Various MLP architectures on MNIST Data set

In [0]:

```
#importing required libraires
from keras.utils import np_utils
from keras.datasets import mnist
import seaborn as sns
from keras.initializers import RandomNormal
from keras.models import Sequential
from keras.layers import Dense, Activation ,Dropout
%matplotlib inline
```

In [3]:

```
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import time
from keras.datasets import mnist
# 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()
```

```
Using TensorFlow backend.
```

In [0]:

```
# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

In [0]:

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1]*X_train.shape[2])
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1]*X_test.shape[2])
X_train = X_train/255
X_test = X_test/255
output_dim = 10
input_dim = X_train.shape[1]
```

In [0]:

```
batch_size = 128
nb_epoch = 50
```

In [0]:

```
Y_train = np_utils.to_categorical(y_train, 10)
Y_test = np_utils.to_categorical(y_test, 10)
```

Model1: Layer1: 512 neurons with acivation as sigmoid + Layer2: 128 neurons with acivation as sigmoid + Layer3: 64 neurons with acivation as sigmoid + sofmax layer

In [26]:

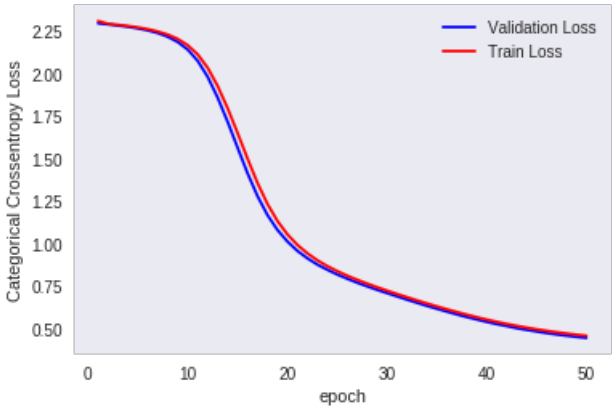
```
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='sigmoid',input_shape=(input_dim,)))

model.add(Dense(128,activation='sigmoid'))
```

```
model.add(Dense(64,activation='sigmoid'))

model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 6s 97us/step - loss: 2.3082 - acc: 0.1101 -
val_loss: 2.2944 - val_acc: 0.1139
Epoch 2/50
60000/60000 [==============================] - 5s 91us/step - loss: 2.2919 - acc: 0.1252 -
val_loss: 2.2887 - val_acc: 0.1909
Epoch 3/50
60000/60000 [==============================] - 5s 90us/step - loss: 2.2859 - acc: 0.1311 -
val_loss: 2.2817 - val_acc: 0.1175
Epoch 4/50
60000/60000 [==============================] - 5s 92us/step - loss: 2.2789 - acc: 0.1414 -
val_loss: 2.2742 - val_acc: 0.1816
Epoch 5/50
60000/60000 [==============================] - 5s 90us/step - loss: 2.2705 - acc: 0.1823 -
val_loss: 2.2650 - val_acc: 0.2733
Epoch 6/50
60000/60000 [==============================] - 5s 91us/step - loss: 2.2600 - acc: 0.2404 -
val_loss: 2.2530 - val_acc: 0.1944
Epoch 7/50
60000/60000 [==============================] - 5s 91us/step - loss: 2.2464 - acc: 0.2741 -
val_loss: 2.2373 - val_acc: 0.3598
Epoch 8/50
60000/60000 [==============================] - 5s 90us/step - loss: 2.2280 - acc: 0.3314 -
val_loss: 2.2153 - val_acc: 0.3892
Epoch 9/50
60000/60000 [==============================] - 5s 91us/step - loss: 2.2024 - acc: 0.3847 -
val_loss: 2.1844 - val_acc: 0.3898
Epoch 10/50
60000/60000 [==============================] - 5s 91us/step - loss: 2.1655 - acc: 0.4211 -
val_loss: 2.1397 - val_acc: 0.4897
Epoch 11/50
60000/60000 [==============================] - 5s 90us/step - loss: 2.1112 - acc: 0.4611 -
val_loss: 2.0732 - val_acc: 0.4738
Epoch 12/50
60000/60000 [==============================] - 5s 90us/step - loss: 2.0322 - acc: 0.4881 -
val_loss: 1.9789 - val_acc: 0.5193
Epoch 13/50
60000/60000 [==============================] - 5s 90us/step - loss: 1.9250 - acc: 0.5191 -
val_loss: 1.8566 - val_acc: 0.5271
Epoch 14/50
60000/60000 [==============================] - 5s 90us/step - loss: 1.7942 - acc: 0.5486 -
val_loss: 1.7145 - val_acc: 0.5581
Epoch 15/50
60000/60000 [==============================] - 6s 95us/step - loss: 1.6481 - acc: 0.5863 -
val_loss: 1.5619 - val_acc: 0.6226
Epoch 16/50
60000/60000 [==============================] - 5s 88us/step - loss: 1.4970 - acc: 0.6176 -
val_loss: 1.4109 - val_acc: 0.6471
Epoch 17/50
```

```
60000/60000 [==============================] - 5s 86us/step - loss: 1.3534 - acc: 0.6431 -
val_loss: 1.2764 - val_acc: 0.6569
Epoch 18/50
60000/60000 [==============================] - 5s 85us/step - loss: 1.2298 - acc: 0.6626 -
val_loss: 1.1647 - val_acc: 0.6800
Epoch 19/50
60000/60000 [==============================] - 5s 86us/step - loss: 1.1297 - acc: 0.6800 -
val_loss: 1.0772 - val_acc: 0.6910
Epoch 20/50
60000/60000 [==============================] - 5s 85us/step - loss: 1.0511 - acc: 0.6952 -
val_loss: 1.0090 - val_acc: 0.7092
Epoch 21/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.9891 - acc: 0.7096 -
val_loss: 0.9539 - val_acc: 0.7179
Epoch 22/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.9393 - acc: 0.7198 -
val_loss: 0.9103 - val_acc: 0.7256
Epoch 23/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.8986 - acc: 0.7298 -
val_loss: 0.8732 - val_acc: 0.7401
Epoch 24/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.8641 - acc: 0.7408 -
val_loss: 0.8427 - val_acc: 0.7468
Epoch 25/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.8343 - acc: 0.7504 -
val_loss: 0.8147 - val_acc: 0.7578
Epoch 26/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.8076 - acc: 0.7591 -
val_loss: 0.7899 - val_acc: 0.7650
Epoch 27/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.7835 - acc: 0.7672 -
val_loss: 0.7664 - val_acc: 0.7744
Epoch 28/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.7610 - acc: 0.7749 -
val_loss: 0.7449 - val_acc: 0.7825
Epoch 29/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.7397 - acc: 0.7825 -
val_loss: 0.7242 - val_acc: 0.7907
Epoch 30/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.7195 - acc: 0.7898 -
val_loss: 0.7051 - val_acc: 0.7956
Epoch 31/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.6998 - acc: 0.7982 -
val_loss: 0.6857 - val_acc: 0.8024
Epoch 32/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.6808 - acc: 0.8046 -
val_loss: 0.6666 - val_acc: 0.8102
Epoch 33/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.6622 - acc: 0.8113 -
val_loss: 0.6481 - val_acc: 0.8141
Epoch 34/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.6442 - acc: 0.8173 -
val_loss: 0.6300 - val_acc: 0.8223
Epoch 35/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.6267 - acc: 0.8232 -
val_loss: 0.6127 - val_acc: 0.8277
Epoch 36/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.6098 - acc: 0.8287 -
val_loss: 0.5960 - val_acc: 0.8343
Epoch 37/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.5935 - acc: 0.8336 -
val_loss: 0.5796 - val_acc: 0.8366
Epoch 38/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.5780 - acc: 0.8386 -
val_loss: 0.5640 - val_acc: 0.8418
Epoch 39/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.5632 - acc: 0.8430 -
val_loss: 0.5497 - val_acc: 0.8469
Epoch 40/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.5492 - acc: 0.8469 -
val_loss: 0.5354 - val_acc: 0.8497
Epoch 41/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.5359 - acc: 0.8511 -
val_loss: 0.5227 - val_acc: 0.8532
Epoch 42/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.5236 - acc: 0.8546 -
val_loss: 0.5110 - val_acc: 0.8567
```

```
Epoch 43/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.5122 - acc: 0.8572 -
val_loss: 0.4984 - val_acc: 0.8608
Epoch 44/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.5017 - acc: 0.8606 -
val_loss: 0.4883 - val_acc: 0.8632
Epoch 45/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.4918 - acc: 0.8628 -
val_loss: 0.4792 - val_acc: 0.8645
Epoch 46/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.4829 - acc: 0.8647 -
val_loss: 0.4695 - val_acc: 0.8672
Epoch 47/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.4747 - acc: 0.8672 -
val_loss: 0.4615 - val_acc: 0.8702
Epoch 48/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.4671 - acc: 0.8691 -
val_loss: 0.4541 - val_acc: 0.8730
Epoch 49/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.4601 - acc: 0.8710 -
val_loss: 0.4471 - val_acc: 0.8744
Epoch 50/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.4537 - acc: 0.8723 -
val_loss: 0.4411 - val_acc: 0.8746
Test score: 0.4410694149255753
Test accuracy: 0.8746
```



We are getting train accuracy of 87.23% after 50 epochs and test accuracy 87.46%.Lets try relu activation units

Model2: Layer1: 512 neurons with acivation as relu + Layer2: 128 neurons with acivation as relu + Layer3: 64 neurons with acivation as relu + sofmax layer

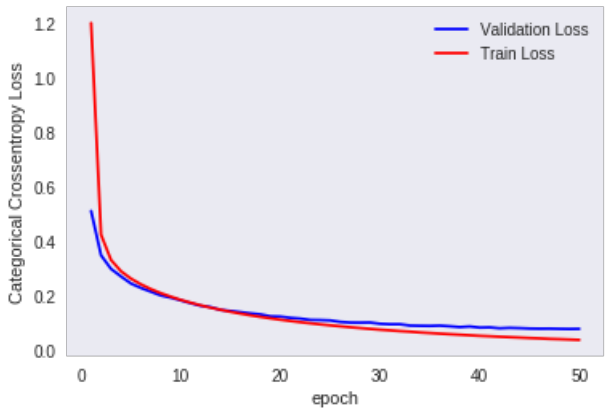
In [27]:

```
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(input_dim,)))

model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))

model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 6s 99us/step - loss: 1.2023 - acc: 0.7060 -
val_loss: 0.5115 - val_acc: 0.8722
Epoch 2/50
60000/60000 [==============================] - 5s 91us/step - loss: 0.4242 - acc: 0.8872 -
val_loss: 0.3483 - val_acc: 0.9038
Epoch 3/50
60000/60000 [==============================] - 5s 90us/step - loss: 0.3313 - acc: 0.9072 -
val_loss: 0.2985 - val_acc: 0.9190
Epoch 4/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.2898 - acc: 0.9180 -
val_loss: 0.2708 - val_acc: 0.9220
Epoch 5/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.2628 - acc: 0.9254 -
val_loss: 0.2447 - val_acc: 0.9291
Epoch 6/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.2418 - acc: 0.9305 -
val_loss: 0.2284 - val_acc: 0.9341
Epoch 7/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.2242 - acc: 0.9360 -
val_loss: 0.2147 - val_acc: 0.9389
Epoch 8/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.2092 - acc: 0.9398 -
val_loss: 0.2008 - val_acc: 0.9411
Epoch 9/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.1964 - acc: 0.9442 -
val_loss: 0.1927 - val_acc: 0.9430
Epoch 10/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.1845 - acc: 0.9469 -
val_loss: 0.1829 - val_acc: 0.9468
Epoch 11/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.1741 - acc: 0.9505 -
val_loss: 0.1725 - val_acc: 0.9477
Epoch 12/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.1646 - acc: 0.9533 -
val_loss: 0.1630 - val_acc: 0.9522
Epoch 13/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.1558 - acc: 0.9559 -
val_loss: 0.1579 - val_acc: 0.9533
Epoch 14/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.1482 - acc: 0.9576 -
val_loss: 0.1477 - val_acc: 0.9565
Epoch 15/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.1407 - acc: 0.9602 -
val_loss: 0.1437 - val_acc: 0.9573
Epoch 16/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.1341 - acc: 0.9617 -
val_loss: 0.1395 - val_acc: 0.9591
Epoch 17/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.1277 - acc: 0.9639 -
val_loss: 0.1350 - val_acc: 0.9594
Epoch 18/50
60000/60000 [==============================] - 6s 93us/step - loss: 0.1220 - acc: 0.9659 -
val_loss: 0.1313 - val_acc: 0.9600
Epoch 19/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.1168 - acc: 0.9670 -
val_loss: 0.1242 - val_acc: 0.9626
Epoch 20/50
60000/60000 [==============================] - 5s 89us/step - loss: 0.1116 - acc: 0.9686 -
val_loss: 0.1231 - val_acc: 0.9627
Epoch 21/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.1071 - acc: 0.9698 -
val_loss: 0.1186 - val_acc: 0.9656
Epoch 22/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.1025 - acc: 0.9713 -
val_loss: 0.1157 - val_acc: 0.9669
Epoch 23/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0985 - acc: 0.9721 -
val_loss: 0.1110 - val_acc: 0.9665
Epoch 24/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0946 - acc: 0.9733 -
val_loss: 0.1104 - val_acc: 0.9673
Epoch 25/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0910 - acc: 0.9742 -
```

```
val_loss: 0.1092 - val_acc: 0.9674
Epoch 26/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0875 - acc: 0.9754 -
val_loss: 0.1037 - val_acc: 0.9702
Epoch 27/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0840 - acc: 0.9760 -
val_loss: 0.1012 - val_acc: 0.9705
Epoch 28/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0810 - acc: 0.9769 -
val_loss: 0.1009 - val_acc: 0.9690
Epoch 29/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0779 - acc: 0.9784 -
val_loss: 0.1014 - val_acc: 0.9697
Epoch 30/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0750 - acc: 0.9794 -
val_loss: 0.0967 - val_acc: 0.9718
Epoch 31/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0725 - acc: 0.9800 -
val_loss: 0.0951 - val_acc: 0.9723
Epoch 32/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0698 - acc: 0.9811 -
val_loss: 0.0951 - val_acc: 0.9714
Epoch 33/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0677 - acc: 0.9815 -
val_loss: 0.0902 - val_acc: 0.9728
Epoch 34/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0648 - acc: 0.9822 -
val_loss: 0.0894 - val_acc: 0.9734
Epoch 35/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0628 - acc: 0.9825 -
val_loss: 0.0890 - val_acc: 0.9734
Epoch 36/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0605 - acc: 0.9833 -
val_loss: 0.0901 - val_acc: 0.9732
Epoch 37/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0584 - acc: 0.9843 -
val_loss: 0.0876 - val_acc: 0.9737
Epoch 38/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0565 - acc: 0.9845 -
val_loss: 0.0845 - val_acc: 0.9745
Epoch 39/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0545 - acc: 0.9851 -
val_loss: 0.0869 - val_acc: 0.9743
Epoch 40/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0524 - acc: 0.9861 -
val_loss: 0.0828 - val_acc: 0.9749
Epoch 41/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0509 - acc: 0.9865 -
val_loss: 0.0835 - val_acc: 0.9756
Epoch 42/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0491 - acc: 0.9871 -
val_loss: 0.0803 - val_acc: 0.9751
Epoch 43/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0476 - acc: 0.9875 -
val_loss: 0.0818 - val_acc: 0.9746
Epoch 44/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0460 - acc: 0.9879 -
val_loss: 0.0808 - val_acc: 0.9750
Epoch 45/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0444 - acc: 0.9883 -
val_loss: 0.0793 - val_acc: 0.9767
Epoch 46/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0430 - acc: 0.9890 -
val_loss: 0.0788 - val_acc: 0.9758
Epoch 47/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0412 - acc: 0.9897 -
val_loss: 0.0790 - val_acc: 0.9763
Epoch 48/50
60000/60000 [==============================] - 5s 87us/step - loss: 0.0401 - acc: 0.9897 -
val_loss: 0.0783 - val_acc: 0.9762
Epoch 49/50
60000/60000 [==============================] - 5s 88us/step - loss: 0.0388 - acc: 0.9904 -
val_loss: 0.0777 - val_acc: 0.9769
Epoch 50/50
60000/60000 [==============================] - 5s 86us/step - loss: 0.0374 - acc: 0.9907 -
val_loss: 0.0782 - val_acc: 0.9762
Test score: 0.07823057521777227
```

Test accuracy: 0.9762



We are getting train accuracy of 99.07% after 50 epochs and test accuracy 97.62%.As there is some diffrence in test and rain accuracy .Lets try relu activation units + dropouts

Model3: Layer1: 512 neurons with acivation as relu + Dropout (0.25)+Layer2: 128 neurons with acivation as relu + Layer3: 64 neurons with acivation as relu+ Dropout (0.25) + sofmax layer

In [0]:

```
```

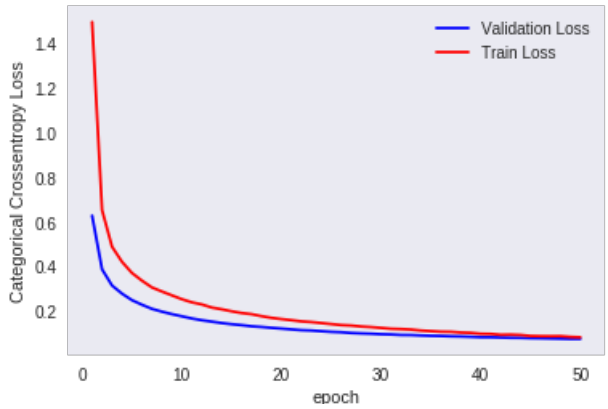
In [28]:

```
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(input_dim,)))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 6s 104us/step - loss: 1.4936 - acc: 0.5363 -
val_loss: 0.6266 - val_acc: 0.8494
Epoch 2/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.6514 - acc: 0.8029 -
val_loss: 0.3862 - val_acc: 0.8974
Epoch 3/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.4874 - acc: 0.8552 -
val_loss: 0.3137 - val_acc: 0.9127
Epoch 4/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.4198 - acc: 0.8760 -
val_loss: 0.2774 - val_acc: 0.9223
Epoch 5/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.3699 - acc: 0.8928 -
val_loss: 0.2482 - val_acc: 0.9283
```

```
Epoch 6/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.3355 - acc: 0.9016 -
val_loss: 0.2277 - val_acc: 0.9339
Epoch 7/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.3052 - acc: 0.9114 -
val_loss: 0.2092 - val_acc: 0.9391
Epoch 8/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.2870 - acc: 0.9168 -
val_loss: 0.1968 - val_acc: 0.9419
Epoch 9/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.2697 - acc: 0.9207 -
val_loss: 0.1856 - val_acc: 0.9454
Epoch 10/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.2526 - acc: 0.9258 -
val_loss: 0.1760 - val_acc: 0.9479
Epoch 11/50
60000/60000 [==============================] - 6s 102us/step - loss: 0.2385 - acc: 0.9310 -
val_loss: 0.1662 - val_acc: 0.9500
Epoch 12/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.2291 - acc: 0.9344 -
val_loss: 0.1583 - val_acc: 0.9526
Epoch 13/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.2152 - acc: 0.9371 -
val_loss: 0.1520 - val_acc: 0.9528
Epoch 14/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.2073 - acc: 0.9399 -
val_loss: 0.1457 - val_acc: 0.9549
Epoch 15/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1983 - acc: 0.9425 -
val_loss: 0.1405 - val_acc: 0.9565
Epoch 16/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1906 - acc: 0.9452 -
val_loss: 0.1363 - val_acc: 0.9575
Epoch 17/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1854 - acc: 0.9468 -
val_loss: 0.1311 - val_acc: 0.9593
Epoch 18/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1765 - acc: 0.9483 -
val_loss: 0.1281 - val_acc: 0.9605
Epoch 19/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.1689 - acc: 0.9512 -
val_loss: 0.1242 - val_acc: 0.9609
Epoch 20/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.1634 - acc: 0.9513 -
val_loss: 0.1209 - val_acc: 0.9614
Epoch 21/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.1585 - acc: 0.9537 -
val_loss: 0.1172 - val_acc: 0.9631
Epoch 22/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.1530 - acc: 0.9553 -
val_loss: 0.1137 - val_acc: 0.9645
Epoch 23/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.1500 - acc: 0.9573 -
val_loss: 0.1122 - val_acc: 0.9641
Epoch 24/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1456 - acc: 0.9583 -
val_loss: 0.1091 - val_acc: 0.9655
Epoch 25/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1410 - acc: 0.9588 -
val_loss: 0.1062 - val_acc: 0.9660
Epoch 26/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1366 - acc: 0.9605 -
val_loss: 0.1045 - val_acc: 0.9671
Epoch 27/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1345 - acc: 0.9610 -
val_loss: 0.1011 - val_acc: 0.9667
Epoch 28/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1302 - acc: 0.9622 -
val_loss: 0.0993 - val_acc: 0.9675
Epoch 29/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1276 - acc: 0.9627 -
val_loss: 0.0979 - val_acc: 0.9681
Epoch 30/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1240 - acc: 0.9642 -
val_loss: 0.0956 - val_acc: 0.9692
Epoch 31/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1201 - acc: 0.9649 -
```

```
val_loss: 0.0945 - val_acc: 0.9700
Epoch 32/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.1187 - acc: 0.9649 -
val_loss: 0.0922 - val_acc: 0.9703
Epoch 33/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1165 - acc: 0.9665 -
val_loss: 0.0919 - val_acc: 0.9712
Epoch 34/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1121 - acc: 0.9669 -
val_loss: 0.0900 - val_acc: 0.9711
Epoch 35/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1094 - acc: 0.9682 -
val_loss: 0.0886 - val_acc: 0.9708
Epoch 36/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.1069 - acc: 0.9693 -
val_loss: 0.0878 - val_acc: 0.9717
Epoch 37/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.1064 - acc: 0.9686 -
val_loss: 0.0862 - val_acc: 0.9722
Epoch 38/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.1032 - acc: 0.9699 -
val_loss: 0.0855 - val_acc: 0.9724
Epoch 39/50
60000/60000 [==============================] - 6s 97us/step - loss: 0.1018 - acc: 0.9705 -
val_loss: 0.0840 - val_acc: 0.9724
Epoch 40/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0975 - acc: 0.9716 -
val_loss: 0.0822 - val_acc: 0.9736
Epoch 41/50
60000/60000 [==============================] - 6s 96us/step - loss: 0.0969 - acc: 0.9718 -
val_loss: 0.0821 - val_acc: 0.9736
Epoch 42/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0932 - acc: 0.9722 -
val_loss: 0.0808 - val_acc: 0.9739
Epoch 43/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0937 - acc: 0.9726 -
val_loss: 0.0797 - val_acc: 0.9742
Epoch 44/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0922 - acc: 0.9730 -
val_loss: 0.0793 - val_acc: 0.9742
Epoch 45/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0878 - acc: 0.9746 -
val_loss: 0.0780 - val_acc: 0.9751
Epoch 46/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0869 - acc: 0.9749 -
val_loss: 0.0774 - val_acc: 0.9749
Epoch 47/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0866 - acc: 0.9746 -
val_loss: 0.0765 - val_acc: 0.9752
Epoch 48/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0869 - acc: 0.9741 -
val_loss: 0.0756 - val_acc: 0.9756
Epoch 49/50
60000/60000 [==============================] - 6s 94us/step - loss: 0.0836 - acc: 0.9754 -
val_loss: 0.0746 - val_acc: 0.9759
Epoch 50/50
60000/60000 [==============================] - 6s 95us/step - loss: 0.0819 - acc: 0.9755 -
val_loss: 0.0748 - val_acc: 0.9758
Test score: 0.07479424988320098
Test accuracy: 0.9758
```



In [0]:

```
```

we can there is less difference in train and test accuracy using dropouts which is generally used to avoid over fitting.We are getting train accuracy of 99.07% after 50 epochs and test accuracy 97.62%. .Lets try relu activation units + dropouts+Batch Normalization

Model4: Layer1: 512 neurons with acivation as relu + Dropout (0.25)+Layer2: 128 neurons with acivation as relu+ Batch normalization + Layer3: 64 neurons with acivation as relu+ Dropout (0.25) + sofmax layer

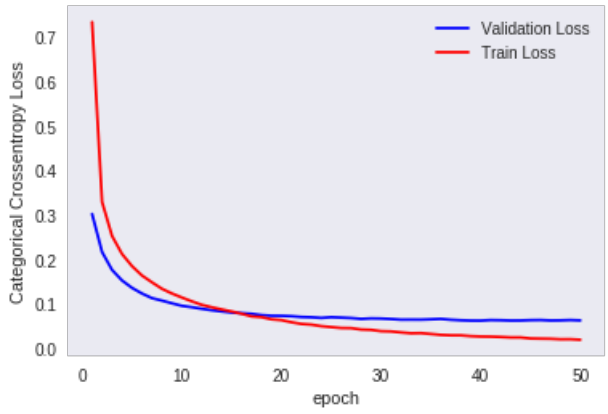
In [30]:

```
from keras.layers.normalization import BatchNormalization
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(input_dim,)))
model.add(Dropout(0.25))

model.add(Dense(128,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(64,activation='relu'))

model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 7s 114us/step - loss: 0.7333 - acc: 0.7824 -
val_loss: 0.3020 - val_acc: 0.9134
Epoch 2/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.3293 - acc: 0.9043 -
val_loss: 0.2166 - val_acc: 0.9354
Epoch 3/50
60000/60000 [==============================] - 6s 102us/step - loss: 0.2527 - acc: 0.9258 -
val_loss: 0.1764 - val_acc: 0.9475
Epoch 4/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.2123 - acc: 0.9380 -
val_loss: 0.1527 - val_acc: 0.9530
Epoch 5/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.1850 - acc: 0.9460 -
val_loss: 0.1361 - val_acc: 0.9600
Epoch 6/50
60000/60000 [==============================] - 6s 98us/step - loss: 0.1639 - acc: 0.9513 -
val_loss: 0.1234 - val_acc: 0.9619
Epoch 7/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.1486 - acc: 0.9559 -
val_loss: 0.1130 - val_acc: 0.9654
Epoch 8/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.1340 - acc: 0.9607 -
val_loss: 0.1074 - val_acc: 0.9671
Epoch 9/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.1235 - acc: 0.9630 -
val_loss: 0.1014 - val_acc: 0.9679
Epoch 10/50
60000/60000 [==============================] - 6s 102us/step - loss: 0.1144 - acc: 0.9655 -
val_loss: 0.0957 - val_acc: 0.9700
Epoch 11/50
```

```
60000/60000 [==============================] - 6s 102us/step - loss: 0.1058 - acc: 0.9687 -
val_loss: 0.0926 - val_acc: 0.9716
Epoch 12/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0979 - acc: 0.9705 -
val_loss: 0.0890 - val_acc: 0.9731
Epoch 13/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0925 - acc: 0.9720 -
val_loss: 0.0858 - val_acc: 0.9736
Epoch 14/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0875 - acc: 0.9730 -
val_loss: 0.0831 - val_acc: 0.9735
Epoch 15/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0829 - acc: 0.9747 -
val_loss: 0.0806 - val_acc: 0.9753
Epoch 16/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0778 - acc: 0.9762 -
val_loss: 0.0791 - val_acc: 0.9757
Epoch 17/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0719 - acc: 0.9778 -
val_loss: 0.0777 - val_acc: 0.9761
Epoch 18/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0702 - acc: 0.9788 -
val_loss: 0.0748 - val_acc: 0.9772
Epoch 19/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0654 - acc: 0.9802 -
val_loss: 0.0734 - val_acc: 0.9777
Epoch 20/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0635 - acc: 0.9804 -
val_loss: 0.0732 - val_acc: 0.9777
Epoch 21/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0587 - acc: 0.9815 -
val_loss: 0.0724 - val_acc: 0.9778
Epoch 22/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0550 - acc: 0.9830 -
val_loss: 0.0706 - val_acc: 0.9790
Epoch 23/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0534 - acc: 0.9833 -
val_loss: 0.0699 - val_acc: 0.9795
Epoch 24/50
60000/60000 [==============================] - 6s 102us/step - loss: 0.0499 - acc: 0.9849 -
val_loss: 0.0685 - val_acc: 0.9783
Epoch 25/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0478 - acc: 0.9850 -
val_loss: 0.0701 - val_acc: 0.9782
Epoch 26/50
60000/60000 [==============================] - 7s 109us/step - loss: 0.0458 - acc: 0.9863 -
val_loss: 0.0692 - val_acc: 0.9796
Epoch 27/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0455 - acc: 0.9857 -
val_loss: 0.0682 - val_acc: 0.9795
Epoch 28/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0424 - acc: 0.9867 -
val_loss: 0.0663 - val_acc: 0.9799
Epoch 29/50
60000/60000 [==============================] - 6s 99us/step - loss: 0.0416 - acc: 0.9872 -
val_loss: 0.0674 - val_acc: 0.9789
Epoch 30/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0387 - acc: 0.9884 -
val_loss: 0.0671 - val_acc: 0.9795
Epoch 31/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0381 - acc: 0.9885 -
val_loss: 0.0660 - val_acc: 0.9800
Epoch 32/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0359 - acc: 0.9886 -
val_loss: 0.0646 - val_acc: 0.9799
Epoch 33/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0339 - acc: 0.9899 -
val_loss: 0.0647 - val_acc: 0.9801
Epoch 34/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0343 - acc: 0.9894 -
val_loss: 0.0646 - val_acc: 0.9797
Epoch 35/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.0324 - acc: 0.9897 -
val_loss: 0.0653 - val_acc: 0.9801
Epoch 36/50
60000/60000 [==============================] - 7s 112us/step - loss: 0.0303 - acc: 0.9905 -
val_loss: 0.0663 - val_acc: 0.9798
```

```
Epoch 37/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.0294 - acc: 0.9909 -
val_loss: 0.0644 - val_acc: 0.9807
Epoch 38/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0293 - acc: 0.9906 -
val_loss: 0.0632 - val_acc: 0.9807
Epoch 39/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0275 - acc: 0.9918 -
val_loss: 0.0625 - val_acc: 0.9816
Epoch 40/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0267 - acc: 0.9919 -
val_loss: 0.0624 - val_acc: 0.9813
Epoch 41/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0264 - acc: 0.9920 -
val_loss: 0.0636 - val_acc: 0.9806
Epoch 42/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0257 - acc: 0.9919 -
val_loss: 0.0633 - val_acc: 0.9818
Epoch 43/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0246 - acc: 0.9927 -
val_loss: 0.0626 - val_acc: 0.9809
Epoch 44/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0246 - acc: 0.9922 -
val_loss: 0.0627 - val_acc: 0.9805
Epoch 45/50
60000/60000 [==============================] - 6s 102us/step - loss: 0.0222 - acc: 0.9928 -
val_loss: 0.0635 - val_acc: 0.9810
Epoch 46/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0219 - acc: 0.9931 -
val_loss: 0.0637 - val_acc: 0.9819
Epoch 47/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0216 - acc: 0.9935 -
val_loss: 0.0628 - val_acc: 0.9810
Epoch 48/50
60000/60000 [==============================] - 6s 101us/step - loss: 0.0205 - acc: 0.9935 -
val_loss: 0.0630 - val_acc: 0.9812
Epoch 49/50
60000/60000 [==============================] - 6s 100us/step - loss: 0.0206 - acc: 0.9940 -
val_loss: 0.0637 - val_acc: 0.9807
Epoch 50/50
60000/60000 [==============================] - 6s 103us/step - loss: 0.0194 - acc: 0.9939 -
val_loss: 0.0629 - val_acc: 0.9817
Test score: 0.06288771041215223
Test accuracy: 0.9817
```



we are getting train accuracy 99.39 and test accuracy 98.17

Model5: Layer1: 512 neurons with acivation as relu + Dropout (0.25)+Layer2: 128 neurons with acivation as relu+ Dropout (0.5)+ Batch normalization + Layer3: 64 neurons with acivation as relu + sofmax layer

In [31]:

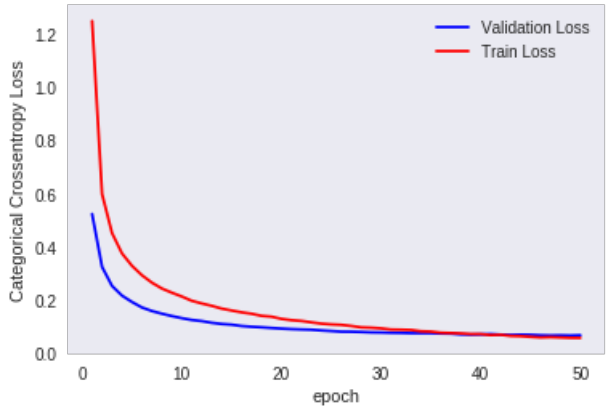
```
from keras.layers.normalization import BatchNormalization
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(input_dim,)))
model.add(Dropout(0.25))

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

```
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(64,activation='relu'))

model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 8s 126us/step - loss: 1.2475 - acc: 0.6113 -
val_loss: 0.5233 - val_acc: 0.8738
Epoch 2/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.5982 - acc: 0.8293 -
val_loss: 0.3246 - val_acc: 0.9119
Epoch 3/50
60000/60000 [==============================] - 7s 109us/step - loss: 0.4517 - acc: 0.8688 -
val_loss: 0.2532 - val_acc: 0.9283
Epoch 4/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.3752 - acc: 0.8907 -
val_loss: 0.2169 - val_acc: 0.9385
Epoch 5/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.3288 - acc: 0.9049 -
val_loss: 0.1931 - val_acc: 0.9445
Epoch 6/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.2934 - acc: 0.9150 -
val_loss: 0.1723 - val_acc: 0.9490
Epoch 7/50
60000/60000 [==============================] - 6s 104us/step - loss: 0.2655 - acc: 0.9244 -
val_loss: 0.1591 - val_acc: 0.9527
Epoch 8/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.2435 - acc: 0.9291 -
val_loss: 0.1490 - val_acc: 0.9549
Epoch 9/50
60000/60000 [==============================] - 7s 111us/step - loss: 0.2279 - acc: 0.9342 -
val_loss: 0.1402 - val_acc: 0.9570
Epoch 10/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.2135 - acc: 0.9380 -
val_loss: 0.1322 - val_acc: 0.9595
Epoch 11/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.1975 - acc: 0.9433 -
val_loss: 0.1255 - val_acc: 0.9613
Epoch 12/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.1874 - acc: 0.9456 -
val_loss: 0.1213 - val_acc: 0.9622
Epoch 13/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.1786 - acc: 0.9477 -
val_loss: 0.1149 - val_acc: 0.9652
Epoch 14/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.1681 - acc: 0.9502 -
val_loss: 0.1104 - val_acc: 0.9664
Epoch 15/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.1610 - acc: 0.9528 -
val_loss: 0.1079 - val_acc: 0.9673
Epoch 16/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.1541 - acc: 0.9542 -
val_loss: 0.1028 - val_acc: 0.9688
Epoch 17/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.1488 - acc: 0.9563 -
val_loss: 0.1000 - val_acc: 0.9693
Epoch 18/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.1410 - acc: 0.9587 -
```

```
val_loss: 0.0979 - val_acc: 0.9703
Epoch 19/50
60000/60000 [==============================] - 7s 112us/step - loss: 0.1378 - acc: 0.9587 -
val_loss: 0.0953 - val_acc: 0.9699
Epoch 20/50
60000/60000 [==============================] - 7s 113us/step - loss: 0.1296 - acc: 0.9626 -
val_loss: 0.0931 - val_acc: 0.9719
Epoch 21/50
60000/60000 [==============================] - 7s 109us/step - loss: 0.1247 - acc: 0.9633 -
val_loss: 0.0912 - val_acc: 0.9724
Epoch 22/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.1218 - acc: 0.9633 -
val_loss: 0.0899 - val_acc: 0.9723
Epoch 23/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.1166 - acc: 0.9654 -
val_loss: 0.0889 - val_acc: 0.9731
Epoch 24/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.1117 - acc: 0.9665 -
val_loss: 0.0862 - val_acc: 0.9735
Epoch 25/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.1089 - acc: 0.9677 -
val_loss: 0.0839 - val_acc: 0.9741
Epoch 26/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.1072 - acc: 0.9678 -
val_loss: 0.0816 - val_acc: 0.9752
Epoch 27/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.1030 - acc: 0.9689 -
val_loss: 0.0813 - val_acc: 0.9752
Epoch 28/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0978 - acc: 0.9707 -
val_loss: 0.0803 - val_acc: 0.9753
Epoch 29/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.0964 - acc: 0.9709 -
val_loss: 0.0788 - val_acc: 0.9756
Epoch 30/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0939 - acc: 0.9710 -
val_loss: 0.0784 - val_acc: 0.9763
Epoch 31/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0896 - acc: 0.9733 -
val_loss: 0.0773 - val_acc: 0.9766
Epoch 32/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0890 - acc: 0.9730 -
val_loss: 0.0769 - val_acc: 0.9769
Epoch 33/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0879 - acc: 0.9737 -
val_loss: 0.0759 - val_acc: 0.9779
Epoch 34/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.0831 - acc: 0.9743 -
val_loss: 0.0758 - val_acc: 0.9775
Epoch 35/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0819 - acc: 0.9748 -
val_loss: 0.0758 - val_acc: 0.9769
Epoch 36/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0775 - acc: 0.9760 -
val_loss: 0.0747 - val_acc: 0.9787
Epoch 37/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0764 - acc: 0.9768 -
val_loss: 0.0742 - val_acc: 0.9778
Epoch 38/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.0750 - acc: 0.9773 -
val_loss: 0.0713 - val_acc: 0.9788
Epoch 39/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.0727 - acc: 0.9778 -
val_loss: 0.0708 - val_acc: 0.9782
Epoch 40/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0725 - acc: 0.9773 -
val_loss: 0.0715 - val_acc: 0.9786
Epoch 41/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0688 - acc: 0.9793 -
val_loss: 0.0725 - val_acc: 0.9790
Epoch 42/50
60000/60000 [==============================] - 6s 105us/step - loss: 0.0700 - acc: 0.9781 -
val_loss: 0.0706 - val_acc: 0.9792
Epoch 43/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0656 - acc: 0.9800 -
val_loss: 0.0693 - val_acc: 0.9799
Epoch 44/50
```

```
60000/60000 [==============================] - 6s 106us/step - loss: 0.0646 - acc: 0.9801 -
val_loss: 0.0702 - val_acc: 0.9796
Epoch 45/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.0621 - acc: 0.9812 -
val_loss: 0.0698 - val_acc: 0.9796
Epoch 46/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0599 - acc: 0.9814 -
val_loss: 0.0680 - val_acc: 0.9804
Epoch 47/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0610 - acc: 0.9810 -
val_loss: 0.0673 - val_acc: 0.9801
Epoch 48/50
60000/60000 [==============================] - 6s 107us/step - loss: 0.0597 - acc: 0.9816 -
val_loss: 0.0679 - val_acc: 0.9810
Epoch 49/50
60000/60000 [==============================] - 6s 106us/step - loss: 0.0584 - acc: 0.9823 -
val_loss: 0.0671 - val_acc: 0.9802
Epoch 50/50
60000/60000 [==============================] - 6s 108us/step - loss: 0.0581 - acc: 0.9815 -
val_loss: 0.0682 - val_acc: 0.9801
Test score: 0.06824740088986582
Test accuracy: 0.9801
```



we are getting 98.15 train accuracy and 98.01 test accuracy .Lets try the same model with tanh activation

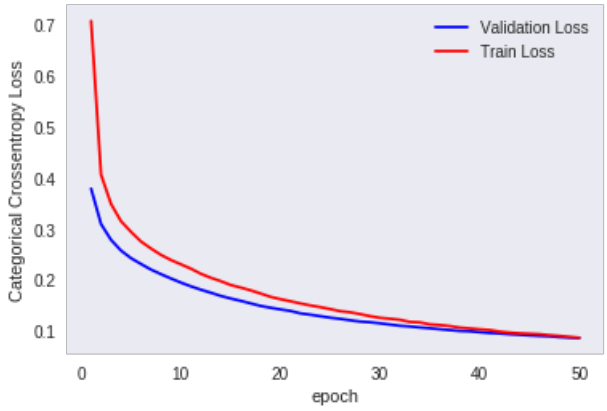
Model6: Layer1: 512 neurons with acivation as tanh + Dropout (0.25)+Layer2: 128 neurons with acivation as tanh+ Dropout (0.5)+ Batch normalization + Layer3: 64 neurons with acivation as tanh + sofmax layer

In [32]:

```
from keras.layers.normalization import BatchNormalization
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='tanh',input_shape=(input_dim,)))
model.add(Dropout(0.25))

model.add(Dense(128,activation='tanh'))
model.add(BatchNormalization())
model.add(Dense(64,activation='tanh'))

model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 10s 159us/step - loss: 0.7072 - acc: 0.7971 - val_l
oss: 0.3784 - val_acc: 0.8977
Epoch 2/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.4058 - acc: 0.8860 -
val_loss: 0.3089 - val_acc: 0.9140
Epoch 3/50
60000/60000 [==============================] - 9s 147us/step - loss: 0.3485 - acc: 0.9011 -
val_loss: 0.2780 - val_acc: 0.9221
Epoch 4/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.3146 - acc: 0.9103 -
val_loss: 0.2570 - val_acc: 0.9289
Epoch 5/50
60000/60000 [==============================] - 9s 146us/step - loss: 0.2938 - acc: 0.9161 -
val_loss: 0.2421 - val_acc: 0.9313
Epoch 6/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.2748 - acc: 0.9212 -
val_loss: 0.2306 - val_acc: 0.9343
Epoch 7/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.2611 - acc: 0.9259 -
val_loss: 0.2197 - val_acc: 0.9372
Epoch 8/50
60000/60000 [==============================] - 9s 149us/step - loss: 0.2486 - acc: 0.9298 -
val_loss: 0.2107 - val_acc: 0.9405
Epoch 9/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.2385 - acc: 0.9321 -
val_loss: 0.2023 - val_acc: 0.9429
Epoch 10/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.2300 - acc: 0.9340 -
val_loss: 0.1942 - val_acc: 0.9439
Epoch 11/50
60000/60000 [==============================] - 9s 146us/step - loss: 0.2215 - acc: 0.9364 -
val_loss: 0.1869 - val_acc: 0.9468
Epoch 12/50
60000/60000 [==============================] - 8s 140us/step - loss: 0.2115 - acc: 0.9397 -
val_loss: 0.1803 - val_acc: 0.9485
Epoch 13/50
60000/60000 [==============================] - 9s 150us/step - loss: 0.2037 - acc: 0.9421 -
val_loss: 0.1743 - val_acc: 0.9501
Epoch 14/50
60000/60000 [==============================] - 9s 150us/step - loss: 0.1970 - acc: 0.9436 -
val_loss: 0.1680 - val_acc: 0.9520
Epoch 15/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.1895 - acc: 0.9453 -
val_loss: 0.1629 - val_acc: 0.9539
Epoch 16/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1844 - acc: 0.9469 -
val_loss: 0.1584 - val_acc: 0.9545
Epoch 17/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1789 - acc: 0.9483 -
val_loss: 0.1534 - val_acc: 0.9558
Epoch 18/50
60000/60000 [==============================] - 8s 128us/step - loss: 0.1727 - acc: 0.9507 -
val_loss: 0.1483 - val_acc: 0.9570
Epoch 19/50
60000/60000 [==============================] - 7s 122us/step - loss: 0.1662 - acc: 0.9521 -
val_loss: 0.1448 - val_acc: 0.9577
Epoch 20/50
60000/60000 [==============================] - 7s 124us/step - loss: 0.1616 - acc: 0.9535 -
val_loss: 0.1415 - val_acc: 0.9587
Epoch 21/50
60000/60000 [==============================] - 7s 124us/step - loss: 0.1575 - acc: 0.9547 -
val_loss: 0.1385 - val_acc: 0.9595
Epoch 22/50
60000/60000 [==============================] - 8s 131us/step - loss: 0.1532 - acc: 0.9557 -
val_loss: 0.1337 - val_acc: 0.9609
Epoch 23/50
60000/60000 [==============================] - 7s 122us/step - loss: 0.1493 - acc: 0.9566 -
val_loss: 0.1311 - val_acc: 0.9611
Epoch 24/50
60000/60000 [==============================] - 8s 127us/step - loss: 0.1460 - acc: 0.9573 -
val_loss: 0.1282 - val_acc: 0.9626
Epoch 25/50
60000/60000 [==============================] - 7s 123us/step - loss: 0.1422 - acc: 0.9591 -
val_loss: 0.1252 - val_acc: 0.9630
```

```
Epoch 26/50
60000/60000 [==============================] - 8s 135us/step - loss: 0.1380 - acc: 0.9603 -
val_loss: 0.1230 - val_acc: 0.9635
Epoch 27/50
60000/60000 [==============================] - 8s 136us/step - loss: 0.1358 - acc: 0.9602 -
val_loss: 0.1201 - val_acc: 0.9636
Epoch 28/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1323 - acc: 0.9613 -
val_loss: 0.1176 - val_acc: 0.9645
Epoch 29/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1282 - acc: 0.9631 -
val_loss: 0.1163 - val_acc: 0.9643
Epoch 30/50
60000/60000 [==============================] - 8s 130us/step - loss: 0.1250 - acc: 0.9639 -
val_loss: 0.1142 - val_acc: 0.9654
Epoch 31/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1232 - acc: 0.9641 -
val_loss: 0.1118 - val_acc: 0.9662
Epoch 32/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1211 - acc: 0.9643 -
val_loss: 0.1094 - val_acc: 0.9658
Epoch 33/50
60000/60000 [==============================] - 8s 139us/step - loss: 0.1169 - acc: 0.9659 -
val_loss: 0.1080 - val_acc: 0.9670
Epoch 34/50
60000/60000 [==============================] - 8s 133us/step - loss: 0.1160 - acc: 0.9666 -
val_loss: 0.1061 - val_acc: 0.9670
Epoch 35/50
60000/60000 [==============================] - 8s 139us/step - loss: 0.1121 - acc: 0.9681 -
val_loss: 0.1045 - val_acc: 0.9679
Epoch 36/50
60000/60000 [==============================] - 8s 137us/step - loss: 0.1108 - acc: 0.9673 -
val_loss: 0.1027 - val_acc: 0.9681
Epoch 37/50
60000/60000 [==============================] - 8s 138us/step - loss: 0.1088 - acc: 0.9681 -
val_loss: 0.1009 - val_acc: 0.9677
Epoch 38/50
60000/60000 [==============================] - 8s 135us/step - loss: 0.1058 - acc: 0.9690 -
val_loss: 0.0992 - val_acc: 0.9691
Epoch 39/50
60000/60000 [==============================] - 8s 138us/step - loss: 0.1042 - acc: 0.9699 -
val_loss: 0.0985 - val_acc: 0.9691
Epoch 40/50
60000/60000 [==============================] - 7s 125us/step - loss: 0.1025 - acc: 0.9701 -
val_loss: 0.0968 - val_acc: 0.9703
Epoch 41/50
60000/60000 [==============================] - 7s 111us/step - loss: 0.1010 - acc: 0.9705 -
val_loss: 0.0952 - val_acc: 0.9701
Epoch 42/50
60000/60000 [==============================] - 7s 118us/step - loss: 0.0979 - acc: 0.9711 -
val_loss: 0.0943 - val_acc: 0.9708
Epoch 43/50
60000/60000 [==============================] - 7s 110us/step - loss: 0.0959 - acc: 0.9719 -
val_loss: 0.0928 - val_acc: 0.9716
Epoch 44/50
60000/60000 [==============================] - 7s 113us/step - loss: 0.0946 - acc: 0.9729 -
val_loss: 0.0918 - val_acc: 0.9715
Epoch 45/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.0938 - acc: 0.9727 -
val_loss: 0.0905 - val_acc: 0.9717
Epoch 46/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0929 - acc: 0.9723 -
val_loss: 0.0895 - val_acc: 0.9723
Epoch 47/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0910 - acc: 0.9734 -
val_loss: 0.0887 - val_acc: 0.9717
Epoch 48/50
60000/60000 [==============================] - 9s 146us/step - loss: 0.0895 - acc: 0.9738 -
val_loss: 0.0872 - val_acc: 0.9724
Epoch 49/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.0875 - acc: 0.9743 -
val_loss: 0.0861 - val_acc: 0.9724
Epoch 50/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.0859 - acc: 0.9748 -
val_loss: 0.0856 - val_acc: 0.9721
Test score: 0.08557037137448788
Test accuracy: 0.9721
```



we are getting 97.48 test accuracy and test accuracy 97.21.Lets try the same architecture with sigmoid

Model7: Layer1: 512 neurons with acivation as sigmoid + Dropout (0.25)+Layer2: 128 neurons with acivation as sigmoid+ Dropout (0.5)+ Batch normalization + Layer3: 64 neurons with acivation as sigmoid + sofmax layer

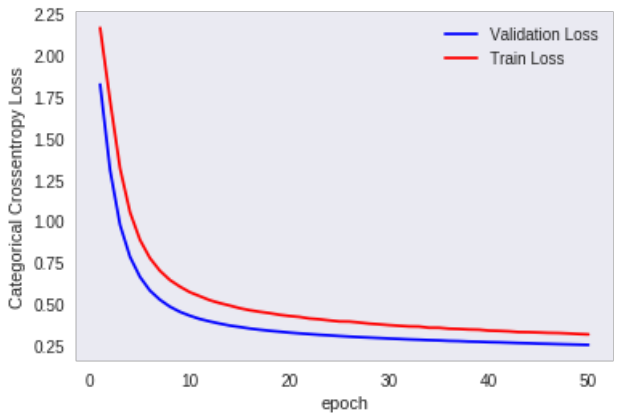
In [33]:

```
from keras.layers.normalization import BatchNormalization
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='sigmoid',input_shape=(input_dim,)))
model.add(Dropout(0.25))

model.add(Dense(128,activation='sigmoid'))
model.add(BatchNormalization())
model.add(Dense(64,activation='sigmoid'))

model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 10s 169us/step - loss: 2.1653 - acc: 0.2596 - val_l
oss: 1.8242 - val_acc: 0.6444
Epoch 2/50
60000/60000 [==============================] - 9s 152us/step - loss: 1.7265 - acc: 0.5192 -
val_loss: 1.3079 - val_acc: 0.7212
Epoch 3/50
60000/60000 [==============================] - 9s 151us/step - loss: 1.3187 - acc: 0.6442 -
val_loss: 0.9771 - val_acc: 0.7758
Epoch 4/50
60000/60000 [==============================] - 9s 150us/step - loss: 1.0535 - acc: 0.7142 -
val_loss: 0.7846 - val_acc: 0.8162
Epoch 5/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.8873 - acc: 0.7578 -
val_loss: 0.6639 - val_acc: 0.8445
Epoch 6/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.7772 - acc: 0.7828 -
val_loss: 0.5813 - val_acc: 0.8606
Epoch 7/50
60000/60000 [==============================] - 9s 149us/step - loss: 0.7010 - acc: 0.8016 -
val_loss: 0.5264 - val_acc: 0.8694
```

```
Epoch 8/50
60000/60000 [==============================] - 10s 160us/step - loss: 0.6452 - acc: 0.8151 - val_l
oss: 0.4846 - val_acc: 0.8756
Epoch 9/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.6057 - acc: 0.8235 -
val_loss: 0.4533 - val_acc: 0.8816
Epoch 10/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.5717 - acc: 0.8341 -
val_loss: 0.4300 - val_acc: 0.8841
Epoch 11/50
60000/60000 [==============================] - 8s 140us/step - loss: 0.5477 - acc: 0.8385 -
val_loss: 0.4102 - val_acc: 0.8886
Epoch 12/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.5235 - acc: 0.8458 -
val_loss: 0.3955 - val_acc: 0.8907
Epoch 13/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.5059 - acc: 0.8495 -
val_loss: 0.3824 - val_acc: 0.8939
Epoch 14/50
60000/60000 [==============================] - 8s 136us/step - loss: 0.4913 - acc: 0.8532 -
val_loss: 0.3706 - val_acc: 0.8956
Epoch 15/50
60000/60000 [==============================] - 7s 124us/step - loss: 0.4753 - acc: 0.8601 -
val_loss: 0.3620 - val_acc: 0.8975
Epoch 16/50
60000/60000 [==============================] - 8s 131us/step - loss: 0.4632 - acc: 0.8622 -
val_loss: 0.3528 - val_acc: 0.8996
Epoch 17/50
60000/60000 [==============================] - 8s 127us/step - loss: 0.4536 - acc: 0.8634 -
val_loss: 0.3456 - val_acc: 0.9021
Epoch 18/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.4455 - acc: 0.8663 -
val_loss: 0.3393 - val_acc: 0.9020
Epoch 19/50
60000/60000 [==============================] - 8s 136us/step - loss: 0.4355 - acc: 0.8692 -
val_loss: 0.3337 - val_acc: 0.9039
Epoch 20/50
60000/60000 [==============================] - 9s 149us/step - loss: 0.4281 - acc: 0.8706 -
val_loss: 0.3281 - val_acc: 0.9047
Epoch 21/50
60000/60000 [==============================] - 9s 149us/step - loss: 0.4226 - acc: 0.8716 -
val_loss: 0.3233 - val_acc: 0.9048
Epoch 22/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.4132 - acc: 0.8751 -
val_loss: 0.3193 - val_acc: 0.9068
Epoch 23/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.4084 - acc: 0.8746 -
val_loss: 0.3148 - val_acc: 0.9076
Epoch 24/50
60000/60000 [==============================] - 9s 150us/step - loss: 0.4015 - acc: 0.8784 -
val_loss: 0.3114 - val_acc: 0.9084
Epoch 25/50
60000/60000 [==============================] - 9s 151us/step - loss: 0.3954 - acc: 0.8815 -
val_loss: 0.3075 - val_acc: 0.9091
Epoch 26/50
60000/60000 [==============================] - 9s 142us/step - loss: 0.3946 - acc: 0.8801 -
val_loss: 0.3039 - val_acc: 0.9110
Epoch 27/50
60000/60000 [==============================] - 8s 138us/step - loss: 0.3889 - acc: 0.8814 -
val_loss: 0.3008 - val_acc: 0.9123
Epoch 28/50
60000/60000 [==============================] - 8s 134us/step - loss: 0.3823 - acc: 0.8823 -
val_loss: 0.2981 - val_acc: 0.9130
Epoch 29/50
60000/60000 [==============================] - 8s 140us/step - loss: 0.3781 - acc: 0.8853 -
val_loss: 0.2953 - val_acc: 0.9131
Epoch 30/50
60000/60000 [==============================] - 9s 143us/step - loss: 0.3732 - acc: 0.8866 -
val_loss: 0.2924 - val_acc: 0.9140
Epoch 31/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.3693 - acc: 0.8876 -
val_loss: 0.2900 - val_acc: 0.9145
Epoch 32/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.3655 - acc: 0.8898 -
val_loss: 0.2872 - val_acc: 0.9152
Epoch 33/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.3646 - acc: 0.8898 -
```

```
val_loss: 0.2851 - val_acc: 0.9164
Epoch 34/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.3574 - acc: 0.8906 -
val_loss: 0.2827 - val_acc: 0.9169
Epoch 35/50
60000/60000 [==============================] - 8s 129us/step - loss: 0.3572 - acc: 0.8906 -
val_loss: 0.2810 - val_acc: 0.9178
Epoch 36/50
60000/60000 [==============================] - 7s 123us/step - loss: 0.3518 - acc: 0.8930 -
val_loss: 0.2782 - val_acc: 0.9176
Epoch 37/50
60000/60000 [==============================] - 7s 125us/step - loss: 0.3500 - acc: 0.8940 -
val_loss: 0.2769 - val_acc: 0.9188
Epoch 38/50
60000/60000 [==============================] - 8s 128us/step - loss: 0.3470 - acc: 0.8947 -
val_loss: 0.2743 - val_acc: 0.9194
Epoch 39/50
60000/60000 [==============================] - 7s 124us/step - loss: 0.3457 - acc: 0.8938 -
val_loss: 0.2729 - val_acc: 0.9200
Epoch 40/50
60000/60000 [==============================] - 7s 125us/step - loss: 0.3402 - acc: 0.8956 -
val_loss: 0.2706 - val_acc: 0.9205
Epoch 41/50
60000/60000 [==============================] - 8s 127us/step - loss: 0.3386 - acc: 0.8975 -
val_loss: 0.2694 - val_acc: 0.9208
Epoch 42/50
60000/60000 [==============================] - 8s 128us/step - loss: 0.3358 - acc: 0.8973 -
val_loss: 0.2673 - val_acc: 0.9214
Epoch 43/50
60000/60000 [==============================] - 8s 130us/step - loss: 0.3313 - acc: 0.9002 -
val_loss: 0.2652 - val_acc: 0.9220
Epoch 44/50
60000/60000 [==============================] - 8s 141us/step - loss: 0.3305 - acc: 0.8996 -
val_loss: 0.2638 - val_acc: 0.9221
Epoch 45/50
60000/60000 [==============================] - 9s 148us/step - loss: 0.3288 - acc: 0.9003 -
val_loss: 0.2617 - val_acc: 0.9228
Epoch 46/50
60000/60000 [==============================] - 9s 146us/step - loss: 0.3265 - acc: 0.9008 -
val_loss: 0.2604 - val_acc: 0.9240
Epoch 47/50
60000/60000 [==============================] - 8s 142us/step - loss: 0.3255 - acc: 0.9007 -
val_loss: 0.2586 - val_acc: 0.9239
Epoch 48/50
60000/60000 [==============================] - 9s 145us/step - loss: 0.3229 - acc: 0.9014 -
val_loss: 0.2571 - val_acc: 0.9248
Epoch 49/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.3196 - acc: 0.9038 -
val_loss: 0.2555 - val_acc: 0.9248
Epoch 50/50
60000/60000 [==============================] - 9s 144us/step - loss: 0.3173 - acc: 0.9041 -
val_loss: 0.2541 - val_acc: 0.9257
Test score: 0.2540672867298126
Test accuracy: 0.9257
```



we are getting 90.41 train accuracy and test accuracy 92.57.Lets try the same architecture with relu and adam optimizer

Model 8: Adam optimizer

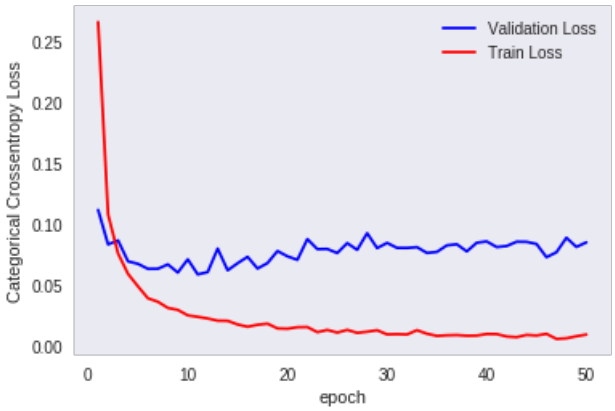
In [35]:

```
from keras.layers.normalization import BatchNormalization
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(input_dim,)))
model.add(Dropout(0.25))

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


model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')

x = list(range(1,nb_epoch+1))
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 [==============================] - 12s 194us/step - loss: 0.2650 - acc: 0.9207 - val_l
oss: 0.1115 - val_acc: 0.9650
Epoch 2/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.1078 - acc: 0.9675 - val_l
oss: 0.0833 - val_acc: 0.9748
Epoch 3/50
60000/60000 [==============================] - 10s 175us/step - loss: 0.0759 - acc: 0.9768 - val_l
oss: 0.0866 - val_acc: 0.9731
Epoch 4/50
60000/60000 [==============================] - 11s 176us/step - loss: 0.0592 - acc: 0.9805 - val_l
oss: 0.0695 - val_acc: 0.9769
Epoch 5/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0490 - acc: 0.9838 - val_l
oss: 0.0674 - val_acc: 0.9787
Epoch 6/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.0393 - acc: 0.9874 - val_l
oss: 0.0633 - val_acc: 0.9788
Epoch 7/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.0364 - acc: 0.9881 - val_l
oss: 0.0634 - val_acc: 0.9797
Epoch 8/50
60000/60000 [==============================] - 10s 175us/step - loss: 0.0314 - acc: 0.9899 - val_l
oss: 0.0670 - val_acc: 0.9810
Epoch 9/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0298 - acc: 0.9900 - val_l
oss: 0.0603 - val_acc: 0.9819
Epoch 10/50
60000/60000 [==============================] - 11s 184us/step - loss: 0.0253 - acc: 0.9912 - val_l
oss: 0.0711 - val_acc: 0.9809
Epoch 11/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0241 - acc: 0.9916 - val_l
oss: 0.0588 - val_acc: 0.9838
Epoch 12/50
60000/60000 [==============================] - 11s 175us/step - loss: 0.0228 - acc: 0.9923 - val_l
oss: 0.0607 - val_acc: 0.9842
Epoch 13/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0210 - acc: 0.9931 - val_l
oss: 0.0799 - val_acc: 0.9796
Epoch 14/50
60000/60000 [==============================] - 11s 176us/step - loss: 0.0208 - acc: 0.9930 - val_l
oss: 0.0621 - val_acc: 0.9852
Epoch 15/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0179 - acc: 0.9937 - val_l
```

```
oss: 0.0679 - val_acc: 0.9834
Epoch 16/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0160 - acc: 0.9944 - val_l
oss: 0.0732 - val_acc: 0.9825
Epoch 17/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0176 - acc: 0.9944 - val_l
oss: 0.0636 - val_acc: 0.9838
Epoch 18/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0185 - acc: 0.9938 - val_l
oss: 0.0680 - val_acc: 0.9835
Epoch 19/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0146 - acc: 0.9952 - val_l
oss: 0.0779 - val_acc: 0.9822
Epoch 20/50
60000/60000 [==============================] - 11s 176us/step - loss: 0.0144 - acc: 0.9953 - val_l
oss: 0.0736 - val_acc: 0.9834
Epoch 21/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0156 - acc: 0.9946 - val_l
oss: 0.0707 - val_acc: 0.9849
Epoch 22/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0158 - acc: 0.9946 - val_l
oss: 0.0876 - val_acc: 0.9821
Epoch 23/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0116 - acc: 0.9959 - val_l
oss: 0.0795 - val_acc: 0.9842
Epoch 24/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0134 - acc: 0.9957 - val_l
oss: 0.0796 - val_acc: 0.9830
Epoch 25/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0112 - acc: 0.9960 - val_l
oss: 0.0762 - val_acc: 0.9846
Epoch 26/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0134 - acc: 0.9953 - val_l
oss: 0.0843 - val_acc: 0.9820
Epoch 27/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0110 - acc: 0.9960 - val_l
oss: 0.0789 - val_acc: 0.9836
Epoch 28/50
60000/60000 [==============================] - 11s 185us/step - loss: 0.0119 - acc: 0.9961 - val_l
oss: 0.0925 - val_acc: 0.9816
Epoch 29/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0131 - acc: 0.9957 - val_l
oss: 0.0803 - val_acc: 0.9831
Epoch 30/50
60000/60000 [==============================] - 11s 175us/step - loss: 0.0097 - acc: 0.9967 - val_l
oss: 0.0845 - val_acc: 0.9845
Epoch 31/50
60000/60000 [==============================] - 10s 172us/step - loss: 0.0099 - acc: 0.9967 - val_l
oss: 0.0805 - val_acc: 0.9849
Epoch 32/50
60000/60000 [==============================] - 11s 175us/step - loss: 0.0097 - acc: 0.9968 - val_l
oss: 0.0804 - val_acc: 0.9844
Epoch 33/50
60000/60000 [==============================] - 10s 173us/step - loss: 0.0132 - acc: 0.9958 - val_l
oss: 0.0811 - val_acc: 0.9830
Epoch 34/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.0103 - acc: 0.9966 - val_l
oss: 0.0763 - val_acc: 0.9841
Epoch 35/50
60000/60000 [==============================] - 10s 173us/step - loss: 0.0085 - acc: 0.9973 - val_l
oss: 0.0772 - val_acc: 0.9839
Epoch 36/50
60000/60000 [==============================] - 10s 173us/step - loss: 0.0090 - acc: 0.9969 - val_l
oss: 0.0825 - val_acc: 0.9848
Epoch 37/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.0092 - acc: 0.9967 - val_l
oss: 0.0834 - val_acc: 0.9845
Epoch 38/50
60000/60000 [==============================] - 11s 178us/step - loss: 0.0087 - acc: 0.9971 - val_l
oss: 0.0776 - val_acc: 0.9853
Epoch 39/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0088 - acc: 0.9971 - val_l
oss: 0.0847 - val_acc: 0.9835
Epoch 40/50
60000/60000 [==============================] - 10s 173us/step - loss: 0.0101 - acc: 0.9969 - val_l
oss: 0.0857 - val_acc: 0.9838
Epoch 41/50
```

```
60000/60000 [==============================] - 11s 176us/step - loss: 0.0100 - acc: 0.9969 - val_l
oss: 0.0811 - val_acc: 0.9849
Epoch 42/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.0080 - acc: 0.9975 - val_l
oss: 0.0820 - val_acc: 0.9848
Epoch 43/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0075 - acc: 0.9974 - val_l
oss: 0.0855 - val_acc: 0.9837
Epoch 44/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0093 - acc: 0.9971 - val_l
oss: 0.0854 - val_acc: 0.9848
Epoch 45/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0088 - acc: 0.9970 - val_l
oss: 0.0837 - val_acc: 0.9841
Epoch 46/50
60000/60000 [==============================] - 11s 178us/step - loss: 0.0102 - acc: 0.9968 - val_l
oss: 0.0728 - val_acc: 0.9857
Epoch 47/50
60000/60000 [==============================] - 10s 174us/step - loss: 0.0061 - acc: 0.9979 - val_l
oss: 0.0769 - val_acc: 0.9863
Epoch 48/50
60000/60000 [==============================] - 11s 176us/step - loss: 0.0066 - acc: 0.9979 - val_l
oss: 0.0887 - val_acc: 0.9835
Epoch 49/50
60000/60000 [==============================] - 11s 175us/step - loss: 0.0082 - acc: 0.9977 - val_l
oss: 0.0812 - val_acc: 0.9850
Epoch 50/50
60000/60000 [==============================] - 11s 176us/step - loss: 0.0096 - acc: 0.9970 - val_l
oss: 0.0850 - val_acc: 0.9855
Test score: 0.08498437745065353
Test accuracy: 0.9855
```



we are getting 99.70 test accuracy and test accuracy 98.55.Lets try the some different architecture with relu and adam optimizer

Model 9:

In [36]:

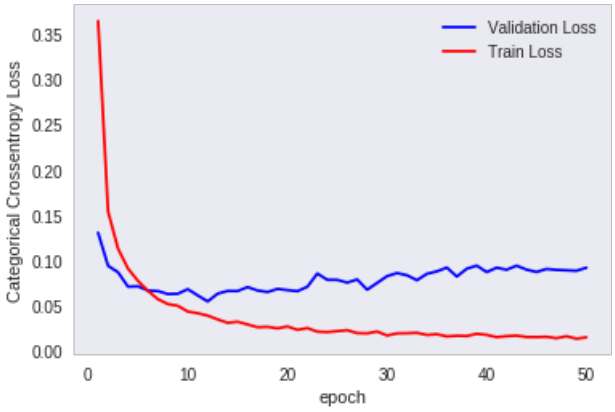
```
from keras.layers.normalization import BatchNormalization
%matplotlib inline
model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(input_dim,)))
model.add(Dropout(0.25))

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

model.add(Dropout(0.5))
model.add(Dense(output_dim, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation
_data=(X_test, Y_test))
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')
```

```
x = list(range(1,nb_epoch+1))
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 [==============================] - 12s 198us/step - loss: 0.3647 - acc: 0.8910 - val_l
oss: 0.1301 - val_acc: 0.9611
Epoch 2/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.1537 - acc: 0.9557 - val_l
oss: 0.0935 - val_acc: 0.9704
Epoch 3/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.1125 - acc: 0.9666 - val_l
oss: 0.0863 - val_acc: 0.9741
Epoch 4/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0904 - acc: 0.9730 - val_l
oss: 0.0704 - val_acc: 0.9780
Epoch 5/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0768 - acc: 0.9763 - val_l
oss: 0.0707 - val_acc: 0.9796
Epoch 6/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0657 - acc: 0.9801 - val_l
oss: 0.0661 - val_acc: 0.9818
Epoch 7/50
60000/60000 [==============================] - 11s 177us/step - loss: 0.0565 - acc: 0.9820 - val_l
oss: 0.0655 - val_acc: 0.9804
Epoch 8/50
60000/60000 [==============================] - 11s 176us/step - loss: 0.0511 - acc: 0.9837 - val_l
oss: 0.0622 - val_acc: 0.9825
Epoch 9/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0492 - acc: 0.9844 - val_l
oss: 0.0626 - val_acc: 0.9821
Epoch 10/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0428 - acc: 0.9865 - val_l
oss: 0.0676 - val_acc: 0.9835
Epoch 11/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0409 - acc: 0.9865 - val_l
oss: 0.0607 - val_acc: 0.9819
Epoch 12/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0383 - acc: 0.9881 - val_l
oss: 0.0540 - val_acc: 0.9845
Epoch 13/50
60000/60000 [==============================] - 11s 186us/step - loss: 0.0341 - acc: 0.9888 - val_l
oss: 0.0627 - val_acc: 0.9838
Epoch 14/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0303 - acc: 0.9901 - val_l
oss: 0.0656 - val_acc: 0.9833
Epoch 15/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0315 - acc: 0.9893 - val_l
oss: 0.0655 - val_acc: 0.9820
Epoch 16/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0285 - acc: 0.9907 - val_l
oss: 0.0699 - val_acc: 0.9823
Epoch 17/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0255 - acc: 0.9920 - val_l
oss: 0.0661 - val_acc: 0.9844
Epoch 18/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0258 - acc: 0.9922 - val_l
oss: 0.0645 - val_acc: 0.9842
Epoch 19/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0242 - acc: 0.9920 - val_l
oss: 0.0679 - val_acc: 0.9839
Epoch 20/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0263 - acc: 0.9917 - val_l
oss: 0.0666 - val_acc: 0.9844
Epoch 21/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0226 - acc: 0.9925 - val_l
oss: 0.0654 - val_acc: 0.9854
Epoch 22/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0245 - acc: 0.9924 - val_l
oss: 0.0703 - val_acc: 0.9845
Epoch 23/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0205 - acc: 0.9931 - val_l
oss: 0.0849 - val_acc: 0.9820
Epoch 24/50
```

```
60000/60000 [==============================] - 11s 190us/step - loss: 0.0201 - acc: 0.9937 - val_l
oss: 0.0780 - val_acc: 0.9827
Epoch 25/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0211 - acc: 0.9936 - val_l
oss: 0.0779 - val_acc: 0.9829
Epoch 26/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0222 - acc: 0.9926 - val_l
oss: 0.0749 - val_acc: 0.9836
Epoch 27/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0189 - acc: 0.9938 - val_l
oss: 0.0784 - val_acc: 0.9842
Epoch 28/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0186 - acc: 0.9938 - val_l
oss: 0.0670 - val_acc: 0.9867
Epoch 29/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0206 - acc: 0.9936 - val_l
oss: 0.0745 - val_acc: 0.9846
Epoch 30/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0163 - acc: 0.9952 - val_l
oss: 0.0820 - val_acc: 0.9837
Epoch 31/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0186 - acc: 0.9942 - val_l
oss: 0.0854 - val_acc: 0.9838
Epoch 32/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0188 - acc: 0.9941 - val_l
oss: 0.0830 - val_acc: 0.9851
Epoch 33/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0192 - acc: 0.9942 - val_l
oss: 0.0775 - val_acc: 0.9844
Epoch 34/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0169 - acc: 0.9945 - val_l
oss: 0.0846 - val_acc: 0.9835
Epoch 35/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0177 - acc: 0.9943 - val_l
oss: 0.0873 - val_acc: 0.9842
Epoch 36/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0153 - acc: 0.9954 - val_l
oss: 0.0915 - val_acc: 0.9832
Epoch 37/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0160 - acc: 0.9952 - val_l
oss: 0.0816 - val_acc: 0.9852
Epoch 38/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0157 - acc: 0.9955 - val_l
oss: 0.0903 - val_acc: 0.9841
Epoch 39/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0182 - acc: 0.9940 - val_l
oss: 0.0937 - val_acc: 0.9841
Epoch 40/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0170 - acc: 0.9948 - val_l
oss: 0.0865 - val_acc: 0.9849
Epoch 41/50
60000/60000 [==============================] - 11s 188us/step - loss: 0.0145 - acc: 0.9957 - val_l
oss: 0.0914 - val_acc: 0.9838
Epoch 42/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0155 - acc: 0.9954 - val_l
oss: 0.0891 - val_acc: 0.9847
Epoch 43/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0162 - acc: 0.9951 - val_l
oss: 0.0936 - val_acc: 0.9843
Epoch 44/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0146 - acc: 0.9953 - val_l
oss: 0.0892 - val_acc: 0.9848
Epoch 45/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0145 - acc: 0.9957 - val_l
oss: 0.0867 - val_acc: 0.9836
Epoch 46/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0148 - acc: 0.9952 - val_l
oss: 0.0900 - val_acc: 0.9829
Epoch 47/50
60000/60000 [==============================] - 11s 180us/step - loss: 0.0133 - acc: 0.9960 - val_l
oss: 0.0890 - val_acc: 0.9853
Epoch 48/50
60000/60000 [==============================] - 11s 183us/step - loss: 0.0154 - acc: 0.9953 - val_l
oss: 0.0885 - val_acc: 0.9838
Epoch 49/50
60000/60000 [==============================] - 11s 181us/step - loss: 0.0127 - acc: 0.9960 - val_l
oss: 0.0880 - val_acc: 0.9850
```

```
Epoch 50/50
60000/60000 [==============================] - 11s 182us/step - loss: 0.0142 - acc: 0.9955 - val_l
oss: 0.0913 - val_acc: 0.9839
Test score: 0.09134189505399432
Test accuracy: 0.9839
```



we are getting 99.55 test accuracy and test accuracy 98.39

Performance table:

In [37]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names= ['Model','Train_accuracy','Test_accuracy']
x.add_row(['Model1','87.17','87.52'])
x.add_row(['Model2','99.01','97.62'])
x.add_row(['Model3','97.08','97.05'])
x.add_row(['Model4','99.39','98.17'])
x.add_row(['Model5','98.15','98.01'])
x.add_row(['Model6','97.48','97.21'])
x.add_row(['Model7','90.77','92.47'])
x.add_row(['Model8','99.70','98.55'])
x.add_row(['Model9','99.55','98.39'])
print(x)
```

```
+--------+----------------+---------------+
| Model  | Train_accuracy | Test_accuracy |
+--------+----------------+---------------+
| Model1 |     87.17      |     87.52     |
| Model2 |     99.01      |     97.62     |
| Model3 |     97.08      |     97.05     |
| Model4 |     99.39      |     98.17     |
| Model5 |     98.15      |     98.01     |
| Model6 |     97.48      |     97.21     |
| Model7 |     90.77      |     92.47     |
| Model8 |     99.70      |     98.55     |
| Model9 |     99.55      |     98.39     |
+--------+----------------+---------------+
```

Conclusion : seeing above table we can say model8 is the bestmodel