In [2]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

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

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [4]:

```
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

In [5]:

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

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

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

In [6]:

```
%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()
    plt.show()
#     fig.canvas.draw()
#     plt.show()
```

# 1) ConvNet with 3 x3 kernel

## 1.1) ConvNet(32-64) | 2 Dropouts | Maxpool | Dense(128-10) | 1 Flatten | ReLU | Adadelta

In [17]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 48s 794us/step - loss: 0.2667 -
accuracy: 0.9190 - val_loss: 0.0583 - val_accuracy: 0.9817
Epoch 2/12
60000/60000 [==============================] - 49s 811us/step - loss: 0.0867 -
accuracy: 0.9738 - val_loss: 0.0395 - val_accuracy: 0.9857
Epoch 3/12
60000/60000 [==============================] - 47s 785us/step - loss: 0.0656 -
accuracy: 0.9804 - val_loss: 0.0358 - val_accuracy: 0.9878
Epoch 4/12
60000/60000 [==============================] - 48s 793us/step - loss: 0.0536 -
accuracy: 0.9839 - val_loss: 0.0304 - val_accuracy: 0.9898
Epoch 5/12
60000/60000 [==============================] - 49s 815us/step - loss: 0.0478 -
accuracy: 0.9860 - val_loss: 0.0315 - val_accuracy: 0.9894
Epoch 6/12
60000/60000 [==============================] - 48s 803us/step - loss: 0.0415 -
accuracy: 0.9875 - val_loss: 0.0277 - val_accuracy: 0.9902
Epoch 7/12
60000/60000 [==============================] - 47s 789us/step - loss: 0.0368 -
accuracy: 0.9889 - val_loss: 0.0280 - val_accuracy: 0.9904
Epoch 8/12
60000/60000 [==============================] - 47s 775us/step - loss: 0.0339 -
accuracy: 0.9894 - val_loss: 0.0320 - val_accuracy: 0.9895
Epoch 9/12
60000/60000 [==============================] - 47s 784us/step - loss: 0.0325 -
accuracy: 0.9897 - val_loss: 0.0243 - val_accuracy: 0.9916
Epoch 10/12
60000/60000 [==============================] - 48s 792us/step - loss: 0.0292 -
accuracy: 0.9910 - val_loss: 0.0284 - val_accuracy: 0.9904
Epoch 11/12
```

```
60000/60000 [==============================] - 49s 821us/step - loss: 0.0275 -
accuracy: 0.9918 - val_loss: 0.0241 - val_accuracy: 0.9919
Epoch 12/12
60000/60000 [==============================] - 47s 787us/step - loss: 0.0262 -
accuracy: 0.9917 - val_loss: 0.0289 - val_accuracy: 0.9914
Test loss: 0.028938814725703924
Test accuracy: 0.9914000034332275
```

In [18]:

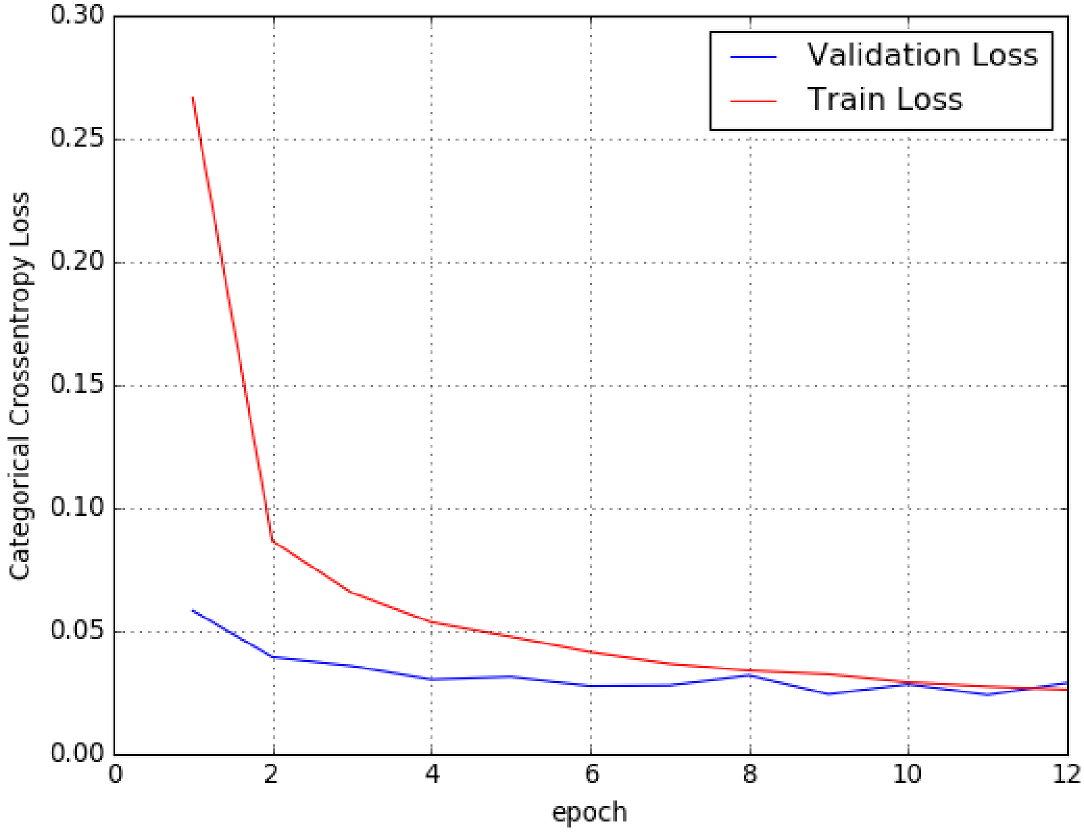
```
history.history['val_loss']
```

Out[18]:

```
[0.058340036510489884,
 0.03950693163461983,
 0.035849299174919726,
 0.030375180654320866,
 0.03146639704736881,
 0.02769776120893657,
 0.027960741236479954,
 0.03204326269463636,
 0.024335926883982027,
 0.02840736617653456,
 0.024061167104181366,
 0.028938813609300996]
```

In [19]:

```
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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```



## 1.2) ConvNet(32-32-64) | 1 Dropouts | 3 MaxPools | Dense(64-10) |1 Flatten | ReLU | Adam

In [20]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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))
```

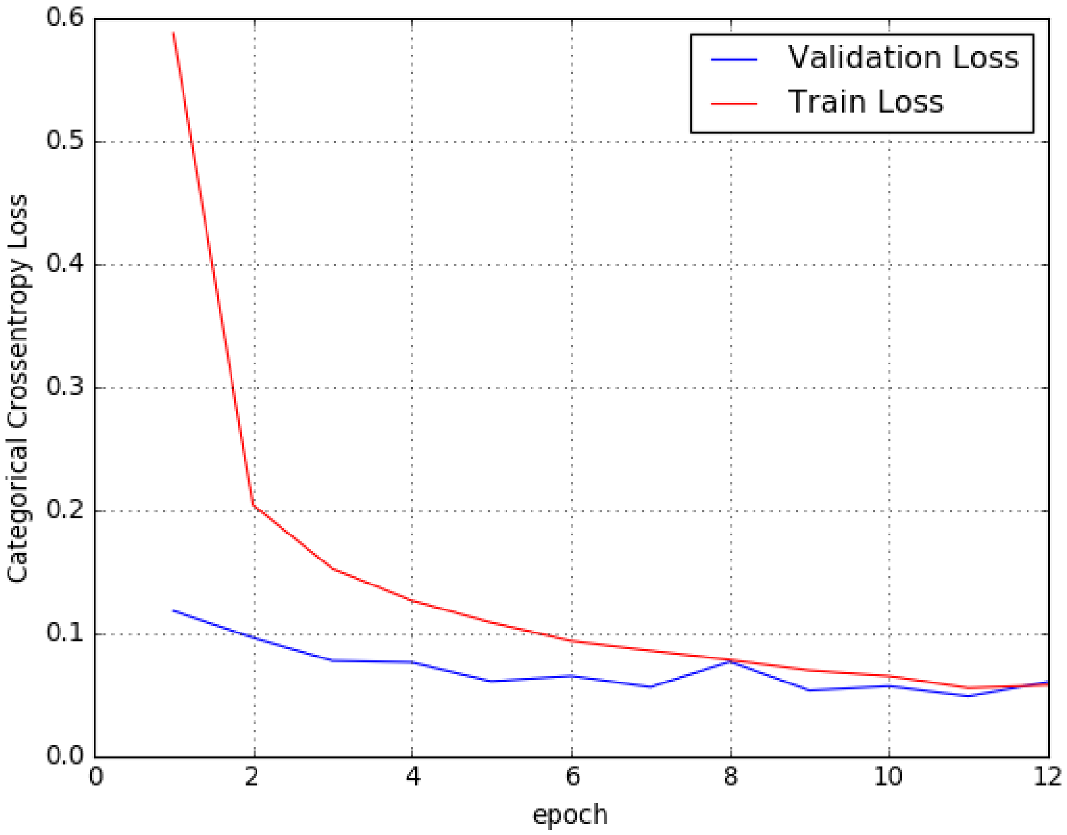
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 12s 205us/step - loss: 0.5875 -
accuracy: 0.8144 - val_loss: 0.1185 - val_accuracy: 0.9638
Epoch 2/12
60000/60000 [==============================] - 12s 199us/step - loss: 0.2047 -
accuracy: 0.9418 - val_loss: 0.0966 - val_accuracy: 0.9702
Epoch 3/12
60000/60000 [==============================] - 12s 200us/step - loss: 0.1527 -
accuracy: 0.9571 - val_loss: 0.0780 - val_accuracy: 0.9764
Epoch 4/12
60000/60000 [==============================] - 12s 208us/step - loss: 0.1269 -
accuracy: 0.9650 - val_loss: 0.0766 - val_accuracy: 0.9782
Epoch 5/12
60000/60000 [==============================] - 13s 212us/step - loss: 0.1090 -
accuracy: 0.9701 - val_loss: 0.0612 - val_accuracy: 0.9828
Epoch 6/12
60000/60000 [==============================] - 13s 212us/step - loss: 0.0941 -
accuracy: 0.9734 - val_loss: 0.0656 - val_accuracy: 0.9804
Epoch 7/12
60000/60000 [==============================] - 13s 217us/step - loss: 0.0861 -
accuracy: 0.9762 - val_loss: 0.0566 - val_accuracy: 0.9841
Epoch 8/12
60000/60000 [==============================] - 13s 221us/step - loss: 0.0786 -
accuracy: 0.9779 - val_loss: 0.0772 - val_accuracy: 0.9782
Epoch 9/12
60000/60000 [==============================] - 13s 214us/step - loss: 0.0702 -
accuracy: 0.9801 - val_loss: 0.0538 - val_accuracy: 0.9853
Epoch 10/12
```

