ECE371 Neural Networks and Deep Learning Assignment 1

Tianxiang Fei 22308035
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
feitx@mail2.sysu.edu.cn

Abstract: This paper presents a fine-tuned ResNet-50 model for flower image classification, addressing the challenge of recognizing diverse floral species with high accuracy. Leveraging transfer learning, we adapt the pretrained ResNet-50 architecture [1] on the five flowers dataset in the structure of ImageNet, achieving competitive performance through targeted adjustments to some of the layers and optimization hyperparameters. Experimental results demonstrate an accuracy of 94.9% on the val set, outperforming baseline CNN models. Our approach highlights the efficacy of fine-tuning large-scale pretrained models for domain-specific computer vision tasks while reducing computational costs.

Keywords: ResNet, DenseNet, ImageNet, Fine-tuning

1 Introduction

Flower Classification is a famous task on Kaggle which the goal is to build a model that can classify images of flowers into different categories. This is a commonly used supervised learning task in the field of computer vision. To accomplish this fine-tuning task, I explored ResNet and DenseNet [2] architectures, optimizing hyperparameters and comparing their performance. ResNet is a convolutional neural network introduced by Microsoft Research in 2015. The key idea of ResNet is the residual block, which uses skip connections to allow the network to learn the difference between the input and output of a few layers, instead of learning everything from scratch. ResNet achieved great success in computer vision tasks and won the ImageNet competition in 2015. DenseNet is another convolutional neural network architecture that introduces direct connections between any two layers with the same feature-map size. In DenseNet, each layer receives the feature maps of all preceding layers as additional inputs, and passes its own feature maps to all subsequent layers. This dense connectivity pattern helps to mitigate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters compared to traditional convolutional networks.

2 Related Work

The task of image classification has undergone significant evolution, marked by key architectural innovations. Early breakthroughs were led by AlexNet [3], which demonstrated the effectiveness of deep convolutional neural networks when trained on large datasets like ImageNet. This work introduced ReLU [4] activation and dropout, paving the way for deeper models.
Subsequent models, such as VGGNet [5], explored deeper but uniform architectures with small convolutional filters. However, these networks suffered from computational inefficiency and vanishing gradients, which motivated further advances.
To address these limitations, ResNet introduced residual connections, allowing networks to be trained with over 100 layers. This innovation significantly improved feature learning and gradient flow, establishing a new baseline for deep learning performance.

Building on this, DenseNet proposed densely connected layers, where each layer receives input from all preceding layers. This design encourages feature reuse, reduces parameter count, and enhances performance especially in small-data scenarios.
More recently, the rise of transformer-based models such as Vision Transformer (ViT) [6] has shifted the trend toward attention mechanisms. ViT treats images as sequences of patches, leveraging self-attention for global context modeling. While transformers often require large-scale data and pre-training, they have outperformed CNNs in several benchmarks when properly scaled.
In summary, the field has evolved from simple deep CNNs to architectures emphasizing feature reuse, residual learning, and attention. These advancements reflect a broader trend toward deeper, more efficient, and data-hungry models, often supported by transfer learning and large-scale pre-training.

3 Method

This task involves enhancing the provided code by completing the following components: data augmentation module, final fully connected layer, loss function implementation, optimizer configuration, and backward pass with optimization procedures.
Data augmentation has become a standard technique for dataset expansion, particularly in small-data scenarios similar to this task. It artificially diversifies training samples through geometric and photometric transformations, including rotation and shifting, while preserving label correctness. Data augmentation serves three primary purposes: to mitigate data scarcity, improve model generalizability, and reduce overfitting risks. In the data augmentation pipeline, I produced five data augmentation methods, including random cropping and resizing, random horizontal flipping, color jittering, random vertical flipping,, and normalizing.
The fully connected layer represents a fundamental architectural component in neural networks. In classification tasks, the output dimensionality of the final FC layer typically corresponds to the number of target classes. Its primary function involves integrating locally or globally extracted features from the preceding layers to generate higher-level semantic representations. In the present task, we constructed a new FC layer based on the original model's final FC layer, with its output dimension adjusted to match the number of classes in our specific task, thereby enabling fine-tuning of the pre-trained model.
The loss function serves as the optimization objective in machine learning, which quantitatively measures the discrepancy between predicted values and ground truth labels, while guiding parameter updates through backpropagation. Commonly employed loss functions include cross-entropy loss, MSE loss, and L1 loss, among others. Given that the current project addresses a single-label multiclass classification problem, the cross-entropy loss function was adopted.
The optimizer serves as a parameter update strategy in deep learning that employs various methodologies to adjust model parameters based on the gradients of the loss function. Commonly utilized optimizers include SGD, Adam [7], and Adagrad [8], among others. For the current task of model fine-tuning, which requires meticulous parameter optimization, the SGD optimizer was selected. Furthermore, an adaptive learning rate adjustment mechanism was incorporated to dynamically modify the learning rate according to training epochs, thereby enhancing the precision of parameter optimization.

4 Experiments

I evaluate my method on the ImageNet 5 flower classification dataset. The dataset was partitioned into training and test sets with an 8:2 ratio, while data augmentation techniques were applied to expand the sample size. The original dataset comprised 2,278 training images and 570 validation images. I valuate top-1 error rate.
Using ResNet-18 as the baseline model, I first validated the impact of batch size on model

performance. The result is displayed in the Table 1.

batchsize	32	64	128
val acc	90.2	91.4	92.6

Table 1: Impact of Batch Size on Model Performance

Experimental results demonstrate that larger batch sizes consistently yield superior model convergence, potentially attributable to more comprehensive feature learning per gradient update during fine-tuning.

I evaluated the fine-tuning performance of three pretrained architectures (ResNet-18, ResNet-50, and DenseNet) on our target dataset. All models were trained with identical hyperparameters and learning strategies. The comparative results are presented in Table 2.

Structure	ResNet18	ResNet50	DenseNet
val acc	92.6	94.5	94.2

Table 2: Impact of Structure on Model Performance

The experimental results demonstrate that ResNet-50 achieves the best validation accuracy (94.5%), followed closely by DenseNet (94.2%) and then ResNet-18 (92.6%). This performance hierarchy suggests that deeper architectures (ResNet-50) generally yield better feature extraction capabilities for this task, while DenseNet's competitive performance despite its relatively compact structure highlights the effectiveness of its dense connectivity pattern in feature reuse. The 1.9% accuracy gap between ResNet-18 and ResNet-50 indicates that increased model depth significantly benefits performance, though the marginal 0.3% difference between ResNet-50 and DenseNet suggests their capabilities may be comparable for this specific application. These findings provide practical guidance for model selection, where ResNet-50 would be preferred for maximum accuracy, while DenseNet offers an excellent alternative when considering the trade-off between performance and model efficiency.

I also evaluated the impact of freezing training on fine-tuning performance based on the ResNet-50 framework. Specifically, I froze the first two layers of the ResNet-50 model, as these early layers are theoretically responsible for extracting low-level visual features. Given that our training dataset is relatively small, allowing these parameters to be updated during training could lead to the degradation of the generalizable representations that the model had previously learned from large-scale pretraining. The results are presented in Table 3.

Freeze	Yes	No
val acc	94.5	93.2

Table 3: Impact of Structure on Model Performance

As shown in Table 3, freezing the first two layers of the ResNet-50 model led to a higher validation accuracy (94.5%) compared to not freezing them (93.2%). This indicates that freezing the early layers can be beneficial when fine-tuning on a relatively small dataset. A likely reason is that the early layers of a pretrained model capture general low-level features that are transferable across tasks. Allowing these layers to update on limited data may cause the model to overfit and lose the robust representations learned from large-scale pretraining.

After comparing various training strategies and model architectures, the final best model was obtained by fine-tuning a ResNet-50-based network with the first two layers frozen. Additional hyperparameters, such as learning rate and optimizer, were also tuned and evaluated. Due to space limitations, detailed comparisons are omitted here. The optimal training configuration has been documented in the script file, achieving a top-1 accuracy of approximately 94.5%.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[2] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2018. URL https://arxiv.org/abs/1608.06993.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi:10.1145/3065386. URL https://doi.org/10.1145/3065386.

[4] A. F. Agarap. Deep learning using rectified linear units (relu), 2019. URL https://arxiv.org/abs/1803.08375.

[5] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL https://arxiv.org/abs/1409.1556.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[8] L. Luo, Y. Xiong, Y. Liu, and X. Sun. Adaptive gradient methods with dynamic bound of learning rate, 2019. URL https://arxiv.org/abs/1902.09843.