Assignment 13 CNN_MNIST

June 17, 2019

In [9]:
```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

%matplotlib inline
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
import matplotlib.pyplot as plt
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()



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)
```

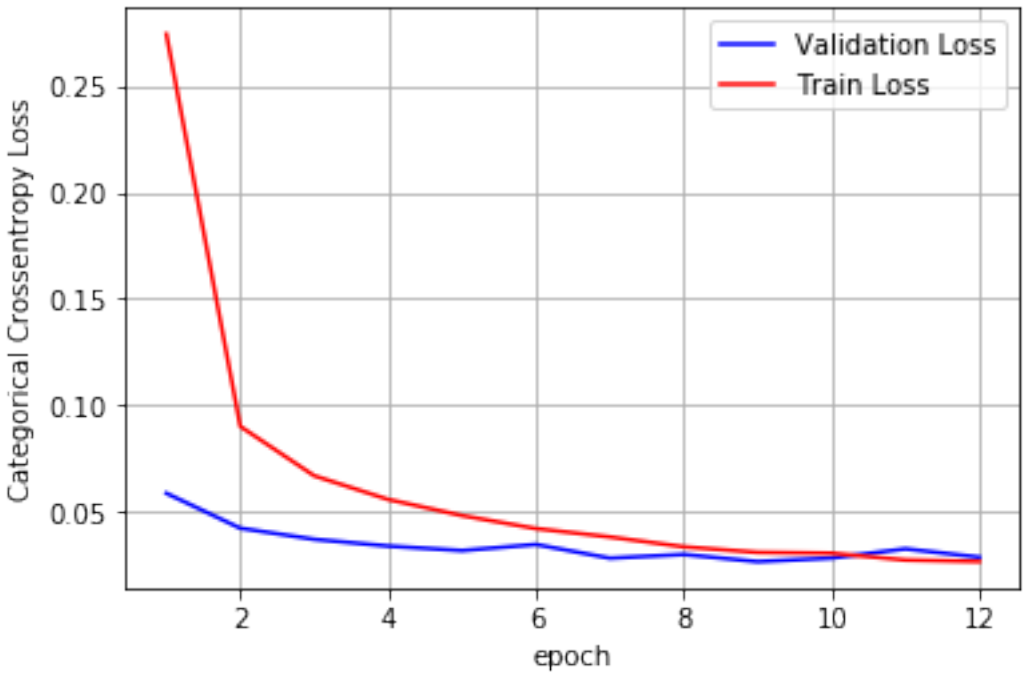
```
        x_train = x_train.astype('float32')
        x_test = x_test.astype('float32')
        x_train /= 255
        x_test /= 255
        print('x_train shape:', x_train.shape)
        print(x_train.shape[0], 'train samples')
        print(x_test.shape[0], 'test samples')

        # 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)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

```
In [10]:
         def model():
           model = Sequential()
           model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
           model.add(Conv2D(64, (3, 3), activation='relu'))
           model.add(MaxPooling2D(pool_size=(2, 2)))
           model.add(Dropout(0.25))
           model.add(Flatten())
           model.add(Dense(128, activation='relu'))
           model.add(Dropout(0.5))
           model.add(Dense(num_classes, activation='softmax'))
           model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
           graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
           score = model.evaluate(x_test, y_test, verbose=0)
           print('Test loss:', score[0])
           print('Test accuracy:', score[1])
           fig,ax = plt.subplots(1,1)
           ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
           x = list(range(1,epochs+1))
           vy = graph.history['val_loss']
           ty = graph.history['loss']
           plt_dynamic(x, vy, ty, ax)
         model()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.2742 - acc: 0.9153 - val
Epoch 2/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0902 - acc: 0.9738 - val
Epoch 3/12
```

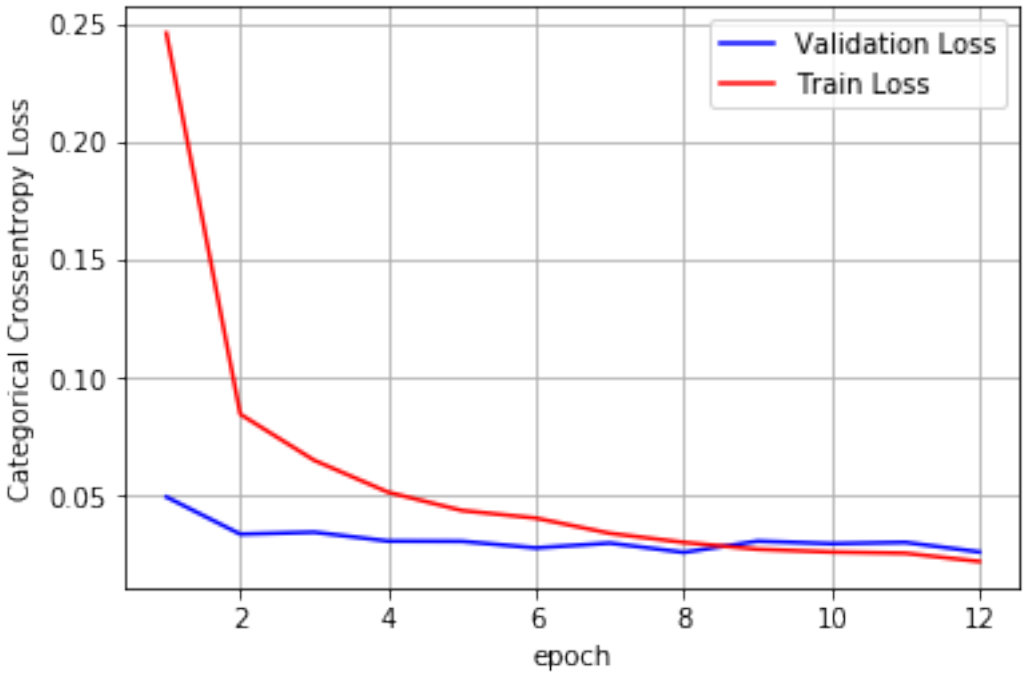
```
60000/60000 [==============================] - 5s 79us/step - loss: 0.0671 - acc: 0.9802 - val
Epoch 4/12
60000/60000 [==============================] - 5s 79us/step - loss: 0.0560 - acc: 0.9832 - val
Epoch 5/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0484 - acc: 0.9851 - val
Epoch 6/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0423 - acc: 0.9870 - val
Epoch 7/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0384 - acc: 0.9884 - val
Epoch 8/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0337 - acc: 0.9898 - val
Epoch 9/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0311 - acc: 0.9906 - val
Epoch 10/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0307 - acc: 0.9909 - val
Epoch 11/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0275 - acc: 0.9916 - val
Epoch 12/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0268 - acc: 0.9917 - val
Test loss: 0.028881798116410572
Test accuracy: 0.9911
```




