In [0]:

```
```

In [3]:

```
# if you keras is not using tensorflow as backend set "KERAS_BACKEND=tensorflow" use this command
from keras.utils import np_utils
from keras.datasets import mnist
import seaborn as sns
from keras.initializers import RandomNormal
from keras.models import Sequential
from keras.layers import Dense, Activation
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info.

In [0]:

```
%matplotlib notebook
%matplotlib inline
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
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()
```

In [0]:

```
```

In [5]:

```
# 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

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)

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.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:66: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_de
fault_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder
instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:4432: The name tf.random_uniform is deprecated. Please use tf.random.uniform in
stead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:4267: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instea
d.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:148: The name tf.placeholder_with_default is deprecated. Please use tf.compat.v
1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:3733: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is dep
recated and will be removed in a future version.
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:793: T
```

```
he name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead
.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:3576: The name tf.log is deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops
/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will b
e removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:1033: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add
instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:1020: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:3005: The name tf.Session is deprecated. Please use tf.compat.v1.Session instea
d.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:190: The name tf.get_default_session is deprecated. Please use tf.compat.v1.get
_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:197: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto
instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:207: The name tf.global_variables is deprecated. Please use tf.compat.v1.global
_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:216: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1
.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:223: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.v
ariables_initializer instead.

60000/60000 [==============================] - 139s 2ms/step - loss: 0.2830 - acc: 0.9140
- val_loss: 0.0567 - val_acc: 0.9820
Epoch 2/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0944 - acc: 0.9720
- val_loss: 0.0442 - val_acc: 0.9861
Epoch 3/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0696 - acc: 0.9791
- val_loss: 0.0391 - val_acc: 0.9872
Epoch 4/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0592 - acc: 0.9826
- val_loss: 0.0348 - val_acc: 0.9887
Epoch 5/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0501 - acc: 0.9847
- val_loss: 0.0283 - val_acc: 0.9900
Epoch 6/12
60000/60000 [==============================] - 137s 2ms/step - loss: 0.0454 - acc: 0.9862
- val_loss: 0.0301 - val_acc: 0.9899
Epoch 7/12
60000/60000 [==============================] - 137s 2ms/step - loss: 0.0420 - acc: 0.9873
- val_loss: 0.0272 - val_acc: 0.9918
Epoch 8/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.0375 - acc: 0.9884
- val_loss: 0.0288 - val_acc: 0.9911
Epoch 9/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.0355 - acc: 0.9890
- val_loss: 0.0311 - val_acc: 0.9901
Epoch 10/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.0338 - acc: 0.9896
```

```
- val_loss: 0.0281 - val_acc: 0.9908
Epoch 11/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.0300 - acc: 0.9906
- val_loss: 0.0254 - val_acc: 0.9919
Epoch 12/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.0282 - acc: 0.9911
- val_loss: 0.0270 - val_acc: 0.9915
Test loss: 0.027048864217419758
Test accuracy: 0.9915
```

In [0]:

```

```

In [0]:

```

```

In [0]:

```

```

3-Layered CNN

In [0]:

```
nb_epoch=epochs
```

In [7]:

```
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(Conv2D(128, (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.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 12, 12, 64)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 10, 10, 128)       73856     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 128)         0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 5, 5, 128)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 3200)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               409728    
```

