```
# 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.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.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

```
%matplotlib inline
import matplotlib.pyplot as plt

def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Test Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

MODEL1 : Three Convolutional Network with Adadelta as optimizer

1.1 with kernal_size = (3,3), Dropout = 0.5 and MaxPooling(pool_size =(2,2))

```
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(Conv2D(96, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(75, 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 [==============================] - 186s 3ms/step - loss: 0.2920 - acc: 0.9100 - val_loss: 0.0505 - val_acc: 0.9849
Epoch 2/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0919 - acc: 0.9726 - val_loss: 0.0427 - val_acc: 0.9868
Epoch 3/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0659 - acc: 0.9804 - val_loss: 0.0275 - val_acc: 0.9907
Epoch 4/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0556 - acc: 0.9842 - val_loss: 0.0280 - val_acc: 0.9901
Epoch 5/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0448 - acc: 0.9873 - val_loss: 0.0250 - val_acc: 0.9922
Epoch 6/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0395 - acc: 0.9890 - val_loss: 0.0267 - val_acc: 0.9923
Epoch 7/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0354 - acc: 0.9898 - val_loss: 0.0267 - val_acc: 0.9913
Epoch 8/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0317 - acc: 0.9906 - val_loss: 0.0263 - val_acc: 0.9922
Epoch 9/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0295 - acc: 0.9911 - val_loss: 0.0215 - val_acc: 0.9935
Epoch 10/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0267 - acc: 0.9923 - val_loss: 0.0170 - val_acc: 0.9951
Epoch 11/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0245 - acc: 0.9927 - val_loss: 0.0193 - val_acc: 0.9939
Epoch 12/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0216 - acc: 0.9932 - val_loss: 0.0242 - val_acc: 0.9937
Test loss: 0.024166463349014566
Test accuracy: 0.9937
```

```
%matplotlib inline
import matplotlib.pyplot as plt
```

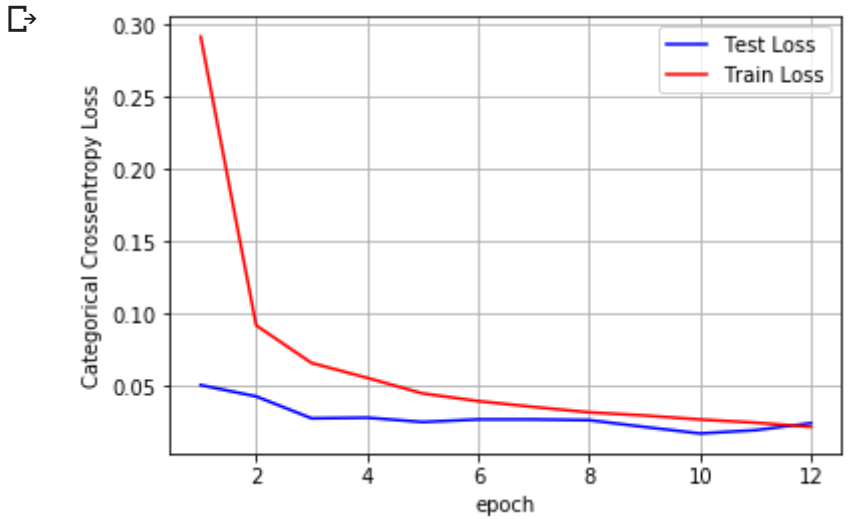
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy

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



1.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(44, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    model.add(Conv2D(66, (2, 2), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(BatchNormalization())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    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 [==============================] - 35s 578us/step - loss: 0.1689 - acc: 0.9509 - val_loss: 0.1136 - val_acc: 0.964
Epoch 2/12
60000/60000 [==============================] - 34s 563us/step - loss: 0.0549 - acc: 0.9830 - val_loss: 0.0489 - val_acc: 0.984
Epoch 3/12
60000/60000 [==============================] - 34s 563us/step - loss: 0.0372 - acc: 0.9887 - val_loss: 0.0573 - val_acc: 0.983
Epoch 4/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.0285 - acc: 0.9911 - val_loss: 0.0525 - val_acc: 0.984
Epoch 5/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.0219 - acc: 0.9932 - val_loss: 0.0971 - val_acc: 0.971
Epoch 6/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.0180 - acc: 0.9948 - val_loss: 0.0396 - val_acc: 0.988
Epoch 7/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.0135 - acc: 0.9962 - val_loss: 0.0425 - val_acc: 0.988
Epoch 8/12
60000/60000 [==============================] - 34s 567us/step - loss: 0.0112 - acc: 0.9968 - val_loss: 0.0532 - val_acc: 0.986
Epoch 9/12
60000/60000 [==============================] - 34s 563us/step - loss: 0.0094 - acc: 0.9974 - val_loss: 0.0415 - val_acc: 0.988
Epoch 10/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.0077 - acc: 0.9980 - val_loss: 0.0377 - val_acc: 0.989
Epoch 11/12
60000/60000 [==============================] - 34s 563us/step - loss: 0.0065 - acc: 0.9981 - val_loss: 0.0372 - val_acc: 0.989
Epoch 12/12
60000/60000 [==============================] - 34s 563us/step - loss: 0.0052 - acc: 0.9986 - val_loss: 0.0540 - val_acc: 0.985
Test loss: 0.054049926586002404
Test accuracy: 0.9858
```

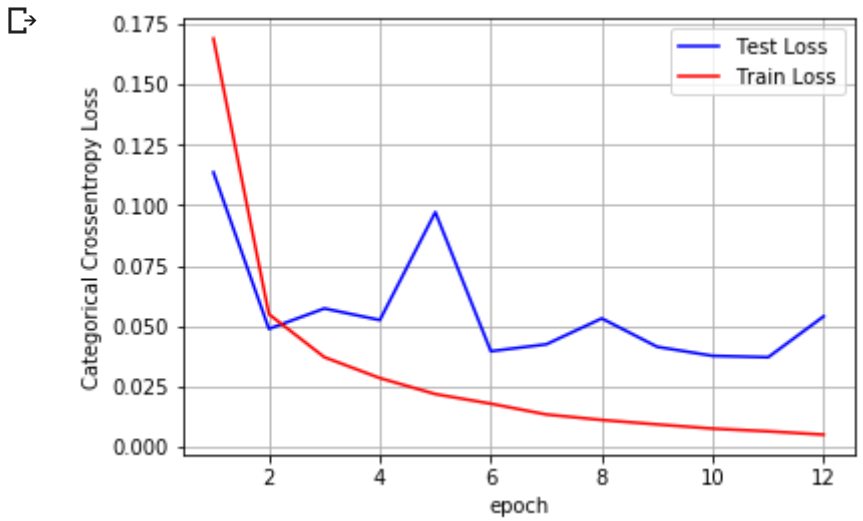
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



1.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(44, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(89, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(165, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(97, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))
```

```
    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])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholde

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_un

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_po

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholde

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is depreca

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is de

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<
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 [==============================] - 188s 3ms/step - loss: 0.7024 - acc: 0.7695 - val_loss: 4.6582 - val_acc: 0.3859
Epoch 2/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.3975 - acc: 0.8714 - val_loss: 4.8570 - val_acc: 0.3057
Epoch 3/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.2876 - acc: 0.9090 - val_loss: 0.3090 - val_acc: 0.9044
Epoch 4/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.2319 - acc: 0.9283 - val_loss: 1.1523 - val_acc: 0.7380
Epoch 5/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.2063 - acc: 0.9359 - val_loss: 0.9241 - val_acc: 0.7260
Epoch 6/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.2268 - acc: 0.9288 - val_loss: 0.8064 - val_acc: 0.7751
Epoch 7/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.1863 - acc: 0.9432 - val_loss: 0.5246 - val_acc: 0.8310
Epoch 8/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.1662 - acc: 0.9490 - val_loss: 0.3421 - val_acc: 0.9035
Epoch 9/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.1653 - acc: 0.9505 - val_loss: 0.2114 - val_acc: 0.9320
Epoch 10/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.1435 - acc: 0.9574 - val_loss: 0.3748 - val_acc: 0.8851
Epoch 11/12
```

```
Epoch 11/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.1591 - acc: 0.9512 - val_loss: 0.5295 - val_acc: 0.8370
Epoch 12/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.1404 - acc: 0.9579 - val_loss: 0.3768 - val_acc: 0.8728
Test loss: 0.376783286190033
Test accuracy: 0.8728
```

```
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))

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy

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

MODEL2 : Five Convolutional Network with Adadelta as optimizer

2.1 with kernal_size = (3,3), Dropout = 0.25 and MaxPooling(pool_size =(3,3))

```
model = Sequential()
model.add(Conv2D(25, kernel_size=(3, 3), activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(55, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(78, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(96, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(108, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
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 [==============================] - 50s 833us/step - loss: 12.4907 - acc: 0.1164 - val_loss: 14.5482 - val_acc: 0.0
Epoch 2/12
60000/60000 [==============================] - 49s 816us/step - loss: 14.5415 - acc: 0.0977 - val_loss: 14.5482 - val_acc: 0.0
Epoch 3/12
60000/60000 [==============================] - 49s 814us/step - loss: 14.5400 - acc: 0.0978 - val_loss: 14.5482 - val_acc: 0.0
Epoch 4/12
60000/60000 [==============================] - 49s 820us/step - loss: 14.5445 - acc: 0.0975 - val_loss: 14.5482 - val_acc: 0.0
Epoch 5/12
60000/60000 [==============================] - 49s 817us/step - loss: 14.5453 - acc: 0.0976 - val_loss: 14.5482 - val_acc: 0.0
Epoch 6/12
60000/60000 [==============================] - 49s 818us/step - loss: 14.5436 - acc: 0.0977 - val_loss: 14.5482 - val_acc: 0.0
Epoch 7/12
60000/60000 [==============================] - 49s 817us/step - loss: 14.5463 - acc: 0.0975 - val_loss: 14.5482 - val_acc: 0.0
Epoch 8/12
60000/60000 [==============================] - 49s 817us/step - loss: 14.5463 - acc: 0.0975 - val_loss: 14.5482 - val_acc: 0.0
Epoch 9/12
60000/60000 [==============================] - 49s 821us/step - loss: 14.5457 - acc: 0.0975 - val_loss: 14.5482 - val_acc: 0.0
Epoch 10/12
60000/60000 [==============================] - 49s 815us/step - loss: 14.5459 - acc: 0.0975 - val_loss: 14.5482 - val_acc: 0.0
Epoch 11/12
60000/60000 [==============================] - 49s 813us/step - loss: 14.5462 - acc: 0.0975 - val_loss: 14.5482 - val_acc: 0.0
Epoch 12/12
60000/60000 [==============================] - 49s 815us/step - loss: 14.5476 - acc: 0.0974 - val_loss: 14.5482 - val_acc: 0.0
Test loss: 14.548192700195312
Test accuracy: 0.0974
```

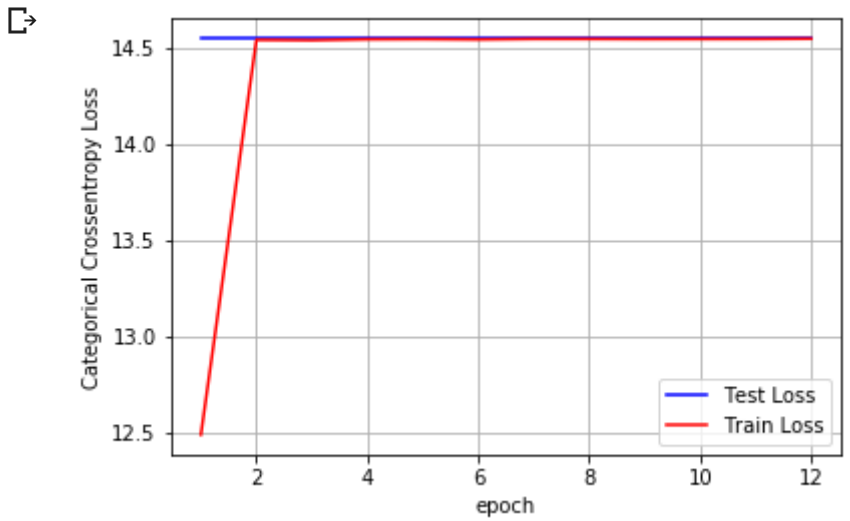
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



2.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(29, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(38, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(49, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(65, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(85, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(133, activation='relu'))
model.add(BatchNormalization())
```

```
model.add(Dense(95, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(76, activation='relu'))
model.add(BatchNormalization())

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 [==============================] - 47s 782us/step - loss: 0.1992 - acc: 0.9398 - val_loss: 0.3477 - val_acc: 0.886
Epoch 2/12
60000/60000 [==============================] - 45s 751us/step - loss: 0.0638 - acc: 0.9800 - val_loss: 1.9318 - val_acc: 0.508
Epoch 3/12
60000/60000 [==============================] - 46s 759us/step - loss: 0.0454 - acc: 0.9859 - val_loss: 0.2556 - val_acc: 0.916
Epoch 4/12
60000/60000 [==============================] - 46s 759us/step - loss: 0.0361 - acc: 0.9890 - val_loss: 0.1088 - val_acc: 0.967
Epoch 5/12
60000/60000 [==============================] - 45s 755us/step - loss: 0.0286 - acc: 0.9910 - val_loss: 0.3016 - val_acc: 0.919
Epoch 6/12
60000/60000 [==============================] - 45s 754us/step - loss: 0.0236 - acc: 0.9924 - val_loss: 0.0625 - val_acc: 0.981
Epoch 7/12
60000/60000 [==============================] - 45s 754us/step - loss: 0.0226 - acc: 0.9926 - val_loss: 0.0944 - val_acc: 0.972
Epoch 8/12
60000/60000 [==============================] - 45s 751us/step - loss: 0.0187 - acc: 0.9940 - val_loss: 0.1087 - val_acc: 0.966
Epoch 9/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0163 - acc: 0.9948 - val_loss: 0.0717 - val_acc: 0.979
Epoch 10/12
60000/60000 [==============================] - 46s 762us/step - loss: 0.0144 - acc: 0.9950 - val_loss: 0.1359 - val_acc: 0.958
Epoch 11/12
60000/60000 [==============================] - 46s 761us/step - loss: 0.0131 - acc: 0.9957 - val_loss: 0.1143 - val_acc: 0.964
Epoch 12/12
60000/60000 [==============================] - 46s 759us/step - loss: 0.0116 - acc: 0.9960 - val_loss: 0.0472 - val_acc: 0.987
Test loss: 0.04721413791068335
Test accuracy: 0.987
```

