Project_2

Fashion_MNIST

Name: - Amany Azzam

ID: - 20399133

Supervised by

Prof. Hazem Abbas

Data Preparation

1. **Describe the data.**

   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.

2. **Clean the data.**

   Data do not contain missing values.

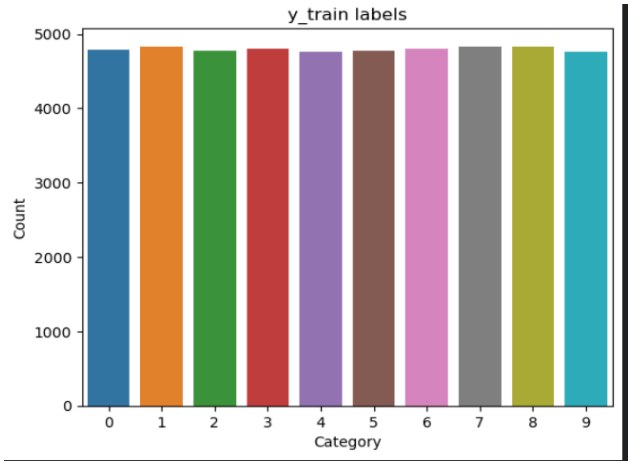
3. **Check the data for duplicating data.**

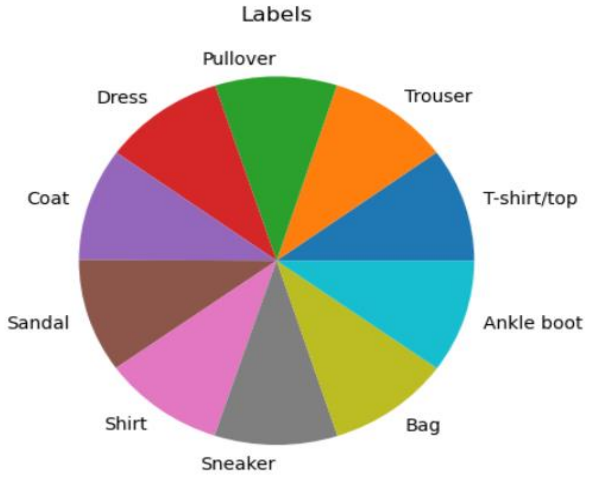
```
print("Duplicated data:", train_df.duplicated().sum())

print("Duplicated data:", test_df.duplicated().sum())
```

```
Duplicated data: 43
Duplicated data: 1
```

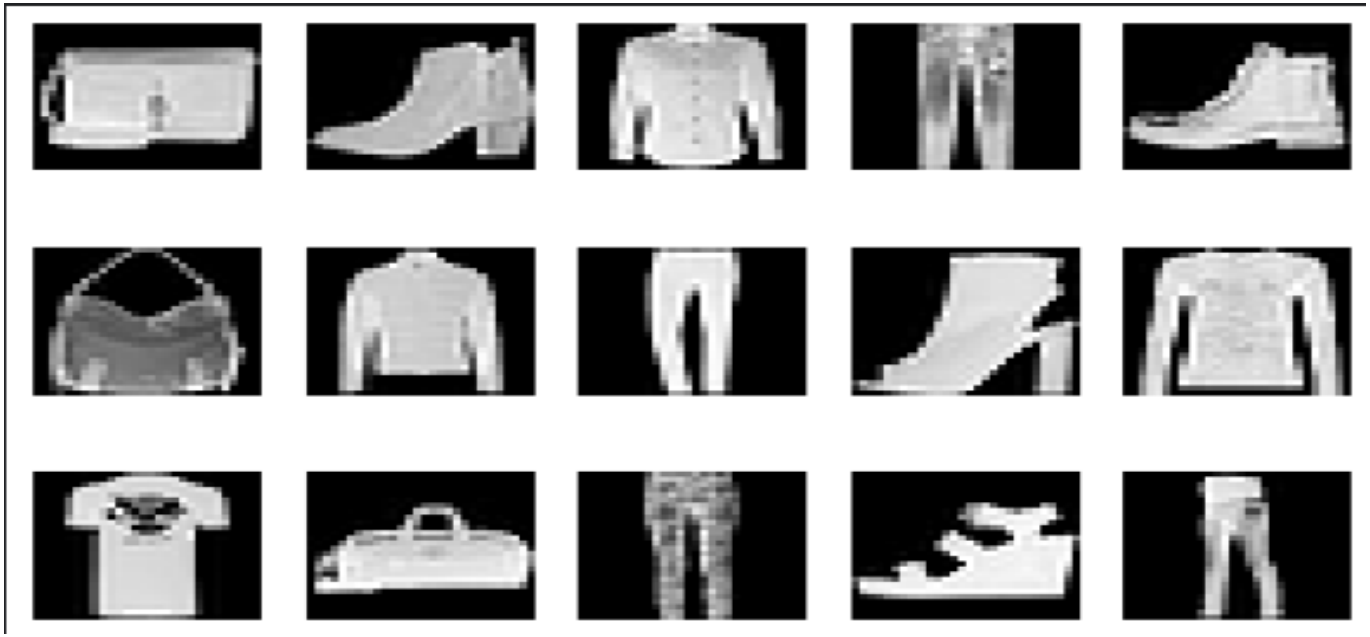
```
# Remove duplicates
train_df = train_df.drop_duplicates()

test_df = test_df.drop_duplicates()
```