```
In [14]: def model():
             model = Sequential()
```

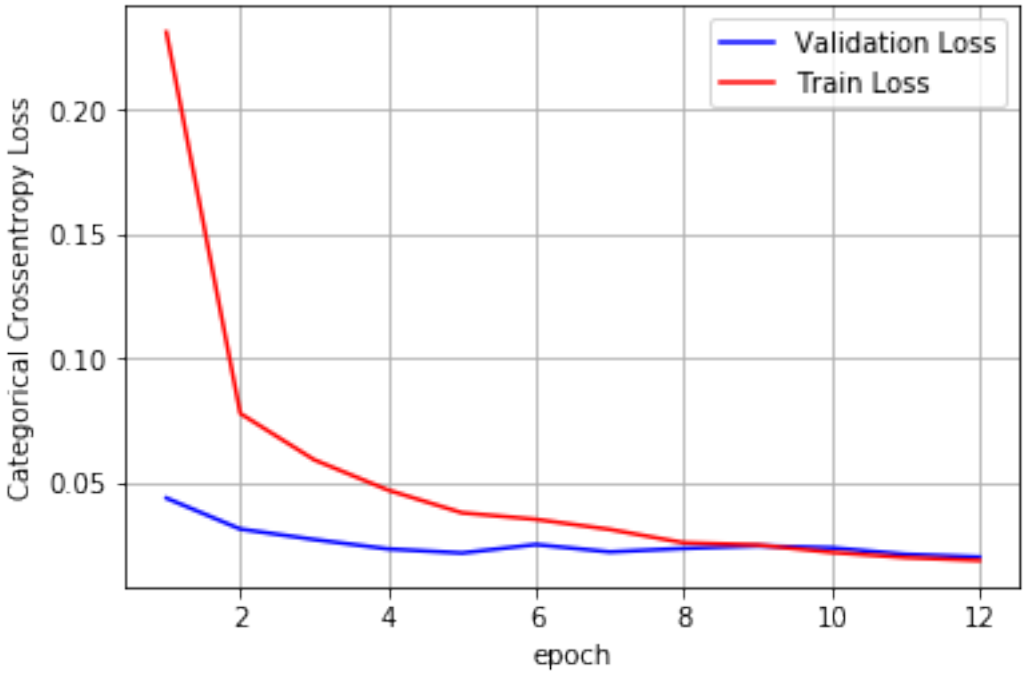
```
            model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
            model.add(Conv2D(64, (3, 3), activation='relu'))
            model.add(MaxPooling2D(pool_size=(2, 2)))
            model.add(Dropout(0.25))
            model.add(Flatten())
            model.add(Dense(128, activation='relu'))
            model.add(Dropout(0.5))
            model.add(Dense(num_classes, activation='softmax'))
            model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
            graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
            score = model.evaluate(x_test, y_test, verbose=0)
            print('Test loss:', score[0])
            print('Test accuracy:', score[1])
            fig,ax = plt.subplots(1,1)
            ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
            x = list(range(1,epochs+1))
            vy = graph.history['val_loss']
            ty = graph.history['loss']
            plt_dynamic(x, vy, ty, ax)
        model()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 6s 97us/step - loss: 0.2461 - acc: 0.9247 - val
Epoch 2/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0846 - acc: 0.9751 - val
Epoch 3/12
60000/60000 [==============================] - 5s 77us/step - loss: 0.0651 - acc: 0.9802 - val
Epoch 4/12
60000/60000 [==============================] - 5s 77us/step - loss: 0.0515 - acc: 0.9842 - val
Epoch 5/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0438 - acc: 0.9869 - val
Epoch 6/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0405 - acc: 0.9872 - val
Epoch 7/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0341 - acc: 0.9891 - val
Epoch 8/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0302 - acc: 0.9901 - val
Epoch 9/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0273 - acc: 0.9911 - val
Epoch 10/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0262 - acc: 0.9917 - val
Epoch 11/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0257 - acc: 0.9914 - val
Epoch 12/12
60000/60000 [==============================] - 5s 76us/step - loss: 0.0222 - acc: 0.9926 - val
Test loss: 0.026217569957539353
Test accuracy: 0.9927
```



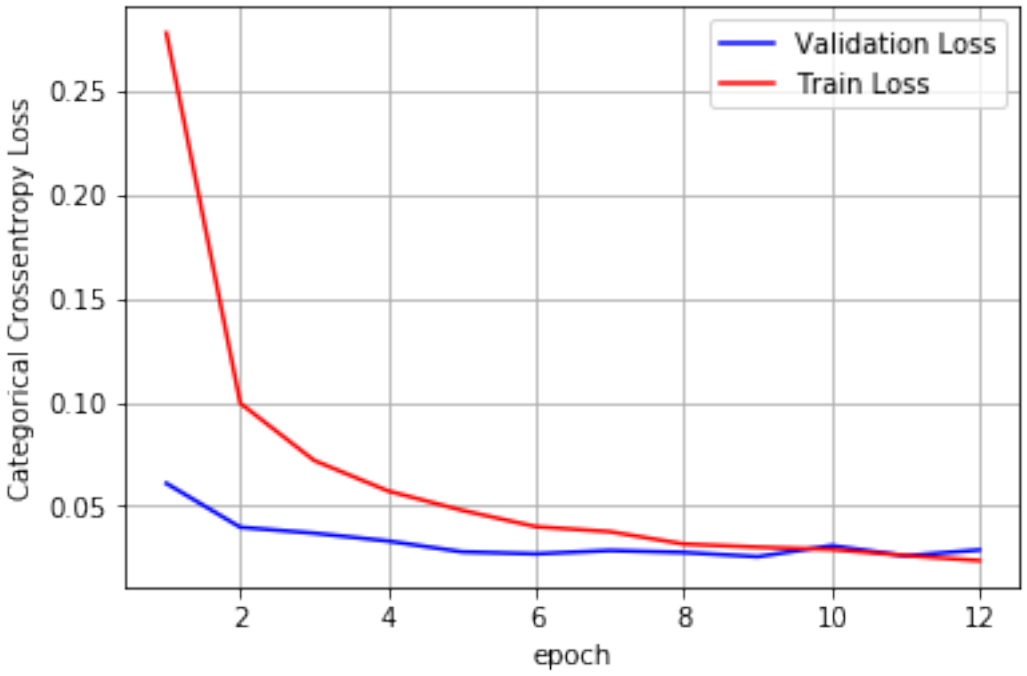

