Hindawi
Journal of Advanced Transportation
Volume 2019, Article ID 6782803, 8 pages
https://doi.org/10.1155/2019/6782803

Research Article

Classification of Metro Facilities with Deep Neural Networks

Deqiang He, Zhou Jiang, Jiyong Chen, Jianren Liu, Jian Miao, and Abid Shah

Guangxi Key Laboratory of Manufacturing System & Advanced Manufacturing Technology, College of Mechanical Engineering, Guangxi University, 530004, Nanning, China

Correspondence should be addressed to Zhou Jiang; xyq031256@163.com

Received 1 May 2018; Revised 15 December 2018; Accepted 10 January 2019; Published 3 February 2019

Guest Editor: Mihai Dimian

Metro barrier-detection has been one of the most popular research fields. How to detect obstacles quickly and accurately during metro operation is the key issue in the study of automatic train operation. Intelligent monitoring systems based on computer vision not only complete safeguarding tasks efficiently but also save a great deal of human labor. Deep convolutional neural networks (DCNNs) are the most state-of-the-art technology in computer vision tasks. In this paper, we evaluated the effectiveness in classifying the common facility images in metro tunnels based on Google's Inception V3 DCNN. The model requires fewer computational resources. The number of parameters and the computational complexity are much smaller than similar DCNNs. We changed its architecture (the last softmax layer and the auxiliary classifier) and used transfer learning technology to retrain the common facility images in the metro tunnel. We use mean average precision (mAP) as the metric for performance evaluation. The results indicate that our recognition model achieved 90.81% mAP. Compared with the existing method, this method is a considerable improvement.

1. Introduction

With the rapid development of urban public transport in recent years, urban rail transport has become the preferred choice for many people because of its various advantages, such as high speed, punctuality, and environmental friendliness. Urban rail transit in China is developing at an amazing pace. The metro is an important part of urban rail transit. On pace with the improvement of metro train design, communication technology, and automation technology, the metro has seen immediate development. Its advanced technology with high safety makes it effective for solving saturated line conditions and enhances transport capacity. However, the main problem that influences driverless metro operation is obstacles. A collision with an obstacle will cause train impulses, derailments, vehicle equipment damage, and other problems. Thus, how to detect obstacles on the track quickly and accurately during operation has become a key issue in the study of safe metro operation. Currently, the detection of obstacles in metro tunnels is performed through manual observation. Detection that is dependent on manual labor has many drawbacks, including a high rate of missed detection, low efficiency, and low reliability. An improvement is automatic detection based on object detection and recognition. However, there are too many metro tunnel images, and this detection method has very high requirements for the efficiency and accuracy of the algorithm. The current research status of object detection and recognition technology can be classified into two categories. One is based on traditional methods [1] in image processing. The other uses DCNNs [2]. Traditional methods have three steps: target feature extraction, target recognition, and target location. The features used in this method can be categorized into two groups: global features including a color histogram [3] or circular shapes [4] and local features such as pixel color [5] or SIFT [6] features, which are all designed manually. Reference [7] proposed a method for visualizing pedestrian traffic flow using SIFT feature point tracking. Reference [8] used the strongly supervised deformable part models for object detection. Due to the diversity of features, it is difficult to extract features standardly and find the best way to represent them [9]. However, this problem could be solved if there were a general method to learn how to extract features automatically. Thus, the advantages of DCNNs have received more attention. A DCNN has recently developed a new kind of method for classification and recognition. It is a multilayer

cascade consisting of linear and nonlinear processing units that are able to extract features automatically and integrate with the process of classification and recognition and then learn by themselves. Reference [10] proposed a DCNN for the detection of arcs in pantograph-catenary systems. Reference [11] used a DCNN for car detection. Although the DCNN has shown impressive results, limited data and high computational resources are barriers to its use [2]. Therefore, this is a barrier in metro obstacle detection. In this paper, we studied the detection of obstacles in metro tunnels using a modified DCNN. In view of a few appropriate examples, we used a common facility image in a metro tunnel to replace the obstacles.