```
60000/60000 [==============================] - 13s 215us/step - loss: 0.0655 -
accuracy: 0.9816 - val_loss: 0.0574 - val_accuracy: 0.9849
Epoch 11/12
60000/60000 [==============================] - 13s 210us/step - loss: 0.0559 -
accuracy: 0.9839 - val_loss: 0.0491 - val_accuracy: 0.9861
Epoch 12/12
60000/60000 [==============================] - 12s 203us/step - loss: 0.0582 -
accuracy: 0.9829 - val_loss: 0.0608 - val_accuracy: 0.9844
```

In [21]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.06079796881068669
Test accuracy: 0.9843999743461609
```



# 1.3: ConvNet(32-64) | 2 Dropouts | 2 MaxPools | Dense(128-10) | 1 Flatten | ReLU | Adam | Padding: "same"

In [22]:

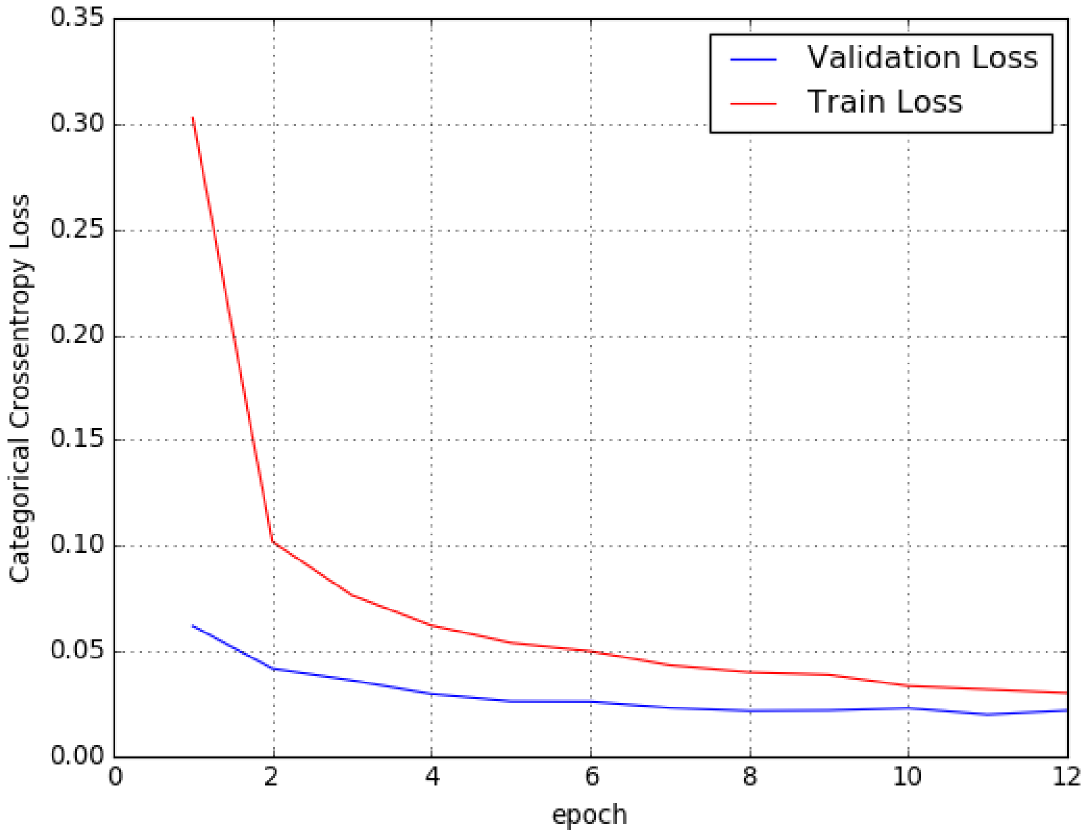
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape,
                padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())

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

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

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))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 18s 307us/step - loss: 0.3029
- accuracy: 0.9059 - val_loss: 0.0620 - val_accuracy: 0.9798
Epoch 2/12
60000/60000 [==============================] - 18s 303us/step - loss: 0.1020
- accuracy: 0.9698 - val_loss: 0.0418 - val_accuracy: 0.9856
Epoch 3/12
60000/60000 [==============================] - 18s 305us/step - loss: 0.0766
- accuracy: 0.9768 - val_loss: 0.0362 - val_accuracy: 0.9881
Epoch 4/12
60000/60000 [==============================] - 18s 297us/step - loss: 0.0623
- accuracy: 0.9808 - val_loss: 0.0298 - val_accuracy: 0.9888
Epoch 5/12
60000/60000 [==============================] - 18s 296us/step - loss: 0.0540
- accuracy: 0.9835 - val_loss: 0.0263 - val_accuracy: 0.9907
Epoch 6/12
60000/60000 [==============================] - 18s 306us/step - loss: 0.0500
- accuracy: 0.9847 - val_loss: 0.0262 - val_accuracy: 0.9916
Epoch 7/12
60000/60000 [==============================] - 18s 301us/step - loss: 0.0433
- accuracy: 0.9868 - val_loss: 0.0232 - val_accuracy: 0.9927
Epoch 8/12
60000/60000 [==============================] - 18s 293us/step - loss: 0.0400
- accuracy: 0.9880 - val_loss: 0.0217 - val_accuracy: 0.9931
Epoch 9/12
60000/60000 [==============================] - 18s 300us/step - loss: 0.0389
- accuracy: 0.9882 - val_loss: 0.0219 - val_accuracy: 0.9918
Epoch 10/12
```

```
60000/60000 [==============================] - 18s 303us/step - loss: 0.0335
- accuracy: 0.9895 - val_loss: 0.0230 - val_accuracy: 0.9927
Epoch 11/12
60000/60000 [==============================] - 18s 308us/step - loss: 0.0319
- accuracy: 0.9900 - val_loss: 0.0199 - val_accuracy: 0.9934
Epoch 12/12
60000/60000 [==============================] - 19s 309us/step - loss: 0.0302
- accuracy: 0.9906 - val_loss: 0.0219 - val_accuracy: 0.9934
```

In [23]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.021854717017415988
Test accuracy: 0.993399977684021
```



## 1.4: ConvNet(32-32) | 2 Dropouts | 2 MaxPools | Dense(64-10) | 1 Flatten | ReLU | Adam | Padding: "same" | Batch Normalization

In [8]:

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

```
In [11]: model = Sequential()
         model.add(Conv2D(32, kernel_size=(3, 3),
                          activation='relu',
                          input_shape=input_shape,
                         padding = 'same'))
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(32, (3, 3), activation='relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Dropout(0.25))
         model.add(Flatten())

         model.add(Dense(64, activation='relu'))
         model.add(Dropout(0.5))
         model.add(BatchNormalization())
         model.add(Dense(num_classes, activation='softmax'))

         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))
```

