An Efficient and Slight Convolutional Neural Network for Vehicle Type Classification

Haibo Zheng, Naijie Gu and Xiaoci Zhang

Anhui Province Key Laboratory of Computing and Communication Software, School of Computer and Technology, University of Science and Technology of China, Hefei, China
boiboibo@mail.ustc.edu.cn, gunj@ustc.edu.cn, zxiaoci@mail.ustc.edu.cn

Abstract. With the popularity of small storage embedded devices, recent researches on deep convolutional neural networks no longer only focus on improving the classification accuracy but start to attach great importance to the size of convolutional neural networks. This paper proposes a small deep convolutional neural network for vehicle type classification, based on the Fire module and the residual module. The Fire module is applied to maintain a small number of parameters. The residual module is employed to accelerate the training process and improves the classification accuracy. Experimental results show that our model achieves 95.57% accuracy of vehicle type classification on an open source dataset BIT-Vehicle. This outperforms previous researches and some classical convolutional neural networks such as AlexNet, ResNet-32 and SqueezeNet. What is more, the size of our model is only 5.3 MB, which has 40X fewer parameters than AlexNet and has 15X fewer parameters than ResNet-32.

1. Introduction

Vehicle type classification is an important part of modern intelligent transportation system that relieves more and more austere traffic pressure. With the widespread use of traffic surveillance cameras, numerous recent researches on vehicle type classification have focused mainly on image-based methods. Until then, existing image-based methods are roughly divided into two categories: model-based methods and appearance-based methods. For model-based methods [1, 2, 3, 4], the 3D model of vehicles are usually required to recover by vehicles’ 3D parameters such as length, width and height. For appearance-based methods [5, 6, 7, 8, 9], appearance features are extracted to perform vehicle type classification. However, the classification accuracy of these methods might drop sharply under cutthroat environmental conditions like severe illumination and extreme weather. Besides, multifarious image processing steps are adopted in these methods, which is considerably time-consuming and reduces greatly the real-time performance of models.

Since AlexNet [10] popularized deep convolutional neural networks (DCNNs) by winning the ImageNet Challenge ILSVRC 2012 [11], a great number of efficient convolutional neural networks (CNNs) such as VGG [12], ResNet [13], and GoogleNet [14] have sprung up on image classification. Compared with traditional image-based methods, CNNs not only avoid complicated image processing steps, but also obtain more satisfactory classification accuracy on computer vision datasets. Therefore, more and more methods based on CNNs have already been applied to vehicle type classification. Dong et al. [15] designed a vehicle type classification model using a semi-supervised multi-layer feed-forward CNN, and introduced sparse Laplacian filters into their CNN to capture more discriminative features of unlabelled data. Chen et al. [16] constructed a useful vehicle type classification network based on vehicle rear-view images. Kim et al. [17] built a CNN for vehicle type classification, and...
made their CNN more robust by bagging methods, the data augmentation and the post-processing for imbalanced data. Liu et al. [18] integrated DCNNs with balanced sampling to design a rich-functional vehicle type classification system. However, these researches focused mainly on the accuracy but usually ignored the size of models when building the architecture of CNNs.

But luckily for us, many model compression methods make it possible to design a high-efficient and small DCNN. Denton et al. [19] proposed the singular value decomposition (SVD) that was applied to per-training CNNs. Han et al. [20] developed network pruning for pre-training CNNs that replaced parameters below a certain threshold with zero to form a sparse matrix and then performed a few iterations of training on this sparse matrix. Iandola et al. [21] replaced big filters with smaller filters and decrease the number of input channels to big filters in order to keep a small total parameters number of CNNs.

In this paper, a high-efficient vehicle type classification model based on a DCNN is proposed. To begin with, our CNN employs the Fire module to maintain a small size. In addition, unlike traditional CNNs, the scale layer that can shift and scale the input data is adopted for all convolutional layers in our CNN. Finally, we accelerate the training process and improve the accuracy by the residual module [13].

The rest of this paper is constructed as follows. Firstly, Section 2 describes our CNN architecture and demonstrates the design of important modules: the Fire module and the residual module. Secondly, details and results of experiments are shown in Section 3. Finally, Section 4 concludes our works briefly.

2. Architecture of the Convolutional Neural Network

In this Section, our CNN architecture is illustrated in detail. As demonstrated in Figure 1, our CNN starts with a traditional convolutional layer (conv1), followed by 12 Fire modules (fire1-12) that are responsible for holding a small total number of parameters, ending with a standard convolutional layer (conv2). The softmax is selected as the classifier of our CNN. First of all, the number of filters per Fire module from the beginning to the end of our CNN is increased gradually. In addition, max-pooling with stride of 2 are performed after Fire modules fire1, fire3, fire6, and fire9. Then, our CNN also performs average pooling after the convolutional layer (conv2). Finally, the dropout layer with a ratio of 50% is applied after the Fire module fire12, which contributes to preventing our CNN from overfitting [22].

![Figure 1. Architecture of our CNN](image)

In Table 1, parameters of our CNN architecture are described in detail. In a Fire module, $s_{1x1}$ is the number of 1×1 filters in the squeeze layer; $e_{1x1}$ is the number of 1×1 filters in the expand layer; $e_{3x3}$ is the number of 3×3 filters in the expand layer; $m$

...
is the number of 3×3 filters in the expand layer. The output of convolutional layers have three dimensions: \((C \times H \times W)\); \(C\) denotes the output channels number of convolutional layers; \(H\) denotes the height of output feature maps, and \(W\) denotes the width of output feature maps. But in table 1, we do not demonstrate the concrete information of other layers such as the output concatenation layer and the element-wise layer.

In following Section 2.1 and 2.2, the Fire module and the residual module are described in detail.

| layer name | filter size / stride | \(s_{1x1} / e_{1x1} / e_{3x3}\) | output size \((C \times H \times W)\) |
|------------|----------------------|-------------------------------|----------------------------------|
| data       |                      |                               | 3×227×227                        |
| conv1      | 3×3 / 2              |                               | 64×113×113                       |
| fire1      |                      |                               | 64×113×113                       |
| maxpool1   | 3×3 / 2              | 8 / 32 / 32                   | 64×56×56                         |
| fire2      |                      | 16 / 64 / 64                 | 128×56×56                        |
| fire3      |                      | 16 / 64 / 64                 | 128×56×56                        |
| maxpool2   | 3×3 / 2              |                               | 128×28×28                        |
| fire4      |                      | 32 / 128 / 128               | 256×28×28                        |
| fire5      |                      | 32 / 128 / 128               | 256×28×28                        |
| fire6      |                      | 32 / 128 / 128               | 256×28×28                        |
| maxpool3   | 3×3 / 2              |                               | 256×14×14                        |
| fire7      |                      | 64 / 256 / 256              | 512×14×14                        |
| fire8      |                      | 64 / 256 / 256              | 512×14×14                        |
| fire9      |                      | 64 / 256 / 256              | 512×14×14                        |
| maxpool4   | 3×3 / 2              |                               | 512×7×7                          |
| fire10     |                      | 64 / 256 / 256              | 512×7×7                          |
| fire11     |                      | 64 / 256 / 256              | 512×7×7                          |
| fire12     |                      | 64 / 256 / 256              | 512×7×7                          |
| conv2      | 3×3 / 1              |                               | 512×7×7                          |
| avgpool    | 7×7 / 1              |                               | 7×1×1                            |

### 2.1. Fire Module

As shown in Figure 2, the Fire module is illustrated in detail. A Fire module consists of a squeeze convolutional layer with 1×1 filters, an expanded convolutional layer with 1×1 filters and an expanded convolutional layer with 3×3 filters.

In a Fire module, each convolutional layer is followed by a scale layer and a ReLU (Rectified Linear Unit) layer. The scale layer is an irreplaceable part of our Fire module, and can shift and scale the input data. In Section 3, related experiments are conducted in order to verify the positive effect of the scale layer on our Fire modules.

In the front, we briefly state that the Fire module is adopted to hold a small total parameters number. Now, concrete steps are described in detail. As we all know, the size of a CNN is the sum of each layer’s parameters number. Therefore, to maintain a small size of our CNN, we try hard to reduce parameters of each layer. As shown in Formula 1, parameters of a layer \(KP\) depends multiplicatively on the number of input channels \(M\), the number of output channels \(N\), and the size of filter \(s \times s\).

\[
KP = M \times N \times (s \times s) \quad (1)
\]

Apparently, in order to control the size of a CNN, two main strategies can be employed when building a CNN architecture:

- Replace 3×3 filters with smaller 1×1 filters.
- Decrease the number of input channels to 3×3 filters.

As for the first strategy, 1×1 filters instead of 3×3 filters are chosen for all squeeze convolutional layers and part of expand convolutional layers in our CNN. Assuming that both the input and output of
a convolutional layer with 3×3 filters has C channels, it requires $3^2 \times C^2 = 9C^2$ parameters. In the meantime, if a convolutional layer with 1×1 filters also has C input and output channels, it would only require $1^2 \times C^2 = C^2$ parameters. Therefore, a convolutional layer with 1×1 filters has 9X fewer parameters than a convolutional layer with 3×3 filters.

As for the second strategy, in order to hold a small size of our CNN, only replacing 3×3 filters with 1×1 filters is far from enough. We also need to decrease the number of input channels to 3×3 filters, because parameters of convolutional layers with 3×3 filters account for the largest part of our model parameters. So, for all Fire modules, we set $s_{1\times1} < e_{1\times1} + e_{3\times3}$; $s_{1\times1}$ denotes the output channels number of the squeeze layer; $e_{1\times1}$ denotes the output channels number of the expand layer with 1×1 filters; $e_{3\times3}$ denotes the output channels number of the expand layer with 3×3 filters. For all Fire modules, the squeeze layer helps to reduce parameters number of expand layers with 3×3 filters by limiting the number of input channels to the expand layer with 3×3 filter. Consequently, our CNN holds a small total number of parameters for the reason that parameters of the expand layer with 3×3 filters reduces tremendously.

**Figure 2.** Architecture of Fire module. A Fire module consists of a squeeze convolutional layer with 1×1 filters, an expanded convolutional layer with 1×1 filters and an expanded convolutional layer with 3×3 filters. Besides, each convolutional layer in a Fire module is followed by a scale layer and a ReLU layer.

### 2.2 Residual Module

Research shows that there would be a degradation phenomenon, as the depth of CNNs increases [24]. Fortunately, He et al. [13] have already found a simple and effective degradation resolution named as residual learning that can also accelerate the training process of CNNs.

For simplicity, assuming that the desired underlying mapping of traditional neural networks is $H(x)$, $H(x)$ is recast into $F(x) + x$ by the residual module. The mapping $F(x) + x$ is performed by a shortcut connection and element-wise addition. He et al [13] firmly believe that it is more convenient and efficient to optimize the residual mapping than to optimize the original mapping. Besides, identity shortcut connections in the residual module can be easily implemented on open source deep learning frameworks Caffe [25] and TensorFlow [26]. What’s more, the residual mapping does not increase redundant burden on parameters and computing power. In Section 3, related experiments are conducted in order to prove the significant influence of the residual module on our CNN.

The residual module of our CNN is illustrated in Figure 3. The residual mapping is added to Fire modules fire2, fire5, fire8 and fire11. As shown in Table 1, we also notice that the output channels number of Fire module fire2 is inconsistent with the output channels number of the max-pooling layer maxpool1. Therefore, in order to ensure that they have the same output channels number, a convolutional layer with 1×1 filters and 128 output channels is employed after the max-pooling layer maxpool1.
3. Experiment Results and Analysis

In this section, we introduce our datasets, our experimental environment, and details of training and evaluation for the vehicle type classification model.

3.1. Datasets

The open source dataset BIT-Vehicle [15] is adopted to verify the versatility of our CNN in this paper. BIT-Vehicle dataset includes 9850 images with sizes of 1600×1200 and 1920×1080 captured from two cameras at different time (daytime and nighttime) and places. The proportion of nighttime images in the whole dataset is about 10%. For all images, there are rich changes in illumination condition, the scale, the surface colour of vehicles, and the viewpoint. All images contain 10053 vehicles, because some images maybe have 2 or more than 2 vehicles. As demonstrated in Figure 4, 10053 vehicle images fall into six categories: bus, microbus, minivan, sedan, SUV, and truck. The number of vehicles per category is 558, 883, 476, 5922, 1392 and 822 respectively.

3.2. Training Setup

Our CNN is trained on Caffe [25]. According to the ratio of 7:3, BIT-Vehicle dataset is randomly split into the training set and the validation set. In this paper, physical hardware conditions and training parameter configurations are consistent in all our experiments. Now we introduce physical hardware and training parameter configurations of our experiments in detail.
**Physical hardware of experiment.** Eight NVIDIA Tesla K40m GPUs and two Intel XeonE5-2620 V2 CPUs with 128 GB of main memory are used.

**Training parameter configurations.** Batch size is 32 for training and 8 for validation in all our experiments. All our models are trained with stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 0.0005. The models are trained for up to 160K iterations. The learning rate is set initially to $10^{-3}$, and then decayed by 10 at 100K, 120K and 140K iterations respectively.

### 3.3. Comparison Results

In Table 2, our CNN is compared with previous work based on a semi-supervised CNN [15] and some classical CNNs such as AlexNet [10], ResNet-32 [13], and SqueezeNet [21]. Our CNN achieves the best classification accuracy of 95.57% among methods mentioned in table 2. Firstly, compared with AlexNet, our CNN has a 0.74% higher accuracy, and 40X fewer parameters. Secondly, our CNN is better than ResNet-32 by 2.79%, and has 15X fewer parameters. Thirdly, our CNN has 0.95% higher accuracy than SqueezeNet, but the size of our CNN is only 0.6MB larger than SqueezeNet. Finally, the size of our CNN is only 5.3MB, and still less than 10MB. In other words, our CNN could be stored directly on small storage embedded devices instead of being limited by memory bandwidth.

| Methods        | The Size of Model | Accuracy  |
|----------------|-------------------|-----------|
| Dong et al. [15] | —                 | 88.11%    |
| AlexNet        | 218M              | 94.83%    |
| ResNet-32      | 82M               | 92.78%    |
| SqueezeNet     | 4.7M              | 94.62%    |
| Ours           | 5.3M              | 95.57%    |

### 3.4. Discuss the Impact of the Residual Module on the CNN

In Table 3, a set of experiment is conducted so as to prove the significant influence of the residual module on our CNN.

**Original CNN “residual, scale”**. As shown in Figure 1, the CNN employs the residual module, and adopts the scale layer for all convolutional layers.

**Counterpart CNN “plain, scale”**. The CNN is based on the above original network without the residual module.

In Table 3, the classification accuracy of our original CNN “residual, scale” is 0.91% higher than the counterpart CNN “plain, scale”. What’s more, the size of our original CNN is only 0.1MB more than the counterpart CNN. As He et al. [13] said, the residual module would not increase redundant burden on parameters of our CNN. In the meantime, we also notice that the residual module accelerates the training process and improves the accuracy in Figure 5.

![Figure 5](image-url)
Table 3. The influence of the residual module on classification accuracy of the CNN

| Methods                      | The Size of Model | Accuracy |
|------------------------------|-------------------|----------|
| Ours (plain, scale)          | 5.2M              | 94.66%   |
| Ours (residual, scale)       | 5.3M              | 95.57%   |

3.5. Discuss the Impact of the Scale Layer on the CNN

In order to verify the positive effect of the scale layer on our CNN, a group of experiments are designed in Table 4.

**Original CNN “residual, scale”.** As demonstrated in Figure 1, the CNN employs the residual module, and adopts the scale layer for all convolutional layers.

**Counterpart CNN “residual, non-scale”**. The CNN is based on original CNN but discards all scale layers.

In Table 4, the classification accuracy of original CNN is 0.42% higher than counterpart CNN. It is satisfying because the size of counterpart CNN is only 0.1MB less than original CNN. In other words, the scale layer does not add too many extra parameters to our CNN.

Table 4. The influence of the scale layer on classification accuracy of the CNN

| Methods                    | The Size of Model | Accuracy |
|----------------------------|-------------------|----------|
| Ours (residual, non-scale) | 5.2M              | 95.15%   |
| Ours (residual, scale)     | 5.3M              | 95.57%   |

4. Conclusion

In this paper, we propose a DCNN based on the Fire module and the residual module for vehicle type classification. Firstly, the Fire module is adopted to maintain a small total parameters number of our CNN. Secondly, we accelerate the training process and improve the classification accuracy by the residual module. Finally, open source dataset BIT-Vehicle is used to evaluate our CNN. Results of experiments demonstrate that our CNN achieves the best accuracy 95.57% compared with previous research and some classical CNNs such as AlexNet, ResNet-32, and SqueezeNet. What’s more, our CNN has only 5.3MB parameters, which is 40X fewer parameters than AlexNet and 14X fewer parameters than ResNet-32. In other words, our CNN could be stored directly on small storage embedded devices instead of being limited by memory.

5. References

[1] Hsieh, Jun-Wei, et al. "Automatic traffic surveillance system for vehicle tracking and classification." IEEE Transactions on Intelligent Transportation Systems 7. 2 (2006): 175-187.

[2] Zhang, Zhaoxiang, et al. "Three-dimensional deformable-model-based localization and recognition of road vehicles." IEEE Transactions on Image Processing 21.1 (2012): 1-13.

[3] Lai, Andrew HS, George SK Fung, and Nelson HC Yung. "Vehicle type classification from visual-based dimension estimation." Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE. IEEE, 2001.

[4] Gupte, Surendra, et al. "Detection and classification of vehicles." IEEE Transactions on intelligent transportation systems 3. 1 (2002): 37-47.

[5] Shan, Ying, Harpreet S. Sawhney, and Rakesh Kumar. "Unsupervised learning of discriminative edge measures for vehicle matching between nonoverlapping cameras." IEEE transactions on pattern analysis and machine intelligence 30.4 (2008): 700 - 711.

[6] Jiang, Mengwan, and Haoliang Li. "Vehicle classification based on hierarchical support vector machine." Computer Engineering and Networking. Springer, Cham, 2014. 593 - 600.

[7] Ma, Xiaoxu, and W. Eric L. Grimson. "Edge-based rich representation for vehicle classification." Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Vol. 2. IEEE, 2005.
[8] Zhang, Chengcui, Xin Chen, and Wei-bang Chen. "A PCA-based vehicle classification framework." Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on. IEEE, 2006.

[9] Ji, Peijin, Lianwen Jin, and Xutao Li. "Vision-based vehicle type classification using partial Gabor filter bank." Automation and Logistics, 2007 IEEE International Conference on. IEEE, 2007.

[10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[11] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International Journal of Computer Vision 115.3 (2015): 211-252.

[12] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv: 1409.1556 (2014).

[13] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[14] Szegedy, Christian, et al. "Going deeper with convolutions." Cvpr, 2015.

[15] Dong, Zhen, et al. "Vehicle type classification using a semisupervised convolutional neural network." IEEE transactions on intelligent transportation systems 16.4 (2015): 2247-2256.

[16] Chen, Yanjun, et al. "Vehicle type classification based on convolutional neural network." Chinese Automation Congress (CAC), 2017. IEEE, 2017.

[17] Kim, Pyong-Kun, and Kil-Taek Lim. "Vehicle Type Classification Using Bagging and Convolutional Neural Network on Multi View Surveillance Image." Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on. IEEE, 2017.

[18] Liu, Wei, et al. "An Ensemble Deep Learning Method for Vehicle Type Classification on Visual Traffic Surveillance Sensors." IEEE Access 5 (2017): 24417-24425.

[19] Denton, Emily L., et al. "Exploiting linear structure within convolutional networks for efficient evaluation." Advances in neural information processing systems. 2014.

[20] Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems. 2015.

[21] Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv: 1602.07360 (2016).

[22] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[23] Sandler, Mark, et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation." arXiv preprint arXiv: 1801.04381 (2018).

[24] He, Kaiming, and Jian Sun. "Convolutional neural networks at constrained time cost." Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on. IEEE, 2015.

[25] Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014.

[26] Abadi, Martin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv: 1603.04467 (2016).