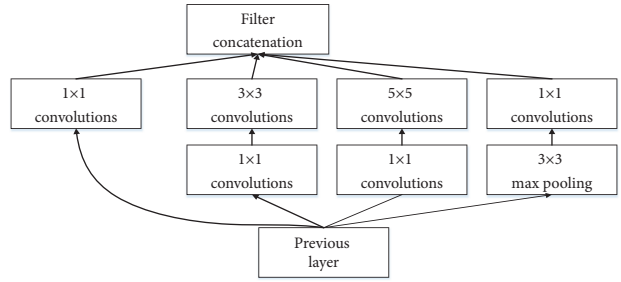

Figure 1: Inception module.

2. Methodology

To achieve high performance with the DCNN, an extremely common practice increases the size of layers or widths [12]. In theory, with a higher width and depth of the DCNN, it has a stronger learning capacity and higher forecasting precision. However, this method may have several drawbacks. The first is that a large DCNN has more convolutions and more layers, which means that the network requires training more parameters. It requires tremendous computational resources and makes the whole network more prone to overfitting the training set, especially if the training data are limited [13]. The second is that changes in feature distributions lead to model failure. In most cases where the DCNN achieved good performance, both the training set and the testing set were obtained from the same feature space and the same distribution. This easily causes the model to fail where feature distributions have large changes. Thus, before the application, the above issues must be settled. Based on massive experimentation with various convolutional network structures, GoogLeNet designers provided several general guiding principles [14]:

(1) Avoid representational bottlenecks, especially early in the network. The representation size should gently decrease from the inputs to the outputs rather than being extremely compressed before reaching the final representation used for the current task

(2) Higher dimensional representations are easier to process locally within a network

In the CNN, increasing the activations per tile can obtain more disentangled features. This allows resulting networks to be trained faster.

(3) Spatial aggregation can be performed over lower dimensional embedding without much or any loss in representational power

(4) When designing the network structure, one must consider the balance of the width and depth of the network. A reasonable network structure shoulder distributes computational resources on its structure, which contributes to higher quality networks

2.1. GoogLeNet. GoogLeNet first appeared in the ILSVRC 2014 competition and won first place by a wide margin. The first version is often called Inception V1 [15]. The main feature of Inception V1 is that it showed good results while limiting the number of computations and parameters—by 93.33% as top 5, which is less than half of AlexNet. Inception V1 proposed a module named Inception. This module contains four branches (the architecture is given in Figure 1): the first branch makes convolution with inputs by a $1*1$ size convolution kernel that is also an important structure proposed in the NIN (network in network) [16]. The $1*1$ convolution kernel, an excellent structure, adds a layer for feature transformation and nonlinear changes with few computational resources, which enhances the ability of the expression of the whole network and increases or decreases the dimension of the outputs. As Figure 1 shows, all branches use a $1*1$ size convolution kernel for low-cost cross-channel feature transformation. The second branch first uses a convolution of $1*1$ and then connects to a convolution of $3*3$, which is equivalent to twice the feature transformations. The third one is similar to the second, but it connects to a $5*5$ size convolution kernel, and the last one has a $3*3$ max pooling and $1*1$ size convolution. The Inception module allows the depth and width of the network to be extended efficiently, improving the accuracy and avoiding overfitting.

Inception V2 [17], the second version of GoogLeNet, replaces the $5*5$ size convolution kernels with two $3*3$ convolution kernels to reduce the number of parameters and overfitting. It proposed a very effective regularization method called batch normalization (BN). BN speeds up the training rate of large-scale convolutional networks by many times, and the classification accuracy after convergence is also greatly improved. When used in the network layers, BN normalizes the interior of the mini-batch data to output the normally distributed data and decreases internal covariate shift. In traditional deep neural networks, the input distribution of each layer varies during training, making it difficult to train. Therefore, only by setting a small learning rate can the training process continue. However, using batch normalization in each layer of the network could solve this problem because the training allows a high learning rate to run, in which the number of iterations is considerably reduced. After this measure was employed, the training time of Inception V2 was fourteen times faster compared to Inception V1 and had higher convergence accuracy.

The third version, Inception V3 [14], had two major improvements. One was the introduction of the thought of factorization into small convolutions. The other was that the

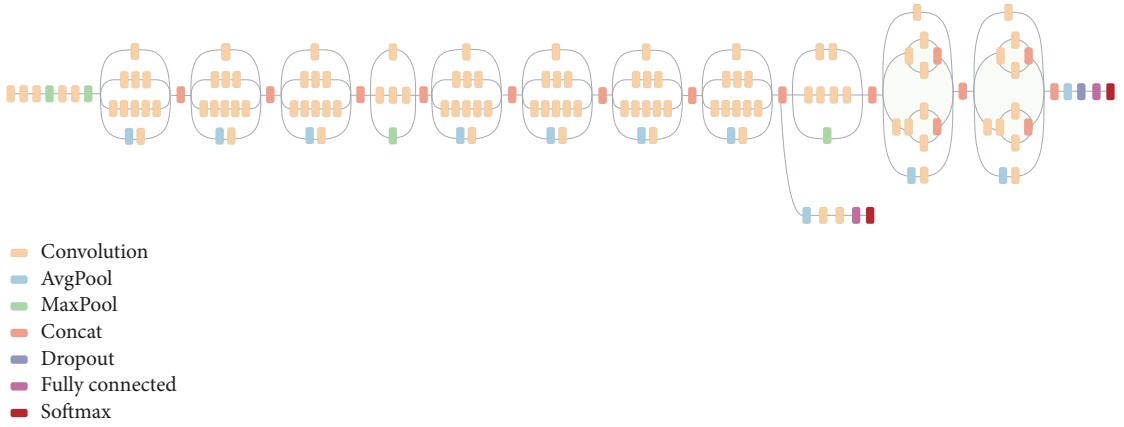

Figure 2: Model architecture.

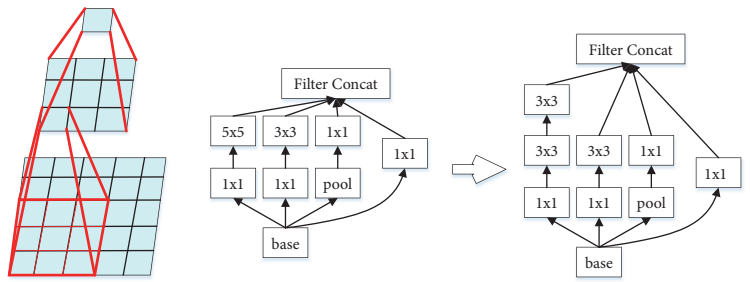

Figure 3: Mini-network replacing the 5 × 5 convolutions.

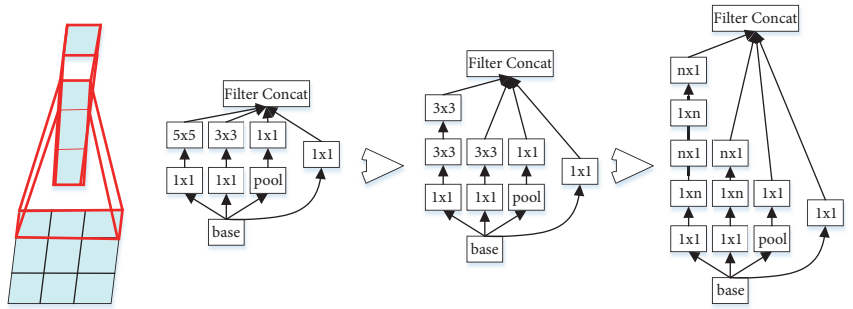

Figure 4: Mini-network replacing the 3 × 3 convolutions. The lower layer of this network consists of a 3 × 1 convolution with 3 output units.

Inception module was optimized. The visualization of the model architecture is given in Figure 2.

The ideal of factorization into small convolutions is a large improvement of Inception V3. As shown in Figure 3, a large convolution layer can be replaced by a multilayer network with fewer parameters.

For example, a 5 × 5 convolution with n filters over a grid with m filters is 25/9 = 2.78 times more computationally expensive than a 3 × 3 convolution with the same number of filters. Thus, two 3 × 3 convolutions replace one 5 × 5 convolution having (9 + 9)/25 × reduction with a relative gain of 28% [14]. Furthermore, spatial factorization into asymmetric convolutions can factorize a convolution into smaller convolutions. For instance, using a 3 × 1 convolution followed by a 1 × 3 convolution is equivalent to a two-layer network with the same 3 × 3 receptive field (Figure 4).

If the number of input and output filters is equal, the two-layer solution is 33% cheaper for the same number of output filters.

The Inception model helps to reduce computational complexity and increase the width and number of stages. It has 1×1, 3×3, and 5×5 convolution layers. The 1 × 1 convolution layers are used to increase the network depth and improve the network nonlinearity. It also reduces the number of 3×3 and 5×5 convolution layers, which is the main reason that the GoogLeNet network model is expanded in terms of depth and width, but the total number of parameters is smaller than that of the classical VGG network. The Inception module accepts the previous input and forms the output of the Inception module through the parallel processing of different scales and functional branches, thus achieving multiscale feature fusion in Inception V3 that was achieved by setting a top 5 error rate

Table 1: The outline of the proposed network architecture.

Type	Patch size/stride or remarks	Input size
conv	3×3 / 2	299×299×3
conv	3×3 / 1	149×149×32
conv padded	3×3 / 1	147×147×32
pool	3×3 / 2	147×147×64
conv	3×3 / 1	73×73×64
conv	3×3 / 2	71×71×80
conv	3×3 /1	35×35×192
Inception module	3×Inception	35×35×288
Inception module	5×Inception	17×17×768
Inception module	2×Inception	8×8×1280
pool	8×8	8×8×2048
linear	logits	1×1×2048
softmax	classifier	1×1×1000

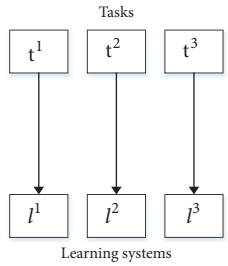

Figure 5: Traditional machine learning.

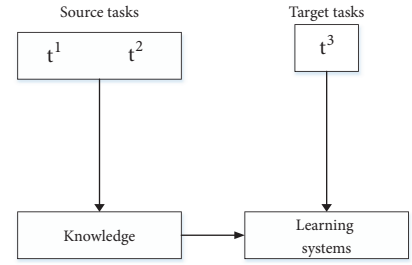

Figure 6: Transfer learning.

of 3.64% on the 2012 validation dataset. Our measure is based on Inception V3.

2.2. Transfer Learning. In network training procedures, there is an important hypothesis that the training and testing dataset must be in the same feature space and have the same distribution. However, it cannot be held in many applications. For example, we sometimes have a classification task in a domain of interest, but we have sufficient training data in another domain, and the latter data may be in different feature spaces or follow different data distributions. In this case, the successful transfer of knowledge will avoid a large amount of expensive data markup work and greatly improve the performance of learning. Transfer learning [18], as the name implies, simply transfers learned-trained model parameters to a new model to help train the model. Because most of the data or tasks are related, a trained model can share its parameters (also understood as knowledge learned by the model) to a new model in a way that expedites and optimizes the learning rate of the new model without starting over as is required in most other networks. Thus, transfer learning is a simple method to transfer knowledge between task domains. The following is the difference between traditional machine learning and transfer learning. Traditional machine learning needs to be retrained for different target tasks, while transfer learning is not necessary (see Figures 5 and 6).

