In the Name of God

University of Tehran

Faculty of Electrical and Computer Engineering

Neural Networks and Deep Learning

Assignment 2

c

Assignment Details	
Student Name Student ID	Mohammad Taha Majlesi 810101504
Submission Deadline	1404/01/26 (April 15, 2025)

Prepared by: Mohammad Taha Majlesi - 810101504

Contents

List of Figures

List of Tables

1 Question 2 – Implementing a Car Classification System using VGG16 and SVM

1.1 Image Review

The dataset used for this section was collected in 2019 by Occulta Insights. These images were gathered from web pages using web crawlers. According to the dataset description, the accuracy of the existing labels is approximately 95%. The dataset includes 38 different car models from the Toyota brand.

First, we observe the frequency distribution of the different car models in the chart below.

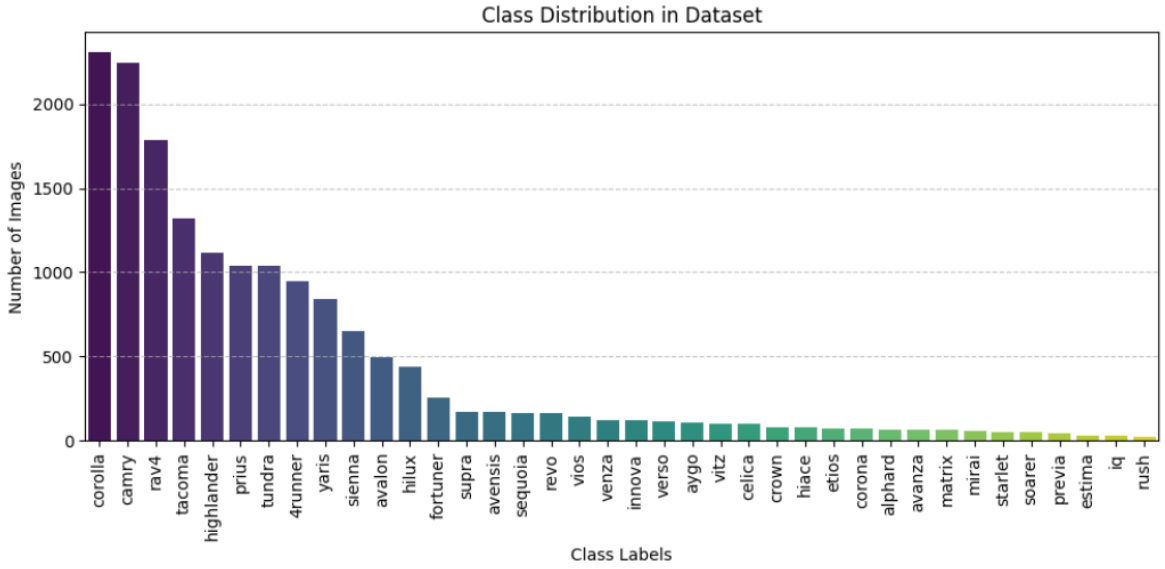


Figure 1: Distribution of image classes before loading.

As is evident, the class distribution is not balanced. For instance, there are over two thousand images of the Camry model, while there are fewer than one hundred samples of the Vitz model. Due to the significant impact of having more images on model performance, we selected the ten car models with the highest frequency as our classes. The second reason for this choice is that after reviewing the models selected in the reference paper, we noticed that apart from the C-HR and Land Cruiser models, which are not present in our dataset, and the Avalon model, which is the eleventh model with the most images, the other seven classes from the paper are among the models we have selected. This might indicate that the paper also attempted to select more common car models, as our dataset, which was crawled from the internet, also has more images of these models.

Some of the images in the dataset were corrupted and could not be accessed. We first identified these images and, after loading the images of the selected classes, we removed the corrupted ones. The frequency distribution for the selected images is as follows:

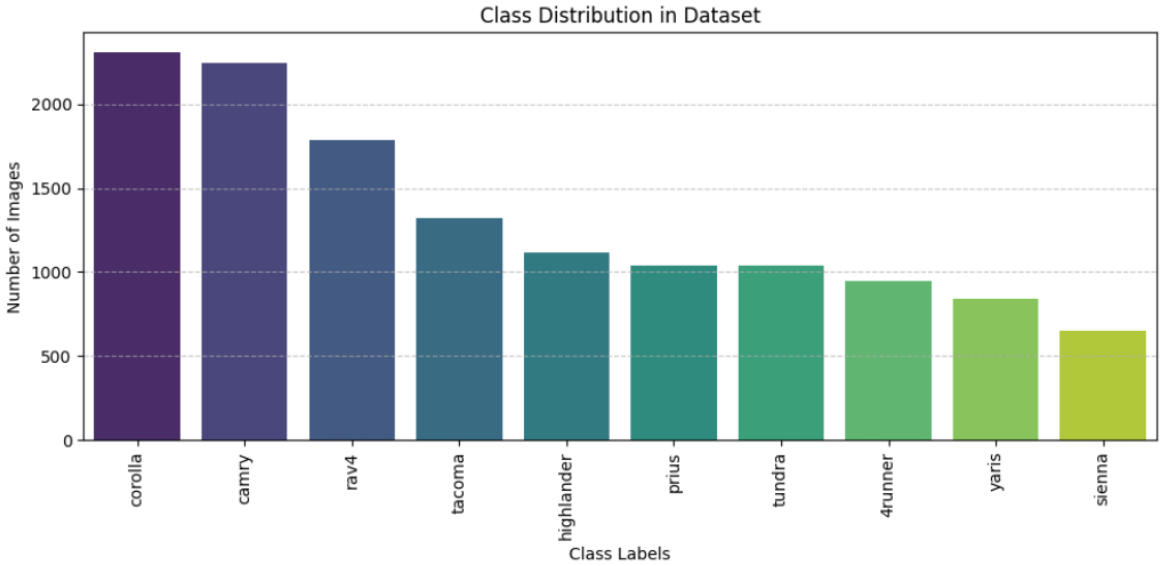


Figure 2: Frequency of images for the selected classes.

We can observe some sample images from the selected classes in Figure 3. It is noticeable that some images are taken from inside the car or are close-ups of parts like the wheel or headlight, which can make classification challenging.

Figure 3: Sample images from the dataset.

1.2 Feature Extraction

The reference paper provides no details about the training process or the model architectures. Therefore, we will try to achieve a similar performance by experimenting with different architectures and parameters. Also, since the AlexNet model is not available in the TensorFlow framework, we will use pre-trained PyTorch models.

After separating the feature extraction part of the two models, once with the VGG16 model and once with the AlexNet model, we extract the features present in the images. For this, we separate the final fully connected part of these two models and use the initial part, which is a convolutional network, for feature extraction.

We have 13,292 images in total. With the AlexNet model, we generate 9,216 features, and with the VGG16 model, we generate 25,088 features for each image. We know that the AlexNet model is an older and less complex model, which is why its output dimension is also smaller, and a lower performance can be expected from it.

1.3 Model Building

First, we separate the image features into an 80-20 ratio to have one set for training and another for evaluation. To train the VGG16 and AlexNet models, the extracted features must be passed through a neural network for training. To do this, we restore the final parts of the models that perform the classification task and replace only their final layer with a 10-neuron layer to place our images into one of the 10 classes.

For training the models, we use the Adam optimizer with the common learning rate of 0.001 and the cross-entropy loss function. We also set the batch size to the common value of 32. We train the models for 15 epochs.

