In [1]:
```
from keras.layers import Dense, Dropout, Flatten, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D, Activation
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.models import load_model
import matplotlib.pyplot as plt
from keras import backend as K
import pandas as pd
import numpy as np
import itertools
import keras
import cv2

# emotion = ['Angry', 'Happy', 'Sad', 'Surprise', 'Neutral']
emotion = ['Angry', "Disgust", 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
img_rows, img_cols = 48, 48
# num_classes = 5
num_classes = 7

data = pd.read_csv('data/fer2013-7.csv', delimiter=',')
# data_test = data[27215:]
data_test = data[32300:]
y_test = data_test['emotion'].values
print('Len ', len(y_test))
x_test = np.zeros((y_test.shape[0], 48*48))
for i in range(y_test.shape[0]):
    try:
        # x_test[i] = np.fromstring(data_test['pixels'][27215+i], dtype=int, s
ep=' ')
        x_test[i] = np.fromstring(data_test['pixels'][32300+i], dtype=int, sep
=' ')
    except:
        pass

x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
x_test_rev = np.flip(x_test, 2)

input_shape = (img_rows, img_cols, 1)
x_test = x_test.astype('float32')
x_test_rev = x_test_rev.astype('float32')
x_test /= 255
x_test_rev /= 255
y_test = keras.utils.to_categorical(y_test, num_classes)

print('Loading Models...')
print('0/3')
model = []
for i in range(2):
    m = load_model('saved_model/' + 'cnn'+str(i)+'.h5')
    print(str(i+1) + '/3')
    model.append(m)

m = load_model('saved_model/ensemble.h5')
model.append(m)
print('3/3')
print("Loading Complete!")
```

```
def plot_confusion_matrix(cm):
    print(cm)
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title('Confusion Matrix on Private Test Data')
    plt.colorbar()
    tick_marks = np.arange(len(emotion))
    plt.xticks(tick_marks, emotion)
    plt.yticks(tick_marks, emotion)

    fmt = '.2f'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True Emotions')
    plt.xlabel('Predicted Emotions')
    plt.tight_layout()

p = np.zeros((y_test.shape[0],14))
# print('y_test ',y_test.shape)
# p = np.zeros((1,14))
p[:,0:7] = model[0].predict(x_test)
p[:,7:14] = model[1].predict(x_test_rev)
print('p ',p.shape)
y_pred = model[2].predict(p)
yp = np.argmax(y_pred, axis=1)
yt = np.argmax(y_test, axis=1)
cm = confusion_matrix(yt, yp)
plot_confusion_matrix(cm)
plt.savefig("img/cm.png")
plt.show()
```

```
Using TensorFlow backend.

Len  3587
Loading Models...
0/3
1/3
2/3
3/3
Loading Complete!
p  (3587, 14)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-1-6df1cc3c437f> in <module>
     81 p[:,7:14] = model[1].predict(x_test_rev)
     82 print('p ',p.shape)
---> 83 y_pred = model[2].predict(p)
     84 yp = np.argmax(y_pred, axis=1)
     85 yt = np.argmax(y_test, axis=1)

~\Anaconda3\envs\tf-gpu\lib\site-packages\keras\engine\training.py in predict
(self, x, batch_size, verbose, steps)
   1147                                'argument.')
   1148         # Validate user data.
-> 1149         x, _, _ = self._standardize_user_data(x)
   1150         if self.stateful:
   1151             if x[0].shape[0] > batch_size and x[0].shape[0] % batch_s
ize != 0:

~\Anaconda3\envs\tf-gpu\lib\site-packages\keras\engine\training.py in _standa
rdize_user_data(self, x, y, sample_weight, class_weight, check_array_lengths,
batch_size)
    749             feed_input_shapes,
    750             check_batch_axis=False,  # Don't enforce the batch size.
--> 751             exception_prefix='input')
    752
    753         if y is not None:

~\Anaconda3\envs\tf-gpu\lib\site-packages\keras\engine\training_utils.py in s
tandardize_input_data(data, names, shapes, check_batch_axis, exception_prefi
x)
    126                         ': expected ' + names[i] + ' to have ' +
    127                         str(len(shape)) + ' dimensions, but got array
'
--> 128                         'with shape ' + str(data_shape))
    129                 if not check_batch_axis:
    130                     data_shape = data_shape[1:]

ValueError: Error when checking input: expected batch_normalization_8_input t
o have 4 dimensions, but got array with shape (3587, 14)
```