4. Visualize the data using proper visualization methods.





5. Draw some of the images.



6. correlation analysis

1.000000	0.297899	0.067551	0.046607	0.026630	0.026172	0.012096	0.012225	0.009644	0.000056	-0.001536	-0.001703	-0.003406	-0.004325	-0.004088	-0.004536	-0.003627
0.297899	1.000000	0.575033	0.138709	0.054353	0.033184	0.022766	0.017138	0.016821	0.010920	0.001104	-0.004765	-0.009051	-0.009799	-0.009812	-0.009117	-0.005704
0.067551	0.575033	1.000000	0.387468	0.118136	0.087300	0.060937	0.035942	0.029674	0.021493	0.013902	0.008735	-0.000249	-0.003643	-0.003400	-0.002603	0.003539
0.046607	0.138709	0.387468	1.000000	0.573172	0.325683	0.242987	0.141033	0.085302	0.051147	0.024978	0.015852	0.008657	0.005696	0.006290	0.005201	0.010298
0.026630	0.054353	0.118136	0.573172	1.000000	0.692892	0.423635	0.230693	0.136391	0.075677	0.035616	0.020559	0.015814	0.016219	0.017737	0.011675	0.013669
0.026172	0.033184	0.087300	0.325683	0.692892	1.000000	0.655459	0.324748	0.181620	0.095664	0.045679	0.028909	0.020924	0.020725	0.021248	0.016798	0.021459
0.012096	0.022766	0.060937	0.242987	0.423635	0.655459	1.000000	0.636615	0.324932	0.161713	0.060007	0.029046	0.018908	0.017065	0.017968	0.014382	0.020052
0.012225	0.017138	0.035942	0.141033	0.230693	0.324748	0.636615	1.000000	0.665618	0.317476	0.108920	0.031307	0.007191	0.003906	0.004136	0.001732	0.016081
0.009644	0.016821	0.029674	0.085302	0.136391	0.181620	0.324932	0.665618	1.000000	0.627682	0.226297	0.077899	0.015395	0.006265	0.003341	0.008142	0.049342
0.000056	0.010920	0.021493	0.051147	0.075677	0.095664	0.161713	0.317476	0.627682	1.000000	0.588959	0.271935	0.131868	0.096078	0.091350	0.113526	0.216056
-0.001536	0.001104	0.013902	0.024978	0.035616	0.045679	0.060007	0.108920	0.226297	0.588959	1.000000	0.703323	0.397845	0.327483	0.304434	0.350053	0.536369

7. Encode the labels.

Encode the labels

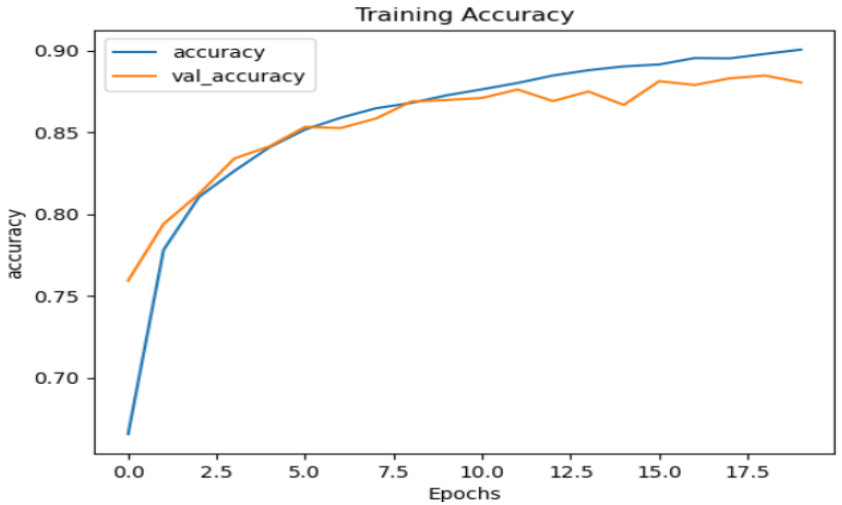
[141]:

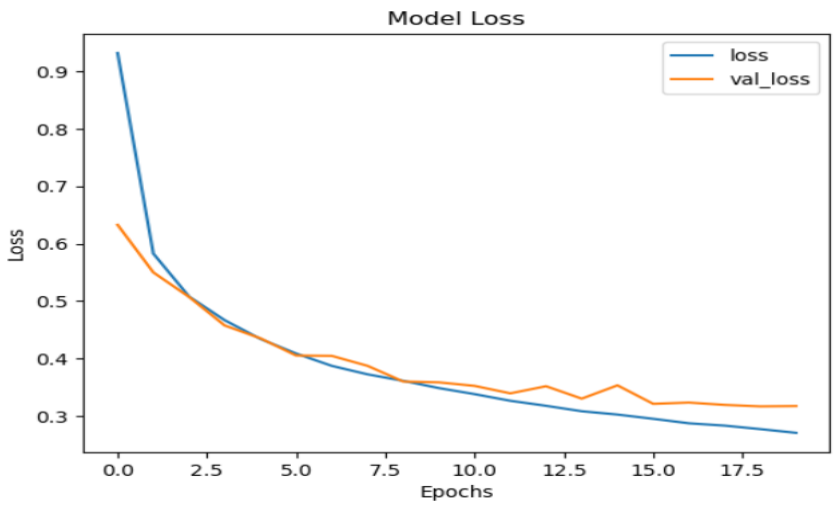
```
# One Hot Encoding
y_train = keras.utils.to_categorical(y_train, 10)
y_val = keras.utils.to_categorical(y_val, 10)
y_test = keras.utils.to_categorical(y_test , 10)
```

Training a CNN neural network

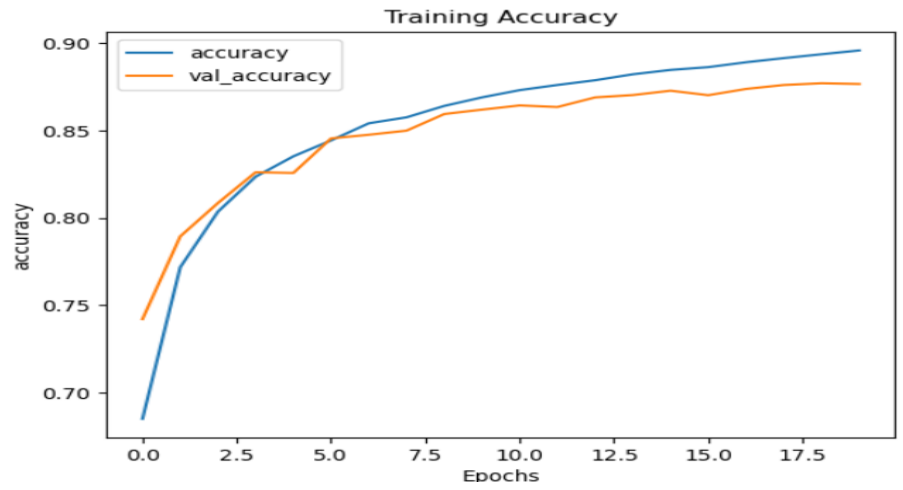
- LeNet-5 network.

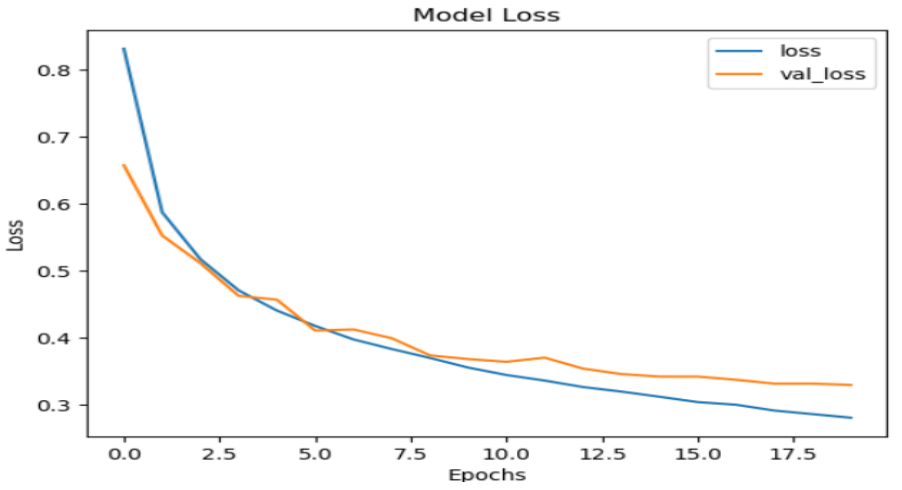
Model_1



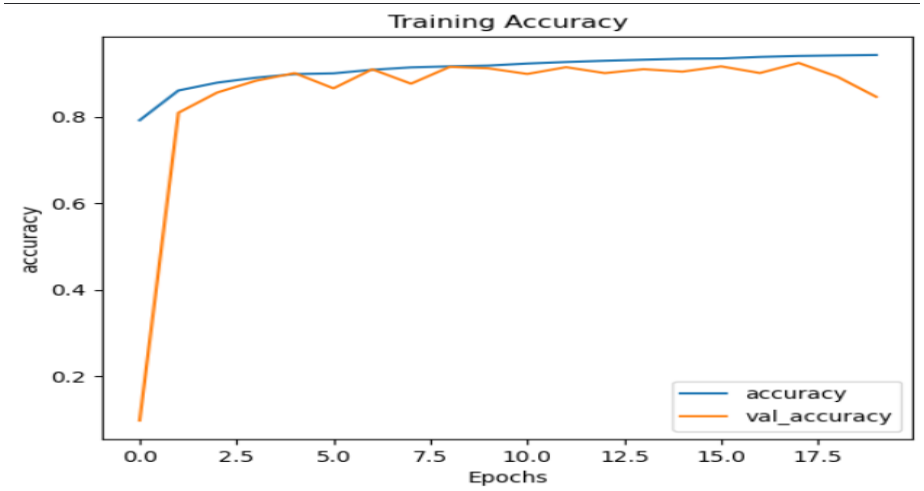


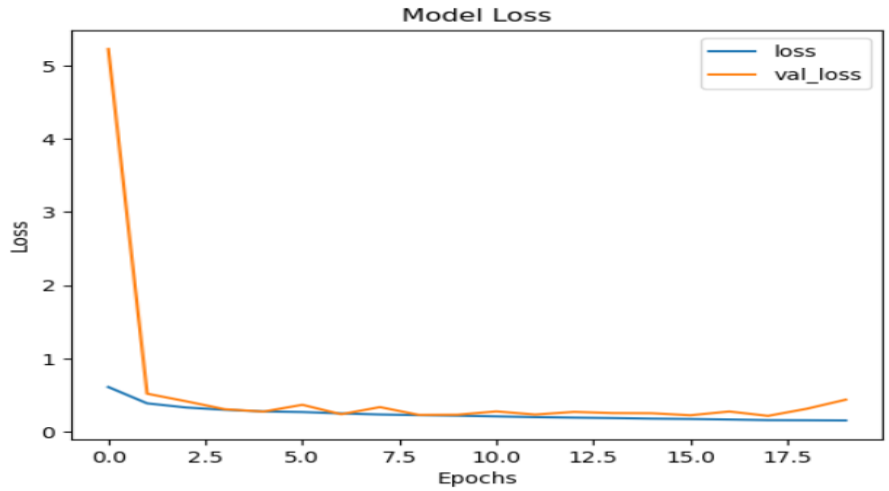
Model_2



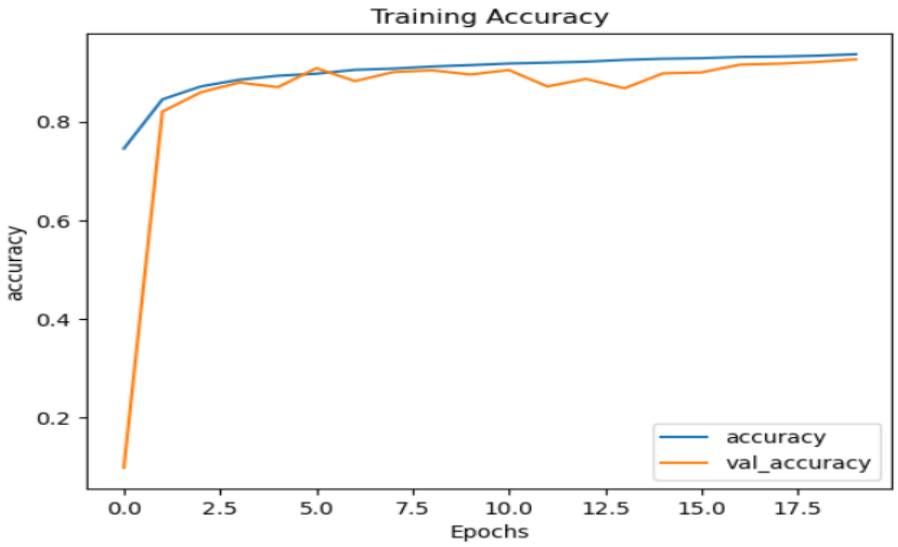


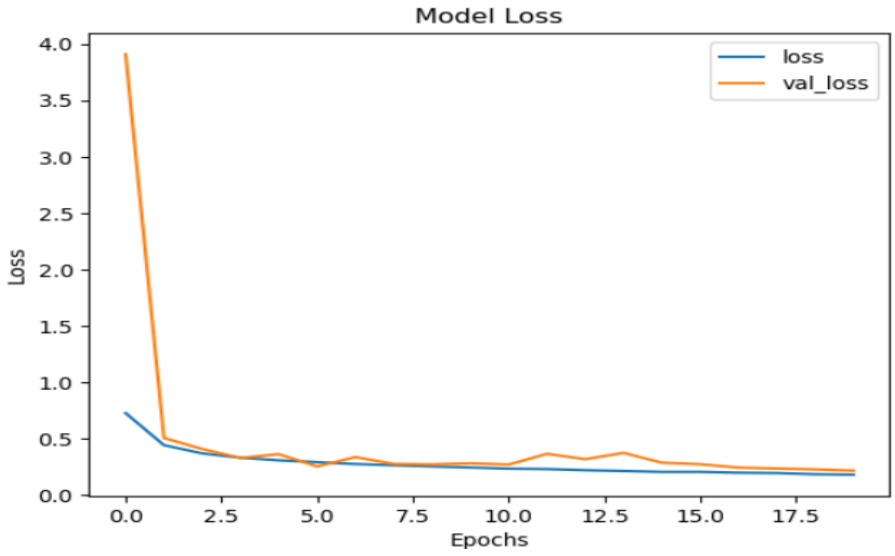
Model_3





Model_4





Modify hyperparameters to get to the best performance you can achieve.

This table to show which model is the best one.

Model_Name	loss	accuracy	Val_loss	Val_accuracy	Test_loss	Test_accuracy
Model_1	0.2708	0.9005	0.3174	0.8804	0.3041	0.8865
Model_2	0.2809	0.8957	0.3297	0.8765	0.3172	0.8803
Model_3	0.1556	0.9425	0.4392	0.8458	0.4137	0.8503
Model_4	0.1787	0.9363	0.2150	0.9260	0.1952	0.9323

