Youtube_architecture

October 13, 2018

In [1]:
```
from keras.utils import np_utils
import itertools
from sklearn.datasets import load_files
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.optimizers import Adagrad, adam
from keras.preprocessing import image
from keras import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dropout, Dense, BatchNormalization
import numpy as np
import pandas as pd
import os
from keras.regularizers import l2
from sklearn.metrics import confusion_matrix, classification_report
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from sklearn.model_selection import train_test_split


def extract_data(filepath):
    dataset = pd.read_csv( filepath, delimiter=',', dtype='a')
    labels = np.array(dataset['emotion'], np.float32)
    raw_data = np.array(dataset['pixels'])
    data = np.array([np.fromstring(i, np.uint8, sep=" ") for i in raw_data])
    data = (data - 128) / 255
    data = data.reshape(data.shape[0], 48, 48, 1)
    return data, labels
```

```
Using TensorFlow backend.
```

In [2]:
```
filepath = 'fer2013.csv'
label_names = ['Anger', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
data, labels = extract_data(filepath)
num_class = len(set(labels))

train_data, x_test, train_labels, y_test = train_test_split(data, labels, test_size=0.2,
x_train, x_valid, y_train, y_valid = train_test_split(train_data, train_labels, test_siz
print(len(x_train), len(x_valid), len(x_test))
```

```
        y_train = (np.arange(num_class) == y_train[:, None]).astype(np.float32)
        y_valid = (np.arange(num_class) == y_valid[:, None]).astype(np.float32)
        y_test = (np.arange(num_class) == y_test[:, None]).astype(np.float32)
```

```
22967 5742 7178
```

```
In [3]: def model_1():
            model = Sequential()
            model.add(Conv2D(filters=32, kernel_size=5, activation='relu', input_shape=(48, 48,
            model.add(BatchNormalization())
            # model.add(Conv2D(filters=32, kernel_size=5, activation='relu', padding='same', nam
            model.add(MaxPool2D(pool_size=2, padding='same'))
            model.add(Dropout(0.1))

            model.add(Conv2D(filters=64, kernel_size=5, activation='relu', padding='same', name=
            model.add(BatchNormalization())
            model.add(MaxPool2D(pool_size=2, padding='same'))
            model.add(Dropout(0.1))

            model.add(Conv2D(filters=128, kernel_size=5, activation='relu', padding='same', name
            model.add(BatchNormalization())
            model.add(MaxPool2D(pool_size=2, padding='same'))
            model.add(Dropout(0.3))

            model.add(Flatten())
            model.add(Dense(128, activation='relu', name='Dense1', kernel_regularizer=l2(1e-4)))
            model.add(BatchNormalization())
            model.add(Dense(256, activation='relu', name='Dense2', kernel_regularizer=l2(1e-4)))
            model.add(BatchNormalization())
            model.add(Dropout(0.5))
            model.add(Dense(len(set(labels)), activation='softmax', name='Dense3'))
            opt = adam(lr=0.001, epsilon=1e-8, decay=1e-4)
            model.compile(optimizer=opt, metrics=['accuracy'], loss='categorical_crossentropy')
            model.summary()
            return model
```

```
In [4]: model = model_1()
        best_model= ModelCheckpoint('yt-2_Facial_Expression.hdf5', save_best_only=True, verbose=
        stop = EarlyStopping(monitor='val_loss', patience=6)
        history = model.fit(x_train, y_train, batch_size=32, epochs=30, verbose=1, shuffle=True,
        # model.load_weights('yt-1_Facial_Expression.hdf5')
        print('Evaluate Accuracy ', model.evaluate(x_test, y_test, batch_size=64))
```

