In [1]:
```
# if you keras is not using tensorflow as backend set "KERAS_BACKEND=tensorflow" use this command
from keras.utils import np_utils
from keras.datasets import mnist
import seaborn as sns
from keras.initializers import RandomNormal
```

```
Using TensorFlow backend.
```

In [0]:
```
%matplotlib notebook
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 [3]:
```
# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz (https://s3.amazonaws.com/img-datasets/mnist.npz)
11493376/11490434 [==============================] - 2s 0us/step
```

In [4]:
```
print("Number of training examples :", X_train.shape[0], "and each image is of shape (%d, %d)"%(X_train.shape[1], X_train.shape[2]))
print("Number of training examples :", X_test.shape[0], "and each image is of shape (%d, %d)"%(X_test.shape[1], X_test.shape[2]))
```

```
Number of training examples : 60000 and each image is of shape (28, 28)
Number of training examples : 10000 and each image is of shape (28, 28)
```

In [0]:
```
# if you observe the input shape its 2 dimensional vector
# for each image we have a (28*28) vector
# we will convert the (28*28) vector into single dimensional vector of 1 * 784

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1]*X_train.shape[2])
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1]*X_test.shape[2])
```

In [6]:
```
# after converting the input images from 3d to 2d vectors

print("Number of training examples :", X_train.shape[0], "and each image is of shape (%d)"%(X_train.shape[1]))
print("Number of training examples :", X_test.shape[0], "and each image is of shape (%d)"%(X_test.shape[1]))
```

```
Number of training examples : 60000 and each image is of shape (784)
Number of training examples : 10000 and each image is of shape (784)
```

In [7]:
```python
# An example data point
print(X_train[0])
```

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   3  18  18  18 126 136 175  26 166 255
 247 127   0   0   0   0   0   0   0   0   0   0   0   0  30  36  94 154
 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0   0   0
   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82
  82  56  39   0   0   0   0   0   0   0   0   0   0   0   0  18 219 253
 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0  11 190 253  70   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  35 241
 225 160 108   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
 253 207   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253
 253 201  78   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0  18 171 219 253 253 253 253 195
  80   9   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0 136 253 253 253 212 135 132  16
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
```

In [0]:
```
# if we observe the above matrix each cell is having a value between 0-255
# before we move to apply machine learning algorithms lets try to normalize the data
# X => (X - Xmin)/(Xmax-Xmin) = X/255

X_train = X_train/255
X_test = X_test/255
```

In [9]:
```
# example data point after normlizing
print(X_train[0])
```

```
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```

In [10]:
```
# here we are having a class number for each image
print("Class label of first image :", y_train[0])

# lets convert this into a 10 dimensional vector
# ex: consider an image is 5 convert it into 5 => [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
# this conversion needed for MLPs

Y_train = np_utils.to_categorical(y_train, 10)
Y_test = np_utils.to_categorical(y_test, 10)

print("After converting the output into a vector : ",Y_train[0])
```

```
Class label of first image : 5
After converting the output into a vector :  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

In [0]:
```
# some model parameters

output_dim = 10
input_dim = X_train.shape[1]

batch_size = 128
nb_epoch = 50
```

MLP + ReLU Without Batch Normalization & Dropout (With 2 hidden layers)

In [33]:
```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.initializers import he_normal
from keras.layers import Dense, Activation
from keras.layers import Dropout

model_one = Sequential()

model_one.add(Dense(456, activation='relu', input_shape=(input_dim,), kernel_initializer=he_normal(seed=None)))

model_one.add(Dense(248, activation='relu', kernel_initializer=he_normal(seed=None)) )

model_one.add(Dense(output_dim, activation='softmax'))

model_one.summary()
```

```
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.

Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_27 (Dense)             (None, 456)               357960    
_________________________________________________________________
dense_28 (Dense)             (None, 248)               113336    
_________________________________________________________________
dense_29 (Dense)             (None, 10)                2490      
=================================================================
Total params: 473,786
Trainable params: 473,786
Non-trainable params: 0
_________________________________________________________________
```

In [34]:
```
model_one.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model_one.fit(X_train, Y_train, batch_size=batch_size, epochs=50, verbose=1, validation_data=(X_test, Y_test))
```

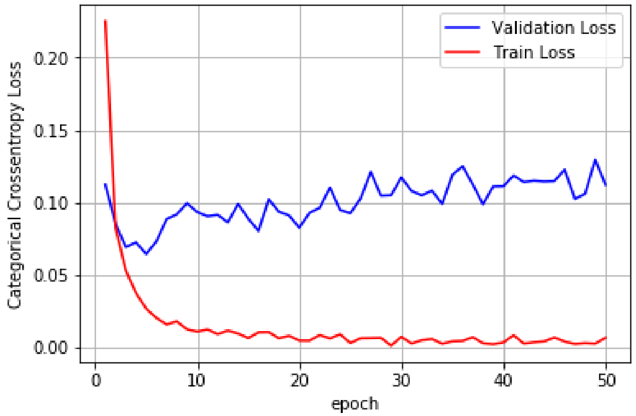
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.2251 - acc: 0.9334 - val_loss: 0.1123 - val_acc: 0.9671
Epoch 2/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0824 - acc: 0.9746 - val_loss: 0.0852 - val_acc: 0.9715
Epoch 3/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0531 - acc: 0.9833 - val_loss: 0.0693 - val_acc: 0.9772
Epoch 4/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0375 - acc: 0.9882 - val_loss: 0.0725 - val_acc: 0.9790
Epoch 5/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0267 - acc: 0.9914 - val_loss: 0.0644 - val_acc: 0.9816
Epoch 6/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0203 - acc: 0.9936 - val_loss: 0.0732 - val_acc: 0.9781
Epoch 7/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0158 - acc: 0.9948 - val_loss: 0.0885 - val_acc: 0.9772
Epoch 8/50
60000/60000 [==============================] - 3s 52us/step - loss: 0.0180 - acc: 0.9941 - val_loss: 0.0918 - val_acc: 0.9770
Epoch 9/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0124 - acc: 0.9960 - val_loss: 0.0995 - val_acc: 0.9782
Epoch 10/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0109 - acc: 0.9961 - val_loss: 0.0934 - val_acc: 0.9779
Epoch 11/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0123 - acc: 0.9958 - val_loss: 0.0905 - val_acc: 0.9797
Epoch 12/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0092 - acc: 0.9970 - val_loss: 0.0915 - val_acc: 0.9793
Epoch 13/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0116 - acc: 0.9962 - val_loss: 0.0863 - val_acc: 0.9812
Epoch 14/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0096 - acc: 0.9969 - val_loss: 0.0990 - val_acc: 0.9771
Epoch 15/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0064 - acc: 0.9978 - val_loss: 0.0889 - val_acc: 0.9828
Epoch 16/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0104 - acc: 0.9967 - val_loss: 0.0803 - val_acc: 0.9826
Epoch 17/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0106 - acc: 0.9965 - val_loss: 0.1023 - val_acc: 0.9788
Epoch 18/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0063 - acc: 0.9980 - val_loss: 0.0937 - val_acc: 0.9822
Epoch 19/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0080 - acc: 0.9974 - val_loss: 0.0911 - val_acc: 0.9829
Epoch 20/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0047 - acc: 0.9985 - val_loss: 0.0826 - val_acc: 0.9840
Epoch 21/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0047 - acc: 0.9987 - val_loss: 0.0929 - val_acc: 0.9811
Epoch 22/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0085 - acc: 0.9972 - val_loss: 0.0964 - val_acc: 0.9818
Epoch 23/50
```

```
60000/60000 [==============================] - 3s 54us/step - loss: 0.0062 - acc: 0.9981 - val_loss: 0.1103 - val_acc: 0.9811
Epoch 24/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0090 - acc: 0.9974 - val_loss: 0.0948 - val_acc: 0.9829
Epoch 25/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0030 - acc: 0.9991 - val_loss: 0.0926 - val_acc: 0.9827
Epoch 26/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0063 - acc: 0.9976 - val_loss: 0.1027 - val_acc: 0.9819
Epoch 27/50
60000/60000 [==============================] - 3s 51us/step - loss: 0.0064 - acc: 0.9979 - val_loss: 0.1211 - val_acc: 0.9796
Epoch 28/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0066 - acc: 0.9978 - val_loss: 0.1047 - val_acc: 0.9814
Epoch 29/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0013 - acc: 0.9996 - val_loss: 0.1050 - val_acc: 0.9828
Epoch 30/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0073 - acc: 0.9977 - val_loss: 0.1173 - val_acc: 0.9818
Epoch 31/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0027 - acc: 0.9990 - val_loss: 0.1080 - val_acc: 0.9822
Epoch 32/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0049 - acc: 0.9984 - val_loss: 0.1049 - val_acc: 0.9829
Epoch 33/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0058 - acc: 0.9983 - val_loss: 0.1082 - val_acc: 0.9812
Epoch 34/50
60000/60000 [==============================] - 3s 52us/step - loss: 0.0025 - acc: 0.9992 - val_loss: 0.0990 - val_acc: 0.9836
Epoch 35/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0042 - acc: 0.9989 - val_loss: 0.1191 - val_acc: 0.9802
Epoch 36/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0045 - acc: 0.9988 - val_loss: 0.1250 - val_acc: 0.9794
Epoch 37/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0070 - acc: 0.9979 - val_loss: 0.1125 - val_acc: 0.9819
Epoch 38/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0029 - acc: 0.9992 - val_loss: 0.0987 - val_acc: 0.9837
Epoch 39/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0022 - acc: 0.9993 - val_loss: 0.1111 - val_acc: 0.9828
Epoch 40/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0034 - acc: 0.9988 - val_loss: 0.1113 - val_acc: 0.9831
Epoch 41/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0084 - acc: 0.9978 - val_loss: 0.1186 - val_acc: 0.9804
Epoch 42/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0026 - acc: 0.9992 - val_loss: 0.1142 - val_acc: 0.9813
Epoch 43/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0036 - acc: 0.9990 - val_loss: 0.1151 - val_acc: 0.9821
Epoch 44/50
60000/60000 [==============================] - 3s 52us/step - loss: 0.0041 - acc: 0.9988 - val_loss: 0.1145 - val_acc: 0.9820
Epoch 45/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0069 - acc: 0.9979 - val_loss: 0.1148 - val_acc: 0.9825
Epoch 46/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0040 - acc: 0.9988 - val_loss: 0.1228 - val_acc: 0.9824
Epoch 47/50
60000/60000 [==============================] - 3s 52us/step - loss: 0.0024 - acc: 0.9994 - val_loss: 0.1025 - val_acc: 0.9838
Epoch 48/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0030 - acc: 0.9993 - val_loss: 0.1060 - val_acc: 0.9839
```

```
Epoch 49/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0026 - acc: 0.9992 - val_loss: 0.1297 - val_acc: 0.9815
Epoch 50/50
60000/60000 [==============================] - 3s 53us/step - loss: 0.0067 - acc: 0.9980 - val_loss: 0.1123 - val_acc: 0.9818
```

In [35]:
```
%matplotlib inline
score = model_one.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# 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: 0.11229222209781944
Test accuracy: 0.9818
```



MLP + ReLU + Batch Normalization + Dropout (With 2 hidden layers)

In [36]:
```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Activation
from keras.layers import Dropout

model = Sequential()

model.add(Dense(456, activation='relu', input_shape=(input_dim,), kernel_initializer=he_normal(seed=None)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(248, activation='relu', kernel_initializer=he_normal(seed=None)) )
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_30 (Dense)             (None, 456)               357960    
_________________________________________________________________
batch_normalization_11 (Batc (None, 456)               1824      
_________________________________________________________________
dropout_11 (Dropout)         (None, 456)               0         
_________________________________________________________________
dense_31 (Dense)             (None, 248)               113336    
_________________________________________________________________
batch_normalization_12 (Batc (None, 248)               992       
_________________________________________________________________
dropout_12 (Dropout)         (None, 248)               0         
_________________________________________________________________
dense_32 (Dense)             (None, 10)                2490      
=================================================================
Total params: 476,602
Trainable params: 475,194
Non-trainable params: 1,408
_________________________________________________________________
```

In [37]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=50, verbose=1, validation_data=(X_test, Y_test))
```

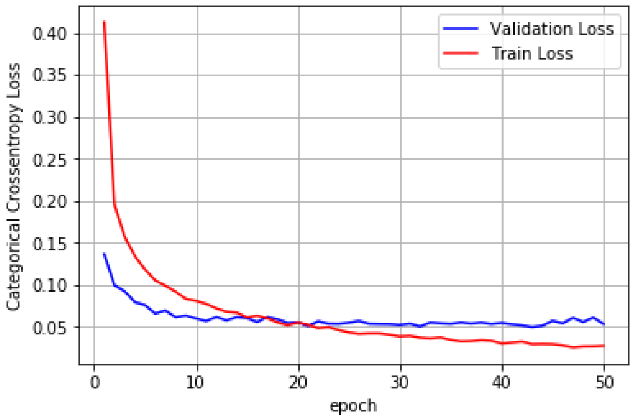
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 7s 117us/step - loss: 0.4123 - acc: 0.8744 - val_loss: 0.1364 - val_acc: 0.9566
Epoch 2/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.1955 - acc: 0.9409 - val_loss: 0.0995 - val_acc: 0.9686
Epoch 3/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.1572 - acc: 0.9525 - val_loss: 0.0922 - val_acc: 0.9713
Epoch 4/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.1341 - acc: 0.9578 - val_loss: 0.0793 - val_acc: 0.9733
Epoch 5/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.1182 - acc: 0.9633 - val_loss: 0.0757 - val_acc: 0.9748
Epoch 6/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.1051 - acc: 0.9670 - val_loss: 0.0657 - val_acc: 0.9801
Epoch 7/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0988 - acc: 0.9689 - val_loss: 0.0694 - val_acc: 0.9779
Epoch 8/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0919 - acc: 0.9714 - val_loss: 0.0613 - val_acc: 0.9808
Epoch 9/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.0831 - acc: 0.9732 - val_loss: 0.0633 - val_acc: 0.9801
Epoch 10/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0808 - acc: 0.9742 - val_loss: 0.0598 - val_acc: 0.9824
Epoch 11/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.0772 - acc: 0.9757 - val_loss: 0.0567 - val_acc: 0.9809
Epoch 12/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0718 - acc: 0.9769 - val_loss: 0.0616 - val_acc: 0.9793
Epoch 13/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0679 - acc: 0.9781 - val_loss: 0.0574 - val_acc: 0.9818
Epoch 14/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0669 - acc: 0.9786 - val_loss: 0.0614 - val_acc: 0.9810
Epoch 15/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0611 - acc: 0.9805 - val_loss: 0.0602 - val_acc: 0.9802
Epoch 16/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0631 - acc: 0.9794 - val_loss: 0.0555 - val_acc: 0.9830
Epoch 17/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0600 - acc: 0.9803 - val_loss: 0.0617 - val_acc: 0.9822
Epoch 18/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0551 - acc: 0.9815 - val_loss: 0.0588 - val_acc: 0.9828
Epoch 19/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0517 - acc: 0.9835 - val_loss: 0.0544 - val_acc: 0.9843
Epoch 20/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0549 - acc: 0.9822 - val_loss: 0.0552 - val_acc: 0.9837
Epoch 21/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0524 - acc: 0.9833 - val_loss: 0.0508 - val_acc: 0.9854
Epoch 22/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0484 - acc: 0.9837 - val_loss: 0.0563 - val_acc: 0.9841
Epoch 23/50
```

```
60000/60000 [==============================] - 5s 90us/step - loss: 0.0495 - acc: 0.9839 - val_loss: 0.0536 - val_acc: 0.9844
Epoch 24/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0460 - acc: 0.9845 - val_loss: 0.0533 - val_acc: 0.9840
Epoch 25/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0433 - acc: 0.9856 - val_loss: 0.0547 - val_acc: 0.9850
Epoch 26/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0416 - acc: 0.9860 - val_loss: 0.0567 - val_acc: 0.9844
Epoch 27/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.0421 - acc: 0.9861 - val_loss: 0.0534 - val_acc: 0.9846
Epoch 28/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.0422 - acc: 0.9859 - val_loss: 0.0532 - val_acc: 0.9855
Epoch 29/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0405 - acc: 0.9866 - val_loss: 0.0530 - val_acc: 0.9853
Epoch 30/50
60000/60000 [==============================] - 5s 92us/step - loss: 0.0387 - acc: 0.9868 - val_loss: 0.0521 - val_acc: 0.9847
Epoch 31/50
60000/60000 [==============================] - 6s 92us/step - loss: 0.0392 - acc: 0.9871 - val_loss: 0.0536 - val_acc: 0.9849
Epoch 32/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.0370 - acc: 0.9880 - val_loss: 0.0504 - val_acc: 0.9865
Epoch 33/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0362 - acc: 0.9881 - val_loss: 0.0550 - val_acc: 0.9856
Epoch 34/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0373 - acc: 0.9876 - val_loss: 0.0540 - val_acc: 0.9858
Epoch 35/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.0344 - acc: 0.9885 - val_loss: 0.0534 - val_acc: 0.9860
Epoch 36/50
60000/60000 [==============================] - 6s 92us/step - loss: 0.0326 - acc: 0.9893 - val_loss: 0.0550 - val_acc: 0.9854
Epoch 37/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0329 - acc: 0.9888 - val_loss: 0.0537 - val_acc: 0.9859
Epoch 38/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0341 - acc: 0.9891 - val_loss: 0.0549 - val_acc: 0.9857
Epoch 39/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0333 - acc: 0.9889 - val_loss: 0.0532 - val_acc: 0.9860
Epoch 40/50
60000/60000 [==============================] - 6s 92us/step - loss: 0.0300 - acc: 0.9902 - val_loss: 0.0545 - val_acc: 0.9857
Epoch 41/50
60000/60000 [==============================] - 6s 92us/step - loss: 0.0310 - acc: 0.9894 - val_loss: 0.0528 - val_acc: 0.9853
Epoch 42/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0323 - acc: 0.9896 - val_loss: 0.0516 - val_acc: 0.9864
Epoch 43/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0292 - acc: 0.9901 - val_loss: 0.0496 - val_acc: 0.9867
Epoch 44/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0294 - acc: 0.9901 - val_loss: 0.0512 - val_acc: 0.9853
Epoch 45/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0291 - acc: 0.9905 - val_loss: 0.0570 - val_acc: 0.9849
Epoch 46/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0277 - acc: 0.9908 - val_loss: 0.0539 - val_acc: 0.9849
Epoch 47/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.0254 - acc: 0.9915 - val_loss: 0.0606 - val_acc: 0.9840
Epoch 48/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0265 - acc: 0.9913 - val_loss: 0.0555 - val_acc: 0.9856
```

```
Epoch 49/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0266 - acc: 0.9911 - val_loss: 0.0610 - val_acc: 0.9842
Epoch 50/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.0272 - acc: 0.9907 - val_loss: 0.0532 - val_acc: 0.9863
```

In [38]:
```
%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,nb_epoch+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: 0.05319240672139513
Test accuracy: 0.9863
```



MLP + ReLU without Batch Normalization & Dropout (With 3 hidden layers)

In [39]:

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Activation
from keras.layers import Dropout

model_three = Sequential()

model_three.add(Dense(368, activation='relu', input_shape=(input_dim,), kernel_initializer=he_normal(seed=None)))

model_three.add(Dense(224, activation='relu', kernel_initializer=he_normal(seed=None)) )

model_three.add(Dense(132, activation='relu', kernel_initializer=he_normal(seed=None)) )


model_three.add(Dense(output_dim, activation='softmax'))

model_three.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_33 (Dense)             (None, 368)               288880    
_________________________________________________________________
dense_34 (Dense)             (None, 224)               82656     
_________________________________________________________________
dense_35 (Dense)             (None, 132)               29700     
_________________________________________________________________
dense_36 (Dense)             (None, 10)                1330      
=================================================================
Total params: 402,566
Trainable params: 402,566
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
model_three.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model_three.fit(X_train, Y_train, batch_size=batch_size, epochs=50, verbose=1, validation_data=(X_test, Y_test))
```

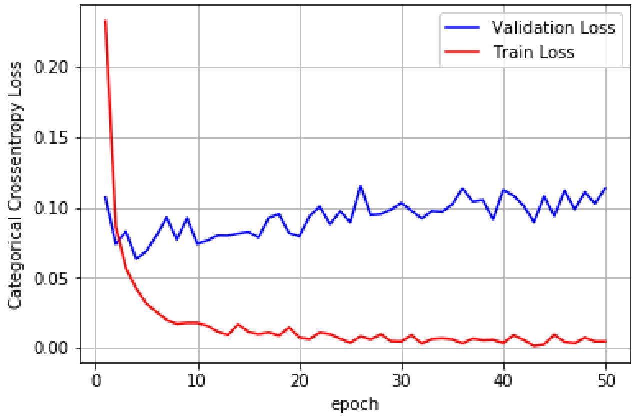
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 5s 81us/step - loss: 0.2326 - acc: 0.9320 - val_loss: 0.1069 - val_acc: 0.9681
Epoch 2/50
60000/60000 [==============================] - 3s 57us/step - loss: 0.0864 - acc: 0.9732 - val_loss: 0.0738 - val_acc: 0.9755
Epoch 3/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0566 - acc: 0.9827 - val_loss: 0.0828 - val_acc: 0.9742
Epoch 4/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0421 - acc: 0.9869 - val_loss: 0.0631 - val_acc: 0.9802
Epoch 5/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0312 - acc: 0.9900 - val_loss: 0.0689 - val_acc: 0.9806
Epoch 6/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0253 - acc: 0.9916 - val_loss: 0.0798 - val_acc: 0.9761
Epoch 7/50
60000/60000 [==============================] - 3s 57us/step - loss: 0.0197 - acc: 0.9935 - val_loss: 0.0929 - val_acc: 0.9740
Epoch 8/50
60000/60000 [==============================] - 3s 58us/step - loss: 0.0169 - acc: 0.9943 - val_loss: 0.0769 - val_acc: 0.9785
Epoch 9/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0177 - acc: 0.9942 - val_loss: 0.0924 - val_acc: 0.9767
Epoch 10/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0176 - acc: 0.9942 - val_loss: 0.0739 - val_acc: 0.9799
Epoch 11/50
60000/60000 [==============================] - 3s 58us/step - loss: 0.0155 - acc: 0.9951 - val_loss: 0.0763 - val_acc: 0.9813
Epoch 12/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0112 - acc: 0.9962 - val_loss: 0.0799 - val_acc: 0.9814
Epoch 13/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0090 - acc: 0.9970 - val_loss: 0.0797 - val_acc: 0.9813
Epoch 14/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0165 - acc: 0.9945 - val_loss: 0.0812 - val_acc: 0.9814
Epoch 15/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0111 - acc: 0.9961 - val_loss: 0.0824 - val_acc: 0.9828
Epoch 16/50
60000/60000 [==============================] - 3s 52us/step - loss: 0.0094 - acc: 0.9970 - val_loss: 0.0783 - val_acc: 0.9799
Epoch 17/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0108 - acc: 0.9967 - val_loss: 0.0924 - val_acc: 0.9796
Epoch 18/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0085 - acc: 0.9974 - val_loss: 0.0951 - val_acc: 0.9791
Epoch 19/50
60000/60000 [==============================] - 3s 57us/step - loss: 0.0142 - acc: 0.9955 - val_loss: 0.0816 - val_acc: 0.9814
Epoch 20/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0072 - acc: 0.9977 - val_loss: 0.0791 - val_acc: 0.9835
Epoch 21/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0059 - acc: 0.9983 - val_loss: 0.0935 - val_acc: 0.9811
Epoch 22/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0107 - acc: 0.9965 - val_loss: 0.1006 - val_acc: 0.9791
Epoch 23/50
```

```
60000/60000 [==============================] - 3s 54us/step - loss: 0.0095 - acc: 0.9969 - val_loss: 0.0879 - val_acc: 0.9814
Epoch 24/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0063 - acc: 0.9979 - val_loss: 0.0971 - val_acc: 0.9809
Epoch 25/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0034 - acc: 0.9989 - val_loss: 0.0892 - val_acc: 0.9837
Epoch 26/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0080 - acc: 0.9975 - val_loss: 0.1151 - val_acc: 0.9786
Epoch 27/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0058 - acc: 0.9982 - val_loss: 0.0945 - val_acc: 0.9807
Epoch 28/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0094 - acc: 0.9975 - val_loss: 0.0951 - val_acc: 0.9815
Epoch 29/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0046 - acc: 0.9985 - val_loss: 0.0983 - val_acc: 0.9829
Epoch 30/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0043 - acc: 0.9988 - val_loss: 0.1030 - val_acc: 0.9817
Epoch 31/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0089 - acc: 0.9972 - val_loss: 0.0976 - val_acc: 0.9832
Epoch 32/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0029 - acc: 0.9992 - val_loss: 0.0919 - val_acc: 0.9839
Epoch 33/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0061 - acc: 0.9983 - val_loss: 0.0973 - val_acc: 0.9824
Epoch 34/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0067 - acc: 0.9978 - val_loss: 0.0967 - val_acc: 0.9822
Epoch 35/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0059 - acc: 0.9982 - val_loss: 0.1021 - val_acc: 0.9825
Epoch 36/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0030 - acc: 0.9990 - val_loss: 0.1133 - val_acc: 0.9807
Epoch 37/50
60000/60000 [==============================] - 3s 58us/step - loss: 0.0064 - acc: 0.9982 - val_loss: 0.1039 - val_acc: 0.9834
Epoch 38/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0052 - acc: 0.9986 - val_loss: 0.1052 - val_acc: 0.9819
Epoch 39/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0056 - acc: 0.9983 - val_loss: 0.0913 - val_acc: 0.9846
Epoch 40/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0033 - acc: 0.9989 - val_loss: 0.1122 - val_acc: 0.9815
Epoch 41/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0087 - acc: 0.9978 - val_loss: 0.1082 - val_acc: 0.9819
Epoch 42/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0055 - acc: 0.9985 - val_loss: 0.1011 - val_acc: 0.9825
Epoch 43/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0013 - acc: 0.9997 - val_loss: 0.0894 - val_acc: 0.9854
Epoch 44/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0023 - acc: 0.9995 - val_loss: 0.1079 - val_acc: 0.9828
Epoch 45/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0091 - acc: 0.9976 - val_loss: 0.0937 - val_acc: 0.9842
Epoch 46/50
60000/60000 [==============================] - 3s 56us/step - loss: 0.0040 - acc: 0.9990 - val_loss: 0.1119 - val_acc: 0.9820
Epoch 47/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0031 - acc: 0.9992 - val_loss: 0.0986 - val_acc: 0.9842
Epoch 48/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0071 - acc: 0.9981 - val_loss: 0.1107 - val_acc: 0.9809
```

```
Epoch 49/50
60000/60000 [==============================] - 3s 54us/step - loss: 0.0044 - acc: 0.9987 - val_loss: 0.1025 - val_acc: 0.9831
Epoch 50/50
60000/60000 [==============================] - 3s 55us/step - loss: 0.0044 - acc: 0.9988 - val_loss: 0.1135 - val_acc: 0.9817
```

In [41]:
```
%matplotlib inline
score = model_three.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# 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: 0.1135252954211466
Test accuracy: 0.9817
```



MLP + ReLU + Batch Normalization + Dropout (With 3 hidden layers)

In [21]:
```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Activation
from keras.layers import Dropout

model_two = Sequential()

model_two.add(Dense(368, activation='relu', input_shape=(input_dim,), kernel_initializer=RandomNormal(mean=0.0, stddev=0.039, seed=None)))
model_two.add(BatchNormalization())
model_two.add(Dropout(0.5))

model_two.add(Dense(224, activation='relu', kernel_initializer=RandomNormal(mean=0.0, stddev=0.55, seed=None)) )
model_two.add(BatchNormalization())
model_two.add(Dropout(0.5))

model_two.add(Dense(132, activation='relu', kernel_initializer=RandomNormal(mean=0.0, stddev=0.55, seed=None)) )
model_two.add(BatchNormalization())
model_two.add(Dropout(0.5))

model_two.add(Dense(output_dim, activation='softmax'))

model_two.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_11 (Dense)             (None, 368)               288880    
_________________________________________________________________
batch_normalization_3 (Batch (None, 368)               1472      
_________________________________________________________________
dropout_3 (Dropout)          (None, 368)               0         
_________________________________________________________________
dense_12 (Dense)             (None, 224)               82656     
_________________________________________________________________
batch_normalization_4 (Batch (None, 224)               896       
_________________________________________________________________
dropout_4 (Dropout)          (None, 224)               0         
_________________________________________________________________
dense_13 (Dense)             (None, 132)               29700     
_________________________________________________________________
batch_normalization_5 (Batch (None, 132)               528       
_________________________________________________________________
dropout_5 (Dropout)          (None, 132)               0         
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1330      
=================================================================
Total params: 405,462
Trainable params: 404,014
```

