In [0]:

```
batch_size = 128
num_classes = 10
epochs = 50

# input image dimensions
img_rows, img_cols = 28, 28
```

In [12]:

```
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
```

```
Using TensorFlow backend.
```

In [20]:

```
# 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)
```

```
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
```

In [0]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py




model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (4, 4), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

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

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

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

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 12s 204us/step - loss: 0.5542 - acc: 0.8218 - val_l
oss: 0.0947 - val_acc: 0.9721
Epoch 2/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.1532 - acc: 0.9567 -
val_loss: 0.0647 - val_acc: 0.9796
Epoch 3/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.1115 - acc: 0.9693 -
val_loss: 0.0588 - val_acc: 0.9823
Epoch 4/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0912 - acc: 0.9748 -
val_loss: 0.0494 - val_acc: 0.9853
Epoch 5/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0751 - acc: 0.9792 -
val_loss: 0.0441 - val_acc: 0.9874
Epoch 6/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0668 - acc: 0.9815 -
val_loss: 0.0404 - val_acc: 0.9882
Epoch 7/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0571 - acc: 0.9839 -
val_loss: 0.0500 - val_acc: 0.9859
Epoch 8/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0496 - acc: 0.9861 -
val_loss: 0.0451 - val_acc: 0.9872
Epoch 9/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0464 - acc: 0.9874 -
val_loss: 0.0379 - val_acc: 0.9888
Epoch 10/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0434 - acc: 0.9876 -
val_loss: 0.0363 - val_acc: 0.9896
Epoch 11/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0392 - acc: 0.9890 -
val_loss: 0.0381 - val_acc: 0.9886
Epoch 12/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0358 - acc: 0.9901 -
val_loss: 0.0373 - val_acc: 0.9903
Epoch 13/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0328 - acc: 0.9911 -
val_loss: 0.0373 - val_acc: 0.9912
Epoch 14/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0309 - acc: 0.9911 -
val_loss: 0.0325 - val_acc: 0.9912
Epoch 15/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0274 - acc: 0.9921 -
val_loss: 0.0380 - val_acc: 0.9907
Epoch 16/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0254 - acc: 0.9929 -
val_loss: 0.0392 - val_acc: 0.9905
Epoch 17/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0246 - acc: 0.9933 -
val_loss: 0.0405 - val_acc: 0.9910
Epoch 18/50
```

```
Epoch 18/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0226 - acc: 0.9937 -
val_loss: 0.0407 - val_acc: 0.9907
Epoch 19/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0226 - acc: 0.9936 -
val_loss: 0.0398 - val_acc: 0.9906
Epoch 20/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0200 - acc: 0.9946 -
val_loss: 0.0405 - val_acc: 0.9913
Epoch 21/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0192 - acc: 0.9948 -
val_loss: 0.0455 - val_acc: 0.9896
Epoch 22/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0184 - acc: 0.9945 -
val_loss: 0.0457 - val_acc: 0.9901
Epoch 23/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0188 - acc: 0.9947 -
val_loss: 0.0399 - val_acc: 0.9907
Epoch 24/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0174 - acc: 0.9949 -
val_loss: 0.0427 - val_acc: 0.9914
Epoch 25/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0162 - acc: 0.9956 -
val_loss: 0.0415 - val_acc: 0.9904
Epoch 26/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0154 - acc: 0.9958 -
val_loss: 0.0443 - val_acc: 0.9914
Epoch 27/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0160 - acc: 0.9955 -
val_loss: 0.0412 - val_acc: 0.9912
Epoch 28/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0143 - acc: 0.9960 -
val_loss: 0.0411 - val_acc: 0.9921
Epoch 29/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0139 - acc: 0.9961 -
val_loss: 0.0488 - val_acc: 0.9914
Epoch 30/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0141 - acc: 0.9957 -
val_loss: 0.0463 - val_acc: 0.9914
Epoch 31/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0121 - acc: 0.9967 -
val_loss: 0.0550 - val_acc: 0.9910
Epoch 32/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0130 - acc: 0.9965 -
val_loss: 0.0470 - val_acc: 0.9911
Epoch 33/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0115 - acc: 0.9964 -
val_loss: 0.0519 - val_acc: 0.9908
Epoch 34/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0134 - acc: 0.9963 -
val_loss: 0.0441 - val_acc: 0.9913
Epoch 35/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0114 - acc: 0.9967 -
val_loss: 0.0501 - val_acc: 0.9911
Epoch 36/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0106 - acc: 0.9971 -
val_loss: 0.0538 - val_acc: 0.9906
Epoch 37/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0110 - acc: 0.9969 -
val_loss: 0.0478 - val_acc: 0.9918
Epoch 38/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0110 - acc: 0.9968 -
val_loss: 0.0490 - val_acc: 0.9912
Epoch 39/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0099 - acc: 0.9973 -
val_loss: 0.0548 - val_acc: 0.9892
Epoch 40/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0104 - acc: 0.9971 -
val_loss: 0.0507 - val_acc: 0.9905
Epoch 41/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0089 - acc: 0.9974 -
val_loss: 0.0499 - val_acc: 0.9915
Epoch 42/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0095 - acc: 0.9974 -
val_loss: 0.0577 - val_acc: 0.9912
Epoch 43/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0097 - acc: 0.9972 -
```