```
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290      
=================================================================
Total params: 503,690
Trainable params: 503,690
Non-trainable params: 0
_________________________________________________________________
```

In [8]:

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
history=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

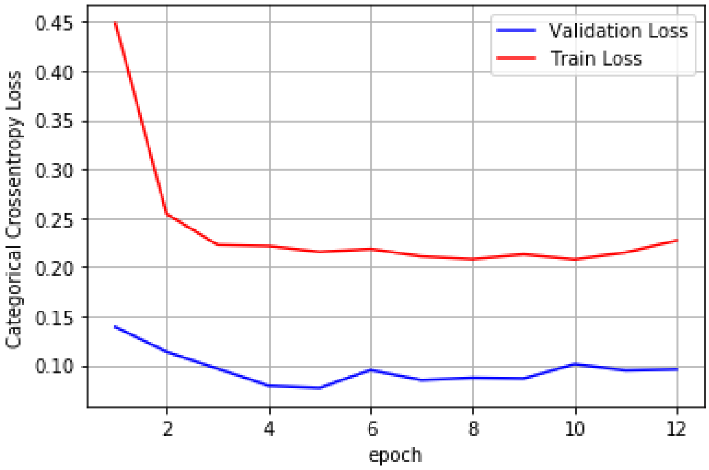
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.4483 - acc: 0.8660
- val_loss: 0.1388 - val_acc: 0.9587
Epoch 2/12
60000/60000 [==============================] - 192s 3ms/step - loss: 0.2542 - acc: 0.9321
- val_loss: 0.1136 - val_acc: 0.9658
Epoch 3/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.2226 - acc: 0.9436
- val_loss: 0.0964 - val_acc: 0.9704
Epoch 4/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.2214 - acc: 0.9461
- val_loss: 0.0792 - val_acc: 0.9772
Epoch 5/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.2153 - acc: 0.9491
- val_loss: 0.0769 - val_acc: 0.9777
Epoch 6/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.2184 - acc: 0.9508
- val_loss: 0.0950 - val_acc: 0.9765
Epoch 7/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.2109 - acc: 0.9516
- val_loss: 0.0847 - val_acc: 0.9800
Epoch 8/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.2081 - acc: 0.9544
- val_loss: 0.0870 - val_acc: 0.9803
Epoch 9/12
60000/60000 [==============================] - 192s 3ms/step - loss: 0.2129 - acc: 0.9545
- val_loss: 0.0862 - val_acc: 0.9782
Epoch 10/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.2078 - acc: 0.9551
- val_loss: 0.1011 - val_acc: 0.9774
Epoch 11/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.2148 - acc: 0.9542
- val_loss: 0.0946 - val_acc: 0.9796
Epoch 12/12
60000/60000 [==============================] - 198s 3ms/step - loss: 0.2270 - acc: 0.9541
- val_loss: 0.0958 - val_acc: 0.9757
Test loss: 0.09575681054808083
Test accuracy: 0.9757
```

In [9]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[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,nb_epoch+1))

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

```
Test score: 0.09575681054808083
Test accuracy: 0.9757
```



In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

```
from keras.layers.normalization import BatchNormalization
```

5-Layered CNN

In [11]:

```
model1=Sequential() # Initializing the model

# First ConvNet
model1.add(Conv2D(32,kernel_size=(5,5),
                    activation='relu',
                    padding='same',
                    input_shape=input_shape))

model1.add(Conv2D(64,kernel_size=(5,5),
                    padding='same',
                    activation='relu'))#Second Convnet
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(128,kernel_size=(5,5),
                    padding='same',
                   activation='relu'))   # 3rd ConvNet
#maxpooling by (2,2 ) ,dropout,flattening
```

```
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(256,kernel_size=(5,5),
                    padding='same',
                    activation='relu'))#fourth Convnet
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.add(Dropout(0.25))
model1.add(Conv2D(512,kernel_size=(5,5),
                    padding='same',
                    activation='relu'))#fifth Convnet
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.add(Dropout(0.25))
model1.add(Flatten())

#hidden_layer
model1.add(Dense(256,
                   activation='relu',
                   kernel_initializer=RandomNormal(mean=0.0, stddev=0.55, seed=None)))
model1.add(BatchNormalization())
model1.add(Dropout(0.5))
model1.add(Dense(num_classes,activation='softmax'))
print(model1.summary())
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_b
ackend.py:4409: The name tf.random_normal is deprecated. Please use tf.random.normal inst
ead.

Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 28, 28, 32)        832       
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 28, 28, 64)        51264     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 64)        0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 14, 14, 128)       204928    
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 128)         0         
_________________________________________________________________
dropout_7 (Dropout)          (None, 7, 7, 128)         0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 7, 7, 256)         819456    
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 3, 3, 256)         0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 3, 3, 256)         0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 512)         3277312   
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 1, 1, 512)         0         
_________________________________________________________________
dropout_9 (Dropout)          (None, 1, 1, 512)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 256)               131328    
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024      
_________________________________________________________________
dropout_10 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 10)                2570      
=================================================================
Total params: 4,488,714
Trainable params: 4,488,202
Non-trainable params: 512
_________________________________________________________________
```

```
None
```

