CS230

What Car Is It?
Vehicle Classification with Deep CNN

Sean Li
Department of Computer Science
Stanford University
seanli19@stanford.edu

Abstract

The ability to accurately identify a vehicle's make, model and production year from images can be of great value to many applications. This project explores the use of a deep convolutional neural network for this very task. Through training in stages, transfer learning and fine tuning of hyperparameters, an InceptionV3 network achieves Top-3 accuracy of 95.7% on a dataset with over 600 classes.

1 Introduction

With the increasing number of video recording devices existing in our society today, the ability to make sense of the recorded data has become increasingly important. One interesting aspect of this is to recognize the particular make, model and production year of a recorded vehicle, which can be of great value to fields like security, surveillance, investigative activities and even autonomous driving. This project explores using deep convolutional neural network to identify the make, model and year of a vehicle using only its picture as input.

2 Related work

Several groups have attempted vehicle classification with convolutional neural networks (CNN) in the past. For some specific task, such as automatic toll collection, projects like the one led by Y. Zhou, H. Nejati et al. [3] focus on classifying the type of vehicle (e.g. truck v.s. sedan) rather than make and model. However, the projects that do predict the make and model of a vehicle [1][2][4] typically omit the manufacturing year from the classification. F. Tafazzoli, H. Frigui, K. Nishiyama proposed a new dataset that contains the "year" information as part of the image label and subsequently trained a ResNet50 network that achieved Top-1 and Top-5 accuracy of 51.76% and 92.90% [5]. This project uses the same dataset but aims at obtaining a more accurate model by exploring a more complex network architecture. It's also worth mentioning that some of these previous projects use techniques such as YOLO [3] to more accurately outline the vehicle of interest from input picture prior to training the classification network. While interesting and useful for certain applications, this technique is not being utilized in this project as it's deemed unnecessary considering the type of input samples used.

3 Dataset

As mentioned previously, an existing dataset, namely Vehicle Make and Model Recognition database (VMMRdb), published by F.Tafazzoli et al. [5] in 2017 will be used. This team cleverly

collected pictures from vehicle listings on various classified websites across North America, resulting in 291K images spanning over 9,170 classes of vehicles. Thanks to the nature of this data collection mechanism, this dataset contains "real-life" images from various photographing devices, camera angles, lighting conditions, background environments and a range of resolutions. In this dataset, each unique combination of a vehicle's make (e.g. Toyota), model (e.g. Camry) and year (e.g. 2007) is categorized as a class (e.g. toyota_camry_2007). Due to the nature of this approach to data collection, some classes have a relatively large sample size of more than 800 images while some less popular cars only have a handful of images. To concentrate the efforts as well as computing resources on obtaining a satisfying end result, this project will only use the classes for newer vehicles (i.e. year 2000 and later) with more than 100 images each. This trims the dataset down to ~118K images with 605 unique classes. Figure 1 below is the visualization of these classes with the size of a circle representing the relative number of samples in that class.

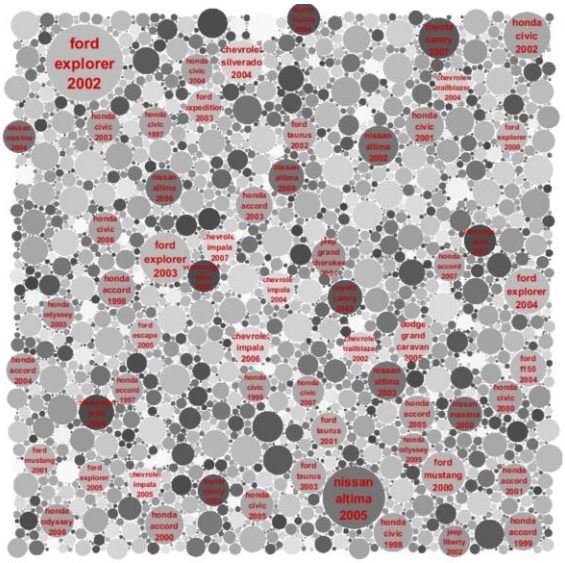

Figure 1. Distribution of images per class in the dataset [5]

This dataset is further divided into train/dev/test sets following an 80/10/10 split. For InceptionV3, the images have their resolutions normalized to (299,299,3) and 3 color channels normalized with division by 255 – same configurations that the default weights are trained with using ImageNet. The training set only has around 155 images on average per class, so data augmentation is applied. Further, the actual class distribution is quite imbalanced as shown in Figure 1 above, the technique of weighted loss is explored so that smaller classes can have relatively higher weights.

4 Methods

Various CNN architectures (VGG16, ResNet50 and InceptionV3) are evaluated on a small subset of the dataset and InceptionV3 is selected as the final architecture to further fine-tune using the full dataset.

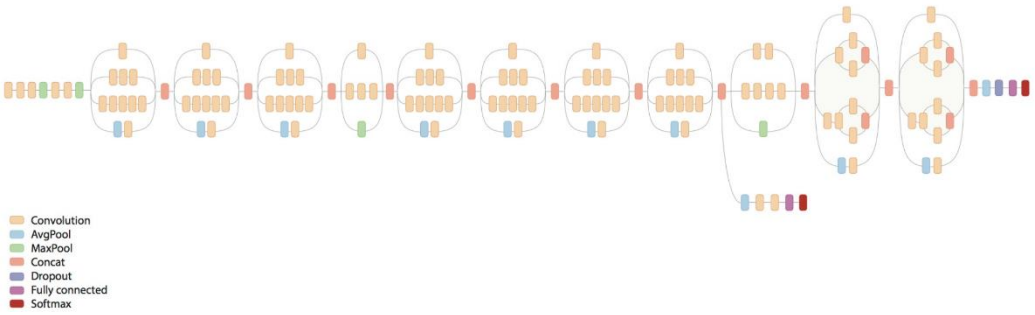

Figure 2. The block diagram of InceptionV3 architecture

As shown in Figure 2 above, InceptionV3 model consists of many sub-blocks and each block contains multiple filters of different kernel sizes running in parallel. Comparing to manually pre-define filter, this design brings extra flexibility as the network automatically tunes to the best performing filters for a given layer through the training process [6]. Furthermore, InceptionV3 is over 300 layers deep and has over 24 million trainable parameters, enabling the detection of very complicated features.

The model is initialized with pre-trained weights using ImageNet with all convolutional layers frozen and only the fully connected (FC) layers being trainable. The idea is that the technique of transfer learning could potentially benefit this project, which fundamentally is also an image classification problem.

5 Experiments/Results/Discussion

5.1 Training Strategy

As mentioned previously, the model is initialized with pre-trained weights (using ImageNet) to take advantage of transfer learning. Initial training has all the convolutional layers frozen, but it yields unsatisfactory dev accuracy. Next, under the assumption that the low-level features of VMMRdb images are similar to those of ImageNet's and their weights can be "transferred", final convolutional layers targeting high level feature detection are unfrozen. This results in slight performance improvement but with much more to be desired. Similarly, initial convolutional layers are then configured as trainable instead of the final layers, which doesn't yield dramatic changes either. At this point, it's realized that not only does the vehicle classification task require rather unique low-level feature detection, the entire InceptionV3 network should be unlocked for training. This is necessary even though the 24 million trainable parameters mean that the training time per epoch will increase multiple folds.

5.2 Hyperparameter Tuning

In order to speed up the process of hyperparameter tuning, a subset of the full dataset is selected, specifically 40 (out of the 605) classes with the most images. An initial search of various parameters is then carried out and the top 3 sets with the highest dev accuracies are listed in Table 1 below.

Set #	Epoch	Optimizer	Learning Rate	Momentum	Batch Size	LR Decay
1	30	SGD	2.00E-04	0.8	32	0
2	30	SGD	1.00E-04	0.8	32	0
3	30	SGD	1.00E-04	0.8	32	1e-2/epochs

Table 1. Top 3 hyperparameter sets from the initial search

A discrete desktop class NVIDIA Titan V board with GV100 GPU and 12GB of memory is used for training. With all layers trainable, it supports a maximum batch size of 32, which happens to be the best performing among the tested batch size values. Adam, SGD and RMSProp optimizers are tested and SGD has the best results. Since all 3 hyperparameter sets have similar accuracies, plots of their accuracies as well as losses during the 30 epochs of training are compared and set #3 shows the most potential for significant improvement over longer period of training, it is therefore selected as the candidate for further tuning. First, longer duration of training is carried out to establish baseline model/weights to be used for training with the full dataset later. Figure 3 below shows the accuracy and loss plots during 200 epochs (~9 hours) of training with the "top 40" dataset using hyperparameter set #3.

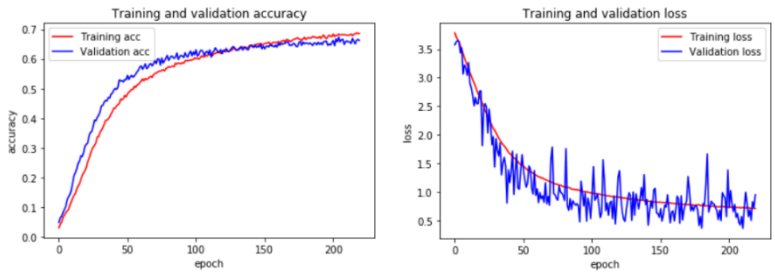

Figure 3. Plots of accuracy and loss of hyperparameter set #3 over 200 epochs of training

I now have a good baseline model and weights, which fit the "top 40" dataset well, to be tested on the full dataset. Considering the full dataset has 605 classes and 6x more images, the current model performs poorly, as expected. Therefore, another round of hyperparameter tuning ensues. Using 30 epochs of training internals, it appears the main issue now is the variance gap between training and dev accuracies, and changes in current hyperparameters don't necessarily reflect as improvements in the results – regularization is then experimented with. Eventually, L2 regularization as well as a dropout unit in the FC layers leads to the best results, as shown in Table 2 below.

Epoch	Optimizer	Learning Rate	Momentum	L2	Batch Size	LR Decay	Dropout
30	SGD	1.00E-04	0.8	0.005	32	1e-2/epochs	0.5

Table 2. Finalized hyperparameters (with regularization terms)

Once the hyperparameters are settled, longer training sessions are carried out. After ~300 epochs and over 60 hours of training, the network saturates at 44.7% Top-1 accuracy and 95.7% Top-3 accuracy.

5.3 Error Analysis

Even though the obtained Top-3 accuracy is much better than the results achieved by F. Tafazzoli et al. [5], error analysis is still needed to better understand the errors in the predictions and potentially improve the model. As a first step, activations of various convolutional layers are visualized to examine what features the network is looking for. Figure 4 below contains the outputs of the first conv2d layer's 32 filters, using a mis-predicted image as input.

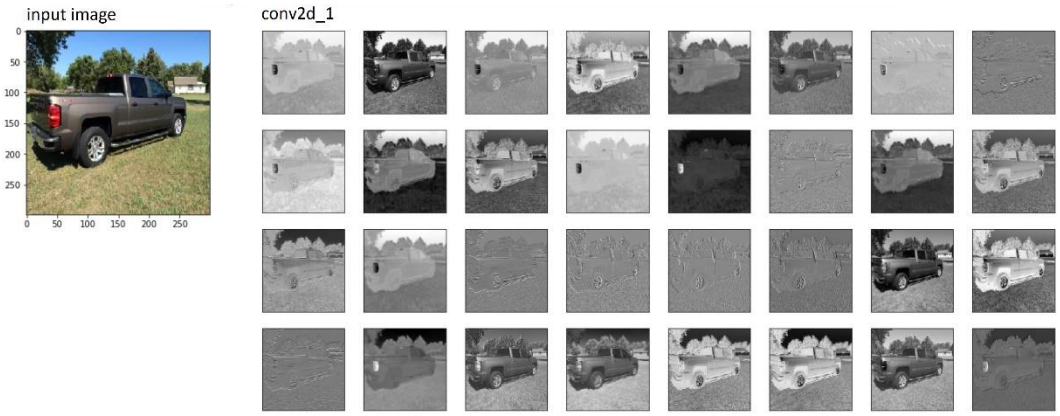

Figure 4. An example input image and its activations from the first conv2d layer

It appears that the filters are somewhat detecting the right features. We can observe that the outline of the truck is highlighted correctly and certain features such as the rims and taillights are

also very prominent. Activations from subsequent layers are also examined but as the layer gets deeper, the visualizations are more convoluted and make less sense visually. As there's no obvious issues observed in the filter outputs, it's decided to look at the incorrectly predicted images more closely next.

By plotting the confusion matrix of selected classes, it's soon noticed that many wrong predictions are correct for the make and model portion but slightly off in the year number. To quantitively understand the percentage of wrong year predictions, the list of 11622 total predictions are then processed to have the year number removed and compared to the true label again. Surprisingly this leads to a mere 562 incorrect predictions. In other words, among the 11622 images, 44.7% have their make/model/year predicted correctly, 50.5% have only make/model (but not year) predicted correctly and only 4.8% have their make/model mistaken. Common sense suggests that vehicles of same make and model but from different production year tend to have very similar exterior features and would be difficult to extinguish even for a human expert. Manual inspection through this category of images seems to confirm this theory in many instances. Same treatment of visual inspection is applied to some random samples from the 4.8% error category as well, it appears, not so surprisingly, that many of the original and predicted vehicles have striking similarities. Figure 5 below shows some examples of this case.

Figure 5. Examples of similar-looking labeled and predicted vehicles

Even though some of these vehicles are difficult to distinguish even for a human expert (e.g. first set of pictures depicting the minivans in Figure 5 above), others are technically possible for a deep CNN to identify. Features like the brand logo or even the model's name (e.g. texts "325i" and "330i" printed on the BMW's in above Figure) are present in some input images but are clearly ignored by the InceptionV3 network. To capture these fine features, perhaps a standalone network using YOLO algorithm can be deployed first to localize these features and classify them before sending the image through the InceptionV3 network.

6 Conclusion/Future Work

For the chosen dataset with 117.6K images of 605 unique vehicle classes, after tuning weights from all parameters, InceptionV3 network with hyperparameters described in Table 2 is able to achieve Top-1 accuracy of 44.7%, Top-3 accuracy of 95.7% for make/model/year prediction, and Top-1 accuracy of 95.2% for make/model only prediction. For future improvement, an even bigger and more complex network can be explored to solve the problems of missing features such as logos and texts. Furthermore, the Top-1 dev accuracy plateaus during training while training accuracy keeps increasing, indicating a variance problem. The original authors of the VMMRdb dataset focused on Craigslist and Amazon postings but didn't collect anything from the vast network of dealerships' websites across the country. If a crawler script can be written to (somehow) utilize this data source, this jump in the amount of input data could potentially improve the performance of my model even more.

7 Contributions

I, Sean Li, am the sole person to have worked on this project. All references have been included in the References section, with open source code credited accordingly in my project code submission.

8 Code Submission

https://github.com/seannlii/deepcar

References

[1] Hyo Jong Lee, Ihsan Ullah, Weiguo Wan, Yongbin Gao and Zhijun Fang (2019) Real-Time Vehicle Make and Model Recognition with the Residual SqueezeNet Architecture

[2] Derrick Liu, Yushi Wang (2015) Monza: Image Classification of Vehicle Make and Model Using Convolutional *Neural Networks and Transfer Learnin*g

[3] Yiren Zhou, Hossein Nejati, Thanh-Toan Do, Ngai-Man Cheung, Lynette Cheah (2016) Image-based Vehicle Analysis using Deep Neural Network: A Systematic Study

[4] Burak Satar, Ahmet Emir Dirik (2019) Deep Learning Based Vehicle Make-Model Classification

[5] Faezeh Tafazzoli, Hichem Frigui, Keishin Nishiyama (2017) A Large and Diverse Dataset for Improved Vehicle Make and Model RecognitionF. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna (2015) Rethinking the Inception Architecture for Computer Vision