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(Dense(output_dim = 128, activation = 'relu'))
classifier.add(Dropout(rate = 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
_________________________________________________________________
dense_1 (Dense)              (None, 128)               2097280
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
=================================================================
Total params: 2,173,831
Trainable params: 2,173,831
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 360 images belonging to 7 classes.
Found 120 images belonging to 7 classes.
```

In [4]:
```
results = classifier.fit_generator(training_set,
                         samples_per_epoch = 360,
                         nb_epoch = 100,
                         validation_data = test_set,
                         nb_val_samples = 120)
```

```
Epoch 1/100
11/11 [==============================] - 146s 13s/step - loss: 1.9613 - acc:
0.2022 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9169 - val_
acc: 0.2500 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/100
11/11 [==============================] - 142s 13s/step - loss: 1.9138 - acc:
0.2528 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8880 - val_
acc: 0.2500 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 3/100
11/11 [==============================] - 145s 13s/step - loss: 1.8884 - acc:
0.2664 - precision: 0.0915 - recall: 0.0086 - val_loss: 1.8845 - val_acc: 0.2
500 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 4/100
11/11 [==============================] - 142s 13s/step - loss: 1.8896 - acc:
0.2393 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8334 - val_
acc: 0.2500 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 5/100
11/11 [==============================] - 137s 12s/step - loss: 1.8750 - acc:
0.2574 - precision: 0.3684 - recall: 0.0115 - val_loss: 1.8026 - val_acc: 0.3
250 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 6/100
11/11 [==============================] - 143s 13s/step - loss: 1.7972 - acc:
0.2986 - precision: 0.1372 - recall: 0.0114 - val_loss: 1.7260 - val_acc: 0.3
583 - val_precision: 0.5800 - val_recall: 0.0250
Epoch 7/100
11/11 [==============================] - 148s 13s/step - loss: 1.8252 - acc:
0.2727 - precision: 0.2273 - recall: 0.0256 - val_loss: 1.7198 - val_acc: 0.3
750 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 8/100
11/11 [==============================] - 141s 13s/step - loss: 1.7114 - acc:
0.3521 - precision: 0.4477 - recall: 0.0507 - val_loss: 1.6737 - val_acc: 0.3
667 - val_precision: 0.6754 - val_recall: 0.0833
Epoch 9/100
11/11 [==============================] - 142s 13s/step - loss: 1.6942 - acc:
0.3300 - precision: 0.6886 - recall: 0.0593 - val_loss: 1.6235 - val_acc: 0.3
667 - val_precision: 0.5689 - val_recall: 0.0250
Epoch 10/100
11/11 [==============================] - 144s 13s/step - loss: 1.6777 - acc:
0.3472 - precision: 0.7526 - recall: 0.0907 - val_loss: 1.6779 - val_acc: 0.3
250 - val_precision: 0.5036 - val_recall: 0.0333