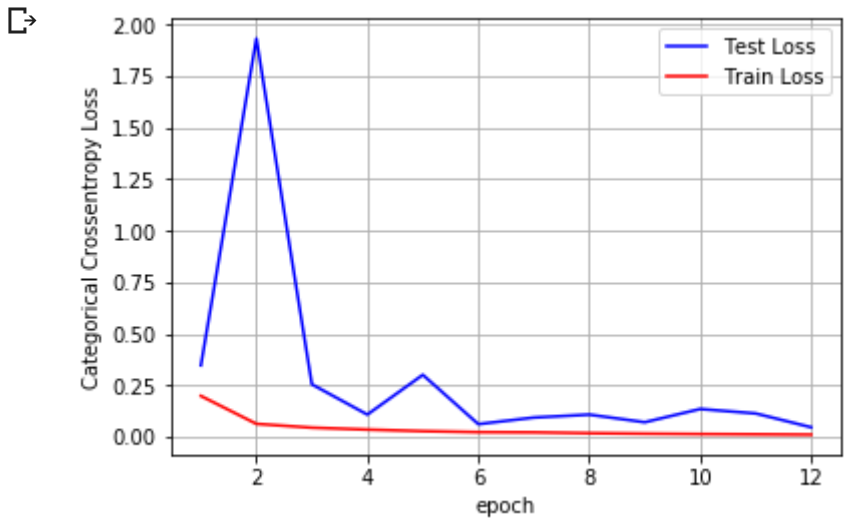
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



2.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(43, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(78, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(93, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(117, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(86, activation='relu'))
```

```
    model.add(Dropout(0.3))
    model.add(Dense(65, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(45, activation='relu'))
    model.add(Dropout(0.3))
    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 [==============================] - 178s 3ms/step - loss: 14.0103 - acc: 0.1022 - val_loss: 14.5353 - val_acc: 0.09
Epoch 2/12
60000/60000 [==============================] - 177s 3ms/step - loss: 14.5056 - acc: 0.1000 - val_loss: 14.5353 - val_acc: 0.09
Epoch 3/12
60000/60000 [==============================] - 177s 3ms/step - loss: 14.4976 - acc: 0.1005 - val_loss: 14.5353 - val_acc: 0.09
Epoch 4/12
60000/60000 [==============================] - 177s 3ms/step - loss: 14.5168 - acc: 0.0993 - val_loss: 14.5353 - val_acc: 0.09
Epoch 5/12
60000/60000 [==============================] - 176s 3ms/step - loss: 14.5251 - acc: 0.0988 - val_loss: 14.5353 - val_acc: 0.09
Epoch 6/12
60000/60000 [==============================] - 176s 3ms/step - loss: 14.5358 - acc: 0.0982 - val_loss: 14.5353 - val_acc: 0.09
Epoch 7/12
60000/60000 [==============================] - 177s 3ms/step - loss: 14.5408 - acc: 0.0979 - val_loss: 14.5353 - val_acc: 0.09
Epoch 8/12
60000/60000 [==============================] - 176s 3ms/step - loss: 14.5186 - acc: 0.0992 - val_loss: 14.5353 - val_acc: 0.09
Epoch 9/12
60000/60000 [==============================] - 176s 3ms/step - loss: 14.5149 - acc: 0.0995 - val_loss: 14.5353 - val_acc: 0.09
Epoch 10/12
60000/60000 [==============================] - 176s 3ms/step - loss: 14.4942 - acc: 0.1007 - val_loss: 14.5353 - val_acc: 0.09
Epoch 11/12
60000/60000 [==============================] - 175s 3ms/step - loss: 14.5326 - acc: 0.0984 - val_loss: 14.5353 - val_acc: 0.09
Epoch 12/12
60000/60000 [==============================] - 175s 3ms/step - loss: 14.5049 - acc: 0.1001 - val_loss: 14.5353 - val_acc: 0.09
Test loss: 14.535298265075683
Test accuracy: 0.0982
```

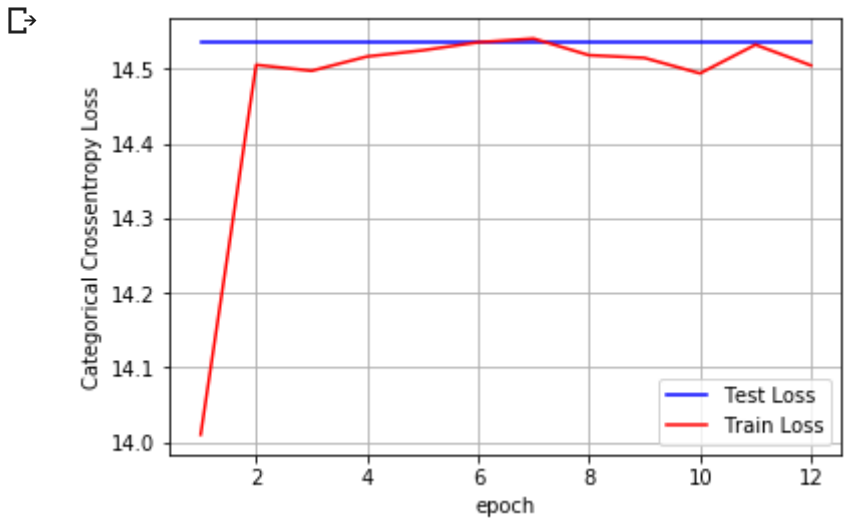
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



MODEL3 : Seven Convolutional Network with Adadelta as optimizer

3.1 with kernal_size = (3,3), Dropout = 0.25 and MaxPooling(pool_size =(3,3))

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

model.add(Conv2D(45, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(60, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(78, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(95, (3, 3), activation='relu', padding='same'))
```

```
    model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

    model.add(Conv2D(112, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

    model.add(Conv2D(145, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    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 [==============================] - 53s 882us/step - loss: 2.3015 - acc: 0.1119 - val_loss: 2.3008 - val_acc: 0.113
Epoch 2/12
60000/60000 [==============================] - 51s 851us/step - loss: 2.2902 - acc: 0.1231 - val_loss: 2.3012 - val_acc: 0.113
Epoch 3/12
60000/60000 [==============================] - 51s 848us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 4/12
60000/60000 [==============================] - 51s 847us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 5/12
60000/60000 [==============================] - 51s 847us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 6/12
60000/60000 [==============================] - 51s 850us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 7/12
60000/60000 [==============================] - 51s 846us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 8/12
60000/60000 [==============================] - 51s 847us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 9/12
60000/60000 [==============================] - 51s 849us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 10/12
60000/60000 [==============================] - 51s 850us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 11/12
60000/60000 [==============================] - 51s 848us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 12/12
60000/60000 [==============================] - 51s 853us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Test loss: 2.3010340282440187
Test accuracy: 0.1135
```

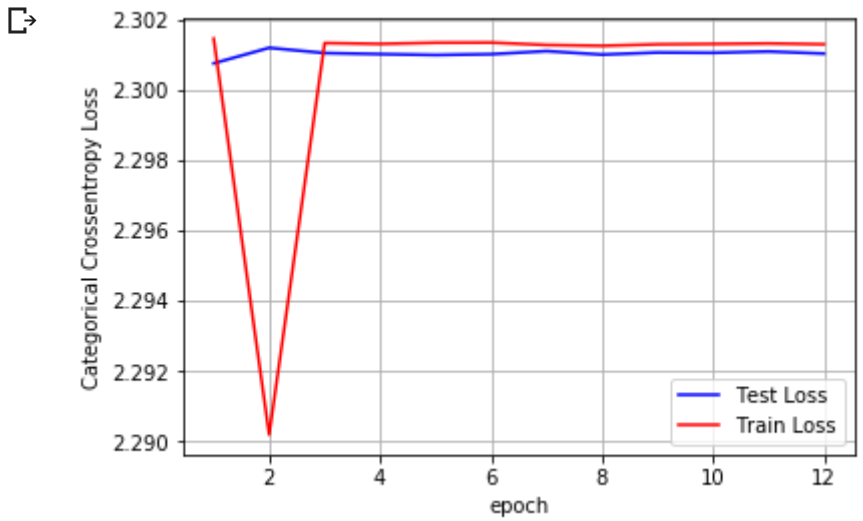
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



3.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(45, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(65, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(97, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(114, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(126, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
    model.add(Conv2D(143, (2, 2), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

    model.add(Flatten())
    model.add(Dense(97, activation='relu'))
    model.add(BatchNormalization())

    model.add(Dense(65, activation='relu'))
    model.add(BatchNormalization())

