In [1]:
```
# Convolutional Neural Network

# Installing Theano
# pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git

# Installing Tensorflow
# Install Tensorflow from the website: https://www.tensorflow.org/versions/r0.
12/get_started/os_setup.html

# Installing Keras
# pip install --upgrade keras

# Part 1 - Building the CNN

# Importing the Keras libraries and packages
import numpy as np
import os
import keras_metrics
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import TimeDistributed
from keras.layers import LSTM
from keras.layers import Reshape

import warnings
warnings.filterwarnings('ignore')

# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Convolution2D(64, (3, 3), padding = 'same', input_shape = (128,
 128, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer
classifier.add(Convolution2D(64, (3, 3), padding = 'same', activation = 'relu'
))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a third conolutional layer
classifier.add(Convolution2D(64, (3, 3), padding = 'same', activation = 'relu'
))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())
classifier.add(Dropout(rate = 0.5))

# Step 4 - Full connection
```

```
classifier.add(Reshape((4*4, 1024)))
classifier.add(LSTM(units = 50, return_sequences = True, dropout = 0.5))
classifier.add(LSTM(units = 20, return_sequences = False, dropout = 0.5))
classifier.add(Dense(output_dim = 7, activation = 'softmax'))

classifier.summary()
```

```
Z:\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion
of the second argument of issubdtype from `float` to `np.floating` is depreca
ted. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 64)      1792      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        36928     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        36928     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 16384)             0         
_________________________________________________________________
reshape_1 (Reshape)          (None, 16, 1024)          0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 16, 50)            215000    
_________________________________________________________________
lstm_2 (LSTM)                (None, 20)                5680      
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 147       
=================================================================
Total params: 296,475
Trainable params: 296,475
Non-trainable params: 0
_________________________________________________________________
```

In [2]:
```
# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metr
ics = ['accuracy', keras_metrics.precision(), keras_metrics.recall()])
```

In [3]:
```
# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   height_shift_range =  0.1,
                                   width_shift_range = 0.1,
                                   channel_shift_range = 10)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('train/',
                                                 target_size = (128, 128),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory('test/',
                                            target_size = (128, 128),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 399 images belonging to 7 classes.
Found 136 images belonging to 7 classes.
```

In [4]:
```
results = classifier.fit_generator(training_set,
                         samples_per_epoch = 399,
                         nb_epoch = 100,
                         validation_data = test_set,
                         nb_val_samples = 136)
```

```
Epoch 1/100
12/12 [==============================] - 49s 4s/step - loss: 1.9143 - acc: 0.
2069 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9084 - val_ac
c: 0.2362 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/100
12/12 [==============================] - 48s 4s/step - loss: 1.9038 - acc: 0.
2474 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9125 - val_ac
c: 0.2352 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 3/100
12/12 [==============================] - 48s 4s/step - loss: 1.9245 - acc: 0.
2082 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8963 - val_ac
c: 0.2354 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 4/100
12/12 [==============================] - 48s 4s/step - loss: 1.8978 - acc: 0.
2461 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8850 - val_ac
c: 0.2352 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 5/100
12/12 [==============================] - 50s 4s/step - loss: 1.8921 - acc: 0.
2292 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8680 - val_ac
c: 0.2370 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 6/100
12/12 [==============================] - 48s 4s/step - loss: 1.8520 - acc: 0.
2688 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8176 - val_ac
c: 0.2424 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 7/100
12/12 [==============================] - 48s 4s/step - loss: 1.7921 - acc: 0.
3177 - precision: 0.0278 - recall: 0.0026 - val_loss: 1.8917 - val_acc: 0.250
3 - val_precision: 0.6616 - val_recall: 0.1320
Epoch 8/100
12/12 [==============================] - 48s 4s/step - loss: 1.7237 - acc: 0.
3533 - precision: 0.4216 - recall: 0.0448 - val_loss: 1.6879 - val_acc: 0.287
3 - val_precision: 0.7339 - val_recall: 0.0886
Epoch 9/100
12/12 [==============================] - 48s 4s/step - loss: 1.7189 - acc: 0.
3159 - precision: 0.4940 - recall: 0.0395 - val_loss: 1.6180 - val_acc: 0.344
8 - val_precision: 0.6448 - val_recall: 0.1102
Epoch 10/100
12/12 [==============================] - 47s 4s/step - loss: 1.6805 - acc: 0.
3295 - precision: 0.5380 - recall: 0.0366 - val_loss: 1.5894 - val_acc: 0.338
0 - val_precision: 0.3031 - val_recall: 0.0146
Epoch 11/100
12/12 [==============================] - 48s 4s/step - loss: 1.6081 - acc: 0.