Epoch 11/100
11/11 [==============================] - 138s 13s/step - loss: 1.6771 - acc:
0.3446 - precision: 0.5770 - recall: 0.1086 - val_loss: 1.5952 - val_acc: 0.4
083 - val_precision: 0.6776 - val_recall: 0.1083
Epoch 12/100
11/11 [==============================] - 154s 14s/step - loss: 1.6809 - acc:
0.3381 - precision: 0.6502 - recall: 0.0682 - val_loss: 1.5851 - val_acc: 0.3
583 - val_precision: 0.7222 - val_recall: 0.1250
Epoch 13/100
11/11 [==============================] - 149s 14s/step - loss: 1.5695 - acc:
0.3571 - precision: 0.7043 - recall: 0.1042 - val_loss: 1.5439 - val_acc: 0.3
667 - val_precision: 0.6020 - val_recall: 0.1917
Epoch 14/100
11/11 [==============================] - 145s 13s/step - loss: 1.6928 - acc:
0.3415 - precision: 0.6919 - recall: 0.0936 - val_loss: 1.5909 - val_acc: 0.3
750 - val_precision: 0.6783 - val_recall: 0.1000
Epoch 15/100
```

```
11/11 [==============================] - 143s 13s/step - loss: 1.5620 - acc:
0.3843 - precision: 0.6187 - recall: 0.1307 - val_loss: 1.5593 - val_acc: 0.3
583 - val_precision: 0.6856 - val_recall: 0.1667
Epoch 16/100
11/11 [==============================] - 150s 14s/step - loss: 1.5531 - acc:
0.4034 - precision: 0.6141 - recall: 0.1136 - val_loss: 1.5319 - val_acc: 0.3
750 - val_precision: 0.6791 - val_recall: 0.1417
Epoch 17/100
11/11 [==============================] - 144s 13s/step - loss: 1.5772 - acc:
0.4120 - precision: 0.7405 - recall: 0.1558 - val_loss: 1.5402 - val_acc: 0.4
000 - val_precision: 0.7022 - val_recall: 0.0917
Epoch 18/100
11/11 [==============================] - 143s 13s/step - loss: 1.5465 - acc:
0.3500 - precision: 0.6926 - recall: 0.0936 - val_loss: 1.5426 - val_acc: 0.3
583 - val_precision: 0.5958 - val_recall: 0.2083
Epoch 19/100
11/11 [==============================] - 143s 13s/step - loss: 1.5614 - acc:
0.3986 - precision: 0.5216 - recall: 0.1086 - val_loss: 1.5088 - val_acc: 0.3
833 - val_precision: 0.6568 - val_recall: 0.1583
Epoch 20/100
11/11 [==============================] - 148s 13s/step - loss: 1.5404 - acc:
0.3949 - precision: 0.5578 - recall: 0.1108 - val_loss: 1.4986 - val_acc: 0.4
333 - val_precision: 0.6823 - val_recall: 0.1750
Epoch 21/100
11/11 [==============================] - 141s 13s/step - loss: 1.5228 - acc:
0.4058 - precision: 0.6625 - recall: 0.1316 - val_loss: 1.4773 - val_acc: 0.3
667 - val_precision: 0.6301 - val_recall: 0.1833
Epoch 22/100
11/11 [==============================] - 143s 13s/step - loss: 1.5104 - acc:
0.3807 - precision: 0.6965 - recall: 0.1420 - val_loss: 1.4595 - val_acc: 0.3
833 - val_precision: 0.6726 - val_recall: 0.1750
Epoch 23/100
11/11 [==============================] - 140s 13s/step - loss: 1.4491 - acc:
0.4442 - precision: 0.6091 - recall: 0.1785 - val_loss: 1.4971 - val_acc: 0.3
917 - val_precision: 0.6458 - val_recall: 0.1917
Epoch 24/100
11/11 [==============================] - 144s 13s/step - loss: 1.5156 - acc:
0.4556 - precision: 0.6296 - recall: 0.1442 - val_loss: 1.4656 - val_acc: 0.4
250 - val_precision: 0.6908 - val_recall: 0.1833
Epoch 25/100
11/11 [==============================] - 144s 13s/step - loss: 1.4699 - acc:
0.3851 - precision: 0.5969 - recall: 0.1343 - val_loss: 1.4875 - val_acc: 0.3
667 - val_precision: 0.7742 - val_recall: 0.1167
Epoch 26/100
11/11 [==============================] - 145s 13s/step - loss: 1.4740 - acc:
0.3700 - precision: 0.6646 - recall: 0.1393 - val_loss: 1.4623 - val_acc: 0.4
167 - val_precision: 0.7255 - val_recall: 0.1833
Epoch 27/100
11/11 [==============================] - 146s 13s/step - loss: 1.5046 - acc:
0.4347 - precision: 0.5685 - recall: 0.1591 - val_loss: 1.4965 - val_acc: 0.3
833 - val_precision: 0.6099 - val_recall: 0.2167
Epoch 28/100
11/11 [==============================] - 137s 12s/step - loss: 1.4656 - acc:
0.4050 - precision: 0.6952 - recall: 0.1604 - val_loss: 1.5307 - val_acc: 0.3
917 - val_precision: 0.5992 - val_recall: 0.2250
Epoch 29/100
11/11 [==============================] - 140s 13s/step - loss: 1.4774 - acc:
```

```
0.4278 - precision: 0.5232 - recall: 0.1235 - val_loss: 1.4411 - val_acc: 0.4
167 - val_precision: 0.7331 - val_recall: 0.1667
Epoch 30/100
11/11 [==============================] - 141s 13s/step - loss: 1.4658 - acc:
0.4080 - precision: 0.6595 - recall: 0.1772 - val_loss: 1.4608 - val_acc: 0.3
917 - val_precision: 0.6278 - val_recall: 0.1833
Epoch 31/100
11/11 [==============================] - 141s 13s/step - loss: 1.4240 - acc:
0.4379 - precision: 0.5851 - recall: 0.1686 - val_loss: 1.4605 - val_acc: 0.3
833 - val_precision: 0.6785 - val_recall: 0.2167
Epoch 32/100
11/11 [==============================] - 138s 13s/step - loss: 1.3805 - acc:
0.4564 - precision: 0.6547 - recall: 0.1744 - val_loss: 1.4137 - val_acc: 0.4
250 - val_precision: 0.6396 - val_recall: 0.2500
Epoch 33/100
11/11 [==============================] - 108s 10s/step - loss: 1.4697 - acc:
0.3737 - precision: 0.5368 - recall: 0.1543 - val_loss: 1.5325 - val_acc: 0.3
583 - val_precision: 0.5288 - val_recall: 0.2250
Epoch 34/100
11/11 [==============================] - 112s 10s/step - loss: 1.3882 - acc:
0.4460 - precision: 0.6754 - recall: 0.2017 - val_loss: 1.4267 - val_acc: 0.3
917 - val_precision: 0.6856 - val_recall: 0.2167
Epoch 35/100
11/11 [==============================] - 107s 10s/step - loss: 1.4786 - acc:
0.4075 - precision: 0.6619 - recall: 0.1604 - val_loss: 1.4355 - val_acc: 0.4
167 - val_precision: 0.7452 - val_recall: 0.1917
Epoch 36/100
11/11 [==============================] - 113s 10s/step - loss: 1.3895 - acc:
0.4318 - precision: 0.6964 - recall: 0.1619 - val_loss: 1.4341 - val_acc: 0.4
083 - val_precision: 0.6497 - val_recall: 0.2500
Epoch 37/100
11/11 [==============================] - 115s 10s/step - loss: 1.4072 - acc:
0.4064 - precision: 0.6223 - recall: 0.1686 - val_loss: 1.4769 - val_acc: 0.4
250 - val_precision: 0.7197 - val_recall: 0.2583
Epoch 38/100
11/11 [==============================] - 119s 11s/step - loss: 1.3615 - acc:
0.4985 - precision: 0.6891 - recall: 0.2149 - val_loss: 1.5087 - val_acc: 0.4
333 - val_precision: 0.5465 - val_recall: 0.2917
Epoch 39/100
