In [0]:

```
from keras.utils import np_utils
from keras.datasets import mnist
from keras.initializers import RandomNormal
```

```
Using TensorFlow backend.
```

In [0]:

```
import matplotlib.pyplot as plt
import numpy as np
import time
import pandas as pd
import seaborn as sns

import keras
from keras.models import Sequential
from keras.layers import Dense, Activation,Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

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

In [0]:

```
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]:

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

img_rows = 28
img_cols = 28

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)
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
```

In [0]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

x_train = x_train/255
x_test = x_test/255

y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
```

In [0]:

```
batch_size = 128
batch_size = 128
num_classes = 10
epochs = 12
```

3 Convolutional Layers

In [0]:

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),padding='same',activation='relu',input_shape=input_shape))
model.add(Conv2D(64,kernel_size = (5,5),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

model.add(Conv2D(32, kernel_size=(3, 3),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (5,5),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(Conv2D(32, kernel_size=(3, 3),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (5,5), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(4,4),padding='same'))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_36 (Conv2D)           (None, 28, 28, 32)        320       
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 28, 28, 64)        51264     
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 14, 14, 32)        18464     
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 14, 14, 64)        51264     
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 5, 5, 32)          18464     
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 5, 5, 64)          51264     
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 2, 2, 64)          0         
_________________________________________________________________
dropout_14 (Dropout)         (None, 2, 2, 64)          0         
_________________________________________________________________
flatten_6 (Flatten)          (None, 256)               0         
_________________________________________________________________
dense_14 (Dense)             (None, 256)               65792     
_________________________________________________________________
batch_normalization_7 (Batch (None, 256)               1024      
_________________________________________________________________
dropout_15 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_15 (Dense)             (None, 128)               32896     
_________________________________________________________________
batch_normalization_8 (Batch (None, 128)               512       
_________________________________________________________________
dropout_16 (Dropout)         (None, 128)               0         
_________________________________________________________________
```

```
dense_16 (Dense)             (None, 10)                1290
=================================================================
Total params: 292,554
Trainable params: 291,786
Non-trainable params: 768
_________________________________________________________________
```

In [0]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_t
est, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 672s 11ms/step - loss: 0.6168 - acc: 0.8232 - val_loss:
0.0631 - val_acc: 0.9816
Epoch 2/12
60000/60000 [==============================] - 663s 11ms/step - loss: 0.0885 - acc: 0.9766 - val_loss:
0.0367 - val_acc: 0.9897
Epoch 3/12
60000/60000 [==============================] - 662s 11ms/step - loss: 0.0605 - acc: 0.9844 - val_loss:
0.0385 - val_acc: 0.9898
Epoch 4/12
60000/60000 [==============================] - 659s 11ms/step - loss: 0.0491 - acc: 0.9868 - val_loss:
0.0263 - val_acc: 0.9932
Epoch 5/12
60000/60000 [==============================] - 659s 11ms/step - loss: 0.0422 - acc: 0.9891 - val_loss:
0.0232 - val_acc: 0.9937
Epoch 6/12
60000/60000 [==============================] - 657s 11ms/step - loss: 0.0363 - acc: 0.9907 - val_loss:
0.0272 - val_acc: 0.9924
Epoch 7/12
60000/60000 [==============================] - 647s 11ms/step - loss: 0.0309 - acc: 0.9916 - val_loss:
0.0264 - val_acc: 0.9928
Epoch 8/12
60000/60000 [==============================] - 644s 11ms/step - loss: 0.0297 - acc: 0.9921 - val_loss:
0.0255 - val_acc: 0.9934
Epoch 9/12
60000/60000 [==============================] - 646s 11ms/step - loss: 0.0237 - acc: 0.9934 - val_loss:
0.0312 - val_acc: 0.9909
Epoch 10/12
60000/60000 [==============================] - 652s 11ms/step - loss: 0.0213 - acc: 0.9943 - val_loss:
0.0337 - val_acc: 0.9913
Epoch 11/12
60000/60000 [==============================] - 660s 11ms/step - loss: 0.0210 - acc: 0.9946 - val_loss:
0.0250 - val_acc: 0.9934
Epoch 12/12
60000/60000 [==============================] - 659s 11ms/step - loss: 0.0177 - acc: 0.9953 - val_loss:
0.0362 - val_acc: 0.9908
```

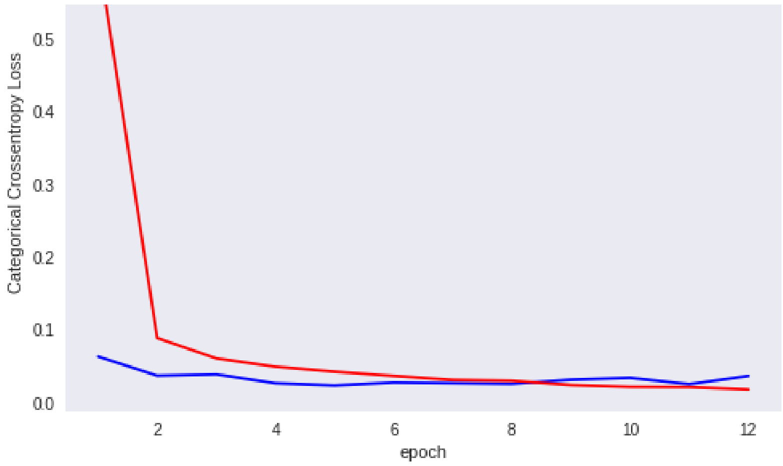
In [0]:

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

```
Test loss: 0.03619625380380894
Test accuracy: 0.9908
```

In [0]:

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



5 Convolution Layers

In [0]:

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),padding='same',activation='relu',input_shape=input_shape))
model.add(Conv2D(64,kernel_size = (5,5),padding = 'same',activation = 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(Conv2D(32, kernel_size=(4,4),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (5,5),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(Conv2D(32, kernel_size=(5,5),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (6,6),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(Conv2D(32, kernel_size=(4,4),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (6,6),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(Conv2D(32, kernel_size=(5,5),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (6,6),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_113 (Conv2D)          (None, 28, 28, 32)        320       
_________________________________________________________________
conv2d_114 (Conv2D)          (None, 28, 28, 64)        51264     
_________________________________________________________________
max_pooling2d_54 (MaxPooling (None, 10, 10, 64)        0         
_________________________________________________________________
conv2d_115 (Conv2D)          (None, 10, 10, 32)        32800     
_________________________________________________________________
conv2d_116 (Conv2D)          (None, 10, 10, 64)        51264     
_________________________________________________________________
max_pooling2d_55 (MaxPooling (None, 4, 4, 64)          0         
_________________________________________________________________
conv2d_117 (Conv2D)          (None, 4, 4, 32)          51232     
_________________________________________________________________
conv2d_118 (Conv2D)          (None, 4, 4, 64)          73792     
_________________________________________________________________
max_pooling2d_56 (MaxPooling (None, 2, 2, 64)          0         
_________________________________________________________________
conv2d_119 (Conv2D)          (None, 2, 2, 32)          32800     
_________________________________________________________________
conv2d_120 (Conv2D)          (None, 2, 2, 64)          73792     
_________________________________________________________________
max_pooling2d_57 (MaxPooling (None, 1, 1, 64)          0         
_________________________________________________________________
conv2d_121 (Conv2D)          (None, 1, 1, 32)          51232     
_________________________________________________________________
conv2d_122 (Conv2D)          (None, 1, 1, 64)          73792     
_________________________________________________________________
max_pooling2d_58 (MaxPooling (None, 1, 1, 64)          0         
_________________________________________________________________
dropout_38 (Dropout)         (None, 1, 1, 64)          0         
_________________________________________________________________
flatten_14 (Flatten)         (None, 64)                0         
_________________________________________________________________
dense_38 (Dense)             (None, 256)               16640     
_________________________________________________________________
batch_normalization_23 (Batc (None, 256)               1024      
_________________________________________________________________
dropout_39 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_39 (Dense)             (None, 256)               65792     
_________________________________________________________________
batch_normalization_24 (Batc (None, 256)               1024      
_________________________________________________________________
dropout_40 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_40 (Dense)             (None, 10)                2570      
=================================================================
Total params: 579,338
Trainable params: 578,314
Non-trainable params: 1,024
_________________________________________________________________
```

In [0]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 604s 10ms/step - loss: 1.4790 - acc: 0.3850 - val_loss: 1.0822 - val_acc: 0.5015
Epoch 2/12
60000/60000 [==============================] - 602s 10ms/step - loss: 0.5329 - acc: 0.8220 - val_loss: 0.3030 - val_acc: 0.9029
Epoch 3/12
60000/60000 [==============================] - 602s 10ms/step - loss: 0.2093 - acc: 0.9533 - val_loss: 0.1224 - val_acc: 0.9765
Epoch 4/12
60000/60000 [==============================] - 604s 10ms/step - loss: 0.1478 - acc: 0.9695 - val_loss: 0.0933 - val_acc: 0.9807
Epoch 5/12
60000/60000 [==============================] - 602s 10ms/step - loss: 0.1011 - acc: 0.9798 - val_loss: 0.0652 - val_acc: 0.9860
Epoch 6/12
60000/60000 [==============================] - 602s 10ms/step - loss: 0.0941 - acc: 0.9813 - val_loss: 0.0956 - val_acc: 0.9787
Epoch 7/12
60000/60000 [==============================] - 601s 10ms/step - loss: 0.0918 - acc: 0.9822 - val_loss: 0.0749 - val_acc: 0.9851
Epoch 8/12
60000/60000 [==============================] - 600s 10ms/step - loss: 0.0851 - acc: 0.9825 - val_loss: 0.0588 - val_acc: 0.9884
Epoch 9/12
60000/60000 [==============================] - 598s 10ms/step - loss: 0.0786 - acc: 0.9839 - val_loss: 0.0580 - val_acc: 0.9885
```

```
[illegible] - val_acc: [illegible]
Epoch 10/12
60000/60000 [==============================] - 601s 10ms/step - loss: 0.0638 - acc: 0.9867 - val_loss:
0.0378 - val_acc: 0.9930
Epoch 11/12
60000/60000 [==============================] - 610s 10ms/step - loss: 0.0480 - acc: 0.9907 - val_loss:
0.0488 - val_acc: 0.9893
Epoch 12/12
60000/60000 [==============================] - 608s 10ms/step - loss: 0.0527 - acc: 0.9898 - val_loss:
0.0447 - val_acc: 0.9887
```

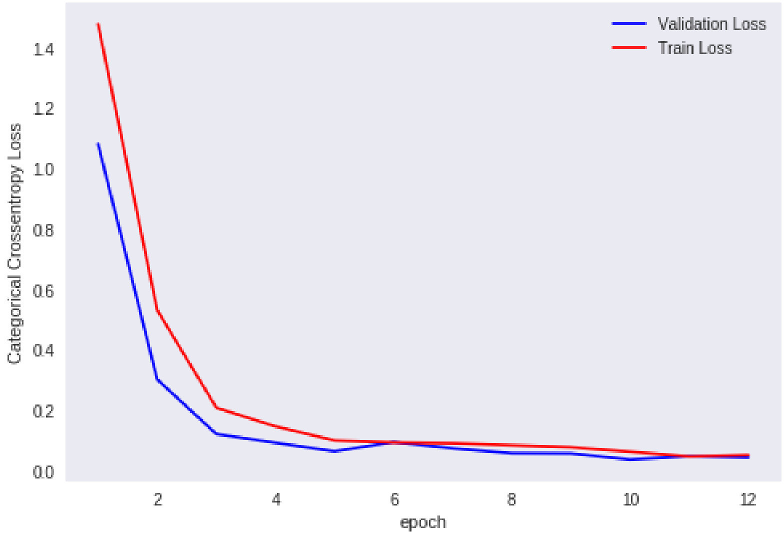
In [0]:

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

```
Test loss: 0.044668429520819335
Test accuracy: 0.9887
```

In [0]:

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



7 Convolutional Layers

In [0]:

```
model = Sequential()

model.add(Conv2D(32,kernel_size = (3,3),padding = 'same',activation = 'relu',input_shape = input_shape)
)
model.add(Conv2D(64,kernel_size = (5,5),padding = 'same',activation = 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(4,4),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (5,5),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(4,4),padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(5,5),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (6,6),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(5,5),padding='same'))
model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(4,4),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (6,6),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(6,6),padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(5,5),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (6,6),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(5,5),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (5,5),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(4,4),padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(4,4),padding='same',activation='relu'))
model.add(Conv2D(64,kernel_size = (7,7),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(5,5),padding='same'))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        51264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 32)        32800
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 64)        51264
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
batch_normalization_2 (Batch (None, 3, 3, 64)          256
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 32)          51232
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 3, 64)          73792
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
batch_normalization_3 (Batch (None, 1, 1, 64)          256
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 1, 1, 32)          32800
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 1, 1, 64)          73792
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
batch_normalization_4 (Batch (None, 1, 1, 64)          256
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 1, 1, 32)          51232
```

```
conv2d_9 (Conv2D)            (None, 1, 1, 32)          51232     
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 1, 1, 64)          73792     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0         
_________________________________________________________________
batch_normalization_5 (Batch (None, 1, 1, 64)          256       
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 1, 1, 32)          51232     
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 1, 1, 64)          51264     
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 1, 1, 64)          0         
_________________________________________________________________
batch_normalization_6 (Batch (None, 1, 1, 64)          256       
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 1, 1, 32)          32800     
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 1, 1, 64)          100416    
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 1, 1, 64)          0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 1, 64)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               16640     
_________________________________________________________________
batch_normalization_7 (Batch (None, 256)               1024      
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792     
_________________________________________________________________
batch_normalization_8 (Batch (None, 256)               1024      
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570      
=================================================================
Total params: 816,586
Trainable params: 814,794
Non-trainable params: 1,792
_________________________________________________________________
```

In [0]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_t
est, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 603s 10ms/step - loss: 0.9687 - acc: 0.6452 - val_loss:
0.5443 - val_acc: 0.8350
Epoch 2/12
60000/60000 [==============================] - 607s 10ms/step - loss: 0.2516 - acc: 0.9378 - val_loss:
0.1349 - val_acc: 0.9707
Epoch 3/12
60000/60000 [==============================] - 621s 10ms/step - loss: 0.1385 - acc: 0.9714 - val_loss:
0.1593 - val_acc: 0.9672
Epoch 4/12
60000/60000 [==============================] - 627s 10ms/step - loss: 0.1065 - acc: 0.9781 - val_loss:
0.0760 - val_acc: 0.9816
Epoch 5/12
60000/60000 [==============================] - 580s 10ms/step - loss: 0.0860 - acc: 0.9825 - val_loss:
0.0876 - val_acc: 0.9794
Epoch 6/12
60000/60000 [==============================] - 577s 10ms/step - loss: 0.0791 - acc: 0.9843 - val_loss:
0.1046 - val_acc: 0.9739
Epoch 7/12
60000/60000 [==============================] - 578s 10ms/step - loss: 0.0708 - acc: 0.9856 - val_loss:
0.0503 - val_acc: 0.9885
Epoch 8/12
60000/60000 [==============================] - 592s 10ms/step - loss: 0.0661 - acc: 0.9865 - val_loss:
0.0513 - val_acc: 0.9892
```

```
Epoch 9/12
60000/60000 [==============================] - 597s 10ms/step - loss: 0.0619 - acc: 0.9874 - val_loss:
0.0408 - val_acc: 0.9912
Epoch 10/12
60000/60000 [==============================] - 595s 10ms/step - loss: 0.0608 - acc: 0.9879 - val_loss:
0.0651 - val_acc: 0.9890
Epoch 11/12
60000/60000 [==============================] - 612s 10ms/step - loss: 0.0512 - acc: 0.9897 - val_loss:
0.0411 - val_acc: 0.9922
Epoch 12/12
60000/60000 [==============================] - 638s 11ms/step - loss: 0.0481 - acc: 0.9900 - val_loss:
0.0510 - val_acc: 0.9905
```

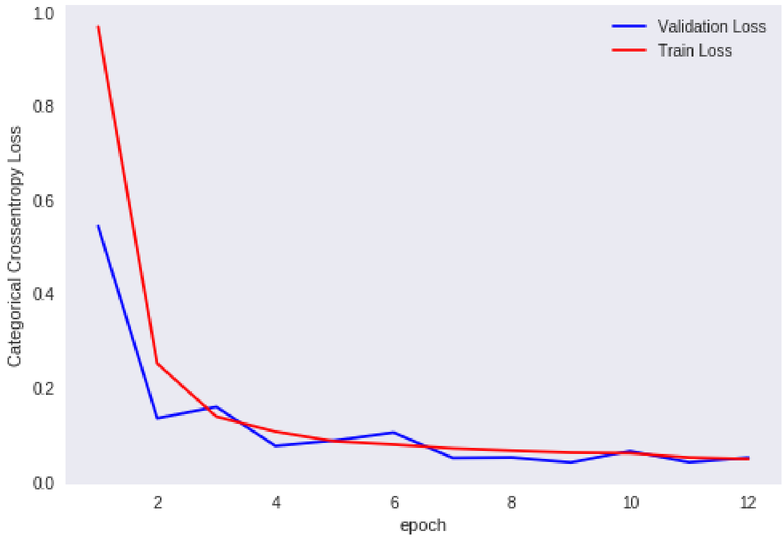
In [0]:

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

```
Test loss: 0.05102709092050791
Test accuracy: 0.9905
```

In [0]:

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


Validation Loss
Train Loss
Categorical Crossentropy Loss
epoch