Assignment:Different CNN Architectures on MNIST dataset

In [1]:
```
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
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

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()
```

```
Using TensorFlow backend.
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
```

In [2]:
```
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)
```

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

In [0]:
```
# this function is used draw Categorical Crossentropy Loss VS No. of epochs plot
def plt_dynamic(x, vy, ty):
  ax.plot(x, vy, 'b', label="Validation Loss")
  ax.plot(x, ty, 'r', label="Train Loss")
  plt.legend()
  plt.grid()
  plt.show()
```

CNN with 3 Convolutional layers and kernel size - (3X3)

In [4]:
```
# Initialising the model
model3 = Sequential()
```

```
# Adding first conv layer
model3.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=
input_shape))

# Adding second conv layer
model3.add(Conv2D(64, (3, 3), activation='relu'))

# Adding Maxpooling layer
model3.add(MaxPooling2D(pool_size=(2, 2)))

# Adding Dropout
model3.add(Dropout(0.25))

# Adding third conv layer
model3.add(Conv2D(128, (3, 3), activation='relu'))

# Adding Maxpooling layer
model3.add(MaxPooling2D(pool_size=(2, 2)))

# Adding Dropout
model3.add(Dropout(0.25))

# Adding flatten layer
model3.add(Flatten())

# Adding first hidden layer
model3.add(Dense(256, activation='relu',kernel_initializer=he_normal(se
ed=None)))

# Adding Dropout
model3.add(Dropout(0.5))

# Adding output layer
model3.add(Dense(num_classes, activation='softmax'))

# Printing model Summary
print(model3.summary())

# Compiling the model
```

```
model3.compile(optimizer='adam', loss='categorical_crossentropy', metri
cs=['accuracy'])

# Fitting the data to the model
history = model3.fit(x_train, y_train,batch_size=batch_size,epochs=epoc
hs,verbose=1,validation_data=(x_test, y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorfl
ow/python/framework/op_def_library.py:263: colocate_with (from tensorfl
ow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:3445: calling dropout (from tensorflow.pyth
on.ops.nn_ops) with keep_prob is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate =
1 - keep_prob`.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               819456
```

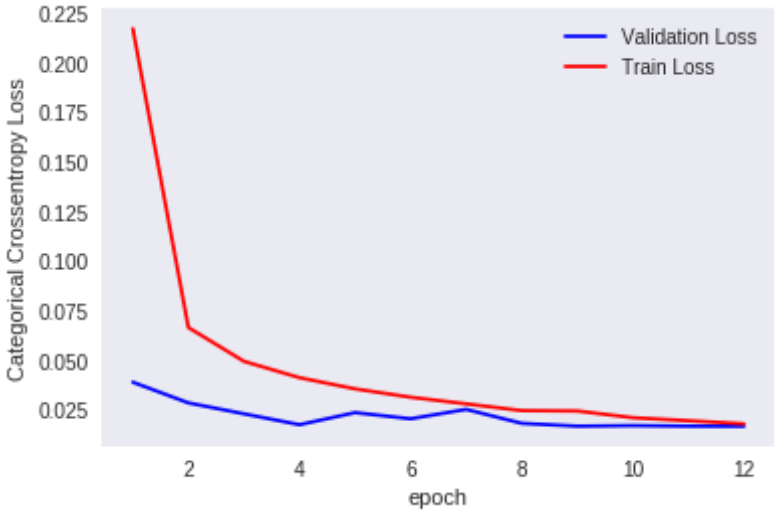
```
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 914,698
Trainable params: 914,698
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorfl
ow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.ma
th_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 229s 4ms/step - loss: 0.
2175 - acc: 0.9307 - val_loss: 0.0392 - val_acc: 0.9871
Epoch 2/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.
0667 - acc: 0.9797 - val_loss: 0.0287 - val_acc: 0.9902
Epoch 3/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0497 - acc: 0.9852 - val_loss: 0.0232 - val_acc: 0.9918
Epoch 4/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0414 - acc: 0.9873 - val_loss: 0.0177 - val_acc: 0.9944
Epoch 5/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0358 - acc: 0.9887 - val_loss: 0.0238 - val_acc: 0.9918
Epoch 6/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0316 - acc: 0.9900 - val_loss: 0.0207 - val_acc: 0.9933
Epoch 7/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0282 - acc: 0.9911 - val_loss: 0.0255 - val_acc: 0.9907
Epoch 8/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
```

```
0248 - acc: 0.9921 - val_loss: 0.0184 - val_acc: 0.9937
Epoch 9/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0246 - acc: 0.9922 - val_loss: 0.0170 - val_acc: 0.9945
Epoch 10/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.
0213 - acc: 0.9933 - val_loss: 0.0173 - val_acc: 0.9941
Epoch 11/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.
0197 - acc: 0.9940 - val_loss: 0.0170 - val_acc: 0.9947
Epoch 12/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.
0182 - acc: 0.9939 - val_loss: 0.0169 - val_acc: 0.9948
```

In [5]:

```
# Evaluating the model
score = model3.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')

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,epochs+1))

# Validation loss
vy = history.history['val_loss']
# Training loss
ty = history.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Test score: 0.016924540818311743
Test accuracy: 0.9948
```



CNN with 5 Convolutional layers and kernel size - (5X5)

In [6]:
```
# Initialising the model
model5 = Sequential()

# Adding first conv layer
model5.add(Conv2D(10, kernel_size=(5, 5),padding='same',activation='rel
u',input_shape=input_shape))

# Adding second conv layer
model5.add(Conv2D(20, (5, 5), activation='relu'))

# Adding Maxpooling layer
model5.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

# Adding Dropout
model5.add(Dropout(0.25))

# Adding third conv layer
```

```
model5.add(Conv2D(35, (5, 5),padding='same', activation='relu'))

# Adding Maxpooling layer
model5.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

# Adding Dropout
model5.add(Dropout(0.25))

# Adding fourth conv layer
model5.add(Conv2D(70, (5, 5),padding='same',activation='relu'))

# Adding fifth conv layer
model5.add(Conv2D(80, (5, 5), activation='relu'))

# Adding Maxpooling layer
model5.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

# Adding Dropout
model5.add(Dropout(0.25))

# Adding flatten layer
model5.add(Flatten())

# Adding first hidden layer
model5.add(Dense(256, activation='relu',kernel_initializer=he_normal(se
ed=None)))

# Adding Batch Normalization
model5.add(BatchNormalization())

# Adding Dropout
model5.add(Dropout(0.5))


# Adding output layer
model5.add(Dense(num_classes, activation='softmax'))

# Printing model Summary
print(model5.summary())
```

```
# Compiling the model
model5.compile(optimizer='adam', loss='categorical_crossentropy', metri
cs=['accuracy'])

# Fitting the data to the model
history= model5.fit(x_train, y_train,batch_size=batch_size,epochs=epoch
s,verbose=1,validation_data=(x_test, y_test))
```

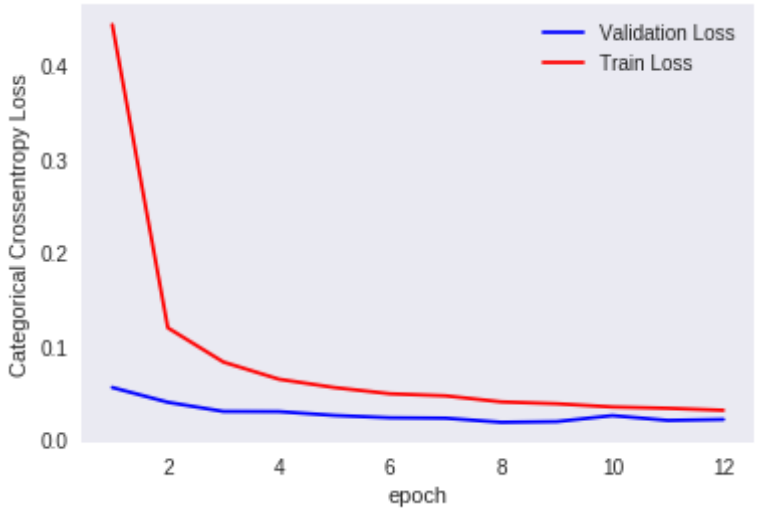
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 28, 28, 10)        260
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 24, 24, 20)        5020
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 20)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 12, 12, 20)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 35)        17535
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 35)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 6, 6, 35)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 6, 70)          61320
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 2, 2, 80)          140080
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 80)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 1, 1, 80)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 80)                0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               20736
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
```

```
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 248,545
Trainable params: 248,033
Non-trainable params: 512
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
4436 - acc: 0.8613 - val_loss: 0.0557 - val_acc: 0.9809
Epoch 2/12
60000/60000 [==============================] - 231s 4ms/step - loss: 0.
1195 - acc: 0.9660 - val_loss: 0.0399 - val_acc: 0.9879
Epoch 3/12
60000/60000 [==============================] - 231s 4ms/step - loss: 0.
0829 - acc: 0.9772 - val_loss: 0.0301 - val_acc: 0.9902
Epoch 4/12
60000/60000 [==============================] - 231s 4ms/step - loss: 0.
0645 - acc: 0.9817 - val_loss: 0.0299 - val_acc: 0.9911
Epoch 5/12
60000/60000 [==============================] - 231s 4ms/step - loss: 0.
0556 - acc: 0.9842 - val_loss: 0.0258 - val_acc: 0.9921
Epoch 6/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
0490 - acc: 0.9864 - val_loss: 0.0233 - val_acc: 0.9928
Epoch 7/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
0468 - acc: 0.9872 - val_loss: 0.0229 - val_acc: 0.9929
Epoch 8/12
60000/60000 [==============================] - 231s 4ms/step - loss: 0.
0403 - acc: 0.9887 - val_loss: 0.0185 - val_acc: 0.9950
Epoch 9/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
0383 - acc: 0.9896 - val_loss: 0.0192 - val_acc: 0.9941
Epoch 10/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
```

```
0349 - acc: 0.9902 - val_loss: 0.0256 - val_acc: 0.9919
Epoch 11/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
0334 - acc: 0.9904 - val_loss: 0.0205 - val_acc: 0.9941
Epoch 12/12
60000/60000 [==============================] - 232s 4ms/step - loss: 0.
0313 - acc: 0.9912 - val_loss: 0.0215 - val_acc: 0.9932
```

In [7]:
```
# Evaluating the model
score = model5.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')

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,epochs+1))

# Validation loss
vy = history.history['val_loss']
# Training loss
ty = history.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Test score: 0.021511626279931806
Test accuracy: 0.9932
```



CNN with 7 Convolutional layers and kernel size - (2X2)

In [8]:
```
model7 = Sequential()

# Adding first conv layer
model7.add(Conv2D(10, kernel_size=(2, 2),padding='same',activation='rel
u',input_shape=input_shape))

# Adding second conv layer
model7.add(Conv2D(20, (2, 2), activation='relu'))

# Adding Maxpooling layer
model7.add(MaxPooling2D(pool_size=(3, 3), strides=(1,1)))

# Adding Dropout
model7.add(Dropout(0.3))

# Adding third conv layer
model7.add(Conv2D(40, (2, 2), activation='relu'))
```

```
# Adding Maxpooling layer
model7.add(MaxPooling2D(pool_size=(2, 2),padding='same'))


# Adding fourth conv layer
model7.add(Conv2D(60, (2, 2),padding='same',activation='relu'))

# Adding fifth conv layer
model7.add(Conv2D(120, (2, 2), activation='relu'))

# Adding Maxpooling layer
model7.add(MaxPooling2D(pool_size=(3, 3),padding='same'))

# Adding Dropout
model7.add(Dropout(0.25))

# Adding sixth conv layer
model7.add(Conv2D(120, (2, 2),padding='same',activation='relu'))

# Adding seventh conv layer
model7.add(Conv2D(240, (2, 2), activation='relu'))

# Adding Maxpooling layer
model7.add(MaxPooling2D(pool_size=(2, 2), strides=(1,1)))

# Adding Dropout
model7.add(Dropout(0.25))

# Adding flatten layer
model7.add(Flatten())

# Adding first hidden layer
model7.add(Dense(256, activation='relu',kernel_initializer=he_normal(se
ed=None)))

# Adding Batch Normalization
model7.add(BatchNormalization())
```

```
# Adding Dropout
model7.add(Dropout(0.5))

# Adding second hidden layer
model7.add(Dense(128, activation='relu',kernel_initializer=he_normal(se
ed=None)))

# Adding Dropout
model7.add(Dropout(0.25))

# Adding output layer
model7.add(Dense(num_classes, activation='softmax'))

# Printing model Summary
print(model7.summary())

# Compiling the model
model7.compile(optimizer='adam', loss='categorical_crossentropy', metri
cs=['accuracy'])

# Fitting the data to the model
history = model7.fit(x_train, y_train,batch_size=batch_size,epochs=epoc
hs,verbose=1,validation_data=(x_test, y_test))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 28, 28, 10)        50
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 27, 27, 20)        820
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 25, 25, 20)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 25, 25, 20)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 40)        3240
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 40)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 12, 12, 60)        9660
```

