CNN on MNIST using Keras

In [1]:

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


import warnings
warnings.filterwarnings("ignore")
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.utils import np_utils
import seaborn as sns
from keras.initializers import RandomNormal
%matplotlib notebook
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import time
from keras.layers import Activation
from datetime import datetime
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic:
more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

1.0 Loading & splitting the data

In [3]:

```
num_classes = 10
# input image dimensions
img_rows, img_cols = 28, 28

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

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

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

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

In [8]:

```
print(input_shape)
```

```
(28, 28, 1)
```

2.0 Utility functions

In [0]:

```
batch_size = 150
num_classes = 10
epochs = 15
```

In [0]:

```
# function to plot epoch vs loss

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()
```

3.0 Three layered CNN using 3X3 kernel

In [0]:

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

start = datetime.now()

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', kernel_initia
lizer= 'he_normal',
                 activation='relu',input_shape=input_shape))

model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(90, (3, 3), activation='relu', kernel_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu',kernel_initializer= 'he_normal'))
model.add(Dropout(0.5))

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

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

fit1=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))

print('Time taken :', datetime.now() - start)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated.
Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Pleas
e use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:4479: The name tf.truncated_normal is deprecated.
Please use tf.random.truncated_normal instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:4267: The name tf.nn.max_pool is deprecated. Plea
se use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:148: The name tf.placeholder_with_default is depr
ecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.op
s.nn_ops) with keep_prob is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. P
lease use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optim
izers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.com
pat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:3576: The name tf.log is deprecated. Please use t
f.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_
core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array
_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Pleas
e use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please us
e tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:3005: The name tf.Session is deprecated. Please u
se tf.compat.v1.Session instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/15
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:190: The name tf.get_default_session is deprecate
d. Please use tf.compat.v1.get_default_session instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Pleas
e use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:207: The name tf.global_variables is deprecated.
Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:216: The name tf.is_variable_initialized is depre
cated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:223: The name tf.variables_initializer is depreca
ted. Please use tf.compat.v1.variables_initializer instead.

60000/60000 [==============================] - 167s 3ms/step - loss: 0.338
1 - acc: 0.8939 - val_loss: 0.0503 - val_acc: 0.9845
Epoch 2/15
60000/60000 [==============================] - 165s 3ms/step - loss: 0.096
0 - acc: 0.9716 - val_loss: 0.0341 - val_acc: 0.9888
Epoch 3/15
60000/60000 [==============================] - 165s 3ms/step - loss: 0.070
9 - acc: 0.9794 - val_loss: 0.0278 - val_acc: 0.9897
Epoch 4/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.060
4 - acc: 0.9828 - val_loss: 0.0230 - val_acc: 0.9921
Epoch 5/15
60000/60000 [==============================] - 163s 3ms/step - loss: 0.052
5 - acc: 0.9839 - val_loss: 0.0216 - val_acc: 0.9936
Epoch 6/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.044
7 - acc: 0.9863 - val_loss: 0.0245 - val_acc: 0.9927
Epoch 7/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.042
0 - acc: 0.9876 - val_loss: 0.0254 - val_acc: 0.9924
Epoch 8/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.037
0 - acc: 0.9882 - val_loss: 0.0176 - val_acc: 0.9942
Epoch 9/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.034
7 - acc: 0.9899 - val_loss: 0.0163 - val_acc: 0.9946
Epoch 10/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.031
6 - acc: 0.9906 - val_loss: 0.0213 - val_acc: 0.9931
Epoch 11/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.030
0 - acc: 0.9906 - val_loss: 0.0171 - val_acc: 0.9945
Epoch 12/15
60000/60000 [==============================] - 163s 3ms/step - loss: 0.028
8 - acc: 0.9910 - val_loss: 0.0161 - val_acc: 0.9946
Epoch 13/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.026
1 - acc: 0.9920 - val_loss: 0.0185 - val_acc: 0.9945
Epoch 14/15
60000/60000 [==============================] - 163s 3ms/step - loss: 0.026
4 - acc: 0.9918 - val_loss: 0.0178 - val_acc: 0.9945
Epoch 15/15
60000/60000 [==============================] - 164s 3ms/step - loss: 0.024
```

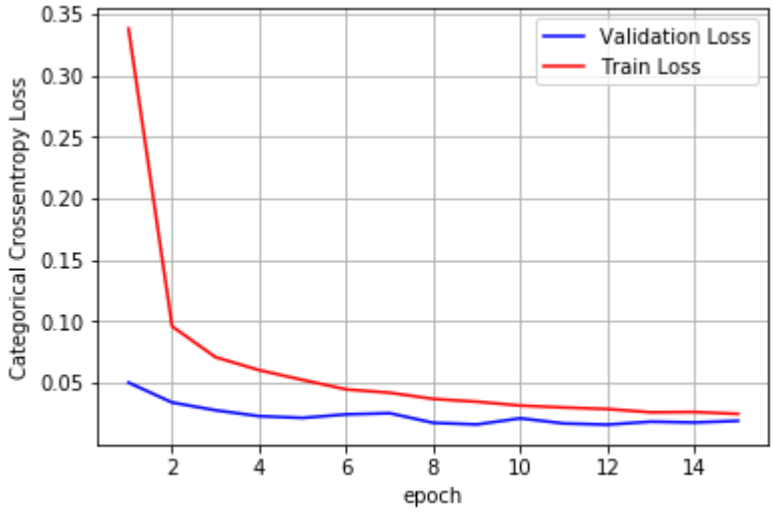
```
8 - acc: 0.9924 - val_loss: 0.0192 - val_acc: 0.9941
Time taken : 0:40:59.787039
```

3.1 Results & Plot

In [0]:

```
score1 = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score1[0])
print('Test accuracy:', score1[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, ve
rbose=1, validation_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 ep
ochs


vy = fit1.history['val_loss']
ty = fit1.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.019186536238695043
Test accuracy: 0.9941
```



4.0 Five layered CNN using 2X2 kernel with batch normalization & dropout

In [0]:

```
batch_size = 200
num_classes = 10
epochs = 10
```

In [30]:

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

start = datetime.now()

model = Sequential()

model.add(Conv2D(32, kernel_size=(2, 2), strides=(1, 1), padding='same', kernel_initial
izer= 'he_normal',
                 activation='relu',input_shape=input_shape))

model.add(Conv2D(64, (2, 2), strides=(1, 1),padding='same',activation='relu', kernel_in
itializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(80, (2, 2), strides=(1, 1), padding='same', activation='relu', kernel_
initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(100, (2, 2), strides=(1, 1), padding='same', activation='relu', kernel
_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(120, (2, 2), strides=(1, 1), padding='same', activation='relu', kernel
_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu',kernel_initializer= 'he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

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

fit2=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))

print('Time taken :', datetime.now() - start)
```

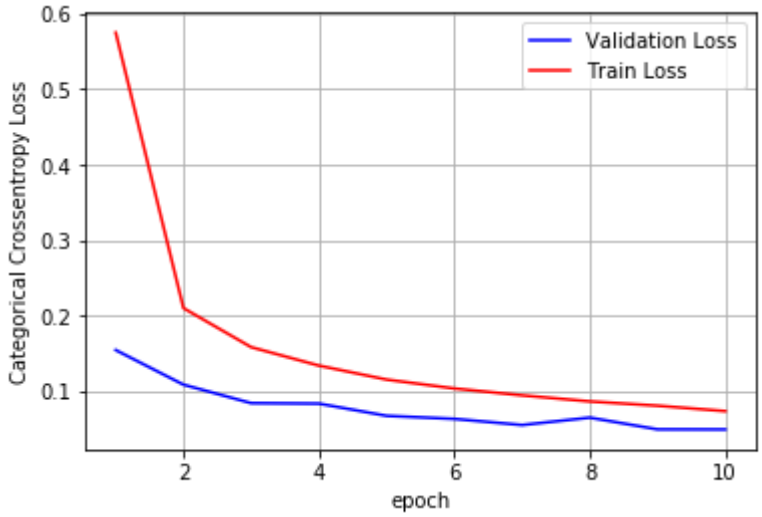
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 117s 2ms/step - loss: 0.574
7 - acc: 0.8229 - val_loss: 0.1548 - val_acc: 0.9518
Epoch 2/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.210
3 - acc: 0.9351 - val_loss: 0.1090 - val_acc: 0.9648
Epoch 3/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.158
6 - acc: 0.9506 - val_loss: 0.0845 - val_acc: 0.9723
Epoch 4/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.134
2 - acc: 0.9588 - val_loss: 0.0839 - val_acc: 0.9729
Epoch 5/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.115
7 - acc: 0.9641 - val_loss: 0.0677 - val_acc: 0.9771
Epoch 6/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.103
9 - acc: 0.9682 - val_loss: 0.0637 - val_acc: 0.9784
Epoch 7/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.094
9 - acc: 0.9706 - val_loss: 0.0556 - val_acc: 0.9816
Epoch 8/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.086
8 - acc: 0.9726 - val_loss: 0.0654 - val_acc: 0.9790
Epoch 9/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.081
1 - acc: 0.9753 - val_loss: 0.0498 - val_acc: 0.9835
Epoch 10/10
60000/60000 [==============================] - 114s 2ms/step - loss: 0.074
0 - acc: 0.9772 - val_loss: 0.0497 - val_acc: 0.9844
Time taken : 0:19:04.763188
```

4.1 Results & Plot

In [31]:

```
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))

# plot
vy = fit2.history['val_loss']
ty = fit2.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.049710892591020094
Test accuracy: 0.9844
```



5.0 Seven layered CNN using 5X5 kernel with batch normalization & dropout

In [0]:

```
batch_size = 200
num_classes = 10
epochs = 10
```

In [25]:

