```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
%matplotlib inline
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

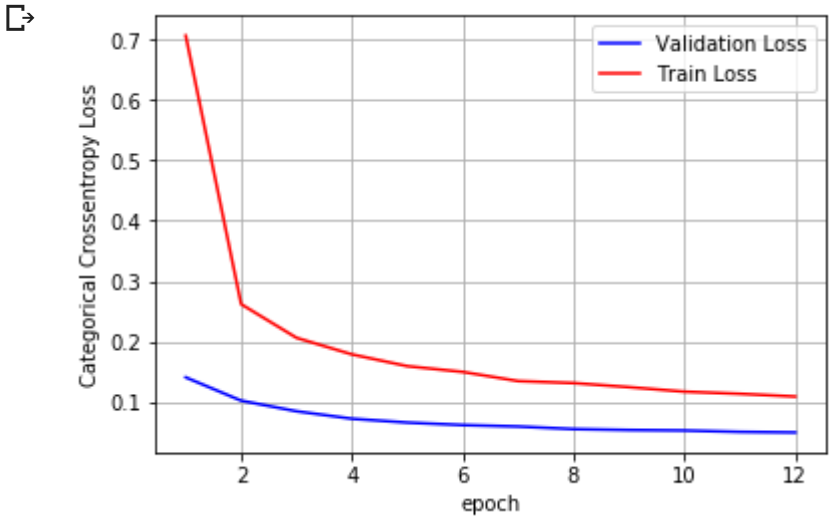
3x3 kernel and 3 layered CNN using Adadelta

```
model1 = Sequential()
model1.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))
model1.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))
model1.add(Conv2D(64,kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dropout(0.25))
model1.add(Dense(num_classes, activation='softmax'))
model1.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),m
history=model1.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_da
score1= model1.evaluate(x_test,y_test, verbose=0)
print('Test loss:', score1[0])
print('Test accuracy:',score1[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 35s 583us/step - loss: 0.7067 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 34s 570us/step - loss: 0.2618 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 34s 572us/step - loss: 0.2060 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 35s 577us/step - loss: 0.1785 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 34s 567us/step - loss: 0.1591 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 34s 561us/step - loss: 0.1495 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 34s 573us/step - loss: 0.1343 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 34s 570us/step - loss: 0.1310 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 34s 560us/step - loss: 0.1243 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 34s 560us/step - loss: 0.1168 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 34s 562us/step - loss: 0.1132 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 34s 561us/step - loss: 0.1087 - acc: 0
Test loss: 0.04917911008154042
Test accuracy: 0.985
```

```
%matplotlib inline
```

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```



5 Layer CNN with 5x5 kernel

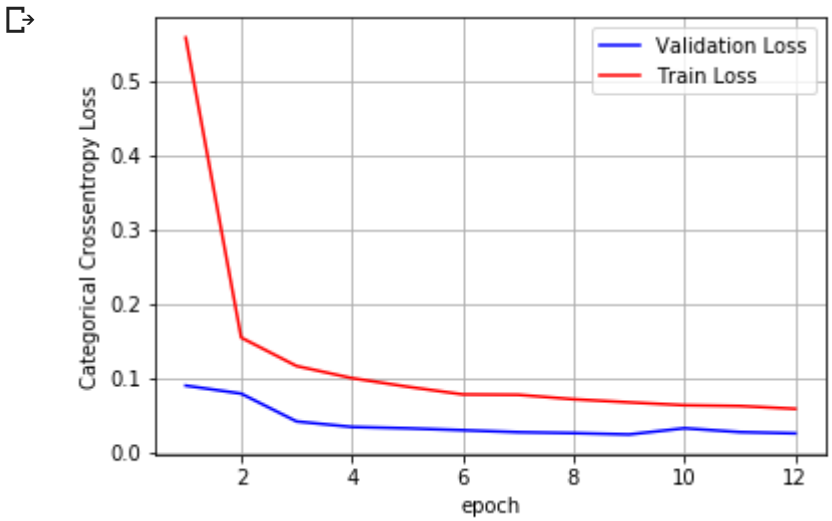
```
from keras.layers import Dense, Dropout, Flatten, Activation, BatchNormalization
model2 = Sequential()
model2.add(Conv2D(32, kernel_size=5,input_shape=(28, 28, 1), activation = 'relu',padding='same'))
model2.add(MaxPooling2D(2,2))
model2.add(BatchNormalization())
model2.add(Dropout(0.4))
model2.add(Conv2D(32, kernel_size=5, activation = 'relu',padding='same'))
model2.add(MaxPooling2D(2,2))
model2.add(BatchNormalization())
model2.add(Dropout(0.4))
model2.add(Conv2D(64, kernel_size=5,activation = 'relu',padding='same'))
model2.add(MaxPooling2D(2,2))
model2.add(BatchNormalization())
model2.add(Dropout(0.4))
model2.add(Conv2D(64, kernel_size=5,activation = 'relu',padding='same'))
model2.add(MaxPooling2D(2,2))
model2.add(BatchNormalization())
model2.add(Dropout(0.4))
model2.add(Conv2D(128, kernel_size=3, activation = 'relu',padding='same'))
model2.add(BatchNormalization())
model2.add(Flatten())
model2.add(Dense(256, activation = "relu"))
model2.add(Dropout(0.4))
model2.add(Dense(128,activation = "relu"))
model2.add(Dropout(0.4))
model2.add(Dense(10, activation = "softmax"))
```