```
val_loss: 0.0511 - val_acc: 0.9908
Epoch 44/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0099 - acc: 0.9975 -
val_loss: 0.0550 - val_acc: 0.9897
Epoch 45/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0102 - acc: 0.9973 -
val_loss: 0.0585 - val_acc: 0.9903
Epoch 46/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0091 - acc: 0.9974 -
val_loss: 0.0575 - val_acc: 0.9913
Epoch 47/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.0090 - acc: 0.9975 -
val_loss: 0.0546 - val_acc: 0.9919
Epoch 48/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0092 - acc: 0.9975 -
val_loss: 0.0464 - val_acc: 0.9917
Epoch 49/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0100 - acc: 0.9972 -
val_loss: 0.0531 - val_acc: 0.9912
Epoch 50/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0089 - acc: 0.9976 -
val_loss: 0.0615 - val_acc: 0.9914
Test loss: 0.06147524185255345
Test accuracy: 0.9914
```

In [0]:

```
%matplotlib notebook
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [0]:

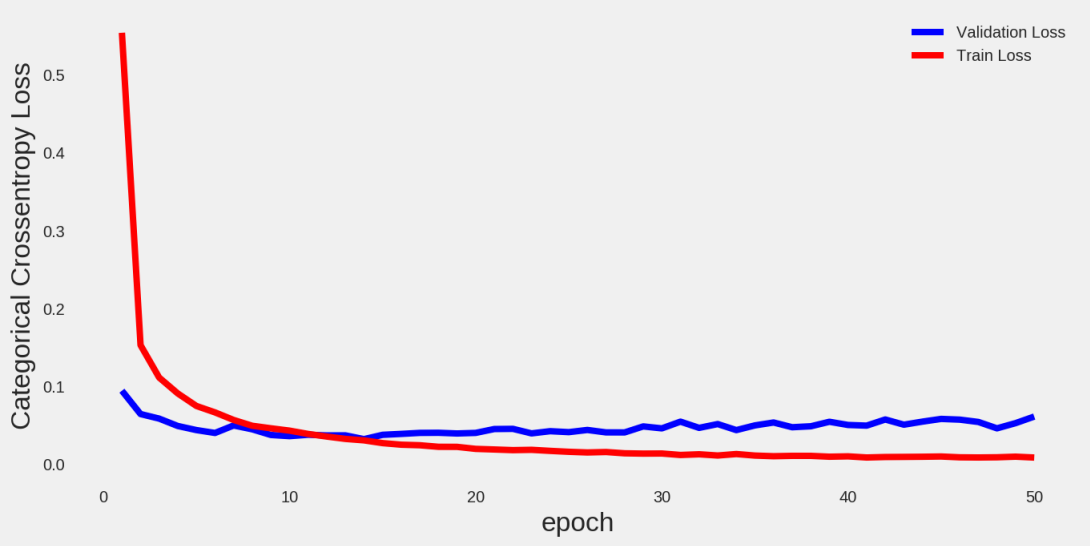
```
import warnings
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = [10, 5]
warnings.filterwarnings("ignore", category=FutureWarning)
%config InlineBackend.figure_format = 'retina'
```

In [0]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, va
lidation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs
```

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

```
Test score: 0.06147524185255345
Test accuracy: 0.9914
```



Adam optimizer

In [0]:

```
# 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



# 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)

model = Sequential()
model.add(Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (4, 4), activation='relu'))
```

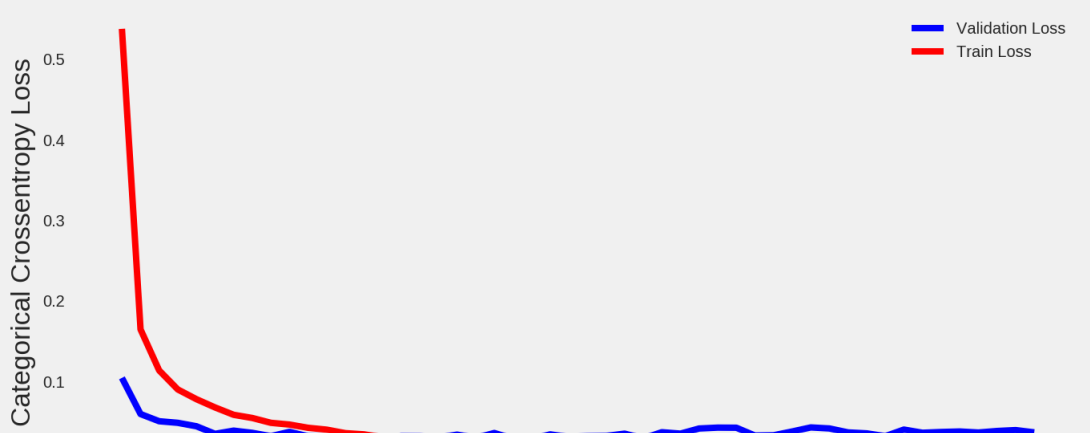
```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))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

model.compile(optimizer='adam',loss=keras.losses.categorical_crossentropy,
              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])
```

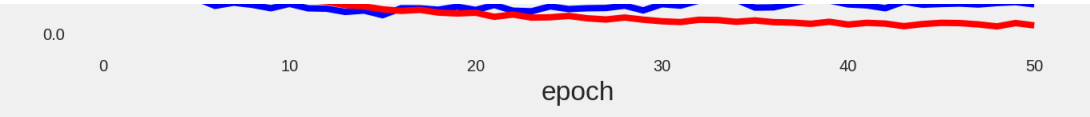
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, va
lidation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

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

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 13s 212us/step - loss: 0.5378 - acc: 0.8265 - val_l
oss: 0.1044 - val_acc: 0.9673
Epoch 2/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.1646 - acc: 0.9543 -
val_loss: 0.0596 - val_acc: 0.9810
Epoch 3/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.1137 - acc: 0.9684 -
val_loss: 0.0506 - val_acc: 0.9852
Epoch 4/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0901 - acc: 0.9745 -
val_loss: 0.0487 - val_acc: 0.9856
Epoch 5/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0780 - acc: 0.9789 -
val_loss: 0.0444 - val_acc: 0.9861
Epoch 6/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0678 - acc: 0.9816 -
val_loss: 0.0350 - val_acc: 0.9897
```

```
Epoch 7/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0586 - acc: 0.9836 -
val_loss: 0.0389 - val_acc: 0.9892
Epoch 8/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0547 - acc: 0.9845 -
val_loss: 0.0362 - val_acc: 0.9891
Epoch 9/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0487 - acc: 0.9862 -
val_loss: 0.0320 - val_acc: 0.9911
Epoch 10/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0465 - acc: 0.9869 -
val_loss: 0.0375 - val_acc: 0.9895
Epoch 11/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0425 - acc: 0.9883 -
val_loss: 0.0319 - val_acc: 0.9904
Epoch 12/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0402 - acc: 0.9887 -
val_loss: 0.0315 - val_acc: 0.9917
Epoch 13/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0358 - acc: 0.9897 -
val_loss: 0.0274 - val_acc: 0.9922
Epoch 14/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0343 - acc: 0.9905 -
val_loss: 0.0292 - val_acc: 0.9918
Epoch 15/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0307 - acc: 0.9914 -
val_loss: 0.0234 - val_acc: 0.9941
Epoch 16/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0288 - acc: 0.9918 -
val_loss: 0.0320 - val_acc: 0.9913
Epoch 17/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0301 - acc: 0.9916 -
val_loss: 0.0319 - val_acc: 0.9917
Epoch 18/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0266 - acc: 0.9924 -
val_loss: 0.0302 - val_acc: 0.9927
Epoch 19/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0255 - acc: 0.9929 -
val_loss: 0.0340 - val_acc: 0.9915
Epoch 20/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0265 - acc: 0.9926 -
val_loss: 0.0299 - val_acc: 0.9919
Epoch 21/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0213 - acc: 0.9939 -
val_loss: 0.0360 - val_acc: 0.9904
Epoch 22/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0243 - acc: 0.9926 -
val_loss: 0.0289 - val_acc: 0.9934
Epoch 23/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0204 - acc: 0.9942 -
val_loss: 0.0282 - val_acc: 0.9935
Epoch 24/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0208 - acc: 0.9936 -
val_loss: 0.0343 - val_acc: 0.9931
Epoch 25/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0225 - acc: 0.9937 -
val_loss: 0.0310 - val_acc: 0.9928
Epoch 26/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0195 - acc: 0.9944 -
val_loss: 0.0320 - val_acc: 0.9924
Epoch 27/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0180 - acc: 0.9952 -
val_loss: 0.0323 - val_acc: 0.9924
Epoch 28/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0205 - acc: 0.9938 -
val_loss: 0.0352 - val_acc: 0.9913
Epoch 29/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0178 - acc: 0.9944 -
val_loss: 0.0293 - val_acc: 0.9927
Epoch 30/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0158 - acc: 0.9951 -
val_loss: 0.0370 - val_acc: 0.9919
Epoch 31/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0148 - acc: 0.9956 -
val_loss: 0.0352 - val_acc: 0.9924
Epoch 32/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0177 - acc: 0.9946 -
```

```
val_loss: 0.0416 - val_acc: 0.9912
Epoch 33/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0173 - acc: 0.9949 -
val_loss: 0.0427 - val_acc: 0.9907
Epoch 34/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0151 - acc: 0.9956 -
val_loss: 0.0426 - val_acc: 0.9910
Epoch 35/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0169 - acc: 0.9949 -
val_loss: 0.0328 - val_acc: 0.9931
Epoch 36/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0148 - acc: 0.9958 -
val_loss: 0.0332 - val_acc: 0.9928
Epoch 37/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0142 - acc: 0.9961 -
val_loss: 0.0379 - val_acc: 0.9934
Epoch 38/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0127 - acc: 0.9964 -
val_loss: 0.0430 - val_acc: 0.9915
Epoch 39/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0152 - acc: 0.9960 -
val_loss: 0.0417 - val_acc: 0.9913
Epoch 40/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0117 - acc: 0.9966 -
val_loss: 0.0368 - val_acc: 0.9926
Epoch 41/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0139 - acc: 0.9963 -
val_loss: 0.0356 - val_acc: 0.9927
Epoch 42/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0129 - acc: 0.9961 -
val_loss: 0.0320 - val_acc: 0.9925
Epoch 43/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0097 - acc: 0.9972 -
val_loss: 0.0403 - val_acc: 0.9923
Epoch 44/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0124 - acc: 0.9964 -
val_loss: 0.0363 - val_acc: 0.9938
Epoch 45/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0139 - acc: 0.9960 -
val_loss: 0.0373 - val_acc: 0.9936
Epoch 46/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0136 - acc: 0.9963 -
val_loss: 0.0379 - val_acc: 0.9930
Epoch 47/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0119 - acc: 0.9968 -
val_loss: 0.0366 - val_acc: 0.9933
Epoch 48/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0095 - acc: 0.9971 -
val_loss: 0.0385 - val_acc: 0.9930
Epoch 49/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.0136 - acc: 0.9964 -
val_loss: 0.0396 - val_acc: 0.9932
Epoch 50/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0107 - acc: 0.9968 -
val_loss: 0.0368 - val_acc: 0.9925
Test loss: 0.036823302400706436
Test accuracy: 0.9925
Test score: 0.036823302400706436
Test accuracy: 0.9925
```





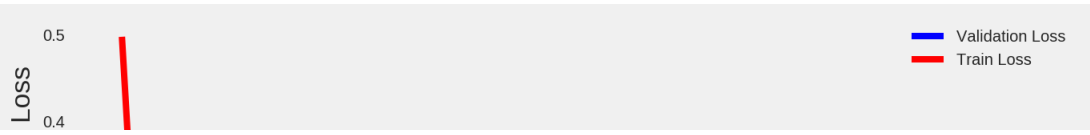
In [0]:

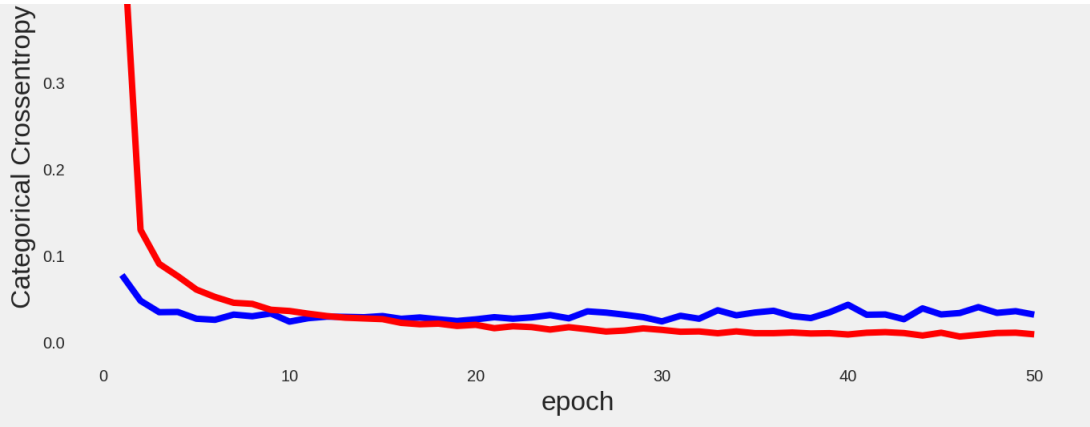
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu'))

model.add(Conv2D(64, (4, 4), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam',loss=keras.losses.categorical_crossentropy,
              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])




score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, va
lidation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

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

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 16s 270us/step - loss: 0.4977 - acc: 0.8391 - val_l
oss: 0.0779 - val_acc: 0.9766
Epoch 2/50
60000/60000 [==============================] - 12s 202us/step - loss: 0.1300 - acc: 0.9636 - val_l
oss: 0.0481 - val_acc: 0.9852
```

```
Epoch 3/50
60000/60000 [==============================] - 12s 202us/step - loss: 0.0908 - acc: 0.9756 - val_l
oss: 0.0349 - val_acc: 0.9901
Epoch 4/50
60000/60000 [==============================] - 12s 202us/step - loss: 0.0765 - acc: 0.9792 - val_l
oss: 0.0351 - val_acc: 0.9890
Epoch 5/50
60000/60000 [==============================] - 12s 202us/step - loss: 0.0610 - acc: 0.9834 - val_l
oss: 0.0273 - val_acc: 0.9906
Epoch 6/50
60000/60000 [==============================] - 12s 202us/step - loss: 0.0525 - acc: 0.9858 - val_l
oss: 0.0261 - val_acc: 0.9920
Epoch 7/50
60000/60000 [==============================] - 12s 201us/step - loss: 0.0458 - acc: 0.9880 - val_l
oss: 0.0321 - val_acc: 0.9906
Epoch 8/50
60000/60000 [==============================] - 12s 201us/step - loss: 0.0445 - acc: 0.9880 - val_l
oss: 0.0302 - val_acc: 0.9916
Epoch 9/50
60000/60000 [==============================] - 12s 202us/step - loss: 0.0377 - acc: 0.9902 - val_l
oss: 0.0334 - val_acc: 0.9910
Epoch 10/50
60000/60000 [==============================] - 12s 201us/step - loss: 0.0363 - acc: 0.9896 - val_l
oss: 0.0240 - val_acc: 0.9930
Epoch 11/50
60000/60000 [==============================] - 12s 201us/step - loss: 0.0332 - acc: 0.9909 - val_l
oss: 0.0279 - val_acc: 0.9925
Epoch 12/50
60000/60000 [==============================] - 12s 201us/step - loss: 0.0304 - acc: 0.9915 - val_l
oss: 0.0297 - val_acc: 0.9919
Epoch 13/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0285 - acc: 0.9921 - val_l
oss: 0.0295 - val_acc: 0.9921
Epoch 14/50
60000/60000 [==============================] - 12s 201us/step - loss: 0.0277 - acc: 0.9924 - val_l
oss: 0.0290 - val_acc: 0.9908
Epoch 15/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0269 - acc: 0.9927 - val_l
oss: 0.0304 - val_acc: 0.9916
Epoch 16/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0225 - acc: 0.9936 - val_l
oss: 0.0272 - val_acc: 0.9925
Epoch 17/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0209 - acc: 0.9940 - val_l
oss: 0.0288 - val_acc: 0.9926
Epoch 18/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0216 - acc: 0.9941 - val_l
oss: 0.0266 - val_acc: 0.9928
Epoch 19/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0190 - acc: 0.9946 - val_l
oss: 0.0247 - val_acc: 0.9931
Epoch 20/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0203 - acc: 0.9944 - val_l
oss: 0.0266 - val_acc: 0.9939
Epoch 21/50
60000/60000 [==============================] - 12s 198us/step - loss: 0.0164 - acc: 0.9953 - val_l
oss: 0.0291 - val_acc: 0.9934
Epoch 22/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0186 - acc: 0.9948 - val_l
oss: 0.0273 - val_acc: 0.9931
Epoch 23/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0177 - acc: 0.9951 - val_l
oss: 0.0288 - val_acc: 0.9934
Epoch 24/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0148 - acc: 0.9959 - val_l
oss: 0.0316 - val_acc: 0.9931
Epoch 25/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0175 - acc: 0.9951 - val_l
oss: 0.0278 - val_acc: 0.9930
Epoch 26/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0152 - acc: 0.9955 - val_l
oss: 0.0359 - val_acc: 0.9924
Epoch 27/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0125 - acc: 0.9962 - val_l
oss: 0.0344 - val_acc: 0.9939
Epoch 28/50
60000/60000 [==============================] - 12s 198us/step - loss: 0.0137 - acc: 0.9965 - val_l
```

```
oss: 0.0320 - val_acc: 0.9936
Epoch 29/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0161 - acc: 0.9955 - val_l
oss: 0.0292 - val_acc: 0.9929
Epoch 30/50
60000/60000 [==============================] - 12s 198us/step - loss: 0.0144 - acc: 0.9960 - val_l
oss: 0.0242 - val_acc: 0.9943
Epoch 31/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0123 - acc: 0.9967 - val_l
oss: 0.0307 - val_acc: 0.9934
Epoch 32/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0126 - acc: 0.9965 - val_l
oss: 0.0273 - val_acc: 0.9941
Epoch 33/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0105 - acc: 0.9967 - val_l
oss: 0.0371 - val_acc: 0.9920
Epoch 34/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0127 - acc: 0.9963 - val_l
oss: 0.0312 - val_acc: 0.9939
Epoch 35/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0106 - acc: 0.9969 - val_l
oss: 0.0345 - val_acc: 0.9926
Epoch 36/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0106 - acc: 0.9968 - val_l
oss: 0.0365 - val_acc: 0.9937
Epoch 37/50
60000/60000 [==============================] - 12s 198us/step - loss: 0.0114 - acc: 0.9968 - val_l
oss: 0.0304 - val_acc: 0.9931
Epoch 38/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0103 - acc: 0.9972 - val_l
oss: 0.0281 - val_acc: 0.9934
Epoch 39/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0106 - acc: 0.9969 - val_l
oss: 0.0347 - val_acc: 0.9934
Epoch 40/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0091 - acc: 0.9975 - val_l
oss: 0.0435 - val_acc: 0.9920
Epoch 41/50
60000/60000 [==============================] - 12s 198us/step - loss: 0.0111 - acc: 0.9969 - val_l
oss: 0.0319 - val_acc: 0.9940
Epoch 42/50
60000/60000 [==============================] - 12s 198us/step - loss: 0.0118 - acc: 0.9968 - val_l
oss: 0.0323 - val_acc: 0.9924
Epoch 43/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0107 - acc: 0.9969 - val_l
oss: 0.0267 - val_acc: 0.9947
Epoch 44/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0079 - acc: 0.9978 - val_l
oss: 0.0393 - val_acc: 0.9917
Epoch 45/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0111 - acc: 0.9967 - val_l
oss: 0.0323 - val_acc: 0.9947
Epoch 46/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0067 - acc: 0.9983 - val_l
oss: 0.0339 - val_acc: 0.9939
Epoch 47/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0087 - acc: 0.9976 - val_l
oss: 0.0407 - val_acc: 0.9926
Epoch 48/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0108 - acc: 0.9971 - val_l
oss: 0.0341 - val_acc: 0.9932
Epoch 49/50
60000/60000 [==============================] - 12s 199us/step - loss: 0.0112 - acc: 0.9969 - val_l
oss: 0.0361 - val_acc: 0.9935
Epoch 50/50
60000/60000 [==============================] - 12s 200us/step - loss: 0.0093 - acc: 0.9974 - val_l
oss: 0.0321 - val_acc: 0.9935
Test loss: 0.03206922948194228
Test accuracy: 0.9935
Test score: 0.03206922948194228
Test accuracy: 0.9935
```





Batch Normalisation

In [0]:

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

In [21]:

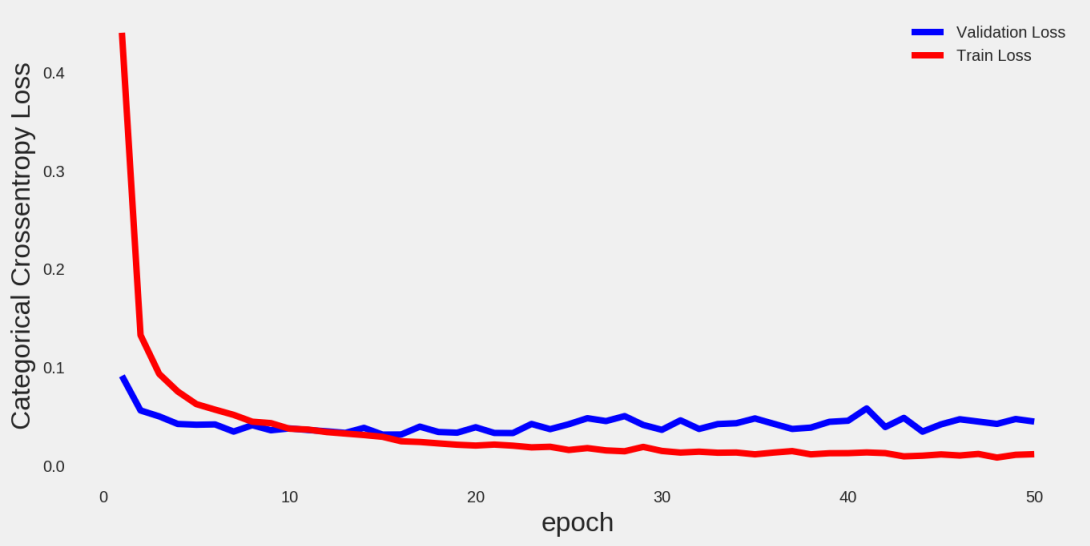
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=input_shape))
BatchNormalization(axis=1)
model.add(Conv2D(64, (4, 4), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
BatchNormalization(axis=1)
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
BatchNormalization(axis=1)
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
BatchNormalization(axis=1)
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam',loss=keras.losses.categorical_crossentropy,
              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])




score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, va
lidation_data=(X_test, Y_test))
```

```
# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

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

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future
version.
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/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 14s 230us/step - loss: 0.4405 - acc: 0.8596 - val_l
oss: 0.0913 - val_acc: 0.9722
Epoch 2/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.1328 - acc: 0.9607 -
val_loss: 0.0561 - val_acc: 0.9828
Epoch 3/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0934 - acc: 0.9730 -
val_loss: 0.0501 - val_acc: 0.9859
Epoch 4/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0754 - acc: 0.9779 -
val_loss: 0.0424 - val_acc: 0.9883
Epoch 5/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.0626 - acc: 0.9820 -
val_loss: 0.0416 - val_acc: 0.9879
Epoch 6/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0570 - acc: 0.9830 -
val_loss: 0.0419 - val_acc: 0.9883
Epoch 7/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0516 - acc: 0.9847 -
val_loss: 0.0347 - val_acc: 0.9907
Epoch 8/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0447 - acc: 0.9868 -
val_loss: 0.0411 - val_acc: 0.9891
Epoch 9/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0432 - acc: 0.9870 -
val_loss: 0.0360 - val_acc: 0.9914
Epoch 10/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0376 - acc: 0.9882 -
val_loss: 0.0379 - val_acc: 0.9899
Epoch 11/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0367 - acc: 0.9891 -
val_loss: 0.0362 - val_acc: 0.9905
Epoch 12/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0341 - acc: 0.9897 -
val_loss: 0.0349 - val_acc: 0.9911
Epoch 13/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0326 - acc: 0.9901 -
val_loss: 0.0333 - val_acc: 0.9903
Epoch 14/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0311 - acc: 0.9906 -
val_loss: 0.0384 - val_acc: 0.9897
Epoch 15/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0293 - acc: 0.9910 -
val_loss: 0.0316 - val_acc: 0.9911
Epoch 16/50
```

```
60000/60000 [==============================] - 9s 153us/step - loss: 0.0248 - acc: 0.9926 -
val_loss: 0.0317 - val_acc: 0.9906
Epoch 17/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0241 - acc: 0.9924 -
val_loss: 0.0397 - val_acc: 0.9896
Epoch 18/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0226 - acc: 0.9929 -
val_loss: 0.0342 - val_acc: 0.9912
Epoch 19/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0213 - acc: 0.9931 -
val_loss: 0.0335 - val_acc: 0.9917
Epoch 20/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0204 - acc: 0.9936 -
val_loss: 0.0390 - val_acc: 0.9913
Epoch 21/50
60000/60000 [==============================] - 9s 153us/step - loss: 0.0214 - acc: 0.9936 -
val_loss: 0.0332 - val_acc: 0.9914
Epoch 22/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0203 - acc: 0.9937 -
val_loss: 0.0331 - val_acc: 0.9919
Epoch 23/50
60000/60000 [==============================] - 9s 154us/step - loss: 0.0186 - acc: 0.9946 -
val_loss: 0.0424 - val_acc: 0.9906
Epoch 24/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.0192 - acc: 0.9939 -
val_loss: 0.0371 - val_acc: 0.9902
Epoch 25/50
60000/60000 [==============================] - 9s 146us/step - loss: 0.0159 - acc: 0.9953 -
val_loss: 0.0421 - val_acc: 0.9896
Epoch 26/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0178 - acc: 0.9943 -
val_loss: 0.0482 - val_acc: 0.9882
Epoch 27/50
60000/60000 [==============================] - 9s 146us/step - loss: 0.0155 - acc: 0.9952 -
val_loss: 0.0453 - val_acc: 0.9905
Epoch 28/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0146 - acc: 0.9954 -
val_loss: 0.0504 - val_acc: 0.9903
Epoch 29/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0191 - acc: 0.9946 -
val_loss: 0.0414 - val_acc: 0.9910
Epoch 30/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0149 - acc: 0.9954 -
val_loss: 0.0364 - val_acc: 0.9912
Epoch 31/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0133 - acc: 0.9959 -
val_loss: 0.0461 - val_acc: 0.9899
Epoch 32/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0142 - acc: 0.9956 -
val_loss: 0.0373 - val_acc: 0.9910
Epoch 33/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0131 - acc: 0.9960 -
val_loss: 0.0423 - val_acc: 0.9912
Epoch 34/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0133 - acc: 0.9961 -
val_loss: 0.0432 - val_acc: 0.9912
Epoch 35/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0115 - acc: 0.9965 -
val_loss: 0.0480 - val_acc: 0.9899
Epoch 36/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0133 - acc: 0.9959 -
val_loss: 0.0426 - val_acc: 0.9914
Epoch 37/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0148 - acc: 0.9951 -
val_loss: 0.0373 - val_acc: 0.9923
Epoch 38/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0115 - acc: 0.9964 -
val_loss: 0.0387 - val_acc: 0.9916
Epoch 39/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0126 - acc: 0.9963 -
val_loss: 0.0446 - val_acc: 0.9917
Epoch 40/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0127 - acc: 0.9964 -
val_loss: 0.0457 - val_acc: 0.9909
Epoch 41/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0134 - acc: 0.9961 -
val_loss: 0.0582 - val_acc: 0.9878
```

```
Epoch 42/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0127 - acc: 0.9962 -
val_loss: 0.0393 - val_acc: 0.9916
Epoch 43/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0094 - acc: 0.9971 -
val_loss: 0.0486 - val_acc: 0.9902
Epoch 44/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0101 - acc: 0.9969 -
val_loss: 0.0346 - val_acc: 0.9932
Epoch 45/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0114 - acc: 0.9964 -
val_loss: 0.0420 - val_acc: 0.9917
Epoch 46/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0103 - acc: 0.9973 -
val_loss: 0.0472 - val_acc: 0.9903
Epoch 47/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0119 - acc: 0.9964 -
val_loss: 0.0448 - val_acc: 0.9917
Epoch 48/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0082 - acc: 0.9973 -
val_loss: 0.0425 - val_acc: 0.9916
Epoch 49/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0110 - acc: 0.9965 -
val_loss: 0.0474 - val_acc: 0.9909
Epoch 50/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0116 - acc: 0.9968 -
val_loss: 0.0448 - val_acc: 0.9916
Test loss: 0.04484636384752339
Test accuracy: 0.9916
Test score: 0.04484636384752339
Test accuracy: 0.9916
```



In [24]:

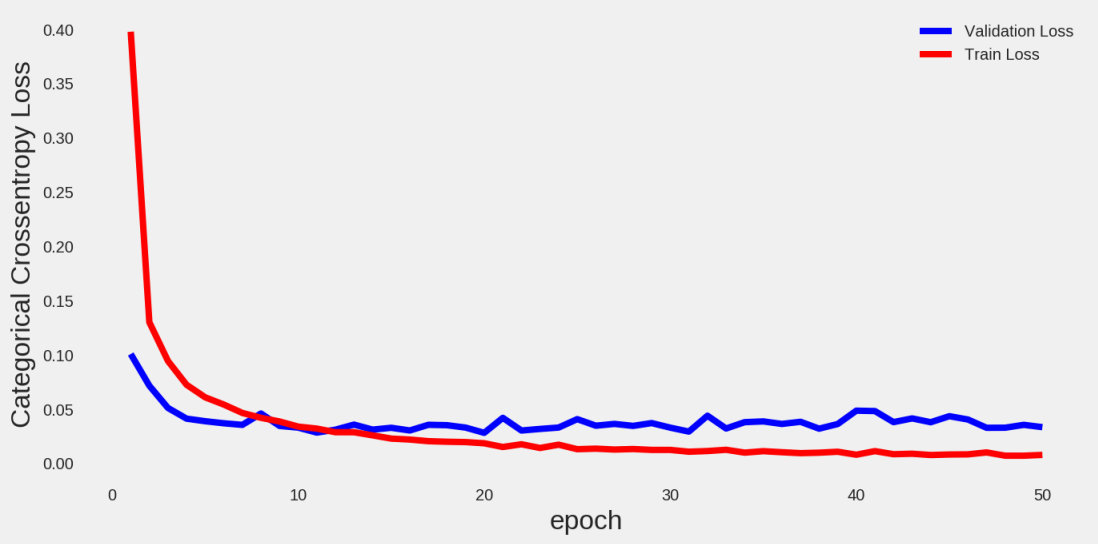
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=input_shape))
BatchNormalization(axis=1)
model.add(Conv2D(64, (4, 4), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
BatchNormalization(axis=1)
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
BatchNormalization(axis=1)
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
BatchNormalization(axis=1)
model.add(Dense(512, activation='relu'))
BatchNormalization(axis=1)
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam',loss=keras.losses.categorical_crossentropy,
```

```
              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])




score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, va
lidation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number of epochs

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

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 9s 152us/step - loss: 0.3979 - acc: 0.8737 -
val_loss: 0.1009 - val_acc: 0.9697
Epoch 2/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.1302 - acc: 0.9610 -
val_loss: 0.0717 - val_acc: 0.9776
Epoch 3/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0945 - acc: 0.9714 -
val_loss: 0.0512 - val_acc: 0.9845
Epoch 4/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0726 - acc: 0.9790 -
val_loss: 0.0414 - val_acc: 0.9872
Epoch 5/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0610 - acc: 0.9816 -
val_loss: 0.0390 - val_acc: 0.9885
Epoch 6/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0544 - acc: 0.9832 -
val_loss: 0.0371 - val_acc: 0.9900
Epoch 7/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0466 - acc: 0.9856 -
val_loss: 0.0356 - val_acc: 0.9886
Epoch 8/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0422 - acc: 0.9871 -
val_loss: 0.0461 - val_acc: 0.9864
Epoch 9/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0388 - acc: 0.9881 -
val_loss: 0.0347 - val_acc: 0.9897
Epoch 10/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0340 - acc: 0.9897 -
val_loss: 0.0331 - val_acc: 0.9913
Epoch 11/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0322 - acc: 0.9898 -
val_loss: 0.0285 - val_acc: 0.9913
```

```
Epoch 12/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0289 - acc: 0.9911 -
val_loss: 0.0312 - val_acc: 0.9915
Epoch 13/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0289 - acc: 0.9911 -
val_loss: 0.0359 - val_acc: 0.9897
Epoch 14/50
60000/60000 [==============================] - 8s 142us/step - loss: 0.0260 - acc: 0.9919 -
val_loss: 0.0313 - val_acc: 0.9918
Epoch 15/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0230 - acc: 0.9927 -
val_loss: 0.0330 - val_acc: 0.9908
Epoch 16/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0221 - acc: 0.9932 -
val_loss: 0.0305 - val_acc: 0.9920
Epoch 17/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.0206 - acc: 0.9939 -
val_loss: 0.0357 - val_acc: 0.9902
Epoch 18/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0201 - acc: 0.9934 -
val_loss: 0.0353 - val_acc: 0.9914
Epoch 19/50
60000/60000 [==============================] - 8s 140us/step - loss: 0.0198 - acc: 0.9939 -
val_loss: 0.0331 - val_acc: 0.9914
Epoch 20/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0187 - acc: 0.9943 -
val_loss: 0.0283 - val_acc: 0.9927
Epoch 21/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0153 - acc: 0.9952 -
val_loss: 0.0421 - val_acc: 0.9904
Epoch 22/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0178 - acc: 0.9944 -
val_loss: 0.0304 - val_acc: 0.9921
Epoch 23/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0144 - acc: 0.9952 -
val_loss: 0.0319 - val_acc: 0.9924
Epoch 24/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0174 - acc: 0.9950 -
val_loss: 0.0332 - val_acc: 0.9922
Epoch 25/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0132 - acc: 0.9960 -
val_loss: 0.0409 - val_acc: 0.9905
Epoch 26/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0137 - acc: 0.9956 -
val_loss: 0.0349 - val_acc: 0.9922
Epoch 27/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0129 - acc: 0.9960 -
val_loss: 0.0365 - val_acc: 0.9917
Epoch 28/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0134 - acc: 0.9958 -
val_loss: 0.0347 - val_acc: 0.9927
Epoch 29/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0126 - acc: 0.9963 -
val_loss: 0.0373 - val_acc: 0.9919
Epoch 30/50
60000/60000 [==============================] - 8s 142us/step - loss: 0.0126 - acc: 0.9959 -
val_loss: 0.0331 - val_acc: 0.9922
Epoch 31/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0109 - acc: 0.9966 -
val_loss: 0.0294 - val_acc: 0.9923
Epoch 32/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0115 - acc: 0.9964 -
val_loss: 0.0441 - val_acc: 0.9889
Epoch 33/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0127 - acc: 0.9959 -
val_loss: 0.0323 - val_acc: 0.9926
Epoch 34/50
60000/60000 [==============================] - 8s 142us/step - loss: 0.0101 - acc: 0.9969 -
val_loss: 0.0381 - val_acc: 0.9931
Epoch 35/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0114 - acc: 0.9964 -
val_loss: 0.0388 - val_acc: 0.9923
Epoch 36/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0104 - acc: 0.9969 -
val_loss: 0.0365 - val_acc: 0.9928
Epoch 37/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0096 - acc: 0.9970 -
```

```
val_loss: 0.0385 - val_acc: 0.9929
Epoch 38/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0100 - acc: 0.9970 -
val_loss: 0.0322 - val_acc: 0.9937
Epoch 39/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0109 - acc: 0.9966 -
val_loss: 0.0364 - val_acc: 0.9933
Epoch 40/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0081 - acc: 0.9974 -
val_loss: 0.0487 - val_acc: 0.9900
Epoch 41/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.0114 - acc: 0.9966 -
val_loss: 0.0484 - val_acc: 0.9909
Epoch 42/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0086 - acc: 0.9974 -
val_loss: 0.0382 - val_acc: 0.9916
Epoch 43/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0090 - acc: 0.9971 -
val_loss: 0.0416 - val_acc: 0.9915
Epoch 44/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0079 - acc: 0.9979 -
val_loss: 0.0381 - val_acc: 0.9918
Epoch 45/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0083 - acc: 0.9974 -
val_loss: 0.0436 - val_acc: 0.9914
Epoch 46/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0085 - acc: 0.9974 -
val_loss: 0.0406 - val_acc: 0.9929
Epoch 47/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0103 - acc: 0.9970 -
val_loss: 0.0330 - val_acc: 0.9933
Epoch 48/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0073 - acc: 0.9979 -
val_loss: 0.0330 - val_acc: 0.9936
Epoch 49/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0072 - acc: 0.9978 -
val_loss: 0.0357 - val_acc: 0.9924
Epoch 50/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0080 - acc: 0.9977 -
val_loss: 0.0335 - val_acc: 0.9935
Test loss: 0.03354739974554673
Test accuracy: 0.9935
Test score: 0.03354739974554673
Test accuracy: 0.9935
```



In [25]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["S.R",'Model',"Test loss","Accuracy",]

x.add_row([(1),"AdaDelta_Optimizer", 0.0614, '99.14%'])
x.add_row([(2),"Adam_optimizer",  0.036, '99.25%'])
```

```
x.add_row([(3),'deep network with adam', 0.032, '99.35%'])
x.add_row([(4),'Batch normalisation', 0.044, '99.16%'])
x.add_row([(5),'Replacing The dropout with Batch normalisation', 0.033, '99.35%'])


print(x.get_string(title = "-----SUMMARY-----"))
```

```
+-----+-------------------------------------------------+-----------+----------+
| S.R |                      Model                      | Test loss | Accuracy |
+-----+-------------------------------------------------+-----------+----------+
|  1  |               AdaDelta_Optimizer                |   0.0614  |  99.14%  |
|  2  |                 Adam_optimizer                  |   0.036   |  99.25%  |
|  3  |             deep network with adam              |   0.032   |  99.35%  |
|  4  |              Batch normalisation                |   0.044   |  99.16%  |
|  5  | Replacing The dropout with Batch normalisation  |   0.033   |  99.35%  |
+-----+-------------------------------------------------+-----------+----------+
```

Conclusion:

1. Adding different kernel sizes gave us better results
2. We did batch normalisation also.
3. **Removing the dropout layer gave us preety good result. It is almost as good as my best model.**