In transfer learning, the network weights and biases are initialized with existing useful values, which can obviously reduce the training time to finish the final training task and significantly lower the required amount of training data. Deep convolutional networks with transfer learning are the best way to solve this detection problem and achieve the most advanced performance with the lowest computational requirements.

2.3. Model. The layout of the entire network is given in Table 1.

The output size of each module is the input size of the next module. To keep the size of the grid, the convolution is marked with zero padding, and the inside of the Inception modules also uses zero padding for the same purpose. The network uses softmax for classification. The original architecture has 1,000 object classes, so we adjusted its structure to fit our project. We changed the final classification layer (the last softmax layer and the auxiliary classifier) on a pretrained Inception V3 network and retrained it with our dataset, fine-tuning the parameters across all layers.

3. Experiments

3.1. Data Collection and Processing. Our dataset was collected during a metro inspection, authorized by Nanning Rail Transit Co. It contains 6000 original images that were

Figure 7: A few examples.

divided into 6 classes: distribution box (DB), jet fan (JF), wireless communication base station module (AP), passenger information system wireless terminal box (PIS), radio transmission equipment (TRE), and billboard (BI). Figure 7 shows a few examples.

We extended the dataset by applying a series of methods, as follows:

(1) Randomly rotating each image; the number of images doubled

(2) Resizing each image to 299×299

(3) Adjusting the brightness of the image

(4) Adjusting the image contrast

(5) Adjusting the image saturation

To evaluate the results, each category was divided into a training set and a testing set: 75% for the training set and 25% for the testing set.

We use mean average precision (mAP) [19] as the metric for performance evaluation. The average precision (AP) is the area under the precision-recall curve. It is widely used for object detection and is calculated by averaging the precision over a group of spaced recall levels [0,0.1,...,1], and the mAP is the AP calculated over all classes. The details are as follows:

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots 1\}} p_{interp}(r). \quad (1)$$

The precision at each recall level r is interpolated by taking the maximum precision measured for a method for which the corresponding recall exceeds r:

$$p_{\text{interp}}(r) = \max_{\tilde{r}:\tilde{r} \geq r} (\tilde{r}), \quad (2)$$

where $p(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

3.2. Training Methodology. Our experiment was conducted on Google's TensorFlow [20]. TensorFlow is an open source software library for machine learning of various perception and language comprehension tasks. It aims to promote the study of machine learning and the rapid and simple transition from prototype research to production.

It is challenging to train deep neural networks on smaller datasets. However, by transfer learning, a large number of acquired feature parameters are extracted from one of the largest datasets—ImageNet—that can be transferred to our new model, which provided better results in detection. The whole network was trained using backpropagation and used a global learning rate of 0.001 using RMSProp with a decay of 0.9 and a momentum of 0.9. The training stopped when there were no more obvious improvements.

3.3. Experimental Results and Comparisons. The probability that a photo belongs to a category was given by the multinomial logistic regression. The category with the highest probability was taken as the predicted category. Parts of the results are shown in Figure 8, and category predictions are accompanied by probabilities.

This work aims at classifying the common facility images in metro tunnels. To the best of our knowledge, this is the first work for this task. There are limited approaches that we could compare in the literature. We resort to other image classification approaches for comparison. Since random forest has been widely used in image classification and object detection [21], we compare our model with random forest. This approach first appeared in [22] and was further developed in [23]. Random forest for this experiment is provided by [23]. We also compare the proposed method with deformable part models (DPM) [1], which have achieved state-of-the-art results on the PASCAL and INRIA person datasets. It is based on mixtures of multiscale deformable part models to represent highly variable object classes. The original implementation of DPM provided by [1] is used for this experiment. Since both random forest and DPM are traditional image recognition, we compare with another method based on a plain CNN. The architecture we used was introduced by [2], which won the 2012 ILSVRC competition.

The experimental data are the same as the data used by Inception V3 and are also divided into 75% for the training set and 25% for the testing set. Parts of the experimental examples are shown in Figures 9–11. Table 2 shows the experimental results.

