Conducting Fine-Grained Vehicle Classification with Deep CNNs and Casper

Xin Shen

Research School of Computer Science, Australian National University
u6498962@anu.edu.au

Abstract. Building an effective Intelligent transportation system (ITS) is really necessary for modern cities. Vehicles are common objects that relate more with people's daily life in this system, and the number of vehicles increases fast in modern society. To better recognize vehicles in surveillance cameras, fine-grained vehicle classification based on vehicle IDs would be a useful approach that can be applied. In this paper, we conduct vehicle classification for real-world images on a dataset called VehicleX with 1,362 classes. Given the vehicle images, we firstly apply a deep convolutional neural network (CNN), ResNet50, to get their representations and conduct classification, since the ResNet50 model is really powerful to extract features from images. However, the number of neurons in each layer of ResNet50 is pre-defined when we want to use it. In order to dynamically add neurons in the network, we then utilize Casper algorithm to decide the structure between the final layer of the last stage and the output layer in a ResNet50. Finally, we test the performance of both ResNet50 and ResNet50-Casper models on VehicleX, and further compare their results with other baselines. Experimental results show that ResNet50 achieves similar Accuracy and mAP scores to those of ResNet50-Casper on the test set, but costs less time than ResNet50-Casper.

Keywords: Fine-grained vehicle classification, deep convolutional neural networks, Casper algorithm.

1 Introduction

Intelligent transportation system (ITS) is an indispensable part of modern cities and has aroused lots of interest in both academia and industry. Among different sub-systems in ITS, vehicle monitoring plays an important role in various situations including the management of traffic flow, automated parking systems, and security enforcement (Luo et al., 2018). Vehicle classification, aiming to separate given or detected vehicles into several classes (Shokravi et al., 2020), is an underlying task for vehicle monitoring.

For conventional vehicle classification, researchers used datasets (Krause et al., 2013[1]; Yang et al., 2015[2]) consisting of images that are captured by non-surveillance cameras, i.e., images in them are collected in good lighting conditions and have good resolutions. However, in real scenarios, vehicle images captured by surveillance cameras are influenced by many factors, such as viewpoint, lighting, and occlusion. In this sense, we apply VehicleX (Yao et al., 2019), a synthetic dataset that can approximate the attributes in real-world datasets, to provide robust and differentiated visual information in effective ITS. VehicleX is originally designed for vehicle re-identification task, which is *an image retrieval task* and tries to find concerned vehicles in different cameras, i.e., no classification results in the original paper can be used to compare the performance. However, for each vehicle ID, the dataset contains diverse images for it, based on above mentioned factors. Therefore, a fine-grained vehicle classification with 1,362 classes/IDs can be conducted on it.

To fulfill the fine-grained vehicle classification task, we use two kinds of deep convolutional neural networks, namely ResNet50[3] (He et al., 2016) and ResNet50-Casper that combines the ResNet50 model with Cascade Network with Progressive RPROP (Casper) (Treadgold, 1997). The former is an effective model with pre-defined network structure for image classification and detection, while the latter integrates a cascade network that add neurons one at a time between the final layer of the last stage and the output layer in ResNet50. Therefore, the small network structure in ResNet50-Casper can be dynamically decided based on the training performance.

We adjust the hyper-parameter setting and get several experimental results to compare, either in each model or between two models. The final results show that, ResNet50 and ResNet50-Casper achieves similar Accuracy and mAP scores on the test set of VehicleX, but the training process of ResNet50 is much faster. Besides, we also compare ResNet50/ResNet50-Casper with other baselines, and the results show that ResNet50/ResNet50-Casper trained on VehicleX is more capable to extract image features and conduct vehicle classification.

2 Method

In this section, we introduce two neural networks used in our fine-grained vehicle classification task: ResNet50 (He et al., 2016) and ResNet50-Casper (Treadgold, 1997).

[1] http://ai.stanford.edu/jkrause/cars/car_dataset.html
[2] http://mmlab.ie.cuhk.edu.hk/datasets/comp_cars/index.html
[3] There are several versions of ResNet, e.g., ResNet34, ResNet50, and ResNet152. Because of the restriction of our computational resources, it is more effective to use ResNet50 that has a fairly good performance for vehicle classification.

