Assignment 2

Depp Learning  
Matheus Vinicius Ferreira Figueiredo Teixeira  
301236904

# Build, Train, and Validate a baseline CNN Model

## Display (print) a summary of the model using summary(). **Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons) and number of weights in each layer.**

A diagram of a diagram

Description automatically generated

# Test and analyze the model

## Display (plot) the Training Vs Validation Accuracy of the baseline CNN Model as a line graph using matplotlib. Provide proper axis labels, title and a legend. Use different line color's for training and validation accuracy. **Compare and analyze the training and validation accuracy in your report.**

A graph of a graph

Description automatically generated As shown in the plot, the model learned to classify the training data, achieving approximately 72.09% accuracy on the training dataset and 70.33% on the validation dataset. The similarity between the validation and training curves suggests that the model is generalizing well. Although the accuracy is not particularly high, it is still acceptable, and importantly, the model does not appear to be overfitting.

## Evaluate the cnn model with the test dataset using Tensorflow's evaluate() and display (Print) the test accuracy. **Compare and discuss the test accuracy to the validation accuracy in your report**

A screenshot of a computer program

Description automatically generatedThe image on the left shows that the test accuracy (75.33%) is very close to both the training accuracy (72.09%) and the validation accuracy (70.33%). Interestingly, the test accuracy is even higher than the training and validation accuracies, further confirming the model's acceptable performance and generalization ability.

## A graph with numbers and a number in the center Description automatically generated with medium confidence**Analyze and discuss the confusion matrix in your report**

The confusion matrix confirms that the model is acceptably accurate, as seen by the large values along the main diagonal. However, it appears that category 6 (Shirt) is the most challenging to classify correctly, with the lowest number of true labels predicted accurately. Specifically, the model frequently misclassifies this category:

* **10 times** as category 0 (T-shirt/top)
* **10 times** as category 2 (Pullover)
* **13 times** as category 3 (Dress)
* **27 times** as category 4 (Coat)
* **2 times** as category 8 (Bag)

This misclassification pattern also occurs in reverse:

* Category 2 (Pullover) is misclassified as 6 (Shirt) **2 times**.
* Category 3 (Dress) is misclassified as 6 (Shirt) **4 times**.

Out of 68 examples for category 6 (Shirt), only **6 are correctly predicted**, resulting in an accuracy of approximately **8.82%**. This is significantly lower than the overall accuracy of the model on the test data, which is around **72%**.

On the other hand, category 9 (Ankle boot) and category 1 (Trousers) are easily recognized, with accuracies above **93%**. This is likely because, in the MNIST\_Fashion dataset, there are no items that closely resemble trousers (like shorts), and the other types of shoes present in the dataset (Sandal and Sneaker) are very different from Ankle boot, making them easier for the model to classify.

# Build, Train, and Validate a baseline CNN Model

## Display (print) a summary of the model using summary(). **Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons) and number of weights in each layer.**

A diagram of a diagram

Description automatically generated

# Build and perform transfer learning on a CNN with the Autoencoder

## Display (print) a summary of the model using summary(). **Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons) and number of weights in each layer.**

A diagram of a diagram

Description automatically generated

# Test and analyze the model

## Display (plot) the Training Vs Validation Accuracy of the pretrained CNN Model as a line graph using matplotlib. Provide proper axis labels, title and a legend. Use different line color's for training and validation accuracy. **Compare and analyze the training and validation accuracy in your report.**

A graph of a graph showing the difference between training and validation accuracy

Description automatically generatedAs shown in the plot, the model learned to classify the training data, achieving approximately 75.77% accuracy on the training dataset and 72.17% on the validation dataset. The similarity between the validation and training curves suggests that the model is generalizing well. Although the accuracy is not particularly high, it is still acceptable, and importantly, the model does not appear to be overfitting.

## A screenshot of a computer program Description automatically generatedEvaluate the cnn model with the test dataset using Tensorflow's evaluate() and display (Print) the test accuracy. Compare and discuss the test accuracy to the validation accuracy in your report

The image on the left shows that the test accuracy (74.50%) is very close to both the training accuracy (75.77%) and the validation accuracy (72.17%). Interestingly, the test accuracy is even higher than the validation accuracy, further confirming the model's acceptable performance and generalization ability.

## A graph of numbers and a diagram Description automatically generated with medium confidence**Analyze and discuss the confusion matrix in your report**

The confusion matrix confirms that the model is acceptably accurate, as seen by the large values along the main diagonal. However, it appears that category 6 (Shirt) is the most challenging to classify correctly, with the lowest number of true labels predicted accurately. Specifically, the model frequently misclassifies this category:

* **10 times** as category 0 (T-shirt/top)
* **1 times** as category 2 (Pullover)
* **4 times** as category 3 (Dress)
* **25 times** as category 4 (Coat)
* **3 times** as category 5 (Sandal)
* **3 times** as category 8 (Bag)

This misclassification pattern also occurs in reverse:

* Category 0 (T-shirt/top) is misclassified as 6 (Shirt) **3 times**.
* Category 2 (Pullover) is misclassified as 6 (Shirt) **10 times**.
* Category 3 (Dress) is misclassified as 6 (Shirt) **8 times**.
* Category 4 (Coat) is misclassified as 6 (Shirt) **6 times**.
* Category 8 (Bag) is misclassified as 6 (Shirt) **2 times**.

Out of 68 examples for category 6 (Shirt), only **22 are correctly predicted**, resulting in an accuracy of approximately **32.35%**. This is significantly lower than the overall accuracy of the model on the test data, which is around **74%**.

Although class 2 (Pullover) has a lower absolute value, its relative accuracy is higher than that of category 6 (Shirt). This is because there are 68 examples for category 6 but only 55 examples for category 2; thus, the accuracy for category 2 (20/55 = **36.36%**) is slightly better than for category 6 (22/68 = **32.35%**). It is also worth mentioning that the pre-training had a backlash effect on category 2, as the original (baseline) classifier correctly classified **34** pullover images, whereas the new classifier with pre-trained layers only classified **20** pullover images correctly.

On the other hand, category 9 (Ankle boot) and category 1 (Trousers) are easily recognized, with accuracies above **95%**. This is likely because, in the MNIST\_Fashion dataset, there are no items that closely resemble trousers (like shorts), and the other types of shoes present in the dataset (Sandal and Sneaker) are very different from Ankle boot, making them easier for the model to classify.

# Compare the performance of the baseline CNN model to the pretrained model in your report

## Compare and analyze the test accuracy in your report.

A graph of a line

Description automatically generated As shown in the plot, overall, the pre-trained model performed better than the baseline model, with slightly higher accuracies. However, as observed earlier in the confusion matrices, the correct classification for a few specific classes decreased. For instance, category 2 saw its true positives drop from 34 to 20. On the other hand, some other categories, such as category 6, experienced a considerable increase in true positives, compensating for the setback in category 2 and contributing to an overall increase in model accuracy. The number of true positives for category 6 rose significantly, from 6 to 22!