Project Report: Fashion MNIST

Dataset Description

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

Labels

Each training and test example is assigned to one of the following labels:

- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

Part I: Data Preparation

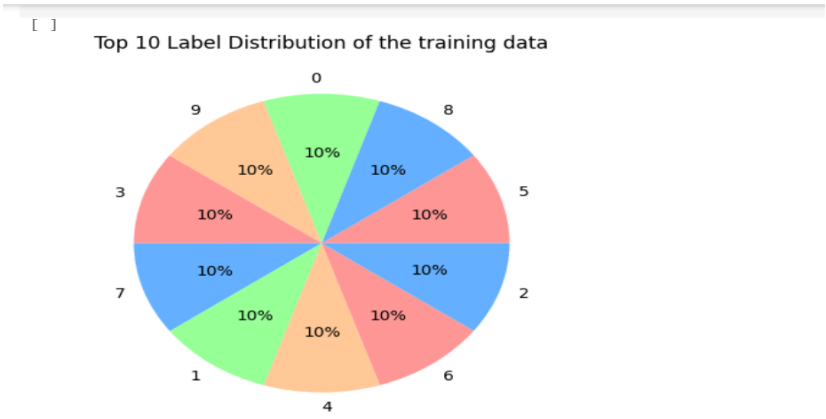
1. importing the required libraries.

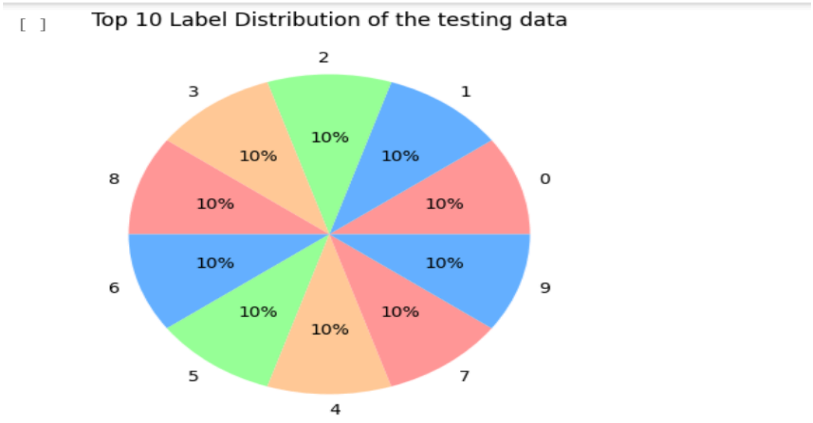
```
[ ]  #import important libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import tensorflow as tf
     from sklearn.model_selection import KFold
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
     from sklearn.decomposition import PCA
     from keras.applications import VGG19
     from tensorflow.keras.utils import to_categorical
     from tensorflow.keras.preprocessing.image import img_to_array
     from tensorflow.image import resize
     from tensorflow.keras.layers import Input,Concatenate,GlobalMaxPooling2D,Dropout
     from tensorflow.keras.layers import BatchNormalization
     from tensorflow.keras.models import Model
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.losses import SparseCategoricalCrossentropy
     from tensorflow.keras.applications import ResNet50
     from tensorflow.keras.layers.experimental import preprocessing
     from tensorflow.keras.layers.experimental.preprocessing import Resizing, Rescaling
     from keras import models
     from keras import layers
```

2. Loading the dataset

3. data cleaning (Ensure data cleanliness by checking for null values and duplicates. By addressing these issues, the dataset can be rendered free of any nulls or duplicate entries, enhancing its reliability and suitability for analysis**)**

4. visualization

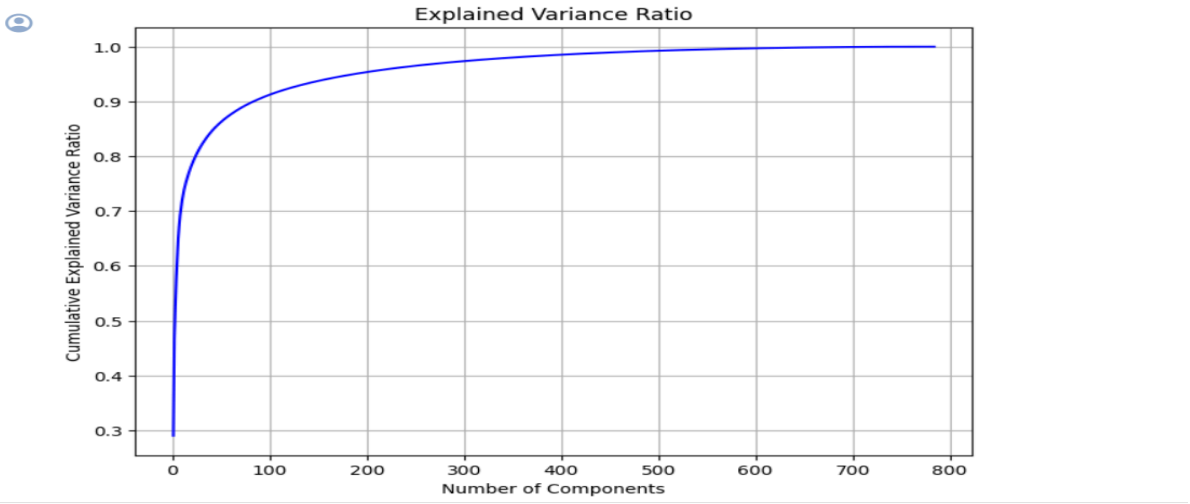




as we can see in the above graphs both the training and the testing data are balanced so we can use the accuracy as a metric for our classification problem.

5. Preprocessing

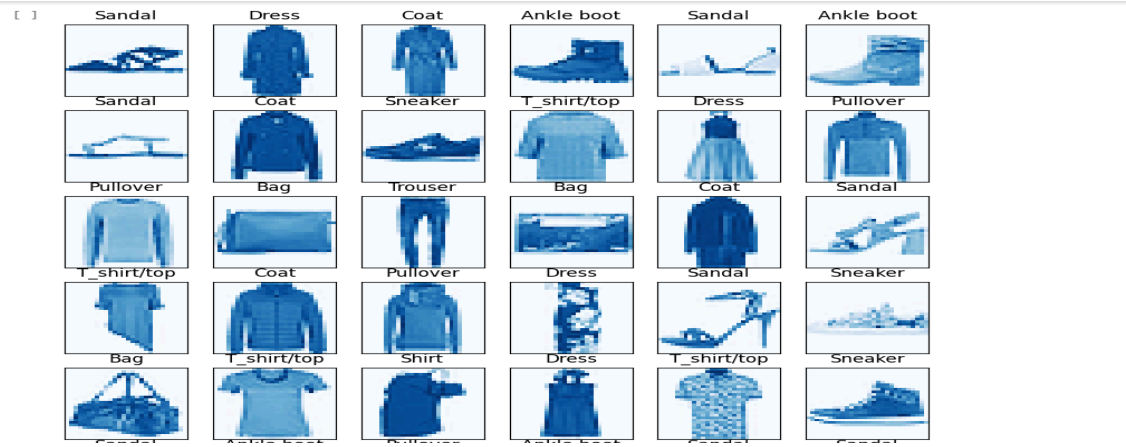
we need to normalize the values of the pixels by dividing by 255 to avoid the saturation problems which can cause the vanishing gradient problem.



comments on the graph:

- X-axis: The x-axis represents the number of principal components used. as we can see, it goes up to 500 because we have a high-dimensional dataset (image pixels).
- Y-axis: The y-axis shows the cumulative explained variance ratio. The values near 1 on the y-axis indicate that the first few principal components capture most of the variance in the data. Saturation: The graph saturating around 0.9-1.0 on the y-axis is expected. It means that a relatively small number of principal components (likely less than 500 in our case) can explain a significant portion of the data's variance. This is the dimensionality reduction benefit of PCA.

6. Drawing some images

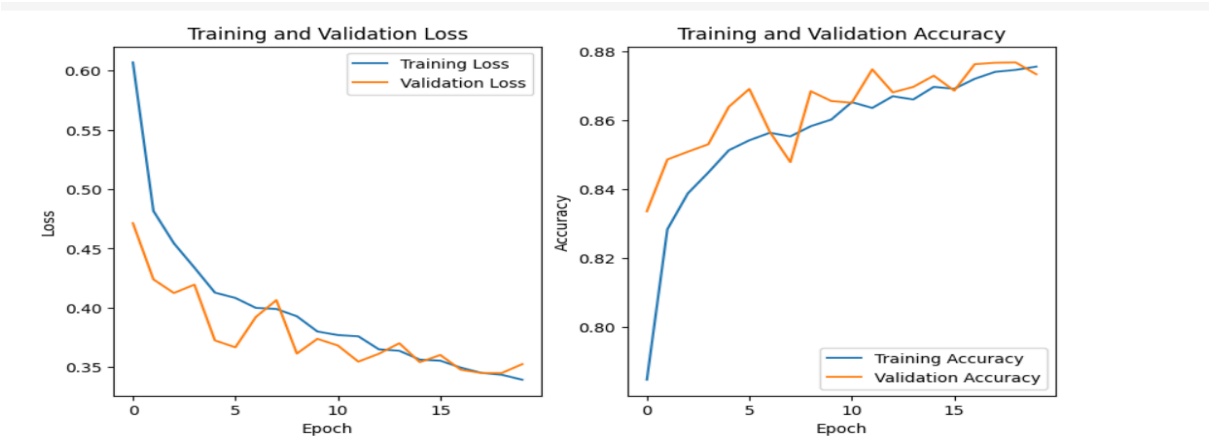


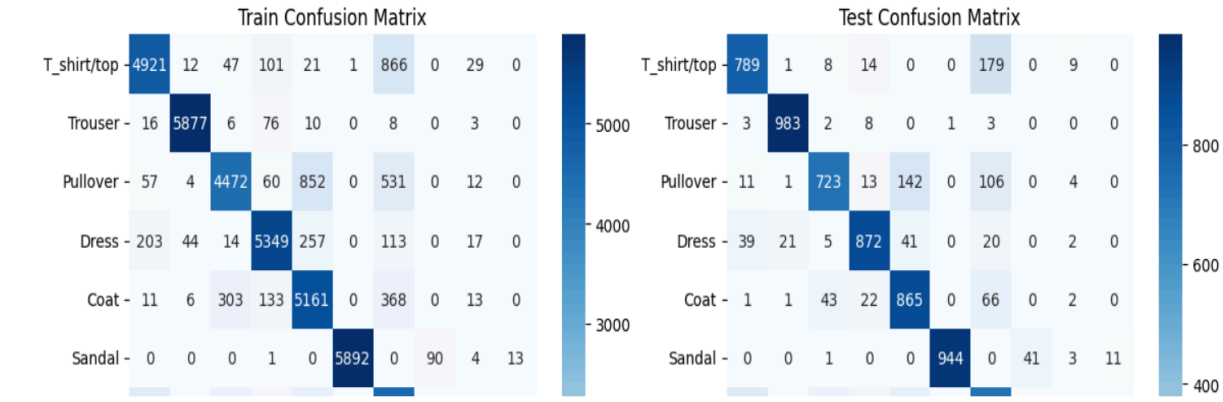
7. Label Encoding

label encoding is unnecessary because the labels are already encoded. Instead, I'll use one-hot encoding to ensure that the output of the neural network matches the required format.

Part II: Training a CNN neural network

Trying the Fully connected neural network:





comment on the Fully connected neural network:

After training the model on the Fashion mnist dataset, the evaluation steps give about 89 % accuracy on the training data and about 87 % accuracy on the test data.
from the confusion matrix, we can see that the model couldn't catch the pattern that can differentiate between the T_shirt/top and the Shirt as the number of misclassifications for those two classes is really high compared to the other combinations.

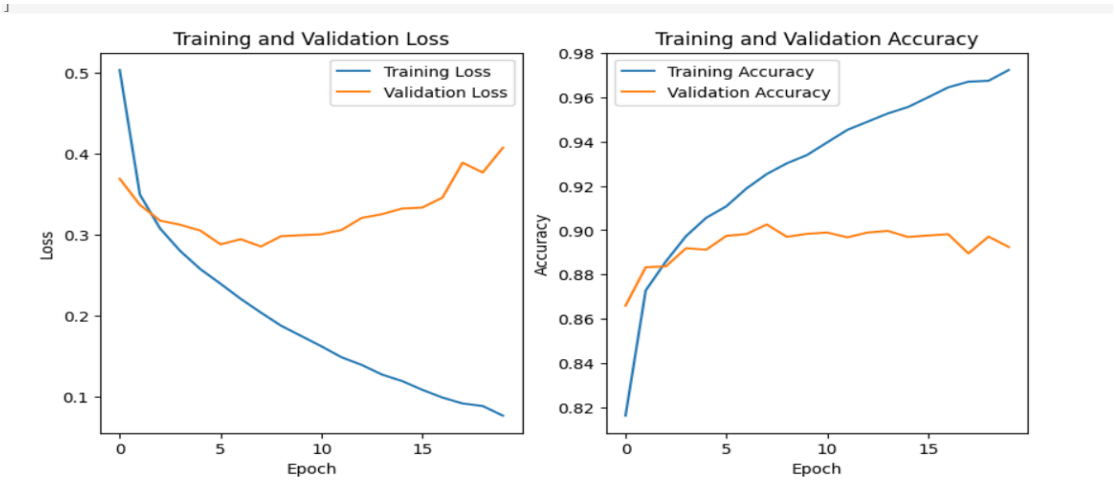
in the next trials I will try to implement LeNet 5 model and compare its results with the results that I get from the FCNN.

trail 1:LeNet 5 Model

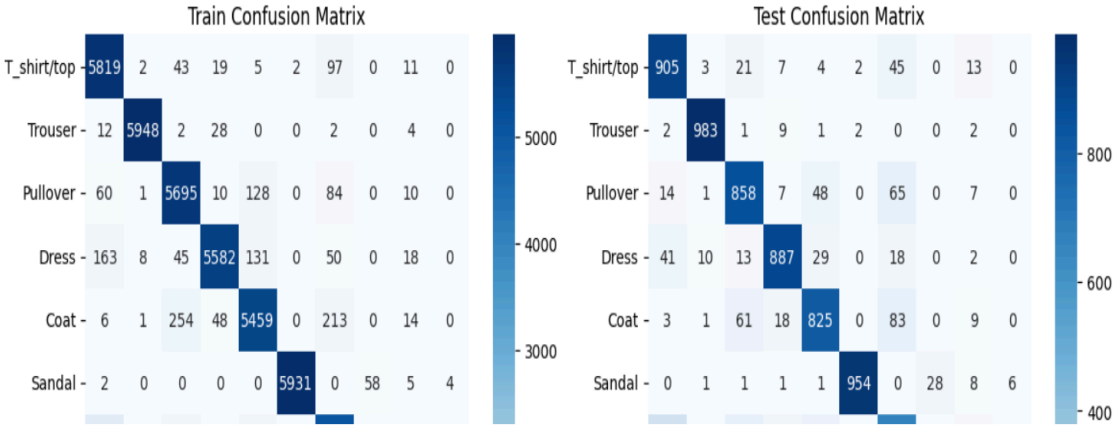
Implementing the LeNet-5 Model without cross-validation. I will do so by following the representation given by the doctor from this source: https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/#:~:text=End%20Notes-,What%20is%20Lenet5%3F,handwritten%20and%20machine%2Dprinted%20characters.

as we can see the Lenet-5 Architecture consists of three convolution layers followed by two Dense layers and the Activation used is Tanh. Also, the input shape of the image is 32*32 which means we need also a preprocessing step.

These graphs represent the validation loss and validation accuracy computed during the training process using the training dataset




```
Evaluate LeNet  Model
313/313 [==============================] - 1s 2ms/step - loss: 0.3772 -
Accuracy: 0.8956


[0.3772267699241638, 0.8956000208854675]
```



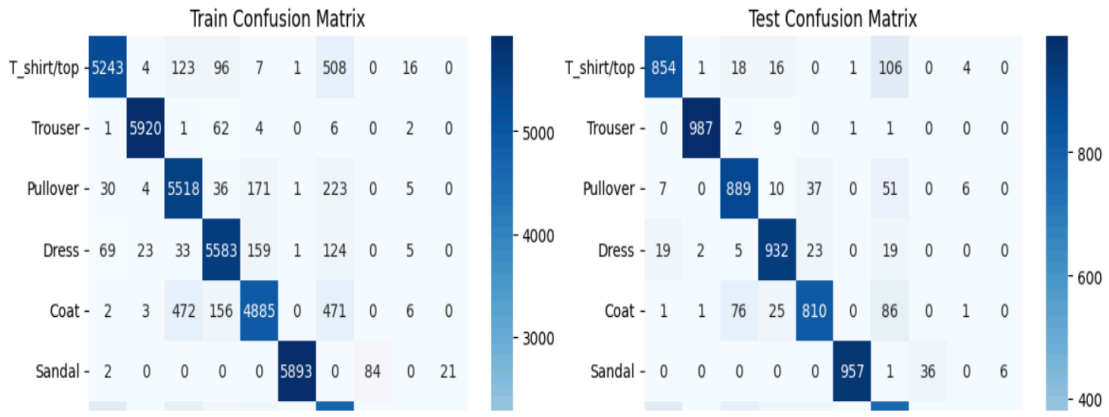
as we can see here the model is making many mistakes in the Shirt class and most of the prediction errors were by predicting the Shirt as T_shirt/top which makes sense as the images are very small and in grayscale.

trial 2

we need to train our model with different hyperparameters to enhance our model's performance. in this trial, i will use batch normalization and dropout to test if they enhance the model performance. also, i will implement a function that accepts the dropout ratio and the number of kernels and number of nodes in the network. in this trail i will use the following : num_kernels_1=6, num_kernels_2=16, num_kernels_3=120,num_nodes1=84,drop_ratio=0.25,batch_norm=True. the difference from the original lenet is that i have used relu as an activation function instead of the tanh and also i used both the batch norm and the dropout techniques to avoid the overfitting that we have seen in the previous trial.

These graphs represent the validation loss and validation accuracy computed during the training process using the training dataset

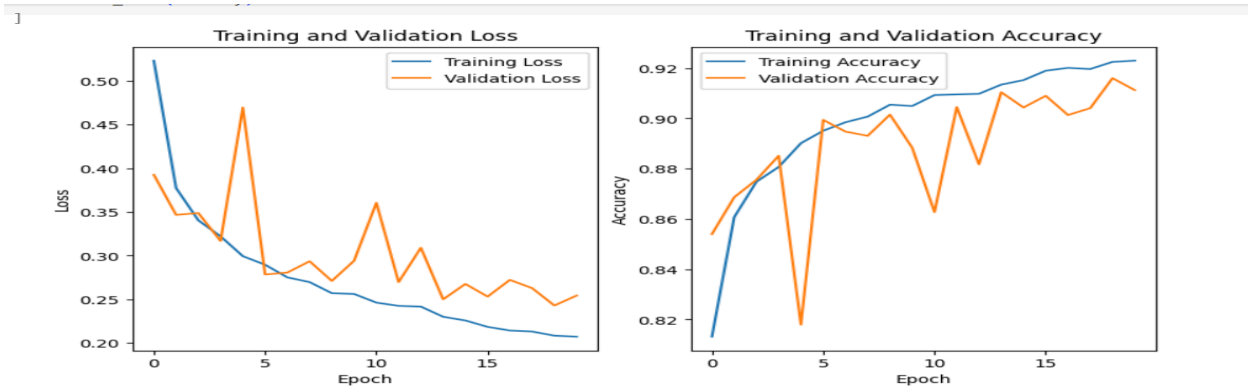
```
Evaluate Enhanced LeNet  Model
313/313 [==============================] - 1s 4ms/step - loss: 0.2457 -
Accuracy: 0.9100


[0.24567143619060516, 0.9100000262260437]
```

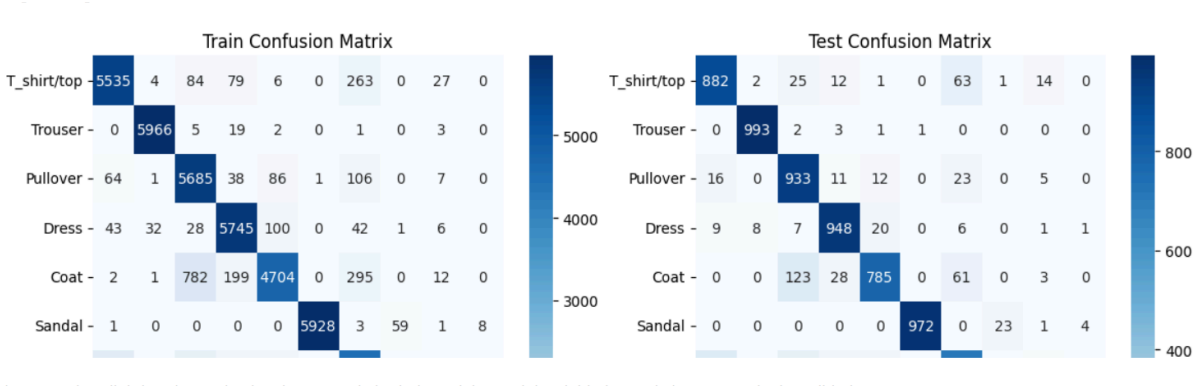


as we can see the deviation between the accuracy curves of both the training and validation set in the plots of the first trail is not found here in the plot of the second trail as both the training and validation accuracies were near to each other and about 90%.this is due to the effect of using both batch normalization and dropout. we can also see that the problem of the shirt being predicted as T/shirt still persists so in the next trial I will try to increase the model capacity by increasing the number of kernels in the convolution layer if the problem still persists I will perform another trial in which I will increase also the number of nodes in the Fully connected layer.

trail 3:

in this trial i will double the number of kernels in each convolution layer num_kernels_1=12, num_kernels_2=32, num_kernels_3=240,num_nodes1=84,drop_ratio=0.25,batch_norm=True



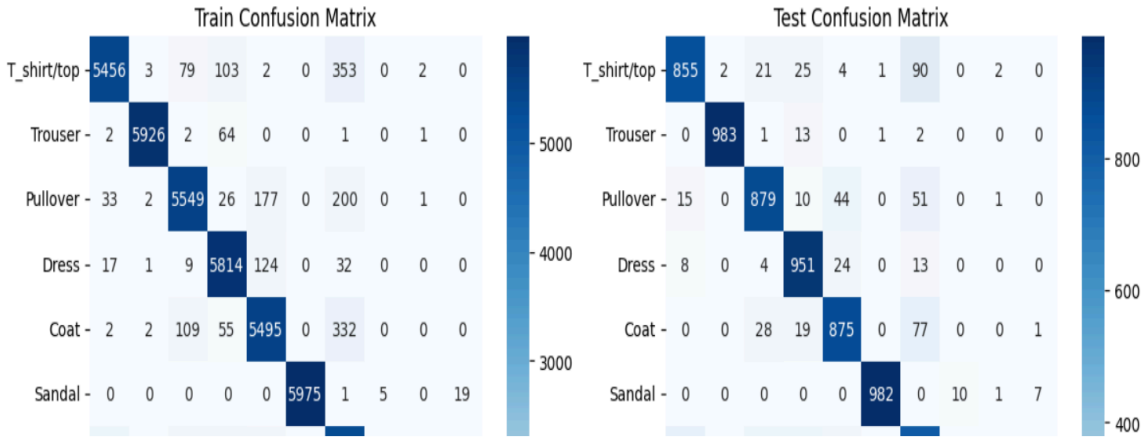


the accuracies slightly enhanced to be about 92 % in both the training and the validation and about 91.5% in the validation.

trail 4

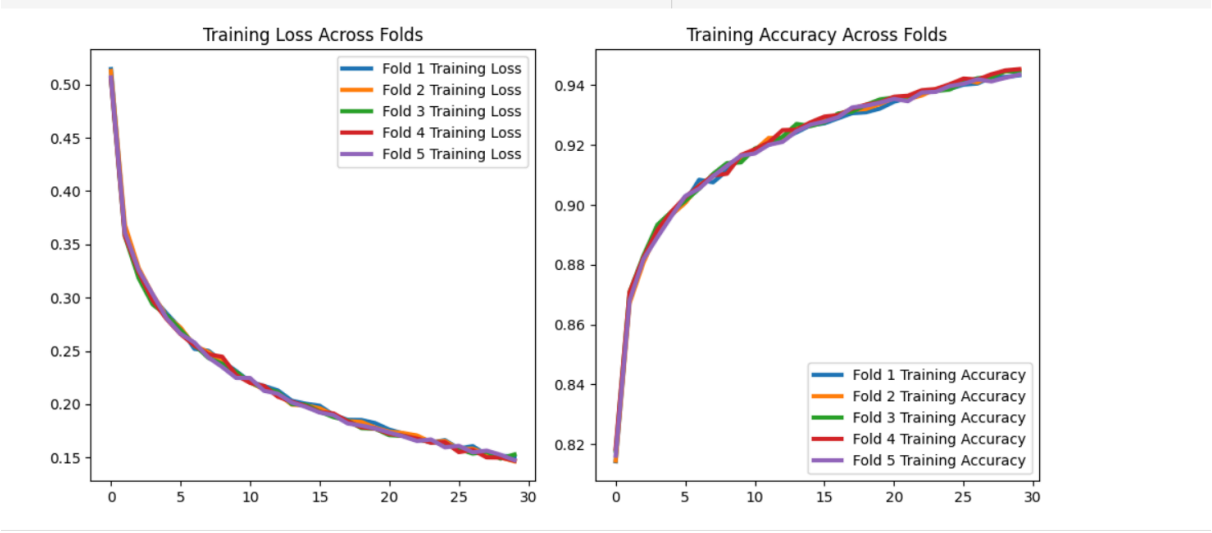
I will try to increase also the number of nodes in the fully connected layer and see the effect of that. num_kernels_1=32, num_kernels_2=64, num_kernels_3=128,num_nodes1=256,drop_ratio=0.25,batch_norm=True also increase the number of epochs to 50


```
Evaluate Trail 4  Model
313/313 [==============================] - 1s 3ms/step - loss: 0.2246 -
Accuracy: 0.9255


[0.22455771267414093, 0.9254999756813049]
```



as we can see the accuracy on the training data and validation data was about 93% and also the accuracy on the test increased to 92.5% also this can be seen from the confusion matrix of the test data as the number of shirt predicted as Tshirt/top decreased to 76 from 119 in the previous trail.

Evaluate the model through cross-validation:



as we can see from the average and the standard deviation of the cross-validation results we can accept this model. Also both the accuracy and loss curves for the fold are near to each other.

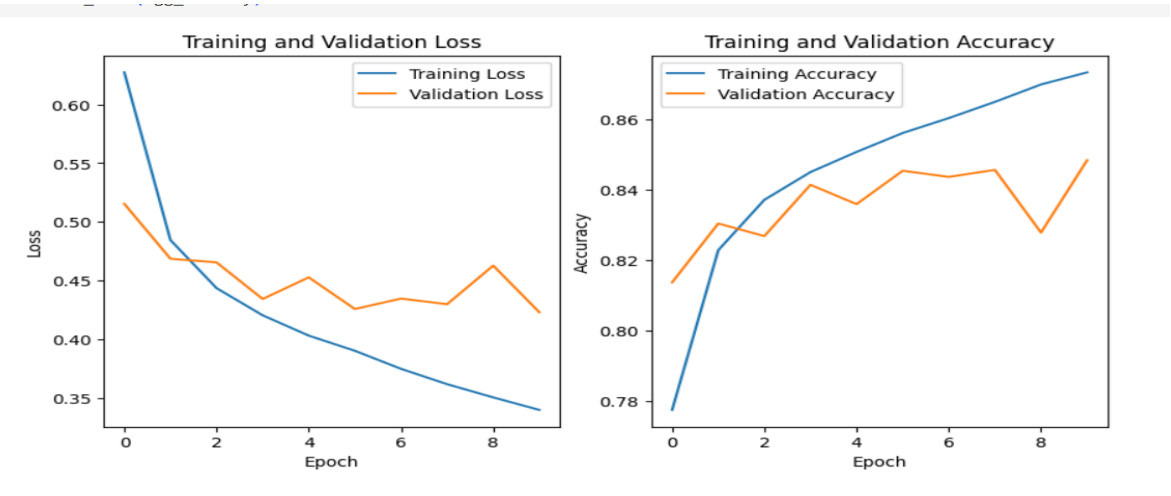
Comment on why you think LeNet-5 further improves the accuracy if any at all.

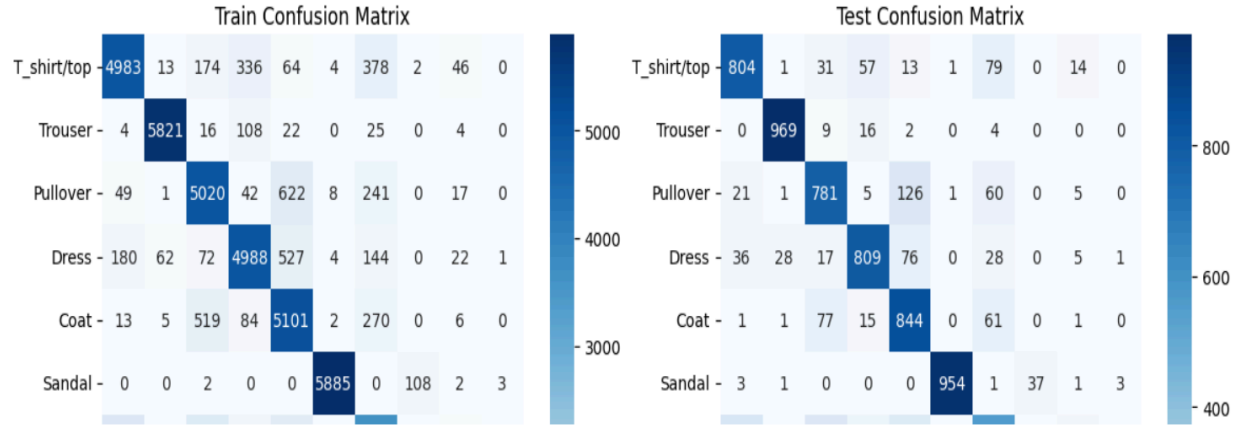
- The leNet Architecture could achieve better performance on the Fashion-mnist dataset.

- as the accuracy on the test data was about 93% for the best trial, while in the FCNN it was about 86% on the test data.
- this is because of the:
  - Local Connectivity: CNNs use convolutional layers with learnable filters to act on specific areas of the picture. This enables them to record local details such as edges, textures, and basic forms. while FCNNs treat pictures as flattened vectors, which removes the spatial correlations between pixels. This may make it challenging for them to collect the critical elements required for proper picture categorization.
  - Parameter Efficiency: To link each neuron in one layer to every neuron in the next layer, FCNNs require a high number of parameters, CNNs, utilizing local filters and weight sharing, drastically minimize the amount of parameters required.
  - Shared Weights: Convolutional filters are applied to the whole image, catching the same characteristics regardless of location. This weight sharing decreases the number of parameters and allows the network to learn features that are insensitive to tiny translations.
- in this project the Fcnn neural network could achieve a good performance as the data is simple after all the image size is 28X28 and in grayscale but when there are classes close to each other as the T_shirt/top and Shirt the model couldn't catch the pattern to differentiate between them.

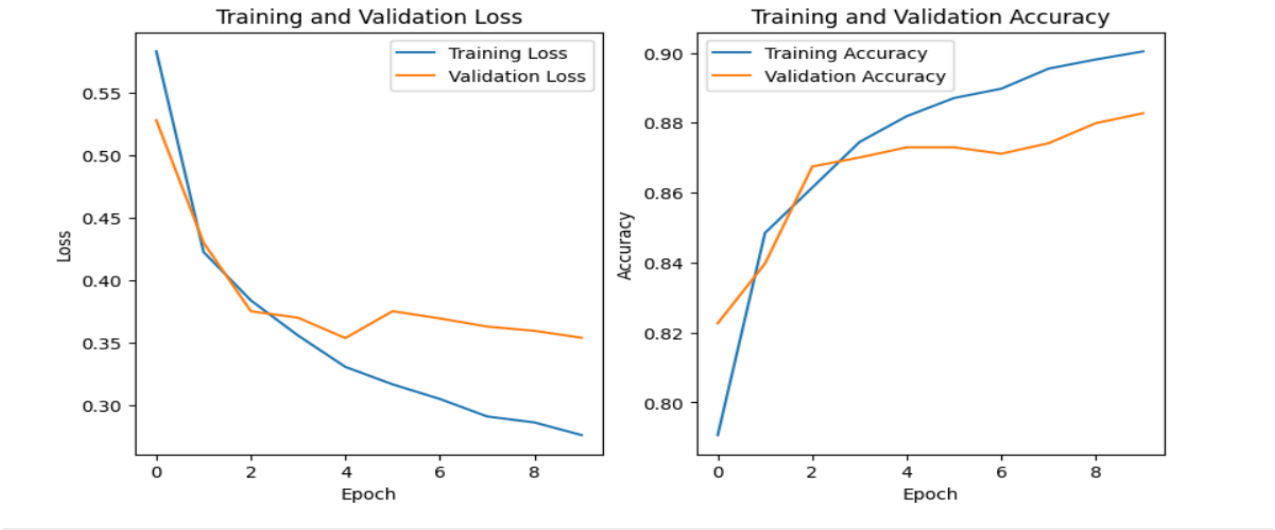
Transfer learning :

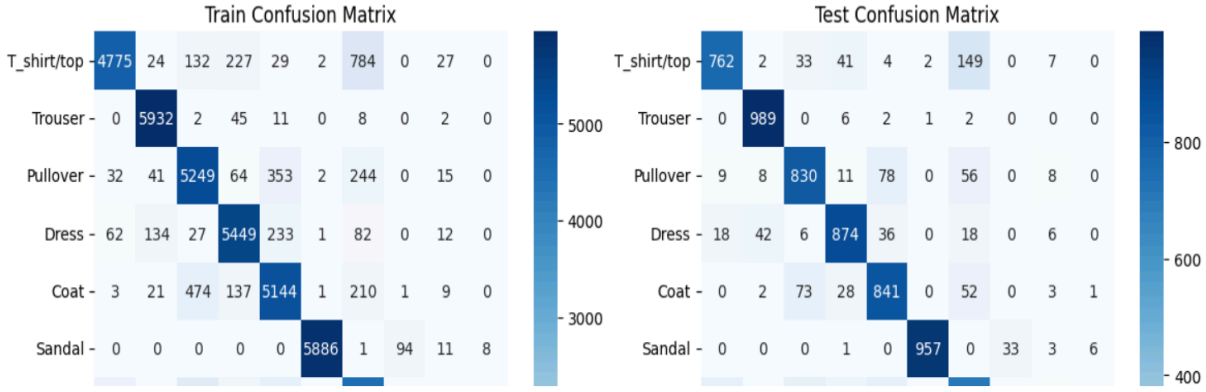
in this step I will try different pre-trained models I will start with VGG 19





the results is worse than that of enhanced lenet 5 so i will unfreeze some layers and test it again.



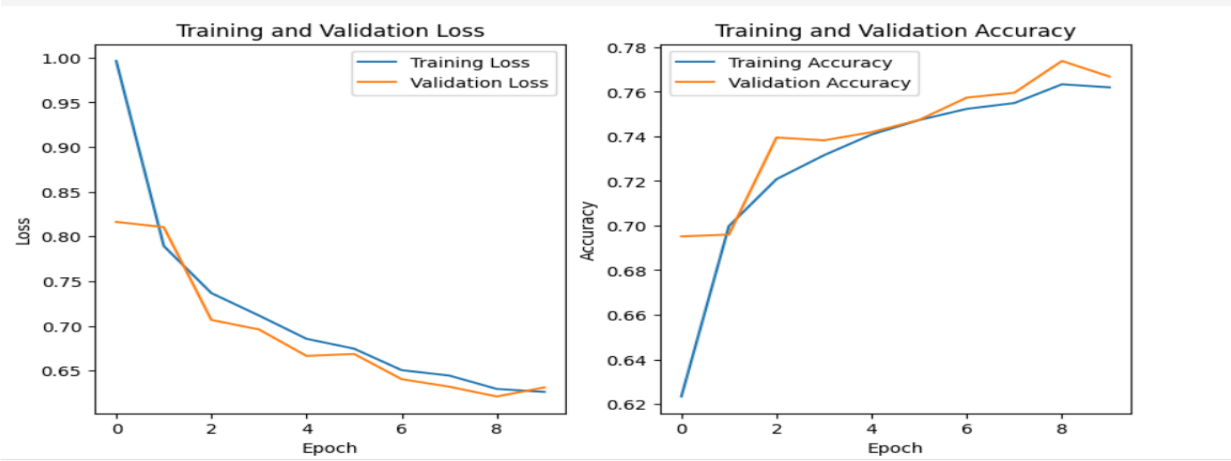


Comment:

after unfreezing some layers in the convolution part of the vgg19 network this achieved better results than freezing the layers from learning as the accuracy increased from 85 on the test data to 89 on the test data, but the enhanced lenet-5 still the model with the highest performance till now, of course, the vgg-19 model with unfreezing some layers may achieve higher accuracy if we train it for longer epochs but colab keeps crashing due to the limitations on the resources.
in the next trial, i will try to use resnet 50

tuning the ResNet50:

the input shape accepted by the resnet 50 is 32X32



the resnet model gives poor accuracy on both the training and testing data so I will unfreeze the layers and evaluate it again.

comment on Resnet:

After unfreezing the layers the accuracy increases to 89 % from 77% in only 10 epochs, increasing the number of epochs may achieve better results.