ECE657A

Assignment 3

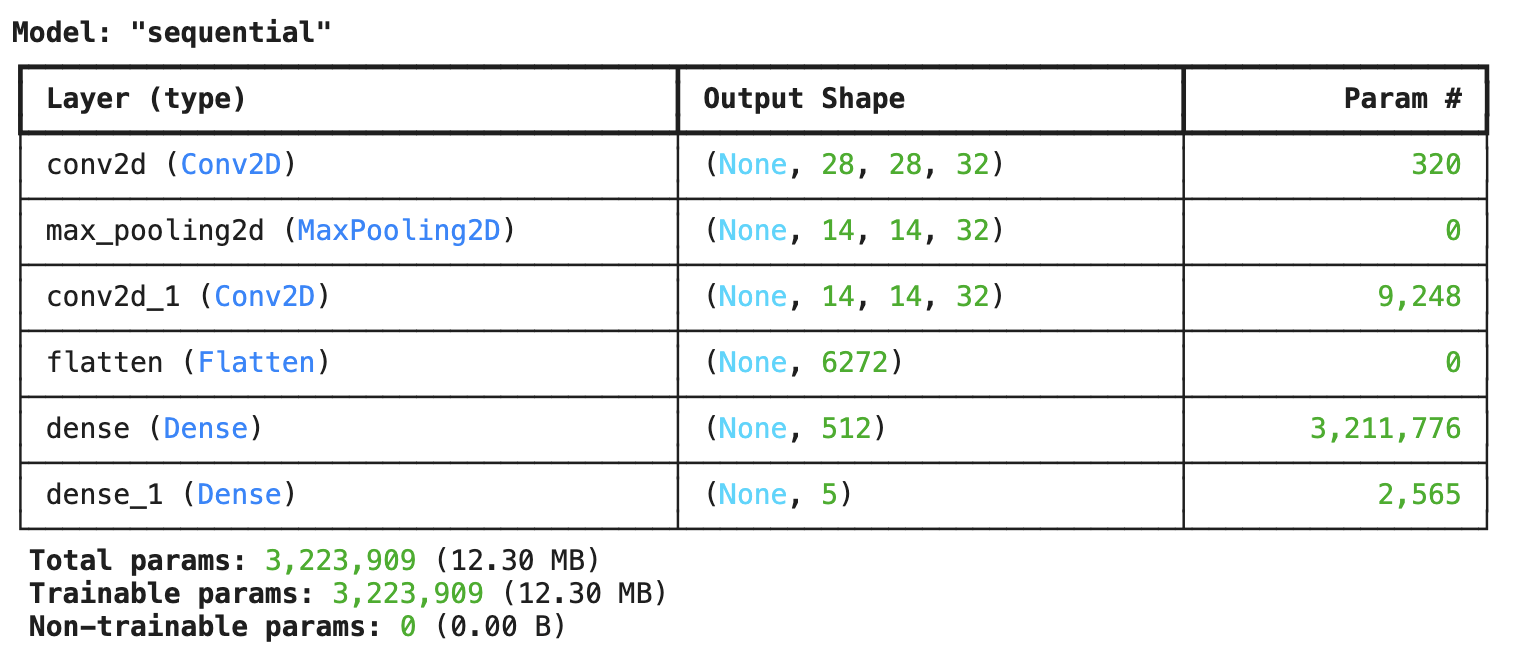
Munia Humaira and Mishaal Khan

April 4, 2025

#### 1. Classification with Convolutional Neural Networks

##### Q1. Default Network

Using TensorFlow, a default Convolutional Neural Network architecture has been modeled as instructed in the assignment. Figure 1 shows the model summary.



**Figure 1.** Default CNN model architecture

The model was then compiled and trained on the training data with a batch size of 64 for 10 epochs. All models were trained with a T4 GPU. The total training time was 50.55 seconds, with every epoch running for 4.7 seconds.

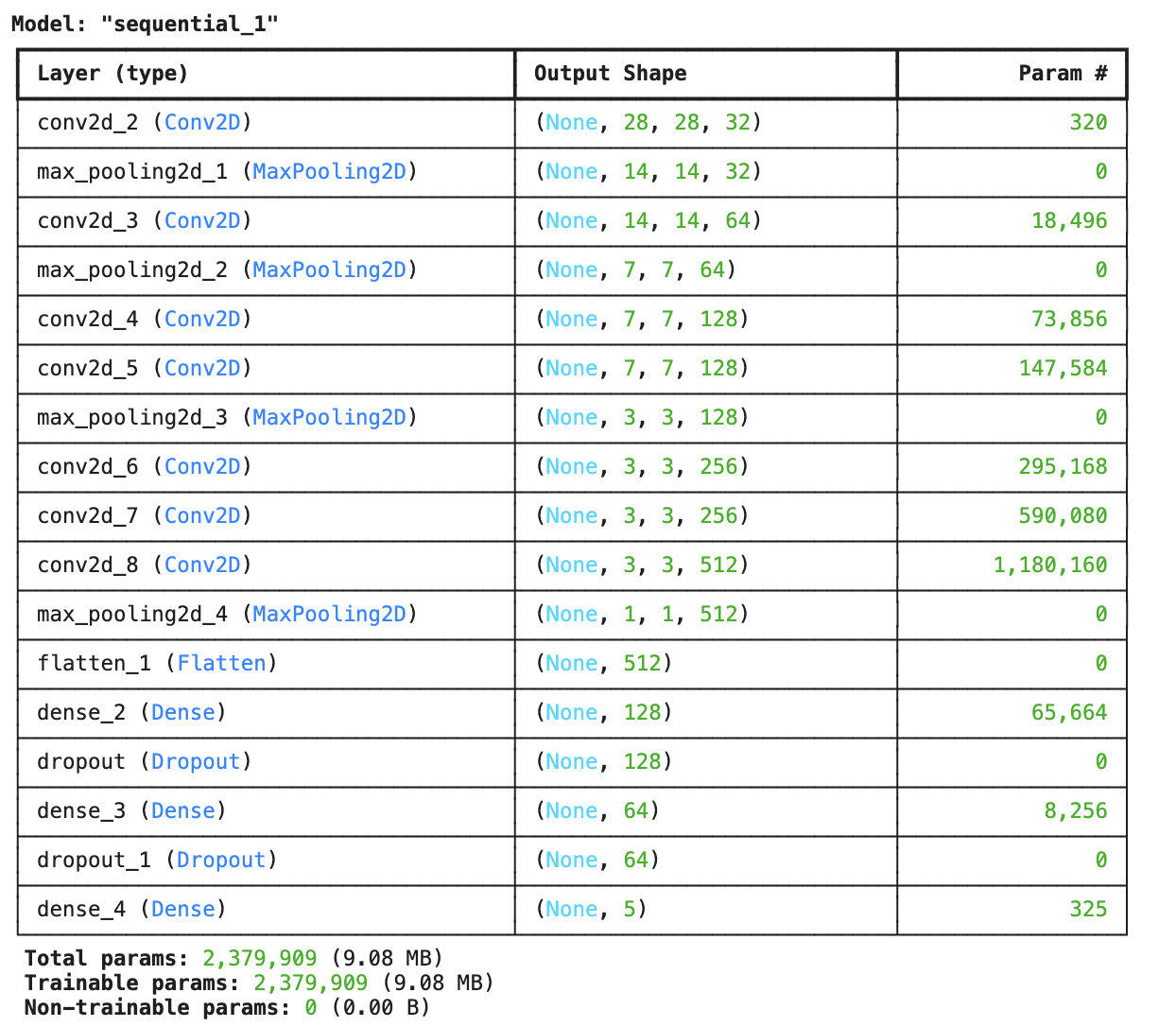
##### Q2. Own Improved Network

The default model architecture has been modified and adapted to make the network bigger and improve accuracy [1]. Our model has significantly more layers than the default one. However, we have tried to keep the network size limited so that it can be run with the limited resources. The first convolutional layers employ 32 filters, followed by 64, 128, 256, and 512 filters, while maintaining constant kernel sizes of 3x3. By halving the spatial dimensions, each max-pooling layer downsamples the feature maps.