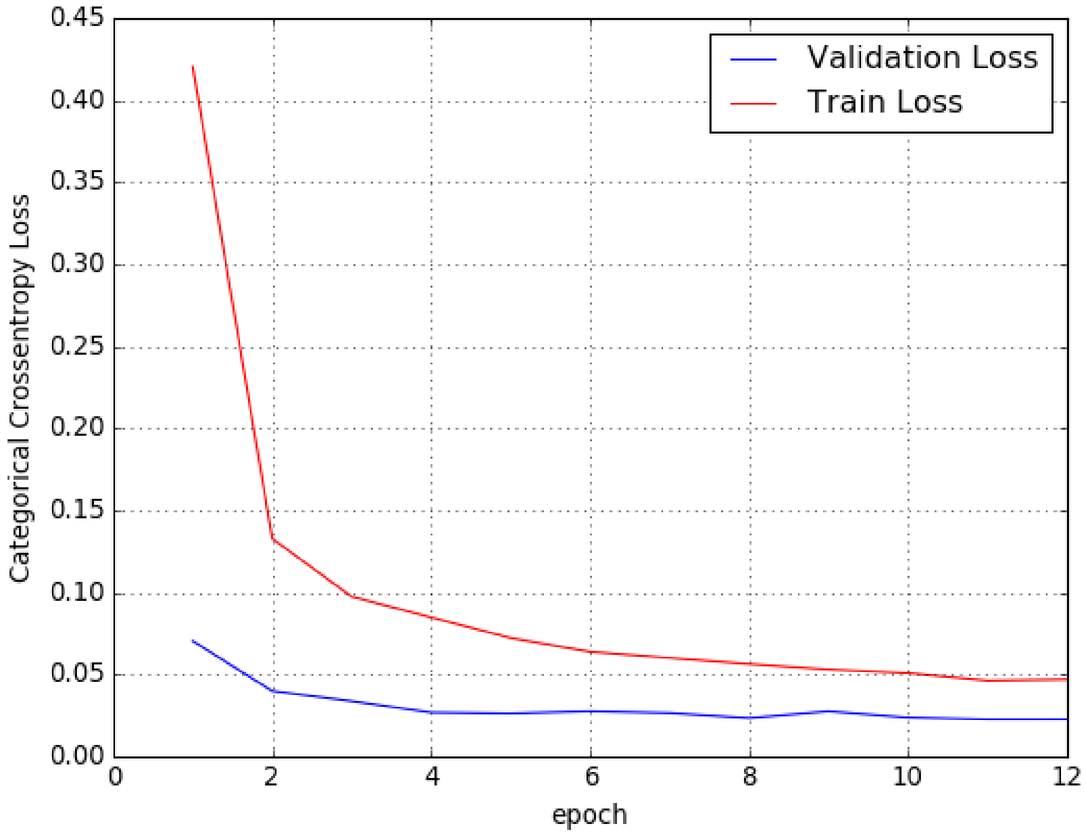
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 15s 250us/step - loss: 0.4205
- accuracy: 0.8844 - val_loss: 0.0705 - val_accuracy: 0.9819
Epoch 2/12
60000/60000 [==============================] - 14s 236us/step - loss: 0.1329
- accuracy: 0.9644 - val_loss: 0.0400 - val_accuracy: 0.9871
Epoch 3/12
60000/60000 [==============================] - 14s 234us/step - loss: 0.0975
- accuracy: 0.9728 - val_loss: 0.0338 - val_accuracy: 0.9881
Epoch 4/12
60000/60000 [==============================] - 14s 235us/step - loss: 0.0849
- accuracy: 0.9754 - val_loss: 0.0269 - val_accuracy: 0.9915
Epoch 5/12
60000/60000 [==============================] - 14s 234us/step - loss: 0.0723
- accuracy: 0.9797 - val_loss: 0.0264 - val_accuracy: 0.9914
Epoch 6/12
60000/60000 [==============================] - 14s 241us/step - loss: 0.0640
- accuracy: 0.9816 - val_loss: 0.0276 - val_accuracy: 0.9908
Epoch 7/12
60000/60000 [==============================] - 15s 245us/step - loss: 0.0601
- accuracy: 0.9829 - val_loss: 0.0266 - val_accuracy: 0.9914
Epoch 8/12
60000/60000 [==============================] - 15s 242us/step - loss: 0.0565
- accuracy: 0.9828 - val_loss: 0.0236 - val_accuracy: 0.9920
Epoch 9/12
60000/60000 [==============================] - 15s 251us/step - loss: 0.0532
- accuracy: 0.9842 - val_loss: 0.0276 - val_accuracy: 0.9915
Epoch 10/12
```

```
60000/60000 [==============================] - 14s 236us/step - loss: 0.0509
- accuracy: 0.9846 - val_loss: 0.0239 - val_accuracy: 0.9910
Epoch 11/12
60000/60000 [==============================] - 15s 243us/step - loss: 0.0464
- accuracy: 0.9858 - val_loss: 0.0228 - val_accuracy: 0.9925
Epoch 12/12
60000/60000 [==============================] - 15s 249us/step - loss: 0.0471
- accuracy: 0.9856 - val_loss: 0.0228 - val_accuracy: 0.9934
```

In [12]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.02282937448550165
Test accuracy: 0.993399977684021
```



# 2) ConvNet with 5 x 5 kernel

## 2.1: ConvNet(128-64-32) | 3 Dropouts | 2 Maxpool | Dense(128-10) | 1 Flatten | ReLU | Adam

In [10]:

```
model = Sequential()
model.add(Conv2D(128, 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(Dropout(0.25))

model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.55))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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))
```

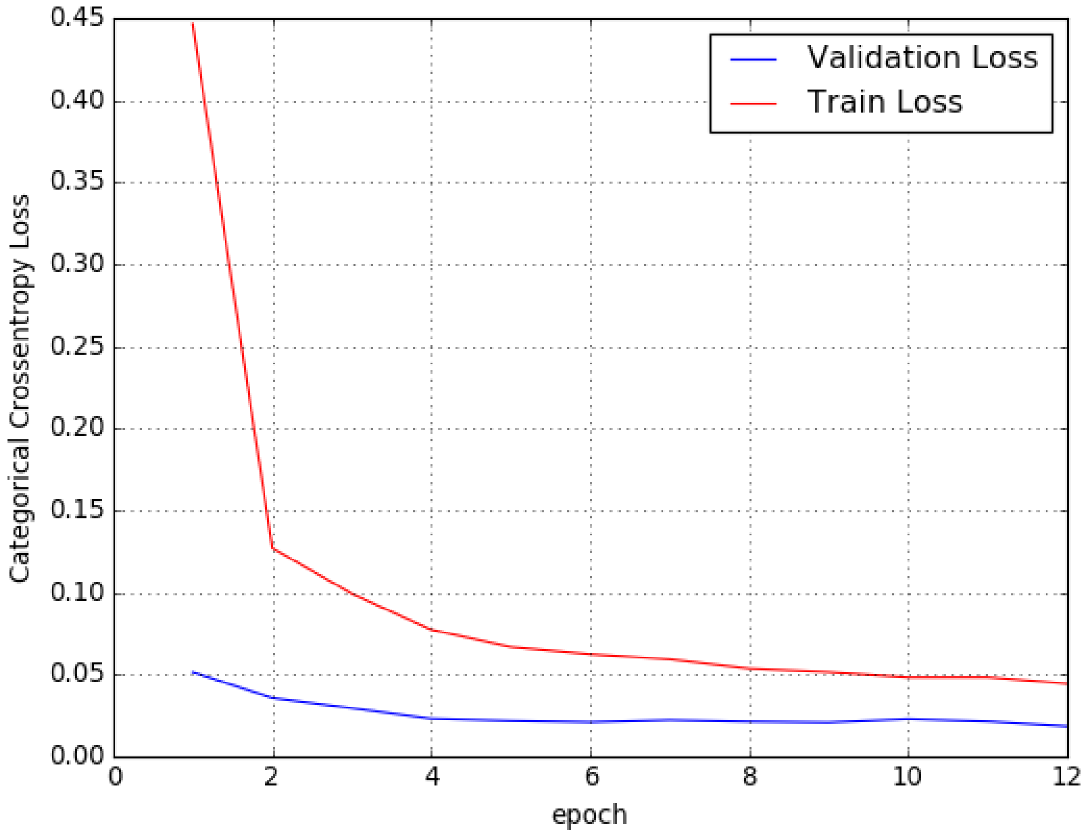
```
WARNING:tensorflow:Large dropout rate: 0.55 (>0.5). In TensorFlow 2.x, dropou
t() uses dropout rate instead of keep_prob. Please ensure that this is intend
ed.
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 257s 4ms/step - loss: 0.4465 -
accuracy: 0.8522 - val_loss: 0.0515 - val_accuracy: 0.9845
Epoch 2/12
60000/60000 [==============================] - 265s 4ms/step - loss: 0.1272 -
accuracy: 0.9619 - val_loss: 0.0358 - val_accuracy: 0.9893
Epoch 3/12
60000/60000 [==============================] - 262s 4ms/step - loss: 0.0994 -
accuracy: 0.9710 - val_loss: 0.0296 - val_accuracy: 0.9912
Epoch 4/12
60000/60000 [==============================] - 266s 4ms/step - loss: 0.0774 -
accuracy: 0.9771 - val_loss: 0.0232 - val_accuracy: 0.9931
Epoch 5/12
60000/60000 [==============================] - 263s 4ms/step - loss: 0.0669 -
accuracy: 0.9806 - val_loss: 0.0219 - val_accuracy: 0.9934
Epoch 6/12
60000/60000 [==============================] - 263s 4ms/step - loss: 0.0625 -
accuracy: 0.9823 - val_loss: 0.0210 - val_accuracy: 0.9929
Epoch 7/12
60000/60000 [==============================] - 264s 4ms/step - loss: 0.0593 -
accuracy: 0.9829 - val_loss: 0.0224 - val_accuracy: 0.9934
Epoch 8/12
60000/60000 [==============================] - 260s 4ms/step - loss: 0.0537 -
accuracy: 0.9842 - val_loss: 0.0214 - val_accuracy: 0.9942
Epoch 9/12
```

```
60000/60000 [==============================] - 255s 4ms/step - loss: 0.0516 -
accuracy: 0.9848 - val_loss: 0.0209 - val_accuracy: 0.9939
Epoch 10/12
60000/60000 [==============================] - 256s 4ms/step - loss: 0.0484 -
accuracy: 0.9857 - val_loss: 0.0230 - val_accuracy: 0.9922
Epoch 11/12
60000/60000 [==============================] - 255s 4ms/step - loss: 0.0484 -
accuracy: 0.9856 - val_loss: 0.0215 - val_accuracy: 0.9940
Epoch 12/12
60000/60000 [==============================] - 255s 4ms/step - loss: 0.0445 -
accuracy: 0.9871 - val_loss: 0.0186 - val_accuracy: 0.9944
```

