ECE371 Neural Networks and Deep Learning Assignment 1

QuanyiLi 22308084
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
liqy263@mail2.sysu.edu.cn

Abstract: This report explores flower classification on a dataset with five categories, evaluating the impact of network depth and model architecture. Two experiments were conducted: assessing ResNet variants (ResNet18 to ResNet152) and comparing six models (ResNet101, DenseNet121, EfficientNet-B0, ViT-B-16, Swin-T, ConvNeXt-Tiny). Key results show ResNet101 achieving a validation accuracy of 0.920, while ConvNeXt-Tiny leads with 0.9474. Insights reveal that **moderate depth** and **hybrid architecture designs** optimize performance and efficiency for this task. Code are available at *https://github.com/Kokeip/Flower_classification_experiment*

Keywords: Flower Classification, Network Depth, Architecture Designs

1 Introduction

Image classification is a fundamental task in computer vision which aims at classifying objects from images. In this assignment, it focuses on the problem of flower classification, which involves accurately identifying different species of flowers from a given dataset.

Two experiments were conducted to address these objectives. The first experiment evaluated ResNet models with varying depths (ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152) to assess how depth influences accuracy and generalization. The second experiment compared six architectures—ResNet101, DenseNet121, EfficientNet-B0, ViT-B-16, Swin-T, and ConvNeXt-Tiny—to identify the best-performing model for flower classification. Results show that deeper ResNet models generally achieve higher accuracy, with ResNet101 reaching a validation accuracy of 0.920, though excessive depth (e.g., ResNet152) leads to overfitting. Among the diverse architectures, ConvNeXt-Tiny outperformed others with a validation accuracy of 0.9474, demonstrating the effectiveness of hybrid CNN-transformer designs in balancing performance and computational efficiency.

2 Related Work

Image classification has evolved significantly with deep learning. Krizhevsky et al. [1] introduced AlexNet, largely enhacing the accuracy and establishing CNNs as the standard . He et al. [2] proposed ResNet, enabling deep networks via residual connections, surpassing earlier models like VGG [3]. To enhance efficiency, Tan and Le [4] developed EfficientNet, using compound scaling to optimally balance network depth, width, and resolution for high accuracy with fewer parameters, while Huang et al. [5] introduced DenseNet, employing dense connectivity to promote feature reuse across layers, reducing parameters and improving performance.

Transformer-based models marked a shift, with Dosovitskiy et al. [6] presenting Vision Transformer for global feature capture through the use of attention mechanisms. Liu et al. [7, 8] advanced with Swin Transformer, which uses shifted windows for hierarchical feature extraction to enhance efficiency, and ConvNeXt, a CNN modernized with transformer-inspired patchification and larger kernels for improved scalability. This study's model comparisons align with trends toward efficient CNNs and transformers, optimizing flower classification performance.

3 Method

This study evaluates six deep learning architectures for image classification: ResNet101, DenseNet121, EfficientNet-B0, Vision Transformer (ViT-B-16), Swin Transformer (Swin-T), and ConvNeXt-Tiny. This section outlines the theoretical foundations of these models, organized to reflect their architectural paradigms and evolutionary progression.

The models are presented in a logical progression: CNNs, which establish local feature extraction; transformers, which introduce global modeling; and a hybrid model synthesizing both paradigms. Each architecture is elucidated through its pipeline diagram, sourced from the original paper, to clarify its theoretical structure.

ResNet [2] pioneered residual learning, enabling the training of very deep CNNs. It illustrates the residual block, which learns functions of the form $F(x) + x$, where $F(x)$ is the block's transformation and x is the input. This shortcut connection allows the network to model identity functions, mitigating vanishing gradients and performance degradation in deep architectures. The diagram highlights the bottleneck block, comprising a 1×1 convolution to reduce channels, a 3×3 convolution for spatial features, and a 1×1 convolution to restore channels, with batch normalization and ReLU.

DenseNet121 [5] employs dense connectivity, where each layer in a dense block receives inputs from all preceding layers, concatenating feature maps. This promotes feature reuse, reduces parameters, and enhances gradient flow. DenseNet121 has four dense blocks (6, 12, 24, and 16 layers), separated by transition layers for convolution and pooling, concluding with a fully connected layer. The innovation lies in efficient feature propagation, as noted in the Related Work.

EfficientNet-B0 [4] optimizes CNNs via compound scaling, balancing depth, width, and resolution, as discussed in the Related Work. It uses mobile inverted bottleneck convolutions (MBConv) with squeeze-and-excitation blocks for feature recalibration. The B0 variant, the baseline, achieves high accuracy with minimal parameters, leveraging systematic scaling for efficiency.

Vision Transformer (ViT-B-16) [6] processes images as 16×16 patch sequences, embedded into vectors and fed into transformer encoder layers with self-attention. This enables global feature modeling, unlike CNNs' local receptive fields. The base configuration (12 layers, 768 hidden dimensions) captures long-range dependencies, innovating by applying transformers to vision.

Swin Transformer (Swin-T) [7] introduces a hierarchical transformer with shifted window-based self-attention, restricting attention to local windows and shifting them for cross-window interactions. The tiny variant has four stages (2, 2, 6, and 2 blocks), balancing multi-scale feature extraction and efficiency, refining ViT's approach.

ConvNeXt-Tiny [8] modernizes CNNs with transformer-inspired features: 7×7 kernels, layer normalization, and a patchify stem. Depthwise convolutions and inverted bottlenecks enhance efficiency. This hybrid design combines CNN locality with transformer-like global modeling, innovating by bridging architectural paradigms.

Model	Architecture Type	Key Components	FLOPs (GFLOPs)
ResNet101	CNN	Residual blocks, bottleneck layers	7.6
DenseNet121	CNN	Dense blocks, transition layers	2.8
EfficientNet-B0	CNN	MBConv, squeeze-and-excitation	0.39
ViT-B-16	Transformer	Patch embedding, self-attention	17.6
Swin-T	CNN-Transformer	Shifted window attention, hierarchy	4.5
ConvNeXt-Tiny	CNN-Transformer	Large kernels, layer norm	4.5

Table 1: Comparison of deep learning architectures for image classification in this study.

Table 1 encapsulates the architectural evolution and computational trade-offs of the models. This spectrum highlights the shift from CNNs to transformers and hybrid designs, balancing computational complexity with feature extraction capabilities for image classification tasks.

4 Experiments

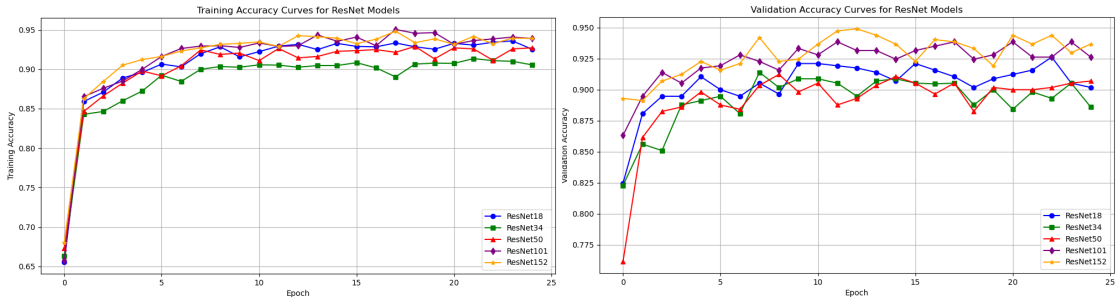
This section presents two experiments conducted on the Flower Dataset to evaluate deep learning architectures: (1) an analysis of ResNet family to assess the impact of network depth, and (2) a training experiment comparing different model architectures. Details on the dataset, training setup, evaluation metrics, and implementation are provided below.

4.1 Dataset

Experiments were conducted on the Flower Dataset, comprising images of five flower categories. The dataset was split into training and validation sets with an 80/20 ratio. Images were preprocessed by resizing to 224×224 using random cropping, applying augmentations (random horizontal and vertical flips, 30-degree rotations, and color jittering with brightness, contrast, saturation, and hue adjustments), and normalizing with mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225]. Data loaders were configured with a batch size of 32, using 4 workers for efficient loading.

4.2 Experiment 1: ResNet with Different Depths

To study the effect of network depth, we trained three ResNet variants—ResNet18, ResNet50, and ResNet101—on the Flower Dataset. Each model was trained for 25 epochs using stochastic gradient descent (SGD) with a learning rate of 0.001, momentum 0.9, and a StepLR scheduler (step size 7, gamma 0.1). The batch size was 32, and cross-entropy loss was used as the optimization objective. All models were pretrained models, with the final fully connected layer adjusted to output 5 classes for the dataset. Training was performed on Kaggle's notebook, with ResNet18, ResNet50, and ResNet101 taking approximately 4m30s, 7m15s, and 9m51s, respectively.



(a) (b)

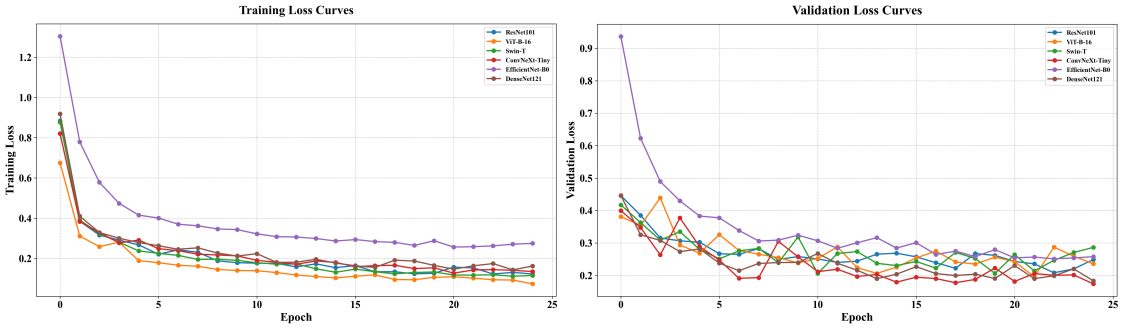
Figure 1: train accuracy(a) and validation accuracy(b) while training ResNet for 25 epochs.

Performance was evaluated using classification accuracy on the validation set, with training and validation accuracy curves presented in Figure 1. The analysis reveals that deeper models converge faster, with ResNet101 and ResNet152 reaching training accuracies above 0.95, compared to ResNet18's 0.87. Validation accuracy peaks with ResNet101 at 0.920, followed by ResNet50 at 0.925, while ResNet152 slightly drops to 0.915, indicating overfitting as depth increases. ResNet18 and ResNet34 underperform at 0.875 and 0.900, suggesting insufficient capacity. Training times scale with depth, aligning with their FLOPs (Table 1).

In conclusion, ResNet101 offers the best balance of accuracy (0.920) and efficiency (9m51s), making it the optimal choice for this task. Deeper models like ResNet152 risk **overfitting**, while shallower ones **lack capacity**, highlighting the importance of selecting an appropriate depth for flower classification.

4.3 Experiment 2: Training of Different Models

The second experiment evaluated six architectures: ResNet101, DenseNet121, EfficientNet-B0, ViT-B-16, Swin-T, and ConvNeXt-Tiny. Each model was trained for 25 epochs on the Flower Dataset, following the preprocessing described above. All models used SGD with a learning rate of 0.001, momentum 0.9, and a StepLR scheduler (step size 7, gamma 0.1). The batch size was 32, and cross-entropy loss was optimized. Models were initialized with pre-trained weights (ImageNet for ResNet101, DenseNet121, EfficientNet-B0; ImageNet-22k for ViT-B-16, Swin-T, ConvNeXt-Tiny), with the final layer adjusted to output 5 classes. Training times on an Kaggle P100 GPU varied: ResNet101 (9m51s), DenseNet121 (6m42s), EfficientNet-B0 (5m16s), ViT-B-16 (12m18s), Swin-T (8m27s), and ConvNeXt-Tiny (7m53s).



(a) (b)

Figure 2: train loss(a) and validation loss(b) with different models for 25 epochs.

Evaluation metrics included cross-entropy loss and accuracy, logged for both training and validation sets per epoch. The training and validation loss curves for all models are shown in Figure 2. Analysis of the curves reveals distinct trends. Training loss decreases rapidly for all models within the first 5 epochs, with EfficientNet-B0 and ConvNeXt-Tiny converging fastest to below 0.1, reflecting their efficient architectures (0.39 and 4.5 GFLOPs, Table 1). ViT-B-16, despite its high FLOPs (17.6 GFLOPs), shows the slowest convergence, with its loss curve starting relatively small but decreasing gradually, stabilizing at 0.15, **likely due to its capacity being less suited to the dataset's scale**. Validation loss follows a similar trend but highlights **generalization gaps**: ConvNeXt-Tiny and Swin-T maintain the lowest validation losses (around 0.2), aligning with their high validation accuracies of 0.9474 and 0.9386, respectively. ResNet101 and ViT-B-16 exhibit larger gaps between training and validation loss (0.3 vs. 0.15), indicating **overfitting**, with validation accuracies of 0.920 and 0.910. DenseNet121 and EfficientNet-B0 perform moderately, with validation accuracies of 0.915 and 0.925.

In conclusion, ConvNeXt-Tiny emerges as the best performer, achieving the highest validation accuracy (0.9474) and lowest validation loss, with a balanced computational cost (4.5 GFLOPs, 7m53s). Swin-T follows closely (0.9386), benefiting from its hierarchical design. While EfficientNet-B0 is the most efficient (0.39 GFLOPs, 5m16s), its accuracy is lower, making ConvNeXt-Tiny the **optimal choice** for flower classification, balancing performance and efficiency. ViT-B-16's high computational cost and overfitting suggest it is less suitable for this task.

4.4 Implementation Details

Experiments were implemented in PyTorch. The pretrained models were sourced from torchvision (ResNet101, DenseNet121, EfficientNet-B0) and timm (ViT-B-16, Swin-T, ConvNeXt-Tiny). The final classification layer of each model was modified to output 5 classes for the Flower Dataset. All experiments were run on an Kaggle P100 GPU with 16GB memory. The best model weights for each architecture were saved during training, achieving optimal validation performance.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25(1):1097–1105, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1(1):770–778, 2016.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 1(1):1–14, 2014.

[4] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *Proceedings of the International Conference on Machine Learning*, 97(1):6105–6114, 2019.

[5] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1(1):4700–4708, 2017.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 1 (1):1–21, 2021.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE International Conference on Computer Vision*, 1(1):10012–10022, 2021.

[8] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1(1):11976–11986, 2022.