11/11 [==============================] - 112s 10s/step - loss: 1.3673 - acc:
0.4628 - precision: 0.5673 - recall: 0.1886 - val_loss: 1.5767 - val_acc: 0.4
250 - val_precision: 0.5008 - val_recall: 0.2667
Epoch 40/100
11/11 [==============================] - 112s 10s/step - loss: 1.4807 - acc:
0.4057 - precision: 0.4901 - recall: 0.1543 - val_loss: 1.4012 - val_acc: 0.4
250 - val_precision: 0.7343 - val_recall: 0.1833
Epoch 41/100
11/11 [==============================] - 113s 10s/step - loss: 1.3807 - acc:
0.4375 - precision: 0.6802 - recall: 0.1591 - val_loss: 1.4150 - val_acc: 0.3
833 - val_precision: 0.7929 - val_recall: 0.1917
Epoch 42/100
11/11 [==============================] - 108s 10s/step - loss: 1.3449 - acc:
0.4420 - precision: 0.7829 - recall: 0.2385 - val_loss: 1.3954 - val_acc: 0.3
833 - val_precision: 0.7145 - val_recall: 0.2333
Epoch 43/100
11/11 [==============================] - 113s 10s/step - loss: 1.3668 - acc:
0.4432 - precision: 0.6241 - recall: 0.2244 - val_loss: 1.4005 - val_acc: 0.4
```

```
250 - val_precision: 0.7459 - val_recall: 0.2167
Epoch 44/100
11/11 [==============================] - 112s 10s/step - loss: 1.3515 - acc:
0.4571 - precision: 0.6633 - recall: 0.2128 - val_loss: 1.3994 - val_acc: 0.4
083 - val_precision: 0.8218 - val_recall: 0.1583
Epoch 45/100
11/11 [==============================] - 107s 10s/step - loss: 1.3884 - acc:
0.4461 - precision: 0.6821 - recall: 0.1665 - val_loss: 1.4153 - val_acc: 0.4
083 - val_precision: 0.7783 - val_recall: 0.1750
Epoch 46/100
11/11 [==============================] - 112s 10s/step - loss: 1.3256 - acc:
0.4886 - precision: 0.6670 - recall: 0.2386 - val_loss: 1.4344 - val_acc: 0.4
250 - val_precision: 0.6089 - val_recall: 0.2833
Epoch 47/100
11/11 [==============================] - 109s 10s/step - loss: 1.3391 - acc:
0.4863 - precision: 0.6692 - recall: 0.2700 - val_loss: 1.4556 - val_acc: 0.4
333 - val_precision: 0.6172 - val_recall: 0.2667
Epoch 48/100
11/11 [==============================] - 108s 10s/step - loss: 1.3719 - acc:
0.4728 - precision: 0.7248 - recall: 0.1843 - val_loss: 1.4105 - val_acc: 0.4
083 - val_precision: 0.8586 - val_recall: 0.1417
Epoch 49/100
11/11 [==============================] - 125s 11s/step - loss: 1.4430 - acc:
0.4636 - precision: 0.6439 - recall: 0.1707 - val_loss: 1.4030 - val_acc: 0.4
333 - val_precision: 0.6310 - val_recall: 0.2583
Epoch 50/100
11/11 [==============================] - 179s 16s/step - loss: 1.3151 - acc:
0.4722 - precision: 0.8132 - recall: 0.1879 - val_loss: 1.4308 - val_acc: 0.4
333 - val_precision: 0.6296 - val_recall: 0.2833
Epoch 51/100
11/11 [==============================] - 176s 16s/step - loss: 1.2921 - acc:
0.4893 - precision: 0.6823 - recall: 0.2529 - val_loss: 1.3931 - val_acc: 0.4
250 - val_precision: 0.6771 - val_recall: 0.2583
Epoch 52/100
11/11 [==============================] - 182s 17s/step - loss: 1.2901 - acc:
0.4716 - precision: 0.6937 - recall: 0.2614 - val_loss: 1.3521 - val_acc: 0.4
500 - val_precision: 0.7400 - val_recall: 0.2833
Epoch 53/100