In Table 2, the first two lines show the results obtained by the traditional method. The performance of the DPM-based method is poor; it only achieves 27.94% mAP. Although DPM has achieved state-of-the-art performance in general object detection, the performance of metro detection is not satisfactory. Random forest is better than DPM, but it only achieves 45.17%. The reason for the low mAP of these two methods is that both methods are based on traditional image processing. Such methods need to manually select the filters that require multiple experiments to determine satisfied filters in type and quantity. The variety of features, rich color changes, and complex environmental backgrounds make such traditional methods difficult to correctly identify. The mAP was better

Figure 8: A few results.

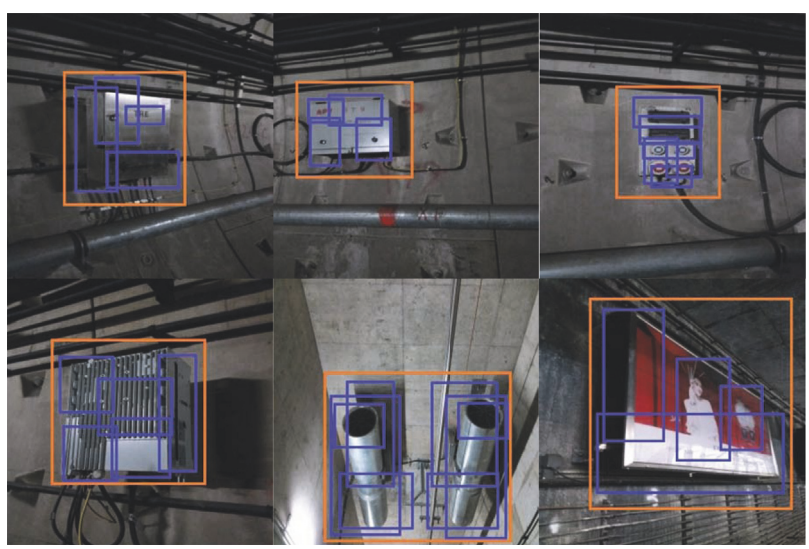

Figure 9: A few results by DPM.

Table 2: Experimental results.

Method	DB(%)	JF(%)	AP(%)	PIS(%)	TRE(%)	BI(%)	mAP(%)
DPM	28.56	27.62	29.53	21.27	29.41	31.24	27.94
Random forest	45.27	41.38	45.79	50.45	42.35	45.78	45.17
CNN	60.92	58.14	55.47	53.33	51.67	53.95	55.58
Inception V3	90.65	88.30	93.09	93.78	85.49	93.55	90.81

Figure 10: A few results by the CNN.

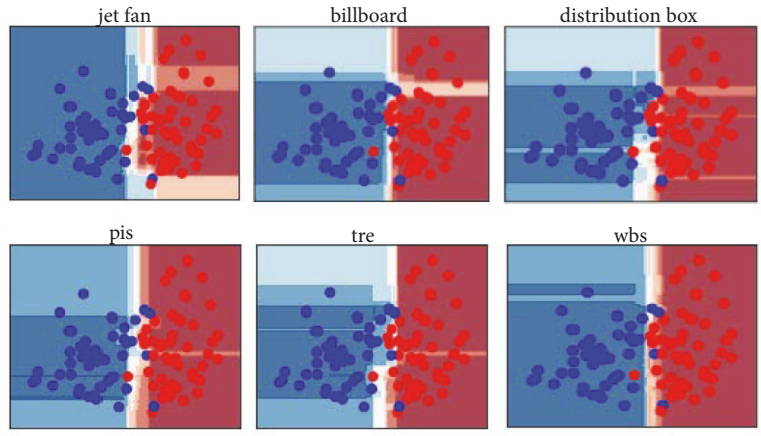

Figure 11: Results by random forest.

when the convolutional neural network was applied, especially using the deep convolution neural network. Instead of manual feature extraction, the feature extraction layer of the CNN learns features directly when training the data, which means that it can avoid the limitations of manual feature extraction. A plain CNN achieves an mAP of 55.58%, and Inception V3 achieves 90.81%. In contrast to the plain CNN, Inception V3 factorizes the convolution filter into a small filter. It reduces many parameters and accelerates the calculation and adds a nonlinear layer extending the whole network expression ability. Such a feature made Inception V3 perform much better than the plain CNN.