    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 [==============================] - 47s 777us/step - loss: 0.1836 - acc: 0.9442 - val_loss: 9.1206 - val_acc: 0.209
Epoch 2/12
60000/60000 [==============================] - 45s 743us/step - loss: 0.0667 - acc: 0.9796 - val_loss: 2.3494 - val_acc: 0.517
Epoch 3/12
60000/60000 [==============================] - 45s 746us/step - loss: 0.0519 - acc: 0.9842 - val_loss: 8.5230 - val_acc: 0.370
Epoch 4/12
60000/60000 [==============================] - 45s 758us/step - loss: 0.0492 - acc: 0.9849 - val_loss: 6.4195 - val_acc: 0.357
Epoch 5/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0375 - acc: 0.9884 - val_loss: 12.8700 - val_acc: 0.10
Epoch 6/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0318 - acc: 0.9898 - val_loss: 2.4923 - val_acc: 0.485
Epoch 7/12
60000/60000 [==============================] - 46s 759us/step - loss: 0.0285 - acc: 0.9911 - val_loss: 4.3455 - val_acc: 0.409
Epoch 8/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0260 - acc: 0.9916 - val_loss: 8.7995 - val_acc: 0.232
Epoch 9/12
60000/60000 [==============================] - 45s 758us/step - loss: 0.0222 - acc: 0.9929 - val_loss: 8.4333 - val_acc: 0.247
Epoch 10/12
60000/60000 [==============================] - 45s 745us/step - loss: 0.0219 - acc: 0.9927 - val_loss: 8.0134 - val_acc: 0.203
Epoch 11/12
60000/60000 [==============================] - 45s 750us/step - loss: 0.0179 - acc: 0.9942 - val_loss: 8.0351 - val_acc: 0.440
Epoch 12/12
60000/60000 [==============================] - 45s 752us/step - loss: 0.0173 - acc: 0.9944 - val_loss: 4.9741 - val_acc: 0.506
Test loss: 4.974070930480957
Test accuracy: 0.5066
```

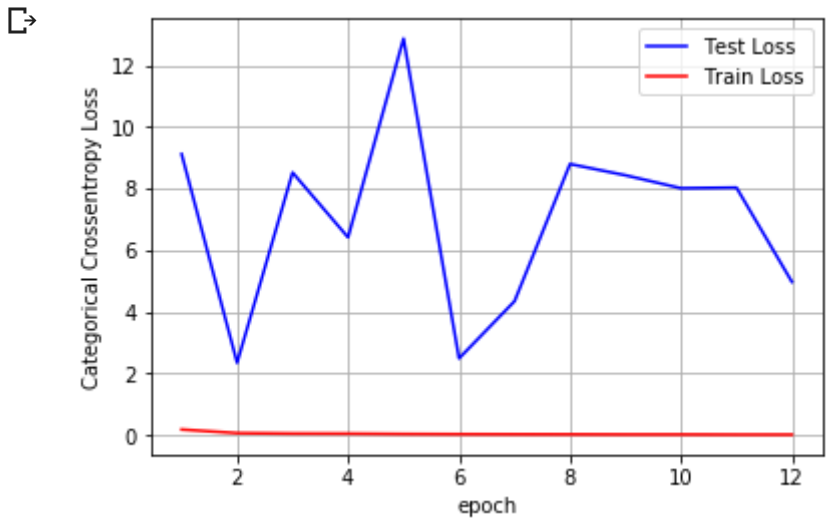
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



3.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(12, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(43, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(60, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(85, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(112, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(143, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
model.add(Conv2D(157, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(98, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())

model.add(Dense(56, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))


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 [==============================] - 214s 4ms/step - loss: 2.2220 - acc: 0.2064 - val_loss: 6.6590 - val_acc: 0.1135
Epoch 2/12
60000/60000 [==============================] - 212s 4ms/step - loss: 1.3812 - acc: 0.4362 - val_loss: 1.4635 - val_acc: 0.3390
Epoch 3/12
60000/60000 [==============================] - 211s 4ms/step - loss: 1.1214 - acc: 0.5511 - val_loss: 10.6865 - val_acc: 0.101
Epoch 4/12
60000/60000 [==============================] - 211s 4ms/step - loss: 1.1125 - acc: 0.5517 - val_loss: 0.8543 - val_acc: 0.7123
Epoch 5/12
60000/60000 [==============================] - 211s 4ms/step - loss: 1.2372 - acc: 0.5118 - val_loss: 5.7753 - val_acc: 0.1164
Epoch 6/12
60000/60000 [==============================] - 212s 4ms/step - loss: 1.1486 - acc: 0.5430 - val_loss: 3.4401 - val_acc: 0.1302
Epoch 7/12
60000/60000 [==============================] - 212s 4ms/step - loss: 1.0421 - acc: 0.5859 - val_loss: 2.2447 - val_acc: 0.2163
Epoch 8/12
60000/60000 [==============================] - 212s 4ms/step - loss: 0.9904 - acc: 0.6094 - val_loss: 0.9847 - val_acc: 0.5665
Epoch 9/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.9875 - acc: 0.6078 - val_loss: 3.0872 - val_acc: 0.3077
Epoch 10/12
60000/60000 [==============================] - 212s 4ms/step - loss: 0.9539 - acc: 0.6268 - val_loss: 1.1809 - val_acc: 0.3865
Epoch 11/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.9457 - acc: 0.6278 - val_loss: 0.8144 - val_acc: 0.7113
Epoch 12/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.9212 - acc: 0.6410 - val_loss: 0.7736 - val_acc: 0.7404
Test loss: 0.7736397917747497
Test accuracy: 0.7404
```

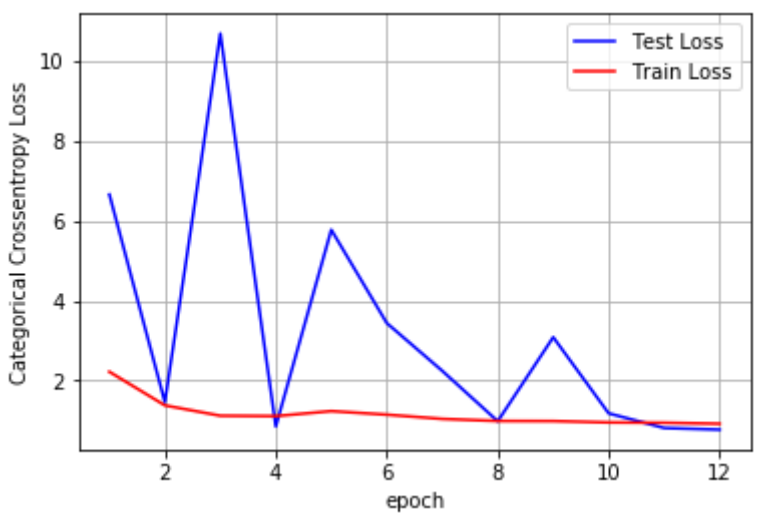
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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




```
#Activation function is relu in each layers
#Adadelta optimizers is used in each case
#MaxPooling is used in each case of cnn_layers of pool_size=(2, 2)

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["No. of CNN_Layers","kernal_size","Batch Normalisations","Dropout","No. of hidden layers", "Test loss", "Test Accuracy"]

x.add_row([3, (3,3), "No", 0.5, 1, 0.024, 0.99])
x.add_row([3, (2,2), "Yes", "No dropout", 1, 0.054, 0.98])
x.add_row([3, (5,5), "Yes", 0.3, 2, 0.376, 0.87])
x.add_row([5, (3,3), "No", 0.25, 1, 14.548, 0.09])
x.add_row([5, (2,2), "Yes", "No droput", 3, 0.047, 0.98])
x.add_row([5, (5,5), "Yes", 0.3, 3, 14.53, 0.09])
x.add_row([7, (3,3), "No", 0.25, 1, 2.30, 0.11])
x.add_row([7, (2,2), "Yes", "No dropout", 2, 4.974, 0.50])
x.add_row([7, (5,5), "Yes", 0.3, 2, 0.773, 0.74])
print(x)
```

No. of CNN_Layers	kernal_size	Batch Normalisations	Dropout	No. of hidden layers	Test loss	Test Accuracy
3	(3, 3)	No	0.5	1	0.024	0.99
3	(2, 2)	Yes	No dropout	1	0.054	0.98
3	(5, 5)	Yes	0.3	2	0.376	0.87
5	(3, 3)	No	0.25	1	14.548	0.09
5	(2, 2)	Yes	No droput	3	0.047	0.98
5	(5, 5)	Yes	0.3	3	14.53	0.09
7	(3, 3)	No	0.25	1	2.3	0.11
7	(2, 2)	Yes	No dropout	2	4.974	0.5
7	(5, 5)	Yes	0.3	2	0.773	0.74

step by step procedure for above experiment

1.Three different architecture of convolution layer has been implemented.There are three , five and seven convolution layers with "**Adadelta optimizers**" in each case

2.**First architecture i.e. three convolution layer** has been further experimented in three more categories.

3.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

4.For the first categories kernal size is (3,3), dropout=0.5 and maxpoolsize=(3,3).

5.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

6.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

7.for each categories test score and test accuracy has been evaluated with different parameter.

8.**Second architecture (i.e. five convolution layer)** has been further experimented in three more categories in the same way as first architecture.

9.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

10.For the first categories kernal size is (3,3), dropout=0.25 and maxpoolsize=(3,3).

11.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

12.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

13.for each categories test score and test accuracy has been evaluated with different parameter.

14.**Third architecture (i.e. Seven convolution layer)** has been further experimented in three more categories in the same way as first and second architecture.

15.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

16.For the first categories kernal size is (3,3), dropout=0.25 and maxpoolsize=(3,3).

17.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

18.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

19.for each categories test score and test accuracy has been evaluated with different parameter.

20.see the **preetytable** to observe the different result in different case

MODEL1 : Three Convolutional Network with Adam as optimizer

1.1 with kernal_size = (3,3), Dropout = 0.5 and MaxPooling(pool_size =(2,2))

```
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(Conv2D(96, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(75, 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))
```

```
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 [==============================] - 179s 3ms/step - loss: 0.3042 - acc: 0.9072 - val_loss: 0.0479 - val_acc: 0.9850
Epoch 2/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0999 - acc: 0.9708 - val_loss: 0.0312 - val_acc: 0.9885
Epoch 3/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0728 - acc: 0.9787 - val_loss: 0.0310 - val_acc: 0.9893
Epoch 4/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0574 - acc: 0.9839 - val_loss: 0.0229 - val_acc: 0.9916
Epoch 5/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0463 - acc: 0.9864 - val_loss: 0.0223 - val_acc: 0.9924
Epoch 6/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0416 - acc: 0.9875 - val_loss: 0.0229 - val_acc: 0.9924
Epoch 7/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0369 - acc: 0.9887 - val_loss: 0.0196 - val_acc: 0.9929
Epoch 8/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0319 - acc: 0.9904 - val_loss: 0.0195 - val_acc: 0.9941
Epoch 9/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0298 - acc: 0.9913 - val_loss: 0.0197 - val_acc: 0.9931
Epoch 10/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0248 - acc: 0.9922 - val_loss: 0.0200 - val_acc: 0.9941
Epoch 11/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0257 - acc: 0.9920 - val_loss: 0.0221 - val_acc: 0.9939
Epoch 12/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0221 - acc: 0.9931 - val_loss: 0.0197 - val_acc: 0.9946
Test loss: 0.019714530481706516
Test accuracy: 0.9946
```

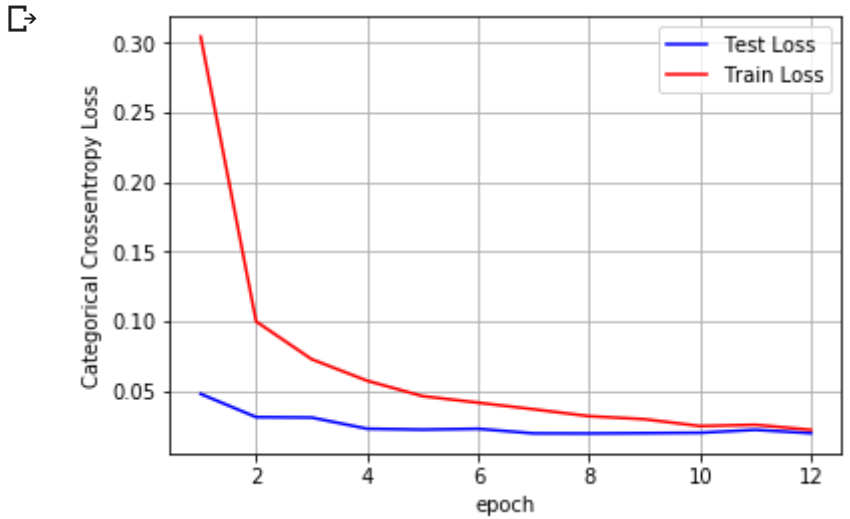
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy
```

