# Machine Learning Engineer Nanodegree

## Capstone Project

### Car Make and Model Image Recognition

Chu Nguyen Van

July 31, 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.
* Perform exploratory analysis and some visualizations on the dataset to understand how the data is distributed.
* Preprocess, 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.
* Validate benchmarking model performance by charting out the learning curve.
* Fine-tune Xception Model.
* Test transfer learning model against test set.
* Validate transfer learning model performance by charting out the learning curve.
* Compare the accuracy of the transfer learning model 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.

### A confusion matrix can also be used to show the car classes that the model struggles the most to discern.

### 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.

A sample image from the data set is shown below.

|  |  |
| --- | --- |
| Original image with bounding box | Cropped car image within the bounding box |

### 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*: 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.

### Detailed work-flow is provided below.

* Data Collection: 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.
* Explore the dataset: construct a bar chart to understand the number of car images associated with each car make and model.
* Image Preprocessing: 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.
* Data preparation:
  + Split original training dataset into 2 sets:
    - Training: 5,700 images – 70% of the original training set.
    - Validation: 2,444 images – 30% of the original training set.
  + Use the original test set as a Public Test set of 8041 images.
  + 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.
* Train and test benchmarking model against Test set.
* Fine-tune Xception 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.
  + 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.
* Fine-tune InceptionV3 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.
  + 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.
* Compare the accuracy of the transfer learning fine-tuned models against the benchmarking model.

### 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

### Implementation

### Refinement

### IV. Results

### Model Evaluation and Validation

### Justification

### V. Conclusion

### Free-form Visualization

### Reflection

### Improvement

### 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*.