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

start = datetime.now()

model = Sequential()

model.add(Conv2D(32, kernel_size=(5, 5), strides=(2, 2), padding='same', kernel_initial
izer= 'he_normal',
                 activation='relu',input_shape=input_shape))

model.add(Conv2D(64, (5, 5), strides=(2, 2),padding='same', activation='relu', kernel_i
nitializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(80, (5, 5), strides=(2, 2),padding='same', activation='relu', kernel_i
nitializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(100, (5, 5), strides=(2, 2), padding='same', activation='relu', kernel
_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(120, (5, 5), strides=(2, 2), padding='same', activation='relu', kernel
_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(150, (5, 5), strides=(2, 2), padding='same', activation='relu', kernel
_initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(170, (5, 5), strides=(2, 2),padding='same', activation='relu', kernel_
initializer= 'he_normal'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(150, activation='relu',kernel_initializer= 'he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

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

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

fit3=model.fit(x_train, y_train,
```

```
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))

print('Time taken :', datetime.now() - start)
```

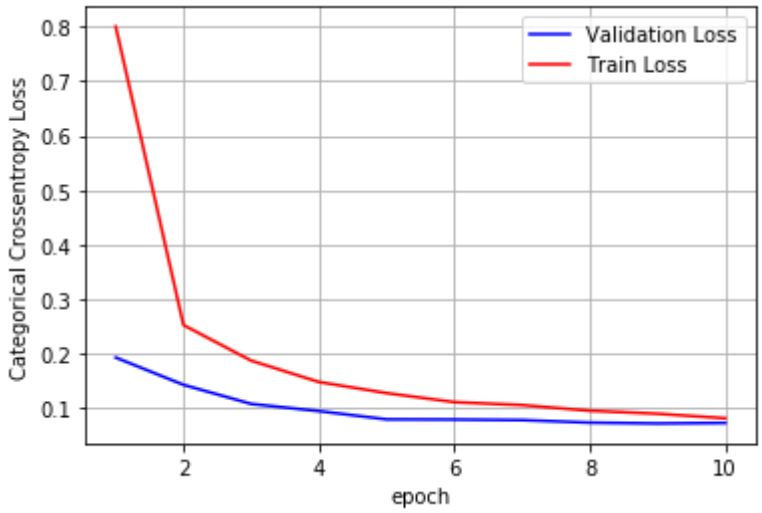
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 248s 4ms/step - loss: 0.799
9 - acc: 0.7518 - val_loss: 0.1932 - val_acc: 0.9486
Epoch 2/10
60000/60000 [==============================] - 242s 4ms/step - loss: 0.252
8 - acc: 0.9306 - val_loss: 0.1432 - val_acc: 0.9610
Epoch 3/10
60000/60000 [==============================] - 243s 4ms/step - loss: 0.187
5 - acc: 0.9498 - val_loss: 0.1082 - val_acc: 0.9714
Epoch 4/10
60000/60000 [==============================] - 243s 4ms/step - loss: 0.148
4 - acc: 0.9595 - val_loss: 0.0950 - val_acc: 0.9734
Epoch 5/10
60000/60000 [==============================] - 243s 4ms/step - loss: 0.128
0 - acc: 0.9654 - val_loss: 0.0799 - val_acc: 0.9763
Epoch 6/10
60000/60000 [==============================] - 243s 4ms/step - loss: 0.111
5 - acc: 0.9697 - val_loss: 0.0795 - val_acc: 0.9772
Epoch 7/10
60000/60000 [==============================] - 242s 4ms/step - loss: 0.106
1 - acc: 0.9712 - val_loss: 0.0786 - val_acc: 0.9791
Epoch 8/10
60000/60000 [==============================] - 242s 4ms/step - loss: 0.095
7 - acc: 0.9736 - val_loss: 0.0739 - val_acc: 0.9800
Epoch 9/10
60000/60000 [==============================] - 242s 4ms/step - loss: 0.090
2 - acc: 0.9747 - val_loss: 0.0724 - val_acc: 0.9800
Epoch 10/10
60000/60000 [==============================] - 243s 4ms/step - loss: 0.081
9 - acc: 0.9774 - val_loss: 0.0733 - val_acc: 0.9788
Time taken : 0:40:35.118176
```

5.1 Results & Plot

In [26]:

```
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))

# plot
vy = fit3.history['val_loss']
ty = fit3.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
Test score: 0.07328068963296246
Test accuracy: 0.9788
```



6.0 Summary

In [38]:

```
import pandas as pd

df= pd.DataFrame(columns=["Number of Layers","Kernel size","Strides","Pooling window si
ze","Batch size","Epoch","Test Loss","Test Accuracy"],index=['I','II','III'])
df.loc['I']=[3,('3X3'),(1,1), (2,2),150,15, 0.0192, '99.41%']
df.loc['II']=[5,('2X2'),(1,1), (1,1),200,10, 0.0497, '98.44%']
df.loc['III']=[7,('5X5'),(2,2), (1,1),200,10, 0.0733, '97.88%']
df
```

Out[38]:

	Number of Layers	Kernel size	Strides	Pooling window size	Batch size	Epoch	Test Loss	Test Accuracy
I	3	3X3	(1, 1)	(2, 2)	150	15	0.0192	99.41%
II	5	2X2	(1, 1)	(1, 1)	200	10	0.0497	98.44%
III	7	5X5	(2, 2)	(1, 1)	200	10	0.0733	97.88%