Source Code

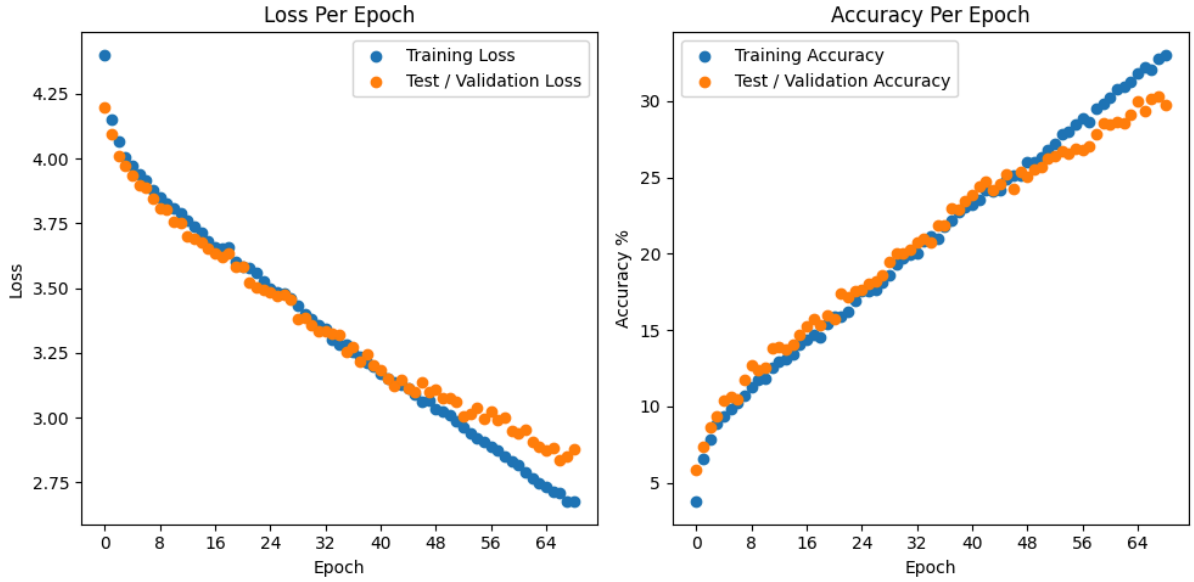
Remote Repository: https://github.com/Mikestriken/Deep-Learning-Class
Name: Michael Marais
Student ID: 801177649
Homework Number: 6

Problem 1

Your goal is to design a Vision Transformer architecture from scratch tailored for CIFAR-100, which consists of 100 classes and 32x32 RGB images, and then analyze how different configurations impact computational complexity and performance compared to a ResNet-18 baseline. Begin by creating a ViT with patch embedding, transformer encoder blocks, and a classification head, experimenting with configurations such as patch sizes of 4x4 and 8x8, embedding dimensions of 256 and 512, transformer layers of 4 and 8, attention heads of 2 and 4, and an MLP hidden dimension set to four times the embedding dimension (e.g., 256 for an embedding dimension of 128). Write a complete PyTorch script to train your ViT on CIFAR-100, incorporating data loading with torchvision.datasets. CIFAR100 and standard training hyperparameters like a batch size of 64, 20-50 epochs, and an Adam optimizer with a learning rate of 0.001. Next, analyze the computational complexity by calculating the theoretical number of parameters for each configuration, estimating FLOPs per forward pass using a tool like torchinfo or manual computation, and measuring training time. For comparison, implement or use a pretrained ResNet-18 from torchvision.models, train it on CIFAR-100 with the same hyperparameters, and evaluate test accuracy after 10 epochs, number of parameters, FLOPs, and training time per epoch against your ViT configurations. In your report, include a table summarizing results for at least four ViT configurations and ResNet-18, and discuss the trade-offs between accuracy, model size, and computational complexity, explaining why certain configurations might outperform or underperform ResNet-18.