```
model2.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),m
history2=model2.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
score2 = model2.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score2[0])
print('Test accuracy:', score2[1])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.5580 - acc: 0.
Epoch 2/12
60000/60000 [==============================] - 176s 3ms/step - loss: 0.1541 - acc: 0.
Epoch 3/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.1158 - acc: 0.
Epoch 4/12
60000/60000 [==============================] - 182s 3ms/step - loss: 0.0995 - acc: 0.
Epoch 5/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0879 - acc: 0.
Epoch 6/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0775 - acc: 0.
Epoch 7/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0770 - acc: 0.
Epoch 8/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0710 - acc: 0.
Epoch 9/12
60000/60000 [==============================] - 176s 3ms/step - loss: 0.0668 - acc: 0.
Epoch 10/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0630 - acc: 0.
Epoch 11/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0619 - acc: 0.
Epoch 12/12
60000/60000 [==============================] - 176s 3ms/step - loss: 0.0581 - acc: 0.
Test loss: 0.025024591614387873
Test accuracy: 0.9941
```

```
%matplotlib inline
```

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history2.history['val_loss']
ty = history2.history['loss']
plt_dynamic(x, vy, ty, ax)
```



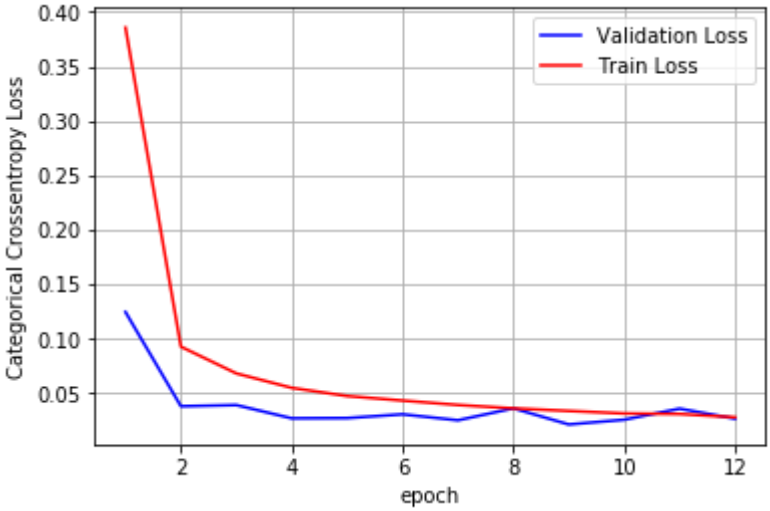
7 Hidden layer with few Convolution layers having 5x5 kernel Maxpooling, dropout

```
from keras.layers import Dense, Dropout, Flatten, Activation, BatchNormalization
model3 = Sequential()
model3.add(Conv2D(32, kernel_size=5,input_shape=(28, 28, 1), activation = 'relu',padding='same'))
model3.add(Conv2D(32, kernel_size=5, activation = 'relu',padding='same'))
model3.add(MaxPooling2D(2,2))
model3.add(BatchNormalization())
model3.add(Dropout(0.4))
model3.add(Conv2D(64, kernel_size=5,activation = 'relu',padding='same'))
model3.add(Conv2D(64, kernel_size=5,activation = 'relu',padding='same'))
model3.add(MaxPooling2D(2,2))
model3.add(BatchNormalization())
model3.add(Dropout(0.4))
model3.add(Conv2D(64, kernel_size=3, activation = 'relu',padding='same'))
model3.add(BatchNormalization())
model3.add(Conv2D(64, kernel_size=5,activation = 'relu',padding='same'))
model3.add(Conv2D(64, kernel_size=5,activation = 'relu',padding='same'))
model3.add(MaxPooling2D(2,2))
model3.add(BatchNormalization())
model3.add(Dropout(0.4))
model3.add(Flatten())
model3.add(Dense(256, activation = "relu"))
model3.add(Dropout(0.4))
model3.add(Dense(128, activation = "relu"))
model3.add(Dropout(0.4))
model3.add(Dense(10,activation = "softmax"))
model3.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),m
history3=model3.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:79

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 753s 13ms/step - loss: 0.3855 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 753s 13ms/step - loss: 0.0928 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 749s 12ms/step - loss: 0.0682 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 753s 13ms/step - loss: 0.0550 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 757s 13ms/step - loss: 0.0474 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 754s 13ms/step - loss: 0.0433 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 748s 12ms/step - loss: 0.0393 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 748s 12ms/step - loss: 0.0361 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 752s 13ms/step - loss: 0.0338 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 750s 13ms/step - loss: 0.0315 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 750s 12ms/step - loss: 0.0310 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 751s 13ms/step - loss: 0.0280 - acc: 0
```

```
%matplotlib inline
score3 = model3.evaluate(x_test,y_test, verbose=0)
print('Test loss:', score3[0])
print('Test accuracy:',score3[1])
score5 = model3.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score3[0])
print('Test accuracy:',score3[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history3.history['val_loss']
ty = history3.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.026654037279491352
Test accuracy: 0.994
Test loss: 0.026654037279491352
Test accuracy: 0.994
```



3x3 kernel and 3 layered CNN using SGD

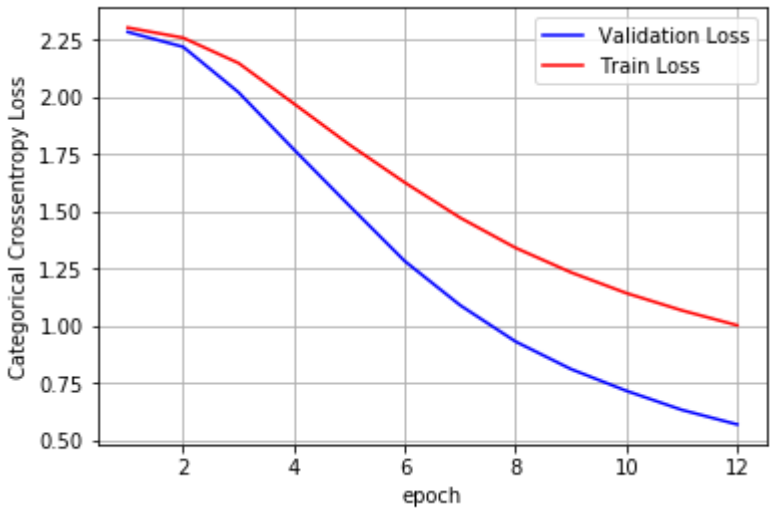
```
model4 = Sequential()
model4.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Dropout(0.5))
model4.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Dropout(0.5))
model4.add(Conv2D(64,kernel_size=(3, 3), activation='relu'))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Dropout(0.5))
model4.add(Flatten())
model4.add(Dense(128, activation='relu'))
model4.add(Dropout(0.5))
model4.add(Dense(num_classes, activation='softmax'))
model4.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.SGD(),metric
history4=model4.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
score4 = model4.evaluate(x_test,y_test, verbose=0)
```

```
print('Test loss:', score4[0])
print('Test accuracy:',score4[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 35s 579us/step - loss: 2.3039 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 33s 553us/step - loss: 2.2594 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 33s 552us/step - loss: 2.1491 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 33s 553us/step - loss: 1.9716 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 33s 555us/step - loss: 1.7928 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 33s 557us/step - loss: 1.6274 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 34s 563us/step - loss: 1.4735 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 33s 555us/step - loss: 1.3420 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 33s 554us/step - loss: 1.2349 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 33s 554us/step - loss: 1.1442 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 33s 556us/step - loss: 1.0689 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 33s 554us/step - loss: 1.0037 - acc: 0
Test loss: 0.5706797729969024
Test accuracy: 0.872
```

```
fig,ax = plt.subplots(1,1)
```

```
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

vy = history4.history['val_loss']
ty = history4.history['loss']
plt_dynamic(x, vy, ty, ax)
```



3x3 kernel and 3 layered CNN using SGD with sigmoid activation

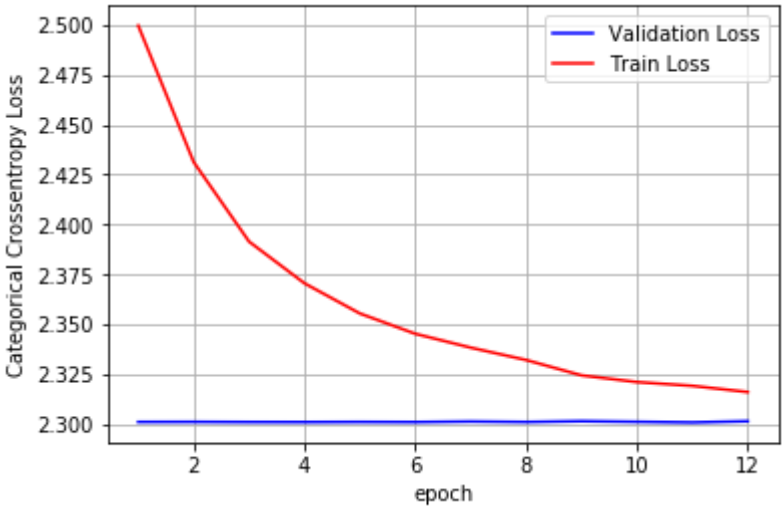
```
model5 = Sequential()
model5.add(Conv2D(16, kernel_size=(3, 3),activation='sigmoid',input_shape=input_shape))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.5))
model5.add(Conv2D(32, kernel_size=(3, 3),activation='sigmoid',input_shape=input_shape))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.5))
model5.add(Conv2D(64,kernel_size=(3, 3), activation='sigmoid'))
model5.add(MaxPooling2D(pool_size=(2,2)))
model5.add(Dropout(0.5))
model5.add(Flatten())
model5.add(Dense(128, activation='sigmoid'))
model5.add(Dropout(0.5))
model5.add(Dense(num_classes, activation='softmax'))
model5.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.SGD(),metric
history5=model5.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 35s 576us/step - loss: 2.4994 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 34s 559us/step - loss: 2.4312 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 34s 563us/step - loss: 2.3915 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 34s 569us/step - loss: 2.3706 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 34s 573us/step - loss: 2.3555 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 35s 576us/step - loss: 2.3454 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 34s 561us/step - loss: 2.3385 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 34s 572us/step - loss: 2.3323 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 35s 582us/step - loss: 2.3246 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 35s 583us/step - loss: 2.3212 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 34s 575us/step - loss: 2.3194 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 34s 566us/step - loss: 2.3163 - acc: 0
```

```
%matplotlib inline
```

```
score5 = model5.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score5[0])
print('Test accuracy:',score5[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

vy = history5.history['val_loss']
ty = history5.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 2.3016598129272463
Test accuracy: 0.1135
```



CNN using 3x3 kernel matrix with tanh activation function

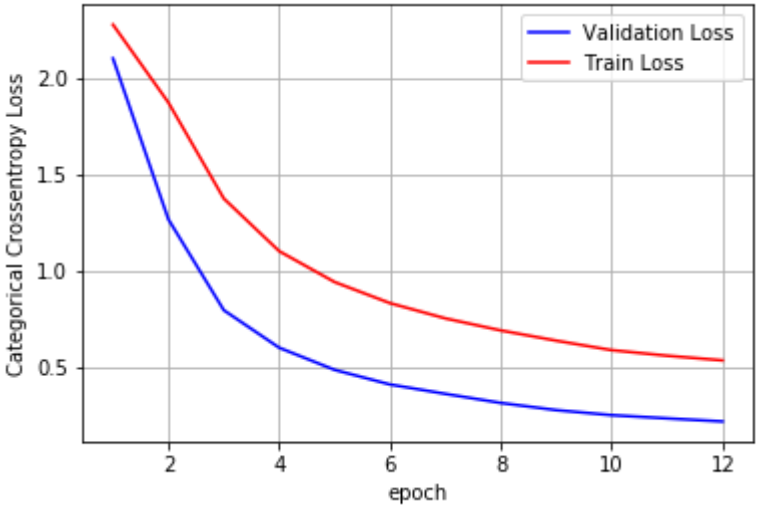
```
model6 = Sequential()
model6.add(Conv2D(16, kernel_size=(3, 3),activation='tanh',input_shape=input_shape))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.4))
model6.add(Conv2D(32, kernel_size=(3, 3),activation='tanh',input_shape=input_shape))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.4))
model6.add(Conv2D(64,kernel_size=(3, 3), activation='tanh'))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.4))
model6.add(Flatten())
model6.add(Dense(128, activation='tanh'))
model6.add(Dropout(0.4))
model6.add(Dense(num_classes,activation='softmax'))
model6.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.SGD(),metric
history6=model6.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 35s 585us/step - loss: 2.2758 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 34s 571us/step - loss: 1.8721 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 34s 564us/step - loss: 1.3745 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 35s 576us/step - loss: 1.1018 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 34s 571us/step - loss: 0.9419 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.8325 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 34s 567us/step - loss: 0.7535 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 34s 567us/step - loss: 0.6911 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 33s 555us/step - loss: 0.6381 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 34s 574us/step - loss: 0.5890 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 35s 584us/step - loss: 0.5601 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 35s 582us/step - loss: 0.5360 - acc: 0
```

```
%matplotlib inline
```

```
score5 = model5.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score5[0])
print('Test accuracy:',score5[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

vy = history6.history['val_loss']
ty = history6.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 2.3016598129272463
Test accuracy: 0.1135
```



3x3 kernel and 3 layered CNN using adam with ReLu activation

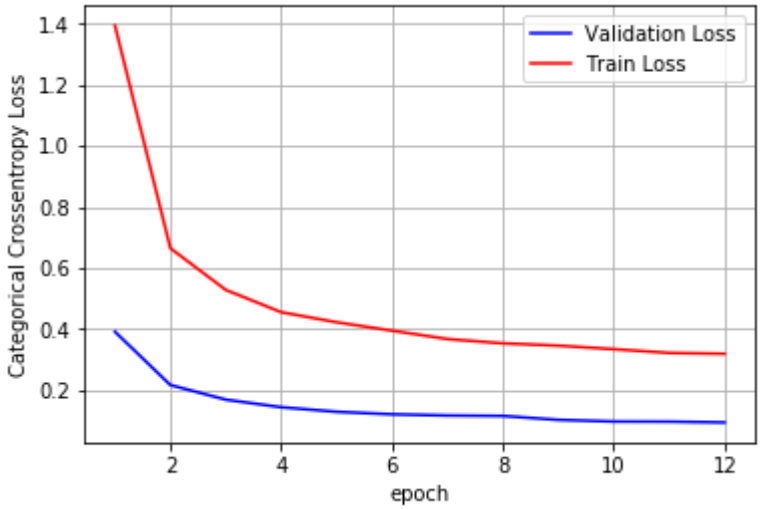
```
model7 = Sequential()
model7.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model7.add(MaxPooling2D(pool_size=(2, 2)))
model7.add(Dropout(0.5))
model7.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model7.add(MaxPooling2D(pool_size=(2, 2)))
model7.add(Dropout(0.5))
model7.add(Conv2D(64,kernel_size=(3, 3), activation='relu'))
model7.add(MaxPooling2D(pool_size=(2, 2)))
model7.add(Dropout(0.5))
model7.add(Flatten())
model7.add(Dense(128, activation='relu'))
model7.add(Dropout(0.5))
model7.add(Dense(num_classes, activation='softmax'))
model7.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.adam(),metri
history7=model7.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
score7 = model7.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score7[0])
print('Test accuracy:',score7[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 36s 599us/step - loss: 1.3935 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 35s 578us/step - loss: 0.6644 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 35s 580us/step - loss: 0.5280 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 35s 583us/step - loss: 0.4550 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 34s 574us/step - loss: 0.4220 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 35s 579us/step - loss: 0.3945 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 35s 576us/step - loss: 0.3671 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 35s 575us/step - loss: 0.3530 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 34s 571us/step - loss: 0.3455 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 35s 581us/step - loss: 0.3339 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 35s 580us/step - loss: 0.3216 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 35s 580us/step - loss: 0.3189 - acc: 0
Test loss: 0.09384770478904247
Test accuracy: 0.9714
```

```
%matplotlib inline
```

```
score7 = model7.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score7[0])
print('Test accuracy:',score7[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.history we will have a list of length equal to number of epochs
vy = history7.history['val_loss']
ty = history7.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 2.3016598129272463
Test accuracy: 0.1135
```



3x3 + 3 layered + Adam + sigmoid + dropout

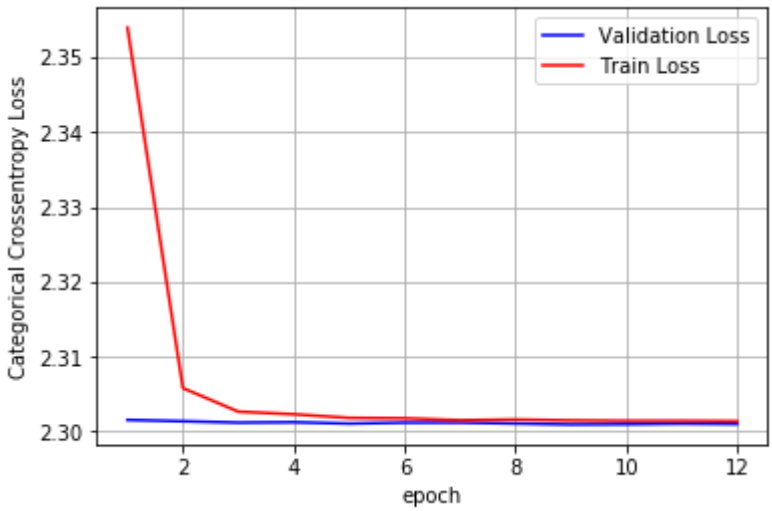
```
model8 = Sequential()
model8.add(Conv2D(16, kernel_size=(3, 3),activation='sigmoid',input_shape=input_shape))
model8.add(MaxPooling2D(pool_size=(2, 2)))
model8.add(Dropout(0.5))
model8.add(Conv2D(32, kernel_size=(3, 3),activation='sigmoid',input_shape=input_shape))
model8.add(MaxPooling2D(pool_size=(2, 2)))
model8.add(Dropout(0.5))
model8.add(Conv2D(64,kernel_size=(3, 3), activation='sigmoid'))
model8.add(MaxPooling2D(pool_size=(2, 2)))
model8.add(Dropout(0.5))
model8.add(Flatten())
model8.add(Dense(128, activation='sigmoid'))
model8.add(Dropout(0.5))
model8.add(Dense(num_classes, activation='softmax'))
model8.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.adam(),metri
history8=model8.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
score8 = model8.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score8[0])
print('Test accuracy:',score8[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 30s 500us/step - loss: 2.3539 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 29s 486us/step - loss: 2.3058 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3027 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 29s 488us/step - loss: 2.3023 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3018 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 29s 484us/step - loss: 2.3018 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 29s 489us/step - loss: 2.3015 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3016 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 29s 482us/step - loss: 2.3015 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 29s 484us/step - loss: 2.3014 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3014 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 29s 481us/step - loss: 2.3014 - acc: 0
Test loss: 2.301026412200928
Test accuracy: 0.1135
```

```
%matplotlib inline
```

```
score8 = model8.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score8[0])
print('Test accuracy:',score8[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history8.history['val_loss']
ty = history8.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 2.301026412200928
Test accuracy: 0.1135
```



3x3 + 3 layered + Adam + tanh + dropout

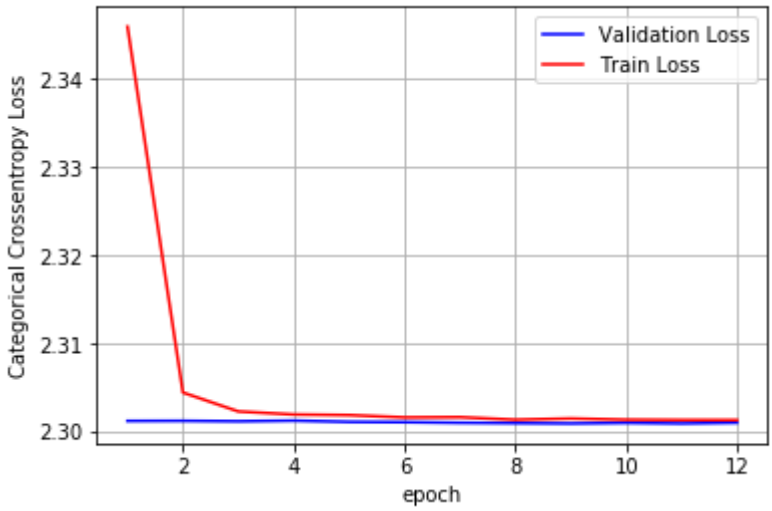
```
model9 = Sequential()
model9.add(Conv2D(16, kernel_size=(3, 3),activation='tanh',input_shape=input_shape))
model9.add(MaxPooling2D(pool_size=(2, 2)))
model9.add(Dropout(0.5))
model9.add(Conv2D(32, kernel_size=(3, 3),activation='tanh',input_shape=input_shape))
model9.add(MaxPooling2D(pool_size=(2, 2)))
model9.add(Dropout(0.5))
model9.add(Conv2D(64,kernel_size=(3, 3), activation='tanh'))
model9.add(MaxPooling2D(pool_size=(2, 2)))
model9.add(Dropout(0.5))
model9.add(Flatten())
model9.add(Dense(128, activation='tanh'))
model9.add(Dropout(0.5))
model9.add(Dense(num_classes, activation='softmax'))
model9.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.adam(),metri
history9=model9.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation_d
score9 = model9.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score9[0])
print('Test accuracy:',score9[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 30s 503us/step - loss: 2.3460 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 29s 489us/step - loss: 2.3045 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 29s 488us/step - loss: 2.3023 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 29s 486us/step - loss: 2.3020 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 29s 486us/step - loss: 2.3019 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 29s 483us/step - loss: 2.3016 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3017 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 29s 481us/step - loss: 2.3014 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 29s 478us/step - loss: 2.3015 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 29s 478us/step - loss: 2.3014 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 29s 480us/step - loss: 2.3013 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 29s 482us/step - loss: 2.3013 - acc: 0
Test loss: 2.3010480514526366
Test accuracy: 0.1135
```

```
%matplotlib inline
```

```
score9 = model9.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score9[0])
print('Test accuracy:',score9[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history9.history['val_loss']
ty = history9.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 2.3010480514526366
Test accuracy: 0.1135
```



3x3 + 3 layered + Adagrad + sigmoid + dropout

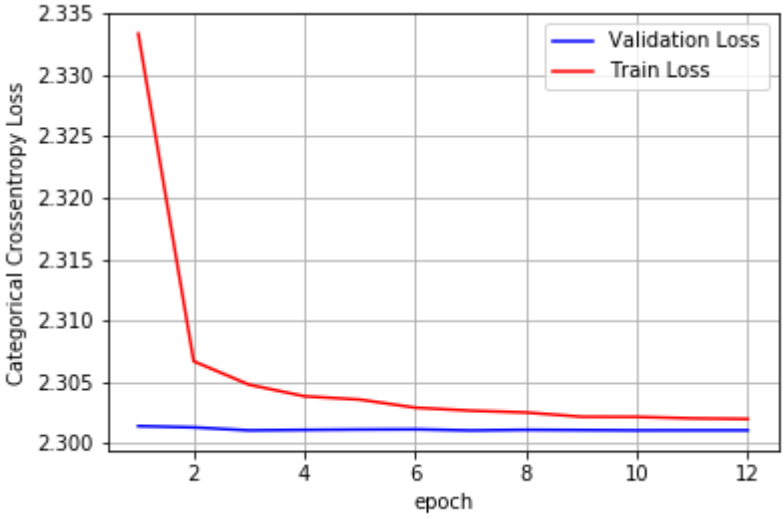
```
model10 = Sequential()
model10.add(Conv2D(16, kernel_size=(3, 3),activation='sigmoid',input_shape=input_shape))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.5))
model10.add(Conv2D(32, kernel_size=(3, 3),activation='sigmoid',input_shape=input_shape))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.5))
model10.add(Conv2D(64,kernel_size=(3, 3), activation='sigmoid'))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.5))
model10.add(Flatten())
model10.add(Dense(128,activation='sigmoid'))
model10.add(Dropout(0.5))
model10.add(Dense(num_classes, activation='softmax'))
model10.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adagrad(),m
history10=model10.fit(x_train, y_train,batch_size=batch_size,epochs=epochs, verbose=1, validation
score10 = model10.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score10[0])
print('Test accuracy:',score10[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 30s 508us/step - loss: 2.3334 - acc: 0
Epoch 2/12
60000/60000 [==============================] - 29s 491us/step - loss: 2.3067 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 29s 490us/step - loss: 2.3048 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3038 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 29s 482us/step - loss: 2.3036 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 29s 484us/step - loss: 2.3029 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3027 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 29s 483us/step - loss: 2.3025 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 29s 485us/step - loss: 2.3022 - acc: 0
Epoch 10/12
60000/60000 [==============================] - 29s 487us/step - loss: 2.3022 - acc: 0
Epoch 11/12
60000/60000 [==============================] - 29s 482us/step - loss: 2.3020 - acc: 0
Epoch 12/12
60000/60000 [==============================] - 29s 487us/step - loss: 2.3020 - acc: 0
Test loss: 2.3010630470275877
Test accuracy: 0.1135
```

```
%matplotlib inline
```

```
score10 = model10.evaluate(x_test,y_test,verbose=0)
print('Test loss:', score10[0])
print('Test accuracy:',score10[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, v
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history10.history['val_loss']
ty = history10.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 2.3010630470275877
Test accuracy: 0.1135
```




```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Hiddden layers", "kernel size",'activation function','optimizer',"accuracy"]
x.add_row([3,'3*3','relu','adadelta',98.54])
x.add_row([5,'5*5','relu','adadelta', 99.5])
x.add_row([7,'5*5','relu','adadelta', 99.41])
x.add_row([3,'3*3','sigmoid','SGD', 11.35])
x.add_row([3,'3*3','tanh','SGD', 11.35])
x.add_row([3,'3*3','relu','SGD', 87.2])
x.add_row([3,'3*3','sigmoid','adam',11.35])
x.add_row([3,'3*3','tanh','adam', 11.35])
x.add_row([3,'3*3','relu','adam', 97.14])
x.add_row([3,'3*3','sigmoid','adagrad',11.35])
print(x)
```

Hiddden layers	kernel size	activation function	optimizer	accuracy
3	3*3	relu	adadelta	98.54
5	5*5	relu	adadelta	99.5
7	5*5	relu	adadelta	99.41
3	3*3	sigmoid	SGD	11.35
3	3*3	tanh	SGD	11.35
3	3*3	relu	SGD	87.2
3	3*3	sigmoid	adam	11.35
3	3*3	tanh	adam	11.35
3	3*3	relu	adam	97.14
3	3*3	sigmoid	adagrad	11.35

Conclusion:

1. 5 layers with relu activation and adadelta optimizer has better the accuracy.
2. Relu works better than all other activation functions
3. Adadelta is best optimizer.
4. Sigmoid and tanh activation function have bad performances.
5. SGD with relu performed betterthan SGD with other activation functions.