Project Report - ECE 285

Transfer Learning Based Automobile Image Classification

Ruize Xuan
ECE
A59024784

Zeyu Chang
ECE
A59023417

Abstract

This project focuses on addressing an image classification problem specific to automobile images using Convolutional Neural Network (CNN) architectures based on transfer learning. The methodology encompasses two key approaches: the feature extraction method and the deeper layer fine-tune method. By combining these approaches, the project aims to efficiently and effectively classify automobile images, demonstrating the potential of transfer learning in image classification tasks.

1 Introduction

In this project, we aim to tackle an image classification problem using knowledge gained from our class. Image classification plays a crucial role in various aspects of our lives, including safety, traffic management, environmental impact analysis, and insurance processing. Our primary objective is to develop a method to classify car images into specific categories using Convolutional Neural Network (CNN) architectures based on transfer learning.

In this project, we chose transfer learning for several key reasons. Leveraging pre-trained models reduces training time and computational resources. Pre-trained models, especially those trained on large datasets like ImageNet, achieve high accuracy and reduce the risk of overfitting, particularly with smaller datasets. Pre-trained models are general enough to be fine-tuned for specific tasks, ensuring good performance. By using transfer learning, we aim to efficiently and effectively solve our car image classification problem. The main difference between traditional deep learning and transfer learning is indicated in Figure 2.

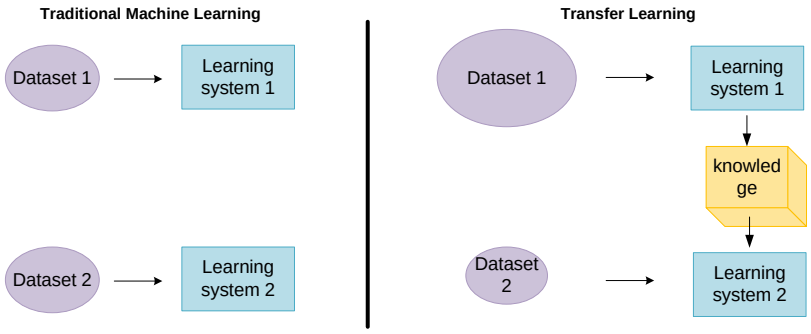

Figure 1: Comparison between traditional learning and transfer learning

The problem involves several key components, such as annotating and labeling our dataset, selecting appropriate CNN architectures, employing data augmentation techniques, tuning hyperparameters, and ensuring model interpretability. The size and characteristics of our chosen dataset will signifi-

cantly influence the methods we implement. As a result, transfer learning-based algorithms work pretty well and can achieve 82.15% accuracy at most.

2 Related Work

The concept of transfer learning was first introduced in the context of image classification using CNNs[1]. It demonstrated the effectiveness of transferring knowledge from pre-trained models on large datasets to new, related tasks with smaller datasets. The principles outlined in this work laid the groundwork for our approach in leveraging pre-trained models for car image classification.

Some very well-known network architectures have been created before. In this project, we will pick four basic network architectures and use them as a basis for transfer learning. A pioneer work on the AlexNet architecture demonstrated the power of deep learning on large-scale image classification tasks, leading to significant improvements in accuracy[2]. AlexNet's success on the ImageNet dataset highlights the importance of using large, diverse datasets for pre-training. A deep CNN architecture known for its simplicity and effectiveness in image classification tasks is the VGG network[3]. The VGG model's architecture, which uses small 3x3 convolution filters and deep layers, serves as a baseline for many transfer learning tasks. The ResNet architecture[4], which addresses the degradation problem in deep networks by introducing residual connections is another milestone work. ResNet models are known for their robustness and high performance in image classification. The creation of GoogleNet (Inception) architecture significantly improved state-of-the-art image classification by using a novel module structure that allows for increased depth and width of the network while keeping computational costs manageable[5].

These papers are fundamental to our approach, as they provide the architectures and methodologies that we adapt and fine-tune for our specific problem of car image classification. By leveraging these well-established models, we aim to achieve high accuracy and efficient performance in our project.

3 Method

3.1 Transfer learning

In this part, we will explain how we implement transfer learning in detail. We will not reiterate the familiar network architectures in reference papers. We will mainly focus on transfer learning aspects.

There are basically two frameworks for transfer learning. They are the feature extraction method and the deeper layer fine-tune method.

Feature extraction method On a high-level basis, each CNN structure is a feature extraction system. There are different features, represented by high-dimension tensors, given by each layer. So the most intuitive way is to replace the last fully connected layer with a customized linear layer or directly train an SVM classifier. The only difference between a pre-trained network and a customized network is that they have different output classes. It's denoted in left two of Figure 2. It can also be done by cutting in any previous layers. For example, if a network has multiple linear layers, we can train a shallow classifier on that specific position.

Deeper layer fine-tune method The second option for transfer learning considers more about the intrinsic properties of CNN. This practice is driven by the observation that the initial layers of a ConvNet encapsulate more universal features (such as edge or color blob detectors) that hold relevance across various tasks. However, as we progress through the layers of the ConvNet, they tend to specialize further, focusing on the intricate details specific to the classes present in the original dataset. So if we assume the front layers already can do a good job of grasping the general patterns, we can freeze all of them and only do backpropagation on the last layer or last block. It's denoted on the right side of Figure 2.

For this project specifically, we will describe how we chose which layers to fine-tune for the four networks that we mentioned. For AlexNet, it consists of 5 convolutional layers and 3 fully connected layers. In our experiments, we froze the first few convolutional layers and unfroze the last number of convolutional and fully connected layers for fine-tuning. For VGG, the VGG model increases the depth of the network by repeatedly stacking the convolutional layers, and this repetitive structure

simplifies the network design while improving the abstraction of features. For our experiments, we chose the VGG19 network model, which contains 16 convolutional layers, 5 pooling layers, and 3 fully connected layers, and we froze most of the front blocks and unfroze the last few; For Resnet, it consists of several repetitive blocks with residual paths. So we will freeze early blocks and unfreeze the last few blocks. Take Resnet50 as an example, it is structured by initial convolution and pooling layers, four sequential layers (layer 1 to layer 4), and final layers. For the middle part, four layers contain 3,4,6, and 3 Bottleneck blocks. So we may unfreeze the layer 4 blocks only. For Googlenet, it's a bit more complicated because of the inception structure and auxiliary classifiers. As for the final output, it has inception 5a and 5b blocks so we may unfreeze these to update parameters.

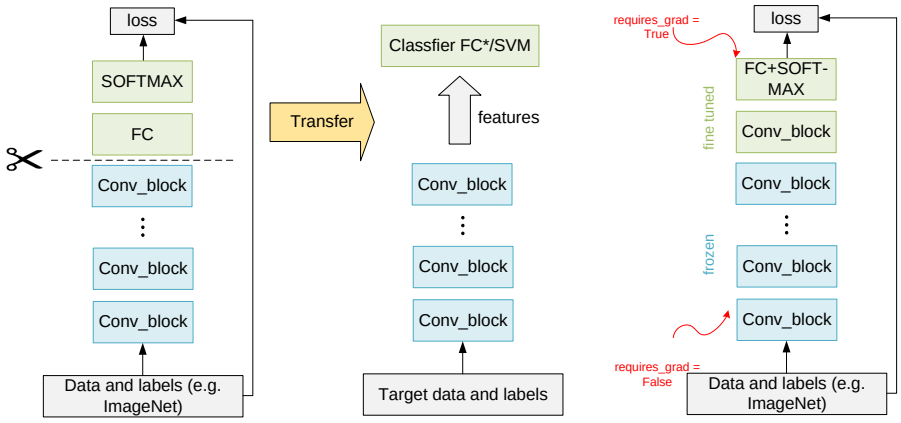

Figure 2: (**Left two**) Basic workflow of feature extraction method. **(Right)** Basic workflow of deeper layer fine-tune method.

3.2 Scheduler

Since our data is of small size and it's inherently a lot different from the Imagenet, we found that sometimes using grid search to fine-tune the parameters such as learning rate is inefficient and time-consuming. So our proposal is to use a scheduler so that it can adaptively adjust the learning rate given the performance of the model. The learning rate will be reduced when the training accuracy has stopped exceeding a certain threshold within allowed epochs.

4 Experiments

4.1 Dataset

In this section, I will first introduce the dataset that we use and then present the results.

This project is about car classification for the Stanford car dataset. The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

4.2 Parameter and numerical result

The general parameter setting is given in Table 1.

The experiment result is indicated by the following Table 2. We tested over 50 sets of parameters for each architecture and got the best result for each architecture. Among all of these experiments, Resnet152 achieves the best accuracy on the test set with 82.15%. On average, VGG achieves better than Alexnet primarily due to its deeper architecture, which consists of 16 or 19 layers compared to AlexNet's 8 layers, allowing it to learn more complex and detailed features. The use of smaller 3x3 filters in VGG captures finer details more effectively than the larger filters used in AlexNet. Additionally, VGG's uniform architecture with repeated layers enhances feature learning

Table 1: Parameter Settings

Parameter	Description	Value
Pretrained Weight	The initialization of model parameters.	IMAGENET1K_V1, IMAGENET1K_V2
Learning Rate	The rate at which the model learns.	0.01
Batch Size	Number of samples in a batch.	32
Number of Epochs	Total number of complete passes.	20/30
Patience	The number of allowed epochs with no improvement after which the learning rate will be reduced.	3
Threshold	The threshold for measuring the new optimum, to only focus on significant changes.	0.9

and optimization, and its parameter efficiency from smaller filters leads to better feature extraction. VGG also incorporates advanced regularization techniques like dropout and batch normalization, reducing overfitting and improving generalization. These characteristics collectively enable VGG to outperform AlexNet in capturing intricate patterns and adapting to new datasets in transfer learning scenarios. It turned out that Googlenet is the most difficult to fine-tune on our smaller dataset. At first, we just froze the last inception block and it got an even worse score than VGG, which is not consistent with our expectations. So what we implemented in the last is to add three linear classifiers after the inception 5a and 5b. It improved accuracy performance from 30% to about 72%.

On the whole, we found that the feature extraction method performs better than the deeper layer fine-tune method on shallow networks. But as layers go deeper and deeper, we may prefer deeper layer fine-tuning methods.

Table 2: Performance of Different Architectures on Stanford Car Dataset

Best Performance				
Networks	**Feature extraction**		**Deeper layer fine-tune**	
	train accuracy	test accuracy	train accuracy	test accuracy
Alexnet	99.85%	30.33%	97.54%	22.00%
VGG	99.85%	27.46%	97.76%	45.00%
Resnet	97.47%	46.74%	99.57%	82.15%
Googlenet	96.82%	40.95%	70.79%	72.00%

4.3 Loss and accuracy plot

The loss and accuracy plot is given in Figure 3.

From the Figure 3, we gained several conclusions. For both AlexNet and VGG, the training loss decreases gradually in all settings as the number of unfrozen layers increases, indicating that the models fit the training data better with the help of more degrees of freedom (i.e., trainable parameters).The training loss for AlexNet decreases at a more similar rate and magnitude for different unfrozen layer settings, whereas the decrease in the loss for VGG is much more pronounced, especially in the unfrozen four and unfreezing five layers. The training accuracy of both AlexNet and VGG improves with unfreezing more layers. However, VGG's improvement in training accuracy is more pronounced, and AlexNet's training accuracy improvement is more modest, which may indicate that its network architecture is not as responsive as VGG's to deeper training adjustments on this kind of task. In terms of test accuracy, AlexNet shows some overfitting, especially when the number of unfrozen layers is high. In contrast, the test accuracy of VGG increased significantly with the number of thawed layers. This suggests that the VGG19 structure is effective in improving the generalization ability when more trainable parameters are utilized.VGG shows better generalization ability than AlexNet in the setting of more unfrozen layers, which demonstrates its more sophisticated feature extraction ability. For more advanced architectures such as Resnet and Googlenet, we found that introducing

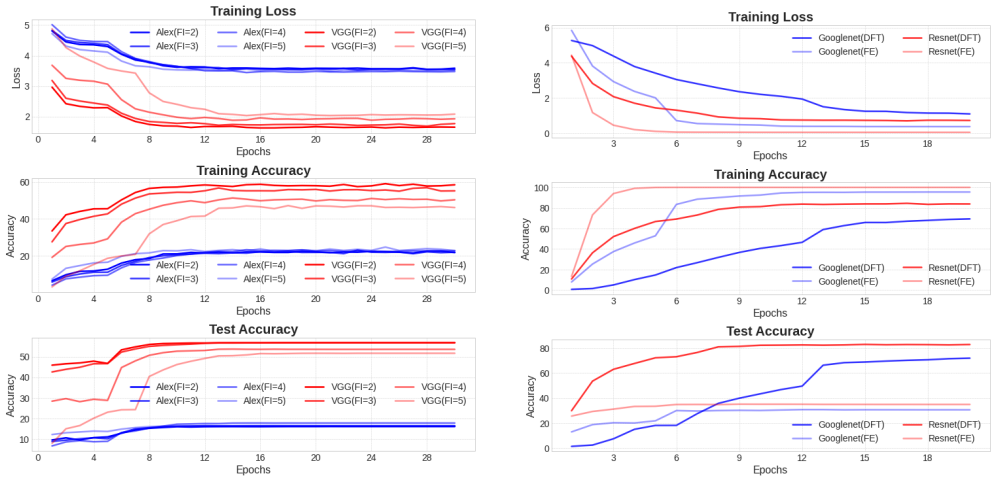

Figure 3: Experiment loss and accuracy. (**Left**) Results for Alexnet and VGG. (**Right**) Results for Resnet and Googlenet.

deeper layers would increase the accuracy performance to some extent. For example, Resnet 152 performed better than Resnet 18 and Resnet 50. But the deeper layer fine-tunes method and feature extraction method take on the opposite outcome on our training and testing set. The latter method beat the former in training but lost in testing. This could be explained by the following reasons. Our target dataset is small and very different from the original dataset. This leads to our model such as Googlenet, underfitting the training data but may achieve better by coincidence in the test set.

So we got some key points and inferences about transfer learning as well. We will consider the size and similarity of the new dataset compared with the dataset that is used to pre-train the network. If the dataset is small and very different from the original dataset as in our project, we may encounter this reversed effect as described above. We could try the deeper layer fine-tuning method. But be aware of the underfitting problem. When the dataset is small but very similar to the original dataset, we may have more confidence in doing this.

4.4 Extra analysis

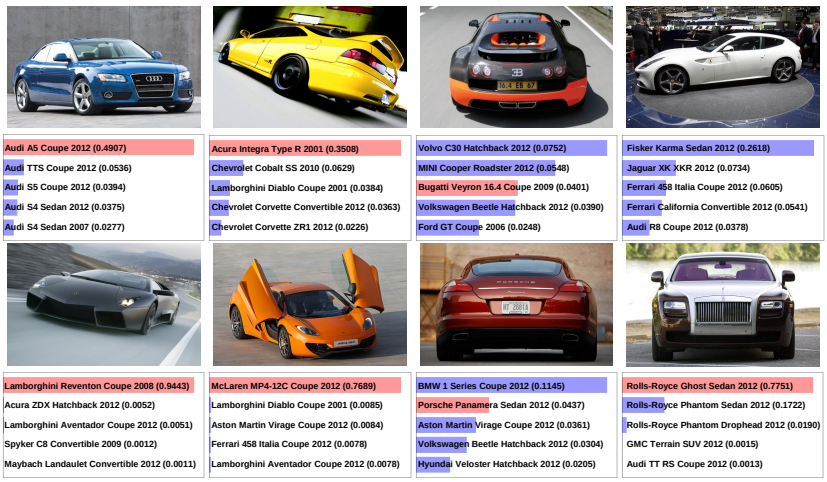

Figure 4: Eight test data points in Standford Car Dataset and the top five class that is given by our best model ranked by confidence from high to low. The true class is shown with a red bar and all the other wrong classes are in a blue bar.

In Figure 4, we presented the model prediction for each test data. The confidence that the model believed in the class of the car is given in the bar plot. After testing several cases, we found that the car can be mostly identified by its front side. This is probably due to its distinctive features such as the grille, headlights, bumper, and emblem, which vary significantly between different car models and brands, providing rich visual cues. Many car manufacturers prominently place their branding elements on the front side, such as logos or emblems, aiding in accurate classification. Additionally, certain design elements on the front side, like headlight shapes or grille styles, may follow standardized patterns within specific car models or brands, offering reliable visual cues. Since images of cars are often captured from the front or front-side angle in real-world scenarios, the front side provides more training data for CNN models, aligning with human perception biases that prioritize features most informative from the front.

In Figure 5, we presented our result for similarity analysis. For each test image, we found the top 5 nearest neighbors in the training set. It shows that our best model can handle this situation very well.

Figure 5: Four test data points in Standford Car Dataset in the first column. The remaining columns show the pictures in the train set that has the shortest feature vector distance just after the final hidden layer.

The goal of this analysis is to demonstrate how well the model can identify visually similar cars. For each test image, the model identifies five cars from the training set that have the closest feature vector distances, indicating high visual similarity. For the first test image (Yellow Jeep), all neighbors are off-road vehicles, primarily similar models in terms of shape and design. For the second test image, neighbors include similar body types, showing the model's effectiveness in grouping similar body types. For the third test image (Red Sedan), the neighbors are mostly sedans, emphasizing the model's ability to distinguish between different car types accurately. For the fourth test image (Orange Sports Car), neighbors consist of various sports cars, highlighting the model's proficiency in recognizing high-performance vehicles.

The ability of the model to find visually similar cars based on feature vector distances demonstrates its robustness in feature extraction. The model can effectively differentiate between various car types, such as SUVs, sedans, and sports cars, which is crucial for accurate classification. This similarity analysis suggests that the feature vectors learned by the model capture essential visual characteristics of the cars. The model's effectiveness in grouping similar cars together implies the potential for high performance in real-world automobile image classification tasks.

5 Implementation

In this project, we borrow the idea from the pre-trained model basic structure, using the pre-trained weights for initialization. We don't re-implement the structure by ourselves for the following reasons: our dataset is relatively small compared to classical training sets. Since we also use the transfer learning technique, it would be better to modify pre-trained models instead of rewriting the whole framework. What we did from scratch included data loading, augmentation, training, evaluation

Table 3: Our implementation

Section	**Description or function names (with lines count)**
Preparation	Data transform, loading, augmentation (35 lines)
Training functions	*train_model* (73 lines, method 2), *train_linear_classifier* (60 lines), *train_svm_classifier* (20 lines)
Evaluation functions	*eval_model* (27 lines), *eval_model_extract* (11 lines)
Method functions	*extract_features* (20 lines), *feature_extraction* (20 lines), *fine_tune_deep_alex* (26 lines), *fine_tune_deep_vgg* (24 lines), *feature_extraction_2* (17 lines, method 1)
Others	*CombinedModel* class (10 lines), confidence ranking (82 lines), similarity analysis (97 lines)

functions for different ways of achieving transfer learning, parameter tuning, loss and accuracy plots, confidence rank function, and similarity grouping functions of different test images. More details are indicated by the Table 3.

References

[1] Maxime Oquab et al. "Transfer learning using convolutional neural networks". In: *arXiv preprint arXiv:1411.1792* (2014).

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[3] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[5] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.