In [1]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py
#importing all then necessary modules

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 tensorflow.keras import backend as K
```

```
Using TensorFlow backend.
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:516:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:517:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:518:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:519:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:520:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
WARNING: Logging before flag parsing goes to stderr.
W0911 08:38:40.618894 140189713626944 __init__.py:308] Limited tf.compat.v2.summary API due to mis
sing TensorBoard installation.
```

In [2]:

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

# input image dimensions
img_rows, img_cols = 28, 28
```

In [3]:

```
#Train and test split
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [4]:

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

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

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

In [5]:

```
print(y_train.shape)
```

```
(60000, 10)
```

In [6]:

```
%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):
  fig = plt.figure( facecolor='y', edgecolor='k')
  plt.plot(x, vy, 'b', label="Validation Loss")
  plt.plot(x, ty, 'r', label="Train Loss")
  plt.xlabel('Epochs')
  plt.ylabel('Categorical Crossentropy Loss')
  plt.legend()
  plt.grid()
  plt.show()
```

1 Model 1:CNN with 3 ConvNet & 3x3 kernel size

Stack all the layers

In [7]:

```
from keras.initializers import he_normal

convnet3=Sequential() # Initializing the model

# First ConvNet
convnet3.add(Conv2D(32,kernel_size=(3,3),
                    activation='relu',
                    input_shape=input_shape))

convnet3.add(Conv2D(64,kernel_size=(3,3),
                    activation='relu'))

convnet3.add(Dropout(0.25))

convnet3.add(Conv2D(128,kernel_size=(3,3),
                   activation='relu'))
#maxpooling by (2,2 ) ,dropout,flattening
convnet3.add(MaxPooling2D(pool_size=(2,2)))
convnet3.add(Dropout(0.25))
convnet3.add(Flatten())

#hidden_layer
convnet3.add(Dense(256,
                   activation='relu',
```

```
                 kernel_initializer=he_normal(seed=None)))
convnet3.add(Dropout(0.5))
convnet3.add(Dense(num_classes,activation='softmax'))
print(convnet3.summary())
```

```
W0911 08:49:22.863518 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:74: The nam
e tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0911 08:49:22.885082 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:517: The na
me tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

W0911 08:49:22.891038 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:4138: The n
ame tf.random_uniform is deprecated. Please use tf.random.uniform instead.

W0911 08:49:22.915654 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:133: The na
me tf.placeholder_with_default is deprecated. Please use tf.compat.v1.placeholder_with_default ins
tead.

W0911 08:49:22.920943 140189713626944 deprecation.py:506] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:3445:
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`.
W0911 08:49:22.941576 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:3976: The n
ame tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

W0911 08:49:22.963110 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:4185: The n
ame tf.truncated_normal is deprecated. Please use tf.random.truncated_normal instead.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 11, 11, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 15488)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               3965184
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 4,060,426
Trainable params: 4,060,426
Non-trainable params: 0
_________________________________________________________________
None
```

Model compile and fit

In [8]:

```
#Model compilation
convnet3.compile(optimizer=keras.optimizers.Adam(),
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
```

```
                metrics=['accuracy'])
convnet3_history=convnet3.fit(x_train,y_train,batch_size=batch_size,
                              epochs=epochs,
                              verbose=1,
                              validation_data=(x_test, y_test))
```

```
W0911 08:49:32.391674 140189713626944 deprecation_wrapper.py:119] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/keras/optimizers.py:790: The name
tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

W0911 08:49:32.525590 140189713626944 deprecation.py:323] From
/home/ubuntu/anaconda3/lib/python3.6/site-packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (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
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.1822 - acc: 0.9431 - val_lo
ss: 0.0421 - val_acc: 0.9862
Epoch 2/12
60000/60000 [==============================] - 140s 2ms/step - loss: 0.0634 - acc: 0.9810 - val_lo
ss: 0.0295 - val_acc: 0.9911
Epoch 3/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0440 - acc: 0.9862 - val_lo
ss: 0.0248 - val_acc: 0.9918
Epoch 4/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0358 - acc: 0.9890 - val_lo
ss: 0.0244 - val_acc: 0.9919
Epoch 5/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0308 - acc: 0.9902 - val_lo
ss: 0.0229 - val_acc: 0.9927
Epoch 6/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0266 - acc: 0.9913 - val_lo
ss: 0.0244 - val_acc: 0.9923
Epoch 7/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0228 - acc: 0.9925 - val_lo
ss: 0.0260 - val_acc: 0.9920
Epoch 8/12
60000/60000 [==============================] - 142s 2ms/step - loss: 0.0216 - acc: 0.9930 - val_lo
ss: 0.0249 - val_acc: 0.9925
Epoch 9/12
60000/60000 [==============================] - 139s 2ms/step - loss: 0.0183 - acc: 0.9940 - val_lo
ss: 0.0261 - val_acc: 0.9919
Epoch 10/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0159 - acc: 0.9951 - val_lo
ss: 0.0203 - val_acc: 0.9937
Epoch 11/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.0153 - acc: 0.9952 - val_lo
ss: 0.0242 - val_acc: 0.9936
Epoch 12/12
60000/60000 [==============================] - 158s 3ms/step - loss: 0.0137 - acc: 0.9956 - val_lo
ss: 0.0222 - val_acc: 0.9936
```

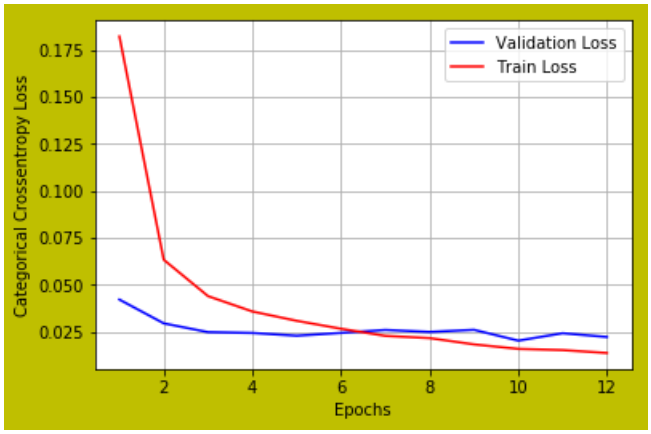
Evaluating model 1

In [9]:

```
#evaluating model

score=convnet3.evaluate(x_test,y_test,verbose=0)
test_score3=score[0]
test_accuracy3=score[1]
train_accuracy3=max(convnet3_history.history['acc'])
print('test score :',test_score3)
print('test sccuracy :',test_accuracy3)
# error plot
x=list(range(1,epochs+1))
vy=convnet3_history.history['val_loss'] #validation loss
ty=convnet3_history.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
test score : 0.022231986878935232
test sccuracy : 0.9936
```



Changing dropout layer to 0.5, weight initializer to 'glorot_uniform' and optimizer to rmsprop

Stack all the layers

In [10]:

```
from keras.initializers import he_normal

convnet3=Sequential() # Initializing the model

# First ConvNet
convnet3.add(Conv2D(32,kernel_size=(3,3),
                    activation='relu',
                    input_shape=input_shape))

convnet3.add(Conv2D(64,kernel_size=(3,3),
                    activation='relu'))

convnet3.add(Dropout(0.5))

convnet3.add(Conv2D(128,kernel_size=(3,3),
                   activation='relu'))
#maxpooling by (2,2 ) ,dropout,flattening
convnet3.add(MaxPooling2D(pool_size=(2,2)))
convnet3.add(Dropout(0.5))
convnet3.add(Flatten())

#hidden_layer
convnet3.add(Dense(256,
                   activation='relu',
                   kernel_initializer='glorot_uniform'))
convnet3.add(Dropout(0.5))
convnet3.add(Dense(num_classes,activation='softmax'))
print(convnet3.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
dropout_4 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 22, 22, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 11, 11, 128)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 15488)             0
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 256)               3965184
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 4,060,426
Trainable params: 4,060,426
Non-trainable params: 0
_________________________________________________________________
None
```

Model compile and fit

In [11]:

```
#Model compilation
convnet3.compile(optimizer=keras.optimizers.RMSprop(),
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
convnet3_history=convnet3.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 [==============================] - 180s 3ms/step - loss: 0.2205 - acc: 0.9327 - val_lo
ss: 0.0535 - val_acc: 0.9846
Epoch 2/12
60000/60000 [==============================] - 185s 3ms/step - loss: 0.0782 - acc: 0.9771 - val_lo
ss: 0.0389 - val_acc: 0.9879
Epoch 3/12
60000/60000 [==============================] - 185s 3ms/step - loss: 0.0650 - acc: 0.9808 - val_lo
ss: 0.0572 - val_acc: 0.9841
Epoch 4/12
60000/60000 [==============================] - 185s 3ms/step - loss: 0.0627 - acc: 0.9823 - val_lo
ss: 0.0355 - val_acc: 0.9888
Epoch 5/12
60000/60000 [==============================] - 186s 3ms/step - loss: 0.0611 - acc: 0.9821 - val_lo
ss: 0.0354 - val_acc: 0.9889
Epoch 6/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0605 - acc: 0.9823 - val_lo
ss: 0.0314 - val_acc: 0.9893
Epoch 7/12
60000/60000 [==============================] - 140s 2ms/step - loss: 0.0593 - acc: 0.9835 - val_lo
ss: 0.0418 - val_acc: 0.9898
Epoch 8/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.0576 - acc: 0.9836 - val_lo
ss: 0.0528 - val_acc: 0.9891
Epoch 9/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0586 - acc: 0.9835 - val_lo
ss: 0.1287 - val_acc: 0.9837
Epoch 10/12
60000/60000 [==============================] - 162s 3ms/step - loss: 0.0576 - acc: 0.9831 - val_lo
ss: 0.0435 - val_acc: 0.9899
Epoch 11/12
60000/60000 [==============================] - 140s 2ms/step - loss: 0.0569 - acc: 0.9837 - val_lo
ss: 0.0405 - val_acc: 0.9904
Epoch 12/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0552 - acc: 0.9848 - val_lo
ss: 0.0348 - val_acc: 0.9885
```

Evaluating model

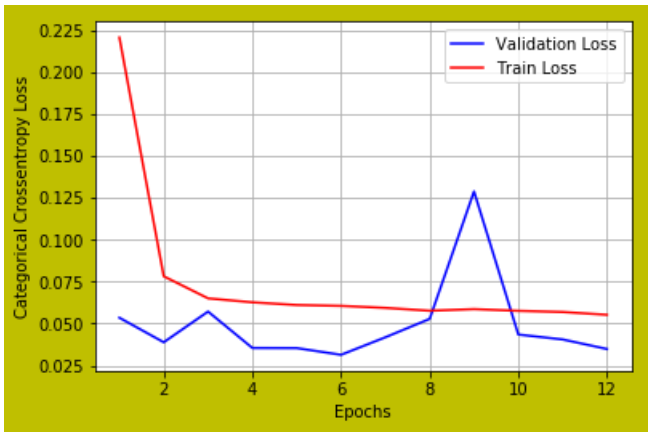
In [12]:

```
#evaluating model

score=convnet3.evaluate(x_test,y_test,verbose=0)
test score3=score[0]
```

```
test_accuracy3=score[1]
train_accuracy3=max(convnet3_history.history['acc'])
print('test score :',test_score3)
print('test sccuracy :',test_accuracy3)
# error plot
x=list(range(1,epochs+1))
vy=convnet3_history.history['val_loss'] #validation loss
ty=convnet3_history.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
test score : 0.034848048690008
test sccuracy : 0.9885
```



2 Model2:CNN with 5 ConvNet & kernel_size=(5x5)

5 convNet followed by maxpooling(2,2) and dropout

Stack all the layers

In [14]:

```
from keras.layers.normalization import BatchNormalization
convnet5=Sequential() # Initializing the model

# First ConvNet
convnet5.add(Conv2D(32,kernel_size=(5,5),
                    activation='relu',
                    padding='same',
                    input_shape=input_shape))

convnet5.add(Conv2D(64,kernel_size=(5,5),
                    padding='same',
                    activation='relu'))#Second Convnet
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.25))

convnet5.add(Conv2D(96,kernel_size=(5,5),
                    padding='same',
                   activation='relu'))  # 3rd ConvNet
#maxpooling by (2,2 ) ,dropout,flattening
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.25))

convnet5.add(Conv2D(128,kernel_size=(5,5),
                    padding='same',
                    activation='relu'))#fourth Convnet
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.25))
convnet5.add(Conv2D(164,kernel_size=(5,5),
                    padding='same',
                    activation='relu'))#fifth Convnet
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.25))
convnet5.add(Flatten())
```

```
#hidden_layer
convnet5.add(Dense(256,
                   activation='relu',
                   kernel_initializer=he_normal(seed=None)))
convnet5.add(BatchNormalization())
convnet5.add(Dropout(0.5))
convnet5.add(Dense(num_classes,activation='softmax'))
print(convnet5.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 64)        51264
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 14, 14, 96)        153696
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 96)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 7, 7, 96)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 7, 7, 128)         307328
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
dropout_9 (Dropout)          (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 3, 3, 164)         524964
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 1, 1, 164)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 1, 1, 164)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 164)               0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               42240
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dropout_11 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,083,918
Trainable params: 1,083,406
Non-trainable params: 512
_________________________________________________________________
None
```

Model compile and fit

In [15]:

```
#Model compilation
from datetime import datetime
start = datetime.now()
convnet5.compile(optimizer=keras.optimizers.Adam(),
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
convnet5_history=convnet5.fit(x_train,y_train,batch_size=batch_size,
                              epochs=epochs,
                              verbose=1,
                              validation_data=(x_test, y_test))
print("Time taken :", datetime.now() - start)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 241s 4ms/step - loss: 2.2567 - acc: 0.2412 - val_lo
```

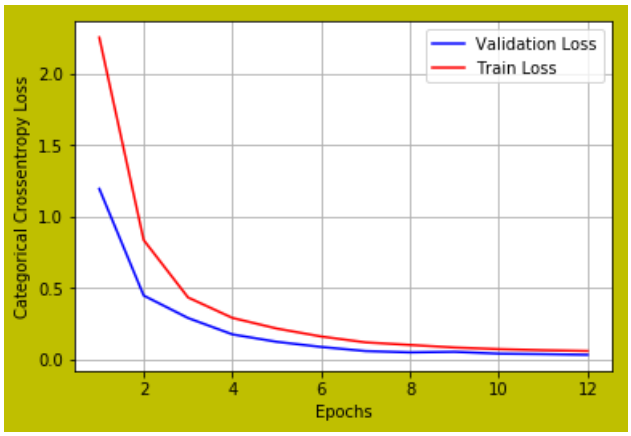
```
ss: 1.1946 - val_acc: 0.5927
Epoch 2/12
60000/60000 [==============================] - 269s 4ms/step - loss: 0.8350 - acc: 0.7161 - val_lo
ss: 0.4455 - val_acc: 0.8568
Epoch 3/12
60000/60000 [==============================] - 284s 5ms/step - loss: 0.4320 - acc: 0.8646 - val_lo
ss: 0.2878 - val_acc: 0.9078
Epoch 4/12
60000/60000 [==============================] - 251s 4ms/step - loss: 0.2885 - acc: 0.9122 - val_lo
ss: 0.1728 - val_acc: 0.9456
Epoch 5/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.2139 - acc: 0.9354 - val_lo
ss: 0.1207 - val_acc: 0.9624
Epoch 6/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.1581 - acc: 0.9530 - val_lo
ss: 0.0848 - val_acc: 0.9747
Epoch 7/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.1168 - acc: 0.9656 - val_lo
ss: 0.0559 - val_acc: 0.9833
Epoch 8/12
60000/60000 [==============================] - 197s 3ms/step - loss: 0.0988 - acc: 0.9715 - val_lo
ss: 0.0466 - val_acc: 0.9841
Epoch 9/12
60000/60000 [==============================] - 197s 3ms/step - loss: 0.0809 - acc: 0.9765 - val_lo
ss: 0.0506 - val_acc: 0.9833
Epoch 10/12
60000/60000 [==============================] - 197s 3ms/step - loss: 0.0694 - acc: 0.9799 - val_lo
ss: 0.0377 - val_acc: 0.9872
Epoch 11/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.0618 - acc: 0.9822 - val_lo
ss: 0.0343 - val_acc: 0.9887
Epoch 12/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.0565 - acc: 0.9832 - val_lo
ss: 0.0296 - val_acc: 0.9915
Time taken : 0:43:25.544832
```

Evalutating Model 2

In [16]:

```
#evaluating model
score=convnet5.evaluate(x_test,y_test,verbose=0)
test_score5=score[0]
test_accuracy5=score[1]
train_accuracy5=max(convnet5_history.history['acc'])
print('test score :',test_score5)
print('test Accuracy :',test_accuracy5)
# error plot
x=list(range(1,epochs+1))
vy=convnet5_history.history['val_loss'] #validation loss
ty=convnet5_history.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
test score : 0.029589390929794172
test Accuracy : 0.9915
```



Changing the dropout to 0.5 , activation to 'LeakyRelu', optimizer to Adadelta, weight initalizer to 'glorot_uniform'

In [21]:

```
from keras.layers import LeakyReLU
convnet5=Sequential() # Initializing the model

#First ConvNet
convnet5.add(Conv2D(32,kernel_size=(5,5),

                    padding='same',
                    input_shape=input_shape))
convnet5.add(LeakyReLU(alpha=0.05))

convnet5.add(Conv2D(64,kernel_size=(5,5),
                    padding='same',))#Second Convnet
convnet5.add(LeakyReLU(alpha=0.05))
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.5))

convnet5.add(Conv2D(96,kernel_size=(5,5),
                    padding='same'))  # 3rd ConvNet
#maxpooling by (2,2 ) ,dropout,flattening
convnet5.add(LeakyReLU(alpha=0.05))

convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.5))

convnet5.add(Conv2D(128,kernel_size=(5,5),
                    padding='same'))#fourth Convnet
convnet5.add(LeakyReLU(alpha=0.05))
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.5))
convnet5.add(Conv2D(164,kernel_size=(5,5),
                    padding='same'))#fifth Convnet
convnet5.add(LeakyReLU(alpha=0.05))
convnet5.add(MaxPooling2D(pool_size=(2,2)))
convnet5.add(Dropout(0.5))
convnet5.add(Flatten())

#hidden_layer
convnet5.add(Dense(256,kernel_initializer='glorot_uniform'))
convnet5.add(LeakyReLU(alpha=0.05))
convnet5.add(BatchNormalization())
convnet5.add(Dropout(0.5))
convnet5.add(Dense(num_classes,activation='softmax'))
print(convnet5.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 28, 28, 32)        832
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 28, 28, 64)        51264
_________________________________________________________________
leaky_re_lu_13 (LeakyReLU)   (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_21 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 14, 14, 96)        153696
_________________________________________________________________
leaky_re_lu_14 (LeakyReLU)   (None, 14, 14, 96)        0
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 7, 7, 96)          0
_________________________________________________________________
dropout_22 (Dropout)         (None, 7, 7, 96)          0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 7, 7, 128)         307328
_________________________________________________________________
leaky_re_lu_15 (LeakyReLU)   (None, 7, 7, 128)         0
```

```
_  _  _
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_23 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 3, 3, 164)         524964
_________________________________________________________________
leaky_re_lu_16 (LeakyReLU)   (None, 3, 3, 164)         0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 1, 1, 164)         0
_________________________________________________________________
dropout_24 (Dropout)         (None, 1, 1, 164)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 164)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               42240
_________________________________________________________________
leaky_re_lu_17 (LeakyReLU)   (None, 256)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 256)               1024
_________________________________________________________________
dropout_25 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 10)                2570
=================================================================
Total params: 1,083,918
Trainable params: 1,083,406
Non-trainable params: 512
_________________________________________________________________
None
```

In [22]:

```
#Model compilation
from datetime import datetime
start = datetime.now()
convnet5.compile(optimizer=keras.optimizers.Adadelta(),
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
convnet5_history=convnet5.fit(x_train,y_train,batch_size=batch_size,
                              epochs=epochs,
                              verbose=1,
                              validation_data=(x_test, y_test))
print("Time taken :", datetime.now() - start)
```

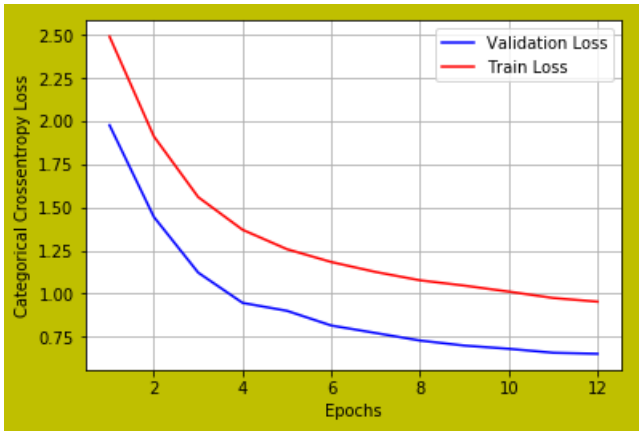
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 200s 3ms/step - loss: 2.4916 - acc: 0.1667 - val_lo
ss: 1.9776 - val_acc: 0.3015
Epoch 2/12
60000/60000 [==============================] - 199s 3ms/step - loss: 1.9123 - acc: 0.3064 - val_lo
ss: 1.4452 - val_acc: 0.5174
Epoch 3/12
60000/60000 [==============================] - 199s 3ms/step - loss: 1.5598 - acc: 0.4484 - val_lo
ss: 1.1219 - val_acc: 0.6472
Epoch 4/12
60000/60000 [==============================] - 202s 3ms/step - loss: 1.3705 - acc: 0.5244 - val_lo
ss: 0.9465 - val_acc: 0.7027
Epoch 5/12
60000/60000 [==============================] - 209s 3ms/step - loss: 1.2579 - acc: 0.5693 - val_lo
ss: 0.9005 - val_acc: 0.7113
Epoch 6/12
60000/60000 [==============================] - 209s 3ms/step - loss: 1.1834 - acc: 0.5980 - val_lo
ss: 0.8151 - val_acc: 0.7361
Epoch 7/12
60000/60000 [==============================] - 203s 3ms/step - loss: 1.1257 - acc: 0.6198 - val_lo
ss: 0.7717 - val_acc: 0.7590
Epoch 8/12
60000/60000 [==============================] - 201s 3ms/step - loss: 1.0774 - acc: 0.6371 - val_lo
ss: 0.7279 - val_acc: 0.7727
Epoch 9/12
60000/60000 [==============================] - 201s 3ms/step - loss: 1.0466 - acc: 0.6480 - val_lo
ss: 0.6988 - val_acc: 0.7753
Epoch 10/12
60000/60000 [==============================] - 201s 3ms/step - loss: 1.0118 - acc: 0.6608 - val_lo
```

```
ss: 0.6802 - val_acc: 0.7812
Epoch 11/12
60000/60000 [==============================] - 202s 3ms/step - loss: 0.9749 - acc: 0.6733 - val_lo
ss: 0.6573 - val_acc: 0.7904
Epoch 12/12
60000/60000 [==============================] - 201s 3ms/step - loss: 0.9535 - acc: 0.6839 - val_lo
ss: 0.6511 - val_acc: 0.8009
Time taken : 0:40:27.634331
```

In [23]:

```
#evaluating model
score=convnet5.evaluate(x_test,y_test,verbose=0)
test_score5=score[0]
test_accuracy5=score[1]
train_accuracy5=max(convnet5_history.history['acc'])
print('test score :',test_score5)
print('test Accuracy :',test_accuracy5)
# error plot
x=list(range(1,epochs+1))
vy=convnet5_history.history['val_loss'] #validation loss
ty=convnet5_history.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
test score : 0.6510871462345124
test Accuracy : 0.8009
```



3 Model3:CNN with 7 ConvNet & kernel_size=(2x2)

5 convNet followed by maxpooling(2,2) and dropout

Stack all the layers

In [24]:

```
convnet7=Sequential() # Initializing the model

# First ConvNet
convnet7.add(Conv2D(16,kernel_size=(2,2),
                    activation='relu',
                    padding='same',strides=(1,1),
                    input_shape=input_shape))


convnet7.add(Conv2D(32,kernel_size=(2,2),
                    padding='same',strides=(2,2),
                    activation='relu'))#Second Convnet
#convnet7.add(MaxPooling2D(pool_size=(2,2)))
#convnet7.add(Dropout(0.25))

convnet7.add(Conv2D(64,kernel_size=(2,2),
                    padding='same',
                   activation='relu'))  # 3rd ConvNet
#maxpooling by (2,2 ) ,dropout,flattening
```

```
#convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.15))

convnet7.add(Conv2D(96,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#fourth Convnet
convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.39))
convnet7.add(Conv2D(128,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#fifth Convnet
convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.3))
convnet7.add(Conv2D(164,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#sixth Convnet
convnet7.add(Conv2D(164,kernel_size=(2,2),
                    padding='same',strides=(1,1),
                    activation='relu'))#seventh Convnet

convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.4))
convnet7.add(Flatten())

#hidden_layer
convnet7.add(Dense(256,
                   activation='relu',
                   kernel_initializer=he_normal(seed=None)))#1 hidden layer
convnet7.add(BatchNormalization())
convnet7.add(Dropout(0.5))
convnet7.add(Dense(148,
                   activation='relu',
                   kernel_initializer=he_normal(seed=None)))#2 hidden layer
convnet7.add(BatchNormalization())
convnet7.add(Dropout(0.5))
convnet7.add(Dense(128,
                   activation='relu',
                   kernel_initializer=he_normal(seed=None)))#3 hidden layer
convnet7.add(BatchNormalization())
convnet7.add(Dropout(0.5))
convnet7.add(Dense(num_classes,activation='softmax'))
print(convnet7.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_28 (Conv2D)           (None, 28, 28, 16)        80
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 14, 14, 32)        2080
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 14, 14, 64)        8256
_________________________________________________________________
dropout_26 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 14, 14, 96)        24672
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 7, 7, 96)          0
_________________________________________________________________
dropout_27 (Dropout)         (None, 7, 7, 96)          0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 7, 7, 128)         49280
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_28 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 3, 3, 164)         84132
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 3, 3, 164)         107748
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 1, 1, 164)         0
_________________________________________________________________
dropout_29 (Dropout)         (None, 1, 1, 164)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 164)               0
_________________________________________________________________
dense_12 (Dense)             (None, 256)               42240
```

```
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dropout_30 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_13 (Dense)             (None, 148)               38036
_________________________________________________________________
batch_normalization_4 (Batch (None, 148)               592
_________________________________________________________________
dropout_31 (Dropout)         (None, 148)               0
_________________________________________________________________
dense_14 (Dense)             (None, 128)               19072
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
dropout_32 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                1290
=================================================================
Total params: 379,014
Trainable params: 377,950
Non-trainable params: 1,064
_________________________________________________________________
None
```

Model compile and fit

In [25]:

```
start=datetime.now()
convnet7.compile(optimizer=keras.optimizers.Adam(),
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
convnet7_history=convnet7.fit(x_train,y_train,batch_size=batch_size,
                              epochs=epochs,
                              verbose=1,
                              validation_data=(x_test, y_test))
print("Time taken :", datetime.now() - start)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 42s 697us/step - loss: 1.9734 - acc: 0.3520 - val_l
oss: 1.1919 - val_acc: 0.5475
Epoch 2/12
60000/60000 [==============================] - 41s 679us/step - loss: 0.4882 - acc: 0.8531 - val_l
oss: 0.0946 - val_acc: 0.9739
Epoch 3/12
60000/60000 [==============================] - 41s 679us/step - loss: 0.2169 - acc: 0.9439 - val_l
oss: 0.0596 - val_acc: 0.9847
Epoch 4/12
60000/60000 [==============================] - 41s 682us/step - loss: 0.1488 - acc: 0.9629 - val_l
oss: 0.0515 - val_acc: 0.9875
Epoch 5/12
60000/60000 [==============================] - 40s 672us/step - loss: 0.1238 - acc: 0.9696 - val_l
oss: 0.0469 - val_acc: 0.9883
Epoch 6/12
60000/60000 [==============================] - 40s 674us/step - loss: 0.1073 - acc: 0.9735 - val_l
oss: 0.0362 - val_acc: 0.9909
Epoch 7/12
60000/60000 [==============================] - 40s 673us/step - loss: 0.0943 - acc: 0.9773 - val_l
oss: 0.0388 - val_acc: 0.9903
Epoch 8/12
60000/60000 [==============================] - 40s 673us/step - loss: 0.0836 - acc: 0.9794 - val_l
oss: 0.0401 - val_acc: 0.9900
Epoch 9/12
60000/60000 [==============================] - 40s 672us/step - loss: 0.0828 - acc: 0.9799 - val_l
oss: 0.0344 - val_acc: 0.9919
Epoch 10/12
60000/60000 [==============================] - 40s 666us/step - loss: 0.0734 - acc: 0.9819 - val_l
oss: 0.0332 - val_acc: 0.9906
Epoch 11/12
60000/60000 [==============================] - 39s 645us/step - loss: 0.0684 - acc: 0.9833 - val_l
oss: 0.0351 - val_acc: 0.9920
Epoch 12/12
```

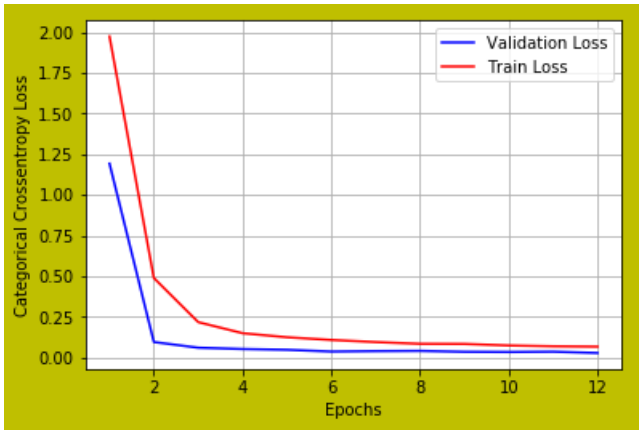
```
Epoch 12/12
60000/60000 [==============================] - 39s 645us/step - loss: 0.0664 - acc: 0.9837 - val_l
oss: 0.0272 - val_acc: 0.9939
Time taken : 0:08:04.429963
```

Evaluating Model 3

In [26]:

```
#evaluating model
score=convnet7.evaluate(x_test,y_test,verbose=0)
test_score7=score[0]
test_accuracy7=score[1]
train_accuracy7=max(convnet7_history.history['acc'])
print('test score :',test_score7)
print('test Accuracy :',test_accuracy7)
# error plot
x=list(range(1,epochs+1))
vy=convnet7_history.history['val_loss'] #validation loss
ty=convnet7_history.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
test score : 0.027164920003572478
test Accuracy : 0.9939
```



Changing dropout layer to 0.4,optimizer to Adadelta and weight initializer to 'random_uniform'

In [27]:

```
convnet7=Sequential() # Initializing the model

# First ConvNet
convnet7.add(Conv2D(16,kernel_size=(2,2),
                    activation='relu',
                    padding='same',strides=(1,1),
                    input_shape=input_shape))


convnet7.add(Conv2D(32,kernel_size=(2,2),
                    padding='same',strides=(2,2),
                    activation='relu'))#Second Convnet
#convnet7.add(MaxPooling2D(pool_size=(2,2)))
#convnet7.add(Dropout(0.25))

convnet7.add(Conv2D(64,kernel_size=(2,2),
                    padding='same',
                   activation='relu'))  # 3rd ConvNet
#maxpooling by (2,2 ) ,dropout,flattening
#convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.4))

convnet7.add(Conv2D(96,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#fourth Convnet
```

```
                    activation='relu'))#fourth Convnet
convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.4))
convnet7.add(Conv2D(128,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#fifth Convnet
convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.4))
convnet7.add(Conv2D(164,kernel_size=(2,2),
                    padding='same',
                    activation='relu'))#sixth Convnet
convnet7.add(Conv2D(164,kernel_size=(2,2),
                    padding='same',strides=(1,1),
                    activation='relu'))#seventh Convnet

convnet7.add(MaxPooling2D(pool_size=(2,2)))
convnet7.add(Dropout(0.4))
convnet7.add(Flatten())

#hidden_layer
convnet7.add(Dense(256,
                   activation='relu',
                   kernel_initializer='random_uniform'))#1 hidden layer
convnet7.add(BatchNormalization())
convnet7.add(Dropout(0.5))
convnet7.add(Dense(148,
                   activation='relu',
                   kernel_initializer='random_uniform'))#2 hidden layer
convnet7.add(BatchNormalization())
convnet7.add(Dropout(0.4))
convnet7.add(Dense(128,
                   activation='relu',
                   kernel_initializer='random_uniform'))#3 hidden layer
convnet7.add(BatchNormalization())
convnet7.add(Dropout(0.4))
convnet7.add(Dense(num_classes,activation='softmax'))
print(convnet7.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_35 (Conv2D)           (None, 28, 28, 16)        80
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 14, 14, 32)        2080
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 14, 14, 64)        8256
_________________________________________________________________
dropout_33 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 14, 14, 96)        24672
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 7, 7, 96)          0
_________________________________________________________________
dropout_34 (Dropout)         (None, 7, 7, 96)          0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 7, 7, 128)         49280
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_35 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 3, 3, 164)         84132
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 3, 3, 164)         107748
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 1, 1, 164)         0
_________________________________________________________________
dropout_36 (Dropout)         (None, 1, 1, 164)         0
_________________________________________________________________
flatten_8 (Flatten)          (None, 164)               0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               42240
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
dropout_37 (Dropout)         (None, 256)               0
```

```
_________________________________________________________________
dense_17 (Dense)             (None, 148)               38036
_________________________________________________________________
batch_normalization_7 (Batch (None, 148)               592
_________________________________________________________________
dropout_38 (Dropout)         (None, 148)               0
_________________________________________________________________
dense_18 (Dense)             (None, 128)               19072
_________________________________________________________________
batch_normalization_8 (Batch (None, 128)               512
_________________________________________________________________
dropout_39 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                1290
=================================================================
Total params: 379,014
Trainable params: 377,950
Non-trainable params: 1,064
_________________________________________________________________
None
```

In [28]:

```
start=datetime.now()
convnet7.compile(optimizer=keras.optimizers.Adadelta(),
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
convnet7_history=convnet7.fit(x_train,y_train,batch_size=batch_size,
                              epochs=epochs,
                              verbose=1,
                              validation_data=(x_test, y_test))
print("Time taken :", datetime.now() - start)
```

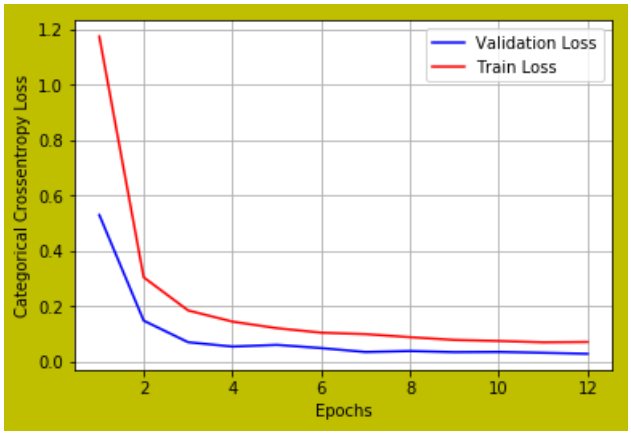
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 43s 710us/step - loss: 1.1741 - acc: 0.6207 - val_l
oss: 0.5301 - val_acc: 0.8582
Epoch 2/12
60000/60000 [==============================] - 41s 684us/step - loss: 0.3050 - acc: 0.9144 - val_l
oss: 0.1487 - val_acc: 0.9572
Epoch 3/12
60000/60000 [==============================] - 41s 684us/step - loss: 0.1855 - acc: 0.9494 - val_l
oss: 0.0708 - val_acc: 0.9798
Epoch 4/12
60000/60000 [==============================] - 41s 684us/step - loss: 0.1451 - acc: 0.9613 - val_l
oss: 0.0550 - val_acc: 0.9855
Epoch 5/12
60000/60000 [==============================] - 41s 686us/step - loss: 0.1214 - acc: 0.9676 - val_l
oss: 0.0612 - val_acc: 0.9834
Epoch 6/12
60000/60000 [==============================] - 41s 686us/step - loss: 0.1052 - acc: 0.9718 - val_l
oss: 0.0495 - val_acc: 0.9868
Epoch 7/12
60000/60000 [==============================] - 41s 690us/step - loss: 0.0999 - acc: 0.9735 - val_l
oss: 0.0354 - val_acc: 0.9907
Epoch 8/12
60000/60000 [==============================] - 42s 695us/step - loss: 0.0889 - acc: 0.9769 - val_l
oss: 0.0390 - val_acc: 0.9895
Epoch 9/12
60000/60000 [==============================] - 42s 694us/step - loss: 0.0790 - acc: 0.9792 - val_l
oss: 0.0349 - val_acc: 0.9900
Epoch 10/12
60000/60000 [==============================] - 41s 691us/step - loss: 0.0752 - acc: 0.9797 - val_l
oss: 0.0357 - val_acc: 0.9908
Epoch 11/12
60000/60000 [==============================] - 42s 693us/step - loss: 0.0703 - acc: 0.9814 - val_l
oss: 0.0326 - val_acc: 0.9916
Epoch 12/12
60000/60000 [==============================] - 42s 696us/step - loss: 0.0716 - acc: 0.9815 - val_l
oss: 0.0289 - val_acc: 0.9928
Time taken : 0:08:18.632244
```

In [29]:

```
#evaluating model
```

```
score=convnet7.evaluate(x_test,y_test,verbose=0)
test_score7=score[0]
test_accuracy7=score[1]
train_accuracy7=max(convnet7_history.history['acc'])
print('test score :',test_score7)
print('test Accuracy :',test_accuracy7)
# error plot
x=list(range(1,epochs+1))
vy=convnet7_history.history['val_loss'] #validation loss
ty=convnet7_history.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
test score : 0.028929874771251342
test Accuracy : 0.9928
```



Summarizing the performance of all the above models using PrettyTable

In [34]:

```
from prettytable import PrettyTable
models=['3ConvNet with kernel 3x3',
        '3ConvNet with kernel 3x3' ,
        '5ConvNet with kernel 5x5',
        '5ConvNet with kernel 5x5',
        '7ConvNet with kernel 2x2',
        '7ConvNet with kernel 2x2']
training_accuracy=[0.9956,0.9848,0.9832,0.6839,0.9837,0.9815]
test_accuracy=[0.9936,0.9885,0.9915,0.8,0.9939,0.9928]
dropouts = [0.25,0.5,0.25,0.5,0.5,0.4]
wt_init = ['he_normal','glorot_uniform','he_normal','glorot_uniform','he_normal','random_uniform']
optimizers = ['Adam','RMSprop','Adam','Adadelta','Adam','Adadelta']
INDEX = [1,2,3,4,5,6]
# Initializing prettytable
Model_Performance = PrettyTable()
# Adding columns
Model_Performance.add_column("INDEX.",INDEX)
Model_Performance.add_column("MODEL_NAME",models)
Model_Performance.add_column("Dropout",dropouts)
Model_Performance.add_column("weight initalizer",wt_init)
Model_Performance.add_column("optimizer",optimizers)
Model_Performance.add_column("TRAIN ACCURACY",training_accuracy)
Model_Performance.add_column("TEST ACCURACY",test_accuracy)


# Printing the Model_Performance
print(Model_Performance)
```

```
+--------+--------------------------+---------+------------------+-----------+----------------+---
---------+
| INDEX. |        MODEL_NAME        | Dropout | weight initalizer | optimizer | TRAIN ACCURACY | TE
ST ACCURACY |
+--------+--------------------------+---------+------------------+-----------+----------------+---
---------+
|   1    | 3ConvNet with kernel 3x3 |   0.25  |     he_normal     |    Adam   |     0.9956     |
0.9936     |
```

```
|   2    | 3ConvNet with kernel 3x3 |   0.5   |   glorot_uniform  |  RMSprop  |     0.9848     |
0.9885    |
|   3    | 5ConvNet with kernel 5x5 |   0.25  |     he_normal     |    Adam   |     0.9832     |
0.9915    |
|   4    | 5ConvNet with kernel 5x5 |   0.5   |   glorot_uniform  |  Adadelta |     0.6839     |
0.8       |
|   5    | 7ConvNet with kernel 2x2 |   0.5   |     he_normal     |    Adam   |     0.9837     |
0.9939    |
|   6    | 7ConvNet with kernel 2x2 |   0.4   |   random_uniform  |  Adadelta |     0.9815     |
0.9928    |
+--------+--------------------------+---------+-------------------+-----------+----------------+---
---------+
```

Conclusions

- In this assignment,we chose three models to train and experiemented with different weight initializers,optimizers and activation functions
- From the Validation/Test accuracy scores,we can find that model 3 has performed slighly better than model 1 with the highest test accuracy of 0.9939
- Model 2 took the longest for training as compared to Model 1 and Model 3
- Model 3 took the least amount of time to train