A comparison between Bidirectional Neural Networks and Convolutional Neural Networks on Vehicle Classification

Tianyi Qi

Research School of Computer Science
Australian National University
Canberra, Australia
tianyi.qi@anu.edu.au

Abstract. Vehicle Classification is a crucial task for Smart AI City. This paper takes two approaches into comparison:Bidirectional Neural Network (BDNN) and Convolutional Neural Network (CNN) with their classification performance on the VehicleX dataset, which is generated using Unity. Experiments show that the end-to-end CNN approach outperforms the BDNN approach in almost every circumstance. Also, in this paper, other investigations like training from scratch, training with Batch Normalization layers, training with different backbone, different number of layers are included to boost the performance of the CNN and BDNN model.

Keywords: BDNN· CNN · VehicleX· Bidirectional· vehicle classification· Neural Network· Classification.

1 Introduction

The vehicle classification problem becomes a key question in AI city. The machine needs to identify, classify, and track the specific vehicle to organize the traffic flow of the entire city. There has been much work previously to solve this problem. SY Cheung's team[2] uses a single magnetic sensor to tackle the traffic measurement and vehicle classification problem, J Gajda's team[5] solves this problem by using inductive loop detectors.

Neural Network (NN) approach has recently become a leading trend to solve this type of classification problem. NN based method reports the charming result on many other fields, like age classification[9], facial emotion classification[10], speech tone classification[1], etc. Furthermore, due to the power of NN, more and more traditional classifiers like Decision Tree, Gradient Boosting Method are stepping out of the stage. For most image-based tasks, Convolutional Neural Network (CNN) is commonly used as a feature extractor. There are many robust structures, like ResNet[6], VGG[12], that proven to be stable on many types of images.

This paper compares the result of a variety configuration of Bidirectional neural network(BDNN)[11] and CNN on the VehicleX[14] dataset. The results shows that end-to-end CNN approach performs better than those BDNN approach.

The rest of this paper is organized in this order: Section 2 introduces the dataset and feature pre-processing and both structure and theory of CNN, simple NN and BDNN. Section 2 also introduces the evaluation metrics. Section 3 is mainly about implantation, result analysis and discussion. Section 4 includes future work and conclusion.

2 Method

2.1 VehicleX dataset, feature preprocessing and data augmentation

VehicleX is a publicly available 3D engine created in Unity, which contains 1,362 vehicles of various 3D models of 10 types of cars with fully editable attributes. In this paper, we use an adapted image set that is directly generated from VehicleX. The same generated dataset is also used for CityFlow AI City Challenge 2020 competition. The generated dataset contains 45,438 images for training, 14,936 images for validation, and 15,412 images for testing. Besides the image itself, the original labeling feature like vehicle orientation, light condition, camera prosperities, vehicle type, and vehicle color is also included. The distribution of the dataset is shown in Table 1.

From Table 1, this training dataset is evenly distributed, and there is no massive variance between the maximum and the minimum number of samples for one class. Thus regular training should produce acceptable results. No additional unbalanced dataset processing is required.

This dataset has two different versions. Version 1 (V1) consists of image features, which are 2048 dimensional feature vectors extracted from an ImageNet[3] pre-trained ResNet model. Version 2(V2) consists of the image itself, shown in Fig. 1. In this paper, due to the input requirements for different types of networks, the V2 dataset will be used on the CNN-based approach, and the V1 dataset will be used on it BDNN based approach.

Table 1: Sample Distribution of the training dataset

Property	Value
Total Class	1362
Max No. Samples for one class	48
Min No. Samples for one class	20
Average No. Samples for one class	33.36
Standard Variance No. Samples for one class	4.705

Fig. 1: The overview of VehicleX Dataset

Other labeling features can be separated into two major classes: image-related (vehicle orientation, light intensity, light direction, camera distance, camera height) and vehicle property (vehicle type, vehicle color).

In this experiment, for CNN based approach, the standard data augmentation techniques like random brightness, random contrast, motion blur, random rotation are used. After data augmentation, other features like orientation, light intensity, light direction are not valid and will be ditched. Only the images are used for CNN based approach. For BDNN based approach, only vehicle property features are used as additional input features. Image properties should be handled by the image feature extractor and may not directly link to the vehicle itself we are trying to classify, thus ditched. Due to both selected are categorical features, these features are processed with one-hot encoding. Both one-hot encoding features are concatenated after the ResNet 2048 dimensional feature, form into a 2071 dimensional feature vector for each sample, and uses as the network input for BDNN. The target output is 1362 dimension vector indicates the 1362 vehicles for both networks.

2.2 CNN based approach

In this experiment, we first compare various CNN backbone network's performance on Vehicle X classification tasks. This include ResNet50, DenseNet161[8], SEResNet50[7], VGG16. These backbones are proven robust on most computer vision tasks and have been adopted into many different tasks and areas. VGG increases the network depth by using tiny convolution filters (3x3), which the author claims a significant improvement compared to the prior-art configurations. In this paper, we use the VGG16 model. ResNet uses a breakthrough residual connection to connect through each network layer, allowing quick gradient flow, better performance, and faster convergence speed. ResNet50 is used in this experiment. Compared to the ResNet structure, DenseNet improves those connections by connecting each layer to every other layer in a feed-forward fashion. In this experiment, we use DenseNet161 for comparison. Also, by adding attention structure into the ResNet architecture, SEResNet outperforms the original ResNet on the ImageNet classification task. This paper use SEResNet50 for comparison. All backbones are connected to a single fully connected layer with 1362 output neurons to perform the final classification. Backbones are pretrained on the ImageNet dataset. All models use cross-entropy(Eq. 1) for calculating the loss.

Besides the pretrained model, this experiment also tries to train the entire model from scratch. Inspired from ScratchDet[15], to further boost the performance of the scratch model, a modified network with Batch Normalization inserted after each convolution layer is also adopted for comparison. Due to the limitation of GPU, in this experiment, we only implement those modifications on VGG16.

Also, the loss function is vital for multi-class classification tasks like this. Recent multi-class classification tasks all tend to modify the loss function to achieve better results. To achieve more discriminative deep features

between each class, we also try to replace the traditional cross-entropy loss function with Arcface[4] loss and Cosface[13] loss. By adding an angular margin penalty on the target angle, those types of losses force the model to learn a more discriminative representation for each class. In this experiment, we implement those modifications on ResNet50.

$$\ell(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right) \tag{1}$$

2.3 Bidirectional Neural Network and Simple Neural Network

In the study of BDNN, besides compare the result to the previous CNN approach, we also set up a baseline using simple fully-connected neural network architecture with m hidden layers and n neurons of the hidden layer. We will look into the impact on accuracy for different values of m,n in the experiment section. Since our input is a 2071 dimensional vector and 1362 class labels, we set the input neurons to 2071 and the output neurons to 1362. We use ReLU (Eq. **??**) as an activation function and a 0.2 dropout rate to prevent overfitting. Also, Batch-Normalization layer will be added as a comparison to the deep neural network. Since this is a multi-class classification task, the Cross-entropy loss with softmax function is selected as the loss function. The overall network architecture is shown in Fig. 2.

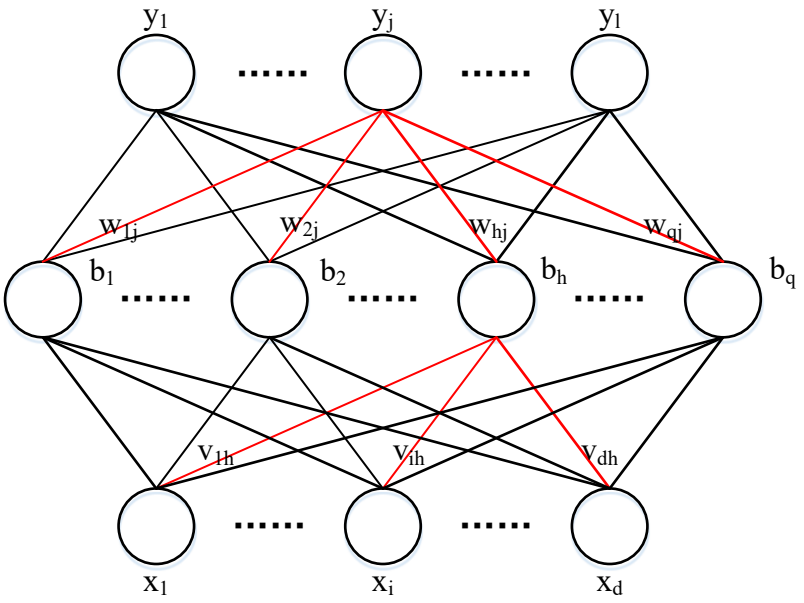

Fig. 2: The overall network architecture (bias not shown)

BDNN network has a similar structure as Simple NN. However, instead of regular NN training only the forward direction and loss backwards in the reversed direction, BDNN trains both directions. At the forward stage (denoted as Stage-F), the network train from left to right(as commonly stated direction, the input are placed on the left-hand side) and back-propagate as usual. However, at reversed stage (denoted as Stage-R), the network takes the original label as the current input, and uses the original input as the target, and train the same network from right to left. The weights for both directions are the same, with no modifications except transpose during the shift. Also notice that although the BDNN includes all the bias term for each layer, when performing the weights mapping, un-matched bias term will disregard and re-initialization. In order to make Stage-R trainable, a one-to-one input-label mapping is required. For our dataset, in order to make one label only maps back a particular input, an extra label is needed. In the original paper this is also introduced as "extra node". The formula to generate special labels for the VehicleX dataset is stated in Eq. 2.

$$U = (10000 * c + e)/10000 \tag{2}$$

where U is the value of the special label, c is the Camera ID and e is counting number corresponding to the specific input. Noted that other approaches should also acceptable as long as one-to-one mapping is satisfied.

After adding this special label as the output, Stage-F has 1363 output neurons. The number of input neurons, hidden neurons, hidden layers, and other hyperparameters remains the same as the simple NN for easy comparison. For Stage-F, the same Cross-entropy loss with softmax function is used, but only calculated towards the first 1362 dimension of output vector (classification result), since the last one is only for one-to-one mapping. For Stage-R, we use mean squared error loss (Eq. 3).

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = (x_n - y_n)^2 \tag{3}$$

2.4 Evaluation metric

Since it is a multi-class classification task, with 1362 classes to be precise, a simple classification accuracy may be too harsh towards our case. We instead use both Top-1 and Top-5 accuracy as our evaluation metric. Top-5 accuracy indicates that any one of our model's top 5 highest probability predictions matches the ground truth. The formula is shown in Eq. 4.

$$Accuracy = \frac{\sum_{i=1}^{N_{test}} \delta_{Top-K}(M_t(x_i), t_i)}{N_{test}} \tag{4}$$

Where N_{test} denotes N samples in test-set, x denotes corresponding input, M indicates model function, t denotes corresponding target.

3 Experiments, results, and analysis

3.1 Implementation and settings

We use the default training testing splitting provided by the dataset. We randomly shuffle the order at every epoch and drop the last partial batch for the training set. For each epoch, we use the validation set to do a quick evaluation on the current model and save only the best models that achieve the top Top-1 Accuracy and Top-5 Accuracy. Then we benchmark these best models on the testing set and get the final result.

For CNN approach, the batch size is set to 64. The image is resized to the target network config: 224 * 224 and normalized using ImageNet config to adopt the fit the pretrained model. We use stochastic gradient descent (SGD) as the optimizer with 0.9 momentum and learning rate set to 0.01, with weight decay to half at 20, 30, and 40 epochs. We trained the model for 45 epochs, and it takes about 5 hours to train on a single V100 GPU.

For simple NN, we use a 3 Layer MLP as our baseline, with 2048 neurons in the hidden layer. The batch size is 1024. The optimizer settings are the same compared to CNN approach. For simple NN, we train for 200 epochs, and the learning rate will be deducted to half at epoch 30, 80, and 130.

Generally, there are two ways to implement BDNN. One is by specifying the reverse direction on the original model. The other is to create a reversed model and sync all the weights when the training direction changes. We choose the second one. We change the training direction every 100 epochs and train the model for 600 epochs, which consists 3 Stage-F and 3 Stage-R. Everything else remains the same as simple NN.

3.2 CNN Approach – Different backbone

This experiment focuses on the performance of different backbone. For each model, we train the model according to the parameter listed in 3.1. Please notice that all the models listed here are all ImageNet pretrained models. The best validation accuracy for each configuration listed in Table 2 is listed in Table 3. We use these models to benchmark the result.

Table 2: Best validation accuracy for each model with different backbone

Model	Best Top-1 Epoch	Best Top-5 Epoch
VGG16	43	30
ResNet50	32	12
SEResNet50	44	27
DenseNet161	19	13

We find out that DenseNet161 in our situation performs the best compared to other models, especially on the Top-1 accuracy. The reason can be summarized into two parts: one is the network depth. The DesNet161

Table 3: Test accuracy of model with different backbone (%)

Model	Top-1 Accuracy	Top-5 Accuracy
VGG16-Top 1	94.347	99.049
VGG16-Top 5	94.353	99.049
ResNet50-Top 1	97.814	99.835
ResNet50-Top 5	97.603	99.835
SEResNet50-Top 1	95.265	99.373
SEResNet50-Top 5	95.093	99.313
DenseNet161-Top 1	98.574	99.934
DenseNet161-Top 5	**98.620**	**99.954**

has 161 layers in total, which is way deeper than other networks. The other reason, which is the primary reason, is the connection between each layer. DenseNet architecture connects each layer to another layer using "Dense Blocks". This architecture significantly accelerates the gradient flow, which is also why DenseNet architecture can achieve the highest accuracy at such few epochs (only 19 and 13 epochs for DenseNet161 for optimal result).

3.3 CNN Approach – Training from scratch

This experiment focuses on the usage of the ImageNet pretrained model. Due to the limitation of GPU resources, we only conduct this experiment on VGG model. Also, inspired by ScratchDet, to improve the performance of the scratch model, we added BN layer after each convolution layer to stabilize the network when training from scratch. The best validation accuracy for each configuration listed in Table 4 is listed in Table 5. We use these models to benchmark the result.

Table 4: Best validation accuracy for each model with different pretrained configurations

Model	Best Top-1 Epoch	Best Top-5 Epoch
VGG16	43	30
VGG16-Scratch	25	22
VGG16-ScratchBN	41	44

Table 5: Test accuracy of model with different pretrained configurations (%)

Model	Top-1 Accuracy	Top-5 Accuracy
VGG16-Top 1	94.347	**99.049**
VGG16-Top 5	**94.353**	**99.049**
VGG16-Scratch-Top 1	73.788	90.120
VGG16-Scratch-Top 5	73.729	90.166
VGG16-ScratchBN-Top 1	88.819	97.239
VGG16-ScratchBN-Top 5	88.760	97.246

We find out that without transfer learning, by training purely from scratch, there is an approximately 20% accuracy drop on Top-1, and approximately 10% accuracy drop on Top-5. We also notice that we can significantly improve the performance when training from scratch by adding BN layer in each convolution layer. This trick is useful when pretrained weights are not available.

3.4 CNN Approach – Various loss functions

This experiment focuses the performance of different loss functions. We adopted Arcface loss function: Additive Angular Margin Loss, which was originally designed for face recognition, and CosFace loss function: Large Margin Cosine Loss. For both loss functions, we set norm parameter (denoted as s) to 30 and margin (denote as m) to 0.5. Due to the limitation of GPU resources, we only conduct this experiment on ResNet model. The best validation accuracy for each configuration listed in Table 6 is listed in Table 7. We use these models to benchmark the result.

From the result, we notice that the original loss: cross-entropy outperforms those modified losses. This may be due to improper parameter settings for margin and norm and needs further investigation.

Table 6: Best validation accuracy for each model with different loss functions

Model	Best Top-1 Epoch	Best Top-5 Epoch
ResNet50-CE	32	12
ResNet50-Arcface	36	44
ResNet50-Cosface	39	35

Table 7: Test accuracy of model with different loss functions (%)

Model	Top-1 Accuracy	Top-5 Accuracy
ResNet50-CE-Top 1	**97.814**	**99.835**
ResNet50-CE-Top 5	97.603	99.835
ResNet50-Arcface-Top 1	97.636	99.808
ResNet50-Arcface-Top 5	97.636	99.802
ResNet50-Cosface-Top 1	97.642	99.709
ResNet50-Cosface-Top 5	97.616	99.703

3.5 BDNN Approach – Different number of hidden layer and neurons

This experiment focuses on the impact of both networks with different hidden neurons and hidden layers. For different hidden neuron experiments, we train 3 different models consisting of 1024 (Denote as S) hidden neurons, 2048, and 4096(Denote as L). The total number of layers of the network is set to 3, unchanged. For different hidden layer experiment, we train 3 different model that consists of original 1 hidden layer, 2 hidden layers, and 3 hidden layers. Thus the 3 models are 3 layer model, 4 Layer model and 5 layer model. All hidden layers got the same number of hidden neurons, 2048. The training epoch for the simple NN approach is set to 300 to ensure complete training and easy comparison, given that the BDNN approach has 300 epochs of stage-F.

The best validation accuracy for each configuration listed in Table 8 is listed in Table 9. We use these model to benchmark the result.

Table 8: Best validation accuracy for each model with different number of hidden neurons and hidden layers

Model	Best Top-1 Epoch	Best Top-5 Epoch
Simple NN-S	166	134
Simple NN	194	147
Simple NN-L	86	42
Simple NN-4Layer	299	299
Simple NN-5Layer	133	133
BDNN-S	442	281
BDNN	256	84
BDNN-L	88	50
BDNN-4Layer	500	500
BDNN-5Layer	422	422

Although overall, the BDNN can sometimes perform better on Top-5 accuracy than its baseline: Simple NN, but the highest Top-1 accuracy (excluding CNN approach) comes from simple NN. Also, when the original one-directional approach meets a multi-layer stacked MLP, the gradient will vanish or significantly weaken due to the backpropagation. Thus, it is very hard for the layer up-front to receive the proper gradients to update its weights. The network may fail to converge when the network is deep enough, like Simple NN-5 Layer Network. But the BDNN approach allows training the network from both sides, which somehow helps the gradient flow. This approach allows a fast converge speed and leads to better accuracy in a limited epoch range. Both approach's accuracy is way below the end-to-end CNN approach, which in this comparison, the ResNet50 model is the same model to perform feature extractor of VehicleX V1 dataset, the one BDNN, and simple NN uses.

3.6 BDNN Approach – Batch Normalization Enabled

As shown in Section 3.4, the 5-Layer model somehow fails to converge due to the network is too deep. This experiment focuses on this new technique and investigates if it could benefit both our models. We select the 5-Layer model as our baseline and added BN layer right after each Linear layer, with the last layer excluded. The other parameter remains the same as the baseline. The best validation accuracy for each configuration stated in Table 10 is listed in Table 11. We use these models to benchmark the result.

Table 9: Test accuracy of model with different number of hidden neurons and hidden layers (%)

Model	Top-1 Accuracy	Top-5 Accuracy
Simple NN -Top 1	38.647	73.927
Simple NN -Top 5	38.489	74.006
Simple NN-4 Layer-Top 1	17.745	53.150
Simple NN-4 Layer-Top 5	17.745	53.150
Simple NN-5 Layer-Top 1	1.155	5.045
Simple NN-5 Layer-Top 5	1.155	5.045
Simple NN-4 Layer-Top 1	17.745	53.150
Simple NN-4 Layer-Top 5	17.745	53.150
Simple NN-5 Layer-Top 1	1.155	5.045
Simple NN-5 Layer-Top 5	1.155	5.045
BDNN -Top 1	38.324	74.224
BDNN -Top 5	38.456	73.986
BDNN -4 Layer-Top 1	35.253	73.465
BDNN -4 Layer-Top 5	35.253	73.465
BDNN -5 Layer-Top 1	3.949	18.406
BDNN -5 Layer-Top 5	3.949	18.406
BDNN -4 Layer-Top 1	35.253	73.465
BDNN -4 Layer-Top 5	35.253	73.465
BDNN -5 Layer-Top 1	3.949	18.406
BDNN -5 Layer-Top 5	3.949	18.406
ResNet50-Top 1	**97.814**	**99.835**
ResNet50-Top 5	97.603	99.835

Table 10: Best validation accuracy for each model with or without BN (5 Layer model)

Model	Best Top-1 Epoch	Best Top-5 Epoch
Simple NN	133	133
Simple NN - BN	126	150
BDNN	422	422
BDNN-BN	86	73

Table 11: Test accuracy of model with or without BN (5-Layer Model) (%)

Model	Top-1 Accuracy	Top-5 Accuracy
Simple NN -Top 1	1.155	5.045
Simple NN -Top 5	1.155	5.045
Simple NN-BN-Top 1	48.930	81.971
Simple NN-BN-Top 5	49.003	82.050
BDNN -Top 1	3.949	18.406
BDNN -Top 5	3.949	18.406
BDNN-BN-Top 1	48.488	81.356
BDNN-BN-Top 5	48.349	81.409
ResNet50-Top 1	**97.814**	**99.835**
ResNet50-Top 5	97.603	99.835

We can notice that after implemented BN, the deep neural network converges, compared to the original one, which fails. Both BDNN and Simple NN benefit from the BN and achieve the best accuracy in their network family. Also, the accuracy gap between CNN approach and BDNN/Simple NN approaches shrink, but still, the end-to-end CNN approach has overall the best accuracy in both Top-1 and Top-5.

4 Conclusion and Future Work

This paper investigates the VehicleX multi-class classification task using Convolutional Neural Network and Bidirectional Neural Network. From the experiment results, it can be concluded that the end-to-end CNN approach with pretrained weights performs better than other approaches. During the experiment, we find that while BDNN may not benefit a lot when the simple baseline NN could able to converge, but when the network is deep and traditional NN fails to converge, by training both directions, BDNN could boost the network performance and aculeate the converge speed. Also, when pretrained weights are not available, insert BN could significantly increase the accuracy.

Recently, more and more advanced techniques like feature matching and feature re-ranking benefit a lot on other multi-class classification tasks. Future work could adopt those types of techniques into the vehicle classification tasks.

As for this project, due to the limitation of GPU computing resources, some experiments do not finalize on time. Future work could focus on the failure of Arcface and Cosface and also transform the bidirectional training concept to the current CNN and investigate the performance.

References

1. Chen, C., Bunescu, R.C., Xu, L., Liu, C.: Tone classification in mandarin chinese using convolutional neural networks. In: INTERSPEECH. pp. 2150–2154 (2016)
2. Cheung, S.Y., Coleri, S., Dundar, B., Ganesh, S., Tan, C.W., Varaiya, P.: Traffic measurement and vehicle classification with single magnetic sensor. Transportation research record **1917**(1), 173–181 (2005)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
4. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2019)
5. Gajda, J., Sroka, R., Stencel, M., Wajda, A., Zeglen, T.: A vehicle classification based on inductive loop detectors. In: IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No. 01CH 37188). vol. 1, pp. 460–464. IEEE (2001)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
7. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
8. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
9. Levi, G., Hassner, T.: Age and gender classification using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 34–42 (2015)
10. Mollahosseini, A., Chan, D., Mahoor, M.H.: Going deeper in facial expression recognition using deep neural networks. In: 2016 IEEE Winter conference on applications of computer vision (WACV). pp. 1–10. IEEE (2016)
11. Nejad, A.F., Gedeon, T.D.: Bidirectional neural networks and class prototypes. In: Proceedings of ICNN'95-International Conference on Neural Networks. vol. 3, pp. 1322–1327. IEEE (1995)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (May 2015)
13. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5265–5274 (2018)
14. Yao, Y., Zheng, L., Yang, X., Naphade, M., Gedeon, T.: Simulating content consistent vehicle datasets with attribute descent. In: ECCV (2020)
15. Zhu, R., Zhang, S., Wang, X., Wen, L., Shi, H., Bo, L., Mei, T.: Scratchdet: Training single-shot object detectors from scratch. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2268–2277 (2019)