Name - Ashish Kumar

Email - ashishok88@gmail.com

AI Minor Project Smartknower

Date - 18-08-2022

Importing the Datasets

In [ ]:
```
from keras.datasets import mnist
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,classification_report
```

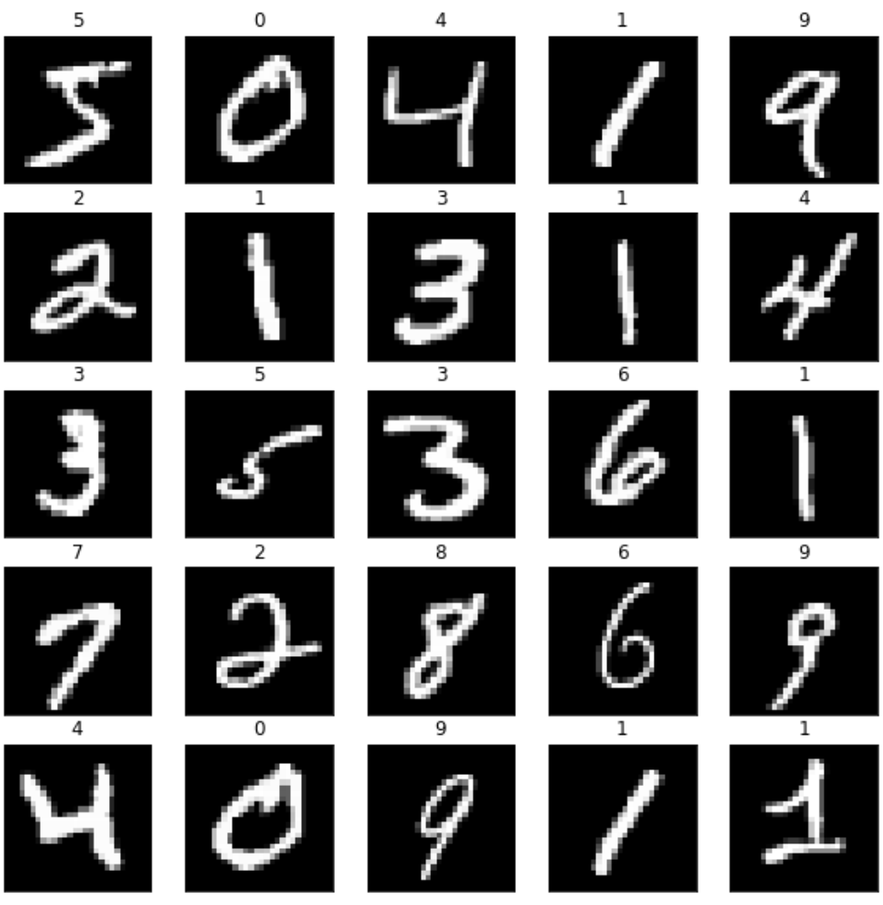
Train and Test split of Data

In [ ]:
```
(x_train, y_train),(x_test, y_test) = mnist.load_data()
```

In [ ]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(10000, 28, 28)
(60000,)
(10000,)
```

In [ ]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i],'gray')
    plt.title(y_train[i])
plt.show()
```



Before Normalization

In [ ]:
```
print(x_train[0])
```

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

In [ ]:
```
#Normalizing the datasets
x_train = x_train/255
x_test = x_test /255
```

After Normalization

Normalization is a technique used to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