To create the convolutional model, since the paper provides no explanation of its architecture, we decided to use the architecture we used for the previous section, and only in the last layer, we place 10 neurons instead of 3. A summary of the model architecture from the previous section is available in Table 2. Thus, the number of parameters for the three models is as follows:

Table 1: Comparison of the number of parameters for the three models: AlexNet, VGG16, and CNN.

	VGG16	**AlexNet**	**CNN**
Trainable parameters	119,586,826	54,575,114	2,462,794

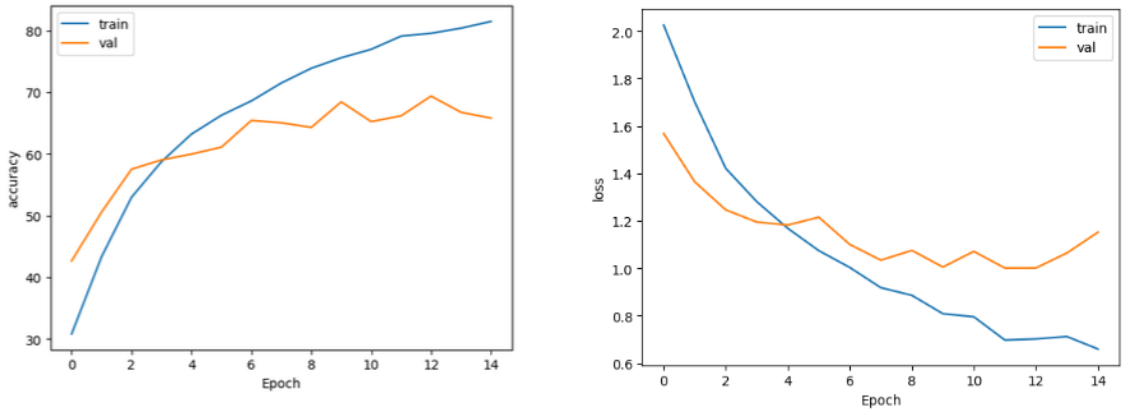
It can be seen that the VGG16 and AlexNet models have a much larger number of parameters compared to the CNN model. The VGG16 model also has about twice the number of parameters as the AlexNet model. This could be one of the reasons for the better performance of VGG16 compared to AlexNet, and AlexNet compared to CNN. Of course, we do not know about the number of parameters of the CNN model used in the paper, and it might have a much larger number of parameters.

The architectures of the AlexNet and VGG16 models have differences, including that the AlexNet model has 8 layers while the VGG16 model is deeper with 16 layers. Also, in the VGG16 model, larger filters with sizes 11 and 5 are used in the first two layers, and the number of filters of size 3 is used. In general, the VGG16 architecture, by increasing the parameters and the number of layers, has achieved better performance than AlexNet.

Finally, we form the main model of the paper. We create an SVM model with a linear kernel and prepare to train it with the features obtained from the VGG16 model.

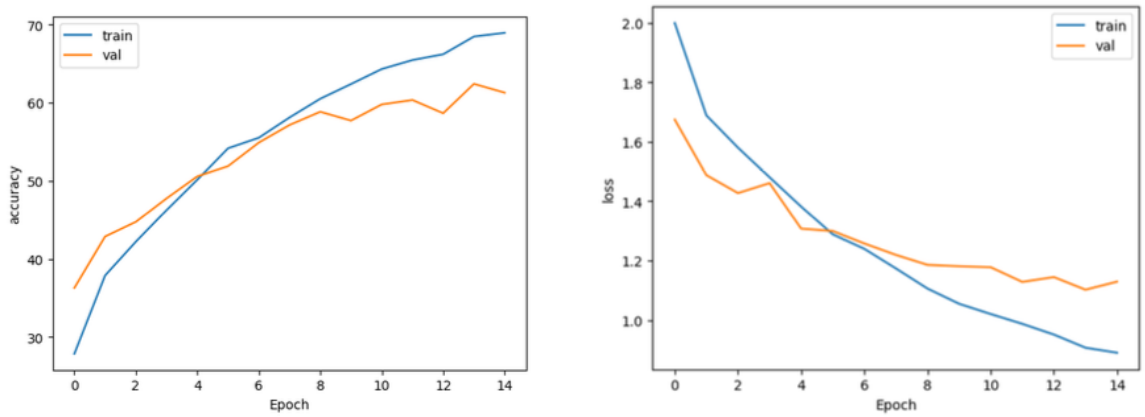
1.4 Training and Evaluation

Now we train the models. The changes in loss and accuracy of the models in different epochs during training are shown in the figures below.



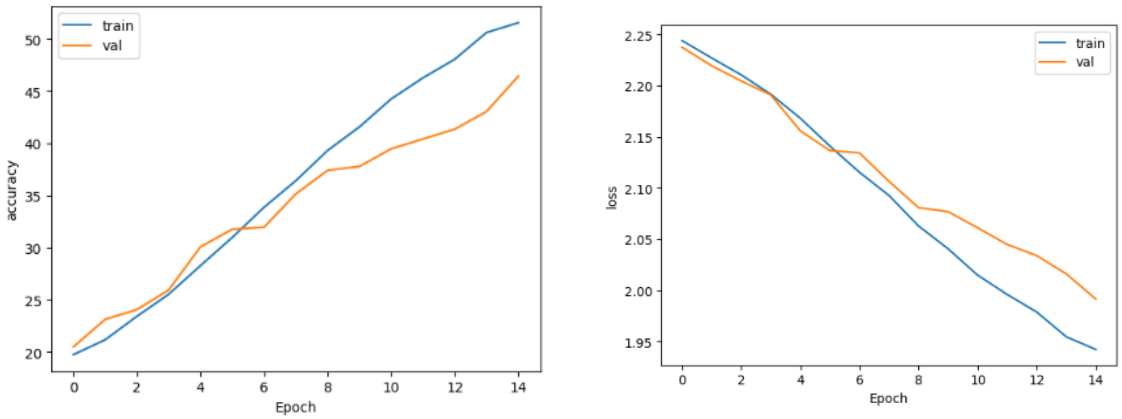
(a) VGG16 Accuracy **(b)** VGG16 Loss

Figure 4: VGG16 model training history.



(a) AlexNet Accuracy **(b)** AlexNet Loss

Figure 5: AlexNet model training history.



(a) CNN Accuracy **(b)** CNN Loss

Figure 6: CNN model training history.

For a better evaluation, we need information about the four evaluation metrics. In the first part, we provided relatively complete explanations about these four metrics. All four are among the most common metrics and each considers a specific type of error in the model, and depending on the type of data and the importance of one type of error over another, using one metric may be more useful than another.

In general, the first metric, accuracy, is the most widely used metric in machine learning. This metric has high interpretability and, due to its simplicity, is widely used. This metric shows the ratio of all correctly classified data to all misclassified data, as follows:

$$\text{accuracy} = \frac{TP + TN}{FP + FN + TP + TN}$$

In the above relation, TP or True Positive indicates the number of data points correctly classified as positive, and TN or True Negative indicates the number of data points correctly classified as negative. Similarly, FP or False Positive indicates the number of negative class data points that the model has incorrectly classified as positive, and FN or False Negative indicates the number of positive class data points that the model has incorrectly classified as negative. As is clear from the above relation, the accuracy metric does not differentiate between the two types of errors and the two types of correct classifications and gives them equal weight. This is the main reason for preferring other metrics in situations where one specific error is more important.

