```
from keras.layers import Dense,Dropout
from tensorflow.keras.utils import to_categorical
from keras.datasets import mnist
from tensorflow.keras.utils import model_to_dot
from IPython.display import SVG
import numpy as np
!pip install livelossplot
import livelossplot
plot_losses = livelossplot.PlotLossesKeras()
%matplotlib inline
```

```
Requirement already satisfied: livelossplot in /usr/local/lib/python3.12/dist-packages (0.5.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from livelossplot
Requirement already satisfied: bokeh in /usr/local/lib/python3.12/dist-packages (from livelossplot) (3
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.12/dist-packages (from bokeh->live
Requirement already satisfied: contourpy>=1.2 in /usr/local/lib/python3.12/dist-packages (from bokeh->l
Requirement already satisfied: narwhals>=1.13 in /usr/local/lib/python3.12/dist-packages (from bokeh->l
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.12/dist-packages (from bokeh->live
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.12/dist-packages (from bokeh->
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from bokeh->live
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.12/dist-packages (from bokeh->li
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.12/dist-packages (from bokeh->liv
Requirement already satisfied: tornado>=6.2 in /usr/local/lib/python3.12/dist-packages (from bokeh->liv
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matpl
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matpl
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplo
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from ma
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from Jinja2>
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateuti
```

```
import pandas as pd
from keras.models import Model
```

```
import tensorflow as tf
from keras.layers import *
```

```
df_train=pd.read_csv("/content/drive/MyDrive/Dataset/train (5).csv")
df_test=pd.read_csv("/content/drive/MyDrive/Dataset/test.csv")
```

```
df_train.head()
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel77
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

```
df_test.head()
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel7
0	0	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 784 columns

```
df_features=df_train.iloc[:,1:785]  #
```

```
df_label=df_train.iloc[:,0]
```

```
x_test=df_test.iloc[:,0:784]
```

```
print(x_test.shape)
```

```
(28000, 784)
```

```
df_label.shape
```

```
(42000,)
```

```
from sklearn.model_selection import train_test_split
x_train,x_cv,y_train,y_cv=train_test_split(df_features,df_label,test_size=0.2,random_state=1212)
```

```
print(x_train.shape)
print(x_cv.shape)
print(y_train.shape)
print(y_cv.shape)
```

```
(33600, 784)
(8400, 784)
(33600,)
(8400,)
```

```
#Because it take less space compare to pandas &execute faster
x_train=x_train.to_numpy().reshape(33600,784)
```

```
x_cv=x_cv.to_numpy().reshape(8400,784)
```

```
x_test=x_test.to_numpy().reshape(28000,784)
```

```
print(x_train.shape)
print(x_test.shape)
print(x_cv.shape)
```

```
(33600, 784)
(28000, 784)
(8400, 784)
```

```
print(min(x_cv[1]),max(x_cv[1]))
x_train=x_train.astype('float32')
x_cv=x_cv.astype('float32')
x_test=x_test.astype('float32')
```

```
0 255
```

```
x_train=x_train/255
x_cv=x_cv/255
x_test/=255
num_digit=10
```

```
num_digit=10
y_train=to_categorical(y_train,num_digit)
y_cv=to_categorical(y_cv,num_digit)
```

```
print(y_train[0])
print(y_cv[3])
```

```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
#input parameter
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
num_digit=10
```

```
x_train.shape
```

```
(33600, 784)
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
#INPUT=layer1,layer2,layer3,layer4->output
model=Model(Inp,Output)
model.summary()
```

```
Model: "functional_7"
```

