Fine-Grained Image Classification for Vehicle Makes & Models using Convolutional Neural Networks

Nicholas Benavides
Department of Computer Science
Stanford University
Stanford, CA
nbenav@stanford.edu

Christian Tae
Department of Electrical Engineering
Stanford University
Stanford, CA
ctae@stanford.edu

***Abstract*—Vehicle identification via computer vision is an important task for traffic control, video surveillance, and security and authentication. However, there are challenges with image classification of vehicles due to the fine-grained details that are inherently harder for a computer to detect. Using limited data from the Stanford Cars dataset, we implement transfer learning on a pre-trained Convolutional Neural Network (CNN) framework to classify vehicles based on 196 classes of different vehicle makes, models, and years. After fine-tuning, we achieve a test accuracy of 85% and top-5 test accuracy of 96.3%, surpassing state-of-the-art results.**

Keywords—Vehicle identification, computer vision, fine-grained details, transfer learning, Convolutional Neural Network,

I. Introduction

Cars are ubiquitous in our daily lives as a means of mobility and transportation. As a result, many different models of cars exist around the world, catering to the specific preferences of drivers in terms of aesthetics and functionality. With around a billion vehicles on the planet, each having their own distinct physical features, the need for efficient and robust identification of such objects has increased over the years. Vehicle recognition has a wide range of applications, such as video surveillance, security and authentication issues, and even accessibility features for ridesharing apps. In computer vision, fine-grained classification can be a very challenging problem because differences between classes tend to be much more nuanced compared to class differences in tasks such as object detection. In this case, fine-grained classification is defined to be classification with categories that share similar basic features. Cars are a perfect example of this, as they share a common physical structure, but can differ widely in physical sub-features such as height, width, wheel type, etc. Our classification problem is accurately discerning a car's make and model from a digital image of the car. Historically, humans are fairly good at vehicle classification, as they can pick up on small details such as a logo or lettering on the vehicle. However, this task has been difficult for computers due to the fine-grained nature of the problem.

This project implements a Convolutional Neural Network (CNN) with transfer learning to accurately classify cars from an image dataset by vehicle make and model. The input to our algorithm is a set of images from the Stanford Cars dataset, which contains 196 classes of cars [1]. We then use a CNN to output a prediction of car make, model, and year. Despite the complexity of the problem and limited amount of data in the set of images, we were able to achieve high quality results from fine-tuning of model hyperparameters.

II. Related Work

Fine-grained vehicle classification has recently become a popular topic of study amongst computer vision researchers due to the availability of image datasets and increase in computational power. CNN's are a popular choice for general object recognition/classification, as shown by their ability to achieve state of the art accuracy on generic image classification [2]. Specifically, convolutional neural networks are an effective tool for computer vision and image recognition due to their ability to analyze spatial coherence of images and reduce the number of trainable parameters. We now go over specific applications of CNN's on fine-grained image classification in past research.

In a survey paper, Zhao et al. proposed a "ensemble of networks-based approach" for subset feature learning. The approach implements two main parts: 1) a domain generic convolutional neural network (pre-trained on large-scale dataset of same domain as target dataset and then fine-tuned on target), and 2) several specific convolutional neural networks [3]. In terms of real-life implementation, Xie et al. used a network of five convolutional layers and two fully connected layers for vehicle recognition on a Stanford cars dataset [4]. The study utilized hyper-class data augmentation, which augments fine-grained data with a large number of auxiliary images labeled by some hyper-classes [4]. The study achieved an accuracy of 80% with multitask learning, and 86% with the model pre-trained on ImageNet with multitask learning [4]. Another study used a two-part model: 1) VGG16 network structure for vehicle detection of images with complex backgrounds, and 2) a CNN with a joint Bayesian network for classification [5]. VGG16 is a CNN architecture that features 13 convolutional layers and 3 fully-connected layers.

Transfer learning is a popular choice for vehicle recognition tasks that use large CNN architectures. In transfer learning, a

base network is pre-trained on dataset to create weights and features. This network, with its trained weights, is then transferred to a new dataset by retraining a subset of the base network's learned weights and features. The overall effect is a classifier that fits the new dataset with state-of-the-art results. One study utilized transfer learning to make predictions on the Stanford Cars dataset, in which a fine-tuned VGGNet model was able to achieve 78.9% accuracy, and 94.2% top-5 accuracy [6]. The top model was GoogLeNet full-train which achieved 80.0% accuracy and 95.01% top-5 accuracy [6]. These results were used to evaluate the performance of our fine-grained classifier.

III. DATASET

We used the Stanford Cars dataset to train and evaluate our vehicle classification models [1]. The dataset contains 16,185 image classification pairs of 196 different classes. Each of the fine-grained 196 classes is determined by year, make, and model of a vehicle.

Fig. 1. Sample Images from Stanford Cars dataset.

As shown above, each image contains a car in the foreground against various backgrounds viewed at different angles. The quality of image, as determined by camera type, lighting, focal length, and positioning, varies from image to image. Some images are professionally-taken shots; others are relatively low quality images taken from classified ads on the internet.

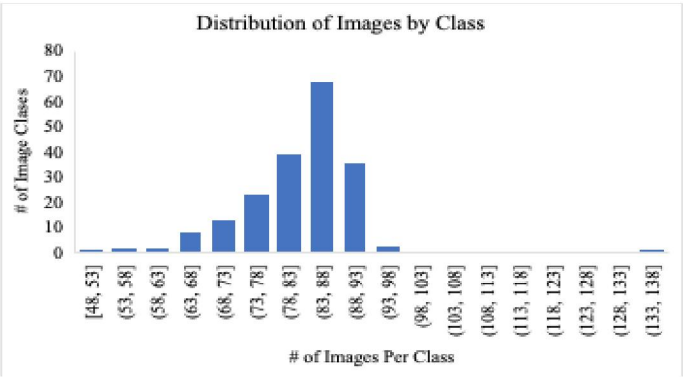

Fig. 2. Histogram of number of images per vehicle class.

The distribution of the number of images per vehicle class is visualized from the histogram above. There is an average of around 83 images per class. There is a minimum of 48 images per class and a maximum of 136 images per class.

In terms of data processing, we first cropped the images based on the bounding boxes provided in the dataset, using a padding of 16 pixels on each side to ensure that parts of the car were not cropped away. From there, we normalized the pixel values for all of the images by dividing them by a factor of 255. Then, we ran a random, stratified train-test split to divide the dataset and preserve the distribution of classes from the full dataset.

We also applied several data augmentation methods on the training set images to increase the size of our training set. These methods included flipping images horizontally, rotating images randomly by up to 30°, shifting images horizontally by up to 20%, and shifting images vertically by up to 10%. Before dataset augmentation, training set size was 11,303 images. Test set size was 2,426 images.

IV. METHODS

For this project and task, we leveraged the VGG-16 network architecture [7], shown below in Figure X. As mentioned before, VGG-16 is a CNN architecture that features 16 trainable layers, namely 13 convolutional layers and 3 fully-connected layers. The convolutional layers are grouped into blocks, with each block separated by a max pooling layer. As an image is passed through the VGG-16 network, the early layers learn simpler features such as edges and shapes, whereas the later layers can learn more complex features such as objects.

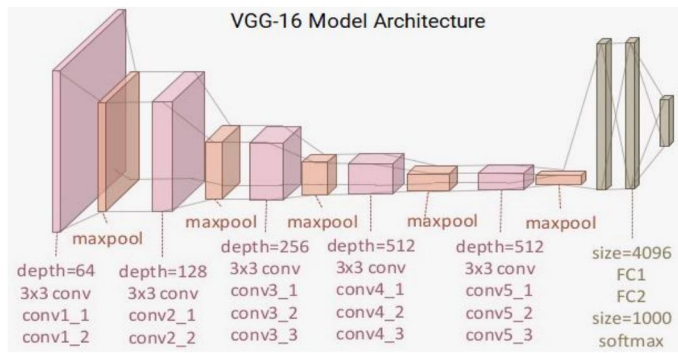

Fig. 3. VGG-16 Model Architecture.

Since our dataset was relatively small and our literature review suggested that transfer learning was a useful approach for fine-grained image classification, we imported VGG-16 weights trained on ImageNet [8] and then fine-tuned them for our particular task. To do so, we removed the 1000-dimension softmax output layer and replaced it with a 196-dimension softmax output layer to model the 196 classes of vehicles in our dataset. The model takes as input color images of size 224 x 224 x 3 and, for each class, assigns a probability that the image is of that class. From there, the model predicts the class with the highest probability. For our cost function, we used categorical cross-entropy loss, which is defined by the equation below for observation o and class c.

$$-\sum_{c=1}^{196} y_{o,c} \log(p_{o,c}) \quad (1)$$

For our baseline model, we fine-tuned the weights on the fully-connected layers of the VGG-16 network, conducting a hyperparameter search but not modifying the model architecture. For our best model, we made several modifications to the original VGG-16 architecture. First, we eliminated the first fully-connected layer and decreased the dimension of the second fully-connected layer from 4096 to 512. In addition, we added a dropout layer after the 512-dimension fully-connected

layer. Both of these changes aimed to reduce the overfitting that we observed in the baseline model. The figure below shows how our modified network differs from the baseline model architecture. Both models feature the same architecture up until the fully-connected layers.

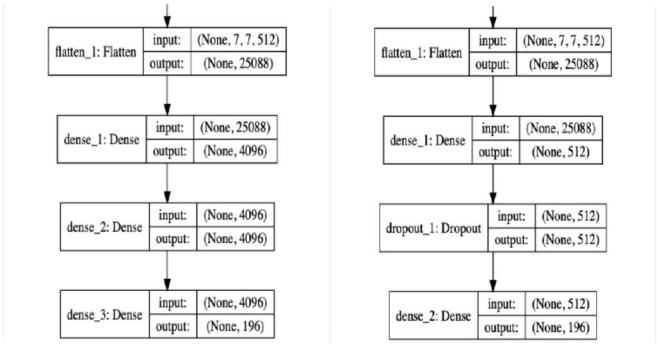

Fig. 4. Last 4 Layers of the Baseline VGG-16 model (left) vs Last 4 Layers of the Modified VGG-16 model (right).

The models themselves were built using NumPy, Tensorflow, and Keras. We also leveraged and adapted code to build the dataset[9], train the VGG-16 models [10] and visualize the network [11].

V. RESULTS

For the baseline model, a hyperparameter search for optimal learning yielded a learning rate =0.005, 10 epochs, and stochastic gradient descent with momentum as our optimizer, with a learning rate of $\alpha = 0.005$, decay = 1e-4, and momentum = 0.5, freezing the first 19 layers of the VGG-16 network. For the best model, a hyperparameter search of the learning rate and weight decay for optimal learning yielded values of α = 1e-4, weight decay = 1e-4, 85 epochs, and Adam as our optimizer, where Beta1 and Beta2 were set to the recommended values of 0.9 and 0.999. The best model also froze the first 17 layers of the VGG-16 network, eliminated the first dense layer, changed the second dense layer to have dimension 512, and adding a dropout layer after the 512-dimension dense layer using a dropout rate of 0.7.

To evaluate our results, we tracked three different accuracy metrics. The first accuracy metric is the overall accuracy, or the percentage of time that the highest probability output by the softmax layer corresponds to the true class. Because of the subtle differences in between classes, we also report the top-3 accuracy, which is the percentage of time that the true class appears in the 3 highest probabilities output by the softmax layer, and the top-5 accuracy, which is the percentage of time that the true class appears in the 5 highest probabilities output by the softmax layer. Table 1 below summarizes our results. Our best model achieved strong performance in all three accuracy metrics on the test set, and solved the overfitting problem we observed with our baseline VGG-16 model.

TABLE I. MODEL RESULTS

Model	Train Acc.	Train Top-3 Acc.	Train Top-5 Acc.	Test Acc.	Test Top-3 Acc.	Test Top-5 Acc.
VGG16 Baseline	85.8%	96.6%	98.2%	52.1%	75.4%	83.83%
Modified VGG16 Network	84.1%	95.6%	97.9%	84.0%	93.9%	96.3%

Figures 5 and 6 below, illustrate the losses and accuracies with respect to the number of training epochs. For both the training and test sets, the loss decreases rapidly and then improves gradually, with the test loss increasing more rapidly at the beginning of training. Similarly, the accuracies increase rapidly at the start of training and then plateau, with the test accuracy accelerating more rapidly than the training accuracy in the earlier epochs.

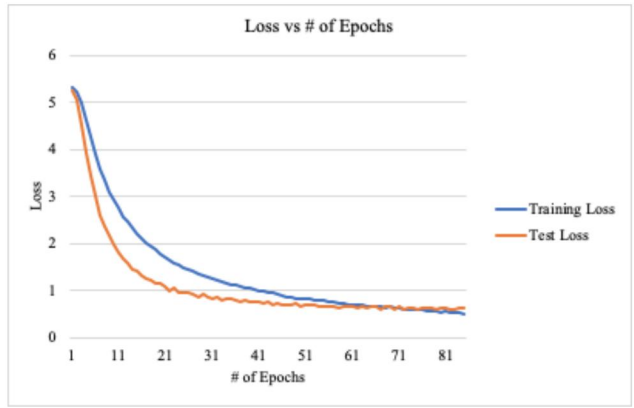

Fig. 5. Training and test losses vs. number of epochs for Modified VGG16 Network.

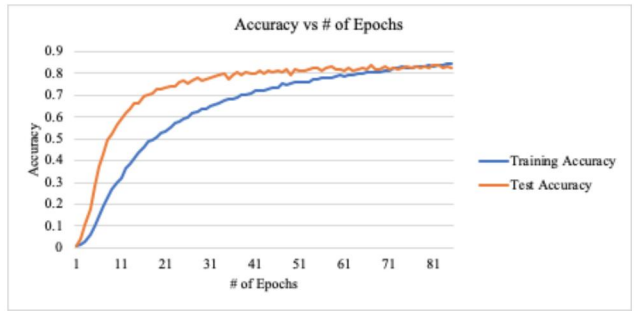

Fig. 6. Training and test accuracies vs. number of epochs for Modified VGG16 Network.

In order to better understand what our fine-tuned model was learning, we generated occlusion maps for random images in the test set. Generally, the maps block out most of the background of the image, indicating that the most important parts of the image for the model is the body of the car itself. An example of one such occlusion map is shown below in Figure 8. The darker areas in the heatmap (middle image of the figure) represent the areas of the image that are most important to the model.

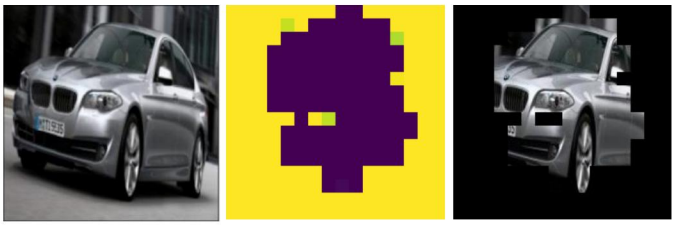

Fig. 7. Original image (left), heatmap (middle), and overlay of binarized heatmap onto original image (right).

In terms of error analysis, we examined 100 images that were not properly classified to gain insights into how the model works and what kinds of errors it was more prone to making. From this analysis, we identified three main groups of misclassifications: one where the model correctly predicts the make but not the model of the car, one where the model predicts a similar style of car from a different make, and rearview images.

Confusing different models from the same car make is an understandable misclassification that even humans make, as manufacturers may have several different models of sedans or SUVs. An example of this error is shown below in Figure 9.

Fig. 8. Model Input = Dodge Charger Sedan 2012 (left), Model Prediction = Dodge Durango SUV 2007.

As shown in the image, different models from the same manufacturer generally have similar bodies. Although the cars do look fairly different, they share similar shapes for the grill at the front of the car as well as the ridges on the hood of the car.

Another common misclassification occurs when the model predicts an image to be a similar car of a different make. These vehicles often have very similar body shapes, with small identifying features like the car's logo or lettering being the most reliable way to distinguish between vehicles, which is very difficult for our model to identify.

Fig. 9. Model Input = Dodge Ram Pickup 3500 Crew Cab 2010 (left), Model Prediction = Ford F-150 Regular Cab 2007.

The final major category of prediction errors stems from images taken from the rear of the car. From a sample of 50 random images in the dataset, we estimate that rearview images make up approximately 5% of the training set. These images make it hard to identify the shape of the car, as seen below in Figure 11.

Fig. 10. Rearview Image of a Maybach Landaulet Convertible 2012.

Thus, while our model performed very well in terms of prediction accuracy, evaluating and classifying the errors made in the model's incorrect predictions gave us insight on what to do next. No model is perfect, and we observed that for some images the model confused different modes from the same car make and predicted an image to be a similar car of a different make. Due to similar body shapes and fine-grained details, it was understandable that our model would incorrectly classify the wrong vehicle. Finally, the fact that the model incorrectly classified many rear-view images can be addressed in future work.

VI. Conclusion

Through our experimentation with the VGG-16 network, we found reducing the number of trainable parameters by eliminating and downsizing dense layers and applying dropout to be an effective approach for reducing overfitting on the training set. Due to the limited images in our dataset, these techniques, in addition to image augmentation and transfer learning, were critical to achieving strong performance. Using this technique, we obtained a test accuracy of 84% and a top-5 test accuracy of 96.3%, surpassing state-of-the-art results for top-5 accuracy. We believe these techniques are applicable beyond vehicle classification and may help to improve performance on a wide range of fine-grained classification problems.

To improve our model's test performance further, we would work to obtain more rear-view images of cars for our training set, as those images represented an outsized number of misclassifications. In addition, we would want to experiment with other model architectures as well as conduct more visualizations of the network to better understand how the model distinguishes between classes.

Code

The code used for this project can be found in [this Github repository](#).

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

Contributions

Nicholas Benavides was primarily responsible for the data processing, image augmentation, and visualizations. Christian Tae was primarily responsible for the literature review and communication of results. Both parties were equally responsible for training models, running experiments, interpreting the results, creating the poster, and writing the final report.

REFERENCES

[1] Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 554-561).

[2] Krizhevsky, A., Sutskever, I. and Hinton, G.: ImageNet Classification with Deep Convolutional Networks. In: NIPS 2012

[3] Zhao, B., Feng, J., Wu, X., & Yan, S. (2017). A survey on deep learning-based fine-grained object classification and semantic segmentation. International Journal of Automation and Computing, 14(2), 119-135.

[4] Xie, S., Yang, T., Wang, X., & Lin, Y. (2015). Hyper-class augmented and regularized deep learning for fine-grained image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2645-2654).

[5] Yang, L., Luo, P., Change Loy, C., & Tang, X. (2015). A large-scale car dataset for fine-grained categorization and verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3973-3981).

[6] Liu, D., & Wang, Y. (2017). Monza: image classification of vehicle make and model using convolutional neural networks and transfer learning.

[7] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[8] Chollet, F. 2016, VGG16, VGG19, and ResNet50, v0.1, https://github.com/fchollet/deep-learning-models/releases/tag/v0.1

[9] Katanforoosh, K. 2018, CS230 Code Examples, https://github.com/kiank/cs230-code-examples

[10] Wang, X. 2019. Fine Tune VGG Networks Based on Stanford Cars. https://github.com/Xiaotian-WANG/Fine-Tune-VGG-Networks-Based-on-Stanford-Cars

[11] Saurabh, P. (2019, May 06). Understanding and Visualizing Neural Networks in Python. Retrieved from https://www.analyticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/