Machine Learning Engineer Nanodegree

Capstone Proposal

Car Make and Model Image Recognition

Chu Nguyen Van
July 10, 2019

Domain Background

Car Make and Model Image Recognition is a sub problem belonging to a large family of Fine-grained Recognition problems. 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.

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 problem is to automate the process of car recognition process from the images, including the make and model of the vehicle.

The aim of this project is develop a model with at least 70% prediction accuracy.

Datasets and Inputs

Dataset Description
The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

Acknowledgement
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.

3D Object Representations for Fine-Grained Categorization
Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei
4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.

Solution Statement

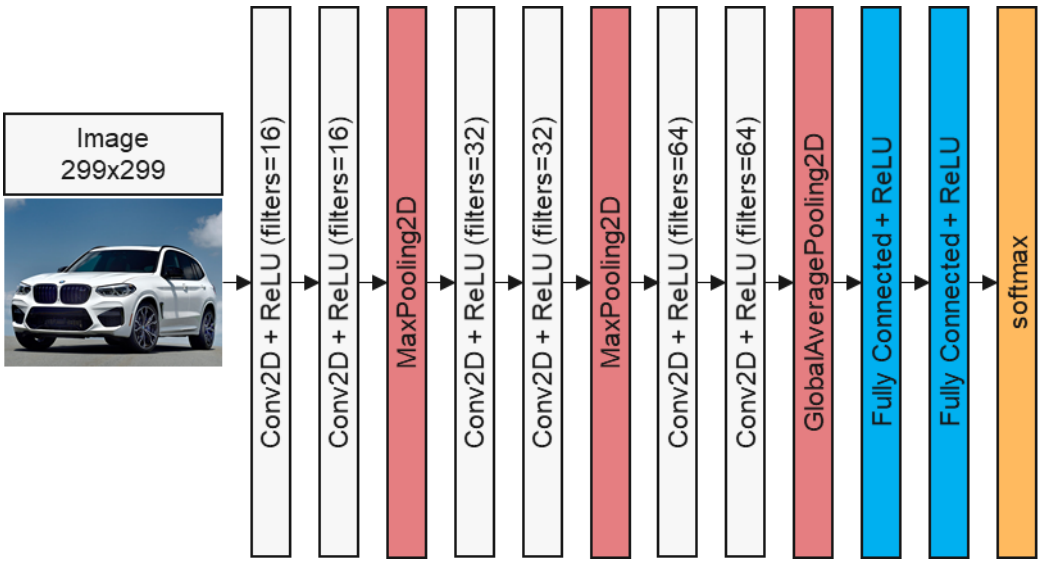
Inspired by the results of the Deep Learning research for Cancer Detection by Sebastian Thrun, I intend to use Transfer Learning to solve this classification problem. To be specific, I will use the Xception model initialised with "imagenet" weights. Since the cars dataset is much smaller than the ImageNet dataset, retraining the whole model will probably result in overfitting. Thus, it is my intention to keep the majority 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 simple CNN model from scratch to serve as a benchmark for my transfer learning model. The architecture of the benchmarking model is shown below.



Evaluation 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.

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

Project Design

Programming language: Python 3.7

Library/Framework: Pandas, Numpy, Scikit-learn, Tensorflow Keras

Workflow:

1. Data Collection: the cars dataset is originated from Stanford University AI Lab and was downloaded from the aforementioned website in the *Datasets and Inputs* section

2. Explore the dataset: construct a bar chart to understand the number of car images associated with each car make and model.

3. 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.

4. Data preparation:
    a. Split original training dataset into 3 sets:
        i. Training: 6515 images – 80% of the original training set.

        ii. Validation: 815 images – 10% of the original training set.
        iii. Private Test: 814 images – 10% of the original training set.
    b. Use the original test set as a Public Test set of 8041 images.
    c. Perform one-hot encoding on the car labels.
    d. Transform datasets into 4D tensors so that they can be used as input for Tensorflow Keras CNN.
    e. Perform image augmentation to add more variety to the images.

5. Train benchmarking model.

6. Test benchmarking model against Private and Public Test set.

7. Validate benchmarking model performance by charting out the learning curve.

8. Fine-tune Xception Model.
    a. Load base Xception model with pre-loaded "imagenet" weights.
    b. Freeze the first 94 layers.
    c. Make the rest of the layers trainable.
    d. 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.
    e. 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".

9. Test transfer learning model against Private and Public Test set.

10. Validate transfer learning model performance by charting out the learning curve.

11. Compare the accuracy of the transfer learning model against the benchmarking model.

Reference

Xception: Deep Learning with Depthwise Separable Convolutions