Efficient Fine–Grained Vehicle Recognition under Class Imbalance

Chiara Tramarin†, Alessio Tuscano†

***Abstract*—Fine-grained vehicle recognition is an important problem in computer vision with direct applications to transportation and surveillance. The CompCars dataset exemplifies its challenges, with hundreds of classes, severe imbalance, and high visual similarity across models.**

We present a systematic study of Convolutional Neural Networks (CNNs) for both classification and verification on CompCars. We compare models trained from scratch with fine-tuned networks, and evaluate Cross-Entropy versus Focal Loss. We also examine the trade-off between accuracy and computational cost across architectures of different complexity.

Results show that fine-tuned networks consistently outperform those trained from scratch. InceptionV3 achieves the highest balanced accuracy, while ResNet18 provides the best compromise between accuracy and efficiency. Focal Loss yields only limited benefits, mainly for larger scratch-trained models. In verification, lightweight pretrained backbones such as Siamese ResNet18 learn robust embeddings and maintain strong ROC-AUC under hard conditions, underscoring the practical value of lightweight models.

This work brings a comparison with many different models even in the verification task, and sets a baseline for a proper study for computational cost in vehicle recognition.

***Index Terms*—CompCars, Convolutional Neural Networks, Vehicle Recognition, Fine-grained Classification, Verification**

I. Introduction

Convolutional neural networks (CNNs) have become the de–facto standard for image classification and verification tasks, showing excellent performance across many domains [1]–[3]. In real–world applications such as intelligent transportation systems, surveillance, and automotive analytics, the ability to distinguish vehicle make and model from images is critically important and challenging.

Vehicle recognition is particularly hard due to datasets like CompCars [4], which include hundreds of classes, extreme class imbalance, large intra–class variation, and subtle inter–class differences. Many existing approaches apply standard CNNs with cross–entropy loss and focus only on classification, often overlooking how imbalance and embedding-based verification affect performance.

In this work, we tackle both classification and verification tasks on the CompCars [4] dataset. We systematically compare Cross–Entropy and Focal Loss in unbalanced conditions, and we analyze the trade-off between prediction performance and computational cost. We show that relatively small pretrained networks can achieve strong results competitively with heavier models, making them appealing for resource-constrained deployment. The insights provided are directly applicable in real–world settings where computational budget and model robustness matter.

Contributions. The key contributions of this paper are:

- A systematic comparison of Cross-Entropy vs. Focal Loss in both classification and verification under class imbalance.
- A cost–performance analysis demonstrating that lightweight pretrained architectures can compete with heavier models in both accuracy and embedding quality.
- Evidence of generalization and robustness in verification embeddings learned under resource constraints, making the proposed methods usable in real deployment contexts.

The remainder of the paper is organized as follows. Section II reviews related work. Section III introduces the dataset and preprocessing pipeline. Section IV describes the network architectures and loss functions. Section V presents the experimental results and analysis. Finally, Section VI offers concluding remarks, lessons learned, and future directions.

II. Related Work

The CompCars dataset was first introduced by Yang et al. to support fine-grained car recognition tasks including classification and verification [4]. It contains both web-nature and surveillance images, annotated with make, model, viewpoint, and other attributes, providing a realistic and challenging benchmark. Subsequent works have extended classification across makes, models and hierarchical levels on CompCars, revealing limitations in dealing with dataset imbalance and verification tasks [5].

In the classification domain, Valev et al. performed a broad evaluation of CNN architectures (ResNet, Inception, MobileNet) on vehicle datasets, comparing fine-tuning vs training from scratch, though they did not deeply address class imbalance or verification performance. More recently, Zhang et al. introduced a lightweight attention–augmented CNN with regularized fine–tuning for fine–grained car classification, demonstrating that compact models can attain strong performance in constrained settings. These works show that pretrained CNNs provide strong baselines, but they leave open issues in handling imbalance and supporting verification tasks.

On the verification side, Siamese architectures have been applied to vehicle matching. Wang et al. developed a local feature–aware Siamese matching model that focuses on informative parts to mitigate viewpoint variation in vehicle re-

†Department of Physics and Department of Information Engineering, University of Padova, Italy. Emails: chiara.tramarin.2@studenti.unipd.it, alessio.tuscano@studenti.unipd.it

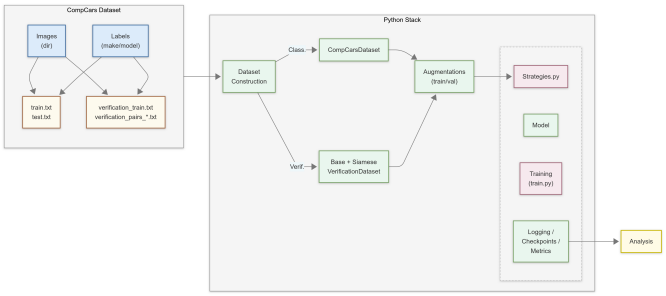

Fig. 1: Processing and training pipeline. Overview of dataset preparation, augmentation, and training flow for classification and verification tasks.

identification datasets [6]. De Oliveira et al. proposed a two-stream Siamese network that fuses coarse vehicle shape and license–plate patches, improving matching even under non-overlapping camera setups [7]. These models illustrate that embedding learning benefits from attention to parts or fusion, though they often assume balanced or simplified verification settings.

Despite these advances, few works systematically contrast classification and verification under class imbalance, or rigorously analyze the impact of alternative loss functions (e.g., Focal Loss) and computational cost trade–offs.

In this work, we complement existing literature by providing:

1) A unified evaluation of classification and verification on CompCars, explicitly addressing class imbalance and comparing Cross–Entropy versus Focal Loss in both tasks.
2) A cost–performance analysis showing that lightweight pretrained architectures can achieve robust embeddings and competitive accuracy more efficiently than deeper scratch–trained models.

III. Processing Pipeline

Before proceeding we explain a few terms used both here and in the related code:

- `task`: the classification or verification task
- `target`: *make* or *model*, the two possible car labels in the dataset for every `task`.
- `strategy`: the set of model and parameters called and passed for training. "Acronym" column in tables I, II, III, IV refers to all `strategies` in the code.

Our processing pipeline follows a standard supervised learning framework adapted to the CompCars dataset. The overall data processing, training and analysis flow is illustrated in figure 1.

IV. Signals and Features

The CompCars dataset has already been described thoroughly in the respective paper [4], however we will briefly remind some of its characteristics to have a better perspective of the choices made during the pipeline setup. Figure 2 shows us that the class distribution is highly imbalanced. Dataset augmentation via GANs could be an interesting future direction. One could also apply dynamic augmentation only to rare classes to simulate a more balanced dataset, but designing such a targeted scheme would require a separate study and was outside the scope of this work. Instead, we applied dynamic augmentation across the entire dataset to enhance feature generalization. To address class imbalance, we compare categorical cross-entropy (CE) with focal loss (FO).

In this study we only used a fixed target size, but in principle one could experiment with different ones and find the gain associated to increasing it. We used a target size of (224×224) for all models for classification but InceptionV3 since this model was intended for (299×299) images. Instead for verification we used a target size of (192×192). These sizes, as well as batch size, were specifically tuned to both cache the verification set in RAM and saturate at the same time the GPU VRAM. Also we note that (224×224) is the same target size as most of the classification benchmarks found online, as well as the common target for ImageNet dataset used for pretraining of the fine–tuned models.

All the test runs were computed on a machine with the following primary specifications:

- CPU: Intel Core i7-4790k,
- RAM: 32GB of DDR3, 1600Mhz
- GPU: Nvidia RTX 3050, 6GB VRAM

For classification, each image is associated with a one-hot encoded label representing either the vehicle *make* or *model*, and the network is trained in a standard multi–class setting. For verification, image pairs are sampled and labeled as either "match" (same class) or "non–match" (different class) depending on the chosen `target`. In this case, the Siamese network outputs embeddings for each image, and similarity scores (based on Euclidean distance) are used to train and evaluate the model. In both `tasks`, the chosen `strategy` produces the metrics required for analysis.

V. Learning Framework

We will now explore the processing pipeline in more detail. As shown in Figure 1, we relied on the official splits already prepared and extensively discussed in the original CompCars paper [4]. For classification we used the `train.txt` and `test.txt` files, covering $\sim 40,000$ web–nature images annotated over 75 *makes* and 431 *models*. For verification we used the `verification_train.txt` and the official `verification_pairs_*.txt` lists, which provide 20,000 pairs for each difficulty level (easy, medium, hard), evenly split between positive and negative examples. Then, given the choice of both `task` and `target` the dataset is constructed accordingly with the augmentation techniques provided by a given `strategy`. For simplicity we used the same augmentations [8] for all the prepared `strategies`:

- A.Resize: to chosen size.
- A.HorizontalFlip(p=0.5): 50% horizontal flip.
- A.RandomBrightnessContrast(p=0.2):
- A.Affine (p=0.3): scale between 90% and 110% of zoom, traslate up to 5% in orizontal/vertical, rotate

between $(-10, +10)$ degree, shear between $(-5, +5)$ and $(-2, +2)$ on x and y.

These augmentation provide a boost to our dataset while avoiding harming training by modifying the dataset too much. Then we normalize using either ImageNet (μ, σ) for transfer learning or the CompCars dataset (μ, σ) for scratch models (precomputed).

This modular pipeline allows us to easily switch between different `task`, `target`, `strategy` and hyper–parameters while maintaining a consistent experimental setup.

We experimented with multiple deep CNN architectures. For transfer learning, we chose the following pretrained models: InceptionV3 [3] classification only, ResNet18 [2], EfficientNet–B0 [9] for the verification `task` only. For custom models we developed two similar networks, both based on the classic ResNet architecture [2]. SimpleResNet is very similar to ResNet18 but with fewer blocks per layer and `Dropout(0.3)` layer in the final head for better generalization. Both SimpleResNet and ResNet18 have $\sim$ 11M parameters. SimpleResNetLarge is the same as SimpleResNet but with more blocks per layer, amounting to $\sim$ 21M parameters. For verification we primarily used pretrained models via fine–tuning, as we'll see in the results this is the only feasible method given our hardware and dataset. Classification training uses the AdamW optimizer with OneCycleLR scheduling, while verification uses AdamW optimizer with ReduceLROnPlateau for a more stable learning environment as the verification `task` is less stable than the classification one due to dataset construction differences. Early stopping based on validation performance is employed to prevent overfitting for the classification `task`.

Regarding the training setup, for classification we compare Cross–Entropy loss with Focal Loss [10], the latter specifically designed to mitigate class imbalance by down-weighting easy examples and emphasizing hard ones. For verification, Siamese networks are trained with binary cross-entropy over similarity scores, using embeddings from the penultimate ResNet layer and Euclidean distance to compute image similarity. Learning rates were tuned to achieve stable performance across all `strategies`; while a full grid search would have been ideal, it was not feasible given the computational cost.

VI. Results

In this section we report the experimental results for both the classification and verification tasks on the CompCars dataset. We first analyze the data distribution, then present classification performance on the *make* and *model* `tasks`, followed by an evaluation of the training stability and computational trade-offs. Verification results are reported in the final part of the section.

A. Dataset imbalance

Figure 2 illustrates the class distribution for both the *make* and *model* tasks across training and test splits. The dataset is clearly imbalanced: a few classes include thousands of samples, while many others are represented by only a handful of images.

Such skewed frequencies make standard accuracy an unreliable metric, as it can be dominated by the performance on majority classes. To address this, we evaluate models also with Balanced Accuracy, defined as the average recall across classes, which gives equal importance to frequent and rare classes alike. In addition, we report macro–F1 scores, which capture the trade-off between precision and recall in imbalanced conditions. These metrics provide a fairer assessment of performance than plain accuracy.

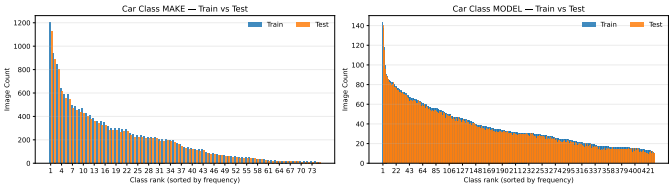

Fig. 2: Class distribution in the CompCars dataset. Sample counts for the *Make* (left) and *Model* (right) tasks in the training and test splits. The strong imbalance, with a few dominant classes and many underrepresented ones, motivates the use of balanced metrics and loss functions.

B. Classification performance

Tables I and II report the classification results for the *Make* and *Model* tasks, while Figure 3 provides a visual comparison of validation Accuracy and Balanced Accuracy across architectures.

The results clearly show that fine-tuned models outperform custom networks trained from scratch. InceptionV3 achieves the highest overall performance, while ResNet18 also delivers strong balanced accuracy. By contrast, custom architectures such as SimpleResNet and SimpleResNetLarge remain significantly behind despite having comparable or even larger parameter counts. These findings confirm the key role of transfer learning from ImageNet in fine–grained vehicle recognition.

The consistently high Top–5 accuracy further shows that even in misclassified cases, the correct label is usually among the top predictions, confirming the discriminative quality of the learned features.

The comparison between Accuracy and Balanced Accuracy highlights that accuracy systematically overestimates performance in this imbalanced setting. While accuracy values are consistently higher across all models, Balanced Accuracy reveals the true performance gap that accuracy alone would conceal. This confirms the necessity of adopting balanced metrics in highly imbalanced datasets such as CompCars. In addition, the training and validation loss curves (Fig. 4) exhibit smooth convergence without overfitting, indicating the reliability of the chosen optimization setup.

C. Cross-Entropy vs. Focal Loss

The comparison between Cross–Entropy (CE) and Focal Loss (FO) highlights how the impact of class imbalance

Acronym	Arch	Params (M)	Input (px)	Loss	Batch	Epochs	Best Ep	LR	Time/Ep [s]	Val Acc	Bal Acc	Top-5	F1
Mk_Inc_CE_B64	Inc	25.3	299×299	CE	64	45	40	1×10^{-3}	195.2	0.985	0.971	0.967	0.841
Mk_Inc_FO_B64	Inc	25.3	299×299	Focal	64	45	44	1×10^{-3}	195.1	0.969	0.965	0.935	0.796
Mk_Res18_CE_B256	Res18	11.2	224×224	CE	256	45	41	5×10^{-5}	54.8	0.962	0.930	0.932	0.828
Mk_Res18_FO_B256	Res18	11.2	224×224	Focal	256	45	43	5×10^{-5}	56.0	0.932	0.913	0.889	0.804
Mk_SRes_CE_B256	SRes	11.2	224×224	CE	256	45	39	1×10^{-3}	52.0	0.856	0.833	0.817	0.626
Mk_SRes_FO_B256	SRes	11.2	224×224	Focal	256	45	44	1×10^{-3}	51.5	0.808	0.795	0.711	0.540
Mk_SResL_FO_B256	SResL	33.3	224×224	Focal	256	75	73	1×10^{-3}	122.7	0.812	0.798	0.746	0.566
Mk_SResL_CE_B256	SResL	33.3	224×224	CE	256	45	43	2×10^{-3}	123.1	0.787	0.772	0.611	0.472

TABLE I: Make classification results. Fine–tuned models (blue) outperform custom architectures trained from scratch (white). Training used AdamW optimizer, OneCycleLR scheduler, label smoothing = 0.1, and early stopping.

Acronym	Arch	Params (M)	Input (px)	Loss	Batch	Epochs	Best Ep	LR	Time/Ep [s]	Val Acc	Bal Acc	Top-5	F1
Md_Inc_CE_B64	Inc	26.0	299×299	CE	64	45	44	1×10^{-3}	199.5	0.983	0.980	0.957	0.865
Md_Inc_FO_B64	Inc	26.0	299×299	Focal	64	45	44	1×10^{-3}	199.4	0.969	0.965	0.935	0.796
Md_Res18_CE_B256	Res18	11.4	224×224	CE	256	45	27	5×10^{-5}	52.8	0.941	0.931	0.842	0.752
Md_Res18_FO_B256	Res18	11.4	224×224	Focal	256	45	43	5×10^{-5}	52.9	0.931	0.917	0.894	0.805
Md_SRes_CE_B256	SRes	11.4	224×224	CE	256	45	33	1×10^{-3}	51.0	0.837	0.821	0.667	0.515
Md_SRes_FO_B256	SRes	11.4	224×224	Focal	256	45	44	1×10^{-3}	50.4	0.808	0.795	0.711	0.540
Md_SResL_FO_B256	SResL	33.5	224×224	Focal	256	75	73	1×10^{-3}	122.3	0.812	0.798	0.746	0.566
Md_SResL_CE_B256	SResL	33.5	224×224	CE	256	45	43	2×10^{-3}	121.4	0.787	0.772	0.611	0.472

TABLE II: Model classification results. Same setup as Table I. Fine–tuned networks (blue) achieve higher balanced accuracy and F1–scores, while custom scratch-trained models lag behind.

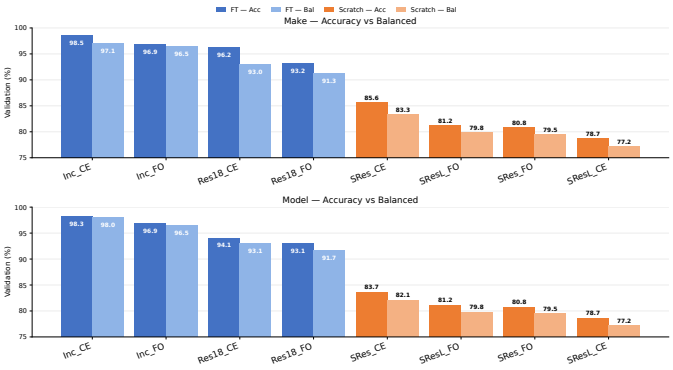

Fig. 3: Validation Accuracy and Balanced Accuracy across models. Results for the *Make* (top) and *Model* (bottom) classification tasks. Accuracy systematically overestimates performance, while Balanced Accuracy provides a fairer evaluation under imbalance.

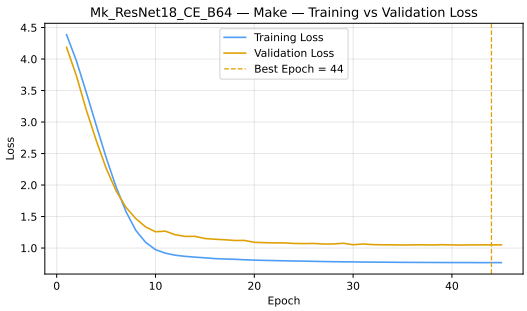

Fig. 4: Training and validation loss curves for ResNet18 (*Make*, Cross-Entropy). Both losses decrease smoothly, with the best validation performance at epoch 44, indicating stable convergence without overfitting.

interacts with model capacity. As shown in Figure 5, Focal Loss leads to modest improvements in Balanced Accuracy for larger networks, particularly SimpleResNetLarge, where it helps mitigate the dominance of majority classes. However, this benefit is not universal: in smaller custom architectures (e.g., SimpleResNet), FO reduces performance, likely because the additional focus on hard or minority examples increases gradient variance and destabilizes learning in low–capacity models.

Interestingly, for fine–tuned models such as ResNet18 and InceptionV3, Focal Loss produces results very close to Cross-Entropy, with no consistent gains. This suggests that the robustness already provided by pretraining on ImageNet largely absorbs the imbalance effects, leaving little room for FO to further improve generalization.

Taken together, these results indicate that Focal Loss is not a one–size–fits–all solution. Its effectiveness depends strongly on the architecture: it can provide measurable benefits on deeper or higher-capacity models trained from scratch, while being neutral or even detrimental for lightweight networks. For practical applications, this means that Focal Loss should be applied selectively, rather than as a default replacement for Cross–Entropy.

D. Computational trade–off

Figure 6 shows the trade–off between accuracy and efficiency across architectures. Fine–tuned models consistently dominate scratch–trained ones, combining higher balanced

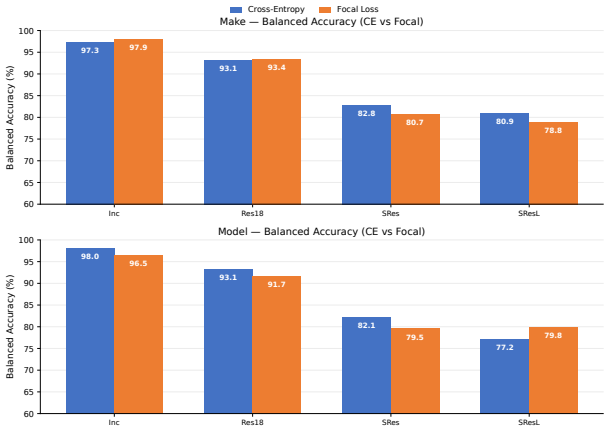

Fig. 5: Cross-Entropy vs. Focal Loss. Balanced Accuracy results for the *Make* (left) and *Model* (right) tasks. Focal Loss slightly improves larger networks but reduces performance in smaller ones, showing its architecture-dependent impact.

accuracy with lower training cost. Among them, ResNet18 stands out as the most efficient option, reaching strong accuracy with training times significantly shorter than InceptionV3.

InceptionV3 remains the most accurate model overall, though at a higher computational cost. Conversely SimpleResNetLarge, despite its size, fails to justify the added complexity, performing worse than both fine–tuned networks. These results highlight that transfer learning not only improves accuracy but also leads to a more favorable cost–performance balance, confirming that transfer learning is essential for cost–effective fine–grained recognition.

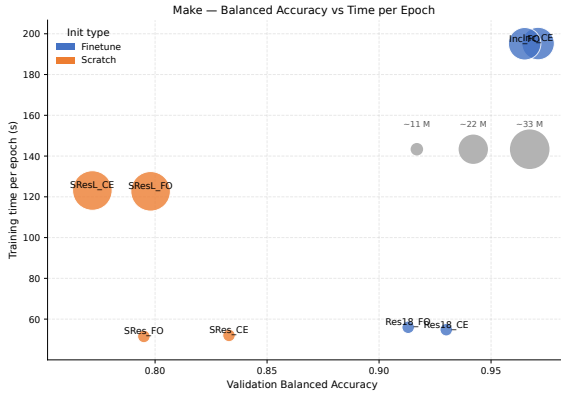

Fig. 6: Balanced Accuracy vs. training time per epoch for the *Make* classification task. Marker size reflects the number of parameters and color indicates initialization (blue = fine–tuned, orange = scratch). Fine–tuned models achieve higher accuracy at lower cost, while larger custom networks are less efficient.

E. Verification results

The verification task was evaluated with Siamese CNNs trained using contrastive loss. Results in Table III show that fine–tuned models clearly outperform the scratch–trained baseline. Lightweight backbones such as EfficientNet-B0 and ResNet18 achieve the best trade–off between accuracy and complexity, while the larger ResNet50 does not provide consistent benefits despite its higher parameter count. Among these, ResNet18 proved the most effective backbone, which motivated its use for the detailed evaluation in Table IV. The table confirms consistent results across difficulty levels, with performance gradually decreasing from easy to hard but remaining reliable overall.

Additional insights come from the ROC curves. Figure 7 illustrates how verification performance improves during training: the AUC rises from 0.79 at epoch 2 to 0.88 by epoch 20, with no further significant gains afterwards, indicating fast and stable convergence. Figure 7 compares results across difficulty levels at epoch 20, with best performance on easy pairs and lower AUC on harder ones. Despite this degradation, the clear separation from the random baseline demonstrates the robustness of the learned embeddings.

Finally, Figure 8 shows the distribution of embedding distances: positive pairs cluster tightly at low values, while negatives shift to higher distances, validating the quality of the embedding space learned by the Siamese network.

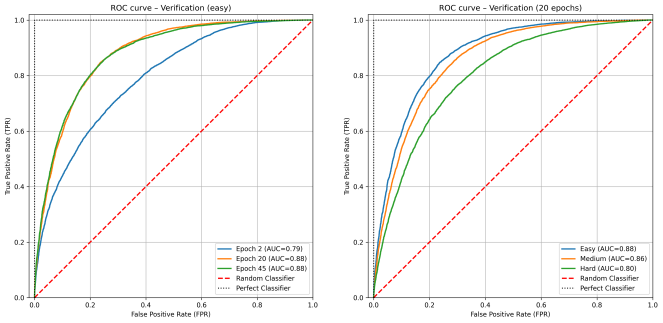

Fig. 7: ROC curves for the verification task with Siamese ResNet18. *Left*: Performance evolution across training epochs (*Make*, easy setting). *Right*: Comparison across difficulty levels at epoch 20.

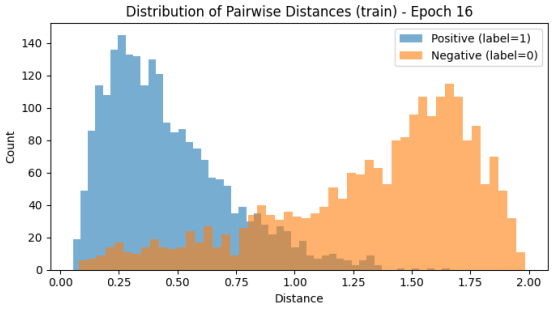

Fig. 8: Embedding distance distribution for Siamese ResNet50. Positive pairs cluster at lower distances, while negatives shift towards higher ones, demonstrating a well-structured embedding space.

Acronym	Arch	Params (M)	Input (px)	Loss	Batch	Epochs	Best Ep	Time/Ep [s]	Val Acc	Bal Acc	ROC-AUC	F1	Prec.	Recall
Vr_SRes18_CT_B128	SRes18	11.4	192×192	Contr.	128	75	43	49.6	0.799	0.799	0.874	0.810	0.767	0.858
Vr_EffB0_CT_B64	EffB0	4.7	192×192	Contr.	64	75	74	60.6	0.796	0.796	0.874	0.807	0.766	0.852
Vr_SRes50_CT_B64	SRes50	28.8	192×192	Contr.	64	75	44	105.9	0.782	0.782	0.861	0.789	0.765	0.814
Vr_SRes18_CT_B128_SCR	SRes18	11.4	192×192	Contr.	128	75	24	48.8	0.639	0.639	0.688	0.550	0.731	0.441

TABLE III: Verification results on the Make task. Comparison of four Siamese CNNs. The scratch-trained model (white) underperforms compared to fine-tuned ones (blue).

Acronym	Arch	Params (M)	Input (px)	Loss	Batch	Epochs	Best Ep	Time/Ep [s]	Val Acc	Bal Acc	ROC-AUC	F1	Prec.	Recall
Vr_SRes18_CT_B128_MkE	SRes18	11.4	192×192	Contr.	128	75	43	49.6	0.799	0.799	0.874	0.810	0.767	0.858
Vr_SRes18_CT_B128_MkM	SRes18	11.4	192×192	Contr.	128	45	38	50.2	0.790	0.790	0.863	0.798	0.768	0.831
Vr_SRes18_CT_B64_MkH	SRes18	11.4	192×192	Contr.	64	45	40	51.2	0.719	0.719	0.789	0.736	0.694	0.783
Vr_SRes18_CT_B128_MdE	SRes18	11.4	192×192	Contr.	128	45	27	49.8	0.797	0.797	0.874	0.805	0.773	0.840
Vr_SRes18_CT_B128_MdM	SRes18	11.4	192×192	Contr.	128	45	40	50.5	0.775	0.775	0.848	0.775	0.775	0.775
Vr_SRes18_CT_B128_MdH	SRes18	11.4	192×192	Contr.	128	45	32	50.5	0.721	0.721	0.788	0.739	0.694	0.790

TABLE IV: Verification results with Siamese ResNet18. Results for Make (top) and Model (bottom) verification, under easy, medium, and hard difficulty settings.

VII. Concluding Remarks

In this project we evaluated CNNs for fine–grained vehicle recognition on the CompCars dataset, addressing both classification and verification. Fine–tuned models consistently outperformed scratch–trained ones, and balanced metrics proved essential for assessing performance under strong class imbalance.

Our findings underline that transfer learning is not only crucial for accuracy but also for efficiency, making lightweight pretrained networks such as ResNet18 attractive candidates for real–world deployment, where computational resources are limited.

A. Challenges and Lessons Learned

During this project we gained hands–on experience with large–scale datasets and end-to-end deep learning workflows. We observed that small choices in dataset construction and preprocessing can markedly affect performance especially in verification. Moreover, in unstable training regimes, simpler scheduling (e.g., `ReduceLROnPlateau`) proved more reliable than aggressive schemes like `OneCycleLR`.

Our main technical challenge was building a modular, maintainable codebase that fully exploited limited hardware. Striking a balance between usability and performance was essential to iterate over many `strategies` efficiently on constrained resources (including a decade-old CPU).

B. Future Work

There is considerable room for improvement on both the original `tasks` and further extensions. One direction is to analyze the impact of image size on performance: smaller inputs could allow caching entire datasets in GPU memory, reducing CPU–GPU transfers and greatly accelerating training (for example we gained ∼2x–4x speedup by caching the validation set in RAM). Another is to explore the raw unfiltered dataset and the surveillance subset to better assess generalization in real-world conditions. Training a competitive verification model entirely from scratch also remains an open challenge, though currently too expensive computationally. Finally, imbalance-mitigation strategies such as class reweighting or targeted augmentation could further improve performance on underrepresented classes, complementing the loss–based approaches studied here.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, (Lake Tahoe, USA), pp. 1097–1105, December 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), pp. 770–778, June 2016.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), pp. 2818–2826, June 2016.

[4] L. Yang, P. Luo, C. Change Loy, and X. Tang, "Compcars: A comprehensive dataset for car recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, USA), pp. 3213–3221, June 2015.

[5] M. Buzzelli and L. Segantin, "Revisiting the compcars dataset for hierarchical car classification: New annotations, experiments, and results," *Sensors*, vol. 21, no. 2, p. 596, 2021.

[6] H. Wang, S. Sun, L. Zhou, L. Guo, X. Min, and C. Li, "Local feature-aware siamese matching model for vehicle re-identification," *Applied Sciences*, vol. 10, no. 7, p. 2474, 2020.

[7] I. O. de Oliveira, K. V. O. Fonseca, and R. Minetto, "A two-stream siamese neural network for vehicle re-identification by using non-overlapping cameras," 2019.

[8] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *arXiv e-prints*, 2018.

[9] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv*, vol. abs/1905.11946, 2019.

[10] T. Lin, P. Goyal, R. Girshick, K. He, and P. DollÃ¡r, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), pp. 2980–2988, October 2017.