3464 - precision: 0.3819 - recall: 0.0208 - val_loss: 1.5653 - val_acc: 0.352
3 - val_precision: 0.7547 - val_recall: 0.0589
Epoch 12/100
12/12 [==============================] - 48s 4s/step - loss: 1.6179 - acc: 0.
3688 - precision: 0.4968 - recall: 0.0653 - val_loss: 1.4874 - val_acc: 0.397
1 - val_precision: 0.6698 - val_recall: 0.1172
Epoch 13/100
12/12 [==============================] - 48s 4s/step - loss: 1.5661 - acc: 0.
3795 - precision: 0.5514 - recall: 0.0813 - val_loss: 1.4719 - val_acc: 0.425
5 - val_precision: 0.7798 - val_recall: 0.0877
Epoch 14/100
12/12 [==============================] - 48s 4s/step - loss: 1.5341 - acc: 0.
3717 - precision: 0.5453 - recall: 0.1308 - val_loss: 1.4250 - val_acc: 0.425
5 - val_precision: 0.6695 - val_recall: 0.1533
Epoch 15/100
```

```
12/12 [==============================] - 48s 4s/step - loss: 1.4931 - acc: 0.
3978 - precision: 0.7114 - recall: 0.1360 - val_loss: 1.4320 - val_acc: 0.396
1 - val_precision: 0.6427 - val_recall: 0.2127
Epoch 16/100
12/12 [==============================] - 48s 4s/step - loss: 1.4968 - acc: 0.
4215 - precision: 0.7140 - recall: 0.1388 - val_loss: 1.3310 - val_acc: 0.440
9 - val_precision: 0.6825 - val_recall: 0.1906
Epoch 17/100
12/12 [==============================] - 48s 4s/step - loss: 1.4361 - acc: 0.
4397 - precision: 0.6642 - recall: 0.1335 - val_loss: 1.3252 - val_acc: 0.469
5 - val_precision: 0.6751 - val_recall: 0.1682
Epoch 18/100
12/12 [==============================] - 48s 4s/step - loss: 1.4433 - acc: 0.
3952 - precision: 0.6403 - recall: 0.1860 - val_loss: 1.2830 - val_acc: 0.485
2 - val_precision: 0.6682 - val_recall: 0.2206
Epoch 19/100
12/12 [==============================] - 48s 4s/step - loss: 1.3958 - acc: 0.
4245 - precision: 0.6977 - recall: 0.1562 - val_loss: 1.3313 - val_acc: 0.507
3 - val_precision: 0.5356 - val_recall: 0.2284
Epoch 20/100
12/12 [==============================] - 47s 4s/step - loss: 1.3072 - acc: 0.
4741 - precision: 0.6750 - recall: 0.1920 - val_loss: 1.2984 - val_acc: 0.543
5 - val_precision: 0.5501 - val_recall: 0.2349
Epoch 21/100
12/12 [==============================] - 48s 4s/step - loss: 1.4891 - acc: 0.
4010 - precision: 0.6871 - recall: 0.1719 - val_loss: 1.4710 - val_acc: 0.477
3 - val_precision: 0.7829 - val_recall: 0.0734
Epoch 22/100
12/12 [==============================] - 48s 4s/step - loss: 1.4168 - acc: 0.
4109 - precision: 0.5798 - recall: 0.1411 - val_loss: 1.2391 - val_acc: 0.544
0 - val_precision: 0.6944 - val_recall: 0.1914
Epoch 23/100
12/12 [==============================] - 47s 4s/step - loss: 1.3369 - acc: 0.
4602 - precision: 0.6661 - recall: 0.1974 - val_loss: 1.2400 - val_acc: 0.552
4 - val_precision: 0.7750 - val_recall: 0.2357
Epoch 24/100
12/12 [==============================] - 48s 4s/step - loss: 1.2956 - acc: 0.
4792 - precision: 0.7068 - recall: 0.2188 - val_loss: 1.1679 - val_acc: 0.574
0 - val_precision: 0.7516 - val_recall: 0.3013
Epoch 25/100
12/12 [==============================] - 48s 4s/step - loss: 1.3447 - acc: 0.
4502 - precision: 0.6346 - recall: 0.2016 - val_loss: 1.1707 - val_acc: 0.559
4 - val_precision: 0.7164 - val_recall: 0.2805
Epoch 26/100
12/12 [==============================] - 48s 4s/step - loss: 1.3265 - acc: 0.
4529 - precision: 0.6670 - recall: 0.2119 - val_loss: 1.1736 - val_acc: 0.587
7 - val_precision: 0.7814 - val_recall: 0.2716
Epoch 27/100
12/12 [==============================] - 48s 4s/step - loss: 1.2744 - acc: 0.
5052 - precision: 0.6836 - recall: 0.2044 - val_loss: 1.1475 - val_acc: 0.581
0 - val_precision: 0.7603 - val_recall: 0.3234
Epoch 28/100
12/12 [==============================] - 61s 5s/step - loss: 1.3447 - acc: 0.
4688 - precision: 0.6126 - recall: 0.2109 - val_loss: 1.1822 - val_acc: 0.544
0 - val_precision: 0.6769 - val_recall: 0.2935
Epoch 29/100
12/12 [==============================] - 81s 7s/step - loss: 1.2383 - acc: 0.
```

```
5215 - precision: 0.6880 - recall: 0.2318 - val_loss: 1.1758 - val_acc: 0.558
6 - val_precision: 0.6489 - val_recall: 0.3091
Epoch 30/100
12/12 [==============================] - 82s 7s/step - loss: 1.2140 - acc: 0.
5211 - precision: 0.7127 - recall: 0.2539 - val_loss: 1.1151 - val_acc: 0.610
7 - val_precision: 0.7081 - val_recall: 0.3596
Epoch 31/100
12/12 [==============================] - 82s 7s/step - loss: 1.2342 - acc: 0.
5053 - precision: 0.6785 - recall: 0.2538 - val_loss: 1.1055 - val_acc: 0.579
9 - val_precision: 0.7020 - val_recall: 0.3518
Epoch 32/100
12/12 [==============================] - 83s 7s/step - loss: 1.1721 - acc: 0.
5339 - precision: 0.7143 - recall: 0.3203 - val_loss: 1.1265 - val_acc: 0.558
3 - val_precision: 0.7045 - val_recall: 0.3896
Epoch 33/100
12/12 [==============================] - 71s 6s/step - loss: 1.2018 - acc: 0.
5528 - precision: 0.7101 - recall: 0.3106 - val_loss: 1.0745 - val_acc: 0.611
5 - val_precision: 0.8038 - val_recall: 0.3901
Epoch 34/100
12/12 [==============================] - 50s 4s/step - loss: 1.1641 - acc: 0.
5421 - precision: 0.6484 - recall: 0.2880 - val_loss: 1.1354 - val_acc: 0.587
7 - val_precision: 0.6812 - val_recall: 0.3604
Epoch 35/100
12/12 [==============================] - 52s 4s/step - loss: 1.1213 - acc: 0.
5547 - precision: 0.7035 - recall: 0.3594 - val_loss: 1.0232 - val_acc: 0.625
5 - val_precision: 0.7767 - val_recall: 0.4347
Epoch 36/100
12/12 [==============================] - 49s 4s/step - loss: 1.2784 - acc: 0.
5078 - precision: 0.6564 - recall: 0.2774 - val_loss: 1.0977 - val_acc: 0.624
5 - val_precision: 0.7483 - val_recall: 0.3969
Epoch 37/100
12/12 [==============================] - 49s 4s/step - loss: 1.1566 - acc: 0.
5731 - precision: 0.7103 - recall: 0.2957 - val_loss: 1.0994 - val_acc: 0.610
2 - val_precision: 0.7048 - val_recall: 0.4050
Epoch 38/100
12/12 [==============================] - 49s 4s/step - loss: 1.1311 - acc: 0.
5576 - precision: 0.6884 - recall: 0.3427 - val_loss: 1.0322 - val_acc: 0.646
3 - val_precision: 0.7894 - val_recall: 0.4409
Epoch 39/100
12/12 [==============================] - 49s 4s/step - loss: 1.1499 - acc: 0.
5369 - precision: 0.6367 - recall: 0.3328 - val_loss: 1.0348 - val_acc: 0.602
1 - val_precision: 0.7006 - val_recall: 0.4627
Epoch 40/100
12/12 [==============================] - 48s 4s/step - loss: 1.1196 - acc: 0.
5577 - precision: 0.6897 - recall: 0.3481 - val_loss: 1.0275 - val_acc: 0.580
5 - val_precision: 0.7443 - val_recall: 0.4916
Epoch 41/100
12/12 [==============================] - 48s 4s/step - loss: 1.2230 - acc: 0.
5127 - precision: 0.6754 - recall: 0.3374 - val_loss: 1.1126 - val_acc: 0.595
3 - val_precision: 0.7604 - val_recall: 0.4406
Epoch 42/100
12/12 [==============================] - 77s 6s/step - loss: 1.1502 - acc: 0.
5578 - precision: 0.7049 - recall: 0.3537 - val_loss: 1.1024 - val_acc: 0.603
7 - val_precision: 0.6448 - val_recall: 0.3755
Epoch 43/100
12/12 [==============================] - 81s 7s/step - loss: 1.1827 - acc: 0.
5365 - precision: 0.6768 - recall: 0.3584 - val_loss: 1.1906 - val_acc: 0.595
```

```
8 - val_precision: 0.6530 - val_recall: 0.3898
Epoch 44/100
12/12 [==============================] - 81s 7s/step - loss: 1.2199 - acc: 0.
5105 - precision: 0.6484 - recall: 0.3194 - val_loss: 1.1453 - val_acc: 0.566
4 - val_precision: 0.6545 - val_recall: 0.3896
Epoch 45/100
12/12 [==============================] - 81s 7s/step - loss: 1.2211 - acc: 0.
5235 - precision: 0.6430 - recall: 0.3322 - val_loss: 1.0914 - val_acc: 0.580
5 - val_precision: 0.7304 - val_recall: 0.4333
Epoch 46/100
12/12 [==============================] - 84s 7s/step - loss: 1.0994 - acc: 0.
5990 - precision: 0.6846 - recall: 0.3490 - val_loss: 0.9547 - val_acc: 0.647
7 - val_precision: 0.7978 - val_recall: 0.4933
Epoch 47/100
12/12 [==============================] - 83s 7s/step - loss: 1.0438 - acc: 0.
5887 - precision: 0.7167 - recall: 0.3769 - val_loss: 1.1012 - val_acc: 0.603
7 - val_precision: 0.6557 - val_recall: 0.4425
Epoch 48/100
12/12 [==============================] - 84s 7s/step - loss: 1.0934 - acc: 0.
5785 - precision: 0.6777 - recall: 0.4187 - val_loss: 0.9904 - val_acc: 0.616
6 - val_precision: 0.7292 - val_recall: 0.5359
Epoch 49/100
12/12 [==============================] - 83s 7s/step - loss: 1.1558 - acc: 0.
5471 - precision: 0.6798 - recall: 0.3612 - val_loss: 1.2080 - val_acc: 0.537
3 - val_precision: 0.6034 - val_recall: 0.3683
Epoch 50/100
12/12 [==============================] - 83s 7s/step - loss: 1.1102 - acc: 0.
5735 - precision: 0.7046 - recall: 0.3613 - val_loss: 1.1741 - val_acc: 0.530
2 - val_precision: 0.6047 - val_recall: 0.3826
Epoch 51/100
12/12 [==============================] - 83s 7s/step - loss: 1.1212 - acc: 0.
5679 - precision: 0.6649 - recall: 0.3822 - val_loss: 1.0577 - val_acc: 0.625
0 - val_precision: 0.7261 - val_recall: 0.4495
Epoch 52/100
12/12 [==============================] - 84s 7s/step - loss: 1.0751 - acc: 0.
5835 - precision: 0.7042 - recall: 0.3819 - val_loss: 0.9550 - val_acc: 0.632
3 - val_precision: 0.8013 - val_recall: 0.4703
Epoch 53/100
12/12 [==============================] - 83s 7s/step - loss: 0.9949 - acc: 0.
6180 - precision: 0.7557 - recall: 0.4295 - val_loss: 0.9898 - val_acc: 0.632
6 - val_precision: 0.7307 - val_recall: 0.5364
Epoch 54/100
12/12 [==============================] - 82s 7s/step - loss: 1.0207 - acc: 0.
5964 - precision: 0.7449 - recall: 0.4453 - val_loss: 0.9379 - val_acc: 0.646
9 - val_precision: 0.7474 - val_recall: 0.5443
Epoch 55/100
12/12 [==============================] - 82s 7s/step - loss: 0.9407 - acc: 0.
6448 - precision: 0.7431 - recall: 0.4685 - val_loss: 0.9085 - val_acc: 0.647
4 - val_precision: 0.7819 - val_recall: 0.5297
Epoch 56/100
12/12 [==============================] - 81s 7s/step - loss: 1.0131 - acc: 0.
6440 - precision: 0.6879 - recall: 0.4529 - val_loss: 1.0041 - val_acc: 0.602
6 - val_precision: 0.6926 - val_recall: 0.4997
Epoch 57/100
12/12 [==============================] - 82s 7s/step - loss: 1.0573 - acc: 0.
5945 - precision: 0.6786 - recall: 0.4007 - val_loss: 1.0654 - val_acc: 0.567
0 - val_precision: 0.6441 - val_recall: 0.4490
```

```
Epoch 58/100
12/12 [==============================] - 83s 7s/step - loss: 1.0266 - acc: 0.
6250 - precision: 0.7574 - recall: 0.4792 - val_loss: 1.0254 - val_acc: 0.602
6 - val_precision: 0.6718 - val_recall: 0.4995
Epoch 59/100
12/12 [==============================] - 82s 7s/step - loss: 1.0629 - acc: 0.
5838 - precision: 0.6720 - recall: 0.4844 - val_loss: 0.9594 - val_acc: 0.646
6 - val_precision: 0.7353 - val_recall: 0.5148
Epoch 60/100
12/12 [==============================] - 82s 7s/step - loss: 1.1706 - acc: 0.
5496 - precision: 0.6628 - recall: 0.4077 - val_loss: 0.9401 - val_acc: 0.624
7 - val_precision: 0.7821 - val_recall: 0.5294
Epoch 61/100
12/12 [==============================] - 82s 7s/step - loss: 1.0560 - acc: 0.
5394 - precision: 0.6578 - recall: 0.4372 - val_loss: 0.9492 - val_acc: 0.661
4 - val_precision: 0.7238 - val_recall: 0.5580
Epoch 62/100
12/12 [==============================] - 83s 7s/step - loss: 0.9551 - acc: 0.
6406 - precision: 0.7597 - recall: 0.4896 - val_loss: 0.9198 - val_acc: 0.669
5 - val_precision: 0.7603 - val_recall: 0.5154
Epoch 63/100
12/12 [==============================] - 82s 7s/step - loss: 0.9547 - acc: 0.
6180 - precision: 0.7030 - recall: 0.4609 - val_loss: 0.9439 - val_acc: 0.632
8 - val_precision: 0.6820 - val_recall: 0.5221
Epoch 64/100
12/12 [==============================] - 81s 7s/step - loss: 0.9761 - acc: 0.
6335 - precision: 0.7094 - recall: 0.4817 - val_loss: 0.9467 - val_acc: 0.655
2 - val_precision: 0.7198 - val_recall: 0.5448
Epoch 65/100
12/12 [==============================] - 81s 7s/step - loss: 0.9414 - acc: 0.
6519 - precision: 0.7360 - recall: 0.4948 - val_loss: 0.9925 - val_acc: 0.580
5 - val_precision: 0.6397 - val_recall: 0.4849
Epoch 66/100
12/12 [==============================] - 80s 7s/step - loss: 0.9581 - acc: 0.
6466 - precision: 0.7410 - recall: 0.4948 - val_loss: 0.9849 - val_acc: 0.625
0 - val_precision: 0.7076 - val_recall: 0.5148
Epoch 67/100
12/12 [==============================] - 80s 7s/step - loss: 0.9648 - acc: 0.
6302 - precision: 0.7132 - recall: 0.4896 - val_loss: 0.9529 - val_acc: 0.603
7 - val_precision: 0.6877 - val_recall: 0.5300
Epoch 68/100
12/12 [==============================] - 79s 7s/step - loss: 0.9092 - acc: 0.
6446 - precision: 0.7375 - recall: 0.5026 - val_loss: 0.9603 - val_acc: 0.667
7 - val_precision: 0.7194 - val_recall: 0.5502
Epoch 69/100
12/12 [==============================] - 82s 7s/step - loss: 0.9202 - acc: 0.
6470 - precision: 0.7220 - recall: 0.5185 - val_loss: 0.9427 - val_acc: 0.661
7 - val_precision: 0.7445 - val_recall: 0.6023
Epoch 70/100
12/12 [==============================] - 81s 7s/step - loss: 0.9292 - acc: 0.
6626 - precision: 0.7406 - recall: 0.5135 - val_loss: 1.0340 - val_acc: 0.610
2 - val_precision: 0.6605 - val_recall: 0.5289
Epoch 71/100
12/12 [==============================] - 84s 7s/step - loss: 0.8843 - acc: 0.
6859 - precision: 0.7569 - recall: 0.5343 - val_loss: 0.9717 - val_acc: 0.639
6 - val_precision: 0.6559 - val_recall: 0.5589
Epoch 72/100
```

```
12/12 [==============================] - 131s 11s/step - loss: 0.9368 - acc:
0.6519 - precision: 0.7168 - recall: 0.5053 - val_loss: 0.9263 - val_acc: 0.6
250 - val_precision: 0.7143 - val_recall: 0.5448
Epoch 73/100
12/12 [==============================] - 165s 14s/step - loss: 0.9409 - acc:
0.6797 - precision: 0.7414 - recall: 0.5469 - val_loss: 0.9450 - val_acc: 0.6
763 - val_precision: 0.7291 - val_recall: 0.5521
Epoch 74/100
12/12 [==============================] - 157s 13s/step - loss: 0.8911 - acc:
0.6387 - precision: 0.7440 - recall: 0.5051 - val_loss: 0.9668 - val_acc: 0.6
404 - val_precision: 0.6713 - val_recall: 0.5524
Epoch 75/100
12/12 [==============================] - 158s 13s/step - loss: 0.8729 - acc:
0.6609 - precision: 0.7314 - recall: 0.5318 - val_loss: 0.9391 - val_acc: 0.5
880 - val_precision: 0.6466 - val_recall: 0.5148
Epoch 76/100
12/12 [==============================] - 159s 13s/step - loss: 0.9177 - acc:
0.6120 - precision: 0.6920 - recall: 0.5234 - val_loss: 1.0650 - val_acc: 0.5
815 - val_precision: 0.6419 - val_recall: 0.5003
Epoch 77/100
12/12 [==============================] - 163s 14s/step - loss: 0.8919 - acc:
0.6754 - precision: 0.7766 - recall: 0.5624 - val_loss: 1.0347 - val_acc: 0.6
533 - val_precision: 0.6960 - val_recall: 0.5729
Epoch 78/100
12/12 [==============================] - 164s 14s/step - loss: 0.9409 - acc:
0.6049 - precision: 0.6971 - recall: 0.5001 - val_loss: 1.0664 - val_acc: 0.6
169 - val_precision: 0.6485 - val_recall: 0.5146
Epoch 79/100
12/12 [==============================] - 155s 13s/step - loss: 0.8490 - acc:
0.6573 - precision: 0.7209 - recall: 0.5341 - val_loss: 0.9413 - val_acc: 0.6
625 - val_precision: 0.7444 - val_recall: 0.5964
Epoch 80/100
12/12 [==============================] - 161s 13s/step - loss: 0.9385 - acc:
0.6441 - precision: 0.7113 - recall: 0.5527 - val_loss: 0.9845 - val_acc: 0.6
180 - val_precision: 0.6900 - val_recall: 0.5732
Epoch 81/100
12/12 [==============================] - 160s 13s/step - loss: 0.8797 - acc:
0.7040 - precision: 0.7817 - recall: 0.5810 - val_loss: 0.9576 - val_acc: 0.6
471 - val_precision: 0.7152 - val_recall: 0.5734
Epoch 82/100
12/12 [==============================] - 166s 14s/step - loss: 0.9156 - acc:
0.6516 - precision: 0.7199 - recall: 0.5419 - val_loss: 0.9609 - val_acc: 0.6
166 - val_precision: 0.6934 - val_recall: 0.5278
Epoch 83/100
12/12 [==============================] - 170s 14s/step - loss: 0.8950 - acc:
0.6510 - precision: 0.7555 - recall: 0.5521 - val_loss: 0.9991 - val_acc: 0.6
245 - val_precision: 0.6776 - val_recall: 0.5583
Epoch 84/100
12/12 [==============================] - 167s 14s/step - loss: 0.9230 - acc:
0.6632 - precision: 0.7297 - recall: 0.5554 - val_loss: 0.9758 - val_acc: 0.6
542 - val_precision: 0.6915 - val_recall: 0.5734
Epoch 85/100
12/12 [==============================] - 162s 14s/step - loss: 0.8586 - acc:
0.6927 - precision: 0.7622 - recall: 0.5677 - val_loss: 0.9533 - val_acc: 0.6
250 - val_precision: 0.6777 - val_recall: 0.5518
Epoch 86/100
12/12 [==============================] - 164s 14s/step - loss: 0.8159 - acc:
```

```
0.6908 - precision: 0.7783 - recall: 0.5888 - val_loss: 0.9839 - val_acc: 0.6
479 - val_precision: 0.6853 - val_recall: 0.5740
Epoch 87/100
12/12 [==============================] - 163s 14s/step - loss: 0.8461 - acc:
0.7025 - precision: 0.7619 - recall: 0.5999 - val_loss: 0.9146 - val_acc: 0.6
768 - val_precision: 0.7191 - val_recall: 0.5818
Epoch 88/100
12/12 [==============================] - 162s 14s/step - loss: 0.7980 - acc:
0.6927 - precision: 0.7839 - recall: 0.5833 - val_loss: 0.9184 - val_acc: 0.6
833 - val_precision: 0.7093 - val_recall: 0.5942
Epoch 89/100
12/12 [==============================] - 162s 14s/step - loss: 0.8901 - acc:
0.6516 - precision: 0.7339 - recall: 0.5548 - val_loss: 0.9394 - val_acc: 0.6
388 - val_precision: 0.7099 - val_recall: 0.5432
Epoch 90/100
12/12 [==============================] - 161s 13s/step - loss: 0.8050 - acc:
0.6988 - precision: 0.7907 - recall: 0.5835 - val_loss: 0.9539 - val_acc: 0.6
253 - val_precision: 0.6726 - val_recall: 0.5740
Epoch 91/100
12/12 [==============================] - 164s 14s/step - loss: 0.7719 - acc:
0.7092 - precision: 0.7818 - recall: 0.5969 - val_loss: 1.0155 - val_acc: 0.6
239 - val_precision: 0.6708 - val_recall: 0.5869
Epoch 92/100
12/12 [==============================] - 159s 13s/step - loss: 0.8346 - acc:
0.7145 - precision: 0.7595 - recall: 0.6098 - val_loss: 0.9758 - val_acc: 0.6
188 - val_precision: 0.6888 - val_recall: 0.5675
Epoch 93/100
12/12 [==============================] - 160s 13s/step - loss: 0.8239 - acc:
0.6777 - precision: 0.7604 - recall: 0.5783 - val_loss: 0.9669 - val_acc: 0.6
547 - val_precision: 0.7004 - val_recall: 0.5521
Epoch 94/100
12/12 [==============================] - 167s 14s/step - loss: 0.8148 - acc:
0.7109 - precision: 0.7819 - recall: 0.6094 - val_loss: 0.9669 - val_acc: 0.6
166 - val_precision: 0.6514 - val_recall: 0.5508
Epoch 95/100
12/12 [==============================] - 160s 13s/step - loss: 0.7690 - acc:
0.7290 - precision: 0.7963 - recall: 0.6263 - val_loss: 0.9907 - val_acc: 0.6
247 - val_precision: 0.6453 - val_recall: 0.5880
Epoch 96/100
12/12 [==============================] - 156s 13s/step - loss: 0.8115 - acc:
0.6727 - precision: 0.7500 - recall: 0.5993 - val_loss: 0.9343 - val_acc: 0.6
477 - val_precision: 0.7154 - val_recall: 0.5888
Epoch 97/100
12/12 [==============================] - 160s 13s/step - loss: 0.7609 - acc:
0.7161 - precision: 0.7820 - recall: 0.6354 - val_loss: 0.9825 - val_acc: 0.6
258 - val_precision: 0.6499 - val_recall: 0.5742
Epoch 98/100
12/12 [==============================] - 159s 13s/step - loss: 0.7765 - acc:
0.6962 - precision: 0.7615 - recall: 0.6386 - val_loss: 0.9483 - val_acc: 0.6
317 - val_precision: 0.6746 - val_recall: 0.5653
Epoch 99/100
12/12 [==============================] - 158s 13s/step - loss: 0.7082 - acc:
0.7250 - precision: 0.7790 - recall: 0.6361 - val_loss: 1.0253 - val_acc: 0.6
031 - val_precision: 0.6665 - val_recall: 0.5737
Epoch 100/100
12/12 [==============================] - 159s 13s/step - loss: 0.7488 - acc:
```

```
0.7200 - precision: 0.7802 - recall: 0.6336 - val_loss: 0.9212 - val_acc: 0.6
183 - val_precision: 0.6329 - val_recall: 0.5157
```

In [10]:
```
test_steps_per_epoch = np.math.ceil(test_set.samples / test_set.batch_size)
predictions = classifier.predict_generator(test_set, steps=test_steps_per_epoc
h)
predicted_classes = np.argmax(predictions, axis=1)
```