```
In [15]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (3, 3), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Conv2D(128, (3, 3), activation='relu'))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
             ty = graph.history['loss']
             plt_dynamic(x, vy, ty, ax)
         model()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 7s 120us/step - loss: 0.2312 - acc: 0.9293 - val
Epoch 2/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0777 - acc: 0.9779 - val_
Epoch 3/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0591 - acc: 0.9834 - val_
Epoch 4/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0469 - acc: 0.9863 - val_
Epoch 5/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0378 - acc: 0.9889 - val_
Epoch 6/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0351 - acc: 0.9890 - val_
Epoch 7/12
60000/60000 [==============================] - 6s 95us/step - loss: 0.0311 - acc: 0.9897 - val_
Epoch 8/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0257 - acc: 0.9918 - val_
Epoch 9/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0248 - acc: 0.9922 - val_
Epoch 10/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0218 - acc: 0.9926 - val_
Epoch 11/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0198 - acc: 0.9935 - val_
Epoch 12/12
60000/60000 [==============================] - 6s 94us/step - loss: 0.0186 - acc: 0.9942 - val_
Test loss: 0.019966480247871912
Test accuracy: 0.994
```



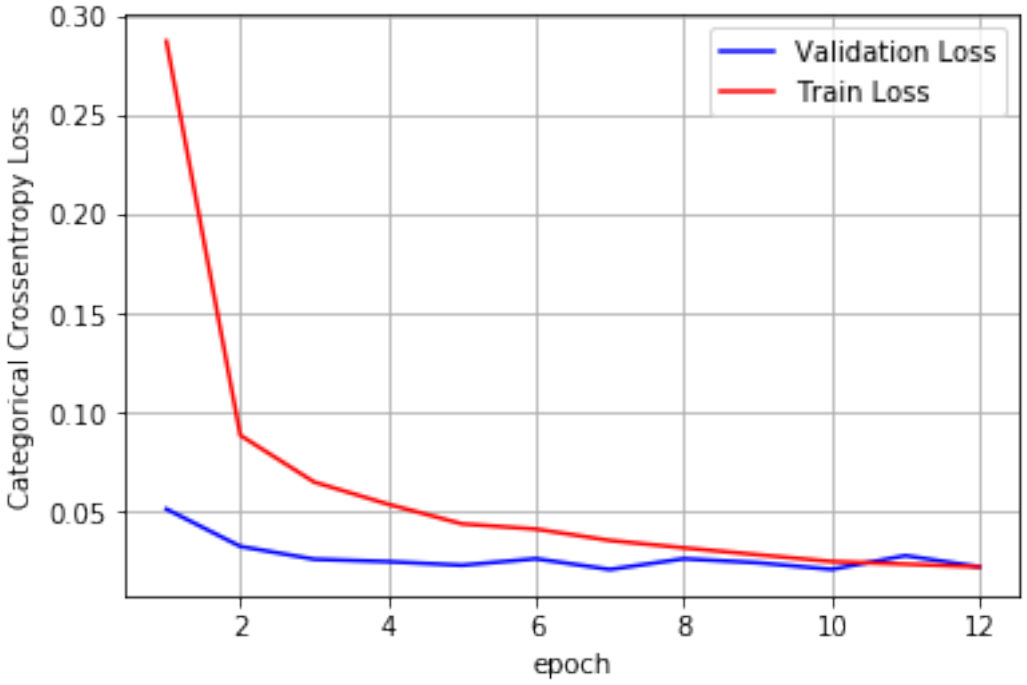