In [ ]:
```
print(x_train[0])
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.01176471 0.07058824 0.07058824 0.07058824 0.49411765 0.53333333
  0.68627451 0.10196078 0.65098039 1.         0.96862745 0.49803922
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.11764706 0.14117647 0.36862745 0.60392157
  0.66666667 0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
  0.88235294 0.6745098  0.99215686 0.94901961 0.76470588 0.25098039
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.19215686 0.93333333 0.99215686 0.99215686 0.99215686
  0.99215686 0.99215686 0.99215686 0.99215686 0.99215686 0.98431373
  0.36470588 0.32156863 0.32156863 0.21960784 0.15294118 0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.07058824 0.85882353 0.99215686 0.99215686 0.99215686
  0.99215686 0.99215686 0.77647059 0.71372549 0.96862745 0.94509804
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.31372549 0.61176471 0.41960784 0.99215686
  0.99215686 0.80392157 0.04313725 0.         0.16862745 0.60392157
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.05490196 0.00392157 0.60392157
  0.99215686 0.35294118 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.54509804
  0.99215686 0.74509804 0.00784314 0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.04313725
  0.74509804 0.99215686 0.2745098  0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.1372549  0.94509804 0.88235294 0.62745098 0.42352941 0.00392157
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.31764706 0.94117647 0.99215686 0.99215686 0.46666667
  0.09803922 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.17647059 0.72941176 0.99215686 0.99215686
  0.58823529 0.10588235 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.0627451  0.36470588 0.98823529
  0.99215686 0.73333333 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.97647059
  0.99215686 0.97647059 0.25098039 0.         0.         0.
  0.         0.         0.         0.        ]
```

```
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.18039216 0.50980392 0.71764706 0.99215686
  0.99215686 0.81176471 0.00784314 0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.15294118 0.58039216 0.89803922 0.99215686 0.99215686 0.99215686
  0.98039216 0.71372549 0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.09411765 0.44705882
  0.86666667 0.99215686 0.99215686 0.99215686 0.99215686 0.78823529
  0.30588235 0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.09019608 0.25882353 0.83529412 0.99215686
  0.99215686 0.99215686 0.99215686 0.77647059 0.31764706 0.00784314
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.07058824 0.67058824 0.85882353 0.99215686 0.99215686 0.99215686
  0.99215686 0.76470588 0.31372549 0.03529412 0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.21568627 0.6745098
  0.88627451 0.99215686 0.99215686 0.99215686 0.99215686 0.95686275
  0.52156863 0.04313725 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.53333333 0.99215686
  0.99215686 0.99215686 0.83137255 0.52941176 0.51764706 0.0627451
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]]
```

In [ ]:

```
x_train = np.expand_dims(x_train,-1)
x_test = np.expand_dims(x_test,-1)
x_train.shape
```

Out[ ]:

```
(60000, 28, 28, 1)
```

In [ ]:

```
print(y_train)
print(y_test)
```

```
[5 0 4 ... 5 6 8]
[7 2 1 ... 4 5 6]
```

Building the Model

In [ ]:

```
from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import Dense,Conv2D,Flatten,Dropout,MaxPooling2D
from tensorflow.python.keras.callbacks import EarlyStopping
```

In [ ]:

```
m1 = Sequential()
m1.add(Conv2D(64,(3,3),input_shape = (28,28,1),activation ="relu"))
m1.add(MaxPooling2D(pool_size=(2,2)))

m1.add(Conv2D(32,(3,3),activation ="relu"))
m1.add(MaxPooling2D(pool_size=(2,2)))

m1.add(Flatten())
m1.add(Dropout(0.25))
m1.add(Dense(64,activation='relu'))
m1.add(Dropout(0.25))
m1.add(Dense(32,activation='relu'))
m1.add(Dropout(0.25))
m1.add(Dense(10,activation='softmax'))
```

```
m1.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [ ]:
```
es = EarlyStopping(monitor='val_loss',patience=5)
```

In [ ]:
```
# m1.summary()
```

In [ ]:
```
h1 = m1.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=30,callbacks=[es])
```

```
Epoch 1/30
1875/1875 [==============================] - 63s 33ms/step - loss: 0.3296 - accuracy: 0.8999 - val_los
s: 0.0544 - val_accuracy: 0.9829
Epoch 2/30
1875/1875 [==============================] - 66s 35ms/step - loss: 0.1238 - accuracy: 0.9657 - val_los
s: 0.0399 - val_accuracy: 0.9874
Epoch 3/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0984 - accuracy: 0.9735 - val_los
s: 0.0378 - val_accuracy: 0.9882
Epoch 4/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0813 - accuracy: 0.9780 - val_los
s: 0.0319 - val_accuracy: 0.9911
Epoch 5/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0695 - accuracy: 0.9812 - val_los
s: 0.0319 - val_accuracy: 0.9898
Epoch 6/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0653 - accuracy: 0.9824 - val_los
s: 0.0341 - val_accuracy: 0.9901
Epoch 7/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0575 - accuracy: 0.9845 - val_los
s: 0.0308 - val_accuracy: 0.9905
Epoch 8/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0542 - accuracy: 0.9855 - val_los
s: 0.0266 - val_accuracy: 0.9920
Epoch 9/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0513 - accuracy: 0.9858 - val_los
s: 0.0274 - val_accuracy: 0.9924
Epoch 10/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0494 - accuracy: 0.9863 - val_los
s: 0.0290 - val_accuracy: 0.9911
Epoch 11/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0449 - accuracy: 0.9879 - val_los
s: 0.0267 - val_accuracy: 0.9921
Epoch 12/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0439 - accuracy: 0.9878 - val_los
s: 0.0273 - val_accuracy: 0.9919
Epoch 13/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0415 - accuracy: 0.9883 - val_los
s: 0.0263 - val_accuracy: 0.9921
Epoch 14/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0416 - accuracy: 0.9886 - val_los
s: 0.0289 - val_accuracy: 0.9923
Epoch 15/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0363 - accuracy: 0.9900 - val_los
s: 0.0248 - val_accuracy: 0.9926
Epoch 16/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0396 - accuracy: 0.9891 - val_los
s: 0.0254 - val_accuracy: 0.9924
Epoch 17/30
1875/1875 [==============================] - 63s 34ms/step - loss: 0.0363 - accuracy: 0.9895 - val_los
s: 0.0244 - val_accuracy: 0.9929
Epoch 18/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0344 - accuracy: 0.9903 - val_los
s: 0.0279 - val_accuracy: 0.9925
Epoch 19/30
1875/1875 [==============================] - 64s 34ms/step - loss: 0.0349 - accuracy: 0.9898 - val_los
s: 0.0260 - val_accuracy: 0.9932
Epoch 20/30
1875/1875 [==============================] - 56s 30ms/step - loss: 0.0330 - accuracy: 0.9905 - val_los
s: 0.0268 - val_accuracy: 0.9933
Epoch 21/30
1875/1875 [==============================] - 51s 27ms/step - loss: 0.0293 - accuracy: 0.9912 - val_los
s: 0.0299 - val_accuracy: 0.9924
Epoch 22/30
1875/1875 [==============================] - 51s 27ms/step - loss: 0.0310 - accuracy: 0.9909 - val_los
s: 0.0304 - val_accuracy: 0.9924
```

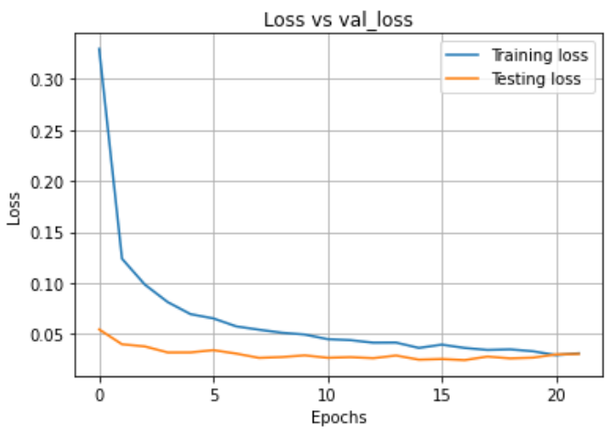
In [ ]:
```
r1 = pd.DataFrame(h1.history)
r1['Epoch'] = h1.epoch
r1
```

Out[ ]:

	loss	accuracy	val_loss	val_accuracy	Epoch
0	0.329579	0.899883	0.054449	0.9829	0
1	0.123824	0.965667	0.039900	0.9874	1
2	0.098423	0.973517	0.037805	0.9882	2
3	0.081296	0.977983	0.031891	0.9911	3
4	0.069451	0.981200	0.031941	0.9898	4
5	0.065269	0.982367	0.034140	0.9901	5
6	0.057474	0.984467	0.030772	0.9905	6
7	0.054180	0.985550	0.026570	0.9920	7
8	0.051251	0.985817	0.027385	0.9924	8
9	0.049444	0.986350	0.029016	0.9911	9
10	0.044950	0.987900	0.026668	0.9921	10
11	0.043935	0.987833	0.027290	0.9919	11
12	0.041457	0.988317	0.026279	0.9921	12
13	0.041551	0.988600	0.028875	0.9923	13
14	0.036328	0.989983	0.024827	0.9926	14
15	0.039556	0.989150	0.025446	0.9924	15
16	0.036314	0.989483	0.024383	0.9929	16
17	0.034363	0.990283	0.027870	0.9925	17
18	0.034915	0.989817	0.026049	0.9932	18
19	0.033017	0.990500	0.026810	0.9933	19
20	0.029269	0.991183	0.029950	0.9924	20
21	0.031033	0.990933	0.030408	0.9924	21

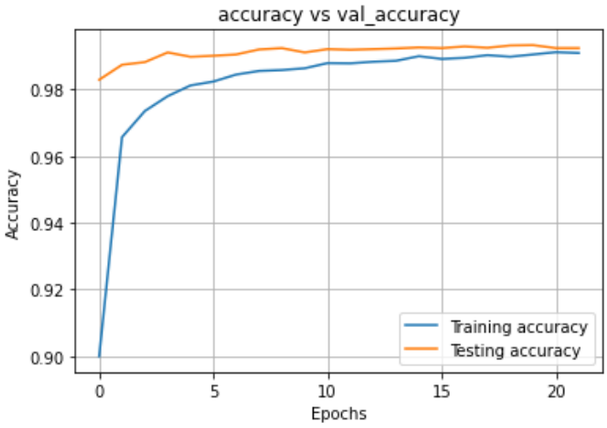
loss vs. val_loss on line chart

In [ ]:
```
plt.plot(r1['Epoch'],r1['loss'],label='Training loss')
plt.plot(r1['Epoch'],r1['val_loss'],label='Testing loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss vs val_loss')
plt.legend()
plt.grid()
plt.show()
```



Accuracy vs. val_accuracy on line chart.

In [ ]:
```
plt.plot(r1['Epoch'],r1['accuracy'],label='Training accuracy')
plt.plot(r1['Epoch'],r1['val_accuracy'],label='Testing accuracy')
plt.xlabel('Epochs')
plt.title('accuracy vs val_accuracy')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()
```



Predictions on test_data

In [ ]:
```
x_test.shape
```

Out[ ]:
```
(10000, 28, 28, 1)
```

In [ ]:
```
ypred = m1.predict(x_test)
print(ypred)
```

```
[[2.22403770e-22 2.33815890e-14 3.48000952e-13 ... 1.00000000e+00
  9.80316353e-19 4.25415649e-12]
 [1.00807485e-13 2.77809620e-09 1.00000000e+00 ... 2.43030371e-14
  3.28084933e-12 1.39159479e-18]
 [1.01202702e-15 1.00000000e+00 4.39212902e-12 ... 5.13374456e-12
  8.87393186e-13 4.88489491e-14]
 ...
 [1.42726269e-23 1.36694667e-16 2.51280699e-18 ... 2.86852858e-14
  4.09258537e-13 4.44604780e-08]
 [1.08984098e-17 6.15559054e-21 4.91154867e-18 ... 2.16584117e-21
  2.32474748e-10 4.75154534e-11]
 [8.45677945e-08 4.62597127e-10 1.70319456e-08 ... 6.04038931e-15
  6.43179078e-07 2.19411538e-11]]
```

In [ ]:
```
print(y_test.shape)
print(ypred.shape)
```

```
(10000,)
(10000, 10)
```

In [ ]:
```
res1 = pd.DataFrame({'y_test':y_test,'y_pred':[*ypred]})
res1.head()
```

Out[ ]:

	y_test	y_pred
0	7	[2.2240377e-22, 2.3381589e-14, 3.4800095e-13, ...
1	2	[1.00807485e-13, 2.7780962e-09, 1.0, 6.368167e...
2	1	[1.012027e-15, 1.0, 4.392129e-12, 5.6413797e-1...
3	0	[1.0, 5.5305414e-14, 2.0595718e-11, 4.4081655e...
4	4	[6.903843e-17, 7.967253e-16, 6.226717e-16, 1.2...

In [ ]:
```
ypred_m1 = [np.argmax(i) for i in ypred]
# print(ypred_m1)
```

Confusion matrix and classification report

In [ ]:
```
cm = confusion_matrix(y_test,ypred_m1)
print(cm)
print(classification_report(y_test,ypred_m1))
```

```
[[ 976    1    0    0    0    0    1    1    1    0]
 [   0 1131    0    1    0    2    0    1    0    0]
 [   2    1 1022    1    0    0    0    5    1    0]
 [   0    0    1 1006    0    1    0    1    1    0]
 [   0    0    0    0  974    0    1    0    1    6]
 [   0    0    0    3    0  886    1    1    0    1]
 [   7    4    0    0    1    3  941    0    2    0]
 [   0    3    3    0    1    0    0 1021    0    0]
 [   1    0    2    0    0    0    0    1  968    2]
 [   0    1    0    0    3    3    0    3    0  999]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       0.99      1.00      0.99      1135
           2       0.99      0.99      0.99      1032
           3       1.00      1.00      1.00      1010
           4       0.99      0.99      0.99       982
           5       0.99      0.99      0.99       892
           6       1.00      0.98      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```