COSE474-2024F: Final Project Final Report
"Comparing Fine-Tuned Legacy Architecture with Non-Fine-Tuned Modern Architecture Model: ResNet-18 vs ConvNeXT-Tiny on CIFAR-10"

Jaehoon Han, 2022320101

1. Introduction

ResNet, a CNN architecture used for image classification, was introduced by Microsoft Research in 2015. It has been continuously developed and many architectures are introduced such as ViT and Swin Transformer, etc. In 2022, Meta AI Research introduced ConvNeXT, a state-of-the-art(SOTA) architecture.

In this research, we explore the effect of model architecture and fine-tuning on performance on specific tasks. This research is helpful to selecting the model for other researches whether fine-tuned legacy architecture or pre-trained modern architecture.

ConvNeXT shows higher performance than ResNet. However, we cannot know exactly how fine-tuning is impacts the performance. We're going to estimate the effect of model architecture and fine-tuning.

To comparing, we will use fine-tuned ResNet-18 and only pre-trained(non-fine-tuned) ConvNeXT-Tiny models.

We can understand the effect of fine-tuning more deeply by this experiment.

2. Methods

This research compares the performance of a fine-tuned legacy architecture (ResNet-18) against a non-fine-tuned modern architecture (ConvNeXT-Tiny) using the CIFAR-10 dataset. ResNet-18, introduced by Microsoft Research in 2015, represents a widely recognized convolutional neural network (CNN) structure for image classification. Conversely, ConvNeXT-Tiny, developed by Meta AI Research in 2022, showcases state-of-the-art (SOTA) performance. The core novelty lies in evaluating the combined influence of fine-tuning and model architecture on classification accuracy across varying data complexities.

Key challenges addressed in this study include:

Fair Comparison

Ensuring a controlled environment for benchmarking, by standardizing training datasets and computational resources.

Fine-Tuning Impact

Investigating how fine-tuning enhances ResNet-18's performance in contrast to a pre-trained ConvNeXT-Tiny.

Model Scalability

Exploring how the architectures scale as the number of classes increases.

By systematically addressing these aspects, the research provides insights into choosing between fine-tuned legacy architectures and modern pre-trained models for specific tasks.

The experiment proceeds in the following steps:

- Prepare the dataset and ResNet for fine-tuning.
- Perform fine-tuning by adjusting the number of classes in the dataset.
- Test the fine-tuned ResNet and ConvNeXT.

3. Experiments

3.1. Dataset

Due to limited resources, the relatively small CIFAR-10 dataset was utilized. The size of the dataset is not expected to significantly impact achieving the objectives of the experiment.

3.2. Computing resource

Processor: Apple M1 Pro
OS: Sequoia 15.1.1
Python: 3.10.16
Torch: 2.5.1

3.3. Experimental design/setup

3.3.1. Dataset Preparation

The experiments were conducted using the CIFAR-10 dataset.

A custom dataset class, FilteredCIFAR10, was implemented to filter specific classes from the original CIFAR-10 dataset.

The number of classes in the dataset was adjusted for fine-tuning experiments by selecting subsets of classes from the original 10 classes. This allowed for evaluating the model's performance with varying levels of task complexity.

Data preprocessing was performed using torchvision.transforms, which included image normalization and augmentations.

3.3.2. Model Architecture

ResNet and ConvNext architectures were employed as the backbone models for the experiments.

ResNet (specifically torchvision.models.resnet18) was fine-tuned for transfer learning, and ConvNext utilized the convnext_tiny architecture.

The final layers of the models were modified to align with the CIFAR-10 classification task (matching the number of selected output classes).

3.3.3. Training Procedure

The optimizer used for training was torch.optim.Adam with specified initial learning rates and weight decay.

The loss function was CrossEntropyLoss.

Data was loaded in batches using DataLoader, with parallel processing enabled to optimize training speed.

3.3.4. Testing

A dedicated test script was implemented to evaluate the performance of the trained models.

Class-wise predictions and overall accuracy were recorded. Metrics such as confusion matrices and quantitative scores were computed.

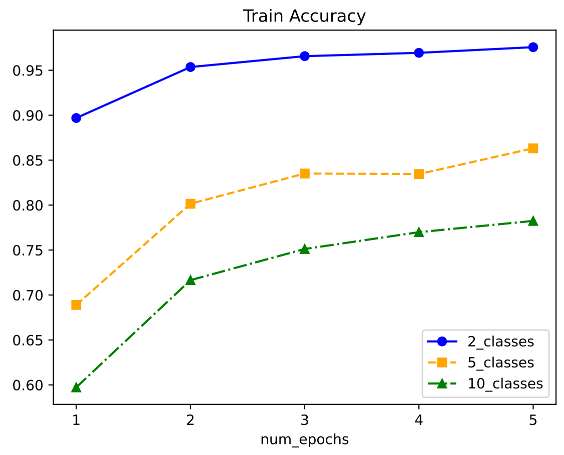
3.3.5. Computational Environment

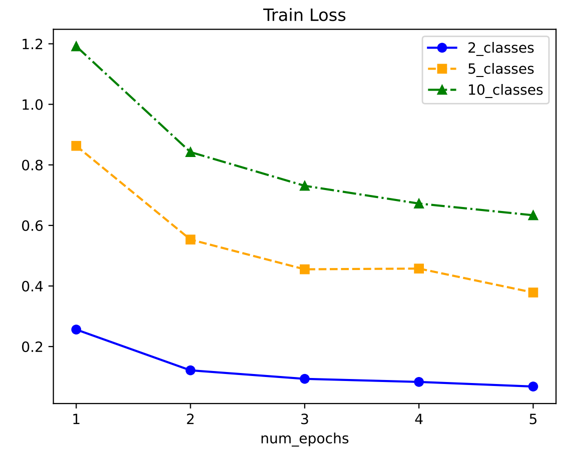
The experiments were conducted on a system using Apple's Metal Performance Shaders (MPS) as the backend for hardware acceleration.