```
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 11, 11, 120)       28920
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 120)         0
_________________________________________________________________
dropout_9 (Dropout)          (None, 4, 4, 120)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 4, 4, 120)         57720
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 3, 3, 240)         115440
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 2, 2, 240)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 2, 2, 240)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 960)               0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               246016
_________________________________________________________________
batch_normalization_2 (Batch (None, 256)               1024
_________________________________________________________________
dropout_11 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_12 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 497,076
Trainable params: 496,564
Non-trainable params: 512
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
5200 - acc: 0.8264 - val_loss: 0.0519 - val_acc: 0.9848
Epoch 2/12
```

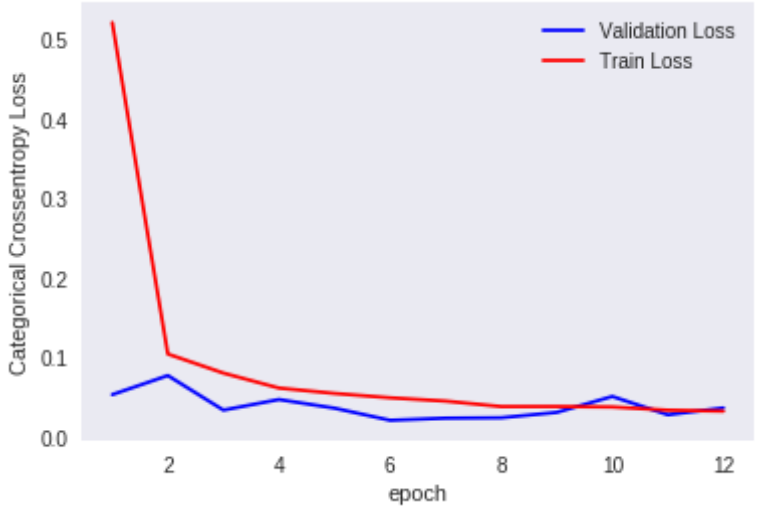
```
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
1031 - acc: 0.9689 - val_loss: 0.0760 - val_acc: 0.9770
Epoch 3/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.
0791 - acc: 0.9770 - val_loss: 0.0325 - val_acc: 0.9903
Epoch 4/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0602 - acc: 0.9823 - val_loss: 0.0458 - val_acc: 0.9862
Epoch 5/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0535 - acc: 0.9848 - val_loss: 0.0349 - val_acc: 0.9896
Epoch 6/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0480 - acc: 0.9860 - val_loss: 0.0196 - val_acc: 0.9935
Epoch 7/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0438 - acc: 0.9871 - val_loss: 0.0223 - val_acc: 0.9932
Epoch 8/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0369 - acc: 0.9892 - val_loss: 0.0228 - val_acc: 0.9931
Epoch 9/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.
0370 - acc: 0.9895 - val_loss: 0.0296 - val_acc: 0.9917
Epoch 10/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0364 - acc: 0.9895 - val_loss: 0.0496 - val_acc: 0.9854
Epoch 11/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0324 - acc: 0.9903 - val_loss: 0.0266 - val_acc: 0.9923
Epoch 12/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.
0312 - acc: 0.9908 - val_loss: 0.0352 - val_acc: 0.9909
```

In [9]:

```
# Evaluating the model
score = model7.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')

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,epochs+1))

# Validation loss
vy = history.history['val_loss']
# Training loss
ty = history.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Test score: 0.03518788337475271
Test accuracy: 0.9909
```



In [11]:
```
import pandas as pd
models = pd.DataFrame({'Model': ['CNN 3-conv layer with kernal size (3,3)', 'CNN 5-conv layer with kernal size(5,5)', "CNN 7-conv layer with kernal size(2,2)"], 'Test score' : [0.019,0.021,0.035],'Accuracy': [0.9943,0.9948,0.9909]}, columns = ["Model","Test score","Accuracy"])
models
```

Out[11]:

	Model	Test score	Accuracy
0	CNN 3-conv layer with kernal size (3,3)	0.019	0.9943
1	CNN 5-conv layer with kernal size(5,5)	0.021	0.9948
2	CNN 7-conv layer with kernal size(2,2)	0.035	0.9909

Conclusion

1.Load mnist dataset.

2.Split the dataset into train and test.

3.Normalize the data.

4.convert class vectors to binary class matrices.

5.Implement Softmax classifier with 3 , 5 and 7 CONV layers of of kernel size 3 , 4 and 2 with hidden layers, batchnormalizer and different dropoutrates .

6.Ploting Categorical Crossentropy Loss VS No.of Epochs plot .