```
In [16]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(2,2),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (2, 2), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Conv2D(64, (2,2 ), activation='relu'))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
             ty = graph.history['loss']
             plt_dynamic(x, vy, ty, ax)
         model()

Train on 60000 samples, validate on 10000 samples
```

```
Epoch 1/12
60000/60000 [==============================] - 6s 108us/step - loss: 0.2776 - acc: 0.9161 - val
Epoch 2/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0995 - acc: 0.9702 - val_
Epoch 3/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0719 - acc: 0.9787 - val_
Epoch 4/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0572 - acc: 0.9825 - val_
Epoch 5/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0479 - acc: 0.9851 - val_
Epoch 6/12
60000/60000 [==============================] - 5s 82us/step - loss: 0.0401 - acc: 0.9871 - val_
Epoch 7/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0377 - acc: 0.9882 - val_
Epoch 8/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0317 - acc: 0.9900 - val_
Epoch 9/12
60000/60000 [==============================] - 5s 82us/step - loss: 0.0302 - acc: 0.9898 - val_
Epoch 10/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0291 - acc: 0.9905 - val_
Epoch 11/12
60000/60000 [==============================] - 5s 81us/step - loss: 0.0262 - acc: 0.9916 - val_
Epoch 12/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0237 - acc: 0.9921 - val_
Test loss: 0.028867433587322012
Test accuracy: 0.9913
```



```
In [17]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(2,2),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (2, 2), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Conv2D(128, (2,2 ), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Conv2D(128, (2,2 ), activation='relu'))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,val
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
             ty = graph.history['loss']
             plt_dynamic(x, vy, ty, ax)
         model()

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 7s 117us/step - loss: 0.2874 - acc: 0.9117 - val
Epoch 2/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0886 - acc: 0.9742 - val_
Epoch 3/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0651 - acc: 0.9809 - val_
Epoch 4/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0539 - acc: 0.9843 - val_
Epoch 5/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0440 - acc: 0.9873 - val_
Epoch 6/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0413 - acc: 0.9879 - val_
Epoch 7/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0356 - acc: 0.9897 - val_
Epoch 8/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0319 - acc: 0.9900 - val_
Epoch 9/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0284 - acc: 0.9912 - val_
```

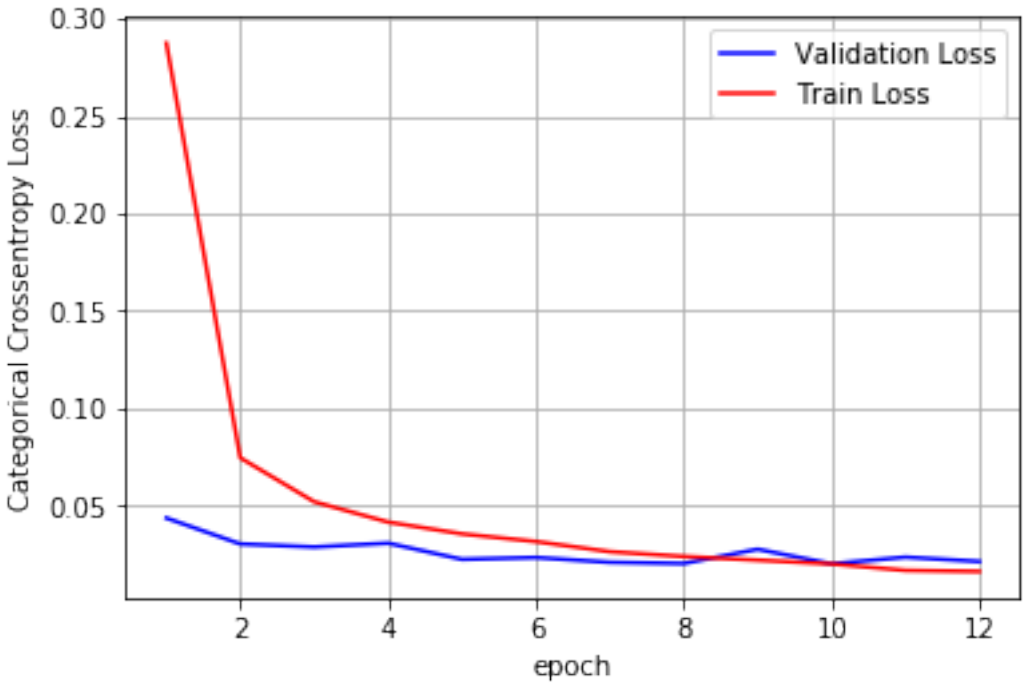
```
Epoch 10/12
60000/60000 [==============================] - 5s 86us/step - loss: 0.0250 - acc: 0.9923 - val
Epoch 11/12
60000/60000 [==============================] - 5s 86us/step - loss: 0.0237 - acc: 0.9927 - val
Epoch 12/12
60000/60000 [==============================] - 5s 86us/step - loss: 0.0223 - acc: 0.9931 - val
Test loss: 0.022352785166477634
Test accuracy: 0.9931
```




```
In [18]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(3,3),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (2, 2), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Conv2D(128, (3,3), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Conv2D(128, (3,3), activation='relu'))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
```

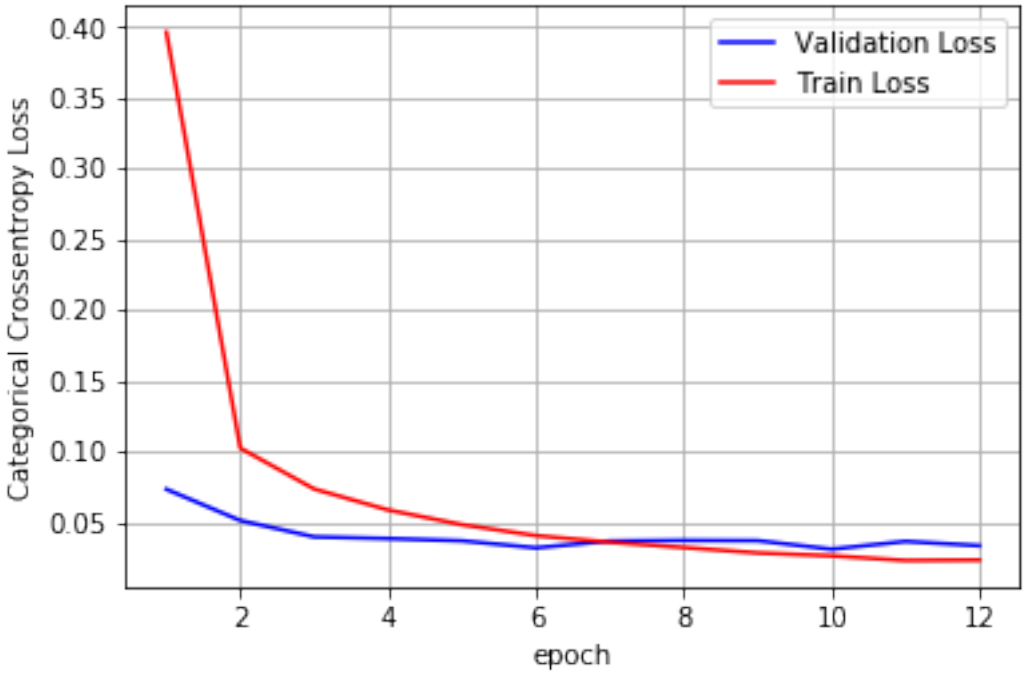
```
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])
    fig,ax = plt.subplots(1,1)
    ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
    x = list(range(1,epochs+1))
    vy = graph.history['val_loss']
    ty = graph.history['loss']
    plt_dynamic(x, vy, ty, ax)
model()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 7s 118us/step - loss: 0.2874 - acc: 0.9084 - val
Epoch 2/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0746 - acc: 0.9781 - val_
Epoch 3/12
60000/60000 [==============================] - 5s 86us/step - loss: 0.0520 - acc: 0.9847 - val_
Epoch 4/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0416 - acc: 0.9880 - val_
Epoch 5/12
60000/60000 [==============================] - 5s 86us/step - loss: 0.0355 - acc: 0.9898 - val_
Epoch 6/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0316 - acc: 0.9910 - val_
Epoch 7/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0264 - acc: 0.9919 - val_
Epoch 8/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0239 - acc: 0.9929 - val_
Epoch 9/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0221 - acc: 0.9935 - val_
Epoch 10/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0201 - acc: 0.9938 - val_
Epoch 11/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0168 - acc: 0.9949 - val_
Epoch 12/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0162 - acc: 0.9951 - val_
Test loss: 0.021370815263250187
Test accuracy: 0.9951
```




```
In [19]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(3,3),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (2, 2), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Conv2D(128, (3,3), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Conv2D(128, (3,3), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
             ty = graph.history['loss']
```

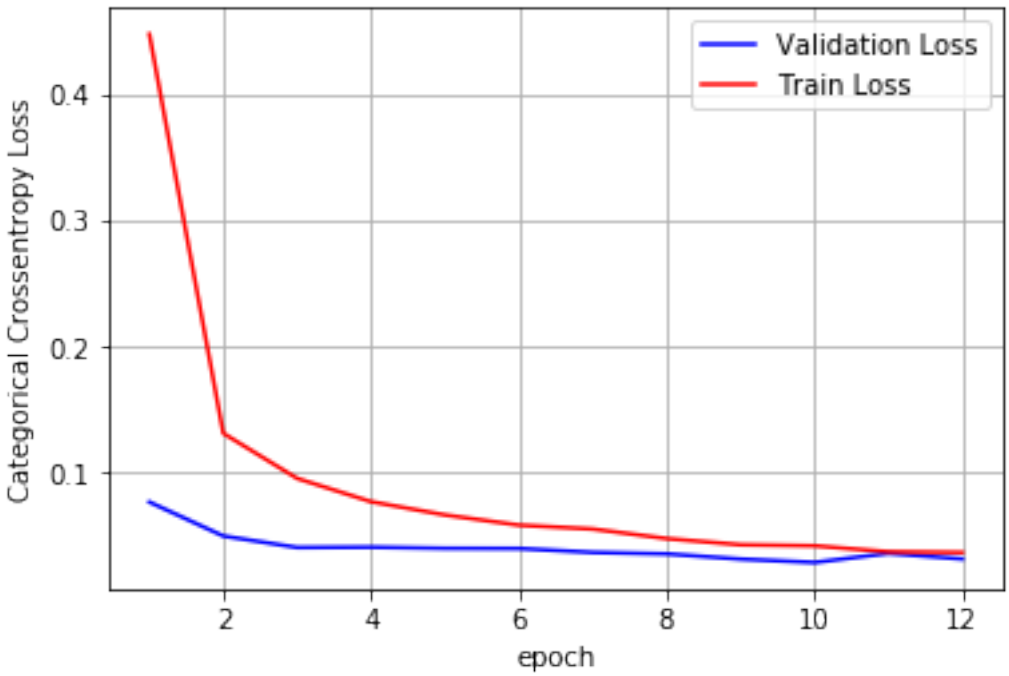
```
          plt_dynamic(x, vy, ty, ax)
        model()

