Machine Learning Engineer Nanodegree

Capstone Project

Athanasios Vaxevanos

April, 2019

Definition

1 Project Overview

Car detection and classification is an important part of the intelligent traffic control and management systems today [1, 2]. Automotive is a competitive environment with each of the car makers to produce a variety of car models. Each of these car models has its own characteristics such as shape, size and color to distinguish from others. There are a number of research papers focused on building deep learning models in order to classify the car models [1, 2, 3]. Different CNN architectures using transfer learning from pre-trained models such as ResNet and CaffeNet have been used to classify the car models with high accuracy [4].

In this project, I created a convolutional neural network using transfer learning in order to classify 196 car models from the Stanford car dataset which is published in the Kaggle's website [5].

2 Problem Statement

The main goal of my project is to create a car model classifier. I will achieve that by implementing a CNN neural network using transfer learning in order to obtain higher accuracy. The algorithm will be able to receive a car image and detect its car model. The main tasks for my project are listed below:

1 Obtain and understand the Stanford Car Dataset.
2 Pre-process the images in the dataset.
3 Train a CNN classifier using VGG16 transfer learning.
4 Calculate the model's accuracy.
5 Visualize some car images with their predictions.

The final model is expected to be useful in order to detect the car model of a car image.

3 Metrics

The accuracy is measured as the percentage of the correct classifications on the test dataset over the total number of the car images on the test dataset. The formula is displayed below:

$$\text{Accuracy} = \frac{Correct\ Classifications\ on\ Test\ Dataset}{Total\ Number\ of\ Test\ Dataset} * 100\%$$

The categorical cross-entropy loss and the accuracy metric available in Keras shall be used in order to measure the probability over the different outputs of my CNN classifier. The accuracy metric is suitable for multi classification problems. It can be seen from the next section on the data visualization that the average number of images per class is around 40. This means that the Stanford dataset is a well balanced dataset with the same average images per class. So, the accuracy metric is suitable in order to measure the average of the performance of my model on the overall classes. The main reason is that the dataset is well balanced and we would like to train the model in order to increase the performance for correct classifications across all the classes.

Analysis

1 Data Exploration

The Stanford car dataset has 16,185 images of 196 car model classes which have been split into training and test datasets. The training dataset has 8,144 images and the test dataset has 8,041 images. The images are sorted out in folders with the respected folder names to be as per their car model classes. The arrangement of the two datasets in separate folders of car classes makes easier to be used with the ImageDataGenerator in Keras. Also, for each image in the dataset, its respective bounding box dimensions are provided. These dimensions will not be used as part of my classifier. It is very important to note that each image represents a car with different backgrounds and viewed by different angles. Some of the images are professional taken images and some others are low quality images taken from the internet or personal shots. The images are colored and they have got various sizes. For these reasons, pre processing of the images and augmentation shall be used in order to improve the classifier's performance. Figure 1 displays some of the images in the datasets.

Figure 1: Sample of Images in the Car Stanford Dataset.

2 Exploratory Visualization

The two figures below display the number of images per car class on both the training and test datasets. It can be seen that the average number of images per class is around 40. This visualization was used in order to prove that there are enough images per class in order to obtain a good accuracy per class. In other words, the images taken are well balanced across the 196 different car model classes.

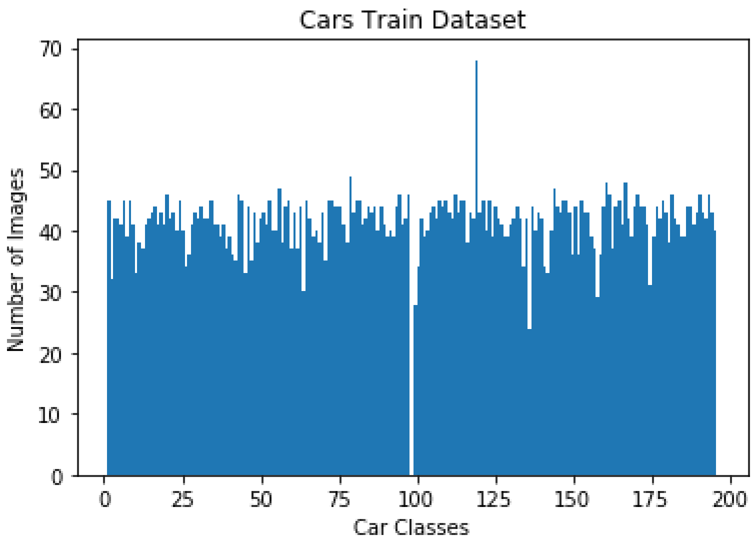

Figure 2: Histogram Plot of Number of Images per car class in the training dataset.

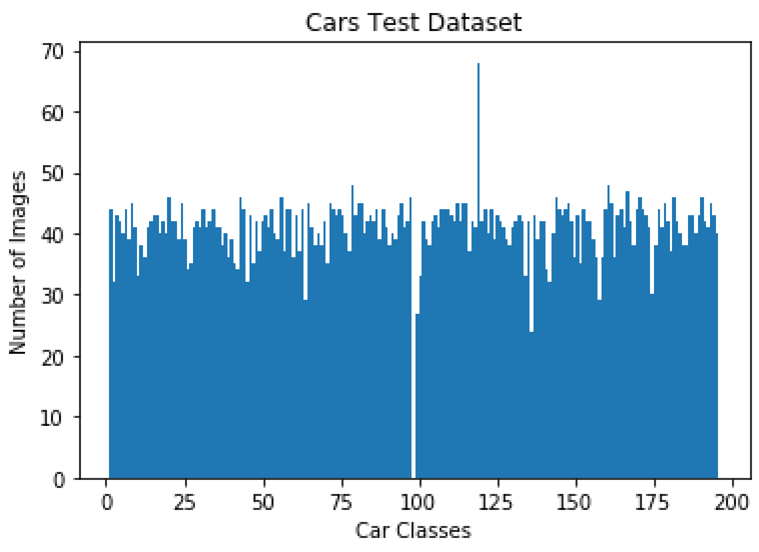

Figure 3: Histogram Plot of Number of Images per car class in the test dataset.

From the above visualization and the analysis in the previous section, it can be concluded that the Stanford car dataset is appropriate for my project as it provides a number of car images associated with their targets (car models).

3 Algorithms and Techniques

For this car model classification problem, I used the transfer learning technique by utilizing the pre-trained VGG16 architecture which is trained on the ImageNet dataset. I utilized this technique because there are a lot of similarities between the ImageNet dataset and the Stanford car dataset. So, I wanted to exploit the already extracted features from the pre-trained VGG16 CNN layers. The input to the model will be colored images which are represented as 3D arrays or vectors by the computers. CNN is the state of art layers which filter each of these arrays in order to extract their features. The pooling layers such as max pooling layers or global average pooling layers are used in order to reduce the dimensionality and the hyper-parameters of the outputs of each CNN. These are mainly used in order for our model to avoid overfitting.

In my model, the inputs which are colored images are converted to 3D matrices or vectors which then are fed into the pre-trained CNN layers and pooling layers of the pre-trained VGG16 model. The output of this will be the extracted features corresponded to my dataset as the last CNN and max pooling layer are trainable. Then the global average pooling layer is used to drastically reduce the dimensionality of the output of the CNNs in order to be fed into a fully connected layer of 196 nodes. The size of the nodes corresponds to the size of the classes for my classification problem.

Finally, CNN layers usually use the Relu activation function where the last fully connected layer of 196 will use the softmax activation method. So, this output will extract the probability for each node. So, the node with the maximum probability will be classed as the prediction.

The parameters that needs to be tuned in order to obtain the best result was the optimizer with the learning rate, the batch size of images and the number of epochs for the training of the model. In order to train my model, I used the Kaggle's kernel with its GPU on and it took me quite a lot of processing time.

4 Benchmark

I identified a benchmark model utilizing the InceptionV3 pre-trained model on the classification problem on the Stanford dataset. This model is published on the Kaggle website and it is used as the benchmark for my approach [6]. This has achieved an almost 50% validation accuracy during training.

Methodology

1 Data Preprocessing

The ImageDataGenerator function which is available in Keras was used for data pre-processing and data augmentation. The following are applied through the Keras function:

1. Image pixels are rescaled in the range of [0, 1].
2. Images converted to the same size of pixels 240x240.
3. The training dataset was spilt by 80% in training and 20% in validation datasets.

Image Augmentation involved for images:

1. Zoomed by a factor of 0.2.
2. Image height shift by a factor of 0.2
3. Image width shift by a factor of 0.2
4. Rotated by a factor of 0.2
5. Horizontally and vertically flipped.

The rescale of all image pixels and sizes was implemented in order to treat all the images in the same manner and contribute evenly in the validation and training loss despite their high or low resolution. The main reason to augment my dataset was to obtain a further better accuracy for my model.

2 Implementation

The implementation phase can be determined by the following stages:

1. **Preprocess Stage:** During this stage, I rescaled, resized, augmented and split into training and validation datasets the images.
2. **Initiate VGG16 architecture:** During this stage, the VGG16 model loaded with the pre-trained weights from ImageNet and it was initiated. The last 3 fully connected layers were removed and these were replaced by a 196 nodes fully connected layer.
3. **Freeze and Train Layers:** During this stage, all the VGG16 model convolutional layers were freezed expect from the last convolutional layer and the new fully connected layer of 196 nodes. I trained the last convolutional layer of the VGG16 model in order for the already pre-trained convolutional layer to extract features which were closer to the Stanford cars dataset.
4. **Train Model:** During this stage, I trained the model utilizing the "rmsprpo" optimizer and the categorical_crossentropy metric. The model was trained with a learning rate of 0.001 for 100 epochs. I tried to keep the learning rate as high as possible for the last convolutional network to learn faster and not to lose a lot of the already ImageNet feature extracts.
5. **Freeze Last CNN and re-train:** During this stage, I loaded the extracted weights from the previous training stage and I freezed the last convolutional network on my model. At that point, the only trainable layer is the last fully connected layer which was further trained for 50 epochs with learning rate of 0.00005 and the same optimizer and metric.
6. **Evaluate the Accuracy:** During this stage, I obtained the accuracy of my model on the test dataset.

<u>Note</u>: In the above process the GlobalAveragePooling2D() and MaxPooling2D() functions were set to trainable during the various stages.

3 Refinement

Initially, I didn't augment my datasets and I only tried to train the last fully connected layer of 196 nodes. However, the best accuracy that I achieved by this implementation and technique was 31% with a learning rate of 0.001 being trained for 200 epochs. The main problem of this implementation was that the model was over fitting significantly. In order to avoid this over fitting problem, I augmented the datasets and I tried to re-evaluate the performance of my model. The model with augmentation achieved a better performance of almost 36%. After that point, my idea was to start training some of the already pre-trained convolutional networks of the VGG16 model. The main thought behind it was that the training of the last CNN will adjust the weights to extract features which are closer to the Stanford cars dataset. By training the last CNN and the last fully connected layer, the performance of the model was improved significantly and it was almost 45%. A further training of the last fully connected layer with the rest of the layers being freezed achieved above 50% accuracy. Different learning rates, batch sizes and optimizers were tried throughout the above implementations to achieve the best possible performance.

Results

1 Model Evaluation and Validation

During the evaluation and validation process, I used and modified quite a lot of the hyper parameters of the model in order to achieve better performance. The model with its hyper parameters which achieved the best performance is listed below:

1. Batch size = 80.
2. Optimizer = rmsprpo.
3. Last Convolutional and fully connected layers were trained with lr=0.001 for 100 epochs.
4. Re-trained last fully connected layer was trained with lr=0.00005 for additional 50 epochs.

The results of this model can be trusted as I tried to train the same model with different batch sizes, optimizers and more epochs but the model was not achieving a better accuracy than the 52%. Also, I tried to make more convolutional layers trainable but the accuracy was not improving. Figures 4 and 5 represent the training and validation accuracies and losses during training of the model over the first 50 epochs. It can be seen that the training and validation loss is decreasing over the epochs. This is a sign that our model is learning over the epochs and the learning is stable. On the other hand, it can be seen that the accuracy over the epochs are increasing on the training and validation datasets. I have suggested some improvements on the relevant section in order to increase the performance of the model.

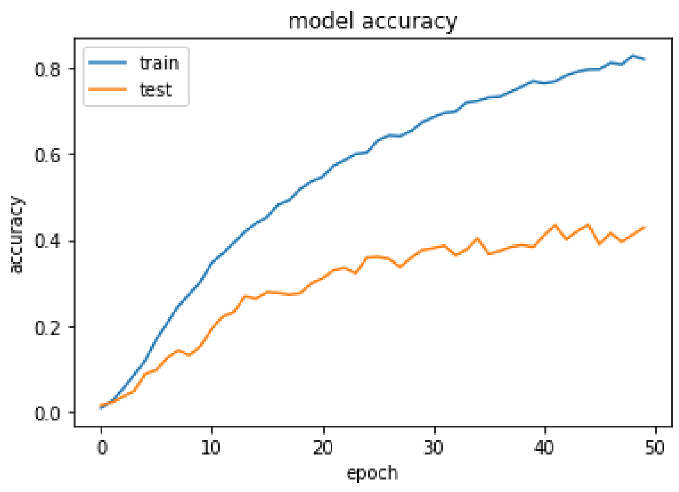

Figure 4: Training and Validation Accuracy over epochs.

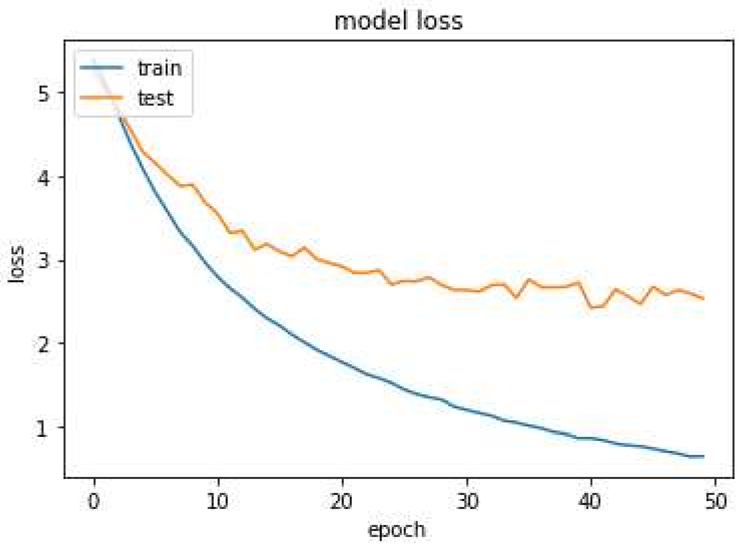

Figure 5: Training and Validation loss over epochs.

2 Justification

The accuracy of my model is slightly above the 50% which replicates the accuracy of the benchmark model. I believe that the accuracy of my model is quite good considering the fact that the images provided have different backgrounds and they have been captured by different angles. Also, I consider the fact that I have got a large number of car models to classify and the number of images available per class is almost 40. So, the dataset is quite small in relation to the number of classes to classify.

Conclusion

1 Free Form Visualization

Figure 5 and 6 represent one correct and one incorrect prediction from our CNN model respectively. It can be seen that for the correct prediction (Fiat 500), the background of the image is not complex as well as the car represents the biggest part of the image from a close camera shot. For this reason, the model extracts easier more features relevant to the Fiat 500 brand and that is one of the reasons for the correct classification. On the other hand, the image of the Ferrari has been taken from a distance as well as the background is complex. So, the model does not have the opportunity to extract more features relevant to the Ferrari in order to make the correct prediction. So, the quality of the images such as the angle of the car, the distance of the camera shot etc. affect the result for my classification.

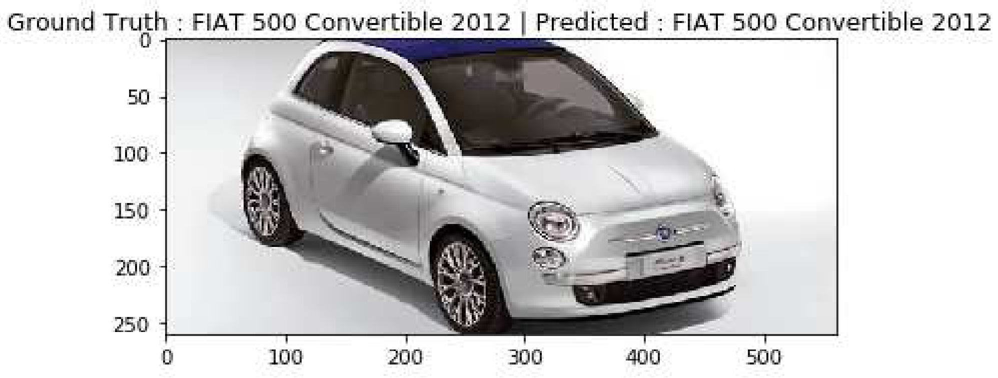
Ground Truth : FIAT 500 Convertible 2012 | Predicted : FIAT 500 Convertible 2012
0
50
100
150
200
250
0
100
200
300
400
500

Figure 6: Fiat 500 correct car brand prediction.

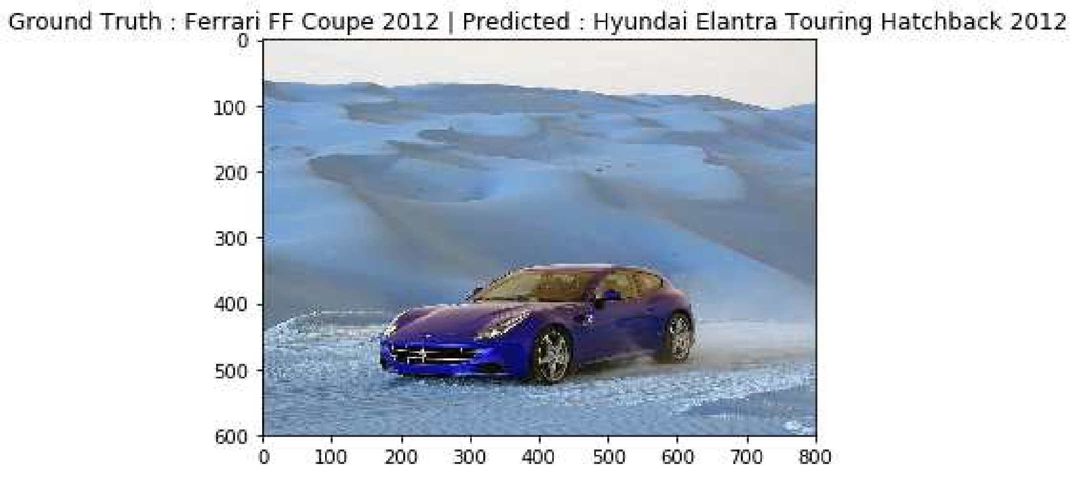
Ground Truth : Ferrari FF Coupe 2012 | Predicted : Hyundai Elantra Touring Hatchback 2012
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800

Figure 7: Ferraru FF Coupe incorrect car brand prediction.

2 Reflection

The steps that I followed to complete this project are outlined below:

1. I researched online to identify a problem which was relevant to automotive sector as I have been working as a systems engineer for JLR.
2. I identified the relevant datasets which were collected by the Kaggle website.
3. Relevant research papers and projects as benchmarks were found.
4. The classifier was trained on the dataset using augmentation and transfer learning.
5. The test accuracy of the classifier was investigated and further improved.
6. Results of predictions from various car brands were displayed.

I found steps 4 and 5 very challenging as I was trying to improve the performance of my classifier. The most challenging part was to tune the hyper-parameters of my model in order to improve the test accuracy. I am very proud of achieving a similar performance with the benchmark model as I have noted above the car images are taken with different backgrounds as well as from different angles. These are some of the biggest challenges for the classifier to improve further its performance. Also, I have learned quite a lot for training and improving the performance of the deep learning networks and this will help me to pursue a career in autonomous driving sector.

3 Improvements

Further improvements that can be made is to obtain a better dataset from car images which will have less complex backgrounds as well as the car shots will be taken by the same distance and angles. I believe this will help my classifier to generalize better and improve further its performance. Another improvement that I could suggest is to try other pre-trained models on the ImageNet dataset in order to investigate if these achieve a better performance than VGG16. Finally, I trained my model using Kaggle Kernel and it was a bit slow in order to train the model for 100 or more epochs. So, it will be better to train the model using faster kernels in order to further train the model for more epochs.

References

[1] Liu, Derrick, and Yushi Wang, “*Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and TransferLearning*”.

[2] Stefan Badura and Stanislav Foltan, “*Advanced scale-space, invariant, low detailed feature recognition from images – car brand recognition*”, Proceeding of the International Multiconference on Computer Science and Information Technology, pp. 19-23, 2010.

[3] Syed Hasib, Akhter Faruqui, Rajitha Meka, “*Vehicle Make and Model Classification Using Convolutional Neural Networks*”, Department of Mechanical Engineering, San Antonio, Texas.

[4] Pranav Dar, “*Top 10 Pretrained Models to get you started with Deep Learning (Part 1 –Computer Vision)*”, [accessed: March, 2019], https://www.analyticsvidhya.com/blog/2018/07/top-10-pretrained-models-get-started-deep-learning-part-1-computer-vision.

[5] Kaggle Website for Stanford Car Dataset, https://www.kaggle.com/jutrera/stanford-car-dataset-by-classes-folder.

[6] Keras InceptionV3 transfer learning, https://www.kaggle.com/kushal1996/keras-inceptionv3-transfer-learning.