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
hidden_layer_2 (Dense)	(None, 100)	30,100
hidden_layer_3 (Dense)	(None, 100)	10,100
hidden_layer_4 (Dense)	(None, 100)	10,100
output_layer (Dense)	(None, 10)	1,010

```
Total params: 286,810 (1.09 MB)
Trainable params: 286,810 (1.09 MB)
Non-trainable params: 0 (0.00 B)
```

```
#Insert hyper parameter
learning_rate=0.01
trainig_epochs=20
batch_size=100
sgd=tf.keras.optimizers.SGD(learning_rate=learning_rate)
```

```
model.compile(
    loss='categorical_crossentropy',
    optimizer=sgd,
    metrics=['accuracy']
)

history1=model.fit(x_train,y_train,
                   batch_size=batch_size,
                   epochs=trainig_epochs,
                   verbose=2,
                   validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 7s - 20ms/step - accuracy: 0.4296 - loss: 1.8695 - val_accuracy: 0.7380 - val_loss: 1.0865
Epoch 2/20
336/336 - 12s - 37ms/step - accuracy: 0.8144 - loss: 0.6914 - val_accuracy: 0.8639 - val_loss: 0.4846
Epoch 3/20
336/336 - 4s - 12ms/step - accuracy: 0.8839 - loss: 0.4141 - val_accuracy: 0.8945 - val_loss: 0.3609
Epoch 4/20
336/336 - 4s - 11ms/step - accuracy: 0.9040 - loss: 0.3338 - val_accuracy: 0.9121 - val_loss: 0.3070
Epoch 5/20
336/336 - 4s - 12ms/step - accuracy: 0.9153 - loss: 0.2928 - val_accuracy: 0.9186 - val_loss: 0.2813
Epoch 6/20
336/336 - 3s - 8ms/step - accuracy: 0.9235 - loss: 0.2636 - val_accuracy: 0.9236 - val_loss: 0.2663
Epoch 7/20
336/336 - 2s - 7ms/step - accuracy: 0.9294 - loss: 0.2419 - val_accuracy: 0.9269 - val_loss: 0.2425
Epoch 8/20
336/336 - 2s - 7ms/step - accuracy: 0.9355 - loss: 0.2228 - val_accuracy: 0.9320 - val_loss: 0.2278
Epoch 9/20
336/336 - 2s - 7ms/step - accuracy: 0.9402 - loss: 0.2070 - val_accuracy: 0.9356 - val_loss: 0.2166
Epoch 10/20
336/336 - 4s - 12ms/step - accuracy: 0.9440 - loss: 0.1927 - val_accuracy: 0.9393 - val_loss: 0.2051
Epoch 11/20
336/336 - 4s - 11ms/step - accuracy: 0.9468 - loss: 0.1811 - val_accuracy: 0.9427 - val_loss: 0.1950
Epoch 12/20
336/336 - 3s - 8ms/step - accuracy: 0.9507 - loss: 0.1697 - val_accuracy: 0.9455 - val_loss: 0.1859
Epoch 13/20
```

```
336/336 - 3s - 8ms/step - accuracy: 0.9529 - loss: 0.1606 - val_accuracy: 0.9448 - val_loss: 0.1816
Epoch 14/20
336/336 - 3s - 9ms/step - accuracy: 0.9561 - loss: 0.1516 - val_accuracy: 0.9490 - val_loss: 0.1724
Epoch 15/20
336/336 - 4s - 11ms/step - accuracy: 0.9582 - loss: 0.1432 - val_accuracy: 0.9498 - val_loss: 0.1670
Epoch 16/20
336/336 - 4s - 12ms/step - accuracy: 0.9604 - loss: 0.1357 - val_accuracy: 0.9505 - val_loss: 0.1632
Epoch 17/20
336/336 - 2s - 7ms/step - accuracy: 0.9632 - loss: 0.1287 - val_accuracy: 0.9518 - val_loss: 0.1602
Epoch 18/20
336/336 - 2s - 7ms/step - accuracy: 0.9636 - loss: 0.1224 - val_accuracy: 0.9512 - val_loss: 0.1586
Epoch 19/20
336/336 - 4s - 12ms/step - accuracy: 0.9665 - loss: 0.1161 - val_accuracy: 0.9574 - val_loss: 0.1453
Epoch 20/20
336/336 - 4s - 11ms/step - accuracy: 0.9681 - loss: 0.1105 - val_accuracy: 0.9580 - val_loss: 0.1442
```

```
Score=model.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 0.1442139893770218
Test_accuracy: 0.9579761624336243
```

MADE CHANGES TO IMPROVE ACCURACY (model 1)(removed SGD and added ADAM command)

```
# input parameter
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
num_digit=10
```

```
x_train.shape
```

```
(33600, 784)
```

```
y_train.shape
```

```
(33600, 10)
```

```
x_cv.shape
```

```
(8400, 784)
```

```
y_cv.shape
```

```
(8400, 10)
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
```

```
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
#INPUT=layer1,layer2,layer3,layer4->output
model2=Model(Inp,Output)
model2.summary()
```

Model: "functional_8"

Layer (type)	Output Shape	Param #
input_layer_8 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
hidden_layer_2 (Dense)	(None, 100)	30,100
hidden_layer_3 (Dense)	(None, 100)	10,100
hidden_layer_4 (Dense)	(None, 100)	10,100
output_layer (Dense)	(None, 10)	1,010