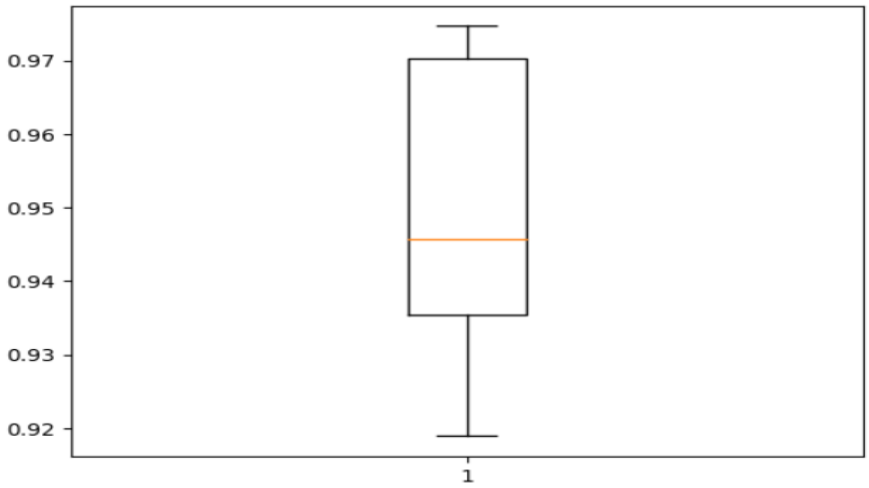
Model _4 is the best one.

Evaluate the model using 5-fold cross-validation.

For 5-fold cross validation

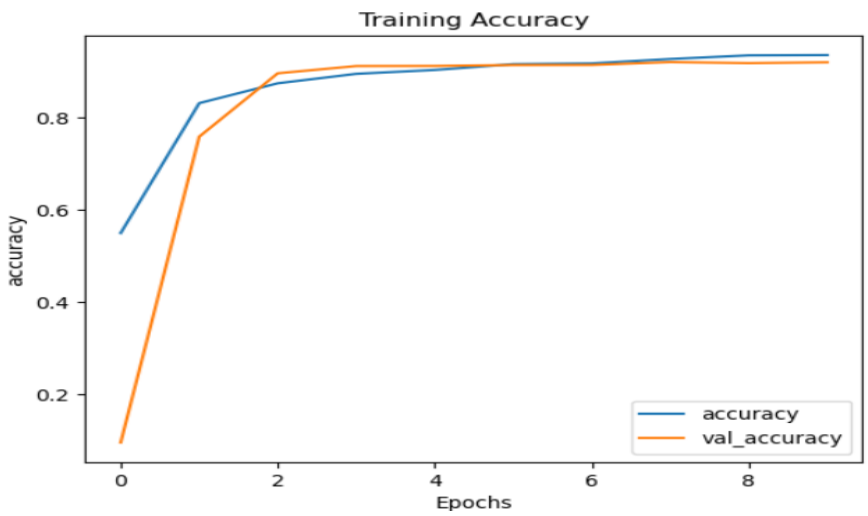
```
]: train_and_cross_validate(model_cv , x_train , y_train , 5 , 20 , 128)

2023-04-07 22:27:46.946026: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout fai
dropout_14/dropout/SelectV2-2-TransposeNHWCToNCHW-LayoutOptimizer
> 91.890
> 93.537
> 94.569
> 97.029
> 97.467
0… ([0.9188991785049438,
  0.9353695511817932,
  0.9456895589828491,
  0.9702908396720886,
  0.9746690392494202],
```

