CasVer: A Cascade Network for vehicle classification with genetic algorithm

Ziang Liu

School of Computer Science, Australian National University
u5927429@anu.edu.au

Abstract. VehicleX is a large-scale synthetic dataset created in recent years. In this paper, we will focus on a classification problem based on VehicleX dataset. We present CasVer, a neural network using techniques from the cascade network algorithm employing Progressive RPROP (Casper) that can classify the types of vehicles. The accuracy for CasVer is 46%, which is higher than some common classification networks (40%). A defect of CasVer is the instability of network structure. We introduced genetic algorithm to improve the network and increased the accuray to 48%. Some advantages and disadvantages anout the model are also discussed in this paper.

Keywords: Neural networks, Casper, Classification, Genetic algorithm

1 Introduction

Classification by neural networks is a well-researched yet still challenging task, and vehicle classification is one of the topics that most people working on. A good vehicle recognition system will be useful in traffic control and will bring some convenience to our life. And This paper will focus on vehicle classification by neural network, where we use data form VehicleX as the input, and output the vehicle type.

In this work, we will use a simple classification networks as a baseline, and a network based on Casper [10] will be designed for comparison (CasVer). Genetic algorithm will be used to further improve the accuracy and robustness of the designed network.

VehicleX [11] is a dataset that contains 1,362 vehicles of various 3D models based on attribute distribution modeling. For those models, they focused on the domain gaps and took some key attributes (such as vehicle orientation, light, and camera pose) into consideration. Those data are very closed to the real images and have high consistency since they are 3D models. It can indicate the performance of our network on real images and more general cases while making the training simple. Another reason why we choose this dataset is that it is a large-scale (more than 75,000 images) dataset with detailed labeling. It is sufficient for us to work on different tasks.

For VehicleX, there are two kinds of data. The first one includes the images of different vehicles, the second one contains image features extracted from ResNet [2] which is pretrained on ImageNet. We decided to use the second one as the dataset. AlexNet [4] and VGG [9] are both famous image classification models. Their networks use convolutional layers to extract features and use fully connected layers for classification. A simple network is hard to extract feature well. However, a deep neural network like VGG requires a long time and a large dataset (such as ImageNet) for training. So, we decided to get the features by a pretrained model. ResNet is a model that won the 2015 ImageNet competition with an error rate of 3.57%, which is better than AlexNet (15.3%) and VGG (7.3). Thus, it is a nice choice to use features extracted from ResNet.

The output for our model will be the vehicle type rather than the vehicle id. There are 1,362 vehicles and only 45,438 images for training. Therefore, if we target the vehicle id, the classes are too many in terms of the dataset. Additionally, VehicleX is a synthetic dataset and may lose some details, which make it harder to distinguish different vehicles. In our previous work, a simple model only gives an accuracy of 17% on vehicle id. To get a reasonable and meaningful result, we only focus on vehicle types. Some types such as bus, truck and pickup are also included in the 1000 image categories for ResNet [2] output. Thus, our input and output will have high consistency.

The main contribution for this paper is that we design a vehicle classification model. Furthermore, we implement some early techniques on recent task and find their potential to be used nowadays.

2 Method and Architecture

In this part we will discuss the methods for data preprocessing and build a simple classification model. We will also show the key ideas in Casper [10] and implement that on our CasVer model. Finally, we will use genetic algorithm to improve this model.

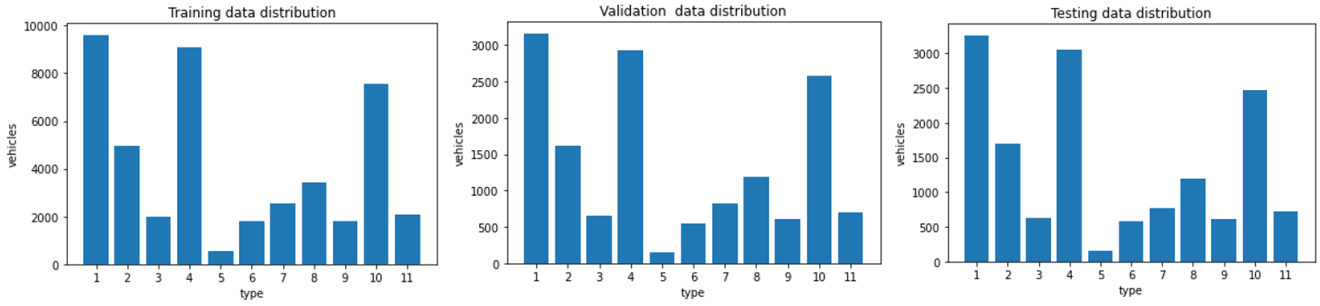

Fig. 1. The distribution of training, validation, and testing dataset. Where we plot the total number of vehicles for each type in the dataset.