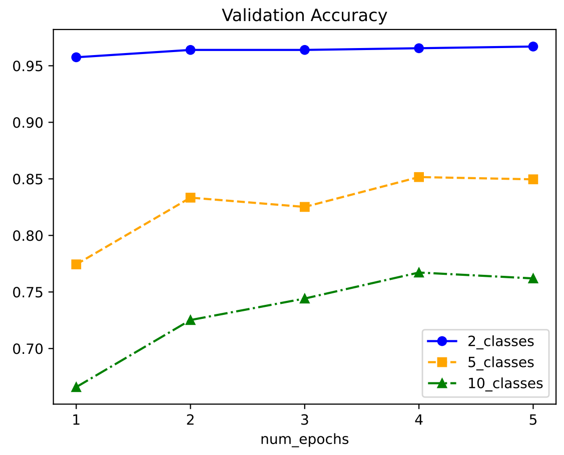
MPS was utilized via PyTorch's support for macOS-based GPU environments, ensuring efficient computation without relying on traditional CUDA or CPU setups.

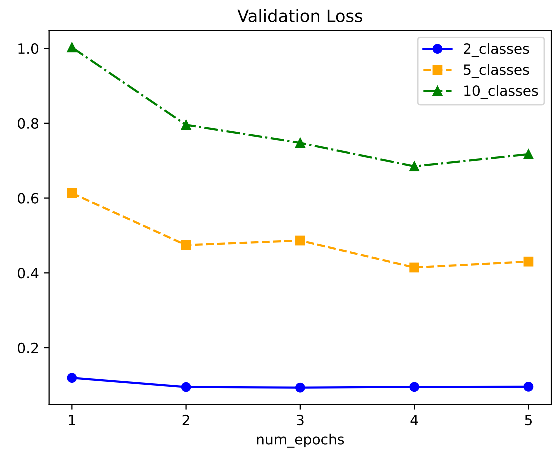
The experiments utilized the latest versions of PyTorch and relevant libraries optimized for the MPS backend.

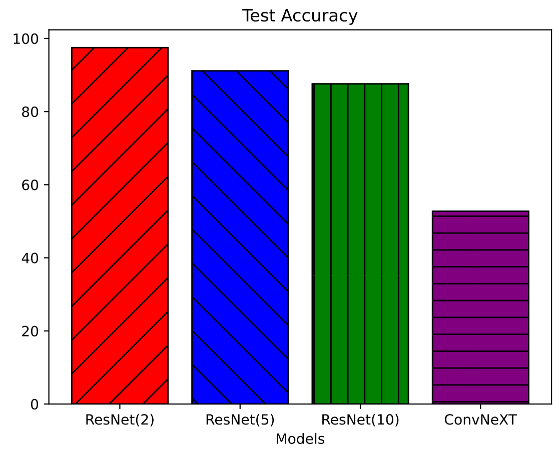
3.4. Results and Figures











4. Future direction

Extending Training Epochs

One potential direction for future work is to increase the number of training epochs to better evaluate the model's performance with more extensive training. This could help in understanding whether the current model has fully converged or if there is room for further improvement in accuracy and generalization.

Exploring Additional Architectures

Experimenting with other advanced architectures, such as Vision Transformers (ViTs) or EfficientNet, could provide insights into the performance differences across various model types.

Fine-tuning with Larger Datasets

Scaling the experiments to larger datasets, such as CIFAR-100 or ImageNet subsets, could help in evaluating the model's performance on more complex and diverse tasks.

Hyperparameter Optimization

Further tuning of hyperparameters, such as learning rates, weight decay, and batch sizes, could potentially yield better performance and efficiency.

Class Imbalance and Data Augmentation

Investigating the effect of class imbalance and employing more advanced augmentation techniques (e.g., MixUp, CutMix) could help improve the robustness of the model.

Evaluating Robustness

Testing the model against adversarial attacks or out-of-distribution data could provide a deeper understanding of its reliability in real-world scenarios.

5. Reference

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 (CVPR), 2016, pp. 770–778.

Z. Liu, H. Mao, C. Wu, et al., “A ConvNet for the 2020s,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” Technical Report, University of Toronto, 2009.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” Advances in Neural Information Processing Systems (NeurIPS), 2014, pp. 3320–3328.

C. Zhang, S. Bengio, M. Hardt, et al., “Understanding deep learning requires rethinking generalization,” Communications of the ACM, vol. 64, no. 3, pp. 107–115, 2021.

D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” International Conference on Learning Representations (ICLR), 2015.