2.1 ResNet50

Convolutional neural networks (CNNs) consist of one or more convolutional layers and a fully connected layer at the top, as well as associated weights and pooling layers. This structure enables convolutional neural networks to use the two-dimensional structure of the input data. In theory, deep CNNs have better performance than shallow ones. However, deep networks are difficult to train due to the existence of degradation problem: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. To solve this problem, He et al. (2016) proposed a residual learning framework, as shown in the left of Fig. 1. Compared with the previous structure design, the authors explicitly reformulated the layers as learning residual functions with reference to the layer inputs, and introduced a shortcut connection into the feedforward network to perform identity mapping in the residual mapping.

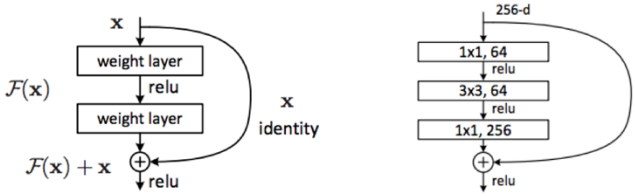


Fig. 1. A building block (left) of residual learning (He et al., 2016) and a "bottleneck" building block (right) for ResNet50.

Because of concerns on the training time, the authors modified the building block as a bottleneck design, as shown in the right of Fig.1. For each residual function, a stack of 3 layers was used instead of 2. To obtain a ResNet50, we just need to replace each 2-layer block in the 34-layer net with this 3-layer bottleneck block. The model architecture of ResNet50 is illustrated in Fig.2. Generally, there are five stages between the INPUT and OUTPUT. Two types of "bottleneck" are included in the model, denoted as bottleneck1 (BTNK1) and bottleneck2 (BTNK2). The input and output dimensions are different for BTNK1, while the input and output dimensions are the same for BTNK2. Please refer to the original paper for more details.

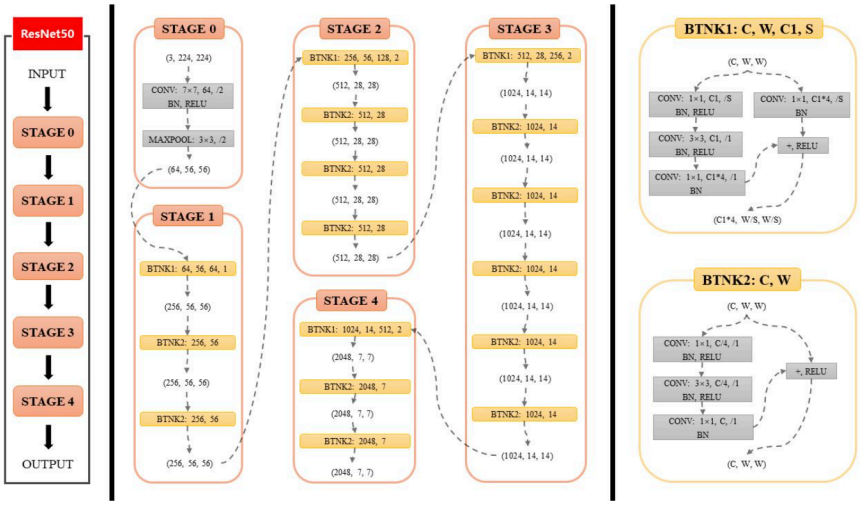


Fig. 2. The model architecture of ResNet50[4].

2.2 Casper

Casper (Treadgold, 1997) was proposed in 1997, and is based on the Cascade Correlation algorithm (Cascor) (Fahlman et al., 1990). Cascor is a powerful method for training neural networks with the usage of weight freezing and a correlation measure. Casper, on the other hand, uses a variant of RPROP (a gradient decent algorithm) (Fernandes et al., 2013) to train the whole network, and can produce more compact networks, which generalise better than Cascor.

When applying RPROP, Casper uses different learning rates for weights in different regions of a network. The corresponding relationship between learning rates and regions can be summarized as follows:

- Region 1 (with learning rate L1): weights connecting to the new neuron from previous hidden and input neurons.
- Region 2 (with learning rate L2): weights connecting from the new neuron to the output neurons.
- Region 3 (with learning rate L3): remaining weights (all weights connected to and coming from the old hidden and input neurons).

[4] Image Source: https://zhuanlan.zhihu.com/p/353235794

To illustrate Region 1 to 3, a case is shown in Fig.3, where a second hidden neuron has been added. In general, L1, L2 and L3 have the following relation: L1 >> L2 > L3, since initially new hidden neuron learns remaining error with little interference from other hidden neurons.

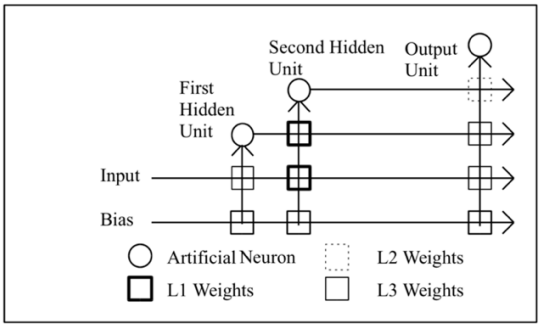


Fig. 3. The Casper architecture - a second hidden unit has just been added (Treadgold, 1997).

The algorithm contains following steps and the procedure is shown in Fig. 4:

1. Start connecting all the neurons in the input layer and all the output layers.
2. Add one neuron at a time, and connect the newly added neuron with all the previous neurons.
3. Use RPORP's gradient return method to reduce loss.
4. Repeat steps 2 and 3 until the loss falls to a tolerable range.

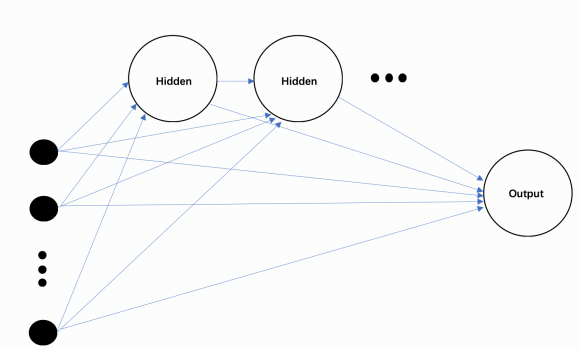


Fig. 4. The algorithm procedure of Casper. The hidden neurons between input and output layer is added one by one.

In Casper, the authors also used weight decay in RPROP. Since RMSprop (Graves et al., 2013) can be considered as the adaptation of RPROP algorithm for mini-batch learning and naturally contains the weight decay characteristic, it is more reasonable to improve Casper with the usage of RMSprop algorithm.

2.3 ResNet50-Casper

The network structure of ResNet50-Capser is shown in Fig.5. After the image is fed into the network, the features are extracted by the five stages of ResNet50. Then, we apply Casper between the final layer of the last stage and the output layer. That is, the “input layer” of the original Casper is replaced by the final layer of the 5th stage, and the “output layer” is still the Softmax layer used for classification. Between these two layers, Casper adds neurons one at a time to get an extra layer with flexible dimension.

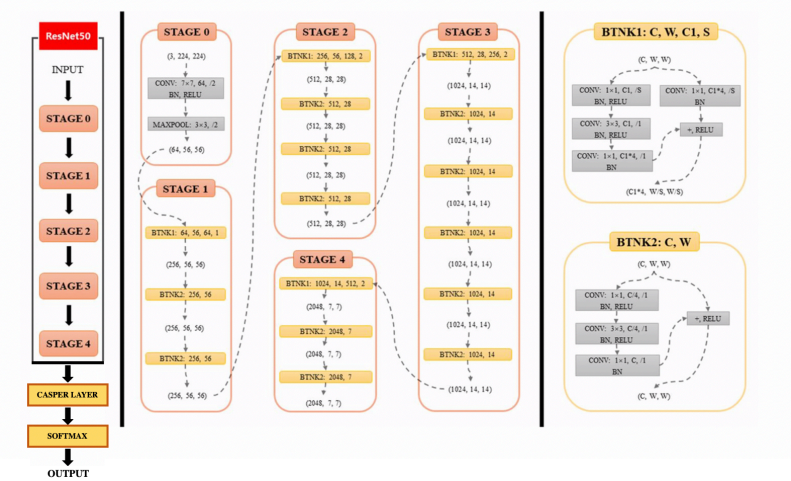


Fig. 5. The model architecture of ResNet50-Casper.

3 Experiments and Discussion

In this section, we conduct experiments to evaluate the performance of ResNet50 and ResNet50-Casper. We firstly introduce data pre-processing, hyper-parameter setting, baselines, and evaluation measures. Then we illustrate our results under the automatic evaluation. Finally, we do further comparisons and discussions over these two models.

3.1 Data Pre-processing

We use the dataset division in the original VehicleX, and the number of images in training/validation/test set is 45,438/14,936/15,142. There are 1,362 classes in this dataset, and each class appears in above three subsets. To illustrate the class distribution, we plot the histogram of each set in Fig. 6. We can find that data samples are not evenly-distributed among all classes. In the training set, most classes contain 30-39 images, while in the validation and test sets, most classes contain 10-14/15 images.

Fig. 6. Data distribution in training/validation/test set.

Each data sample is saved in a ".jpy" file as a color image with the size of 512*512*3. In order to increase the training speed and reduce the memory usage, we resize each image into 64*64*3. The ground-truth label (or target) of each data sample is the vehicle ID, which is included in the name of each ".jpg" file. For example, "0001_c001_33.npy" means the vehicle in this image belongs to class 0001.

3.2 Hyper-parameter Setting

For the implementation of ResNet50 with the best performance in our experiments, the input size and output size are set to 64*64*3 and 1,362, respectively. The Adam optimizer (Kingma et al., 2015) with a learning rate of 0.1 is used to train the model with batch size of 128 and epoch size of 60.

For the implementation of ResNet50-Casper, since the process of neuron adding is really time-consuming, we did limited trials. Finally, the sizes of the "input layer" and "output layer" to apply Casper are set to 2,048 and 1,362, respectively. That is, the output of the 5th layer in ResNet50 is spliced into a 2,048 dimensional vector. The batch size is 128. For the P value in equation 15+P*N (N is the number of currently installed neurons) defined in the Casper paper, we define it to be 1. The RMSprop optimizer (Graves et al., 2013), an improved version of RPROP (Fernandes et al., 2013), with a weight decay of 1e-5 and momentum of 0.9 is used to train the model.

Both of these two neural networks are implemented with Pytorch[5] and trained with GTX 1080Ti GPUs to accelerate the training process.

3.3 Baselines

Since the original paper for VehicleX is used for vehicle re-identification, which is a search problem, no classification-related results can be applied to fulfill the comparison with ResNet50 and ResNet50-Casper. We then design two competitive models here, MLP-Pretrained and Casper-Pretrained. The detailed settings are introduced as follows:

(1) MLP-Pretrained: The features of images in Vehicle-X are extracted from a ResNet with is pretrained on ImageNet (Deng et al., 2009). Then, these features are fed into a Multi-Layer Perceptron whose input size is 2,048 and output size is 1,362. We define the output size of the first hidden layer to be 1,024, and halves it in the following layers. The total number of hidden layers is 4. The Adam optimizer (Kingma et al., 2015) with a learning rate of 0.001 is used to train the model with batch size of 64 and epoch size of 35.

(2) Casper-Pretrained: The input layer size and output layer size are set to 2,048 and 1,362, respectively. The batch size is 1,024. The P value and other parts related to Casper are set the same to those of ResNet50-Casper.

[5] https://pytorch.org/

3.4 Automatic Metrics

Several metrics can be used to evaluate the model performance of a classification task. We use two common metrics for the multi-class classification task: Top-k Categorical Accuracy and Mean Average Precision (mAP) (Hutchinson et al., 2020). Due to the existence of data imbalance, the results of mAP are more persuasive than those of Accuracy. We first introduce following notations:

- $C = \{1, \dots, m\}$ is the set of all m classes.
- $y^{(i)} \in C$: the ground-truth annotation for input $x^{(i)}$ is a single vehicle class label,
- $\hat{y}^{(i)} \in \left\{p_1^{(i)}, \dots, p_m^{(i)}\right\}$ is the model output after conducting the Softmax function, where $p_j^{(i)} \in [0,1]$ is the probability that image $x^{(i)}$ depicts vehicle j.
- $\sum_{j=1}^{m} p_j^{(i)} = 1$ if the model uses softmax output (as is common)

Top-k Categorical Accuracy measures the proportion of times when the ground truth label can be found in the top-k predicted classes for that input. Top-1 accuracy, sometimes simply referred to as *Accuracy*, is the most ubiquitous. To calculate Top-k accuracy, let $\hat{y}_k^{(i)} \subseteq \hat{y}^{(i)}$ be the subset containing the k highest confidence scores for image $x^{(i)}$. The Top-k accuracy over the entire input set, where $\mathbb{I}$ is a 0-1 indicator function, is:

$$a_k = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{\hat{y}_k^{(i)}}(y^{(i)})$$

Mean Average Precision (*mAP*, also known as average Precision of all categories) is obtained by comprehensively weighting the average Precision (AP) of all categories. For a multi-class problem, using mAP as an indicator is a good choice. Formally, $\forall j \in \{1, \dots, m\}$, L_j is a ranked list of outputs such that $\forall a, b \in \{1, \dots, n\}, p_j^{(a)} \geq p_j^{(b)}$. The prediction at rank r in list L_j is a true positive if that image's ground truth label $y^{(i)}$ is class j (i.e. $TP_j(r) = 1 \; if \; y^{(i)} = j$). Using these lists $L_1, \dots, L_m$, precision up to rank k in a given list, interpolated AP over all ranks with unique recall values for a given class, and mAP are calculated as:

$$P_j(k) = \frac{1}{k}\sum_{r=1}^{k} TP_j(r)$$

$$AP(j) = \frac{\sum_{k=1}^{n} P_j(k) * TP_j(k)}{\sum_{k=1}^{n} TP_j(k)}$$

$$mAP = \frac{1}{m}\sum_{j=1}^{m} AP(j)$$

3.5 Results and Comparisons

Comparisons between ResNet50 and ResNet50-Casper

The results of ResNet50 on the test set are listed in Table 1. The epoch, batch size, and learning rate are abbreviated as EP, BS, and LR, respectively. We get a temporary local optimal result by adjusting the hyper-parameters: Accuracy = 97.32% and mAP = 98.48%.

Table 1. The results of ResNet50 on VehicleX test set. EP, BS, and LR represent epoch, batch size, and learning rate, respectively.

Model	Accuracy	mAP
Resnet50_BS128_EP40_LR0.001	59.37%	69.58%
Resnet50_BS64_ EP40_LR0.1	68.49%	78.40%
Resnet50_BS128_ EP40_LR0.1	78.41%	86.06%
Resnet50_BS128_ EP60_LR0.1	**97.32%**	**98.48%**

The process of adjusting parameters is as follows: The original setting of the listed hyper-parameters is: EP=40, BS=128, LR=0.1. We train the corresponding ResNet50, and get the training loss, validation loss, and Accuracy and mAP values on the validation set. Then, we plot the changing values in Fig. 7.

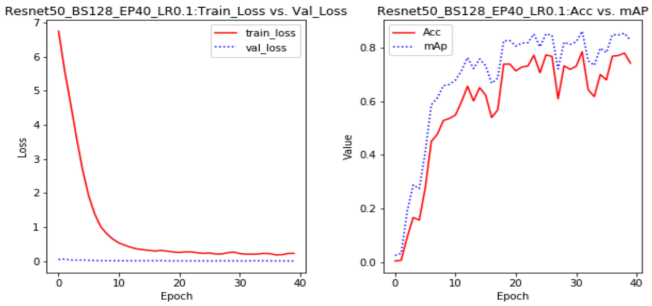


Fig. 7. The curves of training loss/validation loss/validation accuracy/validation mAP of model Resnet50_BS128_EP40_LR0.1.

From Fig. 7, we can find that the curves of training loss and accuracy loss are smooth, which indicate that the setting of training process is reasonable. Since BS=128 consumes more GPU memories, we try to set it to 64, but both Accuracy and mAP drop in a certain degree (Acc: 78.41%->68.49%; mAP: 86.06%->78.40%). Then, we decide to figure out the influence of learning rate value, and change it to 0.001, hoping a smaller learning rate could have a better performance. However, the results drop a lot, which means that 0.001 is a too small learning speed for this task and leads to bad performance.

Finally, we enlarge the epoch size to 60, and to our surprise, the results improve a lot, by 19% and 12% for Accuracy and mAP, respectively. We also plot the changing values of model Resnet50_BS128_EP60_LR0.1 in Fig. 8, and we can see that the curve of training loss begins to vibrate after epoch 40. Correspondingly, the validation accuracy and validation mAP begins to improve in a large margin. The reason why this happens could be that the model finds a local optimal, and goes down to that point fast after epoch 40.