In [0]:

```
nb_epoch=epochs
```

In [13]:

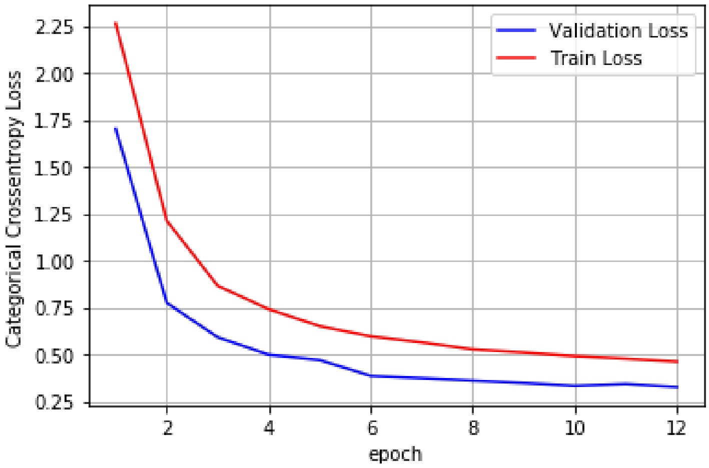
```
model1.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
history=model1.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model1.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

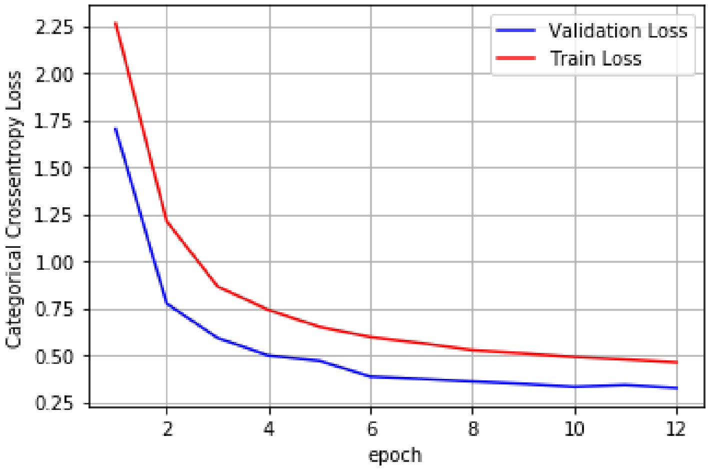
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 1603s 27ms/step - loss: 2.2639 - acc: 0.23
80 - val_loss: 1.7023 - val_acc: 0.3878
Epoch 2/12
59904/60000 [============================>.] - ETA: 2s - loss: 1.2122 - acc: 0.5656Train
on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 1603s 27ms/step - loss: 2.2639 - acc: 0.23
80 - val_loss: 1.7023 - val_acc: 0.3878
Epoch 2/12
60000/60000 [==============================] - 1613s 27ms/step - loss: 1.2117 - acc: 0.56
59 - val_loss: 0.7741 - val_acc: 0.7540
60000/60000 [==============================] - 1613s 27ms/step - loss: 1.2117 - acc: 0.56
59 - val_loss: 0.7741 - val_acc: 0.7540
Epoch 3/12
Epoch 3/12
60000/60000 [==============================] - 1629s 27ms/step - loss: 0.8645 - acc: 0.70
47 - val_loss: 0.5902 - val_acc: 0.7989
60000/60000 [==============================] - 1629s 27ms/step - loss: 0.8645 - acc: 0.70
47 - val_loss: 0.5902 - val_acc: 0.7989
Epoch 4/12
Epoch 4/12
60000/60000 [==============================] - 1633s 27ms/step - loss: 0.7399 - acc: 0.75
10 - val_loss: 0.4974 - val_acc: 0.8326
60000/60000 [==============================] - 1633s 27ms/step - loss: 0.7399 - acc: 0.75
10 - val_loss: 0.4974 - val_acc: 0.8326
Epoch 5/12
Epoch 5/12
60000/60000 [==============================] - 1635s 27ms/step - loss: 0.6501 - acc: 0.78
29 - val_loss: 0.4692 - val_acc: 0.8447
60000/60000 [==============================] - 1635s 27ms/step - loss: 0.6501 - acc: 0.78
29 - val_loss: 0.4692 - val_acc: 0.8447
Epoch 6/12
Epoch 6/12
60000/60000 [==============================] - 1658s 28ms/step - loss: 0.5952 - acc: 0.80
59 - val_loss: 0.3848 - val_acc: 0.8728
