```
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
```

```
Collecting livelossplot
  Downloading livelossplot-0.5.6-py3-none-any.whl.metadata (8.9 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from livelossplot) (3.10.0)
Requirement already satisfied: bokeh in /usr/local/lib/python3.12/dist-packages (from livelossplot) (3.7.3)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (3.1.6)
Requirement already satisfied: contourpy>=1.2 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (1.3.3)
Requirement already satisfied: narwhals>=1.13 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (2.7.0)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (2.0.2)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (25.0)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (2.2.2)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (11.3.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (6.0.3)
Requirement already satisfied: tornado>=6.2 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (6.4.2)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.12/dist-packages (from bokeh->livelossplot) (2025.4.(
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->livelossplot) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->livelossplot) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->livelossplot) (1.4.9)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->livelossplot) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib->livelossplot) (2.9.(
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from Jinja2>=2.9->bokeh->livelossplot) (3
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->bokeh->livelossplot) (2025
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->bokeh->livelossplot) (20
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib->livelossp
Downloading livelossplot-0.5.6-py3-none-any.whl (23 kB)
Installing collected packages: livelossplot
Successfully installed livelossplot-0.5.6
```

```
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	pixel775	pixel776	pixel777	pixel77
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	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	pixel775	pixel776	pixel777	pixel7
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	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"

Layer (type)	Output Shape	Param #
input_layer (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 - 6s - 19ms/step - accuracy: 0.5498 - loss: 1.6466 - val_accuracy: 0.8177 - val_loss: 0.7706
Epoch 2/20
336/336 - 6s - 17ms/step - accuracy: 0.8540 - loss: 0.5438 - val_accuracy: 0.8875 - val_loss: 0.4139
```

```
Epoch 3/20
336/336 - 9s - 26ms/step - accuracy: 0.8934 - loss: 0.3746 - val_accuracy: 0.9020 - val_loss: 0.3380
Epoch 4/20
336/336 - 9s - 28ms/step - accuracy: 0.9087 - loss: 0.3160 - val_accuracy: 0.9114 - val_loss: 0.2947
Epoch 5/20
336/336 - 8s - 24ms/step - accuracy: 0.9185 - loss: 0.2820 - val_accuracy: 0.9196 - val_loss: 0.2736
Epoch 6/20
336/336 - 4s - 12ms/step - accuracy: 0.9251 - loss: 0.2567 - val_accuracy: 0.9254 - val_loss: 0.2533
Epoch 7/20
336/336 - 4s - 10ms/step - accuracy: 0.9305 - loss: 0.2371 - val_accuracy: 0.9283 - val_loss: 0.2410
Epoch 8/20
336/336 - 2s - 7ms/step - accuracy: 0.9351 - loss: 0.2209 - val_accuracy: 0.9268 - val_loss: 0.2424
Epoch 9/20
```

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

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
```

```
y_train.shape
```

```
x_cv.shape
```

```
y_cv.shape
```

```
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()
```

```
#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))
```

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

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
```

```
y_train.shape
```

```
x_cv.shape
```

```
y_cv.shape
```

```
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()
```

```
#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))
```

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

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
```

```
y_train.shape
```

```
x_cv.shape
```

```
y_cv.shape
```

```
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()
```

```
#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))
```

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

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()
```

```
#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))
```

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

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()
```

```
#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))
```

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

```
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()
```

```
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))
```

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

```
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()
```