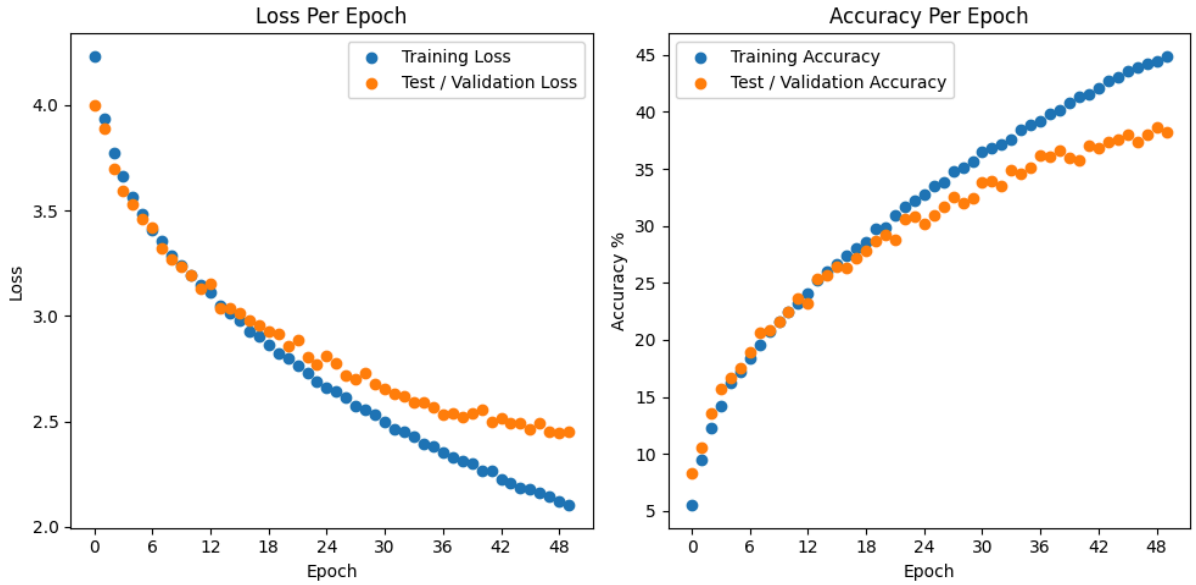
Vision Transformer (Patch_Size x Patch_Size) -- Embedding Size -- (Layers, Heads)

(8x8) -- 256 -- (8,4)



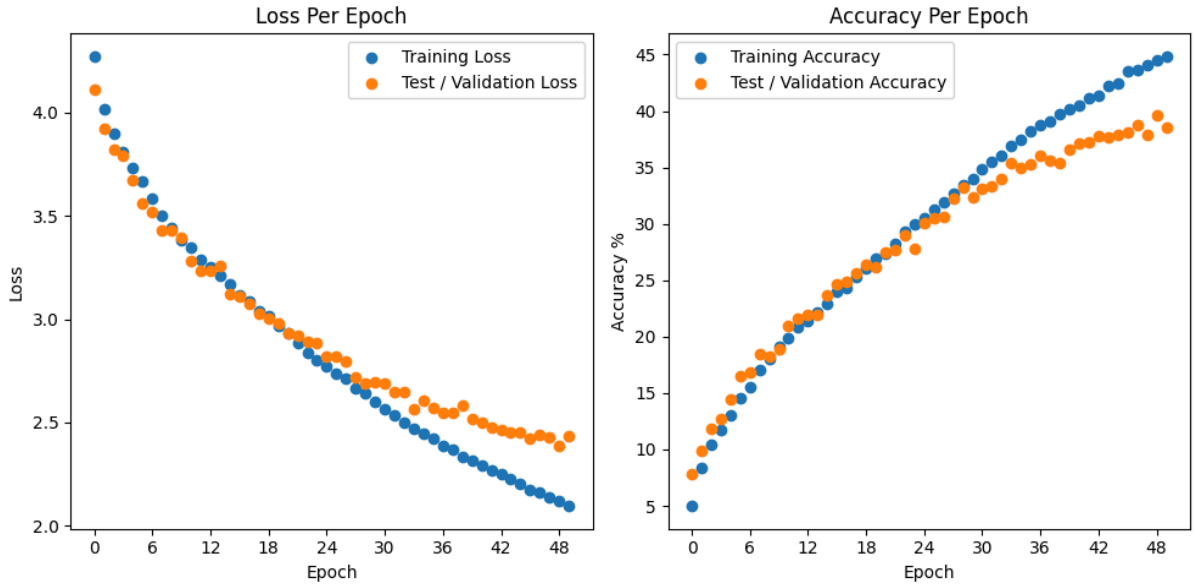
Last Best Epoch: 67, Train Time ≈ 1 hr, 7 min, 38 sec, Avg Epoch Time ≈ 0 min, 58 sec
Train Loss: 2.67836, Train Accuracy: 32.79%
Test Loss: 2.85204, Test Accuracy: 30.31%
Model Parameters: 6,398,308, MACS: 6,979,797,024

(4x4) -- 256 -- (4,4)



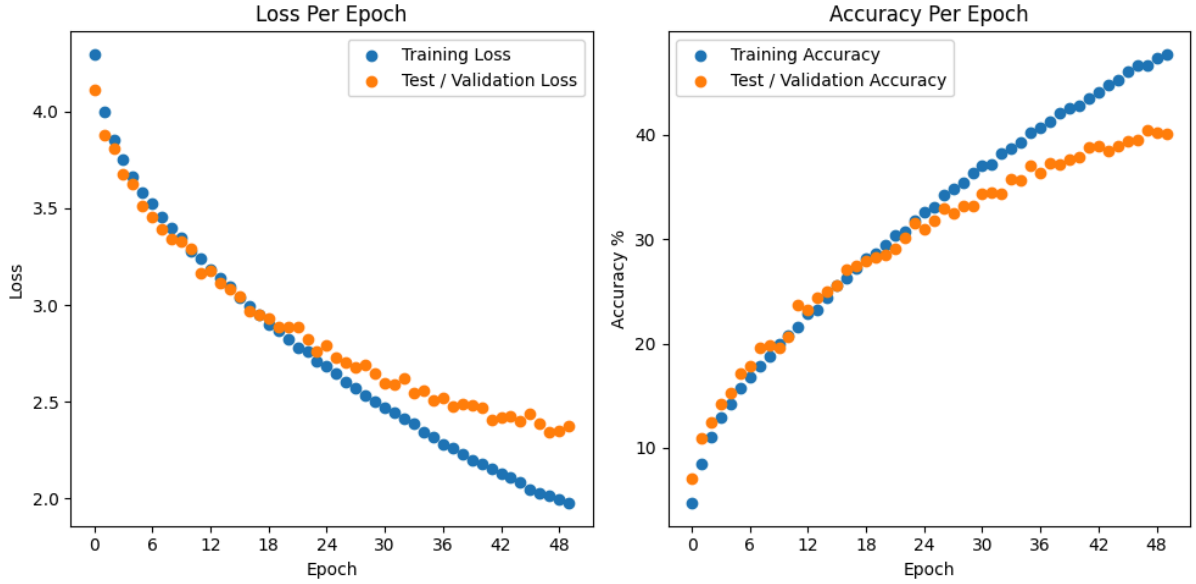
Last Best Epoch: 49, Train Time ≈ 0 hr, 42 min, 11 sec, Avg Epoch Time ≈ 0 min, 50 sec
Train Loss: 2.10139, Train Accuracy: 44.90%
Test Loss: 2.45206, Test Accuracy: 38.26%
Model Parameters: 3,214,692, MACS: 13,705,822,224

(4x4) -- 256 -- (8,2)



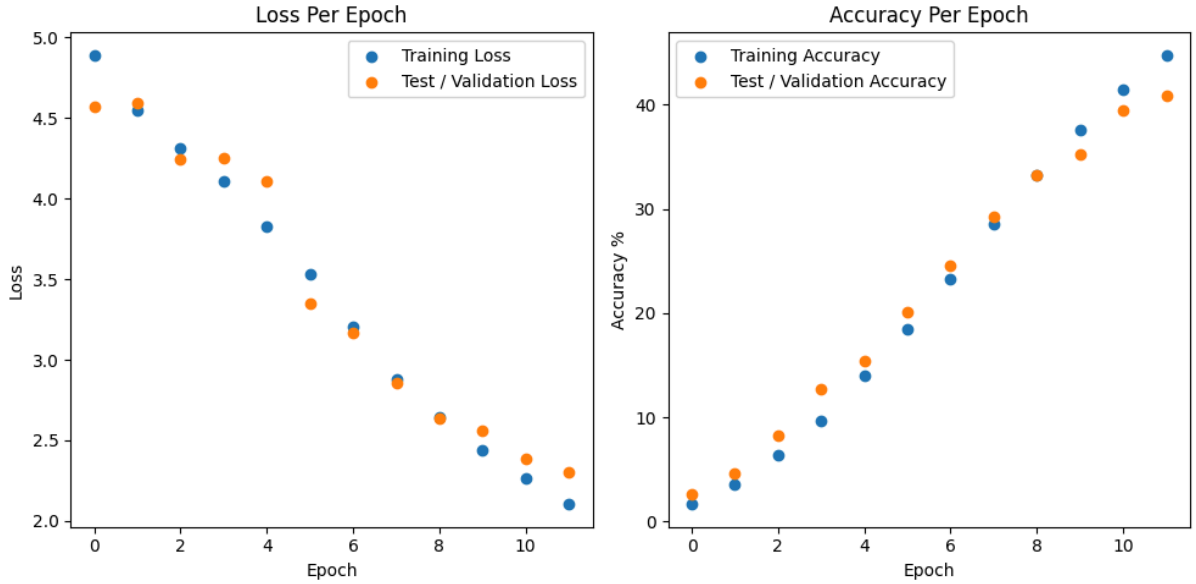
Last Best Epoch: 49, Train Time ≈ 0 hr, 55 min, 50 sec, Avg Epoch Time ≈ 1 min, 7 sec
Train Loss: 2.09357, Train Accuracy: 44.87%
Test Loss: 2.43254, Test Accuracy: 38.56%
Model Parameters: 6,373,732, MACS: 27,358,609,440