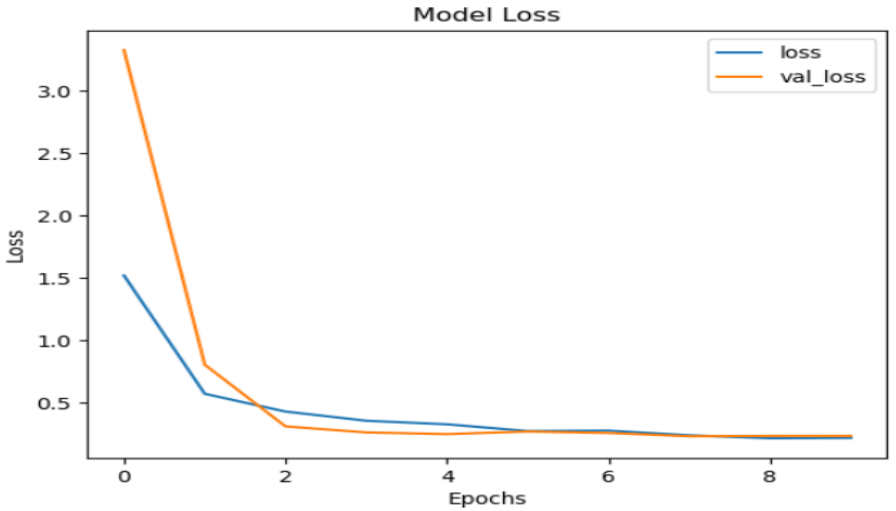
box blot for mean accuracy for 5-fold



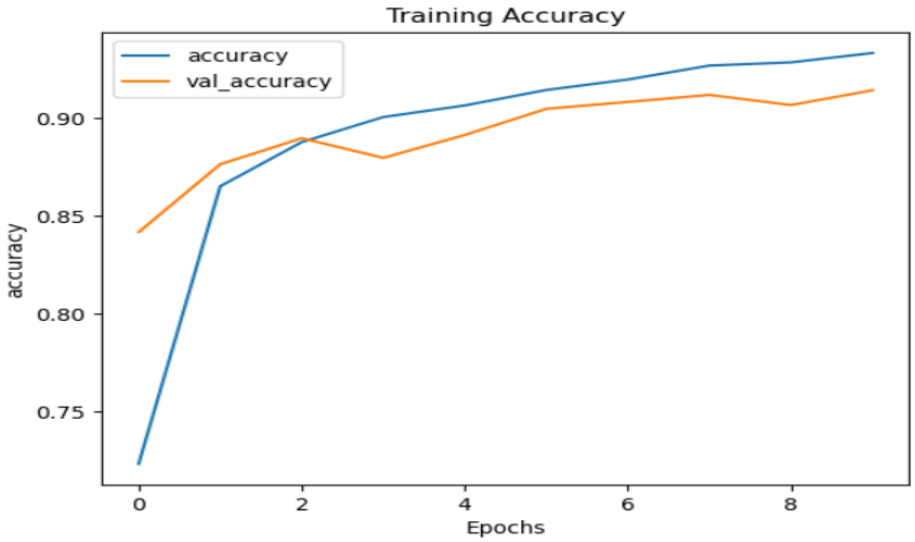
Transfer Learning

ResNet50





VGG16



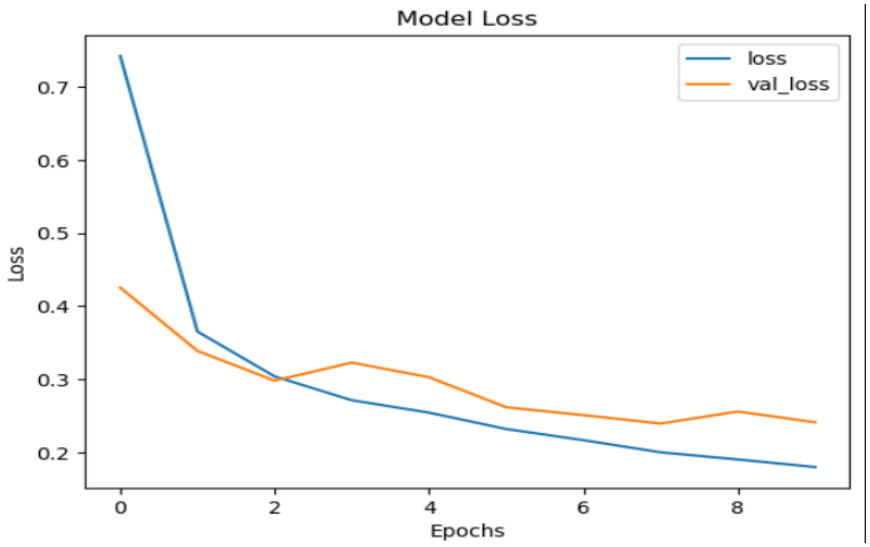


Table show the result when use transfer learning.

Model_name	loss	accuracy	Val_loss	Val_acc	Test_loss	Test_acc
ResNet50	**0.2210**	**0.9352**	**0.2370**	**0.9195**	**0.2238**	**0.9254**
VGG16	**0.1802**	**0.9330**	**0.2413**	**0.9139**	**0.2277**	**0.9219**

AS we see from all the previous trails, the leNet (model_4) is the best model built for these reason:-

- ✓**Training accuracy is. 0.9363**
- ✓**Validation accuracy is. 0. 9260**
- ✓**Has the highest evaluation score. 0.9323**
- ✓**Has the least evaluation loss. 0.1952**

References

1. https://keras.io/api/
2. https://paperswithcode.com/method/vgg-16
3. https://iq.opengenus.org/resnet50-architecture/
4. https://paperswithcode.com/method/resnet

THANK YOU