In [11]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.018588077808042364
Test accuracy: 0.9944000244140625
```



# 2.2: ConvNet(64-32) | 3 Dropouts | 3 MaxPools | Dense(128-32) | 1 Flatten | ReLU | Adam

In [12]:
```
model = Sequential()
model.add(Conv2D(64, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))

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

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))
```

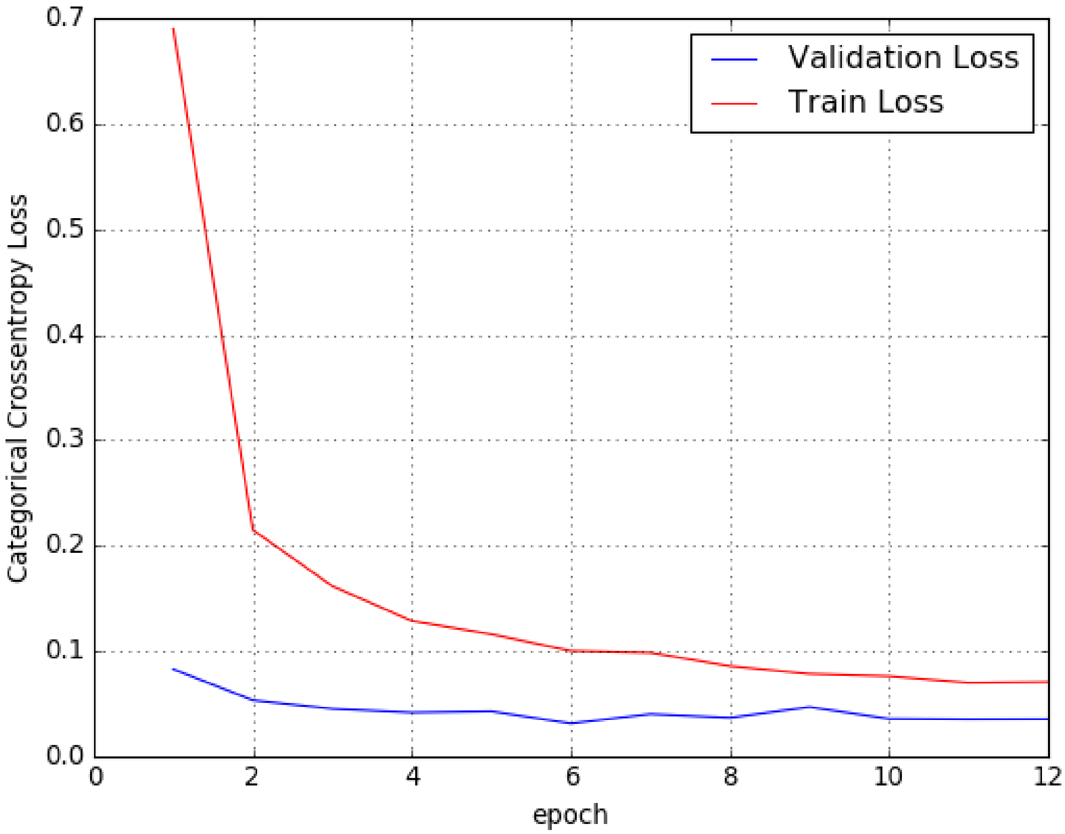
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 25s 412us/step - loss: 0.6901
- accuracy: 0.7758 - val_loss: 0.0829 - val_accuracy: 0.9776
Epoch 2/12
60000/60000 [==============================] - 24s 404us/step - loss: 0.2150
- accuracy: 0.9410 - val_loss: 0.0535 - val_accuracy: 0.9848
Epoch 3/12
60000/60000 [==============================] - 24s 402us/step - loss: 0.1616
- accuracy: 0.9573 - val_loss: 0.0455 - val_accuracy: 0.9887
Epoch 4/12
60000/60000 [==============================] - 24s 403us/step - loss: 0.1287
- accuracy: 0.9653 - val_loss: 0.0418 - val_accuracy: 0.9901
Epoch 5/12
60000/60000 [==============================] - 24s 405us/step - loss: 0.1160
- accuracy: 0.9690 - val_loss: 0.0430 - val_accuracy: 0.9890
Epoch 6/12
60000/60000 [==============================] - 25s 410us/step - loss: 0.1003
- accuracy: 0.9725 - val_loss: 0.0315 - val_accuracy: 0.9920
Epoch 7/12
60000/60000 [==============================] - 24s 403us/step - loss: 0.0986
- accuracy: 0.9749 - val_loss: 0.0404 - val_accuracy: 0.9905
Epoch 8/12
60000/60000 [==============================] - 24s 408us/step - loss: 0.0859
- accuracy: 0.9762 - val_loss: 0.0369 - val_accuracy: 0.9902
Epoch 9/12
60000/60000 [==============================] - 24s 398us/step - loss: 0.0785
```

```
- accuracy: 0.9776 - val_loss: 0.0471 - val_accuracy: 0.9893
Epoch 10/12
60000/60000 [==============================] - 24s 399us/step - loss: 0.0764
- accuracy: 0.9791 - val_loss: 0.0359 - val_accuracy: 0.9913
Epoch 11/12
60000/60000 [==============================] - 24s 400us/step - loss: 0.0701
- accuracy: 0.9803 - val_loss: 0.0355 - val_accuracy: 0.9924
Epoch 12/12
60000/60000 [==============================] - 24s 401us/step - loss: 0.0708
- accuracy: 0.9814 - val_loss: 0.0356 - val_accuracy: 0.9918
```

In [13]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.03562645074729423
Test accuracy: 0.9918000102043152
```



## 2.3: ConvNet(128-64-32) | 3 Dropouts | 3 MaxPools | Dense(64-32) | 1 Flatten | ReLU | Adam | Padding: "same"

In [18]:

```
model = Sequential()
model.add(Conv2D(128, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape,
                padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

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

model.add(Conv2D(32, (5, 5), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))

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

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))
```

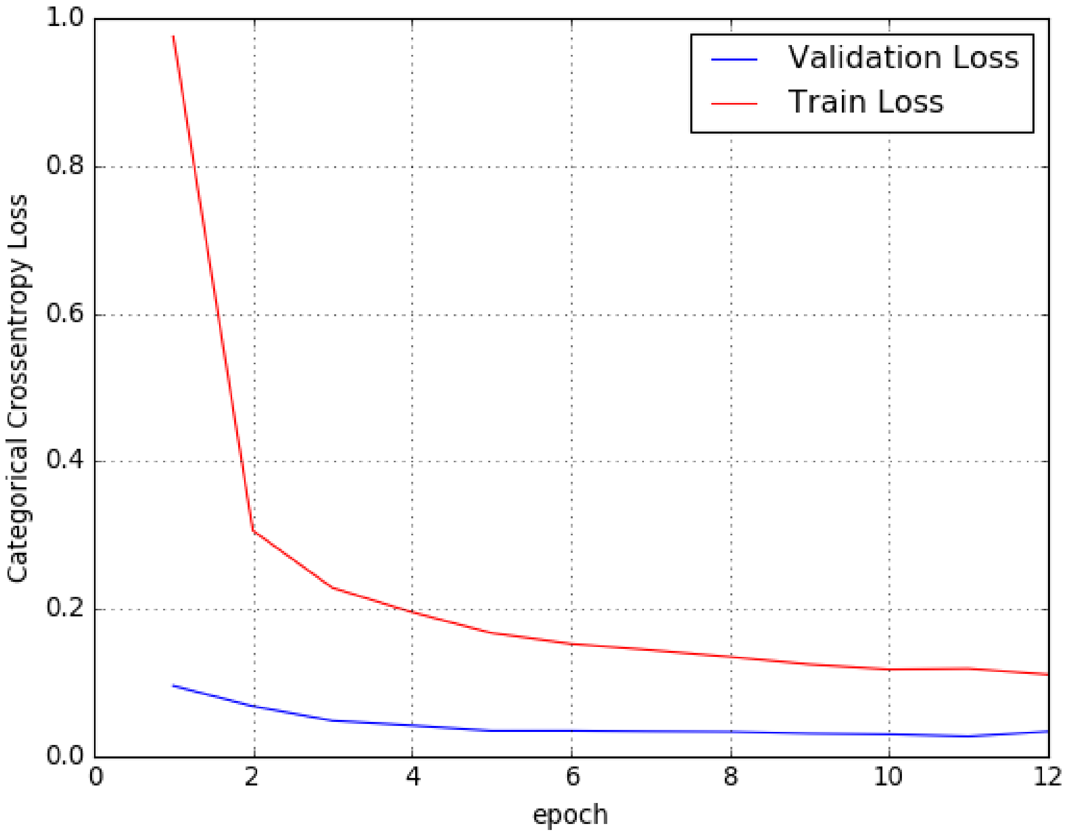
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.9753 -
accuracy: 0.6654 - val_loss: 0.0959 - val_accuracy: 0.9736
Epoch 2/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.3063 -
accuracy: 0.9153 - val_loss: 0.0684 - val_accuracy: 0.9830
Epoch 3/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.2286 -
accuracy: 0.9386 - val_loss: 0.0487 - val_accuracy: 0.9873
Epoch 4/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.1958 -
accuracy: 0.9506 - val_loss: 0.0422 - val_accuracy: 0.9896
Epoch 5/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.1675 -
accuracy: 0.9560 - val_loss: 0.0353 - val_accuracy: 0.9917
Epoch 6/12
60000/60000 [==============================] - 151s 3ms/step - loss: 0.1529 -
accuracy: 0.9603 - val_loss: 0.0353 - val_accuracy: 0.9920
Epoch 7/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.1444 -
accuracy: 0.9622 - val_loss: 0.0343 - val_accuracy: 0.9917
```

```
Epoch 8/12
60000/60000 [==============================] - 151s 3ms/step - loss: 0.1352 -
accuracy: 0.9661 - val_loss: 0.0337 - val_accuracy: 0.9928
Epoch 9/12
60000/60000 [==============================] - 151s 3ms/step - loss: 0.1250 -
accuracy: 0.9681 - val_loss: 0.0314 - val_accuracy: 0.9926
Epoch 10/12
60000/60000 [==============================] - 151s 3ms/step - loss: 0.1184 -
accuracy: 0.9688 - val_loss: 0.0303 - val_accuracy: 0.9927
Epoch 11/12
60000/60000 [==============================] - 151s 3ms/step - loss: 0.1194 -
accuracy: 0.9695 - val_loss: 0.0274 - val_accuracy: 0.9929
Epoch 12/12
60000/60000 [==============================] - 151s 3ms/step - loss: 0.1112 -
accuracy: 0.9708 - val_loss: 0.0341 - val_accuracy: 0.9917
```

In [19]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.03410061263831351
Test accuracy: 0.9916999936103821
```



## 2.4: ConvNet(64-32) | 3 Dropouts | 2 MaxPools | Dense(64-32) | 1 Flatten | ReLU | Adam | Padding: "same" | Batch Normalization

In [9]:

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

model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())

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

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))
```

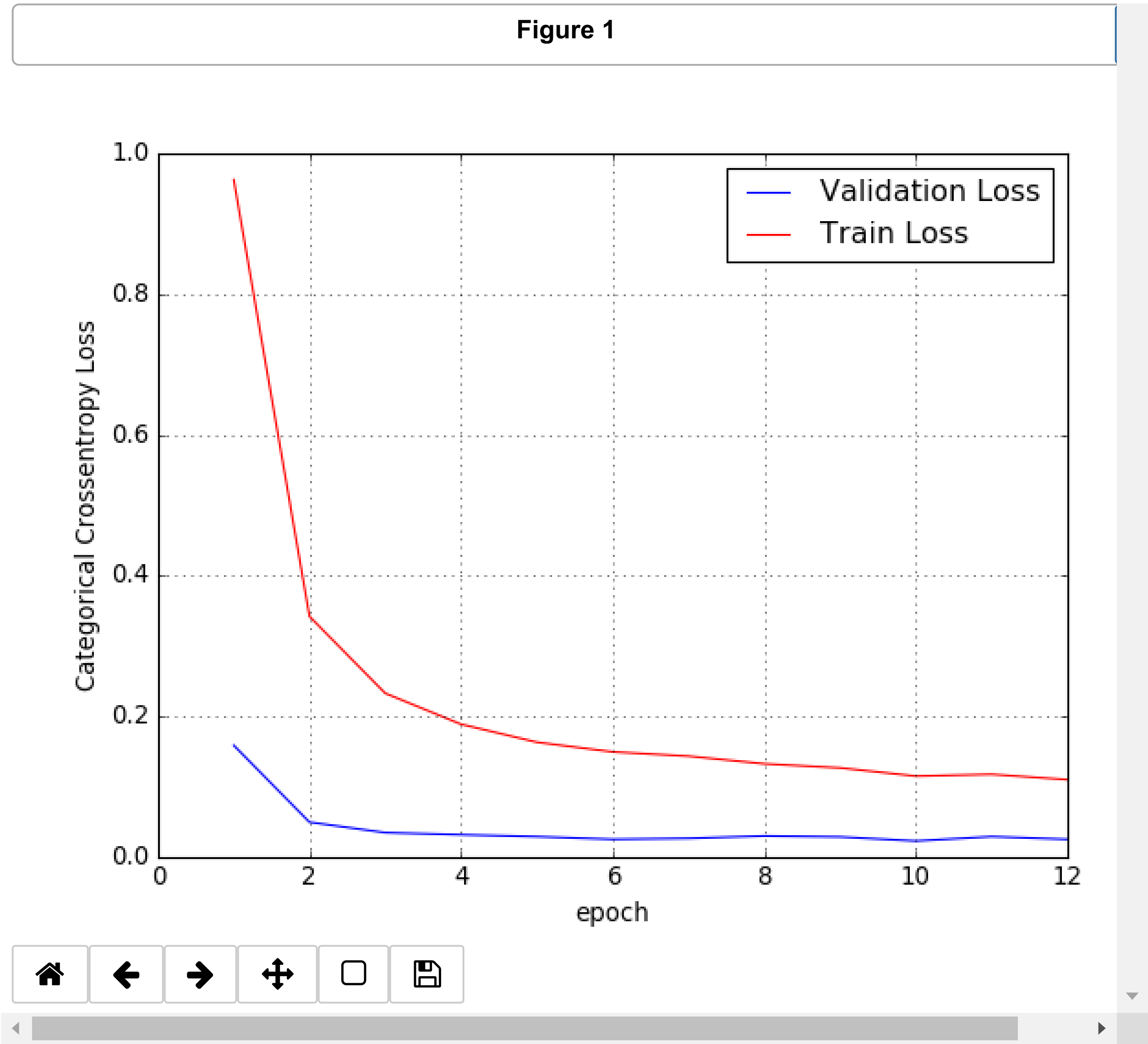
```
WARNING:tensorflow:From /home/komalumrethe/anaconda3/lib/python3.5/site-packa
ges/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is
deprecated. Please use tf.compat.v1.global_variables instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 34s 565us/step - loss: 0.9625
- accuracy: 0.7107 - val_loss: 0.1589 - val_accuracy: 0.9801
Epoch 2/12
60000/60000 [==============================] - 33s 553us/step - loss: 0.3423
- accuracy: 0.9082 - val_loss: 0.0498 - val_accuracy: 0.9865
Epoch 3/12
60000/60000 [==============================] - 33s 547us/step - loss: 0.2330
- accuracy: 0.9379 - val_loss: 0.0349 - val_accuracy: 0.9896
Epoch 4/12
60000/60000 [==============================] - 33s 552us/step - loss: 0.1890
- accuracy: 0.9484 - val_loss: 0.0319 - val_accuracy: 0.9907
Epoch 5/12
60000/60000 [==============================] - 33s 549us/step - loss: 0.1635
- accuracy: 0.9544 - val_loss: 0.0292 - val_accuracy: 0.9915
Epoch 6/12
60000/60000 [==============================] - 33s 550us/step - loss: 0.1497
- accuracy: 0.9583 - val_loss: 0.0254 - val_accuracy: 0.9927
Epoch 7/12
```

```
60000/60000 [==============================] - 33s 546us/step - loss: 0.1436
- accuracy: 0.9601 - val_loss: 0.0268 - val_accuracy: 0.9915
Epoch 8/12
60000/60000 [==============================] - 33s 553us/step - loss: 0.1328
- accuracy: 0.9632 - val_loss: 0.0302 - val_accuracy: 0.9917
Epoch 9/12
60000/60000 [==============================] - 33s 546us/step - loss: 0.1270
- accuracy: 0.9641 - val_loss: 0.0288 - val_accuracy: 0.9918
Epoch 10/12
60000/60000 [==============================] - 33s 546us/step - loss: 0.1153
- accuracy: 0.9662 - val_loss: 0.0231 - val_accuracy: 0.9930
Epoch 11/12
60000/60000 [==============================] - 33s 549us/step - loss: 0.1178
- accuracy: 0.9661 - val_loss: 0.0293 - val_accuracy: 0.9919
Epoch 12/12
60000/60000 [==============================] - 33s 552us/step - loss: 0.1103
- accuracy: 0.9679 - val_loss: 0.0254 - val_accuracy: 0.9927
```

In [10]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.025404230587064557
Test accuracy: 0.9926999807357788
```



# 3) ConvNet with 7 x 7 kernel

## 3.1: ConvNet (128-64-32) | Dense (128) | 3 Dropouts | 2 Maxpool | 1 Flatten | ReLU | Adam

In [7]:

```
model = Sequential()
model.add(Conv2D(128, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))

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

model.add(Conv2D(32, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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))
```

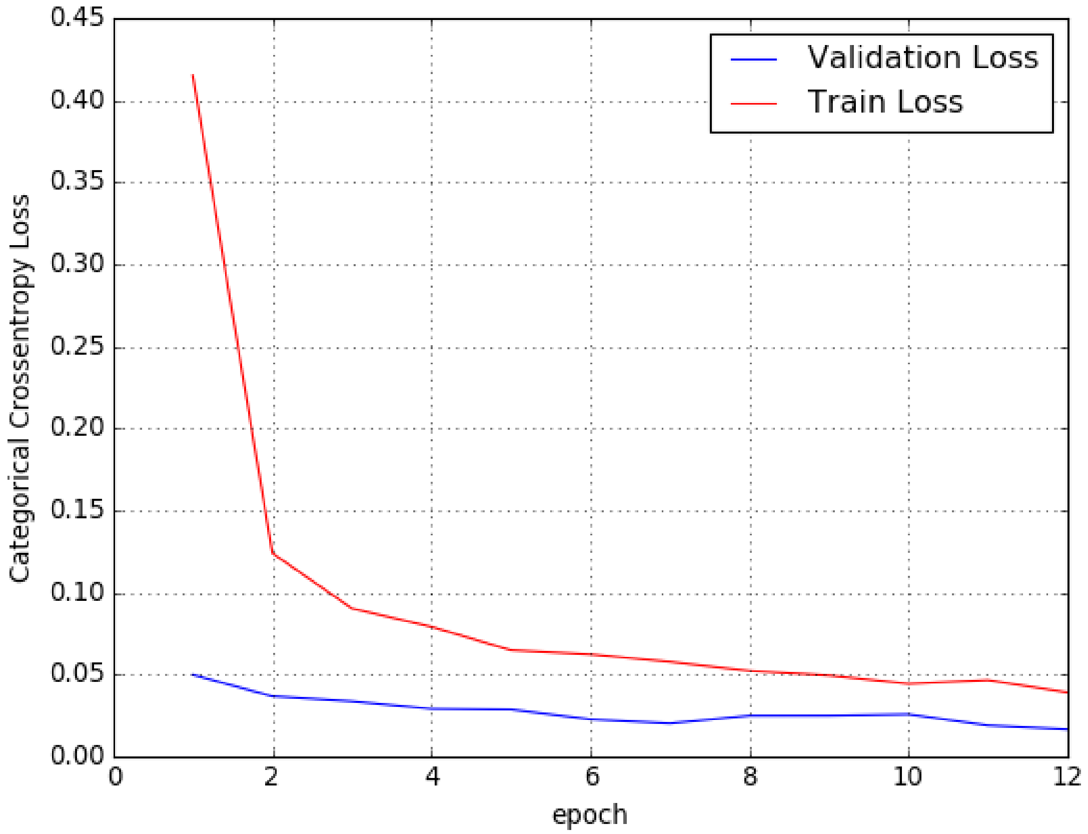
```
WARNING:tensorflow:From /home/komalumrethe/anaconda3/lib/python3.5/site-package
s/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecat
ed. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /home/komalumrethe/anaconda3/lib/python3.5/site-package
s/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is depr
ecated. Please use tf.compat.v1.global_variables instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 322s 5ms/step - loss: 0.4152 - a
ccuracy: 0.8676 - val_loss: 0.0499 - val_accuracy: 0.9861
Epoch 2/12
60000/60000 [==============================] - 322s 5ms/step - loss: 0.1240 - a
ccuracy: 0.9655 - val_loss: 0.0368 - val_accuracy: 0.9894
Epoch 3/12
60000/60000 [==============================] - 322s 5ms/step - loss: 0.0904 - a
ccuracy: 0.9753 - val_loss: 0.0338 - val_accuracy: 0.9904
Epoch 4/12
60000/60000 [==============================] - 322s 5ms/step - loss: 0.0792 - a
ccuracy: 0.9786 - val_loss: 0.0292 - val_accuracy: 0.9917
Epoch 5/12
60000/60000 [==============================] - 319s 5ms/step - loss: 0.0649 - a
ccuracy: 0.9822 - val_loss: 0.0287 - val_accuracy: 0.9917
Epoch 6/12
60000/60000 [==============================] - 326s 5ms/step - loss: 0.0624 - a
ccuracy: 0.9832 - val_loss: 0.0228 - val_accuracy: 0.9932
Epoch 7/12
```

```
60000/60000 [==============================] - 324s 5ms/step - loss: 0.0578 - a
ccuracy: 0.9844 - val_loss: 0.0203 - val_accuracy: 0.9942
Epoch 8/12
60000/60000 [==============================] - 331s 6ms/step - loss: 0.0524 - a
ccuracy: 0.9856 - val_loss: 0.0250 - val_accuracy: 0.9929
Epoch 9/12
60000/60000 [==============================] - 333s 6ms/step - loss: 0.0495 - a
ccuracy: 0.9866 - val_loss: 0.0250 - val_accuracy: 0.9916
Epoch 10/12
60000/60000 [==============================] - 331s 6ms/step - loss: 0.0445 - a
ccuracy: 0.9876 - val_loss: 0.0258 - val_accuracy: 0.9922
Epoch 11/12
60000/60000 [==============================] - 334s 6ms/step - loss: 0.0467 - a
ccuracy: 0.9869 - val_loss: 0.0191 - val_accuracy: 0.9934
Epoch 12/12
60000/60000 [==============================] - 325s 5ms/step - loss: 0.0392 - a
ccuracy: 0.9891 - val_loss: 0.0168 - val_accuracy: 0.9948
```

In [8]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.016798403310499272
Test accuracy: 0.9947999715805054
```



## 3.2: ConvNet(128-64) | 3 Dropouts | 2 MaxPools | 1 Flatten | ReLU | Dense(64-10) | Adam | padding: "same"

In [9]:

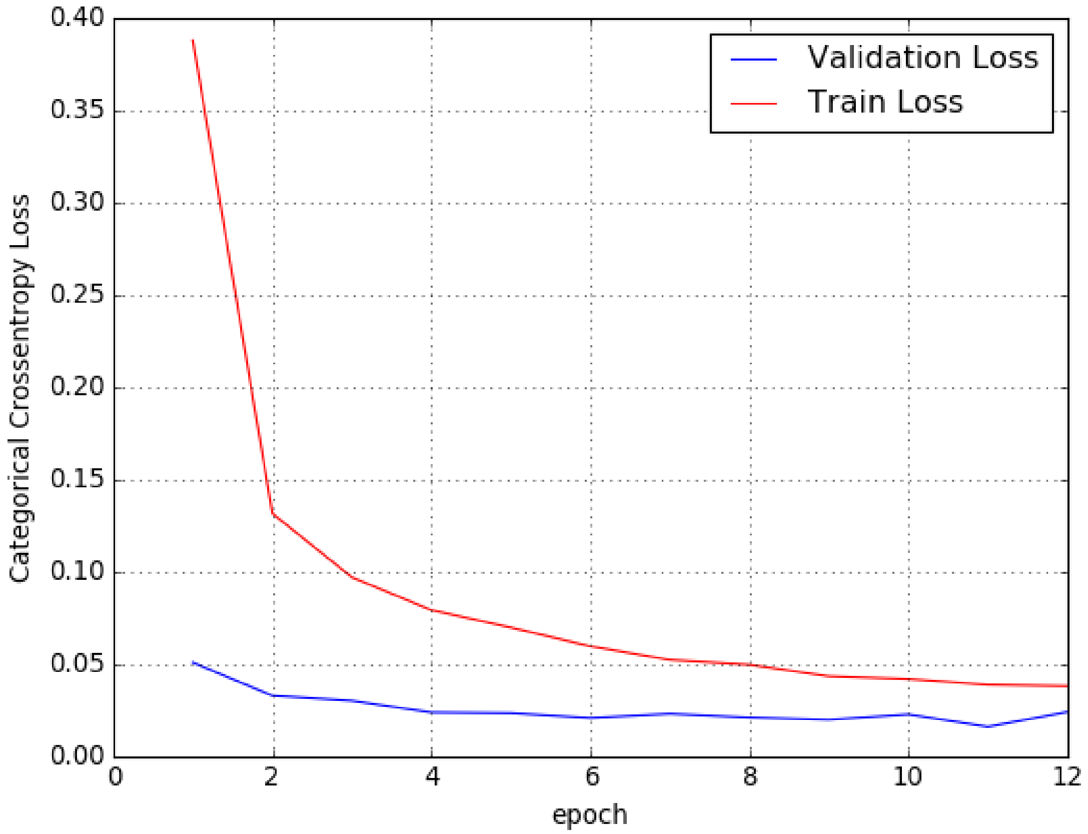
```
model = Sequential()
model.add(Conv2D(128, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape, padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(64, kernel_size=(7, 7),
                 activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 278s 5ms/step - loss: 0.3881 -
accuracy: 0.8764 - val_loss: 0.0510 - val_accuracy: 0.9845
Epoch 2/12
60000/60000 [==============================] - 279s 5ms/step - loss: 0.1318 -
accuracy: 0.9625 - val_loss: 0.0331 - val_accuracy: 0.9895
Epoch 3/12
60000/60000 [==============================] - 278s 5ms/step - loss: 0.0972 -
accuracy: 0.9720 - val_loss: 0.0303 - val_accuracy: 0.9899
Epoch 4/12
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0794 -
accuracy: 0.9765 - val_loss: 0.0240 - val_accuracy: 0.9918
Epoch 5/12
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0700 -
accuracy: 0.9791 - val_loss: 0.0236 - val_accuracy: 0.9920
Epoch 6/12
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0598 -
accuracy: 0.9824 - val_loss: 0.0210 - val_accuracy: 0.9927
Epoch 7/12
60000/60000 [==============================] - 277s 5ms/step - loss: 0.0524 -
accuracy: 0.9844 - val_loss: 0.0233 - val_accuracy: 0.9925
Epoch 8/12
60000/60000 [==============================] - 277s 5ms/step - loss: 0.0498 -
accuracy: 0.9856 - val_loss: 0.0213 - val_accuracy: 0.9934
Epoch 9/12
60000/60000 [==============================] - 274s 5ms/step - loss: 0.0436 -
accuracy: 0.9873 - val_loss: 0.0201 - val_accuracy: 0.9935
Epoch 10/12
60000/60000 [==============================] - 275s 5ms/step - loss: 0.0421 -
```

```
accuracy: 0.9873 - val_loss: 0.0229 - val_accuracy: 0.9932
Epoch 11/12
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0390 -
accuracy: 0.9881 - val_loss: 0.0163 - val_accuracy: 0.9948
Epoch 12/12
60000/60000 [==============================] - 275s 5ms/step - loss: 0.0383 -
accuracy: 0.9880 - val_loss: 0.0242 - val_accuracy: 0.9934
```

```
In [10]: score = model.evaluate(x_test, y_test, verbose=0)
         print('Test loss:', 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_epo

         # we will get val_loss and val_acc only when you pass the paramter validation_dat
         # 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


         vy = history.history['val_loss']
         ty = history.history['loss']
         plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.024228806743490226
Test accuracy: 0.993399977684021
```



# 3.3: ConvNet(256-64-32) | 3 Dropouts | 3 MaxPools | Dense(128-32) | 1 Flatten | ReLU | Adam | Padding: "same"

In [11]:

```
model = Sequential()
model.add(Conv2D(256, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape,
                padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (7, 7), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.75))

model.add(Conv2D(32, (7, 7), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

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

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))


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

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))
```

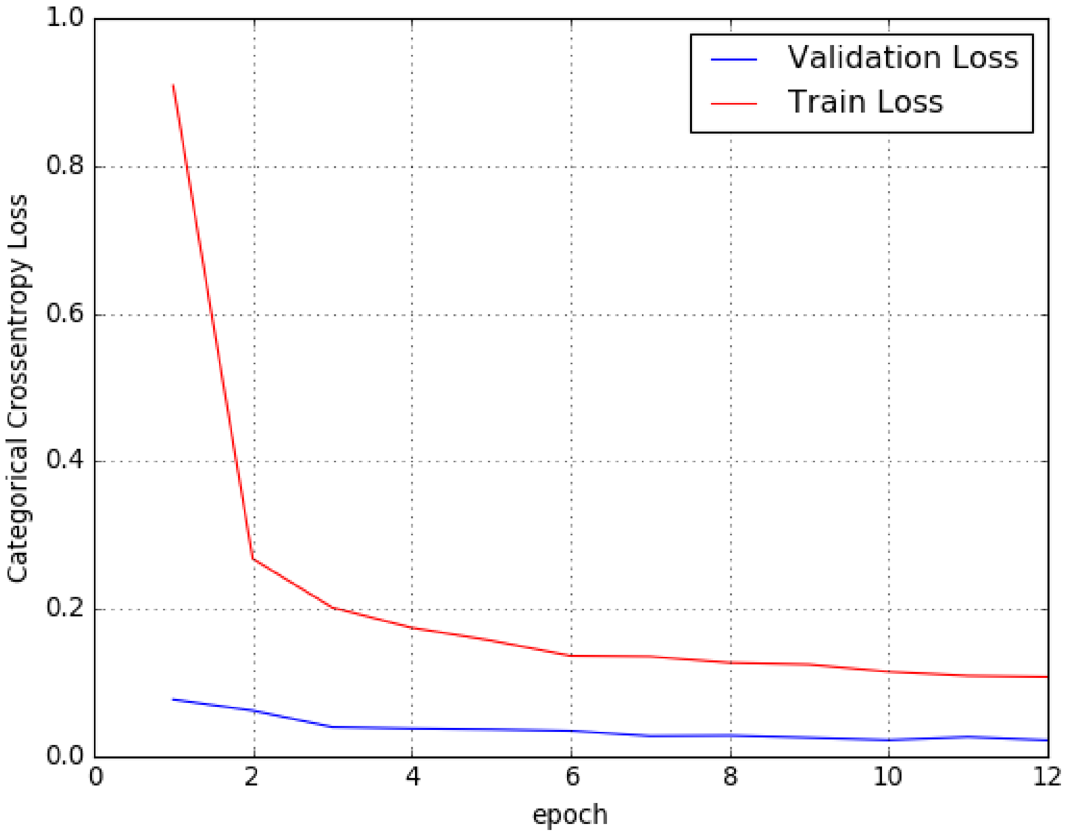
```
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropou
t() uses dropout rate instead of keep_prob. Please ensure that this is intend
ed.
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 561s 9ms/step - loss: 0.9087 -
accuracy: 0.6829 - val_loss: 0.0773 - val_accuracy: 0.9769
Epoch 2/12
60000/60000 [==============================] - 558s 9ms/step - loss: 0.2679 -
accuracy: 0.9274 - val_loss: 0.0624 - val_accuracy: 0.9827
Epoch 3/12
60000/60000 [==============================] - 557s 9ms/step - loss: 0.2020 -
accuracy: 0.9488 - val_loss: 0.0402 - val_accuracy: 0.9890
Epoch 4/12
60000/60000 [==============================] - 558s 9ms/step - loss: 0.1746 -
accuracy: 0.9567 - val_loss: 0.0382 - val_accuracy: 0.9894
Epoch 5/12
60000/60000 [==============================] - 562s 9ms/step - loss: 0.1573 -
accuracy: 0.9624 - val_loss: 0.0371 - val_accuracy: 0.9905
Epoch 6/12
```

```
60000/60000 [==============================] - 559s 9ms/step - loss: 0.1367 -
accuracy: 0.9657 - val_loss: 0.0351 - val_accuracy: 0.9908
Epoch 7/12
60000/60000 [==============================] - 558s 9ms/step - loss: 0.1356 -
accuracy: 0.9660 - val_loss: 0.0280 - val_accuracy: 0.9919
Epoch 8/12
60000/60000 [==============================] - 560s 9ms/step - loss: 0.1272 -
accuracy: 0.9678 - val_loss: 0.0286 - val_accuracy: 0.9932
Epoch 9/12
60000/60000 [==============================] - 559s 9ms/step - loss: 0.1249 -
accuracy: 0.9700 - val_loss: 0.0257 - val_accuracy: 0.9933
Epoch 10/12
60000/60000 [==============================] - 560s 9ms/step - loss: 0.1151 -
accuracy: 0.9710 - val_loss: 0.0226 - val_accuracy: 0.9932
Epoch 11/12
60000/60000 [==============================] - 550s 9ms/step - loss: 0.1095 -
accuracy: 0.9727 - val_loss: 0.0267 - val_accuracy: 0.9933
Epoch 12/12
60000/60000 [==============================] - 549s 9ms/step - loss: 0.1081 -
accuracy: 0.9737 - val_loss: 0.0224 - val_accuracy: 0.9927
```

In [12]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.022369634967670984
Test accuracy: 0.9926999807357788
```



## 3.4: ConvNet(64-32) | 2 Dropouts | 2 MaxPools | Dense(64-32) 1 Flatten | ReLU | Adam | Padding: "same" | 2 Batch Normalization

In [15]:

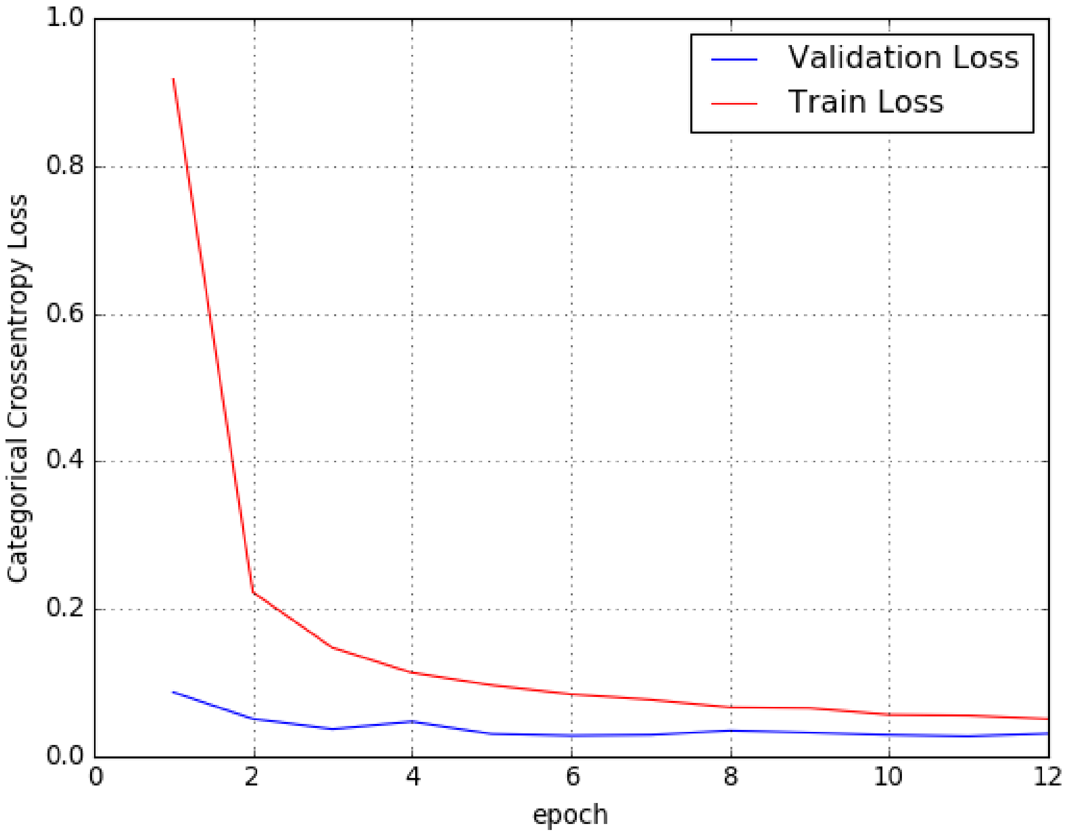
```
model = Sequential()
model.add(Conv2D(64, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape,
                padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())

model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

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

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 42s 705us/step - loss: 0.9177
- accuracy: 0.7233 - val_loss: 0.0872 - val_accuracy: 0.9808
Epoch 2/12
60000/60000 [==============================] - 41s 676us/step - loss: 0.2229
- accuracy: 0.9441 - val_loss: 0.0513 - val_accuracy: 0.9840
Epoch 3/12
60000/60000 [==============================] - 41s 683us/step - loss: 0.1476
- accuracy: 0.9631 - val_loss: 0.0375 - val_accuracy: 0.9872
Epoch 4/12
60000/60000 [==============================] - 40s 671us/step - loss: 0.1138
- accuracy: 0.9717 - val_loss: 0.0474 - val_accuracy: 0.9867
Epoch 5/12
60000/60000 [==============================] - 40s 665us/step - loss: 0.0968
- accuracy: 0.9755 - val_loss: 0.0309 - val_accuracy: 0.9903
Epoch 6/12
60000/60000 [==============================] - 40s 671us/step - loss: 0.0846
- accuracy: 0.9786 - val_loss: 0.0284 - val_accuracy: 0.9920
Epoch 7/12
60000/60000 [==============================] - 40s 669us/step - loss: 0.0772
- accuracy: 0.9800 - val_loss: 0.0295 - val_accuracy: 0.9919
Epoch 8/12
60000/60000 [==============================] - 40s 667us/step - loss: 0.0668
```

