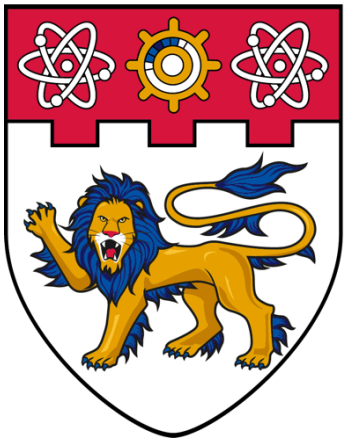

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Neural Networks and Deep Learning

Assignment 2

Fashion-MNIST Clothing Classification

https://github.com/Ruin9999/clothing-classification

Prepared by:

Teo Shao Qi (U2222637B)

Yong Shun Jie (U2221938C)

Nayoyos Kenneth Moli Akiapas (U2222647L)

College of Computing and Data Science

AY 2024/2025

Abstract

This report documents the experimentation of classifying clothing images using the Fashion-MNIST dataset. The data consisted of Zalando's article images - 60,000 for training and 10,000 for testing - uniformly distributed among ten classes. Data augmentation was applied to improve the model's generalisation ability and increase the data count. Advanced techniques like Mixup and RandAugment were applied and found to improve performance across all models explored.

Various network architectures, including convolutional neural networks (CNN) and Vision Transformer (ViT), were analysed and refined for classification performance. An overall accuracy of ~91%, achieved by VGG-8, was observed on the test dataset. The general conclusion of the assignment aligns with Occam's Razor, where the simpler model outperforms larger, deeper networks. This can be further explained by the nature of the classification problem. The images provided by the Fashion-MNIST dataset were in grayscale and small (28 x 28 x 1), and the clothing items were already placed in the center of the image with a plain black background. This greatly simplifies the task, and thus, the solution to the smaller and simpler classification problem was a model with an equivalent level of scale and complexity.[1]

[1] Training and inference code can be found at https://github.com/Ruin9999/clothing-classification

Table of Contents

List of Figures

List of Tables

Chapter 1. Data Collection and Preprocessing

We use the Fashion-MNIST dataset [1] to train and test our models. The dataset contains 70,000 images of clothes categorized into one of ten classes below.

Table 1. List of categories in the Fashion-MNIST dataset.

T-Shirt / Top	Trouser	Pullover	Dress	Coat
Sandal	Shirt	Sneaker	Bag	Ankle Boot

This dataset contains 70,000 28×28 grayscale images, split into a training and test set, with the image distribution equally split between the ten classes. The training set contains 60,000 images, with 6000 images reserved for each class, with the test set contains the remaining 10,000 split into 1000 images for each class.

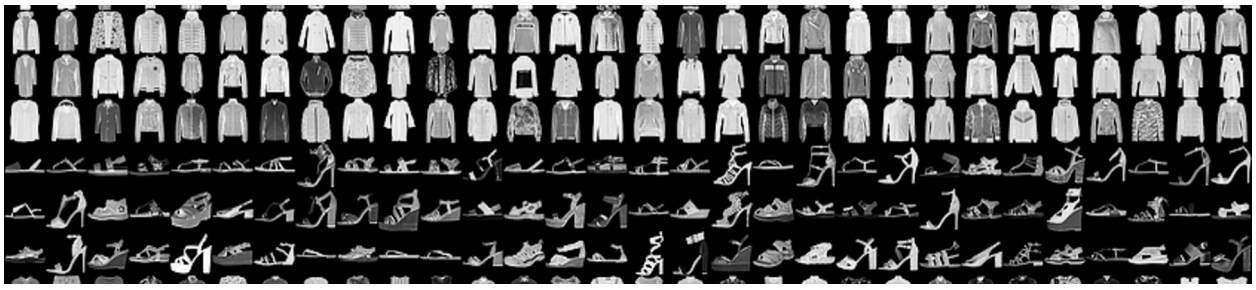

Fig 1. An example of images in the Fashion-MNIST dataset.

1.1 Data Augmentation

We implement two data augmentations on top of the dataset. Specifically, we implement MixUp [2] and RandAugment [3] to increase our dataset size and help our models better generalize for any unseen data.

1.1.1 MixUp

Introduced by Zhang Hongyi et al., [2] proposed the novel data augmentation routine, MixUp, that helps to overcome some fallbacks of Empirical Risk Minimization (ERM) [8]. Essentially, MixUp generates a “smoothened” data sample by taking a blend of two random images from the original dataset. Along with this, MixUp also generates a label for this new sample, combining the categories of both the image samples.

Specifically, MixUp generates new data samples using the formula [2]:

$$\bar{x} = \lambda x_i + (1 - \lambda)x_j, \quad \bar{y} = \lambda y_i + (1 - \lambda)y_j, \quad \lambda \sim \beta(\alpha, \alpha)$$

Where x_i and x_j are two randomly chosen samples, and y_i and y_j are their corresponding one-hot encoded labels. λ is the mixing factor, sampled from a beta distribution β with the hyperparameter α. When α is small, MixUp produces samples close to the original samples, while when α increases, more evenly weighted interpolations are created.