```
# loss : training loss
# acc : train accuracy

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



1.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(44, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(66, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
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))
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 [==============================] - 36s 598us/step - loss: 0.1735 - acc: 0.9476 - val_loss: 0.0835 - val_acc: 0.975
Epoch 2/12
60000/60000 [==============================] - 33s 557us/step - loss: 0.0520 - acc: 0.9848 - val_loss: 0.0563 - val_acc: 0.984
Epoch 3/12
60000/60000 [==============================] - 34s 558us/step - loss: 0.0367 - acc: 0.9889 - val_loss: 0.0633 - val_acc: 0.980
Epoch 4/12
60000/60000 [==============================] - 34s 559us/step - loss: 0.0271 - acc: 0.9919 - val_loss: 0.0431 - val_acc: 0.987
Epoch 5/12
60000/60000 [==============================] - 34s 560us/step - loss: 0.0227 - acc: 0.9929 - val_loss: 0.0541 - val_acc: 0.984
Epoch 6/12
60000/60000 [==============================] - 33s 554us/step - loss: 0.0195 - acc: 0.9937 - val_loss: 0.0389 - val_acc: 0.987
Epoch 7/12
60000/60000 [==============================] - 33s 552us/step - loss: 0.0138 - acc: 0.9960 - val_loss: 0.0715 - val_acc: 0.978
Epoch 8/12
60000/60000 [==============================] - 35s 576us/step - loss: 0.0126 - acc: 0.9962 - val_loss: 0.0516 - val_acc: 0.984
Epoch 9/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0114 - acc: 0.9963 - val_loss: 0.0445 - val_acc: 0.987
Epoch 10/12
60000/60000 [==============================] - 34s 564us/step - loss: 0.0129 - acc: 0.9957 - val_loss: 0.0513 - val_acc: 0.987
Epoch 11/12
60000/60000 [==============================] - 35s 581us/step - loss: 0.0089 - acc: 0.9973 - val_loss: 0.1000 - val_acc: 0.971
Epoch 12/12
60000/60000 [==============================] - 34s 574us/step - loss: 0.0067 - acc: 0.9981 - val_loss: 0.0459 - val_acc: 0.987
Test loss: 0.04589277531141852
Test accuracy: 0.9875
```

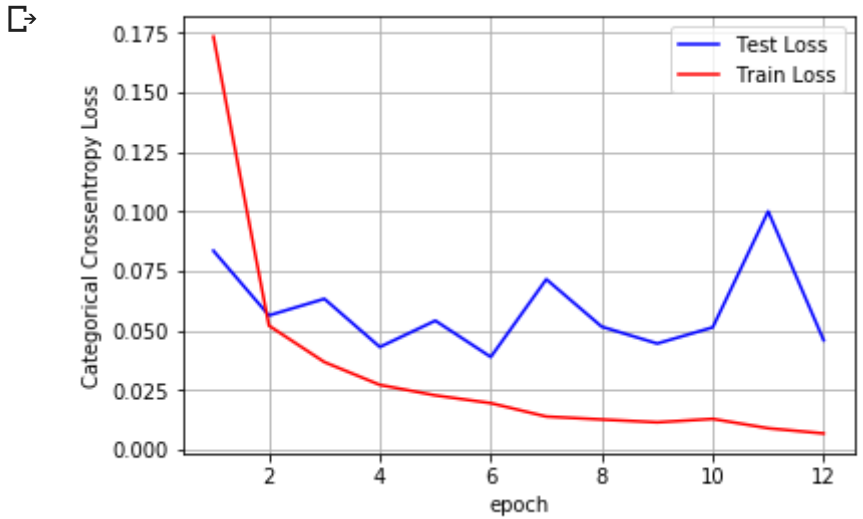
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy

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



1.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(44, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(89, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    model.add(Conv2D(165, (5, 5), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))

    model.add(Dense(97, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))

    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))
    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 [==============================] - 190s 3ms/step - loss: 0.1844 - acc: 0.9458 - val_loss: 0.0639 - val_acc: 0.9786
Epoch 2/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.0554 - acc: 0.9842 - val_loss: 0.0661 - val_acc: 0.9794
Epoch 3/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.0388 - acc: 0.9886 - val_loss: 0.0875 - val_acc: 0.9726
Epoch 4/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0295 - acc: 0.9910 - val_loss: 0.0565 - val_acc: 0.9839
Epoch 5/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0259 - acc: 0.9924 - val_loss: 0.0264 - val_acc: 0.9923
Epoch 6/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.0221 - acc: 0.9935 - val_loss: 0.0581 - val_acc: 0.9826
Epoch 7/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.0203 - acc: 0.9936 - val_loss: 0.0231 - val_acc: 0.9934
Epoch 8/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0159 - acc: 0.9950 - val_loss: 0.0365 - val_acc: 0.9899
Epoch 9/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0138 - acc: 0.9961 - val_loss: 0.0413 - val_acc: 0.9890
Epoch 10/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0122 - acc: 0.9963 - val_loss: 0.0452 - val_acc: 0.9882
Epoch 11/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.0125 - acc: 0.9962 - val_loss: 0.0305 - val_acc: 0.9911
Epoch 12/12
60000/60000 [==============================] - 187s 3ms/step - loss: 0.0104 - acc: 0.9966 - val_loss: 0.0258 - val_acc: 0.9927
Test loss: 0.0258149948295174
Test accuracy: 0.9927
```

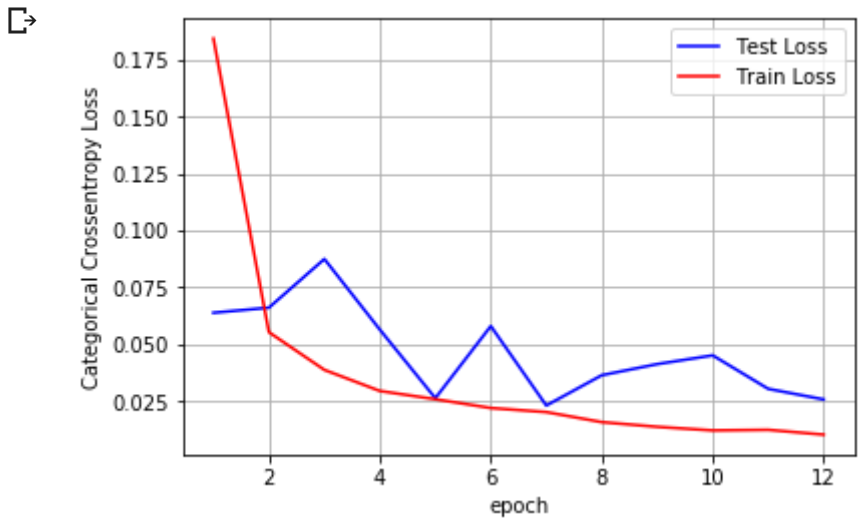
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



MODEL2 : Five Convolutional Network with Adam as optimizer

2.1 with kernal_size = (3,3), Dropout = 0.25 and MaxPooling(pool_size =(3,3))

```
model = Sequential()
model.add(Conv2D(25, kernel_size=(3, 3), activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(55, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(78, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(96, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(108, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
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))
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 [==============================] - 52s 866us/step - loss: 0.5362 - acc: 0.8196 - val_loss: 0.2049 - val_acc: 0.934
Epoch 2/12
60000/60000 [==============================] - 50s 828us/step - loss: 0.1620 - acc: 0.9507 - val_loss: 0.1245 - val_acc: 0.960
Epoch 3/12
60000/60000 [==============================] - 50s 826us/step - loss: 0.1150 - acc: 0.9657 - val_loss: 0.1350 - val_acc: 0.957
Epoch 4/12
60000/60000 [==============================] - 50s 828us/step - loss: 0.0891 - acc: 0.9735 - val_loss: 0.0887 - val_acc: 0.972
Epoch 5/12
60000/60000 [==============================] - 49s 822us/step - loss: 0.0737 - acc: 0.9780 - val_loss: 0.0671 - val_acc: 0.978
Epoch 6/12
60000/60000 [==============================] - 50s 828us/step - loss: 0.0608 - acc: 0.9821 - val_loss: 0.0594 - val_acc: 0.983
Epoch 7/12
60000/60000 [==============================] - 50s 830us/step - loss: 0.0532 - acc: 0.9845 - val_loss: 0.0580 - val_acc: 0.983
Epoch 8/12
60000/60000 [==============================] - 49s 819us/step - loss: 0.0466 - acc: 0.9861 - val_loss: 0.0575 - val_acc: 0.983
Epoch 9/12
60000/60000 [==============================] - 49s 824us/step - loss: 0.0423 - acc: 0.9870 - val_loss: 0.0497 - val_acc: 0.985
Epoch 10/12
60000/60000 [==============================] - 50s 827us/step - loss: 0.0369 - acc: 0.9886 - val_loss: 0.0575 - val_acc: 0.984
Epoch 11/12
60000/60000 [==============================] - 49s 823us/step - loss: 0.0319 - acc: 0.9905 - val_loss: 0.0520 - val_acc: 0.984
Epoch 12/12
60000/60000 [==============================] - 49s 822us/step - loss: 0.0291 - acc: 0.9912 - val_loss: 0.0474 - val_acc: 0.986
Test loss: 0.04744445713285823
Test accuracy: 0.9862
```

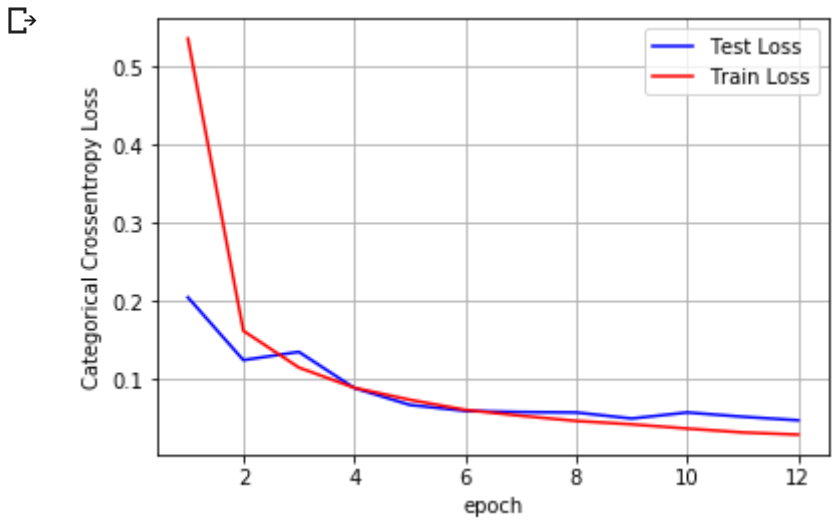
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



2.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(29, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(38, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(49, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(65, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(85, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(133, activation='relu'))
model.add(BatchNormalization())
```

```
model.add(Dense(95, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(76, activation='relu'))
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))
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 806us/step - loss: 0.2226 - acc: 0.9305 - val_loss: 0.5700 - val_acc: 0.830
Epoch 2/12
60000/60000 [==============================] - 45s 755us/step - loss: 0.0601 - acc: 0.9812 - val_loss: 0.2280 - val_acc: 0.918
Epoch 3/12
60000/60000 [==============================] - 45s 754us/step - loss: 0.0442 - acc: 0.9866 - val_loss: 0.3382 - val_acc: 0.899
Epoch 4/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0373 - acc: 0.9880 - val_loss: 0.3386 - val_acc: 0.893
Epoch 5/12
60000/60000 [==============================] - 45s 752us/step - loss: 0.0314 - acc: 0.9896 - val_loss: 0.1397 - val_acc: 0.956
Epoch 6/12
60000/60000 [==============================] - 45s 756us/step - loss: 0.0280 - acc: 0.9911 - val_loss: 0.0629 - val_acc: 0.979
Epoch 7/12
60000/60000 [==============================] - 45s 754us/step - loss: 0.0259 - acc: 0.9916 - val_loss: 0.3566 - val_acc: 0.893
Epoch 8/12
60000/60000 [==============================] - 45s 754us/step - loss: 0.0226 - acc: 0.9927 - val_loss: 0.0680 - val_acc: 0.977
Epoch 9/12
60000/60000 [==============================] - 45s 755us/step - loss: 0.0226 - acc: 0.9926 - val_loss: 0.0622 - val_acc: 0.981
Epoch 10/12
60000/60000 [==============================] - 45s 753us/step - loss: 0.0181 - acc: 0.9941 - val_loss: 0.2622 - val_acc: 0.931
Epoch 11/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0196 - acc: 0.9931 - val_loss: 0.1689 - val_acc: 0.950
Epoch 12/12
60000/60000 [==============================] - 45s 757us/step - loss: 0.0155 - acc: 0.9947 - val_loss: 0.0786 - val_acc: 0.976
Test loss: 0.07857787121404544
Test accuracy: 0.9766
```

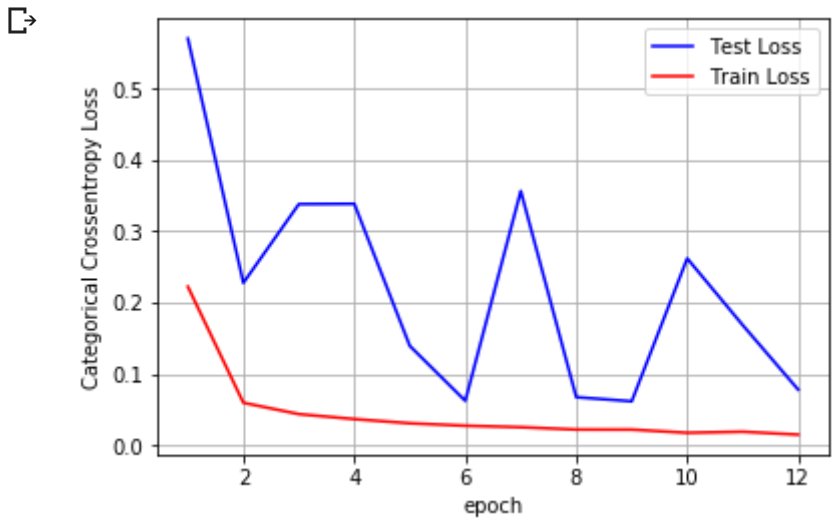
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



2.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(43, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(78, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(93, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(117, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(86, activation='relu'))
```

```
    model.add(Dropout(0.3))
    model.add(Dense(65, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(45, activation='relu'))
    model.add(Dropout(0.3))
    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))
    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 [==============================] - 187s 3ms/step - loss: 0.8045 - acc: 0.7231 - val_loss: 0.1296 - val_acc: 0.9698
Epoch 2/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.1377 - acc: 0.9703 - val_loss: 0.0774 - val_acc: 0.9839
Epoch 3/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.0970 - acc: 0.9804 - val_loss: 0.0457 - val_acc: 0.9895
Epoch 4/12
60000/60000 [==============================] - 182s 3ms/step - loss: 0.0731 - acc: 0.9852 - val_loss: 0.0587 - val_acc: 0.9878
Epoch 5/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.0611 - acc: 0.9874 - val_loss: 0.0421 - val_acc: 0.9890
Epoch 6/12
60000/60000 [==============================] - 199s 3ms/step - loss: 0.0543 - acc: 0.9890 - val_loss: 0.0482 - val_acc: 0.9907
Epoch 7/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.0481 - acc: 0.9902 - val_loss: 0.0443 - val_acc: 0.9916
Epoch 8/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0470 - acc: 0.9900 - val_loss: 0.0403 - val_acc: 0.9917
Epoch 9/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0370 - acc: 0.9926 - val_loss: 0.0551 - val_acc: 0.9888
Epoch 10/12
60000/60000 [==============================] - 179s 3ms/step - loss: 0.0395 - acc: 0.9917 - val_loss: 0.0416 - val_acc: 0.9912
Epoch 11/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0319 - acc: 0.9935 - val_loss: 0.0452 - val_acc: 0.9913
Epoch 12/12
60000/60000 [==============================] - 177s 3ms/step - loss: 0.0315 - acc: 0.9937 - val_loss: 0.0441 - val_acc: 0.9925
Test loss: 0.04412776248144378
Test accuracy: 0.9925
```

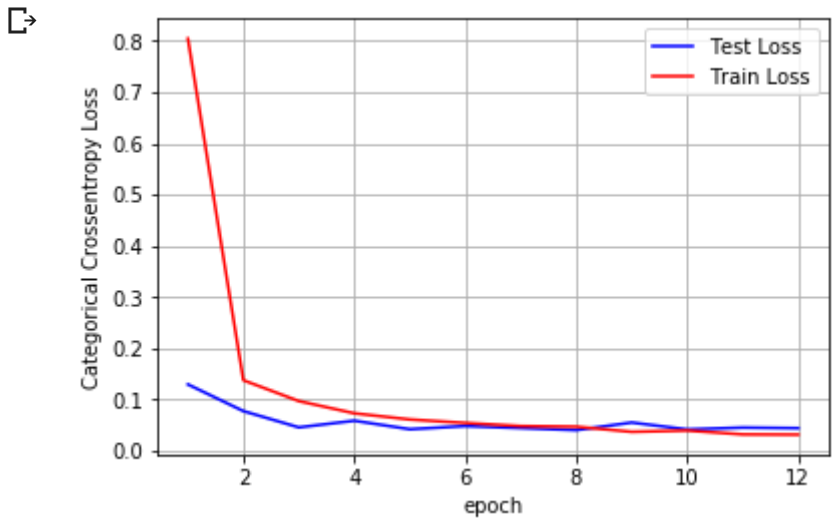
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



MODEL3 : Seven Convolutional Network with Adam as optimizer

3.1 with kernal_size = (3,3), Dropout = 0.25 and MaxPooling(pool_size =(3,3))

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

model.add(Conv2D(45, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(60, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(78, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(95, (3, 3), activation='relu', padding='same'))
```

```
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(112, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(145, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
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))
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 [==============================] - 54s 906us/step - loss: 1.0199 - acc: 0.6124 - val_loss: 0.4529 - val_acc: 0.850
Epoch 2/12
60000/60000 [==============================] - 51s 856us/step - loss: 0.2891 - acc: 0.9129 - val_loss: 0.1822 - val_acc: 0.942
Epoch 3/12
60000/60000 [==============================] - 51s 856us/step - loss: 0.1613 - acc: 0.9527 - val_loss: 0.1207 - val_acc: 0.963
Epoch 4/12
60000/60000 [==============================] - 52s 859us/step - loss: 0.1138 - acc: 0.9666 - val_loss: 0.0989 - val_acc: 0.972
Epoch 5/12
60000/60000 [==============================] - 52s 862us/step - loss: 0.0867 - acc: 0.9756 - val_loss: 0.0634 - val_acc: 0.981
Epoch 6/12
60000/60000 [==============================] - 52s 859us/step - loss: 0.0728 - acc: 0.9794 - val_loss: 0.0558 - val_acc: 0.983
Epoch 7/12
60000/60000 [==============================] - 51s 854us/step - loss: 0.0595 - acc: 0.9834 - val_loss: 0.0591 - val_acc: 0.984
Epoch 8/12
60000/60000 [==============================] - 51s 856us/step - loss: 0.0494 - acc: 0.9862 - val_loss: 0.0631 - val_acc: 0.980
Epoch 9/12
60000/60000 [==============================] - 51s 854us/step - loss: 0.0453 - acc: 0.9872 - val_loss: 0.0457 - val_acc: 0.987
Epoch 10/12
60000/60000 [==============================] - 51s 851us/step - loss: 0.0406 - acc: 0.9888 - val_loss: 0.0453 - val_acc: 0.986
Epoch 11/12
60000/60000 [==============================] - 51s 857us/step - loss: 0.0385 - acc: 0.9887 - val_loss: 0.0584 - val_acc: 0.982
Epoch 12/12
60000/60000 [==============================] - 51s 853us/step - loss: 0.0353 - acc: 0.9901 - val_loss: 0.0439 - val_acc: 0.986
Test loss: 0.04392361495406367
Test accuracy: 0.9868
```

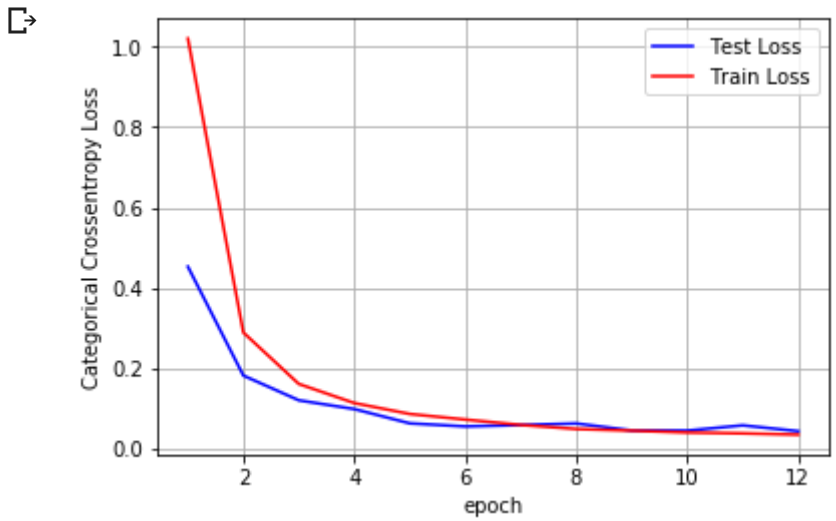
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



3.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(45, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(65, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(97, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(114, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(126, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
model.add(Conv2D(143, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(97, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(65, activation='relu'))
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))
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 [==============================] - 49s 820us/step - loss: 0.1746 - acc: 0.9474 - val_loss: 0.6559 - val_acc: 0.807
Epoch 2/12
60000/60000 [==============================] - 45s 756us/step - loss: 0.0571 - acc: 0.9826 - val_loss: 0.9573 - val_acc: 0.719
Epoch 3/12
60000/60000 [==============================] - 45s 753us/step - loss: 0.0413 - acc: 0.9870 - val_loss: 0.6679 - val_acc: 0.802
Epoch 4/12
60000/60000 [==============================] - 45s 747us/step - loss: 0.0337 - acc: 0.9895 - val_loss: 1.2193 - val_acc: 0.697
Epoch 5/12
60000/60000 [==============================] - 45s 751us/step - loss: 0.0307 - acc: 0.9903 - val_loss: 0.2132 - val_acc: 0.940
Epoch 6/12
60000/60000 [==============================] - 45s 746us/step - loss: 0.0249 - acc: 0.9920 - val_loss: 4.0905 - val_acc: 0.362
Epoch 7/12
60000/60000 [==============================] - 45s 742us/step - loss: 0.0226 - acc: 0.9928 - val_loss: 0.1832 - val_acc: 0.946
Epoch 8/12
60000/60000 [==============================] - 45s 743us/step - loss: 0.0178 - acc: 0.9943 - val_loss: 0.5185 - val_acc: 0.872
Epoch 9/12
60000/60000 [==============================] - 45s 747us/step - loss: 0.0169 - acc: 0.9941 - val_loss: 0.8813 - val_acc: 0.799
Epoch 10/12
60000/60000 [==============================] - 45s 745us/step - loss: 0.0178 - acc: 0.9942 - val_loss: 0.1776 - val_acc: 0.953
Epoch 11/12
60000/60000 [==============================] - 45s 746us/step - loss: 0.0139 - acc: 0.9953 - val_loss: 0.3146 - val_acc: 0.916
Epoch 12/12
60000/60000 [==============================] - 45s 749us/step - loss: 0.0113 - acc: 0.9961 - val_loss: 0.1472 - val_acc: 0.961
Test loss: 0.14715414495209697
Test accuracy: 0.9612
```

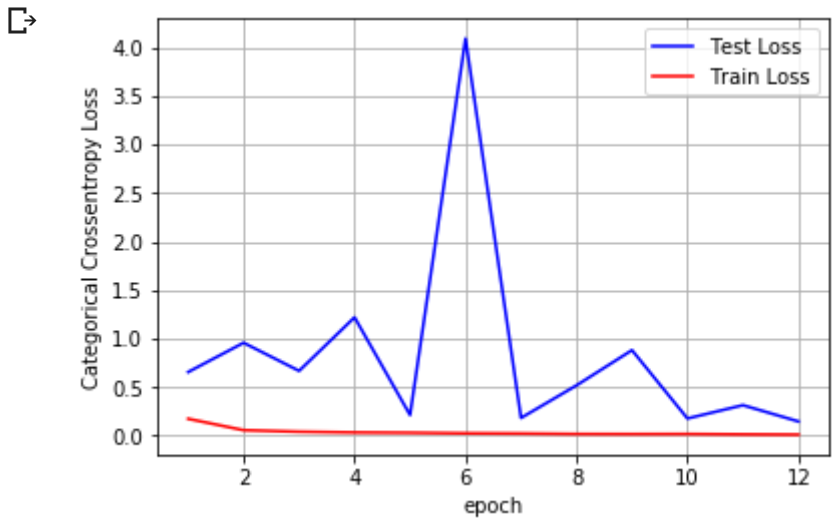
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



3.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(12, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(43, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(60, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(85, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(112, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(143, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
model.add(Conv2D(157, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(98, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())

model.add(Dense(56, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))


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))
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 [==============================] - 215s 4ms/step - loss: 0.5735 - acc: 0.8168 - val_loss: 0.1044 - val_acc: 0.9725
Epoch 2/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.1121 - acc: 0.9726 - val_loss: 0.0711 - val_acc: 0.9831
Epoch 3/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0739 - acc: 0.9814 - val_loss: 0.0470 - val_acc: 0.9873
Epoch 4/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0560 - acc: 0.9859 - val_loss: 0.0749 - val_acc: 0.9805
Epoch 5/12
60000/60000 [==============================] - 212s 4ms/step - loss: 0.0474 - acc: 0.9877 - val_loss: 0.0456 - val_acc: 0.9883
Epoch 6/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0398 - acc: 0.9901 - val_loss: 0.0426 - val_acc: 0.9889
Epoch 7/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0387 - acc: 0.9904 - val_loss: 0.0446 - val_acc: 0.9879
Epoch 8/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0305 - acc: 0.9922 - val_loss: 0.0417 - val_acc: 0.9893
Epoch 9/12
60000/60000 [==============================] - 212s 4ms/step - loss: 0.0263 - acc: 0.9932 - val_loss: 0.0385 - val_acc: 0.9906
Epoch 10/12
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0256 - acc: 0.9932 - val_loss: 0.0400 - val_acc: 0.9901
Epoch 11/12
60000/60000 [==============================] - 212s 4ms/step - loss: 0.0213 - acc: 0.9948 - val_loss: 0.0449 - val_acc: 0.9902
Epoch 12/12
60000/60000 [==============================] - 212s 4ms/step - loss: 0.0209 - acc: 0.9946 - val_loss: 0.0652 - val_acc: 0.9856
Test loss: 0.06518815791808301
Test accuracy: 0.9856
```

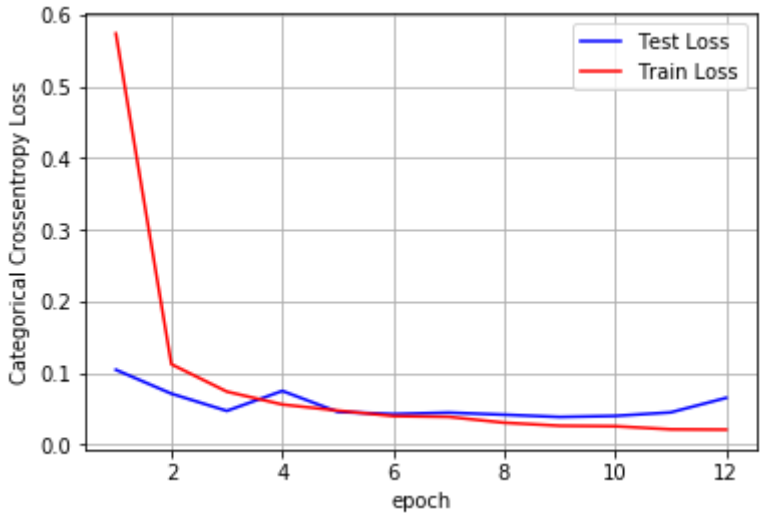
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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




```
#Activation function is relu in each layers
#Adam optimizers is used in each case
#MaxPooling is used in each case of cnn_layers of pool_size=(2, 2)

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["No. of CNN_Layers","kernal_size","Batch Normalisations","Dropout","No. of hidden layers", "Test loss", "Test Accuracy"]

x.add_row([3, (3,3), "No", 0.5, 1, 0.019, 0.99])
x.add_row([3, (2,2), "Yes", "No dropout", 1, 0.045, 0.98])
x.add_row([3, (5,5), "Yes", 0.3, 2, 0.025, 0.99])
x.add_row([5, (3,3), "No", 0.25, 1, 0.047, 0.98])
x.add_row([5, (2,2), "Yes", "No droput", 3, 0.078, 0.97])
x.add_row([5, (5,5), "Yes", 0.3, 3, 0.044, 0.99])
x.add_row([7, (3,3), "No", 0.25, 1, 0.043, 0.98])
x.add_row([7, (2,2), "Yes", "No dropout", 2, 0.147, 0.96])
x.add_row([7, (5,5), "Yes", 0.3, 2, 0.065, 0.98])
print(x)
```

No. of CNN_Layers	kernal_size	Batch Normalisations	Dropout	No. of hidden layers	Test loss	Test Accuracy
3	(3, 3)	No	0.5	1	0.019	0.99
3	(2, 2)	Yes	No dropout	1	0.045	0.98
3	(5, 5)	Yes	0.3	2	0.025	0.99
5	(3, 3)	No	0.25	1	0.047	0.98
5	(2, 2)	Yes	No droput	3	0.078	0.97
5	(5, 5)	Yes	0.3	3	0.044	0.99
7	(3, 3)	No	0.25	1	0.043	0.98
7	(2, 2)	Yes	No dropout	2	0.147	0.96
7	(5, 5)	Yes	0.3	2	0.065	0.98

step by step procedure for above experiment

1.Three different architecture of convolution layer has been implemented.There are three , five and seven convolution layers with "**Adam optimizers**" in each case

2.**First architecture i.e. three convolution layer** has been further experimented in three more categories.

3.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

4.For the first categories kernal size is (3,3), dropout=0.5 and maxpoolsize=(3,3).

5.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

6.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

7.for each categories test score and test accuracy has been evaluated with different parameter.

8.**Second architecture (i.e. five convolution layer)** has been further experimented in three more categories in the same way as first architecture.

9.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

10.For the first categories kernal size is (3,3), dropout=0.25 and maxpoolsize=(3,3).

11.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

12.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

13.for each categories test score and test accuracy has been evaluated with different parameter.

14.**Third architecture (i.e. Seven convolution layer)** has been further experimented in three more categories in the same way as first and second architecture.

15.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

16.For the first categories kernal size is (3,3), dropout=0.25 and maxpoolsize=(3,3).

17.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

18.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

19.for each categories test score and test accuracy has been evaluated with different parameter.

20.see the **preetytable** to observe the different result in different case

MODEL1 : Three Convolutional Network with RMSprop as optimizer

1.1 with kernal_size = (3,3), Dropout = 0.5 and MaxPooling(pool_size =(2,2))

```
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(Conv2D(96, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(75, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.RMSprop(),
              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 [==============================] - 183s 3ms/step - loss: 0.2702 - acc: 0.9158 - val_loss: 0.0478 - val_acc: 0.9850
Epoch 2/12
60000/60000 [==============================] - 179s 3ms/step - loss: 0.0852 - acc: 0.9754 - val_loss: 0.0388 - val_acc: 0.9871
Epoch 3/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0597 - acc: 0.9833 - val_loss: 0.0291 - val_acc: 0.9910
Epoch 4/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0484 - acc: 0.9860 - val_loss: 0.0252 - val_acc: 0.9924
Epoch 5/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0420 - acc: 0.9882 - val_loss: 0.0302 - val_acc: 0.9907
Epoch 6/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0377 - acc: 0.9895 - val_loss: 0.0255 - val_acc: 0.9916
Epoch 7/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0367 - acc: 0.9899 - val_loss: 0.0224 - val_acc: 0.9937
Epoch 8/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0370 - acc: 0.9901 - val_loss: 0.0269 - val_acc: 0.9924
Epoch 9/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0351 - acc: 0.9901 - val_loss: 0.0240 - val_acc: 0.9923
Epoch 10/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0361 - acc: 0.9905 - val_loss: 0.0308 - val_acc: 0.9919
Epoch 11/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0348 - acc: 0.9904 - val_loss: 0.0310 - val_acc: 0.9922
Epoch 12/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0343 - acc: 0.9906 - val_loss: 0.0264 - val_acc: 0.9929
Test loss: 0.026416385893837605
Test accuracy: 0.9929
```

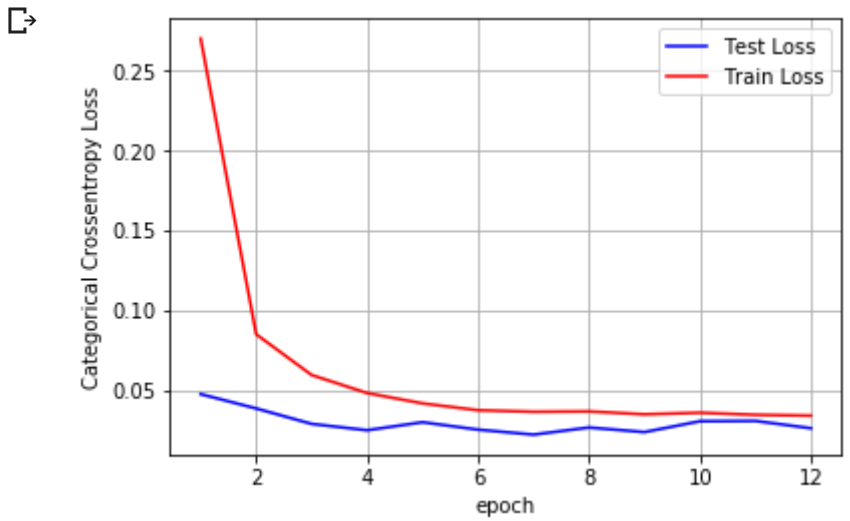
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



1.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(44, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(66, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

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

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.RMSprop(),
              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 [==============================] - 38s 639us/step - loss: 0.1467 - acc: 0.9554 - val_loss: 0.1191 - val_acc: 0.963
Epoch 2/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0515 - acc: 0.9841 - val_loss: 0.0580 - val_acc: 0.982
Epoch 3/12
60000/60000 [==============================] - 34s 570us/step - loss: 0.0360 - acc: 0.9892 - val_loss: 0.0840 - val_acc: 0.974
Epoch 4/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0281 - acc: 0.9911 - val_loss: 0.0596 - val_acc: 0.982
Epoch 5/12
60000/60000 [==============================] - 34s 566us/step - loss: 0.0218 - acc: 0.9932 - val_loss: 0.0499 - val_acc: 0.984
Epoch 6/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0179 - acc: 0.9943 - val_loss: 0.0514 - val_acc: 0.984
Epoch 7/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0146 - acc: 0.9951 - val_loss: 0.0488 - val_acc: 0.986
Epoch 8/12
60000/60000 [==============================] - 34s 567us/step - loss: 0.0125 - acc: 0.9958 - val_loss: 0.0473 - val_acc: 0.986
Epoch 9/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0100 - acc: 0.9968 - val_loss: 0.0515 - val_acc: 0.985
Epoch 10/12
60000/60000 [==============================] - 34s 569us/step - loss: 0.0098 - acc: 0.9965 - val_loss: 0.0507 - val_acc: 0.986
Epoch 11/12
60000/60000 [==============================] - 34s 568us/step - loss: 0.0076 - acc: 0.9975 - val_loss: 0.0622 - val_acc: 0.982
Epoch 12/12
60000/60000 [==============================] - 34s 572us/step - loss: 0.0068 - acc: 0.9978 - val_loss: 0.0746 - val_acc: 0.980
Test loss: 0.0746201684974978
Test accuracy: 0.9806
```

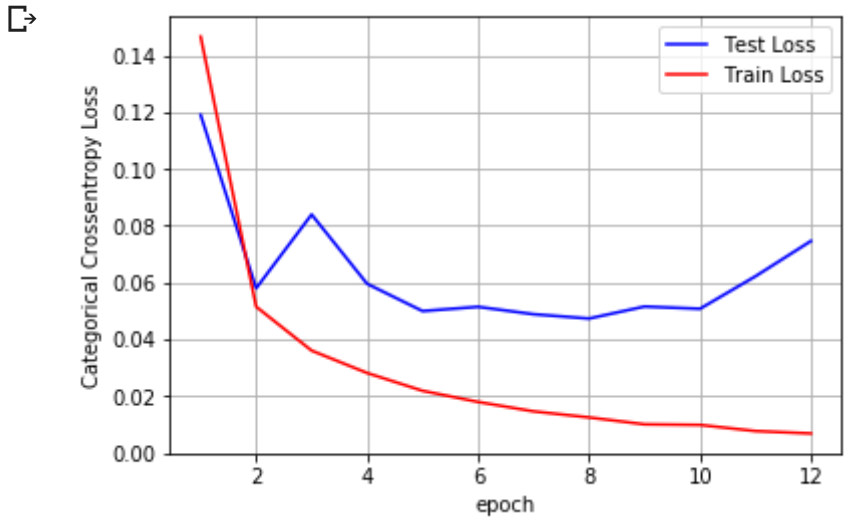
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
```

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy

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



1.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(44, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(89, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(165, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))

    model.add(Dense(97, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))

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

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.RMSprop(),
                  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 [==============================] - 195s 3ms/step - loss: 1.0009 - acc: 0.6653 - val_loss: 3.8346 - val_acc: 0.3946
Epoch 2/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.4697 - acc: 0.8464 - val_loss: 1.8055 - val_acc: 0.5506
Epoch 3/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.3592 - acc: 0.8845 - val_loss: 0.3017 - val_acc: 0.9077
Epoch 4/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.2422 - acc: 0.9255 - val_loss: 0.1909 - val_acc: 0.9426
Epoch 5/12
60000/60000 [==============================] - 188s 3ms/step - loss: 0.1920 - acc: 0.9413 - val_loss: 0.3387 - val_acc: 0.9023
Epoch 6/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.1515 - acc: 0.9537 - val_loss: 0.8797 - val_acc: 0.7712
Epoch 7/12
60000/60000 [==============================] - 192s 3ms/step - loss: 0.1412 - acc: 0.9584 - val_loss: 0.3180 - val_acc: 0.9071
Epoch 8/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.1341 - acc: 0.9601 - val_loss: 1.0845 - val_acc: 0.7558
Epoch 9/12
60000/60000 [==============================] - 192s 3ms/step - loss: 0.1196 - acc: 0.9643 - val_loss: 0.1963 - val_acc: 0.9442
Epoch 10/12
60000/60000 [==============================] - 194s 3ms/step - loss: 0.1096 - acc: 0.9672 - val_loss: 0.1150 - val_acc: 0.9621
Epoch 11/12
60000/60000 [==============================] - 193s 3ms/step - loss: 0.1006 - acc: 0.9697 - val_loss: 0.3534 - val_acc: 0.8884
Epoch 12/12
60000/60000 [==============================] - 189s 3ms/step - loss: 0.0934 - acc: 0.9713 - val_loss: 0.1269 - val_acc: 0.9603
Test loss: 0.12685609666118397
Test accuracy: 0.9603
```

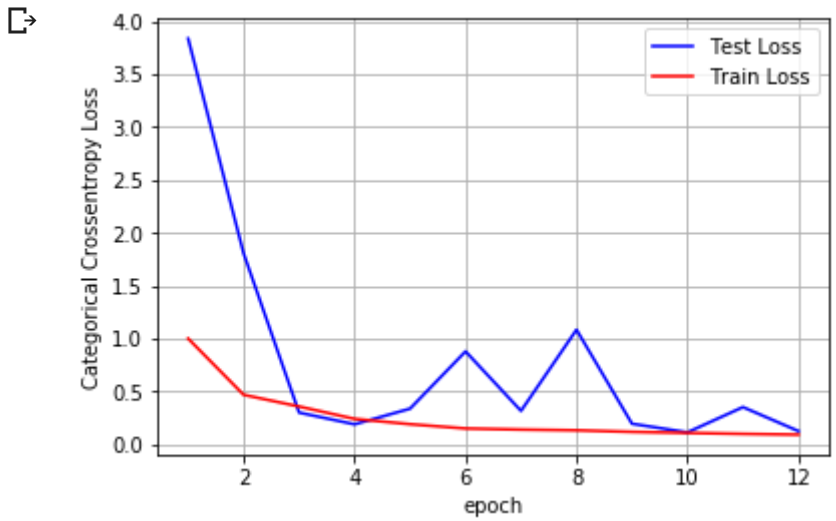
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



MODEL2 : Five Convolutional Network with RMSprop as optimizer

2.1 with kernal_size = (3,3), Dropout = 0.25 and MaxPooling(pool_size =(3,3))

```
model = Sequential()
model.add(Conv2D(25, kernel_size=(3, 3), activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(55, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(78, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(96, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(108, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.RMSprop(),
              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 [==============================] - 54s 900us/step - loss: 0.8243 - acc: 0.7197 - val_loss: 0.7556 - val_acc: 0.819
Epoch 2/12
60000/60000 [==============================] - 50s 834us/step - loss: 0.3361 - acc: 0.9033 - val_loss: 0.3976 - val_acc: 0.888
Epoch 3/12
60000/60000 [==============================] - 50s 834us/step - loss: 0.4302 - acc: 0.8970 - val_loss: 0.3611 - val_acc: 0.919
Epoch 4/12
60000/60000 [==============================] - 50s 836us/step - loss: 0.5801 - acc: 0.8971 - val_loss: 0.9266 - val_acc: 0.855
Epoch 5/12
60000/60000 [==============================] - 50s 833us/step - loss: 1.3400 - acc: 0.8779 - val_loss: 3.6306 - val_acc: 0.768
Epoch 6/12
60000/60000 [==============================] - 49s 822us/step - loss: 3.5538 - acc: 0.7767 - val_loss: 3.5029 - val_acc: 0.781
Epoch 7/12
60000/60000 [==============================] - 50s 829us/step - loss: 5.9515 - acc: 0.6299 - val_loss: 11.6605 - val_acc: 0.27
Epoch 8/12
60000/60000 [==============================] - 49s 824us/step - loss: 7.9589 - acc: 0.5059 - val_loss: 7.1494 - val_acc: 0.556
Epoch 9/12
60000/60000 [==============================] - 49s 816us/step - loss: 9.2002 - acc: 0.4290 - val_loss: 10.7258 - val_acc: 0.33
Epoch 10/12
60000/60000 [==============================] - 49s 820us/step - loss: 11.3325 - acc: 0.2968 - val_loss: 11.5988 - val_acc: 0.2
Epoch 11/12
60000/60000 [==============================] - 49s 818us/step - loss: 10.7564 - acc: 0.3326 - val_loss: 11.5162 - val_acc: 0.2
Epoch 12/12
60000/60000 [==============================] - 49s 815us/step - loss: 12.7129 - acc: 0.2112 - val_loss: 13.5779 - val_acc: 0.1
Test loss: 13.57788892364502
Test accuracy: 0.1576
```

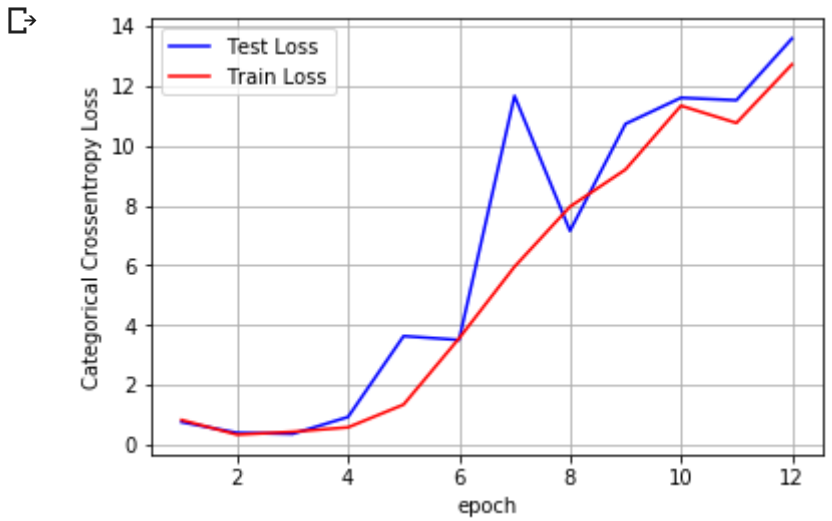
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



2.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(29, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(38, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(49, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(65, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(85, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(133, activation='relu'))
model.add(BatchNormalization())
```

```
model.add(Dense(95, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(76, activation='relu'))
model.add(BatchNormalization())

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

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.RMSprop(),
              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 [==============================] - 50s 840us/step - loss: 0.1994 - acc: 0.9389 - val_loss: 1.0779 - val_acc: 0.701
Epoch 2/12
60000/60000 [==============================] - 46s 761us/step - loss: 0.0684 - acc: 0.9791 - val_loss: 0.2030 - val_acc: 0.935
Epoch 3/12
60000/60000 [==============================] - 46s 762us/step - loss: 0.0495 - acc: 0.9851 - val_loss: 1.3340 - val_acc: 0.643
Epoch 4/12
60000/60000 [==============================] - 46s 765us/step - loss: 0.0399 - acc: 0.9873 - val_loss: 0.1545 - val_acc: 0.954
Epoch 5/12
60000/60000 [==============================] - 46s 769us/step - loss: 0.0350 - acc: 0.9886 - val_loss: 0.3375 - val_acc: 0.898
Epoch 6/12
60000/60000 [==============================] - 46s 767us/step - loss: 0.0287 - acc: 0.9910 - val_loss: 0.3413 - val_acc: 0.910
Epoch 7/12
60000/60000 [==============================] - 46s 763us/step - loss: 0.0253 - acc: 0.9918 - val_loss: 0.2752 - val_acc: 0.918
Epoch 8/12
60000/60000 [==============================] - 46s 764us/step - loss: 0.0230 - acc: 0.9926 - val_loss: 0.0723 - val_acc: 0.976
Epoch 9/12
60000/60000 [==============================] - 46s 760us/step - loss: 0.0201 - acc: 0.9935 - val_loss: 0.2592 - val_acc: 0.940
Epoch 10/12
60000/60000 [==============================] - 46s 759us/step - loss: 0.0180 - acc: 0.9942 - val_loss: 0.0851 - val_acc: 0.974
Epoch 11/12
60000/60000 [==============================] - 46s 763us/step - loss: 0.0172 - acc: 0.9943 - val_loss: 0.1641 - val_acc: 0.950
Epoch 12/12
60000/60000 [==============================] - 46s 761us/step - loss: 0.0152 - acc: 0.9951 - val_loss: 0.4397 - val_acc: 0.897
Test loss: 0.4397493788525462
Test accuracy: 0.8979
```

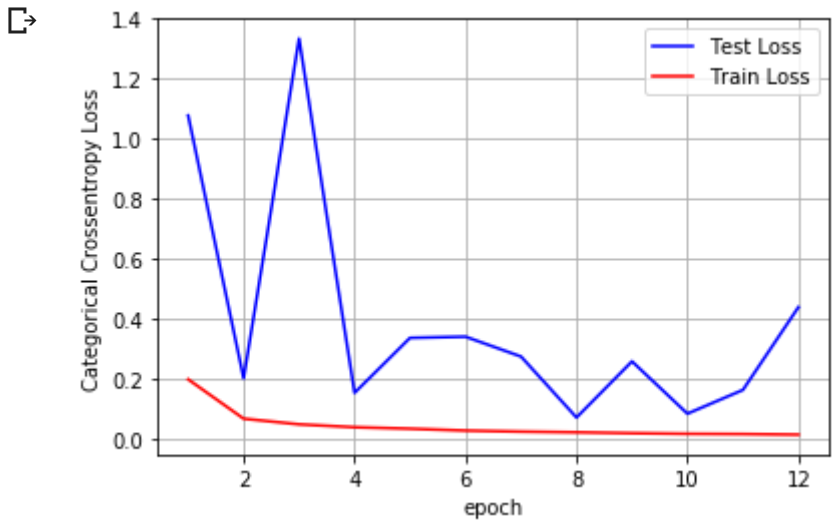
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



2.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(43, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(78, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(93, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(117, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(86, activation='relu'))
```

```
    model.add(Dropout(0.3))
    model.add(Dense(65, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(45, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.RMSprop(),
                  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 [==============================] - 208s 3ms/step - loss: 11.4338 - acc: 0.1039 - val_loss: 14.2887 - val_acc: 0.11
Epoch 2/12
60000/60000 [==============================] - 196s 3ms/step - loss: 13.2848 - acc: 0.1094 - val_loss: 14.2887 - val_acc: 0.11
Epoch 3/12
60000/60000 [==============================] - 200s 3ms/step - loss: 13.3054 - acc: 0.1082 - val_loss: 14.2887 - val_acc: 0.11
Epoch 4/12
60000/60000 [==============================] - 197s 3ms/step - loss: 13.2475 - acc: 0.1097 - val_loss: 14.2887 - val_acc: 0.11
Epoch 5/12
60000/60000 [==============================] - 200s 3ms/step - loss: 13.2673 - acc: 0.1094 - val_loss: 14.2887 - val_acc: 0.11
Epoch 6/12
60000/60000 [==============================] - 200s 3ms/step - loss: 13.2961 - acc: 0.1085 - val_loss: 14.2887 - val_acc: 0.11
Epoch 7/12
60000/60000 [==============================] - 197s 3ms/step - loss: 13.2780 - acc: 0.1099 - val_loss: 14.2887 - val_acc: 0.11
Epoch 8/12
60000/60000 [==============================] - 202s 3ms/step - loss: 13.2838 - acc: 0.1093 - val_loss: 14.2887 - val_acc: 0.11
Epoch 9/12
60000/60000 [==============================] - 202s 3ms/step - loss: 13.2500 - acc: 0.1097 - val_loss: 14.2887 - val_acc: 0.11
Epoch 10/12
60000/60000 [==============================] - 208s 3ms/step - loss: 13.2610 - acc: 0.1097 - val_loss: 14.2887 - val_acc: 0.11
Epoch 11/12
60000/60000 [==============================] - 207s 3ms/step - loss: 13.2702 - acc: 0.1098 - val_loss: 14.2887 - val_acc: 0.11
Epoch 12/12
60000/60000 [==============================] - 205s 3ms/step - loss: 13.2866 - acc: 0.1096 - val_loss: 14.2887 - val_acc: 0.11
Test loss: 14.28869146270752
Test accuracy: 0.1135
```

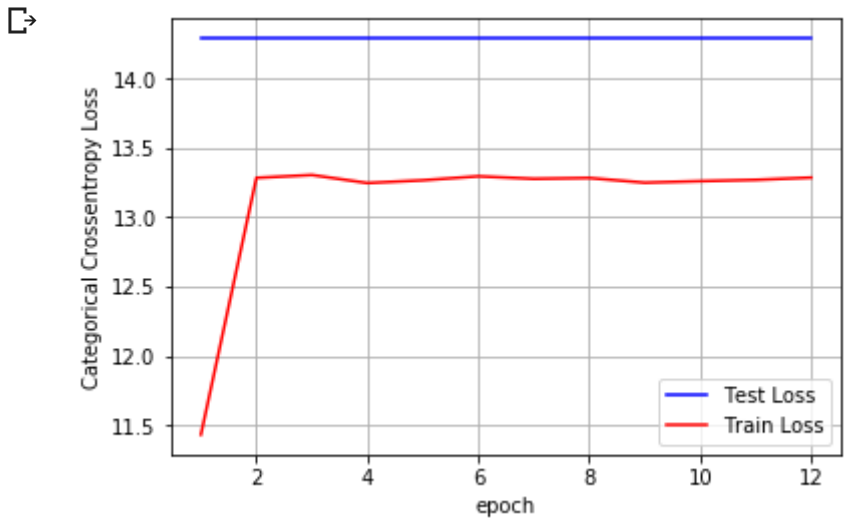
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



MODEL3 : Seven Convolutional Network with RMSprop as optimizer

3.1 with kernal_size = (3,3), Dropout = 0.25 and MaxPooling(pool_size =(3,3))

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

model.add(Conv2D(45, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(60, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(78, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

model.add(Conv2D(95, (3, 3), activation='relu', padding='same'))
```

```
    model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

    model.add(Conv2D(112, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

    model.add(Conv2D(145, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(3, 3), padding='same'))