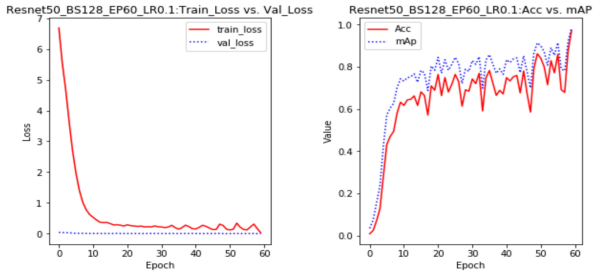


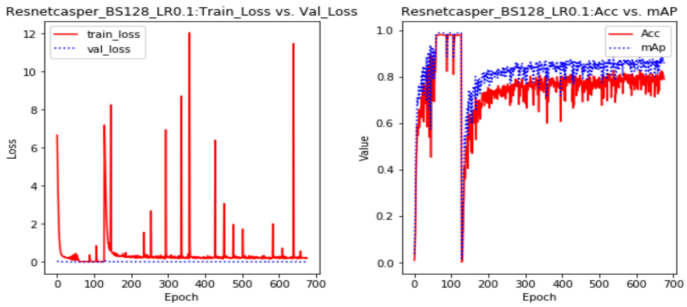
Fig. 8. The curves of training loss/validation loss/validation accuracy/validation mAP of model Resnet50_BS128_EP60_LR0.1.

For ResNet50-Casper, we get a temporary local optimal result by adjusting the hyper-parameter: Accuracy=98.23%, mAP=99.01%. Since the training process of ResNet50-Casper is really time-consuming, i.e., taking around 2.5 days to run 700 epochs with about 14 neurons added, we only apply two groups of settings for the model training, and the results are shown in Table 2. We find that a bigger learning rate helps a lot in this task to get a better result.

Table 2. The results of ResNet50-Casper on VehicleX test set. EP, BS, LR, and N represent epoch, batch size, learning rate, and the number of added neurons, respectively.

Model	N	Accuracy	mAP
Resnetcasper_BS128_EP700_LR0.001	1	58.15%	68.59%
Resnetcasper_BS128_EP700_LR0.1	14	**98.23%**	**99.01%**

We plot the training loss, validation loss, validation accuracy, and validation mAP in Fig. 9. We can see that for model Resnetcasper_BS128_LR0.1, the training loss does not decrease smoothly, this is because Casper tries to add one neuron at a time, which influence the entire model largely. However, model Resnetcasper_BS128_LR0.01 has a smooth training curve. There is only one neuron added in 700 epochs, and the entire model is more like a ResNet-50. Therefore, the training process is similar to that of ResNet-50.



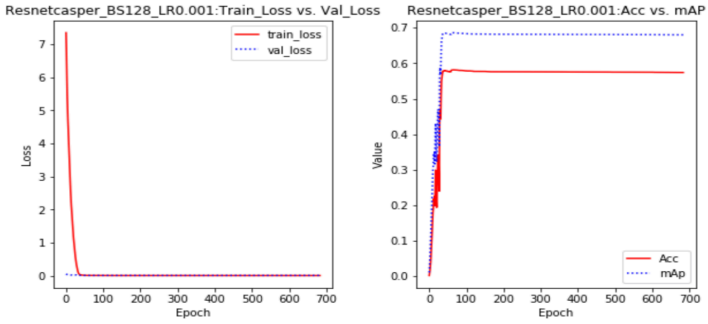


Fig. 9. The curves of training loss/validation loss/validation accuracy/validation mAP of model Resnetcasper_BS128_LR0.1 and Resnetcasper_BS128_LR0.001.

By comparing the Accuracy and mAP of the two models, we find that the results of ResNet50 and ResNet50-Casper are similar on the test set of Vehicle-X. We think this is because that ResNet50 itself is a powerful deep neural network for classification tasks, and there are lots of neurons in it. At the same time, adding a small amount of neurons to ResNet50 by Casper cannot improve the performance significantly. However, the training process of ResNet50 is much faster than that of ResNet50-Casper, which comes from the high degree of parallelism of ResNet50.

Comparisons between ResNet50/ResNet50-Casper and Baselines

Since the VehicleX dataset is new and cited by papers that also focus on vehicle re-identification (Zhu et al., 2020; He et al., 2020; Zheng et al., 2020; Khorramshahi et al., 2020), results in the original paper cannot be used for comparisons.

Here, we use the baselines MLP-Pretrained and Casper-Pretrained introduced above. The temporary local optimal results of them are shown in Table 3. To make clear comparisons, the best results of ResNet50 and ResNet50-Casper are also listed in this table.

Table 3. The temporary local optimal results of ResNet50, ResNet50-Casper, MLP-Pretrained, and Casper-Pretrained on VehicleX test set.

