FINE-GRAIN VEHICLE CLASSIFICATION

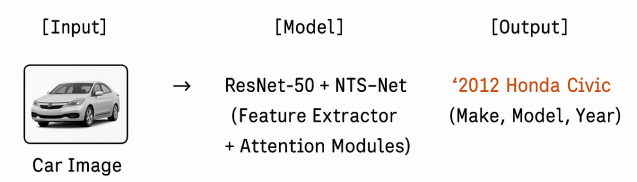
Emily Ye
Student# 1010975526
emilyyj.ye@mail.utoronto.ca

Carole Yang
Student# 1011042346
carole.yang@mail.utoronto.ca

David Yang
Student# 1011088742
davidhengyi.yang@mail.utoronto.ca

Muhammad Arsalan
Student# 1010970549
m.arsalan@mail.utoronto.ca

1 INTRODUCTION



The purpose of this project is to develop a system that can take an image of a car and accurately classify its make, model and release year. This system is intended to support public safety and security by leveraging technology and deep learning. In situations where law enforcement or other agencies need to quickly identify a vehicle—such as during investigations or surveillance—an automated, reliable tool could be highly valuable.

This goal is important not only for improving response times and investigative accuracy but also for contributing to broader efforts to build safer communities. Additionally, as a group of individuals with a shared interest in cars, we were naturally drawn to a project that combines our personal interests with meaningful societal impact.

Deep learning is a suitable approach for this problem because of its ability to extract and learn complex, fine-grained visual features. Our implementation uses a ResNet-50 backbone for feature extraction, combined with NTS-Net's navigator–teacher–scrutinizer framework to focus on the most distinctive regions of each image. This design is particularly effective for distinguishing subtle differences between car models and production years. By using this hybrid architecture along with targeted data augmentation, we aim to achieve higher classification accuracy and demonstrate the effectiveness of combining these methods for fine-grained vehicle recognition.

2 ILLUSTRATION/FIGURE

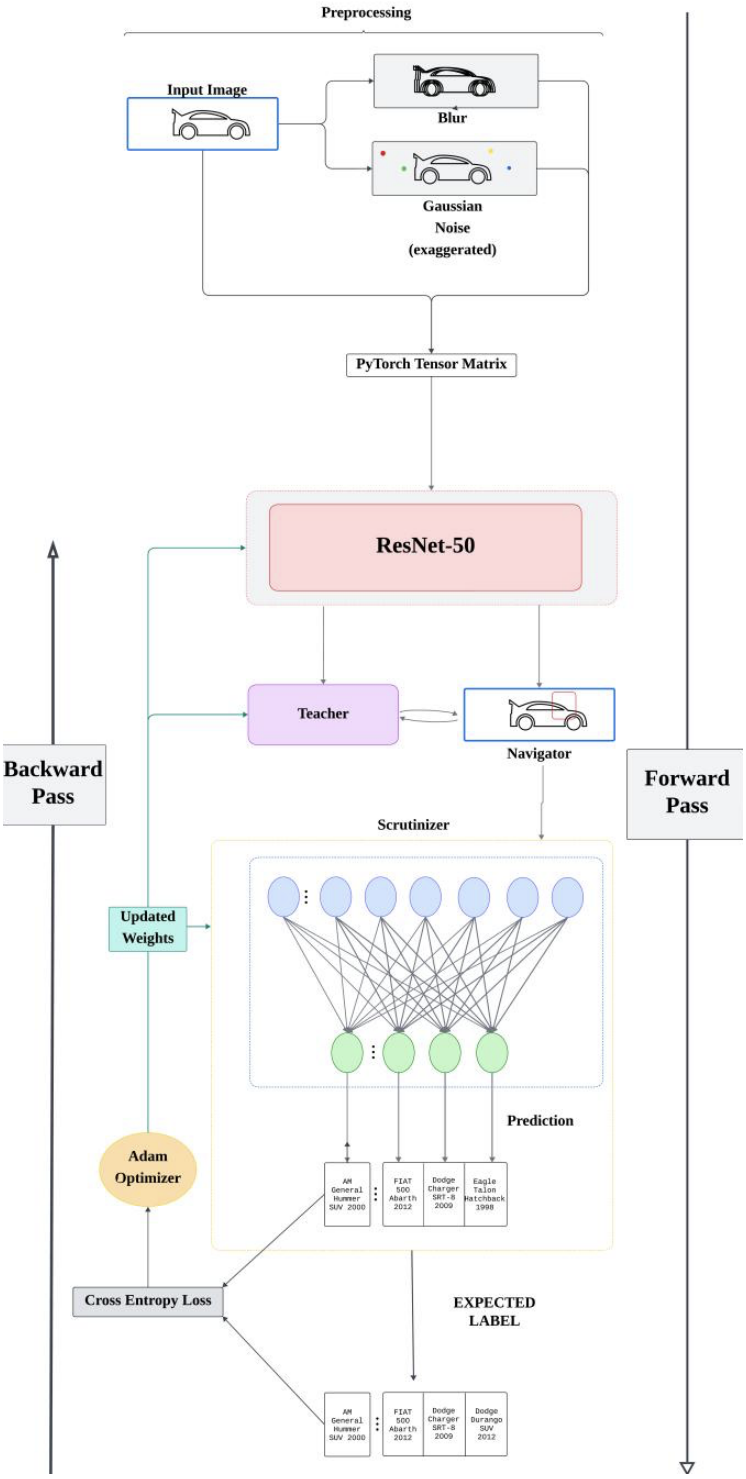


Figure 1: Architecture of Our Model

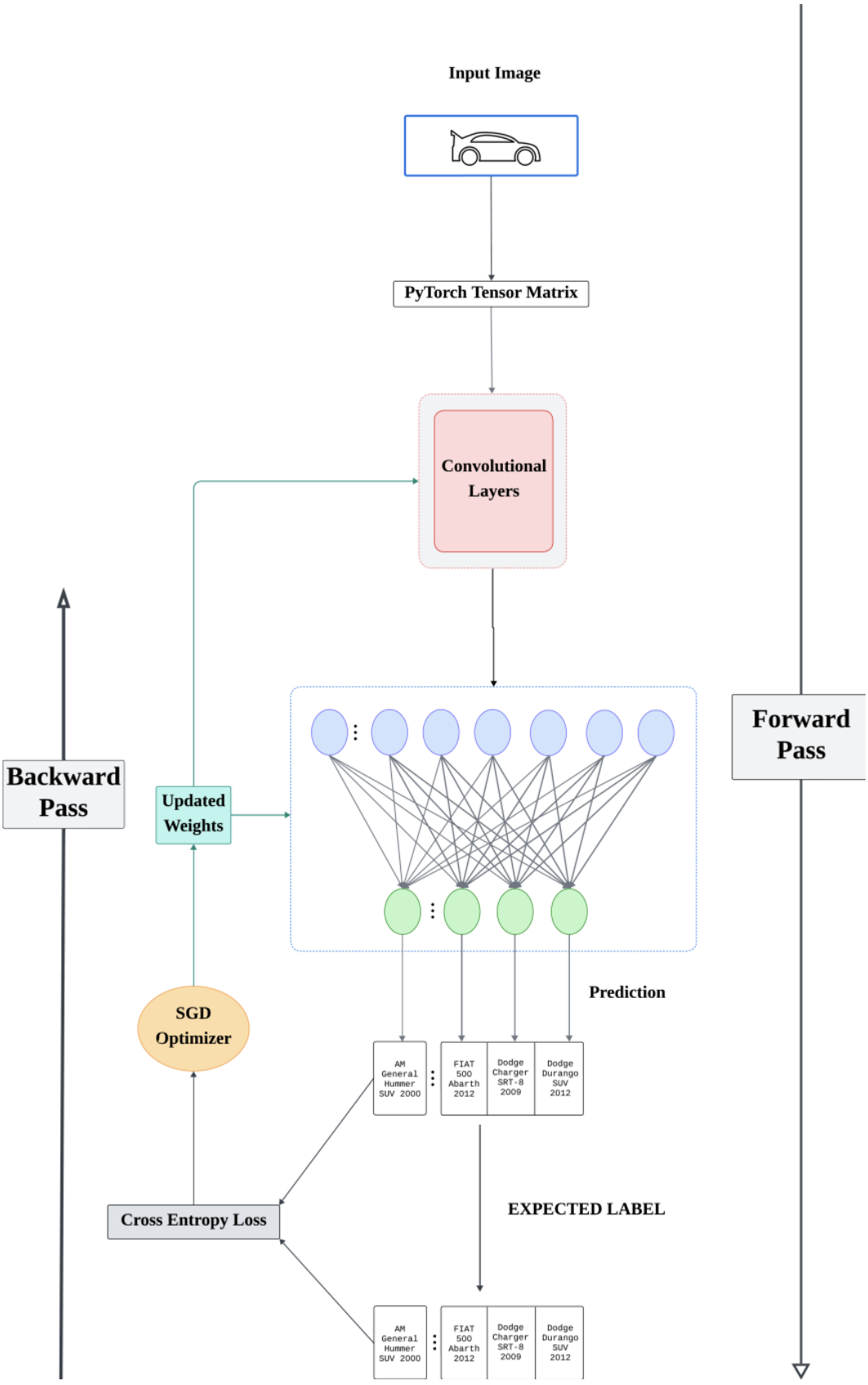


Figure 2: Our Baseline Model

3 BACKGROUND & RELATED WORK

Collecting a Large-Scale Dataset of Fine-Grained Cars — Krause et al. (2013) consolidates meaningful images with a variety of angles, lighting, and vehicle colours has proven to be a challenging task. This paper introduces a large-scale dataset with 16,185 unique vehicle images split between 196 fine-grained classes. This paper also introduces benchmarks using primitive machine learning techniques such as Locality Constrained Linear Coding and BubbleBank. Ultimately, this paper demonstrates that fine-grained vehicle classification systems have immense potential for improvement.

A Systematic Evaluation of Recent Deep Learning Architectures for Fine-Grained Vehicle Classification — Valev et al. (2018) analyzes the performance of several CNN models on a large-scale vehicle classification dataset. The study compares DenseNet, ResNet, Inception, and VGG architectures in distinguishing between 196 fine-grained vehicle classes. Fine-tuning pre-trained models significantly boosted classification accuracy by as much as 50%. Among the tested models, this paper established that DenseNet161 achieved the highest accuracy, while other DenseNet variants also performed competitively. In contrast, the VGG networks yielded the lowest accuracies, and ResNet and Inception architectures performed moderately well.

Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring — Kherraki & El Ouazzani (2022) looks to solve a real-time binary classification problem: determining whether or not a car is an emergency vehicle. Using a dataset of 2352 images, this paper explored the performance of various deep CNN architectures for this classification task. DenseNet121 yielded the highest accuracy, with ResNet-50, VGG16, and MobileNetV2 producing similar results.

Aircraft Classification Using Image Processing Techniques and Artificial Neural Networks — Jain et al. (2015) proposes a deep learning methodology for classifying the type and model of an approaching aircraft using a static image. This method uses an edge detection algorithm with hand-defined, shape-based features to create a feature matrix. These features are then passed into an artificial neural network for classification. This method yielded an accuracy of 97%.

Revisiting the CompCars Dataset for Hierarchical Car Classification — Buzzelli & Segantin (2021)improves the CompCars dataset to better reflect real-world car classification scenarios by identifying bias in the original train/test split and introducing bounding boxes and full type-level annotations. The authors evaluate hierarchical classification methods using CNNs, finding that a two-step cascade with binary cross-entropy achieves the highest MMy accuracy (70.1%). The revised dataset and benchmarks provide a more realistic foundation for fine-grained vehicle classification research.

A Comparative Study of Lightweight, Resource-Efficient Backbones for Image Classification — Pranav Jeevan P & Amit Sethi (2024) benchmarks 11 lightweight CNN and transformer backbones from Torchvision across 18 datasets spanning natural images, textures, remote sensing, plants, astronomy, and medical imaging. ConvNeXt-Tiny performs best on natural images; RegNet and EfficientNetV2-S are consistently strong across domains; and WaveMix excels on low-resolution tasks. Transformer backbones underperform CNNs in low-data fine-tuning, while ShuffleNetV2 outpaces MobileNetV3 for ultra-lightweight use. The study offers practical guidance for choosing backbones under tight compute and memory budgets.

4 DATA PROCESSING

We use the Stanford Cars Dataset (Krause et al., 2013), which contains approximately 16,000 images evenly split into training and testing sets. The dataset includes 196 pre-labeled classes, each specifying the make, model, and year of a car. The dataset is fairly balanced amongst all the classes as shown in Figure 3 Hassan et al. (2021).

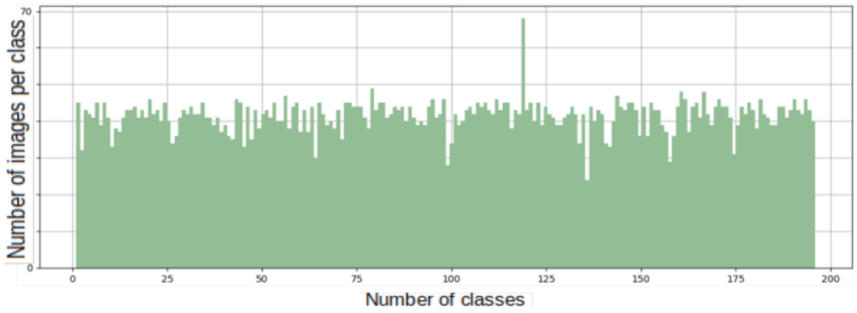


Figure 3: Per-class image counts in the Stanford Cars dataset (196 classes). The distribution is approximately uniform, indicating minimal class imbalance.

Our dataset was split into 80/10/10 for training, validation, and testing respectively for both the backbone ResNet-50 and NTS. The dataset consists of images of cars from a variety of angles, allowing for comprehensive training of the make, model and year. Images in this dataset are mostly high-quality and well-centered, real-world data varies significantly. To combat this, we employ data augmentation techniques to improve model generalization.

The pre-processing consists of two main components. The base transform, applied to all images, first resizes each image to 224×224 pixels to match ResNet input requirements. The images are then converted into PyTorch tensors and normalized using the ImageNet mean [0.485,0.456,0.406] and standard deviation [0.229,0.224,0.225], which are required for ResNet architectures.

The augmentation transform, applied to an additional copy of the training set, also begins by resizing images to 224×224 pixels. It then applies a horizontal flip and a Gaussian blur to introduce variability. The images are subsequently converted into tensors, and Gaussian noise with a standard deviation of 0.05 is added. Finally, the same ImageNet normalization is applied.

5 ARCHITECTURE

The primary architecture that we used was a low-scale NTS network Yang et al. (2018) with a ResNet-50 backbone. NTS is a particularly strong architecture for this task as it is able to discern incredibly small differences between classes—for instance, this is useful when comparing two identical cars with different release years. The main components of NTS are a navigator, teacher, scrutinizer, and backbone.

Firstly, the ResNet-50 backbone was fine-tuned separately as an end-to-end classifier for the Stanford Cars dataset. Transfer learning from the ImageNet dataset was implemented to assist the model with identifying low-level features. The ideal hyperparameters for training ResNet-50 were found to be a batch size 64, learning rate of 0.0001, and 10 epochs. These ResNet-50 weights were saved and used within the Teacher and Scrutinizer modules.

Within NTS, data first was passed through the navigator module, which consists of a 3 layer CNN feature extractor with a kernel size of 1 and batch normalization between layers. This module splits the image into 8 regions and ranks them in a matrix based on how discriminative they are. This matrix is passed to the teacher, which uses the fine-tuned ResNet-50 followed by a single linear layer to more accurately rank these 8 regions within the matrix. The teacher does this by using each region to make a prediction into the image's class. This reordered matrix is then passed into the scrutinizer module, which aggregates the features of each region with the features of the global image by concatenating. Finally, the scrutinizer uses the fine-tuned ResNet-50 along with 3 linear layers (making use of dropout to prevent overfitting) to make a prediction for the image class.

Each module within NTS was backpropagated separately: the navigator used the teacher's ranking for backpropagation and the scrutinizer used the final prediction. The teacher was not backpropagated, as it depended directly on the previously fine-tuned ResNet-50 backbone. Cross Entropy

Loss and the Adam optimizer was used with the following hyperparameter choices: a learning rate of 0.0001, batch size of 64, image split of 8 regions, and 20 epochs.

6 Baseline Model

A realistic baseline for this task requires the ability to extract fine-grained features from images. As such, we used numerous pre-trained and fine-tuned deep CNNs for classifying cars and compared their accuracies. In particular, we analyzed ResNet models—modifying their classification head to output a result for 196 classes—as these were found to produce high accuracy for car classification [source of that one website with all the accuracies]. ResNet-18, ResNet-34, ResNet-50, and ResNet-118 were all tested with fine-tuning and pre-training. Fine-tuning these models for the Stanford Cars dataset caused their accuracies to increase to approximately 65%. Fine-tuned ResNet-50 performed best as an end-to-end classifier while maintaining a reasonable training time, ultimately achieving an accuracy of 74.44%. Transfer learning from the ImageNet dataset was used for all of these models to allow them to better extract low-level features.

7 Quantitative Results

Our model performs well at a testing accuracy of 87.32%, with a validation accuracy of 89.78%. In addition, validation accuracy is 5% less than training accuracy by the 20th epoch, and training accuracy reaches about 95%, indicating the model is overfitting to the training data by a little.

The accuracy of the 5 worst classes was printed to determine the lower bounds of the model's performance on the 196 classes.

```
5 worst classes by index and accuracy:
Class 74 - Accuracy: 11.36%
Class 103 - Accuracy: 19.05%
Class 20 - Accuracy: 26.19%
Class 7 - Accuracy: 26.67%
Class 70 - Accuracy: 28.57%
```

Figure 4: Accuracy of the five worst-performing classes in the model's predictions, identified by class index and percentage accuracy.

As shown, the worst classified class was 74, with the second worst class having almost double its accuracy. After double-checking the training dataset, a lot of the images in this class were from the same angle, and some cars were cut off–this led to the model being undertrained for this specific class due to missing details, explaining its low accuracy of 11.36%. An example of camera angle and cut-off car image for the class can be found below:

Figure 5: Example image from the worst-performing class (Class 74 – Chevrolet Silverado 1500 Regular Cab 2012) showing camera angle and cut-off issues that contributed to low accuracy.

A confusion matrix was designed in order to visually observe the model's performance:

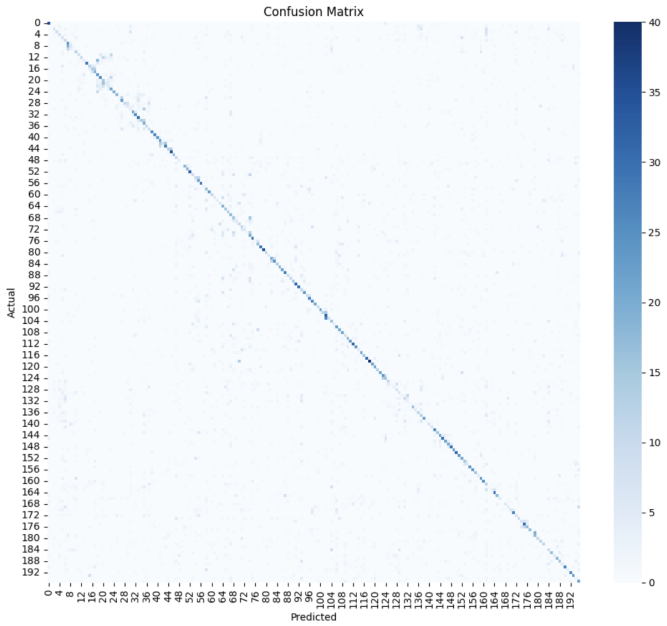


Figure 6: Confusion matrix

There is a strong diagonal through the matrix, which indicates that the model classifies each image into the correct class of cars well, supporting the high test accuracy. However, there are cluster errors in the regions 12-24 and 52-76, indicating the model is mixing up surrounding classes and having a lower performance in those regions. In regions above class 76, the confusion matrix shows no observable cluster errors, indicating stronger performance in the upper half of the classes.

8 Qualitative Results

Displaying the model's predictions on numerous cars made it evident that the images where the camera is positioned behind the car are most challenging. An example of a misclassifications from this angle is displayed below. We hypothesized that this was because there was a deficit of rear-view training images in this region of classes, and later confirmed this by inspecting all of the training images for the class shown below.

Figure 7: Rear-view misclassification example (Predicted 28, Actual 29). The rear-angle view and cropped details make these two classes easy to confuse.

This makes sense as the model had lower performance in regions near 12-24 in the confusion matrix above.

Training graphs were primarily used for hyperparameter tuning. As shown in the following graphs, train accuracy peaks around 95%, with overfitting only beginning to occur at around epoch 20. Similarly, validation accuracy does not experience any overfitting until around the 20th epoch, where validation accuracy plateaus, proving this to be a good stopping point. The model also appears to converge effectively with a learning rate of 0.0001, as decreasing the learning rate causes erratic behaviour in later epochs and a smaller learning rate makes it difficult for the model to converge at all.

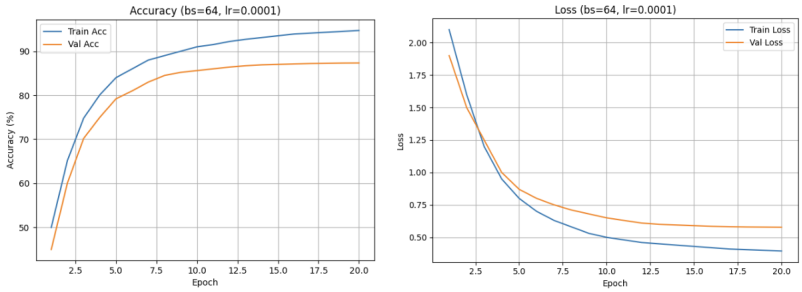


Figure 8: Training and validation accuracy/loss for ResNet-50 (batch size 64, learning rate 1e-4). Accuracy plateaus around 20 epochs while loss declines smoothly; the small train–val gap indicates mild overfitting and informed our stopping point.

9 Evaluation on New Data

While the Stanford cars dataset sets aside approximately 8000 unique images for testing, it was crucial to ensure that these images were representative of real-world conditions and were truly unseen. To ensure that the model had no experience with these test images, the validation set was extracted from the training set to be used for all hyperparameter tuning and architecture tweaks. In addition to this, no augmentations were applied to any of the testing images to ensure that they correctly represented real-world scenarios. Furthermore, 100 images were randomly selected and examined to ensure that visual features were realistic and not unnecessarily clear or easy to classify (therefore skewing results). As displayed below, some unseen images from the testing set were also randomly extracted, and the model correctly predicted 7 out of 8 cars, which is consistent with its accuracy on the entire testing set. Ultimately, the model was determined to produce an accuracy of 87.3% on unseen test results.

Figure 9: 8 randomly displayed images in the testing set, where 7/8 were classified correctly

10 Discussion

The final model achieves a test accuracy of 87%, with a strong diagonal confusion matrix. Errors are not uniformly random; they concentrate in a handful of visually proximate classes. Compared to a plain ResNet-50 baseline (74.4%), the NTS-Net variant delivers a 13% improvement.

We think our model performs well because the navigator–teacher–scrutinizer design directs attention to the most class-distinctive areas, complementing global features from the backbone. This keeps the model from focusing on broad, misleading patterns and helps it to tell apart look-alike cars (like adjacent years or trims). The training/validation curves also show a smaller gap than the baseline, which fits a model that focuses more on small, local details.

What we found interesting was that even though the classes are balanced, most mistakes clump together in a few groups of very similar cars. That indicates look-alike car models and image quality is the main issue, not lack of data. Also, a quick spot check ($7/8 \approx 87.5\%$) lines up with the overall test accuracy, which makes us confident the result isn't just luck because of how the data was split. In addition, some near-misses are off by one "generation" within a model line, which is evidence that the model has largely learned the family but is still sensitive to subtle year-specific cues like grille geometry, taillight shape, or wheel design.

We learned that the backbone and where the model specifically looks really matters, where attention to the most class-informative parts is more important than treating the whole image equally. Data ultimately constrains performance. Basic augmentations such as flips provide only incremental gains and do not address shifts in viewpoint or occlusion so the hardest errors point to domain mismatch rather than model capacity. We also learned that overall accuracy isn't enough. By studying where the model confused labels and checking images, we were able to find the issues and fix errors.

The model's performance is good overall. The main issues are look-alike subclasses, low-resolution, images and domain shift, but we believe the goals are met, with next steps that could be made to improve our model.

11 Ethical Considerations

While our system has the potential to support law enforcement and enhance public safety, it also raises important ethical concerns. One key risk is the possibility of misuse for unauthorized surveillance, which could infringe upon individuals' privacy rights. Such misuse could occur if the system were deployed outside of regulated contexts or without proper oversight, making it essential to establish clear governance and accountability measures.

Another consideration lies in the potential limitations of the dataset. If the dataset lacks diversity—such as having fewer images of older, rare, heavily modified, or non-mainstream car models—it may introduce bias and reduce classification accuracy. Also, challenging conditions such as poor image quality, extreme lighting, or unusual camera angles can further contribute to misclassifications. These limitations underscore the need for careful deployment, regular performance auditing, and continuous model refinement to ensure fairness, reliability, and robustness across diverse real-world scenarios.

12 Project Difficulty / Quality

Vehicle classification has repeatedly been noted to be a difficult task due to the immense number of realistic training images required, the vast number of classes, the extensive training time and computational demand, and the minute differences between identical cars of different generations. While Stanford Cars provides leverage for building effective vehicle classification models, it is not sophisticated enough to allow car classification to become a feasible technology in the real world. Despite using numerous augmentations to represent real-world noise, our model would still struggle to classify cars if implemented in surveillance or police systems.

Despite these challenges, our NTS-Net architecture achieved performance that exceeded typical results for this dataset in a university-level course. While many projects within the past several years exceeded 90% accuracy on this dataset, they used significantly more advanced techniques and train using computers that have significantly stronger processing power than what is available to our team. In specific, a 2018 paper achieved 93.9% accuracy on the Stanford Cars dataset using a more advanced NTS architecture Yang et al. (2018). Therefore, we believe that our network performs exceptionally well for this task given the challenges associated with our task.

REFERENCES

Marco Buzzelli and Luca Segantin. Revisiting the compcars dataset for hierarchical car classification: New annotations, experiments, and results. *Sensors*, 21(2):596, Jan 2021. doi: 10.3390/s21020596. URL https://www.mdpi.com/1424-8220/21/2/596.

Aqsa Hassan, Mohsin Ali, Nouman M. Durrani, and Muhammad Atif Tahir. An empirical analysis of deep learning architectures for vehicle make and model recognition. *IEEE Access*, 9:91487–91499, 2021. doi: 10.1109/ACCESS.2021.3090766. URL https://doi.org/10.1109/ACCESS.2021.3090766.

Avinash Jain, Manju Khare, and Ankit Tiwari. Aircraft classification using image processing techniques and artificial neural networks. *International Journal of Computer Applications*, 119(23): 25–30, 2015. doi: 10.5120/21179-4200.

Ahmed Kherraki and Rachid El Ouazzani. Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring. In *2022 3rd International Conference on Intelligent Systems and Computer Vision (ISCV)*, pp. 1–6. IEEE, 2022. doi: 10.1109/ISCV55399.2022.9798674.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3334–3341. IEEE, 2013.

Pranav Jeevan P and Amit Sethi. Which backbone to use: A resource-efficient domain specific comparison for computer vision. *arXiv*, 2024. URL https://arxiv.org/abs/2406.05612.

Dimitar Valev, Radoslav Pavlov, and Boyan Aleksiev. A systematic evaluation of recent deep learning architectures for fine-grained vehicle classification. *Mathematics and Informatics*, 61(4): 371–384, 2018.

Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 420–436. Springer, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/papers/Ze_Yang_Learning_to_Navigate_ECCV_2018_paper.pdf.