4. Conclusion

In this paper, we used a deep convolutional neural network model, Inception V3, devised by Google. By this module, a very deep network can be built with fewer parameters, thus reducing computational resources. The final classification layer of the network was removed and retrained with our

dataset to construct a satisfactory structure. The application of transfer learning trained a very deep model rapidly from scratch and with a small dataset. Our modified Inception V3 achieved impressive results on the datasets, 90.81%, which were much better than published methods. Considering the number of parameters and the consequent computational cost, this approach could actually be a viable choice for metro intelligent monitoring systems.

Data Availability

The dataset used to support the findings of this study is related to the safety of subway operation and is copyrighted; it is partly available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (Grant No. 51765006), the Natural Science Foundation of Guangxi Province of China (Grant No. 2017GXNSFDA198012), the Key Project of Science and Technology of Guangxi (Grant No. 1598009-6, AB17195046), the Guangxi Manufacturing Systems and Advanced Manufacturing Technology Key Laboratory Director Fund (15-140-30S003), and the Innovation Project of Guangxi Graduate Education (YCSW2018033).

References

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[2] 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.

[3] Z. Zivkovic and B. Kröse, "An EM-like algorithm for color-histogram-based object tracking," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 1, pp. 798–803, IEEE, July 2004.

[4] T. Garlipp and C. H. Müller, "Detection of linear and circular shapes in image analysis," *Computational Statistics & Data Analysis*, vol. 51, no. 3, pp. 1479–1490, 2006.

[5] Y. Deng, B. Manjunath S, and H. Shin, "Color image segmentation," in *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*, pp. 1021–1036, 1999.

[6] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using SIFT features and mean shift," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[7] Y. Tsuduki and H. Fujiyoshi, "A method for visualizing pedestrian traffic flow using SIFT feature point tracking," *Pacific Rim Symposium on Advances in Image and Video Technology*, pp. 25–36, 2009.

[8] H. Azizpour and I. Laptev, "Object detection using strongly-supervised deformable part models," *Lecture Notes in Computer Science*, vol. 7572, no. 1, pp. 836–849, 2012.

[9] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS '14)*, pp. 487–495, December 2014.

[10] G. Karaduman, M. Karakose, and E. Akin, "Deep learning based Arc detection in pantograph-catenary systems," in *Proceedings of the International Conference on Electrical and Electronics Engineering*, pp. 12–63, 2017.

[11] Y. Xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car detection from low-altitude UAV imagery with the faster R-CNN," *Journal of Advanced Transportation*, vol. 2017, Article ID 2823617, 10 pages, 2017.

[12] H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini, and S. Cagnoni, "Food image recognition using very deep convolutional networks," in *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management (MADiMa '16)*, pp. 41–49, 2016.

[13] D. M. Hawkins, "The problem of over-fitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, pp. 2818–2826, July 2016.

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

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of the International Conference on Learning Representations*, vol. 1, pp. 56–63, 2014.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML '15)*, vol. 1, pp. 448–456, July 2015.

[18] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[19] 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.

[20] M. Abadi, P. Barham, J. Chen et al., "TensorFlow: a system for large-scale machine learning," *Operating Systems Design and Implementation*, vol. 1, pp. 265–283, 2016.

[21] A. Bosch, A. Zisserman, and X. Muñoz, "Image classification using random forests and ferns," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, vol. 2, pp. 1–8, IEEE, Rio de Janeiro, Brazil, October 2007.

[22] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[23] L. Breiman, M. Last, and J. Rice, "Random forests: finding quasars[M]//Statistical challenges in astronomy," Springer, New York, NY, 243-254, 2003.