Model	Accuracy	mAP
ResNet50	97.32%	98.48%
ResNet50-Casper	**98.23%**	**99.01%**
MLP-Pretrained	16.64%	28.02%
Casper-Pretrained	12.03%	20.22%

From Table 3, we can see that ResNet50-based models outperforms the baselines significantly. We think that this is because the image distributions of VehicleX and ImageNet are not similar, the former is for vehicles, while the latter contains several objects. Then, the ResNet pretrained on ImageNet may not extract proper features for images in VehicleX, and further affect the performance of classification.

4 Conclusion and Future Work

After training and testing on the VehicleX dataset with two deep convolutional neural networks, we find that the results of ResNet50 and ResNet50-Casper are similar. ResNet50 has better parallelism ability, and we can add neurons in hidden layers in a large amount. Since the training process is fast, we could also adjust the hyper-parameters for multiple times according to the performance on the validation set, which can lead to better overall performance. For ResNet50-Casper, though the number of neurons in the hidden layer can be determined by the algorithm itself to reduce the redundancy of neural networks, every time a neuron is added, the overall loss of the network has to be calculated based on all training samples, which is a time-consuming process.

For the future work, we think there are two ways to improve the classification performance: (1) Design unsupervised or weakly supervised fine-grained vehicle classification models. At present, the cost of producing fine-grained vehicle classification data is relatively high. The deep learning method requires massive data samples to improve the generalization ability of the model, so how to learn features from unlabeled samples is a meaningful subtask in this field. (2) Use more vehicle-related information. The methods mentioned in this paper only take into consideration the vehicle IDs. In the future, the performance of fine-grained vehicle classification can be further improved by using the color information or appearance information of vehicles.

References

1. Luo, Z., Branchaud-Charron, F., Lemaire, C., Konrad, J., Li, S., Mishra, A., ... & Jodoin, P. M. (2018). MIO-TCD: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, *27*(10), 5129-5141.

2. Wu, W., QiSen, Z., & Mingjun, W. (2001, May). A method of vehicle classification using models and neural networks. In *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No. 01CH37202)* (Vol. 4, pp. 3022-3026). IEEE.
3. Yao, Y., Zheng, L., Yang, X., Naphade, M., & Gedeon, T. (2019). Simulating content consistent vehicle datasets with attribute descent. *arXiv preprint arXiv:1912.08855*.
4. Zhu, X., Luo, Z., Fu, P., & Ji, X. (2020). VOC-ReID: Vehicle re-identification based on vehicle-orientation-camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 602-603).
5. Zheng, Z., Ruan, T., Wei, Y., Yang, Y., & Mei, T. (2020). VehicleNet: learning robust visual representation for vehicle re-identification. *IEEE Transactions on Multimedia*.
6. He, S., Luo, H., Chen, W., Zhang, M., Zhang, Y., Wang, F., ... & Jiang, W. (2020). Multi-domain learning and identity mining for vehicle re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 582-583).
7. Khorramshahi, P., Peri, N., Chen, J. C., & Chellappa, R. (2020, August). The devil is in the details: Self-supervised attention for vehicle re-identification. In *European Conference on Computer Vision* (pp. 369-386). Springer, Cham.
8. Shokravi, H., Shokravi, H., Bakhary, N., Heidarrezaei, M., Rahimian Koloor, S. S., & Petrů, M. (2020). A review on vehicle classification and potential use of smart vehicle-assisted techniques. *Sensors*, *20*(11), 3274.
9. Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In Proceedings of the IEEE international conference on computer vision workshops (pp. 554-561).
10. Yang, L., Luo, P., Change Loy, C., & Tang, X. (2015). A large-scale car dataset for fine-grained categorization and verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3973-3981).
11. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
12. Treadgold, N. K., & Gedeon, T. D. (1997, June). A cascade network algorithm employing progressive RPROP. In International Work-Conference on Artificial Neural Networks (pp. 733-742). Springer, Berlin, Heidelberg.
13. Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.
14. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.
15. Hutchinson, M., & Gadepally, V. (2020). Video Action Understanding: A Tutorial. *arXiv preprint arXiv:2010.06647*.
16. Kingma, D. P., & Ba, J. (2015, January). Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.
17. Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.
18. Fernandes, P. O., Teixeira, J. P., Ferreira, J., & Azevedo, S. (2013). Training neural networks by resilient backpropagation algorithm for tourism forecasting. In Management Intelligent Systems (pp. 41-49). Springer, Heidelberg.