Car make classification and verification using Snapshot ensembling

Lorenzo Martinelli†, Arina Ponomareva†, Laura Schulze†

***Abstract*—Striking a good balance between having an accurate model for common computer vision tasks and reducing the time and computational resources needed to train said model is an open challenge in modern computing. In this report, we explore snapshot ensembles as a way to enhance performance without incurring additional training cost, by applying them to car-related computer vision tasks using the CompCars dataset. To optimize both time and required computational resources, we take advantage of EfficientNet as a lightweight, well-established architecture, and leverage transfer learning. We test our model's ability to perform classification and verification based on car make. Combining snapshot ensembling with various data augmentation techniques, we are able to achieve test accuracies of 94.89% for the car make classification, and 83.40% for the car make verification. Our results highlight both the advantage of snapshot ensembling, even with short training periods, and the importance of data augmentation. This holds especially true for the verification task, which exhibits a significant increase in accuracy when an ensemble is used instead of a single, fully trained model.**

***Index Terms*—Convolutional Neural Networks, Image classification, Image verification, Snapshot ensembles, Transfer Learning.**

I. Introduction

Object classification is a classic problem in machine learning with countless real-world applications, especially in the field of computer vision. As such, it is an ongoing subject of studies seeking to optimize and improve solutions further and further. A similar problem is verification, where a network has to decide if a pair (or more) of objects are part of the same category. These tasks are commonly approached with complex Convolutional Neural Network (CNN) architectures. Many complex models are available with pre-trained weights, which are obtained from training with large image databases. One such example is ResNet [1], which is trained using the ImageNet dataset [2]. These pre-trained models can then be used as starting point and fine-tuned for a specific task, saving training time and resources and improving performance with what is commonly known as transfer learning.

Along with the development of more advanced architectures, the so-called ensemble approaches can be employed to further improve the accuracy of the predictions. Despite their advantages, ensembles feature one main limitation, being the time and resources needed to train the single elements that make up an ensemble. Snapshot ensembles [3] aim to mitigate that problem; they are constructed from "snapshots" taken at local minima during the training process of a single model. At test time, the snapshots are combined into an ensemble by averaging their predictions, resulting in improved accuracy without any additional training cost.

In this report, we combine both transfer learning and snapshot ensembling to tackle a specific computer vision task: classifying and verifying car images based on the car make. Using a complex, pre-trained CNN architecture, EfficientNet-B0 [4], we retrain the last layers on the CompCars dataset [5] and extract model snapshots during training to form an ensemble and enhance accuracy at test time. We compare our ensemble results to the performance of some models on the same dataset, as well as analyse how different snapshot selections for the final ensemble affect the outcome. This approach enables a high accuracy at comparatively low training cost, and demonstrates the ease with which a functioning ensemble can be extracted during the fine-tuning of a pre-trained model. The developed pipeline can easily be extended to more fine-grained classification of e.g. car models, or applied to problems with different object types altogether.

The main contributions of this report can be summarized as follows:

1) We exploit snapshot ensembles to create a set of models for car-make classification from a pre-trained CNN architecture. In doing so, we greatly reduce the time expenditure necessary for training, while retaining high levels of accuracy.
2) We extend the same strategy to a car make verification task; we create an ensemble of Siamese Networks to verify whether two images show the same car make.
3) For both tasks, we vary the number of snapshots in the ensemble, and explore the effect of data augmentation techniques on the performance.

This report is structured as follows. In Section II, we provide a brief overview of related literature. Section III gives a high-level description of our approach. The data processing and model architecture are detailed in Sections IV and V, respectively. The results are presented and discussed in Section VI; Section VII concludes our findings.

II. Related Work

Ensembling approaches are known to achieve great accuracy, but preparing multiple networks for an ensemble comes at a high training cost. [3] introduces snapshot ensemble as a time- and resource-saving approach. Following prior works [6], [7], they adapt a cyclic annealing schedule for the learning

†Department of Physics and Astronomy, University of Padova, email:
{lorenzo.martinelli.1, arina.ponomareva, lauramarie.schulze}@studenti.unipd.it

rate, and save the model parameters as "snapshots" at the end of each cycle, leveraging different local minima. In their paper, they show how this technique produces ensembles that surpass single fully trained models without any additional training cost. Due to its effectiveness in achieving high accuracy without increased training cost, we explore the application of this method to image-based car make classification. Furthermore, we extend the approach to car make verification as well. To do so, we use the CompCars dataset, a comprehensive dataset designed for various car-related computer vision tasks. It was first introduced in [5], which additionally provides baseline results for both make and model classification, as well as model verification with CNNs. Rather than performing fine-grained classification and verification by model, our analysis of snapshot ensembling is focussed on the simpler tasks based on car make only.

To further enhance training efficiency, we employ transfer learning with a pre-trained network. As a common approach to reduce training time and improve performance for specialised tasks, pre-trained complex CNNs are widely available and can easily be adapted to the problem of interest. For instance, the baseline results in [5] are obtained with the OverFeat model, pre-trained on ImageNet and fine-tuned with the CompCars dataset. We base our model on the more recent EfficientNet-B0 instead, which is similarly pre-trained on ImageNet. The EfficientNet family is designed to optimize model scaling, and has demonstrated high efficiency and accuracy on image classification tasks, including transfer learning datasets [4].

By combining snapshot ensembles with transfer learning on the CompCars dataset, our goal is to evaluate whether this technique can provide accuracy improvements while maintaining low computational costs.

III. Processing Pipeline

With the overarching goal of exploring the potential of snapshot ensembles, the project is focussed on two distinct applications, car make classification and verification. In the following, we introduce the processing pipelines developed for each task.

A. Car make classification

Our strategy for the make-based classification of car images provided by the CompCars dataset consists of three main stages; data preparation, model training with snapshot extraction, and ensemble construction.

Data preparation. The car images and corresponding car make labels are extracted from the CompCars dataset, and the classes are categorized based on the amount of available samples. To mitigate class imbalance in the training set, we explore different strategies, such as undersampling the most common classes and performing data augmentation through random image transforms. All images are resized and normalized to be compatible with the EfficientNet model.

Model Training and Snapshot extraction. The starting point is the EfficientNet-B0 model with pre-trained weights. To fine-tune it for car classification, the last three stages are retrained with a focal loss function. As described in [5], a cyclical learning rate is used. At the end of each cycle, the current model weights are saved as a "snapshot". The initial learning rate is a hyperparameter to be fine-tuned, as well as the γ-parameter of the focal loss.

Ensemble construction. When training is concluded, we have obtained a number of snapshots, including the final trained model. The snapshots (or a subset thereof) are combined into an ensemble at test time, by taking the average of the outputs as prediction. The number of snapshots to form the final ensemble are subject to investigation.

B. Car make verification

The second goal consists of reliably verifying whether or not two car images depict cars of the same make. Our approach can again be divided into three similar stages.

Data preparation. Besides the car images, the CompCars dataset also provides a list of image pairs for verification tasks, and labels based on whether or not the same car model is depicted. We adjust these labels for car make instead of car model verification. The images themselves are prepared the same way as for the classification task, and image augmentation strategies are explored.

Model Training and Snapshot extraction. Using again the pre-trained EfficientNet-B0, we set the model up as a Siamese Network, taking two images at a time and producing an output vector for each. Training is conducted using contrastive loss with the cosine similarity as a measure of similarity between the output vectors. As with the classification, snapshots are saved at regular intervals during training.

Ensemble construction. The ensemble is constructed the same way as for the classification task. Snapshot outputs are again averaged.

IV. Signals and Features

The CompCars dataset contains car images from web-nature and surveillance-nature scenarios [5]. This project utilizes the web-nature subset of the CompCars dataset, which consists of images obtained from car forums, public websites, and search engines. The images are organized in a hierarchical structure based on car make, model, and year, and feature various attributes and viewpoints. A total of 163 car makes with 1716 car models are represented.

As is common in real-world datasets, a severe class imbalance is present. Two different approaches are taken to tackle this imbalance:

- Undersampling more common classes, and applying standard data augmentation through image transforms to all (from here on referred to as **standard augmentation**).
- Applying different levels of data augmentation depending on the number of samples per class, without undersampling (referred to as **dynamic augmentation**).

For standard augmentation, classes with over 2000 images in the dataset are reduced by 50%, classes containing between 500 and 2000 images are reduced by 25%, all other classes remain in full. The image transforms applied are a

combination of a random horizontal flip and a random affine transformation.

For the dynamic augmentation, the classes are divided into three categories, based on the number of samples s they contain. Based on these categories, the following sets of random transformations are applied:

- $s < 500$: horizontal flip, rotation, color jitter, affine transformation.
- $s \in [500, 2000)$: horizontal flip, color jitter.
- $s \geq 2000$: no transformations applied.

To ensure compatibility with the model, all images are resized to $224\,\text{px} \times 224\,\text{px}$ and their channels (corresponding to RGB colors) are renormalized.

CompCars also provides mappings of positive and negative image pairs for car model verification tasks, with three sets of different difficulties. Positive refers to pairs of the same model, negative to pairs of different models. Here, a combination of the easy and medium sets is used, consisting of 40000 pairs total. To adjust the model-based pairs for make verification, the corresponding negative pairs of different models from the same make were relabelled as positive pairs. This yields an overall distribution of 20256 positive and 19744 negative pairs. The images are augmented through a random combination of horizontal flip and affine transformation, and are again resized and normalized to be transformed into compatible input vectors for EfficientNet.

The fully processed dataset for the classification task with undersampling consists of 104651 samples, and 136726 samples without undersampling. In both cases they are split into training (80%), validation (10%) and test set (10%). The 40000 pairs for the verification task are split identically.

V. Learning Framework

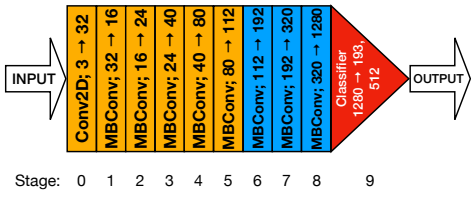

Fig. 1: Block diagram of EfficientNet-B0. The numbers next to the layer type represent the corresponding input and output features. The blue stages and the red head are retrained for our tasks, while the orange stages retain their pre-trained weights.

As highlighted in section III, the core architecture used in this project is EfficientNet-B0. The architecture has been developed with the explicit goal of having a robust and reliable model that uses a lower number of trainable parameters compared to other CNN architectures. A simple architecture scheme can be seen in Figure 1, which illustrates the different "stages" of EfficientNet. The first stage (Stage 0) is a simple convolutional layer, enriched by a batch normalization layer and the usage of the Sigmoid Linear Unit (SiLU) function. Stages 1 through 8 take advantage of a so-called Mobile Bottleneck Convolutional Layer (MBConv). Its main features are summarized in 2. In particular, there are two elements of interest:

i **Depth-wise Convolution Layer (DWConv)**: This layer uses separate filters operating on single channels, instead of a single filter operating through multiple channels. This greatly reduces the number of trainable parameters, and is the main advantage of EfficientNet.

ii **Squeeze-and-Excitation Layer (SE)**: This composed layer is used to understand which channels bear the most information by recalibrating channel-wise feature responses, further increasing the power of a lightweight model for a lower computational price.

MBConv layers use the connection shortcut, where the output of the layer is summed to its input. This is only doable for those layers where the numbers of input and output features match.

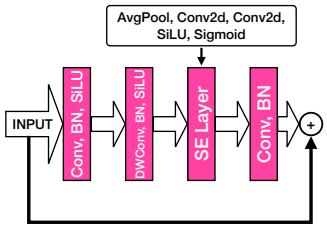

Fig. 2: Block diagram of the MBConv layer.

Instead of training the network from scratch, we use the pre-trained model available in `torchvision.models`, with weights that were obtained with the ImageNet dataset, and only retrain the last stages. The underlying architecture for the two tasks (classification and verification) is very similar, although there are some differences that will be highlighted in the respective subsections.

This architecture is combined with snapshot ensembling; a strategy which considers various "snapshots" of the same model throughout training. If the training loop and especially the learning rate are properly tuned, the single snapshots should be able to capture different features of the input, ultimately leading to higher accuracies. A central aspect to the snapshot ensemble technique is the periodic learning rate. By introducing a learning rate that decreases in time, allowing the model to approach a local minimum, only to sharply rise after a few epochs, we may be able to explore multiple local minima, which we can later combine into an ensemble. As suggested in [3], we use a shifted cosine learning rate $\eta(t)$, defined as:

$$\eta(t) = \frac{\eta_0}{2}\left[\cos\left(\frac{\pi \bmod (t-1, \lceil T/M \rceil)}{\lceil T/M \rceil}\right)\right], \qquad (1)$$

where t is the current epoch, T is the total number of epochs in the training loop, M is the total number of snapshots used, and η_0 is the initial learning rate.

A. Car make classification

For classification, the main task-specific features in the architecture and in the learning framework that have to be adjusted are the classifier head and the loss function. As for the former, EfficientNet-B0 already uses a fully connected dense layer as the final head. Therefore, the only required change consists of properly adjusting the number of final outputs, such that it matches the number of classes.

As for the loss function, we use the focal loss to further mitigate class imbalance, which is defined as

$$\mathcal{L} = -\sum_i \alpha_i \left(1-\hat{p}_i\right)^\gamma y_i \log \hat{p}_i \ . \tag{2}$$

The sum is carried out across all classes; y_i is the ground truth, $\hat{p}_i$ is the predicted probability of class i, γ is a control parameter, and α_i is a weight factor that takes into account the frequency of each class. Each element α_i is calculated as $\alpha_i = \frac{(\text{class count})_\text{i}^{-1}}{\sum_j (\text{class count})_\text{j}^{-1}}$. The hyperparameter γ in particular controls the strength of the focal loss: larger values of γ yield a stronger correction.

Before the full training of the model, we perform a short search for the two hyperparameters η_0 and γ, inspecting the range $\eta_0 \in \{0.01, 0.001\}$ and $\gamma \in \{2, 5\}$. We choose the combination that yields the highest validation accuracy after 15 training epochs and continue to use these values throughout the following tasks.

B. Car make verification

For this task, setup EfficientNet-B0 as a Siamese neural network. The idea is to use two networks with shared sets of weights, which we implement by giving the network two inputs at a time, and comparing how it processes similarities and differences between them. To achieve this, the classifier head and loss function are again modified. Instead of having a dense layer with 163 output features as classifier head, we choose an embedding size of 512 of output features, allowing for the capture of more complex features. The embeddings resulting from two inputs are put through a scoring function evaluating their similarity, which is import for defining loss. In our case, the scoring function is the cosine similarity, which, for two arbitrary embeddings $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, is defined as follows:

$$\text{CS}(\boldsymbol{e}_1, \boldsymbol{e}_2) = \frac{\boldsymbol{e}_1^T \boldsymbol{e}_2}{||\boldsymbol{e}_1||_2 \cdot ||\boldsymbol{e}_2||_2}. \tag{3}$$

This score, or "distance", is then used for the contrastive loss:

$$\mathcal{L} = y_i D^2 + (1-y_i)\min(0, (h-D)^2), \tag{4}$$

where D is the distance obtained through the cosine similarity, h is the margin, and y_i is the label. There are two things worth nothing:

- With the cosine similarity as metric, high values correspond to similar vectors, which might seem counterintuitive when thinking in terms of "distance". Furthermore, note that the label $y_i = 1$ corresponds to a positive pair and $y_i = 0$ corresponds to a negative pair.
- The contrastive loss "pushes" negative pairs towards 0, instead of -1, not maximizing their distance compared to the positive pairs (which are "pushed" towards 1).

To assess the influence of this cosine-similarity scoring over the final results, we carry out an additional test, where we use the cosine distance $\text{CD}(\boldsymbol{e}_1, \boldsymbol{e}_2) = 1 - \text{CS}(\boldsymbol{e}_1, \boldsymbol{e}_2)$ as scoring function instead. In this case, the contrastive loss is defined as

$$\hat{\mathcal{L}} = (1-y_i)D^2 + y_i \min(0, (h-D)^2). \tag{5}$$

VI. Results

A. Car make classification

For the classification task, we start with a hyperparameter search to fine-tune the initial learning rate η_0 and focal loss parameter γ. Using the best parameters, we then continue to fully train a model and extract snapshots for two different data augmentation settings as described in Section IV; first the standard approach using basic augmentation for all images, then a differential augmentation approach to further mitigate class imbalance. The results of both approaches are presented and compared in the following.

The hyperparameter search for initial learning rate η_0 and focal loss parameter γ yields $\gamma = 2$ and $\eta_0 = 0.001$ as best combination. The full validation results for the tested combinations are summarized in Tab. 1.

TABLE 1: Accuracy on the validation set after a 15 epoch-training for various values of the initial learning rate η_0 and focal loss parameter γ; car make classification task. The most accurate combination is marked in **bold**

γ	η_0	Accuracy
2	0.01	53.07%
2	**0.001**	**55.47%**
5	0.01	53.48%
5	0.001	54.59%

After a full 30-epoch training using the found hyperparameter combination, we obtain 6 snapshots. The evolution of training and validation loss throughout the training epochs are depicted in 3. It clearly shows the effect of the cyclic learning rate as described in eq. (1); for every cycle of 5 epochs the losses decrease, then sharply increase at the beginning of the new cycle, reflecting the behaviour of the learning rate. The local minima correspond to the points where the model snapshots are extracted.

For testing, these snapshots are combined into ensembles of different sizes. An ensemble of size k is constructed using the last k snapshots. Tab. 2 lists the top-1 and top-5 accuracies for ensembles of different sizes and different data augmentation strategies. The best accuracies are achieved using an ensemble made of $k = 2$ snapshots trained with dynamic data augmentation, reaching a top-1 accuracy of 94.89% and a top-5 accuracy of 99.14%. Interestingly, the model trained with standard data augmentation reaches its

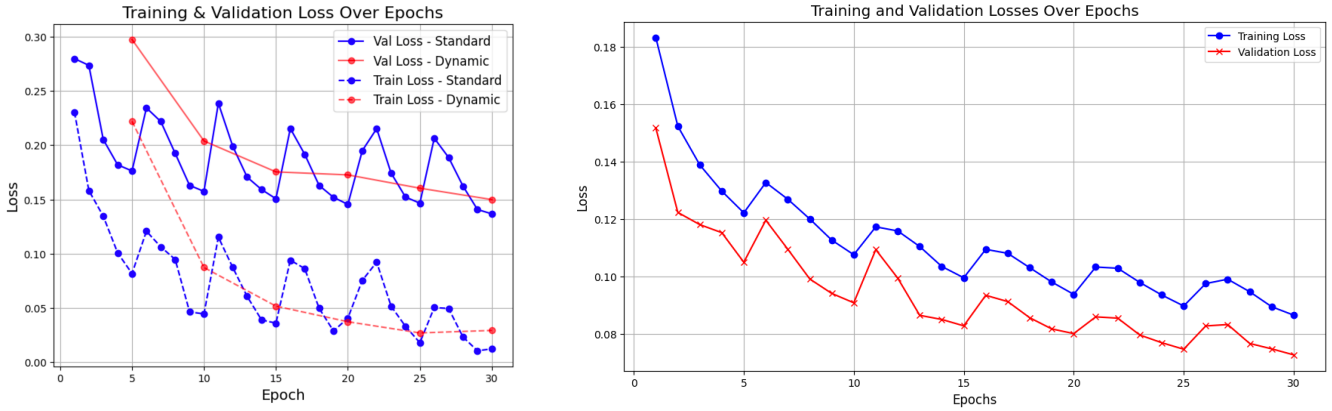

Fig. 3: The training and validation loss of the model throughout the 30 training epochs for the classification (left) and verification (right) tasks. In the classification task, both the standard and dynamic data augmentation approaches are showcased, with the former being represented by the evolution of the losses during each epoch to highlight the cyclical behavior and the latter being shown only once every five epochs to display the actual progress of the loss.

best accuracy level for $k = 4$, which is higher than the optimal number of snapshots in the case of dynamic data augmentation. A reason for why this might be the case could be because the model with standard augmentation might start overfitting the data, especially the less common classes, therefore requiring less trained snapshots to balance out the results of models that are trained more thoroughly. Conversely, the model trained with dynamic augmentation has a larger set of transformations for less common classes, requiring the model more time to properly learn their features. The model consistently outperforms the baseline from [5], who report a top-1 classification accuracy of 82.9%.

TABLE 2: Top-1 and top-5 test accuracies of snapshot ensembles on the car make classification task for the two data augmentation strategies. An ensemble of size k is constructed using the last k snapshots obtained during training. The most accurate ensembles are marked in **bold**.

	Standard augmentation		**Dynamic augmentation**	
k	Top-1	Top-5	Top-1	Top-5
1	84.85%	96.37%	93.84%	99.01%
2	85.52%	96.48%	**94.89%**	**99.14%**
3	85.59%	96.67%	94.72%	99.01%
4	**85.67%**	**96.76%**	94.25%	98.97%
5	85.33%	96.68%	93.83%	98.84%
6	84.95%	96.58%	93.42%	98.72%

B. Car make verification

As for this task, we compare the results of our model, trained with a few variations. As previously stated, these variations are:

i Training with data augmentation and cosine similarity as a scoring function

ii Training with no data augmentation and cosine similarity as a scoring function

iii Training with data augmentation and cosine distance as a scoring function

These models were taken into account as complete ensembles. Furthermore, we compared these results with a single, fully trained model obtained with data augmentation and cosine similarity as a scoring function. The accuracies obtained on a test set are shown in Table 3, where it appears that the complete ensemble with data augmentation and cosine similarity performs the best.

Being a verification task, it is important to assess if the increase in accuracy does not come from an inherent imbalance in the positive-negative pairs. As we have already discussed in Section IV, the split between positive and negative pairs is almost even, so we expect a rather even split in the test set too. However, we decide to show these results in the form of a confusion matrix, as seen in Figure 4. Interestingly, the best model performs better because it seems to be more prone to classifying pairs as *negative*. This makes sense when we consider a training loop which makes use of data augmentation but is very interesting to assess that cosine distance and the "classic" definition of contrastive loss yield a model that performs even worse. The exact reason as of why this is the case is not completely understood, but we believe that it might be because of the way cosine similarity and cosine distance "push" vectors: the former pushes the embeddings of different kinds of pairs to be parallel to each other, while the latter makes them antiparallel, which could be an inherently more complex state for the architecture to reach.

Having assessed which model performs the best, we plot its ROC curve. The result can be seen in Figure 5, where the Area Under Curve (AUC) score is reported too. In particular, it results that $\mathrm{AUC} = 0.9034$, a somewhat good score, considering that a perfect verifier would yield $\mathrm{AUC} = 1$. One should also keep in mind that the model we worked

TABLE 3: Accuracy on the test set for various models. The best results are highlighted in **bold**.

k	Data augm.	Scoring func.	Accuracy
5	✓	**Cosine similarity**	**83.40%**
1	✓	Cosine similarity	79.93%
5	✗	Cosine similarity	79.63%
5	✓	Cosine distance	74.38%

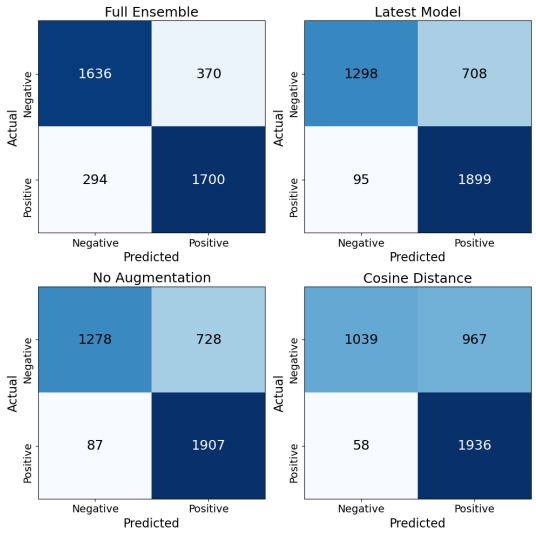

Fig. 4: Confusion matrices for the data verification task for various variants of the model. From top left, clockwise: Complete ensemble with data augmentation and cosine similarity, fully trained single model with data augmentation and cosine similarity, complete ensemble with data augmentation and cosine distance, and complete ensemble without data augmentation and cosine similarity.

with was not excessively tuned for this task and that the technique employed, snapshot ensemble, is computationally very advantageous since it does not introduce any additional training loops.

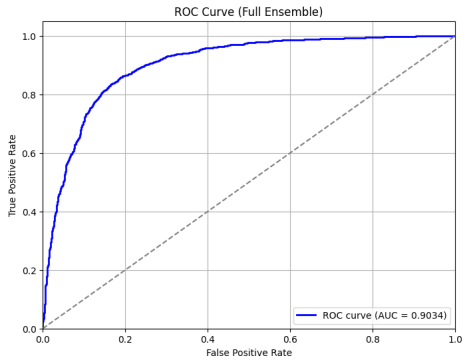

Fig. 5: ROC curve for the complete ensemble with data augmentation and cosine similarity as the scoring function

VII. Concluding Remarks

In this report, we have successfully applied snapshot ensembling to image-based car make classification and verification, highlighting how it can be used to improve the performance of a deep neural network without introducing any additional training cost. By combining this approach with the lightweight EfficientNet architecture, we were able to reduce the time and resources required to train a model that is capable of tackling these problems. Furthermore, we conducted an analysis of different data augmentation and training techniques, highlighting their importance also within the snapshot ensembling framework. Combining these insights, we have achieved highly accurate results especially on the classification task.

In conclusion, our results demonstrate that snapshot ensembles have great potential for improving accuracy even in short training periods. Of course, more research can be done on this technique; our report only takes into account the surface-level tasks of make classification and make verification on the CompCars dataset. The next point of interest could be the more challenging task of characterising the car models with this technique. Additionally, it could be interesting to explore longer training periods and vary the cycle lengths, to see if our admittedly quite short training session of only 30 epochs allowed the model to truly capture local minima, as it is the original goal of the technique.

As for our personal experience, this project gave us the opportunity to further explore the concept of ensemble learning, which is extremely interesting but is not investigated in great detail throughout the main course. It allowed us to "deal with" a real network, with its limitations in terms of time and resources, and it forced us to choose somewhat original strategies to tackle seemingly standard tasks. A big challenge for us was class imbalance; along with the suggested focal loss function, we wanted to explore a few more possibilities that would have helped us in contrasting this problem. Hyperparameter tuning was also another problem and, had we had more time and/or resources, we would have loved to explore that part further.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2015.

[3] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," 2017.

[4] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.

[5] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3973–3981, 2015.

[6] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017.

[7] L. N. Smith, "Cyclical learning rates for training neural networks," 2017.