Various CNN networks on MNIST dataset

In [1]:

```
# To ignore warnings in the below code

import warnings
warnings.filterwarnings("ignore")
```

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
from keras.initializers import he_normal
from keras.layers.normalization import BatchNormalization

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

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

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

x_train = x_train.astype('float32')
x_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)
```

```
Using TensorFlow backend.
```

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

In [3]:

```
%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()
```

1. Model 1: CNN with 3 ConvNets and with kernel size 3x3

In [7]:

```
# MODEL 1
model1 = Sequential()

#convnet 1
model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))

#convnet 2
model1.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model1.add(Dropout(0.25))

#convnet 3
model1.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))
model1.add(Flatten())

#hidden layer
model1.add(Dense(256, activation='relu'))
model1.add(Dropout(0.5))

model1.add(Dense(num_classes, activation='softmax'))

model1.compile(loss=keras.losses.categorical_crossentropy,
               optimizer=keras.optimizers.Adadelta(),
               metrics=['accuracy'])

print(model1.summary())
history = model1.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

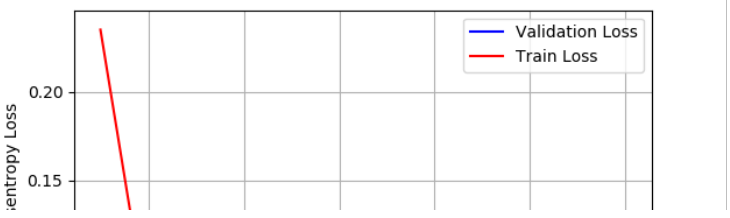
```
WARNING:tensorflow:From C:\Users\NIKHITHA\Anaconda3\lib\site-packages\keras\optimizers.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\NIKHITHA\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3295: The name tf.log is deprecated. Please use tf.math.log instead.


_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
dropout_4 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 22, 22, 128)       73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 11, 11, 128)       0
_________________________________________________________________
flatten_4 (Flatten)          (None, 15488)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               3965184
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
```

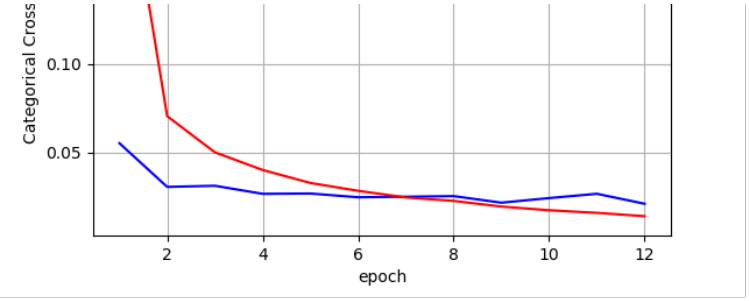
```
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 4,060,426
Trainable params: 4,060,426
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From C:\Users\NIKHITHA\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.2351 - acc: 0.9265 - val_lo
ss: 0.0554 - val_acc: 0.9825
Epoch 2/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.0707 - acc: 0.9789 - val_lo
ss: 0.0307 - val_acc: 0.9887
Epoch 3/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.0502 - acc: 0.9846 - val_lo
ss: 0.0314 - val_acc: 0.9885
Epoch 4/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.0403 - acc: 0.9873 - val_lo
ss: 0.0268 - val_acc: 0.9909
Epoch 5/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.0329 - acc: 0.9902 - val_lo
ss: 0.0270 - val_acc: 0.9908
Epoch 6/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.0285 - acc: 0.9917 - val_lo
ss: 0.0249 - val_acc: 0.9922
Epoch 7/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.0248 - acc: 0.9925 - val_lo
ss: 0.0252 - val_acc: 0.9919
Epoch 8/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.0228 - acc: 0.9926 - val_lo
ss: 0.0255 - val_acc: 0.9916
Epoch 9/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.0196 - acc: 0.9940 - val_lo
ss: 0.0218 - val_acc: 0.9931
Epoch 10/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.0175 - acc: 0.9948 - val_lo
ss: 0.0244 - val_acc: 0.9929
Epoch 11/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.0161 - acc: 0.9948 - val_lo
ss: 0.0268 - val_acc: 0.9922
Epoch 12/12
60000/60000 [==============================] - 195s 3ms/step - loss: 0.0142 - acc: 0.9954 - val_lo
ss: 0.0212 - val_acc: 0.9938
```

In [8]:

```
score = model1.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')
x=list(range(1,epochs+1))
vy=history.history['val_loss'] #validation loss
ty=history.history['loss'] # train loss
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.0212051354131494
Test accuracy: 0.9938
```





2. Model 2: CNN with 5 ConvNets and with kernel size 5x5

In [16]:

```
# MODEL 1
model2 = Sequential()

#convnet 1
model2.add(Conv2D(32, kernel_size=(5, 5), padding = 'same', activation='relu',
input_shape=input_shape))

#convnet 2
model2.add(Conv2D(64, kernel_size=(5, 5), padding = 'same', activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(BatchNormalization())

#convnet 3
model2.add(Conv2D(96, kernel_size=(5, 5), padding = 'same', activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.25))

#convnet 4
model2.add(Conv2D(108, kernel_size=(5, 5), padding = 'same', activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
#model2.add(Dropout(0.25))

#convnet 5
model2.add(Conv2D(164, kernel_size=(5, 5), padding = 'same', activation='relu'))
model2.add(Dropout(0.15))
model2.add(Flatten())

#hidden layer
model2.add(Dense(256, activation='relu', kernel_initializer=he_normal(seed=None)))
model2.add(BatchNormalization())
model2.add(Dropout(0.5))

model2.add(Dense(num_classes, activation='softmax'))

model2.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

print(model2.summary())

history = model2.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

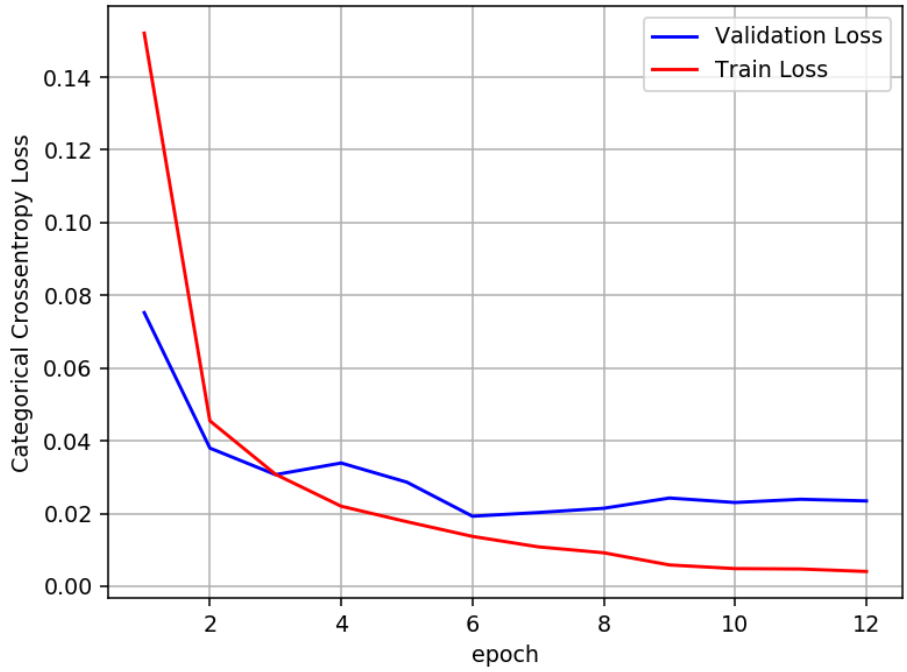
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_45 (Conv2D)           (None, 28, 28, 32)        832
```

```
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 28, 28, 64)        51264
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
batch_normalization_10 (Batc (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 14, 14, 96)        153696
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 7, 7, 96)          0
_________________________________________________________________
dropout_22 (Dropout)         (None, 7, 7, 96)          0
_________________________________________________________________
conv2d_48 (Conv2D)           (None, 7, 7, 108)         259308
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 3, 3, 108)         0
_________________________________________________________________
conv2d_49 (Conv2D)           (None, 3, 3, 164)         442964
_________________________________________________________________
dropout_23 (Dropout)         (None, 3, 3, 164)         0
_________________________________________________________________
flatten_11 (Flatten)         (None, 1476)              0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               378112
_________________________________________________________________
batch_normalization_11 (Batc (None, 256)               1024
_________________________________________________________________
dropout_24 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,290,026
Trainable params: 1,289,386
Non-trainable params: 640
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 411s 7ms/step - loss: 0.1521 - acc: 0.9537 - val_lo
ss: 0.0753 - val_acc: 0.9793
Epoch 2/12
60000/60000 [==============================] - 409s 7ms/step - loss: 0.0455 - acc: 0.9865 - val_lo
ss: 0.0380 - val_acc: 0.9879
Epoch 3/12
60000/60000 [==============================] - 406s 7ms/step - loss: 0.0308 - acc: 0.9904 - val_lo
ss: 0.0307 - val_acc: 0.9914
Epoch 4/12
60000/60000 [==============================] - 387s 6ms/step - loss: 0.0220 - acc: 0.9934 - val_lo
ss: 0.0339 - val_acc: 0.9899
Epoch 5/12
60000/60000 [==============================] - 378s 6ms/step - loss: 0.0177 - acc: 0.9946 - val_lo
ss: 0.0286 - val_acc: 0.9907
Epoch 6/12
60000/60000 [==============================] - 376s 6ms/step - loss: 0.0137 - acc: 0.9956 - val_lo
ss: 0.0193 - val_acc: 0.9943
Epoch 7/12
60000/60000 [==============================] - 376s 6ms/step - loss: 0.0108 - acc: 0.9970 - val_lo
ss: 0.0203 - val_acc: 0.9937
Epoch 8/12
60000/60000 [==============================] - 381s 6ms/step - loss: 0.0092 - acc: 0.9972 - val_lo
ss: 0.0214 - val_acc: 0.9934
Epoch 9/12
60000/60000 [==============================] - 371s 6ms/step - loss: 0.0058 - acc: 0.9981 - val_lo
ss: 0.0243 - val_acc: 0.9940
Epoch 10/12
60000/60000 [==============================] - 313s 5ms/step - loss: 0.0048 - acc: 0.9986 - val_lo
ss: 0.0230 - val_acc: 0.9940
Epoch 11/12
60000/60000 [==============================] - 311s 5ms/step - loss: 0.0047 - acc: 0.9987 - val_lo
ss: 0.0239 - val_acc: 0.9942
Epoch 12/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.0040 - acc: 0.9987 - val_lo
ss: 0.0235 - val_acc: 0.9932
```

In [17]:

```
score = model2.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')
x=list(range(1,epochs+1))
vy=history.history['val_loss'] #validation loss
ty=history.history['loss'] # train loss
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.023452779674043495
Test accuracy: 0.9932
```



3. Model 3: CNN with 7 ConvNets and with kernel size 7x7

In [26]:

```
# MODEL 1
model3 = Sequential()
#convnet 1
model3.add(Conv2D(32, kernel_size=(7, 7), padding = 'same', activation='relu',
input_shape=input_shape))

#convnet 2
model3.add(Conv2D(64, kernel_size=(7, 7), strides = (1, 1), activation='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))

#convnet 3
model3.add(Conv2D(96, kernel_size=(7, 7), padding = 'same', activation='relu'))
model3.add(BatchNormalization())
model3.add(Dropout(0.3))

#convnet 4
model3.add(Conv2D(108, kernel_size=(7, 7), padding = 'same', activation='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.25))

#convnet 5
model3.add(Conv2D(124, kernel_size=(7, 7), padding = 'same', activation='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(BatchNormalization())
#model3.add(Flatten())

#convnet 6
```

```
model3.add(Conv2D(162, kernel_size=(7, 7), padding = 'same', activation='relu'))
model3.add(BatchNormalization())
model3.add(Dropout(0.4))
#model3.add(Flatten())

#convnet 7
model3.add(Conv2D(198, kernel_size=(7, 7), padding = 'same', activation='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Flatten())

#hidden layer
model3.add(Dense(256, activation='relu', kernel_initializer=he_normal(seed=None)))
model3.add(BatchNormalization())
model3.add(Dropout(0.5))

#hidden layer
model3.add(Dense(128, activation='relu', kernel_initializer=he_normal(seed=None)))
model3.add(BatchNormalization())
model3.add(Dropout(0.5))

model3.add(Dense(num_classes, activation='softmax'))

model3.compile(loss=keras.losses.categorical_crossentropy,
               optimizer=keras.optimizers.Adadelta(),
               metrics=['accuracy'])
print(model3.summary())

history = model3.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

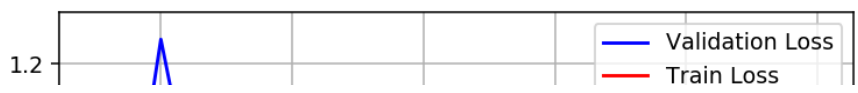
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_90 (Conv2D)           (None, 28, 28, 32)        1600
_________________________________________________________________
conv2d_91 (Conv2D)           (None, 22, 22, 64)        100416
_________________________________________________________________
max_pooling2d_43 (MaxPooling (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_92 (Conv2D)           (None, 11, 11, 96)        301152
_________________________________________________________________
batch_normalization_21 (Batc (None, 11, 11, 96)        384
_________________________________________________________________
dropout_41 (Dropout)         (None, 11, 11, 96)        0
_________________________________________________________________
conv2d_93 (Conv2D)           (None, 11, 11, 108)       508140
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 5, 5, 108)         0
_________________________________________________________________
dropout_42 (Dropout)         (None, 5, 5, 108)         0
_________________________________________________________________
conv2d_94 (Conv2D)           (None, 5, 5, 124)         656332
_________________________________________________________________
max_pooling2d_45 (MaxPooling (None, 2, 2, 124)         0
_________________________________________________________________
batch_normalization_22 (Batc (None, 2, 2, 124)         496
_________________________________________________________________
conv2d_95 (Conv2D)           (None, 2, 2, 162)         984474
_________________________________________________________________
batch_normalization_23 (Batc (None, 2, 2, 162)         648
_________________________________________________________________
dropout_43 (Dropout)         (None, 2, 2, 162)         0
_________________________________________________________________
conv2d_96 (Conv2D)           (None, 2, 2, 198)         1571922
_________________________________________________________________
max_pooling2d_46 (MaxPooling (None, 1, 1, 198)         0
_________________________________________________________________
flatten_14 (Flatten)         (None, 198)               0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               50944
_________________________________________________________________
batch_normalization_24 (Batc (None, 256)               1024
_________________________________________________________________
```

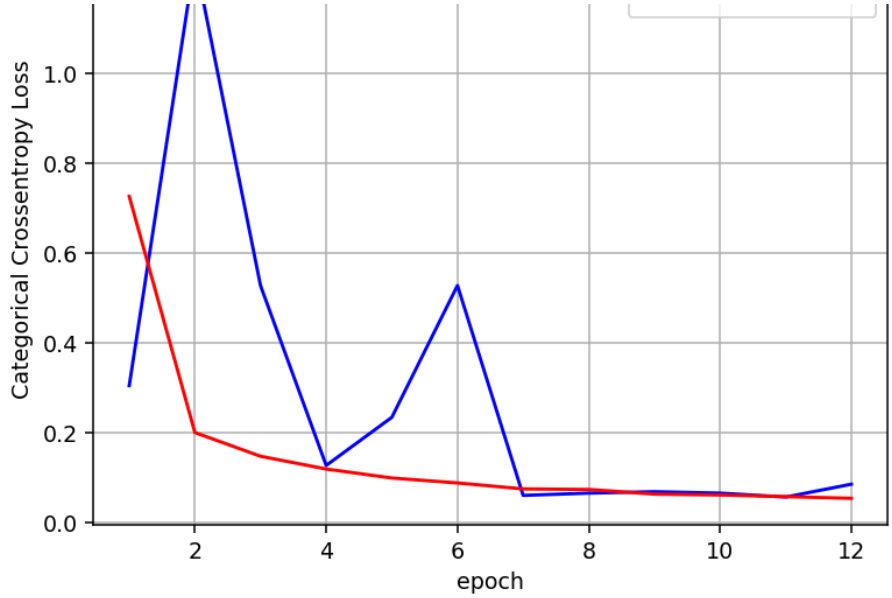
```
_________________________________________________________________
dropout_44 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               32896
_________________________________________________________________
batch_normalization_25 (Batc (None, 128)               512
_________________________________________________________________
dropout_45 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_11 (Dense)             (None, 10)                1290
=================================================================
Total params: 4,212,230
Trainable params: 4,210,698
Non-trainable params: 1,532
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 827s 14ms/step - loss: 0.7266 - acc: 0.7776 - val_l
oss: 0.3053 - val_acc: 0.9263
Epoch 2/12
60000/60000 [==============================] - 925s 15ms/step - loss: 0.2009 - acc: 0.9422 - val_l
oss: 1.2542 - val_acc: 0.7638
Epoch 3/12
60000/60000 [==============================] - 886s 15ms/step - loss: 0.1481 - acc: 0.9582 - val_l
oss: 0.5279 - val_acc: 0.8438
Epoch 4/12
60000/60000 [==============================] - 858s 14ms/step - loss: 0.1197 - acc: 0.9660 - val_l
oss: 0.1277 - val_acc: 0.9613
Epoch 5/12
60000/60000 [==============================] - 840s 14ms/step - loss: 0.0997 - acc: 0.9712 - val_l
oss: 0.2348 - val_acc: 0.9265
Epoch 6/12
60000/60000 [==============================] - 808s 13ms/step - loss: 0.0887 - acc: 0.9744 - val_l
oss: 0.5283 - val_acc: 0.8446
Epoch 7/12
60000/60000 [==============================] - 805s 13ms/step - loss: 0.0755 - acc: 0.9787 - val_l
oss: 0.0610 - val_acc: 0.9832
Epoch 8/12
60000/60000 [==============================] - 989s 16ms/step - loss: 0.0742 - acc: 0.9788 - val_l
oss: 0.0661 - val_acc: 0.9814
Epoch 9/12
60000/60000 [==============================] - 1131s 19ms/step - loss: 0.0639 - acc: 0.9816 - val_
loss: 0.0693 - val_acc: 0.9821
Epoch 10/12
60000/60000 [==============================] - 1130s 19ms/step - loss: 0.0619 - acc: 0.9818 - val_
loss: 0.0661 - val_acc: 0.9813
Epoch 11/12
60000/60000 [==============================] - 1187s 20ms/step - loss: 0.0584 - acc: 0.9834 - val_
loss: 0.0573 - val_acc: 0.9835
Epoch 12/12
60000/60000 [==============================] - 1179s 20ms/step - loss: 0.0545 - acc: 0.9842 - val_
loss: 0.0860 - val_acc: 0.9743
```

In [27]:

```
score = model3.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')
x=list(range(1,epochs+1))
vy=history.history['val_loss'] #validation loss
ty=history.history['loss'] # train loss
plt_dynamic(x, vy, ty, ax)
```

```
Test loss: 0.08595889861029572
Test accuracy: 0.9743
```





4. Conclusion

In [28]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Index", "Model Name", "Accuracy"]
x.add_row(["1", '3 CNN with kernel 3x3', 0.9938])
x.add_row(["2", '5 CNN with kernel 5x5', 0.9932])
x.add_row(["3", '7 CNN with kernel 7x7', 0.9743])

print(x)
```

```
+-------+-----------------------+----------+
| Index |       Model Name      | Accuracy |
+-------+-----------------------+----------+
|   1   | 3 CNN with kernel 3x3 |  0.9938  |
|   2   | 5 CNN with kernel 5x5 |  0.9932  |
|   3   | 7 CNN with kernel 7x7 |  0.9743  |
+-------+-----------------------+----------+
```

In [ ]: