Machine Learning Engineer Nanodegree

Capstone Project

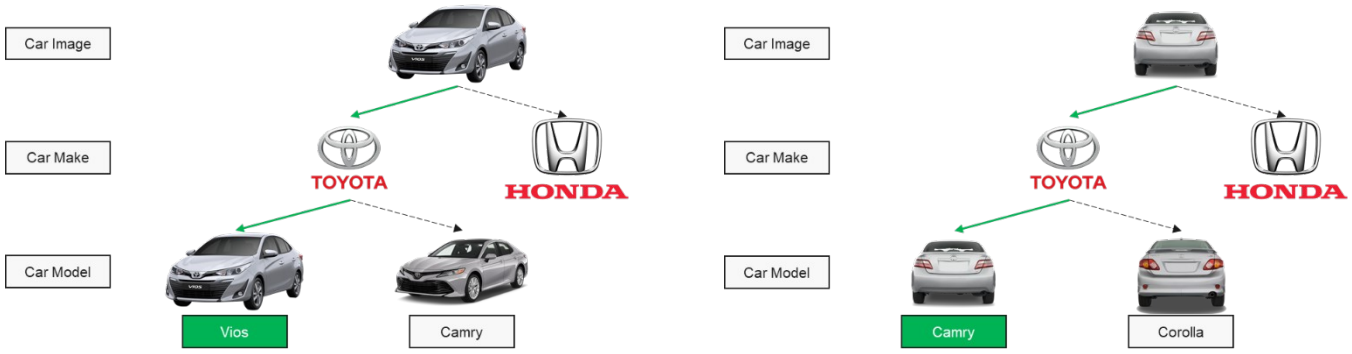
Car Make and Model Image Recognition

Chu Nguyen Van
August 13, 2019

I. Definition

Project Overview

Fine-grained image recognition is the task of distinguishing highly similar objects such as identifying canine breed, bird species, or aircraft model. Car Make and Model Image Recognition is a sub problem belonging to a large family of Fine-grained visual classification problems. This problem is challenging as the differences between cars could be extremely subtle and highly dependent on factors including angle of view and weather to name a few. The figures below show the process of identifying car make and model from an image input.



An algorithm which can effectively distinguish one car make and model from another is extremely beneficial. Some of the benefits include improving Traffic Video Surveillance system or increasing the accuracy of a variety of Traffic Analytics.

On a personal note, I have always wanted to be able to come up with a model to accurately identify the car on the street. The reason is because I have been using lots of ride-sharing services (e.g. Uber, Grab, etc.), and it has always been a struggle for me to identify the correct car in a crowded street. Even though relevant information (e.g. the vehicle number plate, the make, the model, the color of the car, etc.) is visible in the mobile application, I think it is still difficult for a person to correctly spot out the car especially in difficult environment (e.g. bad weather, congestion, etc.). Therefore, it is the motivation for me to work on this project.

Some challenges associated with vehicle recognition problems are shown below.

- Car image acquisition.
- Variations in lighting conditions when the car photo was taken.
- Large variety of car makes and models.

- Similarities between different car makes and models.

Problem Statement

The goal is to train an image classifier model that can correctly label the car make and model based on an input image of a car. This model should have at least 70% classification accuracy. The tasks involved are the following:

- Download the cars dataset which originated from Stanford University AI Lab. Data source and banner image: http://ai.stanford.edu/~jkrause/cars/car_dataset.html contains all bounding boxes and labels for both training and tests.
- Perform exploratory analysis and some visualizations on the dataset to understand how the data is distributed.
- Crop the car out of the original image based on the provided bounding box information. This will eliminate all noises in the image so that the model can have better accuracy thanks to cleaner input. Augment and transform images into 4D tensors which are later used as input for Tensorflow Keras CNN model.
- Train and test benchmarking model against test set.
- Fine-tune Xception and InceptionV3 model.
- Test transfer learning models against test set.
- Validate transfer learning model performance by charting out the learning curve.
- Compare the accuracy of the transfer learning models against the benchmarking model.

Metrics

An accuracy score comparing the model predictions of test images against the true test label will be computed. The higher the accuracy score is, the better the model is. Another reason for choosing this metric is because as explained in a later section, the dataset is highly uniform since the number of cars per class is almost identical.

$$Accuracy = \frac{Number\ of\ Correct\ Classifications\ of\ Car\ Make \wedge Model}{Total\ number\ of\ test\ labels}$$

II. Analysis

Data Exploration

As mentioned above, the Cars dataset is originated from Stanford University AI Lab. The dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images. The number of cars for each class is roughly the same in the training and test set. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

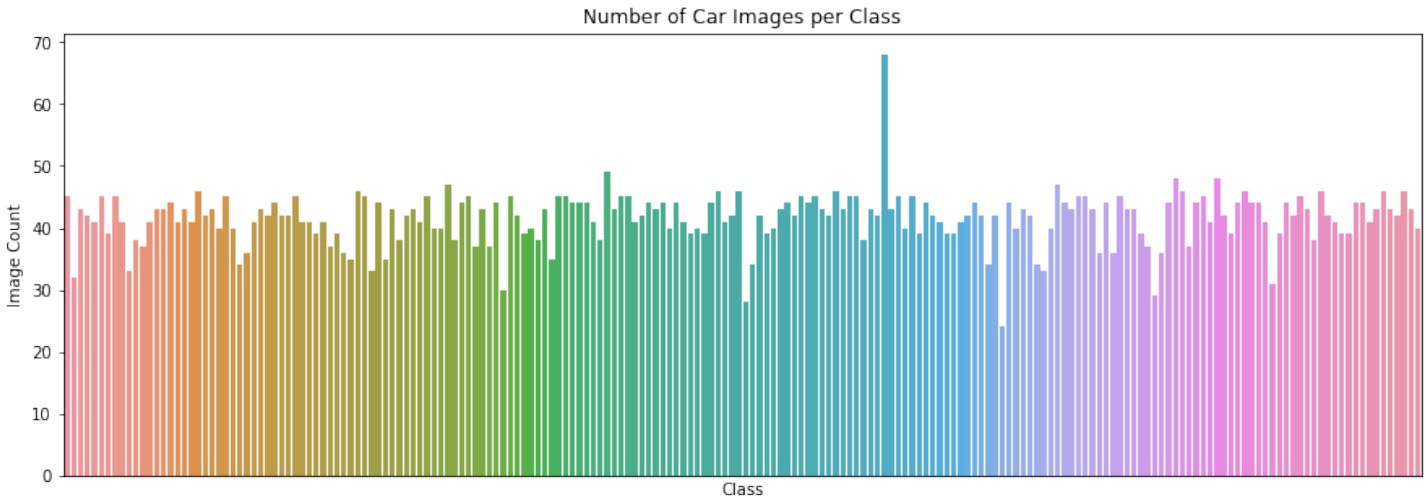
Sample images from the dataset are shown below.

The following files are provided together with the car images. Descriptions of the files are as follows:

- *cars_meta.mat* contains 196 different class names.
- *cars_train_annos.mat* contains bounding box information and class name corresponding to each training image. There are 8,144 training images in total.
    - *bbox_x1*: Min x-value of the bounding box, in pixels.
    - *bbox_y1*: Min y-value of the bounding box, in pixels.
    - *bbox_x2*: Max x-value of the bounding box, in pixels.
    - *bbox_y2*: Max y-value of the bounding box, in pixels.
    - *class_id*: Integral id of the class the image belongs to.
    - *fname*: Filename of the image within the folder of images.
- *cars_test_annos_withlabels.mat* contains bounding box information and class name corresponding to each testing image. There are 8,041 test images in total.

Exploratory Visualization

The bar plot below shows the number of cars per class in the dataset. As indicated by the chart, there is a variation in the number of images per car class. Therefore, I will utilize image augmentation later to introduce more variety to the dataset.



Algorithms and Techniques

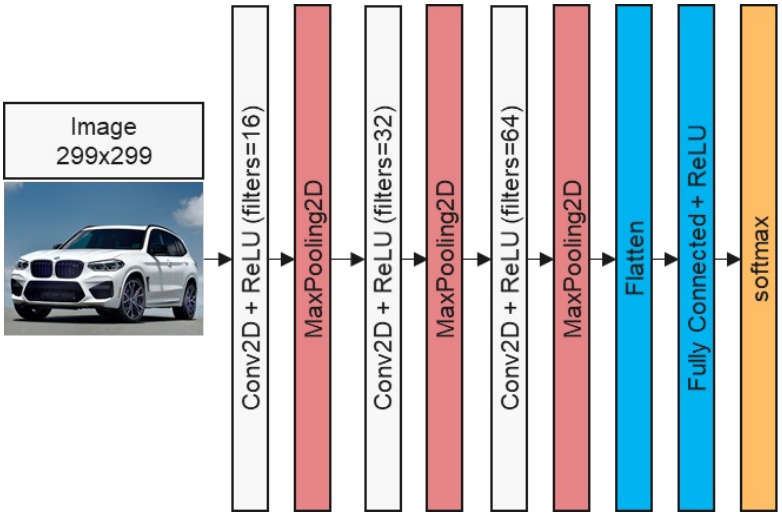
Convolutional Neural Network will be used as the classifier as this state-of-the-art algorithm has proven itself countless times when it comes to various image recognition tasks. Inspired by the results of the Deep Learning research for Cancer Detection by Sebastian Thrun, I intend to use Transfer Learning to solve this classification problem.

To be specific, I will use the Xception and InceptionV3 model initialized with *imagenet* weights. Image augmentation will be performed to add more variety to the input images. Since the cars dataset is much smaller than the ImageNet dataset, retraining the whole model will probably result in overfitting. Thus, it is my intention to keep most of the pre-loaded weights intact so that I can leverage the top model to extract higher level features of the image. The last few layers of the top model will be retrained and subsequently connected with my own logistic layers specific to this problem.

Benchmark Model

A random guess with equal probability assigned for each car class has a ~0.5% (= 1/196) probability of being accurate.

I will train a CNN model from scratch to serve as a benchmark for my transfer learning model. The architecture of the benchmarking model is shown below.



III. Methodology

Data Preprocessing

The data preprocessing process includes the following steps:

- Create 3 pandas dataframe (*cars_info*, *train_info*, and *test_info*) respectively for each of the abovementioned original files that come together with the images (*cars_meta.mat*, *cars_train_annos.mat*, and *cars_test_annos_withlabels.mat)*.
- In order to match the *class_id* values in the *train_info* and *test_info* dataframe with the index column of *cars_info* dataframe, subtract 1 from each *class_id* value since Python index starts from 0.
- Append the relevant paths to training and test images to the *fname* column in *train_info* and *test_info* dataframe. To be specific:
    - Original *fname* value: 00001.jpg
    - *fname* with path: /media/Data/capstone/cars_train/00001.jpg
- Append a *class_names* column with text description of car make and model to the right of *train_info* and *test_info* dataframe.
- After the transformation, the *train_info* and *test_info* dataframe look like the table below.

	bbox_x1	bbox_y1	bbox_x2	bbox_y2	class_id	fname	class_names
0	39	116	569	375	13	/media/Data/capstone/cars_train/00001.jpg	Audi TTS Coupe 2012
1	36	116	868	587	2	/media/Data/capstone/cars_train/00002.jpg	Acura TL Sedan 2012
2	85	109	601	381	90	/media/Data/capstone/cars_train/00003.jpg	Dodge Dakota Club Cab 2007
3	621	393	1484	1096	133	/media/Data/capstone/cars_train/00004.jpg	Hyundai Sonata Hybrid Sedan 2012
4	14	36	133	99	105	/media/Data/capstone/cars_train/00005.jpg	Ford F-450 Super Duty Crew Cab 2012

- In order to improve accuracy of the model, crop the car based on bounding box information for all training and test images.

Original image with bounding box

Cropped car image within the bounding box

- The original training set is divided into a training (70%) and validation set (30%).
- All images get resized to 299 x 299 so that they can be fed into the Xception and InceptionV3 transfer learning model later.
- The pixel values get rescaled from range (0, 255) to range (0, 1).
- Perform one-hot encoding on the car labels.

- Transform datasets into 4D tensors so that they can be used as input for Tensorflow Keras CNN.
- Perform image augmentation with the following adjustable parameters:
    - *shear_range*=0.2. Shear intensity.
    - *rotation_range*=40. Randomly rotate images by a degree.
    - *width_shift_range*=0.2. Randomly shift images horizontally.
    - *height_shift_range*=0.2. Randomly shift images vertically.
    - *zoom_range*=0.2. Assign a range for random zoom.
    - *horizontal_flip*=True. Randomly flip images horizontally.
    - *fill_mode*="nearest". Points outside the boundaries of the input are filled according to the nearest point.

Implementation

The implementation process includes the following steps.

- Load both the training and validation images into RAM and preprocess them as described previously.
- Due to limited RAM – my machine only has 16GB of RAM – the loading and preprocessing of the test images will be done later once the model weights have been obtained.
- There are 3 separate sections in the Jupyter Notebook file for 3 different models:
    - Benchmarking model.
    - Xception fine-tuned model.
    - InceptionV3 fine-tuned model.
- Benchmarking model:
    - Define the network architecture and training parameters for benchmarking model. The benchmarking model summary can be found below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 299, 299, 16)      208
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 16)      0
_________________________________________________________________
dropout (Dropout)            (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 149, 149, 32)      2080
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 74, 74, 64)        8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 37, 37, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 37, 37, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 87616)             0
_________________________________________________________________
dense (Dense)                (None, 512)               44859904
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 196)               100548
=================================================================
Total params: 44,970,996
Trainable params: 44,970,996
Non-trainable params: 0
_________________________________________________________________
```

    - Compile the model with *categorical_crossentropy* as the loss function, *rmsprop* as the optimizer, and *accuracy* as the metrics.
    - Train the network and obtain the best weights for benchmarking model.

- Obtain accuracy score by testing the benchmarking model against the test images.

- Xception fine-tuned model:
  - Load base Xception model with pre-loaded *imagenet* weights.
  - Freeze the first 94 layers.
  - Make the rest of the layers trainable.
  - Append a Batch Normalization layer, a Global Average Pooling 2D layer, 2 fully connected layers with kernel regularization and a Dense layer with *softmax* activation function to the top model.
  - The Xception fine-tuned model network architecture can be found below.

Layer (type)	Output Shape	Param #
xception (Model)	(None, 10, 10, 2048)	20861480
batch_normalization_v1_4 (Ba	(None, 10, 10, 2048)	8192
global_average_pooling2d (Gl	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 196)	50372

Total params: 22,100,460
Trainable params: 11,252,676
Non-trainable params: 10,847,784

  - Train the final model with Model Checkpoint, Early Stopping if there is no improvement in “val_acc” and Reduce Learning Rate if there is no improvement in “val_loss”.
  - Test fine-tuned Xception model against Test set.
  - Validate fine-tuned Xception model performance by charting out the learning curve.
  - If the result is not as good as expected (more than 70%), redefine the transfer learning network architecture.

- InceptionV3 fine-tuned Model.
  - Load base InceptionV3 model with pre-loaded *imagenet* weights.
  - Freeze the first 249 layers.
  - Make the rest of the layers trainable.
  - Append a Batch Normalization layer, a Global Average Pooling 2D layer, 2 fully connected layers and a Dense layer with *softmax* activation function to the top model.
  - The InceptionV3 fine-tuned model network architecture can be found below.

Layer (type)	Output Shape	Param #
inception_v3 (Model)	(None, 8, 8, 2048)	21802784
batch_normalization_v1_94 (B	(None, 8, 8, 2048)	8192
global_average_pooling2d (Gl	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 196)	100548

Total params: 23,223,268
Trainable params: 11,957,828
Non-trainable params: 11,265,440

    - Train the final model with Model Checkpoint, Early Stopping if there is no improvement in “val_acc” and Reduce Learning Rate if there is no improvement in “val_loss”.
    - Test fine-tuned InceptionV3 model against Test set.
    - Validate fine-tuned InceptionV3 model performance by charting out the learning curve.
    - If the result is not as good as expected (more than 70%), redefine the transfer learning network architecture.
- Compare the accuracy of the transfer learning fine-tuned models against the benchmarking model.

Refinement

The accuracy of the Benchmarking model when tested against the test set is only 1.93%. This is expected as the network was only compiled in 5 epochs. The weights were learnt from scratch so running the model in a small number of epochs does not allow the model to find the optimal weights. In addition, 3 Convolutional Layers appear to be insufficient to capture subtle differences between various car makes and models.

In order to achieve higher accuracy, I have made the following changes when fine-tuning Xception and InceptionV3 model.

- Increase the number of epochs from 5 to 70.
- Initialize the weights for the network using the pre-trained weights obtained from training *imagenet* dataset.
- Keep the top layers of the Xception and InceptionV3 model intact and retrain the bottom layers of the Xception and InceptionV3 model.

This allows me to obtain the accuracy of 72.47% and 57.70% on the test dataset for the fine-tuned Xception and InceptionV3 model respectively.

IV. Results

Model Evaluation and Validation

The validation set containing 30% of the original training images was used to evaluate the model. The validation accuracy results of the fine-tuned models are much better than that of the benchmarking model. Nonetheless, due to the increase in the number of epochs, the 2 fine-tuned models takes much longer to finish training.

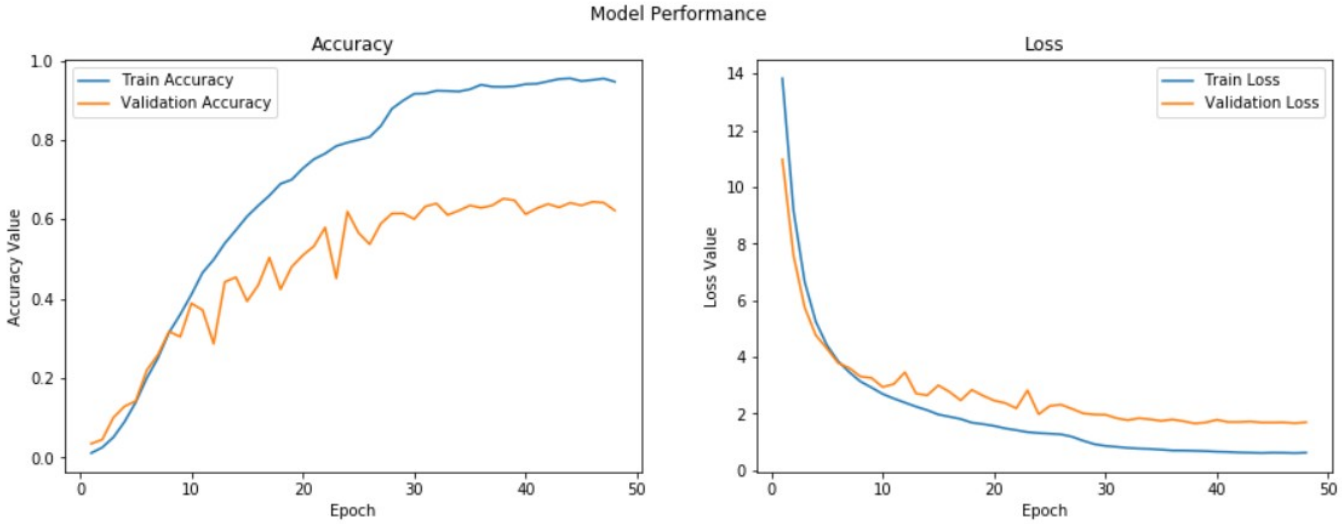
In order to confirm the robustness of the model, all 3 models were tested against the test set, and the Xception fine-tuned model has an accuracy of 72.47% which is higher than the goal of achieving at least 70% classification accuracy in the Problem Statement section.

A detailed summary of results is provided in the table below.

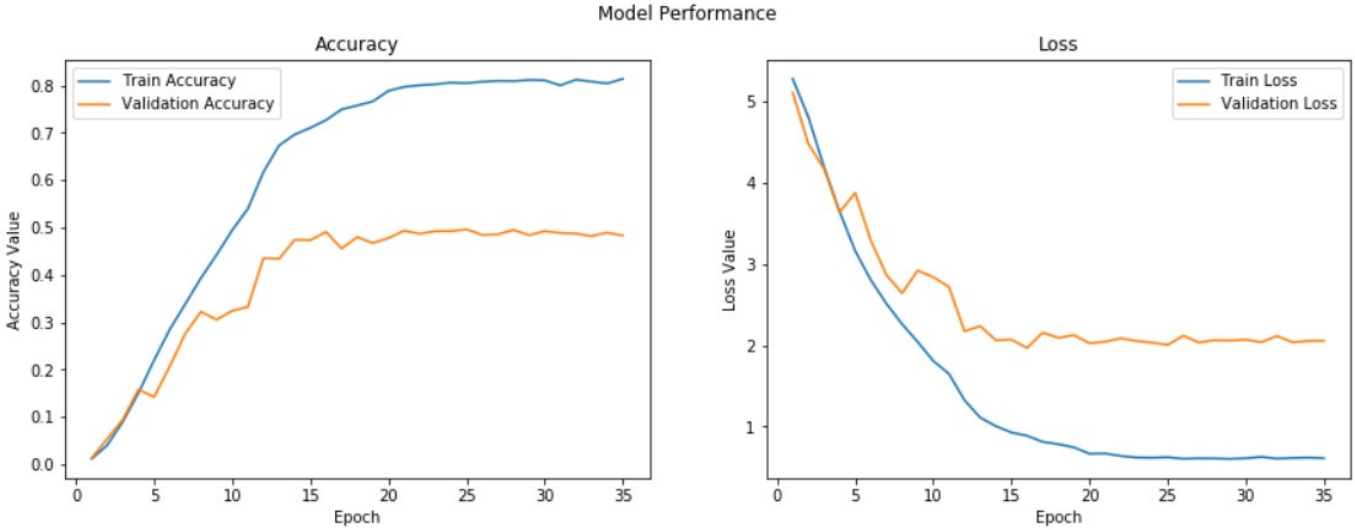
Model	Training	Validation	Test
Benchmarking	1.96%	1.72%	1.93%
Fine-tuned Xception	93.37%	65.22%	72.47%
Fine-tuned InceptionV3	72.68%	49.10%	57.70%

The training / validation accuracy and loss of the fine-tuned models are shown below. That there is a big gap between training and validation curve indicates the 2 fine-tuned models are subject to overfitting.

Xception fine-tuned model:



InceptionV3 fine-tuned model:



Justification

As stated in earlier section, both fine-tuned models have managed to outperform the benchmarking model by more than 30 folds. Additionally, the goal set out earlier before this project has been achieved by the Xception fine-tuned model, and thus, it has been chosen as the final model.

V. Conclusion

Free-form Visualization

Below are the Xception fine-tuned model predictions of 10 car images drawn randomly from the test set. Based on the accuracy reported for the Xception fine-tuned model, I expect the model to be able to predict 7 or 8 car makes and models correctly. In fact, 8 out of 10 predictions are correct.

[True] Actual: Hyundai Veloster Hatchback 2012 | Predicted: Hyundai Veloster Hatchback 2012

[True] Actual: Volkswagen Beetle Hatchback 2012 | Predicted: Volkswagen Beetle Hatchback 2012

[True] Actual: Ford GT Coupe 2006 | Predicted: Ford GT Coupe 2006

[True] Actual: Ford F-150 Regular Cab 2007 | Predicted: Ford F-150 Regular Cab 2007

[False] Actual: Lamborghini Aventador Coupe 2012 | Predicted: Volvo C30 Hatchback 2012

[True] Actual: Audi S6 Sedan 2011 | Predicted: Audi S6 Sedan 2011

[True] Actual: Bentley Continental GT Coupe 2007 | Predicted: Bentley Continental GT Coupe 2007

[True] Actual: Volvo XC90 SUV 2007 | Predicted: Volvo XC90 SUV 2007

[False] Actual: Chevrolet Silverado 1500 Extended Cab 2012 | Predicted: Isuzu Ascender SUV 2008

[True] Actual: Audi 100 Wagon 1994 | Predicted: Audi 100 Wagon 1994

Reflection

A summary of the steps used to complete this project can be found below.

1. Download the cars dataset which originated from Stanford University AI Lab.
2. Perform exploratory analysis on the dataset.
3. Preprocess the images (i.e. car cropping, image augmentation, tensor transformation).
4. Train and test benchmarking model against test set.
5. Fine-tune Xception and InceptionV3 model.
6. Test transfer learning models against test set.
7. Validate transfer learning model performance by charting out the learning curve.
8. Compare the accuracy of the transfer learning models against the benchmarking model.

I find step 5 the most challenging as I had to define an architecture that is not prone to overfitting and decide on the number of layers of the existing model to keep and retrain. In the end, my model is still overfitting even though I tried several preventive methods such as Dropout and Kernel regularization. Despite unideal results, the model has met my accuracy goal, and I find it extremely satisfying when I tested the model predictions against random images from the dataset.

Improvement

Because I leveraged RAM to do the image preprocessing, I ran into an Out of Memory problem when trying to increase batch size. If it was possible to increase batch size from 32 to a higher value (e.g. 64 or 128), the learning process might be slower, but a more stable model with lower variance in classification accuracy could be obtained.

Based on my research, the winning solutions for various Kaggle competitions mostly appear to be an ensemble of models. Inspired by this, I will try to combine my 2 fine-tuned models and see if I can achieve higher accuracy using the ensembled model.

Reference

- Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei (2013). *3D Object Representations for Fine-Grained Categorization*.
- François Chollet (2016). *Xception: Deep Learning with Depthwise Separable Convolutions*.
- Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, Li Fei-Fei (2017). *Fine-Grained Car Detection for Visual Census Estimation*.