```
- accuracy: 0.9827 - val_loss: 0.0354 - val_accuracy: 0.9900
Epoch 9/12
60000/60000 [==============================] - 40s 664us/step - loss: 0.0658
- accuracy: 0.9829 - val_loss: 0.0326 - val_accuracy: 0.9908
Epoch 10/12
60000/60000 [==============================] - 40s 665us/step - loss: 0.0569
- accuracy: 0.9856 - val_loss: 0.0294 - val_accuracy: 0.9913
Epoch 11/12
60000/60000 [==============================] - 40s 664us/step - loss: 0.0558
- accuracy: 0.9860 - val_loss: 0.0276 - val_accuracy: 0.9925
Epoch 12/12
60000/60000 [==============================] - 40s 667us/step - loss: 0.0511
- accuracy: 0.9873 - val_loss: 0.0314 - val_accuracy: 0.9920
```

In [16]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', 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_epo

# we will get val_loss and val_acc only when you pass the paramter validation_dat
# 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


vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.03136349078471503
Test accuracy: 0.9919999837875366
```



# Conclusion:

In [11]:

```
from prettytable import PrettyTable
```

In [12]:

```
x = PrettyTable()
x.field_names = ["convolution kernel", "architecture", "Test Accuracy"]
```

In [14]:

```
x.add_row(["3x3", "ConvNet(32-64)|2 Dropouts|Maxpool|Dense(128-10)|1 Flatten|ReLU
x.add_row(["3x3", "ConvNet(32-32-64)|1 Dropouts|3 MaxPools|Dense(64-10)|1 Flatten
x.add_row(["3x3", "ConvNet(32-64)|2 Dropouts|2 MaxPools|Dense(128-10)|1 Flatten|Re
x.add_row(["3x3", "ConvNet(32-32)|2 Dropouts|2 MaxPools|Dense(64-10)|1 Flatten|Re

x.add_row(["5x5", "ConvNet(128-64-32) | 3 Dropouts | 2 Maxpool | Dense(128-10) |
x.add_row(["5x5", "ConvNet(64-32) | 3 Dropouts | 3 MaxPools | Dense(128-32) | 1 F
x.add_row(["5x5", "ConvNet(128-64-32) | 3 Dropouts | 3 MaxPools | Dense(64-32) |
x.add_row(["5x5", "ConvNet(64-32) | 3 Dropouts | 2 MaxPools | Dense(64-32) | 1 Fl

x.add_row(["7x7", "ConvNet (128-64-32) | Dense (128) | 3 Dropouts | 2 Maxpool | 1
x.add_row(["7x7", "ConvNet(128-64) | 3 Dropouts | 2 MaxPools | 1 Flatten | ReLU |
x.add_row(["7x7", "ConvNet(256-64-32) | 3 Dropouts | 3 MaxPools | Dense(128-32) |
x.add_row(["7x7", "ConvNet(64-32) | 2 Dropouts | 2 MaxPools | Dense(64-32) 1 Flat
print(x)
```

```
+--------------------+----------------------------------------------------------
------------------------------------------------------------------+-------------
---+
| convolution kernel |                                                         a
rchitecture                                                       | Test Accura
cy |
+--------------------+----------------------------------------------------------
------------------------------------------------------------------+-------------
---+
|        3x3         |                    ConvNet(32-64) | 2 Dropouts | Maxpool
| Dense(128-10) | 1 Flatten | ReLU | Adadelta                     |    0.9914
|
|        3x3         |                    ConvNet(32-32-64) | 1 Dropouts | 3 Ma
xPools | Dense(64-10) |1 Flatten | ReLU | Adam                      |    0.9843
|
|        3x3         |            ConvNet(32-64) | 2 Dropouts | 2 MaxPools | De
nse(128-10) | 1 Flatten | ReLU | Adam | Padding: (same)           |    0.9933
|
|        3x3         | ConvNet(32-32) | 2 Dropouts | 2 MaxPools | Dense(64-10)
| 1 Flatten | ReLU | Adam | Padding: (same) | Batch Normalization |    0.9933
|
|        5x5         |                   ConvNet(128-64-32) | 3 Dropouts | 2 Ma
xpool | Dense(128-10) | 1 Flatten | ReLU | Adam                     |    0.9944
|
|        5x5         |                     ConvNet(64-32) | 3 Dropouts | 3 MaxP
ools | Dense(128-32) | 1 Flatten | ReLU | Adam                      |    0.9918
|
|        5x5         |          ConvNet(128-64-32) | 3 Dropouts | 3 MaxPools |
Dense(64-32) | 1 Flatten | ReLU | Adam | Padding: (same)          |    0.9917
|
|        5x5         | ConvNet(64-32) | 3 Dropouts | 2 MaxPools | Dense(64-32)
| 1 Flatten | ReLU | Adam | Padding: (same) | Batch Normalization |    0.9927
|
|        7x7         |                    ConvNet (128-64-32) | Dense (128) | 3
Dropouts | 2 Maxpool | 1 Flatten | ReLU | Adam                   |    0.9947
|
|        7x7         |            ConvNet(128-64) | 3 Dropouts | 2 MaxPools | 1
Flatten | ReLU | Dense(64-10) | Adam | padding: (same)          |    0.9933
|
|        7x7         |          ConvNet(256-64-32) | 3 Dropouts | 3 MaxPools |
Dense(128-32) | 1 Flatten | ReLU | Adam | Padding: (same)         |    0.9926
|
|        7x7         | ConvNet(64-32) | 2 Dropouts | 2 MaxPools | Dense(64-32)
1 Flatten | ReLU | Adam | Padding: (same) | 2 Batch Normalization |    0.9919
|
|        3x3         |                          ConvNet(32-64)|2 Dropouts|Maxpo
ol|Dense(128-10)|1 Flatten|ReLU|Adadelta                           |    0.9914
|
|        3x3         |                          ConvNet(32-32-64)|1 Dropouts|3
MaxPools|Dense(64-10)|1 Flatten|ReLU|Adam                          |    0.9843
|
|        3x3         |                   ConvNet(32-64)|2 Dropouts|2 MaxPools|D
ense(128-10)|1 Flatten|ReLU|Adam|Padding:(same)                   |    0.9933
|
|        3x3         |                  ConvNet(32-32)|2 Dropouts|2 MaxPools|De
nse(64-10)|1 Flatten|ReLU|Adam|Padding: (same)|BN                 |    0.9933
|
|        5x5         |                   ConvNet(128-64-32) | 3 Dropouts | 2 Ma
xpool | Dense(128-10) | 1 Flatten | ReLU | Adam                    |    0.9944
|
|        5x5         |                     ConvNet(64-32) | 3 Dropouts | 3 MaxP
ools | Dense(128-32) | 1 Flatten | ReLU | Adam                     |    0.9918
|
|        5x5         |          ConvNet(128-64-32) | 3 Dropouts | 3 MaxPools |
Dense(64-32) | 1 Flatten | ReLU | Adam | Padding: (same)          |    0.9917
|
|        5x5         | ConvNet(64-32) | 3 Dropouts | 2 MaxPools | Dense(64-32)
| 1 Flatten | ReLU | Adam | Padding: (same) | Batch Normalization |    0.9927
|
|        7x7         |                    ConvNet (128-64-32) | Dense (128) | 3
Dropouts | 2 Maxpool | 1 Flatten | ReLU | Adam                   |    0.9947
|
|        7x7         |            ConvNet(128-64) | 3 Dropouts | 2 MaxPools | 1
Flatten | ReLU | Dense(64-10) | Adam | padding: (same)           |    0.9933
|
|        7x7         |          ConvNet(256-64-32) | 3 Dropouts | 3 MaxPools |
Dense(128-32) | 1 Flatten | ReLU | Adam | Padding: (same)         |    0.9926
|
|        7x7         | ConvNet(64-32) | 2 Dropouts | 2 MaxPools | Dense(64-32)
1 Flatten | ReLU | Adam | Padding: (same) | 2 Batch Normalization |    0.9919
|
+--------------------+----------------------------------------------------------
------------------------------------------------------------------+-----------
---+
```

# Procedure Followed:

- Splitted the MNIST dataset into train and test
- Tried different architectures of CNN with dataset like with muktiple dropout, diffent kernel size, different convolution layers, etc.
- Plotted the epoch vs Train/Test loss of each model