(4x4) -- 256 -- (8,4)



Last Best Epoch: 49, Train Time ≈ 0 hr, 57 min, 0 sec, Avg Epoch Time ≈ 1 min, 8 sec
Train Loss: 1.97499, Train Accuracy: 47.77%
Test Loss: 2.37777, Test Accuracy: 40.14%
Model Parameters: 6,373,732, MACS: 27,358,609,440

ResNet-18



Last Best Epoch: 11, Train Time ≈ 0 hr, 33 min, 26 sec, Avg Epoch Time ≈ 2 min, 47 sec
Train Loss: 2.10346, Train Accuracy: 44.77%
Test Loss: 2.30414, Test Accuracy: 40.85%
Model Parameters: 165,969,812, MACS: 223,606,594,560

Observations

MODEL	MACS	NUM PARAMS	EPOCH TRAIN TIME	TEST ACCURACY
(8x8) -- 256 -- (8,4)	6,979,797,024	6,398,308	0 min, 58 sec	30.31%
(4x4) -- 256 -- (4,4)	13,705,822,224	3,214,692	0 min, 50 sec	38.26%
(4x4) -- 256 -- (8,2)	27,358,609,440	6,373,732	1 min, 7 sec	38.56%
(4x4) -- 256 -- (8,4)	27,358,609,440	6,373,732	1 min, 8 sec	40.14%
ResNet-18	223,606,594,560	165,969,812	2 min, 47 sec	40.85%

Comparing ViT architectures reveals that lower patch size and lower head and layer count led to the best results. This is probably due to the dataset being so small and of low resolution.

ResNet-18 had the best Final Accuracy however, it has much higher MACS and parameter count (over 8x MACS and 27x parameter count) for only a 0.7% increase in test accuracy. ResNet also only took 11 Epochs to train compared to the 49 Epochs that ViT took; thus, ResNet had a faster training time.

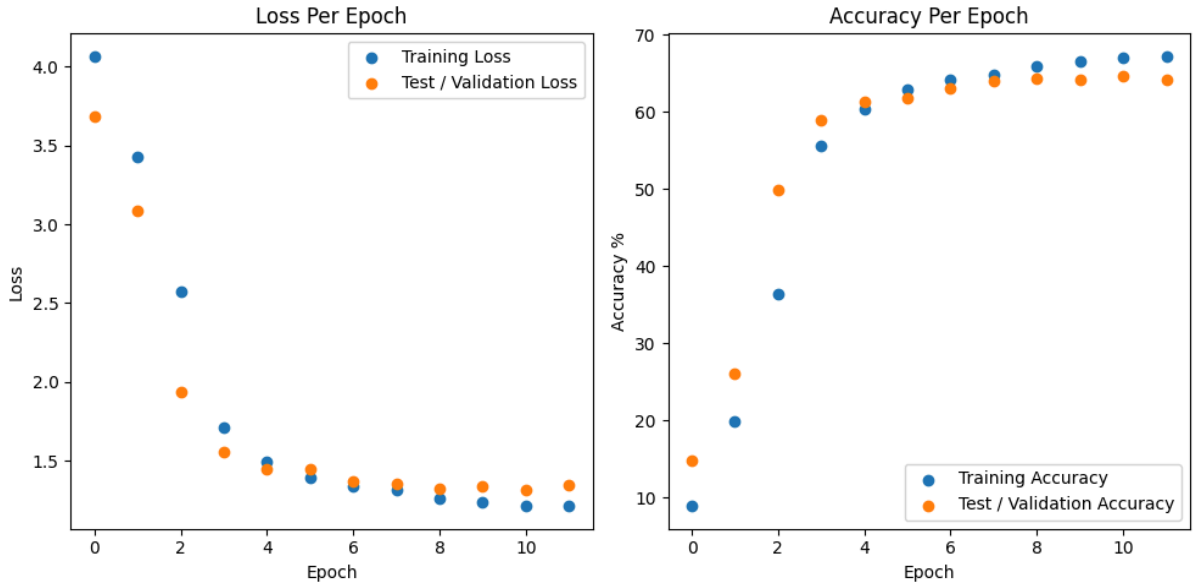
In summary, ResNet-18 is a vastly more complex model and had one of the best test accuracies compared to ViT. It also took significantly less time to train.

Problem 2

You will fine-tune pretrained Swin Transformer models from the Hugging Face Transformers library—specifically the Tiny (microsoft/swin-tiny-patch4-window7-224) and Small (microsoft/swin-small-patch4-window7-224) variants - on CIFAR-100 and compare their performance to a Swin Transformer trained from scratch. Start by loading these pretrained models using SwinForImageClassification.from_pretrained(), adjusting the classification head for 100 classes and freezing the backbone to train only the head. Fine-tune both models for 5 epochs with a batch size of 32, a learning rate of 2e-5, the Adam optimizer. Measure training time per epoch and final test accuracy for each. Then, implement a Swin Transformer from scratch (e.g., adapting a tiny version from Problem 1 or prior examples), train it on CIFAR-100 from random initialization for 5 epochs with a batch size of 32, a learning rate of 0.001, and the same preprocessing, and record its training time and test accuracy. Compare Swin-Tiny (pretrained), Swin-Small (pretrained), and the scratch model based on test accuracy after 5 epochs, training time and model size in terms of number of parameters. In your report, present a table with these results and discuss the benefits and drawbacks of fine-tuning versus training from scratch, the differences between Swin-Tiny and Swin-Small in this context, and reasons why pretrained models might outperform or underperform the scratch model.

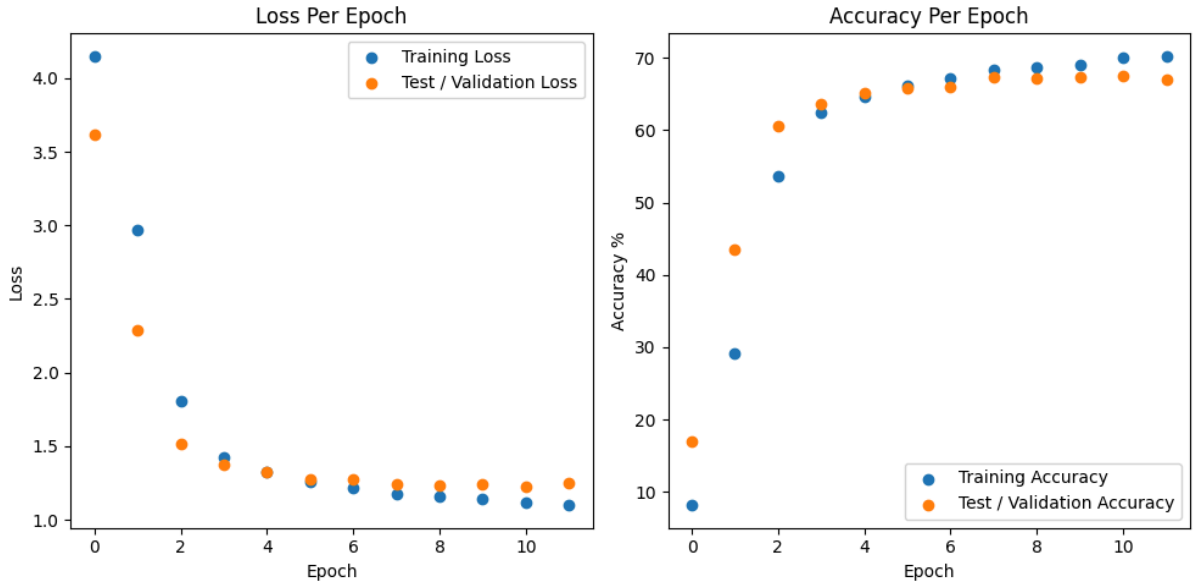
Fine-Tuned SWIN

Tiny SWIN



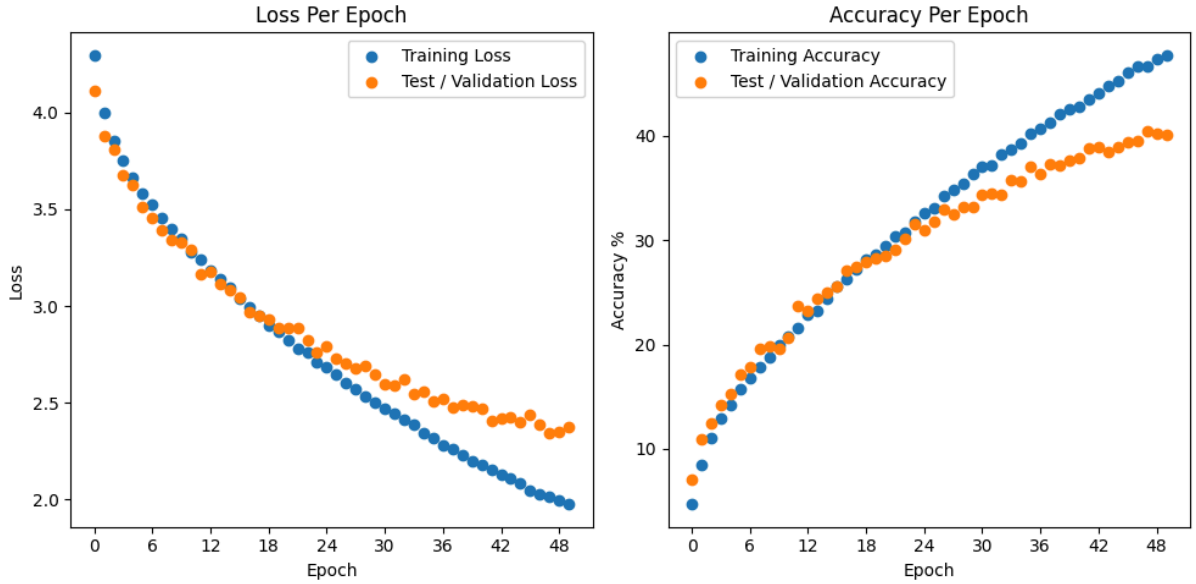
Last Best Epoch: 10, Train Time ≈ 0 hr, 44 min, 44 sec, Avg Epoch Time ≈ 3 min,43 sec
Train Loss: 1.21547, Train Accuracy: 66.97%
Test Loss: 1.31412, Test Accuracy: 64.60%
Model Parameters: 27,668,065, MACS: 185,004,916,753

Small SWIN



Last Best Epoch: 10, Train Time ≈ 1 hr, 4 min, 41 sec, Avg Epoch Time ≈ 5 min,23 sec
Train Loss: 1.11348, Train Accuracy: 69.94%
Test Loss: 1.22347, Test Accuracy: 67.43%
Model Parameters: 48,985,969, MACS: 321,072,594,973

Best ViT From Problem 1: (4x4) -- 256 -- (8,4)



Last Best Epoch: 49, Train Time ≈ 0 hr, 57 min, 0 sec, Avg Epoch Time ≈ 1 min, 8 sec
Train Loss: 1.97499, Train Accuracy: 47.77%
Test Loss: 2.37777, Test Accuracy: 40.14%
Model Parameters: 6,373,732, MACS: 27,358,609,440

Observations

MODEL	**FINAL TEST ACCURACY**	**TRAIN TIME**	**MODEL SIZE (Number of Parameters)**
Tiny SWIN (Pre-Trained)	64.60%	44 min, 44 sec	27,668,065
Small SWIN (Pre-Trained)	67.43%	1 hr, 4 min, 41 sec	48,985,969
Best ViT: (4x4) -- 256 -- (8,4)	40.14%	57 min, 0 sec	6,373,732

Pre-trained models gave approximately a 20% increase in test accuracy thanks to the fact SWIN was pre-trained on other generic data, allowing it to develop a better sense of reasoning compared to ViT. All models took approximately the same amount of time to train.

According to the data I have, pre-trained models have no drawbacks and only serve to increase accuracy. However, theoretically, a pre-trained model lacks specificity and has a lower theoretical upper limit of accuracy.