Fig 2. An example of MixUp applied on the Fashion-MNIST dataset.

1.1.2 RandAugment

In their 2019 paper, Cubuk et al. introduced RandAugment [3], an automated data augmentation pipeline that drastically reduced the augmentation policy search space while improving general model performance.

RandAugment does this by parameterizing the augmentation process into two hyperparameters N and $\mathrm{M}$where N is the number of augmentation operations applied sequentially to an image and M is a global magnitude that controls the strength of all transformations. During the data augmentation process N number of augmentations are randomly picked from a list of transformations and sequentially multiplied and applied with M to produce the new sample.

Table 2. The different augmentations available in RandAugment

Identity	AutoContrast	Equalize	Rotate
Solarize	Color	Posterize	Contrast
Brightness	Sharpness	Shear-X	Shear-Y
Traslate-X	Translate-Y		

Chapter 2. Methodology

In this section, we explore different neural network architectures and analyze their ability to perform on the multi-classification task. More precisely, we examine Convolutional Neural Networks (CNNs), Vision Transformers (ViTs) and their variants.

We standardize some non-model-specific hyperparameters and normalize all data samples into the range [-1, 1] before training and testing our models. Any model-specific hyperparameter not explicitly stated in the report has been fine-tuned; however, this tuning has been omitted in this report for brevity.

Table 3. Preset hyperparameters for training and testing across models

Loss Function	Optimizer	Patience	Epochs	Batch Size
Cross Entropy Loss	Adam Optimizer	5	100	64

2.1 Convolutional Neural Networks

CNNs are highly effective for image classification tasks, making them feasible for the Fashion MNIST dataset. Early architectures like LeNet-5 [12] and AlexNet [11] were key for realising modern CNN models but were either too shallow for complex datasets or prone to overfitting. To address these issues, two later architectures, VGG [4] and ResNet [5], were explored to classify the MNIST dataset into 10 distinct fashion categories. As these models were designed for larger, coloured images, several modifications were made to accommodate the 28×28 grayscale images and improve training efficiency and accuracy.

2.1.1. Visual Geometry Group

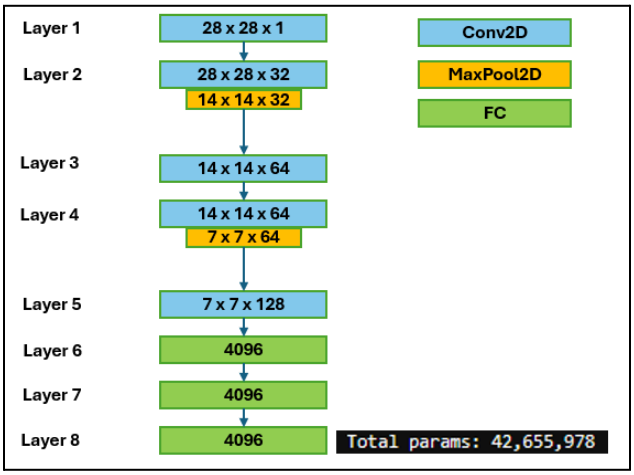

Fig. 3: VGG8 for Fashion-MNIST dataset

Visual Geometry Group (VGG) architectures are known to contain several MaxPooling layers. While they effectively reduce the spatial dimensions of feature maps for larger images, the smaller MNIST images risk shrinking to a single point. Since there were no fixed number of layers and kernels to the VGG architecture, the model was started with 8 layers (thus VGG8) to avoid excessive shrinkage of images. The amount of MaxPooling layers (not included in VGG layer count) was set so that the smallest image resolution would be 7x7, allowing the network to retain essential features without losing important spatial information before flattening.

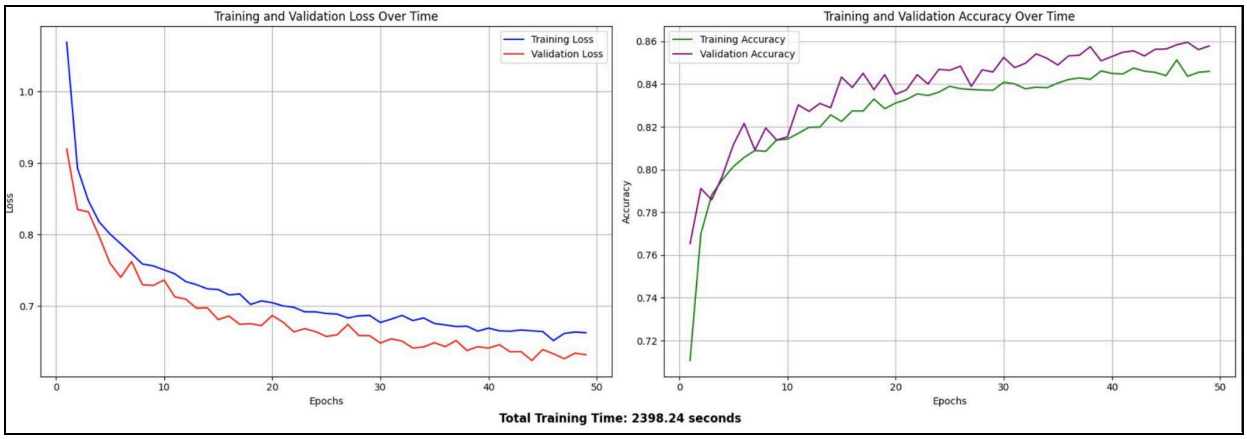

Fig. 4: VGG8 loss/accuracy plot

An average test accuracy of 91.8% was observed from VGG8, with training taking ~2400 seconds for 48 epochs (~ 50 seconds per epoch). The implementation saw over 42 million parameters, contributing to the long training time. An effort to reduce this was made by replacing Layer 2 to 5 with Depthwise-Separable Convolution (DSC). Since MNIST images only contain one channel, DSC for the first layer was unnecessary. DSC reduced the total parameters count by ~100,000, although no significant reduction in training time and improvements to accuracy were made.

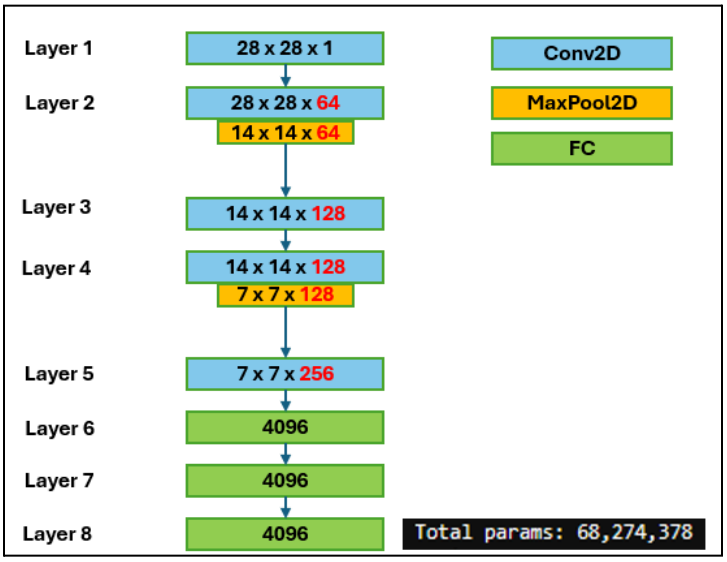

Fig. 5: Increasing kernels in VGG8

An effort to improve accuracy was then made by increasing the number of kernels per layer. While improvements to validation loss/accuracy were observed, the average test accuracy fell to **86%**, an indication of overfitting.