60000/60000 [==============================] - 1658s 28ms/step - loss: 0.5952 - acc: 0.80
59 - val_loss: 0.3848 - val_acc: 0.8728
Epoch 7/12
Epoch 7/12
60000/60000 [==============================] - 1643s 27ms/step - loss: 0.5630 - acc: 0.81
68 - val_loss: 0.3722 - val_acc: 0.8774
60000/60000 [==============================] - 1643s 27ms/step - loss: 0.5630 - acc: 0.81
68 - val_loss: 0.3722 - val_acc: 0.8774
Epoch 8/12
Epoch 8/12
60000/60000 [==============================] - 1629s 27ms/step - loss: 0.5257 - acc: 0.82
92 - val_loss: 0.3591 - val_acc: 0.8803
60000/60000 [==============================] - 1629s 27ms/step - loss: 0.5257 - acc: 0.82
92 - val_loss: 0.3591 - val_acc: 0.8803
Epoch 9/12
Epoch 9/12
60000/60000 [==============================] - 1640s 27ms/step - loss: 0.5093 - acc: 0.83
52 - val_loss: 0.3466 - val_acc: 0.8874
60000/60000 [==============================] - 1640s 27ms/step - loss: 0.5093 - acc: 0.83
52 - val_loss: 0.3466 - val_acc: 0.8874
Epoch 10/12
Epoch 10/12
60000/60000 [==============================] - 1642s 27ms/step - loss: 0.4899 - acc: 0.84
25 - val_loss: 0.3310 - val_acc: 0.8908
60000/60000 [==============================] - 1642s 27ms/step - loss: 0.4899 - acc: 0.84
25 - val_loss: 0.3310 - val_acc: 0.8908
Epoch 11/12
Epoch 11/12
60000/60000 [==============================] - 1637s 27ms/step - loss: 0.4764 - acc: 0.84
87 - val_loss: 0.3410 - val_acc: 0.8884
60000/60000 [==============================] - 1637s 27ms/step - loss: 0.4764 - acc: 0.84
87 - val_loss: 0.3410 - val_acc: 0.8884
Epoch 12/12
Epoch 12/12
60000/60000 [==============================] - 1646s 27ms/step - loss: 0.4612 - acc: 0.85
22 - val_loss: 0.3250 - val_acc: 0.8912
60000/60000 [==============================] - 1646s 27ms/step - loss: 0.4612 - acc: 0.85
22 - val_loss: 0.3250 - val_acc: 0.8912
Test loss: 0.32503178759813306
Test accuracy: 0.8912
Test loss: 0.32503178759813306
Test accuracy: 0.8912
```

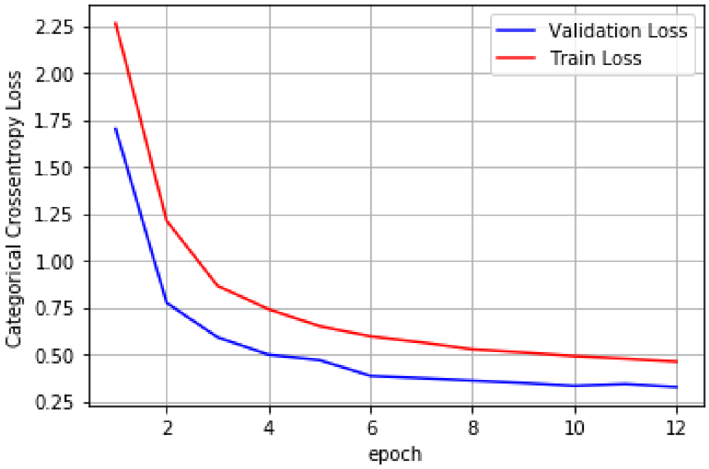
In [14]:

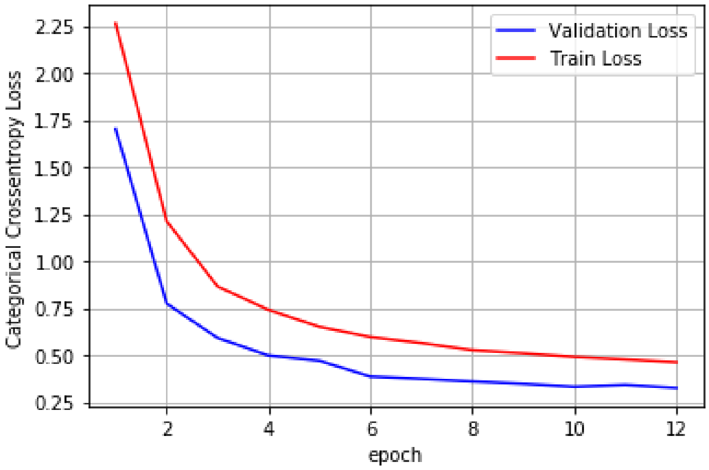
```
score = model1.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

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

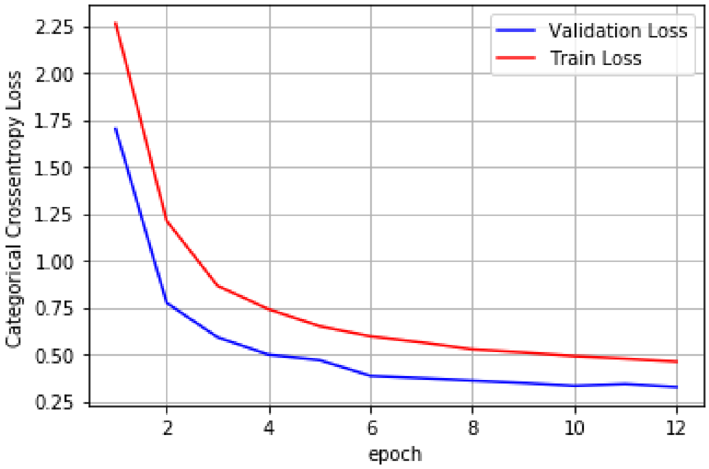
```
Test score: 0.32503178759813306
Test accuracy: 0.8912
Test score: 0.32503178759813306
Test accuracy: 0.8912
Test score: 0.32503178759813306
Test accuracy: 0.8912
Test score: 0.32503178759813306
Test accuracy: 0.8912
Test score: 0.32503178759813306
Test accuracy: 0.8912
Test score: 0.32503178759813306
Test accuracy: 0.8912
```

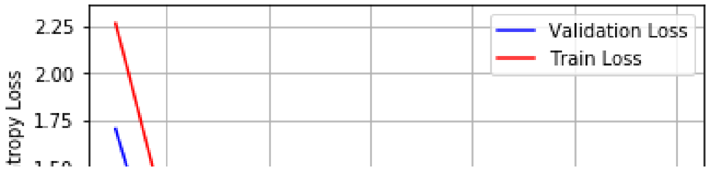


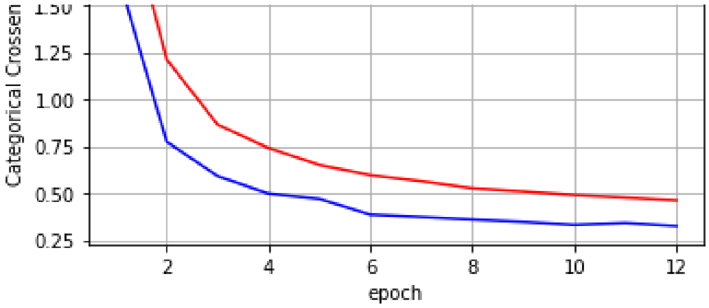

Validation Loss
Train Loss
Categorical Crossentropy Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
2
4
6
8
10
12
epoch


Validation Loss
Train Loss
Categorical Crossentropy Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
2
4
6
8
10
12
epoch


Validation Loss
Train Loss
Categorical Crossentropy Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
2
4
6
8
10
12
epoch


Validation Loss
Train Loss
Categorical Crossentropy Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
2
4
6
8
10
12
epoch


Validation Loss
Train Loss
tropy Loss
2.25
2.00
1.75



In [0]:

```
```

In [0]:

```
```

In [0]:

```
from keras.initializers import he_normal
```

7-Layered CNN

In [20]:

```
model7=Sequential() # Initializing the model

# First ConvNet
model7.add(Conv2D(16,kernel_size=(2,2),
                    activation='relu',
                    padding='same',strides=(1,1),
                    input_shape=input_shape))


model7.add(Conv2D(32,kernel_size=(2,2),
                    padding='same',strides=(2,2),
                    activation='relu'))#Second Convnet
#model7.add(MaxPooling2D(pool_size=(2,2)))
#model7.add(Dropout(0.25))

model7.add(Conv2D(64,kernel_size=(2,2),
                    padding='same',
                   activation='relu'))  # 3rd ConvNet
#maxpooling by  (2,2 ) ,dropout,flattening
#model7.add(MaxPooling2D(pool_size=(2,2)))
model7.add(Dropout(0.15))

model7.add(Conv2D(96,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#fourth Convnet
model7.add(MaxPooling2D(pool_size=(2,2)))
model7.add(Dropout(0.39))
model7.add(Conv2D(128,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#fifth Convnet
model7.add(MaxPooling2D(pool_size=(2,2)))
model7.add(Dropout(0.3))
model7.add(Conv2D(164,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#sixth Convnet
model7.add(Conv2D(164,kernel_size=(2,2),
                    padding='same',strides=(1,1),
                    activation='relu'))#seventh Convnet

model7.add(MaxPooling2D(pool_size=(2,2)))
model7.add(Dropout(0.4))
model7.add(Flatten())
```

```
#hidden_layer
model7.add(Dense(256,
                   activation='relu',
                   kernel_initializer=RandomNormal(mean=0.0, stddev=0.55, seed=None)))#1
hidden layer
model7.add(BatchNormalization())
model7.add(Dropout(0.5))
model7.add(Dense(148,
                   activation='relu',
                   kernel_initializer=RandomNormal(mean=0.0, stddev=0.4, seed=None)))#2
hidden layer
model7.add(BatchNormalization())
model7.add(Dropout(0.5))
model7.add(Dense(128,
                   activation='relu',
                  kernel_initializer=RandomNormal(mean=0.0, stddev=0.58, seed=None)))#3
hidden layer
model7.add(BatchNormalization())
model7.add(Dropout(0.5))
model7.add(Dense(num_classes,activation='softmax'))
print(model7.summary())
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_25 (Conv2D)           (None, 28, 28, 16)        80        
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 14, 14, 32)        2080      
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 14, 14, 64)        8256      
_________________________________________________________________
dropout_22 (Dropout)         (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 14, 14, 96)        24672     
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 7, 7, 96)          0         
_________________________________________________________________
dropout_23 (Dropout)         (None, 7, 7, 96)          0         
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 7, 7, 128)         49280     
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 3, 3, 128)         0         
_________________________________________________________________
dropout_24 (Dropout)         (None, 3, 3, 128)         0         
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 3, 3, 164)         84132     
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 3, 3, 164)         107748    
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 1, 1, 164)         0         
_________________________________________________________________
dropout_25 (Dropout)         (None, 1, 1, 164)         0         
_________________________________________________________________
flatten_6 (Flatten)          (None, 164)               0         
_________________________________________________________________
dense_11 (Dense)             (None, 256)               42240     
_________________________________________________________________
batch_normalization_5 (Batch (None, 256)               1024      
_________________________________________________________________
dropout_26 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_12 (Dense)             (None, 148)               38036     
_________________________________________________________________
batch_normalization_6 (Batch (None, 148)               592       
_________________________________________________________________
dropout_27 (Dropout)         (None, 148)               0         
_________________________________________________________________
dense_13 (Dense)             (None, 128)               19072     
_________________________________________________________________
batch_normalization_7 (Batch (None, 128)               512       
_________________________________________________________________
dropout_28 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1290
=================================================================
Total params: 379,014
Trainable params: 377,950
Non-trainable params: 1,064
_________________________________________________________________
None
```

In [21]:

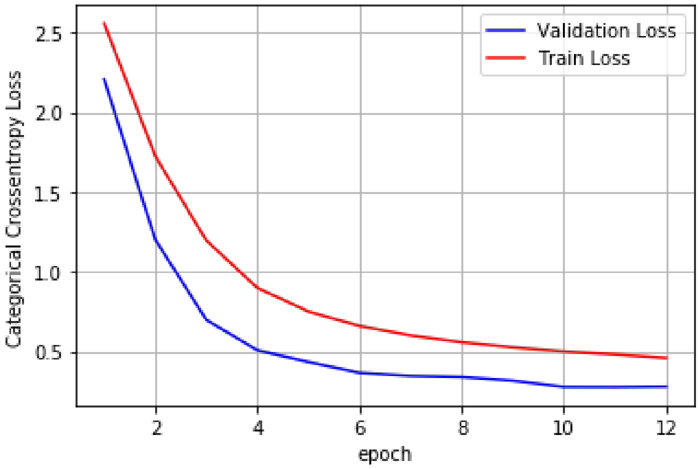
```
model7.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
history=model7.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 139s 2ms/step - loss: 2.5590 - acc: 0.1444
- val_loss: 2.2081 - val_acc: 0.1513
Epoch 2/12
60000/60000 [==============================] - 136s 2ms/step - loss: 1.7260 - acc: 0.3351
- val_loss: 1.2023 - val_acc: 0.5579
Epoch 3/12
60000/60000 [==============================] - 138s 2ms/step - loss: 1.1965 - acc: 0.5705
- val_loss: 0.6982 - val_acc: 0.7830
Epoch 4/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.8984 - acc: 0.7021
- val_loss: 0.5069 - val_acc: 0.8267
Epoch 5/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.7502 - acc: 0.7632
- val_loss: 0.4341 - val_acc: 0.8462
Epoch 6/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.6602 - acc: 0.7950
- val_loss: 0.3656 - val_acc: 0.8845
Epoch 7/12
60000/60000 [==============================] - 134s 2ms/step - loss: 0.6003 - acc: 0.8143
- val_loss: 0.3462 - val_acc: 0.8924
Epoch 8/12
60000/60000 [==============================] - 133s 2ms/step - loss: 0.5575 - acc: 0.8325
- val_loss: 0.3393 - val_acc: 0.9020
Epoch 9/12
60000/60000 [==============================] - 134s 2ms/step - loss: 0.5257 - acc: 0.8436
- val_loss: 0.3161 - val_acc: 0.9037
Epoch 10/12
60000/60000 [==============================] - 136s 2ms/step - loss: 0.4997 - acc: 0.8526
- val_loss: 0.2770 - val_acc: 0.9187
Epoch 11/12
60000/60000 [==============================] - 138s 2ms/step - loss: 0.4810 - acc: 0.8593
- val_loss: 0.2763 - val_acc: 0.9181
Epoch 12/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.4590 - acc: 0.8672
- val_loss: 0.2790 - val_acc: 0.9163
Test loss: 0.09575681054808083
Test accuracy: 0.9757
```

In [23]:

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

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

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



In [0]:

Conclusion

In [0]:

```
from prettytable import PrettyTable
```

In [27]:

```
x=PrettyTable()

x.field_names=(['No of layers','Test Score','Accuracy'])

x.add_row(['3-Layered CNN',0.095,0.975])
x.add_row(['5-Layered CNN',0.3,0.891])
x.add_row(['7-Layered CNN',0.09,0.975])

print(x)
```

```
+---------------+------------+----------+
|  No of layers | Test Score | Accuracy |
+---------------+------------+----------+
| 3-Layered CNN |   0.095    |  0.975   |
| 5-Layered CNN |    0.3     |  0.891   |
| 7-Layered CNN |    0.09    |  0.975   |
+---------------+------------+----------+
```

We can conclude that by using Convolution Neural Network, the accuracy by using 3 layered and 7 layered shown well comparedd to 5-Layered CNN

In [0]:

In [0]:

In [0]: