In [7]:

```python
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

In [8]:

```python
batch_size = 128
num_classes = 10
epochs = 10

# 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':
  #https://machinelearningmastery.com/a-gentle-introduction-to-channels-first-and-channels-last-image-formats-for-deep-learning/
    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://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz (https://storage.googleapis.com/ten
sorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 3s 0us/step
```

In [9]:

```python
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255 #normalizing
x_test /= 255 #normalizing
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 [14]:

```python
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic(x, vy, ty, ax, colors=['b']):
  ax.plot(x, vy, 'b', label="Validation Loss")
  ax.plot(x, ty, 'r', label="Train Loss")
  plt.legend()
  plt.grid()
  fig.canvas.draw()
```

In [15]:

```python
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
#model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 conv2d_1 (Conv2D)           (None, 24, 24, 128)       36992

 conv2d_2 (Conv2D)           (None, 22, 22, 64)        73792

 flatten (Flatten)           (None, 30976)             0

 dense (Dense)               (None, 10)                309770

=================================================================
Total params: 420,874
Trainable params: 420,874
Non-trainable params: 0
_________________________________________________________________
```

In [16]:

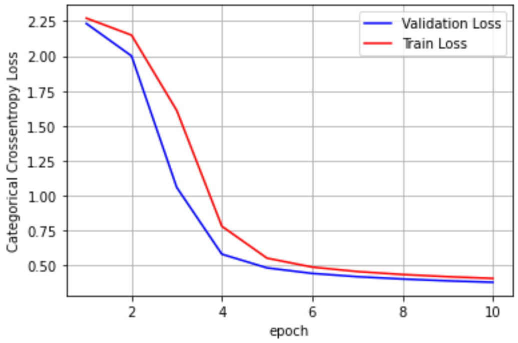
```python
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

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])
```

```
Epoch 1/10
469/469 [==============================] - 257s 542ms/step - loss: 2.2711 - accuracy: 0.3253 - val_loss: 2.2329 - val_accuracy:
0.4806
Epoch 2/10
469/469 [==============================] - 240s 511ms/step - loss: 2.1498 - accuracy: 0.5391 - val_loss: 2.0008 - val_accuracy:
0.6362
Epoch 3/10
469/469 [==============================] - 240s 513ms/step - loss: 1.6109 - accuracy: 0.7204 - val_loss: 1.0603 - val_accuracy:
0.7987
Epoch 4/10
469/469 [==============================] - 242s 516ms/step - loss: 0.7814 - accuracy: 0.8094 - val_loss: 0.5816 - val_accuracy:
0.8428
Epoch 5/10
469/469 [==============================] - 253s 540ms/step - loss: 0.5531 - accuracy: 0.8418 - val_loss: 0.4835 - val_accuracy:
0.8627
Epoch 6/10
469/469 [==============================] - 278s 592ms/step - loss: 0.4891 - accuracy: 0.8579 - val_loss: 0.4437 - val_accuracy:
0.8719
Epoch 7/10
469/469 [==============================] - 273s 583ms/step - loss: 0.4569 - accuracy: 0.8667 - val_loss: 0.4202 - val_accuracy:
0.8787
Epoch 8/10
469/469 [==============================] - 245s 523ms/step - loss: 0.4359 - accuracy: 0.8733 - val_loss: 0.4036 - val_accuracy:
0.8840
Epoch 9/10
469/469 [==============================] - 265s 566ms/step - loss: 0.4202 - accuracy: 0.8778 - val_loss: 0.3905 - val_accuracy:
0.8888
Epoch 10/10
469/469 [==============================] - 256s 546ms/step - loss: 0.4079 - accuracy: 0.8817 - val_loss: 0.3803 - val_accuracy:
0.8919
Test loss: 0.38033363223075867
Test accuracy: 0.8919000029563904
```

In [17]:

```
import matplotlib.pyplot as plt
%matplotlib inline
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.38033363223075867
Test accuracy: 0.8919000029563904
```



In [18]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_3 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 conv2d_4 (Conv2D)           (None, 24, 24, 128)       36992     
                                                                 
 conv2d_5 (Conv2D)           (None, 22, 22, 64)        73792     
                                                                 
 conv2d_6 (Conv2D)           (None, 20, 20, 32)        18464     
                                                                 
 flatten_1 (Flatten)         (None, 12800)             0         
                                                                 
 dense_1 (Dense)             (None, 10)                128010    
                                                                 
=================================================================
Total params: 257,578
Trainable params: 257,578
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

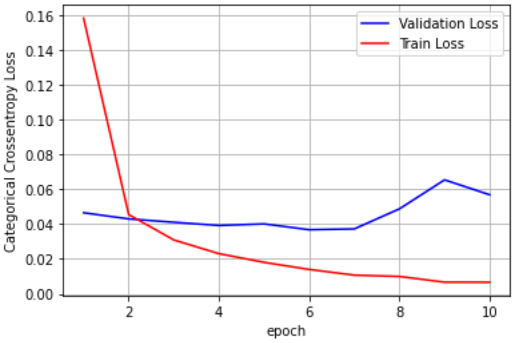
```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

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])
```

```
Epoch 1/10
469/469 [==============================] - 376s 770ms/step - loss: 0.1581 - accuracy: 0.9538 - val_loss: 0.0465 - val_accuracy:
0.9837
Epoch 2/10
469/469 [==============================] - 308s 656ms/step - loss: 0.0454 - accuracy: 0.9861 - val_loss: 0.0430 - val_accuracy:
0.9863
Epoch 3/10
469/469 [==============================] - 301s 641ms/step - loss: 0.0309 - accuracy: 0.9902 - val_loss: 0.0410 - val_accuracy:
0.9870
Epoch 4/10
469/469 [==============================] - 300s 639ms/step - loss: 0.0230 - accuracy: 0.9924 - val_loss: 0.0392 - val_accuracy:
0.9892
Epoch 5/10
469/469 [==============================] - 312s 665ms/step - loss: 0.0181 - accuracy: 0.9942 - val_loss: 0.0401 - val_accuracy:
0.9887
Epoch 6/10
469/469 [==============================] - 402s 856ms/step - loss: 0.0140 - accuracy: 0.9952 - val_loss: 0.0368 - val_accuracy:
0.9888
Epoch 7/10
469/469 [==============================] - 351s 748ms/step - loss: 0.0107 - accuracy: 0.9963 - val_loss: 0.0372 - val_accuracy:
0.9894
Epoch 8/10
469/469 [==============================] - 2942s 6s/step - loss: 0.0100 - accuracy: 0.9966 - val_loss: 0.0487 - val_accuracy:
0.9881
Epoch 9/10
469/469 [==============================] - 318s 677ms/step - loss: 0.0067 - accuracy: 0.9978 - val_loss: 0.0654 - val_accuracy:
0.9855
Epoch 10/10
469/469 [==============================] - 313s 667ms/step - loss: 0.0066 - accuracy: 0.9977 - val_loss: 0.0568 - val_accuracy:
0.9867
Test loss: 0.05681750550866127
Test accuracy: 0.9866999983787537
```

In [20]:

```
import matplotlib.pyplot as plt
%matplotlib inline
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.05681750550866127
Test accuracy: 0.9866999983787537
```



In [22]:

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

model = Sequential()
model.add(Conv2D(32, kernel_size=(5,5),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_7 (Conv2D)           (None, 24, 24, 32)        832       
                                                                 
 conv2d_8 (Conv2D)           (None, 20, 20, 64)        51264     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 10, 10, 64)       0         
 )                                                               
                                                                 
 batch_normalization (BatchN  (None, 10, 10, 64)       256       
 ormalization)                                                   
                                                                 
 dropout (Dropout)           (None, 10, 10, 64)        0         
                                                                 
 conv2d_9 (Conv2D)           (None, 6, 6, 64)          102464    
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 3, 3, 64)         0         
 2D)                                                             
                                                                 
 batch_normalization_1 (Batc  (None, 3, 3, 64)         256       
 hNormalization)                                                 
                                                                 
 dropout_1 (Dropout)         (None, 3, 3, 64)          0         
                                                                 
 flatten_2 (Flatten)         (None, 576)               0         
                                                                 
 dense_2 (Dense)             (None, 10)                5770      
                                                                 
=================================================================
Total params: 160,842
Trainable params: 160,586
Non-trainable params: 256
_________________________________________________________________
```

In [23]:

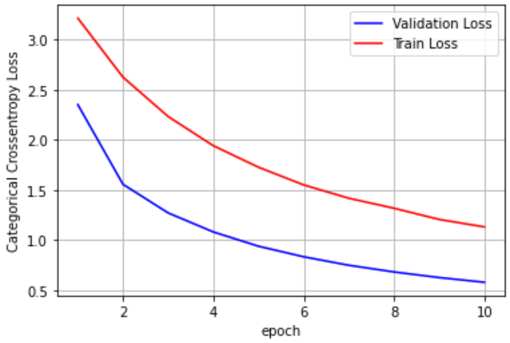
```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

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])
```

```
Epoch 1/10
469/469 [==============================] - 557s 789ms/step - loss: 3.2142 - accuracy: 0.1488 - val_loss: 2.3515 - val_accuracy:
0.2316
Epoch 2/10
469/469 [==============================] - 139s 297ms/step - loss: 2.6254 - accuracy: 0.2350 - val_loss: 1.5569 - val_accuracy:
0.4987
Epoch 3/10
469/469 [==============================] - 136s 291ms/step - loss: 2.2328 - accuracy: 0.3181 - val_loss: 1.2712 - val_accuracy:
0.5992
Epoch 4/10
469/469 [==============================] - 130s 277ms/step - loss: 1.9420 - accuracy: 0.3928 - val_loss: 1.0817 - val_accuracy:
0.6659
Epoch 5/10
469/469 [==============================] - 125s 267ms/step - loss: 1.7272 - accuracy: 0.4520 - val_loss: 0.9391 - val_accuracy:
0.7147
Epoch 6/10
469/469 [==============================] - 129s 274ms/step - loss: 1.5509 - accuracy: 0.5020 - val_loss: 0.8337 - val_accuracy:
0.7529
Epoch 7/10
469/469 [==============================] - 130s 277ms/step - loss: 1.4179 - accuracy: 0.5419 - val_loss: 0.7492 - val_accuracy:
0.7830
Epoch 8/10
469/469 [==============================] - 144s 307ms/step - loss: 1.3182 - accuracy: 0.5712 - val_loss: 0.6826 - val_accuracy:
0.8048
Epoch 9/10
469/469 [==============================] - 147s 312ms/step - loss: 1.2064 - accuracy: 0.6068 - val_loss: 0.6264 - val_accuracy:
0.8224
Epoch 10/10
469/469 [==============================] - 137s 291ms/step - loss: 1.1318 - accuracy: 0.6298 - val_loss: 0.5796 - val_accuracy:
0.8364
Test loss: 0.5795929431915283
Test accuracy: 0.8363999724388123
```

In [24]:

```
import matplotlib.pyplot as plt
%matplotlib inline
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.5795929431915283
Test accuracy: 0.8363999724388123
```



In [25]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2,2),
                 strides=(2,2),
                 padding='same',
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

model.summary()
```

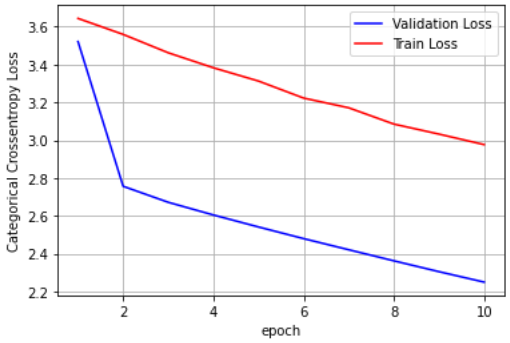
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_10 (Conv2D)          (None, 14, 14, 32)        160       
                                                                 
 conv2d_11 (Conv2D)          (None, 13, 13, 64)        8256      
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 6, 6, 64)         0         
 2D)                                                             
                                                                 
 batch_normalization_2 (Batc  (None, 6, 6, 64)         256       
 hNormalization)                                                 
                                                                 
 dropout_2 (Dropout)         (None, 6, 6, 64)          0         
                                                                 
 conv2d_12 (Conv2D)          (None, 5, 5, 64)          16448     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 2, 2, 64)         0         
 2D)                                                             
                                                                 
 batch_normalization_3 (Batc  (None, 2, 2, 64)         256       
 hNormalization)                                                 
                                                                 
 dropout_3 (Dropout)         (None, 2, 2, 64)          0         
                                                                 
 flatten_3 (Flatten)         (None, 256)               0         
                                                                 
 dense_3 (Dense)             (None, 10)                2570      
                                                                 
=================================================================
Total params: 27,946
Trainable params: 27,690
Non-trainable params: 256
_________________________________________________________________
```

In [26]:
```
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])
```

```
Epoch 1/10
469/469 [==============================] - 85s 132ms/step - loss: 3.6439 - accuracy: 0.1099 - val_loss: 3.5213 - val_accuracy:
0.0902
Epoch 2/10
469/469 [==============================] - 31s 66ms/step - loss: 3.5594 - accuracy: 0.1156 - val_loss: 2.7564 - val_accuracy:
0.0813
Epoch 3/10
469/469 [==============================] - 30s 64ms/step - loss: 3.4622 - accuracy: 0.1222 - val_loss: 2.6716 - val_accuracy:
0.1039
Epoch 4/10
469/469 [==============================] - 31s 66ms/step - loss: 3.3829 - accuracy: 0.1291 - val_loss: 2.6045 - val_accuracy:
0.1298
Epoch 5/10
469/469 [==============================] - 30s 65ms/step - loss: 3.3127 - accuracy: 0.1352 - val_loss: 2.5411 - val_accuracy:
0.1499
Epoch 6/10
469/469 [==============================] - 31s 66ms/step - loss: 3.2227 - accuracy: 0.1437 - val_loss: 2.4795 - val_accuracy:
0.1694
Epoch 7/10
469/469 [==============================] - 29s 63ms/step - loss: 3.1713 - accuracy: 0.1508 - val_loss: 2.4207 - val_accuracy:
0.1817
Epoch 8/10
469/469 [==============================] - 30s 65ms/step - loss: 3.0852 - accuracy: 0.1587 - val_loss: 2.3621 - val_accuracy:
0.1933
Epoch 9/10
469/469 [==============================] - 25s 54ms/step - loss: 3.0322 - accuracy: 0.1666 - val_loss: 2.3047 - val_accuracy:
0.2062
Epoch 10/10
469/469 [==============================] - 26s 55ms/step - loss: 2.9765 - accuracy: 0.1759 - val_loss: 2.2495 - val_accuracy:
0.2164
Test loss: 2.249525308609009
Test accuracy: 0.21639999747276306
```

In [28]:
```
import matplotlib.pyplot as plt
%matplotlib inline
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 2.249525308609009
Test accuracy: 0.21639999747276306
```



In [29]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(7,7),
                 padding='valid',
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(128, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv2D(64, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_13 (Conv2D)          (None, 22, 22, 32)        1600      
                                                                 
 conv2d_14 (Conv2D)          (None, 16, 16, 128)       200832    
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 8, 8, 128)        0         
 2D)                                                             
                                                                 
 batch_normalization_4 (Batc  (None, 8, 8, 128)        512       
 hNormalization)                                                 
                                                                 
 dropout_4 (Dropout)         (None, 8, 8, 128)         0         
                                                                 
 conv2d_15 (Conv2D)          (None, 2, 2, 64)          401472    
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 1, 1, 64)         0         
 2D)                                                             
                                                                 
 batch_normalization_5 (Batc  (None, 1, 1, 64)         256       
 hNormalization)                                                 
                                                                 
 dropout_5 (Dropout)         (None, 1, 1, 64)          0         
                                                                 
 flatten_4 (Flatten)         (None, 64)                0         
                                                                 
 dense_4 (Dense)             (None, 10)                650       
                                                                 
=================================================================
Total params: 605,322
Trainable params: 604,938
Non-trainable params: 384
_________________________________________________________________
```

In [30]:

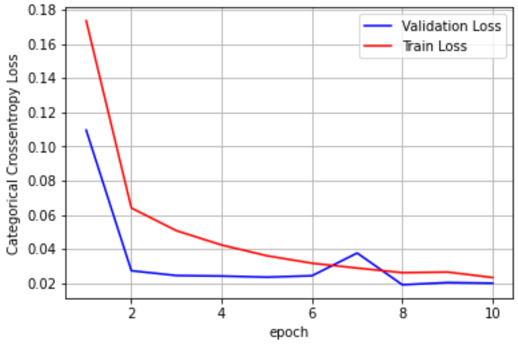
```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

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])
```

```
Epoch 1/10
469/469 [==============================] - 276s 574ms/step - loss: 0.1735 - accuracy: 0.9499 - val_loss: 0.1095 - val_accuracy:
0.9825
Epoch 2/10
469/469 [==============================] - 267s 569ms/step - loss: 0.0640 - accuracy: 0.9810 - val_loss: 0.0273 - val_accuracy:
0.9910
Epoch 3/10
469/469 [==============================] - 252s 537ms/step - loss: 0.0507 - accuracy: 0.9852 - val_loss: 0.0245 - val_accuracy:
0.9914
Epoch 4/10
469/469 [==============================] - 256s 546ms/step - loss: 0.0424 - accuracy: 0.9873 - val_loss: 0.0242 - val_accuracy:
0.9922
Epoch 5/10
469/469 [==============================] - 258s 551ms/step - loss: 0.0361 - accuracy: 0.9892 - val_loss: 0.0236 - val_accuracy:
0.9921
Epoch 6/10
469/469 [==============================] - 266s 567ms/step - loss: 0.0317 - accuracy: 0.9902 - val_loss: 0.0244 - val_accuracy:
0.9923
Epoch 7/10
469/469 [==============================] - 282s 602ms/step - loss: 0.0289 - accuracy: 0.9911 - val_loss: 0.0377 - val_accuracy:
0.9886
Epoch 8/10
469/469 [==============================] - 307s 654ms/step - loss: 0.0262 - accuracy: 0.9918 - val_loss: 0.0191 - val_accuracy:
0.9934
Epoch 9/10
469/469 [==============================] - 293s 625ms/step - loss: 0.0266 - accuracy: 0.9917 - val_loss: 0.0204 - val_accuracy:
0.9936
Epoch 10/10
469/469 [==============================] - 303s 646ms/step - loss: 0.0234 - accuracy: 0.9929 - val_loss: 0.0201 - val_accuracy:
0.9935
Test loss: 0.020070811733603477
Test accuracy: 0.9934999942779541
```

In [31]:

```
import matplotlib.pyplot as plt
%matplotlib inline
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
# list of epoch numbers
x = list(range(1,epochs+1))
# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy
# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.020070811733603477
Test accuracy: 0.9934999942779541
```



In [40]:

```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names=["Model","#Hidden Layers","Kernel-Size","MaxPooling","Dropout/BatchNormalization","Optimizer","Activation","Accurac
x.add_row(["1.","2(128-64)", "3X3","False","False","Adadelta","ReLu","0.989"])
x.add_row(["2.","3(128-64-32)", "3X3","False","False","Adam","ReLu","0.988"])
x.add_row(["3.","2(64-64)", "5X5","2X2","True","Adadelta","ReLu","0.993"])
x.add_row(["4.","2(64-64)", "2X2(s=2,p='same')","2X2","True","Adadelta","ReLu","0.983"])
x.add_row(["5.","2(128-64)", "7X7(s=2,p='valid')","2X2","True","Adam","ReLu","0.992"])
print(x)
```

```
+-------+----------------+--------------------+------------+----------------------------+-----------+------------+----------+
| Model | #Hidden Layers |    Kernel-Size     | MaxPooling | Dropout/BatchNormalization | Optimizer | Activation | Accuracy |
+-------+----------------+--------------------+------------+----------------------------+-----------+------------+----------+
|   1.  |   2(128-64)    |        3X3         |   False    |           False            |  Adadelta |    ReLu    |  0.989   |
|   2.  |  3(128-64-32)  |        3X3         |   False    |           False            |    Adam   |    ReLu    |  0.988   |
|   3.  |    2(64-64)    |        5X5         |    2X2     |            True            |  Adadelta |    ReLu    |  0.993   |
|   4.  |    2(64-64)    | 2X2(s=2,p='same')  |    2X2     |            True            |  Adadelta |    ReLu    |  0.983   |
|   5.  |   2(128-64)    | 7X7(s=2,p='valid') |    2X2     |            True            |    Adam   |    ReLu    |  0.992   |
+-------+----------------+--------------------+------------+----------------------------+-----------+------------+----------+
```