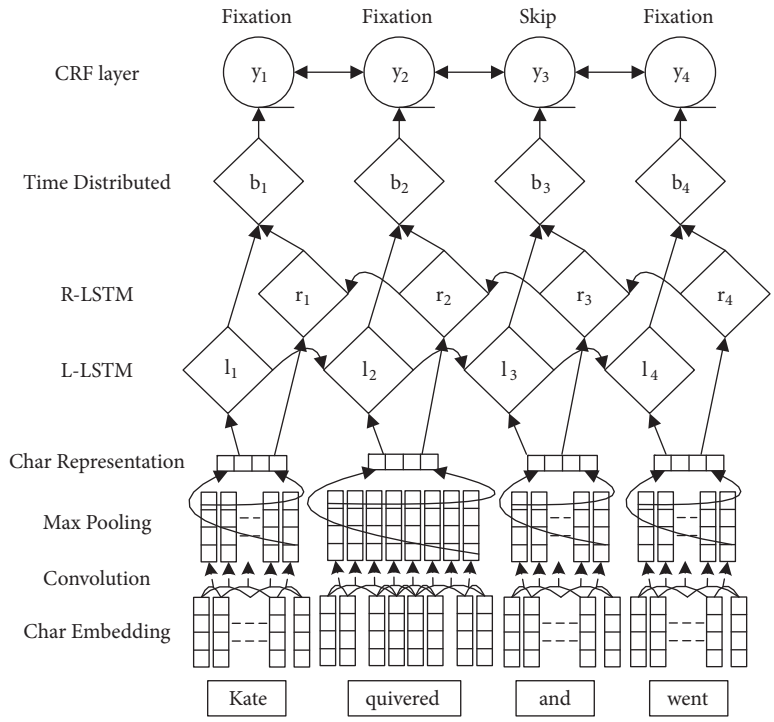


Figure 3: The CNN-LSTM-CRF architecture for predicting fixation in reading. The first layer is the embedding layer, which vectorizes the text corpus by converting each text into a sequence of integers. The second layer is the CNN layer, which extracts word form information from characters in a word and encoding it into a neural representation. The third layer is the bidirectional LSTM neural network, which can effectively use past (left) and future (right) contexts. The fourth layer is the CRF layer, which is used for sentence-level sequence labeling.

(1) **for** epoch **in** epochs:
(2) **for** batch **in** batchs:
(3) (1) bi-LSTM model forward pass:
(4) forward pass for forward state LSTM
(5) forward pass for backward state LSTM
(6) (2) CRF layer forward and backward pass
(7) (3) bi-LSTM model backward pass:
(8) backward pass for forward state LSTM
(9) backward pass for backward state LSTM
(10) (4) update parameters

Algorithm 1: The model training procedure.

forward propagation, and backpropagation (BP) algorithms. The training procedure is shown in Algorithm 1.

For each training sample, the algorithm first initialized random weights and threshold parameters, then provided relevant input examples to the input layer neurons, and forwarded the signals layer by layer (input layer -> hidden layer -> output layer) until the output layer produced an output value. Then, the error of the output were calculated according to the output value, and then the error was reversely propagated to the neurons of the hidden layer, and finally the weight of the connection and the threshold of the neuron were adjusted according to the error calculated by the hidden layer neurons. The BP algorithm continually iteratively looped through the above steps until the conditions of stopping training were reached.

5.4. Tuning Hyperparameters. The hyperparameters that need to be determined include (1) word embeddings dimension, (2) word embeddings length, (3) convolution kernel size, (4) maxpool size, (5) neuron activation function type, (6) cost function, (7) weight initialization method, (8) number of neurons in each hidden layer, (9) optimizer, (10) learning rate, (11) learning epoch, and (12) batch size. Among these hyperparameters, (1) and (2) were determined by the size of the development set, (6), (7), and (8) were determined by some common strategies, and (3), (4), (5), (9), (10), (11), and (12) were determined by random search.

Since there were 1200 different words in the development set, and each sentence contained less than 60 words, we set the word embeddings dimension to 1200 and the word embeddings length to 60. In the one-dimensional convolution operation, it was equivalent to the feature extraction of *n_gram* using neural network, so the optional parameters were 2, 3, and 4.

To find the optima values for hyperparameters, we used the following strategies: first, we determined the type of activation function, and then determined the type of cost function and the method of weight initialization. Secondly, according to the network topology, the number of neurons in each hidden layer in the neural network was determined. Then, for the remaining hyperparameters, a possible value

Table 1: Hyperparameter settings.

Layer	Hyper-parameter	Value	Parameter Range
Embedding	output dim	32	/
	input dim	1200	/
	input length	60	/
CNN	kernel size	3	[2, 3, 4]
	maxpool size	3	[2, 3, 4]
	stride size	1	/
L-LSTM	units	32	/
R-LSTM	units	32	/
Time Distributed	units	50	/
Global	activation	ReLU	[Sigmoid, Tanh, ReLU]
	cost	cross_entropy	/
	kernel initializer	random_normal	/
	optimizer	SGD	[SGD, AdaDelta, Adam, RMSProp]
	learning rate	2.5	[0.025, 0.25, 2.5]
	learning epoch	100	[50, 100, 150, 200]
	batch size	16	[8, 16, 32]
	validation split	0.2	/

was randomly given first. In the cost function, the existence of the regular term was not considered first, and the learning rate was adjusted to obtain a suitable threshold. Half of the threshold was taken as the initial value in the process of adjusting the learning rate. The size of the batch was determined through experiments. Then we used the determined learning rate to carefully adjust the learning rate and the validation data to select good regularization parameters. After the regularization parameters were determined, we went back and reoptimize the learning rate. The number of epochs was determined by a whole observation through the above experiments.

Among all the parameters, the type of neuron activation function should be selected first. The Sigmoid, Tanh, and ReLU functions are the most commonly used activation functions [42], but the Sigmoid and Tanh functions will encounter the problem of gradient disappearance, which is not conducive to the extension of the neural network to a deeper structure. The ReLU overcomes this problem because it solves the problem of gradient disappearance and therefore allows the neural network to extend to deeper layers. So we chose the ReLU as the activation function here.

Since our task was a classification task, the cost function that should be used in the experiment was the cross-entropy. For an input layer with n_{in} neurons, the initialization weight is a Gaussian random distribution with a mean of 0 and a standard deviation of $1/\sqrt{n_{in}}$. We determined the optimizer by random search. The super-parameters to be tuned included SGD [42], AdaDelta [43], Adam [44], and RMSProp [45].

The adjustment steps of the learning rate were as follows: first, we choose the estimation that the cost on the training data immediately began to decrease rather than oscillate or increase as the learning rate threshold, and it was not necessary to be too precise to determine the magnitude. If the cost began to decline in the first few rounds of training, we gradually increased the magnitude of the learning rate. If the cost function curve began to oscillate or increase, we tried to reduce the magnitude until the cost fell at the beginning of the round. Taking half of the threshold determined the learning rate.

Table 1 summarizes the relevant hyperparameters used in the experiments determined by the development set, the common strategies, and the random search method. The last column in the table is the parameter range of the random search. In order to avoid the overfitting problem, we used a simple dropout strategy [46] to drop 10% of the redundant nodes.

6. Experimental Verification

6.1. Evaluation Metrics and Baselines. We followed the evaluation metrics and baselines in NN09 [12] and HMKA12 [17]. First, we quantified the number of words in the test set and used the fixation word rate for each subject as a baseline (see Table 2). Then, the accuracy of the fixation word prediction was used as an evaluation metrics, and the fixation/skip distribution of each word in the verification set was predicted. The accuracy of each predicted distribution was calculated from the distribution in the fixation data. Finally, the accuracy of each word in the validation set was averaged.

6.2. Result Analysis. Based on the analysis in Section 5.2, we set up features to predict the fixation points. The examined features can be classified into two types: low-level visual features and high-level cognitive features. In the experiments,

Table 2: Baseline rates for fixated words in the test data.

Subjects	Sub1	Sub2	Sub3	Sub4	Sub5	Sub6	Sub7	Sub8	Sub9	Sub10
# fixated words	1,907	2,158	2,120	1,788	1,666	2,040	1,856	1,989	1,537	2,046
Rate[%]	69.52	78.67	77.29	65.18	60.74	74.37	67.66	72.51	56.03	74.59
# words in test data	2,743(100%)									

Table 3: Mean accuracy and standard deviation (averaged over 100 runs) for the fixation prediction task.

Subjects	Baseline[%]	POS[%]		WL[%]		POS&WL[%]	
		Mean	Δ_{std}	Mean	Δ_{std}	Mean	Δ_{std}
Sub1	69.52	70.34	1.05	78.36	1.95	79.87	1.50
Sub2	78.67	78.91	1.10	81.39	1.53	83.87	1.80
Sub3	77.29	79.66	1.18	80.83	1.84	82.77	1.58
Sub4	65.18	68.85	1.82	76.65	2.03	78.52	2.04
Sub5	60.74	63.42	1.27	71.31	1.54	74.71	1.35
Sub6	74.37	75.21	1.12	77.63	1.50	79.29	1.05
Sub7	67.66	69.48	1.05	72.58	1.34	75.69	2.13
Sub8	72.51	73.75	1.85	76.67	2.09	78.47	1.76
Sub9	56.03	61.91	1.27	70.76	1.58	73.92	1.16
Sub10	74.59	75.79	0.96	79.09	1.40	79.91	1.72
Average	69.66	71.73	1.27	76.53	1.68	78.70	1.61

we explored the contribution of low-level visual and high-level cognitive features individually and in combinations to prediction accuracy.

Based on the experimental settings discussed in Section 5, we run at least 100 times with different random initialization of parameters. All experiments were trained using Stochastic Gradient Descent (SGD). We reported the mean accuracy and the standard deviation (Δ_{std}) and thus more fully characterize the distribution of accuracies in the predictions.

The experimental results in Table 3 show that the word length feature (for low-level visual features, denoted by "WL") has an accuracy of 76.53%, while the part of speech feature (for high-level cognitive features, denoted by "POS") provides less accuracy.

The achieved test performances can be plotted in a violin plot as shown in Figure 4. The violin plot is similar to a boxplot; however, it estimates from the samples the probability density function and depicts it along the Y-axis. If a violin plot is wide at a certain location, then achieving this test performance is especially likely. Besides the probability density it also shows the median as well as the quartiles. In Figure 4 we can observe that the WL feature usually results in a higher performance than the POS feature for the fixation prediction task. Hence, we can conclude that the low-level visual feature is a better option for this task.

We also considered combinations of the two feature types. From the Figure 5, we can see that adding other features to the WL feature barely contributes to an improvement in accuracy. Additionally, the influence of the POS feature on improving the accuracy is not obvious. The prediction accuracy obtained by the POS feature is similar to the baseline accuracy. These observations seem to imply that high-level cognitive features do not capture very much extra information when using the features separately. This would suggest that combined features work well only in conjunction with low-level visual features.

In summary, we can draw the conclusion that the low-level visual cues have a key impact on the selection of saccade targets compared with the high-level cognitive factors.

The authors in [12, 17] used the Dundee corpus to train and test their models. Owing to licensing restrictions, our experiments are based on the data from the Provo corpus. Because of the different experimental settings, we cannot simply compare our experimental results with those from [12, 17]. However, considering that we were able to obtain similar accuracy as those using NN09 [12] or HMKA12 [17], and, moreover, we used far fewer features and required much less preprocessing than did NN09 or HMKA12 (see Table 4), these results indicate that the proposed RNN-based model performs well in simulating reading eye movements.

6.3. Application Example. Interactive graphics environments require high refresh rates, high resolutions, and low latencies, each of which adds computational burden on the hardware. To address the problems, the state-of-the-art technology that integrates with eye tracking is known as *foveated rendering* [47]. Foveated rendering can make the fixation point that the user is focusing on clearer and replaces the adjacent area with a blurred image, which is in line with the mode of human vision. In this way, the machine does not have to render the entire picture in detail, which can greatly reduce the computational burden on the GPU. However, accurate fixation tracking systems are still expensive and can only be accessed by a limited number of researchers or companies [48]. Another way to compute the fixation point is to use an eye movement model that simulates the gaze behavior of human.

Table 4: Comparison between E-Z Reader, NN09, HMKA12, and RNN-based models.

Parameters	E-Z Reader	NN09	HMKA12	RNN-based Model
# training sentences	/	157.8	157.8	137.5
# training features	/	8	7	2
Average fixation accuracy	57.7%	69.5%	78.601%	78.702%

Table 5: Comparison between reference latency, HMKA12 latency, and RNN-based latency.

Count of pixels	Reference Latency		HMKA12 Latency		RNN-based Latency	
	Single-GPU	Multi-GPU	Single-GPU	Multi-GPU	Single-GPU	Multi-GPU
32M	780 ms	577 ms	96.2 ms	74.0 ms	27.6 ms	19.7 ms
16M	532 ms	410 ms	95.6 ms	65.4 ms	27.6 ms	19.8 ms
8M	385 ms	289 ms	95.0 ms	58.6 ms	27.4 ms	16.9 ms
4M	267 ms	212 ms	94.4 ms	62.2 ms	27.4 ms	17.4 ms
2M	98.6 ms	75.4 ms	93.8 ms	63.1 ms	27.1 ms	17.6 ms
1M	90.4 ms	72.1 ms	89.9 ms	63.7 ms	27.3 ms	18.3 ms

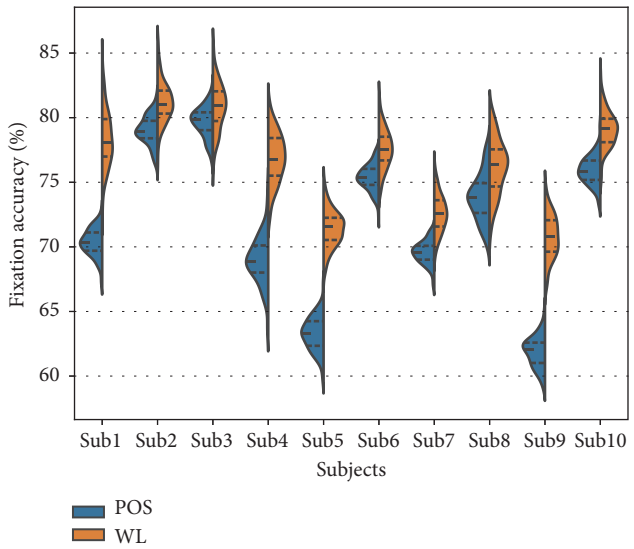


Figure 4: Performance on the fixation prediction task for various subjects using the POS feature or the WL feature.

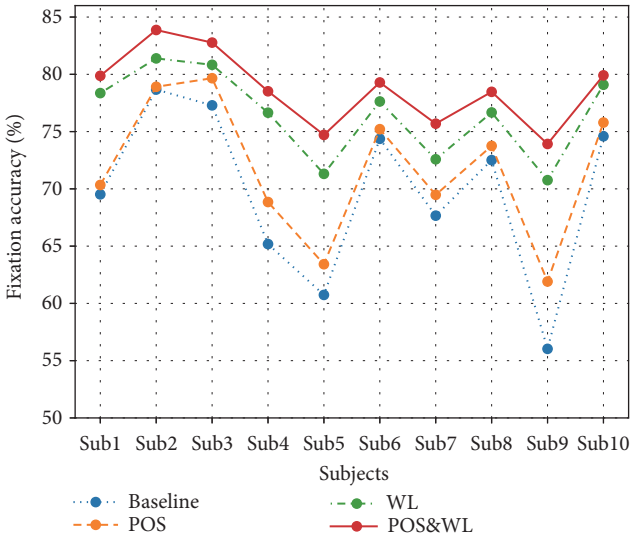


Figure 5: Fixation prediction accuracy comparison using different features on the validation data.

The proposed model requires a smaller number of input features than any of the alternatives, while the prediction accuracy is similar to that of current machine learning models. Requiring fewer features and reducing the preprocessing of data make the proposed model attractive in a range of human-computer application areas. For example, the latency can be reduced in an interactive graphics environment.

We constructed an application example using the architecture proposed in [49] to demonstrate that the proposed model can improve system performance by reducing latency. This was accomplished by a fixation-predicting architecture with a parallel client and server process that accesses a shared scene graph (Figure 6). Low latency and constant delay are ideal features of an interactive system. We examined the latency for single- and multi-GPU fixation-predicting implementations as well as for a standalone regular renderer for reference with a method first documented by Steed [50].

Table 5 lists the results obtained from the experiments and the results show the average delay obtained in several experiments. The reference render has a much higher latency, and the amount of delay depends on the number of pixels and frame rate in the scene. When the number of pixels is reduced to 4M or lower, the multi-GPU delay is slightly increased, which is affected by frequent asynchronous data transmission and context switching between threads, and a single GPU is not affected by this. In the case of small scenes, it is best to use direct rendering. However, when rendering is larger than 2M pixels, the fixation prediction method can significantly reduce the delay, and the RNN-based model has a lower delay.

6.4. Discussion. The RNN-based model achieves similar fixation prediction accuracy to current machine learning models

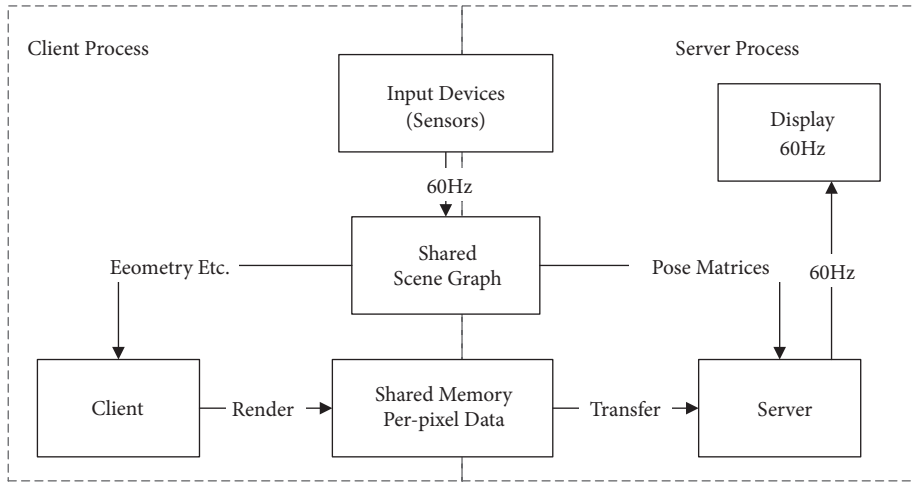


Figure 6: Overview of the gaze point rendering application architecture (adapted from [49]).

while requiring fewer features. We believe that there are two reasons for this. On the one hand, it uses a convolution operation and objectively increases the number of training samples. On the other hand, patterns of fixations and saccades are driven in part by low-level visual cues and high-level linguistic and cognitive processing of the text. CRF can account for the transfer features and state features of label sequences, in line with how humans handle low-level visual features. A RNN is a time-recursive neural network that can process and foresee events following particularly large intervals and delays in a time series, in correspondence with human handling of high-level visual features. Placing the RNN before the CRF is equal to utilizing the language relationships extracted by the RNN to train the CRF. The proposed model takes advantage of the context of the text sequence and the label sequence, which is more in line with the reality of the reading process.

7. Conclusions

In this paper, a new type of eye movement model was developed and evaluated in terms of its ability to simulate eye movements of human reading. Theoretical analysis demonstrated that the RNN-based model always converges to a global solution. In comparison with conventional eye movement models, the new approach was shown to achieve similar accuracy in predicting a user's fixation points during reading. In addition, the proposed model has less reliance on data features and requires less preprocessing than current machine learning models, which makes the proposed model attractive in a range of human-computer application areas. The verification results further demonstrate the novelty and efficacy of RNNs for simulating eye movements of human reading.

Data Availability

The eye-tracking corpus data used to support the findings of this study are available from the Open Science Framework at https://osf.io/sjefs.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants nos. 61231016 and 61871326) and the General Project of Humanities and Social Sciences Research of Ministry of Education of China (Grant no. 18YJCZH180).

References

[1] G. Stockman and L. G. Shapiro, *Computer Vision*, Prentice Hall, 2001.

[2] X. J. Bai and G. L. Yan, *Psychology of Reading*, East China Normal University Press, 2017.

[3] H. X. Meng, *The Selection Mechanism of Saccade Target in Chinese Reading*, World Book Publishing Guangdong Co., Ltd., 2016.

[4] X. L. Liu, *Visual Neurophysiology*, People's Medical Publishing House, 2011.

[5] K. Rayner, "Eye movements in reading and information processing: 20 years of research," *Psychological Bulletin*, vol. 124, no. 3, pp. 372–422, 1998.

[6] R. Radach and G. W. Mcconkie, "Determinants of fixation positions in words during reading," in *Eye Guidance in Reading and Scene Perception*, G. Underwood, Ed., pp. 77–100, Elsevier, 1998.

[7] C. C. Jr, F. Ferreira, J. M. Henderson et al., "Eye movements in reading and information processing: Keith Rayner's 40 year legacy," *Journal of Memory and Language*, vol. 86, no. 1, pp. 1–19, 2016.

[8] S. G. Luke and K. Christianson, "Limits on lexical prediction during reading," *Cognitive Psychology*, vol. 88, no. 6, pp. 22–60, 2016.

[9] E. D. Reichle, "Computational models of reading: a primer," *Language and Linguistics Compass*, vol. 9, no. 7, pp. 271–284, 2015.

[10] E. D. Reichle, K. Rayner, and A. Pollatsek, "The E-Z reader model of eye-movement control in reading: comparisons to other models," *Behavioral and Brain Sciences*, vol. 26, no. 4, pp. 445–476, 2003.

[11] R. Engbert, A. Nuthmann, E. M. Richter et al., "SWIFT: a dynamical model of saccade generation during reading," *Psychological Review*, vol. 112, no. 4, pp. 777–813, 2005.

[12] M. Nilsson and J. Nivre, "Learning where to look: modeling eye movements in reading," in *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL)*, pp. 93–101, Boulder, Colo, USA, 2009.

[13] N. Landwehr, S. Arzt, T. Scheffer, and R. Kliegl, "A model of individual differences in gaze control during reading," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1810–1815, Doha, Qatar, 2014.

[14] M. Hahn and F. Keller, "Modeling human reading with neural attention," 2016, http://cn.arxiv.org/abs/1608.05604.

[15] M. Nilsson and J. Nivre, "Towards a data-driven model of eye movement control in reading," in *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pp. 63–71, Uppsala, Sweden, 2010.

[16] F. Matties and A. Søgaard, "With blinkers on: robust prediction of eye movements across readers," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 803–807, Seattle, Wash, USA, 2013.

[17] T. Hara, D. Mochihashi, Y. Kano et al., "Predicting word fixations in text with a CRF model for capturing general reading strategies among readers," in *Proceedings of the First Workshop on Eye-tracking and Natural Language Processing*, pp. 55–70, Mumbai, India, 2012.

[18] U. R. Acharya, S. L. Oh, Y. Hagiwara et al., "Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals," *Computers in Biology and Medicine*, vol. 100, pp. 270–278, 2018.

[19] J. Jiang, P. Trundle, and J. Ren, "Medical image analysis with artificial neural networks," *Computerized Medical Imaging and Graphics*, vol. 34, no. 8, pp. 617–631, 2010.

[20] J. Ren, "ANN vs. SVM: which one performs better in classification of MCCs in mammogram imaging," *Knowledge-Based Systems*, vol. 26, no. 2, pp. 144–153, 2012.

[21] J. Ren, D. Wang, and J. Jiang, "Effective recognition of MCCs in mammograms using an improved neural classifier," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 4, pp. 638–645, 2011.

[22] Y. Guo, G. Ding, and J. Han, "Robust quantization for general similarity search," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 949–963, 2018.

[23] H. Liu, L. Dai, S. Hou, J. Han, and H. Liu, "Are mid-air dynamic gestures applicable to user identification?" *Pattern Recognition Letters*, vol. 117, pp. 179–185, 2019.

[24] H. Liu, Z. Fu, J. Han, L. Shao, S. Hou, and Y. Chu, "Single image super-resolution using multi-scale deep encoder-decoder with phase congruency edge map guidance," *Information Sciences*, vol. 473, pp. 44–58, 2019.

[25] G. Wu, J. Han, Y. Guo et al., "Unsupervised deep video hashing via balanced code for large-scale video retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2019.

[26] G. Wu, J. Han, Z. Lin, G. Ding, B. Zhang, and Q. Ni, "Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning," *IEEE Transactions on Industrial Electronics*, 2018.

[27] S. Luan, C. Chen, B. Zhang, J. Han, and J. Liu, "Gabor convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4357–4366, 2018.

[28] R. Collobert, J. Weston, L. Bottou et al., "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[29] K. Liang, N. Qin, D. Huang, and Y. Fu, "Convolutional recurrent neural network for fault diagnosis of high-speed train bogie," *Complexity*, vol. 2018, Article ID 4501952, 13 pages, 2018.

[30] C. A. Martín, J. M. Torres, R. M. Aguilar, and S. Diaz, "Using deep learning to predict sentiments: case study in tourism," *Complexity*, vol. 2018, Article ID 7408431, 9 pages, 2018.

[31] S. S. Du, X. Zhai, B. Poczos et al., "Gradient descent provably optimizes over-parameterized neural networks," 2018, http://cn.arxiv.org/abs/1810.02054.

[32] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proceedings of the 31st International Conference on Machine Learning*, pp. 1818–1826, Beijing, China, 2014.

[33] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," 2015, http://cn.arxiv.org/abs/1511.08308.

[34] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," 2015, http://cn.arxiv.org/abs/1505.08075.

[35] M. Boden, "A guide to recurrent neural networks and back propagation," *The Dallas project*, 2002.

[36] Keras-contrib library, 2018, https://github.com/keras-team/keras-contrib.

[37] Keras python library, 2018, https://keras.io.

[38] A. Kennedy, R. Hill, and J. Pynte, "The Dundee corpus," in *Proceedings of the 12th European Conference on Eye Movement*, Dundee, Scotland, 2003.

[39] S. G. Luke and K. Christianson, "The provo corpus: a large eye-tracking corpus with predictability norms," *Behavior Research Methods*, vol. 50, no. 2, pp. 826–833, 2018.

[40] A. Kennedy, J. Pynte, W. S. Murray, and S.-A. Paul, "Frequency and predictability effects in the dundee corpus: an eye movement analysis," *The Quarterly Journal of Experimental Psychology*, vol. 66, no. 3, pp. 601–618, 2013.

[41] R. Garside and N. Smith, "A hybrid grammatical tagger: CLAWS4," in *Corpus Annotation: Linguistic Information from Computer Text Corpora*, R. Garside, G. N. Leech, and T. McEnery, Eds., pp. 102–121, Longman, London, UK, 1997.

[42] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2013, http://cn.arxiv.org/abs/1211.5063v2.

[43] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," 2012, http://cn.arxiv.org/abs/1212.5701.

[44] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[45] Y. N. Dauphin, H. De Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," *Advances in Neural Information Processing Systems*, vol. 35, no. 3, pp. 1504–1512, 2015.

[46] N. Srivastava, G. Hinton, and A. Krizhevsky, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] M. Weier, T. Roth, A. Hinkenjann et al., "Predicting the gaze depth in head-mounted displays using multiple feature regression," in *Proceedings of the 10th ACM Symposium on Eye Tracking Research and Applications (ETRA)*, pp. 1–9, Warsaw, Poland, 2018.

[48] E. Arabadzhiyska, O. T. Tursun, K. Myszkowski et al., "Saccade landing position prediction for gaze-contingent rendering," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–12, 2017.

[49] F. Smit, R. van Liere, S. Beck et al., "A shared-scene-graph image-warping architecture for VR: low latency versus image quality," *Computers and Graphics*, vol. 34, no. 1, pp. 3–16, 2010.

[50] A. Steed, "A simple method for estimating the latency of interactive, real-time graphics simulations," in *Proceedings of the VRST ACM Symposium on Virtual Reality Software and Technology*, pp. 123–129, Bordeaux, France, 2008.

Hindawi
Complexity
Volume 2019, Article ID 9180391, 12 pages
https://doi.org/10.1155/2019/9180391

Research Article

Weakly Supervised Deep Semantic Segmentation Using CNN and ELM with Semantic Candidate Regions

Xinying Xu,[1] Guiqing Li,[1] Gang Xie ,[1,2] Jinchang Ren ,[1,3] and Xinlin Xie[1]

[1]*College of Electrical and Power Engineering, Taiyuan University of Technology, Taiyuan, China*
[2]*College of Electronic Information Engineering, Taiyuan University of Science and Technology, Taiyuan, China*
[3]*University of Strathclyde, Department of Electronic and Electrical Engineering, Glasgow, UK*

Correspondence should be addressed to Gang Xie; xiegang@tyut.edu.cn and Jinchang Ren; jinchang.ren@strath.ac.uk

Received 15 November 2018; Revised 3 February 2019; Accepted 25 February 2019; Published 14 March 2019

Guest Editor: Jungong Han

The task of semantic segmentation is to obtain strong pixel-level annotations for each pixel in the image. For fully supervised semantic segmentation, the task is achieved by a segmentation model trained using pixel-level annotations. However, the pixel-level annotation process is very expensive and time-consuming. To reduce the cost, the paper proposes a semantic candidate regions trained extreme learning machine (ELM) method with image-level labels to achieve pixel-level labels mapping. In this work, the paper casts the pixel mapping problem into a candidate region semantic inference problem. Specifically, after segmenting each image into a set of superpixels, superpixels are automatically combined to achieve segmentation of candidate region according to the number of image-level labels. Semantic inference of candidate regions is realized based on the relationship and neighborhood rough set associated with semantic labels. Finally, the paper trains the ELM using the candidate regions of the inferred labels to classify the test candidate regions. The experiment is verified on the MSRC dataset and PASCAL VOC 2012, which are popularly used in semantic segmentation. The experimental results show that the proposed method outperforms several state-of-the-art approaches for deep semantic segmentation.

1. Introduction

Image semantic segmentation is the understanding of the semantic information contained in images. It uses the computer to extract semantic information of the captured scene from the image for understanding its contents, which can be applied in image recognition, classification, and analysis [1]. Semantic segmentation has been widely used in intelligent robot scene understanding, automatic driving system streetscape recognition, and medical image detection [2]. However, semantic segmentation has become one of the most challenging computer vision tasks due to the scale, position, illumination, and texture changes of objects in the image [3].

In most cases, image semantic segmentation is established as a fully supervised task. The fully supervised methods require using strong pixel-level annotations, which is very limited, expensive, and time-consuming in the labeling process, and it is different due to the subjective understanding of the labeling personnel [4]. However, weakly supervised semantic segmentation only requires image labels at the image-level, which is much cheaper and less time-consuming than pixel-level annotations. Weakly supervised semantic segmentation can be divided into three categories that included bounding box [5], partial marking [6], and image-level labels. At present, with the increasing popularity of image sharing websites (for example, Flickr) and providing a large number of user-labeled images, many studies have focused on image-level labels for weakly supervised semantic segmentation.

Therefore, the semantic segmentation of weakly supervised images based on image-level labels has gradually increased recently. According to the different methods of semantic label inference, the weakly supervised image semantic segmentation can be roughly divided into classifier, multigraph model, and deep convolutional neural network based methods. Among them, the first classifier-based method uses the superpixels or the candidate regions

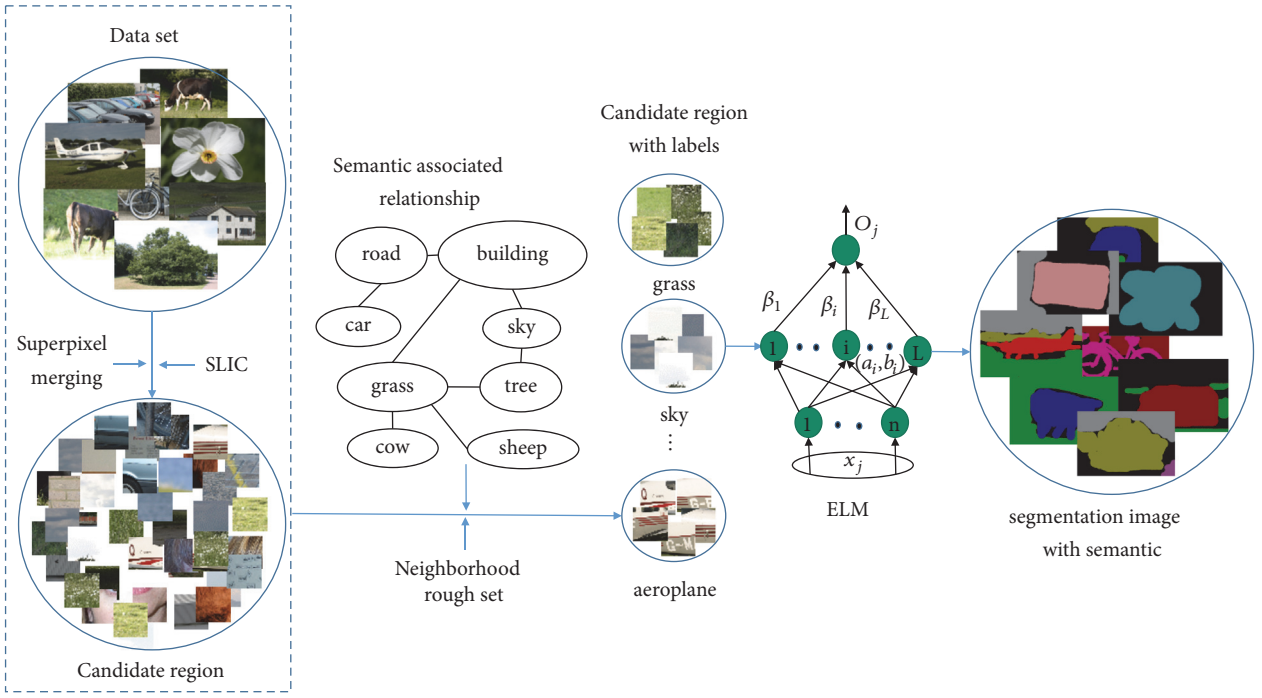


Figure 1: Flow chart of algorithm framework.

generated by superpixel as the basic processing unit to infer semantic label and then selects various classifier models to learn the inferred label. The main idea is that the superpixels or candidate regions with the same semantic label have similar appearance [7]. However, semantic label inference based on superpixel contains more redundant information, which can interfere with the accuracy. Although the methods based on candidate regions contain less redundant information, it is difficult to completely and accurately segment the number of image objects equaling the number of the labels by the current image segmentation techniques. Then the based multigraph model method uses all pixels or superpixels in the image as graph model nodes. And graph model is established with relationship between pixels or superpixels. But this method calculates a one-dimensional potential energy function for each superpixel and the algorithm complexity is high [8]. Fortunately, sparse representation and image hashing are powerful tools for data representation and the combinations of these two tools for scalable image retrieval. It is possible to replace the high-dimensional features with a low-dimensional Hamming space with preserving the similarity between features, which will reduce the computational complexity of the energy function, thereby reducing the complexity of the algorithm [9–14]. In addition, the deep convolutional neural network based method uses a pretrained classification network to obtain objects of the image and then fine tunes by segmentation networks and image-level labels. The methods are sensitive to the accuracy and dataset of the pretrained classification network. And the classification network can only identify small and discernible regions, which is insufficient for the inference of large-scale image-level semantic labels [15].

Although the weakly supervised image semantic segmentation based on the image-level labels is proposed constantly, its segmentation accuracy has a large room for refinement compared with the fully supervised image semantic segmentation. The main obstacles and difficulties lie in how to accurately implement the semantic label inference, that is, the accurate mapping from the image-level labels to image pixel positions. In addition, as a dense pixel-level label prediction task, not all features are equally important and discriminative for learning classification models [16]. Therefore, how to construct an effective model to infer semantic labels is also meaningful for improving the accuracy of weakly supervised image semantic segmentation.

Under the condition of weak supervision, this paper proposes a deep semantic segmentation using CNN and ELM with semantic candidate regions. The proposed method uses candidate region instead of the superpixel as the basic processing unit, and the neighborhood rough set combines with the semantic associated relationship between image-level labels to infer semantic label. In addition, the ELM is trained by candidate region contained semantic information to classify test candidate regions. The algorithm flow chart is shown in Figure 1 and the main contributions of this paper are as follows: (1) A method for merging superpixel into candidate regions is proposed. The method guides superpixel merging with the number of image-level labels as supervised information and generates candidate region with high precision, which can solve the problem that multiple instances are not adjacent in an image. And merging process can reduce complexity of subsequent processing practically. (2) An inference method of candidate region semantic label is proposed. The method uses the neighborhood rough set

to generate different neighborhood particles and starts from the highest frequency semantic label to infer. Then the other candidate regions semantic labels are inferred based on the strongest associated relationship, which solves the problem of semantic label mapping difficultly. (3) An ELM training method is proposed. It uses candidate region with semantic labels to train ELM, which can reduce the introduction of negative sample pixels in the training data and improve accuracy of classification.

2. Related Work

As the simplest and most effective form of weak supervision, image-level labels are widely used in weakly supervised image semantic segmentation. It is difficult to correspond to image objects if only image-level labels data is used for training, since image-level labels cannot provide accurate information to describe boundaries and locations of objects due to inherent ambiguity of image-level labels. According to the different methods of semantic label inference, the paper divides the weakly supervised image segmentation algorithm into three categories: classifier, multigraph model, and deep convolutional neural network based methods.

The classifier based method uses image-level labels as supervised information and divides all pixels or superpixels in the image contained target label into positive samples and other negative samples without target label. Then classifier is trained directly and the best classifier is obtained by iteratively optimizing loss function. For example, Wei et al. [23] trained a multilabel classification network, where pictures are classified through the network, and finally matched the classification information with higher confidence to the original picture to obtain association between semantic labels and locations. However, this method directly introduces the pixel points of target image block as object regions into many negative sample pixels, such as pixels belonging to the background. Subsequently, Wei et al. [19, 22] proposed a simple to complex framework (STC) in 2017, which firstly trains an initial segmentation network using simple images and then predicts the labels of simple images using the network and uses these labels to enhance training semantic segmentation network. Finally, the enhanced network is used to predict labels of more complex images and train a better semantic segmentation network. However, this method requires collecting a large number of simple pictures; otherwise it is difficult to train a higher performance initialization network and continue to improve, and it has many training samples and long training time. Zhang et al. [18] proposed to use the spatial sparse reconstruction method to obtain an effective SVM classifier, which trains classifier by training data with noise, and to use method of subspace reconstruction to denoising and find optimal SVM classifier by iterative optimization. The methods iterate between generating temporary segmentation masks and learning with interim supervision. These methods benefit from pixel-level supervision; but errors easily accumulate in iterations.

The multigraph model based method uses all pixels or superpixels in the image as graph model nodes. And graph model is established with relationship between pixels or superpixels. Vezhnevets et al. [8] proposed a multi-instance learning (MIL) framework for weakly supervised images segmentation. The algorithm regards each superpixel as an instance; each image is represented as a series of instance sets. Only labels of instance set are known, so image segmentation is converted to instance label inference. But the algorithm lacks the labels between superpixel pairs. In order to solve this problem, Vezhnevets et al. [17] proposed a multi-image model (MIM) based on the graph model and built a common probability graph model on the training set and test set using conditional random fields for each superpixel. The one-dimensional potential energy function establishes a binary potential energy function between superpixel pairs and finally approximates parameters of conditional random field by method of graph division. However, this method calculates a one-dimensional potential energy function for each superpixel and the algorithm complexity is high. In order to enrich the description of superpixel features, Vezhnevets et al. [24] further proposed a series of parameterized structured models in which potential energy pairs are formed by multichannel visual features, and weight of each channel is determined by minimizing to distinguish different superpixel labels of trained segmentation model. The above graph-based algorithm has improved segmentation performance in weakly supervised environment, but it is limited by the low descriptiveness of the unary or binary potential energy function.

The method of deep convolutional neural network is based on DCNN framework, which is trained to obtain the object position. Oquab et al. [25] applied DCNN framework to generate a single point to infer the location of the object, but this method cannot detect multiple objects of same class in an image. Pinheiro et al. [21] and Pathak et al. [20] added segmentation constraints to final cost function to optimize parameters of DCNN image-level labels. However, the two methods generate coarse prediction because the algorithms generally do not use low-level cues.

3. The Proposed Method

The paper proposes a weakly supervised image semantic segmentation framework based on candidate regions and ELM. The framework of the paper consists of two phases of learning and testing. Among them, there are three basic steps in the learning phase: (1) candidate region segmentation using superpixel; (2) candidate region semantic inference using semantic label association; (3) candidate regions classification using ELM. In the testing phase, the paper first performs superpixel segmentation and merging on the test image and then predicts the semantic label of each pixel with the candidate region as the basic processing unit.

3.1. Segmentation of Candidate Regions Using Superpixels. Compared with superpixels, the number of candidate regions in the image is smaller, which is more helpful for improving the accuracy of semantic label inference. Therefore it is necessary to merge oversegmented superpixels to obtain candidate regions library. In addition, the several low-level

Input: Data set, image-level label number l.
Output: Cluster center for each target superpixel $C = \{c_i\}_{i=1}^{n}$, the number of target superpixels in the image n.
Step 1. SLIC superpixel segmentation, $X = [x_1, \ldots, x_n]$.
Step 2. While $n \geq 3l$
(a) Extract visual features of each superpixel: LAB(3 dim), Gabor(65 dim), Sift(64 dim), Surf(64 dim);
(b) The adjacency relationship between superpixels is counted and stored in matrix D;
(c) The superpixel similarity S is calculated according to formula (1);
(d) Combine the most similar superpixel pairs with considering the adjacency;
(e) Calculate the mean of the merged superpixel clustering centers as a new clustering center;
(f) Update n.
End
Step 3. Reclassify disconnected areas.

Algorithm 1: Superpixel merging process.

visual features are extracted to preserve the boundary information of each superpixel as much as possible during the merging process. Therefore the paper selects the colour, texture, sift, and surf features representing each superpixel. Specifically, due to the wide colour gamut of the LAB, this paper chooses the LAB as the colour feature. And this paper selects the Gabor filters to represent the texture feature of each superpixel, because the Gabor filter has the capability of dealing with spatial transformations [26].

First, the initial image is divided into superpixels based on the simple linear iterative clustering algorithm (SLIC). And compared with other superpixel segmentation methods, SLIC algorithm has the following advantages [27]: (a) the size of formed superpixels is basically the same; (b) the number of superpixels can be controlled by adjusting the parameter k; (c) the speed is fast and boundary fit between block and target boundary is high; d) the difference of features between pixels within each block is small.

Then, the 196-dimensional visual features are extracted to describe each superpixel, including colour features (3-dimension), texture features (65-dimension), Sift features (64-dimension), and Surf features (64-dimension). Finally, on the basis of superpixel spatial position adjacency, the most similar superpixels are merged by statistical superpixel similarity, and the number of superpixels is combined to be no more than three times of image labels, as shown in Figure 2.

Suppose an image contains n superpixels $X = [x_1, \ldots, x_n] \in R^{m \times n}$, and any superpixel x_i has 196 dimensional visual features to describe, image labels $y = [y_1, y_2, \ldots, y_l]$, and l is the number of image semantic labels. Then similarity of any superpixels x_i and x_j is described as

$$S_{i,j} = \sum_{i=1, j=1, and\ i \neq j}^{m} \left[\delta_1 d_{ij}^{lab} + \delta_2 d_{ij}^{tex} + \delta_3 d_{ij}^{sift} + \delta_4 d_{ij}^{surf}\right] \times D_{i,j} \quad (1)$$

where δ is weight factor of adjusting distance and satisfies $\delta_1 + \delta_2 + \cdots + \delta_4 = 1$; $d_{ij}^{lab}, d_{ij}^{tex}, d_{ij}^{sift}, d_{ij}^{surf}$ are the Euclidean distance to represent the color, texture, Sift, and Surf distance of the superpixels i and j; D stores adjacency relationship between superpixels.

$$D_{i,j} = \begin{cases} 1, & \text{if } c_i \text{ is adjacent to } c_j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The specific steps of superpixel merging algorithm are as shown in Algorithm 1.

3.2. Candidate Region Semantic Inference Using Semantic Label Association. The inference from image-level to pixel-level semantic label is the key of the whole weakly supervised image semantic segmentation algorithm. In the process, the classification of candidate regions directly affects the semantic label inference results; it is necessary to extract rich visual features. Therefore the paper adopts CNN to extract features to ensure effective classification results. However, extracting multilayer visual features increases the data dimension; it will bring great difficulties to subsequent label clustering. The neighborhood classifier [28] has an important advantage in that it can get a subset of the features that are important for decision making through attribute reduction; that is, it can obtain discriminative features that are important for semantic label inference.

As for the candidate region as the basic processing unit, the paper regards the semantic label inference as the most similar neighborhood particle extraction problem; the uniqueness of the program is as follows: (1) The paper stars inferring the semantic label from the semantic label with the most images, as much as possible to ensure the accuracy of prediction of the semantic labels; (2) According to the image-level label number and the proportion of the images corresponding to the semantic label to be inferred, the number of candidate regions is included in each semantic label to be inferred; (3) The inference of each semantic label is based on semantic label association relationship, which reduces the interference of the noise. The detailed steps are as follows:

First, semantic labels can be represented as $L = [l_1, l_2, \ldots, l_k]$; k is the total number of semantic labels categories. According to image-level labels, each semantic label corresponding to the number of images is expressed as $N =$

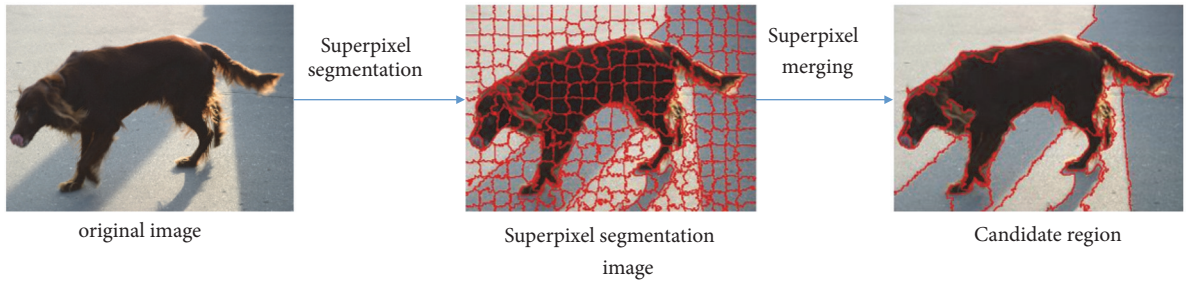


Figure 2: Flow chart of candidate region segmentation based on superpixel.

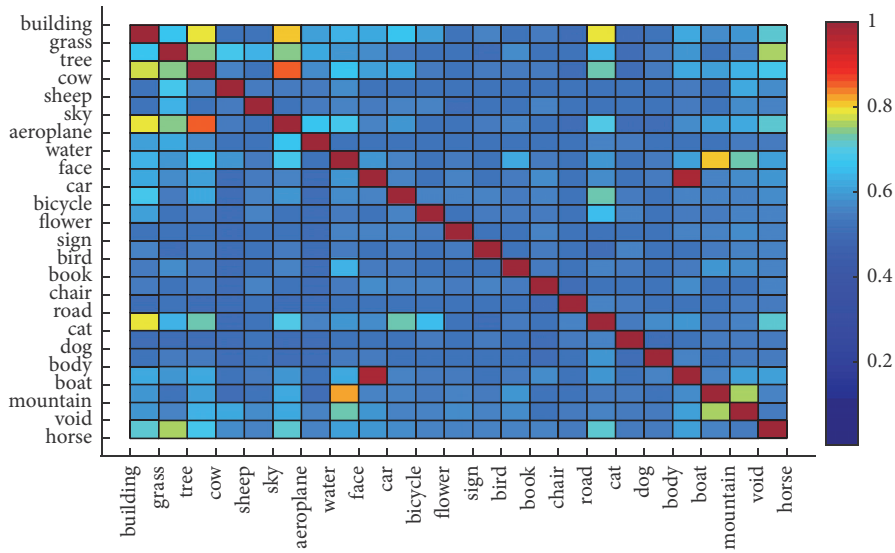


Figure 3: Image semantic correlation intensity map.

$[N(t), t = 1, 2, \ldots, k]$. According to the relationship between L and N, it can obtain a semantic label containing the most images in the data set. Then the number of candidate region set corresponding to the semantic label i can be expressed as

$$R_i = nN(i), \quad n \in R^+ \tag{3}$$

where n is a proportional parameter. It depends on the multiple of the number of image-level labels and the complexity of the training set image. Therefore, the proportion of the candidate region set corresponding to the semantic label i in the entire candidate region library can be expressed as

$$F_i = \frac{R(i)}{\sum_{t=1}^{k} R(t)} \tag{4}$$

Therefore, this paper obtains the range of the proportion of candidate regions set. And the inference of the semantic label is transformed into finding the proportion of candidate region corresponding to the semantic label.

Second, given a set of semantic labels that need to be associated, the semantic association relationship between labels is obtained by calculating the semantic association strength. And the association relationship is saved in a diagonal relationship matrix W expressed as

$$W_{k\times k} = \begin{cases} L_{i,j}, & (i > j) \\ 0, & (i \le j) \end{cases} \tag{5}$$

$$L_{i,j} = \frac{com_{i,j}}{cof_{i,j}} \tag{6}$$

where $L_{i,j}$ is connection strength of two labels i and j in the data set, $com_{i,j}$ is frequency of simultaneous occurrence of labels i and j, and $cof_{i,j}$ is frequency of any one occurrence of labels i and j. Semantic association strength is shown in Figure 3. The color from blue to red indicates that association strength is from weak to strong in the figure. And image semantic self-association is the strongest degree that is expressed as red.

As can be seen from Equation (4) and (5), this paper encourages inference from semantic labels that appear simultaneously in multiple images. Then the sematic labels are inferred from the strongest association. According to the semantic label association relationship and its corresponding semantic label, the proportion of the semantic label can be obtained.

In order to fully extract the features of each candidate region in the candidate region library, the paper adopts CNN to extract features. And the CNN network structure

Table 1: Network structure of CNN.

structure	input	operate			output
		convolution	Nonlinear mapping	Pooling	
Conv1	27×27×3	64×3×3 ×3 stride 1	ReLU		27×27×64
Conv2	27×27×64	256×5×5×64 stride 1	ReLU	2×2 pool	13×13×256
Conv3	13×13×256	256×3×3×256 stride 1	ReLU		13×13×256
Conv4	13×13×256	256×3×3×256 stride 1	ReLU		13×13×256
Conv5	13×13×256	512×3×3×256 stride 1	ReLU	2×2 pool	6×6×512
Fc6	6×6×512	4096	ReLU		4096
Fc7	4096	4096	ReLU		4096
Fc8	1000	1000			1000

is shown in Table 1. It consists of five convolutional layers (cov1~cov5) and three fully connected layers (fc6~fc8). In this paper, five convolutional layers and two full convolutional layers are used for learning. After cov2 and cov5 convolution operations, the max pooling method is used to operate, and finally 4096-dimensional feature vector of fc7 layer is used as an image feature vector output. For CNN input data preparation phase, the sample patch uses an image block of 27×27 pixels in size, and the sampling center is candidate region center. For CNN output, feature extraction model chooses directly to use 4096-dimensional feature vector of fc7 layer as visual feature of candidate region.

According to the feature vector of the candidate region, we construct an information table $IS = \langle U, C, V, f \rangle$, where the sample set of candidate regions $U = \{x_1, x_2, \ldots, x_{k'}\}$, which is described by a series of features. Where k' is the number of candidate regions in the candidate region library, C is feature set describing U, V is a set of attribute values, and f is information function. And the neighborhood particles $\delta(x_i)$ of each candidate region are constructed:

$$F'_{x_i} = \frac{\delta(x_i)}{U} \tag{7}$$

$$\delta(x_i) = \left\{ x_j \mid x_j \in U, \Delta\left(x_i, x_j\right) \leq \delta \right\} \tag{8}$$

$$\Delta\left(x_i, x_j\right) = \left(\sum_{i=1}^{m} \left| f\left(x_I, C\right) - f\left(x_J, C\right) \right|^P \right)^{1/P} \tag{9}$$

where $\delta \geq 0$; $\delta(x_i)$ is called generated neighborhood information particle, which determines the size of the neighborhood particle. P is the norm, Δ is called the similarity measure, and m is dimension of attribute matrix V. According to nature of metric, it can be known that

$$\delta(x_i) \neq \varnothing \tag{10}$$

$$\sum_{i=1}^{k} \delta(x_i) = U \tag{11}$$

If the size of the neighborhood particle is fixed, the neighborhood particle with the most similar candidate regions can be obtained. And f_i can determine the size of the neighborhood particle. Then the paper can get neighborhood thresholds $\delta = \{\delta_1, \delta_2, \ldots, \delta_k\}$ and get the smallest threshold $\delta_v = \min(\delta)$. Therefore, the candidate regions corresponding to the most similar neighborhood particles with the minimum threshold are determined.

Finally, the paper obtains the candidate region corresponding to the semantic label to be inferred and its neighboring particles and completes the inference of the semantic label. After that, the inferred candidate region is removed from the candidate region library, iterating until all inferences of the rest of semantic labels are completed.

3.3. Candidate Regions Classification Using ELM. After completing the inference of all semantic labels, the paper selects the ELM to learn the inferred candidate regions. The main reason is that ELM is a new type of fast machine learning algorithm, which is a supervised algorithm based on single hidden layer feed forward neural network [29]. In addition, ELM trains parameters without iterating, which can improve algorithm efficiency.

First, the ELM is trained based on candidate regions with semantic labels and get trained ELM to classify in the training stage. And the candidate region is still used as the basic processing unit of semantic label prediction. The reason is that the candidate region is well close to the boundary of the target and is not susceptible to noise. In order to obtain the candidate regions corresponding to the test images, the paper first performs superpixel segmentation and superpixel merging to generate candidate regions under the same parameter setting and implementation steps. Then 4096-dimensional features are extracted on the candidate regions corresponding to the test images to ensure the consistency between the testing stage and the training stage.

After that, given an image candidate region $x_i = \{x_1, x_2, \ldots, x_{l'}\}$ in the ELM testing stage, l' is the number of the test candidate regions. The candidate region is directly used as the input of the ELM; then the semantic label is predicted by the ELM. The specific steps of the ELM classification algorithm are shown in Algorithm 2.

4. Experiment

4.1. Dataset and Evaluation. The performance of our algorithm was evaluated on the MSRC [30] dataset, which has 591 images, including natural scenes (such as trees),

Input: Given N training samples (x_i, y_i), $i = 1, 2, \ldots, N$; The number of semantic label categories k; Activation function $g(x)$; The number of hidden layer nodes is l l, Test sample x'.
Output: Predicted result y'.
Step 1. Initialize the weight and bias between the input layer and the hidden layer, Randomly set the value of w and b, given the value of l.
Step 2. Select the activation function of the hidden layer $g(x)$ and calculating the output matrix H.
Step 3. Calculate the output weight of the network β: $\beta = H^T T$ (where H^T is the transpose of H).
Step 4. The output weights of the test samples x': $O_i = H(w_1, \ldots, w_l, x', b_1, \ldots, b_l)\widehat{\beta}$.
Step 5. the output of the predicted result y': $y' = label(x') = \arg\max(O_i)$, $(1 \le i \le k)$.

Algorithm 2: ELM classification algorithm.

structured scenes (such as buildings and roads), and other structures scenes. The dataset provides pixel-level annotation semantic images, and all images corresponding to pixel-level annotations maps are 213×320 pixels in size. And the scene contains a total of 23 semantic categories of objects. The same rules are followed in use of dataset, ignoring the classes of the horse and mountain image type. This article uses 276 images for training and 256 images for testing.

In addition, our method is also evaluated on the PASCAL VOC 2012 segmentation benchmark dataset [31], which is one of the most widely used benchmark datasets for semantic segmentation. It contains one background category and 20 object categories. It consists of three parts: training set (1464 images), validation set (1449 images), and test set (1456 images). In our experiments, our work is also based on the training images (10582 images) amplified by Harry Harlan et al. [32] as a training set, which provides image-level labels for training.

In this paper, evaluation index selects pixel accuracy (PA), mean pixel accuracy (MPA), and mean intersection over union (mIoU). Calculation formula is as follows:

$$PA = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{12}$$

$$MPA = \frac{1}{n_c l}\sum_i \frac{n_{ii}}{t_i} \tag{13}$$

$$mIoU = \frac{1}{n_c l}\sum_i \frac{n_{ii}}{\left(t_i + \sum_j n_{ji} - n_{ii}\right)} \tag{14}$$

where $n_c l$ is the number of categories included in true value, n_{ji} is the pixel of category i divided into category j, and t_i is the total number of pixels of category i in ground truth.

4.2. Parameter Settings. The parameter setting of CNN model is given as follows. The learning rate was set to 0.001, and the performance of three CNN visual features in image clustering is analyzed and compared. The last 3 fully connected extracted visual characteristics of candidate regions, whose outputs are 4096, 4096, and 1000, respectively, are considered feature representations of image. Figure 4 shows comparison of three visual features on MSRC dataset. It can be seen that visual features are selected as output of fc7 layer for image clustering, whose precision is the highest.

The parameter setting of ELM algorithm is given as follows. When designing ELM, the cross-validation method

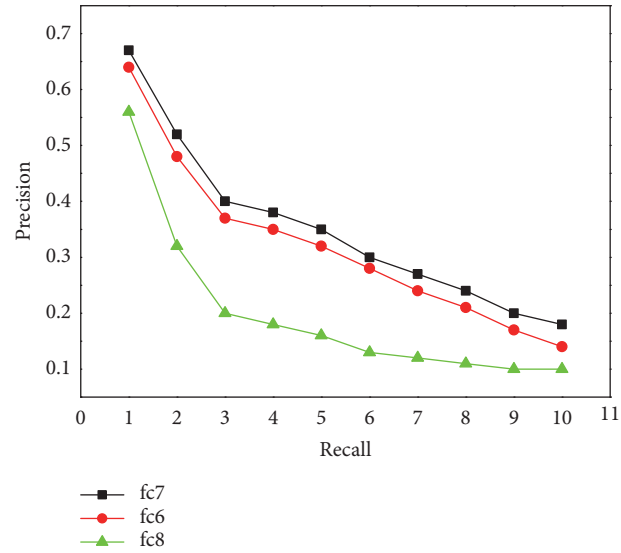


Figure 4: Comparison of recall-precision.

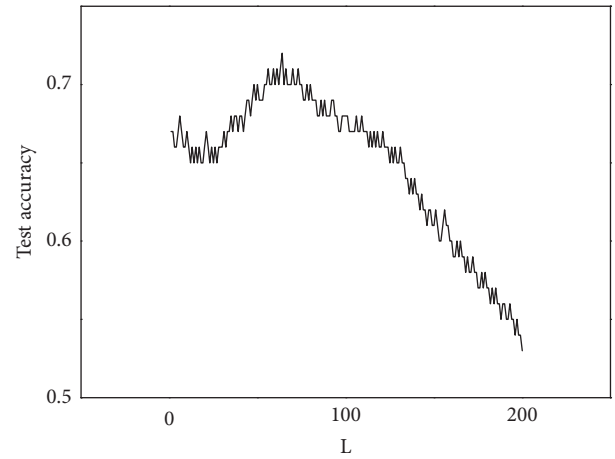


Figure 5: Relationship between L and test accuracy.

is generally used to determine optimal hidden layer node number L within preset range of K value. The simulation is performed on MSRC-21 data. It is assumed that L is increasing from 1 to 200, and classification accuracy of test set is sequentially obtained as shown in Figure 5. It can be seen from Figure 5 that when L value reaches 64, test accuracy is the highest. However, with L value continuing to increase,

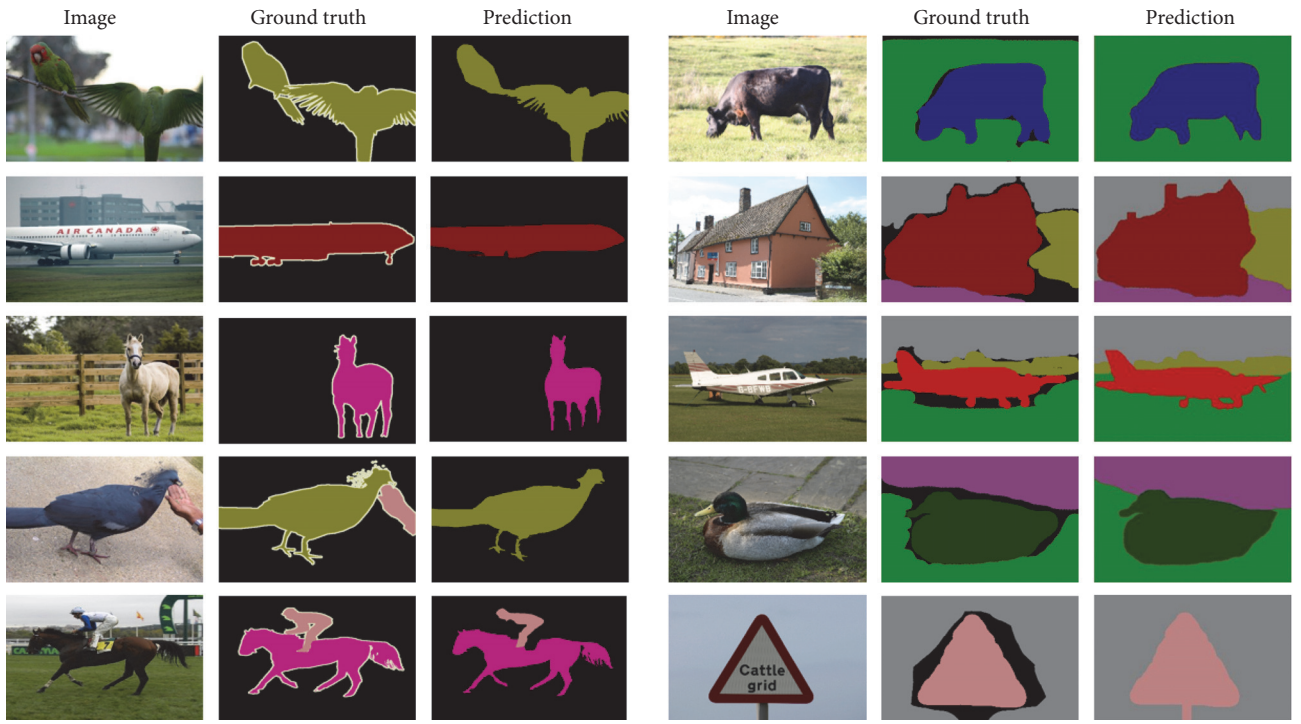


Figure 6: Examples of predicted segmentations (the left are the examples of MSRC and the right are the examples of PASCAL VOC 2012 dataset).

the measurement accuracy of ELM is generally decreasing. So when $60 \leq L \leq 68$, ELM has a good test accuracy.

4.3. Experimental Results. In order to evaluate the performance of the proposed weakly supervised image semantic segmentation method, the experiments were compared with the current weakly supervised image semantic segmentation algorithm on the MSRC-21 dataset and PASCAL VOC 2012 dataset. These comparison algorithms include STC [19], AE [22], SR [18], MIM [17], MIL+ILP+SP-sppxly [21], and CCNN [20], and these weakly supervised image semantic segmentation comparison algorithms are based on image-level labels.

First, the IoU of per-image label and the average IoU (mIoU) of all image labels are as in Tables 2 and 3, respectively, for the proposed method and the current weakly supervised image semantic segmentation algorithm on the MSRC-21 dataset and the PASCAL VOC 2012 dataset. And each column represents different algorithm accuracy of each semantic class on MSRC-21 and PASCAL VOC 2012 dataset, and the last column is average accuracy of all classes. The bold values in the table represent the best segmentation performance.

As shown in Tables 2 and 3, the proposed algorithm obtains comparable and competitive results on the IoU of per-image label and the average IoU (mIoU) of all semantic labels compared with the existing image-level labels weakly supervised image semantic segmentation algorithm method. Although the IoU on some semantic classes is lower than the compared algorithm on the MSCR and the PASCAL VOC 2012 validation set, the proposed algorithm achieves the best segmentation performance on the mIoU. In addition, the segmentation accuracy for the weakly supervised image semantic segmentation algorithm on the MSRC dataset is significantly higher than that of the PASCAL VOC 2012 dataset. The reason is that the images on the PASCAL VOC 2012 dataset contain more complex objects and backgrounds than the images on the MSRC dataset. Although many weakly supervised image semantic segmentation algorithms have been proposed, the segmentation accuracy of each semantic class on the entire dataset still has a relatively large room for improvement.

Then, in order to more intuitively display the segmentation performance of the proposed algorithm, some qualitative segmentation examples of MSRC and PASCAL VOC 2012 dataset are given. The specific segmentation results are shown in Figure 6.

As shown in Figure 6, the weakly supervised deep semantic segmentation using CNN and ELM with semantic candidate regions can achieve better segmentation performance. Moreover, the segmentation result based on the candidate region level can retain the edge information of the object in the image. However, the proposed method relies on semantic label inference and classifier learning at the candidate region level for an object that contains multiple regions with large contrast, which may be misclassified.

5. Conclusions

In this paper, a weakly supervised semantic segmentation method using ELM with semantic candidate regions is

Table 2: Results on MSRC (mIoU in %) for weakly-supervised semantic segmentation with per-image labels.

Methods	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	average
MIM [17]	43	70	59	**84**	**75**	56	65	57	**70**	60	37	56	33	**68**	42	56	**65**	59	57	45	**22**	56
SR [18]	53	63	68	74	61	67	64	**65**	52	64	**60**	47	53	60	58	**67**	57	47	62	57	13	58
STC [19]	**60**	65	**73**	65	57	54	80	47	65	73	53	**68**	61	43	60	65	62	56	**67**	**59**	11	59
Ours	53	**80**	65	81	63	**76**	70	59	68	**75**	48	60	**70**	56	**65**	49	58	**60**	48	**59**	14	**61**

Table 3: Results on PASCAL VOC 2012 (mIoU in %) for weakly-supervised semantic segmentation with only per-image labels.

PASCAL VOC 2012	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	Dining table	dog	horse	motorbike	person	Potted plant	sheep	sofa	train	tv/montir	average
CCNN [20]	69	26	18	25	20	36	47	47	48	16	38	21	44	35	46	41	30	36	22	39	37	35.3
MIL+ILP+SP-sppxly [21]	77	37	18	25	28	32	42	48	51	13	46	15	51	44	39	38	28	44	20	38	35	36.6
STC [19]	82	63	26	62	28	38	**67**	63	**75**	22	53	28	**66**	**58**	62	**53**	33	63	32	45	45	50.7
AE [22]	78	**72**	**29**	**64**	40	**58**	58	54	63	10	61	**36**	62	56	63	43	37	**65**	32	**50**	39	50.9
Ours	**84**	68	26	58	**47**	41	57	**67**	74	**23**	**73**	26	53	57	**74**	38	**43**	63	**37**	40	**48**	**52.2**

proposed. By merging superpixels into candidate regions instead of using a large number of superpixels in an image, the semantic associated relationship and neighborhood rough set are effectively combined to solve the difficulty of mapping from semantic labels into image objects. The image semantic labels quantity information is used as a condition to terminate superpixel merging, which avoids problem of manually set parameters and hence helps to solve the problem of nonadjacent multiple instances. The candidate regions are classified based on neighborhood rough set, where the candidate regions are inferred by using semantic associated relationship. As a result, more reliable candidate region semantic labels can be obtained to improve the classification accuracy. Future works can be extended to combine saliency detection [33, 34] and heuristic optimization in a data fusion framework [35–38].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

The authors would like to express their gratitude for the support from the National Natural Science Foundation of China (61503271; 61603267), Shanxi Scholarship Council of China (2015-045; 2016-044), 100 People Talents Programme of Shanxi, Shanxi Natural Science Foundation of China (201801D121144), and Shanxi Natural Science Foundation of China (201801D221190).

References

[1] Z. Shi and Y. Yang, "Weakly-supervised image annotation and segmentation with objects and attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2525–2538, 2017.

[2] J. Gao, Z. Xie, J. Zhang et al., "Image semantic analysis and understanding: a review," *Pattern recognition and artificial intelligence*, vol. 2, no. 23, pp. 191–202, 2010.

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

[4] 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 2015*, pp. 3431–3440, Boston, Mass, USA, June 2015.

[5] G. Papandreou, L. Chen, K. P. Murphy et al., "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2015)*, pp. 1742–1750, Santiago, Chile, December 2015.

[6] D. Lin, J. Dai, J. Jia et al., "Scribblesup: scribble-supervised convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 3159–3167, Las Vegas, Nev, USA, July 2016.

[7] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, and X. Gao, "Learning from weak and noisy labels for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 486–500, 2017.

[8] A. Vezhnevets and J. M. Buhmann, "Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning," in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2010*, pp. 3249–3256, San Francisco, Calif, USA, June 2010.

[9] Y. Xi, J. Zheng, X. Li, X. Xu, J. Ren, and G. Xie, "SR-POD: sample rotation based on principal-axis orientation distribution for data augmentation in deep object detection," *Cognitive Systems Research*, vol. 52, pp. 144–154, 2018.

[10] Y. Yan, J. Ren, G. Sun et al., "Unsupervised image saliency detection with Gestalt-laws guided optimization and visual attention based refinement," *Pattern Recognition*, vol. 79, pp. 65–78, 2018.

[11] G. Ding, J. Zhou, Y. Guo et al., "Large-scale image retrieval with sparse embedded hashing," *Neurocomputing*, vol. 257, pp. 24–36, 2017.

[12] G. Wu, J. Han, Z. Lin et al., "Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning," *IEEE Transactions on Industrial Electronics*, pp. 1-1, 2018.

[13] G. Wu, J. Han, Y. Guo et al., "Unsupervised deep video hashing via balanced code for large-scale video retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2019.

[14] Y. Guo, G. Ding, J. Han, and Y. Gao, "Zero-shot learning with transferred samples," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3277–3290, 2017.

[15] Y. Wei, H. Xiao, H. Shi et al., "Revisiting dilated convolution: a simple approach for weakly- and semi-supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2018*, pp. 7268–7277, Salt Lake City, Utah, USA, June 2018.

[16] Y. Liu, J. Liu, Z. Li et al., "Weakly-supervised dual clustering for image semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2075–2082, Portland, Ore, USA, June 2013.

[17] A. Vezhnevets, V. Ferrari, and J. M. Buhmann, "Weakly supervised semantic segmentation with a multi-image model," in *Proceedings of the IEEE International Conference on Computer Vision, ICCV 2011*, pp. 643–650, Barcelona, Spain, November 2011.

[18] K. Zhang, W. Zhang, Y. Zheng et al., "Sparse reconstruction for weakly supervised semantic segmentation," in *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2013*, pp. 1889–1895, Beijing, China, August 2013.

[19] Y. Wei, X. Liang, Y. Chen et al., "STC: a simple to complex framework for weakly-supervised semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2314–2320, 2017.

[20] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation,"

in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 1796–1804, Chile, December 2015.

[21] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1713–1721, Boston, Mass, USA, June 2015.

[22] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, "Object region mining with adversarial erasing: a simple classification to semantic segmentation approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Hawaii, USA, July 2017.

[23] Y. Wei, X. Liang, Y. Chen et al., "Learning to segment with image-level annotations," *Pattern Recognition*, vol. 59, pp. 234–244, 2016.

[24] A. Vezhnevets, V. Ferrari, and J. M. Buhmann, "Weakly supervised structured output learning for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012*, pp. 845–852, Providence, RI, USA, June 2012.

[25] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 1717–1724, IEEE, Columbus, Ohio, USA, June 2014.

[26] S. Luan, C. Chen, B. Zhang, J. Han, and J. Liu, "Gabor convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4357–4366, 2018.

[27] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, 2012.

[28] Q. Hu, "Numerical attribute reduction based on neighborhood granulation and rough approximation," *Journal of Software*, vol. 19, no. 3, pp. 640–649, 2008.

[29] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference*, vol. 2, pp. 985–990, Budapest, Hungary, March 2004.

[30] J. Shotton, J. Winn, C. Rother et al., "Textonboost: joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proceedings of the European Conference on Computer Vision, ECCV 2006*, vol. 3951, Graz, Austria, May 2006.

[31] M. Everingham, L. van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[32] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proceedings of the IEEE International Conference on Computer Vision, ICCV 2011*, pp. 991–998, Barcelona, Spain, November 2011.

[33] Z. Wang, J. Ren, D. Zhang, M. Sun, and J. Jiang, "A deep-learning based feature hybrid framework for spatiotemporal saliency detection inside videos," *Neurocomputing*, vol. 287, pp. 68–83, 2018.

[34] J. Han, D. Zhang, X. Hu, L. Guo, J. Ren, and F. Wu, "Background prior based salient object detection via deep reconstruction residual," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 8, pp. 1309–1321, 2015.

[35] Y. Yan, J. Ren, H. Zhao et al., "Cognitive fusion of thermal and visible imagery for effective detection and tracking of pedestrians in videos," *Cognitive Computation*, vol. 10, no. 1, pp. 94–104, 2018.

[36] A. Zhang, G. Sun, J. Ren et al., "A dynamic neighborhood learning-based gravitational search algorithm," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 436–447, 2018.

[37] J. Tschannerl, J. Ren, P. Yuen et al., "MIMR-DGSA: unsupervised hyperspectral band selection based on information theory and a modified discrete gravitational search algorithm," *Information Fusion*, vol. 51, pp. 189–200, 2019.

[38] F. Cao, Z. Yang, and J. Ren, "Local block multilayer sparse extreme learning machine for effective feature extraction and classification of hyperspectral images," *IEEE Trans. Geoscience and Remote Sensing*, 2019.

Hindawi
Complexity
Volume 2019, Article ID 8176489, 11 pages
https://doi.org/10.1155/2019/8176489

Research Article

Passive Initialization Method Based on Motion Characteristics for Monocular SLAM

Yu Yang, Jing Xiong, Xiaoyu She, Chang Liu, ChengWei Yang, and Jie Li

School of Mechatronical Engineering, Beijing Institute of Technology, 5th South Zhongguancun Street, Beijing, China

Correspondence should be addressed to ChengWei Yang; yangchengwei2009@126.com

Received 12 October 2018; Accepted 23 January 2019; Published 5 February 2019

Guest Editor: Jungong Han

Visual SLAM techniques have proven to be effective methods for estimating robust position and attitude in the field of robotics. However, current monocular SLAM algorithms cannot guarantee timeliness of system startup due to the problematic initialization time and the low success rates. This paper introduces a rectilinear platform motion hypothesis and thereby converts the estimation problem into a verification problem to achieve fast monocular SLAM initialization. The proposed method is simulation tested on a fixed-wing UAV. Tests show that the proposed method can produce faster initialization of visual SLAM and that the advantages are more profound on systems with sparse image features.

1. Introduction

In recent years, with the application of Graph-based Optimization [1] and Bundle Adjustment (BA) [2] in Visual Simultaneous Localization and Mapping (vSLAM) and the emergence of excellent open-source libraries [3, 4], vSLAM systems are increasingly used in autonomous motion platforms. Exceptional open-source vSLAM systems also help popularize vSLAM techniques. Presently, vSLAM systems have been applied to UAV autonomous navigation [5, 6] and obstacle avoidance [7, 8] problems in GPS-denied environments. However, current vSLAM systems usually take a long time to initialize [9], posing difficulties for real-world engineering problems.

Currently, there exist many powerful vSLAM methods, such as PTAM [10], ORB-SLAM [11, 12] SVO [13, 14], and semidirect LSD-SLAM [15] and DSO [16]. Their initialization methods are summarized in Table 1.

Generally speaking, feature-based vSLAM techniques rely on epipolar geometry constraints or homography constraints [17]; they obtain the **R** and **t** corresponding to the minimum Reprojection Error with RANSAC or Least Squares methods. As for direct methods, they are usually initialized through randomized approaches, as exact point-to-point mappings cannot be obtained directly, leading to noncomputable **R** and **t**.

As can be seen from the above method, most of the classical monocular vision SLAM method does not consider the motion characteristics of the platform during the initialization phase. However, the basic equations of the SLAM system are composed of equations of motion and observation equations. Most of the current research focuses on the observation equations. This paper believes that the reasonable introduction of motion hypothesis can effectively improve the robustness of observations, especially in the initialization phase.

PTAM's initialization works with the hypothesis that captured images are mainly composed of flat surfaces; initial camera motion **R** and **t** are then computed with homography matrix (**H**) accordingly. ORB-SLAM algorithms are effective extensions of PTAM that compute computing essential matrix (**E**) and H simultaneously; the final initialization method is then selected by comparing the respective scores. LSD-SLAM and DSO, as direct methods, cannot compute **R** and **t** through Reprojection. Therefore, they initialize through random variables. When camera motions cover enough distance, initialization will be effectuated by locking into specific depths. SVO's initialization is similar to that of PTAM, except that SVO integrates an additional assumption that the motion direction is perpendicular to the photographed plane, as SVO is originally designed for rotor UAV use.

Table 1: Summary of initialization methods.

	Method	Initialization	Main Approach	Category
1	PTAM	H	Feature-based	SLAM
2	ORB-SLAM	H+E	Feature-based	SLAM
3	LSD-SLAM	Rand	Direct	SLAM
4	DSO	Rand	Direct	VO
5	SVO	H	Feature-based+Direct	VO

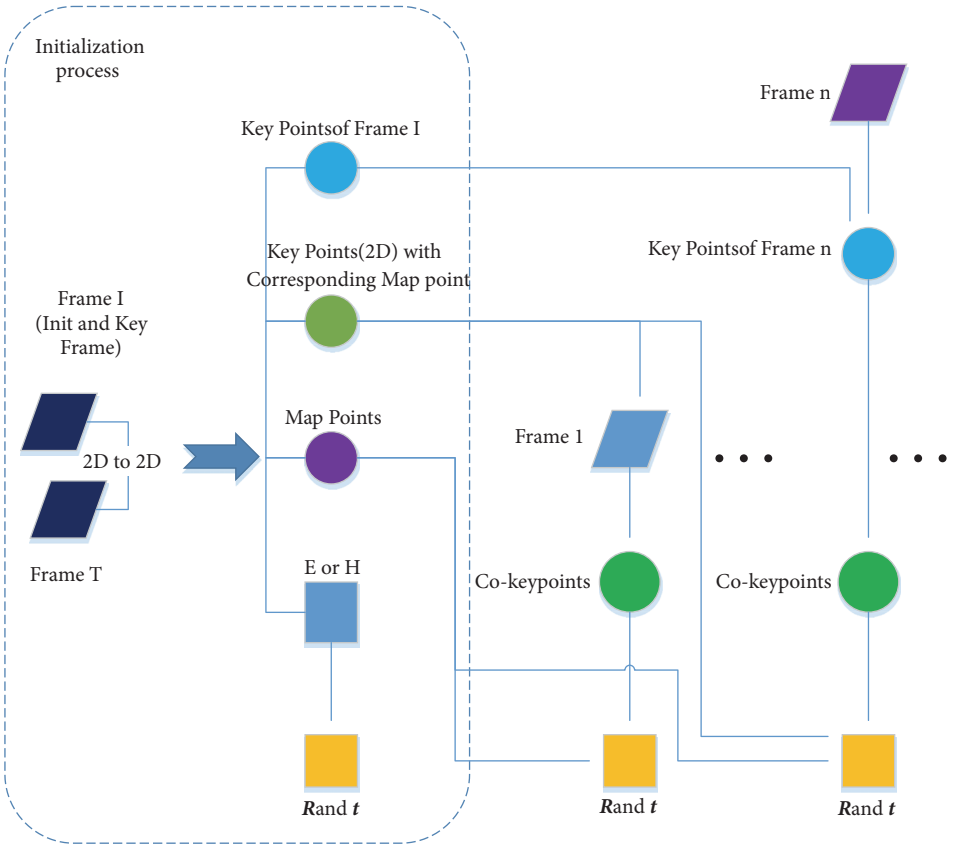


Figure 1: Classic feature-based initialization workflows.

These five classic methods are renowned in the field of monocular SLAM/VO, each possessing unique strengths. They have been successfully applied in their respective environments with satisfactory performance. The initialization workflows of the feature-based algorithms are summarized in Figure 1.

Theoretically, the initialization process depicted herein can initialize any movement except pure rotation. Firstly, corresponding points from separate frames are identified through feature-based or optical flow methods. These point mappings are then utilized along with monocular-camera imaging characteristics in computing **H** or **E** under the epipolar geometry frame. **H** or **E** is then decomposed to produce **R**, t, and finally the initial map points with the additional assumption that the mapped points contain no actual movement. This concludes the traditional initialization process, where the frames can be adjacent or nonadjacent, and the decomposition utilizes RANSAC, Eight-Point Method, or Bundle Adjustment. Subsequent processes will use the initial **R**, **t** and map points (3D) for the chain processes maintaining the monocular SLAM system. Due to the scale uncertainty of the monocular visual SLAM system, no initialization method can produce real-world distance of the map points; the dimensionless depths are provided instead. The initial map points (3D) play an important role for subsequent frames. The indirect 3Dto2D correspondences between pixel points and map points (3D), together with the geocalculated DLT/P3P [18]/EPnP [19]/UPnP [20] or the optimized BA, are used to determine the subsequent frames' positions and orientations relative to the preceding key frame. Frame I is an initialization frame as well as a key frame. As the camera moves on, the number of indirect 3Dto2D correspondences that can be established will gradually decrease, leading to probable failure of the aforementioned chain processes. It is then necessary to consider inserting new key frames to replenish the map points (3D) needed for the chain processes. Complete SLAM

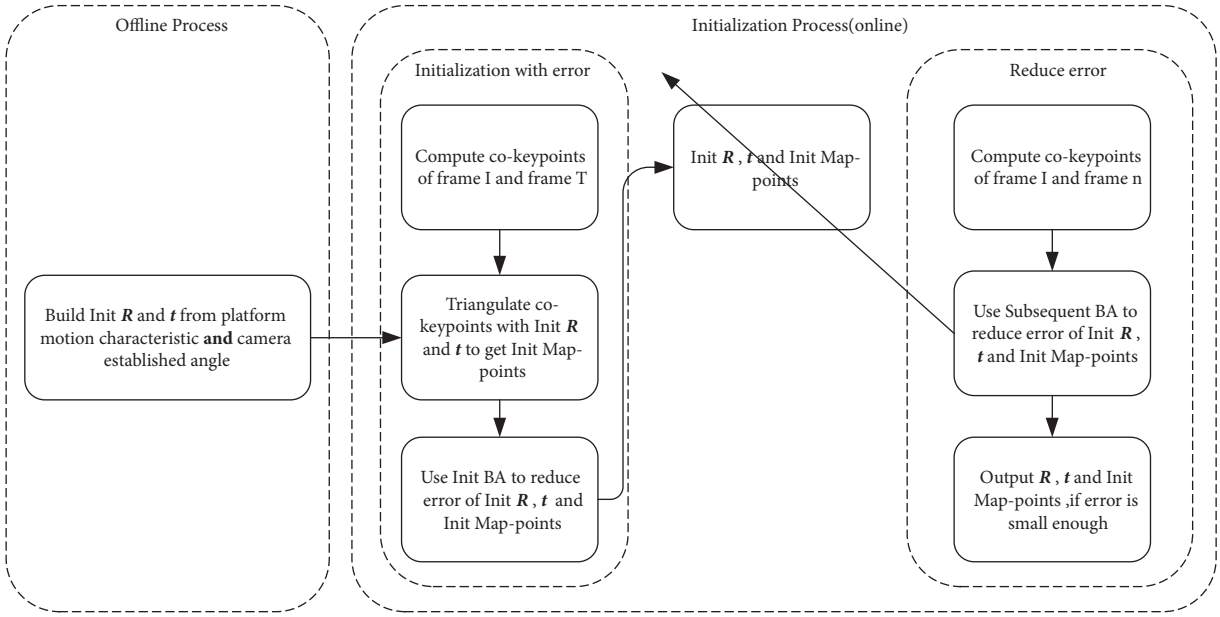


Figure 2: Proposed method flowchart.

algorithms also involve another important process termed loop closure, which will not be discussed further, as it is not much related to the present paper.

It can be seen from above that the **E** or **H** is obtained from point correspondences is the initial enabler of the entire monocular SLAM system. However, when point correspondences are insufficient or too inaccurate, the obtained **E** or **H** may contain large errors, affecting the accuracy of the map points (3D), and thus compromising the subsequent processes. Current methods are based on limited errors of **R** and **t**. Therefore, the actual implementations contain much strict computation on **E** or **H**, leading to low success rates in many cases. When the monocular SLAM systems are applied to fixed-wing unmanned aerial vehicles (UAVs), the initialization success rates are even more worrying [5, 6].

In this paper, we add a generalized motion characteristic hypothesis in the initialization process to transform the solution of camera motion R and t into the error elimination problem during the initialization process. In this way, the success rate of initialization is increased. In view of the error caused by the hypothesis, this paper reduces the error by multiframe optimization method, thus improving the accuracy of the initialization process.

1.1. Contribution. Firstly, this paper proposes the platform motion characteristics, which represents the motion state of the platform in most of the time. Secondly, this paper introduces the platform motion characteristics into the initialization phase of monocular vision SLAM and avoids the solution of the essential matrix and the homography matrix by optimization. Finally, this paper uses the subsequent BA to convert the initialization from a transient process to a convergent process of several consecutive frames.

2. Monocular SLAM Initialization Method Based on Platform Motion Characteristics and Optimization

2.1. Overview. The present paper proposes a monocular SLAM initialization method based on platform motion characteristics and optimization, the flowchart of which is shown in Figure 2.

The proposed method contains an offline process and an online process. In the offline process, initial motions **R** and **t** are computed with platform motion characteristics of the camera installation mode. In the online process, firstly, a set of Frame I and Frame T are used to detect and match the feature points, and then initial map points are generated under the initial motion hypothesis, whose initial errors are eliminated by Init BA. Finally, the subsequent BA is performed with the matched feature points of Frame I and Frame n, further reducing the errors of **R**, **t**, and the map points. The initialization is considered to have succeeded when the errors converge.

2.2. Initial Motion Hypothesis Combining Platform Motion Characteristics and Camera Installation. The proposed method utilizes an initial motion hypothesis to initialize the system. Cars on ground generally run along straight lines, while aerial vehicles usually fly at a fixed angle of attack. This is a very broad description, as ground vehicles may turn, and aircraft may roll. General motion characteristics can be expressed with

Require: Frame I:Initialization Frame
Frame T:Target Frame
T_{mc}:Camera Motion Hypothesis
Ensure: Initialized Map Points $\mathbf{X}_{init}$,$\mathbf{R}_{init}$,$\mathbf{t}_{init}$
(1) Perform feature point matching for Frame I and Frame T, resulting in matched points $\mathbf{x_I}$ and $\mathbf{x_T}$.
(2) Triangulate $\mathbf{x_I}$ and $\mathbf{x_T}$ using hypothesis T_{mc} to generate map points $\mathbf{X}_{init}$
(3) Optimize $\mathbf{X}_{init}$, $\mathbf{R}_{init}$, $\mathbf{t}_{init}$ with Init BA and update accordingly.
(4) **return** $\mathbf{X}_{init}$ $\mathbf{R}_{init}$ $\mathbf{t}_{init}$

Algorithm 1: Monocular SLAM initialization with initial motion hypothesis.

$$T_m = \begin{bmatrix} R_m & t_m \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_m \cos\phi_m & \sin\psi_m \sin\theta_m \cos\phi_m - \cos\psi_m \sin\phi_m & \cos\psi_m \sin\theta_m \cos\phi_m + \sin\psi_m \sin\phi_m & x_m \\ \cos\theta_m \sin\phi_m & \sin\psi_m \sin\theta_m \sin\phi_m + \cos\psi_m \cos\phi_m & \cos\psi_m \sin\theta_m \sin\phi_m - \sin\psi_m \cos\phi_m & y_m \\ -\sin\theta_m & \sin\phi_m \cos\theta_m & \cos\psi_m \cos\theta_m & z_m \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Equation (1) is a 6-DOF rectilinear description of any motion platform. For the monocular vSLAM initialization, the rectilinear hypothesis of the platform motion needs to be expressed in the coordinate system of the camera. The require conversion is derived from the mounting characteristics of the camera and is expressed with

$$T_{mc} = T_m T_{vc} \tag{2}$$

T_{vc} in (2) is the transformation matrix of the camera coordinate system with respect to the platform coordinate system. This matrix can be obtained from the camera installation characteristic. A general form of T_{vc} is given in

$$T_{vc} = \begin{bmatrix} R_{vc} & t_{vc} \\ 0 & 1 \end{bmatrix} \tag{3}$$

Substitute T_m and T_vc in (2) by (1) and (3), respectively, the camera motion model under the aforementioned rectilinear hypothesis is obtained.

$$T_{mc} = \begin{bmatrix} R_m R_{vc} & R_m t_{vc} + t_m \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{mc} & t_{mc} \\ 0 & 1 \end{bmatrix} \tag{4}$$

2.3. Monocular SLAM Initialization with Initial Motion Hypothesis. The established camera motion model is then used in the implementation of the passive initialization process. In conventional initialization methods, **R** and **t** are obtained from the decomposing of the strictly computed **E** or **H**. The proposed method replaces the decomposition results **R** and **t** with $\mathbf{R}_{mc}$ and $\mathbf{t}_{mc}$ and thereby triangulates the feature points to obtain the map points and finally utilizes Init BA to reduce the errors (Algorithm 1).

The Init BA mentioned above minimalizes the Reprojection Error for the feature points of Frame I and Frame T. Let $x_1^n \in \mathbb{R}^2$ and $x_2^n \in \mathbb{R}^2$ be the coordinates of the matched feature points in Frame I and Frame T, respectively. The previously established $\mathbf{T}_{cm}$ is used to triangulate x_1^n and x_2^n to obtain map points $X_m^n \in \mathbb{R}^3$.

$$\begin{bmatrix} -\mathbf{I} & x_1 & 0 & 0 \\ 0 & 0 & -\mathbf{I} & x_2 \end{bmatrix} \begin{pmatrix} K\begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \\ K\begin{bmatrix} \mathbf{R_{mc}} & \mathbf{t_{mc}} \end{bmatrix} \end{pmatrix} \begin{pmatrix} X_m \\ 1 \end{pmatrix} = 0 \tag{5}$$

$\mathbf{T}_{cm}$ introduces the rectilinear hypothesis; therefore, it is necessary to restrain the errors in the map points' coordinates.

$$\{\mathbf{X}_{init}, \mathbf{R}_{init}, \mathbf{t}_{init}\} = \underset{\mathbf{X}_m, \mathbf{R}_{cm}, \mathbf{t}_{cm}}{\arg\min} \sum_{j=1}^{N} \left(\left\| \frac{1}{\lambda_1^j} K X_m^j - \left[x_1^j, 1\right]^T \right\|^2 + \left| \frac{1}{\lambda_2^j} K \left(R_{cm} X_m^j + t_{cm}\right) - \left[x_2^j, 1\right]^T \right\|^2 \right) \tag{6}$$

The map points can be optimized once by (6), which reduces the error caused by the hypothetical model. Due to the quality of feature point matching, only Init BA cannot make the error of X, R, and t small enough, so the method introduces the subsequent BA to further reduce the error.

Equation (6) describes the Init BA optimization of the map points. Due to the quality of the matched feature points, Init BA alone cannot reduce the errors of **X**, **R**, and **t** to an acceptable margin. The propose method utilizes the subsequent BA to further reduce the errors.

2.4. Error Reduction with Subsequent BA. Limited by the number and distribution of matching feature points, the errors contained in $\mathbf{X}_{init}^i$, $\mathbf{R}_{init}$, and $\mathbf{t}_{init}$ cannot be evaluated, so this paper introduces subsequent BA to achieve further error suppression and initialization accuracy evaluation. The main idea of subsequent BA is to optimize X, Rinit, and tinit with each subsequent frame and then decide whether to continue the subsequent optimization by judging its convergence (Algorithm 2).

Require: Frame I: Initialization Frame
Frame n: Subsequent Frame
$\mathbf{x_I}$: Feature points of Frame I
$\mathbf{x^1}, \mathbf{x^2} \ldots \mathbf{x^{n-1}}$: Feature points of previous frames corresponding to $\mathbf{x_I}$
$\mathbf{R}_1, \mathbf{R}_2 \ldots \mathbf{R}_{n-1}$: Optimized rotation matrices of previous frames
$\mathbf{t}_1, \mathbf{t}_2 \ldots \mathbf{t}_{n-1}$: Optimized translation vectors of previous frames
$\mathbf{X}_{init}$: Initialized Map Points
Ensure: Map points $\mathbf{X}_{init}$ $\mathbf{R}_n$ $\mathbf{t}_n$ and Initialization Evaluator v.
(1) Extract feature points of Frame n and match with $\mathbf{x_I}$ to obtain matched feature points $\mathbf{x^n}$.
(2) Optimize the Reprojection Error with Subsequent BA and thereby obtain optimized $\mathbf{X}_{init}$, $\mathbf{R}_n$, $\mathbf{t}_n$.
(3) Calculate current evaluator value v_n.
(4) **if** $v_n \approx v_{n-1}$ **then**
(5) Stop iteration, and return $\mathbf{X}_{init}$, $\mathbf{R}_n$, $\mathbf{t}_n$ as initialization results.
(6) **end if**
(7) **if** n is too big **then**
(8) Stop iteration, and report failure of initialization.
(9) **end if**

Algorithm 2: Error reduction with subsequent BA.

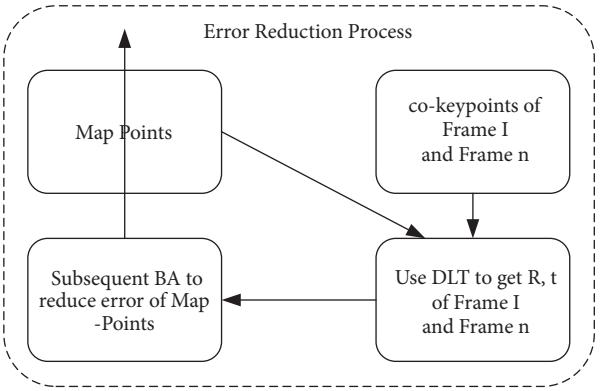


Figure 3: Error reduction process.

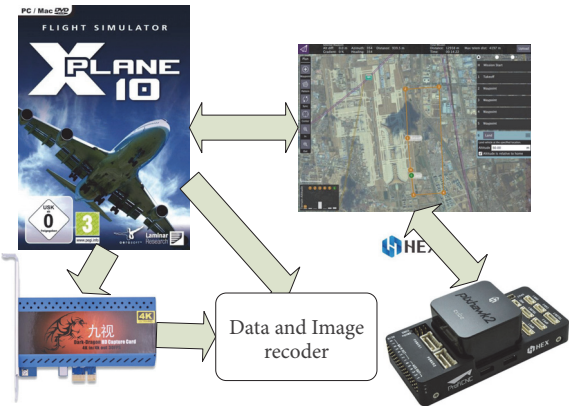


Figure 4: Simulation system.

The errors contained in $\mathbf{X}^i_{init}$, $\mathbf{R}_{init}$, and $\mathbf{t}_{init}$ cannot be evaluated through Init BA, due to the scale and distribution of the matched feature points. Subsequent BA is thus utilized for initial error evaluation and further error reduction. The main idea of subsequent BA is to optimize $\mathbf{X}^i_{init}$, $\mathbf{R}_{init}$, and $\mathbf{t}_{init}$ with each subsequent frame. Convergence evaluation is performed to determine when to stop the subsequent optimization.

For each frame of the subsequent input, the coordinate X_{init} of the map points is optimized as shown in the process of Figure 3. When it converges, the error elimination process is considered to be ended, and the subsequent BA process in the figure is as shown in (7).

For each subsequent frame, X_{init} is optimized with the error reduction process shown in Figure 3 and

$$\begin{aligned} &\left\{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l \mid \mathbf{X}^i \in \mathbf{X}_{init},\ l \in \mathbf{N}_f\right\} \\ &= \arg\min_{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l} \sum_{k \in \mathbf{N}_f} \sum_{j \in \mathbf{X}_{init}} \rho\left(E\left(k, j\right)\right) \\ &E\left(k, j\right) = \left\|\mathbf{x}^j - p\left(\mathbf{R}_k \mathbf{X}^j + \mathbf{t}_k\right)\right\|_2^2 \end{aligned} \tag{7}$$

It can be seen from (7) that the scale of optimization gets larger with continuous input of subsequent frames, which ensures to some extent the reliability of the optimized results. The optimized map points are viewed as initialization results, providing input for subsequent chain processes. The quality of initialization is evaluated with

$$\begin{aligned} v &= \overline{\mathbf{E}}_o \\ \mathbf{E_o} &= \left\{e < \overline{\mathbf{E}_{all}} \mid e \in \mathbf{E}_{all}\right\} \end{aligned} \tag{8}$$

where $\mathbf{E}_{all}$ is the sum of the Reprojection Errors of all map points in all frames participating in the optimization.

3. Simulation

3.1. Simulation System. In order to better reproduce vehicle motion characteristics, the present study builds a hardware-in-the-loop (HIL) simulation system, as illustrated in Figure 4. It consists of four parts, namely, the Xplane10 flight simulation software, the Pixhawk2 flight controller, the

Table 2: Self-evaluation performance indicators.

	Indicator Name	Unit	Alias
1	Number of Convergence Frames	frame	NCF
2	Initial Error	deg	IE
3	Convergence Error	deg	CE
4	Average Number of Convergence Frames	frame	ANCF
5	Average Initial Error	deg	AIE
6	Average Convergence Error	deg	ACE

Table 3: Comparative-evaluation performance indicators.

	Indicator Name	Unit	Alias
1	Success Rate of Initialization	percentage	SRI
2	Average Error of Initialization	deg	AEI

QGroundControl software, and the data logger. Xplane10 and QGroundControl run on PC (CPU: Intel i7-7700K 4.20GHz, graphics card: NVidia GTX 1080 8G, memory: 32GB). The Pixhawk2 controller is linked to PC via a USB port.

Xplane10 plays the most important role in the entire simulation system, providing aircraft models and simulation images. The Pixhawk2 controller performs autonomous control of the fixed-wing aircraft in Xplane10, with QGroundControl acting as the data relay. Specifically, Xplane10 sends the aircraft states to QGroundControl through local loopback UDP; QGroundControl forwards the aircraft data to Pixhawk2 via the USB port using the Mavlink protocol; Pixhawk2 sends out the control commands through the same protocols. The data logger records the uplink-downlink data through UDP and the first-person view (FPV) simulation images through the video capture card.

Data transmitted in the simulation system can be roughly classified as periodic data and sporadic data. Periodic data includes the control commands and the aircraft states. Sporadic data includes the start signal, the waypoint-planning instruction, etc. Through meticulous testing, the frequency of the periodic commands is set at 65HZ, and the image sampling frequency is set at 25HZ.

3.2. Performance Indicators. Reasonable and balanced performance indicators are needed to evaluate the initialization methods. This paper proposes two groups of performance indicators, for self-evaluation (Table 2) and comparative evaluation (Table 3), respectively.

This study holds that the convergence frame number and the initial error are key indicators to assess the proposed passive initialization algorithm. As the initial error is only affected by the aircraft state at the initial time and the rectilinear motion hypothesis, the convergence frame number is a stronger indicator of the usability of the proposed method.

The optimized map points and pose information are only usable after convergence. Since error rotation e_R is quite unintuitive, for ease of understanding, this paper decomposes e_R into e_{pitch}, e_{roll}, e_{yaw} to facilitate the evaluation of performance. The difference in length between $\mathbf{t}_{init}$ and $\mathbf{t}_T$ (true value of $\mathbf{t}_{init}$) is not considered due to the depth uncertainty of monocular vSLAM initializations; only the difference in angle between $\mathbf{t}_{init}$ and $\mathbf{t}_T$ is considered.

3.3. Test Design. In order to thoroughly test the proposed initialization method, we devise a simple self-evaluation test and an advanced comparative-evaluation test. The self-evaluation test measures the inherent capabilities of the new method, while the comparative-evaluation test runs the competing algorithms on different terrains. The test scenarios include: taxing, climbing, level flight, BTT turn, diving, and landing.

3.4. Self-Evaluation Test and Result Analysis. The test results of the algorithm in this paper are shown in Figure 6, and the convergence curve of the algorithm is given, where Figure 6(a) gives the convergence curve for initializing in the running state. It can be seen that in this state, the initial error is small, because the state of motion of the aircraft is very close to the motion assumption in the slipping state, and the error is within 1° even if the error is not eliminated. Figure 6(b) gives the convergence curve for the initialization of the aircraft at the moment of takeoff. It can be seen that there is a large error in the motion state of the aircraft and the motion assumption at this time. Due to the characteristics of the fixed-wing aircraft, it is mostly in a level flight during the cruise flight. In this state, the motion of the aircraft is similar to the motion assumption. Therefore, several special states are selected in the test, including the climbing to level flight (Figure 6(d)), level flight to BTT turn (Figure 6(e)), level flight to dive (Figure 6(f)), etc. Thus Figure 6 gives the convergence of the algorithm in each typical state during a complete flight, which does not reflect the ability of the algorithm in the whole process.

Figures 7 and 8 and Table 4 summarize the convergence-related initialization performance at different poses throughout one complete flight. Figure 7 shows the convergence time statistics of the algorithm in the whole flight process. Figure 8 indicates the error distribution of the algorithm under different thresholds. Table 4 gives the exact values of Figures 7 and 8.

3.5. Comparative-Evaluation Test and Result Analysis. This paper selects ORB-SLAM2 and DSO as the competing classical algorithms for the comparative-evaluation test. In order to better reflect their performance, this study runs the all methods on plain terrain (Figure 5) and mountainous

Table 4: Convergence statistics.

	δ_s	ANCF	AIE	ACE
1	1°	1.49	95.4%	0.83
2	0.7°	1.65	83.5%	0.58
3	0.5°	2.81	78.1%	0.41
4	0.3°	3.94	67.5%	0.25
5	0.1°	5.63	63.3%	0.08

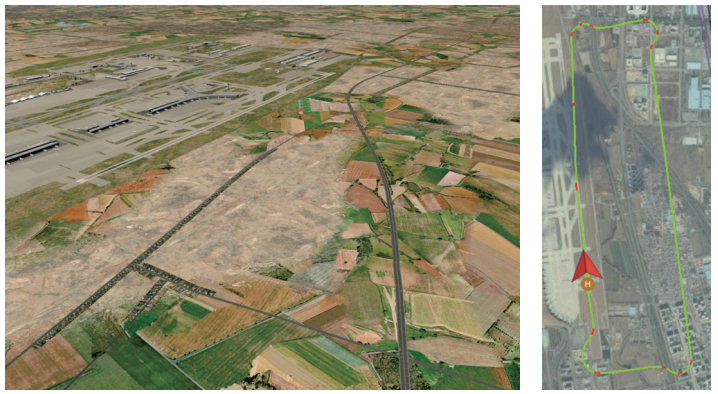


Figure 5: Simulation scenarios on plain terrain.

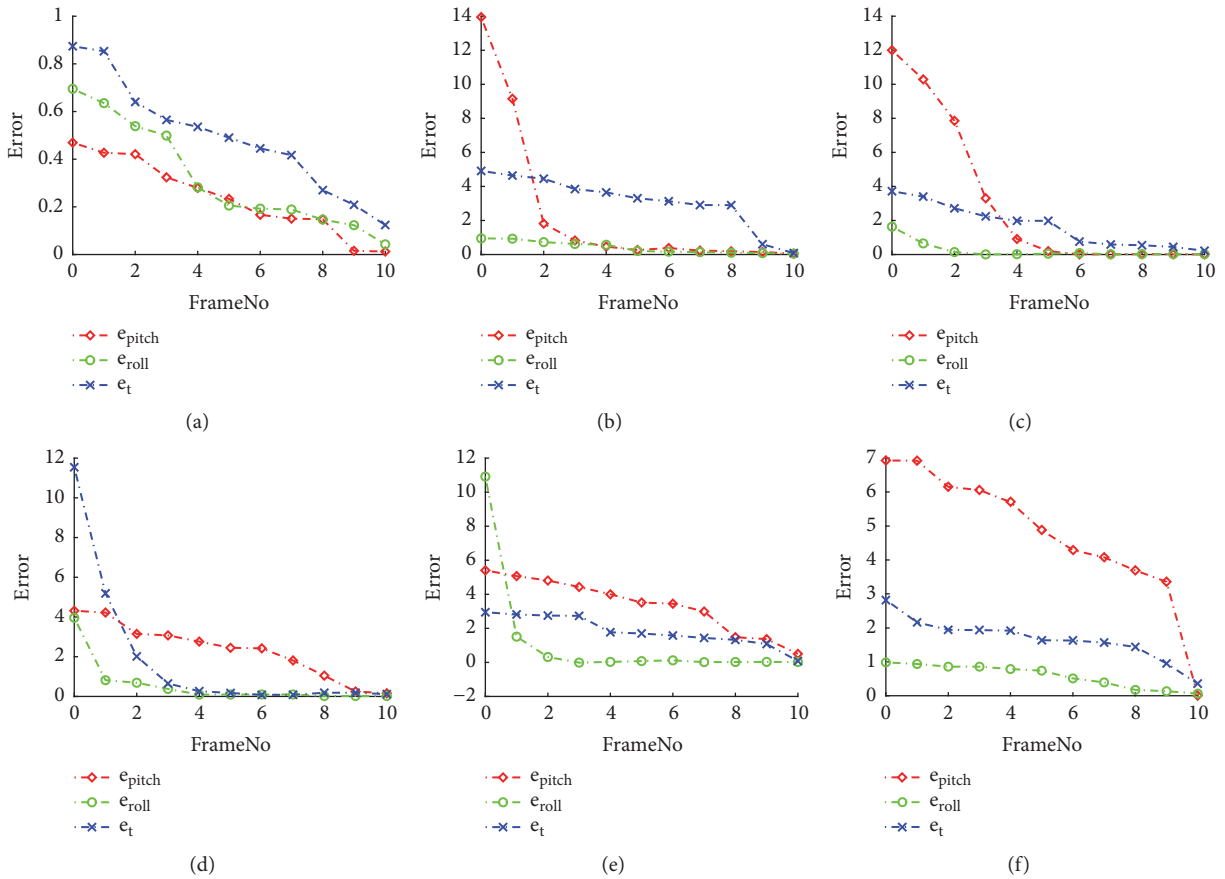


Figure 6: Error time histories.

Table 5: Initialization results under different terrains.

	Method	Terrain	SRI	AEI
1	ours (δ_s = 1)	Plain	95.4%	0.83
2	ours (δ_s = 0.7)	Plain	83.5%	0.58
3	ours (δ_s = 0.5)	Plain	78.1%	0.41
4	ours (δ_s = 0.3)	Plain	67.5%	0.25
5	ours (δ_s = 0.1)	Plain	63.3%	0.08
6	ORB-SLAM2	Plain	8.2%	1.67
7	DSO	Plain	12.2%	1.01
8	ours (δ_s = 1)	Mountainous	23.1%	0.93
9	ours (δ_s = 0.7)	Mountainous	17.1%	0.65
10	ours (δ_s = 0.5)	Mountainous	14.9%	0.44
11	ours (δ_s = 0.3)	Mountainous	11.1%	0.24
12	ours (δ_s = 0.1)	Mountainous	9.1%	0.08
13	ORB-SLAM2	Mountainous	5.4%	1.78
14	DSO	Mountainous	10.4%	1.38

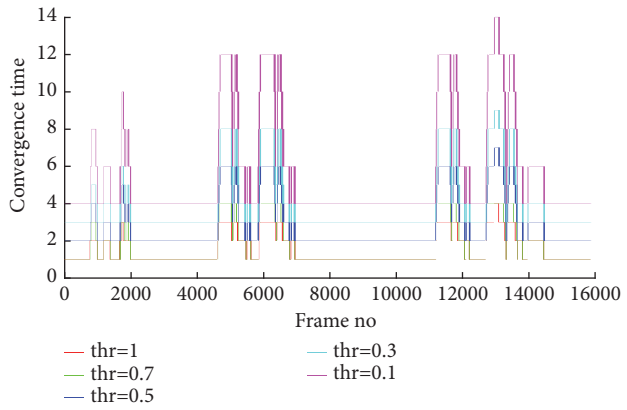


Figure 7: Convergence time histories.

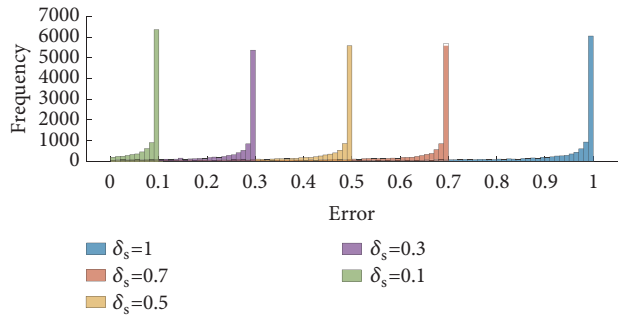


Figure 8: Converged error statistics.

terrain (Figure 9). Considering the stochastic nature of ORB-SLAM2, the study conducts five comparative-evaluation subtests on each terrain. The best subtest results are viewed as the illustrative test results.

Figure 10 gives the initialization results of the three algorithms in two terrains. δ_s in proposed method is set to 0.5 during test. It can be seen that SRI of the proposed method in both plain and hilly terrain is greater than that of ORB-SLAM2 or DSO. Considering the effect of δ_s on proposed method, SRI under different δ_s is compared, as in the Table 5

In addition to the comparative-evaluation performance indicators introduced in Table 3, this paper also compares the number of matched feature points(ANMFP) needed by ORB-SLAM2, DSO and the proposed method, respectively, to effectuate successful initialization (Table 6).

It can be seen from Table 6 that the ANMFP value of the proposed algorithm is between 50 and 70, while the ANMFP of ORB-SLAM2 is above 200. The DSO algorithm requires a larger ANMFP, because it uses a direct method framework. It can be seen that the number of feature points required by the proposed algorithm is much smaller than that of ORB-SLAM2 and DSO. The reason for this result is determined by the basic structure of the algorithm in this paper. The algorithm does not directly calculate the **H** or **E** by relying on the correspondence between the feature points of two adjacent frames, but continuously optimizes the initial pose by using the feature point correspondences that can be continuously observed in successive frames. That is to say, for this method, there is no need to have so many feature points in the initial frame. This method could get an acceptable initial attitude as long as enough points can be continuously observed in successive frames. This also explains from another side why the algorithm can achieve a higher SRI. Therefore, the method in this paper can achieve better results from little feature points when dealing with sparse image features.

4. Conclusion

In this paper, we propose a rectilinear hypothesis of platform motion and thereby derive a passive initialization method for monocular SLAM. Init BA and subsequent BA are utilized in reducing the errors between the actual motion and that of the proposed hypothesis. A simulated fixed-wing aircraft is selected as the test platform for the proposed method. Results show that the success rate of monocular SLAM Initialization

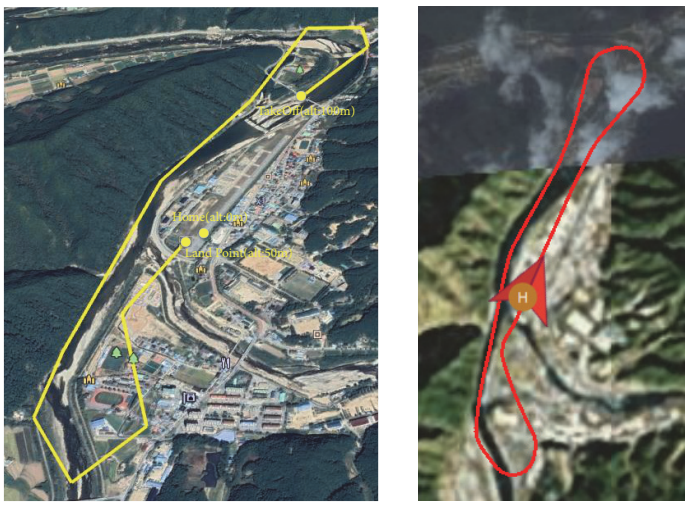


Figure 9: Simulation scenarios on mountainous terrain.

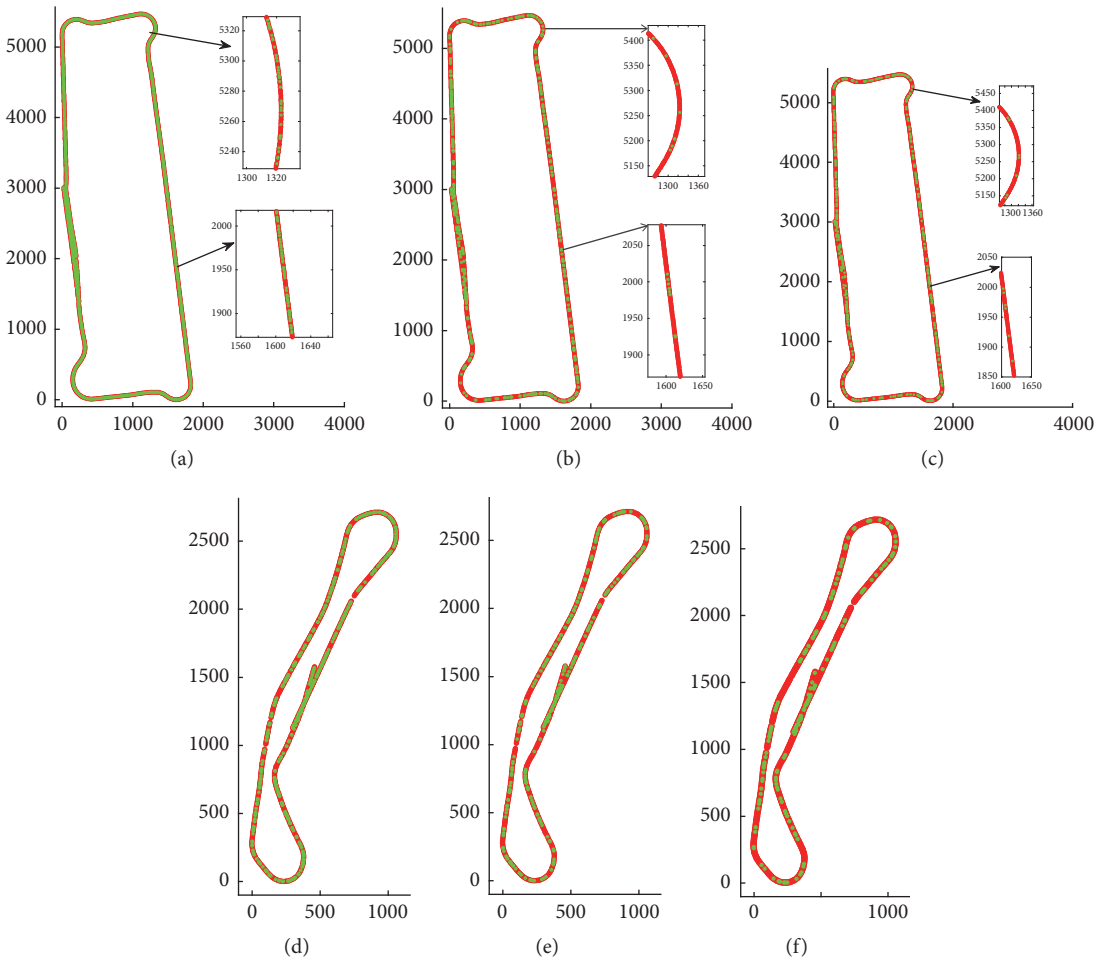


Figure 10: Initialization results comparison (green: success; red: failed), (a) plain (ours), (b) plain (ORB-SLAM2), (c) plain (DSO), (d) mountainous (ours), (e) mountainous (ORB-SLAM2), and (f) mountainous (DSO).

Table 6: Number of matched feature points needed.

	method	Terrain	ANMFP
1	ours ($\delta_s = 1$)	Plain	57.1
2	ours ($\delta_s = 0.7$)	Plain	58.5
3	ours ($\delta_s = 0.5$)	Plain	56.1
4	ours ($\delta_s = 0.3$)	Plain	55.2
5	ours ($\delta_s = 0.1$)	Plain	63.3
6	ORB-SLAM2	Plain	297.3
7	DSO	Plain	1907.1
8	ours ($\delta_s = 1$)	Mountainous	78.7
9	ours ($\delta_s = 0.7$)	Mountainous	68.2
10	ours ($\delta_s = 0.5$)	Mountainous	74.9
11	ours ($\delta_s = 0.3$)	Mountainous	66.2
12	ours ($\delta_s = 0.1$)	Mountainous	71.1
13	ORB-SLAM2	Mountainous	254.8
14	DSO	Mountainous	1953.8

is greatly improved compared with that of ORB-SLAM2. However, this method is only effective on platforms with strong motion characteristics and cannot be used indiscriminately on platforms characterized by randomized motions, such as humans and animals. At present, the method has yet to be tested in real-world environments, which will be rectified in future works.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

The research received no external funding. Among the authors, Yu Yang, Jing Xiong, and Xiaoyu She are graduate students of Beijing Institute of Technology. The research was performed as part of their education. Authors Jie Li, Chengwei Yang, and Chang Liu are employed by Beijing Institute of Technology. They only played a supervising role in the research.

Conflicts of Interest

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

References

[1] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[2] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, ""Bundle adjustment" a modern synthesis," in *Proceedings of the Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., vol. 1883 of *Lecture Notes in Computer Science*, pp. 298–372, Springer, Corfu, Greece, 1999.

[3] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: a general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 3607–3613, IEEE, Shanghai, China, May 2011.

[4] S. Agarwal and K. Mierle, "Ceres solver," http://ceres-solver.org.

[5] Y. Lin, F. Gao, T. Qin et al., "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 23–51, 2018.

[6] F. Nex and F. Remondino, "UAV for 3D mapping applications: a review," *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

[7] S. Lin, M. A. Garratt, and A. J. Lambert, "Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment," *Autonomous Robots*, vol. 41, no. 4, pp. 881–901, 2017.

[8] D. Scaramuzza, M. C. Achtelik, L. Doitsidis et al., "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robotics and Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[9] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg, "Global localization from monocular SLAM on a mobile phone," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 4, pp. 531–539, 2014.

[10] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '07)*, pp. 225–234, Nara, Japan, November 2007.

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[13] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '14)*, pp. 15–22, IEEE, Hong Kong, June 2014.

[14] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation, ICRA 2014*, pp. 2609–2616, IEEE, June 2014.

[15] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Proceedings of the Computer Vision –ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8690, pp. 834–849, Springer International Publishing, 2014.

[16] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[17] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2nd edition, 2003.

[18] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[19] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: an accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[20] L. Kneip, H. Li, and Y. Seo, "UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability," in *Proceedings of the European Conference on Computer Vision*, vol. 8689, pp. 127–142, Springer, 2014.

Hindawi
Complexity
Volume 2019, Article ID 3563674, 16 pages
https://doi.org/10.1155/2019/3563674

Research Article

Neural Personalized Ranking via Poisson Factor Model for Item Recommendation

Yonghong Yu,[1] **Li Zhang**,[2] **Can Wang**,[3] **Rong Gao**,[4] **Weibin Zhao**,[1] **and Jing Jiang**[1]

[1]*College of Tongda, Nanjing University of Posts and Telecommunications, China*
[2]*Department of Computer and Information Sciences, Northumbria University, UK*
[3]*School of Information and Communication Technology, Griffith University, Australia*
[4]*School of Computer Science, Hubei University of Technology, China*

Correspondence should be addressed to Yonghong Yu; yuyh@njupt.edu.cn

Received 22 September 2018; Revised 16 December 2018; Accepted 18 December 2018; Published 3 January 2019

Academic Editor: Rongqing Zhang

Recommender systems have become indispensable for online services since they alleviate the information overload problem for users. Some work has been proposed to support the personalized recommendation by utilizing collaborative filtering to learn the latent user and item representations from implicit interactions between users and items. However, most of existing methods simplify the implicit frequency feedback to binary values, which make collaborative filtering unable to accurately learn the latent user and item features. Moreover, the traditional collaborating filtering methods generally use the linear functions to model the interactions between latent features. The expressiveness of linear functions may not be sufficient to capture the complex structure of users' interactions and degrades the performance of those recommender systems. In this paper, we propose a neural personalized ranking model for collaborative filtering with the implicit frequency feedback. The proposed method integrates the ranking-based poisson factor model into the neural networks. Specifically, we firstly develop a ranking-based poisson factor model, which combines the poisson factor model and the Bayesian personalized ranking. This model adopts a pair-wise learning method to learn the rankings of uses' preferences between items. After that, we propose a neural personalized ranking model on top of the ranking-based poisson factor model, named NRPFM, to capture the complex structure of user-item interactions. NRPFM applies the ranking-based poisson factor model on neural networks, which endows the linear ranking-based poisson factor model with a high level of nonlinearities. Experimental results on two real-world datasets show that our proposed method compares favorably with the state-of-the-art recommendation algorithms.

1. Introduction

Recommender systems [1] have become an indispensable component in E-commerce, online news and social media sites. These systems alleviate the information overload problem for users, by discovering the users' hidden preferences and providing users with the personalized information, products, or services. With such attractive features, recommender systems are widely employed in many online applications, including Amazon, Youtube, and Netflix.

As one of the most widely used techniques for recommender systems, collaborative filtering (CF) [2] has achieved great success in E-commerce. CF methods, which are independent of specific domains, make recommendations by analyzing the past activities of users. The main idea of CF is to learn the latent user preferences and the item characteristics by modelling the user-item interaction behaviors. Among a variety of CF methods, matrix factorization (MF) [3, 4] has drawn a large amount of attentions, due to its effectiveness and efficiency for coping with large datasets. MF method assumes that only a few latent factors contribute to the preferences of users and the characteristics of items. Matrix factorization approach simultaneously embeds both the user and item feature vectors into a low-dimensional latent factor space.

Most of the traditional CF methods work on explicit feedback, i.e., the ratings on items given by users. They generally apply the point-wise regression methods to predict

the ratings for unobserved items. However, explicit feedback may not always be available, since it is comparatively difficult to collect. As a result, using the implicit feedback (e.g., clicks, bookmarks and purchases) to express users' preferences is more common in the practical recommender systems. In CF with implicit feedback, only the positive instances are observed, while the negative instances and the missing values are mixed together, which make the personalized recommendation with implicit feedback more challenging.

Collaborative filtering with implicit feedback is referred to as the One-Class Collaborative Filtering (OCCF) problem [5, 6]. In order to solve the OCCF problem, Pan et al. [5] and Hu et al. [6] proposed a Weighted Regularized Matrix Factorization (WRMF) method. Rendle et al. [7] formulated recommendation as a ranking problem and proposed the Bayesian Personalized Ranking (BPR). Zhao et al. [8] proposed the Social Bayesian Personalized Ranking (SBPR) model, which integrates social connections with the users' implicit feedback to estimate users' rankings of items. However, most of the existing methods for OCCF simplify the implicit interactions between users and items. Regardless of how many times a user interacts with an item, they use the binary implicit feedback to indicate whether a user has clicked or viewed an item [5–8]. In other words, the existing methods generally use the matrix factorization models to quantify the binary implicit interactions between users and items. Intuitively, the number of implicit interactions reflects the degree of the user's preferences for items. The larger the number of interactions, the more preferred. Hence, such simplified schemes make the CF methods unable to accurately capture the users' preferences for items.

In addition, as reported in [9], matrix factorization based models use a linear function (i.e., inner product) to model the interactions between the user latent features and the item latent features. The expressiveness of the linear function is too limited to capture the complex structure of users' interactions, which hinders the performance of recommender systems. Hence, He et al. [9] proposed a general framework, named neural collaborative filtering (NCF) for recommender systems and suggested applying neural networks to learn the nonlinear interaction function from data. The nonlinear interaction function learned from the interaction data endows recommender systems with a high level of nonlinearities. Similar to the traditional recommendation methods that work on the explicit feedback, however, the point-wise learning method in NCF degrades the recommendation performance due to the data sparsity issue.

To tackle the aforementioned issues, in this paper, we propose a neural personalized ranking model for collaborative filtering with implicit frequency feedback, which integrates the ranking-based poisson factor model with the neural networks. Specifically, we basically adopt the poisson factor model (PFM) [10, 11], instead of the classical matrix factorization techniques, to model the implicit interactions between users and items. The poisson factor model replaces the usual Gaussian likelihood in the probabilistic matrix factorization with the Poisson likelihood, which guarantees the nonnegativity of latent factors. Moreover, as pointed out by Ma et al. [10], the poisson factor model is better at modelling the frequency data than the traditional matrix factorization models. However, the data sparsity of implicit frequency feedback limits the performance of the poisson factor model, because the observed feedback available is not sufficient for the poisson factor model to learn latent features. To solve the data sparsity issue, we develop a ranking-based poisson factor model, which combines the poisson factor model and the Bayesian personalized ranking. The ranking-based poisson factor model adopts a pairwise learning method to learn the rankings of preferences between items. In order to capture the complex structure of interactions, moreover, we propose a neural personalized ranking model on top of the ranking-based poisson factor model, called NRPFM. NRPFM integrates the ranking-based poisson factor model with the neural networks, which provides the linear ranking-based poisson factor model with a high level of nonlinearities. In our neural personalized ranking model, we use the multilayer perceptron (MLP) [12] to learn the nonlinear and nontrivial user-item interaction relationships. Hence, our proposed neural personalized ranking model unifies the strengths of the ranking-based poisson model in learning users' preferences ranking between items from implicit frequency feedback, and the neural networks in capturing the nonlinear user-item interaction relationships.

The key contributions of our work are summarized as follows:

(i) We propose a ranking-based poisson factor model, which combines the poisson factor model and the Bayesian personalized ranking to tackle the data sparsity of implicit frequency feedback.

(ii) We propose a neural personalized ranking model for collaborative filtering with implicit frequency feedback. This neural personalized ranking model integrates the ranking-based poisson factor model with the neural networks, which endows the linear ranking-based poisson factor model with a high level of nonlinearities.

(iii) We perform extensive experiments to evaluate our proposed method on real-life datasets. The results show that our proposed method outperforms the state-of-the-art recommendation algorithms.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work in recommender systems. Section 3 introduces some preliminary knowledge. Section 4 describes the details of our proposed item recommendation algorithm. Experiments are evaluated in Section 5. Finally, we conclude this paper and present some directions for future work in Section 6.

2. Related Work

In this section, we review the major related work for recommender systems, including the traditional collaborative filtering methods and the neural network-based recommendation methods.

2.1. Collaborative Filtering. Collaborative filtering (CF) [2] approaches are widely deployed in the modern E-commerce websites and have achieved a great success. CF approaches include two main categories [2]: memory-based algorithms and model-based algorithms, according to different ways of utilizing a user-item rating matrix.

Memory-based CF algorithms, also known as neighbor-based methods, use the entire user-item rating matrix to generate recommendations. Typical memory-based algorithms include user-based methods [2] and item-based methods [13, 14]. The underlying assumption of memory-based methods is that similar users share common interests, and users usually prefer similar items. The key issue of user-based and item-based methods is to adopt suitable similarity measures to calculate the pairwise similarity between users or between items. Typical similarity measures include the cosine similarity, the Pearson correlation coefficient, and the adjusted cosine similarity [13]. Model-based CF methods firstly learn a predictive model, which characterizes the rating behaviors of users, by exploiting the statistical and machine learning techniques. They use the predictive models to predict users' future behaviors. Typical model-based filtering approaches include Bayesian networks [2], clustering model [15, 16], latent semantic analysis [17, 18], and restricted Boltzmann machines [19].

As the most popular approaches among various CF methods, matrix factorization methods (MF) [3, 4] have attracted a lot of attentions due to their effectiveness and efficiency in dealing with a very large scale user-item rating matrix. The basic assumption of matrix factorization is that only a few latent factors contribute to the preferences of users and the characteristics of items. Therefore, matrix factorization approaches simultaneously embed both user and item feature vectors into a low-dimensional latent factor space, where the correlation between user's preference and item characteristics can be computed directly. Typical matrix factorization approaches include NMF [20], PMF [4], SVD++ [21], and MMMF [22].

These above matrix factorization based recommendation algorithms generally learn the latent feature vectors of users and items from users' explicit feedback (i.e., users' ratings on items). Explicit feedback however may not always be available since it is difficult to collect. So, it is more common for recommender systems to present users' preferences using implicit feedback (e.g., clicks, bookmarks and purchases) in real world, since implicit feedback is relatively easy to obtain. However, only positive instances are observed in implicit feedback, and negative instances and missing values are mixed together, which make the personalized recommendation with implicit feedback more challenging. Collaborative filtering with implicit feedback is referred as the One-Class Collaborative Filtering (OCCF) problem [5, 6]. To solve the OCCF problem, Pan et al. [5] and Hu et al. [6] proposed a Weighted Regularized Matrix Factorization (WRMF) method. WRMF treats all missing entries as negative instances and assigns varying confidence to positive and negative instances. Rendle et al. [7] modeled the rankings of feedback and proposed a Bayesian Personalized Ranking (BPR) criterion for recommendation systems based on implicit feedback. Pan et al. [23] extended BPR and proposed the Group Bayesian Personalized Ranking (GBPR), via introducing the richer interactions among users. GBPR aggregates the features of similar users in groups to reduce sampling uncertainty. In [8], Zhao et al. proposed the Social Bayesian Personalized Ranking (SBPR) model, which integrates social connections with users' implicit feedback to estimate users' rankings of items. In [24], Zhao et al. utilized the cross-region community matching technique to generate personalized locally interest locations for users. In particular, they proposed the Bayesian probabilistic tensor factorization with social and location regularization (BPTF-SLR) framework to extract users' latent social dimensions from users' implicit feedback. It must be noted that the recommender systems with implicit feedback are more practical than those with explicit feedback. Therefore, the recent research directions on recommendation have been shifted towards learning users' hidden preferences from implicit feedback, rather than inferring users' tastes from explicit feedback.

2.2. Neural Networks-Based Recommendation Approaches. Recently, many research has employed the neural network technique to design recommendation algorithms, because neural network technique is able to effectively capture the non-linear and non-trivial user-item interaction relationships [25] and extract deep and abstract feature representations for users and items, leading to large improvements in recommendation quality. Representatives of neural network-based recommendation algorithms include Wide & Deep Learning [26], NCF [9], NFM [27], AutoRec [28], CDAE [29], and ConvMF [30].

Among deep neural network techniques, the multilayer perceptron (MLP) [12] is able to approximate measurable function and widely adopted in recommender systems. To provide the App recommendation in Google play, Cheng et al. [26] presented the Wide & Deep Learning approach, which consists of the wide learning model and the deep learning model. The wide learning component is a single layer perceptron and can effectively memorize feature interactions using the cross-product feature transformations. The deep learning component applies the MLP to generalize to the unobserved feature interactions through low-dimensional embeddings. In [9], He et al. proposed a general framework named neural collaborative filtering (NCF) for CF based on neural networks. Specifically, NCF leverages multilayer perceptron to learn the user-item interaction function, which endows NCF modelling with a high level of nonlinearities. Under the NCF framework, three instantiations of NCF are presented, i.e., GMF, MLP, and NeuMF. GMF employs a linear kernel to model the latent feature interactions, and MLP utilizes a nonlinear kernel to the user-item interaction function. Based on GMF and MLP, NeuMF unifies the linearity of GMF and the nonlinearity of MLP for modelling the complex interactions between users and items. Furthermore, Wang et al. [31] extended NCF to solve the cross-domain social recommendation problem (i.e., recommending the relevant items of information domains to the potential users

of social networks) and proposed a neural social collaborative ranking (NSCR) approach. NSCR enhances NCF by plugging a pairwse pooling operation on top of embedding vectors and utilizes the graph regularization technique to model the cross-domain social relations. In addition, in order to simultaneously model the low-order feature interactions and the high-order feature interaction, Guo et al. [32] proposed an end-to-end model, named deepFM, which seamlessly fuses the factorization machine (FM) [33] and MLP. Similar to deepFM, which employes FM and MLP for recommendation, He et al. [27] proposed the neural factorization machine (NFM) for prediction under sparse settings. Unlike other MLP-based methods, NFM introduces a Bi-Interaction pooling component on top of embeddings vectors, which captures the second-order feature interactions in the low-level and greatly facilitates the following hidden layers of NFM to learn the high-order feature interactions. An extension of NFM, called AFM, is also proposed in [34]. AFM takes the importance of different feature interactions into consideration and learns the importance of each feature interaction via a neural attention network.

As one of the core components of deep neural network, autoencoder [35] technique is able to reconstruct inputs in the output layer via a low-dimensional hidden space; some researchers have employed the autoencoder technique in recommender systems to improve the recommendation performance. For example, Sedhain et al. [28] utilized the autoencoder paradigm to make recommendation and proposed AutoRec. Specifically, AutoRec takes the user-partial observed vectors or the item-partial vectors as inputs and embeds them into a low-dimensional hidden space. Finally, AutoRec reconstructs inputs in the output layer by directly optimizing Root Mean Square Error (RMSE). According to the types of inputs, AutoRec includes two variants: U-AutoRec and I-AutoRec, which correspond to take the user-partial observed vectors and the item-partial vectors as inputs, respectively. Compared to the classical autoencoder technique, denoising autoencoder (DAE) [36] techniques are able to discover more robust representations and avoid learning an identity function. In order to take the advantages of DAE, several research work employes DAE techniques for CF [37, 38]. Li et al. [38] proposed the deep collaborative filtering framework (DCF), which unifies the deep learning models with MF based CF. The key deep learning model used in DCF is the marginalized denoising auto-encoders (mDA) [39], which is more computationally efficient and has a closed-form solution to learn model parameters. Moreover, Wu et al. [29] utilized the idea of DAE and proposed the collaborative denoising autoencoder (CDAE) for CF. The key difference between CDAE and the above DAE-based CF methods is that CDAE considers the personalized user factors by encoding a latent vector for each user, which greatly improve the recommendation performance. These above DAE-based recommendation methods [29, 37, 38] assume that the observed user-item interactions are a corrupted version of the user's full preferences and learn the latent representations of the corrupted user-item preferences, which can be used to reconstruct users' full preferences. In contrast, Wang et al. [40] proposed a hierarchical Bayesian model, called CDL. CDL integrates the stacked denoising autoencoder (SDAE) [41] into PMF [4] and jointly uses SDAE to learn the deep representations for item content and utilizes PMF to perform collaborative filtering for ratings matrix. Note that CDL takes the item features as inputs for SDAE while other DAE-based methods (i.e., AutoRec, DCF, and CDAE) make user feedback as inputs. In other words, CDL utilizes the deep learning component to model the auxiliary information rather than model user behaviors.

Convolution neural network (CNN) [42] is a specialized kind of feedforward neural network for processing the data with grid-like topology. CNN-based recommendation methods usually utilize CNN to extract the deep and abstract feature representations [30, 43–45] from images, audio, and text information. Wang et al. [46] proposed a visual content enhanced point-of-interest (POI) recommendation method (VPOI). VPOI incorporates the visual features extracted via CNN into PMF for learning the latent features of users and POIs. He et al. [44] proposed a visual Bayesian personalized ranking (VBPR) algorithm that incorporates the visual features learned from the product images by CNN into MF. In [45], Oord et al. utilized CNN to extract the latent features from music audio for the music recommendation. Gong et al. [43] formulated the hashtag recommendation task as a multiclass classification problem and proposed an attention based CNN architecture for the hashtag recommendation. Zheng et al. [47] proposed the deep cooperative neural network (DeepCoNN). DeepCoNN consists of two parallel CNNs coupled together by a shared common layer to model the user behaviors and the item properties from reviews. Moreover, Kim et al. [30] integrated CNN into the probabilistic matrix factorization and proposed the convolutional matrix factorization for recommendation (ConvMF). ConvMF utilizes CNN to capture the contextual information of item content and further enhance the rating prediction accuracy. Other neural network-based approaches include the recurrent neural network (RNN) based methods [48–50], the restricted Boltzmann machine (RBM) based methods [19, 51], and the generative adversarial network based methods [52].

There are also other cross-domain multimodal research developments on recommendation systems that provide interesting and insightful discussions to guide future directions. As an example, Nie et al. [53] proposed a scheme to rerank web images for complex queries from probabilistic perspective. Specifically, they first proposed a heuristic approach to detect noun-phrase based visual concepts from complex query. Then, they proposed a heterogeneous network to automatically estimate the relevance score of each image, which jointly integrates three layer relationships, spanning from semantic level to visual level. Nie et al. [54] focused on a challenging image search performance prediction problem. By analyzing Normalized Discounted Cumulative Gain (NDCG) and Average Precision (AP), they found that only the prediction of the images' relevance probabilities was required to compute their mathematical expectations. Therefore a query-adaptive graph-based learning approach was proposed to estimate the relevance probability of each image to a given query.

The works that are most related to ours are BPR [7] and NCF [9]. Comparing BPR with NRPFM, the major difference between BPR and NRPFM includes three aspects: (1) BPR is designed to learn latent user and item feature vectors from implicit binary feedback, while NRPFM is designed to learn latent user and item representations from implicit frequency feedback, which is more practical in recommendation scenarios; (2) BPR uses a linear function (i.e., inner product) to model the interactions between user latent features and item latent features. By contrast, NRPFM uses a nonlinear function learned from a multilayer perceptron to model the complex interactions between latent user features and latent item features, which endows NRPFM with a high level of nonlinearity; (3) essentially, BPR assumes that users' implicit feedback follows the Gaussian distribution, which does not fit the heavily skewed frequency data well. But NRPFM assumes that users' implicit feedback follows the Poisson distribution, which is more suitable for fitting the skewed frequency feedback. In addition, although both NCF and NRPFM leverage the multilayer perceptron to model the nonlinear interactions between latent user features and latent item features, the difference between NRPFM and NCF mainly lies in three aspects. (1) Similar to BPR, NCF is also designed for implicit binary feedback, which is a simplified form of implicit frequency feedback. In contrast, NRPFM directly learns users' preferences and items' characteristics from the implicit frequency feedback. (2) Moreover, NCF leverages a point-wise method to learn latent user and item feature vectors, while NRPFM utilizes a pair-wise method to learn latent feature vectors for users and items. For NCF, only the observed feedback has contributions to the learning of latent user and item feature vectors. In contrast, NRPFM makes the missing values also contribute to learning latent user and item features. Hence, NRPFM to some extent better alleviates the data sparsity problem. (3) NCF also assumes that users' implicit feedback follows the Gaussian distribution, while NRPFM makes a different assumption that users' implicit feedback follows the Poisson distribution.

Our proposed Neural Personalized Ranking via Poisson Factor Model is an important attempt of this direction. Moreover, we consider more practical recommendation scenarios, in which users represent their preferences using the implicit frequency feedback, rather than using the simplified implicit binary feedback. Furthermore, although it is direct to use deep learning techniques to extend personalized ranking recommendation models (i.e., BPR), this scheme is not able to infer latent user preferences and item characteristics from implicit frequency feedback because traditional ranking based personalized recommendation models essentially are designed to deal with implicit binary feedback. By contrast, our proposed deep learning based personalized ranking model is built on top of BPR, Poisson factor model, and deep learning technique and unify the strengths of these models to more accurately learn latent user and item features from implicit frequency feedback. In short, our contributions are three-fold. Specifically, (1) we use Poisson factor model to model users' implicit frequency feedback. Subsequently, (2) we utilize the nonlinear function learned from the deep learning algorithm to model the complex interactions between latent user feature vectors and latent item feature vectors. Finally, (3) in order to train the recommendation model, we sample the set of triplets with partial order from implicit frequency feedback, resulting in alleviating the data sparsity problem.

3. Preliminary Knowledge

In this section, we introduce the preliminary knowledge related to our proposed neural personalized ranking based recommendation algorithm. We firstly describe the recommendation problem in Section 3.1. Then, we briefly introduce the Poisson factor model in Section 3.2.

3.1. Problem Description. In the typical recommender systems with implicit frequency feedback, users' implicit feedback is used to construct the user-item interaction matrix $R \in \mathfrak{R}^{M \times N}$, which is comprised of two entity sets: the set of M users $U = \{u_1, u_2, \ldots, u_M\}$ and the set of N items $I = \{v_1, v_2, \ldots, v_N\}$. Each entry r_{ui} of R represents the number of interactions between user u and item i. The number of interactions reflects the users' preferences for certain items. Note that, in the recommender systems with explicit feedback, the feedback r_{ui} usually indicates the value of the rating on item i given by user u. Ratings are usually integers and fall into $[0, 5]$, in which 0 indicates the missing value, since the user has not yet rated that item. In the recommender systems with implicit frequency feedback, however, the feedback r_{ui} has a larger range compared with ratings. For example, in shopping websites, user may click hundreds of times at some products, while user may click few times for other products. Moreover, 0 is the mixture of the missing value and the negative instance in implicit scenario, which indicates the user is not aware of the item, and the user does not like it, respectively. The set of items interacting with the user u is denoted as I_u ($I_u \subseteq I$). In practice, the user-item interaction matrix R is generally very sparse with many unknown entries, since a typical user may have only interacted with a tiny percentage of items. This sparse nature of the user-item interaction matrix leads to the poor recommendation quality. In this paper, we use "feedback" and "interaction" interchangeably.

The task of recommender systems with implicit frequency feedback is to learn the users' hidden preferences by utilizing users' interaction history and provide them with the ranked lists of items that user may be interested in.

3.2. Poisson Factor Model. Poisson factor model (PFM) [10] is a generative probabilistic model, which assumes that each observed element r_{ui} follows the Poisson distribution with the expectation f_{ui}: $r_{ui} \sim Poisson(f_{ui})$. The expected value matrix $\mathbf{F} \in \mathbb{R}^{M \times N}$ is factorized into the user latent feature matrix $\mathbf{P} \in \mathbb{R}^{K \times M}$ and the item latent feature matrix $\mathbf{Q} \in \mathbb{R}^{K \times N}$: $\mathbf{F} \sim \mathbf{P}^T\mathbf{Q}$. Besides assuming that the Poisson distribution generates the observed elements, PFM places Gamma priors over p_{uk} and q_{ik},

$$p(p_{uk} \mid \alpha_k, \beta_k) = \frac{p_{uk}^{\alpha_k - 1} \exp(-p_{uk}/\beta_k)}{\beta_k^{\alpha_k} \Gamma(\alpha_k)},$$
$$p(q_{ik} \mid \alpha_k, \beta_k) = \frac{q_{ik}^{\alpha_k - 1} \exp(-q_{ik}/\beta_k)}{\beta_k^{\alpha_k} \Gamma(\alpha_k)}, \quad (1)$$

where $\Gamma(.)$ is the Gamma function. α_k and β_k are the shape and rate parameters of Gamma distribution, respectively.

The generative process of an observed element r_{ui} is as follows.

(1) For each user u, generate each component of the user latent feature vector: $p_{uk} \sim Gamma(\alpha_k, \beta_k)$.

(2) For each item i, generate each component of the item latent feature vector: $q_{ik} \sim Gamma(\alpha_k, \beta_k)$.

(3) Generate $r_{ui} \sim Poisson(\mathbf{p}_u^T \mathbf{q}_i)$.

The posterior distribution of $\mathbf{P}$ and $\mathbf{Q}$ given the user-item interaction matrix $\mathbf{R}$ is as follows:

$$p(\mathbf{P}, \mathbf{Q} \mid \mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto p(\mathbf{R} \mid \mathbf{F})\, p(\mathbf{P} \mid \boldsymbol{\alpha}, \boldsymbol{\beta})\, p(\mathbf{Q} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}). \quad (2)$$

where $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$, $\boldsymbol{\beta} = \{\beta_1, \beta_2, \ldots, \beta_K\}$.

Maximizing the log-posterior distribution $p(\mathbf{P}, \mathbf{Q} \mid \mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ results in the following objective function:

$$\mathscr{L}^{PFM} = \min_{\mathbf{P},\mathbf{Q}} \sum_{u=1}^{M} \sum_{k=1}^{K} \left(\frac{p_{uk}}{\beta_k} - (\alpha_k - 1) \ln \left(\frac{p_{uk}}{\beta_k} \right) \right) + \sum_{i=1}^{N} \sum_{k=1}^{K} \left(\frac{q_{ik}}{\beta_k} - (\alpha_k - 1) \ln \left(\frac{q_{ik}}{\beta_k} \right) \right) + \sum_{u=1}^{M} \sum_{i=1}^{N} (f_{ui} - r_{ui} \ln f_{ui}) + const. \quad (3)$$

PFM applies the stochastic gradient descent algorithm (SGD) technique to learn the user latent feature matrix $\mathbf{P}$ and the item latent feature matrix $\mathbf{Q}$.

4. Our Approach

Our motivation is to learn user preferences and item characteristics from implicit frequency feedback, as well as model the complex structure of user interactions via deep learning. In order to implement this motivation, we do not directly fit the observed implicit frequency feedback, but fit the partial orders of user preferences for items embedded in the triplets, which are sampled from implicit frequency feedback according to our assumption. To endow the ranking-based Poisson factor model with a high level of nonlinearities, we use the multilayer perceptron to learn the nonlinear and nontrivial user-item interaction relationships. In the following sections, we elaborate our proposed neural personalized ranking model that integrates the neural networks into the ranking based poisson factor model for collaborative filtering with implicit frequency feedback.

4.1. Ranking-Based Poisson Factor Model. In practical recommender systems, the implicit interactions between users and items usually are displayed in the form of frequency data, which reflects the degree of the user's preferences for items. The larger the number of interactions, the more preferred. Traditional recommendation models generally simplify the implicit frequency feedback to be binary feedback, which to some extents leads to information loss. The Poisson factor model [10] replaces the usual Gaussian likelihood in probabilistic matrix factorization with the Poisson likelihood, which guarantees the nonnegativity of latent factors. Moreover, as reported in [10], Poisson factor model is suitable for modelling the frequency data, which displays similar property to implicit interactions. Hence, we basically adopt the poisson factor model to quantify users' interaction behaviors and learn the latent user features and the item features from implicit frequency feedback. Similar to classical matrix factorization models [3, 4], the Poisson factor model basically adopts the point-wise regression method to learn the latent representations for users and items from the observed feedback. In this sense, only the observed feedback has contributions to the learning of the latent user features and the item features. Due to the data sparsity issue that commonly exits in recommender systems, the available observed feedback is not sufficient for the poisson factor model to learn the latent features, resulting in degrading the performance of recommender systems.

The Bayesian personalized Ranking (BPR) [7] is a popular pairwise learning method for collaborative filtering with binary feedback and has been widely adopted in many recommendation models [8, 23]. BPR learns the latent user and item features by optimizing the Bayesian pairwise ranking criterion. Unlike the point-wise learning methods, BPR assumes that users prefer the observed items over the nonobserved items. In fact, BPR essentially makes the missing values contribute to the training of recommendation model. Hence, this pair-wise learning method somehow alleviates the data sparsity problem.

To tackle the sparsity of implicit frequency feedback, we propose a ranking based Poisson factor model, which combines the Poisson factor model and the Bayesian personalized ranking. The ranking based Poisson factor model adopts a pair-wise learning method to learn the rankings of preferences between items. Specifically, we assume that users' preferences for items increase as the numbers of interactions increase. This assumption implies three aspects: (1) the ranking of preferences for the observed item is higher than that of the preferences for nonobserved item; (2) if two items are observed, a user prefers the one with the larger number of interactions over the another one with the fewer number of interactions; (3) for two nonobserved items, we can not infer the order of their preferences. Let r_{ui} denote the number of interactions between the user u and the item i, and r_{uj} for the user u and the item j. If $r_{ui} > 0$ *and* $r_{uj} = 0$, then the user u prefers the item i over the item j; i.e., $i >_u j$, where $>_u$ indicates the preference relationships between user u and items. If $r_{ui} - r_{uj} \geq \tau$ *and* $r_{uj} > 0$, then $i >_u j$, where τ denotes a threshold parameter. In other words, if the difference between r_{ui} and r_{uj} surpasses the threshold τ, we assume that the preference

on the item i is ranked higher than the preference on the item j. Formally, training set D_R (i.e., the set of triplet (u, i, j)) is defined as follows:

$$D_R = \{(u,i,j) \mid (r_{ui} > 0 \wedge (r_u j) = 0) \vee ((r_{ui} - r_{uj}) \geq \tau \wedge (r_{uj} > 0))\}. \tag{4}$$

Given the preference relationships between the user u and items $>_u$, we maximize the posterior probability $p(\Theta \mid >_u)$ to learn the latent user and item features. Θ denotes model parameters; i.e., $\Theta = \{\mathbf{P}, \mathbf{Q}\}$. Through the Bayesian inference, the posterior probability of $\mathbf{P}$ and $\mathbf{Q}$ can be obtained as follows:

$$p(\Theta \mid >_u) \propto p(>_u \mid \Theta)\, p(\Theta). \tag{5}$$

All users are presumed to be independent of each other, and the preference ranking of one (user,item) pair for a specific user is also assumed to be independent of the rankings of other (user,item) pairs. As a result, the likelihood function for all the users is formulated as

$$\prod_{u \in U} p(>_u \mid \Theta) = \prod_{(u,i,j) \in U \times I \times I} p(i >_u j \mid \Theta)^{I((u,i,j) \in D_R)} \times (1 - p(i >_u j \mid \Theta))^{I((u,i,j) \notin D_R)}, \tag{6}$$

where $I(x)$ is the indication function. Based on the totality and antisymmetry properties of preferences relationship, (6) is rewritten as

$$\prod_{u \in U} p(>_u \mid \Theta) = \prod_{(u,i,j) \in D_R} p(i >_u j \mid \Theta). \tag{7}$$

$p(i >_u j \mid \Theta)$ denotes the probability that the user u prefers the item i over the item j and is defined as

$$p(i >_u j \mid \Theta) = \sigma(\widehat{r}_{uij}(\Theta)), \tag{8}$$

where $\sigma(.)$ is the logistic sigmoid function. $\widehat{r}_{uij}(\Theta)$ is a real value function of model parameters and captures the relationship between the user u, the item i, and the item j and is defined as follows:

$$\widehat{r}_{uij}(\Theta) = \widehat{r}_{ui}(\Theta) - \widehat{r}_{uj}(\Theta) = \mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j. \tag{9}$$

In addition, Gamma priors are assumed for the latent user and item feature vectors:

$$p(\Theta) = p(\mathbf{P} \mid \boldsymbol{\alpha}, \boldsymbol{\beta})\, p(\mathbf{Q} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{u=1}^{M} \prod_{k=1}^{K} \frac{p_{uk}^{\alpha_k - 1} \exp(-p_{uk}/\beta_k)}{\beta_k^{\alpha_k} \Gamma(\alpha_k)} \times \prod_{i=1}^{N} \prod_{k=1}^{K} \frac{q_{ik}^{\alpha_k - 1} \exp(-q_{ik}/\beta_k)}{\beta_k^{\alpha_k} \Gamma(\alpha_k)}. \tag{10}$$

Substitute the likelihood function defined in (7) and the model parameter priors defined in (10) into (5); then maximize the log of the posterior probability; we obtain the objective function of the ranking-based poisson factor model as follows:

$$\mathscr{L}^{RPFM} = \min_{\mathbf{P},\mathbf{Q}} - \ln \sigma(\widehat{r}_{uij}(\Theta)) + \sum_{u=1}^{M} \sum_{k=1}^{K} \left(\frac{p_{uk}}{\beta_k} - (\alpha_k - 1) \ln \left(\frac{p_{uk}}{\beta_k} \right) \right) + \sum_{i=1}^{N} \sum_{k=1}^{K} \left(\frac{q_{ik}}{\beta_k} - (\alpha_k - 1) \ln \left(\frac{q_{ik}}{\beta_k} \right) \right). \tag{11}$$

It should be noted that both PFM and ranking-based PFM basically assume that each observed element r_{ui} in the user-item interaction matrix follows the Poisson distribution. Moreover, they both place Gamma priors over each entry of latent user feature matrix and item feature matrix. In addition, they basically are generative probabilistic models. The ranking-based PFM model is an extension of PFM. The difference between the ranking-based PFM and PFM models is that PFM is a point-wise recommendation model, whereas the ranking-based PFM adopts a pair-wise method to learn model parameters. For PFM, only observed feedback has contributions to learning recommendation model parameters. For ranking-based PFM, both observed feedback and missing values contribute to learning model parameters.

4.2. The Architecture of Neural Personalized Ranking Model. As shown in (9), the ranking based Poisson factor model uses a linear function (i.e., inner product) to model the interactions between the user latent features and the item latent features. As reported in [9], the expressiveness of the linear function is limited and may not be sufficient to capture the complex structure of users' interactions, which hinders the performance of the ranking based Poisson factor model. To capture the complex structure of interactions; therefore we develop a neural personalized ranking model on top of the ranking-based Poisson factor model, called NRPFM. This model integrates the neural networks with the ranking-based Poisson factor model, which endows the linear ranking-based Poisson factor model with a high level of nonlinearities. In our neural personalized ranking model, we use the multilayer perceptron (MLP) [12] to learn the nonlinear and nontrivial user-item interaction relationships. Figure 1 presents the architecture of our proposed model, which consists of two branches - The left branch is used to predict score for the positive (user,item) pair and the right branch for the negative pair. Each branch includes four layers: embedding layer, merge layer, hidden layers, and prediction layer.

Embedding Layer. Embedding layer is aimed at mapping users and items into a low-dimensional latent space and uses the compact and dense real value vectors, instead of the sparse and high-dimensional vectors, to represent users and items.

The input of our model is a triplet (u, i, j) that indicates the indexes of user u and the corresponding ranking item pair (i, j). After one-hot encoding the user and item indexes, we obtain the sparse representations of users and items. Then, we

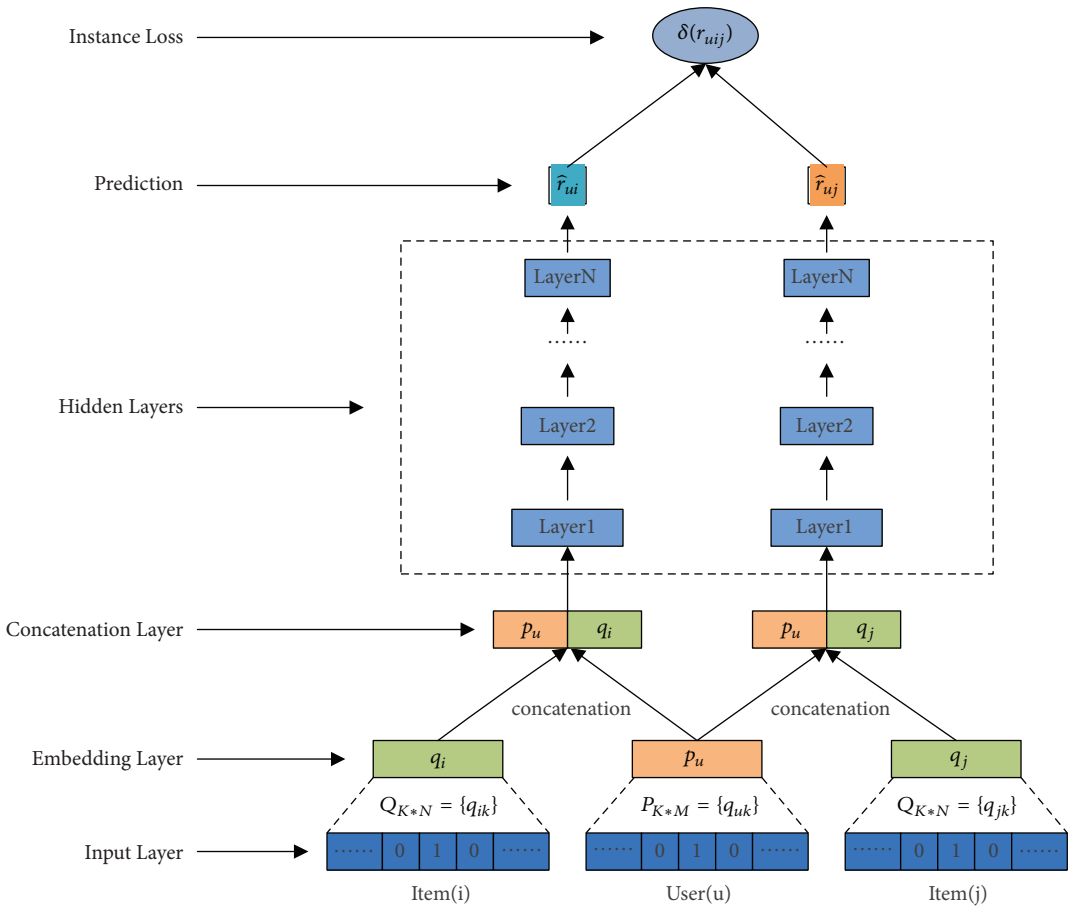


Figure 1: Neural personalized ranking architecture.

use the embedding table lookup to obtain the embeddings of user u and items i and j. Formally,

$$\begin{aligned} p_u &= \mathbf{P}.onehot\,(u)\,, \\ q_i &= \mathbf{Q}.onehot\,(i)\,, \\ q_j &= \mathbf{Q}.onehoe\,(j)\,, \end{aligned} \tag{12}$$

where $onehot(.)$ indicates the result of one-hot encoding for user or item. $\mathbf{P} \in \mathbb{R}^{K\times M}$ and $\mathbf{Q} \in \mathbb{R}^{K\times N}$ are the user embedding matrix and the item embedding matrix, respectively. The embedding matrices and the latent feature matrices derived from matrix factorization models have the same semantics. Hence, $\mathbf{p}_u$ represents the latent feature vector of user u and $\mathbf{q}_i$ indicates the latent feature vector of item i.

Above the embedding layer, we concatenate on the user embedding and the item embedding for each (user,item) pair in the merge layer. The concatenation of the embeddings of user and item jointly encodes the user preferences and the item characteristics. Then, we feed the concatenated embedding into the hidden layers. This design for the emerge layer is widely adopted in the multilayer perceptron- (MLP-) based recommendation methods [9, 26].

Hidden Layers and Prediction Layer. Since the simple concatenation of embeddings does not account for any interactions between the user latent features and the item latent features, we utilize a multilayer perceptron (MLP) to learn the user-item interaction relationships, which endows our model with a high level of nonlinearities. MLP stacks multiple fully connected hidden layers, where each hidden layer nonlinearly transforms the output of the previous hidden layer via the weight matrix and the activation function and feeds their output into the following hidden layer. The entire MLP adopts the tower structure, where the size of layer l is the half of the size of layer $l-1$. Based on this structure, a higher hidden layers is able to learn more abstract features for users and items.

The prediction layer is connected with the last hidden layer and takes ReLu as the activation function to predict the scores on items. Formally, the prediction score on the item i given by the user u is defined as follows:

$$\begin{aligned} \hat{r}_{ui} &= ReLU\left(\mathbf{W}_p^{(i)T}\mathbf{Z}_L^{(i)}\right), \\ \mathbf{Z}_L^{(i)} &= a_L\left(\mathbf{W}_L^{(i)T}\mathbf{Z}_{L-1}^{(i)} + \mathbf{b}_L^{(i)}\right), \\ &\vdots \\ \mathbf{Z}_1^{(i)} &= a_1\left(\mathbf{W}_1^{(i)T}\mathbf{x}^{(i)} + \mathbf{b}_1^{(i)}\right), \\ \mathbf{x}^{(i)} &= \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \end{aligned} \tag{13}$$

where $\mathbf{W}_p^{(i)}$ is the weight matrix of predication layer. $\mathbf{W}_l^{(i)}$, $\mathbf{b}_l^{(i)}$, and a_l denote the weight matrix, the bias vector, and the activation function for the lth hidden layer, respectively. $\mathbf{x}^{(i)}$ is the concatenation of the user embedding $\mathbf{p}_u$ and the item embedding $\mathbf{q}_i$. The prediction score on the item j given by the user u is computed in the same way. We adopt ReLU as the activation functions for hidden layers, because other activation functions, such as sigmoid and tanh, suffer from the saturation and lead to overfitting.

4.3. Model Learning. The prediction scores reflect users' preferences for items. After obtaining the prediction scores for the positive (user, item) pair and the negative (user, item) pair, i.e., $\widehat{r}_{ui}$ and $\widehat{r}_{uj}$, the real value function $\widehat{r}_{uij}(\Theta)$ is rewritten as follows:

$$\begin{aligned}\widehat{r}_{uij}(\Theta) &= \widehat{r}_{ui}(\Theta) - \widehat{r}_{uj}(\Theta) \\ &= ReLU\left(\mathbf{W}_p^{(i)^T}\mathbf{Z}_L^{(i)}\right) - ReLU\left(\mathbf{W}_p^{(j)^T}\mathbf{Z}_L^{(j)}\right).\end{aligned} \tag{14}$$

Integrating the above relationship function $\widehat{r}_{uij}(\Theta)$ with the ranking-based Poisson factor model, the objective function of our neural personalized ranking model is defines as follows:

$$\begin{aligned}\mathscr{L} = \min_{\mathbf{P},\mathbf{Q}} &\sum_{(u,i,j)\in D_R} -\ln\sigma \\ &\cdot\left(ReLU\left(\mathbf{W}_p^{(i)^T}\mathbf{Z}_L^{(i)}\right) - ReLU\left(\mathbf{W}_p^{(j)^T}\mathbf{Z}_L^{(j)}\right)\right) \\ &+\sum_{u=1}^{M}\sum_{k=1}^{K}\left(\frac{p_{uk}}{\beta_k} - (\alpha_k - 1)\ln\left(\frac{p_{uk}}{\beta_k}\right)\right) \\ &+\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\frac{q_{ik}}{\beta_k} - (\alpha_k - 1)\ln\left(\frac{q_{ik}}{\beta_k}\right)\right).\end{aligned} \tag{15}$$

We initialize the user embedding matrix $\mathbf{P}$ and the item embedding matrix $\mathbf{Q}$ with the Poisson distribution, which is parameterized by the shape parameters $\boldsymbol{\alpha}$ and the rate parameters $\boldsymbol{\beta}$. At the training stage, according to the rules of the construction of training set D_R, we uniformly sample triplets (u, i, j) in each iteration and control the number of negative items for each positive instance. After shuffling these sampled triplets, a batch of triplets are fed into our neural personalized ranking model. For the optimization algorithm, we adopt Adam [55] to update gradients, since Adam tunes the learning rate based on the adaptive schemes and thus yields fast convergence.

5. Experiments

In this section, we conduct several experiments on real datasets to compare the performance of our proposed recommendation algorithm with the state-of-the-art methods.

5.1. DataSet Description. There are many datasets used to evaluate the performance of recommendation algorithms, such as MovieLens (https://grouplens.org/datasets/movielens/), Yelp(https://www.yelp.com/dataset/challenge), Epinions(https://snap.stanford.edu/data/soc-Epinions1.html), and Netflix(https://www.netflixprize.com/). However, most of them only consist of explicit feedback (i.e., ratings). In our experiment, we choose two publicly available datasets(http://www.ntu.edu.sg/home/gaocong/datacode.htm): Foursquare and Gowalla, to evaluate the performance of our proposed method, since they include implicit frequency feedback of users. Foursquare (https://foursquare.com/) and Gowalla (http://gowalla.com/) are two popular location-based social networks (LBSNs), which attract lots of attention from both the industry and the academia recently. In both Foursquare and Gowalla, users present their preferences in terms of the check-ins at locations (e.g., restaurants, tourists spots, and stores). In other words, users interact with locations via the check-ins. The number of check-ins to some extend indicates the degree of users' preferences for locations.

Table 1: Statistics of Foursquare and Gowalla.

Statistics	Foursquare	Gowalla
Num. of Check-ins	194,108	242,172
Num. of Users, M	2,321	5,000
Num. of Locations, N	5,596	23,997
Sparsity	99.18%	99.86%
Avg. Locations per User	45.57	32.13

In the Foursquare dataset, all check-ins were collected within Singapore from Aug. 2010 to Jul. 2011. Check-ins of Gowalla dataset were made within California and Nevada from Feb. 2009 to Oct. 2010. In both datasets, users who have checked in fewer than 5 locations have been removed. Meanwhile, locations with less than 5 users have been filtered out. Foursquare dataset contains 194,108 check-in observations from 2,321 users at 5,596 locations. We randomly sample a subset from the original Gowalla dataset for evaluation. The sampled Gowalla dataset includes 242,172 check-in observations from 5,000 users at 23,997 locations. After aggregating the check-in records based on user and location identifiers, we obtain 105,764 entries in the user-location interaction matrix of Foursquare, and 160,689 entries in the user-location interaction matrix of Gowalla, respectively. The sparsity of Foursquare and Gowalla datasets is $1 - 105764/(2321 \times 5596) = 99.19\%$ and $1 - 160689/(5000 \times 23997) = 99.86\%$, respectively. Hence, both the Foursquare and Gowalla datasets are very sparse. In Foursquare dataset, each user checked in 45.57 locations on average. And in Gowalla, each user checked in 32.13 locations on average.

The general statistics of Foursquare and Gowalla are summarized in Table 1.

5.2. Evaluation Metrics. We focus on the recommendation problem with implicit feedback, which is formulated as the item recommendation problem aimed at providing users with top-k highest ranked items. Therefore, we employ two widely used rank metrics to evaluate the performance of different recommendation algorithms, i.e., Precision@k and Recall@k,

where k is the length of ranked recommendation list. Given a user u, the precision and recall are defined as follows:

$$Precision_u@k = \frac{\left|I_u^{rec} \cap I_u^{test}\right|}{k}$$
$$Recall_u@k = \frac{\left|I_u^{rec} \cap I_u^{test}\right|}{\left|I_u^{test}\right|} \quad (16)$$

where I_u^{rec} is the top-k recommended item list for the user u and I_u^{test} is the visited items list by the user u in testing set. The final Precision@k and Recall@k of the entire recommendation algorithm are computed by averaging the precision and recall values over all the users, respectively. For both metrics, we set $k = 3, 5, 10$ to evaluate the performance in our experiments.

5.3. Compared Approaches. In order to evaluate the effectiveness of our proposed method, we compare our method with the following state-of-the-art approaches:

(1) PMF: this method was proposed by Mnih and Salakhutdinov [4] and can be viewed as a probabilistic extension of SVD [56] model. PMF represents the latent user and item feature vector by means of a probabilistic graphic model with Gaussian observation noise.

(2) BPR: BPR adopts a Bayesian Personalized Ranking criterion [7] for item ranking. BPR is a pair-wise learning method for OCCF problem. In our experiments, we employ a uniform sampling strategy to sample the user-item pairs for model training.

(3) MLP: MLP is an instantiation of NCF [9], which concatenates the user and item embeddings, and feeds the concatenation into neural networks to model the nonlinear user-item interactions.

(4) NeuMF: NeuMF is another instantiation of NCF, which is a strong baseline that fuses the generalized matrix factorization and the multilayer perceptron to simultaneously model the linear and nonlinear interactions between the latent user features and the latent item features.

(5) PFM: this method was proposed by Ma et al. [10]. PFM focuses on website recommendation and models users' implicit feedback using the Poisson distribution.

(6) NRPFM: NRPFM is described in Section 4. NRPFM is a neural personalized ranking model for collaborative filtering with implicit frequency feedback, which integrates the ranking-based Poisson factor model with the neural networks.

5.4. Experiment Settings. In order to make a fair comparison, we set the parameters of each method, according to respective references or based on our experiments. Under these parameter settings, each method achieves its best performance. For PMF, we set λ_U and λ_V to be 0.001. For BPR, $\lambda_\Theta = 0.001$, we employ a uniform sampling strategy to sample the user-item pairs for model training. For PMF, BPR, and PFM, we set the learning rate involved in the gradient descent algorithm to be 0.0001. For MLP, we adopt the tower structure for neural networks with three hidden layers, where the sizes of each hidden layer are $(32, 16, 8)$. The embedding size of users and items is equal to the size of the first hidden layer, i.e, 32. For NeuMF, we adopt the default parameters setting of the original paper: the number of hidden layers is set to be 3, and the sizes of each hidden layer are $(32, 16, 8)$, respectively. The number of negative samples for each positive one is set to be 4, and the embedding size of the generalized matrix factorization is set to be 10. For PFM, $\alpha_k = 2, \beta_k = 0.05, k = 1, \ldots, K$. For our proposed NRPFM, we initialize the model parameters using Gamma distribution with the shape parameter $\alpha_k = 2$ and the rate parameter $\beta_k = 0.2$. We set the number of hidden layers to be 3 and the sizes of each hidden layer to be $(32, 16, 8)$. Meanwhile, we set the number of negative samples per positive instance to be 8, $\tau = 2$ and adopt Adam with the batch size of 256 as optimizer.

For each user, we randomly extract 70% of the visited locations as the training set and the remaining 30% of locations as the testing data. We conduct data splitting five times and report the average results on the test sets for each dataset.

5.5. Recommendation Quality Comparisons. Tables 2 and 3 report the recommendation quality of all the compared methods on Foursquare and Gowalla datasets.

From Tables 2 and 3, we have the following observations: (1) on both datasets, PMF performs the worst among all the compared methods. Besides the data sparsity issue, one possible reason is that PMF assumes a user's implicit feedback follows the Gaussian distribution, which is not suitable for modeling the implicit frequency feedback. (2) BPR achieves better performance than PMF. This is because BPR adopts the pair-wise learning method to infer the latent user and item feature vectors, making missing values contribute to the learning of model parameters. Hence, to some extent, this pair-wise learning method is able to alleviate the data sparsity problem. (3) Although MLP and NeuMF apply the point-wise methods to learning the latent representations of users and items, they are generally superior to BPR. This shows the strengths of the neural networks in capturing the complex structure of users' interactions. The performance improvements of MLP and NeuMF over BPR demonstrate that using the neural network to capture the nonlinear user-item interaction relationships is beneficial for collaborative filtering. (4) PFM outperforms PMF by a large margin and achieves comparable performance to BPR. This observation indicates that the Poisson factor model is more suitable for modeling uses' implicit frequency feedback than probabilistic matrix factorization. Meanwhile, the overall performance of PFM is worse than BPR. This is because PFM only uses the observed feedback to learn the latent feature vectors, suffering from the data sparsity issue. (5) NBPFM consistently outperforms other methods, which demonstrates the effectiveness of our proposed model for collaborative filtering

Table 2: Performance comparison on Foursquare.

Metric	PMF	BPR	MLP	NeuMF	PFM	NRPFM
Precision@3	0.00530	0.10777	0.10843	0.11350	0.10492	**0.12075**
Precision@5	0.00497	0.09932	0.10053	0.10523	0.09294	**0.10760**
Precision@10	0.00467	0.08151	0.08173	0.08328	0.06450	**0.08535**
Recall@3	0.00111	0.02329	0.02388	0.02473	0.02306	**0.02700**
Recall@5	0.00184	0.03654	0.03730	0.03933	0.03574	**0.04138**
Recall@10	0.00356	0.06073	0.06128	0.06320	0.05082	**0.06630**

Table 3: Performance comparison on Gowalla.

Metric	PMF	BPR	MLP	NeuMF	PFM	NRPFM
Precision@3	0.00093	0.04442	0.04782	0.04883	0.04436	**0.05320**
Precision@5	0.00085	0.03656	0.03826	0.03848	0.03148	**0.04167**
Precision@10	0.00068	0.02796	0.02602	0.02673	0.01666	**0.02800**
Recall@3	0.00021	0.02115	0.02540	0.02693	0.02442	**0.02817**
Recall@5	0.00044	0.02753	0.03278	0.03343	0.02900	**0.03537**
Recall@10	0.00118	0.03950	0.04104	0.04265	0.03026	**0.04337**

Table 4: Performance of NBPFM with different hidden layers on Foursquare.

Metric	NBPFM-0	NBPFM-1	NBPFM-2	NBPFM-3	NBPFM-4
Precision@3	0.11656	0.11730	0.12378	0.12075	0.11885
Precision@5	0.10643	0.10764	0.11203	0.10760	0.10480
Precision@10	0.08530	0.08614	0.08968	0.08535	0.08475
Recall@3	0.02573	0.02550	0.02743	0.02700	0.02640
Recall@5	0.04003	0.04033	0.04183	0.04138	0.03910
Recall@10	0.06410	0.06485	0.06708	0.06630	0.00620

with implicit frequency feedback. Our proposed method improves the Precision@3 of NeuMF by 6.4% and 8.9% on Foursquare and Gowalla, respectively. In terms of Recall@3, the improvements of NBPFM over NeuMF are 9.2% and 4.6% on Foursquare and Gowalla, respectively. This observation confirms our assumption that integrating the strengths of the ranking-based poisson model in learning users' preferences ranking between items and neural networks in capturing nonlinear user-item interaction relationships is able to boost the recommendation quality. (6) All the compared methods perform better on Foursquare than on Gowalla. The reason is that Gowalla is more sparse than Foursquare. With the dense user-item interactions, recommendation methods is more able to accurately learn the latent user and item feature vectors, resulting in better recommendation performance.

5.6. Sensitivity Analysis

5.6.1. Impact of the Depth of Neural Networks. In our proposed method, we use the neural networks, i.e, MLP, to learn the nonlinear interactions between the user and item feature vectors from users' implicit feedback. The depth of neural networks is an important factor that affects the expressiveness of neural networks. In the section, we conduct a group of experiments to investigate the impact of the depth of neural networks on the recommendation quality. We fix the size of the last hidden layer as 8 and vary the depth of neural networks from 1 to 4. For example, if the depth of neural networks is 4, then the structure of neural networks is $8 \longrightarrow 16 \longrightarrow 32 \longrightarrow 64$, and the embedding size of user and item is 64. Other parameters keep the same settings as described in Section 5.4. We only present the experimental results on Foursquare in Table 4 and the experimental results on Gowalla show similar trends.

In Table 4, the NBPFM-n denotes the NBPFM method with n hidden layers. As demonstrated in Table 4, NBPFM with different number of hidden layers consistently outperforms NeuMF. Moreover, it is not beneficial for NBPFM that stack more hidden layers in MLP to learn the nonlinear interaction functions between the latent user and item feature vectors. A possible reason is that a deeper neural network makes NBPFM have more trainable parameters, which are relatively difficult to learn with limited training data, resulting in the degradation of recommendation performance. In addition, for NBPFM without hidden layers, the performance of NBPFM-0 is even better than that of NeuMF. NBPFM-0 can be viewed as a variant of the ranking-based poisson factor model that utilizes the concatenation of user and item embeddings to predict scores. This observation somehow shows the effectiveness of our proposed ranking-based Poisson factor model for collaborative filtering with implicit frequency feedback.

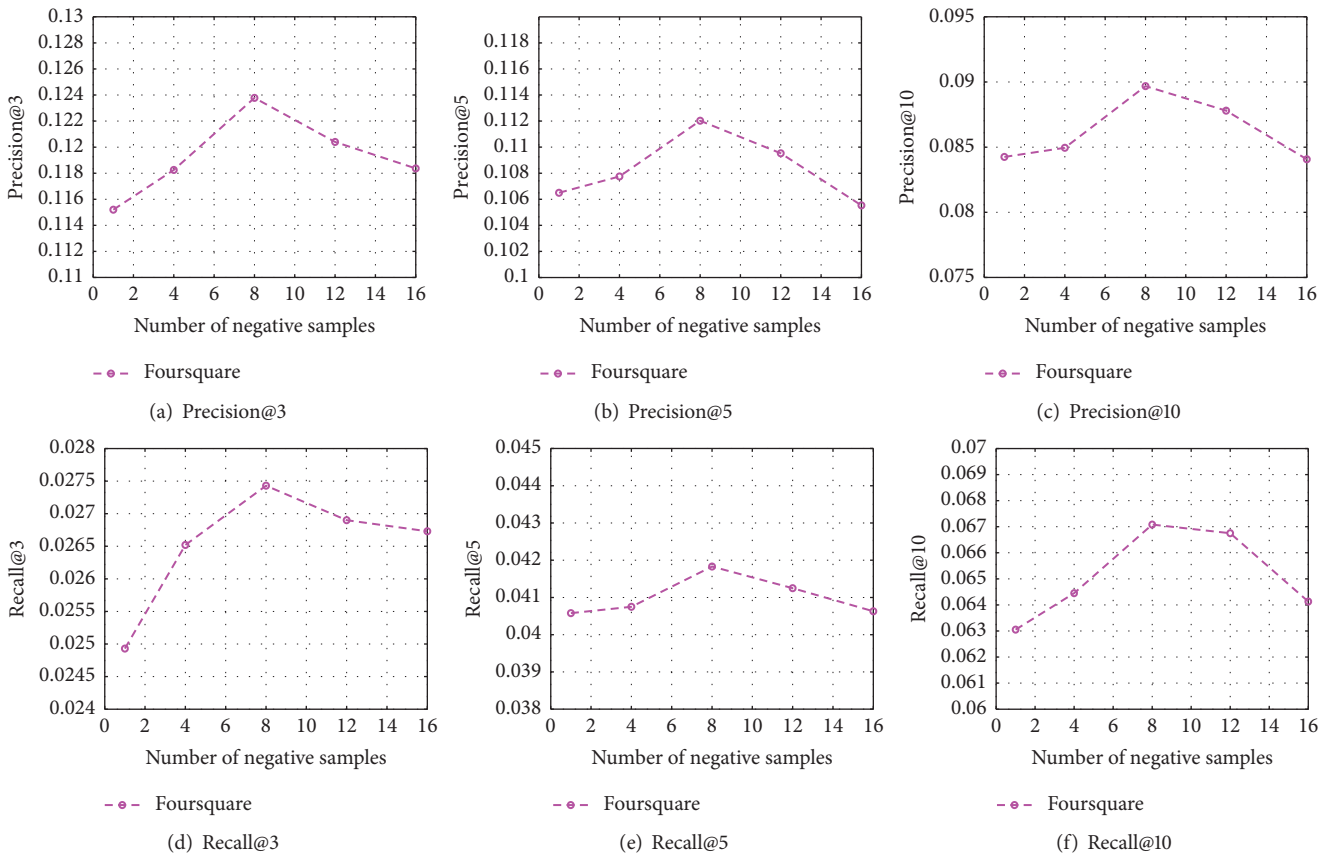


Figure 2: The impact of negative samples.

5.6.2. Impact of Negative Samples. In this section, we conduct another group of experiments to investigate the impact of negative samples on the recommendation quality. We vary the number of negative samples for each positive one from 1 to 16 and observe the changes of recommendation quality. We set the number of hidden layers to be 2 since NBPFM achieves better performance under this settings, which is shown in Table 4. And other parameters remain unchanged. The experimental results of NBPFM on Foursquare are shown in Figure 2.

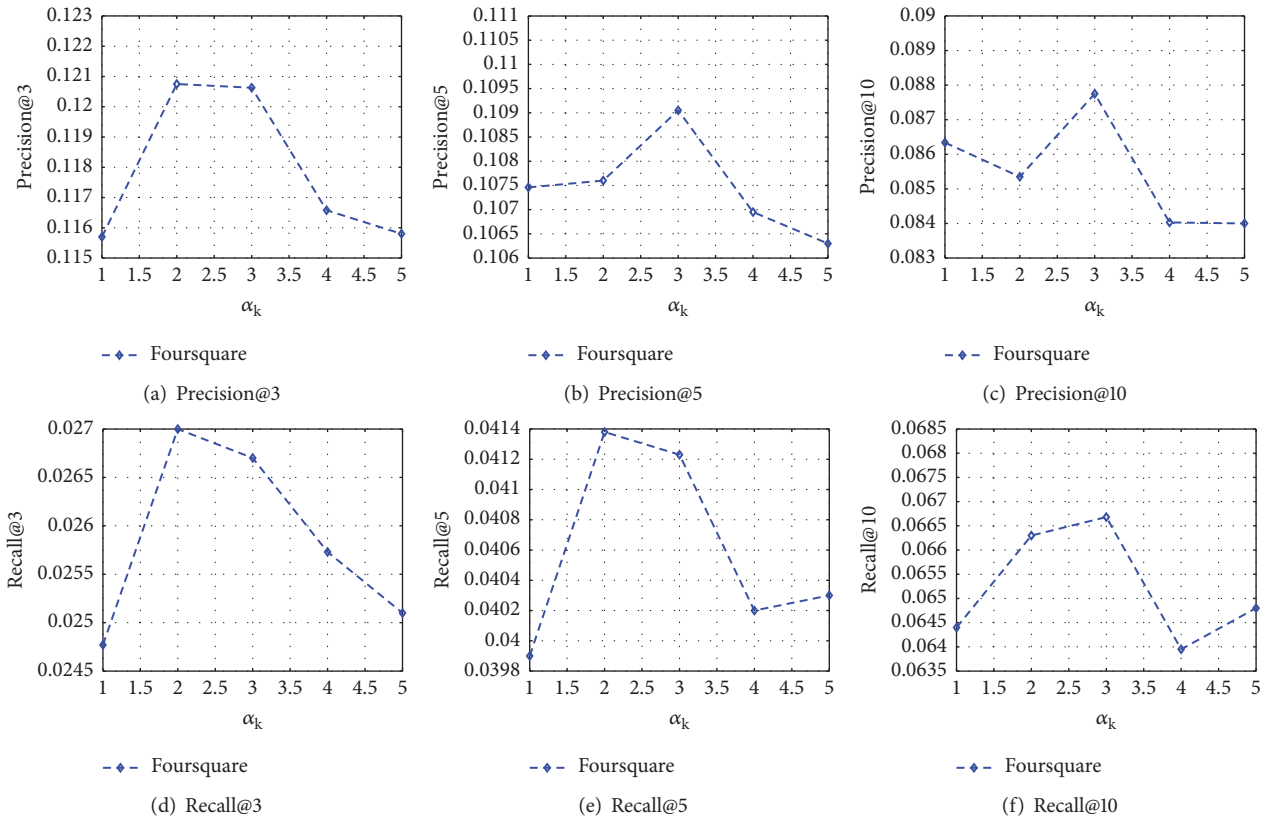
As indicated in Figure 2, our proposed neural personalized model is sensitive to the number of negative samples. The recommendation quality firstly improves as the number of negative samples increases and then degrades as the number of negative samples further increases. This indicates that insufficient or too many negative samples may hurt the recommendation performance of NBPFM. NBPFM achieves the best performance when the number of negative samples is around 8.

5.6.3. Impact of Parameters α_k and β_k. In our proposed method, parameters α_k and β_k control the shapes and scales of the Gamma distributions, which are used to initialize the user embedding matrix and the item embedding matrix. We perform another group of experiments to evaluate the sensitivities of α_k and β_k, by changing the values of α_k from 1 to 5 given $\beta_k = 0.2$, or varying the values of β_k from 0.1 to 0.5 given $\alpha_k = 2$. The experimental results of NBPFM with respect to different α_k and β_k on Foursquare are plotted in Figures 3 and 4, respectively.

As we can see, both α_k and β_k significantly affect the performance of our proposed recommendation model. In the case of α_k with the fixed value 2, NBPFM performs best when β_k is around 0.2, further reducing or increasing the value of β_k leads to worse performance. In the case of fixing the value of β_k to be 0.2, NBPFM shows the similar trends; i.e., all the evaluation metrics firstly move upwards and then begin to drop down, when α_k surpasses a certain threshold. This observation indicates that NBPFM is also sensitive to the initializations of the user embedding matrix and the item embedding matrix, which initially encode the user preferences and the item characteristics, respectively.

6. Conclusion and Future Work

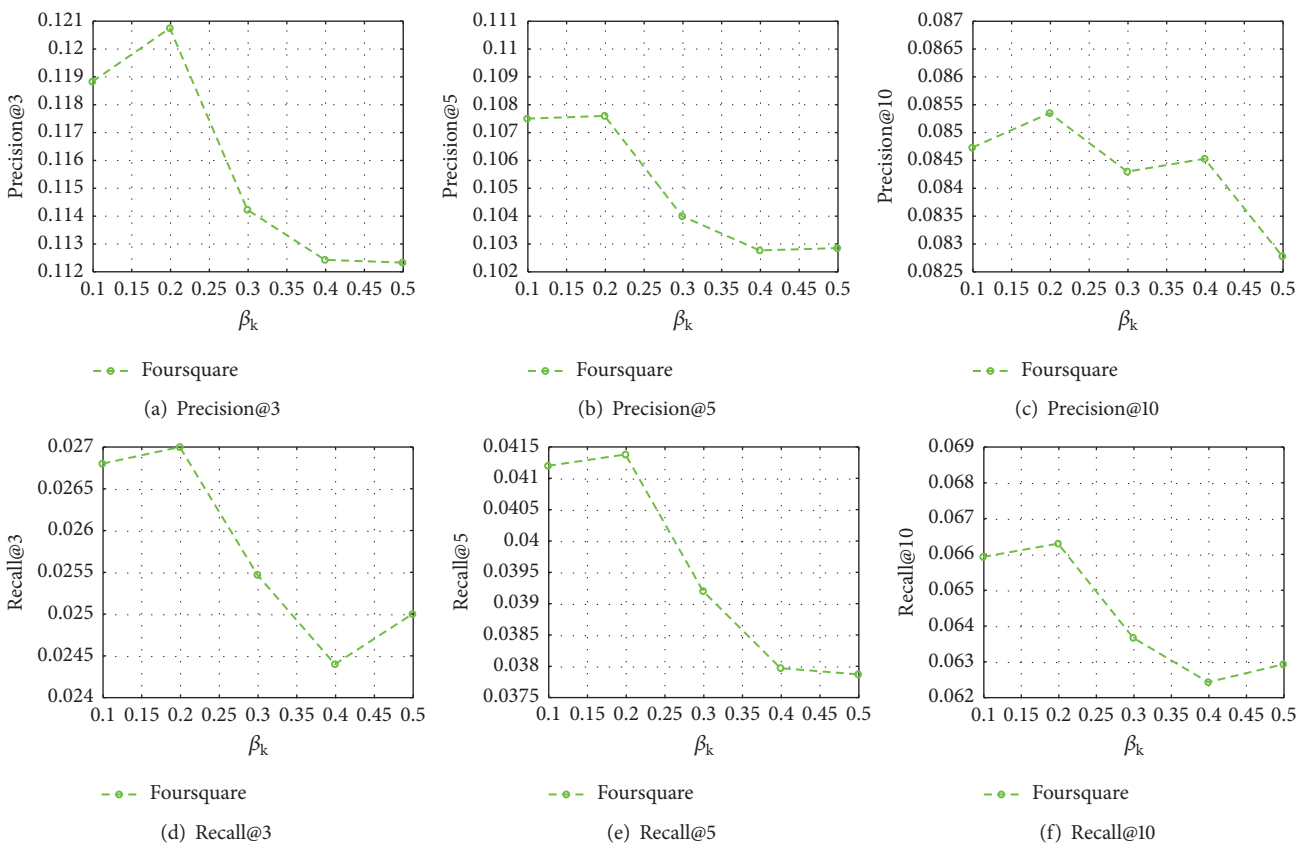
In this paper, we propose a neural personalized ranking model for collaborative filtering with implicit frequency feedback, which integrates the ranking-based poisson factor model with the neural networks. We firstly develop a ranking-based Poisson factor model, which combines the Poisson factor model and the Bayesian personalized ranking to model sparse implicit frequency feedback. The ranking-based Poisson factor model adopts a pair-wise learning method to learn the rankings of preferences between items. Then, we utilize



(a) Precision@3 (b) Precision@5 (c) Precision@10

(d) Recall@3 (e) Recall@5 (f) Recall@10

Figure 3: The impact of α_k.



(a) Precision@3 (b) Precision@5 (c) Precision@10

(d) Recall@3 (e) Recall@5 (f) Recall@10

Figure 4: The impact of β_k.

the neural networks to further extend the ranking-based Poisson factor model, and propose a neural personalized ranking model to capture the complex structure of user-item interactions. The neural personalized ranking model leverages the multilayer perceptron to learn the nonlinear user-item interaction relationships and endows the linear ranking-based Poisson factor model with a high level of nonlinearities. Experimental results on two real-world datasets show that our proposed method outperforms the state-of-the-art recommendation algorithms.

We only infer users' preferences and items' characteristics from users' implicit feedback. Since auxiliary information, such as social relationships and user reviews, is beneficial to recommendation algorithms, we plan to integrate these auxiliary information into our proposed neural personalized ranking model to boost the recommendation performance. In addition, recent advances in deep learning, e.g., attention mechanism, graph convolutional neural network, and generative adversarial network, have shown great potential in the fields of natural language processing and computer vision. Hence, applying the above deep learning techniques to recommender systems would be an interesting direction.

Data Availability

The ZIP data used to support the findings of this study are available at http://www.ntu.edu.sg/home/gaocong/datacode.htm. The ZIP data is publicly available and can be directly downloaded from http://www.ntu.edu.sg/home/gaocong/data/poidata.zip. The description of datasets is presented in Section 5.1 of our manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the Natural Science Foundation of the Higher Education Institutions of Jiangsu Province (Grant no. 17KJB520028), NUPTSF (Grant no. NY217114), Tongda College of Nanjing University of Posts and Telecommunications (Grant no. XK203XZ18002), and Qing Lan Project of Jiangsu Province.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] J. S. Breese, D. Heckerman, and C. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, UAI, Morgan Kaufmann Publishers Inc, pp. 43–52, 1998.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *The Computer Journal*, vol. 42, no. 8, pp. 30–37, 2009.

[4] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *NIPS, NIPS'07*, pp. 1257–1264, 2007.

[5] R. Pan, Y. Zhou, B. Cao et al., "One-class collaborative filtering," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, pp. 502–511, Pisa, Italy, December 2008.

[6] Y. Hu, C. Volinsky, and Y. Koren, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, pp. 263–272, ICDM, Pisa, Italy, December 2008.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '09)*, pp. 452–461, AUAI Press, Montreal, Canada, June 2009.

[8] T. Zhao, J. McAuley, and I. King, "Leveraging social connections to improve personalized ranking for collaborative filtering," in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, CIKM 2014*, pp. 261–270, ACM, China, November 2014.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural Collaborative Filtering," in *WWW, International World Wide Web Conferences Steering Committee*, pp. 173–182, 2017.

[10] H. Ma, C. Liu, I. King, and M. R. Lyu, "Probabilistic factor models for web site recommendation," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'11*, pp. 265–274, ACM, China, July 2011.

[11] P. Gopalan, L. Charlin, and D. M. Blei, "Content-based recommendations with Poisson factorization," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014*, pp. 3176–3184, Canada, December 2014.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 285–295, ACM, 2001.

[14] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[15] G.-R. Xue, C. Lin, Q. Yang et al., "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pp. 114-21, ACM, Salvador, Brazil, 2005.

[16] Y. Yu, C. Wang, Y. Gao, L. Cao, and X. Chen, "A Coupled Clustering Approach for Items Recommendation," in *Advances in Knowledge Discovery and Data Mining*, vol. 7819, pp. 365–376, PAKDD, Springer, Berlin, Germany, 2013.

[17] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information and System Security*, vol. 22, no. 1, pp. 89–115, 2004.

[18] T. Hofmann, "Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis," *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, pp. 259–266, 2003.

[19] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine learning (ICML '07)*, vol. 227, pp. 791–798, ACM, Corvallis, Ore, USA, June 2007.

[20] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *NIPS*, pp. 556–562, 2001.

[21] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 426–434, ACM, New York, NY, USA, August 2008.

[22] N. Srebro, J. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," *NIPS*, pp. 1329–1336, 2005.

[23] W. Pan and L. Chen, "GBPR: Group preference based bayesian personalized ranking for one-class collaborative filtering," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*, pp. 2691–2697, AAAI Press, China, August 2013.

[24] Y. Zhao, L. Nie, X. Wang, and T. Chua, "Personalized Recommendations of Locally Interesting Venues to Tourists via Cross-Region Community Matching," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, pp. 1–26, 2014.

[25] S. Zhang, L. Yao, and A. Sun, *Deep Learning Based Recommender System: A Survey And New Perspectives*, https://arxiv.org/abs/1707.07435.

[26] H.-T. Cheng, L. Koc, J. Harmsen et al., "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS 2016*, pp. 7–10, ACM, USA.

[27] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017*, pp. 355–364, ACM, Japan, August 2017.

[28] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the the 24th International Conference*, pp. 111-112, WWW '15 Companion, Florence, Italy, May 2015.

[29] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proceedings of the 9th ACM International Conference on Web Search and Data Mining, WSDM 2016*, pp. 153–162, ACM, USA, February 2016.

[30] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys 2016*, pp. 233–240, ACM, USA, September 2016.

[31] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017*, pp. 185–194, ACM, Japan, August 2017.

[32] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, pp. 1725–1731, AAAI Press, Australia, August 2017.

[33] S. Rendle, "Factorization machines," in *Proceedings of the 10th IEEE International Conference on Data Mining, ICDM 2010*, pp. 995–1000, Australia, December 2010.

[34] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, pp. 3119–3125, AAAI Press, Australia, August 2017.

[35] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS '06)*, pp. 153–160, Mass, USA, December 2006.

[36] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103, ACM, July 2008.

[37] F. Strub and J. Mary, "Collaborative filtering with stacked denoising autoencoders and sparse inputs in," *NIPS workshop on machine learning for eCommerce*, 2015.

[38] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015*, pp. 811–820, ACM, Australia, October 2015.

[39] M. Chen, Z. Xu, K. Q. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *ICML*, pp. 1627–1634, 2012.

[40] H. Wang, N. Wang, and D. Yeung, "Collaborative Deep Learning for Recommender Systems," in *Proceedings of the the 21th ACM SIGKDD International Conference*, pp. 1235–1244, Sydney, NSW, Australia, August 2015.

[41] P. Vincent, H. Larochelle, I. Lajoie, and P. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[42] Y. LeCun, K. Kavukcuoglu, C. Farabet et al., "Convolutional networks and applications in vision," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2010, pp. 253–256, IEEE, Paris, France, June 2010.

[43] G. Yuyun and Z. Qi, "Hashtag recommendation using attention-based convolutional neural network," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI 2016*, pp. 2782–2788, USA, July 2016.

[44] R. He and J. McAuley, "VBPR: Visual Bayesian personalized ranking from implicit feedback," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 144–150, USA, February 2016.

[45] A. Van den Oord and S. B. Dieleman, "Deep content-based music recommendation," *NIPS*, pp. 2649–2651, 2013.

[46] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What Your Images Reveal: Exploiting Visual Contents for Point-Of-Interest Recommendation," in *International World Wide Web Conferences Steering Committee (WWW)*, pp. 391–400, Perth, Australia, April 2017.

[47] L. Zheng, V. Noroozi, and P. S. Yu, "Joint Deep Modeling of Users and Items Using Reviews for Recommendation," in *Proceedings of the the Tenth ACM International Conference*, pp. 425–434, WSDW, ACM, Cambridge, UK, Feburary 2017.

[48] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, *Session-Based Recommendations with Recurrent Neural Networks*, ICLR, 2016.

[49] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in NetEase," in *Proceedings of the 32nd IEEE International Conference on Data Engineering, ICDE 2016*, pp. 1218–1229, IEEE, Finland, May 2016.

[50] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proceedings of the 11th ACM Conference on Recommender Systems, RecSys 2017*, pp. 130–137, ACM, Italy, August 2017.

[51] K. Georgiev and P. Nakov, "A non-IID framework for collaborative filtering with Restricted Boltzmann Machines," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, pp. 1148–1156, USA, June 2013.

[52] J. Wang, L. Yu, W. Zhang et al., "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017*, pp. 515–524, ACM, Japan, August 2017.

[53] L. Nie, S. Yan, M. Wang, R. Hong, and T.-S. Chua, "Harvesting visual concepts for image search with complex queries," in *Proceedings of the 20th ACM International Conference on Multimedia*, pp. 59–68, ACM, Japan, November 2012.

[54] L. Nie, M. Wang, Z. Zha, and T. Chua, "Oracle in Image Search," *ACM Transactions on Information and System Security*, vol. 30, no. 2, pp. 1–23, 2012.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, pp. 1–15, 2014.

[56] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of Dimensionality Reduction in Recommender System - A Case Study," WebKDD Workshop, 2000.

Hindawi
Complexity
Volume 2019, Article ID 9345861, 11 pages
https://doi.org/10.1155/2019/9345861

Research Article

Focal CTC Loss for Chinese Optical Character Recognition on Unbalanced Datasets

Xinjie Feng, Hongxun Yao, and Shengping Zhang

Harbin Institute of Technology, China

Correspondence should be addressed to Hongxun Yao; h.yao@hit.edu.cn

Received 15 September 2018; Revised 3 December 2018; Accepted 19 December 2018; Published 2 January 2019

Guest Editor: Li Zhang

In this paper, we propose a novel deep model for unbalanced distribution Character Recognition by employing focal loss based connectionist temporal classification (CTC) function. Previous works utilize Traditional CTC to compute prediction losses. However, some datasets may consist of extremely unbalanced samples, such as Chinese. In other words, both training and testing sets contain large amounts of low-frequent samples. The low-frequent samples have very limited influence on the model during training. To solve this issue, we modify the traditional CTC by fusing focal loss with it and thus make the model attend to the low-frequent samples at training stage. In order to demonstrate the advantage of the proposed method, we conduct experiments on two types of datasets: synthetic and real image sequence datasets. The results on both datasets demonstrate that the proposed focal CTC loss function achieves desired performance on unbalanced datasets. Specifically, our method outperforms traditional CTC by 3 to 9 percentages in accuracy on average.

1. Introduction

Recently, Deep Convolutional Neural Networks (DCNN) have achieved great success in various computer vision tasks, such as image classification and object detection [1–6]. Such success is supposed to contribute to large-scale data, dropout [7], and regularization [8–12] techniques. Image sequence recognition, which can be regarded as a variant of object detection, still remains challenging due to the difficulty in detection sequence-like objects. Different from classification and detection problems which predicts one label for an entire image or a region, sequence recognition is required to compute a sequence of labels for an input image, as shown in Figure 1.

In such cases, we cannot readily apply Deep Convolutional Neural (DCNN) Networks [13, 14] to sequence-like recognition tasks since DCNN can only generate label sequences in fixed lengthes depending on the input sequences. This limitation constrains its application in scenes that require to predict sequences of various length.

Traditional methods including [15, 16] are based on a detection-recognition strategy. Individual characters are firstly detected and then recognized to form a full sentence. However, detecting a single character is challenging especially for Chinese words. Different from English, a lot of Chinese words are composed of structural parts in left-right order. This phenomenon restricts the application of detection-recognition methods.

A commonly used method is by overcutting the input sequence into slices and recognizing them through recurrent neural networks (RNN). Compared with the aforementioned detection-recognition methods, this method cuts them into many slices before feeding them into a RNN based recognition model. Due to the great power of remembering past information, it is not required to locate characters for RNN models. The final results are predicted by fusing memories into current state information. There is another challenge for Chinese image-based sequence recognition, i.e., the unbalance of training set. Different from small lexicon language datasets, large lexicon language datasets, such as Chinese, suffer from severe unbalanced sample distribution. Most words except for the small part are rarely used in everyday scenes. In this paper, we refer to the commonly used samples as *easy samples* and others as *hard samples*.

Existing methods for sequence recognition can be classified into two branches: seq2seq fashion [17, 18] and CTC

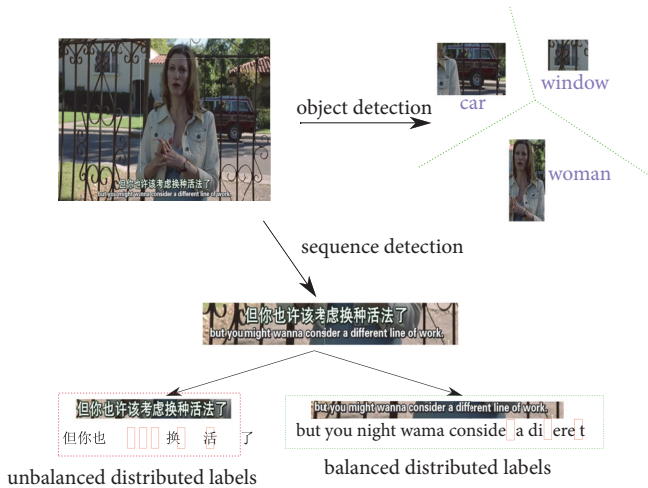


Figure 1: Distinct from object detection which aims to locate and recognize important objects, sequence detection is required to output a sequence of labels in various length. However, the prediction could be challenging if the distribution of labels is unbalanced.

loss function based models [19]. None of the above works take unbanlanced datasets into consideration, especially in Chinese image-based sequence recognition tasks. The unbanlance of a dataset will result in severe overfitting for easy samples and underfitting for hard samples. In order to remedy the unblance problem between easy and hard samples during training, we propose focal CTC loss function to prevent the model from forgetting to train the hard samples. To the best of our knowledge, this is the first work attempting to solve the unbalance problem for sequence recognition.

2. Related Work

2.1. Text Recognition Based on Manually Designed Image Features. Previous work mainly focuses on developing a powerful character classifier with manually designed image features. Wang proposed a HoG feature with random ferns developed for character classification in [20]. Neumann proposed new oriented strokes for character detection and classification in [21]. Manually designed image features always are confined to low-level which limits the performance. Lee developed a mid-level representative of characters with a discriminative feature pooling in [22]. Yao developed a mid-level feature named Strokelets to describe the parts of characters in [23]. Other interesting works brought insightful ideas by proposing to embed word images in a common vectorial subspace and convert word recognition into a retrieval problem [24, 25]. Some works take advantages of RNN which extract a set of geometrical or iamge features from handwritten texts into a sequence of image features [26]. Some other approaches [27] treat scene text recognition as an image classification problem and assign a class label to each English word (90K words in total).

2.2. CTC Loss Function Based Sequence Recognition. Connectionist Temporal Classification (CTC) is proposed in [19], which presents a CTC loss function to train RNNs to label unsegmented sequences directly. CTC is widely used for speech recognition [28, 29]. In this paper, we focus on applying CTC in image-based sequence recognition applications. Graves proposed the first attempt to combine recurrent neural networks and CTC for off-line handwriting recognition in [30]. After the revival of neural networks, deep convolutional neural networks have been devoted to image-based sequence recognition. Hasan applies Bidirectional Long-Short Term Memory (BLSTM) architecture with CTC to recognize printed Urdu texts [31]. Shi proposed a novel neural network architecture, which integrates feature extraction, sequence modeling, and transcription into a unified framework [32, 33]. Deep TextSpotter [34] trains both text detection and recognition in a single end-to-end pass. He developed a Deep-Text Recurrent Network (DTRN) that regards scene text reading as a sequence labeling problem [35].

2.3. Seq2seq Based Sequence Recognition. seq2seq and attention frameworks are prevalent in machine translation research [36]. Recently, such frameworks are adopted for image-based sequence recognition. Ba proposed a deep recurrent neural network trained with reinforcement learning to attend to the most relevant regions of an input image and the model learns to both localize and recognize multiple objects despite being given only class labels during training [37]. Lee presents recursive recurrent neural networks with attention modeling for lexicon-free optical character recognition in natural scene images [18]. Xu introduced an attention based model that automatically learns to describe the content of images [38]. Another interesting work is Spatial Transformer Networks [39], which introduce a new learnable module, the Spatial Transformer. It explicitly allows the spatial manipulation of data within the network. Focal loss was proposed to address class imbalance by reshaping the standard cross entropy loss such that it downweights the loss

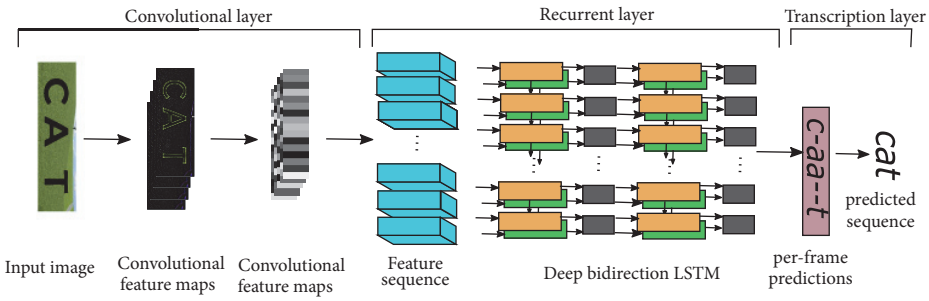


Figure 2: The network architecture. The architecture consists of three parts: convolutional layers, which extract a feature sequence from the input image; recurrent layers, which predict a label distribution for each frame; transcription layer, which translates the per-frame predictions into the final label sequence.

assigned to well-classified examples in an object detection framework [40]. Lee presents recursive recurrent neural networks with attention modeling (R2AM) for lexicon-free optical character recognition in natural scene images [41].

3. Architecture

We design our model by extending the architecture proposed by [32], which consists of three components: convolutional, recurrent, and transcription layers. The overview of the architecture is illustrated in Figure 2. We adopt the Residual Network in [42], which contains 51 layers as the convolutional layers and bidirectional Long-Short Term Memory [43–45] as the recurrent layers. The transcription layer is mainly based on CTC or our focal CTC function.

Similar to many CNN based Computer Vision applications, the convolutional layers are employed for generating feature maps of the given image. As has been already demonstrated by some detection literatures, such as Fast-RCNN and Faster-RCNN, it is possible to locate and recognize objects with feature maps from a pretrained CNN model. Hence, this strategy can be readily adopted by image-based sequence recognition task. Specifically, we first overcut the feature maps into multiple slices. Each slice can be regarded as a representation of a bounding box. These slice feature maps form a fix-length sequence. Then we feed this sequence into a RNN-based layer to predict variable-length label sequence. Note that the deep structure of ResNet is able to provide enough receptive field for a small area that cover the corresponding character. So RNN is not necessary for image-based sequence recognition mission. Despite this fact, we still utilize LSTM to predict label sequences due to its excellent ability in retaining and discarding previous information. The transcription layer is responsible for translating the output of hidden unit of LSTM to labels and finding the label sequence that has the highest probability conditioned on the per-slice predictions. We employ a fully connection layer to interface with the output of LSTM hidden units. Then we calculate the probability of each label through a softmax layer. Finally, we calculate the CTC loss through our focal loss based CTC layer with the predicted and ground truth label sequences as input.

3.1. Feature Extraction. In order to obtain a suitable representation of a given image, many CV models use pretrained CNN to generate feature maps. In fact, one of the advantages for CNNs over other traditional feature extracting methods is that they can capture local textures by different filters. For this reason, a well-trained CNN model can be readily adopted by image sequence recognition tasks. In our case, we use Residual Network, known as ResNet, to extract visual feature maps. It is first proposed by He et al. in [42] to solve classification problems and won 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. Compared with other previously proposed deep CNN models, ResNet has deeper layers but low complexities. The elegant performance of ResNet should be contributed to the introduction of deep residual learning framework. We briefly illustrate this structure in Figure 3. The formulation of residual learning can be viewed as inserting shortcut connections into a plain feed-forward network. In fact, shortcut connections simply perform identity mapping and their outputs are added to the outputs of the following layers (Figure 3). Supposing that the input of residual learning block is denoted as X, the learning process can be formulated as optimising a residual function $F(x)$:

$$F(X) \coloneqq H(X) - X \tag{1}$$

where $H(\cdot)$ denotes an underlying mapping to be fit by several stacked layers. Note that the size of X should be consistent with the output of $H(\cdot)$.

In our experiments, the Residual Network consists of four bottleneck blocks: $9(layers) \times 16(filters)$, $12(layers) \times 32(filters)$, $18(layers) \times 64(filters)$, and $9(layers) \times 128(filters)$. Between two connected bottleneck blocks, a 1×1 filter shortcut is inserted to align the filter dimension.

We take the output of the last CNN layer of ResNet as feature maps that correspond to the entire image. We overcut this feature map into slices. Similar to Fast RCNN or Faster

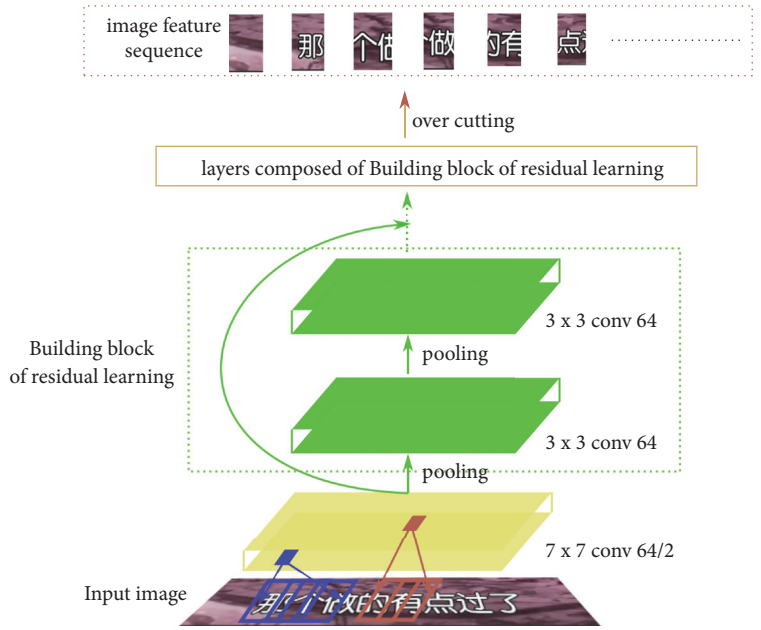


Figure 3: Convolutional layers (ResNet) which are used to extract image feature sequences. The basic building block is residual learning unit, surrounded by the green dash box.

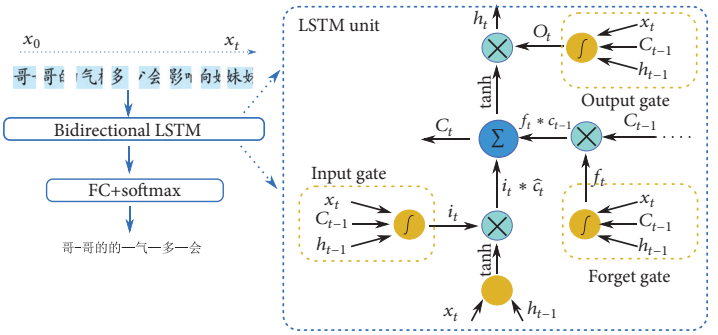


Figure 4: A schematic illustration of a LSTM neuron. Each LSTM neuron has an input gate, a forget gate, and an output gate.

RCNN, each slice contains information of a local area of the original image.

3.2. Label Sequence Prediction. Based on feature maps of image slices, we predict label sequence based on a bidirectional LSTM network. RNN [43] is a class of neural networks for efficient modeling dynamic temporal behavior of sequences through directed cyclic connections between units. Each unit is able to retain internal hidden states which are considered to contain information of previous ones. Generally speaking, RNN can be viewed as an extension of hidden Markov models. In spite of the advantages in dealing with sequential signals, traditional RNN unit suffers from the vanishing gradient problem[46], which limits the range of context it can store and thus makes training process difficult. To solve this issue, Long-Short Term Memory(LSTM), a variant of RNN, is proposed. It is capable of capturing long and short term temporal dependencies than traditional RNN unit. Specifically, LSTM extends RNN by adding three gates to the RNN neuron: a forget gate f controlling to what extent the current information is supposed to be retained; an input gate i to decide how much effect the current input should have on the hidden state; an output gate o to constrain the information of current memory which is available to output the hidden state. These gates enable LSTM to solve long-term dependency problems in sequence recognition tasks. More importantly, LSTM is easier to optimize as these gates help the input signal to effectively propagate through the recurrent hidden states $h(t)$ without affecting the output. Figure 4 is a schematic illustration of a LSTM unit. LSTM also alleviates the gradient vanishing or exploding issues of RNN [47]. In our case, we formulate the operation of a LSTM unit in (8a). For convenience, we omit the indication of forward or backward layers.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2a}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2b}$$

$$\widetilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2c}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2d}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{2e}$$

$$h_t = o_t * \tanh\left(C - t\right) \tag{2f}$$

where σ is the sigmoid activation function, which calculates probabilities for all gates. f_t, i_t, o_t represent the forget gate, input gate, and output gate at tth step, respectively. C_t, C_{t-1} store information of current and last cell state. h_t, h_{t-1} represent the hidden units of the two successive steps. W_* and b_* are the weights and bias, which transform two vectors into one common space. In the literature [48], rectified linear units (ReLU) are also employed as the activation function.

In our case, the label sequence is predicted through a Bidirectional LSTM. The size of hidden unit is 128. Each label is calculated by fusing hidden state of forward and backward hidden layers. At the tth step, h_t^f and h_{T-t}^b are combined through a concatenate layer followed by a full-connect layer. The final results, which take the form of probability distributions, are obtained through a softmax layer.

4. Transcription

Transcription is used to convert the predictions of each slice made by the Bidirectional LSTM into a label sequence. In this section, we first briefly review the definition of CTC loss and then introduce our proposed focal CTC loss. The CTC loss function, which is first proposed in [19], aims to model the conditional probability of label sequences given probability distribution of each predicted label. In essence, a CTC layer is supposed to be a kind of loss functions rather than a network layer. For this reason, the terminology of *CTC* layer is not accurate and may lead to misunderstandings about CTC. The focal CTC loss function is mainly inspired by the focal strategy in object detection applications. The main contribution of this paper is that by employing focal strategy, CTC loss function can be more effective in optimising the entire model.

4.1. Probability of Label Sequences. Let $\mathbb{R}$ and $\mathbb{L}$ be a real number set and a label set, respectively, which are always named as lexicon. Let $\mathcal{X} = \mathbb{R}^{m \times t}$ be the feature space of the input, and $\mathcal{Y} = \mathbb{L}^s$ be the label space, where superscripts m, t, s represent feature dimension, sequence time, and label length, respectively. Following the previous method, the input is overcut into slices. Each slice is supposed to contain a fraction of some single label characters, implying $t \geq s$. CTC loss function can be regarded as modeling the joint probability distribution over $\mathcal{X}$ and $\mathcal{Y}$, which is denoted as $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$.

A CTC loss function has an input of a softmax layer [49]. We add a blank label $\mathbf{B}$ to $\mathbb{L}$ and hence obtain a new label $\mathbb{L}^+ = \mathbb{L} \cup \{\mathbf{B}\}$. An input sequence $\mathbf{x} \in \mathbb{R}^{m \times T}$ is transformed to another sequence $\mathbf{y} \in \mathbb{L}^{+T}$ through the softmax layer. We denote activation of output unit k at time t as y_k^t. Then y_k^t is interpreted as the probability of observing label k at time t, which defines a distribution over the set $\mathbb{L}^{+T}$ of length T sequences of the lexicon $\mathbb{L}^+ = \mathbb{L} \cup \{\mathbf{B}\}$. Reference [19] refers to the elements of $\mathbb{L}^{+T}$ as paths and denotes them as π. We assume that the distribution of the outputs of the network is conditionally independent. Then the probability of path π can be expressed as follows:

$$P\left(\pi \mid x\right) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{3}$$

A sequence-to-sequence mapping function $\mathcal{B}$ is defined on sequence $\pi \in \mathbb{L}^{+T}$. $\mathcal{B}$ maps π onto l by firstly removing the repeated and blank labels. For example, $\mathcal{B}$ maps "**B**1**BB**1**B**220" onto "1120". The conditional probability is then calculated through summing probabilities of all π that are mapped by $\mathcal{B}$ onto l:

$$P\left(l \mid \mathbf{y}\right) = \sum_{\pi:\mathcal{B}(\pi)=l} p\left(\pi \mid \mathbf{y}\right) \tag{4}$$

The time complexity of the naive way to compute the conditional probability of (4) is exponential as $sizeof(\mathbb{L}^+)^T$ paths exist. Reference [19] provides an efficient dynamic programming algorithm to compute the conditional probability. The CTC loss is obviously differentiable since the conditional probability $P(l \mid \mathbf{y})$ only contains addition and multiplication operations.

4.2. Focal CTC Loss. In [40], the cross entropy of focal loss is defined as follows:

$$FL\left(p_t\right) = -\alpha_t \left(1 - p_t\right)^\gamma \log\left(p_t\right) \tag{5}$$

where p_t is the probability of ground truth in the softmax output distribution. α and γ are hyperparameters used to balance the loss. An intuitive understanding of the focal loss is that it can be viewed as multiplying cross entropy by $\alpha_t(1 - p_t)^\gamma$ (the minis belong to cross entropy: $-\log(p_t)$). It is easy to find that the closer p_t approaches to 1, the smaller the focal loss will be. So the focal loss will reduce the effect of examples but pay more attention to hard negative samples during training.

With the focal theory, we redefine our focal CTC loss as follows:

$$F\_CTC\left(l \mid \mathbf{y}\right) = -\alpha_t \left(1 - P\left(l \mid \mathbf{y}\right)\right)^\gamma \log\left(P\left(l \mid \mathbf{y}\right)\right) \tag{6}$$

where $(P(l \mid \mathbf{y})$ is the conditional probability mentioned above. The negative log function converts the optimization process from a maximization problem to a minimization problem in order to adopt gradient decent algorithm. In this way, we can focus our loss on undertrained samples and "ignore" overtrained samples.

5. Evaluation

5.1. Datasets. In this section, we evaluate the focal CTC loss on both synthetic and real datasets. We establish two synthetic datasets by concatenating 5 MNIST [13] images which are resized to 32×32 in y-axis to a long image with a resolution of 32×160. We split the alphabet "$0-9a-z$" into two subalphabets "$0-9a-h$" and "$i-z$" with the same size. The

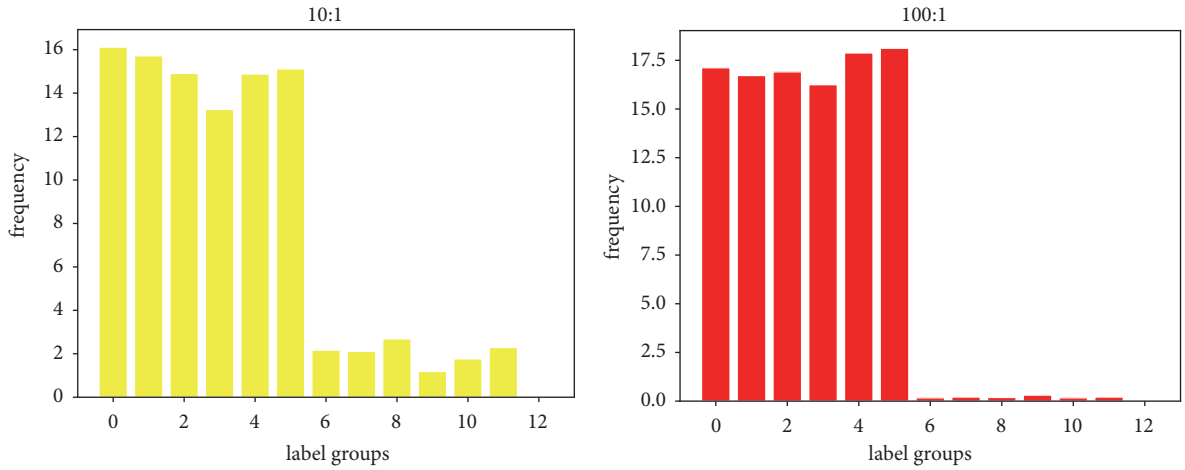


Figure 5: We provide distribution of labels for both two synthetic datasets. Each bar represents the frequency of two characters. For example, the first bar of (a) represents how many 0 and 1 exist in dataset with unbalance ratio 10 : 1.

first dataset with a unbalance ratio of 10:1 consists of two parts, one containing 1,000,000 long images which are concatenated by 5 images randomly sampling from "0 − 9a − h" and the other one containing 100,000 long images concatenated by 5 images randomly from "i − z". The second dataset with an unbalance ratio of 100:1 consists of 1,000,000 long images which are concatenated by 5 images randomly sampling from "0 − 9a − h" and 10,000 long images concatenated by 5 images randomly from "i − z". We use a dataset containing 10,000 images to test the accuracy. The ratio of high-frequency and low-frequency characters we used in training phrase is set to be 1 : 1. We present label distribution of both datasets in Figure 5.

We also test the focal CTC loss on a real Chinese-ocr dataset [50], which consists of 3,607,567 training and 5,000 test samples. Each of them is a 32 × 280 pixel image with a 10- -Chinese-character label. The frequencies of words are illustrated in Figure 6.

5.2. Training Strategy Evaluation Metrics. We implement our focal loss function in tensorflow framework, which is known as a flexible architecture supporting complex computations in machine learning and deep learning. A typical CTC loss function can be formulated as follows:

$$f_{CTC\_loss}(l, y, s) = -\log(P(l \mid \mathbf{y})) \tag{7}$$

where l and y denote label sequences from ground truth and output by RNN units, respectively. Both are constrained in length by an integer scalar s. This function has already been implemented in Tensorflow framework. So we first easily compute $P(l \mid y)$ by calling (7) defined in TF. Then we calculate focal loss according to (5). We summarize the entire process as follows:

$$ctc\_loss = f_{CTC_{loss}}(l, y, s) \tag{8a}$$

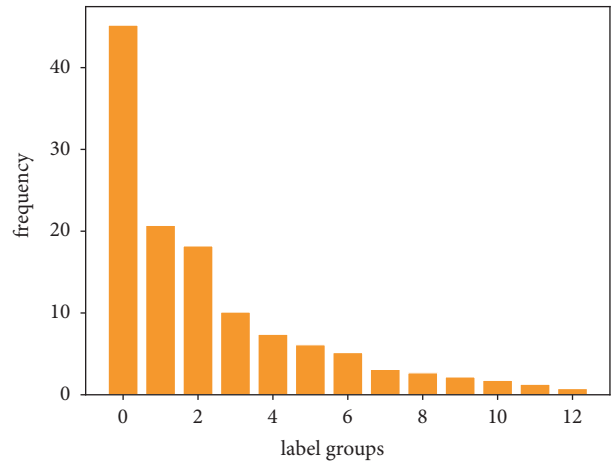


Figure 6: We provide distribution of labels for Chinese-ocr dataset. The first bar represents the most 300 frequently used words in dataset. Obviously, most Chinese words only account for a small part of all words.

$$p = e^{-ctc\_loss} \tag{8b}$$

$$focal_{loss} = \alpha \times (1 - p)^{\gamma} \times ctc\_loss \tag{8c}$$

Before we train our model, we set the learning rate and batch size to be 0.001 and 128, respectively. All parameters except CNN are initialized by sampling from a Gaussian distribution. The weights of CNN are copied from ResNet and kept unchanged during training. We optimise our model using stochastic gradient descent (SGD) with Nesterov momentum [51] set to be 0.9. We run all the experiments on a single NVIDIA M40 GPU. The entire training process is described in Algorithm 1.

We evaluate of our model in terms of two metrics: the naive accuracy and soft_accuracy. The naive one means that the predicted sequence can only be recognized as positive when it is just the same as the ground truth. The soft_accuracy

Require Parameters of RNN: Θ, learning rate $\tau = 0.001$, batch size $B = 128$

1: **for** $i = 1; i < max\_iteration; i++$ **do**
2: $V = getBatch(train\_setm, B)$;
3: $logits = Forward(V)$;
4: $ctc\_loss = tf.nn.ctc\_loss(labels, logits)$
5: $p = tf.exp(-ctc\_loss)$
6: $focal\_loss = \alpha \times (1-p)^{\gamma} \times ctc\_loss$
7: $\nabla focal\_loss(\Theta) = Backward(V, focal\_loss)$;
8: $\Theta^{i} = \Theta^{i-1} - \tau\left(\frac{1}{B}\nabla focal\_loss\,(\Theta)\right)$
9: **end for**

Algorithm 1: Focal CTC implemented in tensorflow.

Table 1: Dataset with unbalance ratio 100:1.

α	γ	accuracy	HF	LF
CTC		**0.538**	**0.739**	**0.337**
0.99	0.5	0.587	0.753	0.421
0.99	1	0.531	0.765	0.296
0.99	2	0.511	0.742	0.280
0.75	**0.5**	**0.628**	**0.755**	**0.501**
0.75	1	0.538	0.709	0.368
0.75	2	0.501	0.704	0.297
0.5	0.5	0.525	0.741	0.310
0.5	1	0.500	0.731	0.269
0.5	2	0.504	0.722	0.287
0.25	***0.5***	***0.614***	***0.731***	***0.498***
0.25	1	0.590	0.728	0.451
0.25	2	0.508	0.685	0.331

Table 2: Dataset with unbalance ratio 10:1.

α	γ	accuracy	HF	LF
CTC		**0.657**	**0.711**	**0.603**
0.99	0.5	0.667	0.721	0.613
0.99	1	0.636	0.729	0.543
0.99	2	0.703	0.745	0.662
0.75	0.5	0.684	0.709	0.659
0.75	1	0.641	0.715	0.567
0.75	2	0.631	0.716	0.546
0.5	0.5	0.667	0.741	0.593
0.5	1	0.680	0.722	0.638
0.5	2	0.682	0.728	0.635
0.25	***0.5***	***0.723***	***0.753***	***0.694***
0.25	**1**	**0.724**	**0.751**	**0.697**
0.25	2	0.707	0.763	0.650

refers to tolerating an edit distance of 1 from prediction to label. The edit distance between two sequences p and q is defined as the minimum number of insertions, substitutions, and deletions required to change p into q.

5.3. Results. Results for various α and γ are shown in Tables 1 and 2 for the synthetic dataset with unbalanced ratios 100:1 and 10:1, respectively. The best gains of the two datasets, as highlighted in bold in Tables 1 and 2, are 9% and 6.7%, respectively. Moreover, the focal CTC not only improves the Low-frequency accuracy, which is a natural result, but also enhances the High-frequency accuracy in the 10:1 dataset. The improvement for 100:1 dataset is mainly due to the enhancement of Low-frequency data. However some choices

Table 3: Real dataset.

α	γ	accuracy	soft_accuracy
CTC		**0.723**	**0.926**
0.99	0.5	0.730	0.933
0.99	**1**	**0.764**	**0.946**
0.99	2	0.631	0.913
0.75	0.5	0.692	0.918
0.75	1	0.724	0.926
0.75	2	0.704	0.930
0.5	0.5	0.733	0.936
0.5	1	0.655	0.908
0.5	2	0.641	0.913
0.25	***0.5***	***0.759***	***0.937***
0.25	1	0.690	0.926
0.25	2	0.667	0.919

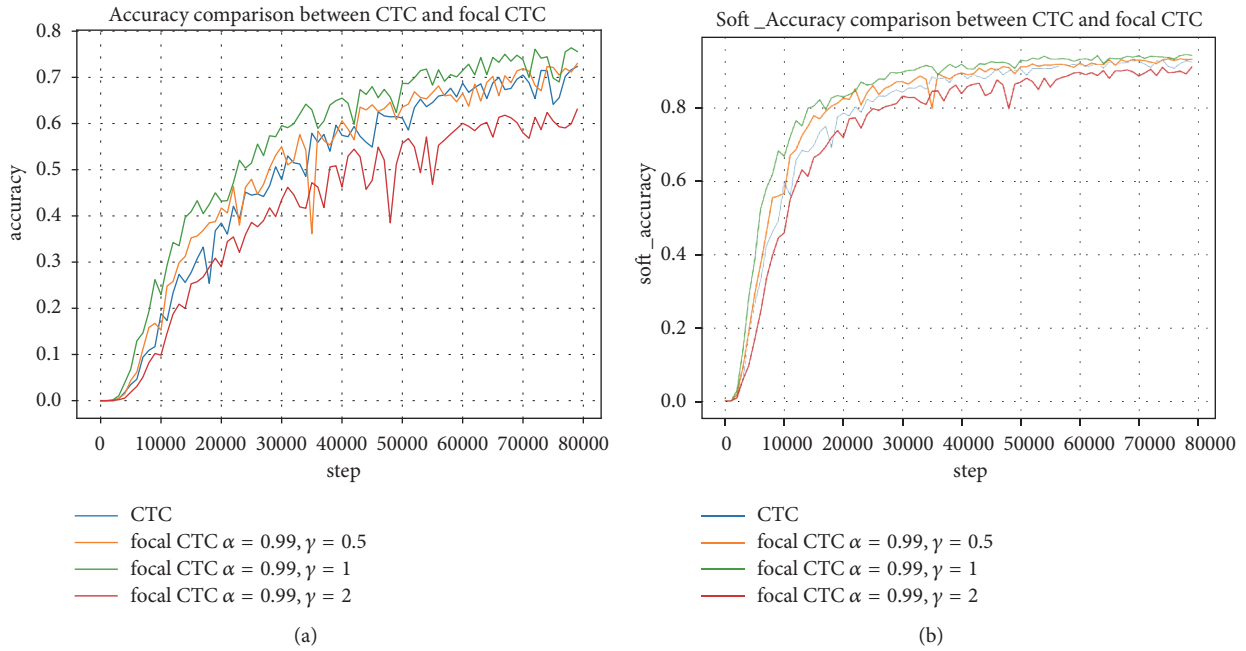


Figure 7: The comparison of convergence speed for different γ of real data with the same $\alpha = 0.99$.

of α and γ perform poor bad such as $\alpha = 0.5, \gamma = 2$ $\alpha = 0.75, \gamma = 2$ and $\alpha = 0.5, \gamma = 1$ for 100:1 dataset. For $\alpha = 0.75$, accuracy of High-frequency drops dramatically. As for $\alpha = 0.5$, accuracy of Low-frequency is not so good. The accuracy of 10:1 dataset achieves promising results for both High-frequency and Low-frequency samples. Similarly, the same issue occured in 100:1 dataset. In addition, some choices of α and γ also result in poor performance such as $\alpha = 0.99, \gamma = 1$ $\alpha = 0.75, \gamma = 1$, and $\alpha = 0.75, \gamma = 2$.

However, a bad choice of α and γ would impair the accuracy as finding that $\alpha = 0.25, \gamma = 0.5$ achieves an exiting promotion of at least 5% on both datasets, despite the existence of both High-frequency and Low-frequency data. We highlighted these results in bold italic font in Tables 1 and 2.

We present the results on the real dataset in Table 3. The best improvement of accuracy, as highlighted in bold, is 4.1%. This is an exciting improvement for a real life application. Additionally, we observe that $\alpha = 0.25, \gamma = 0.5$ makes a promotion of 3.6% which is also a considerable enhancement.

In order to observe the convergence situation for different γ, we plot the test Accuracy and Soft_Accuracy by changing curve of the real dataset for $\alpha = 0.99$. We can see that, for all training process, the focal CTC loss of $\gamma = 0.5$ achieves the best convergence ratio for both the Accuracy and Soft_Accuracy, as shown in Figure 7. The focal CTC loss of $\gamma = 2$ performs bad all the time.

With the above results on both the synthetic and real datasets, we can conclude that the focal CTC loss with $\alpha = 0.25$, and $\gamma = 0.5$ gives a considerable improvement compared with the CTC loss. Some other choices of α and γ may achieve more considerable enhancement. So in real

Figure 8: We present prediction results of Chinese word in (a). We also show some examples of synthetic datasets with unbalance ratios 10 : 1 and 100 : 1 in (b) and (c), respectively.

life applications, we can choose $\alpha = 0.25$ and $\gamma = 0.5$ for unbalanced datasets.

5.4. Qualitative Results. We provide some examples of both synthetic and real datasets in Figure 8. The sequences predicted by CTC or focal CTC are marked in red and green, respectively. All images are sampled from test split. Generally, the prediction is obviously improved with focal CTC loss employed. Due to the extreme unbalance of distribution in Chinese words, it can be seen from Figure 8(a) that some uncommonly used words can not effectively be detected by CTC based model but by our proposed focal CTC based model. As for synthetic dataset, it is interesting that CTC and focal CTC both work well for 10:1 dataset. However, the performance of CTC drops in case that we make the distribution of characters more unbalanced.

6. Conclusion

In this paper, we propose a focal CTC loss function, which can balance the loss between easy and hard samples during training. We test various hyperparameters α and γ on both the synthetic and real datasets. The results show that setting $\alpha = 0.25$ and $\gamma = 0.5$ achieves a considerable improvement for both the synthetic and real dataset. Besides, we also point out that some choices may result in bad performance. To some extent, our proposed focal CTC loss function alleviates the unbalance of big lexicon sequence recognition.

Data Availability

The datasets used in the experiment are from previously reported studies and datasets, which have been cited.

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 the Science and Technology Funding of China (no. 61772158 and no. 61472103) and the Science and Technology Funding Key Program of China (no. U1711265).

References

[1] L. Zhang, A. A. Mohamed, R. Chai, B. Zheng, and S. Wu, "Automated deep-learning method for whole-breast segmentation in diffusion-weighted breast mri," in *Medical Imaging*, SPIE, 2019.

[2] L. Zhang, R. Chai, S. W. Dooman Arefan, and J. Sumkin, "Deep-learning method for tumor segmentation in breast dce-mri," in *Medical Imaging*, SPIE, 2019.

[3] D. Xie, L. Zhang, and L. Bai, "Deep learning in visual computing and signal processing," *Applied Computational Intelligence and Soft Computing*, vol. 2017, Article ID 1320780, 13 pages, 2017.

[4] Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang, "Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 4761–4772, USA, July 2017.

[5] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proceedings of the 23rd IEEE International Conference on Image Processing, ICIP 2016*, pp. 3708–3712, Phoenix, AZ, USA, September 2016.

[6] Y. Qi, S. Zhang, L. Qin et al., "Hedging deep features for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[8] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 58, no. 1, pp. 267–288, 1996.

[9] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data*, Springer Series in Statistics, Springer, New York, NY, USA, 2011.

[10] N. M. Nasrabadi, "Pattern Recognition and Machine Learning," *Journal of Electronic Imaging*, vol. 16, no. 4, p. 049901, 2007.

[11] A. Christmann and D.-X. Zhou, "On the robustness of regularized pairwise learning methods based on kernels," *Journal of Complexity*, vol. 37, pp. 1–33, 2016.

[12] Abhishake and S. Sivananthan, "Multi-penalty regularization in learning theory," *Journal of Complexity*, vol. 36, pp. 141–165, 2016.

[13] 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.

[14] 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.

[15] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR '12)*, pp. 3304–3308, November 2012.

[16] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "PhotoOCR: Reading text in uncontrolled conditions," in *Proceedings of the 2013 14th IEEE International Conference on Computer Vision, ICCV 2013*, pp. 785–792, Australia, December 2013.

[17] Y. Deng, A. Kanervisto, and A. M. Rush, "What you get is what you see: a visual markup decompiler," 2016, https://arxiv.org/abs/1609.04938v1.

[18] C.-Y. Lee and S. Osindero, "Recursive recurrent nets with attention modeling for OCR in the wild," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 2231–2239, USA, July 2016.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks," in *Proceedings of the ICML 2006: 23rd International Conference on Machine Learning*, pp. 369–376, USA, June 2006.

[20] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 1457–1464, IEEE, Barcelona, Spain, November 2011.

[21] L. Neumann and J. Matas, "Scene text localization and recognition with oriented stroke detection," in *Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV '13)*, pp. 97–104, December 2013.

[22] C.-Y. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu, "Region-based discriminative feature pooling for scene text recognition," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, pp. 4050–4057, USA, June 2014.

[23] X. Bai, C. Yao, and W. Liu, "A learned multi-scale representation for scene text recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4042–4049, 2014.

[24] J. Almazan, A. Gordo, A. Fornes, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[25] J. A. Rodriguez-Serrano, A. Gordo, and F. Perronnin, "Label Embedding: A Frugal Baseline for Text Recognition," *International Journal of Computer Vision*, vol. 113, no. 3, pp. 193–207, 2015.

[26] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[27] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[28] A. Hannun, C. Case, J. Casper et al., "Deep speech: Scaling up end-to-end speech recognition," https://arxiv.org/abs/1412.5567.

[29] D. Amodei, S. Ananthanarayanan, R. Anubhai et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, pp. 173–182, 2016.

[30] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems, NIPS 2008*, pp. 545–552, Canada, December 2008.

[31] A. Ul-Hasan, S. B. Ahmed, F. Rashid, F. Shafait, and T. M. Breuel, "Offline printed urdu nastaleeq script recognition with bidirectional LSTM networks," in *Proceedings of the 12th International Conference on Document Analysis and Recognition, ICDAR 2013*, pp. 1061–1065, USA, August 2013.

[32] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[33] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," CoRR abs/1507.05717, https://arxiv.org/abs/1507.05717.

[34] M. Busta, L. Neumann, and J. Matas, "Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework," in *Proceedings of the 16th IEEE International Conference on Computer Vision, ICCV 2017*, pp. 2223–2231, Italy, October 2017.

[35] P. He, W. Huang, Y. Qiao, C. C. Loy, and X. Tang, "Reading scene text in deep convolutional sequences," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 3501–3508, USA, February 2016.

[36] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," https://arxiv.org/abs/1409.0473.

[37] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," https://arxiv.org/abs/1412.7755.

[38] K. Xu, J. L. Ba, R. Kiros et al., "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, pp. 2048–2057, France, July 2015.

[39] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proceedings of the 29th Annual Conference on Neural Information Processing Systems, NIPS 2015*, pp. 2017–2025, Canada, December 2015.

[40] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, Venice, October 2017.

[41] C.-Y. Lee and S. Osindero, "Recursive recurrent nets with attention modeling for ocr in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 770–778, July 2016.

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

[44] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, no. 1, pp. 115–143, 2003.

[45] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural 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. 1127–1137, Beijing, China, July 2015.

[46] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 5, no. 2, pp. 157–166, 1994.

[47] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, pp. 2347–2355, USA, June 2013.

[48] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 8609–8613, May 2013.

[49] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing (Les Arcs, 1989)*, vol. 68, pp. 227–236, Neurocomputing, Springer, Berlin, Germany, 1990.

[50] Y. Chenguang, "chinese_ocr," 2018, https://github.com/YCG09/chinese_ocr.

[51] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On The Importance of Initialization And Momentum in Deep Learning," in *International Conference on Machine Learning*, pp. 1139–1147, June 2013.

Hindawi
Complexity
Volume 2018, Article ID 5676095, 8 pages
https://doi.org/10.1155/2018/5676095

Research Article

Multimodal Deep Feature Fusion (MMDFF) for RGB-D Tracking

Ming-xin Jiang[1,2] **Chao Deng**[3] **Ming-min Zhang**[4,5] **Jing-song Shan**[6] **and Haiyan Zhang**[6]

[1]*Jiangsu Laboratory of Lake Environment Remote Sensing Technologies, Huaiyin Institute of Technology, Huaian, 223003, China*
[2]*Faculty of Electronic information Engineering, Huaiyin Institute of Technology, Huaian, 223003, China*
[3]*School of Physics & Electronic Information Engineering, Henan Polytechnic University, Jiaozuo, 454000, China*
[4]*School of Computer Science & Technology, Zhejiang University, 310058, China*
[5]*Institute of VR and Intelligent System, Hangzhou Normal University, 310012, China*
[6]*Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian, 223003, China*

Correspondence should be addressed to Chao Deng; dengchao_hpu@163.com and Ming-min Zhang; zhangmm95@zju.edu.cn

Received 22 July 2018; Accepted 5 November 2018; Published 28 November 2018

Guest Editor: Li Zhang

Visual tracking is still a challenging task due to occlusion, appearance changes, complex motion, etc. We propose a novel RGB-D tracker based on multimodal deep feature fusion (MMDFF) in this paper. MMDFF model consists of four deep Convolutional Neural Networks (CNNs): Motion-specific CNN, RGB- specific CNN, Depth-specific CNN, and RGB-Depth correlated CNN. The depth image is encoded into three channels which are sent into depth-specific CNN to extract deep depth features. The optical flow image is calculated for every frame and then is fed to motion-specific CNN to learn deep motion features. Deep RGB, depth, and motion information can be effectively fused at multiple layers via MMDFF model. Finally, multimodal fusion deep features are sent into the C-COT tracker to obtain the tracking result. For evaluation, experiments are conducted on two recent large-scale RGB-D datasets and results demonstrate that our proposed RGB-D tracking method achieves better performance than other state-of-art RGB-D trackers.

1. Introduction

As one of the most fundamental problems in computer vision, visual object tracking, which aims to estimate the trajectory of a object in a video, has been successfully addressed in numerous applications, including intelligent traffic control, artificial intelligence, and autonomous driving [1–3]. Despite the research on visual object tracking has made outstanding achievements, many challenges remain when seeking to track objects effectively in practice. For example, it is still quite difficult to track objects when occlusion occurs frequently, appearance changes, the motion of object is complex, and illumination varies [4–6].

The major drawback of the tracking methods only using RGB data is that they are not robust to appearance changes. Thanks to the availability of consumer-grade RGB-D sensors, such as Inter RealSense, Microsoft Kinect, and Asus Xtion, more accurate depth information of objects can be obtained to revisit the existing problems of tracking [7, 8]. Compared with RGB data, the RGB-D data can remarkably improve the performance of tracking due to the access to the depth information complementary to RGB [9]. The depth information is invariant to illumination or color variations [10, 11] and can provide geometrical cues and spatial structure information; thus it shows powerful benefits in visual object tracking. However, how to effectively utilize the depth data provided by RGB-D sensors is still a challenging issue.

However, RGB and depth only encode static information from a single frame so that tracking often fails when the motion of object is complex. Under these circumstances, deep motion features can provide high-level motion information to distinguish the target object [12–14]. The dynamic information can be captured by deep motion features, and that will be complementary to static features extracted from RGB and depth image.

Our motivation is to design a RGB-D object tracker based on multimodal deep feature fusion. Specific emphasis of this paper is placed on exploring three scientific questions: how to fuse deep motion features with static features provided by RGB and depth image; how to fuse the deep RGB and depth features sufficiently; how to effectively derive geometrical cues and spatial structure information from depth data. In summary, the key technical contributions of this study are three-fold:

(i) A novel MMDFF model is designed for RGB-D tracking, including four deep CNN: motion-specific CNN, RGB- specific CNN, Depth-specific CNN, and RGB-Depth correlated CNN. In MMDFF, we can fuse RGB, depth, and motion information at multiple layers via CNN effectively. The proposed method can separately extract RGB and depth features by using RGB-specific CNN and Depth-specific CNN and adequately exploit the correlated relationship between RGB and depth modality by using RGB-Depth correlated CNN. At the same time, Motion-specific CNN can provide an important high-level information about motion for tracking.

(ii) The depth image is encoded into three channels: horizontal disparity, height above ground, and angle with gravity. Then, the three channel images are sent into depth-specific CNN to extract deep depth features. The strong correlation between RGB and depth modality can be learnt by RGB-Depth correlated CNN. In contrast to only using depth information as one channel in many existing RGB-D trackers, we can obtain more useful information for tracking, such as geometrical features and spatial structure information, by encoding the depth image into three channels.

(iii) To evaluate the performance of our proposed RGB-D tracker, we conduct extensive experiments on the two recent challenging RGB-D benchmark datasets: the large-scale Princeton RGB-D Tracking Benchmark (PTB) Dataset [13] and the University of Birmingham RGB-D Tracking Benchmark (BTB) [15]. The experimental results show that our proposed approach is superior to the state-of-the-art RGB-D trackers.

The remainder of this paper is organized as follows. The related work is discussed in Section 2. Section 3 describes the overview and the details of our proposed method. In Section 4, we demonstrate experimental results to evaluate our proposed RGB-D tracker. We conclude our work in Section 5.

2. Related Work

2.1. RGB-D Object Tracking. With the emergence of RGB-D sensors, there has been great interest in visual object tracking using RGB-D data [15–17] to improve tracking performance since depth modality can provide useful information complementary to RGB modality.

A RGB-D tracking method using depth scaling kernelised correlation filters and occlusion handling was proposed in [18]. In [19], authors used Haar-like features and HOG features based on RGB and depth to form a boosting tracking approach. Zheng et al. [20] presented an object tracker based on sparse representation of depth image. Hannuna et al. proposed a RGB-D tracker built upon the KCF tracker, they exploit depth information to handle scale changes, occlusions, and shape changes.

Although the existing RGB-D trackers have made great contributions to promote RGB-D tracking, most of them used hand-crafted features and fused simply RGB and depth information, ignored the strong correlation between RGB and depth modality.

2.2. Deep RGB Features. Owing to the superiority in feature extraction, CNN has been increasingly used in RGB trackers [21, 22]. The CNN includes a number of convolutional layer, pooling layer, and fully connected layer; the features at different layers have different properties. The higher layers can capture the senmantic features and the lower layers can capture the spatial features, and they are both important in the tracking problem [23, 24].

In [25], Song et al. proposed the CREST algorithm, which treated the correlation filter as one convolutional layer and applied residual learning to capture appearance changes. C-COT was presented in [26], which employed an implicit interpolation model to solve the learning problem in the continuous spatial domain. Zhu et al. [27] proposed a tracker named UCT, which designed an end-to-end framework to learn the convolutional features and perform the tracking process simultaneously.

All these trackers only considered that RGB appearance deep features in current frame cannot benefit from geometrical features extracted from depth image and interframe dynamic information. Thus, it is important to get rid of this problem by fusing deep features from RGB-D data and deep motion features.

2.3. Deep Depth Features. In the recent years, deep depth features have received a lot of attention in object recognition [28, 29], object detection [30, 31], indoor semantic segmentation [32, 33], etc. Unfortunately, few existing RGB-D trackers use CNN to extract deep depth features to improve the tracking performance. In [34], Jiang et al. proposed a RGB-D tracker based on cross-modality deep learning, in which Gaussian-Bernoulli deep Boltzmann Machines (DBMs) were adopted to extract the features of RGB and depth image. As far as the drawbacks of DBMs are concerned, one of the most important ones is the high computational cost of inference.

In recent years, CNN has witnessed great success in computer vision. In this context, we will focus on how to use CNN to fuse deep depth feature with deep RGB and motion features effectively for RGB-D tracker.

2.4. Deep Motion Features. Deep motion feature has been successfully applied for action recognition [35] and video classification [36], but it is rarely applied to visual tracking.

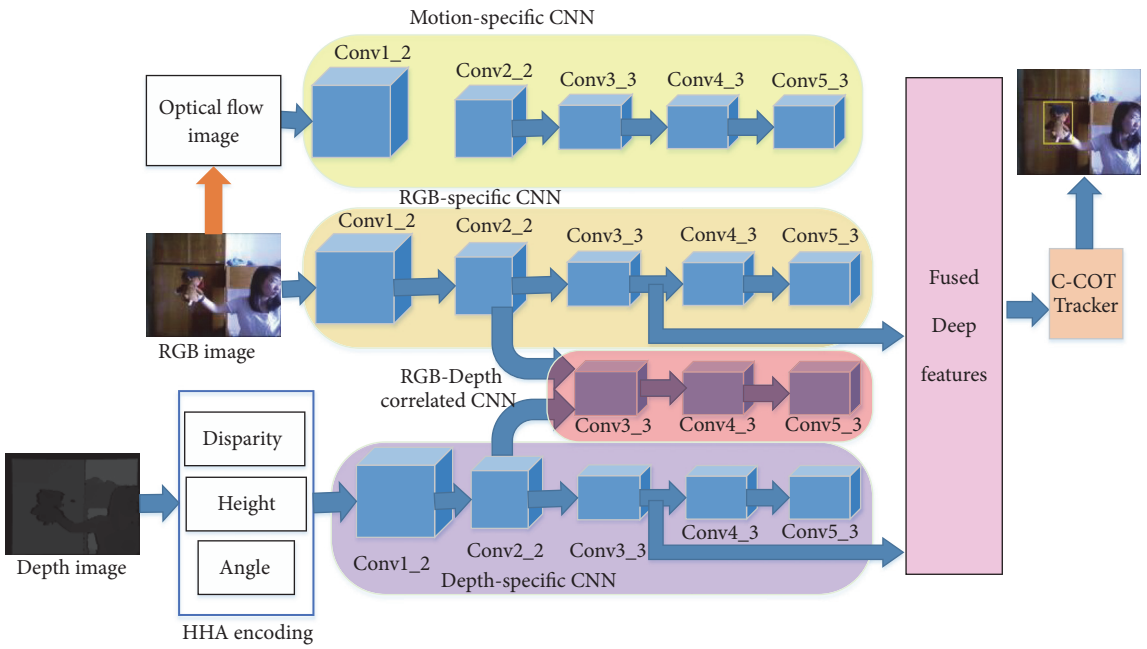


Figure 1: Our MMDFF model for RGB-D Tracking.

Most existing tracking methods only extract appearance features, ignore motion information. In [37], Zhu et al. proposed an end-to-end flow correlation tracker, which focuses on making use of the rich flow information in consecutive frames to improve the feature representation and the tracking accuracy. Danelljan [38] investigated the influence of deep motion features in a RGB tracker. But they have not taken depth information into account.

To the best of our knowledge, deep motion features are yet to be applied for RGB-D tracking. In this paper, we will discuss how to fuse deep motion features with RGB appearance features and geometrical features provided by depth image to improve the performance of RGB-D tracking.

3. RGB-D Tracking Based on MMDFF

3.1. Multimodal Deep Feature Fusion (MMDFF) Model. In this section, a novel MMDFF model is proposed for RGB-D tracking aiming at fusing deep motion features with static appearance and geometrical features provided by RGB and depth data. The overall architecture of our approach is illustrated as Figure 1, and our end-to-end MMDFF model is composed of four deep CNN: motion-specific CNN, RGB-specific CNN, depth-specific CNN, and RGB-Depth correlated CNN. A final fully connected (FC) fusion layer is proposed to effectively fuse deep RGB-D features and deep motion features; then the fused deep features are sent into the C-COT tracker, and the tracking result can be obtained at last.

As described in Section 1, CNN has been shown to significantly outperform traditional machine learning approaches in a wide range of computer vision tasks. It has been widely acknowledged that CNN features extracted from different layers play different important roles in the tracking. A lower layer captures spatial detailed features, which is helpful to precisely localize the target object; at the same time, a higher layer provides semantic features which is robust to occlusion and deformation. In our MMDFF model, we separately adopt hierarchical convolutional features extracted from RGB-specific CNN and depth-specific CNN. To be more specific, two independent CNN networks are adopted: the RGB-specific CNN is for RGB data and the depth-specific CNN is for depth features. And the features on Conv3-3 and Conv5-3 in the two CNNs are sent to the highest fusing FC layer in our experiments.

It is believed that more features extracted from different modalities can be helpful to accurately descript the objects and improve the tracking performance. As abovementioned, most of existing RGB-D trackers directly concatenate features extracted from RGB and depth modalities, not adequately exploiting the correlation between the two modalities. In our method, the strong correlation between RGB and depth modality can be learnt by RGB-Depth correlated CNN.

For human visual system, geometrical and spatial structure information plays an important role in tracking objects. In order to more explicitly derive geometrical and spatial structure information from depth data, we encode it into three channels: horizontal disparity, height above ground, and angle with gravity, using the encoding approach proposed in [39]. Then, the three channel image is sent into depth-specific CNN to extract deep depth features.

The optical flow image is calculated for every frame and then is fed to motion-specific CNN to learn deep motion features, which can capture high-level information about the movement of the object. A pretrained optical flow network provided by [13] is used as our motion-specific CNN,

Table 1: Comparison results: SR using different deep features fusions on the PTB dataset.

Deep Features	All SR	Target type			Target size		Movement		Occlusion		Motion type	
		human	animal	rigid	large	small	slow	fast	yes	no	passive	active
Only RGB	0.73	0.75	0.69	0.74	0.80	0.72	0.71	0.72	0.74	0.85	0.78	0.73
RGB+Depth	0.77	0.79	0.71	0.77	0.81	0.77	0.76	0.73	0.81	0.86	0.79	0.75
RGB+Depth + RGB-depth correlated	0.79	0.81	0.74	0.79	0.83	0.79	0.78	0.74	0.83	0.86	0.81	0.77
RGB+Depth + RGB-depth correlated +Motion	0.84	0.83	0.86	0.85	0.85	0.86	0.82	0.83	0.87	0.87	0.82	0.83

which is pretrained on the UCF101 dataset and includes five convolutional layers.

Thus far, we have obtained multimodal deep features to represent the rich information of the object, including the RGB, horizontal disparity, height above ground, angle with gravity, and motion. Next, we attempt to explore how to fuse the multimodal deep features using the CNN. To solve this problem, we conduct extensive experiments to evaluate the performance using different fusion schemes, each experiment fuses the multimodal deep features at different layer. Inspired by the working mechanism of human visual cortex in the brain which indicates the features should be fused in the high level, so we test fusing at several relatively high layers, such as pool 5, fc 6 and fc 7. We find that fusing the multimodal deep features from fc 6 and fc 7 can obtain better performance.

Let p_i^j represent feature map of j modalities and i denotes the spatial position, $j = \{1, 2, \ldots, J\}$. In our paper, $J = 4$ as we adopt the feature maps from Conv3-3 and Conv5-3 in RGB, depth, RGB-depth correlated, and motion modality. The fusing feature map P_i^{fusion} is the weighted sum of feature maps for three levels in four modalities,

$$P_i^{fusion} = \sum_{j=1}^{J} w_i^j p_i^j \tag{1}$$

where the weigh w_i^j can be computed as follows:

$$w_i^j = \frac{\exp\left(\mathrm{p}_i^j\right)}{\sum_{k=1}^{J} \exp\left(\mathrm{p}_i^k\right)} \tag{2}$$

3.2. C-COT Tracker. The multimodal fusion deep features are sent into the C-COT tracker, which was proposed in [26]. A brief review of the C-COT tracker will be provided in the following of this section, and we will use the same symbols as in [33], for convenience, and more detailed description and proofs can be found in [26].

The C-COT transfers multimodal the fusion feature map to the continuous spatial domain $t \in [0, \mathrm{T})$ by defining the interpolation operator $J_d : \mathbb{R}^{N_d} \longrightarrow L^2(\mathrm{T})$ as follows:

$$J_d\left\{x^d\right\}(t) = \sum_{n=0}^{N_d-1} x^d[n]\, b_d\left(t - \frac{T}{N_d} n\right) \tag{3}$$

The convolution operator is defined as

$$S_f\{x\} = f * J\{x\} - \sum_{d=1}^{D} f^d * J_d\left\{x^d\right\} \tag{4}$$

The objective function is

$$E(f) = \sum_{j=1}^{M} \alpha_j \left\| S_f\left\{x_j\right\} - y_j \right\|_{L^2}^2 - \sum_{d=1}^{D} \left\| \omega f^d \right\|_{L^2}^2 \tag{5}$$

Equation (5) can be minimized in the Fourier domain to learn the filter. The following can be obtained by applying Parseval's formula to (5)

$$E(f) = \sum_{j=1}^{M} \alpha_j \left\| \sum_{d=1}^{D} \widehat{f}^d X_j^d \widehat{b}_d - \widehat{y}_j \right\|_{L^2}^2 - \sum_{d=1}^{D} \left\| \widehat{\omega} * \widehat{f}^d \right\|_{L^2}^2 \tag{6}$$

The desired convolution output $\widehat{y}_j$ can be provide by the following expression:

$$\widehat{y}_j[k] = \frac{\sqrt{2\pi\sigma^2}}{T} \exp\left(-2\sigma^2 \left(\frac{\pi k}{T}\right)^2 - i\frac{2\pi}{T} u_j k\right) \tag{7}$$

4. Experimental Results

The experiments are conducted on two challenging benchmark datasets: BTB dataset [17] with 36 videos and PTB dataset [7] with 100 videos to test the proposed RGB-D tracker using MATLAB R2016b platform with the Caffe toolbox [40] on a PC with an Intel(R) Core(TM) i7-4712MQ CPU@3.40GHz (with 16G memory) and TITAN GPU (12.00 GB memory).

4.1. Deep Features Comparison. The impacts of deep motion features are evaluated by conducting different experiments on PTB dataset. Table 1 indicates the comparison results of different deep features fusions using success rate (SR) as measurements. From Table 1, we find that SRs are lowest when only using deep RGB feautures. SRs increase by fusing deep depth features and RGB-depth correlated features, especially when occlusion occurs. Further, the performance is obviously improved when deep motion features are added, especially when the movement is fast, motion type is active, and target size is small.

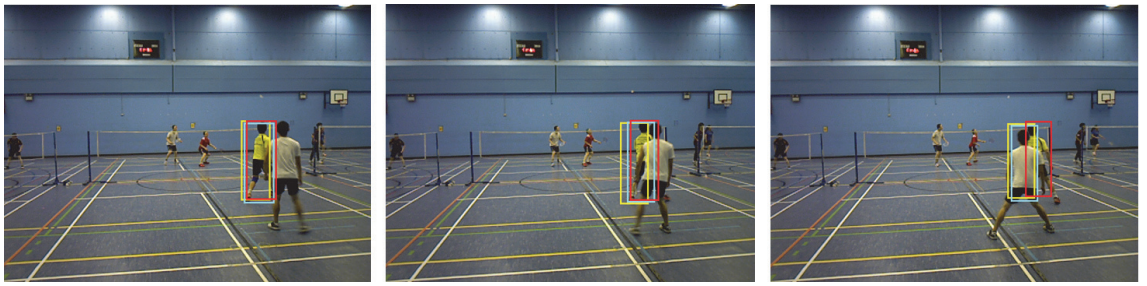


Figure 2: The comparison results on athlete_move video from BTB dataset.

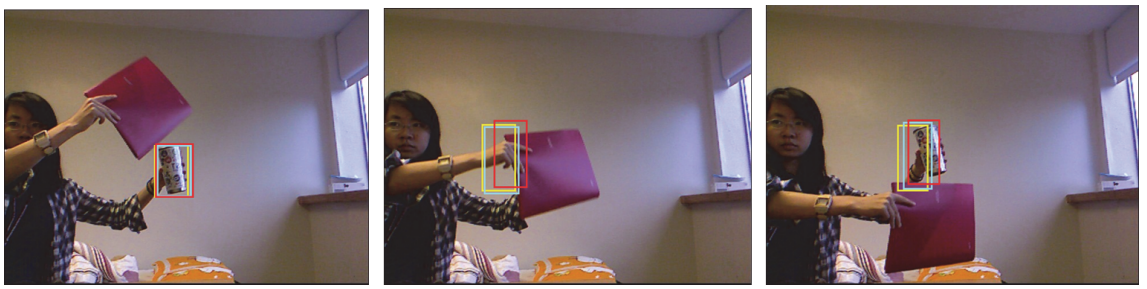


Figure 3: The comparison results on cup_book video from PTB dataset.

Figure 4: The comparison results on zballpat_no1 video from PTB dataset.

To intuitively show the contribution of fusing deep depth features and deep motion features for RGB-D tracking, in the following, we demonstrate part of experimental results on the BTB dataset and PTB dataset that only using deep RGB features obtains unsatisfactory performance. The tracking result only using deep RGB features is in yellow, fusing deep RGB and depth is in blue, adding deep motion features to RGB, and depth is in red.

In Figure 2, the major challenge of the athlete_move video is that the target object moves fast from left to right. As illustrated in Figure 3, the cup is fully occluded by the book. The zballpat_no1 sequence is challenging due to the change of moving direction (Figure 4). The deep motion features are able to improve the tracking performance as they can exploit the motion patterns of the target object.

4.2. Quantitative Evaluation. We present the results of comparing our proposed tracker with four state-of-the-art RGB-D trackers on the BTB dataset and PTB dataset: Prin Tracker [7] (2013), DS-KCF* Tracker [41] (2016), GBM Tracker [34] (2017), and Berming Tracker [17] (2018). We provide the results in terms of success rate (SR) and area-under-curve (AUC) as measurements.

Figure 5 shows the comparison results of SR of different trackers on the PTB dataset. The results illustrates that the overall SR of our tracker is 87%, SR is 86% when the object moves fast, and SR is 84% when the motion type is active. These values are higher than other trackers obviously, especially when the object moves fast.

Figure 6 indicates the AUC comparison results on the BTB dataset. The overall AUC of our tracker is 9.30, the AUC is 9.84 when the camera is stationary, and the AUC is 8.27 when the camera is moving.

From Figures 5-6, we can see that our tracker obtains the best performance, especially when the object is moving fast or the camera is moving. These results show that deep motion feature is helpful to improve the tracking performance.

5. Conclusion

We study the problems in the existing visual object tracking algorithms and find that existing trackers cannot benefit

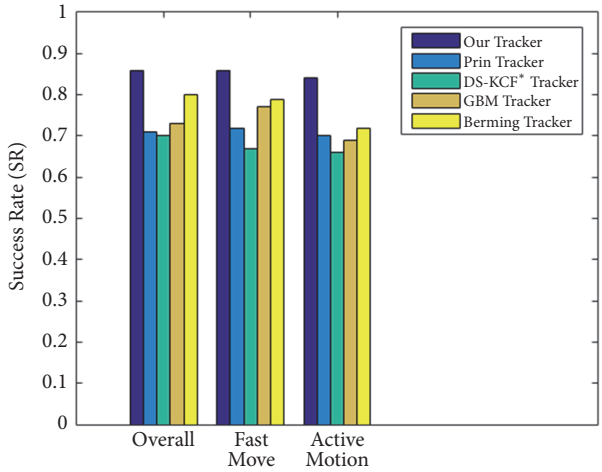


Figure 5: The comparison results of SR on the PTB dataset.

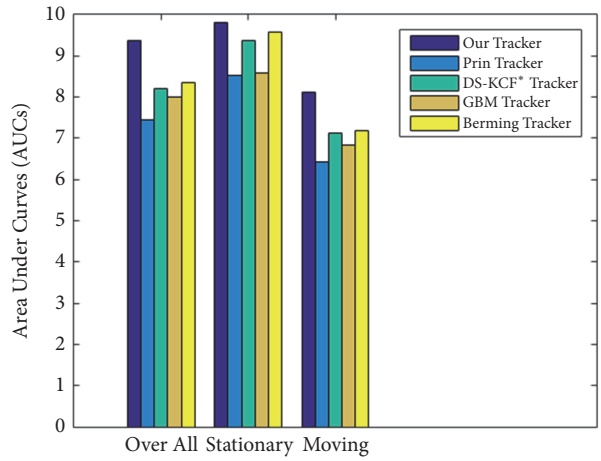


Figure 6: The comparison results of AUC on the BTB dataset.

from geometrical features extracted from depth image and interframe dynamic information by fusing deep RGB, depth, and motion information. We propose MMDFF model to solve these problems. In this model, RGB, depth and motion information are fused at multiple layers via CNN, the correlated relationship between RGB and depth modality exploited by using RGB-Depth correlated CNN. The experimental results show that deep depth feature and deep motion feature provide complementary information to RGB data and the fusing strategy promotes the tracking performance significantly, especially when occlusion occurs, the movement is fast, motion type is active, and target size is small.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D project under Grant no. 2017YFB1002803, Major Program of Natural Science Foundation of the Higher Education Institutions of Jiangsu Province under Grant no. 18KJA520002, Project funded by the Jiangsu Laboratory of Lake Environment Remote Sensing Technologies under Grant no. JSLERS-2018-005, Six-Talent Peak Project in Jiangsu Province under Grant no. 2016XYDXXJS-012, the Natural Science Foundation of Jiangsu Province under Grant no. BK20171267, 533 Talents Engineering Project in Huaian under Grant no. HAA201738, project funded by Jiangsu Overseas Visiting Scholar Program for University Prominent Young & Middle-aged Teachers and Presidents, and the fifth issue 333 high-level talent training project of Jiangsu province.

References

[1] S. Duffner and C. Garcia, "Using Discriminative Motion Context for Online Visual Object Tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 12, pp. 2215–2225, 2016.

[2] X. Zhang, G.-S. Xia, Q. Lu, W. Shen, and L. Zhang, "Visual object tracking by correlation filters and online learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017.

[3] E. Di Marco, S. P. Gray, and K. Jandeleit-Dahm, "Diabetes alters activation and repression of pro- and anti-inflammatory signaling pathways in the vasculature," *Frontiers in Endocrinology*, vol. 4, p. 68, 2013.

[4] M. Jiang, Z. Tang, and L. Chen, "Tracking multiple targets based on min-cost network flows with detection in RGB-D data," *International Journal of Computational Sciences and Engineering*, vol. 15, no. 3-4, pp. 330–339, 2017.

[5] K. Chen, W. Tao, and S. Han, "Visual object tracking via enhanced structural correlation filter," *Information Sciences*, vol. 394-395, pp. 232–245, 2017.

[6] M. Jiang, D. Wang, and T. Qiu, "Multi-person detecting and tracking based on RGB-D sensor for a robot vision system," *International Journal of Embedded Systems*, vol. 9, no. 1, pp. 54–60, 2017.

[7] S. Song and J. Xiao, "Tracking Revisited Using RGBD Camera: Unified Benchmark and Baselines," in *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV)*, pp. 233–240, Sydney, Australia, December 2013.

[8] N. An, X.-G. Zhao, and Z.-G. Hou, "Online RGB-D tracking via detection-learning-segmentation," in *Proceedings of the 23rd International Conference on Pattern Recognition, ICPR 2016*, pp. 1231–1236, Mexico, December 2016.

[9] D. Michel, A. Qammaz, and A. A. Argyros, "Markerless 3D human pose estimation and tracking based on RGBD cameras: An experimental evaluation," in *Proceedings of the 10th ACM International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2017*, pp. 115–122, Greece, June 2017.

[10] A. Wang, J. Cai, J. Lu, and T.-J. Cham, "Modality and component aware feature fusion for RGB-D scene classification," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 5995–6004, USA, July 2016.

[11] S. Hou, Z. Wang, and F. Wu, "Object detection via deeply exploiting depth information," *Neurocomputing*, vol. 286, pp. 58–66, 2018.

[12] A. Dosovitskiy, P. Fischery, E. Ilg et al., "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 2758–2766, Chile, December 2015.

[13] G. Gkioxari and J. Malik, "Finding action tubes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 759–768, USA, June 2015.

[14] S. Karen and Z. Andrew, "Two-Stream Convolutional Networks for Action Recognition in Videos [C]," *Andrew Z. Two-Stream Convolutional Networks for Action Recognition in Videos [C]. NIPS*, 2014.

[15] L. Spinello, M. Luber, and K. O. Arras, "Tracking people in 3D using a bottom-up top-down detector," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1304–1310, Shanghai, China, May 2011.

[16] K. Meshgi, S.-I. Maeda, S. Oba, H. Skibbe, Y.-Z. Li, and S. Ishii, "An occlusion-aware particle filter tracker to handle complex and persistent occlusions," *Computer Vision and Image Understanding*, vol. 150, pp. 81–94, 2016.

[17] J. Xiao, R. Stolkin, Y. Gao, and A. Leonardis, "Robust Fusion of Color and Depth Data for RGB-D Target Tracking Using Adaptive Range-Invariant Depth Models and Spatio-Temporal Consistency Constraints," *IEEE Transactions on Cybernetics*, 2017.

[18] M. Camplani, S. Hannuna, M. Mirmehdi et al., "Real-time RGB-D Tracking with Depth Scaling Kernelised Correlation Filters and Occlusion Handling," in *Proceedings of the British Machine Vision Conference 2015*, pp. 145.1-145.11, Swansea.

[19] M. Luber, L. Spinello, and K. O. Arras, "People tracking in RGB-D data with on-line boosted target models," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems: Celebrating 50 Years of Robotics, IROS'11*, pp. 3844–3849, USA, September 2011.

[20] W.-L. Zheng, S.-C. Shen, and B.-L. Lu, "Online Depth Image-Based Object Tracking with Sparse Representation and Object Detection," *Neural Processing Letters*, vol. 45, no. 3, pp. 745–758, 2017.

[21] H. Li, Y. Li, and F. Porikli, "DeepTrack: learning discriminative feature representations online for robust visual tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1834–1848, 2016.

[22] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9914, pp. 850–865, 2016.

[23] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 3119–3127, Chile, December 2015.

[24] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 3074–3082, Chile, December 2015.

[25] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. H. Lau, and M.-H. Yang, "CREST: Convolutional Residual Learning for Visual Tracking," in *Proceedings of the 16th IEEE International Conference on Computer Vision, ICCV 2017*, pp. 2574–2583, Italy, October 2017.

[26] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, "Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking," in *Computer Vision – ECCV 2016*, vol. 9909 of *Lecture Notes in Computer Science*, pp. 472–488, Springer International Publishing, Cham, 2016.

[27] Z. Zhu, G. Huang, W. Zou, D. Du, and C. Huang, "UCT: Learning Unified Convolutional Networks for Real-Time Visual Tracking," in *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pp. 1973–1982, Venice, October 2017.

[28] Y. Cheng, X. Zhao, K. Huang, and T. Tan, "Semi-supervised learning and feature evaluation for RGB-D object recognition," *Computer Vision and Image Understanding*, vol. 139, pp. 149–160, 2015.

[29] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pp. 681–687, Germany, October 2015.

[30] J. Hoffman, S. Gupta, J. Leong, S. Guadarrama, and T. Darrell, "Cross-modal adaptation for RGB-D detection," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation, ICRA 2016*, pp. 5032–5039, Sweden, May 2016.

[31] X. Xu, Y. Li, G. Wu, and J. Luo, "Multi-modal deep feature learning for RGB-D object detection," *Pattern Recognition*, vol. 72, pp. 300–313, 2017.

[32] Y. Cheng, R. Cai, Z. Li, X. Zhao, and K. Huang, "Locality-sensitive deconvolution networks with gated fusion for RGB-D indoor semantic segmentation," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 1475–1483, USA, July 2017.

[33] S. Lee, S.-J. Park, and K.-S. Hong, "RDFNet: RGB-D Multi-level Residual Feature Fusion for Indoor Semantic Segmentation," in *Proceedings of the 16th IEEE International Conference on Computer Vision, ICCV 2017*, pp. 4990–4999, Italy, October 2017.

[34] M. Jiang, Z. Pan, and Z. Tang, "Visual object tracking based on cross-modality Gaussian-Bernoulli deep Boltzmann machines with RGB-D sensors," *Sensors*, vol. 17, no. 1, 2017.

[35] G. Cheron, I. Laptev, and C. Schmid, "P-CNN: Pose-based CNN features for action recognition," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 3218–3226, Chile, December 2015.

[36] A. Chadha, A. Abbas, and Y. Andreopoulos, "Video Classification With CNNs: Using The Codec As A Spatio-Temporal Activity Sensor," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[37] Z. Zhu, W. Wu, and W. Zou, "End-to-end Flow Correlation Tracking with Spatial-temporal Attention[J]," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[38] M. Danelljan, G. Bhat, S. Gladh, F. S. Khan, and M. Felsberg, "Deep motion and appearance cues for visual tracking," *Pattern Recognition Letters*, 2018.

[39] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8695, no. 7, pp. 345–360, 2014.

[40] Y. Jia, E. Shelhamer, J. Donahue et al., "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the ACM Conference on Multimedia (MM '14)*, pp. 675–678, ACM, Orlando, Fla, USA, November 2014.

[41] S. Hannuna, M. Camplani, J. Hall et al., "DS-KCF: a real-time tracker for RGB-D data," *Journal of Real-Time Image Processing*, pp. 1–20, 2016.

Hindawi
Complexity
Volume 2018, Article ID 5298294, 10 pages
https://doi.org/10.1155/2018/5298294

Research Article

Design and Implementation of an Assistive Real-Time Red Lionfish Detection System for AUV/ROVs

M-Mahdi Naddaf-Sh, Harley Myler, and Hassan Zargarzadeh

Phillip M. Drayer Electrical Engineering Department, Lamar University, Beaumont, TX, USA

Correspondence should be addressed to Hassan Zargarzadeh; hzargarzadeh@lamar.edu

Received 11 September 2018; Revised 19 October 2018; Accepted 1 November 2018; Published 11 November 2018

Guest Editor: Li Zhang

In recent years, the Pterois Volitans, also known as the red lionfish, has become a serious threat by rapidly invading US coastal waters. Being a fierce predator, having no natural predator, being adaptive to different habitats, and being with high reproduction rates, the red lionfish has enervated current endeavors to control their population. This paper focuses on the first steps to reinforce these efforts by employing autonomous vehicles. To that end, an assistive underwater robotic scheme is designed to aid spear-hunting divers to locate and more efficiently hunt the lionfish. A small-sized, open source ROV with an integrated camera is programmed using Deep Learning methods to detect red lionfish in real time. Dives are restricted to a certain depth range, time, and air supply. The ROV program is designed to allow the divers to locate the red lionfish before each dive, so that they can plan their hunt to maximize their catch. Lightweight, portability, user-friendly interface, energy efficiency, and low cost of maintenance are some advantages of the proposed scheme. The developed system's performance is examined in areas currently invaded by the red lionfish in the Gulf of Mexico. The ROV has shown success in detecting the red lionfish with high confidence in real time.

1. Introduction

Biological invasions can cause environmental disruption and biodiversity loss, often due to human-caused global change [1–3]. Invasive species are nonnative species that may have serious effects on ecosystems and habitats. Some of these effects can evolve to global consequences [4]. The terrestrial and freshwater systems appropriate the majority of invasions, while, in the last decade, the rates of the marine invasions have dramatically increased and impacted the stability of ecosystems, raising ecological and economic concerns [1, 5]. Overall, recent studies [6] indicate that invasive species cost 120 billion dollars to the environment and the economy. Among marine species, *Pterios volitans* (red lionfish) is the most invasive and aggressive species that has taken only two decades to populate across a significant portion of the US east coast [7–9].

Regardless of how the red lionfish were first introduced [10–12], their rapid reproduction rate, lack of significant predators, and wide range of dietary consumption have made them a serious threat to coral reefs and many other marine environments [13–16]. As evidence to this threat, in Figure 1, the spread of the red lionfish in 1995 is compared with that of 2015. The red lionfish grow rapidly and reach up to 50 centimeters in 3 years [16]. Both smaller fishes and crustaceans are potential prey for red lionfish [1, 9]; they literally consume anything that they can fit in their mouth. Due to their venomous dorsal, pelvic, and anal spines the red lionfish are fatal [13] to human divers [9] and native predatory species who quickly learn to avoid them. Having no natural predator on one hand and the ability to quickly procreate, which is approximately 2 million eggs per female annually, the red lionfish population is exponentially growing and calling for immediate national resolutions [17].

Being aware of this destructive invasion in recent years, scientists have tried to find effective ways to control the red lionfish to spread and prevent more damage to the ecosystem [17]. One way for controlling the red lionfish population is hunting them by scuba divers equipped with "ZooKeepers" [18] and an "ELF lionfish Spear Tool" [19]. The divers spear the red lionfish and then keep them into the tubular ZooKeeper containments through their one-way

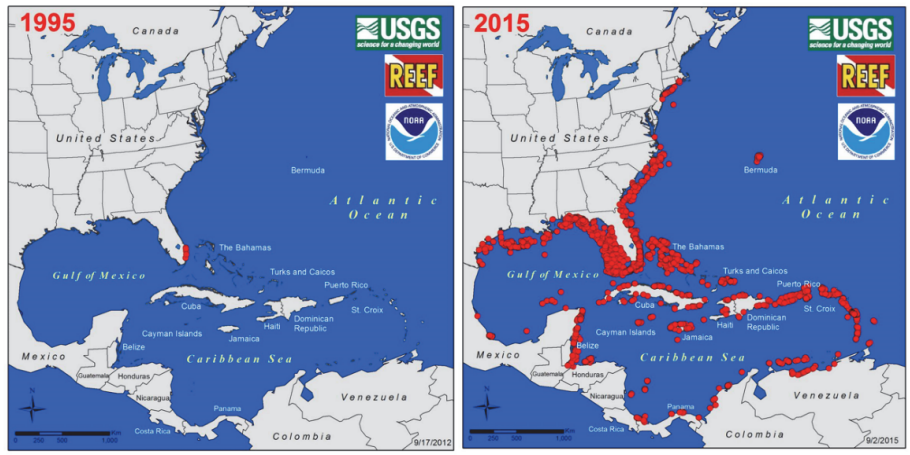


Figure 1: Red lionfish occurrence in the Western North Atlantic and Caribbean Sea (USGS-NAS 2015) (figures courtesy of USGS-NAS database found at http://nas.er.usgs.gov).

gate that holds the fish until the diver returns to the surface. This method is not cost-efficient due to limitations in the number of divers, diving time and the small number of hunts possible in each dive. Divers face many limitations for finding and locating red lionfish underwater due to significant depth, limited air cylinder capacity, low visibility conditions underwater, temperature differences, high pressure, etc. The fast spreading of the red lionfish calls for more efficient and aggressive solutions for the problem [20, 21]. Hence, introducing assistive technological schemes to detect the red lionfish can help to increase the effectiveness of the hunt in each dive.

Recently, several methods to detect fishes underwater are used for different purposes such as fishing, biological research, etc. Some of these methods employ high-resolution sonar scanning or vision-based methods. However, sonar-based fish finders are unable to distinguish fish species. Finding a specific species of fish can be very challenging using the available technologies. Using robots instead of humans in harsh environments is a common solution for such problems [22]. Nonetheless, for the robots, there are challenges to automatically detect objects underwater. Low lighting, moving cameras along with moving objects, limited sight, and background color change due to organic and artificial floating debris are some of the technicalities that add more complexity to red lionfish detection. Although some of these challenges had been addressed in the literature [23], applying them in underwater condition is still challenging. There exists an extensive amount of research for offline detection of fish underwater for different purposes such as counting [24, 25] and measurement of their length [26]. In [25] they perform detection, tracking, and counting fish; the method they used for detection was a moving average algorithm applied offline to recorded videos. In [27] a radio-tag system was developed for monitoring invasive fish. Lin Wu et al. [28] developed underwater object detection based on a gravity gradient, which detects objects underwater using a gravimeter. In [29] several methods were proposed and implemented on offline videos for the purpose of fish classification. In [30] deep Convolutional Neural Networks were used for coral classification. Qin proposed used Deep Learning for underwater imagery analysis [31]. In [32] a survey is conducted over using Deep Learning for various marine objects excluding red lionfish. Moreover, Siddiqui et al. [33] investigated the automated classification systems that have the ability to identify fish from underwater videos and also studied the feasibility and cost efficiency of automated methods like Deep Learning. Although many species are included in [33], unlike the red lionfish, the selected species had a less complex body shape such as P. porosus and A. bengalensis. An automatic image-based system was proposed in [24] that was used for estimating the mass of free-swimming fish. Qin et al. [34] has proposed live fish recognition using deep architecture where both support vector machines and SoftMax for classification are compared. Similar to [33], the outcome of [34] was tested on fishes with simple body shapes, e.g., oval or semicircle shapes and species with variable body shapes were excluded. The challenge in the lionfish detection is that it takes different gestures in different dispositions such as offending, defending, hiding or normal swimming [1]. In other words, unlike salmon, trout, tuna, etc., the red lionfish does not have a certain body shape. To the best of the author's knowledge, the available underwater object detection methods have not been applied to the detection of red lionfish in real time.

In this work, a compact-sized, open source ROV is employed to assist divers in detecting red lionfish in a more efficient manner. The scheme of the mechanism is as follows. The ROV is tethered to a computer on the surface that receives the video from a camera integrated on the ROV. The computer processes the video frames real-in-time and detects the red lionfish. By prespotting the red lionfish, the divers will not need to consume time and air hunting for them.

The ability of Deep Learning to learn patterns from high-dimensional data especially in image processing problems has made it a major tool for object detection or classification

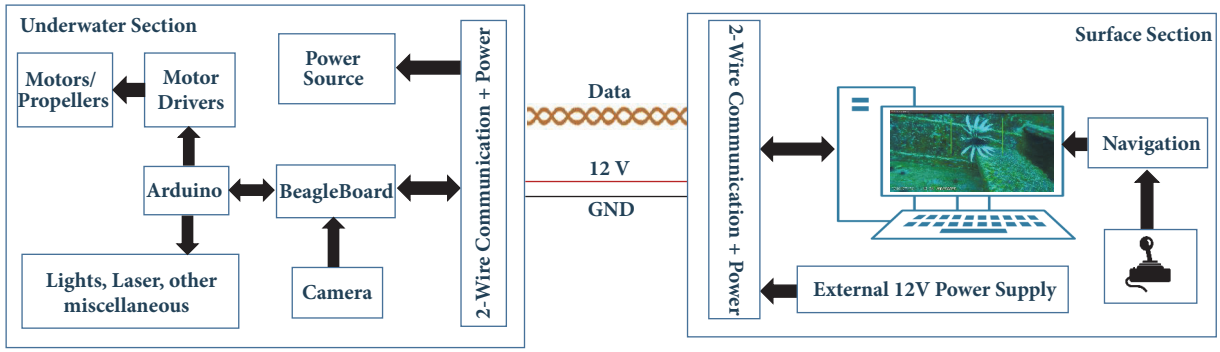


Figure 2: Overall system block diagram. ROV, Designed Interface in MATLAB, and Joystick for navigation are main parts of assistive system.

in recent years in several applications [35]. The proposed real-time red lionfish detection scheme employs the Deep Learning method in a MATLAB-based Graphical User Interface (GUI) in a PC. The user employs a joystick to navigate and investigate underwater while the ROV is programmed to send video from its camera in real time. Each frame of the video is processed, and the detected red lionfish are identified on the screen. The ROV uses its Inertial Measurement Unit (IMU) to report the detected lionfish's location to the surface. This scheme offers a unique platform that can be employed for detecting any other recognizable species. Further, finding, locating, and recording the population of the red lionfish are useful for the biologists to determine the red lionfish spreading patterns [7].

The remainder of this paper is organized as follows. Section 2 gives an overview of the robot specifications and navigation; also it discusses the algorithm and methodology that has been used for detecting the red lionfish. In Section 3, Simulation and real-world testing results and challenges are presented. The conclusion follows in Section 4.

2. Design

The proposed assistive system is comprised of the following subsystems:

(1) OpenROV robot that consists of actuators, control boards, lighting system, camera, and navigation system

(2) Computer and Graphical User Interface

(3) Red Lionfish detection system

The overall block diagram of the designed system is depicted in Figure 2, and the above subsystems are elaborated in the following subsections.

2.1. OpenROV 2.8. The OpenROV 2.8 is a telerobotic submarine with open access to its operational source code. Small dimensions and light weight ($30 \times 20 \times 15$ *cm* and 2.6 *kg*) make it possible to be carried and operated by a single user. It reaches a maximum forward speed of 2 knots, which is sufficient for the calm sea states, where the dives take place. This ROV is a tethered robot and all data are transceived through a 90-meter two-wire twisted cable with 100Mbps throughput. A Tenda HomePlug and a third party board (Topside Interface Board) designed by OpenROV [36] is used to convert Ethernet to the two-wire connection protocol. The communication channel is depicted in Figure 3. Three 700Kv (rpm/v) brushless motors with electronic speed control (ESC) propel the ROV to move in three dimensions. Figures 4 and 5 show the OpenROV layouts.

The OpenROV 2.8 is powered with six 3.3V batteries that last for at least 30 minutes of surfing, when fully charged. All the electronics are in an acrylic case under vacuum mounted as depicted in Figure 4, on an internal chassis. In addition, the maximum frame rate of the mounted camera is 30 fps at HD 720p quality.

2.2. User Interface Panel (UIP). To provide data access and control, a MATLAB-based Graphical User Interface (GUI) was coded that is shown in Figure 6. For all the test and simulations, the program was run on a Core i7 PC with 16 GigaByte of RAM and NVIDIA GTX 745 GPU.

Being open source, the ROV is supported by numerous libraries for accessing motors, camera, etc. They are accessible via socket.io [37] in MATLAB's GUI. A USB joystick was utilized to make the navigation of the ROV more convenient. The joystick is capable of controlling the ROV in all directions by adjusting the thrusters' rotation speed. The horizontal thrusters propel the ROV forward and backward and provide torque to control the yaw. The vertical thruster propels the ROV vertically. In addition, the joystick can control lights, camera recording, and camera tilting upward and downward. A Node.js [38] server was created and ran on BeagleBone Black that is integrated on the ROV.

In Figure 7 the communication process during deployment of the ROV underwater is depicted. Through these interface options, e.g., navigating, monitoring motors, controlling LEDs and lasers, camera tilting and capturing videos or images are accessible. The default method for transmitting data between the ROV and the surface computer is web-based. In this approach, all controlling and feedback values between the ROV and PC are transmitted using Socket.io. In other words, the ROV acts as a server and the browser in the PC is its client. To be able to transmit data without using a web browser, Java code is written for communicating

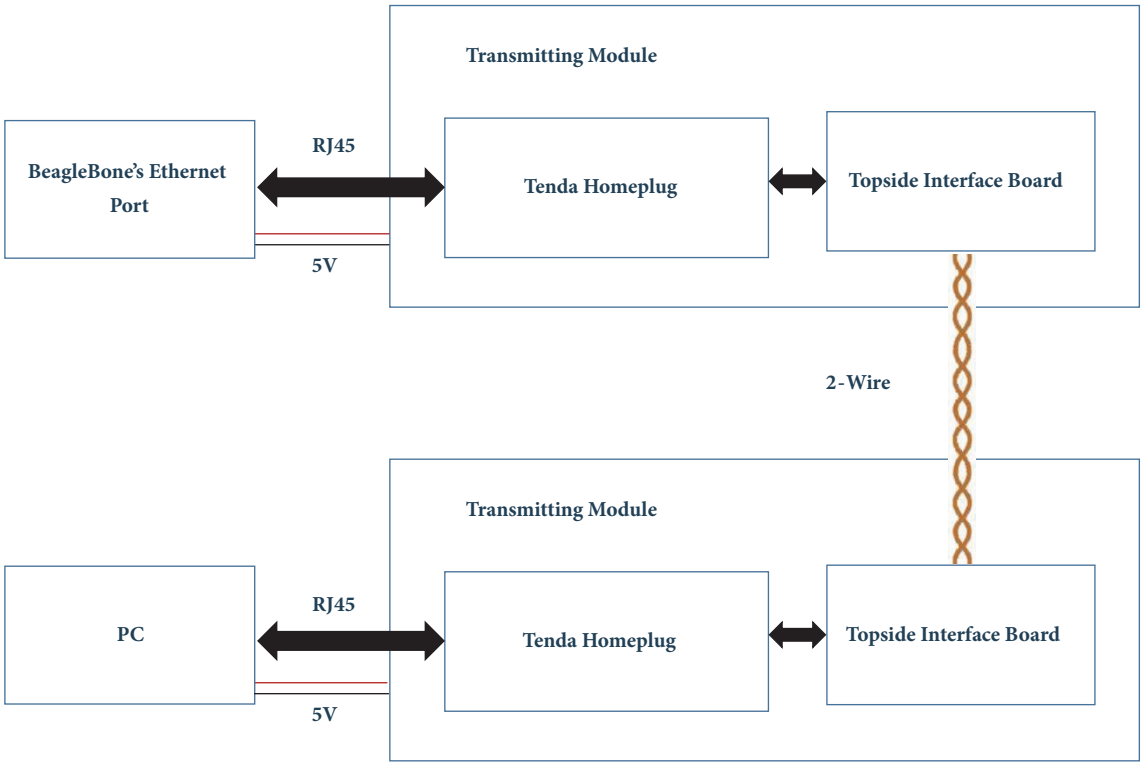


Figure 3: Schematic of data transmission channel between ROV and PC.

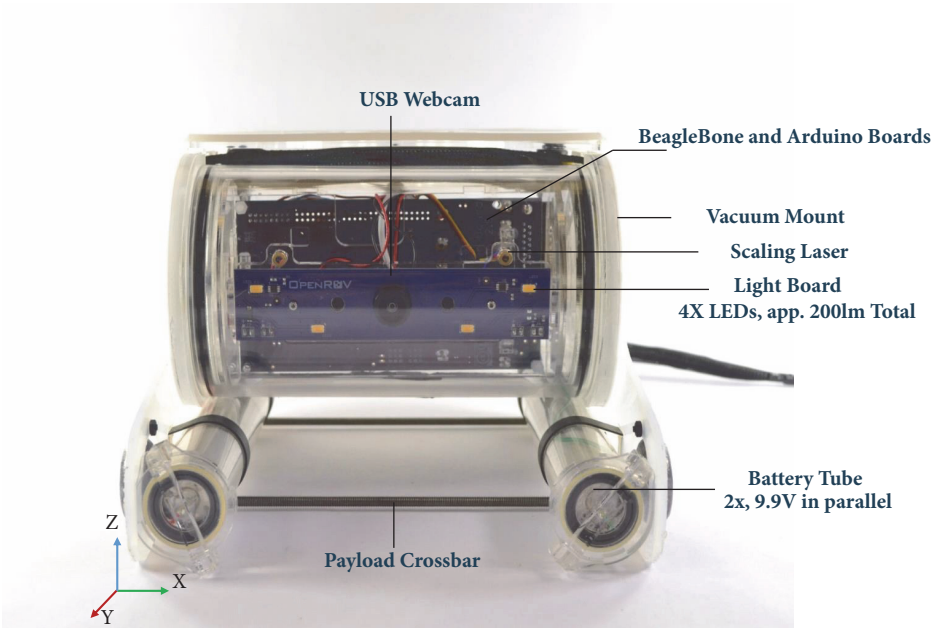


Figure 4: Front view of OpenROV 2.8. All main components of the ROV are marked. Two battery holders are for supplying power and balancing underwater.

with Node.js server inside the ROV directly and it bypasses the web browser for reducing delay between the ROV and UIP. This Java code is then used in the MATLAB GUI and provides complete access to the ROV. Navigational and miscellaneous related commands are sent through Socket.io each 20 milliseconds and the camera image frames are received each 0.8 seconds.

2.3. Object Detection Algorithm. To detect objects underwater, the Deep Learning (DL) method is employed. DL

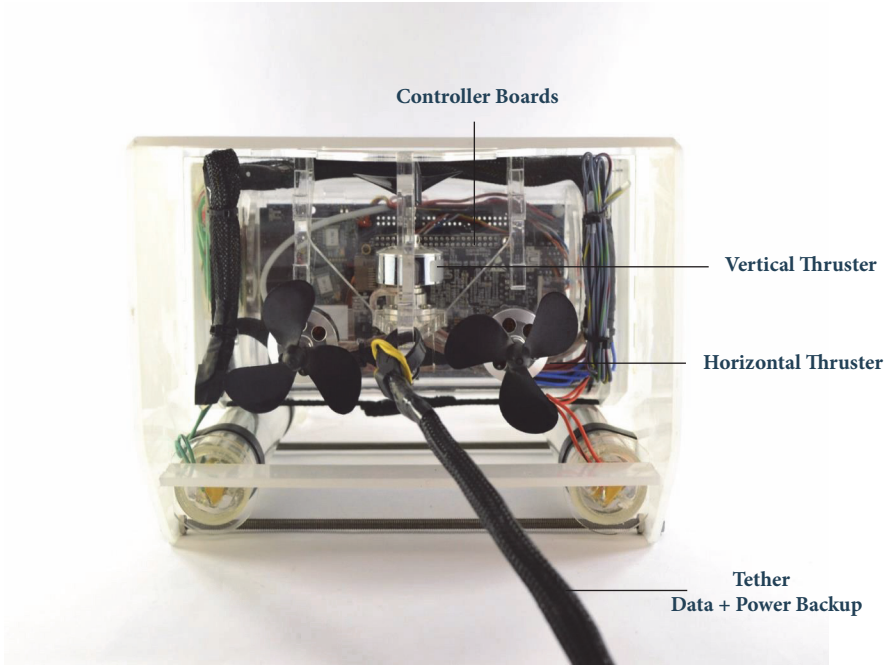


Figure 5: Rear view of OpenROV 2.8. There are three brushless motors for moving the device underwater. The tether cable is used for data transfer.

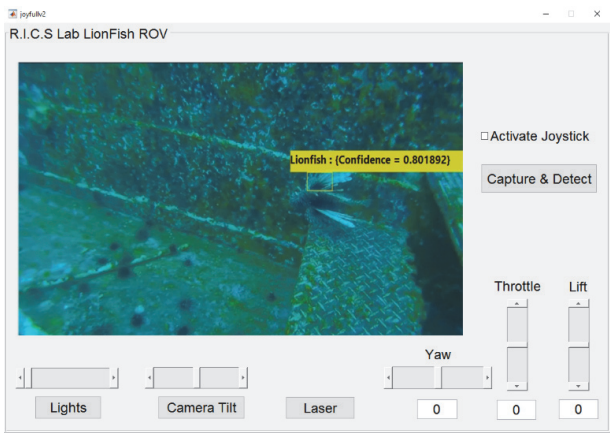


Figure 6: Designed UIP in MATLAB. All three brushless motors, LED lights, camera, laser, and camera tilt are accessible via this UIP. Live camera view also is available on this UIP.

utilizes Convolutional Neural Networks for object detection and classification [39]. For detecting objects of interest, it is necessary to gather a database that includes prototypical images of those specific objects. The more images that contain the object of interest, the more accurate the detection will become. There is no limitation on the number of objects of interest to be detected other than memory if sufficient prototypes are available. There are several databases for various objects available online, e.g., human faces in different poses, human body gestures, cars, etc. But unfortunately, because the red lionfish are a specific species, there are no images or video database available for them. To address that issue, 1500 images were gathered from different royalty free online resources such as ImageNet, Google, and YouTube. Also, the authors contributed to the database by participating in diving excursions in the Gulf of Mexico in different infested areas such as Flower Garden Banks National Marine Sanctuary, artificial reefs off the coast of Pensacola, FL.

Deep Learning consists of many cascaded layers and these layers are nonlinear processing functions used for feature extraction and transformation. The pattern recognition for the database is semisupervised and because of that we introduce the object of interest, which is a red lionfish in this case, and label them in the database. The classification is an unsupervised algorithm that classifies objects of interest from other objects based on defined labels. The database is trained using Regions with Convolutional Neural Networks (R-CNN) [40]. In this project the database images were run through 15 layers of 5×5 convolutions, and the filters were trained using Stochastic Gradient Descent with Momentum (SDGM) [41].

MATLAB was used to label the images. Using the "Training Image Labeler" app in MATLAB allows us to specify all rectangular ROIs in images. Most of images suffered from two problems, the first one was background and the second was low quality images requiring preprocessing. For instance, Figure 8(a) depicts a red lionfish next to an artificial reef, so for preparing the image for the database the brightness of image is modified as shown in Figure 8(b). Also, the size of the images is reduced to avoid other unnecessary objects, although the objects in the background or foreground, such as reefs and underwater debris, are impossible to avoid.

CNNs have the ability to build their own features and transform the input signal using convolutional kernels, an

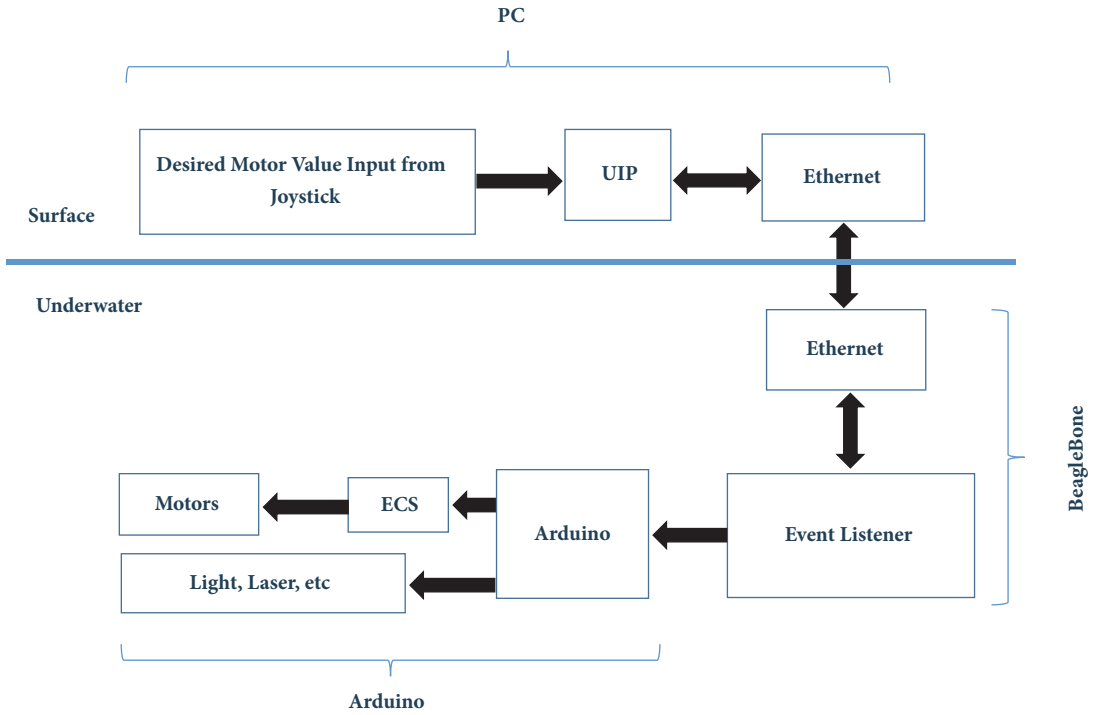
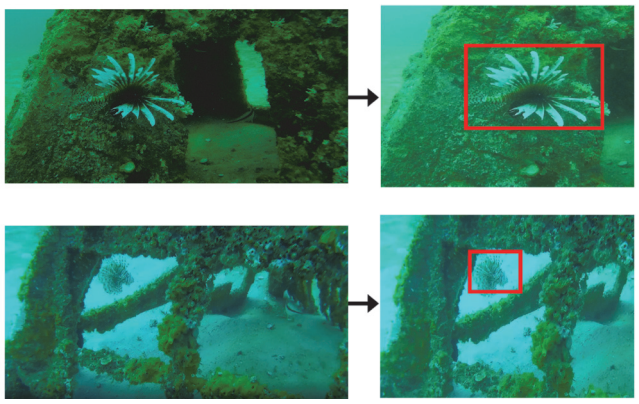


Figure 7: Data transmission diagram between UIP and ROV.



(a) (b)

Figure 8: Two left images in (a) are original images from red lionfish. The right images (b) are preprocessed to become suitable for the database. Images are taken from in Pete Tide II.

activation function, and a pooling phase. The activation function adds nonlinearity to the input, and the pooling phase is for reducing input size and strengthening learning [35]. Finally, in the last convolutional layer all features are sorted as a vector and sent to next layer. In the training step, the database that contains images and their labels are inputs to CNN. Figure 9 shows the architecture of CNN that was used for detecting the red lionfish. Excluding input layer, there were total of 14 layers. The input size of first layer was set to $32 \times 32 \times 3$ for all three channels and to have a coarse-to-fine prediction of features three convolutional layers were used. First convolutional layer size was set to 5×5. Each convolutional layer is followed by a Maxpooling layer. Maxpooling layers were followed by a Rectified Linear Unit (ReLU) layer where this layer is used as a thresholding operator [35, 42].

3. Experimental Results

The trained network was tested real-in-time on collected videos with ROV camera from four artificial reefs off the coast of Pensacola, Florida, USA. Figure 10 shows the sites that are visited during the experimental dives. In order to simulate real conditions, no samples from the recorded videos were added to the database. Therefore, the testing procedures could be assumed as a real-world situation, since the results were completely captured in real environment. Figure 11 shows a screenshot from one of the captured videos. Due to algae, green is the dominant color in this video. Also, the background, which is a sunken ship, has some patterns that can be mistaken as red lionfish stripes by a trained CNN. Moreover, as depicted in Figure 11, the similar stripe patterns were detected as a false positive instance. Confidence of false positives was relatively low. Therefore, to avoid false detection, the acceptable confidence level was set to 80%. Figure 12 is a sample of frame that have detected true positive. As depicted in Figure 12, although the red lionfish stripes are not clearly observable, because of other features such as fins, the trained network could successfully distinguish the red lionfish. In addition, as depicted in Figure 13, in a complicated

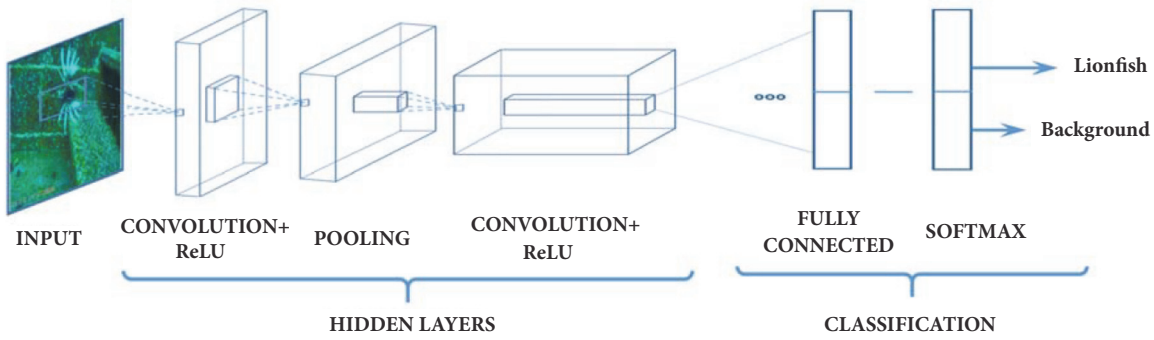


Figure 9: The architecture of the CNN that is used to detect red lionfish from other similar fishes and objects underwater.

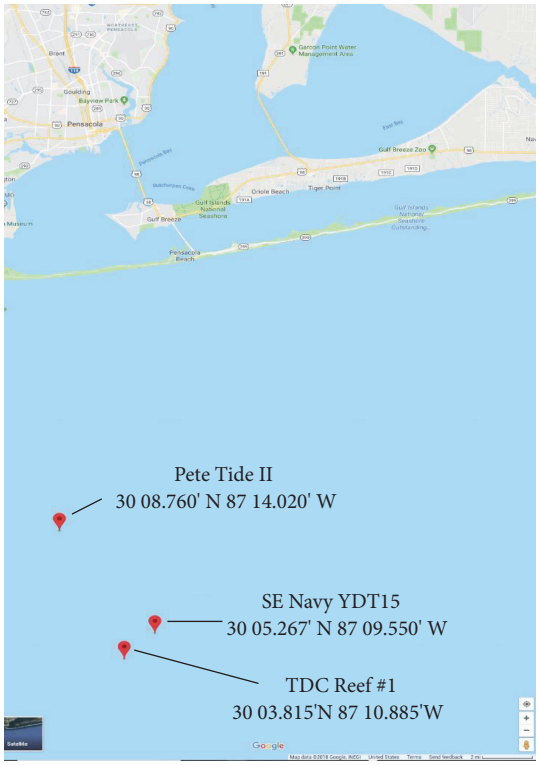


Figure 10: Location of three different sites in Pensacola, Florida, for finding red lionfish.

Figure 11: An example of false positive detection due to similarity of patterns. Although because of strip patterns this stair was detected as a red lionfish, the confidence was about 0.5 that is below the defined rejection threshold. Location of image is TDC Reef #1.

Figure 12: Despite the color change and camouflage, the red lionfish was detected with confidence above 0.8. Location of image is TDC Reef #1.

situation like presence of other fishes, the CNN is capable of detecting the red lionfish with 91% confidence.

In order to find the accuracy of proposed method, 1000 consecutive frames were selected from one of the captured videos from the Pensacola reefs. According to the selected frames, the red lionfish was available in 88.5% of them. Table 1 shows the number of true positive detected red lionfish in 885 frames that contained red lionfish in them. Among the true positive instances there were some frames that contained false positive instances that means despite the presence of red lionfish in that particular frame another object was wrongly detected as red lionfish with a confidence higher than 80%. Moreover, in some frames like Figure 14, the trained CNN was not able to detect the target due to different conditions like instantaneous turning of the red lionfish that cause a blurry image of it, very low light condition, or far distance. However, since these frames sporadically occur in the video, the overall continuousness of the lionfish tracking is not affected.

Table 1: Results of the trained CNN on 1000 frames.

Frames	True Positive Detected	False Positive Detected	Missed Red lionfish
red lionfish	93%	4%	3%

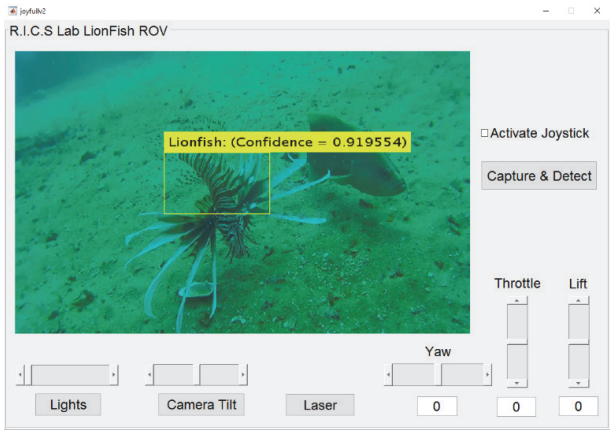


Figure 13: The red lionfish was detected correctly despite the presence of other species in the image. Location of image is SE Navy YDT15.

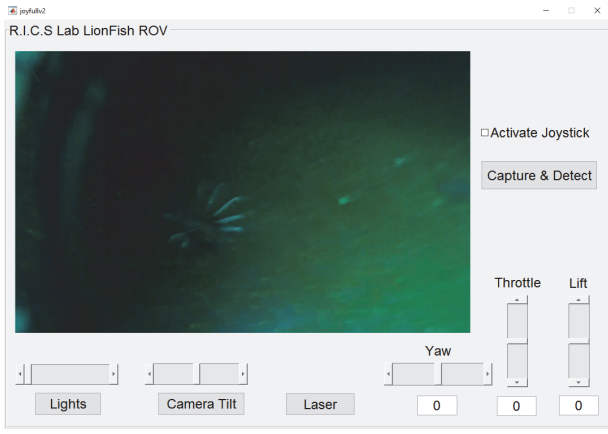


Figure 14: The ROV lights were off while the red lionfish tried to hide in a dark environment. Location of image is TDC Reef #1.

In Figure 15, a video containing 500 frames in presence of a red lionfish was selected for evaluating the real-time performance of the trained CNN real-in-time. The average time for processing each frame was measured as 0.097 seconds that leads to at least 10 frames per seconds. Since the red lionfish swim at low speed, the detection system can still notify the user about the presence of the red lionfish in a real-time manner. Figure 16 depicts the processing time for each of the 500 frames in the processed video. Moreover, Figure 17 shows the confidence percentage of detected red lionfish in each frame. In each frame, if the detected object has a confidence level lower than the threshold of 85%, then it is discarded as false positive. The total number of true positive detected objects in 500 frames was 461, which is 92% of the whole frames. Finally, four live performances of the trained

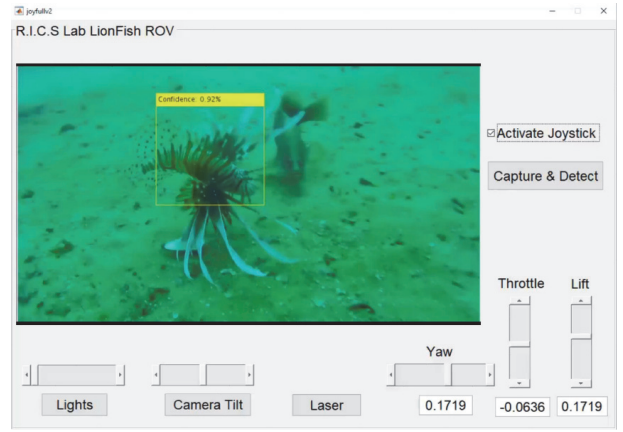


Figure 15: The processed video is available at youtu.be/j43Og_-d_SQ. Location of video is SE Navy YDT15.

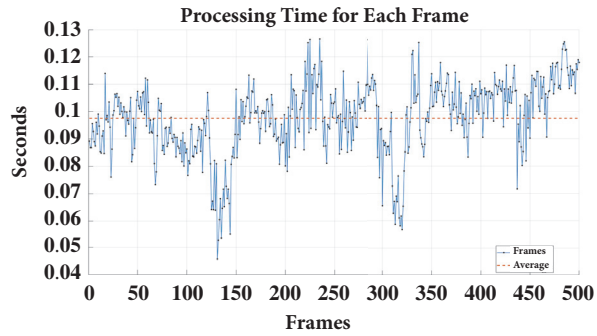


Figure 16: Processing time for each frame in the Figure 15's video. The average time for processing frames is 0.097 seconds.

CNN the red lionfish detection scheme can be found in a YouTube video with the address provided in Figure 18.

As one can recognize from the results, the proposed system shows success in real-time detection of red lionfish in a variety of environments and lighting conditions. However, further investigations are required to prove the effectiveness of the system on the number of catches, the time for each dive, and the number of divers needed in each excursion.

4. Conclusions

This study was a proof of concept for the design and implementation of a real-time CNN-based assistive robotic system for divers to locate the red lionfish. The assistive robot is able to find red lionfish at up to 30 meters in depth. The streaming videos from underwater are sent to the surface and processed in real time to detect the red lionfish. The overall design was driven by the needs of portability, high manoeuvrability, low energy consumption, and user

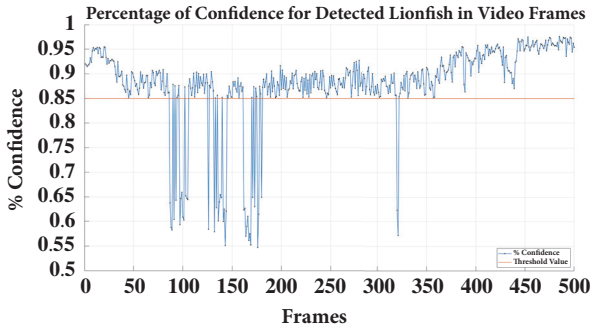


Figure 17: The percentage of confidence in each of 500 frames. Detected objects with confidence lower than 85% discarded as nonred lionfish.

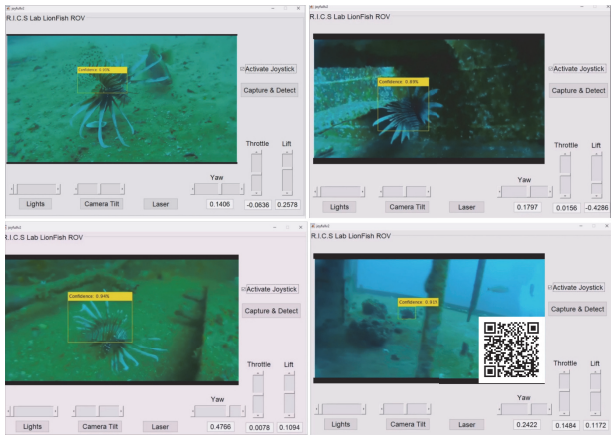


Figure 18: Four CNN live performance on three different sites can be found by scanning the QR code or on YouTube at youtu.be/j43Og_-d_SQ. Locations of videos are SE Navy YDT15, TDC Reef #1, Pete Tide II, SE Navy YDT15.

friendliness. The proposed scheme is able to perform red lionfish detection in real time, while diving in an underwater environment. The detection system was implemented on an open source, low cost ROV equipped with a camera to collect live videos underwater. Experiments were conducted on recorded videos from marine environments. The main achievement of this work was developing a computer-aided scheme on an affordable set of hardware that can be used environmentalist to detect and remove the lionfish. In the next phase of this research, the focus will be on (1) developing a custom-built ROV especially designed for lionfish detection and removal and (2) investigating the effect of utilizing the proposed assistive scheme on the number of hunts in diving excursions.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This paper was supported by an internal grant from Lamar University, College of Engineering.

References

[1] M. A. Albins and M. A. Hixon, "Worst case scenario: Potential long-term effects of invasive predatory lionfish (Pterois volitans) on Atlantic and Caribbean coral-reef communities," *Environmental Biology of Fishes*, vol. 96, no. 10-11, pp. 1151–1157, 2013.

[2] C. S. Elton, *The Ecology of Invasions by Animals and Plants*, University of Chicago Press, USA, 2000.

[3] R. N. Mack, D. Simberloff, W. M. Lonsdale, H. Evans, M. Clout, and F. A. Bazzaz, "Biotic invasions: Causes, epidemiology, global consequences, and control," *Ecological Applications*, vol. 10, no. 3, pp. 689–710, 2000.

[4] L. Pejchar and H. A. Mooney, "Invasive species, ecosystem services and human well-being," *Trends in Ecology & Evolution*, vol. 24, no. 9, pp. 497–504, 2009.

[5] G. M. Ruiz, J. T. Carlton, E. D. Grosholz, and A. H. Hines, "Global invasions of marine and estuarine habitats by non-indigenous species: mechanisms, extent, and consequences," *American Zoologist*, vol. 37, no. 6, pp. 621–632, 1997.

[6] D. Pimentel, R. Zuniga, and D. Morrison, "Update on the environmental and economic costs associated with alien-invasive species in the United States," *Ecological Economics*, vol. 52, no. 3, pp. 273–288, 2005.

[7] P. E. Whitfield, J. A. Hare, A. W. David, S. L. Harter, R. C. Muñoz, and C. M. Addison, "Abundance estimates of the Indo-Pacific lionfish Pterois volitans/miles complex in the Western North Atlantic," *Biological Invasions*, vol. 9, no. 1, pp. 53–64, 2007.

[8] W. J. Sutherland, "Horizon scan of global conservation issues for 2011," *Trends in ecology & evolution*, vol. 26, no. 1, pp. 10–16, 2011.

[9] P. J. Schofield, "Geographic extent and chronology of the invasion of non-native lionfish (Pterois volitans [Linnaeus 1758] and P. miles [Bennett 1828]) in the Western North Atlantic and Caribbean Sea," *Aquatic Invasions*, vol. 4, no. 3, pp. 473–479, 2009.

[10] R. M. Hamner, D. W. Freshwater, and P. E. Whitfield, "Mitochondrial cytochrome b analysis reveals two invasive lionfish species with strong founder effects in the western Atlantic," *Journal of Fish Biology*, vol. 71, pp. 214–222, 2007.

[11] B. X. Semmens, E. R. Buhle, A. K. Salomon, and C. V. Pattengill-Semmens, "A hotspot of non-native marine fishes: Evidence for the aquarium trade as an invasion pathway," *Marine Ecology Progress Series*, vol. 266, pp. 239–244, 2004.

[12] R. Ruiz-Carus, R. E. Matheson Jr., D. E. Roberts Jr., and P. E. Whitfield, "The western Pacific red lionfish, Pterois volitans (Scorpaenidae), in Florida: Evidence for reproduction and parasitism in the first exotic marine fish established in state waters," *Biological Conservation*, vol. 128, no. 3, pp. 384–390, 2006.

[13] M. A. Albins and M. A. Hixon, "Invasive Indo-Pacific lionfish Pterois volitans reduce recruitment of Atlantic coral-reef fishes," *Marine Ecology Progress Series*, vol. 367, pp. 233–238, 2008.

[14] A. B. Barbour, M. L. Montgomery, A. A. Adamson, E. Díaz-Ferguson, and B. R. Silliman, "Mangrove use by the invasive lionfish Pterois volitans," *Marine Ecology Progress Series*, vol. 401, pp. 291–294, 2010.

[15] S. J. Green, J. L. Akins, A. Maljković, and I. M. Côté, "Invasive lionfish drive Atlantic coral reef fish declines," *PLoS ONE*, vol. 7, no. 3, 2012.

[16] J. A. Morris Jr. and J. L. Akins, "Feeding ecology of invasive lionfish (Pterois volitans) in the Bahamian archipelago," *Environmental Biology of Fishes*, vol. 86, no. 3, pp. 389–398, 2009.

[17] L. C. T. Chaves, J. Hall, J. L. L. Feitosa, and I. M. Côté, "Photo-identification as a simple tool for studying invasive lionfish Pterois volitans populations," *Journal of Fish Biology*, vol. 88, no. 2, pp. 800–804, 2016.

[18] A. ElHage, *Lionfish Containment Unit, commonly known as the ZooKeeper*, Google Patents, 2015.

[19] G. Waugh, *System for harvesting marine species members including those that present a danger to a harvester*, Google Patents, 2014.

[20] F. Ali, K. Collins, and R. Peachey, "The role of volunteer divers in lionfish research and control in the caribbean," *Joint International Scientific Diving Symposium*, p. 7.

[21] F. Ali, K. Collins, and R. Peachey, *The role of volunteer divers in lionfish research and control in the Caribbean*, 2013.

[22] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[23] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 2, pp. II-97–II-104, Washington, DC, USA, June 2004.

[24] J. A. Lines, R. D. Tillett, L. G. Ross, D. Chan, S. Hockaday, and N. J. B. McFarlane, "An automatic image-based system for estimating the mass of free-swimming fish," *Computers and Electronics in Agriculture*, vol. 31, no. 2, pp. 151–168, 2001.

[25] C. Spampinato, Y.-H. Chen-Burger, G. Nadarajan, and R. B. Fisher, "Detecting, tracking and counting fish in low quality unconstrained underwater videos," in *Proceedings of the 3rd International Conference on Computer Vision Theory and Applications, VISAPP 2008*, pp. 514–519, Portugal, January 2008.

[26] E. Harvey, M. Cappo, M. Shortis, S. Robson, J. Buchanan, and P. Speare, "The accuracy and precision of underwater measurements of length and maximum body depth of southern bluefin tuna (Thunnus maccoyii) with a stereo-video camera system," *Fisheries Research*, vol. 63, no. 3, pp. 315–326, 2003.

[27] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," *IEEE Robotics and Automation Magazine*, vol. 20, no. 3, pp. 33–41, 2013.

[28] L. Wu, X. Tian, J. Ma, and J. Tian, "Underwater object detection based on gravity gradient," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 2, pp. 362–365, 2010.

[29] J. Anderson, B. Lewis, and C. O'Byrne, *Intelligent Fish Classi_cation in Underwater Video*, Research Experience for Undergraduates (REU) and the University of New Orleans funded by flagrant from the National Science Foundation, 2011.

[30] M. Elawady, *Sparse coral classification using deep convolutional neural networks*, 2015, arXiv preprint arXiv:1511.09067.

[31] H. Qin, X. Li, Z. Yang, and M. Shang, "When underwater imagery analysis meets deep learning: A solution at the age of big visual data," in *Proceedings of the MTS/IEEE Washington, OCEANS 2015*, USA, October 2015.

[32] M. Moniruzzaman, S. M. Islam, M. Bennamoun, and P. Lavery, "Deep Learning on Underwater Marine Object Detection: A Survey," in *Advanced Concepts for Intelligent Vision Systems*, vol. 10617 of *Lecture Notes in Computer Science*, pp. 150–160, Springer International Publishing, Cham, 2017.

[33] S. A. Siddiqui, A. Salman, M. I. Malik et al., "Automatic fish species classification in underwater videos: Exploiting pre-trained deep neural network models to compensate for limited labelled data," *ICES Journal of Marine Science*, vol. 75, no. 1, pp. 374–389, 2018.

[34] H. Qin, X. Li, J. Liang, Y. Peng, and C. Zhang, "DeepFish: Accurate underwater live fish recognition with a deep architecture," *Neurocomputing*, vol. 187, pp. 49–58, 2016.

[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[36] "OpenROV (n.d.)," https://www.openrov.com/.

[37] "Socket.io (n.d.)," https://socket.io/docs/.

[38] "Node.js (n.d.)," https://nodejs.org/en/.

[39] 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.

[40] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[41] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, USA, 2006.

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Mass, USA, 2016.