Following the convolutional blocks, the feature maps undergo a 512-dimensional vector flattening process before being submitted to two fully connected (Dense) layers, each consisting of 128 and 64 units. With dropout rates set to 0.5, dropout layers are added after each Dense layer to reduce overfitting. Utilizing a softmax activation for multi-class classification, the final Dense layer produces a 5-unit vector representing the 5 classes. The model makes use of the Adam optimizer [2], which is a well-liked option for deep neural network training because of its capacity for adaptive learning rate. Adam uses the estimates of the first and second moments of the gradients to modify the learning rate for each parameter. For models with a lot of parameters, like this one, this leads to more effective and stable convergence.

Approximately 2.38 million trainable parameters make up the model. Deep convolutional layers are used in the architecture to capture intricate patterns in the input data, and dropout layers and a gradual increase in filter depth aid in regularizing the model and improving its generalization abilities. This design, which balances computational efficiency and complexity, is common for image classification tasks.

The model summary is given below in Figure 2.

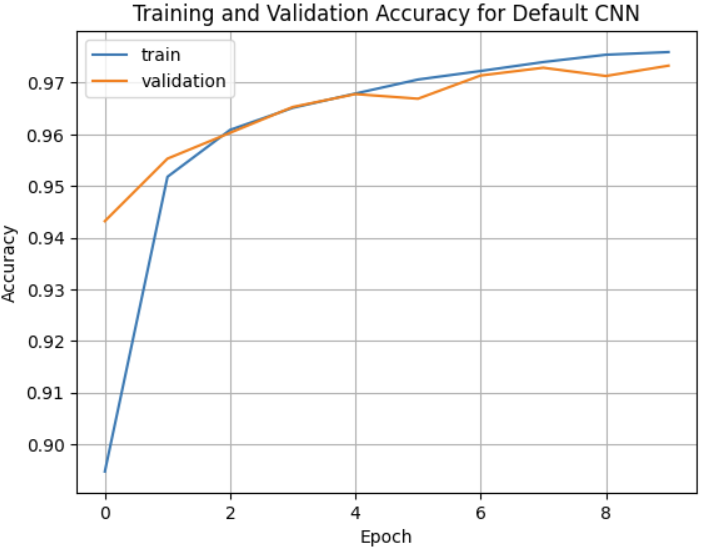
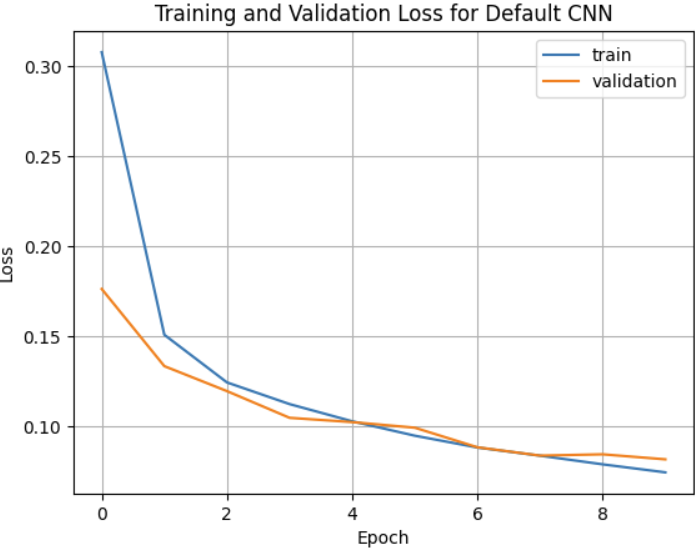
****

**Figure 2.** Improved CNN model architecture

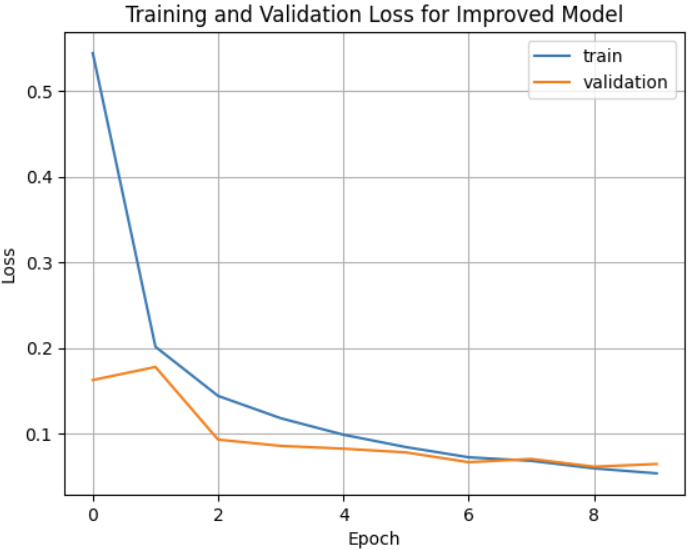
The model was compiled and trained on the training data with a batch size of 128 for 10 epochs. The total training time was 91.28 seconds or about 1.5 minutes, with every epoch running for 9.2 seconds.

##### Q3. Result Analysis

For the default CNN model, Figure 3 shows the training and validation loss versus the training epochs, and Figure 4 shows the training and validation accuracy versus the training epochs. Based on Figure 3, the model is learning as the training and validation losses are decreasing, and it converges in around 10 epochs. The model does not show any signs of overfitting as the validation loss remains close to the training loss and does not start increasing. As mentioned previously, the total training time was 50.55 seconds, and the testing time was 2 seconds. When evaluated on the test set, the model achieves a test loss of 0.082 and a test accuracy of 97.33%.

**Figure 3.** Training and Validation loss for default CNN  **Figure 4.** Training and Validation accuracy for default CNN

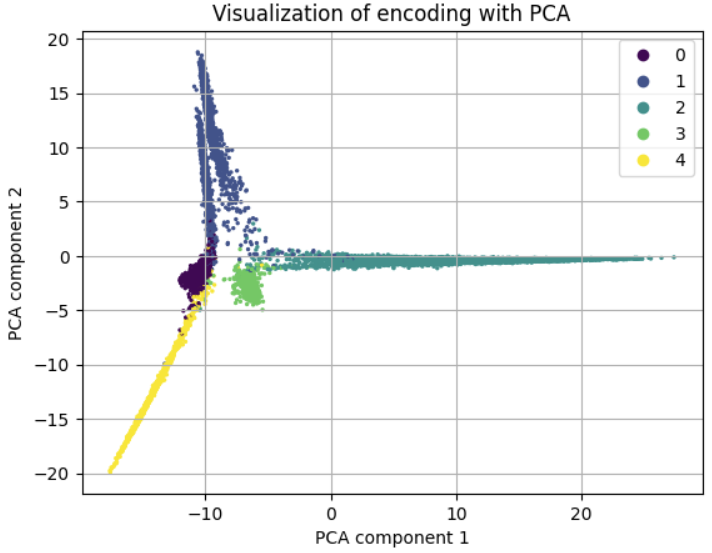
For the improved model, the loss and accuracy plots are shown in Figure 5 and Figure 6, respectively. We see that the training time is higher than that of the default model, as this model has a deeper network than the default model. The testing time was also 2 seconds. The convergence rate of this model is similar to the default model, around 10 epochs. Deeper models can tend to overfit, but as this model employs regularization, through dropout, based on Figure 5 there is no overfitting. For hyperparameter tuning, learning rate was tuned and a learning rate of 0.0001 was chosen as larger rates such as 0.1 and 0.01 led to poor training. When evaluated on the test set, the model achieves a test loss of 0.064 and a test accuracy of 98.21%, which is an improvement to the default model.

**Figure 5.** Training and Validation loss for improved CNN **Figure 6.** Training and Validation accuracy for improved CNN 

##### 

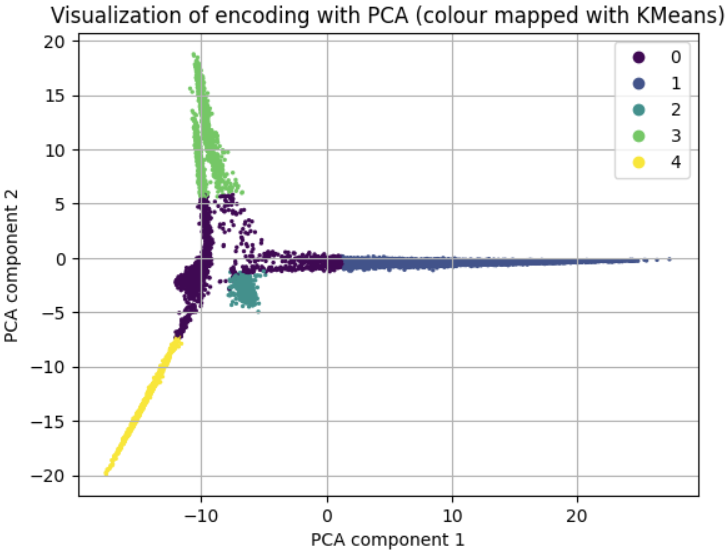
##### Q4. Using own encoding

From our trained multi-layer network, the output of the last Dense layer before the softmax layer has been chosen as our encoding. This dense layer has 64 nodes, meaning the encoding or features is of 64 dimensions. PCA has been performed on these features, and the first two components color-mapped with the true labels have been plotted and shown in Figure 7. From these first two principal components we see the distinction between classes has been captured.

****

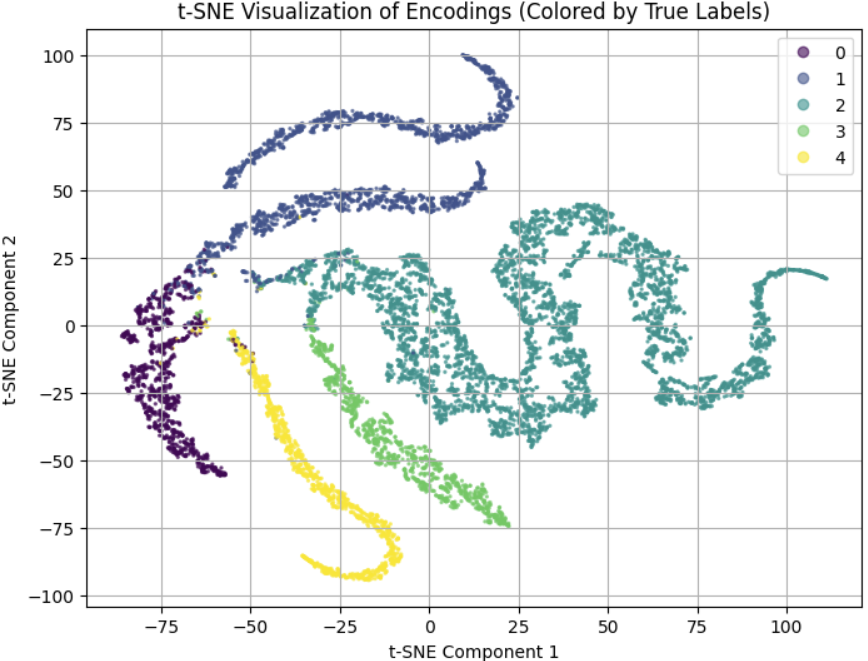
**Figure 7.** Encoding visualized with the first two components from PCA

Clustering was performed using the K-means algorithm on the extracted features, with 5 clusters. Figure 8 shows the above PCA plot but with the colours mapped according to the K-means cluster each data point is assigned. Like PCA, K-means is an unsupervised algorithm. While the PCA visualization in Figure 7 uses the true labels to colour map, the visualization in Figure 8 uses the cluster assignments, so as expected, the colours do not need to match between the two plots necessarily. However, the patterns captured by both are similar, in terms of which points are grouped into the same class.



**Figure 8.** PCA on the encoding colour mapped according to K-means cluster assignment

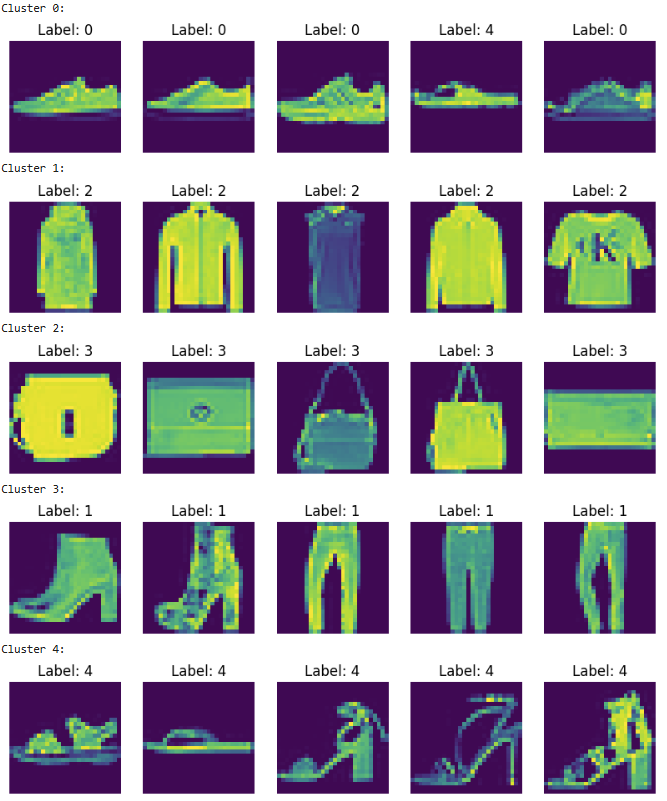
To use the encoding in a different task, t-SNE has been applied to the extracted features. Below is the 2D representation of the t-SNE components shown in Figure 9. The different classes can be visually discriminated between them.



**Figure 9.** t-SNE visualization of the encodings

To guess what the labels are for the given dataset, based on all these results of clustering, we have plotted a random selection of the original images for each cluster and their label values, as shown in Figure 10. The randomly selected images from each cluster reveal the kinds of items that have been clustered together. Generally, sneakers have been clustered together, different kinds of shirts are in a cluster, bags are clustered together, ankle boots and trousers have been clustered together, and sandals have been clustered together. This leads us to deduce that the labels for the given dataset are:

* Class 0: Sneakers
* Class 1: Trousers and ankle boots
* Class 2: T-shirt/top, Pullover, Dress, Coat, and Shirt
* Class 3: Bags
* Class 4: Sandals



**Figure 10.** Sample images and their labels of the given dataset for each cluster.

**References**

[1] S. Dutta, “Designing Your Own Convolutional Neural Network (CNN) Model: A Step-by-Step Guide for Beginners,” Medium, https://medium.com/@sanjay\_dutta/designing-your-own-convolutional-neural-network-cnn-model-a-step-by-step-guide-for-beginners-4e8b57836c81 (accessed Apr. 4, 2025).

[2] F. J. Dijkinga, “The ADAM optimizer,” Medium, https://medium.com/@fernando.dijkinga/the-adam-optimizer-4645719c6c0a#:~:text=This%20optimizer%20maintains%20first%2Dorder,second%20moments%20of%20the%20gradients (accessed Apr. 4, 2025).