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 7s 120us/step - loss: 0.3961 - acc: 0.8728 - val
Epoch 2/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.1023 - acc: 0.9708 - val_
Epoch 3/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0734 - acc: 0.9789 - val_
Epoch 4/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0586 - acc: 0.9826 - val_
Epoch 5/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0482 - acc: 0.9860 - val_
Epoch 6/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0405 - acc: 0.9889 - val_
Epoch 7/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0358 - acc: 0.9889 - val_
Epoch 8/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0321 - acc: 0.9908 - val_
Epoch 9/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0282 - acc: 0.9916 - val_
Epoch 10/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0262 - acc: 0.9919 - val_
Epoch 11/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0229 - acc: 0.9935 - val_
Epoch 12/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0232 - acc: 0.9931 - val_
Test loss: 0.033381988369911145
Test accuracy: 0.9917
```




```
In [20]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(3,3),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (2, 2), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Conv2D(128, (3,3), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Conv2D(128, (3,3), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.3))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
```

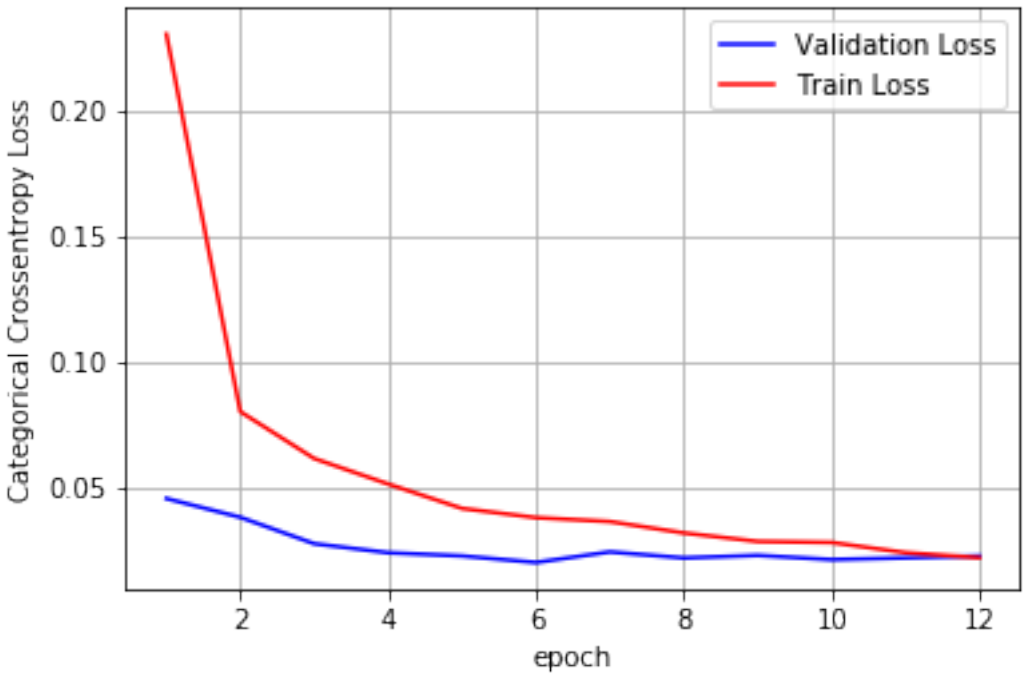
```
            ty = graph.history['loss']
            plt_dynamic(x, vy, ty, ax)
        model()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.4485 - acc: 0.8569 - val
Epoch 2/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.1312 - acc: 0.9625 - val_
Epoch 3/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0953 - acc: 0.9732 - val_
Epoch 4/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0768 - acc: 0.9786 - val_
Epoch 5/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0662 - acc: 0.9812 - val_
Epoch 6/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0582 - acc: 0.9833 - val_
Epoch 7/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0550 - acc: 0.9846 - val_
Epoch 8/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0474 - acc: 0.9864 - val_
Epoch 9/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0425 - acc: 0.9875 - val_
Epoch 10/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0417 - acc: 0.9883 - val_
Epoch 11/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0370 - acc: 0.9892 - val_
Epoch 12/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0364 - acc: 0.9893 - val_
Test loss: 0.03126877400011872
Test accuracy: 0.991
```



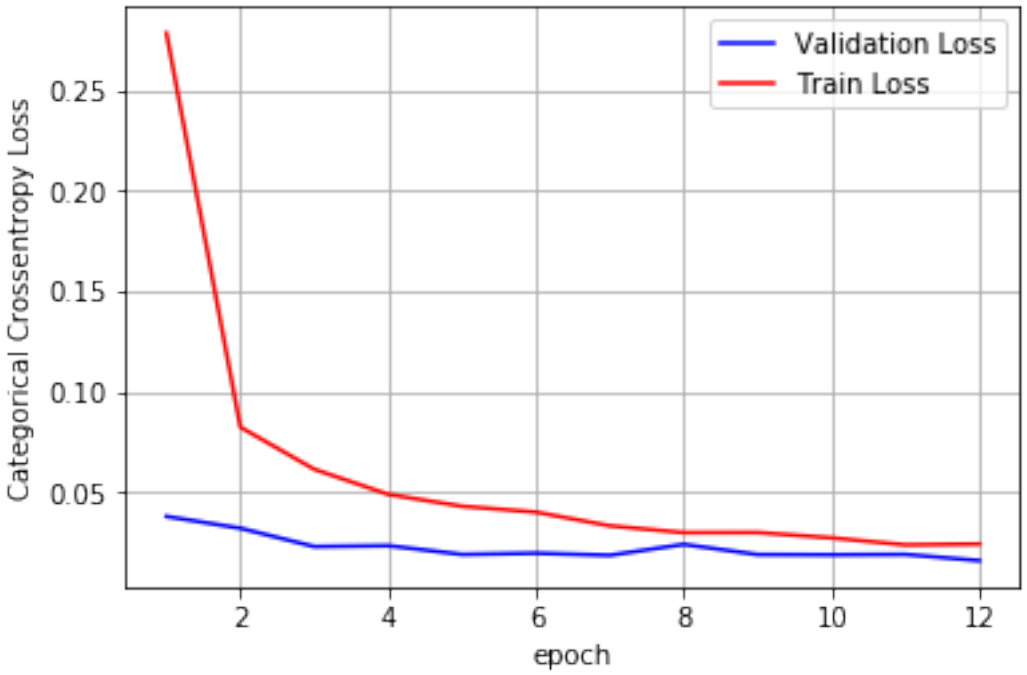

```
In [21]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(5,5),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (5, 5), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
             ty = graph.history['loss']
             plt_dynamic(x, vy, ty, ax)
         model()

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
```

```
60000/60000 [==============================] - 7s 119us/step - loss: 0.2304 - acc: 0.9296 - val
Epoch 2/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0802 - acc: 0.9770 - val
Epoch 3/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0615 - acc: 0.9822 - val
Epoch 4/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0514 - acc: 0.9845 - val
Epoch 5/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0417 - acc: 0.9871 - val
Epoch 6/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0380 - acc: 0.9884 - val
Epoch 7/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0365 - acc: 0.9892 - val
Epoch 8/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0319 - acc: 0.9903 - val
Epoch 9/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0285 - acc: 0.9911 - val
Epoch 10/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0281 - acc: 0.9913 - val
Epoch 11/12
60000/60000 [==============================] - 5s 80us/step - loss: 0.0241 - acc: 0.9923 - val
Epoch 12/12
60000/60000 [==============================] - 5s 79us/step - loss: 0.0221 - acc: 0.9928 - val
Test loss: 0.022752777716280052
Test accuracy: 0.9934
```



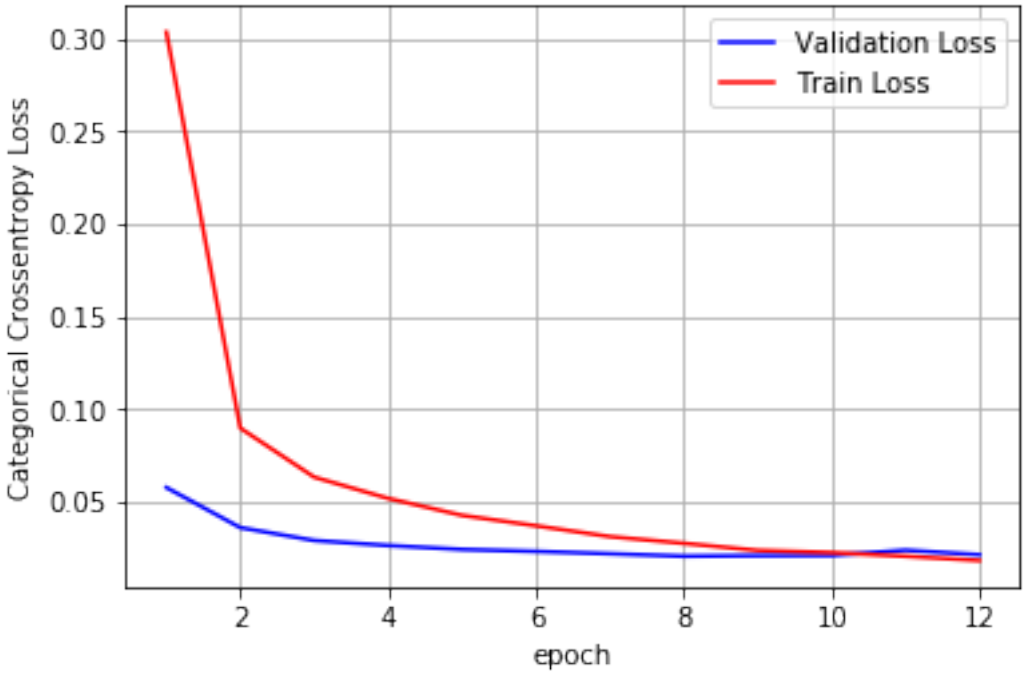
```
In [22]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(5,5),activation='relu',input_shape=input_shape))
             model.add(Conv2D(64, (5, 5), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.2))
             model.add(Conv2D(100, (5, 5), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Dropout(0.4))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,val
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
             ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
             x = list(range(1,epochs+1))
             vy = graph.history['val_loss']
             ty = graph.history['loss']
             plt_dynamic(x, vy, ty, ax)
         model()

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 8s 130us/step - loss: 0.2782 - acc: 0.9105 - val
Epoch 2/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0826 - acc: 0.9759 - val_
Epoch 3/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0616 - acc: 0.9823 - val_
Epoch 4/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0492 - acc: 0.9856 - val_
Epoch 5/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0433 - acc: 0.9868 - val_
Epoch 6/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0403 - acc: 0.9885 - val_
Epoch 7/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.0335 - acc: 0.9900 - val_
Epoch 8/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.0302 - acc: 0.9910 - val_
Epoch 9/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0302 - acc: 0.9910 - val_
Epoch 10/12
```

```
60000/60000 [==============================] - 5s 89us/step - loss: 0.0276 - acc: 0.9919 - val_
Epoch 11/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.0241 - acc: 0.9925 - val_
Epoch 12/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0245 - acc: 0.9923 - val_
Test loss: 0.016260585690253355
Test accuracy: 0.9949
```




```
In [30]: def model():
             model = Sequential()
             model.add(Conv2D(32, kernel_size=(5,5),activation='relu',input_shape=input_shape))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Conv2D(64, (5, 5), activation='relu'))
             model.add(MaxPooling2D(pool_size=(2, 2)))
             model.add(Flatten())
             model.add(Dense(128, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
             graph = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,va
             score = model.evaluate(x_test, y_test, verbose=0)
             print('Test loss:', score[0])
             print('Test accuracy:', score[1])
             fig,ax = plt.subplots(1,1)
```

```
            ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
            x = list(range(1,epochs+1))
            vy = graph.history['val_loss']
            ty = graph.history['loss']
            plt_dynamic(x, vy, ty, ax)
          model()

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 6s 95us/step - loss: 0.3038 - acc: 0.9070 - val
Epoch 2/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0896 - acc: 0.9741 - val
Epoch 3/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0630 - acc: 0.9810 - val
Epoch 4/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0515 - acc: 0.9848 - val
Epoch 5/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0424 - acc: 0.9876 - val
Epoch 6/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0367 - acc: 0.9888 - val
Epoch 7/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0309 - acc: 0.9906 - val
Epoch 8/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0272 - acc: 0.9916 - val
Epoch 9/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0234 - acc: 0.9927 - val
Epoch 10/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0222 - acc: 0.9930 - val
Epoch 11/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0202 - acc: 0.9935 - val
Epoch 12/12
60000/60000 [==============================] - 3s 50us/step - loss: 0.0180 - acc: 0.9942 - val
Test loss: 0.021075438146310035
Test accuracy: 0.9938
```



CONCLUSIONS:

1) Tried with various combinations of architectures and kernels sizes. Best accuracy was obtained with 3x3 kernel with accuracy of 0.994. There were other architectures with better accuracy but also had overfitting.
2) Learnt that there is no specific architecture for obtaining good accuracy. Trial and error has to be done. Since this was a small dataset, for large dataset the architecture might be different.
3) Dropout is needed to avoid overfitting.

```
In [32]: !pip install -q PTable
         from prettytable import PrettyTable

         x = PrettyTable()
         x.field_names = [ "Architecture", "Kernel Size","Overfitting oberserved", "Test Accura
         x.add_row(["conv->conv->m_pool->drop-..","3x3"," no ",0.9911])
         x.add_row(["conv->conv->m_pool->drop-..","3x3"," yes ",0.9927])
         x.add_row(["conv->conv->m_pool->drop->conv-..","3x3"," no ",0.9940])
         x.add_row(["conv->conv->m_pool->drop->conv-..","3x3"," yes ",0.9951])
         x.add_row(["conv->conv->m_pool->drop->conv->m_pool-..","3x3"," yes ",0.9917])
         x.add_row(["conv->conv->m_pool->drop->conv->m_pool-..","3x3"," no ",0.991])
         x.add_row(["conv->conv->m_pool->drop->conv-..","2x2"," no ",0.9913])
         x.add_row(["conv->conv->m_pool->drop->conv->m_pool->conv-..","2x2"," no ",0.9931])
         x.add_row(["conv->conv->m_pool->drop-..","5x5"," no ",0.9934])
         x.add_row(["conv->conv->m_pool->drop->conv->m_pool->drop-..","5x5"," yes ",0.9949])
         x.add_row(["conv->m_pool->conv->m_pool-..","5x5"," no ",0.9938])
         print(x)
```

```
  Building wheel for PTable (setup.py) ... done
```

Architecture	Kernel Size	Overfitting oberserved	Tes
conv->conv->m_pool->drop-..	3x3	no	
conv->conv->m_pool->drop-..	3x3	yes	
conv->conv->m_pool->drop->conv-..	3x3	no	
conv->conv->m_pool->drop->conv-..	3x3	yes	
conv->conv->m_pool->drop->conv->m_pool-..	3x3	yes	
conv->conv->m_pool->drop->conv->m_pool-..	3x3	no	
conv->conv->m_pool->drop->conv-..	2x2	no	
conv->conv->m_pool->drop->conv->m_pool->conv-..	2x2	no	
conv->conv->m_pool->drop-..	5x5	no	
conv->conv->m_pool->drop->conv->m_pool->drop-..	5x5	yes	
conv->m_pool->conv->m_pool-..	5x5	no	

1)Here after " .. " the complete layer is -> flat -> dense -> dropout -> dense. 2) Overfitting is observed as 'yes' but it is subjective compared to other models.