In [1]:
```
%matplotlib inline
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

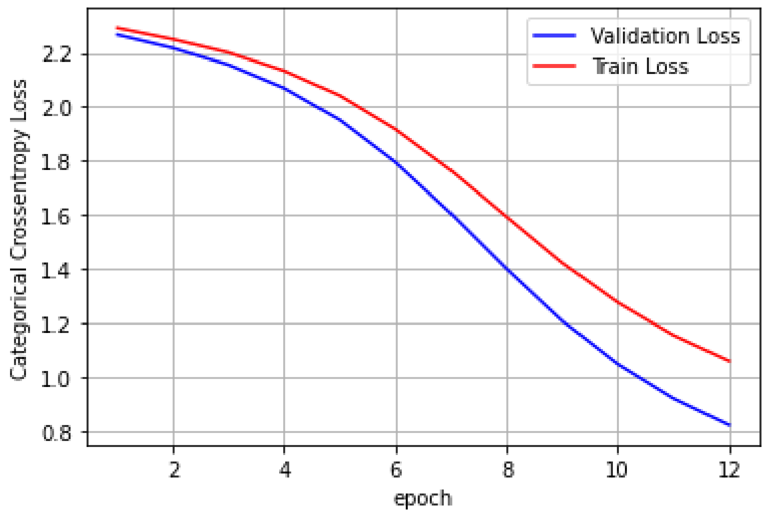
model with 3*3 kernal

In [2]:
```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mni
st_cnn.py


from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

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

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

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

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

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

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

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


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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-ke
ras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/12
469/469 [==============================] - 142s 303ms/step - loss: 2.
2917 - accuracy: 0.1370 - val_loss: 2.2671 - val_accuracy: 0.3519
Epoch 2/12
469/469 [==============================] - 143s 304ms/step - loss: 2.
2506 - accuracy: 0.2508 - val_loss: 2.2183 - val_accuracy: 0.4990
Epoch 3/12
469/469 [==============================] - 143s 304ms/step - loss: 2.
2013 - accuracy: 0.3431 - val_loss: 2.1554 - val_accuracy: 0.5730
```

```
Epoch 4/12
469/469 [==============================] - 148s 316ms/step - loss: 2.
1335 - accuracy: 0.4221 - val_loss: 2.0696 - val_accuracy: 0.6225
Epoch 5/12
469/469 [==============================] - 143s 304ms/step - loss: 2.
0420 - accuracy: 0.4828 - val_loss: 1.9521 - val_accuracy: 0.6600
Epoch 6/12
469/469 [==============================] - 142s 304ms/step - loss: 1.
9179 - accuracy: 0.5285 - val_loss: 1.7958 - val_accuracy: 0.6905
Epoch 7/12
469/469 [==============================] - 143s 304ms/step - loss: 1.
7653 - accuracy: 0.5660 - val_loss: 1.6055 - val_accuracy: 0.7199
Epoch 8/12
469/469 [==============================] - 147s 314ms/step - loss: 1.
5933 - accuracy: 0.6018 - val_loss: 1.4008 - val_accuracy: 0.7509
Epoch 9/12
469/469 [==============================] - 142s 303ms/step - loss: 1.
4232 - accuracy: 0.6286 - val_loss: 1.2090 - val_accuracy: 0.7778
Epoch 10/12
469/469 [==============================] - 143s 304ms/step - loss: 1.
2763 - accuracy: 0.6544 - val_loss: 1.0480 - val_accuracy: 0.7959
Epoch 11/12
469/469 [==============================] - 142s 304ms/step - loss: 1.
1549 - accuracy: 0.6780 - val_loss: 0.9209 - val_accuracy: 0.8109
Epoch 12/12
469/469 [==============================] - 143s 304ms/step - loss: 1.
0595 - accuracy: 0.6970 - val_loss: 0.8224 - val_accuracy: 0.8248
Test loss: 0.8223868012428284
Test accuracy: 0.8248000144958496
```



In [3]:
```
pip install scikit-plot
```

```
Collecting scikit-plot
  Downloading https://files.pythonhosted.org/packages/7c/47/32520e25934
0c140a4ad27c1b97050dd3254fdc517b1d59974d47037510e/scikit_plot-0.3.7-py3
-none-any.whl
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/pyt
hon3.6/dist-packages (from scikit-plot) (0.22.2.post1)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.
6/dist-packages (from scikit-plot) (0.16.0)
Requirement already satisfied: matplotlib>=1.4.0 in /usr/local/lib/pyth
on3.6/dist-packages (from scikit-plot) (3.2.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.6/d
ist-packages (from scikit-plot) (1.4.1)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.
6/dist-packages (from scikit-learn>=0.18->scikit-plot) (1.18.5)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/pyth
on3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4.0->scik
it-plot) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.
```

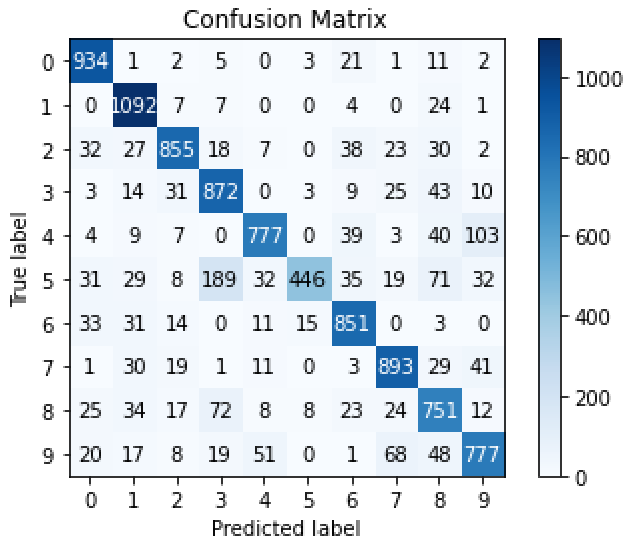
```
6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/p
ython3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (2.8.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-pac
kages (from cycler>=0.10->matplotlib>=1.4.0->scikit-plot) (1.15.0)
Installing collected packages: scikit-plot
Successfully installed scikit-plot-0.3.7
```

In [4]:

```
pred=model.predict(x_test)
#pred= (pred>0.5)
import scikitplot.metrics as skplt
skplt.plot_confusion_matrix(y_test.argmax(axis=1), pred.argmax(axis=1))
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4331ee5940>
```



model with 5*5 kernal

In [5]:

```
from keras.layers import BatchNormalization
model = Sequential()
```

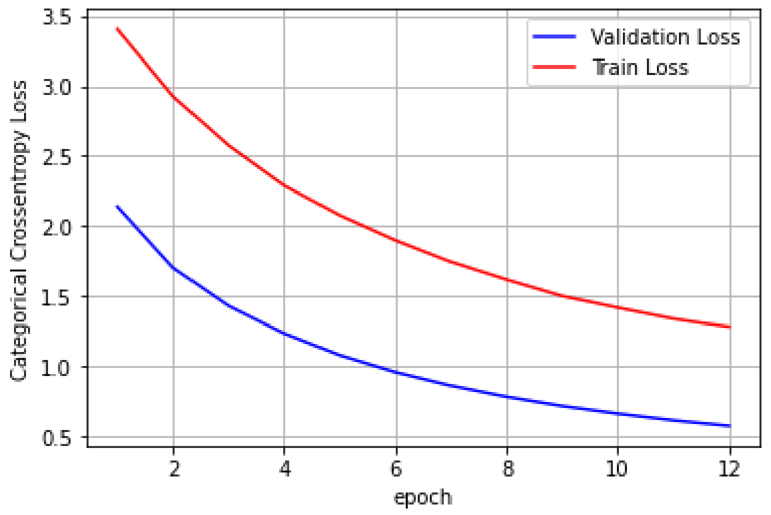
```
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.50))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.7))
model.add(Dense(num_classes, activation='softmax'))