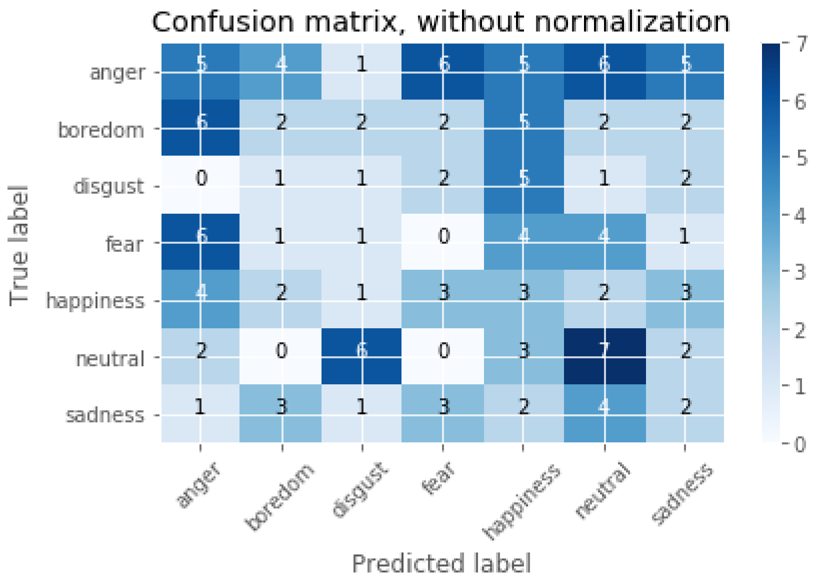
In [11]:
```
true_classes = test_set.classes
class_labels = list(test_set.class_indices.keys())
```

In [12]:
```
import sklearn.metrics as metrics
report = metrics.classification_report(true_classes, predicted_classes, target
_names=class_labels)
print(report)
```

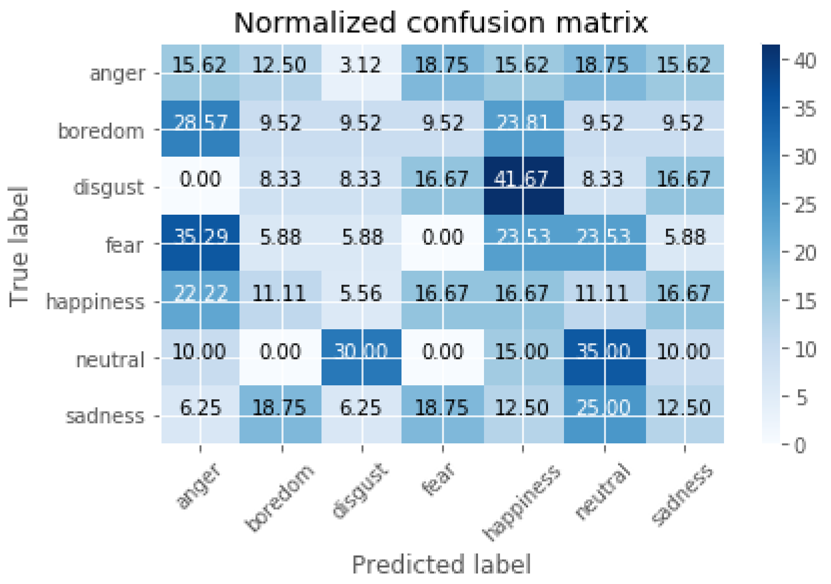
```
             precision    recall  f1-score   support

      anger       0.21      0.16      0.18        32
    boredom       0.15      0.10      0.12        21
    disgust       0.08      0.08      0.08        12
       fear       0.00      0.00      0.00        17
  happiness       0.11      0.17      0.13        18
    neutral       0.27      0.35      0.30        20
    sadness       0.12      0.12      0.12        16

avg / total       0.15      0.15      0.14       136
```

In [13]:
```
import matplotlib.pyplot as plt
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting normalize=True.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]*100
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap, aspect = 'auto')
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    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')

# Compute confusion matrix
cnf_matrix = metrics.confusion_matrix(true_classes, predicted_classes)
np.set_printoptions(precision=4)

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_labels,
                      title='Confusion matrix, without normalization')
plt.savefig("non_normalized_confusion_matrix_cnn_lstm.png")
plt.show()
# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_labels, normalize=True,
                      title='Normalized confusion matrix')
plt.savefig("normalized_confusion_matrix_cnn_lstm.png")
plt.show()
```

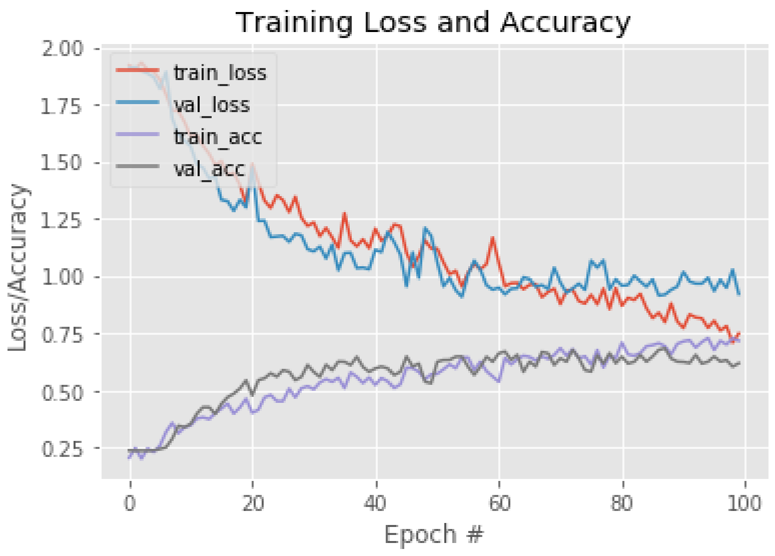
```
Confusion matrix, without normalization
[[5 4 1 6 5 6 5]
 [6 2 2 2 5 2 2]
 [0 1 1 2 5 1 2]
 [6 1 1 0 4 4 1]
 [4 2 1 3 3 2 3]
 [2 0 6 0 3 7 2]
 [1 3 1 3 2 4 2]]
```



```
Normalized confusion matrix
[[15.625   12.5      3.125   18.75    15.625   18.75    15.625 ]
 [28.5714  9.5238  9.5238  9.5238 23.8095  9.5238  9.5238]
 [ 0.       8.3333  8.3333 16.6667 41.6667  8.3333 16.6667]
 [35.2941  5.8824  5.8824  0.      23.5294 23.5294  5.8824]
 [22.2222 11.1111  5.5556 16.6667 16.6667 11.1111 16.6667]
 [10.       0.      30.       0.      15.      35.      10.     ]
 [ 6.25    18.75     6.25    18.75    12.5     25.      12.5    ]]
```



In [14]:
```
import matplotlib.pyplot as plt
plt.style.use("ggplot")
plt.figure()
N = 100
plt.plot(np.arange(0, N), results.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), results.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), results.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), results.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="upper left")
plt.savefig("plot_cnn_lstm.png")
```


Training Loss and Accuracy
Epoch #
Loss/Accuracy
train_loss
val_loss
train_acc
val_acc