### Deliverable 2: Technical Implementation Report

### 1. Data Handling

The Fashion-MNIST dataset, one of the benchmarks, was loaded to initiate the project with the help of the torchvision.datasets utility. A preprocessing pipeline that was critical was developed and made the images ready to the neural networks. The transforms.Compose used to build this pipeline initially used transforms.ToTensor to convert the 28x28 grayscale PIL images into transforms.Compose tensors and also scaled pixel values in the range[0, 255] of integer values to a range of [0.0, 1.0] of floats. After that, the transforms.Normalize(0.5, 0.5,) was utilized to mean-centre the data so to put the pixel values on the range of [ -1.0, 1.0]. The step of normalization is essential in assisting the models to converge faster and with more stability. This training dataset of 60,000 images was then split strategically with the help of random split to form a training set of 50,000 images and a validation set of 10,000 images which is imperative in checking overfitting. The individual 10,000-image test set was to be used in the last and the unbiased performance test. Lastly, all three sets (train, validation, and test) were then assigned DataLoader objects with a batch size of 64. A training loader was set with shuffle=True to randomize the data at every epoch, which is necessary to have an efficient stochastic gradient descent.

### 2. Network Topology (ANN)

The model that was initially applied was that of an Artificial Neural Network (ANN), or Multi-Layer Perceptron (MLP). This model was a non-spatial baseline model, which was meant to be a simple, fully-connected classifier. It was designed around a nn.Flatten() layer to transform the 2D image tensors (28x28) into 1D 784 features. A nn.Sequential stack was then applied to this flat vector. The initial hidden layer was a nn.Linear layer of 128 units, and then a non-linear activation function, nn.ReLU (), was added. To prevent overfitting a nn.Dropout(0.2) layer was included to regularize the model. The information was then passed to a second hidden layer of 64 units, which was also run by nn.ReLU(). The last layer was a nn.Linear layer that had 10 output units (one per class) and generated the raw logits of the classification. The whole architecture of this ANN does not consider the 2D structure of the picture and even all the pixels are viewed as independent features.

### 3. Network Topology (CNN)

The second was a Convolutional Neural Network (CNN), an architecture that was specifically created to process and learn 2D spatial information. Its architecture started with a convolutional stack. The initial nn.Conv2d had 32 3x3-filtered images on the single-channel input image with padding=1 to maintain the 28x28 size and then a nn.ReLU() activation. This was then followed by a nn.MaxPool2d layer with a 2 by 2 kernel that reduced the size of the feature maps to 14x14. This Conv-Pool block was repeated: a second round of nn.Conv2d, using 64 filters, was used, followed by nn.ReLU, and again another round of nn.MaxPool2d, downsizing the feature maps to 7x7. The hierarchy of features is successfully removed by this stack: the edges of the first layer, the textures and shapes of the second layer. This was then fed through a nn.Flatten() layer which turned the 3D feature data into a 1D feature (64 7 7 = 3136 features). This feature vector was then inputted into a classifier block, which comprises of a nn.Linear layer with 128 units, a nn.ReLU() activation, 10-unit strong nn.Dropout(0.5) layer to regularize it, and the last 10-unit nn.Linear output layer.

### 4. Training Setup

The two models had the same set up under which they were trained so that they could be fairly compared. The torch.optim.Adam optimizer was chosen which is an extremely efficient adaptive learning rate optimizer and in general converges quickly with little optimization. The learning rate was set to a standard of 1e-3 (0.001). The multi-class classification problem loss function was nn.CrossEntropyLoss, which is the industry standard since it is a combination of LogSoftmax and NLLLoss in one, efficient function. The two models were trained by 10 epochs. Any training and evaluation steps that had the presence of a GPU (cuda device) were explicitly transferred to it, making sure that there was a substantial reduction in the time-consumption of the computationally intensive training loops. tqdm progress bars were used which gave a real-time plot of the training progress each epoch, and each batch.

### 5. Evaluation

Models were tested on the 10,000-image test set held out on completion of training. CNN model showed definite improvement, with an ultimate test precision of 91.22%. ANN model trailed greatly behind with a score of 87.59. Such a difference in accuracy of 3.63 percent is a significant margin on this benchmark data.

This performance gap is also demonstrated by the training history graphs. The training and validation curves (in terms of loss and accuracy) of CNN follow each other closely. The validation loss steadily drops and the validation accuracy steadily increases to its peak which is a sign that the model is generalizing well. In sharp contrast, the graphs of the ANN indicate overfitting; the loss during validation is also irregular and still significantly higher than the loss during training and indicates that the model is memorizing the training data and not learning its internal patterns.

A graph of a line

AI-generated content may be incorrect.A graph of a graph of a graph

AI-generated content may be incorrect.

The performance in the classes through the confusion matrices analysis shows the areas where the models performed well and where they failed. Both models were very successful in recognizing objects with very different silhouettes like Trouser (986/1000 correct CNN) or Bag (984/1000 correct CNN). The main error that both models made was the confusion in separating the categories that are related to the shirt-like. The CNN was, however, much better at this. Indicatively, in the classification of the Shirt (row 6), the ANN falsely classified 148 of the Shirts as T-shirt/top. The CNN was far more critical and it made this same mistake just 93 times. This trend followed with other similar confusion classes such as the Pullover and Coat, and it was evident that CNN outdid the other on the best feature extraction.

A graph with numbers and a number in the center

AI-generated content may be incorrect.A graph with numbers and squares

AI-generated content may be incorrect.

### 6. Critical Analysis: ANN vs. CNN

The huge disparity in the performance is not by coincidence, it is an immediate consequence of architectural design. The weakness of the ANN to do this is that it lacks spatial awareness. The flattening of the image to a 784-unit vector kills all information relating to the adjacency of pixels. It has no built-in knowledge of the fact that two pixels are next to each other and must be taught that, and this is very inefficient. The CNN, on the other hand, is created to handle spatial data. Its patterns are learned through its 3x3 kernels, which are local receptive fields that move over the image. This design combines two strong ideas: parameter sharing (the same 32 filters are applied to the whole image, and thus it only needs to learn a single edge detector, not thousands of them), and hierarchical feature extraction (the first layer learns simple edges, the second ones unites them into textures, etc.). This quality of complex features being constructed out of simple ones is what enables the CNN to make the difference between a "Coat" and a "Pullover" by identifying such features as lapels or V-necks, and the ANN simply identifies two similarly-shaped blobs.

### 7. Reflection and Improvements

Although this is a good percentage of 91.22, it is possible to make it better. Data augmentation would be the most effective short-term measure. CRM-TTR would be artificially augmented by using transforms.RandomHorizontalFlip and transforms.RandomRotation to the training images. This would also teach the model that a Bag is still a Bag even inverted or a little object rotated, which makes the model stronger and even less overfitting. To achieve greater profits, we might use a more serious, modern network such as a ResNet (Residual Network). Such architectures are made to solve the vanishing gradient problem and are able to learn far more complex feature hierarchies. Lastly, a learning rate scheduler (e.g. ReduceLROnPlateau) might be included so as to automatically reduce the learning rate when the validation loss levels off, enabling the model to fine-tune its weights to achieve a small, final accuracy improvement.