Vehicle Classification using Image Recognition

River Meckstroth (meckstr3)
Vanshika Kadian (kadianva)
Aadhaya Makkar (makkaraa)
Quinn Fransen (fransenq)
Kunal Kale (kalekuna)
https://github.com/rivmeck/CSE-404-Group-Project/tree/main

Abstract

Problem Statement: Use an image recognition machine learning model to classify the make and model of a vehicle.

Image recognition is a growing field that has exciting and practical applications. It can be used to classify all types of people, places and things. We want to create a model that helps accurately classify cars using bounding boxes and Convolution Neural Networks (CNNs). We will be training our model on the Stanford Cars dataset. This dataset provides high quality images making it optimal for our task. Our research aims to solve relevant problems ranging from providing accurate data by recognizing cars on the road, to injury prevention by car collision avoidance.

Introduction

Vehicle classification within the larger domain of image recognition has essential real-world applications. Efficient and accurate classification of car make and model can help in traffic monitoring, advanced driver-assistance systems, and collision prevention, all of which contribute to overall road safety. Our proposed approach will use CNNs on the Stanford Cars dataset, which offers high-resolution images and interpreted bounding boxes of numerous vehicle models.

Initially, to gain an understanding of the data and how we might properly build a model to best predict a vehicle make and model, we will train a simple neural network. While understanding image classification is quite complex and a simple model may not be able to achieve the intended effect or level of accuracy that we will strive for, we hope to use the experience as a stepping stone for a more well-tuned convolutional neural network (CNN). As our understanding of machine learning and neural networks deepens, we will adjust and tune the CNN appropriately to increase model performance and accuracy.

Beyond achieving optimal classification performance, our goal is to address potential biases in training data. By researching and experimenting with techniques such as data augmentation and

rebalancing, we hope to develop a model that is both fair and accurate. This project will help exhibit how meticulous dataset selection and model design can improve AI systems' reliability and eventually support safer transportation and more equitable applications of machine learning.

Related Works

Prior work has been done on vehicle classification which mainly focused on improving accuracy through deep convolutional neural networks and transfer learning. Some references include:

- **Krause et al. (2013)** introduced the Stanford Cars Dataset, highlighting its use in fine-grained classification.
- **He et al. (2016)** demonstrated that ResNet improves vehicle classification over traditional CNNs.
- **Dosovitskiy et al. (2020)** proposed Vision Transformers (ViTs) for image classification, outperforming CNNs in some tasks.
- **Buolamwini & Gebru (2018)** highlighted bias in computer vision datasets, showing higher error rates in underrepresented demographic groups.
- **Kim et al. (2021)** investigated fairness in object recognition, emphasizing dataset rebalancing as a solution.

We aim to explore strategies to reduce bias, such as data augmentation, rebalancing, and adversarial debiasing.

Dataset/Methodology

The data we have chosen to use is the Stanford Cars dataset (https://paperswithcode.com/dataset/stanford-cars). It consists of 16,185 images of cars that have been broken up into training and testing sets. The photos are taken from a variety of angles, and each differs in background and lighting. Furthermore, the images are given in .jpg format and have varying pixel sizes. The variability in the data will ensure the robustness of the model by forcing it to learn and adapt to many different situations and surroundings. The data was taken directly from the kaggle database and was stored in github for easy accessibility for the whole team.

The training dataset contains 8144 images, while the testing set contains 8041 images. Additionally, there are bounding boxes of the cars and the specific car make and model for each image in the training and testing set. A bounding box contains 4 different coordinates (two points), that are used to draw a rectangle around the perimeter of the car. Our model will use these bounding boxes to gain spatial understanding as well as to ensure the correct car is being classified. The visual below displays a bounding box around one of our images.

Fig 1. Bounding Box Visual

The dataset we are using is also well-balanced. There are a total of 196 different classes. The average class size is 82.58 images when combining the training and testing sets, and there is a standard deviation of 8.69 images. Additionally, we have a minimum class size of 48 and a maximum class size of 136. We can conclude that the dataset label frequency is well-balanced. When comparing the label frequency of the testing and training sets, it remains well-balanced. The maximum differential between the number of images in the training dataset and the number of images in the testing dataset is just 1. So, the characteristics of the training and testing datasets are very similar.

The data provided can be utilized in a variety of different ways with several approaches that can be used to build valuable tools. First, we could use the bounding boxes to detect cars in an image. This has significant use when considering Advanced Driver Assistance Systems and AI-driven traffic regulation. Cameras are a relatively cheap sensing equipment to put in vehicles, so it is important to be able to recognize where cars are in images to assist in emergency braking and collision avoidance. This model would output whether it detects a vehicle and its estimation of the bounding box that contains the vehicle.

Second, our trained model can be used to classify certain types of vehicles. Understanding the type of vehicle can bring about further insight into vehicle features. When combining a vehicle prediction from a neural network model with other information about the vehicle, more insight can be drawn that could assist nearby drivers, traffic monitoring cameras, and even law enforcement equipment. For example, a neural network could take an input of a vehicle image and return the probabilities of the make and model the car is most likely associated with, then additional data such as vehicle fuel economy (https://catalog.data.gov/dataset/vehicle-fuel-economy/resource/fd761199-110d-4318-a72e-0f875aa22ab5) could assist the vehicle type prediction and could further predict potential road damage and the amount of emissions that are coming from the use of a portion of road.

Initially, we constructed a simple neural network to experiment with model creation. Before model construction and training, data preprocessing steps needed to be taken to ensure the images were usable. Without proper preprocessing steps, the simple model would be unable to

correctly weight image features because of the varying sizes and pixel numbers across the dataset. Thus, the images are scaled down to the same size and quality. Next, we compute the Histogram of Oriented Gradients (HOGs) using the processed images. The HOG algorithm includes taking gradients of an image in the x and y dimension, computing a histogram of those gradients, and returning a flattened, normalized feature vector across. More info on how the HOG algorithm works can be found here: (https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html#). The flattened vectors returned from the HOG algorithm are then fed as input into the neural network. The process through which an image receives its predicted class is demonstrated below in figure 2.

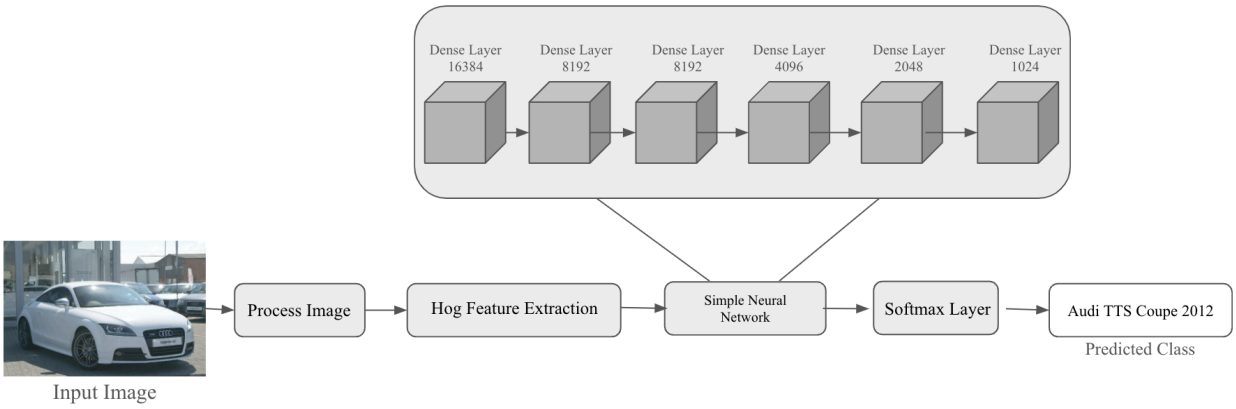

Figure 2. Simple Neural Network Diagram

We tested several different sizes of neural networks with varying layers and parameters in an attempt to increase the accuracy of the premature model. The model was finalized with 40716 input parameters, matching the number of parameters created after image preprocessing. In an attempt to increase the robustness of the model, several hidden layers were added. Each hidden layer was a dense layer with a Rectified Linear Unit (ReLU) activation function and batch normalization. The number of parameters per layer were 16384, 8192, 8192, 4096, 2048, and 1024 respectively. The final dense layer was an output layer with a softmax activation function, and a parameter for each unique class in the dataset. As a result, the model returned a vector with a softmax score that indicated the likelihood an image belonged to a given class. The process through which an image receives its predicted class is demonstrated below in figure 2.

Initial Experimental Results

Despite our best efforts, the initial implementation of the baseline model using a simple neural network architecture combined with HOG image preprocessing achieved limited performance, with a testing accuracy of just 0.49%. While convolutional neural networks are traditionally the model of choice for image classification, it is clear that our simplistic model was unable to handle the complexity of our dataset and problem. Such performance indicates substantial room

for improvement, prompting critical reflection on both preprocessing methods and model architecture.

Analysis of initial results suggested several key areas for improvement. The model exhibited low performance in all four basic metrics: precision, recall, accuracy, and F1 score. The use of HOG features was clearly insufficient for capturing the detailed characteristics necessary for accurate and precise vehicle classification. Thus, we transitioned to a CNN architecture known for its effectiveness in fine-grained image classification, the ResNet50. Moreover, integrating more in depth data augmentation and preprocessing techniques along with carefully adjusting the learning rate and other hyperparameters could provide additional performance gains.

Initial Improvements

As discussed above, there were few things that could be done to improve the current, simplistic model's attempts at image classification. However, transitioning to a more complex model with robust features gave hope for enhancement. First, the image preprocessing operations were revamped. Images were restricted and scaled within the bounding boxes given by the dataset so all that remained within each image was the respective vehicle. Because of the background padding space in each of the images, external or unintended features could be learned by the neural network that could vastly decrease the processing ability and accuracy of the model. Since those unnecessary features could only harm the model, removing the excess space by cropping each image was a logical step toward overall model improvement.

In addition to cropping each image, a change in HOGs calculation was implemented. By increasing the number of potential orientations calculated for each of the gradients, more information for each image could be calculated and fed into the model. Another option would be to adjust the number of pixels per cell and the number of cells per block. By shrinking the total size of each block, more information can be given to the neural network, the tradeoff being the cost of more training time and the total size of the neural network.

Moreover, as a replacement to our previous simple neural network, we chose to attempt to fine tune a pre-trained model that has the robustness for a sophisticated image classification problem. TensorFlow's pretrained image model, ResNet50, had the potential to significantly improve the accuracy for our vehicle prediction model. Tuning and adding to a pre-trained model for such a complex task like image classification would heavily increase chances of improvement.

As a result, we began to implement and tune a new ResNet50 model. The model takes in an input of a 224x224x3 RBF image. As discussed above, the image has been scaled within its vehicle bounding box and resized to fit ResNet50 parameters. In order to retain learned visual features, backbone freezing was implemented within the first 140 layers of ResNet50. Few modifications were made to the internal layers of TensorFlow's pre-trained model. Figure 3

depicts the inner workings of the model, the output prediction returned, as well as the preprocessing steps.

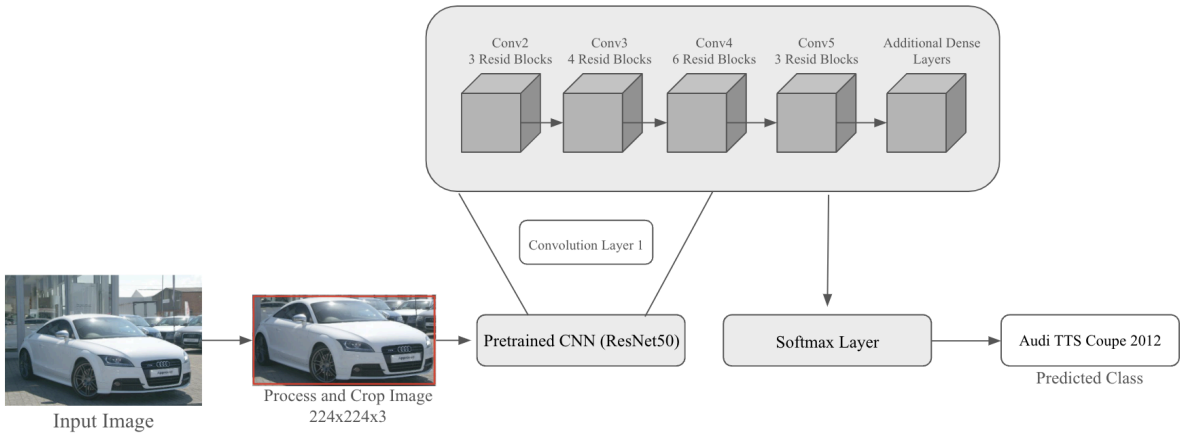

Figure 3. ResNet50 Diagram

As seen above in Figure 3, there is an initial convolution layer and 4 other convolution layers with varying levels of residual blocks. Convolution layer 1 contains a ReLU activation function, batch normalization, and a max pooling layer. This initial layer is designed to extract the most prevalent features from the image. Edges, shape, and contours are examples of basic features that when captured early, can be processed in much larger depth. These features are condensed by the max pooling layer to reduce computational cost as the output is passed to the next layers. The early layers of ResNet50 learn general features in order to produce highly effective, fine-tuned predictions.

After initial feature extraction, convolution layers 2 through 5 contain 3, 4, 6, and 3 residual bottleneck blocks respectively. These blocks are designed to decrease computational expenses while increasing the depth and complexity of the model. ResNet50 has 50 layers in total, many of which are a result of these residual blocks. Each convolution layer also contains batch normalization and ReLU activation to ensure consistency in the output that is passed between each layer. Downsampling techniques are also utilized for each internal layer as a method of reducing spatial dimensionality and extracting higher level features than previously attained. All of these attributes ensure that high quality features are as essential as possible when passed to the output layer.

Similarly to the simple neural network, a softmax activation function was utilized within the dense output layer. Additionally, an average global pooling layer was added along with a dropout layer with a rate of 0.2 was added to decrease the likelihood of overfitting. These customizations resulted in the model returning the same output as our prior model, a vector with percentage predictions corresponding to the number of car classes.

Final Experiment Results

Following the implementation of a simple dense neural network model utilizing Histogram of Oriented Gradients (HOG) features, which only achieved 0.49% accuracy, we significantly restructured our approach by leveraging transfer learning through a ResNet50-based Convolutional Neural Network (CNN). This enhanced model uses the Stanford Cars dataset images, cropped with bounding boxes and preprocessed via ResNet-specific normalization, along with extensive data augmentation, including random flips, rotations, and zooming.

We trained the model for an initial 10 epochs, achieving 81.37% validation accuracy. Encouraged by this performance, we continued training for an additional 20 epochs, with the model steadily improving and ultimately reaching a validation accuracy of 84.44% and a final training accuracy of 98.60% at epoch 30. The best performance was obtained using Adam optimizer with a learning rate of 1e-4, sparse_categorical_crossentropy loss, and a batch size of 8.

To further assess the model, we evaluated it on the test set (8041 images). The final test accuracy was 83%, with a macro-averaged F1-score of 0.83, demonstrating significant improvements across nearly all classes compared to the initial dense model. The model also achieved high precision and recall values across most of the 196 vehicle classes, including several near-perfect F1-scores (for e.g., Class 182: F1 = 0.98, Class 141: F1 = 0.99).

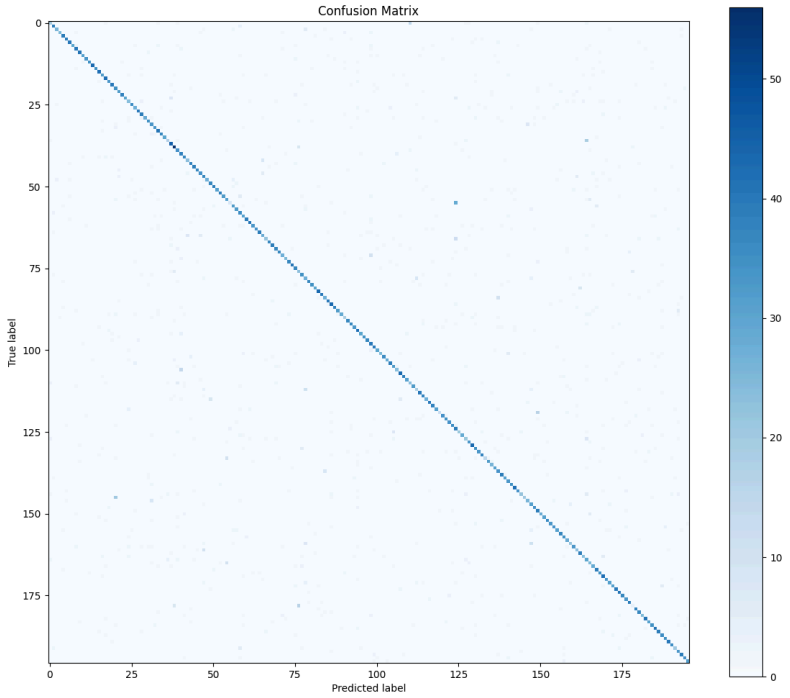

Figure 4. Confusion Matrix

Due to the large number of classes, the confusion matrix in Figure 4 is not a very effective visualization of the results. However, it is able to demonstrate the increased levels of accuracy. As seen in the heatmap, darker blue colors indicate strong F1 scores. Since the darkest blue square across the predicted label typically falls in line with the same true label on the y axis, we can see high levels of accuracy in our predictions.

A few classes still presented challenges, especially those with high visual similarity or fewer training samples (for e.g., Class 178: F1 = 0.24, Class 191: F1 = 0.63), suggesting a potential need for class-specific rebalancing or attention-based mechanisms in future work.

Summary of Key Metrics:

	Metric	**Value**
1	Training Accuracy	98.60%
2	Validation Accuracy	84.44%
3	Test Accuracy	83.00%
4	Macro F1-Score	0.83
5	Number of Classes	196

Table 1: Model Results containing performance metrics and their values

This ResNet-based approach significantly outperformed the earlier baseline model and lays a strong foundation for more advanced architectures, such as Vision Transformers (ViTs) or EfficientNet, and for incorporating bias mitigation techniques in subsequent research phases.

Future Work

From our results we can see that the training accuracy is around 14% higher than the validation accuracy. These results could indicate our model is experiencing overfitting, and there is room for improvement when it comes to generalizing well with new images.

A solution could include adding more randomization to our data augmentation pipeline. Currently, our pipeline randomly flips and rotates our images to introduce some variance. To improve the robustness of our model we could introduce contrast and brightness into randomly specified images. This could help our model generalize better for pictures with different real-world lighting conditions.

Another way to improve our model is to freeze more layers from our ResNet50 base model. The lower layers of this model capture general features of images. Our current implementation freezes the first 140 layers. By freezing more layers we can force the model to rely on more general features in these frozen layers rather than memorizing the training data.

Experimentation with different forms of regularization could also positively affect the generalization of our model. L2 regularization could help reduce the complexity of our model by shrinking our weights.

Conclusion

We proposed a car classification model using ResNet50 as our base model and Stanford Cars for training purposes. We achieved a validation accuracy of 84.44% for over 8000 images. We experienced noteworthy performances in classes with F1-scores of 0.98 and 0.99 for Class 182 and 141.

Our model handles complex features from visual images in order to accurately classify car images. We introduced techniques to ensure our model generalizes to real world scenarios. These results could potentially contribute to road safety and other relevant automobile injury prevention efforts.

While our model performed well overall, there is still room for improvement for classes with high visual similarity such as Class 178 and and 191. Efforts to address these issues in class imbalance strengthen our accuracy. To improve these limitations going forward, we can further improve our data augmentation, regularization and cross validation techniques.