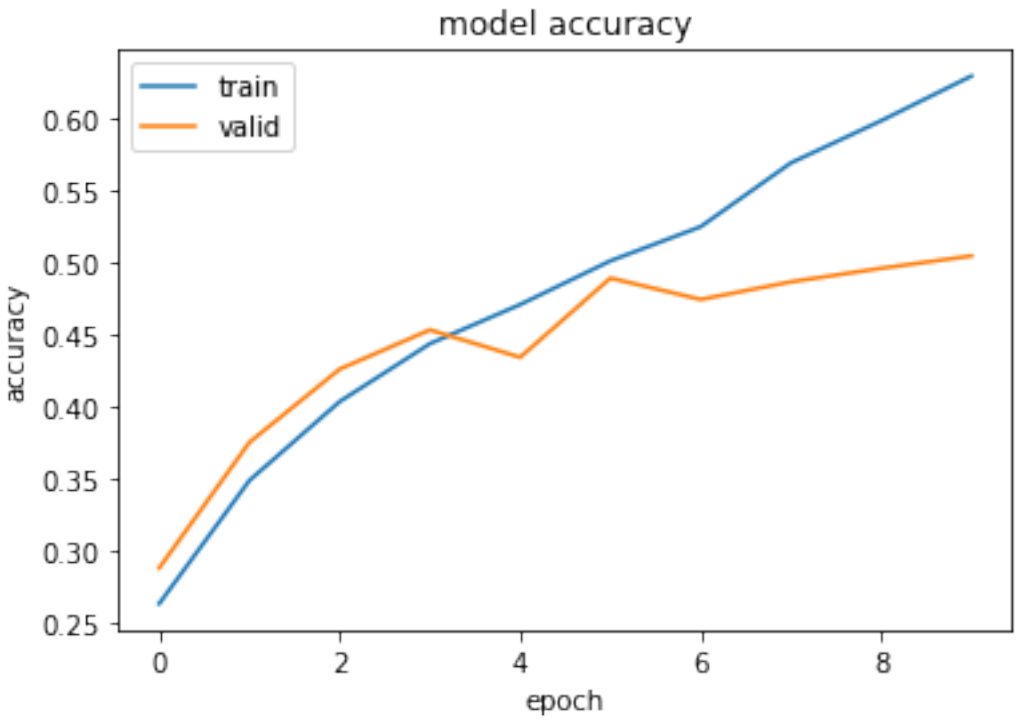
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 48, 48, 32)        832
```

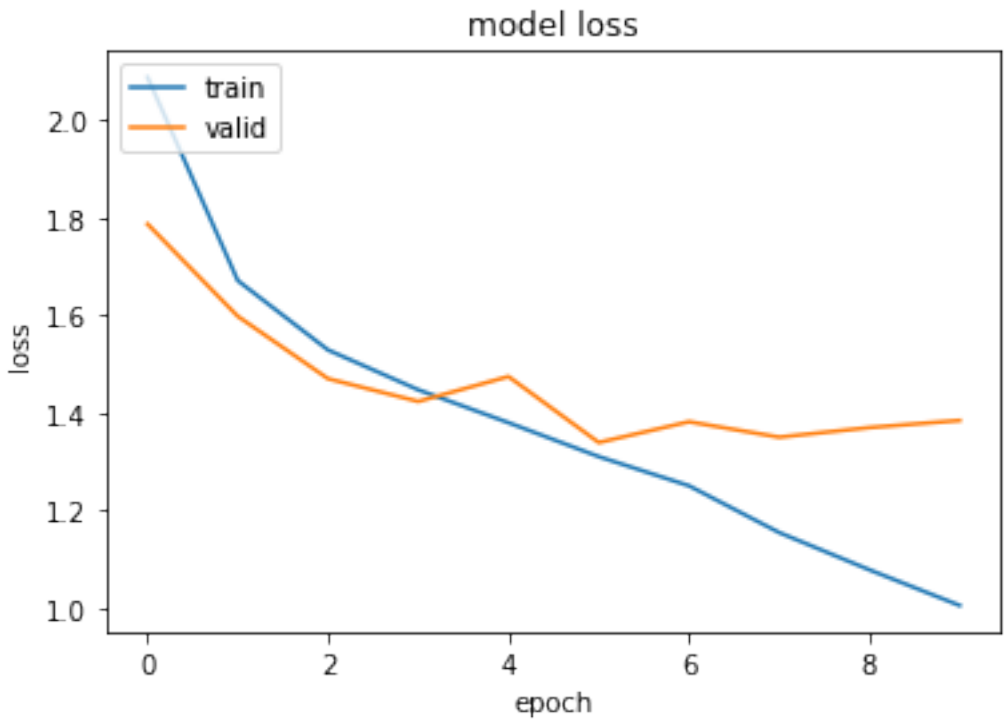
```
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 24, 24, 64)        51264
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
conv3 (Conv2D)               (None, 12, 12, 128)       204928
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
Dense1 (Dense)               (None, 128)               589952
_________________________________________________________________
batch_normalization_4 (Batch (None, 128)               512
_________________________________________________________________
Dense2 (Dense)               (None, 256)               33024
_________________________________________________________________
batch_normalization_5 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
Dense3 (Dense)               (None, 7)                 1799
=================================================================
Total params: 884,231
Trainable params: 883,015
Non-trainable params: 1,216
_________________________________________________________________
Train on 22967 samples, validate on 5742 samples
Epoch 1/30
22944/22967 [============================>.] - ETA: 0s - loss: 2.1407 - acc: 0.2632Epoch 00001:
22967/22967 [==============================] - 23s 1ms/step - loss: 2.1402 - acc: 0.2633 - val_l
Epoch 2/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.7132 - acc: 0.3499Epoch 00002:
22967/22967 [==============================] - 21s 931us/step - loss: 1.7133 - acc: 0.3498 - val
```

```
Epoch 3/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.5884 - acc: 0.4053Epoch 00003:
22967/22967 [==============================] - 21s 925us/step - loss: 1.5886 - acc: 0.4052 - val
Epoch 4/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.5131 - acc: 0.4400Epoch 00004:
22967/22967 [==============================] - 21s 932us/step - loss: 1.5132 - acc: 0.4399 - val
Epoch 5/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.4581 - acc: 0.4667Epoch 00005:
22967/22967 [==============================] - 21s 927us/step - loss: 1.4580 - acc: 0.4666 - val
Epoch 6/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.3864 - acc: 0.5018Epoch 00006:
22967/22967 [==============================] - 21s 927us/step - loss: 1.3861 - acc: 0.5019 - val
Epoch 7/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.3259 - acc: 0.5338Epoch 00007:
22967/22967 [==============================] - 21s 931us/step - loss: 1.3259 - acc: 0.5337 - val
Epoch 8/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.2537 - acc: 0.5651Epoch 00008:
22967/22967 [==============================] - 21s 923us/step - loss: 1.2537 - acc: 0.5652 - val
Epoch 9/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.1655 - acc: 0.6053Epoch 00009:
22967/22967 [==============================] - 21s 925us/step - loss: 1.1653 - acc: 0.6055 - val
Epoch 10/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.0965 - acc: 0.6402Epoch 00010:
22967/22967 [==============================] - 21s 928us/step - loss: 1.0961 - acc: 0.6404 - val
Epoch 11/30
22944/22967 [============================>.] - ETA: 0s - loss: 1.0183 - acc: 0.6760Epoch 00011:
22967/22967 [==============================] - 21s 925us/step - loss: 1.0184 - acc: 0.6759 - val
Epoch 12/30
22944/22967 [============================>.] - ETA: 0s - loss: 0.9373 - acc: 0.7065Epoch 00012:
22967/22967 [==============================] - 21s 923us/step - loss: 0.9373 - acc: 0.7065 - val
Epoch 13/30
22944/22967 [============================>.] - ETA: 0s - loss: 0.8689 - acc: 0.7340Epoch 00013:
22967/22967 [==============================] - 21s 924us/step - loss: 0.8689 - acc: 0.7340 - val
7178/7178 [==============================] - 1s 190us/step
Evaluate Accuracy  [1.6522178171271567, 0.50264697688208482]
```

```
In [5]: # summarize history for accuracy
        plt.figure(1)

        plt.plot(history.history['acc'])
        plt.plot(history.history['val_acc'])
        plt.title('model accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('epoch')
        plt.legend(['train', 'valid'], loc='upper left')
        plt.show()
        # summarize history for loss
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```

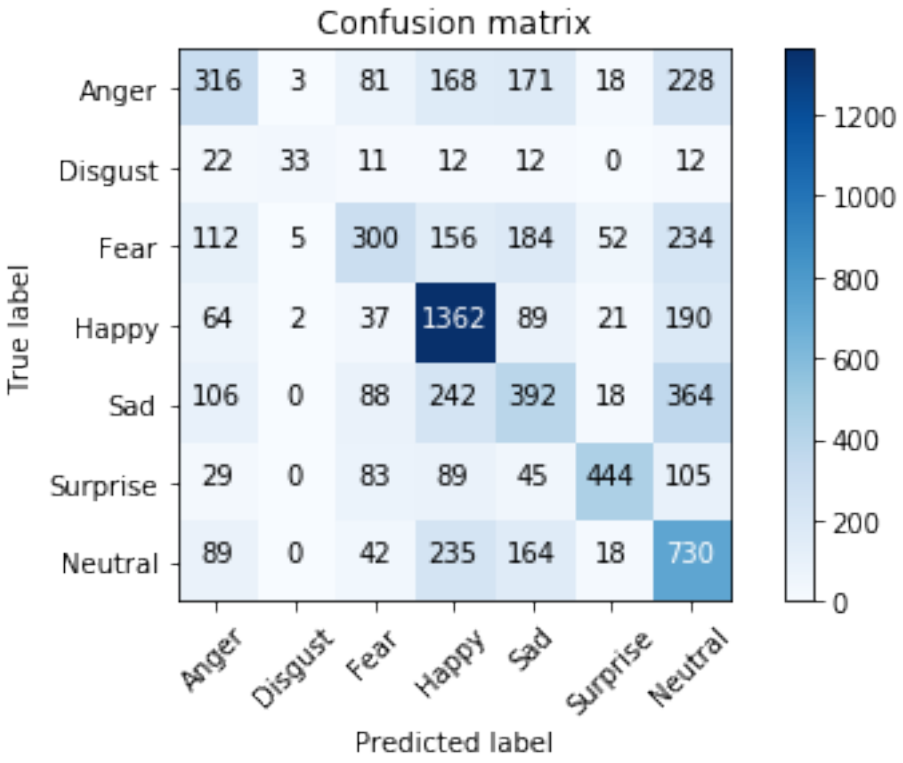





```
In [6]: y_pred = np.argmax(model.predict(x_test), axis=1)
        print('Confusion Matrix ')


        def plot_confusion_matrix(cm, normalize=False, title='Confusion matrix', cmap=plt.cm.Blu
            plt.imshow(cm, interpolation='nearest', cmap=cmap)
            plt.title(title)
            plt.colorbar()
            tick_marks = np.arange(7)
            plt.xticks(tick_marks, label_names, rotation=45)
            plt.yticks(tick_marks, label_names)
            fmt = '.2f' if normalize else 'd'
            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.tight_layout()
            plt.ylabel('True label')
            plt.xlabel('Predicted label')


        cm = confusion_matrix(np.argmax(y_test, axis=1), y_pred)
```

```
        np.set_printoptions(precision=2)
        print('Confusion matrix, without normalization')
        print(cm)
        plt.figure()
        plot_confusion_matrix(cm)
        plt.show()
```

```
Confusion Matrix
Confusion matrix, without normalization
[[ 381    2  124  116  199   16  147]
 [  22   42   10   12   11    1    4]
 [ 104    4  369  127  268   47  124]
 [  76    4   81 1314  148   27  115]
 [ 116    7  128  181  513   18  247]
 [  38    3  125   83   38  431   77]
 [ 108    1   98  235  255   23  558]]
```



In [ ]: