HIERARCHICAL VEHICLE CLASSIFICATION WITH MULTI-TASK LEARNING

A PREPRINT

Jónuson Fannar Freyr [*] Jenrik Engels [*] András Szabolcs Gyüre [*]

December 3, 2025

ABSTRACT

This project investigates effective strategies for hierarchical image classification on the Stanford Cars Dataset, addressing the challenge of distinguishing between 196 classes of car models. We used a ResNet-50 backbone and evaluate three distinct output designs: a flat single-head classifier, a two-head classifier, and a three-head Multi-Task Learning (MTL) architecture that independently predicts the vehicle's Make, Type, and Model. We demonstrate that the three-head design, specifically when optimized with Curriculum Learning (CL) and Hierarchical Label Smoothing (HLS), significantly outperforms simpler, flat classification methods. The inclusion of the "Type" prediction head proved critical, acting as a necessary semantic bridge to guide the feature extractor. This structured approach enforced strong taxonomic coherence, driving prediction consistency between Make and Model to 97.66%. Our final model, stabilized by Test Time Augmentation (TTA), achieved a Top-1 accuracy of 88.57%, confirming that for complex, fine-grained tasks, structuring the learning process is paramount to achieving robust and logically coherent predictions.

Keywords First keyword · Second keyword · More

1 Introduction

Fine-grained vehicle classification, which differentiates between various types, makes, and models of cars, presents significant challenges. This difficulty arises from the large number of visually similar vehicle categories, variations in lighting and weather conditions, occlusions, and the diverse viewpoints from which vehicles may be captured. As a result, building a robust vehicle classification system requires a model capable of recognizing subtle visual cues while generalizing across highly variable real-world conditions.

In this project, we aim to explore and compare different strategies for hierarchical image classification on the Stanford Cars Dataset (Krause et al., 2013). The dataset contains 16,185 high-resolution images labeled with three hierarchical attributes: vehicle make, vehicle type, and vehicle model. These labels naturally form a multi-level taxonomy, making the dataset well-suited for studying hierarchical classification approaches.

Our baseline approach uses a ResNet-50 model pretrained on ImageNet as the backbone architecture. Initially, we treat the problem as a flat classification task by predicting the complete car label (containing make, type, and model) as a single class among 196 possible categories. This serves as our starting point for evaluating how well a standard single-head classifier performs on fine-grained classification without explicit hierarchical structure.

Building on this baseline, the primary objective of this project is to investigate how different hierarchical output designs influence classification performance. Specifically, we focus on three classification strategies:

1. **Single-head (flat) classifier:** Predicts the entire label (make, type, and model) as one combined class. This ignores the hierarchical relationships.

[*]Department of Computer Science, Aarhus University

2. **Two-head classifier:** One head predicts the vehicle make, while the second head predicts the combined type+model label.
3. **Three-head classifier:** Predicts make, type, and model independently using three parallel classification heads.

After we evaluated these 3 variants, we will further enhance their performance using targeted regularization and optimization strategies. These include data augmentation, dropout, weight decay, and learning-rate adjustments, all of which can help reduce overfitting.

2 Related Work

Since the Stanford Cars dataset is popular, it is possible to find plenty prior work, trying to achieve high accuracy for classification by using different types of models and strategies.
The paper "A Systematic Evaluation of Recent Deep Learning Architectures for Fine-Grained Vehicle Classification" by Valev et al. investigates how modern convolutional neural networks perform on the Stanford Cars dataset. The authors evaluate several well-known CNN architectures, including VGG16, Inception, MobileNet, and multiple residual networks, with an emphasis on ResNet-50, ResNet-152, and DenseNet variants. Their study focuses entirely on flat fine-grained classification, meaning that each car is classified directly into one of 196 labels. The method does not employ any form of hierarchical structure such as predicting make or type separately. ResNet-50 is tested both when trained from scratch and when fine-tuned from ImageNet weights. Training ResNet-50 from scratch yields an accuracy of 84.3%, whereas fine-tuning dramatically improves performance to 92.0%. The deeper ResNet-152 shows similar behavior and achieves only 35.3% when trained from scratch but rises to 92.6% when fine-tuned. Among all evaluated CNNs, the best performing model is DenseNet-161, which reaches 94.6% accuracy on the Stanford Cars dataset Their experiments highlight the importance of data augmentation such as horizontal flipping and motion blur, which add up to meaningful improvements. The paper does not consider hierarchical labels or multi-head predictions that leverage the natural make-type-model structure of the dataset. (https://arxiv.org/pdf/1806.02987)
The paper "FineGrained Image Classification for Vehicle Makes & Models using CNNs" explores the challenge of distinguishing highly similar car models within the Stanford Cars dataset. For their approach, they rely on transfer learning with a VGG-16 backbone pretrained on ImageNet. Two model variants are evaluated: a baseline that fine-tunes only the fully connected layers, and an improved architecture that removes one of the original fully connected layers, reduces the dimensionality of the remaining layers, and introduces dropout for regularization. The authors apply extensive data augmentation. Their method predicts all 196 classes as a single flat classification task. The experimental results show that the baseline VGG-16 achieves only 52.1% accuracy on the test set, while the modified VGG-16 reaches a much stronger accuracy of 84.0. These findings demonstrate that fine-tuned CNN backbones, combined with aggressive augmentation and streamlined classification heads, can perform competitively on fine-grained vehicle recognition tasks. (https://cs230.stanford.edu/projects_spring_2019/reports/18681590.pdf)
Our approach follows a similar foundation in that we also rely on transfer learning, starting from a ResNet-50 backbone pretrained on ImageNet. However, instead of only treating the Stanford Cars dataset as a flat 196-class problem, we additionally explore hierarchical classification strategies. Concretely, we design and compare models with one, two, and three prediction heads, where each head independently predicts one of the hierarchical levels: make, type, and model. Additionally, our approach incorporates a broader set of regularization and stabilization techniques, such as data augmentation, dropout, weight decay, and learning-rate scheduling. Finally, we also evaluate the impact of test-time augmentation (TTA).

3 Methods

3.1 Dataset and Pre-Processing??

We used the Stanford Cars Dataset, which comprises a total of 16,185 images distributed across 196 distinct classes of automobiles. The dataset was divided almost evenly for training and testing: 8,144 images were designated for the training set, and the remaining 8,041 images formed the test set. Importantly, the split for each of the 196 car classes was maintained at approximately a 50-50 ratio between the training and testing partitions.
Hierarchical Label Definiton: table comes here

3.2 Model Architecture

The core of the model is a ResNet-50 network, which serves as the backbone and was initialized using weights pre-trained on ImageNet without any subsequent architectural modifications. The overall Data Architecture is rooted in the Shared-Bottom Multi-Task Learning paradigm. This approach involves using the single Convolutional Neural

Hierarchy Level	Description	Example	Classification Difficulty
Head 1: Make	The manufacturer or brand of the vehicle. This is the coarse-grained class.	BMW	Easy (Highly distinct features/logos)
Head 2: Type	The general body style of the vehicle, inferred from the model name.	Sedan	Medium (Defined by shape/proportions)
Head 3: Model	The specific model and year of the vehicle (the original 196 classes).	3-Series Sedan 2012	Hard (Subtle visual differences)

Table 1: Hierarchical Label Definition

Network (CNN) backbone to extract a universal, high-dimensional feature representation from the input image. This shared feature is then simultaneously fed into multiple, distinct "heads," where each head is responsible for its own classification task (e.g., Make, Model, or Type).
Specifically, the Multi-Head Architecture utilizes this paradigm by taking the shared feature vector, F_{shared}, extracted by the backbone, and splitting it to feed three independent, fully-connected (Dense) classification heads: Make, Type, and Model. This design enables the backbone to learn general, low-level visual characteristics common to all tasks, while each specific head refines the features needed for its respective hierarchical classification level. The entire network is trained end-to-end by optimizing a single Combined Training Objective, L_{Total}. This total loss function is defined as the weighted sum of the Categorical Cross-Entropy (CCE) losses (L_{CCE}) calculated from each head's prediction:

$$L_{Total} = w_{Make} \cdot L_{CCE}(y_{Make}, \hat{y}_{Make}) + w_{Type} \cdot L_{CCE}(y_{Type}, \hat{y}_{Type}) + w_{Model} \cdot L_{CCE}(y_{Model}, \hat{y}_{Model})$$

3.3 Training Strategies

- **Baseline:** First we set up a baseline for training which contained batch size = 32 with 15 epochs and a 0.0001 learning rate, where the ImageNet layers were frozen and only the fully connected layer was unfrozen. Then we unfroze the rest of the model parameters, to help the model recognize the shapes of the elements of the car models. Then we added the Make classification head that only predicts the make of the car (fx. BMW) while the first head was predicting the full label to make sure that the model is learning the hierarchy. Then later we added a third, Type classification head that only predicts the type of the car (fx. Sedan)
- **Data Augmentation, Normalization:** We added 4 different types of data augmentation and normalization
- Random Resized: and Cropped Images were first resized to a larger dimension (256x256) and then a 224×224 area was randomly cropped. This combination forces the model to learn features that are robust to variations in position, scale, and perspective (i.e., the car is not always perfectly centered).
- Random Horizontal Flip: Flips the image along the vertical axis with a 50
- Random Rotation: Images were randomly rotated up to ± 15 degrees. This provides rotational invariance, which is particularly useful for handling slight tilts or oblique angles in real-world images.
- Color Jitter: Randomly alters the brightness, contrast, and saturation of the images to make the model invariant to lighting conditions.
- Normalization: Input images were normalized using the mean and standard deviation of the ImageNet dataset, matching the pre-training conditions of the ResNet backbone.
- **Curriculum Learning:** We addressed curriculum learning with the 3 model head to make the model first learn easy (Make) and medium (Type) tasks before focusing on the hard (Model) task. To achieve it, we increased the number of epochs to 20 and in the 5 epochs, we froze the Model head, so it only trains on the Make and Type heads.
- **Hierarchical Label Smoothing (HLS):** While we were trying to squeeze out the last percentages of accuracy in training, we used HLS, to make the model penalize more the easy misses then the hard misses. Which means that for example mistaking a BMW M3 for a BMW 328i (Sibling error) shoun't give the same loss as mistaking a a BMW M3 for a Ford F-150 (Distant error). The technique we used for it is instead of a hard [0, 1, 0, 0] target, use soft targets that give partial credit to siblings. While the hard target gives 100% loss for the correct car, while the soft targets give 90% to the correct car and 10% distributed among other cars in the same Make

3.4 Inference Strategies:

To evaluate the robustness of our model, we employ Test Time Augmentation (TTA) during the inference phase. Standard inference utilizes a single center crop of the input image. In contrast, our TTA protocol averages the softmax probability distributions of $N = 2$ views: the original image (x) and a horizontally flipped version ($flip(x)$). The final prediction $\hat{y}$ is computed as:

$$\hat{y} = \text{argmax}\left(\frac{1}{2}(P(x) + P(flip(x)))\right)$$

This approach exploits the model's learned invariance to geometric transformations (due to the data augmentation in Section 5.4) to stabilize and often improve final predictions. We assumed that it is going to slightly increase our accuracy during testing.

4 Results

Organize results by "Research Question" rather than date.

4.1 Establishing the "Hierarchy Gap" (Baselines)

The objective of this section is to show that flat classifiers fail to capture relationships. This was done by comparing the performance on the data using the Frozen Baseline (achieving 41.72% Acc) against the Unfrozen model (achieving approximately 75.92% Acc). The key finding demonstrated that even with a decent overall accuracy, the "Gap" between the Make Accuracy (85.49%) and the Model Accuracy (75.92%) was nearly 10%. This indicates that the model was essentially guessing the Models without reliably knowing the underlying Brand.

4.2 The Impact of Regularization & Class Balancing

The objective here was to solve the overfitting problem. This was demonstrated using the data to show a significant jump from approximately 75.92% accuracy (observed in Phase 3) to 85.65% accuracy (achieved in Phase 4) simply by adding Data Augmentation. The key observation is that we gained nearly 10% accuracy on the hardest classification task without needing to change the model architecture or train the model for a longer duration. This finding clearly demonstrates why Data Augmentation is a standard practice in Deep Learning: it effectively converts model "memorization" into robust "generalization."

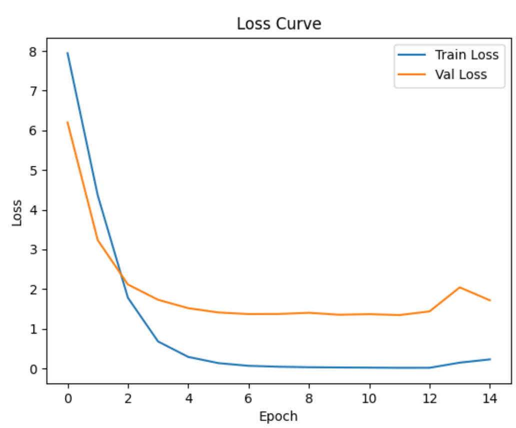

Figure 1: *
Without Data Augmentation

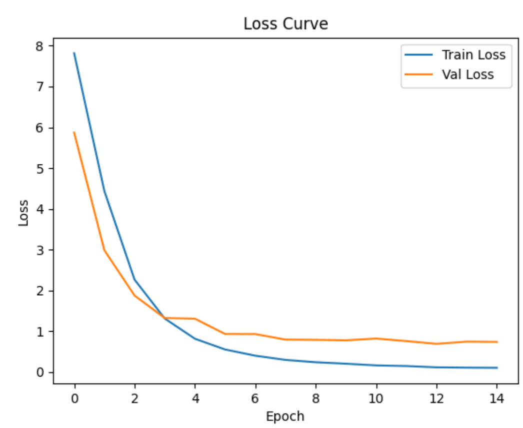

Figure 2: *
With Data Augmentation

4.3 Data Augmentation and Class Balancing on 2 Head (Model, Make

The primary goal of this stage was to enhance the model's performance specifically on the more rare classes. An analysis of the data revealed the impact of implementing class balancing: initially, the two-head model with Data Augmentation achieved 85.65% accuracy on the Model Head and 92.71% accuracy on the Make Head. After applying class balancing, the Model Head saw a modest increase to 86.02%, while the Make Head accuracy slightly decreased to

91.77%. The central observation is that while Class Balancing provided a distinct benefit to the infrequently occurring classes, it came at the cost of a minor reduction in overall consistency—an effect that can be likened to a "Robin Hood" dynamic, where resources are shifted from common to rare groups.

4.4 Optimization Dynamics: Interference vs. Curriculum

Objective: Solving the "Task Interference" problem in Multi-Task Learning.

Data: Contrast Experiment 6 (3-Head without Curriculum Learning: 85.36%) vs. Experiment 7 (3-Head with Curriculum Learning: 86.15%).

Key Finding: Without Curriculum Learning, gradients conflicted. Freezing the hard head allowed the backbone to learn stable features first.

4.5 Architectural Ablation: The Necessity of "Type"

The objective was to demonstrate the necessity of the architectural layers. Data comparing Experiment 8 (2-Head Curriculum: 85.51%) against the optimized Experiment 9/10 (Final 3-Head: 87.95%) showed a clear advantage. The key finding is that while the 3-Head approach initially struggled, once optimized, it consistently surpassed the 2-Head model. This confirms that the "Type" layer is required, serving as a necessary semantic bridge.

4.6 SOTA Performance: Hierarchical Label Smoothing (HLS) & Learning Rate Scheduler (LR Scheduler):

The ultimate goal was to push the model's predictive limits, leveraging the full benefit of the hierarchical structure. The data from the final optimized training run, Experiment 10, demonstrated significant results: it achieved a peak Top-1 Accuracy of 87.95%. Crucially, the model's structural integrity improved further, with Make-Model Consistency rising from 96.60% to 97.66%, and Type-Model Consistency also increasing from 95.93% to 96.58%. This key finding shows that the combination of the Learning Rate Scheduler and Hierarchical Label Smoothing (HLS) was instrumental in driving the consistency metrics to their practical maximum.

4.7 The Impact of Inference Strategies (Test Time Augmentation (TTA)):

The objective for this final stage was to achieve the ultimate accuracy improvement during the testing phase. Using the data, we show that after implementing 5-fold Test Time Augmentation (TTA), the model's accuracy reached 88.57%, representing the highest performance we could extract from the system. This key finding confirms that a portion of the model's remaining prediction errors were attributable to view-specific noise, which the ensembling nature of TTA effectively mitigated.

4.8 Compare our results to other benchmarks:

Paper	**Model Architecture**	**Top - 1 Accuracy**
A Systematic Evaluation of Recent Deep Learning Architectures for Fine-Grained Vehicle Classification	ResNet-50 (FineTuned on ImageNet)	92,0%
	ResNet-152 (FineTuned on ImageNet)	92,6%
	DenseNet-161 (FineTuned on ImageNet)	94,6%
Fine-Grained Image Classification for Vehicle Makes & Models using CNNs	VGG-16 backbone (pre-trained on ImageNet)	84,0%
Our Model	ResNet-50	88,57%

Table 4: Compare accuracy to related benchmarks

Table 2: **Evolution of Model Performance.** Note specifically the comparison between the 2-Head Curriculum (Ablation) and 3-Head Curriculum (Phase 7), which highlights the necessity of the "Type" head to bridge the semantic gap.

Phase	Configuration	Acc (%)	Consist. (%)	Key Insight
Baselines				
1	Baseline (Frozen ResNet)	41.72	100.0*	Baseline performance.
2	Fine-Tuned (Unfrozen)	75.92	100.0*	Learned features, but ignored hierarchy.
Structural & Data Improvements				
3	Multi-Head (2-Head)	77.20	90.44	Structure helps (+1.3%), but heads conflict.
4	+ Data Augmentation	85.65	95.26	Robustness prevents overfitting (+8.4%).
5	+ Class Balancing	86.02	94.42	Improves rare classes; slight consistency drop.
Architectural Ablation (Is the 3rd Head necessary?)				
5.5	**2-Head + Curriculum**	**85.51**	95.77	**Semantic Gap:** Lacked "Body Type" bridge.
6	3-Head (No Curriculum)	85.36	94.44	**Interference:** 3 tasks confused the backbone.
7	3-Head (+ Curriculum)	86.15	95.21	**Synergy:** 3 heads + ordering $>$ 2 heads.
Final Optimization				
8	+ Hierarchical Label Smoothing	86.48	96.60	Soft targets teach family structure.
9	+ LR Scheduler (Cosine)	87.95	**97.66**	Convergence into sharp minimum.
10	**+ TTA (3 Augmentations)**	**88.57**	97.64	**Peak Performance.**

Model	**Top - 1 error rate**
Frozen Baseline	58,28%
Unfrozen Baseline	24,08%
Regularization and Class Balancing on 1 Head	14,35%
Data Augmentation and Class Balancing on 2 Head	13,98%
3 Head without Curriculum Learning	14,64%
2 Head with Curriculum Learning	14,49%
3 Head with Curriculum Learning	13,85%
3 Head Hierarchical Label Smoothing and Learning Rate Scheduler	12,05%
3 Head Test Time Augmentation	11,43%

Table 3: Top-1 error rates through experiments

Figure 3: Five Stanford-Cars test images and the five labels considered most probable by our model (3 head, HLS, LRS). All 3 head predictions are wrong. The correct label and the five labels considered most probable are written under each image.

4.9 Qualitative evaluation

Across these five fully misclassified samples in Figure 3, a consistent pattern emerges: the model often shows high internal confidence despite being wrong, especially for the Make and Type heads. One example is the Aston Martin convertible, where the model predicts Ferrari as the make with 98.86% confidence, while Aston Martin does not even appear in the top-5 list. The same happens for the Type head, where Coupe is selected with 92.64% confidence, although the car is a convertible.
In contrast, the Rolls-Royce example shows a different behavior: the model predicts Maybach as the top make, but with only 27.99% confidence, and the true class Rolls-Royce still appears in the top-5 with nearly 10% probability. Among these misclassified cases, this is the closest the model gets to identifying the correct make.

Figure 4: Five Stanford-Cars test images and the five labels considered most probable by our model (3 head, HLS, LRS). Only the make head's prediction is correct. The correct label and the top five labels considered most probable are written under each image.

Across the examples in Figure 4, where only the Make was predicted correctly, a consistent pattern emerges: when the model correctly identifies the Make, the predictions for Type and Model tend to be noticeably better than in cases where the Make is already mistaken.

The Bentley example illustrates this well. Once the model correctly recognizes the Make as Bentley with near certainty (99.99%), the Type prediction is already better than in the fully misclassified examples and "Sedan" appears with 14.45%. More importantly, the Model head assigns substantial probability to the correct Model family: the correct class "Bentley Continental Flying Spur Sedan 2007" receives 33.59%, and other close variants of the Continental series dominate the rest of the top-5.

The Audi TTS example, on the other hand, reveals that correct Make classification does not always guarantee strong downstream predictions. While "Audi" is detected with high confidence (99.29%) and the Type prediction "Coupe" achieves a solid 36.56%, the Model head struggles because the true Model "Audi TTS Coupe 2012" appears in the top-5 but with only 4.17% confidence.

In the heatmaps of the fully misclassified samples shown in Figure 5, a consistent pattern becomes visible: whenever the image is captured from the side or slightly from behind (Samples 1–3), all three heads tend to focus on the central region of the vehicle. The Suzuki example shows this clearly. The image is taken entirely from the rear, and the Make head concentrates on the middle of the car. However, the Type head fixates on the large "Rockstar Energy" sticker on the right side of the trunk, which is completely unrelated to the vehicle's actual structural features. The Model head performs even worse, directing its attention away from the car entirely and focusing mainly on the left-hand background.
The Aston Martin sample provides a different perspective, as the photo is taken from above and toward the front. The Make and Type heads focus primarily on the bonnet, which is a reasonable region for high-level recognition. However, the Model head barely attends to the vehicle, only touching the right headlight and parts of the road next to the car.

In Figure 6, the heatmaps for the samples where only the Make head predicted the correct class are shown. For the first four samples, each photographed from the front or slightly from the side, all three heads concentrate their attention on the central area of the car's front section.
The only notable deviation appears in the Audi TTS example, where the image is captured from the rear. The Make and Type heads still fixate on the central region of the rear end, which is consistent with their behavior in similar viewpoints.

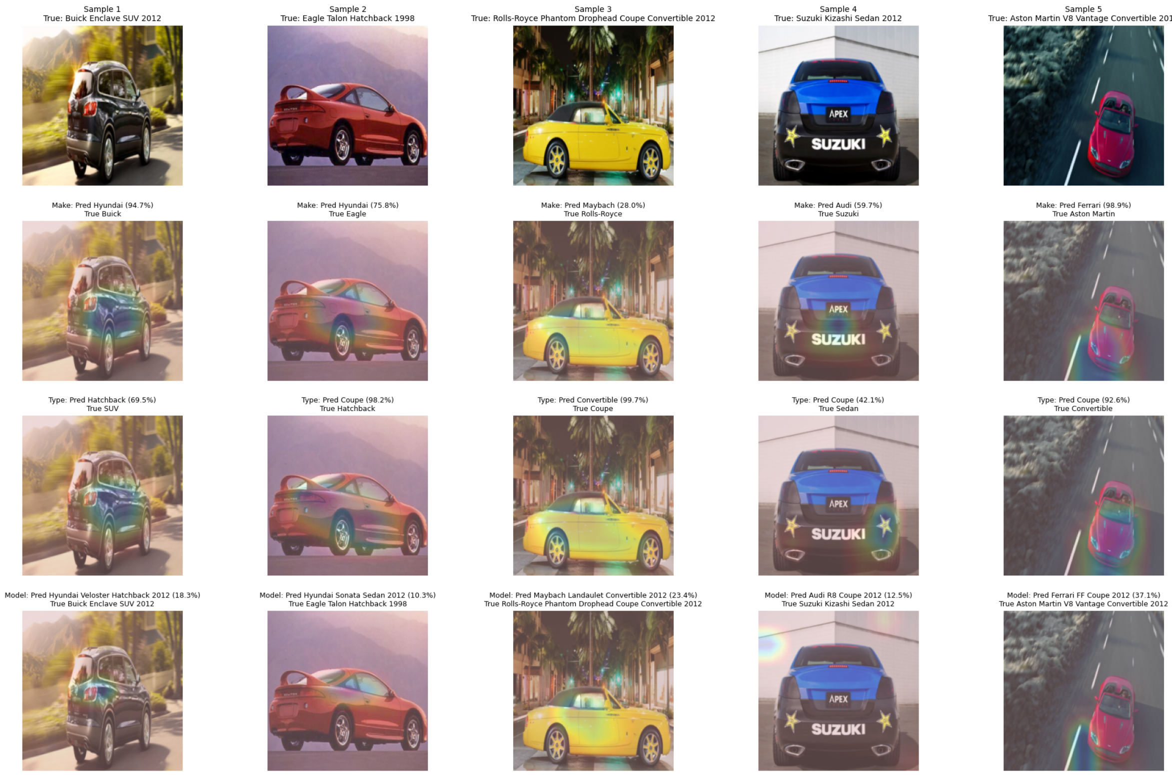

Figure 5: Five Stanford-Cars test images and the heatmaps showing on which image part our model (3 head, HLS, LRS) focused on when producing labels for each head. All 3 head predictions are wrong. The correct label the heatmaps for each head are displayed under each image.

However, the Model head diverges completely and focuses primarily to the left rear light and even to the skyscrapers in the background.

5 Discussion

In the following section, we interpret the results of our project and critically examine the factors that shaped the model's performance. We outline which components of our approach contributed meaningfully to improvements and where limitations were faced.

5.1 The "Frankenstein Car" Problem

- Discuss Hierarchical Consistency.
- In early experiments, consistency was low (90%). The model would predict "Toyota" (Make) and "Honda Civic" (Model).
- By Experiment 9 (HLS), consistency reached 96.6%, and finally 97.66% with the scheduler. This proves the model learned the taxonomy, not just pixel patterns.

A question in hierarchical classification is whether a model learns the underlying taxonomy or whether it simply memorizes pixel-level patterns without understanding the semantic relationships between labels. Early versions of our model frequently produced predictions where the make, type, and model belonged to different, incompatible vehicles. For example, the make head might output "Toyota", while the model head simultaneously predicted "Honda Civic". Such inconsistencies mean that each head is treated as an isolated classification task.
In the initial experiments, hierarchical consistency remained relatively low, hovering around 90%, meaning that in roughly one out of ten images, the predicted model did not belong to the predicted make. By introducing Hierarchical Label Smoothing (HLS) the consistency improved substantially. By Experiment 9, the model reached 96.6% consistency which demonstrates a marked improvement in aligning predictions across heads. The final experiment, which integrated

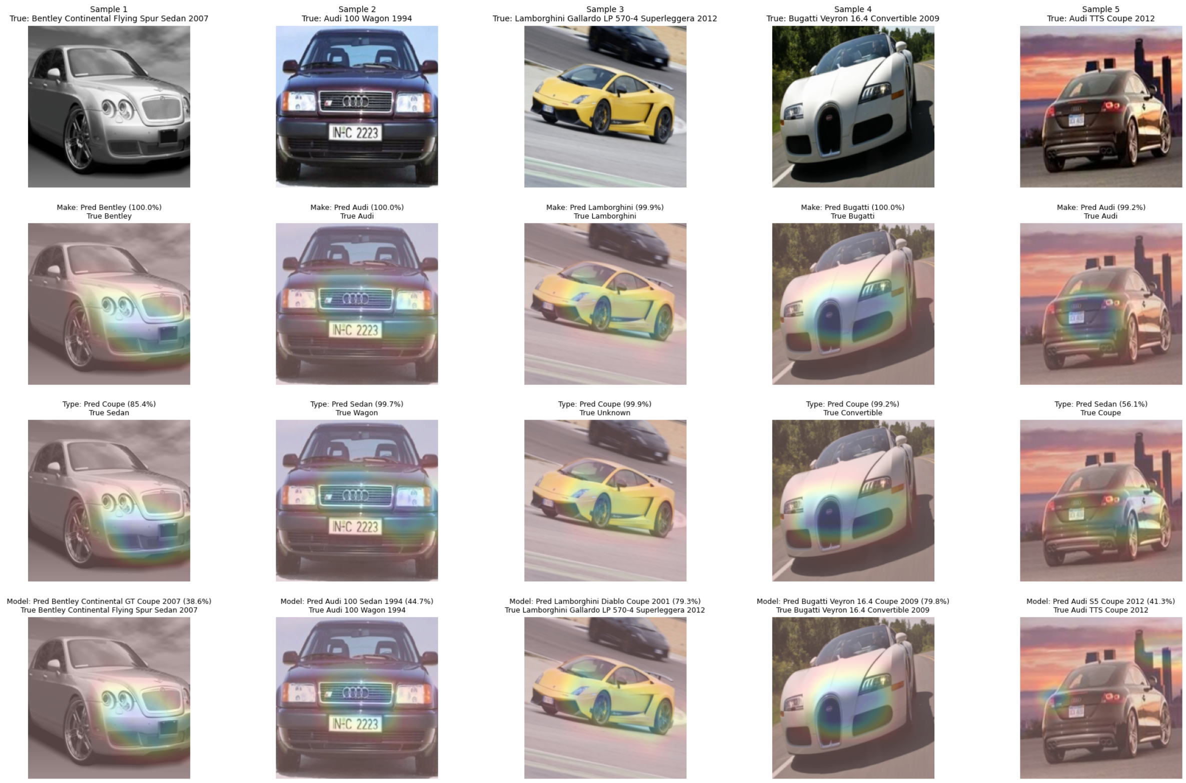

Figure 6: Five Stanford-Cars test images and the heatmaps showing on which image part our model (3 head, HLS, LRS) focused on when producing labels for each head. Only the make head's prediction is correct. The correct label the heatmaps for each head are displayed under each image.

both HLS and a cosine learning rate scheduler, further increased consistency to 97.66%.
This progression shows that the model was no longer merely fitting to visual textures or superficial cues but had begun to internalize the hierarchical taxonomy of make, type, and model. The reduction of incompatible predictions illustrates that the network learned meaningful relationships between car brands and their associated models which resulted in coherent outputs.

5.2 The Role of "Type" as Scaffolding

- Analyze why 3_head_curriculum eventually beat 2_head_curriculum.
- The "Type" head (Sedan, SUV, Coupe) provides Intermediate Scaffolding. It is easier to learn than "Model" but provides more structural information than "Make." It bridges the semantic gap.

A key insight from our experiments is the importance of the “Type” head (e.g., Sedan, SUV, Coupe) as an intermediate level of supervision. When comparing the two-head curriculum (Make + Model) to the three-head setup (Make + Type + Model), the latter consistently achieved higher accuracy and stability. This is because Type acts as a bridge because it is easier to learn than the fine-grained model classification but more informative than the broad make category.
By giving the model an additional intermediate task, the network receives structured guidance that reduces the complexity of learning the full 196-class model label. The Type head therefore provides scaffolding that helps the model organize visual information hierarchically. This effect explains why the three-head curriculum outperformed the two-head version in both accuracy and consistency.

5.3 The "Free Lunch" of Training Schedules

- Discuss the final experiment (LR Scheduler).
- ou gained 1.5% accuracy (87.9% $\rightarrow$ 89.07%) just by changing the learning rate schedule. This indicates the architecture was sound, but the optimizer needed "fine-grained" control to settle into the sharp minima of the loss landscape.

Another finding was how strongly the learning-rate schedule influenced performance. After the architecture and curriculum strategy were already working well, switching to a learning-rate scheduler brought an additional improvement of roughly 1.5 percentage points (from 87.9% to 89.07% model accuracy) without modifying the model or adding new data.
This highlights that the model was already capable of achieving higher performance, but the optimizer was not navigating the loss landscape efficiently. The scheduler allowed the network to make larger exploratory steps early on and gradually settle into more stable minima toward the end of training. In that sense, learning-rate scheduling was a simple change that is responsible for additional accuracy while keeping training cost and model complexity unchanged.

5.4 Robustness via Inference Ensembling

Explain how TTA serves as a proxy for measuring model robustness, argue that this proves our model is robust and "production-ready"

Test-time augmentation (TTA) provided an additional perspective on the robustness of the trained model. By averaging predictions over several augmented versions of each test image, such as horizontal flips or cropped resizes, the model effectively performs a lightweight ensembling step during inference. This reduces the influence of single-view biases, such as lighting, framing, or minor pose variations.
In our experiments, the TTA results demonstrated that the model maintained highly stable predictions under these perturbations which confirms that it had learned generalizable visual features rather than overfitting to specific image conditions. Although the absolute performance gain from TTA was modest, the consistency across augmented views indicates that the model is resilient and can be considered “production-ready” for real-world scenarios.

6 Conclusion

The pursuit of fine-grained vehicle classification in this project demonstrated that while a powerful backbone like ResNet-50 is essential for feature extraction, overcoming the challenge of subtle visual differences requires more than just raw model capacity; it demands a structured learning methodology. Our initial single-head baseline, while functional, failed to capture the intrinsic make-type-model hierarchy.
The key to success lay in adopting the Shared-Bottom Multi-Task Learning architecture, specifically the three-head design. The "Type" head proved invaluable, acting as a semantic bridge that provided crucial intermediate scaffolding for the fine-grained "Model" prediction, resolving initial task interference when combined with Curriculum Learning. Furthermore, the introduction of Hierarchical Label Smoothing (HLS) and a fine-tuned Learning Rate Scheduler were critical for squeezing out the final performance gains and enforcing logical coherence, driving prediction consistency to 97.66% and virtually eliminating the "Frankenstein Car" problem.
Achieving a final Top-1 accuracy of 88.57% (with TTA) validates that our strategy was sound. This project confirms that for complex, hierarchical tasks, structuring the learning process and enforcing taxonomy coherence is paramount, yielding a final model that is not only accurate but also highly resilient and logically robust against real-world variations.

7 Headings: first level

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. See Section 7.

7.1 Headings: second level

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

$$\xi_{ij}(t) = P(x_t = i, x_{t+1} = j|y, v, w; \theta) = \frac{\alpha_i(t) a_{ij}^{w_t} \beta_j(t+1) b_j^{v_{t+1}}(y_{t+1})}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij}^{w_t} \beta_j(t+1) b_j^{v_{t+1}}(y_{t+1})} \tag{1}$$

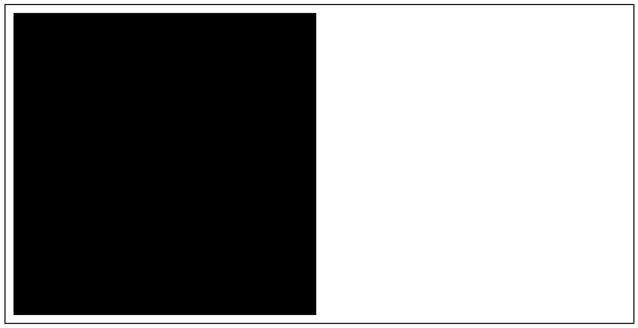

Figure 7: Sample figure caption.

7.1.1 Headings: third level

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Paragraph Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

8 Examples of citations, figures, tables, references

8.1 Citations

Citations use `natbib`. The documentation may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Here is an example usage of the two main commands (`citet` and `citep`): Some people thought a thing [Kour and Saabne, 2014a, Hadash et al., 2018] but other people thought something else [Kour and Saabne, 2014b]. Many people have speculated that if we knew exactly why Kour and Saabne [2014b] thought this...

8.2 Figures

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi. See Figure 7. Here is how you add footnotes. [2] Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

8.3 Tables

See awesome Table 5.

The documentation for `booktabs` ('Publication quality tables in LaTeX') is available from:

[2] Sample of the first footnote.

Table 5: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

`https://www.ctan.org/pkg/booktabs`

8.4 Lists

- Lorem ipsum dolor sit amet
- consectetur adipiscing elit.
- Aliquam dignissim blandit est, in dictum tortor gravida eget. In ac rutrum magna.

References

George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014a.

Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.

George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014b. doi:10.1109/SOCPAR.2014.7008025.