In [0]:

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

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.

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/compat/v2_compat.py:68: disable_resource_variables (from
tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [0]:

```
# 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.layers.normalization import BatchNormalization
from keras import backend as K
```

```
Using TensorFlow backend.
```

In [0]:

```
%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, ty, 'r', label="Train Loss")
    ax.plot(x, vy, 'b', label="Validation Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
    plt.show()
```

In [0]:

```
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_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)
```

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

1) **Model-1:** 3 Convnet layers

In [0]:

```
%%time
model = Sequential()
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu',
                 input_shape=input_shape, padding='same',
                 kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(16, (3, 3), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
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'])

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])

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

# list of epoch numbers
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Plea
se use tf.compat.v1.get_default_graph instead.

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

e tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4479: The name tf.truncated_normal is deprecated. Please use tf.random.truncated_normal instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholder_with_default is deprecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated 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/backend/tensorflow_backend.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_backend.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_backend.py:203: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:2041: The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

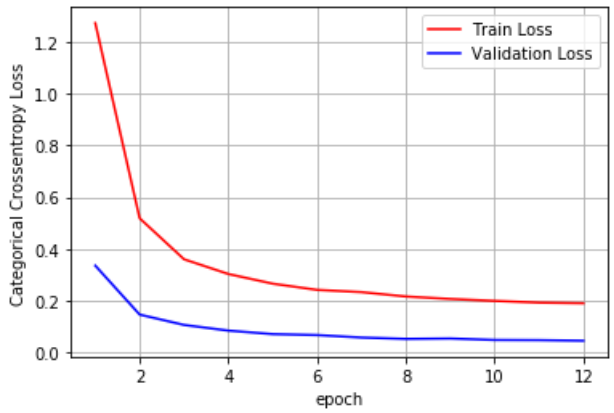
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is deprecated. Please use tf.math.log instead.

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 be 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_backend.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_backend.py:1020: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 33s 545us/step - loss: 1.2727 - acc: 0.5789 - val_loss: 0.3357 - val_acc: 0.9200
Epoch 2/12
60000/60000 [

```
60000/60000 [==============================] - 32s 536us/step - loss: 0.5187 - acc: 0.8368 - val_l
oss: 0.1465 - val_acc: 0.9559
Epoch 3/12
60000/60000 [==============================] - 32s 532us/step - loss: 0.3602 - acc: 0.8909 - val_l
oss: 0.1065 - val_acc: 0.9658
Epoch 4/12
60000/60000 [==============================] - 32s 529us/step - loss: 0.3033 - acc: 0.9093 - val_l
oss: 0.0845 - val_acc: 0.9754
Epoch 5/12
60000/60000 [==============================] - 32s 530us/step - loss: 0.2659 - acc: 0.9207 - val_l
oss: 0.0711 - val_acc: 0.9781
Epoch 6/12
60000/60000 [==============================] - 32s 530us/step - loss: 0.2419 - acc: 0.9267 - val_l
oss: 0.0673 - val_acc: 0.9784
Epoch 7/12
60000/60000 [==============================] - 32s 532us/step - loss: 0.2333 - acc: 0.9323 - val_l
oss: 0.0578 - val_acc: 0.9814
Epoch 8/12
60000/60000 [==============================] - 32s 535us/step - loss: 0.2163 - acc: 0.9363 - val_l
oss: 0.0526 - val_acc: 0.9839
Epoch 9/12
60000/60000 [==============================] - 32s 529us/step - loss: 0.2069 - acc: 0.9405 - val_l
oss: 0.0541 - val_acc: 0.9819
Epoch 10/12
60000/60000 [==============================] - 32s 527us/step - loss: 0.1994 - acc: 0.9411 - val_l
oss: 0.0484 - val_acc: 0.9852
Epoch 11/12
60000/60000 [==============================] - 32s 532us/step - loss: 0.1927 - acc: 0.9446 - val_l
oss: 0.0478 - val_acc: 0.9842
Epoch 12/12
60000/60000 [==============================] - 32s 536us/step - loss: 0.1904 - acc: 0.9452 - val_l
oss: 0.0455 - val_acc: 0.9845
Test loss: 0.045479313206672665
Test accuracy: 0.9845
```




```
CPU times: user 10min 51s, sys: 29.3 s, total: 11min 20s
Wall time: 6min 27s
```

Observation: This model is doing good with a relatively lower loss and higher accuracy and is not overfitting much.

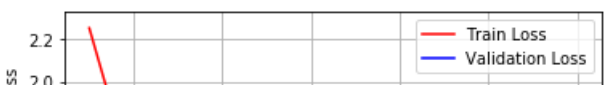
2) **Variation of Model-1:** 3 Convnet layers

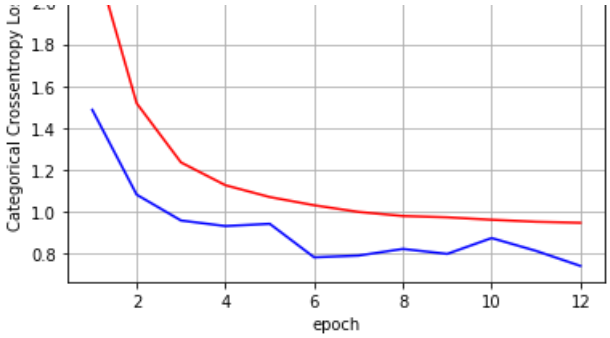
In [0]:

```
%%time
model = Sequential()
model.add(Conv2D(1, kernel_size=(3, 3), activation='relu',
                 input_shape=input_shape, padding='same',
                 kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(3, (3, 3), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))
```

```
model.add(Conv2D(5, (3, 3), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
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'])

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])

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

# list of epoch numbers
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 24s 404us/step - loss: 2.2551 - acc: 0.2406 - val_l
oss: 1.4862 - val_acc: 0.5907
Epoch 2/12
60000/60000 [==============================] - 23s 379us/step - loss: 1.5176 - acc: 0.4647 - val_l
oss: 1.0804 - val_acc: 0.6886
Epoch 3/12
60000/60000 [==============================] - 23s 379us/step - loss: 1.2345 - acc: 0.5740 - val_l
oss: 0.9566 - val_acc: 0.7149
Epoch 4/12
60000/60000 [==============================] - 23s 378us/step - loss: 1.1255 - acc: 0.6186 - val_l
oss: 0.9302 - val_acc: 0.6993
Epoch 5/12
60000/60000 [==============================] - 22s 372us/step - loss: 1.0689 - acc: 0.6400 - val_l
oss: 0.9409 - val_acc: 0.6849
Epoch 6/12
60000/60000 [==============================] - 22s 371us/step - loss: 1.0297 - acc: 0.6595 - val_l
oss: 0.7807 - val_acc: 0.7493
Epoch 7/12
60000/60000 [==============================] - 22s 368us/step - loss: 0.9978 - acc: 0.6691 - val_l
oss: 0.7890 - val_acc: 0.7371
Epoch 8/12
60000/60000 [==============================] - 22s 368us/step - loss: 0.9783 - acc: 0.6782 - val_l
oss: 0.8208 - val_acc: 0.7222
Epoch 9/12
60000/60000 [==============================] - 22s 367us/step - loss: 0.9720 - acc: 0.6818 - val_l
oss: 0.7970 - val_acc: 0.7334
Epoch 10/12
60000/60000 [==============================] - 22s 371us/step - loss: 0.9603 - acc: 0.6872 - val_l
oss: 0.8728 - val_acc: 0.7003
Epoch 11/12
60000/60000 [==============================] - 22s 370us/step - loss: 0.9507 - acc: 0.6872 - val_l
oss: 0.8115 - val_acc: 0.7192
Epoch 12/12
60000/60000 [==============================] - 22s 370us/step - loss: 0.9458 - acc: 0.6904 - val_l
oss: 0.7395 - val_acc: 0.7609
Test loss: 0.739545515537262
Test accuracy: 0.7609
```






```
CPU times: user 7min 16s, sys: 24.3 s, total: 7min 40s
Wall time: 4min 32s
```

Observation: The train loss and the Validation loss vary a little, but does not seem to converge.

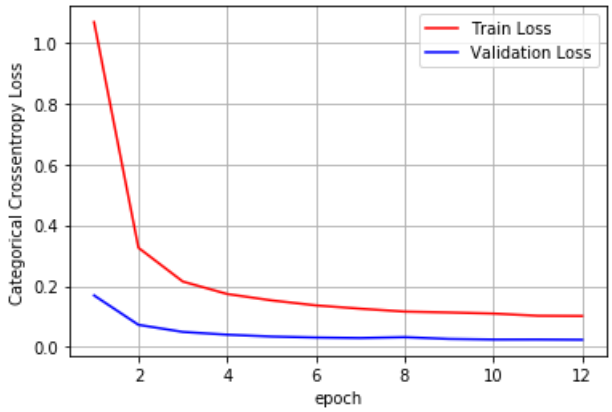
3) **Variation of Model-1:** 3 Convnet layers

In [0]:

```
%%time
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                 input_shape=input_shape, padding='same',
                 kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(32, (3, 3), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
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'])

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])

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

# list of epoch numbers
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 67s 1ms/step - loss: 1.0675 - acc: 0.6597 - val_los
s: 0.1692 - val_acc: 0.9555
Epoch 2/12
```

```
60000/60000 [==============================] - 64s 1ms/step - loss: 0.3258 - acc: 0.9035 - val_los
s: 0.0726 - val_acc: 0.9760
Epoch 3/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.2148 - acc: 0.9364 - val_los
s: 0.0492 - val_acc: 0.9834
Epoch 4/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1739 - acc: 0.9499 - val_los
s: 0.0402 - val_acc: 0.9865
Epoch 5/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1529 - acc: 0.9560 - val_los
s: 0.0341 - val_acc: 0.9881
Epoch 6/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1364 - acc: 0.9604 - val_los
s: 0.0308 - val_acc: 0.9899
Epoch 7/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1256 - acc: 0.9643 - val_los
s: 0.0292 - val_acc: 0.9892
Epoch 8/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1161 - acc: 0.9663 - val_los
s: 0.0320 - val_acc: 0.9899
Epoch 9/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1132 - acc: 0.9677 - val_los
s: 0.0262 - val_acc: 0.9905
Epoch 10/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1096 - acc: 0.9681 - val_los
s: 0.0242 - val_acc: 0.9916
Epoch 11/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1023 - acc: 0.9708 - val_los
s: 0.0243 - val_acc: 0.9907
Epoch 12/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.1017 - acc: 0.9706 - val_los
s: 0.0236 - val_acc: 0.9910
Test loss: 0.023584829754967358
Test accuracy: 0.991
```




```
CPU times: user 23min 24s, sys: 42.7 s, total: 24min 6s
Wall time: 12min 55s
```

Observation: This model is doing very good with a relatively lower loss and higher accuracy and is not overfitting very much.

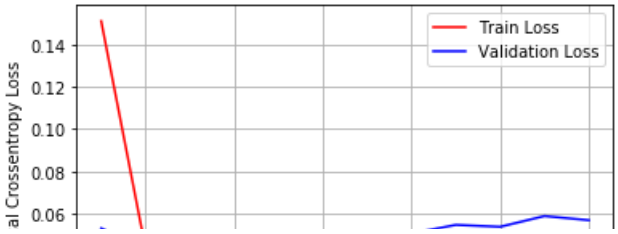
4) **Variation of Model-1:** 3 Convnet layers

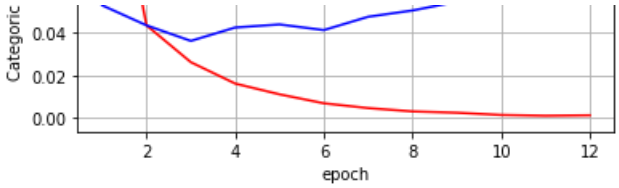
In [0]:

```
%%time
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                 input_shape=input_shape, padding='valid',
                 kernel_initializer='he_normal'))

model.add(Conv2D(32, (3, 3), activation='relu',
                 padding='valid', kernel_initializer='he_normal'))

model.add(Conv2D(32, (3, 3), activation='relu',
                 padding='valid', kernel_initializer='he_normal'))

model.add(Flatten())
```

```
model.add(Dense(32, activation='relu'))

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

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])

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

# list of epoch numbers
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 155s 3ms/step - loss: 0.1510 - acc: 0.9534 - val_lo
ss: 0.0531 - val_acc: 0.9850
Epoch 2/12
60000/60000 [==============================] - 155s 3ms/step - loss: 0.0436 - acc: 0.9865 - val_lo
ss: 0.0436 - val_acc: 0.9862
Epoch 3/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0262 - acc: 0.9923 - val_lo
ss: 0.0363 - val_acc: 0.9894
Epoch 4/12
60000/60000 [==============================] - 155s 3ms/step - loss: 0.0161 - acc: 0.9949 - val_lo
ss: 0.0426 - val_acc: 0.9881
Epoch 5/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0110 - acc: 0.9964 - val_lo
ss: 0.0441 - val_acc: 0.9885
Epoch 6/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0068 - acc: 0.9979 - val_lo
ss: 0.0414 - val_acc: 0.9900
Epoch 7/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0045 - acc: 0.9984 - val_lo
ss: 0.0477 - val_acc: 0.9874
Epoch 8/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0030 - acc: 0.9991 - val_lo
ss: 0.0507 - val_acc: 0.9896
Epoch 9/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0023 - acc: 0.9993 - val_lo
ss: 0.0547 - val_acc: 0.9890
Epoch 10/12
60000/60000 [==============================] - 155s 3ms/step - loss: 0.0013 - acc: 0.9997 - val_lo
ss: 0.0538 - val_acc: 0.9905
Epoch 11/12
60000/60000 [==============================] - 153s 3ms/step - loss: 9.1630e-04 - acc: 0.9998 - va
l_loss: 0.0589 - val_acc: 0.9899
Epoch 12/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0011 - acc: 0.9998 - val_lo
ss: 0.0569 - val_acc: 0.9901
Test loss: 0.056943806147187616
Test accuracy: 0.9901
```






```
CPU times: user 58min 7s, sys: 1min 10s, total: 59min 18s
Wall time: 30min 55s
```

Observation: The train loss and the Validation loss vary much here.

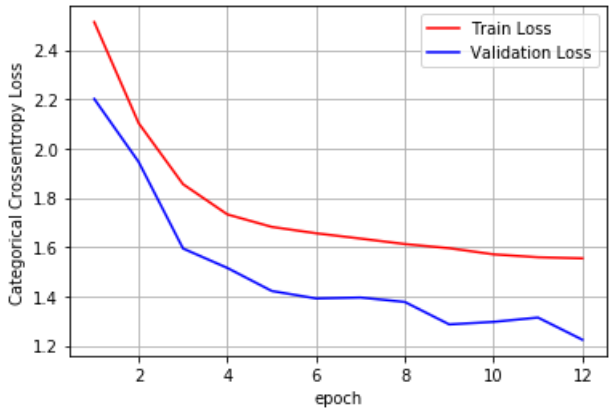
5) **Model -2:** 4 - Convnet layers

In [0]:

```
%%time
model = Sequential()
model.add(Conv2D(1, kernel_size=(4, 4), activation='relu',
                 input_shape=input_shape, padding='same',
                 kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(3, (4, 4), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(5, (4, 4), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Conv2D(7, (4, 4), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(16, activation='relu'))
model.add(BatchNormalization())
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'])

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])

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

# list of epoch numbers
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 27s 450us/step - loss: 2.5135 - acc: 0.1224 - val_l
oss: 2.2016 - val_acc: 0.1240
Epoch 2/12
```

```
Epoch 2/12
60000/60000 [==============================] - 26s 426us/step - loss: 2.1033 - acc: 0.2154 - val_l
oss: 1.9461 - val_acc: 0.2744
Epoch 3/12
60000/60000 [==============================] - 26s 432us/step - loss: 1.8557 - acc: 0.3156 - val_l
oss: 1.5946 - val_acc: 0.4629
Epoch 4/12
60000/60000 [==============================] - 26s 426us/step - loss: 1.7329 - acc: 0.3486 - val_l
oss: 1.5156 - val_acc: 0.4682
Epoch 5/12
60000/60000 [==============================] - 26s 430us/step - loss: 1.6822 - acc: 0.3613 - val_l
oss: 1.4222 - val_acc: 0.4849
Epoch 6/12
60000/60000 [==============================] - 26s 427us/step - loss: 1.6564 - acc: 0.3741 - val_l
oss: 1.3917 - val_acc: 0.4994
Epoch 7/12
60000/60000 [==============================] - 26s 429us/step - loss: 1.6348 - acc: 0.3833 - val_l
oss: 1.3957 - val_acc: 0.4645
Epoch 8/12
60000/60000 [==============================] - 26s 428us/step - loss: 1.6126 - acc: 0.3936 - val_l
oss: 1.3775 - val_acc: 0.4769
Epoch 9/12
60000/60000 [==============================] - 26s 428us/step - loss: 1.5958 - acc: 0.3970 - val_l
oss: 1.2864 - val_acc: 0.5122
Epoch 10/12
60000/60000 [==============================] - 26s 435us/step - loss: 1.5708 - acc: 0.4063 - val_l
oss: 1.2968 - val_acc: 0.5025
Epoch 11/12
60000/60000 [==============================] - 26s 430us/step - loss: 1.5590 - acc: 0.4105 - val_l
oss: 1.3142 - val_acc: 0.5041
Epoch 12/12
60000/60000 [==============================] - 26s 427us/step - loss: 1.5545 - acc: 0.4191 - val_l
oss: 1.2243 - val_acc: 0.5520
Test loss: 1.2242729833602906
Test accuracy: 0.552
```




```
CPU times: user 8min 28s, sys: 17 s, total: 8min 45s
Wall time: 5min 13s
```

Observation: The train loss and the Validation loss vary much here.

6) **Model -3:** 5 - Convnet layers

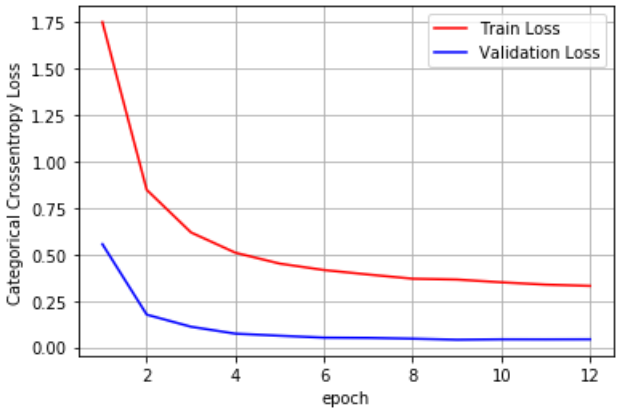
In [0]:

```
#%%time
model = Sequential()
model.add(Conv2D(16, kernel_size=(5, 5), activation='relu',
                 input_shape=input_shape, padding='same',
                 kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(16, (5, 5), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
#model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
```

```
model.add(Dropout(0.25))

model.add(Conv2D(16, (5, 5), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Conv2D(16, (5, 5), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Conv2D(16, (5, 5), activation='relu',
                 padding='same', kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
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'])

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])

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

# list of epoch numbers
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 101s 2ms/step - loss: 1.7502 - acc: 0.4025 - val_lo
ss: 0.5551 - val_acc: 0.8840
Epoch 2/12
60000/60000 [==============================] - 99s 2ms/step - loss: 0.8473 - acc: 0.7273 - val_los
s: 0.1756 - val_acc: 0.9696
Epoch 3/12
60000/60000 [==============================] - 99s 2ms/step - loss: 0.6179 - acc: 0.8120 - val_los
s: 0.1104 - val_acc: 0.9759
Epoch 4/12
60000/60000 [==============================] - 98s 2ms/step - loss: 0.5089 - acc: 0.8485 - val_los
s: 0.0732 - val_acc: 0.9828
Epoch 5/12
60000/60000 [==============================] - 98s 2ms/step - loss: 0.4502 - acc: 0.8660 - val_los
s: 0.0621 - val_acc: 0.9846
Epoch 6/12
60000/60000 [==============================] - 98s 2ms/step - loss: 0.4156 - acc: 0.8765 - val_los
s: 0.0517 - val_acc: 0.9871
Epoch 7/12
60000/60000 [==============================] - 99s 2ms/step - loss: 0.3915 - acc: 0.8855 - val_los
s: 0.0504 - val_acc: 0.9867
Epoch 8/12
60000/60000 [==============================] - 99s 2ms/step - loss: 0.3689 - acc: 0.8921 - val_los
s: 0.0462 - val_acc: 0.9882
Epoch 9/12
60000/60000 [==============================] - 99s 2ms/step - loss: 0.3645 - acc: 0.8933 - val_los
s: 0.0405 - val_acc: 0.9894
Epoch 10/12
60000/60000 [==============================] - 98s 2ms/step - loss: 0.3496 - acc: 0.8993 - val_los
```

```
s: 0.0427 - val_acc: 0.9894
Epoch 11/12
60000/60000 [==============================] - 98s 2ms/step - loss: 0.3371 - acc: 0.9011 - val_los
s: 0.0425 - val_acc: 0.9893
Epoch 12/12
60000/60000 [==============================] - 98s 2ms/step - loss: 0.3303 - acc: 0.9060 - val_los
s: 0.0429 - val_acc: 0.9904
Test loss: 0.042885919146612286
Test accuracy: 0.9904
```



Observation: This model is doing good with a relatively low loss and high accuracy and is not overfitting very much.

Summary

In [3]:

```
from prettytable import PrettyTable
pt = PrettyTable()
pt.field_names = ["Number of Convolutional Layers", "Activation", "Kernel dimension", "Channels", "
BatchNormalization", "Dropout", "Max Pooling (2x2)", "Test loss","Test accuracy"]
pt.add_row(["3", "relu", "3,3", "8,16,32", "Yes", "Yes (0.25,0.25,0.4,0.5)", "Yes", "0.045","0.985"
])
pt.add_row(["3", "relu", "3,3", "1,3,5", "Yes", "Yes (0.25,0.25,0.4,0.5)", "Yes", "0.740","0.761"])
pt.add_row(["3", "relu", "3,3", "32,32,32", "Yes", "Yes (0.25,0.25,0.4,0.5)", "Yes", "0.024","0.991
"])
pt.add_row(["3", "relu", "3,3", "32,32,32", "No", "No", "No", "0.057","0.990"])
pt.add_row(["4", "relu", "4,4", "1,3,5,7", "Yes", "Yes (0.25,0.25,0.3,0.4,0.5)", "Yes", "1.224","0.
552"])
pt.add_row(["5", "relu", "5,5", "16,16,16,16,16", "Yes", "Yes (0.25,0.25,0.3,0.4,0.5)", "Yes", "0.0
43","0.990"])
print(pt)
```

```
+-------------------------------+------------+-----------------+----------------+----------------
-+-------------------------------+------------------+-----------+---------------+
| Number of Convolutional Layers | Activation | Kernel dimension |    Channels    |
BatchNormalization |           Dropout           | Max Pooling (2x2) | Test loss | Test accuracy |
+-------------------------------+------------+-----------------+----------------+----------------
-+-------------------------------+------------------+-----------+---------------+
|               3               |    relu    |       3,3        |    8,16,32     |        Yes
|   Yes (0.25,0.25,0.4,0.5)     |        Yes        |   0.045   |     0.985     |
|               3               |    relu    |       3,3        |     1,3,5      |        Yes
|   Yes (0.25,0.25,0.4,0.5)     |        Yes        |   0.740   |     0.761     |
|               3               |    relu    |       3,3        |    32,32,32    |        Yes
|   Yes (0.25,0.25,0.4,0.5)     |        Yes        |   0.024   |     0.991     |
|               3               |    relu    |       3,3        |    32,32,32    |         No
|              No               |         No        |   0.057   |     0.990     |
|               4               |    relu    |       4,4        |    1,3,5,7     |        Yes
| Yes (0.25,0.25,0.3,0.4,0.5) |        Yes        |   1.224   |     0.552     |
|               5               |    relu    |       5,5        | 16,16,16,16,16 |        Yes
| Yes (0.25,0.25,0.3,0.4,0.5) |        Yes        |   0.043   |     0.990     |
+-------------------------------+------------+-----------------+----------------+----------------
-+-------------------------------+------------------+-----------+---------------+
```

Conclusion:

- The best model is the one with 3 Convnet layers, 3x3x32 Kernel filters along with techniques like Batch normalization, Dropout, Max pooling having the minimum Test loss and Maximum Test accuracy without overfitting.
- Techniques like Batch normalization and Dropout are important for a model's better performance, Max pooling is required to reduce the matrix dimension to reduce time and space utilization.
- The Number of channels is also important to get a minimum Test loss and Maximum Test accuracy without overfitting.
- It is better to have the Kernel's dimension in odd number and the number of channels in powers of 2.
- For a 28x28 image 3x3 kernel seems enough and as the image's dimension gets bigger filters like 5x5 or even 7x7 can be used.
- Since the MNIST dataset is not a very complex dataset even a 3 layer Convnet seems sufficient.