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

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


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

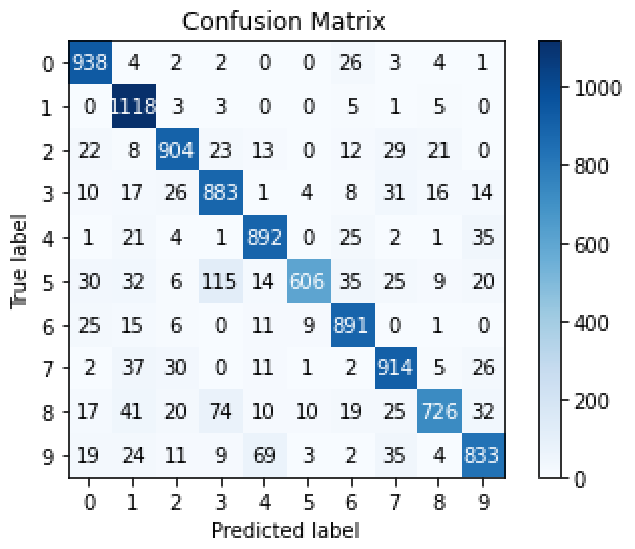
```python
pred=model.predict(x_test)
#pred= (pred>0.5)
import scikitplot.metrics as skplt
skplt.plot_confusion_matrix(y_test.argmax(axis=1), pred.argmax(axis=1))
```

```
Epoch 1/12
469/469 [==============================] - 240s 511ms/step - loss: 3.
4053 - accuracy: 0.1143 - val_loss: 2.1387 - val_accuracy: 0.2530
Epoch 2/12
469/469 [==============================] - 234s 499ms/step - loss: 2.
9262 - accuracy: 0.1647 - val_loss: 1.7026 - val_accuracy: 0.5195
Epoch 3/12
469/469 [==============================] - 239s 510ms/step - loss: 2.
5755 - accuracy: 0.2241 - val_loss: 1.4310 - val_accuracy: 0.6585
Epoch 4/12
469/469 [==============================] - 234s 499ms/step - loss: 2.
2903 - accuracy: 0.2851 - val_loss: 1.2336 - val_accuracy: 0.7352
Epoch 5/12
469/469 [==============================] - 234s 500ms/step - loss: 2.
0765 - accuracy: 0.3413 - val_loss: 1.0781 - val_accuracy: 0.7770
Epoch 6/12
469/469 [==============================] - 234s 499ms/step - loss: 1.
8990 - accuracy: 0.3873 - val_loss: 0.9578 - val_accuracy: 0.8026
Epoch 7/12
469/469 [==============================] - 233s 496ms/step - loss: 1.
7444 - accuracy: 0.4339 - val_loss: 0.8611 - val_accuracy: 0.8206
Epoch 8/12
469/469 [==============================] - 232s 496ms/step - loss: 1.
6180 - accuracy: 0.4736 - val_loss: 0.7819 - val_accuracy: 0.8339
Epoch 9/12
469/469 [==============================] - 233s 497ms/step - loss: 1.
5013 - accuracy: 0.5087 - val_loss: 0.7160 - val_accuracy: 0.8450
Epoch 10/12
469/469 [==============================] - 233s 496ms/step - loss: 1.
4185 - accuracy: 0.5367 - val_loss: 0.6611 - val_accuracy: 0.8549
Epoch 11/12
469/469 [==============================] - 233s 497ms/step - loss: 1.
3404 - accuracy: 0.5610 - val_loss: 0.6135 - val_accuracy: 0.8628
```

```
Epoch 12/12
469/469 [==============================] - 238s 507ms/step - loss: 1.
2808 - accuracy: 0.5809 - val_loss: 0.5738 - val_accuracy: 0.8705
Test loss: 0.5738340616226196
Test accuracy: 0.8705000281333923
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4331e3b198>
```





model with 7*7 kernal

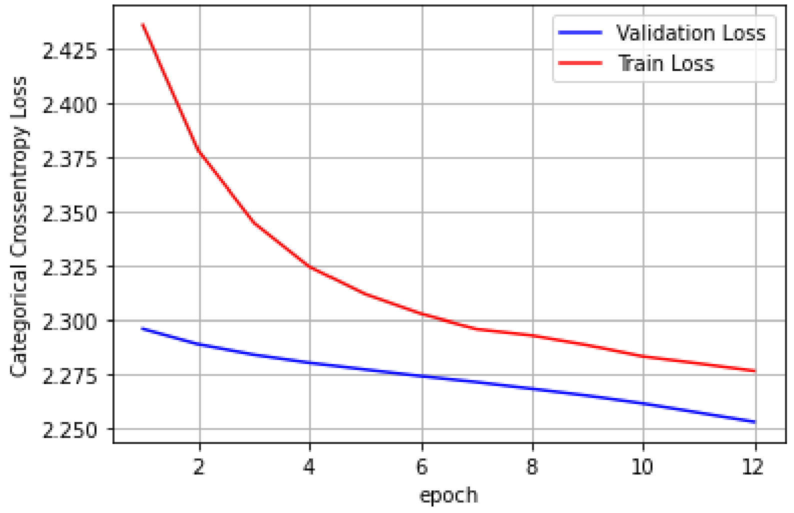
In [6]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.8))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.9))
model.add(Dense(num_classes, activation='softmax'))