As is clear from the definition of the four metrics TP, TN, FP, and TP, these metrics are used for the classification of two-class data, and consequently, the four metrics accuracy, precision, recall, and f1-score are also used for the classification of two-class data. In cases where the data of interest has more than two classes, as in this exercise, to solve this problem, we can take a kind of average of the value of each metric between the different classes. However, in this exercise, we intend to examine the performance of the weak model by examining each class individually, so we calculate each metric for each class separately.

The next metric, precision, only prevents the FP error of the model. This error occurs when the model incorrectly classifies negative class data as positive. For example, for detecting spam emails as the positive class, we do not want any non-spam emails to be classified as spam, so using this model can be helpful. Thus, the precision value can be calculated from the following relation:

$$\text{precision} = \frac{TP}{TP + FP}$$

The next metric is recall. This metric, unlike the previous metric, prevents the creation of FN error. This error occurs when the model incorrectly classifies a data point that should have been a member of the positive class as a member of the negative class. For example, for diagnosing the coronavirus, the importance of this metric is twofold because even not diagnosing one sick person can endanger the lives of many others, whereas mistakenly diagnosing a healthy person as sick is less harmful. Thus, recall can be calculated as:

$$\text{recall} = \frac{TP}{TP + FN}$$

The last metric is the F1-score, whose goal is to give importance to both precision and recall. For this, this metric calculates the harmonic mean of the two mentioned metrics. One of the advantages of the harmonic mean over the other two means is that

even if one of the two metrics, precision or recall, is low, the f1-score will also be low, and only if both metrics are high will this metric also have an acceptable value. This metric is also calculated as follows:

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

In this section, a specific model is not more important than the other models, and our goal is to reduce both errors, but the number of images from different classes is different, and to solve this problem, we can take a weighted average for one of the metrics and consider it for evaluation. However, to examine each class and the effect of the number of images of the classes on the performance, we calculate each metric for all classes.

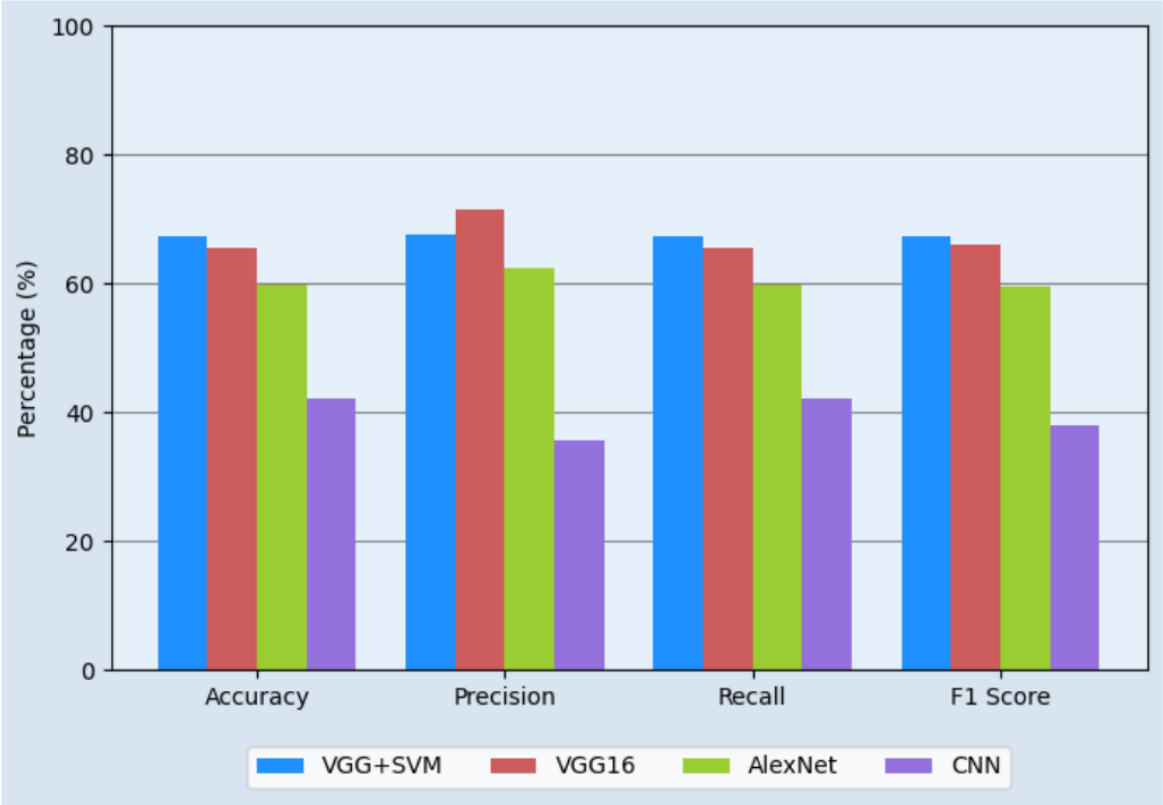


Figure 7: Comparison of the performance of the models with different metrics.

In the chart above, which is drawn with a weighted average of the metrics for the classes, it is clear that the performance of the two models AlexNet and VGG16 is consistent with the paper and has even improved slightly, but the VGG+SVM and CNN models are weaker. The reason for the weaker performance of the CNN could be due to the difference in architecture. Of course, we must consider that the images used by us are also different from the paper's images. But the VGG+SVM model, which is the main model, does not have acceptable performance.

Another useful metric for evaluating models for each class is the confusion matrix. In the cell at row x and column y of this matrix, we keep the number of samples of class x that have been mistakenly classified as class y. To make the values of this matrix easier to see, we usually color each cell relative to its value. In an ideal model without any

errors, all the cells on the main diagonal of the matrix are filled, and the rest of the cells will be empty. If a cell in the table besides the main diagonal has a large value, it can indicate the model's error in distinguishing those two classes, and as a result, we can investigate those two classes further.

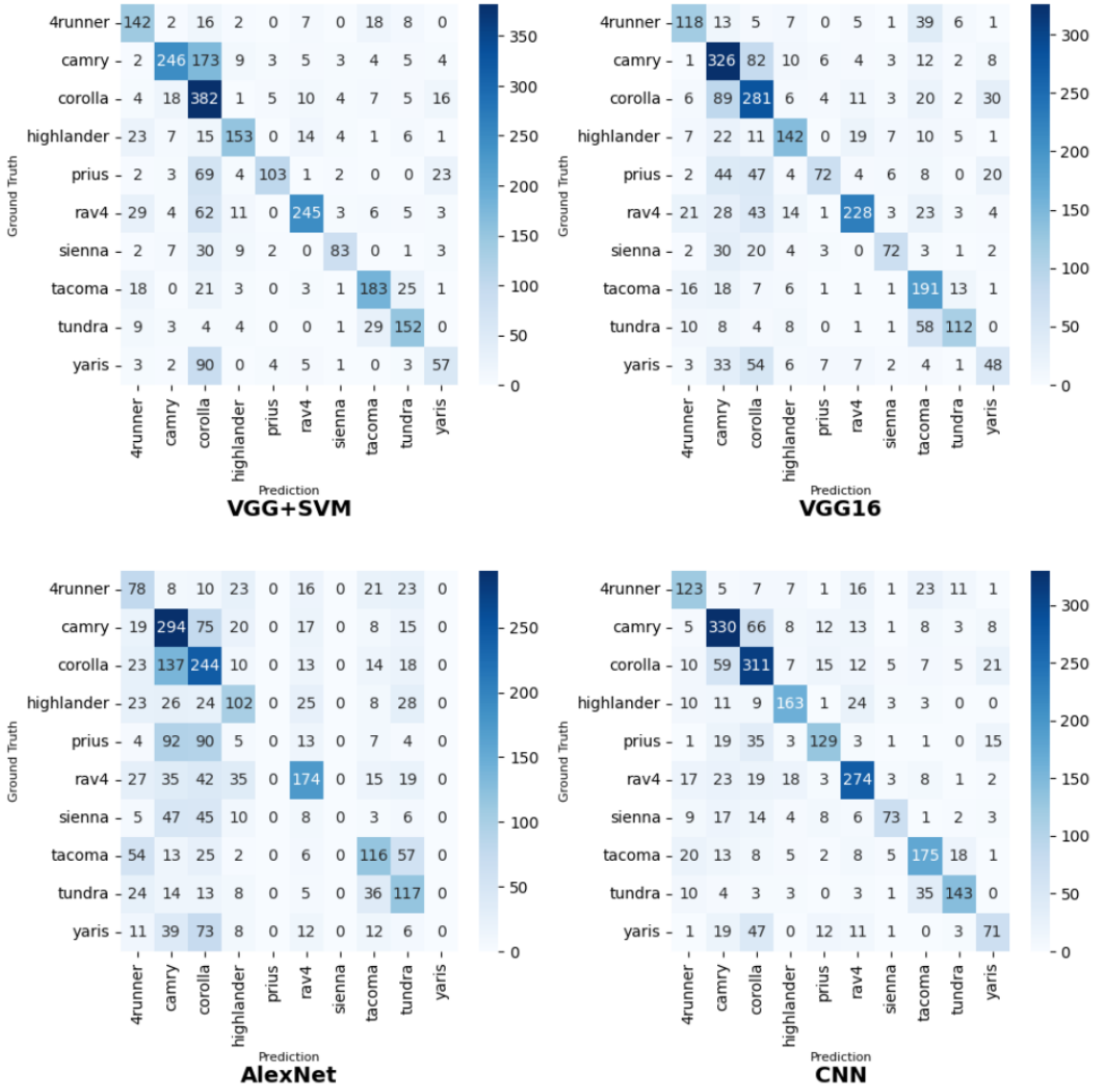


Figure 8: Confusion matrix for each of the four models.

From the confusion matrices, it is clear that all four models, and especially the two models AlexNet and VGG+SVM, have difficulty in distinguishing the two classes Camry and Corolla. In the table below, the numerical values of each metric for each model, which are averaged over the classes with the macro method, are shown.

Table 2: Values of the metrics for each model.

	Accuracy	**Precision**	**Recall**	**F1 Score**
VGG+SVM	67.39	67.75	67.39	67.27
VGG16	65.66	71.53	65.66	66.15
AlexNet	59.80	68.58	59.80	59.56
CNN	42.31	35.69	42.31	38.07

It is clear that the main model of the paper, i.e., VGG+SVM, has a much weaker performance compared to the paper, and in the following, we try to improve its performance.

1.5 Adding Augmented Images

Data augmentation is generally a method that has many applications in computer vision and natural language processing because, with its help, the number of training images can be multiplied, which reduces the cost of data collection. In some fields like medicine, due to the high cost of diagnosis and data collection, especially for rare diseases, creating augmentation can help in training the models. Also, by multiplying the data, if the changes are made up to a limit, we can increase the generalization ability of the model's predictions on unseen data.

In the previous section, because the images were medical, our hands were tied in the type of changes in the images to create augmented images. In this section, because the images are scraped from the internet and the type of camera or photography is not specific, and also given the diversity that exists in car images in the real world, we can change the images with various methods to generate new images. Thus, we apply the following changes to the image of each class to bring the number of images of each class closer to each other.

As explained in the previous section, the changes made to create augmentation should not be such that they create images that are not likely to occur in the real world, and on the other hand, they should not be too similar to the images themselves to cause the model to overfit the images.

By taking each image, we first change the image slightly in terms of geometry. Geometric changes are logical because cars can be photographed from different views and angles. Specifically, we use reflection around the vertical axis, cropping a part of the image, and changing the angle. We believe that reflection around the horizontal axis is not correct for these images because usually in all images, the car wheels are on the ground, and creating images where this is not the case not only does not help the model but may also harm its performance.

Apart from geometric changes, due to environmental and temporal changes and differences in imaging devices, we expect the features related to the color of the images to be different. For this reason, we randomly change the light, contrast, and color of the images.

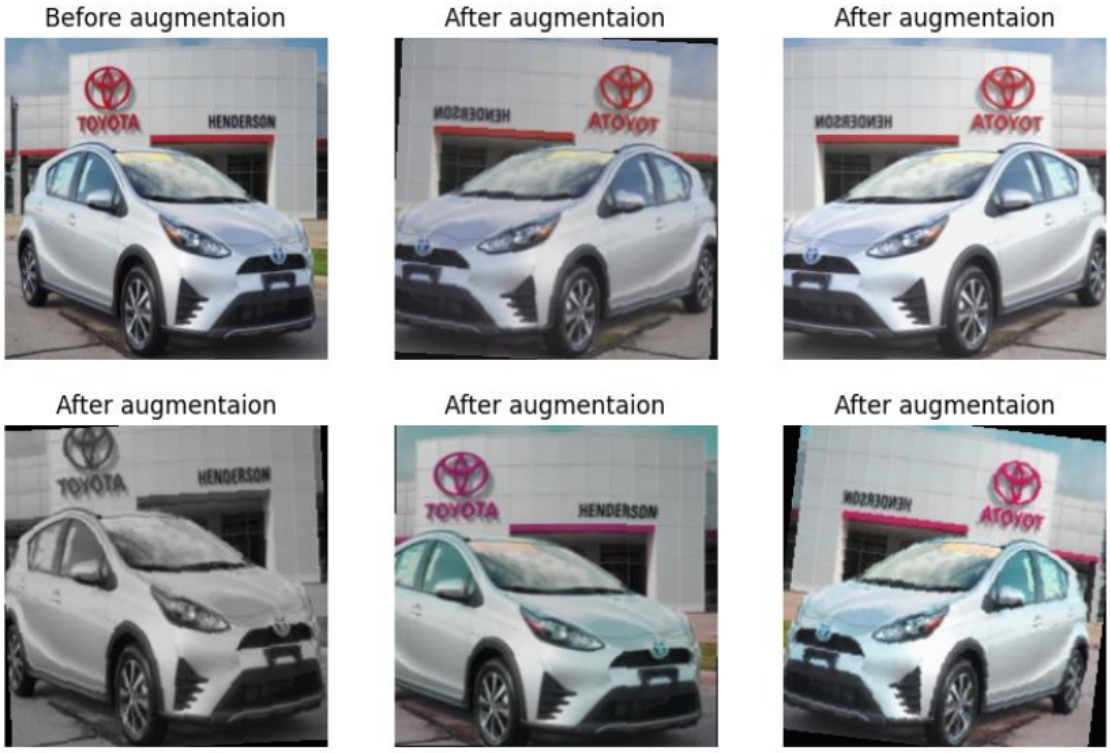


Figure 9: Augmented images for one sample image.

From the image above, it is clear that with the creation of changes in color, reflection, and angle, new images are generated. Thus, from each of the 10 car models, according to the chart below, we have 2000 training images.

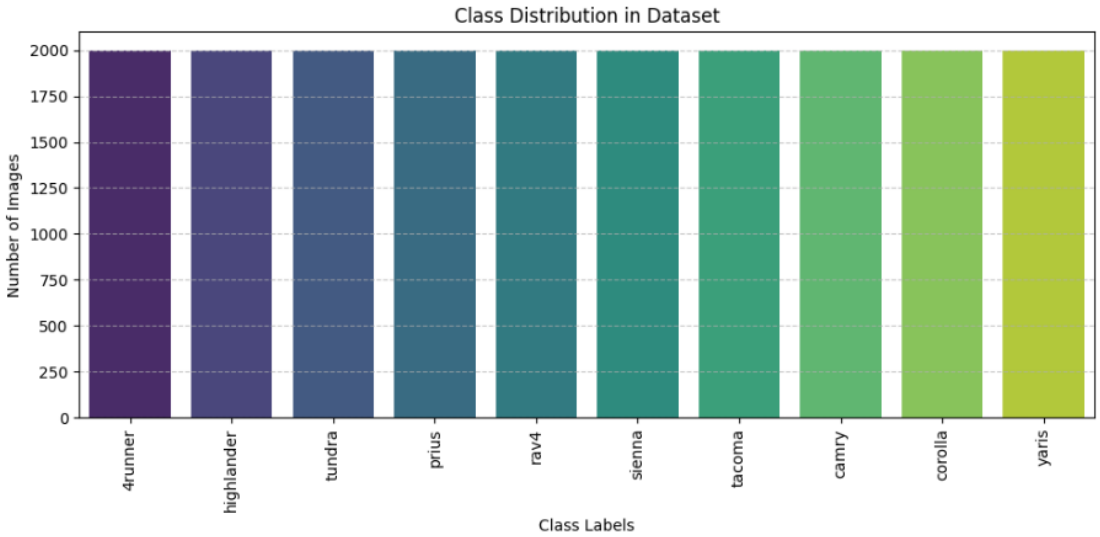
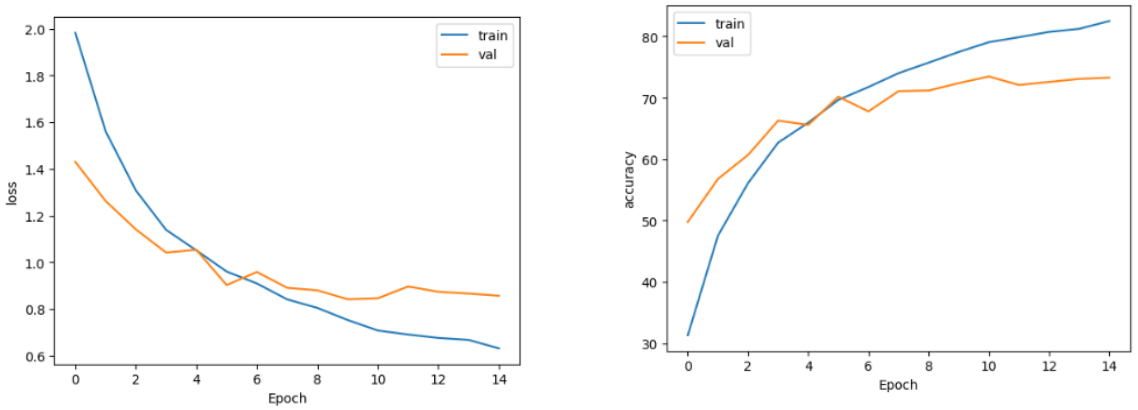


Figure 10: Frequency distribution of training images of each class after augmentation.

Finally, after obtaining the features with the help of the frozen models as before, we are ready to train the models.

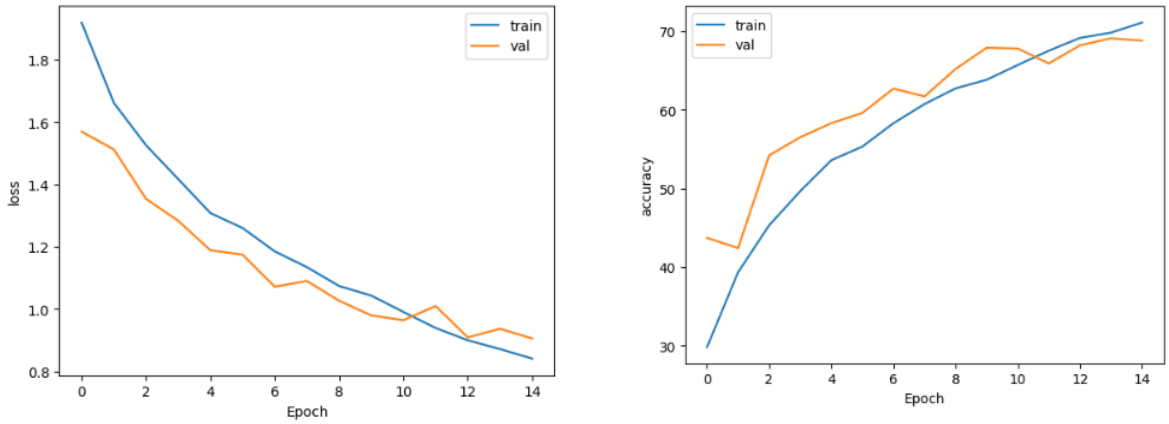
1.6 Training and Evaluation with Augmented Images

Thus, we train the models with 20,000 augmented images.



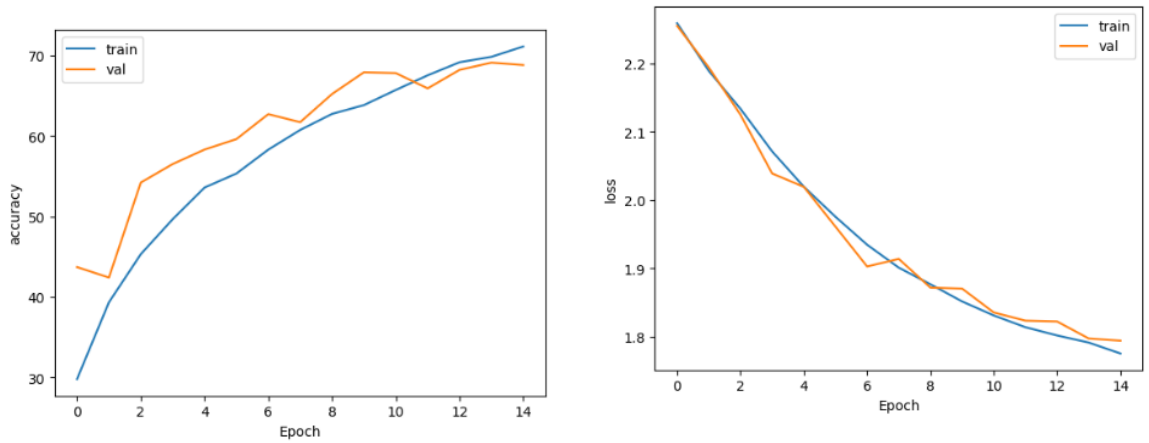
(a) VGG16 Loss **(b)** VGG16 Accuracy

Figure 11: VGG16 training history with augmented images.



(a) AlexNet Loss **(b)** AlexNet Accuracy

Figure 12: AlexNet training history with augmented images.



(a) CNN Loss **(b)** CNN Accuracy

Figure 13: CNN training history with augmented images.

Thus, the weighted performance of the models on the ten classes is shown in the chart below.

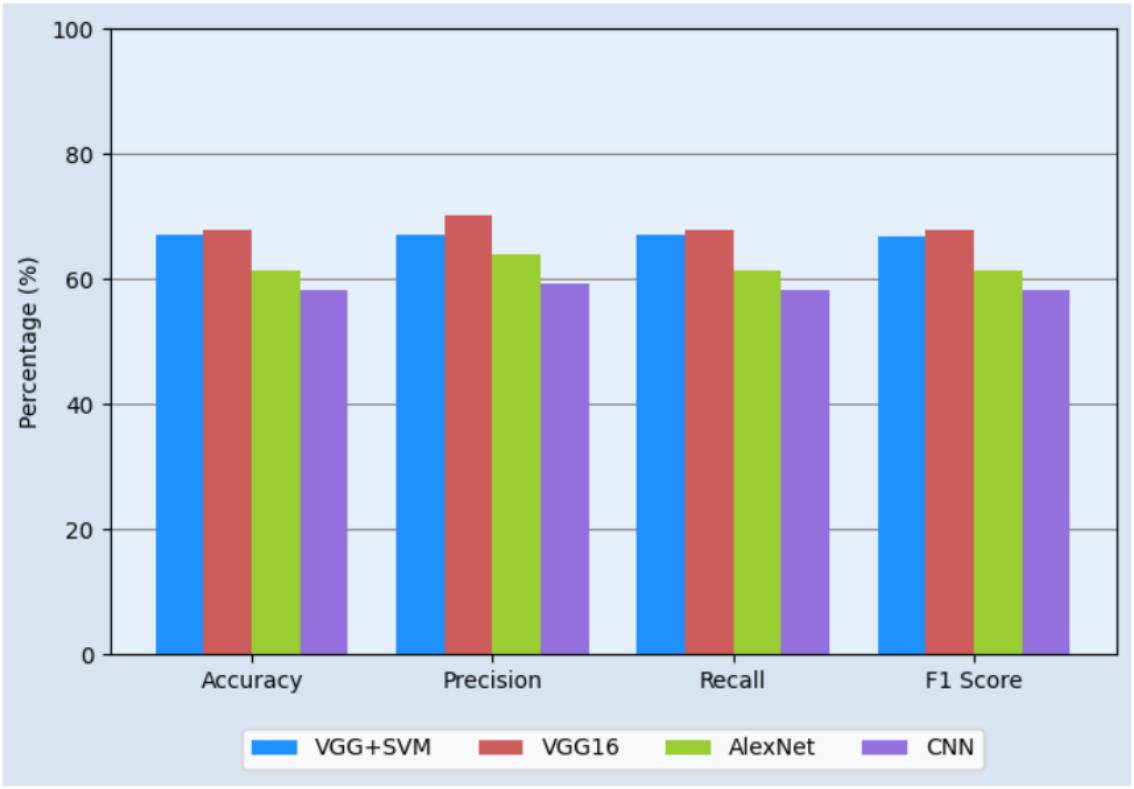


Figure 14: Comparison of model performance with different metrics with augmented images.

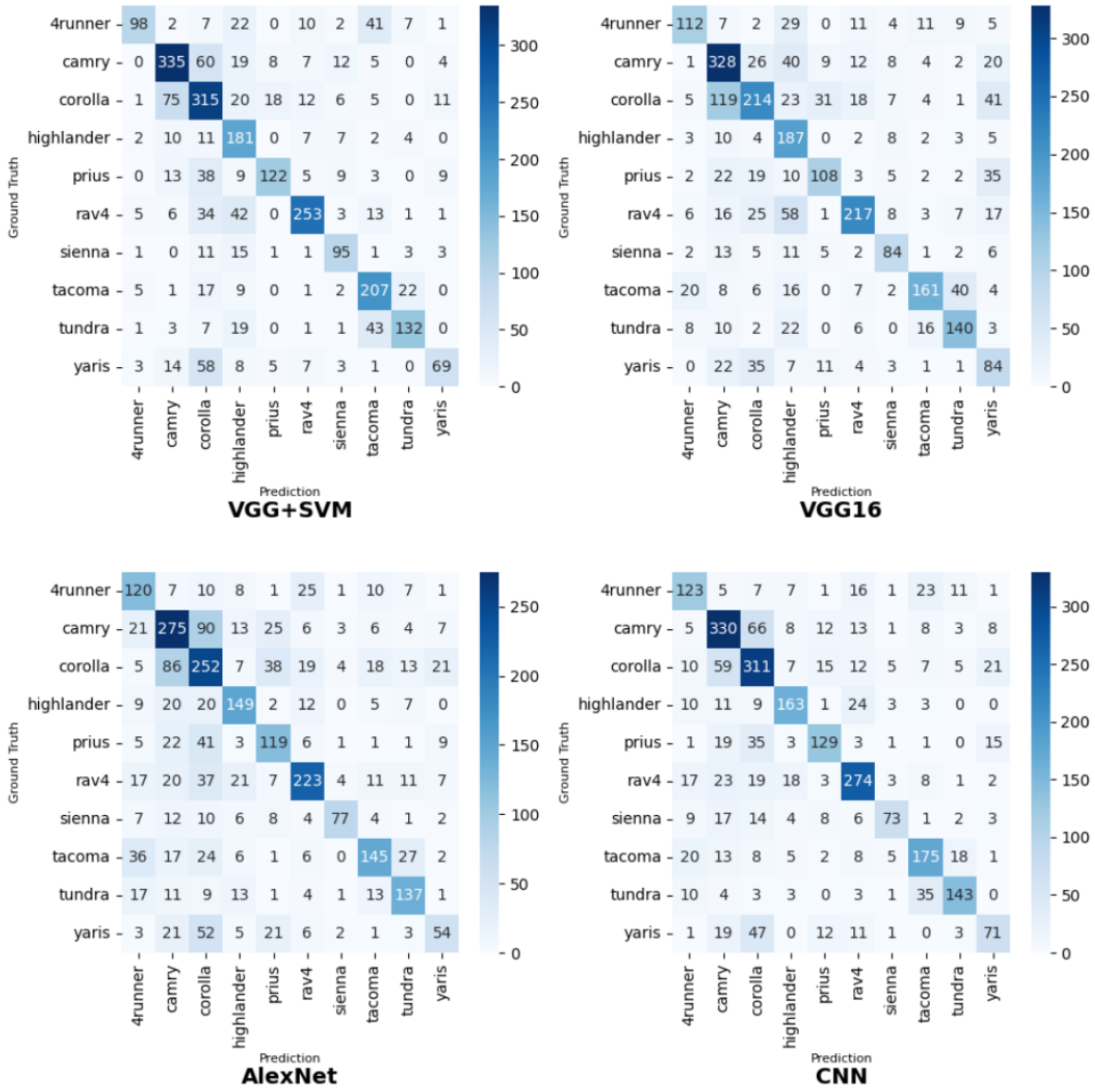


Figure 15: Confusion matrix for each of the four models with augmented images.

The values of the metrics for each model are shown in the table below.

Table 3: Values of the metrics for each model with augmented images.

	Accuracy	**Precision**	**Recall**	**F1 Score**
VGG+SVM	67.03	67.19	67.03	66.95
VGG16	67.86	70.22	67.86	67.85
AlexNet	61.40	64.04	61.40	61.53
CNN	58.24	59.42	58.24	58.33

Although by creating augmented images, we balanced the classes and tried to increase the generalization ability of the models by creating various images, and we were successful in increasing the performance of the CNN, so that its accuracy increased from 42 percent to 58 percent, it is still far from the paper's performance for the VGG+SVM model, with a difference of more than 30 percent.

1.7 Training and Evaluation with Normalized Augmented Images

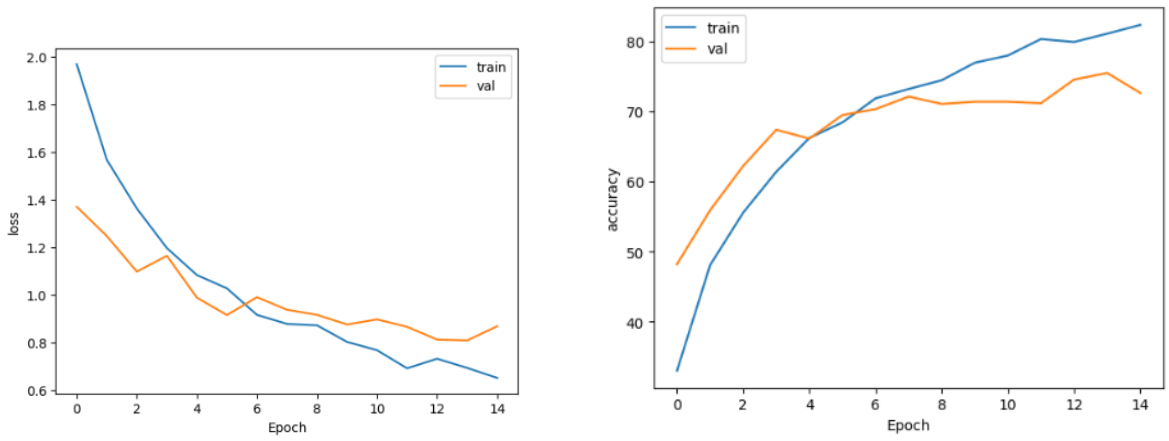
Both AlexNet and VGG16 models have been trained on tens of thousands of images from the ImageNet dataset. One of the common methods that acts like batch normalization for these models is to normalize the images with the mean and standard deviation of the ImageNet images. Thus, the performance of the model can be improved by giving images similar to the images it was trained on. Thus, we see a sample of the images.

Figure 16: Sample images after normalization.

Also, like before, to balance the classes, we create augmented images, a sample of which is as follows:

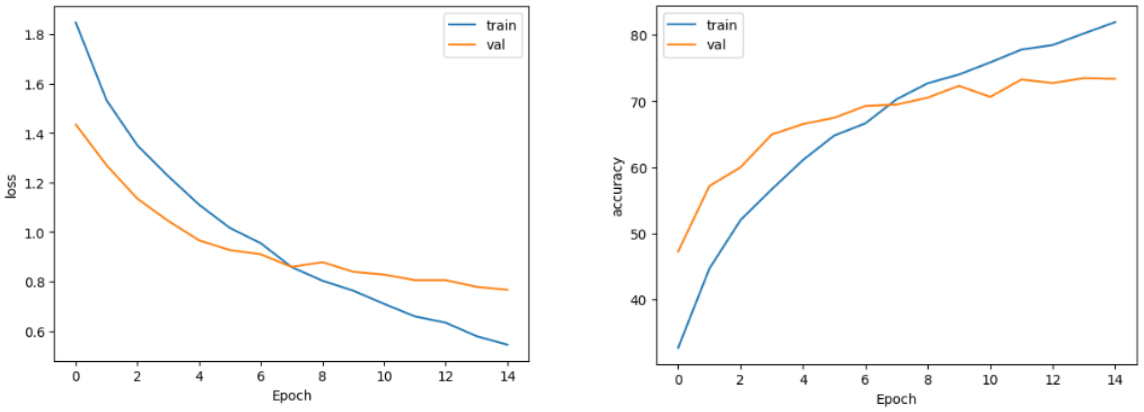
Figure 17: Sample of creating augmented images after normalization.

We can expect that by having a general shape regardless of color and light, the model's predictions will be more generalizable. Now we train the models as before.



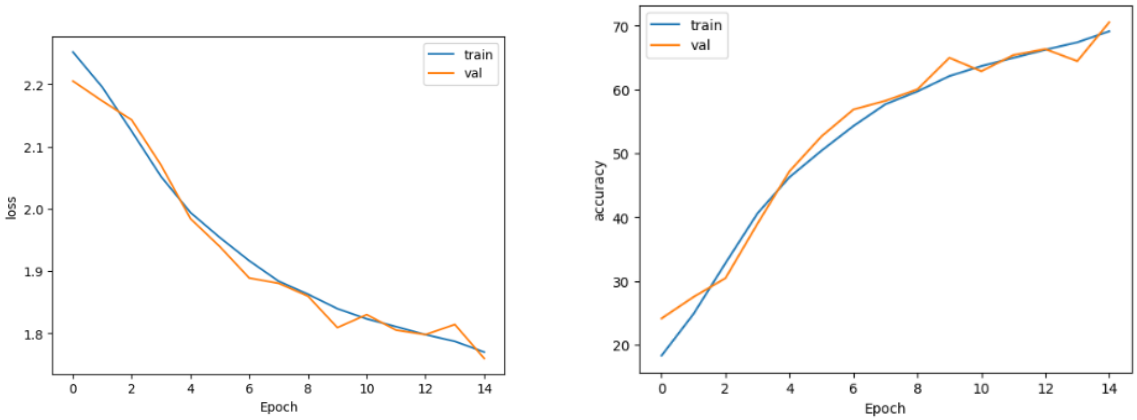
(a) VGG16 Loss **(b)** VGG16 Accuracy

Figure 18: VGG16 training history with normalized augmented images.



(a) AlexNet Loss **(b)** AlexNet Accuracy

Figure 19: AlexNet training history with normalized augmented images.



(a) CNN Loss **(b)** CNN Accuracy

Figure 20: CNN training history with normalized augmented images.

Similar to before, it can be seen that the training of the models is such that overfitting is prevented, and their performance is almost stable. As we saw in the previous sections, all three models reach a performance similar to the paper, and only the performance of the fourth model is different.

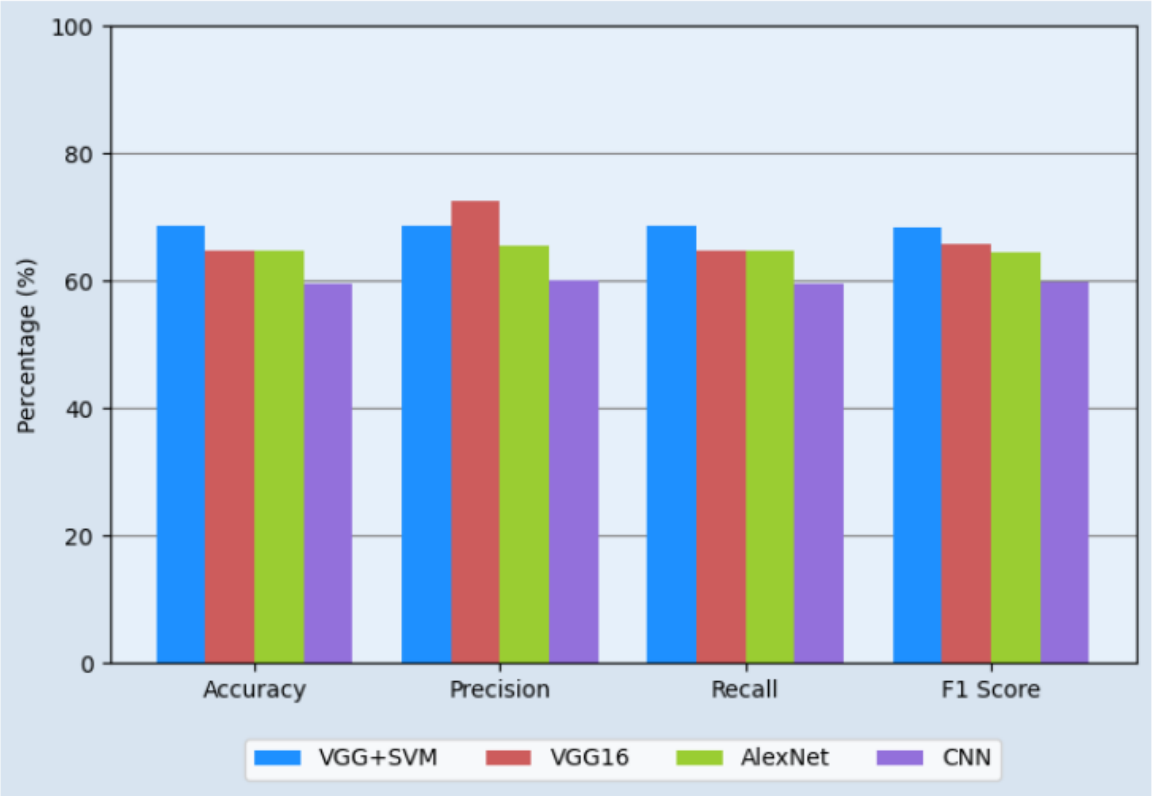


Figure 21: Comparison of model performance with different metrics with normalized augmented images.

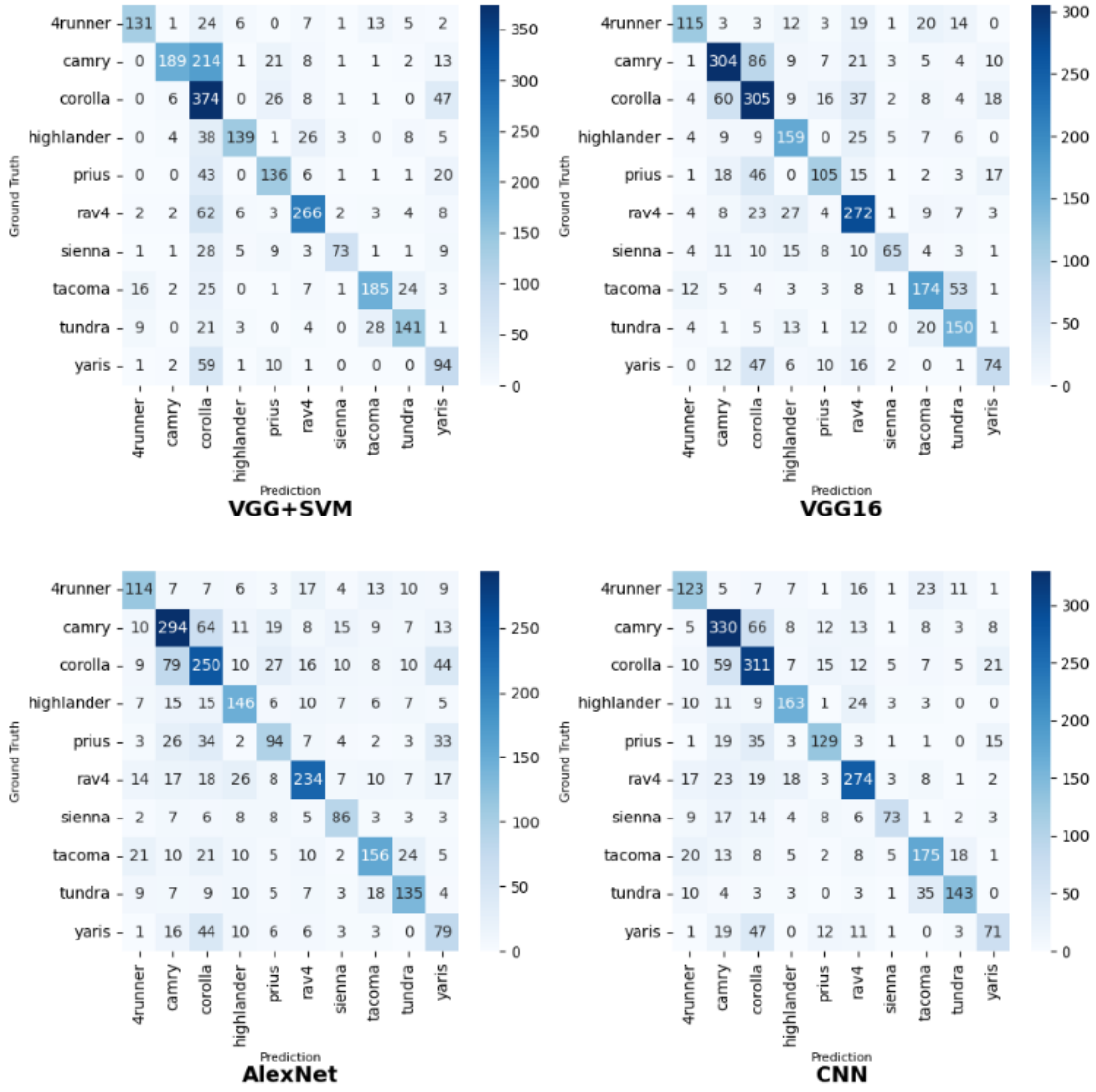


Figure 22: Confusion matrix for each of the four models with normalized augmented images.

Table 4: Values of the metrics for each model with normalized augmented images.

	Accuracy	Precision	Recall	F1 Score
VGG+SVM	68.61	68.67	68.61	68.55
VGG16	64.89	72.66	64.59	65.71
AlexNet	64.70	65.57	64.70	64.51
CNN	59.63	60.22	59.63	59.79

It seems that the CNN model has improved slightly, but the main model still has a weak performance.

1.8 Summary

In this section, we tried to use the method used in the paper to extract features of Toyota car images with the help of the VGG16 model and classify ten models of these cars. In

general, by extracting features with the help of powerful models and training smaller models with these features, it is optimal in terms of training cost and can lead to good performance.

The three models used, VGG16, AlexNet, and CNN, had a performance similar to the paper. The main model of the paper, which is created with the help of SVM, reached a performance of 99 percent in the paper, but for us, the best performance of this model was obtained with augmented images and with the rbf kernel.

We used four models from the paper for different classes of cars, but ultimately, it seemed logical to choose the ten classes with the most images. However, the results with the selection of the other classes were also similar. We also increased the images by creating augmentation and balanced the number of images of the classes. We also used the method of normalizing images with the ImageNet dataset. Also, with different parameters of the SVM model, we trained it several times, but still, we did not reach a performance similar to the paper.

In our opinion, there can be various reasons for this. One is that the dataset used in the paper and the classes of cars used are different from our dataset. In the dataset we used, many of the images were of low quality and had incorrect labels. Also, the paper provides almost no details of the training or other details. On the other hand, the number of images was very large, and training the SVM model, which is trained with the help of the CPU, was very slow. Also, this model usually performs best for a smaller number of data, and it is possible that training a neural network can lead to much better results.