```
Non-trainable params: 1,448
_________________________________________________________________
```

In [22]:
```
!pip install plotly --upgrade
```

```
Collecting plotly
  Downloading https://files.pythonhosted.org/packages/f7/05/3c32c6bc85acbd30a18fbc3ba732fed5e48e5f8fd60d2a148877970f4a61/plotly-4.2.1-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/f7/05/3c32c6bc85acbd30a18fbc3ba732fed5e48e5f8fd60d2a148877970f4a61/plotly-4.2.1-py2.py3-none-any.whl) (7.2MB)
     |████████████████████████████████| 7.2MB 28kB/s
Requirement already satisfied, skipping upgrade: six in /usr/local/lib/python3.6/dist-packages (from plotly) (1.12.0)
Requirement already satisfied, skipping upgrade: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly) (1.3.3)
Installing collected packages: plotly
  Found existing installation: plotly 4.1.1
    Uninstalling plotly-4.1.1:
      Successfully uninstalled plotly-4.1.1
Successfully installed plotly-4.2.1
```

In [23]:
```
model_two.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model_two.fit(X_train, Y_train, batch_size=batch_size, epochs=50, verbose=1, validation_data=(X_test, Y_test))
```

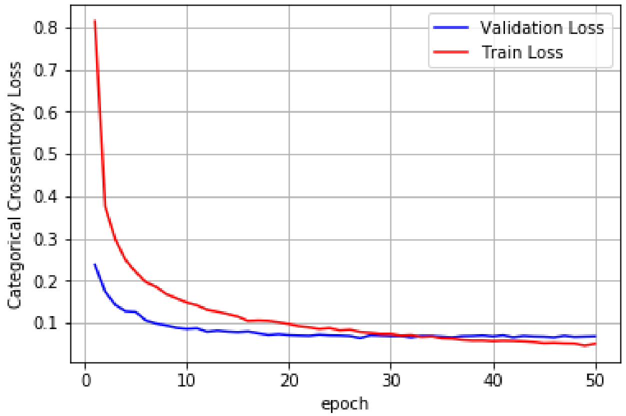
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 7s 117us/step - loss: 0.8143 - acc: 0.7470 - val_loss: 0.2372 - val_acc: 0.9265
Epoch 2/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.3755 - acc: 0.8894 - val_loss: 0.1737 - val_acc: 0.9462
Epoch 3/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.2974 - acc: 0.9131 - val_loss: 0.1423 - val_acc: 0.9543
Epoch 4/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.2486 - acc: 0.9262 - val_loss: 0.1271 - val_acc: 0.9601
Epoch 5/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.2201 - acc: 0.9352 - val_loss: 0.1255 - val_acc: 0.9631
Epoch 6/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1961 - acc: 0.9415 - val_loss: 0.1051 - val_acc: 0.9692
Epoch 7/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.1852 - acc: 0.9476 - val_loss: 0.0980 - val_acc: 0.9703
Epoch 8/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1678 - acc: 0.9518 - val_loss: 0.0934 - val_acc: 0.9729
Epoch 9/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1578 - acc: 0.9539 - val_loss: 0.0880 - val_acc: 0.9753
Epoch 10/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1480 - acc: 0.9564 - val_loss: 0.0854 - val_acc: 0.9760
Epoch 11/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.1413 - acc: 0.9582 - val_loss: 0.0869 - val_acc: 0.9755
Epoch 12/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1307 - acc: 0.9610 - val_loss: 0.0784 - val_acc: 0.9782
Epoch 13/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1260 - acc: 0.9634 - val_loss: 0.0810 - val_acc: 0.9770
Epoch 14/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.1206 - acc: 0.9646 - val_loss: 0.0787 - val_acc: 0.9795
Epoch 15/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.1145 - acc: 0.9661 - val_loss: 0.0775 - val_acc: 0.9790
Epoch 16/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.1042 - acc: 0.9692 - val_loss: 0.0791 - val_acc: 0.9769
Epoch 17/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.1054 - acc: 0.9689 - val_loss: 0.0749 - val_acc: 0.9784
Epoch 18/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1045 - acc: 0.9692 - val_loss: 0.0709 - val_acc: 0.9808
Epoch 19/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.1010 - acc: 0.9699 - val_loss: 0.0729 - val_acc: 0.9793
Epoch 20/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0969 - acc: 0.9711 - val_loss: 0.0706 - val_acc: 0.9806
Epoch 21/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0917 - acc: 0.9725 - val_loss: 0.0696 - val_acc: 0.9801
Epoch 22/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0891 - acc: 0.9735 - val_loss: 0.0689 - val_acc: 0.9811
Epoch 23/50
```

```
60000/60000 [==============================] - 6s 98us/step - loss: 0.0852 - acc: 0.9739 - val_loss: 0.0717 - val_acc: 0.9814
Epoch 24/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0875 - acc: 0.9735 - val_loss: 0.0700 - val_acc: 0.9809
Epoch 25/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0819 - acc: 0.9747 - val_loss: 0.0696 - val_acc: 0.9814
Epoch 26/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.0836 - acc: 0.9751 - val_loss: 0.0684 - val_acc: 0.9809
Epoch 27/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.0780 - acc: 0.9758 - val_loss: 0.0635 - val_acc: 0.9818
Epoch 28/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0761 - acc: 0.9771 - val_loss: 0.0702 - val_acc: 0.9810
Epoch 29/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.0736 - acc: 0.9778 - val_loss: 0.0693 - val_acc: 0.9808
Epoch 30/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.0738 - acc: 0.9776 - val_loss: 0.0683 - val_acc: 0.9817
Epoch 31/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0695 - acc: 0.9788 - val_loss: 0.0686 - val_acc: 0.9827
Epoch 32/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0709 - acc: 0.9788 - val_loss: 0.0649 - val_acc: 0.9822
Epoch 33/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.0662 - acc: 0.9800 - val_loss: 0.0687 - val_acc: 0.9808
Epoch 34/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0676 - acc: 0.9797 - val_loss: 0.0684 - val_acc: 0.9820
Epoch 35/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0631 - acc: 0.9809 - val_loss: 0.0671 - val_acc: 0.9827
Epoch 36/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0623 - acc: 0.9812 - val_loss: 0.0652 - val_acc: 0.9831
Epoch 37/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.0593 - acc: 0.9816 - val_loss: 0.0682 - val_acc: 0.9824
Epoch 38/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0577 - acc: 0.9820 - val_loss: 0.0686 - val_acc: 0.9818
Epoch 39/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0578 - acc: 0.9823 - val_loss: 0.0699 - val_acc: 0.9809
Epoch 40/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0565 - acc: 0.9824 - val_loss: 0.0676 - val_acc: 0.9832
Epoch 41/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0573 - acc: 0.9821 - val_loss: 0.0704 - val_acc: 0.9826
Epoch 42/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0569 - acc: 0.9825 - val_loss: 0.0654 - val_acc: 0.9829
Epoch 43/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0558 - acc: 0.9828 - val_loss: 0.0686 - val_acc: 0.9822
Epoch 44/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0543 - acc: 0.9833 - val_loss: 0.0673 - val_acc: 0.9834
Epoch 45/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0514 - acc: 0.9837 - val_loss: 0.0668 - val_acc: 0.9824
Epoch 46/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0521 - acc: 0.9841 - val_loss: 0.0652 - val_acc: 0.9823
Epoch 47/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0510 - acc: 0.9841 - val_loss: 0.0690 - val_acc: 0.9828
Epoch 48/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.0506 - acc: 0.9841 - val_loss: 0.0662 - val_acc: 0.9833
```

```
Epoch 49/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0458 - acc: 0.9856 - val_loss: 0.0671 - val_acc: 0.9836
Epoch 50/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.0500 - acc: 0.9846 - val_loss: 0.0677 - val_acc: 0.9828
```

In [24]:
```
%matplotlib inline
score = model_two.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# 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: 0.06771488659109018
Test accuracy: 0.9828
```



MLP + ReLU without Batch Normalization & Dropout (With 5 hidden layers)

In [42]:
```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Activation
from keras.layers import Dropout

model_four = Sequential()

model_four.add(Dense(424, activation='relu', input_shape=(input_dim,), kernel_initializer=he_normal(seed=None)))

model_four.add(Dense(314, activation='relu', kernel_initializer=he_normal(seed=None)) )

model_four.add(Dense(256, activation='relu', kernel_initializer=he_normal(seed=None)) )

model_four.add(Dense(128, activation='relu', kernel_initializer=he_normal(seed=None)) )

model_four.add(Dense(64, activation='relu', kernel_initializer=he_normal(seed=None)) )

model_four.add(Dense(output_dim, activation='softmax'))

model_four.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_37 (Dense)             (None, 424)               332840    
_________________________________________________________________
dense_38 (Dense)             (None, 314)               133450    
_________________________________________________________________
dense_39 (Dense)             (None, 256)               80640     
_________________________________________________________________
dense_40 (Dense)             (None, 128)               32896     
_________________________________________________________________
dense_41 (Dense)             (None, 64)                8256      
_________________________________________________________________
dense_42 (Dense)             (None, 10)                650       
=================================================================
Total params: 588,732
Trainable params: 588,732
Non-trainable params: 0
_________________________________________________________________
```

In [43]:
```
model_four.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model_four.fit(X_train, Y_train, batch_size=batch_size, epochs=50, verbose=1, validation_data=(X_test, Y_test))
```

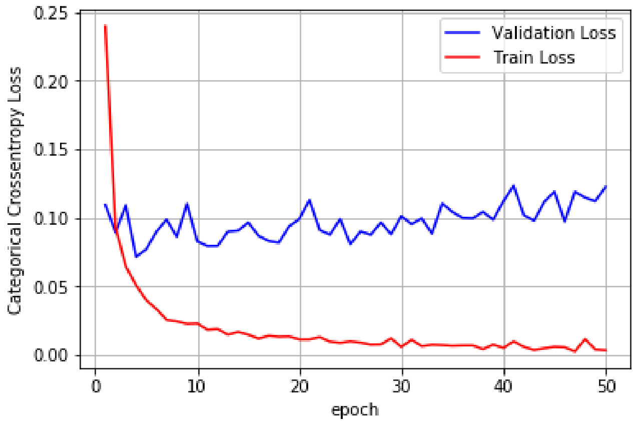
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.2399 - acc: 0.9263 - val_loss: 0.1095 - val_acc: 0.9658
Epoch 2/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0928 - acc: 0.9719 - val_loss: 0.0892 - val_acc: 0.9730
Epoch 3/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0645 - acc: 0.9794 - val_loss: 0.1092 - val_acc: 0.9668
Epoch 4/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0508 - acc: 0.9838 - val_loss: 0.0714 - val_acc: 0.9783
Epoch 5/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0398 - acc: 0.9872 - val_loss: 0.0771 - val_acc: 0.9785
Epoch 6/50
60000/60000 [==============================] - 4s 69us/step - loss: 0.0334 - acc: 0.9891 - val_loss: 0.0899 - val_acc: 0.9765
Epoch 7/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0255 - acc: 0.9920 - val_loss: 0.0988 - val_acc: 0.9758
Epoch 8/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0244 - acc: 0.9923 - val_loss: 0.0860 - val_acc: 0.9777
Epoch 9/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0226 - acc: 0.9928 - val_loss: 0.1103 - val_acc: 0.9718
Epoch 10/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0229 - acc: 0.9929 - val_loss: 0.0829 - val_acc: 0.9768
Epoch 11/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0183 - acc: 0.9942 - val_loss: 0.0794 - val_acc: 0.9804
Epoch 12/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0188 - acc: 0.9942 - val_loss: 0.0795 - val_acc: 0.9820
Epoch 13/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0147 - acc: 0.9954 - val_loss: 0.0899 - val_acc: 0.9814
Epoch 14/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0167 - acc: 0.9947 - val_loss: 0.0906 - val_acc: 0.9789
Epoch 15/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0147 - acc: 0.9956 - val_loss: 0.0966 - val_acc: 0.9791
Epoch 16/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0118 - acc: 0.9967 - val_loss: 0.0869 - val_acc: 0.9803
Epoch 17/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0139 - acc: 0.9959 - val_loss: 0.0831 - val_acc: 0.9805
Epoch 18/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0131 - acc: 0.9961 - val_loss: 0.0819 - val_acc: 0.9824
Epoch 19/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0134 - acc: 0.9960 - val_loss: 0.0937 - val_acc: 0.9804
Epoch 20/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0112 - acc: 0.9966 - val_loss: 0.0991 - val_acc: 0.9790
Epoch 21/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0112 - acc: 0.9967 - val_loss: 0.1131 - val_acc: 0.9768
Epoch 22/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0129 - acc: 0.9964 - val_loss: 0.0910 - val_acc: 0.9803
Epoch 23/50
```

```
60000/60000 [==============================] - 4s 72us/step - loss: 0.0096 - acc: 0.9975 - val_loss: 0.0877 - val_acc: 0.9825
Epoch 24/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0086 - acc: 0.9973 - val_loss: 0.0991 - val_acc: 0.9809
Epoch 25/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0098 - acc: 0.9972 - val_loss: 0.0807 - val_acc: 0.9846
Epoch 26/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0088 - acc: 0.9976 - val_loss: 0.0900 - val_acc: 0.9828
Epoch 27/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0074 - acc: 0.9978 - val_loss: 0.0876 - val_acc: 0.9830
Epoch 28/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0076 - acc: 0.9977 - val_loss: 0.0965 - val_acc: 0.9832
Epoch 29/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0120 - acc: 0.9964 - val_loss: 0.0880 - val_acc: 0.9824
Epoch 30/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0057 - acc: 0.9984 - val_loss: 0.1012 - val_acc: 0.9795
Epoch 31/50
60000/60000 [==============================] - 4s 69us/step - loss: 0.0109 - acc: 0.9971 - val_loss: 0.0953 - val_acc: 0.9817
Epoch 32/50
60000/60000 [==============================] - 4s 69us/step - loss: 0.0062 - acc: 0.9984 - val_loss: 0.0996 - val_acc: 0.9814
Epoch 33/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0073 - acc: 0.9980 - val_loss: 0.0883 - val_acc: 0.9843
Epoch 34/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0070 - acc: 0.9980 - val_loss: 0.1105 - val_acc: 0.9816
Epoch 35/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0065 - acc: 0.9981 - val_loss: 0.1043 - val_acc: 0.9802
Epoch 36/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0068 - acc: 0.9982 - val_loss: 0.1000 - val_acc: 0.9832
Epoch 37/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0068 - acc: 0.9983 - val_loss: 0.0997 - val_acc: 0.9822
Epoch 38/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0041 - acc: 0.9991 - val_loss: 0.1044 - val_acc: 0.9820
Epoch 39/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0074 - acc: 0.9981 - val_loss: 0.0987 - val_acc: 0.9826
Epoch 40/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0050 - acc: 0.9988 - val_loss: 0.1121 - val_acc: 0.9796
Epoch 41/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0097 - acc: 0.9973 - val_loss: 0.1234 - val_acc: 0.9745
Epoch 42/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0058 - acc: 0.9983 - val_loss: 0.1020 - val_acc: 0.9821
Epoch 43/50
60000/60000 [==============================] - 4s 72us/step - loss: 0.0035 - acc: 0.9989 - val_loss: 0.0979 - val_acc: 0.9830
Epoch 44/50
60000/60000 [==============================] - 4s 74us/step - loss: 0.0047 - acc: 0.9989 - val_loss: 0.1116 - val_acc: 0.9828
Epoch 45/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0058 - acc: 0.9987 - val_loss: 0.1192 - val_acc: 0.9819
Epoch 46/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0055 - acc: 0.9986 - val_loss: 0.0972 - val_acc: 0.9834
Epoch 47/50
60000/60000 [==============================] - 4s 71us/step - loss: 0.0024 - acc: 0.9994 - val_loss: 0.1191 - val_acc: 0.9822
Epoch 48/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0114 - acc: 0.9970 - val_loss: 0.1147 - val_acc: 0.9810
```

```
Epoch 49/50
60000/60000 [==============================] - 4s 72us/step - loss: 0.0038 - acc: 0.9990 - val_loss: 0.1121 - val_acc: 0.9820
Epoch 50/50
60000/60000 [==============================] - 4s 70us/step - loss: 0.0033 - acc: 0.9991 - val_loss: 0.1225 - val_acc: 0.9828
```

In [44]:
```
%matplotlib inline
score = model_four.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# 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: 0.12247530582656879
Test accuracy: 0.9828
```



MLP + ReLU + Batch Normalization + Dropout (With 5 hidden layers)

In [45]:
```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Activation
from keras.layers import Dropout

model_five = Sequential()

model_five.add(Dense(424, activation='relu', input_shape=(input_dim,), kernel_initializer=he_normal(seed=None)))
model_five.add(BatchNormalization())
model_five.add(Dropout(0.5))

model_five.add(Dense(314, activation='relu', kernel_initializer=he_normal(seed=None)) )
model_five.add(BatchNormalization())
model_five.add(Dropout(0.5))

model_five.add(Dense(256, activation='relu', kernel_initializer=he_normal(seed=None)) )
model_five.add(BatchNormalization())
model_five.add(Dropout(0.5))

model_five.add(Dense(128, activation='relu', kernel_initializer=he_normal(seed=None)) )
model_five.add(BatchNormalization())
model_five.add(Dropout(0.5))

model_five.add(Dense(64, activation='relu', kernel_initializer=he_normal(seed=None)) )
model_five.add(BatchNormalization())
model_five.add(Dropout(0.5))

model_five.add(Dense(output_dim, activation='softmax'))

model_five.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_43 (Dense)             (None, 424)               332840    
_________________________________________________________________
batch_normalization_13 (Batc (None, 424)               1696      
_________________________________________________________________
dropout_13 (Dropout)         (None, 424)               0         
_________________________________________________________________
dense_44 (Dense)             (None, 314)               133450    
_________________________________________________________________
batch_normalization_14 (Batc (None, 314)               1256      
_________________________________________________________________
dropout_14 (Dropout)         (None, 314)               0         
_________________________________________________________________
dense_45 (Dense)             (None, 256)               80640     
_________________________________________________________________
batch_normalization_15 (Batc (None, 256)               1024      
```

```
_________________________________________________________________
dropout_15 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_46 (Dense)             (None, 128)               32896     
_________________________________________________________________
batch_normalization_16 (Batc (None, 128)               512       
_________________________________________________________________
dropout_16 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_47 (Dense)             (None, 64)                8256      
_________________________________________________________________
batch_normalization_17 (Batc (None, 64)                256       
_________________________________________________________________
dropout_17 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_48 (Dense)             (None, 10)                650       
=================================================================
Total params: 593,476
Trainable params: 591,104
Non-trainable params: 2,372
_________________________________________________________________
```

In [46]:
```
model_five.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model_five.fit(X_train, Y_train, batch_size=batch_size, epochs=50, verbose=1, validation_data=(X_test, Y_test))
```

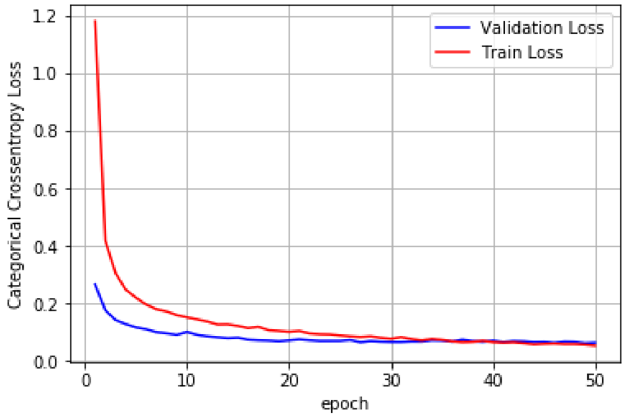
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 12s 202us/step - loss: 1.1809 - acc: 0.6278 - val_loss: 0.2667 - val_acc: 0.9225
Epoch 2/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.4186 - acc: 0.8829 - val_loss: 0.1771 - val_acc: 0.9510
Epoch 3/50
60000/60000 [==============================] - 9s 157us/step - loss: 0.3061 - acc: 0.9172 - val_loss: 0.1431 - val_acc: 0.9605
Epoch 4/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.2487 - acc: 0.9347 - val_loss: 0.1290 - val_acc: 0.9643
Epoch 5/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.2208 - acc: 0.9412 - val_loss: 0.1176 - val_acc: 0.9693
Epoch 6/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.1970 - acc: 0.9487 - val_loss: 0.1111 - val_acc: 0.9720
Epoch 7/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.1806 - acc: 0.9539 - val_loss: 0.1009 - val_acc: 0.9734
Epoch 8/50
60000/60000 [==============================] - 10s 165us/step - loss: 0.1729 - acc: 0.9555 - val_loss: 0.0970 - val_acc: 0.9743
Epoch 9/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1601 - acc: 0.9592 - val_loss: 0.0915 - val_acc: 0.9751
Epoch 10/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.1525 - acc: 0.9612 - val_loss: 0.1018 - val_acc: 0.9737
Epoch 11/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.1454 - acc: 0.9632 - val_loss: 0.0915 - val_acc: 0.9768
Epoch 12/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1378 - acc: 0.9643 - val_loss: 0.0863 - val_acc: 0.9783
Epoch 13/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1277 - acc: 0.9669 - val_loss: 0.0830 - val_acc: 0.9793
Epoch 14/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1282 - acc: 0.9672 - val_loss: 0.0799 - val_acc: 0.9799
Epoch 15/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1225 - acc: 0.9677 - val_loss: 0.0816 - val_acc: 0.9783
Epoch 16/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1153 - acc: 0.9702 - val_loss: 0.0747 - val_acc: 0.9810
Epoch 17/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1194 - acc: 0.9698 - val_loss: 0.0727 - val_acc: 0.9814
Epoch 18/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.1078 - acc: 0.9720 - val_loss: 0.0716 - val_acc: 0.9832
Epoch 19/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.1051 - acc: 0.9726 - val_loss: 0.0692 - val_acc: 0.9822
Epoch 20/50
60000/60000 [==============================] - 9s 157us/step - loss: 0.1019 - acc: 0.9740 - val_loss: 0.0723 - val_acc: 0.9815
Epoch 21/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.1049 - acc: 0.9728 - val_loss: 0.0759 - val_acc: 0.9816
Epoch 22/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0966 - acc: 0.9746 - val_loss: 0.0727 - val_acc: 0.9823
Epoch 23/50
```

```
60000/60000 [==============================] - 9s 154us/step - loss: 0.0939 - acc: 0.9754 - val_loss: 0.0704 - val_acc: 0.9824
Epoch 24/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0931 - acc: 0.9756 - val_loss: 0.0705 - val_acc: 0.9827
Epoch 25/50
60000/60000 [==============================] - 9s 157us/step - loss: 0.0894 - acc: 0.9768 - val_loss: 0.0706 - val_acc: 0.9828
Epoch 26/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0863 - acc: 0.9766 - val_loss: 0.0737 - val_acc: 0.9814
Epoch 27/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0839 - acc: 0.9787 - val_loss: 0.0656 - val_acc: 0.9842
Epoch 28/50
60000/60000 [==============================] - 9s 157us/step - loss: 0.0864 - acc: 0.9776 - val_loss: 0.0699 - val_acc: 0.9828
Epoch 29/50
60000/60000 [==============================] - 10s 159us/step - loss: 0.0814 - acc: 0.9786 - val_loss: 0.0677 - val_acc: 0.9828
Epoch 30/50
60000/60000 [==============================] - 9s 158us/step - loss: 0.0779 - acc: 0.9798 - val_loss: 0.0667 - val_acc: 0.9842
Epoch 31/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0836 - acc: 0.9785 - val_loss: 0.0666 - val_acc: 0.9841
Epoch 32/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0771 - acc: 0.9796 - val_loss: 0.0684 - val_acc: 0.9837
Epoch 33/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0724 - acc: 0.9817 - val_loss: 0.0682 - val_acc: 0.9845
Epoch 34/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0772 - acc: 0.9793 - val_loss: 0.0725 - val_acc: 0.9831
Epoch 35/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0743 - acc: 0.9814 - val_loss: 0.0719 - val_acc: 0.9817
Epoch 36/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0688 - acc: 0.9814 - val_loss: 0.0682 - val_acc: 0.9852
Epoch 37/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0659 - acc: 0.9828 - val_loss: 0.0750 - val_acc: 0.9820
Epoch 38/50
60000/60000 [==============================] - 9s 157us/step - loss: 0.0671 - acc: 0.9820 - val_loss: 0.0698 - val_acc: 0.9836
Epoch 39/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0712 - acc: 0.9810 - val_loss: 0.0673 - val_acc: 0.9844
Epoch 40/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0660 - acc: 0.9831 - val_loss: 0.0713 - val_acc: 0.9837
Epoch 41/50
60000/60000 [==============================] - 10s 161us/step - loss: 0.0645 - acc: 0.9828 - val_loss: 0.0664 - val_acc: 0.9846
Epoch 42/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0655 - acc: 0.9826 - val_loss: 0.0699 - val_acc: 0.9822
Epoch 43/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0626 - acc: 0.9836 - val_loss: 0.0690 - val_acc: 0.9848
Epoch 44/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0586 - acc: 0.9842 - val_loss: 0.0662 - val_acc: 0.9856
Epoch 45/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0604 - acc: 0.9845 - val_loss: 0.0670 - val_acc: 0.9846
Epoch 46/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0614 - acc: 0.9843 - val_loss: 0.0639 - val_acc: 0.9843
Epoch 47/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0595 - acc: 0.9842 - val_loss: 0.0678 - val_acc: 0.9841
Epoch 48/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0595 - acc: 0.9842 - val_loss: 0.0672 - val_acc: 0.9850
```

```
Epoch 49/50
60000/60000 [==============================] - 9s 156us/step - loss: 0.0577 - acc: 0.9848 - val_loss: 0.0628 - val_acc: 0.9853
Epoch 50/50
60000/60000 [==============================] - 9s 155us/step - loss: 0.0543 - acc: 0.9861 - val_loss: 0.0644 - val_acc: 0.9853
```

In [47]:
```
%matplotlib inline
score = model_five.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# 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: 0.06441039258484961
Test accuracy: 0.9853
```



In [49]:
```
# Please write down few lines about what you observed from this assignment.
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
x.field_names = [ "Model","Number of Hidden Layers", "Train Accuracy", "Test Accuracy"]
x.add_row(["MLP + ReLU Without Batch Normalization & Dropout", "2", "0.9980", "0.9818"])
x.add_row(["MLP + ReLU With Batch Normalization & Dropout", "2", "0.9907", "0.9863"])
x.add_row(["MLP + ReLU Without Batch Normalization & Dropout", "3", "0.9988", "0.9817"])
x.add_row(["MLP + ReLU With Batch Normalization & Dropout", "3", "0.9846", "0.9828"])
x.add_row(["MLP + ReLU Without Batch Normalization & Dropout", "5", "0.9991", "0.9828"])
x.add_row(["MLP + ReLU With Batch Normalization & Dropout", "5", "0.9861", "0.9853"])
print(x)
```

```
+--------------------------------------------------+-------------------------+----------------+---------------+
|                      Model                       | Number of Hidden Layers | Train Accuracy | Test Accuracy |
+--------------------------------------------------+-------------------------+----------------+---------------+
| MLP + ReLU Without Batch Normalization & Dropout |            2            |     0.9980     |     0.9818    |
|  MLP + ReLU With Batch Normalization & Dropout   |            2            |     0.9907     |     0.9863    |
| MLP + ReLU Without Batch Normalization & Dropout |            3            |     0.9988     |     0.9817    |
|  MLP + ReLU With Batch Normalization & Dropout   |            3            |     0.9846     |     0.9828    |
| MLP + ReLU Without Batch Normalization & Dropout |            5            |     0.9991     |     0.9828    |
|  MLP + ReLU With Batch Normalization & Dropout   |            5            |     0.9861     |     0.9853    |
+--------------------------------------------------+-------------------------+----------------+---------------+
```

In [ ]: