Matthew Caro, Kevin Huang, Ankith Nagabandi, Yarid Tyran

12/2/2024

CSE 5717 Team 1

Project Report

Vehicle-Type Identification Project

Problem Statement

The goal of this project is to train and develop a model capable of identifying a vehicle's body type from an image. The model aims to classify a given vehicle image into one of the following 6 categories: SUV, 4-door, 2-door, van, pickup, and convertible. As a result, this project falls under a classification-based project. The purpose for this project's creation is in its potential practical applications.

Consumers, automotive enthusiasts, and even law enforcement agencies with limited knowledge about vehicle types can utilize this model to learn and understand more about the classification of vehicles and their types. Additionally, in cases where details about a vehicle are limited, such as blurry surveillance images or incomplete descriptions, this model can provide a reliable and automated assistance in narrowing down the type of vehicle.

This project utilizes specific performance metrics such as classification accuracy, precision, validation accuracy, loss, etc., to help evaluate the effectiveness of the trained model.

Data Description/Dataset Preparation

Figure 1: Sample Bad or Uninformative Data

The dataset used for this project contains around 60,000 images. It included a vast range of images, of all types of vehicles, models and makes, including different years as well. However, not all these images were high quality and proper images that could be understood. They varied from whole body images to interior images to largely zoomed in images showcasing one part of the car. To clean this as far as we can, we utilized a combination of scripts and manual cleaning.

To start, we ran scripts to normalize the data based off quality and then more scripts to further filter images. However, even after post processing and cleaning, there was still some bad or unrelated data that shouldn't be used (like those shown in Figure 1), but we cleaned it as far as we could without manually going through the whole dataset of over 60,000 images and deleting bad images. A technique we used in processing was using a pretrained model to determine whether an image is relevant to our project. This involved giving this pretrained model a prompt to follow that would determine whether this image was of the exterior or the interior and removing the irrelevant images. At the same time, we also removed duplicate images from the dataset. This helps with balancing the data and only using unique images in the dataset.

Figure 2: Sample images from the dataset

Furthermore, the dataset was not sorted or organized, but they were labeled in extensive detail. As shown in Figure 2 above, the naming scheme for the images started with the make of the car, followed by the model, year, and other information. This information was separated by an underscore. The ending of the names were the type of body and a unique identifier as there were many images of the same vehicle taken at different angles. To sort them, we wrote a script to read all the files, split the name by the underscores and took the second to last element in the resulting array to find the class. Then, we directly change the directory of the image to their respective class folder.

After sorting the dataset into their respective classes, we split the data into the training, testing, and validation sets. We wrote a script that would randomly put the images into the folders based on the split we chose- 70% train and 30% test. And with the data split up into training and test data, we are ready to talk about the models we chose to work with.

Model Selection

We explored multiple different options when selecting our model. As it was an image classification problem, we dove deeper into researching specific techniques in CNNs. We had decided to have a baseline and then compare the other models beyond that for the image classification problem. From our individual research, we concluded that there were 2 models that consistently performed the best in image classification: ResNet50 and MobileNet. Thus, we took these models, along with a baseline model (Sequential) and compared their performances on our training dataset. From there, we can choose the best performing model to move forward with in our project and work on fine-tuning and training multiple iterations of that model to extract the maximum and best possible performance.

ResNet50 is a deep convolutional neural network (CNN), which is an architecture belonging to the ResNet family. The main architectural style is within Deep Networks. It involves the concept of residual learning, where there are different mapping identities, which are added to the layers of the network. The “50” means that there are 50 layers to be able to extract individual features. As it's mainly used in feature extraction, it's a popular option in computer vision problems, such as Image Class and Obj. Detection. In our application, we saw that it has computational efficiency, a reason why we had chosen it as one of the options for our project.

MobileNet is a lightweight CNN architecture designed for mobile and embedded devices, where computational power and memory are constrained. It uses depth wise separable convolutions to significantly reduce the number of parameters and computational cost while still achieving competitive performance on tasks like image classification and object detection. MobileNet is particularly effective for real-time computer vision applications due to its

efficiency. We chose it, as it fits perfectly into the computer vision, image classification task that we are looking at.

The Sequential model is a high-level abstraction provided by frameworks like TensorFlow and Keras for building deep learning models, including CNNs. It allows for creating a linear stack of layers, where each layer is sequentially connected to the next. This simplicity benefits prototyping CNN architectures used in computer vision tasks, such as image classification. Usually, Sequential models are often used in smaller, straightforward applications where CNNs are employed for tasks like digit recognition or basic object detection. We decided to use this for our case, because it's a good baseline for CNNs and classification problems, based on our research.

Model Techniques

In the first iteration of training ResNet50, the goal was to create a baseline for vehicle classification. The model utilized a ResNet50 base pretrained on ImageNet and had only the last 10 layers unfrozen to enable fine-tuning. This strategy used the pretrained knowledge of ResNet50 while limiting the computational cost of training by freezing most of the layers. The model also had multiple layers such as a global average pooling layer, a dense layer with 256 neurons and ReLU activation, a dropout layer with a rate of 0.5 to prevent overfitting, and a final SoftMax layer to output probabilities for the 5 vehicle classes. To handle the imbalance of classes within the dataset, class weights were computed using scikit-learn's compute_class_weight function and applied it during training. Data augmentation techniques such as rotation, width and height shifts, and horizontal flipping were also applied to improve model generalization. The training was conducted with a learning rate of 1e-5, and the Adam optimizer was used to update model weights. Key callbacks included ModelCheckpoint to save

progress occasionally, EarlyStopping to stop training if validation loss plateaued, and ReduceLROnPlateau to dynamically adjust the learning rate based on validation loss. This iteration provided a baseline model and helped identify areas where improvements could be made in the next iterations.

In the second iteration of training ResNet50, we introduced several changes to improve the model's performance. First, the learning rate was increased from 1e-5 to 1e-4, and the optimizer was changed from Adam to Stochastic Gradient Descent (SGD) with a momentum of 0.9. This adjustment aimed to increase stability and utilize SGD's ability to converge more effectively. Additionally, 50 layers of the ResNet50 architecture were unfrozen to allow fine-tuning of a larger portion of the model, enabling it to learn more task-specific features for vehicle classification. Additional data augmentation techniques were introduced such as brightness adjustments, shear transformations, and channel shifts, along with the rotation, width, and height shifts used in the first iteration. This helped the model generalize better to varied real-world scenarios. Batch normalization layers were also added to improve training stability and convergence. Usage of class weights and callbacks remained the same except the patience for ReduceLROnPlateau was reduced to allow learning rate reductions more aggressively. The number of epochs was also increased to 100 to provide the model with more training time to implement these changes effectively. These enhancements collectively resulted in a more refined model with improved accuracy and generalization compared to the baseline established in the first iteration.

In the third iteration, the focus was on improving regularization techniques and fine-tuning specific layers to enhance model performance. We unfroze the last 15 layers of ResNet50 to fine-tune more task-relevant features while keeping the earlier layers frozen to retain the

pretrained knowledge. A significant change we made was the addition of an L2 regularization term in the dense layer to reduce overfitting. This effect was further enhanced by using batch normalization layers to stabilize training and improve convergence. The data augmentation techniques were adjusted slightly to balance diversity and efficiency but largely remained the same. Unlike earlier versions, however, this iteration introduced stricter early stopping criteria, combined with a ReduceLROnPlateau callback that even more aggressively reduced the learning rate after three epochs of plateaued validation loss to achieve even faster convergence. We reintroduced the Adam optimizer but with a lower learning rate of 1e-5 and used it to maintain stable training dynamics. The overall architecture remained similar to the second iteration, with a global average pooling layer, a dense layer with ReLU activation, a dropout layer with a rate of 0.5, and a SoftMax output layer for classification.

The final iteration of ResNet50 training focused on optimizing training strategies and refining model performance for vehicle classification. This iteration unfreezes the last 50 layers of the ResNet50 architecture, enabling deeper fine-tuning to capture more nuanced features in the dataset. A structured learning rate scheduler was introduced to adjust the learning rate dynamically during training, with a decay rate of 0.1 applied every 30 epochs. This adjustment balanced stability and convergence to the best degree, ensuring consistent improvements in validation loss. Regularization methods were further enhanced by adding L2 penalties to the dense layer, preventing overfitting, especially in the highly trainable layers. Batch normalization layers were still utilized to maintain stable gradients and faster convergence, while dropout layers with a rate of 0.5 were kept to further combat overfitting. The architecture remained very similar but with some slight modifications. It now consisted of the global average pooling layer, a dense layer with 256 neurons, and a SoftMax output layer for class probabilities. Class weights

also continued to play a pivotal role in handling dataset imbalance. The Adam optimizer with a learning rate of 1e-5 was still kept from the previous iteration and allowed for smaller and more controlled weight updates. Callbacks like ModelCheckpoint, EarlyStopping, and LearningRateScheduler were still used to save the best model, stop training when no further improvement was observed, and dynamically adjust learning rates.

Evaluation

The dataset of vehicle images was split into 70% for training and 30% for testing and evaluation. This split ensured that the model had a substantial portion of data to learn from while reserving enough data to thoroughly test its performance and generalization capabilities on unseen images. The training set was used to optimize the model's parameters through backpropagation, while the test set was used exclusively for evaluation purposes to simulate real-world performance.

The 30% test set was important for generating key metrics such as accuracy, loss, validation accuracy, and validation loss. These metrics provided valuable insights into a model's performance, capturing both its ability to learn patterns during training and its capacity to generalize those patterns to unseen data. More specifically, the training accuracy and loss represented the model's performance on the data it was exposed to during training, while the validation accuracy and loss reflected its ability to generalize during the training process itself. By using a separate test set, we ensured that the final evaluation metrics were unbiased and independent of the data used to optimize the model.

Metrics like test accuracy and test loss served as important indicators of the model's effectiveness. Accuracy measured the percentage of correctly classified vehicle types, while loss showcased the predictive error on the dataset. These metrics provided a clear assessment of a model's generalization and classification capabilities. Over the course of the different iterations of training ResNet50, accuracy steadily improved, with the final iteration achieving a test accuracy of around 80%.

Unfortunately, this number is a lot lower than what we had hoped, most likely due to a few main reasons which will be explained later. However, the steady increase in this number through every iteration highlights the success and effectiveness of our training strategies, where each iteration ultimately did refine the model's ability to classify vehicle images across six categories: Van, SUV, Pickup, Convertible, 4-door, and 2-door. Similarly, test loss values consistently decreased across iterations, demonstrating the model's improved ability to learn meaningful patterns while avoiding overfitting. This balanced improvement in both accuracy and loss reflected a robust and reliable training methodology.

Additionally, we also utilized confusion matrices for trained models, which played an important role in the evaluation process and ultimately choosing our best model. It detailed the individual model's performance for each vehicle class and showed where classification errors occurred. This analysis provided insights into which classes were more challenging for the model and indicated areas for further refinement.

Evaluation (Graphs and Figures)

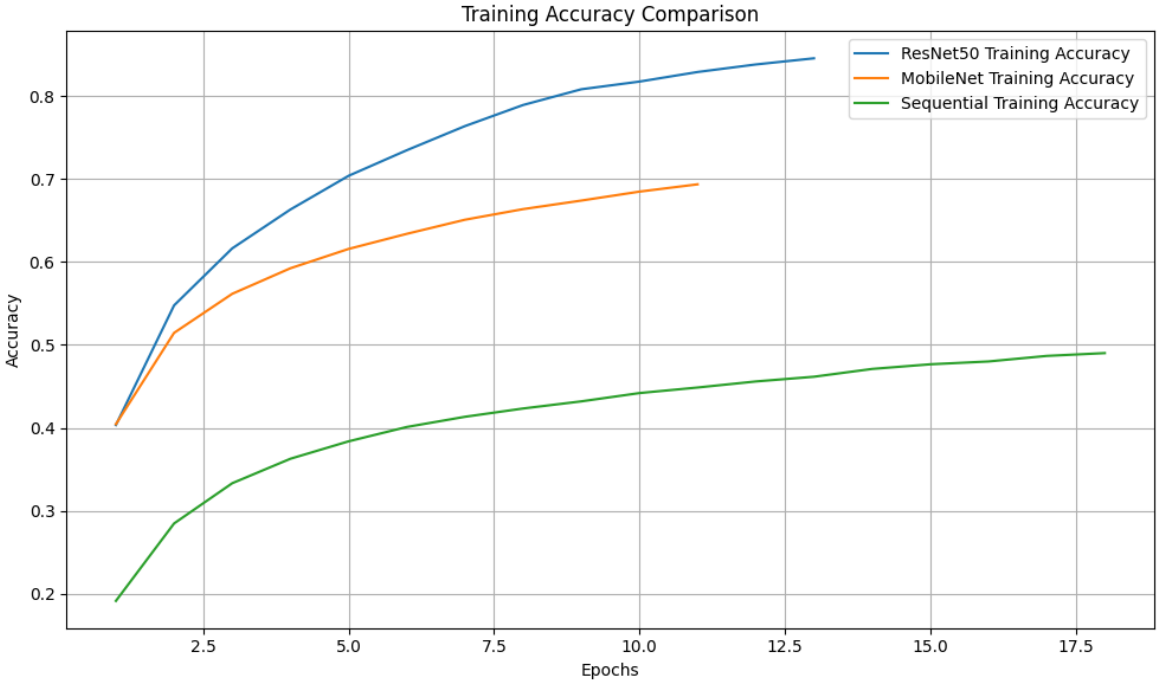

Figure 3 - Training Accuracy of All 3 Models

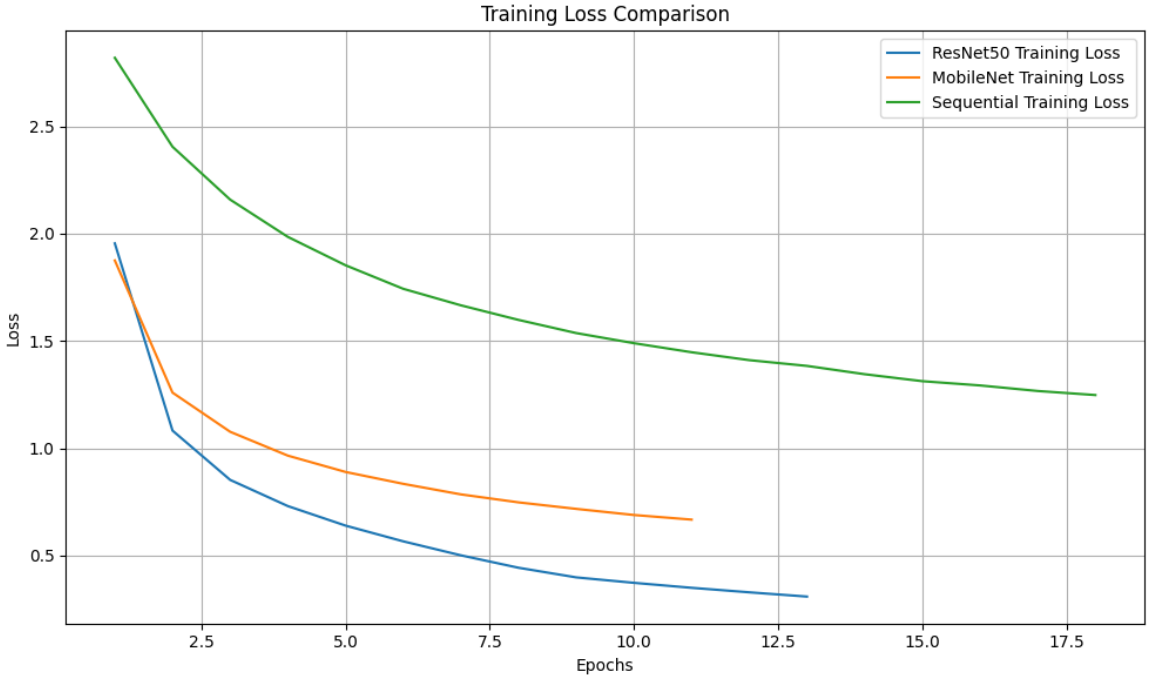

Figure 4 – Training Loss of All 3 Models

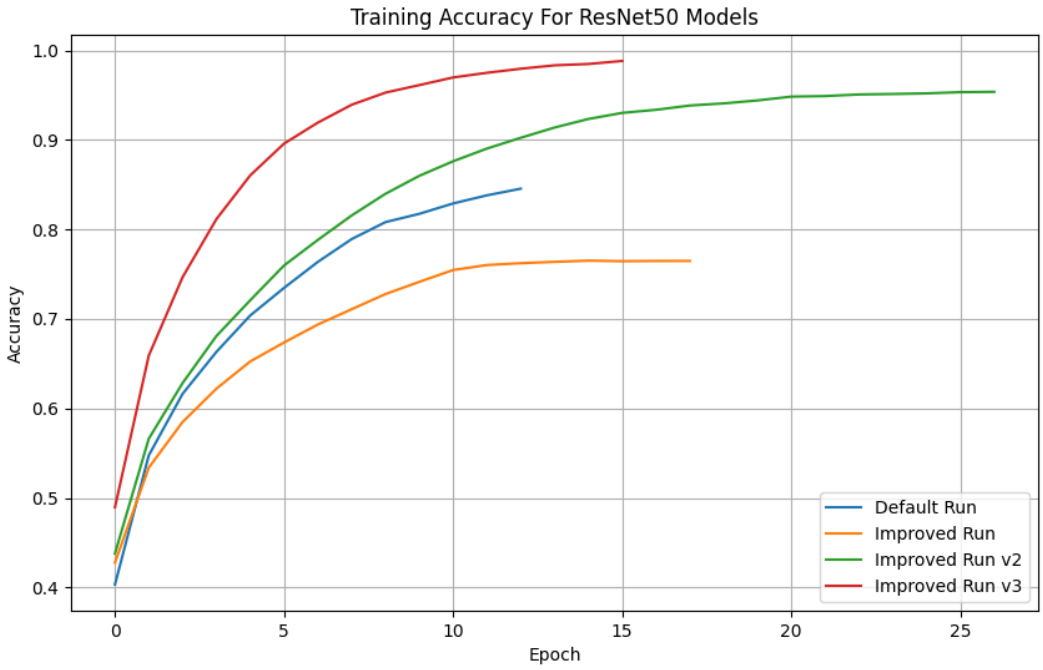

Figure 5 – Training Accuracy for all ResNet50 Model Iterations

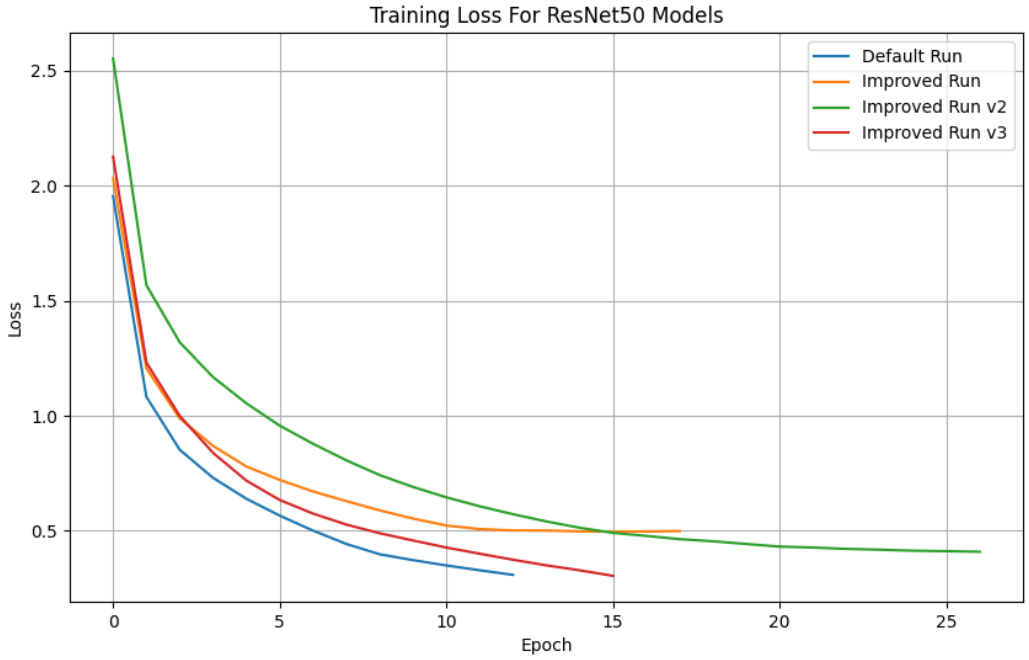

Figure 6 – Training Loss for all ResNet50 Model Iterations

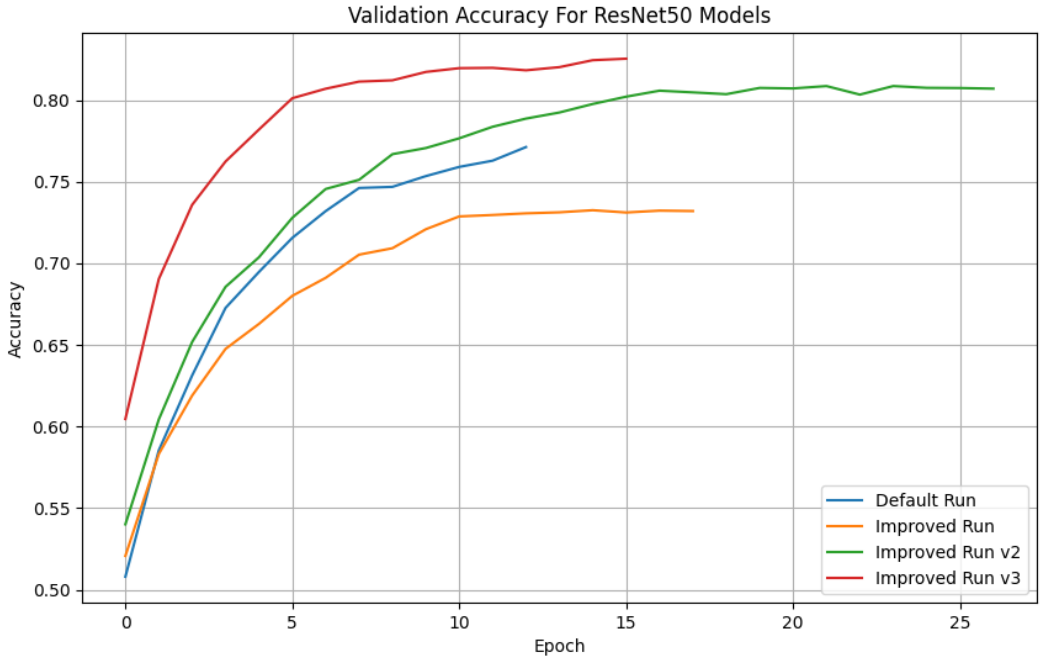

Figure 7 – Validation Accuracy for all ResNet50 Model Iterations

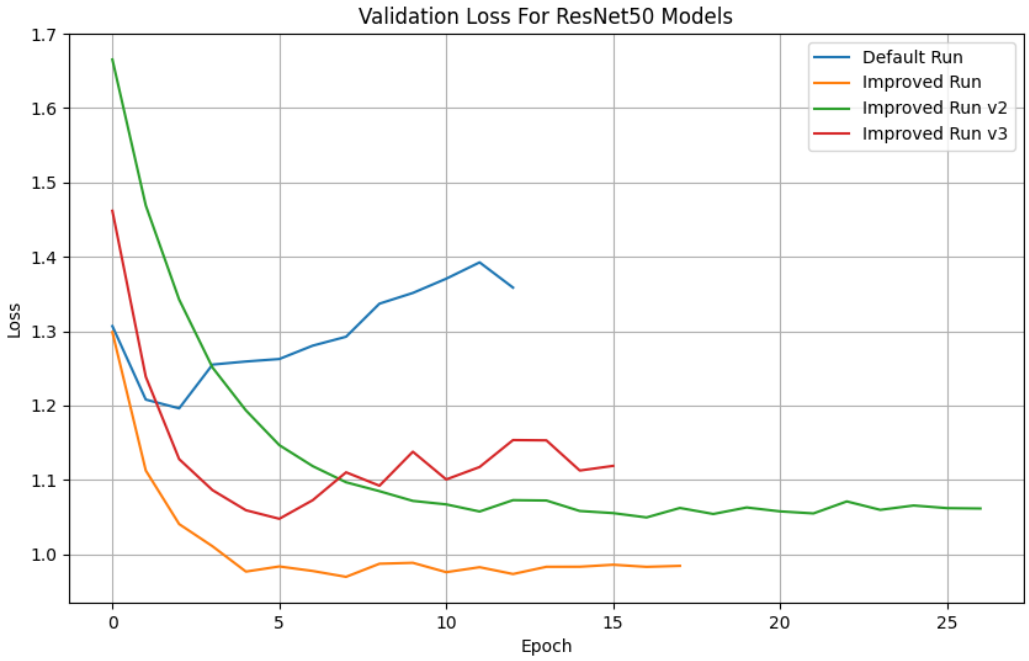

Figure 8 – Validation Loss for all ResNet50 Model Iterations

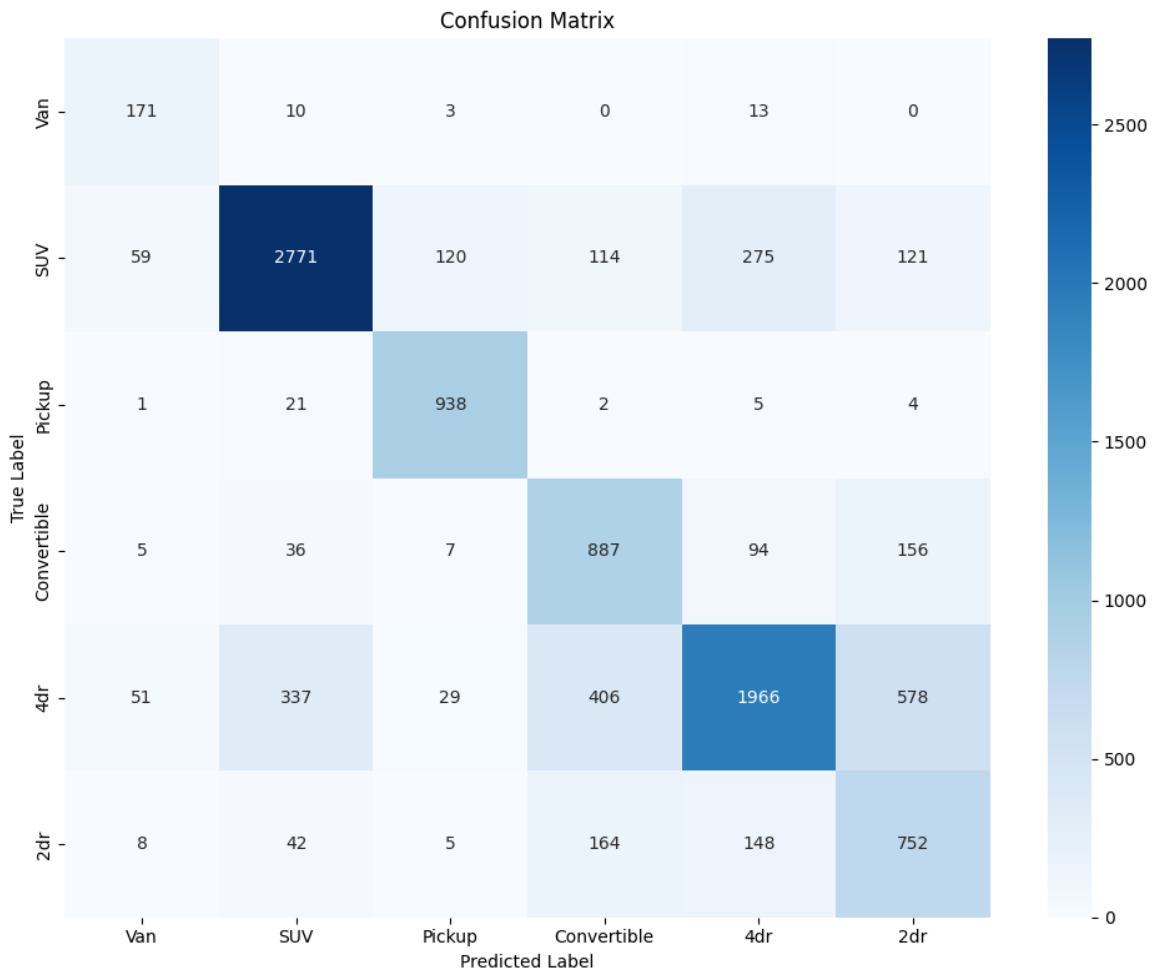

Figure 9 – Confusion Matrix for Default ResNet50 Iteration

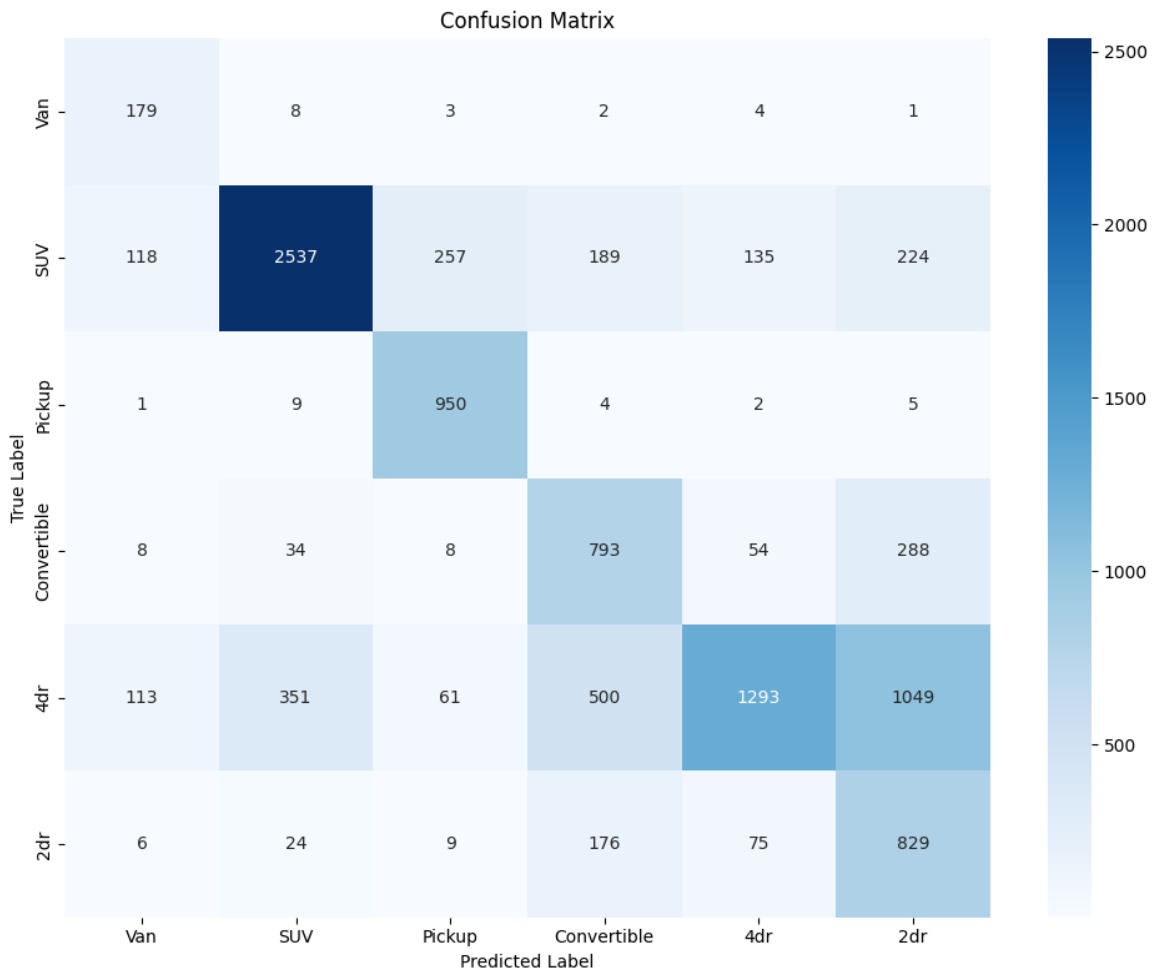

Figure 10 – Confusion Matrix for Improved V1 ResNet50 Iteration on Testing Set

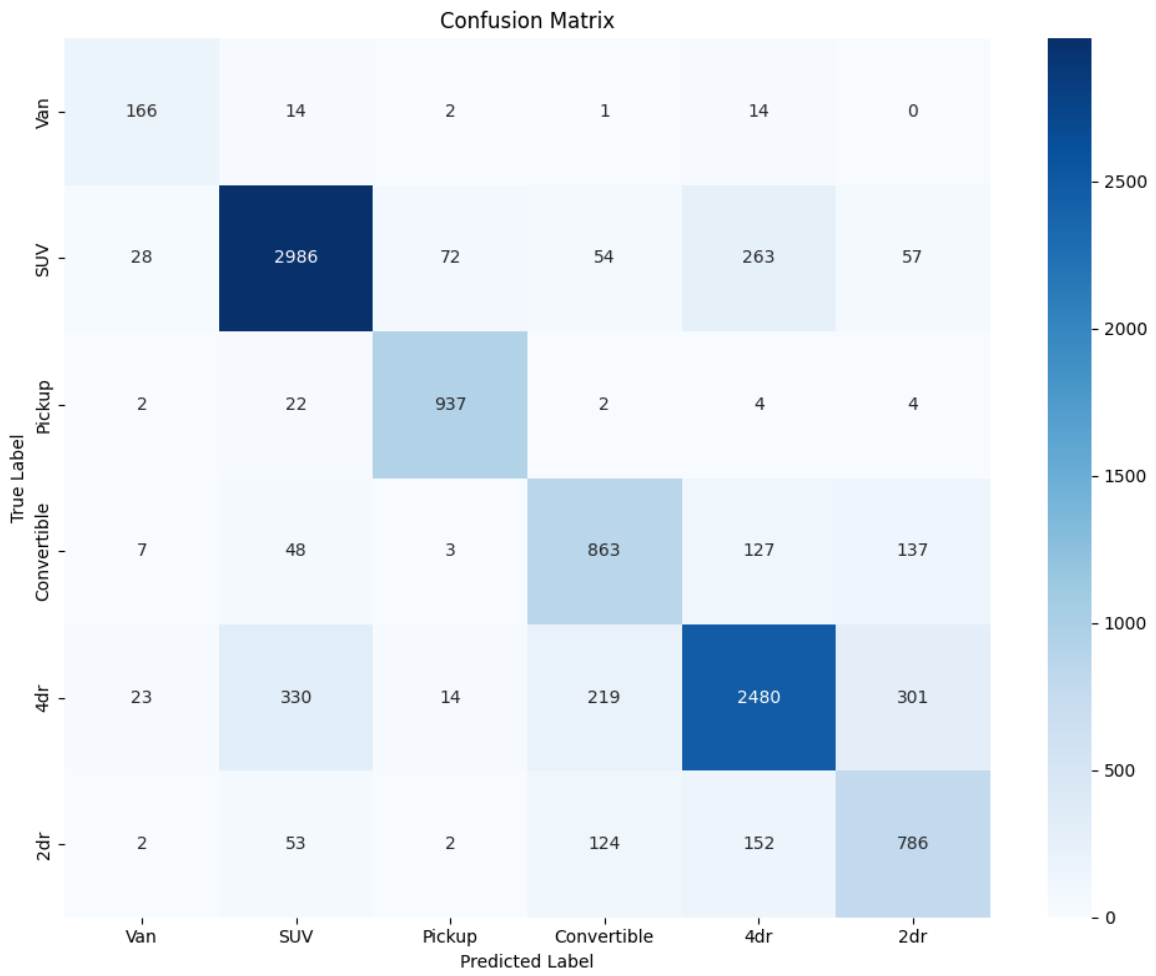

Figure 11 – Confusion Matrix for ResNet50 Iteration V2 on Testing Set

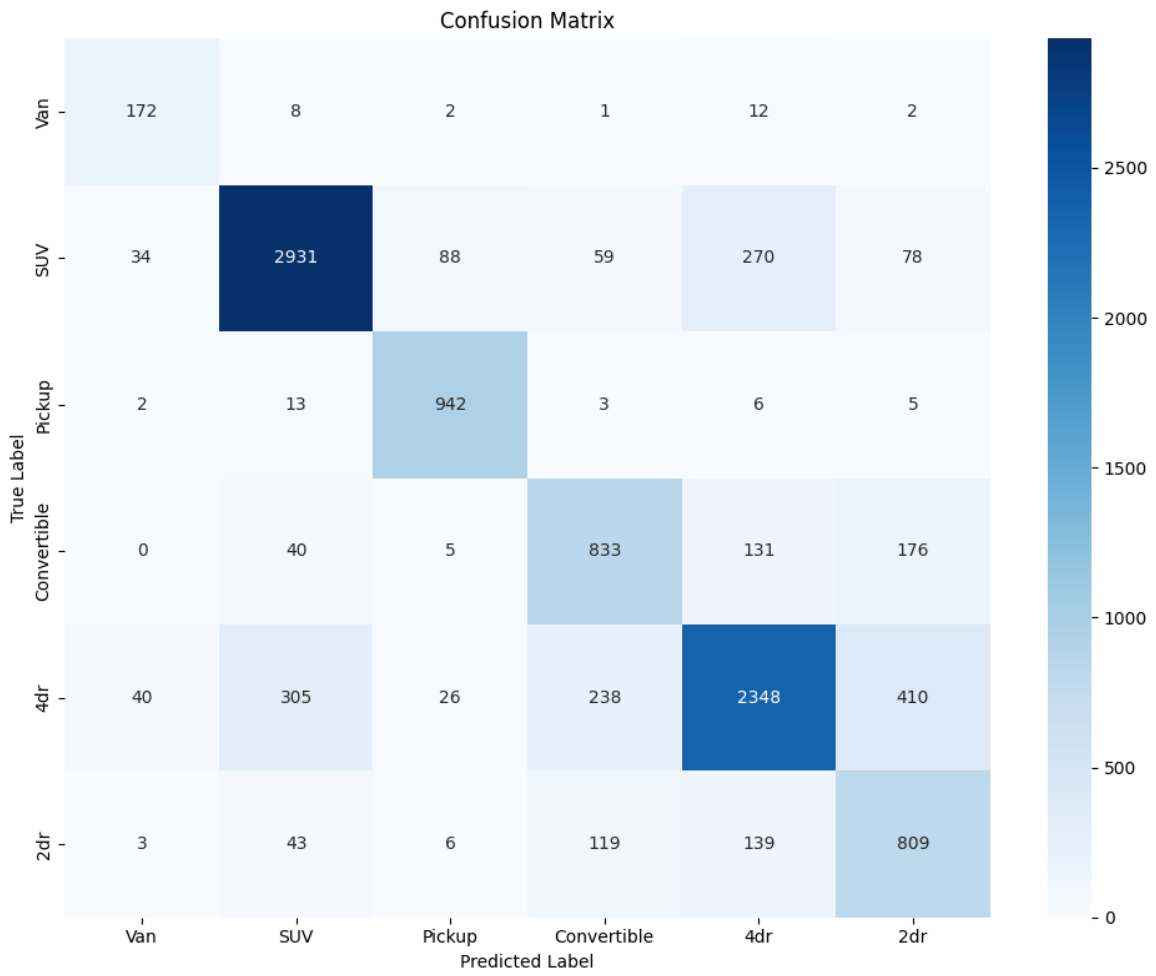

Figure 12 – Confusion Matrix for ResNet50 Iteration V3 on Testing Set

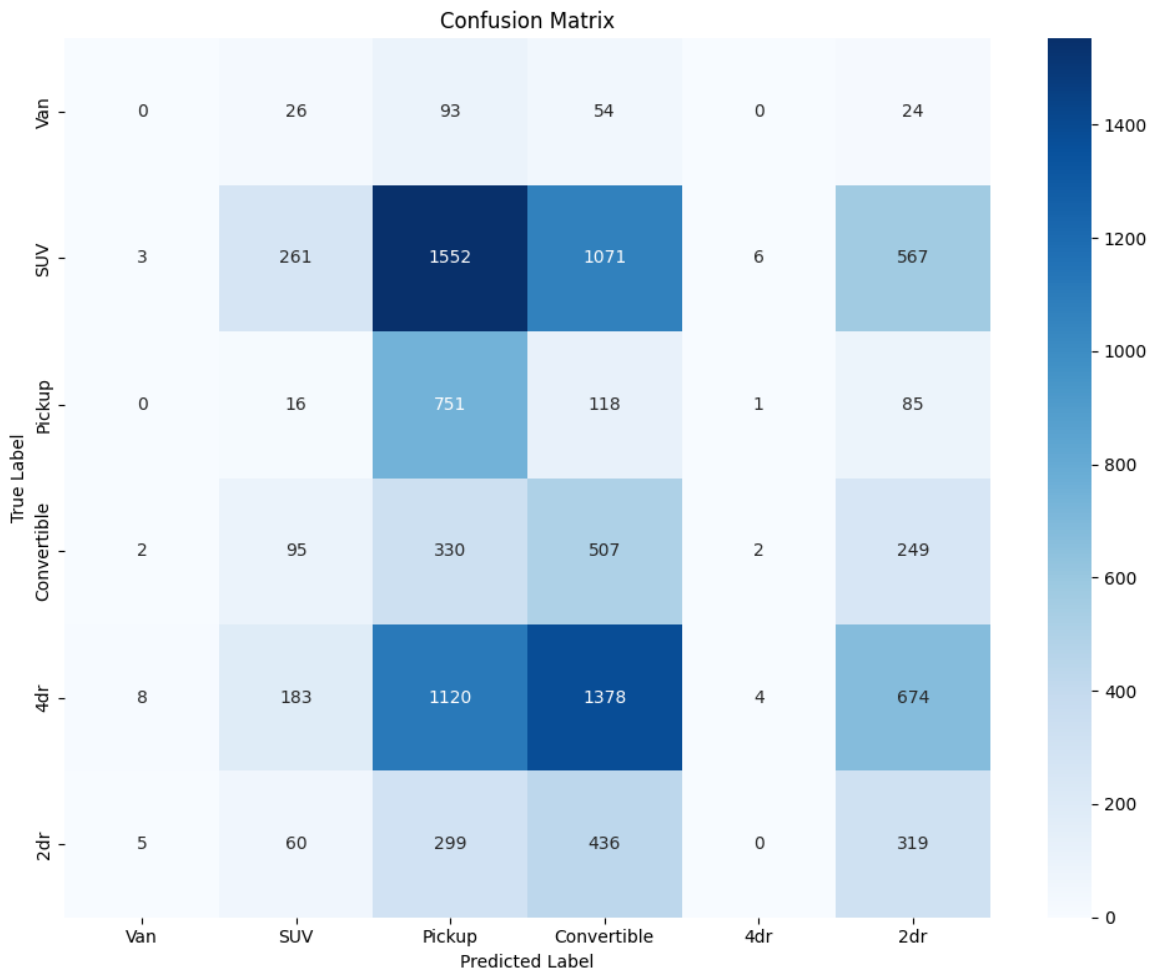

Figure 13 – Confusion Matrix for Sequential Model on Testing Set

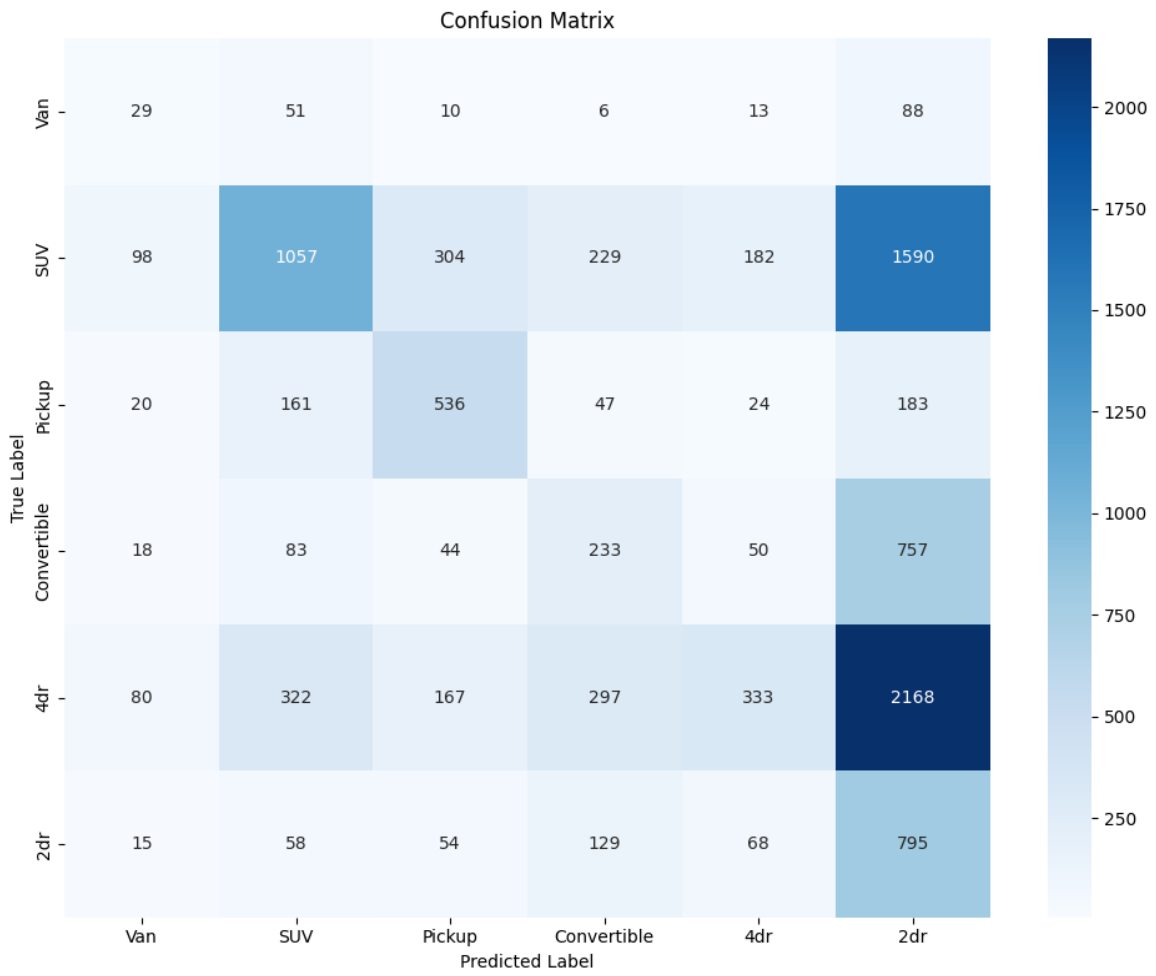

Figure 14 – Confusion Matrix for MobileNet Model on Testing Set

Error Analysis

The error analysis of the different models and iterations revealed important information about limitations and challenges faced. One such challenge was the misclassification of certain vehicle types, particularly between similar categories like "SUV" and "Van" or "2-door" and "4-door" cars. These errors can be attributed to overlapping features such as size, shape, and design, which made differentiation difficult. Confusion matrices proved the frequent misclassifications in these specific categories, reflecting the challenge for each model to distinguish these visual cues. This

limitation highlights the need for more fine-grained features or additional pretraining on datasets specifically tailored to vehicle classification.

Another source of error came from the dataset itself, which, despite extensive cleaning efforts, still contained inconsistencies and error images. The "Van" class was particularly underrepresented, with significantly fewer samples compared to the dominant classes like "SUV" and "4-door," which had approximately 8,000 training images each compared to the Van's 500. Meanwhile, classes such as "Convertible," "Pickup," and "2-door" fell into a middle range, with datasets varying between 2,000 and 4,000 images. This large disparity between the classes made it challenging for the model to learn balanced representations across all categories, even with the utilization of class weights to address the imbalance.

Additionally, noisy data, such as zoomed-in images of tires or other car parts, partial views, and even interior images irrelevant to the task, further hindered classification accuracy. These error images, along with the imbalance in class distributions, likely contributed to the misclassifications seen in the confusion matrices. While our automated cleaning scripts and filtering helped mitigate some of these issues, the size of the dataset made it impractical to eliminate all problematic images. As a result, the model's final test accuracy on the test dataset capped at around 80%, falling short of expectations and emphasizing the need for a more refined and balanced dataset in the future.

The iterative approach and strategy for training the ResNet50 model helped address some errors, but challenges continued to persist. Early iterations showed signs of overfitting, particularly seen in the discrepancies between training and validation accuracy. While regularization techniques such as dropout layers and L2 penalties were introduced in later iterations to address this issue, the initial overfitting further emphasizes the need for a larger and

more diverse dataset. Additionally, while more fine-tuning in iterations v2 and v3 improved overall metrics, version 2 outperformed or matched version 3 on certain classes, indicating that the later improvements were not fully beneficial across all categories. This suggests a trade-off between generalization and specificity that requires further optimization and fine-tune adjustments.

Conclusion and Areas for Improvement

From the beginning of this project, we all had expected that a convolutional neural network (CNN) would give us the best opportunity to achieve our goal and make accurate a model to correctly identify different types of vehicles. Early on we started doing research into different CNN models and which we believed would be best as well as what the differences were between the models. This allowed us to be ahead of the curve once we were formally introduced to neural networks in lecture allowing us to quickly progress and build a quality vehicle identification model.

Even though we did make a model that we feel is indicative of our extensive knowledge and abilities in the field of Big Data, that does not mean that there isn't space to improve. The main areas that we feel could be better implemented or built upon include the dataset size, cleaning, and the implementation of more models.

When initially choosing our dataset, we had chosen it because of the extensive labeling as well as the belief that it would be enough data for our project. While it is certainly a decent amount of data, in the world of big data and neural networks this is just a drop in the bucket and if we had a similar dataset but with hundreds of thousands or millions of images it would

certainly help. This is because unlike most traditional machine learning algorithms, neural networks do not stagnate in accuracy once hitting a certain amount of data. Instead, neural networks continually gain accuracy the more data they are given so if we had a dataset of a million labeled images, we could likely grow our accuracy by a wide margin. More data would also allow for better training of the different vehicle types too, ensuring that classes like van were not limited to 500 only 500 images. The only major issue that this would bring would be training time but ultimately this would be a worthwhile trade off.

As stated earlier in this paper, we went through extensive efforts to clean the dataset. Despite this there were still inconsistencies and bad images that were able to make it through the cracks. To solve this issue, we could research even more methods of cleaning, but this has no guarantee of 100% accuracy in removing bad images. The only way to guarantee a clean data set would be to manually clean all of the bad images from the dataset but under the time constraints of this project it was unrealistic to do this.

The last logical improvement we could make is trying more models. While we did do research as to what models would likely work best this doesn't mean that they are going to be best. Simply speaking, there is a very good chance that there is something else out there that we simply do not know of or had overlooked that would perform just as well if not better as ResNet50. If we choose to continue working on this project following the end of the class this will likely be what we do in order to continue learning and hope to surpass our current model.

Finally, moving forward we have talked about continuing to work on this project as it is good experience and there are many ways that we could continue to build. The first way and likely the first thing we will do is use it as a test set to learn and use more machine learning algorithms. There is also the idea of adding more features to the predictive nature such as having

it predict a car’s year, brands and more based on the extensive labeling. In the end, we are satisfied with the results of the project and look forward to improving on it in the future.