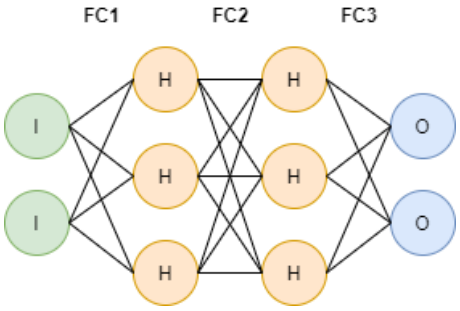

Fig. 2. The structure for Baseline Model. Where FC1, FC2, and FC3 are Fully Connected layers, and ReLU is used as the activation function.

2.1 Data Preprocessing

As we mentioned before, the input for CasVer are features extracted from ResNet [2]. In ResNet, batch normalization has been adopted and the input values can be directly used without normalization. We have three separate folders for training, validation, and testing (where contain 45,438 images, 14,936 images and 15,142 images respectively). And we load them directly as arrays with the size of 2048.

The dataset also provides detailed labeling in an XML file with the structure TrainingImages – Items – Item (which include different labels). The output for CasVer will be typeIDs form this file and link to the input data based on the imageName labels. There are 11 different types of vehicles in the dataset, so we labeled them from class 0 to 10.

The distributions of the datasets (training, validation, and testing) are shown in Figure 1. It seems that they are not very balanced, but the datasets are consistent, so they are still fine to be used. We can use sampling to balance the dataset, but this may also reduce the accuracy since we do not have many type-5 vehicles.

2.2 Baseline Model

According to AlexNet [4] and VGG [9], 3 fully connected layers can be used for classification. We built a simple network based on VGG. The structure is shown in Figure 2.

Rectified Linear Units (ReLUs) are used as activation functions to improve the efficiency. In VGG, dropout ratio was set to 0.5. However, we used a smaller dataset and thus adjust the ratio to 0.2. Based on some pretests, we set the learning rate to 0.0001 and use 0.9 for momentum. We used RMSProp to train the model and used cross-entropy for loss function. This is to match the CasVer model, and we will show that in Section 2.4.

2.3 Casper

Casper [10] is a Cascade network algorithm employing Progressive RPROP. There are three key points for Casper.

The first one is the architecture. Casper starts with a single hidden neuron and a new hidden unit will be added when necessary. This new hidden unit will be connected to all previous hidden and input neurons. In addition, the network is divided into three different regions with learning rate L1, L2 and L3. L1 is for all weights connecting the new unit and previous neurons. L2 is for the connection between the new unit and the output unit. And L3 is for the rest weights. For their values, we also need to ensure that L1 >> L2 > L3.

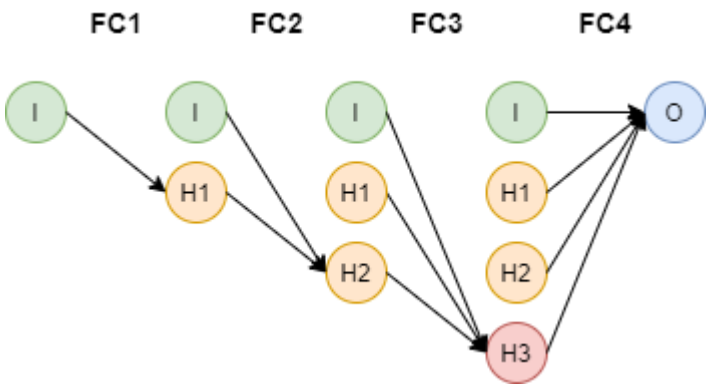

Fig. 3. The structure for CasVer network when we have 3 hidden units. I is the input unit. H1, H2, and H3 are the hidden neurons that were added one by one. O is the output unit. FC1, FC2, FC3, and FC4 are Fully Connected layers where the lines show the connection.

The second one is the gradient descent algorithm. RPROP [8] is used to train Casper which only depends on the sign of the gradient. Simulated Annealing is also applied to the weight decay for Casper, and gives the error gradient as shown below:

$$\delta E/\delta w_{ij} = \delta E/\delta w_{ij} - k * sign(w_{ij}) * {w_{ij}}^2 * 2^{-0.01*HEpoch} \quad (1)$$

Where HEpoch is the number of epochs since addition of last hidden unit and k is a parameter for weight decay.

The third key point is to determine the addition of new units. For Casper, a new neuron is installed when the RMS error decreases to a specific number. The network will stop when the loss cannot fall by 1% in a time period 15+P*N, where P is a parameter for the time period and N is the number of hidden units.

2.4 CasVer Network

The network we deigned is an implementation of Casper [10] on VehicleX [11], so we call it CasVer. Similar to Casper, we used three steps to design the network.

First, we designed a neural network with three input parameters input_size, num_classes and num_hid. Input size is the size of input array, which is 2048. Number of classes refer to the output size, which is 11. Number of hidden units is the number of neurons that are added to the network. Figure 3 shows the network when we have 3 hidden units. The hidden neurons are added one by one, and each hidden unit will be connected to the previous hidden and input neurons. FC1, FC2, FC3, and FC4 are all Fully Connected layers. According to Casper, we used Sigmoid as the activation function, and set the learning rate L1/ L2/ L3 to 0.02/0.0005/0.0001.

Different from Casper, we used cross-entropy for loss function instead of RMS, which is the equation below.

$$H(p,q) = -\sum_{x \in X} p(x) \log q(x) \quad (2)$$

According to [10], as we are doing a classification task, cross-entropy is better. Also, we used a large-scale dataset which means mini-batch is necessary for the training [8]. However, RPROP is not appropriate for mini-batch learning. So, we used RMSProp to train the model, which is closed to RPROP but able to deal with mini-batch learning.

The second step for CasVer is to determine the setting for some parameters. We need to find out a suitable batch size, number of epochs and the learning rates. Moreover, we can set momentum and weight decay for the optimizer in PyTorch. Test accuracy is the most important thing for the classification, so we use is to determine the parameters.

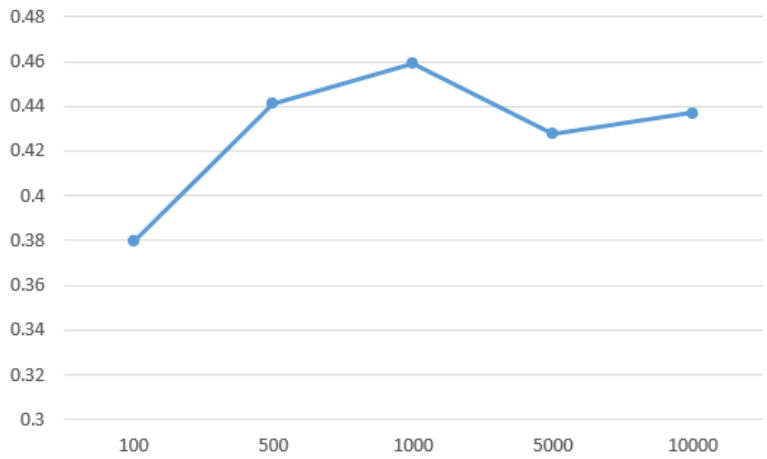

Fig. 4. The test accuracy for different batch size, where 100, 500, 1000, 5000, and 10000 indicate different batch size.

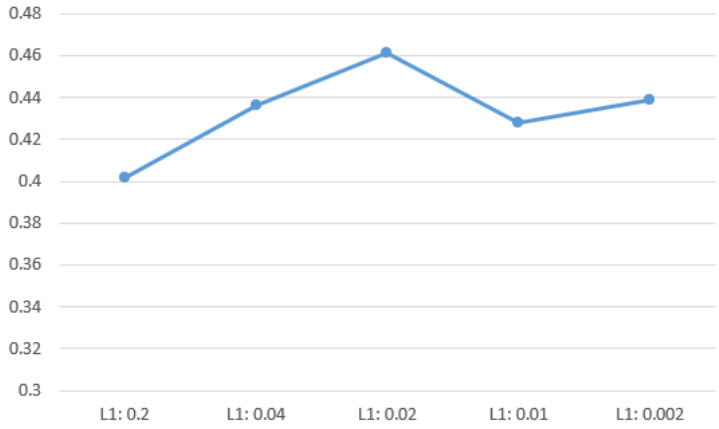

Fig. 5. The test accuracy for different learning rates, where 0.2, 0.04, 0.02, 0.004, and 0.002 indicate different values for L1.

Batch size. Large batch size may lead to poor generalization and small batch can make the model hard to converge [1]. We used 45,438 images for training, so we tested 100, 500, 1000, 5000, and 10000 as the batch size. As shown in Figure 4, 1000 is a sensible choice for batch size.

Number of epochs. CasVer is not a complex model and cannot get an excellent result for classification, so we do not need to set a large number for epochs. We found that the accuracy increases slowly after 300 epochs and set the epoch number to be 300.

Learning rate. According to Casper, the values of learning rates L1, L2, and L3 were set to 0.2, 0.005, and 0.001 respectively. However, our case is more complex and requires smaller learning rates. We kept that ratio and used 0.2, 0.04, 0.02, 0.004, 0.002 as L1 for testing. According to Figure 5, the final value we chose for L1, L2, and L3 are 0.02, 0.0005, and 0.0001 respectively.

Optimizer. We can set momentum and weight decay for the optimizer. Although the Casper uses weight decay, we found that setting a weight decay will reduce the test accuracy for our model. The momentum is usually set to 0.9, so we used 0.85, 0.9 and 0.95 for testing. As shown in Figure 6, the common value 0.9 is good to be used.

The final step is the method for the network to add hidden neurons. Classification is not easy for CasVer, and the loss will not always decrease during training. The basic idea for this network is that we should add units to make the network more complex when it is underfitting. And this is the reason why Casper increase the hidden neurons when a small loss is reached. However, we do not want the network to be overfitting because of too many hidden units. So, Casper stops when the loss become hard to decrease (when the loss cannot fall by 1% in a time period 15+P*N as we mentioned before).

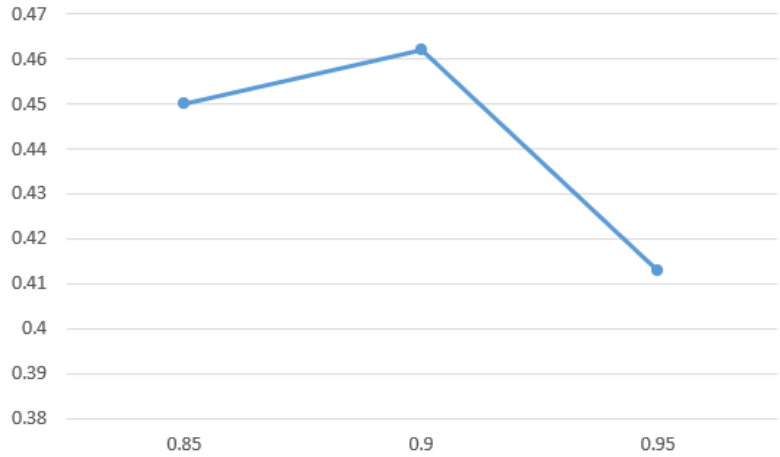

Fig. 6. The test accuracy for different learning rates, where 0.85, 0.9, and 0.95 indicate different momentum for the optimizer.

For CasVer, we used the validation data to test overfitting. If the accuracy for training dataset increases slowly, it is likely that the model is underfitting. However, we should also check the validation accuracy. If the training accuracy increases fast but the validation accuracy stops increasing, the network seems to overfitting, and we should freeze the network to be the final one. The approach is summarized in Algorithm 1 below.

Algorithm 1: Training CasVer

```
initialize the number of hidden units and the overfit state;
while not overfit do
        build CasVer based on the hidden number;
        if hidden number > 20 then
           overfit;
        train CasVer;
        if training accuracy increases < 3% then
           underfit;
        if validation accuracy decreases then
           if underfit then
              add hidden unit;
           else
              overfit;
end
```

We will start with one hidden unit according to Casper, then build and train the network. We check every 25 epochs. If the validation accuracy decreases, we assume that the following epochs will not bring much improvement on this model. Therefore, we need to determine whether we should use this model to be the final one or build a new model with an additional hidden unit. We now check the training accuracy. If the increase of accuracy is less than 3% (which is the value that gives best results in our experiment), we can use a more complex model to improve the performance. If the training accuracy still increases fast, it is likely that the model starts to overfit since the validation accuracy is decreasing. So, we stop training and use this model to be the final network. We also found that in some extreme cases, the number of hidden units will be too large, and we set the upper limit to be 20 (the model with more than 20 hidden units can hardly perform well in our tests).

This method may lead to the instability of the model structure. However, it is inevitable when we do complex tasks based on the early techniques (Casper). Therefore, we will introduce some additional techniques to solve this in following section.

2.5 Genetic Algorithm

Genetic algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation [3].

In our tests, we found that CasVer is not very stable when detecting overfitting and underfitting. CasVer will be initialized each time when a hidden unit is added, so the key idea for the network is just to find a suitable structure (number of hidden units). This is a task that can be done by genetic algorithm [7], where each model (or individual) is a

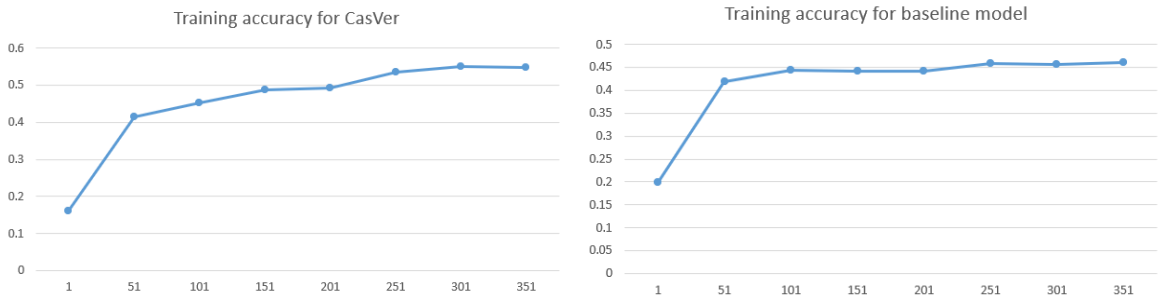

Fig. 7. The training accuracy for CasVer and baseline model, where 1 to 351 indicate the epochs.

trained architecture, and the model's accuracy on a separate validation dataset is a measure of the individual's quality or fitness.

Another thing we found during the experiment is that only add one hidden unit seems not the best choice, and we want to figure out the best number of hidden units to be added each time. So, for the genetic algorithm, the individuals will be CasVer networks with different number of hidden layers and different number of hidden units to be added in each layer. We use the validation accuracy to be the fitness score to each individual, and model with higher fitness value has higher chance to be selected. In this case, the relation between model structure and model behavior may not be linear, so we did not use crossover to get next generation. The idea is more like self or asexual reproduction, each offspring only has one parent, and the mutation will increase or decrease the number of hidden layers and units.

For our work, we initialize population by building 10 CasVer models with random number of hidden layers and units. We get a best model structure after 5 generations with a mutation rate of 0.2. Since the individuals are a bit complex, we used small population and generation number. The mutation will not change the model significantly, so we used large mutation rate. Moreover, the number of hidden layers is between 1 and 10, the number of hidden units to be added in each layer is less than 100, and therefore no extreme cases will happen.

3 Results and Discussion

In this part we will show the training of CasVer and the results. Those results will be compared to some simple networks. In addition, we will talk about the result after the implementation of genetic algorithm. And finally, we will discuss the performance of CasVer on this problem and the limitation for CasVer.

3.1 CasVer Result

As we mentioned before, CasVer is based on early techniques and a large-scale vehicle dataset is used for training. So, the training accuracy will become hard to increase after 56% without overfitting. Despite this, the testing accuracy will still be more than 45% which is good enough for this network. The loss depends on batch size and is not very clear to be used for showing the training. We used the accuracy instead and an example of training is shown in Figure 7. The increasing of training accuracy stops at about 200 epochs.

The results for 5 tests are shown in Table 1. We can see that the testing accuracy is around 46% without much fluctuation. However, the number of hidden units seems to be very random. This is because we introduced validation dataset for overfitting checking which bring many uncertainties. Moreover, since we aimed to implement Casper [10], some design, such us the optimizer may not be the best choice. We also found that the training accuracy may start to fluctuate at a very early step. Therefore, the number of hidden neurons can be very different for different tests.

3.2 Baseline Model Comparison

We also built a simple neuron network for comparison. The detail of this network is covered in Section 2.2. This paper aims to find out the advantages and limitations for CasVer, so we used cross-entropy loss and RMSprop for the baseline model, which is the same as CasVer.

The training accuracy for baseline model is shown in Figure 7. We can see that it stops at about 45% at a very early stage. Compared to this model, CasVer will take longer for training but also perform better.

We did 5 tests on this simple network, and the result is in Table 2. The average test accuracy is 39.74%, which is smaller than CasVer (45.82%). In addition, the performance of CasVer is more stable than this network.

Table 1. The test accuracy and the final number of hidden units for 5 different tests on CasVer.

Test	Test accuracy	Number of hidden units
Test 1	45.69%	7
Test 2	46.15%	3
Test 3	46.07%	2
Test 4	45.41%	1
Test 5	45.81%	3

Table 2. The test accuracy for 5 different tests on simple network.

Test	Test accuracy
Test 1	39.81%
Test 2	40.54%
Test 3	39.70%
Test 4	38.64%
Test 5	40.03%

3.2 CasVer with Genetic Algorithm

As we mentioned before, the structure of CasVer is very different for different tests, and therefore we use genetic algorithm to find a best structure.

The final model we got is a CasVer network with 6 hidden layers and we add 10 units in each hidden layer. The final test accuracy is 48.1%, which is higher than a CasVer (46%).

3.3 Discussion

CasVer has better performance than a baseline classification model for VehicleX [11]. However, this accuracy is still much less than today's network. The main problem for CasVer is that the input size is 2048 and we only add one hidden neuron each time, so that the hidden layers are dominated by the inputs. And sometimes it happens that the model structure become very random as we mentioned before. We used genetic algorithm to improve these and brought a noticeable increase on accuracy. In addition, VehicleX is not a balanced dataset for type classification, and this may also affect the performance for our model.

For CasVer, the commendable part is the idea of adding hidden neurons and changing learning rates. This is the reason that it gives better result than a baseline model. We also found that genetic algorithm works well with CasVer. CasVer does not have a stable way for adding and stop adding units, and genetic algorithm can provide a nice network structure, which helps to improve the accuracy and robustness for CasVer.

However, the training time for CasVer will be longer than the baseline model. And if we apply genetic algorithm, this time will be increased significantly. This means our model may not behave well when dealing with complex tasks (requires long time and large memory).

4 Conclusion and Future Work

In this paper, CasVer network is designed to solve vehicle classification problem on VehicleX dataset. The accuracy for this network can be more than 46%, which is higher than a baseline model. We presented the settings and algorithm for CasVer in this work. A noticeable problem for CasVer is that the model structure is not very stable, we found that genetic algorithm can be used to improve this and will give a better result on test accuracy.

For future work, we can focus on the image dataset rather than the image features extracted from a pretrained model. VehicleX is only a synthetic dataset, and we can work on more datasets for real-world vehicles. Additionally, we can simplify the genetic algorithm on CasVer to achieve a higher efficiency.

References

1. Effect of batch size on training dynamics, https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR (2016)
3. Introduction to Genetic Algorithms, https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
4. Krizhevsky, A., Sutskever, I. and Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems (2012)
5. Li, M., Zhang, T., Chen, Y., Smola, A.J.: Efficient mini-batch training for stochastic optimization. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (2014)
6. Mannor, S., Peleg, D., Rubinstein, R.: The cross entropy method for classification. In: Proceedings of the 22nd international conference on Machine learning (2005)
7. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. In: International Conference on Machine Learning (2017)
8. Riedmiller, M., Braun, H.: Rprop-a fast adaptive learning algorithm. In: Proc. of ISCIS VII (1992)
9. Simonyan, K., Zisserman, A.: VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. In: ICLR (2015)
10. Treadgold, N.K., Gedeon, T.D.: A Cascade Network Algorithm Employing Progressive RPROP. In: International Work-Conference on Artificial and Natural Neural Networks (1997)
11. Yao, Y., Zheng, L., Yang, X., Gedeon, T.D.: Simulating Content Consistent Vehicle Datasets with Attribute Descent. In: ECCV (2020)