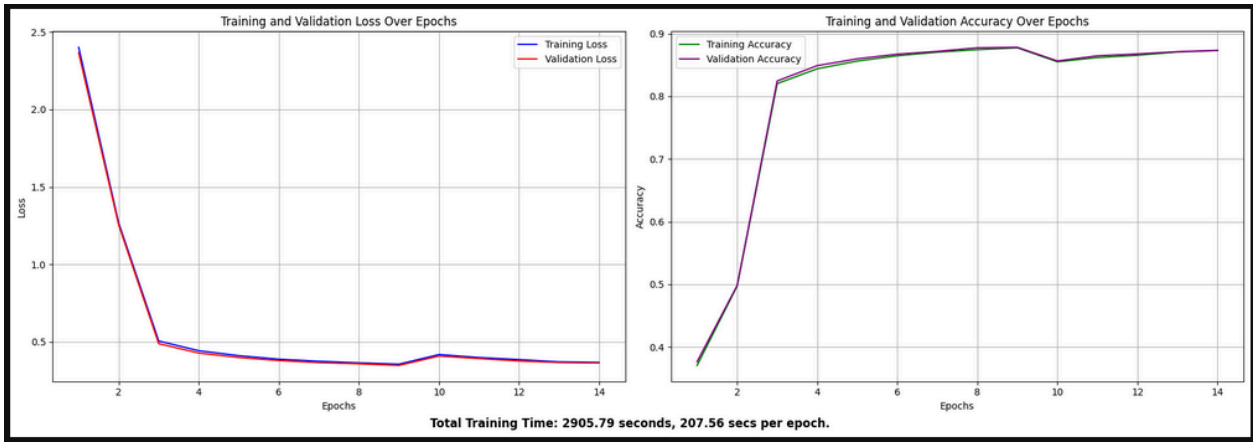

Fig. 6: VGG8 (increased kernels) loss/accuracy plot

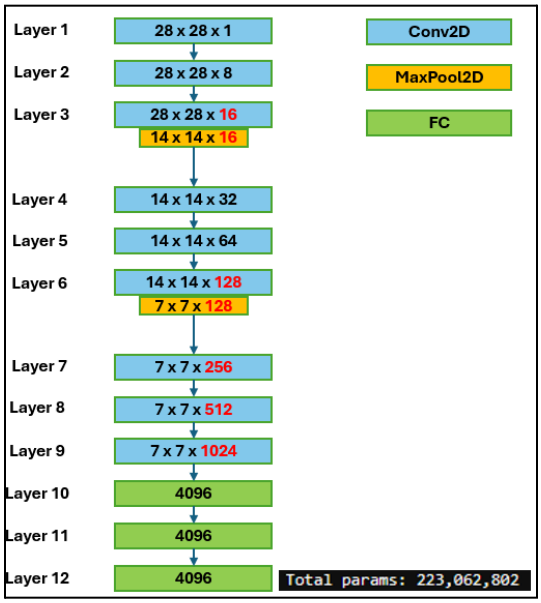

Fig. 7: VGG12 for Fashion-MNIST dataset

To address this, a deeper network with 12 layers (VGG12) was experimented with to increase the model's complexity while mitigating overfitting. The number of MaxPooling layers in the network was maintained, with zero paddings applied to ensure that the resolution remained at 7x7 before flattening.

```
Epoch 1 - Train Loss: 5.1239, Train Acc: 0.1152, Val Loss: 5.6622, Val Acc: 0.1168,  Test Loss: 2.3060, Test Acc: 0.0995
Epoch 2 - Train Loss: 2.3056, Train Acc: 0.1029, Val Loss: 2.3045, Val Acc: 0.1051,  Test Loss: 2.3053, Test Acc: 0.1004
Epoch 3 - Train Loss: 2.3055, Train Acc: 0.1008, Val Loss: 2.3045, Val Acc: 0.1031,  Test Loss: 2.3056, Test Acc: 0.0995
Epoch 4 - Train Loss: 2.3056, Train Acc: 0.1028, Val Loss: 2.3046, Val Acc: 0.1041,  Test Loss: 2.3078, Test Acc: 0.0998
Epoch 5 - Train Loss: 2.3057, Train Acc: 0.1015, Val Loss: 2.3047, Val Acc: 0.1039,  Test Loss: 2.3076, Test Acc: 0.1007
Epoch 6 - Train Loss: 2.3056, Train Acc: 0.1013, Val Loss: 2.3045, Val Acc: 0.1031,  Test Loss: 2.3111, Test Acc: 0.0998
Epoch 7 - Train Loss: 2.3061, Train Acc: 0.1005, Val Loss: 2.3051, Val Acc: 0.1027,  Test Loss: 2.3057, Test Acc: 0.1001
```

Fig. 8: VGG12 performance failed to improve after several epochs

Despite the increased depth, losses and accuracies failed to improve with epochs. A significant increase in training time per epoch was also noted. This was likely caused by vanishing gradients, a result of small images being propagated through many layers. As gradients diminished with each layer, the earlier layers struggled to update effectively, making deeper VGG networks less practical for the Fashion MNIST dataset.

2.1.2. ResNet

ResNet [4] was explored to address the problem of prolonged training times and vanishing gradients of deeper VGG networks. It uses residual connections to allow information to bypass layers, helping the network learn more effectively. These shortcut connections result in faster and more stable training, enabling the exploration of deeper architectures without issues like vanishing gradients.

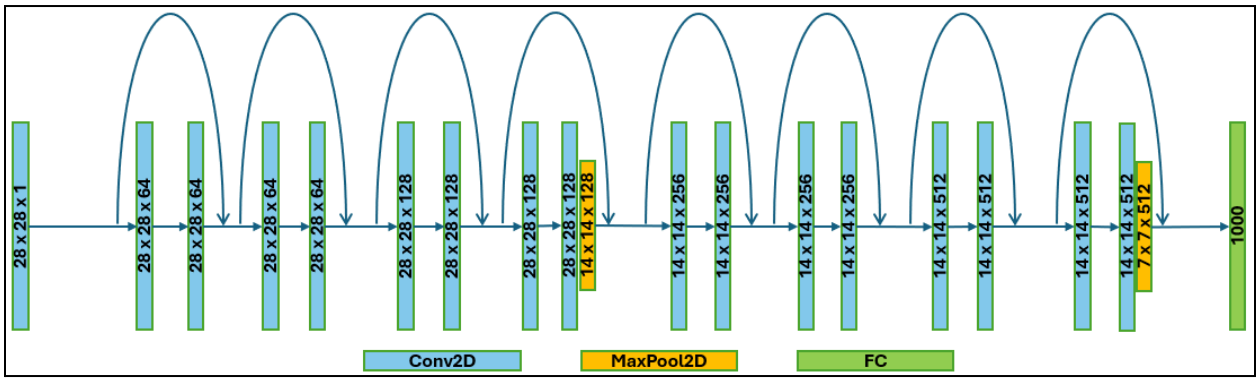

Fig. 9: ResNet18 for MNIST dataset

The architecture was implemented with 18 layers (ResNet18), a lightweight variant of the ResNet family suitable for MNIST datasets. As before, the number of MaxPooling layers and use of zero padding was allocated to ensure a minimum image resolution of 7x7 before flattening.

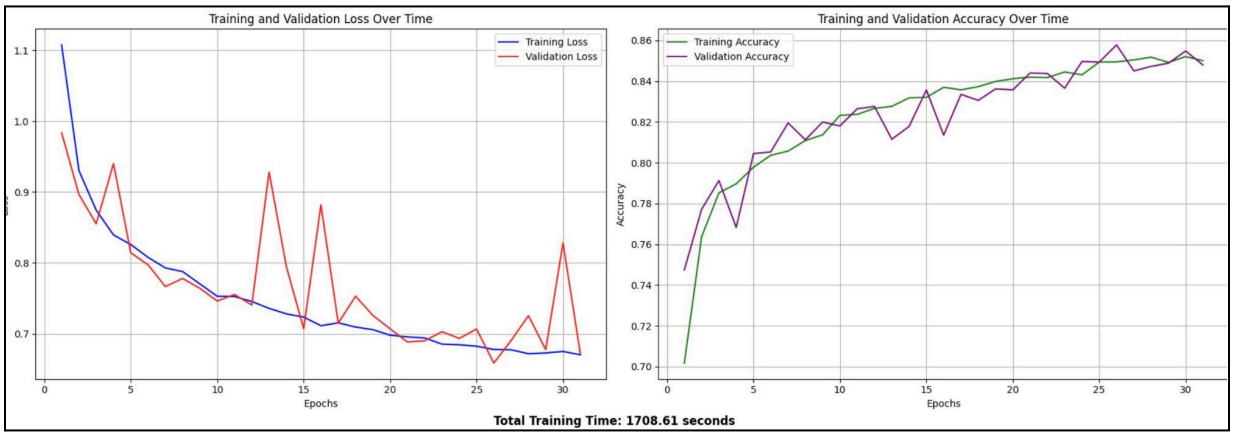

Fig. 10: ResNet18 loss/accuracy plot

Despite having more layers, ResNet18 took less time to complete one training epoch compared to VGG12, an advantage of residual connections in improving training efficiency. However, its performance was slightly poor compared to that of VGG8, with an average test accuracy of 88.6%.

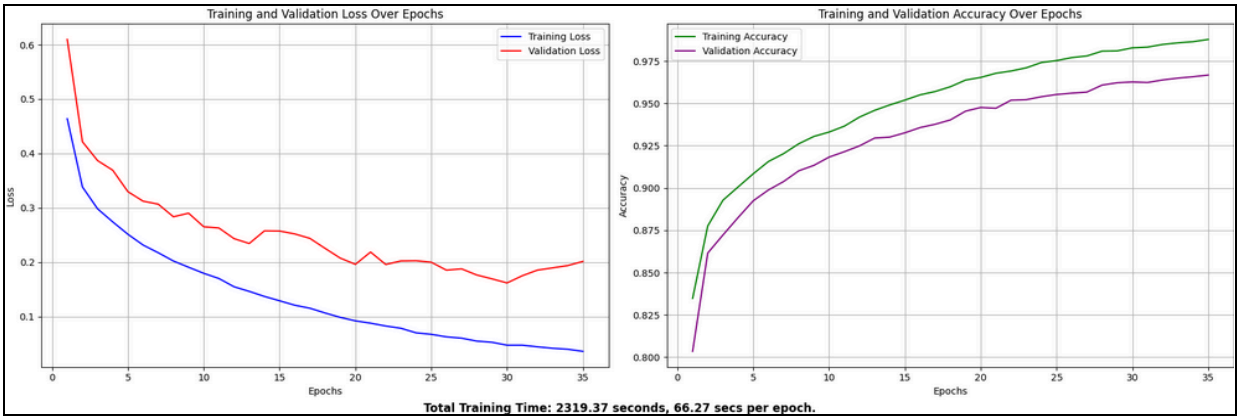

Fig. 11: ResNet18 (kernels reduced) loss/accuracy plot

The kernels of ResNet18 were increased to improve test accuracy. The average test accuracy improved to 90%, with some overfitting observed on the validation dataset.

Another attempt to improve the accuracy while minimising overfitting was done using dilated convolution [13] (dilation in Conv2d set to 1). This expands the model's receptive field to help better detect small and larger details. However, no significant improvement was noticed. This was likely because MNIST images were small, and skipping pixels in dilation may have caused the model to lose useful details in the images.

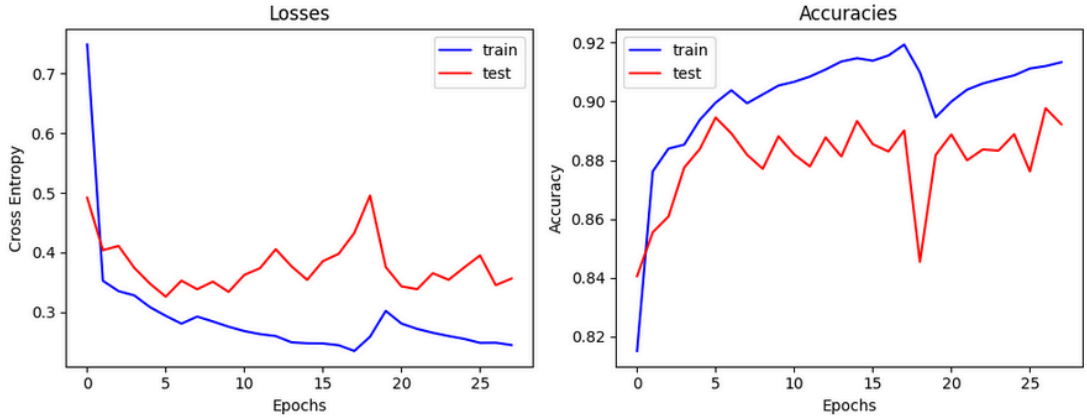

Fig. 12: ResNet18 (dilated convolution) loss/accuracy plot

2.2 Vision Transformers

Introduced by Ashish V. et al. in 2017, [9] presented the transformer architecture to solve Natural Language Processing (NLP) tasks like translation and have since become the foundation of many state-of-the-art models. In a typical transformer model, the overall architecture can be divided into two main components: an encoder and a decoder. The encoder focuses on understanding and generating latent representations of an input sequence. At the same time, the decoder handles the mixing and understanding of the latent representation fed in by the encoder, together with its previously generated outputs.

Central to the transformer's effectiveness is the attention mechanism, located inside both its encoder and decoder portions. Inside the encoder, the attention mechanism allows the model to understand the long-range relationships between each input token in the sequence and every other token, enabling it to focus on the most essential parts of the input. In the decoder, the same is done, but instead, between the output of the encoder and the model's previously generated output.

Building upon the success of transformers in text-to-text operations, [6] proposed the ViT. Using only the encoder portion of the transformer architecture, ViTs would first split images up into patches with pre-determined heights and widths. These images would then be flattened and fed into the transformer encoder, just like how text would be processed.

2.2.1 Base Vision Transformer

We implemented a modified version of the original ViT with improvements suggested in [10]. Mainly, we swapped out the use of a class token for global average-pooling and replaced the multi-layer perceptron layer with a simple linear layer to perform classification.

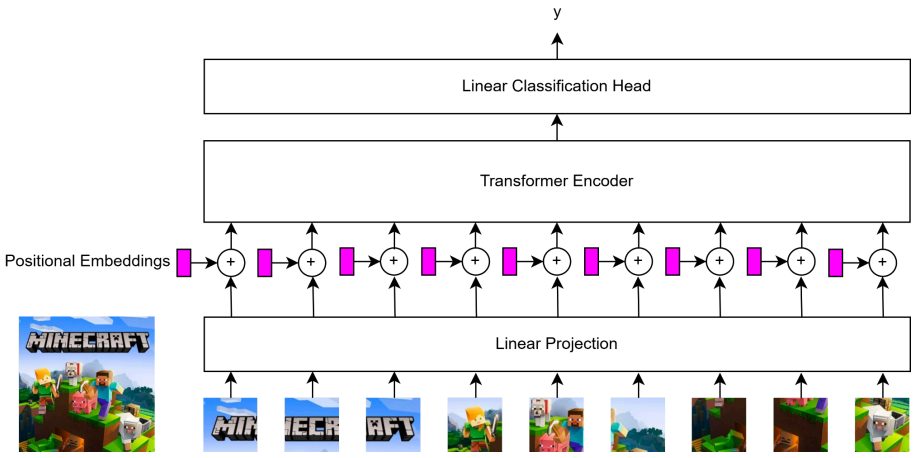

Fig. 13. The modified base vision transformer implemented in our report.

Since we were working with images with the 28×28 dimensions, a stark difference from the original paper using high-resolution images, we set a patch size of 7 instead of 16 and limited ourselves to 4 transformer layers. The other hyperparameters follow those from [10].

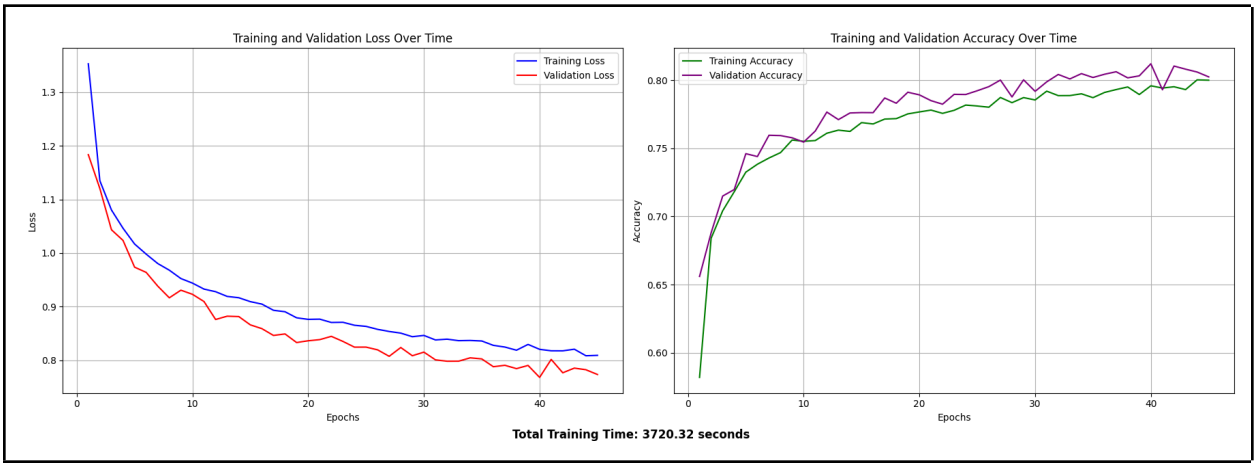

Fig. 14. Training accuracies and losses for the base vision transformer.

We find that our vision transformer can reliably classify images of clothes, achieving a test accuracy of 86.34%, training for 62 minutes. However, with previous studies proving ViTs to be able to outperform traditional CNNs on vision tasks, this is far from expected.

2.2.2 Depthwise Convolutional Vision Transformer

We suspected that the lower-than-expected accuracy might have been due to previous papers relying on large datasets before being able to outperform CNNs. To counteract this, we implement a variation of [7].

While the attention mechanism in transformers allows it to focus on the global relationship between pixels, it lacks an inductive bias when training on image or video modalities. In contrast, traditional CNNs utilize local filters and inherently possess an inductive bias. This bias allows them to be more efficient and reach convergence better than a ViT, especially when working with smaller datasets.

Zhang T. et al. [7], proposed adding a residual connection between transformer layers and passing it through a lightweight depthwise convolution module to help the ViT better learn on datasets with sizes similar to ours.

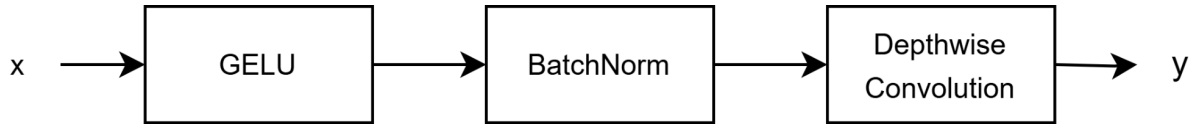

Fig 15. A visual representation of the depthwise convolution module.

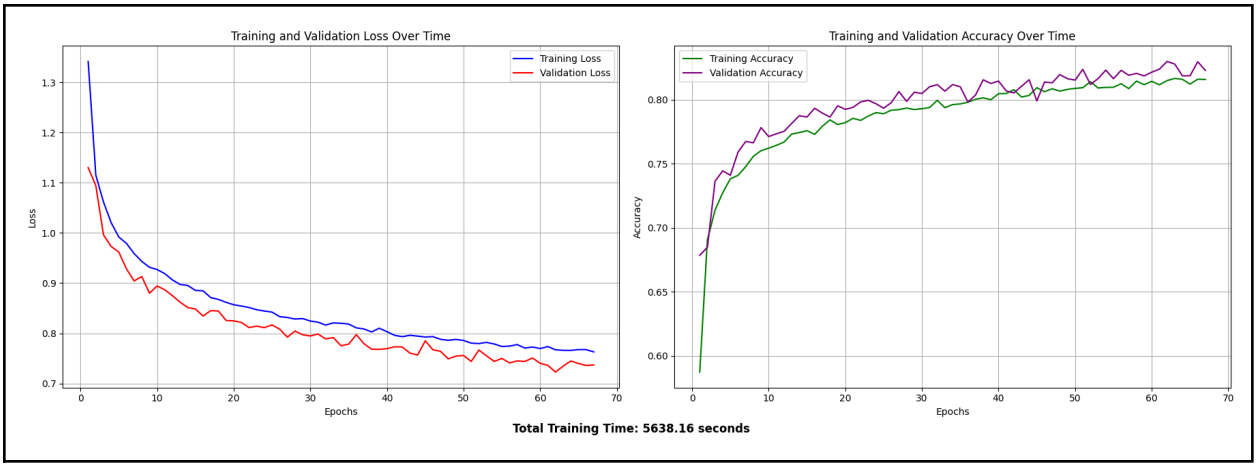

Fig. 16. Training accuracies and losses for the depthwise vision transformer.

We find that the depthwise convolution vision transformer performs better than the base vision transformer model, achieving a test accuracy of 88%, taking 62 minutes to train. We realized that the model unexpectedly took longer to train compared to the base transformer despite having very similar trainable parameter counts ($\sim 2\%$), but we theorize that this was solely because of the residual convolutional layer helping to stabilize training and preventing early stopping.

2.2.3 Patch-Free Vision Transformer

As we are working with images far tinier than what was introduced in the original ViT paper [6] (28×28 compared to 512×512), we theorized that the model might not have been able to accurately learn the dependencies between pixels due to the small patch sizes of 7×7 pixels. We train a slight modification to the ViT architecture, replacing the patching layer with a simple linear layer and positional encodings for each pixel coordinate.

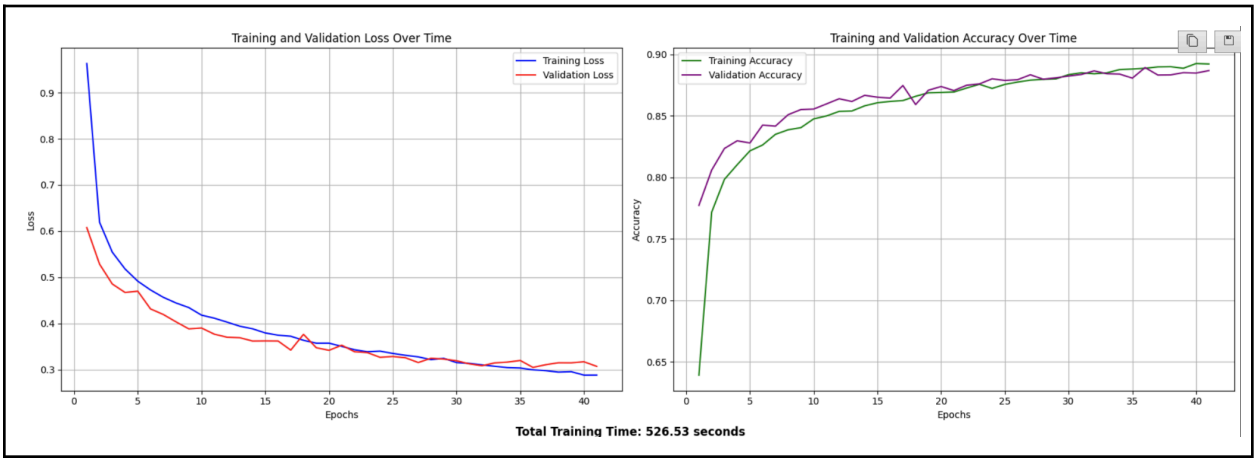

Fig. 17. Training accuracies and losses for the patchless vision transformer.

We recognise that the patchless version can perform slightly better than the base ViT at 86.7%, however, it is noted that there is a one-fold increase in trainable parameters, which we find unnecessary.

References

[1] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” arXiv.org, Aug. 25, 2017. https://arxiv.org/abs/1708.07747

[2] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” arXiv.org, Oct. 25, 2017. https://arxiv.org/abs/1710.09412

[3] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le V., “RandAugment: Practical automated data augmentation with a reduced search space,” arXiv.org, Sep. 30, 2019. https://arxiv.org/abs/1909.13719

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for Large-Scale image recognition,” arXiv.org, Sep. 04, 2014. https://arxiv.org/abs/1409.1556

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv.org, Dec. 10, 2015. https://arxiv.org/abs/1512.03385

[6] A. Dosovitskiy et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” arXiv.org, Oct. 22, 2020. https://arxiv.org/abs/2010.11929

[7] T. Zhang, W. Xu, B. Luo, and G. Wang, “Depth-Wise Convolutions in Vision Transformers for efficient training on small datasets,” Neurocomputing, p. 128998, Nov. 2024, doi: 10.1016/j.neucom.2024.128998

[8] Vapnik, V. (1998) Statistical Learning Theory. John Wiley & Sons, Chichester. - References - Scientific Research Publishing.” https://www.scirp.org/reference/referencespapers?referenceid=84632

[9] A. Vaswani et al., “Attention is all you need,” arXiv.org, Jun. 12, 2017. https://arxiv.org/abs/1706.03762

[10] L. Beyer, X. Zhai, and A. Kolesnikov, “Better plain ViT baselines for ImageNet-1k,” arXiv.org, May 03, 2022. https://arxiv.org/abs/2205.01580

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” 2012. https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[12] Wikipedia contributors, “LeNet,” Wikipedia, Mar. 26, 2025. https://en.wikipedia.org/wiki/LeNet

[13] F. Yu and V. Koltun, “Multi-Scale context aggregation by dilated convolutions,” arXiv.org, Nov. 23, 2015. https://arxiv.org/abs/1511.07122