    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.RMSprop(),
                  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 [==============================] - 57s 954us/step - loss: 2.3342 - acc: 0.1844 - val_loss: 2.3024 - val_acc: 0.113
Epoch 2/12
60000/60000 [==============================] - 53s 878us/step - loss: 2.3016 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 3/12
60000/60000 [==============================] - 52s 873us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 4/12
60000/60000 [==============================] - 52s 873us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 5/12
60000/60000 [==============================] - 52s 874us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 6/12
60000/60000 [==============================] - 52s 870us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 7/12
60000/60000 [==============================] - 52s 873us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Epoch 8/12
60000/60000 [==============================] - 52s 870us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 9/12
60000/60000 [==============================] - 52s 867us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 10/12
60000/60000 [==============================] - 52s 870us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 11/12
60000/60000 [==============================] - 52s 867us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3010 - val_acc: 0.113
Epoch 12/12
60000/60000 [==============================] - 52s 869us/step - loss: 2.3013 - acc: 0.1124 - val_loss: 2.3011 - val_acc: 0.113
Test loss: 2.3010625061035155
Test accuracy: 0.1135
```

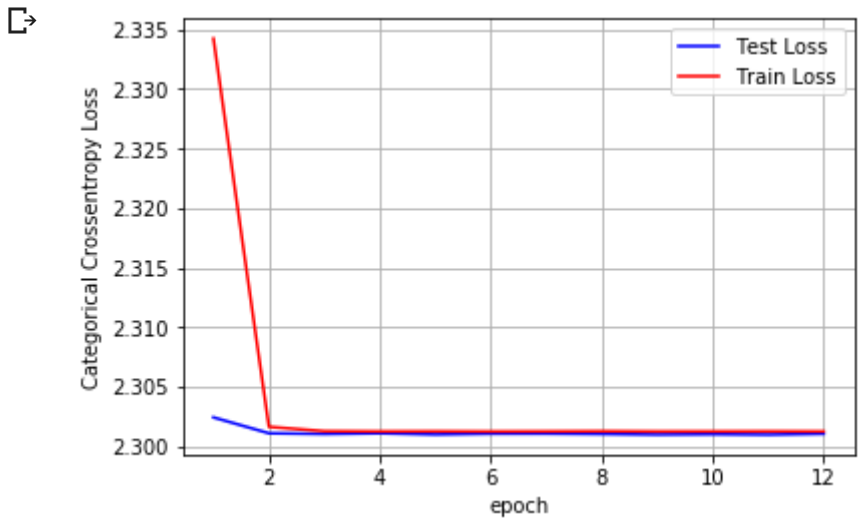
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



3.2 with kernal_size = (2,2), BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(22, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(45, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(65, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(97, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(114, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(126, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
    model.add(Conv2D(143, (2, 2), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

    model.add(Flatten())
    model.add(Dense(97, activation='relu'))
    model.add(BatchNormalization())

    model.add(Dense(65, activation='relu'))
    model.add(BatchNormalization())

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

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.RMSprop(),
                  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 [==============================] - 51s 852us/step - loss: 0.2137 - acc: 0.9356 - val_loss: 12.6879 - val_acc: 0.21
Epoch 2/12
60000/60000 [==============================] - 46s 758us/step - loss: 0.0835 - acc: 0.9751 - val_loss: 12.6732 - val_acc: 0.09
Epoch 3/12
60000/60000 [==============================] - 45s 750us/step - loss: 0.2566 - acc: 0.9276 - val_loss: 14.4918 - val_acc: 0.10
Epoch 4/12
60000/60000 [==============================] - 47s 776us/step - loss: 0.1935 - acc: 0.9418 - val_loss: 14.4902 - val_acc: 0.10
Epoch 5/12
60000/60000 [==============================] - 47s 782us/step - loss: 0.0639 - acc: 0.9805 - val_loss: 14.5477 - val_acc: 0.09
Epoch 6/12
60000/60000 [==============================] - 47s 784us/step - loss: 0.0788 - acc: 0.9764 - val_loss: 14.4918 - val_acc: 0.10
Epoch 7/12
60000/60000 [==============================] - 48s 794us/step - loss: 0.0602 - acc: 0.9819 - val_loss: 10.7008 - val_acc: 0.19
Epoch 8/12
60000/60000 [==============================] - 47s 783us/step - loss: 0.0403 - acc: 0.9876 - val_loss: 11.6996 - val_acc: 0.10
Epoch 9/12
60000/60000 [==============================] - 47s 779us/step - loss: 0.0362 - acc: 0.9881 - val_loss: 14.2824 - val_acc: 0.11
Epoch 10/12
60000/60000 [==============================] - 47s 780us/step - loss: 0.0509 - acc: 0.9839 - val_loss: 14.5740 - val_acc: 0.09
Epoch 11/12
60000/60000 [==============================] - 47s 778us/step - loss: 0.1001 - acc: 0.9695 - val_loss: 9.4936 - val_acc: 0.298
Epoch 12/12
60000/60000 [==============================] - 47s 783us/step - loss: 0.0902 - acc: 0.9722 - val_loss: 2.0876 - val_acc: 0.654
Test loss: 2.087562216567993
Test accuracy: 0.6546
```

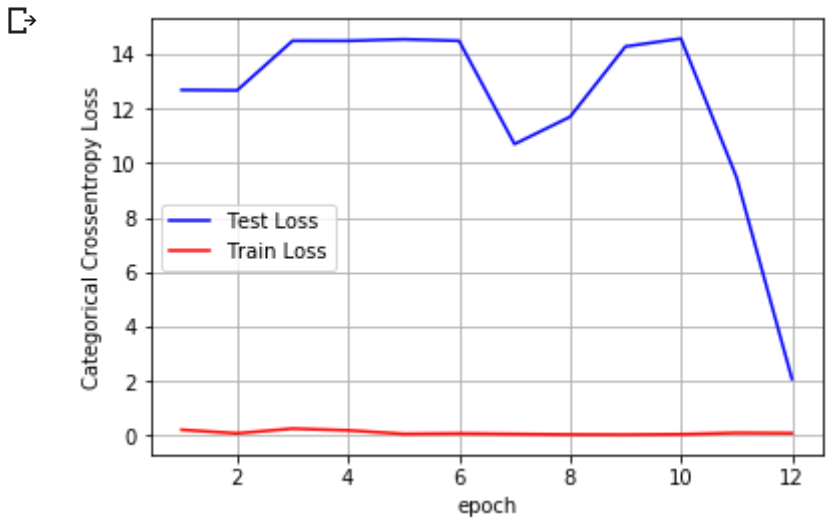
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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



3.3 with kernal_size = (5,5), Dropout = 0.3, BN and MaxPooling

```
model = Sequential()
model.add(Conv2D(12, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(43, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(60, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(85, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(112, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Conv2D(143, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
```

```
model.add(Conv2D(157, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model.add(Flatten())
model.add(Dense(98, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())

model.add(Dense(56, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))


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

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.RMSprop(),
              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 [==============================] - 221s 4ms/step - loss: 2.2941 - acc: 0.1903 - val_loss: 5.8873 - val_acc: 0.1135
Epoch 2/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.1791 - acc: 0.1547 - val_loss: 2.9495 - val_acc: 0.1135
Epoch 3/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3021 - acc: 0.1100 - val_loss: 2.3133 - val_acc: 0.1135
Epoch 4/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3018 - acc: 0.1108 - val_loss: 2.3051 - val_acc: 0.1135
Epoch 5/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3017 - acc: 0.1109 - val_loss: 2.3050 - val_acc: 0.0974
Epoch 6/12
60000/60000 [==============================] - 213s 4ms/step - loss: 2.3016 - acc: 0.1113 - val_loss: 2.3030 - val_acc: 0.1135
Epoch 7/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3016 - acc: 0.1119 - val_loss: 2.3042 - val_acc: 0.0974
Epoch 8/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3015 - acc: 0.1125 - val_loss: 2.3100 - val_acc: 0.1135
Epoch 9/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3016 - acc: 0.1123 - val_loss: 2.3022 - val_acc: 0.1028
Epoch 10/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3015 - acc: 0.1121 - val_loss: 2.3015 - val_acc: 0.1135
Epoch 11/12
60000/60000 [==============================] - 214s 4ms/step - loss: 2.3015 - acc: 0.1124 - val_loss: 2.3049 - val_acc: 0.1135
Epoch 12/12
60000/60000 [==============================] - 212s 4ms/step - loss: 2.3016 - acc: 0.1124 - val_loss: 2.3056 - val_acc: 0.1135
Test loss: 2.305607932281494
Test accuracy: 0.1135
```

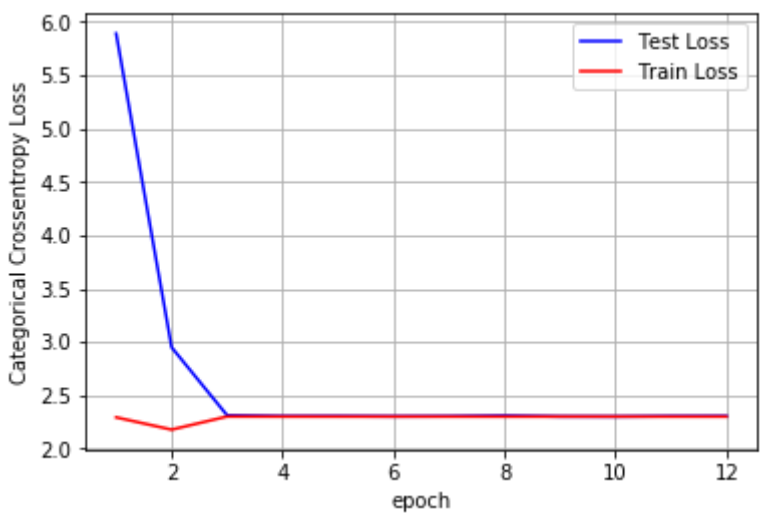
```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
```

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




```
#Activation function is relu in each layers
#RMSprop optimizers is used in each case
#MaxPooling is used in each case of cnn_layers of pool_size=(2, 2)

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["No. of CNN_Layers","kernal_size","Batch Normalisations","Dropout","No. of hidden layers", "Test loss", "Test Accuracy"]

x.add_row([3, (3,3), "No", 0.5, 1, 0.026, 0.99])
x.add_row([3, (2,2), "Yes", "No dropout", 1, 0.074, 0.98])
x.add_row([3, (5,5), "Yes", 0.3, 2, 0.126, 0.96])
x.add_row([5, (3,3), "No", 0.25, 1, 13.57, 0.15])
x.add_row([5, (2,2), "Yes", "No droput", 3, 0.439, 0.89])
x.add_row([5, (5,5), "Yes", 0.3, 3, 14.28, 0.11])
x.add_row([7, (3,3), "No", 0.25, 1, 2.30, 0.11])
x.add_row([7, (2,2), "Yes", "No dropout", 2, 2.08, 0.65])
x.add_row([7, (5,5), "Yes", 0.3, 2, 2.30, 0.11])
print(x)
```

No. of CNN_Layers	kernal_size	Batch Normalisations	Dropout	No. of hidden layers	Test loss	Test Accuracy
3	(3, 3)	No	0.5	1	0.026	0.99
3	(2, 2)	Yes	No dropout	1	0.074	0.98
3	(5, 5)	Yes	0.3	2	0.126	0.96
5	(3, 3)	No	0.25	1	13.57	0.15
5	(2, 2)	Yes	No droput	3	0.439	0.89
5	(5, 5)	Yes	0.3	3	14.28	0.11
7	(3, 3)	No	0.25	1	2.3	0.11
7	(2, 2)	Yes	No dropout	2	2.08	0.65
7	(5, 5)	Yes	0.3	2	2.3	0.11

step by step procedure far above experiment

1.Three different architecture of convolution layer has been implemented.There are three , five and seven convolution layers with "**Adadelta optimizers**" in each case

2.**First architecture i.e. three convolution layer** has been further experimented in three more categories.

3.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

4.For the first categories kernal size is (3,3), dropout=0.5 and maxpoolsize=(3,3).

5.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

6.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

7.for each categories test score and test accuracy has been evaluated with different parameter.

8.**Second architecture (i.e. five convolution layer)** has been further experimented in three more categories in the same way as first architecture.

9.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

10.For the first categories kernal size is (3,3), dropout=0.25 and maxpoolsize=(3,3).

11.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

12.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

13.for each categories test score and test accuracy has been evaluated with different parameter.

14.**Third architecture (i.e. Seven convolution layer)** has been further experimented in three more categories in the same way as first and second architecture.

15.In each categories, with different kernal size, dropout and maxpooling experiment has been performed

16.For the first categories kernal size is (3,3), dropout=0.25 and maxpoolsize=(3,3).

17.For the second categories kernal size is(2,2), same maxpooling as above with bathnormalisation and with no dropout experiment has been performed.

18.For the third categories kernal size is(5,5), same maxpooling with bathnormalisation and dropout= 0.3 experiment has been performed.

19.for each categories test score and test accuracy has been evaluated with different parameter.

20.see the **preetytable** to observe the different result in different case

Overall observations

1.We can observe Adam optimizer is better than Adadelta and RMSprop optimizer

2.With Adadelta optimizer test score is very worse when there is five and seven convolution layer

3.with batch normalisations and dropout accuracy can be improved much

4.with different kernal size we can see different result with lower kernal size test accuracy is much better

5.In this assignment I have experimented with different parameter and observe their result in each of the cases using prettytable.

6.I enjoyed a lot while doing this experiment.