Total params: 286,810 (1.09 MB)
Trainable params: 286,810 (1.09 MB)
Non-trainable params: 0 (0.00 B)

```
#Insert hyper parameter
learning_rate=0.1
trainig_epochs=20
batch_size=100
adam=tf.keras.optimizers.Adam(learning_rate=learning_rate)
```

```
model2.compile(
   loss='categorical_crossentropy',
                  optimizer= adam,
                  metrics= ['accuracy']
   )
```

```
history3=model2.fit(x_train,y_train,
                  batch_size=batch_size,
                  epochs=trainig_epochs,
                  verbose=2,
                  validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 5s - 15ms/step - accuracy: 0.1045 - loss: 4.7784 - val_accuracy: 0.1086 - val_loss: 2.3047
Epoch 2/20
336/336 - 5s - 14ms/step - accuracy: 0.1046 - loss: 2.3078 - val_accuracy: 0.1086 - val_loss: 2.3037
Epoch 3/20
336/336 - 3s - 9ms/step - accuracy: 0.1031 - loss: 2.3074 - val_accuracy: 0.1038 - val_loss: 2.3057
Epoch 4/20
336/336 - 3s - 9ms/step - accuracy: 0.1040 - loss: 2.3074 - val_accuracy: 0.0963 - val_loss: 2.3069
Epoch 5/20
336/336 - 3s - 9ms/step - accuracy: 0.1026 - loss: 2.3078 - val_accuracy: 0.1038 - val_loss: 2.3082
Epoch 6/20
336/336 - 5s - 14ms/step - accuracy: 0.1033 - loss: 2.3084 - val_accuracy: 0.1086 - val_loss: 2.3093
Epoch 7/20
336/336 - 3s - 10ms/step - accuracy: 0.1051 - loss: 2.3081 - val_accuracy: 0.1086 - val_loss: 2.3073
Epoch 8/20
336/336 - 3s - 9ms/step - accuracy: 0.1056 - loss: 2.3082 - val_accuracy: 0.1035 - val_loss: 2.3112
```

```
Epoch 9/20
336/336 - 3s - 10ms/step - accuracy: 0.1017 - loss: 2.3100 - val_accuracy: 0.0963 - val_loss: 2.3136
Epoch 10/20
336/336 - 5s - 14ms/step - accuracy: 0.1041 - loss: 2.3088 - val_accuracy: 0.1086 - val_loss: 2.3092
Epoch 11/20
336/336 - 4s - 11ms/step - accuracy: 0.1022 - loss: 2.3086 - val_accuracy: 0.1018 - val_loss: 2.3116
Epoch 12/20
336/336 - 3s - 9ms/step - accuracy: 0.1044 - loss: 2.3085 - val_accuracy: 0.1035 - val_loss: 2.3072
Epoch 13/20
336/336 - 3s - 9ms/step - accuracy: 0.1038 - loss: 2.3086 - val_accuracy: 0.1038 - val_loss: 2.3044
Epoch 14/20
336/336 - 5s - 14ms/step - accuracy: 0.1038 - loss: 2.3081 - val_accuracy: 0.1086 - val_loss: 2.3118
Epoch 15/20
336/336 - 3s - 9ms/step - accuracy: 0.1072 - loss: 2.3078 - val_accuracy: 0.1018 - val_loss: 2.3089
Epoch 16/20
336/336 - 5s - 16ms/step - accuracy: 0.1039 - loss: 2.3074 - val_accuracy: 0.1038 - val_loss: 2.3079
Epoch 17/20
336/336 - 8s - 23ms/step - accuracy: 0.1045 - loss: 2.3079 - val_accuracy: 0.1086 - val_loss: 2.3072
Epoch 18/20
336/336 - 3s - 9ms/step - accuracy: 0.1052 - loss: 2.3077 - val_accuracy: 0.1086 - val_loss: 2.3054
Epoch 19/20
336/336 - 3s - 9ms/step - accuracy: 0.1038 - loss: 2.3081 - val_accuracy: 0.1086 - val_loss: 2.3040
Epoch 20/20
336/336 - 4s - 13ms/step - accuracy: 0.1035 - loss: 2.3087 - val_accuracy: 0.1038 - val_loss: 2.3076
```

```
Score=model2.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 2.30759334564209
Test_accuracy: 0.10380952060222626
```

MADE CHANGES TO IMROVE ACCURACY(model 3)(lr = 0.5)

```
# input parameter
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
num_digit=10
```

```
x_train.shape
```

```
(33600, 784)
```

```
y_train.shape
```

```
(33600, 10)
```

```
x_cv.shape
```

```
(8400, 784)
```

```
y_cv.shape
```

```
(8400, 10)
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
#INPUT=layer1,layer2,layer3,layer4->output
model3=Model(Inp,Output)
model3.summary()
```

Model: "functional_9"

Layer (type)	Output Shape	Param #
input_layer_9 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
hidden_layer_2 (Dense)	(None, 100)	30,100
hidden_layer_3 (Dense)	(None, 100)	10,100
hidden_layer_4 (Dense)	(None, 100)	10,100
output_layer (Dense)	(None, 10)	1,010

Total params: 286,810 (1.09 MB)
Trainable params: 286,810 (1.09 MB)
Non-trainable params: 0 (0.00 B)

```
#Insert hyper parameter
learning_rate=0.5
trainig_epochs=20
batch_size=100
adam=tf.keras.optimizers.Adam(learning_rate=learning_rate)
```

```
model3.compile(
   loss='categorical_crossentropy',
                  optimizer= adam,
                  metrics= ['accuracy']
   )
```

```
history4=model3.fit(x_train,y_train,
                  batch_size=batch_size,
                  epochs=trainig_epochs,
                  verbose=2,
                  validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 5s - 16ms/step - accuracy: 0.1017 - loss: 1744.7648 - val_accuracy: 0.1035 - val_loss: 2.3139
Epoch 2/20
336/336 - 3s - 10ms/step - accuracy: 0.1026 - loss: 2.3216 - val_accuracy: 0.1086 - val_loss: 2.3285
Epoch 3/20
336/336 - 5s - 14ms/step - accuracy: 0.0998 - loss: 2.3280 - val_accuracy: 0.1038 - val_loss: 2.3274
Epoch 4/20
336/336 - 3s - 9ms/step - accuracy: 0.1018 - loss: 2.3244 - val_accuracy: 0.0963 - val_loss: 2.3302
Epoch 5/20
336/336 - 3s - 9ms/step - accuracy: 0.1002 - loss: 2.3309 - val_accuracy: 0.0989 - val_loss: 2.3193
Epoch 6/20
336/336 - 3s - 10ms/step - accuracy: 0.1018 - loss: 2.3261 - val_accuracy: 0.0963 - val_loss: 2.3305
```

```
Epoch 7/20
336/336 - 5s - 14ms/step - accuracy: 0.0988 - loss: 2.3309 - val_accuracy: 0.1086 - val_loss: 2.3125
Epoch 8/20
336/336 - 3s - 10ms/step - accuracy: 0.1017 - loss: 2.3334 - val_accuracy: 0.0995 - val_loss: 2.3309
Epoch 9/20
336/336 - 3s - 9ms/step - accuracy: 0.1032 - loss: 2.3316 - val_accuracy: 0.1035 - val_loss: 2.3436
Epoch 10/20
336/336 - 3s - 9ms/step - accuracy: 0.1018 - loss: 2.3308 - val_accuracy: 0.0963 - val_loss: 2.3415
Epoch 11/20
336/336 - 5s - 16ms/step - accuracy: 0.1008 - loss: 2.3342 - val_accuracy: 0.1018 - val_loss: 2.3352
Epoch 12/20
336/336 - 3s - 9ms/step - accuracy: 0.0981 - loss: 2.3334 - val_accuracy: 0.0963 - val_loss: 2.3105
Epoch 13/20
336/336 - 6s - 17ms/step - accuracy: 0.1024 - loss: 2.3313 - val_accuracy: 0.1086 - val_loss: 2.3384
Epoch 14/20
336/336 - 4s - 12ms/step - accuracy: 0.1006 - loss: 2.3334 - val_accuracy: 0.1038 - val_loss: 2.3528
Epoch 15/20
336/336 - 3s - 9ms/step - accuracy: 0.1013 - loss: 2.3303 - val_accuracy: 0.0989 - val_loss: 2.3310
Epoch 16/20
336/336 - 5s - 15ms/step - accuracy: 0.1032 - loss: 2.3371 - val_accuracy: 0.1086 - val_loss: 2.3200
Epoch 17/20
336/336 - 5s - 14ms/step - accuracy: 0.1008 - loss: 2.3318 - val_accuracy: 0.0929 - val_loss: 2.3289
Epoch 18/20
336/336 - 3s - 10ms/step - accuracy: 0.1032 - loss: 2.3284 - val_accuracy: 0.0963 - val_loss: 2.3268
Epoch 19/20
336/336 - 3s - 9ms/step - accuracy: 0.0997 - loss: 2.3307 - val_accuracy: 0.0985 - val_loss: 2.3216
Epoch 20/20
336/336 - 3s - 9ms/step - accuracy: 0.1028 - loss: 2.3316 - val_accuracy: 0.1035 - val_loss: 2.3204
```

```
Score=model3.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 2.320424795150757
Test_accuracy: 0.10345238447189331
```

MADE CHANGES TO IMROVE ACCURACY(model 4)(Added hidden layer) bold text

```
# input parameter
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
n_hidden_5=100
num_digit=10
```

```
x_train.shape
```

```
(33600, 784)
```

```
y_train.shape
```

```
(33600, 10)
```

```
x_cv.shape
```

```
(8400, 784)
```

```
y_cv.shape
```

```
(8400, 10)
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
X=Dense(n_hidden_5,activation='relu',name='hidden_layer_5')(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
#INPUT=layer1,layer2,layer3,layer4->,LAYER5,output
model3=Model(Inp,Output)
model3.summary()
```

Model: "functional_10"

Layer (type)	Output Shape	Param #
input_layer_10 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
hidden_layer_2 (Dense)	(None, 100)	30,100
hidden_layer_3 (Dense)	(None, 100)	10,100
hidden_layer_4 (Dense)	(None, 100)	10,100
hidden_layer_5 (Dense)	(None, 100)	10,100
output_layer (Dense)	(None, 10)	1,010

Total params: 296,910 (1.13 MB)
Trainable params: 296,910 (1.13 MB)
Non-trainable params: 0 (0.00 B)

```
#Insert hyper parameter
learning_rate=0.01
trainig_epochs=20
batch_size=100
adam=tf.keras.optimizers.Adam(learning_rate=learning_rate)
```

```
model3.compile(
    loss='categorical_crossentropy',
                    optimizer= adam,
                    metrics= ['accuracy']
    )
```

```
history4=model3.fit(x_train,y_train,
                    batch_size=batch_size,
                    epochs=trainig_epochs,
                    verbose=2,
                    validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 6s - 18ms/step - accuracy: 0.8850 - loss: 0.3799 - val_accuracy: 0.9227 - val_loss: 0.2808
Epoch 2/20
336/336 - 3s - 10ms/step - accuracy: 0.9505 - loss: 0.1880 - val_accuracy: 0.9573 - val_loss: 0.1622
Epoch 3/20
336/336 - 3s - 9ms/step - accuracy: 0.9582 - loss: 0.1627 - val_accuracy: 0.9562 - val_loss: 0.1733
Epoch 4/20
336/336 - 5s - 15ms/step - accuracy: 0.9653 - loss: 0.1308 - val_accuracy: 0.9540 - val_loss: 0.1887
Epoch 5/20
336/336 - 3s - 9ms/step - accuracy: 0.9710 - loss: 0.1116 - val_accuracy: 0.9623 - val_loss: 0.1525
Epoch 6/20
336/336 - 3s - 9ms/step - accuracy: 0.9737 - loss: 0.1041 - val_accuracy: 0.9618 - val_loss: 0.1612
Epoch 7/20
336/336 - 3s - 9ms/step - accuracy: 0.9721 - loss: 0.1172 - val_accuracy: 0.9661 - val_loss: 0.1744
Epoch 8/20
336/336 - 5s - 14ms/step - accuracy: 0.9740 - loss: 0.1039 - val_accuracy: 0.9614 - val_loss: 0.1769
Epoch 9/20
336/336 - 3s - 10ms/step - accuracy: 0.9747 - loss: 0.1093 - val_accuracy: 0.8965 - val_loss: 0.5277
Epoch 10/20
336/336 - 3s - 9ms/step - accuracy: 0.9767 - loss: 0.0975 - val_accuracy: 0.9635 - val_loss: 0.1752
Epoch 11/20
336/336 - 3s - 9ms/step - accuracy: 0.9775 - loss: 0.1040 - val_accuracy: 0.9671 - val_loss: 0.1695
Epoch 12/20
336/336 - 5s - 14ms/step - accuracy: 0.9760 - loss: 0.1077 - val_accuracy: 0.9693 - val_loss: 0.1812
Epoch 13/20
336/336 - 3s - 9ms/step - accuracy: 0.9789 - loss: 0.0900 - val_accuracy: 0.9707 - val_loss: 0.1508
Epoch 14/20
336/336 - 3s - 10ms/step - accuracy: 0.9816 - loss: 0.0776 - val_accuracy: 0.9704 - val_loss: 0.1515
Epoch 15/20
336/336 - 3s - 10ms/step - accuracy: 0.9832 - loss: 0.0723 - val_accuracy: 0.9688 - val_loss: 0.1528
Epoch 16/20
336/336 - 5s - 14ms/step - accuracy: 0.9840 - loss: 0.0648 - val_accuracy: 0.9658 - val_loss: 0.1778
Epoch 17/20
336/336 - 3s - 9ms/step - accuracy: 0.9849 - loss: 0.0664 - val_accuracy: 0.9656 - val_loss: 0.1937
Epoch 18/20
336/336 - 3s - 10ms/step - accuracy: 0.9821 - loss: 0.0703 - val_accuracy: 0.9677 - val_loss: 0.1828
Epoch 19/20
336/336 - 4s - 11ms/step - accuracy: 0.9834 - loss: 0.0739 - val_accuracy: 0.9631 - val_loss: 0.1917
Epoch 20/20
336/336 - 5s - 15ms/step - accuracy: 0.9861 - loss: 0.0589 - val_accuracy: 0.9694 - val_loss: 0.1801
```

```
Score=model3.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 0.1800655871629715
Test_accuracy: 0.9694047570228577
```

MADE CHANGES TO IMROVE ACCURACY(model 4)(removing adam)

```
# input parameter
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
num_digit=10
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
model4=Model(Inp,Output)
model4.summary()
```

Model: "functional_11"

Layer (type)	Output Shape	Param #
input_layer_11 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
hidden_layer_2 (Dense)	(None, 100)	30,100
hidden_layer_3 (Dense)	(None, 100)	10,100
hidden_layer_4 (Dense)	(None, 100)	10,100
output_layer (Dense)	(None, 10)	1,010

Total params: 286,810 (1.09 MB)
Trainable params: 286,810 (1.09 MB)
Non-trainable params: 0 (0.00 B)

```
#Insert hyper parameter
trainig_epochs=20
batch_size=100
```

```
model4.compile(
loss='categorical_crossentropy',
optimizer= 'Adam',
metrics= ['accuracy']
)
```

```
history5=model4.fit(x_train,y_train,
batch_size=batch_size,
epochs=trainig_epochs,
verbose=2,
validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 5s - 15ms/step - accuracy: 0.9026 - loss: 0.3323 - val_accuracy: 0.9557 - val_loss: 0.1469
Epoch 2/20
336/336 - 5s - 15ms/step - accuracy: 0.9632 - loss: 0.1224 - val_accuracy: 0.9623 - val_loss: 0.1229
Epoch 3/20
336/336 - 3s - 10ms/step - accuracy: 0.9753 - loss: 0.0790 - val_accuracy: 0.9688 - val_loss: 0.1061
Epoch 4/20
336/336 - 5s - 14ms/step - accuracy: 0.9805 - loss: 0.0608 - val_accuracy: 0.9702 - val_loss: 0.1020
Epoch 5/20
336/336 - 7s - 19ms/step - accuracy: 0.9870 - loss: 0.0417 - val_accuracy: 0.9756 - val_loss: 0.0844
Epoch 6/20
336/336 - 3s - 9ms/step - accuracy: 0.9903 - loss: 0.0302 - val_accuracy: 0.9762 - val_loss: 0.0942
Epoch 7/20
336/336 - 3s - 9ms/step - accuracy: 0.9895 - loss: 0.0306 - val_accuracy: 0.9719 - val_loss: 0.1059
Epoch 8/20
336/336 - 3s - 9ms/step - accuracy: 0.9916 - loss: 0.0264 - val_accuracy: 0.9757 - val_loss: 0.1025
```

```
Epoch 9/20
336/336 - 5s - 14ms/step - accuracy: 0.9938 - loss: 0.0197 - val_accuracy: 0.9719 - val_loss: 0.1215
Epoch 10/20
336/336 - 3s - 10ms/step - accuracy: 0.9927 - loss: 0.0226 - val_accuracy: 0.9761 - val_loss: 0.0984
Epoch 11/20
336/336 - 3s - 10ms/step - accuracy: 0.9948 - loss: 0.0168 - val_accuracy: 0.9763 - val_loss: 0.1069
Epoch 12/20
336/336 - 3s - 10ms/step - accuracy: 0.9950 - loss: 0.0171 - val_accuracy: 0.9742 - val_loss: 0.1124
Epoch 13/20
336/336 - 5s - 14ms/step - accuracy: 0.9946 - loss: 0.0158 - val_accuracy: 0.9689 - val_loss: 0.1475
Epoch 14/20
336/336 - 3s - 9ms/step - accuracy: 0.9952 - loss: 0.0153 - val_accuracy: 0.9764 - val_loss: 0.1176
Epoch 15/20
336/336 - 5s - 15ms/step - accuracy: 0.9939 - loss: 0.0186 - val_accuracy: 0.9757 - val_loss: 0.1097
Epoch 16/20
336/336 - 4s - 13ms/step - accuracy: 0.9954 - loss: 0.0141 - val_accuracy: 0.9750 - val_loss: 0.1098
Epoch 17/20
336/336 - 4s - 10ms/step - accuracy: 0.9958 - loss: 0.0134 - val_accuracy: 0.9768 - val_loss: 0.1174
Epoch 18/20
336/336 - 5s - 14ms/step - accuracy: 0.9978 - loss: 0.0067 - val_accuracy: 0.9770 - val_loss: 0.1189
Epoch 19/20
336/336 - 3s - 10ms/step - accuracy: 0.9962 - loss: 0.0106 - val_accuracy: 0.9732 - val_loss: 0.1457
Epoch 20/20
336/336 - 4s - 13ms/step - accuracy: 0.9954 - loss: 0.0144 - val_accuracy: 0.9752 - val_loss: 0.1206
```

```
Score=model4.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 0.12060751765966415
Test_accuracy: 0.9752380847930908
```

MADE CHANGES TO IMROVE ACCURACY(model 5)(DROPOUT)

```
# input parameter
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
num_digit=10
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X = Dropout(0.3)(X)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X = Dropout(0.3)(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
X = Dropout(0.3)(X)
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
X = Dropout(0.3)(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
model5=Model(Inp,Output)
model5.summary()
```

```
Model: "functional_12"
```

Layer (type)	Output Shape	Param #
input_layer_12 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
dropout_9 (Dropout)	(None, 300)	0
hidden_layer_2 (Dense)	(None, 100)	30,100
dropout_10 (Dropout)	(None, 100)	0
hidden_layer_3 (Dense)	(None, 100)	10,100
dropout_11 (Dropout)	(None, 100)	0
hidden_layer_4 (Dense)	(None, 100)	10,100
dropout_12 (Dropout)	(None, 100)	0
output_layer (Dense)	(None, 10)	1,010

```
Total params: 286,810 (1.09 MB)
Trainable params: 286,810 (1.09 MB)
Non-trainable params: 0 (0.00 B)
```

```
#Insert hyper parameter
trainig_epochs=20
batch_size=100
```

```
model5.compile(
   loss='categorical_crossentropy',
                  optimizer= 'Adam',
                  metrics= ['accuracy']
)
```

```
history6=model5.fit(x_train,y_train,
                  batch_size=batch_size,
                  epochs=trainig_epochs,
                  verbose=2,
                  validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 5s - 16ms/step - accuracy: 0.7777 - loss: 0.6930 - val_accuracy: 0.9423 - val_loss: 0.1988
Epoch 2/20
336/336 - 4s - 12ms/step - accuracy: 0.9250 - loss: 0.2703 - val_accuracy: 0.9582 - val_loss: 0.1491
Epoch 3/20
336/336 - 4s - 12ms/step - accuracy: 0.9453 - loss: 0.1960 - val_accuracy: 0.9607 - val_loss: 0.1421
Epoch 4/20
336/336 - 4s - 12ms/step - accuracy: 0.9532 - loss: 0.1684 - val_accuracy: 0.9685 - val_loss: 0.1114
Epoch 5/20
336/336 - 4s - 11ms/step - accuracy: 0.9620 - loss: 0.1364 - val_accuracy: 0.9695 - val_loss: 0.1210
Epoch 6/20
336/336 - 5s - 16ms/step - accuracy: 0.9668 - loss: 0.1201 - val_accuracy: 0.9682 - val_loss: 0.1208
Epoch 7/20
336/336 - 3s - 10ms/step - accuracy: 0.9691 - loss: 0.1108 - val_accuracy: 0.9707 - val_loss: 0.1081
Epoch 8/20
336/336 - 3s - 10ms/step - accuracy: 0.9736 - loss: 0.0941 - val_accuracy: 0.9749 - val_loss: 0.0961
Epoch 9/20
336/336 - 4s - 11ms/step - accuracy: 0.9732 - loss: 0.0903 - val_accuracy: 0.9746 - val_loss: 0.0950
Epoch 10/20
```

```
336/336 - 5s - 13ms/step - accuracy: 0.9768 - loss: 0.0831 - val_accuracy: 0.9748 - val_loss: 0.0994
Epoch 11/20
336/336 - 3s - 10ms/step - accuracy: 0.9772 - loss: 0.0801 - val_accuracy: 0.9775 - val_loss: 0.0957
Epoch 12/20
336/336 - 3s - 10ms/step - accuracy: 0.9798 - loss: 0.0747 - val_accuracy: 0.9773 - val_loss: 0.0904
Epoch 13/20
336/336 - 4s - 12ms/step - accuracy: 0.9807 - loss: 0.0678 - val_accuracy: 0.9780 - val_loss: 0.0879
Epoch 14/20
336/336 - 4s - 12ms/step - accuracy: 0.9813 - loss: 0.0670 - val_accuracy: 0.9767 - val_loss: 0.0936
Epoch 15/20
336/336 - 3s - 10ms/step - accuracy: 0.9806 - loss: 0.0652 - val_accuracy: 0.9770 - val_loss: 0.0939
Epoch 16/20
336/336 - 3s - 10ms/step - accuracy: 0.9831 - loss: 0.0604 - val_accuracy: 0.9764 - val_loss: 0.0991
Epoch 17/20
336/336 - 4s - 13ms/step - accuracy: 0.9846 - loss: 0.0523 - val_accuracy: 0.9775 - val_loss: 0.1020
Epoch 18/20
336/336 - 4s - 11ms/step - accuracy: 0.9841 - loss: 0.0551 - val_accuracy: 0.9811 - val_loss: 0.0878
Epoch 19/20
336/336 - 3s - 10ms/step - accuracy: 0.9853 - loss: 0.0511 - val_accuracy: 0.9792 - val_loss: 0.1020
Epoch 20/20
336/336 - 3s - 10ms/step - accuracy: 0.9854 - loss: 0.0513 - val_accuracy: 0.9783 - val_loss: 0.1012
```

```
Score=model5.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 0.10122441500425339
Test_accuracy: 0.9783333539962769
```

```
n_input=784
n_hidden_1=300
n_hidden_2=100
n_hidden_3=100
n_hidden_4=100
n_hidden_5=200
num_digit=10
```

```
Inp=Input(shape=(784,))
X=Dense(n_hidden_1,activation='relu',name='hidden_layer_1')(Inp)
X = Dropout(0.3)(X)
X=Dense(n_hidden_2,activation='relu',name='hidden_layer_2')(X)
X = Dropout(0.3)(X)
X=Dense(n_hidden_3,activation='relu',name='hidden_layer_3')(X)
X = Dropout(0.3)(X)
X=Dense(n_hidden_4,activation='relu',name='hidden_layer_4')(X)
X = Dropout(0.3)(X)
X=Dense(n_hidden_5,activation='relu',name='hidden_layer_5')(X)
X=Dropout(0.3)(X)
Output=Dense(num_digit,activation='softmax',name='output_layer')(X)
```

```
model5=Model(Inp,Output)
model5.summary()
```

```
Model: "functional_13"
```

Layer (type)	Output Shape	Param #
input_layer_13 (InputLayer)	(None, 784)	0
hidden_layer_1 (Dense)	(None, 300)	235,500
dropout_13 (Dropout)	(None, 300)	0
hidden_layer_2 (Dense)	(None, 100)	30,100
dropout_14 (Dropout)	(None, 100)	0
hidden_layer_3 (Dense)	(None, 100)	10,100
dropout_15 (Dropout)	(None, 100)	0
hidden_layer_4 (Dense)	(None, 100)	10,100
dropout_16 (Dropout)	(None, 100)	0
hidden_layer_5 (Dense)	(None, 200)	20,200
dropout_17 (Dropout)	(None, 200)	0
output_layer (Dense)	(None, 10)	2,010

```
Total params: 308,010 (1.17 MB)
Trainable params: 308,010 (1.17 MB)
Non-trainable params: 0 (0.00 B)
```

```
trainig_epochs=20
batch_size=100
```

```
model5.compile(
   loss='categorical_crossentropy',
                   optimizer= 'Adam',
                   metrics= ['accuracy']
)
```

```
history6=model5.fit(x_train,y_train,
                   batch_size=batch_size,
                   epochs=trainig_epochs,
                   verbose=2,
                   validation_data=(x_cv,y_cv))
```

```
Epoch 1/20
336/336 - 7s - 21ms/step - accuracy: 0.7439 - loss: 0.7705 - val_accuracy: 0.9395 - val_loss: 0.2158
Epoch 2/20
336/336 - 4s - 11ms/step - accuracy: 0.9220 - loss: 0.2852 - val_accuracy: 0.9558 - val_loss: 0.1626
Epoch 3/20
336/336 - 5s - 15ms/step - accuracy: 0.9405 - loss: 0.2163 - val_accuracy: 0.9623 - val_loss: 0.1390
Epoch 4/20
336/336 - 4s - 13ms/step - accuracy: 0.9513 - loss: 0.1811 - val_accuracy: 0.9661 - val_loss: 0.1274
Epoch 5/20
336/336 - 4s - 11ms/step - accuracy: 0.9587 - loss: 0.1558 - val_accuracy: 0.9682 - val_loss: 0.1169
Epoch 6/20
336/336 - 4s - 13ms/step - accuracy: 0.9646 - loss: 0.1368 - val_accuracy: 0.9708 - val_loss: 0.1174
Epoch 7/20
336/336 - 5s - 15ms/step - accuracy: 0.9662 - loss: 0.1217 - val_accuracy: 0.9721 - val_loss: 0.1032
Epoch 8/20
336/336 - 4s - 11ms/step - accuracy: 0.9698 - loss: 0.1116 - val_accuracy: 0.9732 - val_loss: 0.1076
```

```
Epoch 9/20
336/336 - 4s - 11ms/step - accuracy: 0.9725 - loss: 0.1035 - val_accuracy: 0.9726 - val_loss: 0.1048
Epoch 10/20
336/336 - 5s - 16ms/step - accuracy: 0.9756 - loss: 0.0909 - val_accuracy: 0.9726 - val_loss: 0.1101
Epoch 11/20
336/336 - 4s - 11ms/step - accuracy: 0.9763 - loss: 0.0884 - val_accuracy: 0.9755 - val_loss: 0.1024
Epoch 12/20
336/336 - 5s - 16ms/step - accuracy: 0.9760 - loss: 0.0871 - val_accuracy: 0.9750 - val_loss: 0.1003
Epoch 13/20
336/336 - 5s - 16ms/step - accuracy: 0.9765 - loss: 0.0860 - val_accuracy: 0.9775 - val_loss: 0.0930
Epoch 14/20
336/336 - 4s - 11ms/step - accuracy: 0.9788 - loss: 0.0777 - val_accuracy: 0.9767 - val_loss: 0.0977
Epoch 15/20
336/336 - 5s - 16ms/step - accuracy: 0.9797 - loss: 0.0746 - val_accuracy: 0.9763 - val_loss: 0.1057
Epoch 16/20
336/336 - 5s - 16ms/step - accuracy: 0.9818 - loss: 0.0672 - val_accuracy: 0.9760 - val_loss: 0.1038
Epoch 17/20
336/336 - 4s - 11ms/step - accuracy: 0.9827 - loss: 0.0641 - val_accuracy: 0.9742 - val_loss: 0.1011
Epoch 18/20
336/336 - 4s - 12ms/step - accuracy: 0.9822 - loss: 0.0647 - val_accuracy: 0.9769 - val_loss: 0.0977
Epoch 19/20
336/336 - 6s - 17ms/step - accuracy: 0.9837 - loss: 0.0581 - val_accuracy: 0.9780 - val_loss: 0.0919
Epoch 20/20
336/336 - 4s - 13ms/step - accuracy: 0.9841 - loss: 0.0571 - val_accuracy: 0.9780 - val_loss: 0.0959
```

```
Score=model5.evaluate(x_cv,y_cv,verbose=0)
print('Test_loss:',Score[0])
print('Test_accuracy:',Score[1])
```

```
Test_loss: 0.09589793533086777
Test_accuracy: 0.9779762029647827
```

```
test_pred=pd.DataFrame(model5.predict(x_test,batch_size=200))
test_pred=pd.DataFrame(test_pred.idxmax(axis=1))
test_pred.index.name='Image Id'
test_pred=test_pred.rename(columns={0:'Label'}).reset_index()
test_pred['Image Id']=test_pred['Image Id']+1
test_pred.head()
```

```
140/140 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step
```

	Image Id	Label
0	1	2
1	2	0
2	3	9
3	4	4
4	5	3

Next steps: Generate code with `test_pred` New interactive sheet