Deep Learning Car Classification of the Stanford Cars Dataset

Group Members:
Khoi Ngo, Branden Shin, Andrew Lee, Brandon Low-On

1. Motivation and Significance

Learning the theory and skills behind Deep Learning has opened up a whole new dimension of relatable problems to tackle. Our team selected a problem that is collectively interesting for the group, utilizes the theory and skills from the class, and has a scope that is solvable and within reach of our skill sets. The objective of this project is to train a classifier that distinguishes the make (or brand) of automobiles from the Stanford Cars dataset [1].

This dataset lends itself towards a classification problem where we can utilize many concepts learned in this course. Cars and the model of each car are of large interest for the group as we are all enthusiasts of the automotive industry. Being able to create a classifier for a topic we are passionate about will lead to more interest from the group to do well.

In terms of real-world applications, we believe this project has the potential to help police services. A well trained classifier can efficiently determine the make or brand of a car based off of an image. For example, for police chases or hit and runs, if an individual was able to capture an image of the culprit's car the classifier will be able to help identify what brand it belongs to and narrow down the suspects. Meyer and Kuschk [2] talk about how object detection can be achieved using deep learning on radar and camera images, showing a classifier of this nature has potential. In the future, companies can also easily scan traffic data that could contain hundreds of cars in it and quickly identify the brand and model of each individual car [3]. This data can be collected and applied to a variety of research projects.

2. Methodology

2.1 - The Stanford Cars Dataset

This project uses the Stanford Cars dataset [1]. The original dataset contains 16,185 images, split into a training (8144 images) and testing set (8041 images). There are 196 classes of cars based on its make, model, and year (ex: "*Acura Integra Type R 2001*"). The image quality unfortunately varies from image to image, as they capture the car at various angles, backgrounds, lighting, and distance. Figure 1 below shows a few of the images from the dataset and their labels.

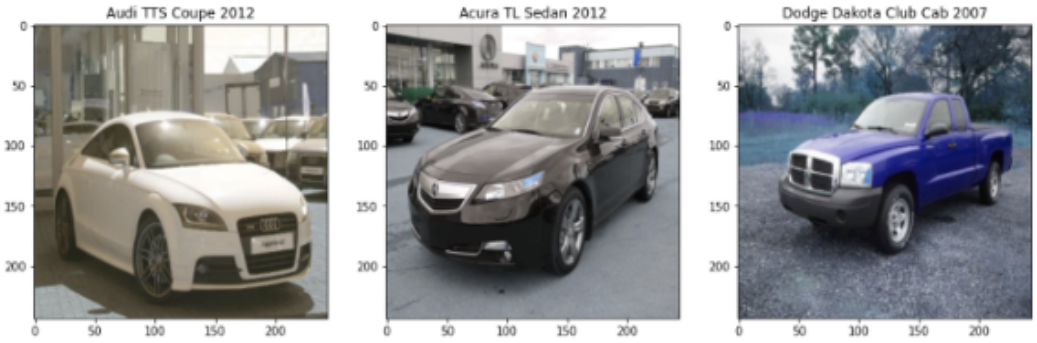

Figure 1: Example images with class labels

2.2 - Adjustments Made to the Dataset

After attempting to train our models on the original dataset, we realized there were some alterations that we could make to the data itself to improve results.

The first adjustment we made was to increase the number of training samples we had by moving half of the testing dataset to the training dataset. More training data generally leads to better results and helps mitigate overfitting [4]. After this adjustment, we were able to train our models with 50% more training data (12,159 training images and 4026 testing images).

Secondly, we cropped each image to remove background noise in the samples and ensure that the focal point of each image was the car. Stanford conveniently provided a separate file detailing the bounding boxes for each image, and a few examples can be seen in Figure 2 below. Cropping the images around these boxes would reduce background noise and improve results. An adapted dataset was provided online that contained images cropped to their respective bounding boxes [5].

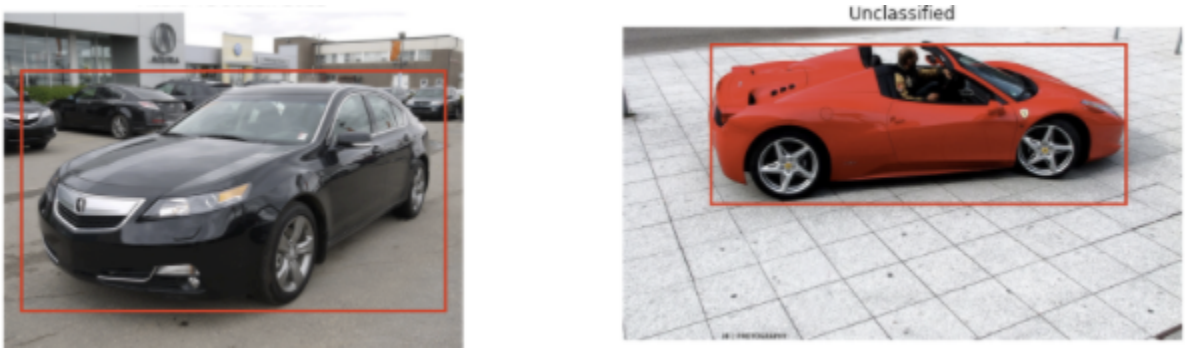

Figure 2: Bounding boxes on car images to isolate the vehicle and remove background noise

The original Stanford Cars dataset is a classification problem to determine the make *and the model* of each vehicle. This was a 196 class problem, with an average number of 83 samples per class. The final adjustment we made to the dataset was to reduce the complexity of the problem

by combining samples of the same make to reduce the number of classes from 196 to 49. For example, the classes Acura RL Sedan 2012, Acura TL Sedan 2012, Acura TL Type-S 2008, Acura TSX Sedan 2012, Acura Integra Type R 2001, and Acura ZDX Hatchback 2012 were all combined to become one "Acura" class. This increased the average samples per class from 83 to 330. Thus, we reduced the original classification problem of the make and model to just the make of each vehicle.

2.3 - Pre-Processing

We also employed several data pre-processing and augmentation techniques. First and foremost, we normalize image size by re-sizing each image to dimensions of 224 by 224. Additionally, we scaled the images by dividing each pixel by 255 (min-max scaling). After resizing and scaling, we employed several data augmentation techniques including randomized rotations by 20^{o}, zooms of 10%, horizontal flips, shifts in width/height by 10%, and shear rotations of 20%. Finally, the training data is split into 75-25 training and validation sets with a stratified class distribution (results in 9138 training, and 3021 validation images).

3. Machine Learning Approach and Algorithms

With a plethora of deep-learning networks and strategies taught in this course, our group decided to employ many different techniques to gauge the effects on model performance. We developed several convolutional neural networks (CNN) to learn from this data, and experimented with techniques such as transfer learning to achieve the best possible results.

The following subsections of this report will describe the different strategies our group experimented with. We will evaluate the results of each of these strategies to determine the best performing model on our dataset. For our preliminary midterm results, we ran the models on the "raw" dataset discussed in Section 2.1 (196 classes). For the final results, we ran the models again on the "adjusted" dataset discussed in Section 2.2 (49 classes).

To be consistent, all images were imported and pre-processed using the same code, and each model was trained with at least 100 epochs, a learning rate of 0.0001 (for transfer learning, fine tune with a learning rate of 0.000001), and a batch size of 128.

3.1 - CNN's Trained From Scratch

In this section, we will discuss the CNN models we trained from scratch on the car data. These models were inspired by existing architectures, and our objective was to evaluate if training models from scratch was suitable for this task.

CNN Control Model

For our first model, we developed a CNN architecture from scratch to act as our control model for the next three existing CNN models (AlexNet, VGG-16, and ResNet50). A secondary objective of this model was to explore if it is better to use a predefined architecture, or design a model from scratch. Using methods and CNN architecture learned from class, the first model was designed with 4 total convolutional layers with a dropout layer after each set of two convolutional layers. Also included is a one max pooling layer for our model. Each convolutional layer uses a Relu activation, and a softmax activation layer is used at the end to perform the classification. We hope that with our first initial model we can set the benchmark in terms of accuracy for our three later designed models. This model had 59,153,761 total trainable parameters.

AlexNet Architecture

The AlexNet CNN architecture was one of the first models to distribute the neurons on multiple GPUs for training large datasets [6]. The model contains five consecutive convolutional layers followed by three fully connected layers. The convolutional layers are all followed by a Relu non-linear activation layer, and layers 1, 3, and 5 are followed by a batch normalization layer and a 3x3 max pooling layer that performs overlapping pooling. The three fully connected layers classify the image at the end of the model. Each of the dense layers uses Relu activation and have a dropout probability of 0.5. The final layer of the model is a 49 dimension softmax layer that assigns probabilities to our neurons for each of our 49 classes. The model's architecture results in a total of 46,950,513 trainable parameters.

3.2 - CNN's With Transfer Learning

Transfer learning is a technique where the weights of a network pre-trained on another dataset are leveraged and adapted for another problem [7]. This allows you to repurpose a previously learned classifier for a new task. Transfer learning is suitable when the dataset in question is smaller, and helps mitigate overfitting and reduce computation time. CNN models pre-trained on the ImageNet dataset are notoriously popular in transfer learning, and have seen successful results [8]. Our group explored two different CNN architectures using base models trained on the ImageNet dataset, and fine-tuned them for our task.

Both of the transfer learning architectures were also trained from scratch to better evaluate the effects of transfer learning on the cars dataset. This was added to our strategy after the midterm checkpoint, and was not performed for the original 196-class dataset.

VGG-16 Architecture with Transfer Learning

Like the name suggests, VGG-16 is a CNN architecture with 16 layers: 13 convolutional layers of 3x3 filter and 3 fully-connected layers. The convolutional layers are organized into blocks containing 2 to 3 layers, and a 2x2 max-pooling layer is placed between each block. The fully-connected layers are featured at the end of the model to classify the image [9]. For our task,

we imported VGG-16 model weights pre-trained on the ImageNet dataset for transfer learning. We fine-tuned the model on the cars dataset and added a soft-max layer with 49 dimensions. For our preliminary results (196 classes) we developed a model with 19,632,132 trainable parameters. After adjusting the dataset this number was reduced to 15,944,049 trainable parameters.

ResNet50 Architecture with Transfer Learning

ResNet50 is a Deep Residual Network similar to CNN's. The network uses identity connection between layers [10]. The ResNet50 architecture comprises 4 stages. The network starts with a convolution layer of 7x7 kernel and then a max-pooling layer of 3x3 kernel. The first stage has three residual blocks with each block performing a convolution operation on three layers. The layers alternate kernels of 1x1 and 3x3. As it moves from one block to another, there is an identity connection from the start to the end of the block, hence these blocks are called residual blocks. The next stages replicate stage one, but we double the channel width and half the input size [10]. The last part of the architecture has a Flatten layer and a fully connected Dense layer. In total, there are 46 layers and we had 19,669,188 trainable parameters. Similar to the VGG-16 model, we import weights pre-trained on the ImageNet dataset and use it for transfer learning. We added a Flatten and Dense layer to converge our neurons into our 49 classes. When we applied our new image preprocessing steps, the number of trainable parameters decreased to 4,917,297 parameters.

4. Results

In total, 6 models were developed and the accuracy scores on the test set were recorded. As mentioned in Section 2 of this report, each model was trained using the exact same pre-processing methods on the input data, and with the same hyperparameters. Table 1 below shows the results of our experimentation, where we compare the results of our models on the adjusted dataset versus our results on the original dataset from the midterm.

Table 1: Accuracy on Test Set for each Model

Model	Preliminary test accuracy (196 classes)	Test accuracy (49 classes)
Control Model (From Scratch)	0.0572	0.2754
AlexNet (From Scratch)	0.0295	0.1686
VGG-16 (Transfer Learning)	0.3441	0.7052
VGG-16 (From Scratch)	-	0.2064
ResNet50 (Transfer Learning)	0.0515	0.2131

ResNet50 (From Scratch)	-	0.2072

In addition to these metrics, we also plotted the training and validation accuracy curves for each model. The plots are shown in Figures 3 and 4 below.

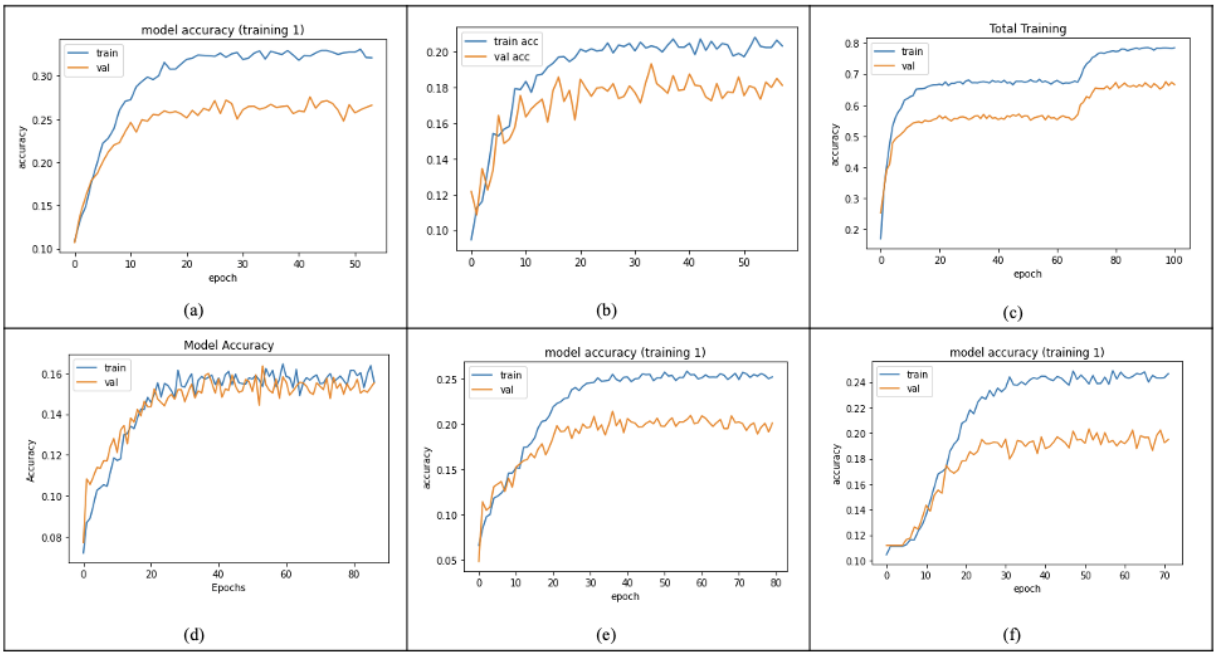

Figure 3: Final training and validation accuracy curves for (a) Control model CNN (b) ResNet50 (from scratch), (c) VGG-16 (from scratch), (d) AlexNet (e) ResNet50 (transfer learning), and (f) VGG-16 (transfer learning)

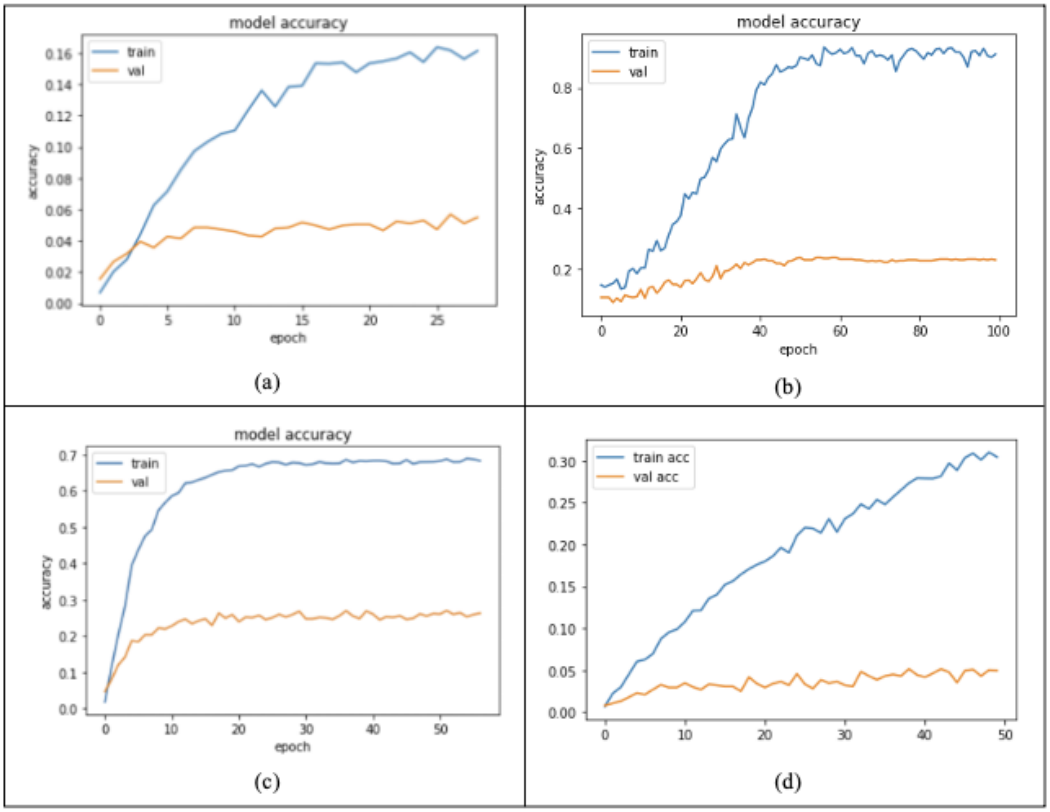

Figure 4: Midterm training and validation accuracy curves for (a) Control model CNN (b) AlexNet, (c) VGG-16, and (d) ResNet50 models

5. Discussion

5.1 - Discussion of our Results

Table 1 shows that our models saw an increase in accuracy after adjusting the dataset. We believe one reason for this is due to the reduction of model complexity. Reducing the number of classes from 196 to 49 also reduced the number of trainable parameters, and this improved results. The dataset had more training samples per class, and we believe this played a role as well. Since each class in the original dataset had only 83 photos on average, we would need more samples per class in order to develop a decently performing model. However, increasing the number of training data will surely increase the time for the model to learn. We found that our models were training slowly due to the limitations in our technology, so it may not be feasible to increase the number of photos unless the technology is powerful enough.

When adjusting the dataset, we also cropped each image to the bounding box of the car. This helped reduce the background noise, and may have played a role in improving model accuracy. It is important to note that this was only possible because Stanford provided the bounding boxes for the car in each image. If this information was not provided, image segmentation would have been necessary to isolate the car from the background.

And finally, adjusting the percentage of training and testing data also contributed to our improvements. For the midterm, we used a roughly 50-50 split, but we changed this to roughly 75-25 for the final results. This manoeuvre provided our models with 50% more training data to learn from. In general, more training data improves model performance, and this contributed to the increase in accuracy between the midterm and final results.

For our midterm results, we observed a significant deviation between the training accuracies and the validation accuracies (Figure 4), which is indicative of overfitting. For our final results (Figure 3), we have reduced the overfitting in our models, which is shown by the smaller deviation between the two curves. This is largely due to the reduction of model complexity in our adjusted dataset.

5.2 - Comparison with Literature

Some of the findings from our experimentation were echoed by the findings of Liu and Wang from Stanford [11]. Similar to our results, they found that using the bounding boxes of the cars to remove background noise helped improve their model accuracy. Additionally, they found that using transfer learning generally increased model accuracy as well. This finding is comparable to our results, where our transfer learning models performed much better than the models we trained from scratch (see Table 1). From this, we can safely conclude that due to the small amount of data and the large number of classes, transfer learning is necessary to develop models with good performance.

From our results, it is also clear that the VGG16 model is superior to the other architectures. This is a finding again echoed by Liu and Wang, as they also leveraged a VGG16 model in their paper and had excellent results [11]. This finding is further backed by another paper by Benavides and Tae, also from Stanford, who developed a VGG16 model with transfer learning to develop their model as well [12]. With the results of our experimentation and the results of these two papers, we can conclude that the VGG16 architecture is well-suited for this classification problem.

In the paper by Benavides and Tae, they also concluded that the number of trainable parameters directly influences overfitting on the training set [12]. When we applied our new image preprocessing step, we saw a decrease in the number of trainable parameters and the models were not as overfitted as the midterm models.

6. Conclusion and Future Work

Through our experimentation, our group was successfully able to develop a CNN model that can predict the make of a vehicle with an accuracy of 70.52% on the testing set. This was accomplished using a VGG-16 model with weights pre-trained on the ImageNet dataset.

There were many key learnings for our group upon the completion of this project. Firstly, we found that techniques such as transfer learning are critical for improving model results when given a limited dataset. Additionally, we found that factors such as having more training data, having more samples per class, and reducing the complexity of the model play a significant role in improving the results of our models.

To further improve model performance, our group suggests further testing with the inclusion of more training data for the models to use as introducing more samples of each car class would help improve the accuracy of our results. For future work there were several features that we wanted to incorporate but were unable to do so due to hardware/RAM and time limitations. These issues directly prohibited us from using ImageDataGenerators for our model. Further improvements could have been done to our models such as incorporating features learned from class such as AutoML and SSL but due to the limitations noted previously we were unable to accomplish this goal. As a team we firmly believe that our classifier's accuracy of the VGG-16 model would have improved with the usage of AutoML and SSL.

One of our future goals we aimed to accomplish next would be to develop and create a machine learning model that can classify car models within a specific car brand. An example of this would be to train the model to look at only Audi cars and then allow it to distinguish and classify the individual models of the Audi brand such as an Audi A4 and Audi Q7. Secondly, another important objective would have been to train the classifier to examine more than one car at a time starting with just two. Finding a way to segment the image for the classifier to distinguish that there are two car models on the image will prove to be a difficult challenge, but it has more real-world applications to it as we are not always going to have one car perfectly isolated in an image.

Code

Overall, we consider our project a relative success, as we have been able to divulge observations from the vehicle images and proved the performance of a model to classify the data. All code for this project can be found in this GitHub repository.

References

[1] J. Li, "Stanford Cars Dataset", *Kaggle.com*, 2018. [Online]. Available: https://www.kaggle.com/jessicali9530/stanford-cars-dataset. [Accessed: 16- Feb- 2021].

[2] M. Meyer and G. Kuschk, "Deep Learning Based 3D Object Detection for Automotive Radar and Camera," 2019 16th European Radar Conference (EuRAD), PARIS, France, 2019, pp. 133-136.

[3] Ni, X., Huttunen, H. Vehicle Attribute Recognition by Appearance: Computer Vision Methods for Vehicle Type, Make and Model Classification. J Sign Process Syst (2020). https://doi.org/10.1007/s11265-020-01567-6

[4] A. Müller and S. Guido, *Introduction to machine learning with Python* =, 2nd ed. Sebastopol: O'Reilly Media, Inc, 2017, pp. 29-52.

[5] "Vehicle-detected Stanford Cars Data classes folder", *Kaggle.com*, 2019. [Online]. Available: https://www.kaggle.com/sungtheillest/vehicledetected-stanford-cars-data-classes-folder. [Accessed: 31- Mar- 2021].

[6] Krizhevsky, A., Sutskever, I., & Hinton, G. (n.d.). ImageNet Classification with Deep Convolutional Neural Networks. Retrieved March 15, 2021, from https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[7] S. Jialin Pan and Q. Yang, "A Survey on Transfer Learning", IEEE transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345-1359, 2021. [Accessed 11 March 2021].

[8] M. Huh, P. Agrawal and A. Efros, "What makes ImageNet good for Transfer Learning?", 2016. Available: https://arxiv.org/abs/1608.08614. [Accessed 11 March 2021].

[9] M. ul Hassan, "VGG16 - Convolutional Network for Classification and Detection", Neurohive.io, 2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 11- Mar- 2021].

[10] "Detailed Guide to Understand and Implement ResNets - CV-Tricks.com", CV-Tricks.com, 2021. [Online]. Available: https://cv-tricks.com/keras/understand-implement-resnets/. [Accessed: 12- Mar- 2021].

[11] D. Liu and Y. Wang, "Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning". [Online]. Available: http://cs231n.stanford.edu/reports/2015/pdfs/lediurfinal.pdf. [Accessed: 16- Apr- 2021].

[12] N. Benevides and C. Tae, *Cs230.stanford.edu*, 2019. [Online]. Available: http://cs230.stanford.edu/projects_spring_2019/reports/18681590.pdf. [Accessed: 16- Apr- 2021].