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

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


vy = history.history['val_loss']
```

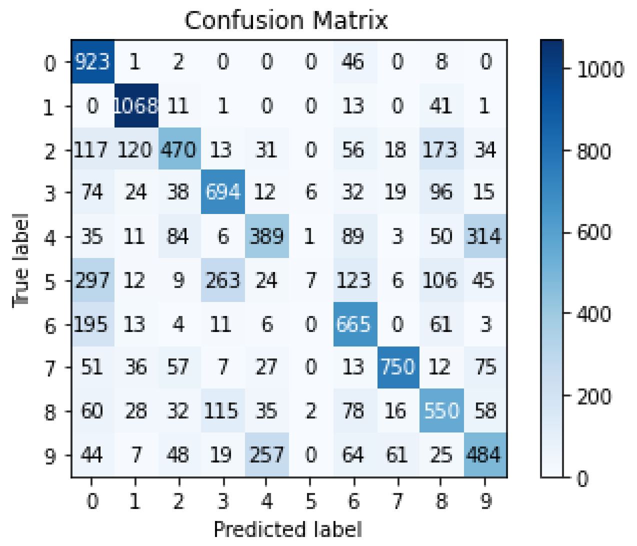
```
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)


pred=model.predict(x_test)
#pred= (pred>0.5)
import scikitplot.metrics as skplt
skplt.plot_confusion_matrix(y_test.argmax(axis=1), pred.argmax(axis=1))
```

```
Epoch 1/12
469/469 [==============================] - 278s 593ms/step - loss: 2.43
56 - accuracy: 0.1020 - val_loss: 2.2958 - val_accuracy: 0.1097
Epoch 2/12
469/469 [==============================] - 282s 601ms/step - loss: 2.37
77 - accuracy: 0.1048 - val_loss: 2.2887 - val_accuracy: 0.1421
Epoch 3/12
469/469 [==============================] - 276s 589ms/step - loss: 2.34
46 - accuracy: 0.1075 - val_loss: 2.2839 - val_accuracy: 0.2034
Epoch 4/12
469/469 [==============================] - 282s 601ms/step - loss: 2.32
43 - accuracy: 0.1099 - val_loss: 2.2802 - val_accuracy: 0.2901
Epoch 5/12
469/469 [==============================] - 276s 589ms/step - loss: 2.31
17 - accuracy: 0.1140 - val_loss: 2.2770 - val_accuracy: 0.3738
Epoch 6/12
469/469 [==============================] - 278s 593ms/step - loss: 2.30
27 - accuracy: 0.1213 - val_loss: 2.2740 - val_accuracy: 0.4372
Epoch 7/12
469/469 [==============================] - 278s 593ms/step - loss: 2.29
57 - accuracy: 0.1226 - val_loss: 2.2712 - val_accuracy: 0.4861
Epoch 8/12
469/469 [==============================] - 276s 589ms/step - loss: 2.29
28 - accuracy: 0.1251 - val_loss: 2.2682 - val_accuracy: 0.5183
Epoch 9/12
469/469 [==============================] - 275s 587ms/step - loss: 2.28
83 - accuracy: 0.1284 - val_loss: 2.2650 - val_accuracy: 0.5441
Epoch 10/12
469/469 [==============================] - 275s 587ms/step - loss: 2.28
31 - accuracy: 0.1323 - val_loss: 2.2614 - val_accuracy: 0.5666
Epoch 11/12
```

```
469/469 [==============================] - 276s 588ms/step - loss: 2.27
99 - accuracy: 0.1348 - val_loss: 2.2571 - val_accuracy: 0.5844
Epoch 12/12
469/469 [==============================] - 275s 587ms/step - loss: 2.27
64 - accuracy: 0.1393 - val_loss: 2.2529 - val_accuracy: 0.6000
Test loss: 2.2528669834136963
Test accuracy: 0.6000000238418579
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f433f437780>
```





	0	1	2	3	4	5	6	7	8	9
0	923	1	2	0	0	0	46	0	8	0
1	0	1068	11	1	0	0	13	0	41	1
2	117	120	470	13	31	0	56	18	173	34
3	74	24	38	694	12	6	32	19	96	15
4	35	11	84	6	389	1	89	3	50	314
5	297	12	9	263	24	7	123	6	106	45
6	195	13	4	11	6	0	665	0	61	3
7	51	36	57	7	27	0	13	750	12	75
8	60	28	32	115	35	2	78	16	550	58
9	44	7	48	19	257	0	64	61	25	484

In [7]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["model","accuracy"]
x.add_row(["model with 3*3 kernal",one])
x.add_row(["model with 5*5 kernal",two])
x.add_row(["model with 7*7 kernal",three])
print(x)
```

```
+-----------------------+--------------------+
|         model         |      accuracy      |
+-----------------------+--------------------+
| model with 3*3 kernal | 0.8248000144958496 |
| model with 5*5 kernal | 0.8705000281333923 |
| model with 7*7 kernal | 0.6000000238418579 |
+-----------------------+--------------------+
```

from above chart, model with 5*5 keral has .87 max accuracy, so will consider this model.