Executive Report: Neural Systems Coursework

Student ID: P2963116

1 Background

This report outlines my implementation of artificial neural networks to build a classifier for use with the Fashion-MNIST [**?**] image dataset using PyTorch [**?**]. The dataset contains 60,000 greyscale images, each 28x28 pixels in size. The images depict 10 categories of fashion items (for example t-shirt, shirt, ankle boot) and is split into two subsets, 50,000 for training and 10,000 for testing. The training set is subdivided to include a random sample of 10,000 for use in validation, tests performed during training to monitor how the model performs on unseen data. We will record loss; a measure of error used in training to adjust model parameters along with accuracy as our primary metric for performance on unseen data.

1.1 Models

I implemented two types of neural network when building the classifiers. The first is a Multi-Layer Perceptron or ANN [Table 2a], a collection of 'neurons' each of which calculate a weighted linear sum based on the training parameters and input. Before passing the result to the neurons in the next layer, we pass the value through an activation function, helping the model learn complex patterns in the training data.
I also created a Convolutional Neural Network (CNN) [Table 3, Figure 1], a network type particularly effective at dealing with tabular data such as images. We perform convolution operations on the input with smaller matrices to generate feature maps and downsize the input matrix. After this the image is flattened and passed through a series of dense layers, which have a similar structure to the ANN.

2 Key Findings

Model	Training Time	Accuracy	Training Loss	Validation Loss
ANN	4.5 minutes	88.67%	0.0945	0.467
ANN (modified)	5 minutes	89.77%	0.152	0.298
CNN	12.5 minutes	91.95%	0.0666	0.309

Table 1: Key metrics after training

2.1 Notable Innovations

Training was completed over the full 30 epochs to emphasise the impact of potential issues. One of which is high validation loss compared to training loss, a phenomenon is known as overfitting. To mitigate this, I implemented dropout and batch normalisation. Dropout [**?**] randomly disables connections between layers during training, which helps to stabilise the loss over each epoch. Similarly, batch normalisation [**?**] attempts to stabilise training and improve convergence speed. Results show that these significantly reduced the validation loss by the end of training without having a noticeable negative impact on accuracy.
An alternative fix to this issue is through early stopping as the training results showed diminishing returns after 15-20 epochs, especially prior to adding dropout and batch normalisation.
Optimiser choice may further impact network performance and testing showed Adam [**?**] is superior due to its faster convergence and lower loss throughout training when compared to other algorithms such as Stochastic Gradient Descent [**?**].
Data was visualised using Seaborn [**?**] and SciKit Learn [**?**] through a series of graphs (Figure **??**) and by plotting the Confusion Matrix (Figure 2) associated with each model.

3 Evaluation

Further steps that could be taken to improve the models would be to augment training data by applying random rotations, reflections and scaling to the existing images. It would also be reasonable to adjust the architecture of my networks to reflect existing high performance classifier architectures such as ResNet, GoogLeNet and VisionTransformer, all of which would likely get higher accuracy scores when compared to my models due to their increased complexity.
Overall, while more complex, a CNN is far superior when it comes to computer vision classification task. This is likely due to the architecture facilitating the storage and understanding of spatial information by virtue of the matrix structure.

A Tables and Figures

Layer	Type	Size	Activation
Input	Image	28x28	-
Flatten	Flatten	784	-
1	Linear	392	ReLU
2	Linear	196	ReLU
3	Linear	98	ReLU
Output	Linear	10	Log Softmax

(a) Initial ANN

Layer	Type	Size	Activation
Input	Image	28x28	
Flatten	Flatten	784	
1	Linear	392	ReLU
	Batch Norm		
	Dropout	p=0.2	
2	Linear	196	ReLU
	Batch Norm		
	Dropout	p=0.2	
3	Linear	98	ReLU
	Batch Norm		
	Dropout	p=0.2	
Output	Linear	10	Log Softmax

(b) Modified ANN

Table 2: Network Structures

Layer	Type	Feature Maps	Size	Kernel	Stride	Activation
Input	Image		28x28x1			
1	Convolution	32	26x26x32	3x3	(1,1)	ReLU
2	Convolution	32	24x24x32	3x3	(1,1)	ReLU
	Max Pooling		12x12x32	2x2	(2,2)	
3	Convolution	64	10x10x64	3x3	(1,1)	ReLU
4	Convolution	64	8x8x64	3x3	(1,1)	ReLU
	Max Pooling		4x4x64	2x2	(2,2)	
	Flatten		1024			
5	Linear		128			ReLU
	Batch Norm					
	Dropout		p=0.4			
6	Linear		64			ReLU
	Batch Norm					
	Dropout		p=0.4			
Output	Linear		10			Log Softmax

Table 3: CNN Structure

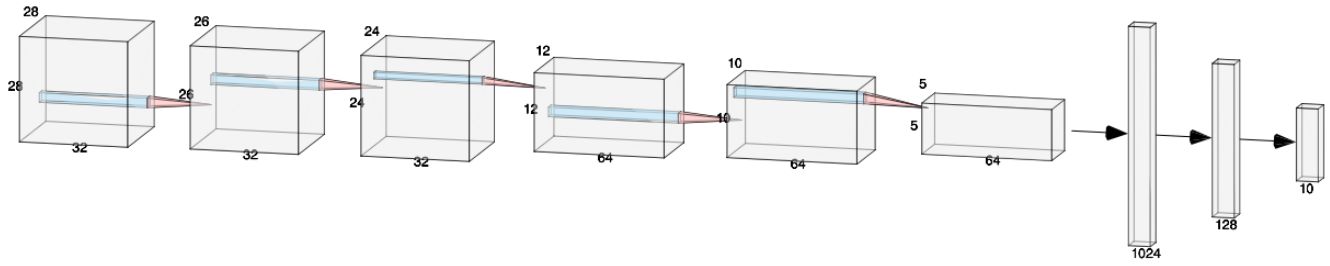
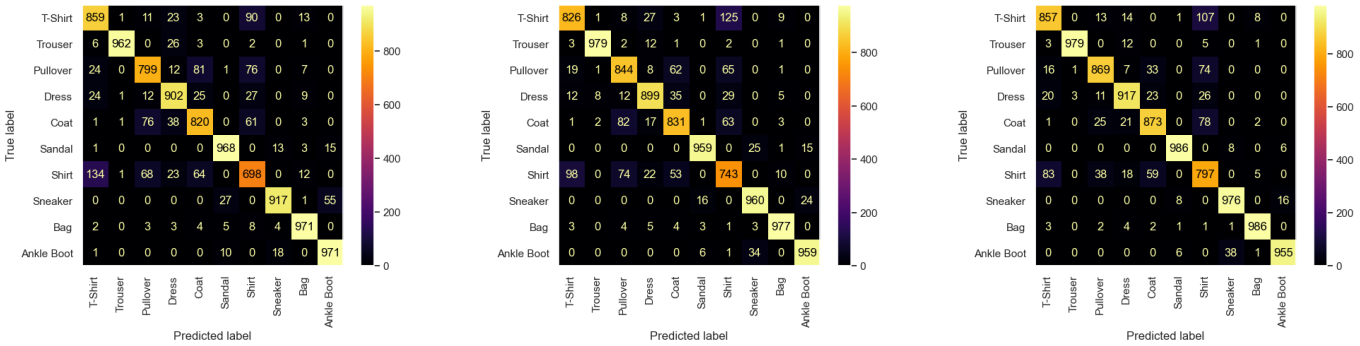


Figure 1: AlexNet-like Visualisation of my CNN Structure
Source: NN SVG [**?**]

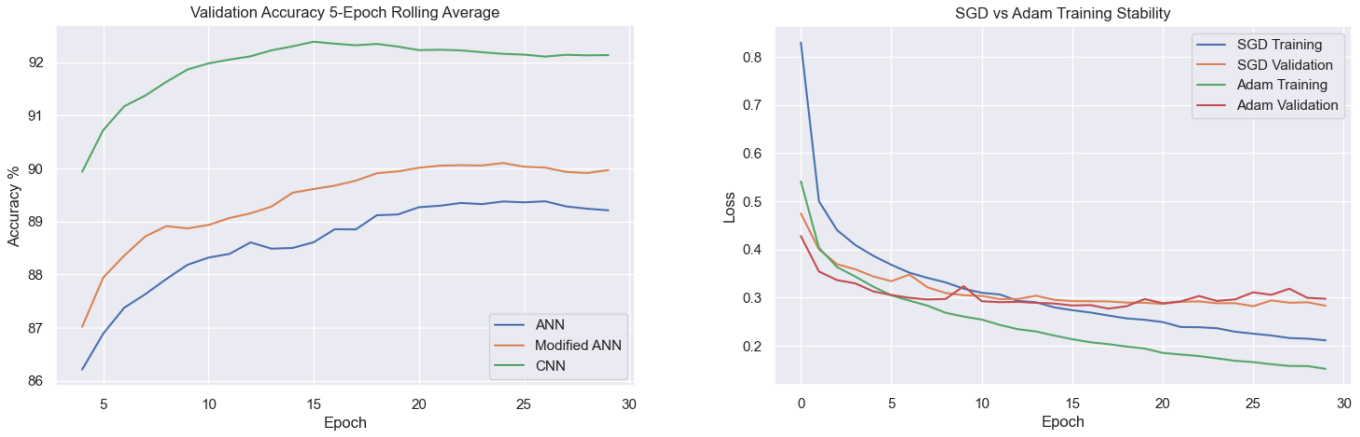


(a) ANN (b) ANN (modified) (c) CNN

Figure 2: Confusion Matrices

(a) ANN (b) ANN (modified) (c) CNN

Figure 3: Loss History Plot During Training



(a) Rolling Avg. Validation Set Accuracy (b) Optimiser Comparison

Figure 4: Comparison Graphs