11/11 [==============================] - 179s 16s/step - loss: 1.2590 - acc:
0.5029 - precision: 0.6986 - recall: 0.2924 - val_loss: 1.3852 - val_acc: 0.4
917 - val_precision: 0.7120 - val_recall: 0.3083
Epoch 54/100
11/11 [==============================] - 176s 16s/step - loss: 1.3006 - acc:
0.4987 - precision: 0.6542 - recall: 0.2372 - val_loss: 1.3691 - val_acc: 0.4
250 - val_precision: 0.7202 - val_recall: 0.1917
Epoch 55/100
11/11 [==============================] - 180s 16s/step - loss: 1.2731 - acc:
0.4972 - precision: 0.7281 - recall: 0.2557 - val_loss: 1.3319 - val_acc: 0.4
667 - val_precision: 0.7357 - val_recall: 0.2583
Epoch 56/100
11/11 [==============================] - 180s 16s/step - loss: 1.1942 - acc:
0.5335 - precision: 0.7432 - recall: 0.3685 - val_loss: 1.3260 - val_acc: 0.4
500 - val_precision: 0.7155 - val_recall: 0.3083
Epoch 57/100
11/11 [==============================] - 176s 16s/step - loss: 1.2499 - acc:
0.5093 - precision: 0.6405 - recall: 0.2565 - val_loss: 1.4002 - val_acc: 0.4
000 - val_precision: 0.6440 - val_recall: 0.2750
```

```
Epoch 58/100
11/11 [==============================] - 177s 16s/step - loss: 1.2803 - acc:
0.5415 - precision: 0.7502 - recall: 0.3252 - val_loss: 1.3640 - val_acc: 0.4
417 - val_precision: 0.6391 - val_recall: 0.2833
Epoch 59/100
11/11 [==============================] - 185s 17s/step - loss: 1.2435 - acc:
0.4943 - precision: 0.6369 - recall: 0.2557 - val_loss: 1.3089 - val_acc: 0.4
583 - val_precision: 0.7186 - val_recall: 0.2750
Epoch 60/100
11/11 [==============================] - 175s 16s/step - loss: 1.2338 - acc:
0.4987 - precision: 0.7210 - recall: 0.2879 - val_loss: 1.3271 - val_acc: 0.4
750 - val_precision: 0.7060 - val_recall: 0.2833
Epoch 61/100
11/11 [==============================] - 177s 16s/step - loss: 1.1752 - acc:
0.5593 - precision: 0.6980 - recall: 0.3550 - val_loss: 1.4305 - val_acc: 0.4
417 - val_precision: 0.6115 - val_recall: 0.3167
Epoch 62/100
11/11 [==============================] - 177s 16s/step - loss: 1.2603 - acc:
0.5343 - precision: 0.7123 - recall: 0.2864 - val_loss: 1.3921 - val_acc: 0.4
917 - val_precision: 0.7489 - val_recall: 0.3250
Epoch 63/100
11/11 [==============================] - 173s 16s/step - loss: 1.2948 - acc:
0.4600 - precision: 0.7139 - recall: 0.2664 - val_loss: 1.3294 - val_acc: 0.4
500 - val_precision: 0.7446 - val_recall: 0.2417
Epoch 64/100
11/11 [==============================] - 180s 16s/step - loss: 1.1942 - acc:
0.5483 - precision: 0.7791 - recall: 0.3011 - val_loss: 1.3626 - val_acc: 0.4
500 - val_precision: 0.6738 - val_recall: 0.2750
Epoch 65/100
11/11 [==============================] - 181s 16s/step - loss: 1.2158 - acc:
0.5606 - precision: 0.7354 - recall: 0.3371 - val_loss: 1.3624 - val_acc: 0.4
333 - val_precision: 0.7052 - val_recall: 0.2833
Epoch 66/100
11/11 [==============================] - 176s 16s/step - loss: 1.1262 - acc:
0.5465 - precision: 0.7288 - recall: 0.4080 - val_loss: 1.3355 - val_acc: 0.4
667 - val_precision: 0.7113 - val_recall: 0.3250
Epoch 67/100
11/11 [==============================] - 175s 16s/step - loss: 1.2000 - acc:
0.5337 - precision: 0.7298 - recall: 0.3269 - val_loss: 1.3371 - val_acc: 0.4
750 - val_precision: 0.7008 - val_recall: 0.2917
Epoch 68/100
11/11 [==============================] - 179s 16s/step - loss: 1.1833 - acc:
0.5171 - precision: 0.7161 - recall: 0.3293 - val_loss: 1.3315 - val_acc: 0.4
167 - val_precision: 0.7243 - val_recall: 0.2750
Epoch 69/100
11/11 [==============================] - 177s 16s/step - loss: 1.1375 - acc:
0.5899 - precision: 0.7215 - recall: 0.3443 - val_loss: 1.2721 - val_acc: 0.4
833 - val_precision: 0.7466 - val_recall: 0.3250
Epoch 70/100
11/11 [==============================] - 182s 17s/step - loss: 1.1638 - acc:
0.5312 - precision: 0.6828 - recall: 0.3324 - val_loss: 1.4408 - val_acc: 0.4
083 - val_precision: 0.6556 - val_recall: 0.3000
Epoch 71/100
11/11 [==============================] - 180s 16s/step - loss: 1.2764 - acc:
0.4909 - precision: 0.5626 - recall: 0.2744 - val_loss: 1.4598 - val_acc: 0.4
250 - val_precision: 0.6178 - val_recall: 0.2417
Epoch 72/100
```

```
11/11 [==============================] - 176s 16s/step - loss: 1.2208 - acc:
0.5086 - precision: 0.6921 - recall: 0.2879 - val_loss: 1.3056 - val_acc: 0.5
000 - val_precision: 0.8137 - val_recall: 0.2500
Epoch 73/100
11/11 [==============================] - 180s 16s/step - loss: 1.1771 - acc:
0.5408 - precision: 0.7443 - recall: 0.3121 - val_loss: 1.3356 - val_acc: 0.4
833 - val_precision: 0.6659 - val_recall: 0.3000
Epoch 74/100
11/11 [==============================] - 175s 16s/step - loss: 1.1719 - acc:
0.5208 - precision: 0.7198 - recall: 0.3350 - val_loss: 1.2998 - val_acc: 0.4
667 - val_precision: 0.6966 - val_recall: 0.3250
Epoch 75/100
11/11 [==============================] - 176s 16s/step - loss: 1.1838 - acc:
0.5568 - precision: 0.6933 - recall: 0.3409 - val_loss: 1.2644 - val_acc: 0.5
167 - val_precision: 0.6730 - val_recall: 0.2917
Epoch 76/100
11/11 [==============================] - 173s 16s/step - loss: 1.0893 - acc:
0.5724 - precision: 0.7167 - recall: 0.3935 - val_loss: 1.2687 - val_acc: 0.4
750 - val_precision: 0.6998 - val_recall: 0.3333
Epoch 77/100
11/11 [==============================] - 153s 14s/step - loss: 1.1383 - acc:
0.5507 - precision: 0.7343 - recall: 0.3558 - val_loss: 1.2671 - val_acc: 0.4
833 - val_precision: 0.6653 - val_recall: 0.3167
Epoch 78/100
11/11 [==============================] - 142s 13s/step - loss: 1.0761 - acc:
0.5535 - precision: 0.7259 - recall: 0.3657 - val_loss: 1.3143 - val_acc: 0.5
000 - val_precision: 0.6529 - val_recall: 0.3917
Epoch 79/100
11/11 [==============================] - 145s 13s/step - loss: 1.1250 - acc:
0.5284 - precision: 0.6915 - recall: 0.3523 - val_loss: 1.2692 - val_acc: 0.5
083 - val_precision: 0.6943 - val_recall: 0.3250
Epoch 80/100
11/11 [==============================] - 142s 13s/step - loss: 1.0292 - acc:
0.6168 - precision: 0.7507 - recall: 0.4322 - val_loss: 1.2632 - val_acc: 0.5
250 - val_precision: 0.7227 - val_recall: 0.3500
Epoch 81/100
11/11 [==============================] - 143s 13s/step - loss: 1.0100 - acc:
0.6080 - precision: 0.7549 - recall: 0.4375 - val_loss: 1.3038 - val_acc: 0.5
083 - val_precision: 0.6911 - val_recall: 0.4083
Epoch 82/100
11/11 [==============================] - 147s 13s/step - loss: 1.0562 - acc:
0.5358 - precision: 0.6849 - recall: 0.4072 - val_loss: 1.3580 - val_acc: 0.4
750 - val_precision: 0.6253 - val_recall: 0.3583
Epoch 83/100
11/11 [==============================] - 144s 13s/step - loss: 0.9226 - acc:
0.6500 - precision: 0.7882 - recall: 0.4621 - val_loss: 1.2727 - val_acc: 0.5
083 - val_precision: 0.6635 - val_recall: 0.3750
Epoch 84/100
11/11 [==============================] - 143s 13s/step - loss: 1.0107 - acc:
0.5915 - precision: 0.7267 - recall: 0.4722 - val_loss: 1.3630 - val_acc: 0.4
750 - val_precision: 0.6132 - val_recall: 0.3833
Epoch 85/100
11/11 [==============================] - 146s 13s/step - loss: 1.1186 - acc:
0.5330 - precision: 0.6765 - recall: 0.4022 - val_loss: 1.3294 - val_acc: 0.4
750 - val_precision: 0.6806 - val_recall: 0.3917
Epoch 86/100
11/11 [==============================] - 147s 13s/step - loss: 1.1267 - acc:
```

```
0.5192 - precision: 0.7014 - recall: 0.3399 - val_loss: 1.2972 - val_acc: 0.4
917 - val_precision: 0.7058 - val_recall: 0.3583
Epoch 87/100
11/11 [==============================] - 148s 13s/step - loss: 1.0461 - acc:
0.6165 - precision: 0.7555 - recall: 0.4261 - val_loss: 1.3205 - val_acc: 0.4
750 - val_precision: 0.6441 - val_recall: 0.3333
Epoch 88/100
11/11 [==============================] - 144s 13s/step - loss: 1.0018 - acc:
0.5943 - precision: 0.7839 - recall: 0.4200 - val_loss: 1.4124 - val_acc: 0.4
750 - val_precision: 0.6646 - val_recall: 0.3833
Epoch 89/100
11/11 [==============================] - 147s 13s/step - loss: 1.1697 - acc:
0.5465 - precision: 0.6615 - recall: 0.3573 - val_loss: 1.3465 - val_acc: 0.4
583 - val_precision: 0.6506 - val_recall: 0.3417
Epoch 90/100
11/11 [==============================] - 144s 13s/step - loss: 1.0884 - acc:
0.5899 - precision: 0.7153 - recall: 0.4135 - val_loss: 1.2781 - val_acc: 0.5
083 - val_precision: 0.6875 - val_recall: 0.3833
Epoch 91/100
11/11 [==============================] - 145s 13s/step - loss: 0.9850 - acc:
0.6049 - precision: 0.7347 - recall: 0.4396 - val_loss: 1.3467 - val_acc: 0.4
833 - val_precision: 0.6162 - val_recall: 0.3500
Epoch 92/100
11/11 [==============================] - 146s 13s/step - loss: 1.0727 - acc:
0.5694 - precision: 0.6589 - recall: 0.4051 - val_loss: 1.3170 - val_acc: 0.4
917 - val_precision: 0.6888 - val_recall: 0.3917
Epoch 93/100
11/11 [==============================] - 148s 13s/step - loss: 1.0636 - acc:
0.5994 - precision: 0.7677 - recall: 0.4261 - val_loss: 1.2820 - val_acc: 0.4
583 - val_precision: 0.6931 - val_recall: 0.3417
Epoch 94/100
11/11 [==============================] - 146s 13s/step - loss: 1.0396 - acc:
0.5736 - precision: 0.6967 - recall: 0.4057 - val_loss: 1.3228 - val_acc: 0.5
083 - val_precision: 0.6775 - val_recall: 0.3833
Epoch 95/100
11/11 [==============================] - 144s 13s/step - loss: 0.9730 - acc:
0.6385 - precision: 0.7858 - recall: 0.4792 - val_loss: 1.2685 - val_acc: 0.5
333 - val_precision: 0.6849 - val_recall: 0.3417
Epoch 96/100
11/11 [==============================] - 142s 13s/step - loss: 0.9108 - acc:
0.6578 - precision: 0.7487 - recall: 0.4813 - val_loss: 1.3118 - val_acc: 0.5
417 - val_precision: 0.5949 - val_recall: 0.3917
Epoch 97/100
11/11 [==============================] - 143s 13s/step - loss: 1.0452 - acc:
0.5936 - precision: 0.7045 - recall: 0.4849 - val_loss: 1.2662 - val_acc: 0.5
083 - val_precision: 0.7057 - val_recall: 0.3583
Epoch 98/100
11/11 [==============================] - 146s 13s/step - loss: 1.0468 - acc:
0.6080 - precision: 0.7329 - recall: 0.4432 - val_loss: 1.3177 - val_acc: 0.4
917 - val_precision: 0.6383 - val_recall: 0.3833
Epoch 99/100
11/11 [==============================] - 142s 13s/step - loss: 0.9975 - acc:
0.6176 - precision: 0.7387 - recall: 0.4309 - val_loss: 1.3810 - val_acc: 0.4
833 - val_precision: 0.6907 - val_recall: 0.4083
Epoch 100/100
11/11 [==============================] - 142s 13s/step - loss: 0.9511 - acc:
```

```
0.6279 - precision: 0.7773 - recall: 0.5135 - val_loss: 1.3155 - val_acc: 0.5
167 - val_precision: 0.6612 - val_recall: 0.3583
```

In [5]:
```
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)
```

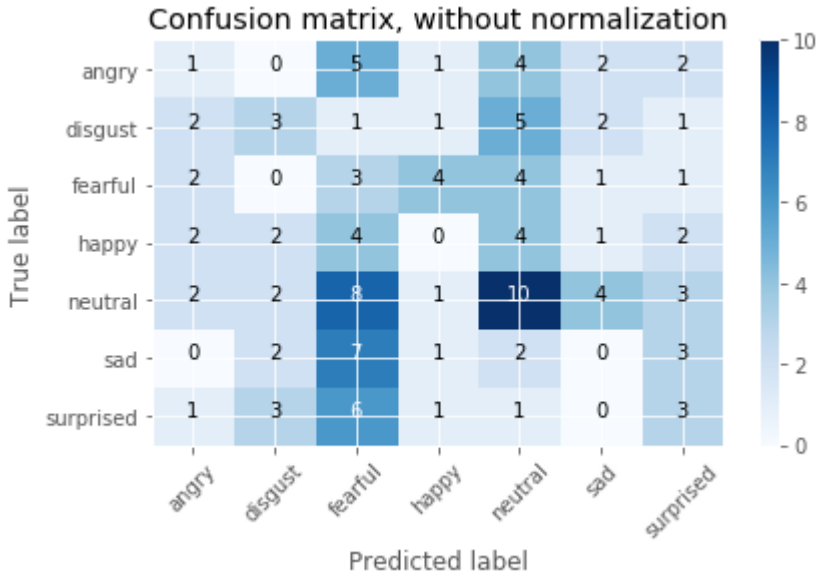
In [6]:
```
true_classes = test_set.classes
class_labels = list(test_set.class_indices.keys())
```

In [7]:
```
import sklearn.metrics as metrics
report = metrics.classification_report(true_classes, predicted_classes, target
_names=class_labels)
print(report)
```

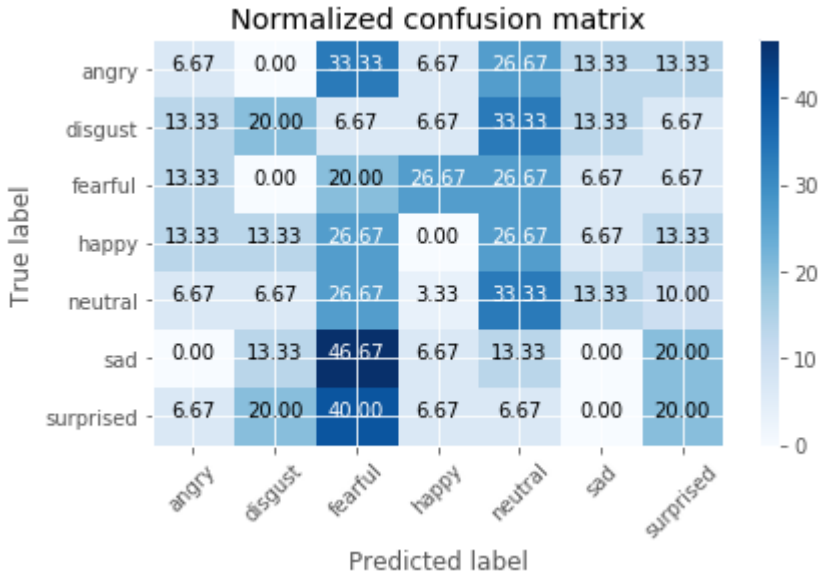
```
             precision    recall  f1-score   support

      angry       0.10      0.07      0.08        15
    disgust       0.25      0.20      0.22        15
    fearful       0.09      0.20      0.12        15
      happy       0.00      0.00      0.00        15
    neutral       0.33      0.33      0.33        30
        sad       0.00      0.00      0.00        15
  surprised       0.20      0.20      0.20        15

avg / total       0.16      0.17      0.16       120
```

In [10]:
```
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.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.png")
plt.show()
```

```
Confusion matrix, without normalization
[[ 1  0  5  1  4  2  2]
 [ 2  3  1  1  5  2  1]
 [ 2  0  3  4  4  1  1]
 [ 2  2  4  0  4  1  2]
 [ 2  2  8  1 10  4  3]
 [ 0  2  7  1  2  0  3]
 [ 1  3  6  1  1  0  3]]
```




```
Normalized confusion matrix
[[ 6.6667  0.     33.3333  6.6667 26.6667 13.3333 13.3333]
 [13.3333 20.      6.6667  6.6667 33.3333 13.3333  6.6667]
 [13.3333  0.     20.     26.6667 26.6667  6.6667  6.6667]
 [13.3333 13.3333 26.6667  0.     26.6667  6.6667 13.3333]
 [ 6.6667  6.6667 26.6667  3.3333 33.3333 13.3333 10.    ]
 [ 0.     13.3333 46.6667  6.6667 13.3333  0.     20.    ]
 [ 6.6667 20.     40.      6.6667  6.6667  0.     20.    ]]
```



In [11]:
```
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.png")
```