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 = 8, 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, 8)	1032

```
Total params: 2,173,960
Trainable params: 2,173,960
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 1080 images belonging to 8 classes.
Found 360 images belonging to 8 classes.
```

In [4]:
```
results = classifier.fit_generator(training_set,
                         samples_per_epoch = 1080,
                         nb_epoch = 100,
                         validation_data = test_set,
                         nb_val_samples = 360)
```

```
Epoch 1/100
33/33 [==============================] - 544s 16s/step - loss: 2.0757 - acc:
0.1361 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 2.0382 - val_
acc: 0.1889 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/100
33/33 [==============================] - 523s 16s/step - loss: 2.0212 - acc:
0.1834 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9979 - val_
acc: 0.2083 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 3/100
33/33 [==============================] - 524s 16s/step - loss: 1.9900 - acc:
0.2027 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8428 - val_
acc: 0.2361 - val_precision: 0.0800 - val_recall: 0.0028
Epoch 4/100
33/33 [==============================] - 528s 16s/step - loss: 1.9602 - acc:
0.2263 - precision: 0.2223 - recall: 0.0076 - val_loss: 1.8788 - val_acc: 0.2
444 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 5/100
33/33 [==============================] - 521s 16s/step - loss: 1.9196 - acc:
0.2572 - precision: 0.1008 - recall: 0.0050 - val_loss: 1.7784 - val_acc: 0.3
000 - val_precision: 0.2496 - val_recall: 0.0083
Epoch 6/100
33/33 [==============================] - 524s 16s/step - loss: 1.8731 - acc:
0.2705 - precision: 0.2556 - recall: 0.0133 - val_loss: 1.7799 - val_acc: 0.3
083 - val_precision: 0.3030 - val_recall: 0.0111
Epoch 7/100
33/33 [==============================] - 519s 16s/step - loss: 1.8443 - acc:
0.2914 - precision: 0.4041 - recall: 0.0170 - val_loss: 1.6935 - val_acc: 0.3
528 - val_precision: 0.7714 - val_recall: 0.0806
Epoch 8/100
33/33 [==============================] - 522s 16s/step - loss: 1.8267 - acc:
0.2964 - precision: 0.4007 - recall: 0.0379 - val_loss: 1.6081 - val_acc: 0.4
194 - val_precision: 0.8580 - val_recall: 0.0833
Epoch 9/100
33/33 [==============================] - 526s 16s/step - loss: 1.7909 - acc:
0.3144 - precision: 0.6268 - recall: 0.0432 - val_loss: 1.5703 - val_acc: 0.4
083 - val_precision: 0.7055 - val_recall: 0.1111
Epoch 10/100
33/33 [==============================] - 445s 13s/step - loss: 1.7027 - acc:
0.3276 - precision: 0.6447 - recall: 0.0824 - val_loss: 1.5271 - val_acc: 0.4
361 - val_precision: 0.6562 - val_recall: 0.1500
Epoch 11/100
33/33 [==============================] - 431s 13s/step - loss: 1.6921 - acc:
0.3551 - precision: 0.6272 - recall: 0.0994 - val_loss: 1.6006 - val_acc: 0.4
250 - val_precision: 0.7906 - val_recall: 0.0861
Epoch 12/100
33/33 [==============================] - 435s 13s/step - loss: 1.7011 - acc:
0.3532 - precision: 0.5550 - recall: 0.0783 - val_loss: 1.5561 - val_acc: 0.4
111 - val_precision: 0.8163 - val_recall: 0.1222
Epoch 13/100
33/33 [==============================] - 437s 13s/step - loss: 1.6782 - acc:
0.3665 - precision: 0.6540 - recall: 0.0950 - val_loss: 1.5159 - val_acc: 0.4
500 - val_precision: 0.6792 - val_recall: 0.1306
Epoch 14/100
33/33 [==============================] - 438s 13s/step - loss: 1.6783 - acc:
0.3627 - precision: 0.5840 - recall: 0.0938 - val_loss: 1.5488 - val_acc: 0.4
222 - val_precision: 0.6667 - val_recall: 0.1111
Epoch 15/100
```

```
33/33 [==============================] - 436s 13s/step - loss: 1.6190 - acc:
0.3791 - precision: 0.6233 - recall: 0.1225 - val_loss: 1.5438 - val_acc: 0.4
750 - val_precision: 0.8187 - val_recall: 0.1111
Epoch 16/100
33/33 [==============================] - 430s 13s/step - loss: 1.5990 - acc:
0.3949 - precision: 0.6134 - recall: 0.1326 - val_loss: 1.4697 - val_acc: 0.4
333 - val_precision: 0.7273 - val_recall: 0.1833
Epoch 17/100
33/33 [==============================] - 429s 13s/step - loss: 1.5613 - acc:
0.4145 - precision: 0.6334 - recall: 0.1376 - val_loss: 1.4509 - val_acc: 0.4
778 - val_precision: 0.5884 - val_recall: 0.2361
Epoch 18/100
33/33 [==============================] - 282s 9s/step - loss: 1.5753 - acc:
0.4044 - precision: 0.6442 - recall: 0.1373 - val_loss: 1.4317 - val_acc: 0.4
667 - val_precision: 0.6798 - val_recall: 0.2167
Epoch 19/100
33/33 [==============================] - 232s 7s/step - loss: 1.5658 - acc:
0.4053 - precision: 0.6224 - recall: 0.1487 - val_loss: 1.5572 - val_acc: 0.4
028 - val_precision: 0.6717 - val_recall: 0.1833
Epoch 20/100
33/33 [==============================] - 235s 7s/step - loss: 1.5542 - acc:
0.4012 - precision: 0.6592 - recall: 0.1569 - val_loss: 1.4077 - val_acc: 0.4
806 - val_precision: 0.6745 - val_recall: 0.2000
Epoch 21/100
33/33 [==============================] - 238s 7s/step - loss: 1.5005 - acc:
0.4258 - precision: 0.7042 - recall: 0.1947 - val_loss: 1.3900 - val_acc: 0.5
139 - val_precision: 0.8115 - val_recall: 0.2056
Epoch 22/100
33/33 [==============================] - 237s 7s/step - loss: 1.4917 - acc:
0.4337 - precision: 0.6307 - recall: 0.1922 - val_loss: 1.3602 - val_acc: 0.4
917 - val_precision: 0.7734 - val_recall: 0.2111
Epoch 23/100
33/33 [==============================] - 231s 7s/step - loss: 1.5031 - acc:
0.4324 - precision: 0.7388 - recall: 0.1809 - val_loss: 1.3081 - val_acc: 0.5
472 - val_precision: 0.7619 - val_recall: 0.2417
Epoch 24/100
33/33 [==============================] - 228s 7s/step - loss: 1.4297 - acc:
0.4529 - precision: 0.6959 - recall: 0.2250 - val_loss: 1.3712 - val_acc: 0.4
667 - val_precision: 0.7530 - val_recall: 0.2611
Epoch 25/100
33/33 [==============================] - 228s 7s/step - loss: 1.4734 - acc:
0.4328 - precision: 0.6198 - recall: 0.1843 - val_loss: 1.3307 - val_acc: 0.5
361 - val_precision: 0.8045 - val_recall: 0.2389
Epoch 26/100
33/33 [==============================] - 236s 7s/step - loss: 1.4306 - acc:
0.4394 - precision: 0.6726 - recall: 0.2162 - val_loss: 1.3244 - val_acc: 0.4
944 - val_precision: 0.7121 - val_recall: 0.2778
Epoch 27/100
33/33 [==============================] - 231s 7s/step - loss: 1.4319 - acc:
0.4627 - precision: 0.6883 - recall: 0.2254 - val_loss: 1.3187 - val_acc: 0.5
361 - val_precision: 0.7023 - val_recall: 0.3083
Epoch 28/100
33/33 [==============================] - 230s 7s/step - loss: 1.4167 - acc:
0.4811 - precision: 0.6731 - recall: 0.2348 - val_loss: 1.3577 - val_acc: 0.5
361 - val_precision: 0.6782 - val_recall: 0.2639
Epoch 29/100
33/33 [==============================] - 229s 7s/step - loss: 1.4026 - acc:
```

```
0.4665 - precision: 0.6574 - recall: 0.2272 - val_loss: 1.2603 - val_acc: 0.5
333 - val_precision: 0.7070 - val_recall: 0.3333
Epoch 30/100
33/33 [==============================] - 233s 7s/step - loss: 1.4340 - acc:
0.4517 - precision: 0.6646 - recall: 0.2317 - val_loss: 1.2612 - val_acc: 0.5
583 - val_precision: 0.8135 - val_recall: 0.2528
Epoch 31/100
33/33 [==============================] - 229s 7s/step - loss: 1.3593 - acc:
0.4934 - precision: 0.7124 - recall: 0.2652 - val_loss: 1.2463 - val_acc: 0.5
639 - val_precision: 0.7834 - val_recall: 0.2944
Epoch 32/100
33/33 [==============================] - 229s 7s/step - loss: 1.3609 - acc:
0.4902 - precision: 0.6867 - recall: 0.2648 - val_loss: 1.3349 - val_acc: 0.5
000 - val_precision: 0.6613 - val_recall: 0.3000
Epoch 33/100
33/33 [==============================] - 228s 7s/step - loss: 1.3365 - acc:
0.4994 - precision: 0.6719 - recall: 0.2617 - val_loss: 1.2701 - val_acc: 0.5
528 - val_precision: 0.7372 - val_recall: 0.3139
Epoch 34/100
33/33 [==============================] - 228s 7s/step - loss: 1.3597 - acc:
0.4814 - precision: 0.7189 - recall: 0.2579 - val_loss: 1.3522 - val_acc: 0.5
000 - val_precision: 0.6497 - val_recall: 0.3139
Epoch 35/100
33/33 [==============================] - 233s 7s/step - loss: 1.3644 - acc:
0.4902 - precision: 0.6706 - recall: 0.2645 - val_loss: 1.2468 - val_acc: 0.5
833 - val_precision: 0.7516 - val_recall: 0.3028
Epoch 36/100
33/33 [==============================] - 233s 7s/step - loss: 1.3756 - acc:
0.4810 - precision: 0.7066 - recall: 0.2405 - val_loss: 1.2735 - val_acc: 0.5
083 - val_precision: 0.6668 - val_recall: 0.3722
Epoch 37/100
33/33 [==============================] - 236s 7s/step - loss: 1.3257 - acc:
0.5085 - precision: 0.7180 - recall: 0.2812 - val_loss: 1.2699 - val_acc: 0.5
417 - val_precision: 0.7795 - val_recall: 0.2778
Epoch 38/100
33/33 [==============================] - 233s 7s/step - loss: 1.3328 - acc:
0.4707 - precision: 0.6587 - recall: 0.2630 - val_loss: 1.2150 - val_acc: 0.5
500 - val_precision: 0.7582 - val_recall: 0.3278
Epoch 39/100
33/33 [==============================] - 233s 7s/step - loss: 1.2709 - acc:
0.5237 - precision: 0.7243 - recall: 0.2961 - val_loss: 1.2396 - val_acc: 0.5
306 - val_precision: 0.7438 - val_recall: 0.3667
Epoch 40/100
33/33 [==============================] - 233s 7s/step - loss: 1.3421 - acc:
0.5072 - precision: 0.6857 - recall: 0.2983 - val_loss: 1.2094 - val_acc: 0.5
611 - val_precision: 0.7658 - val_recall: 0.3250
Epoch 41/100
33/33 [==============================] - 230s 7s/step - loss: 1.3111 - acc:
0.5186 - precision: 0.7187 - recall: 0.2850 - val_loss: 1.2952 - val_acc: 0.5
167 - val_precision: 0.7245 - val_recall: 0.3444
Epoch 42/100
33/33 [==============================] - 230s 7s/step - loss: 1.2842 - acc:
0.5180 - precision: 0.7186 - recall: 0.2926 - val_loss: 1.2642 - val_acc: 0.5
389 - val_precision: 0.6756 - val_recall: 0.3667
Epoch 43/100
33/33 [==============================] - 229s 7s/step - loss: 1.2863 - acc:
0.5386 - precision: 0.7261 - recall: 0.3188 - val_loss: 1.2248 - val_acc: 0.5
```

```
444 - val_precision: 0.7262 - val_recall: 0.3472
Epoch 44/100
33/33 [==============================] - 228s 7s/step - loss: 1.2514 - acc:
0.5246 - precision: 0.7068 - recall: 0.3150 - val_loss: 1.1754 - val_acc: 0.5
778 - val_precision: 0.7171 - val_recall: 0.4083
Epoch 45/100
33/33 [==============================] - 228s 7s/step - loss: 1.2415 - acc:
0.5277 - precision: 0.6926 - recall: 0.3197 - val_loss: 1.1670 - val_acc: 0.5
778 - val_precision: 0.7599 - val_recall: 0.3861
Epoch 46/100
33/33 [==============================] - 227s 7s/step - loss: 1.2910 - acc:
0.5123 - precision: 0.7126 - recall: 0.2914 - val_loss: 1.1463 - val_acc: 0.5
861 - val_precision: 0.7622 - val_recall: 0.4028
Epoch 47/100
33/33 [==============================] - 228s 7s/step - loss: 1.1988 - acc:
0.5410 - precision: 0.7187 - recall: 0.3488 - val_loss: 1.1621 - val_acc: 0.5
778 - val_precision: 0.7473 - val_recall: 0.3833
Epoch 48/100
33/33 [==============================] - 227s 7s/step - loss: 1.1895 - acc:
0.5410 - precision: 0.7325 - recall: 0.3564 - val_loss: 1.2638 - val_acc: 0.5
389 - val_precision: 0.6768 - val_recall: 0.3972
Epoch 49/100
33/33 [==============================] - 226s 7s/step - loss: 1.1930 - acc:
0.5590 - precision: 0.7586 - recall: 0.3614 - val_loss: 1.1983 - val_acc: 0.5
639 - val_precision: 0.7195 - val_recall: 0.4278
Epoch 50/100
33/33 [==============================] - 233s 7s/step - loss: 1.3033 - acc:
0.5180 - precision: 0.7200 - recall: 0.3182 - val_loss: 1.2718 - val_acc: 0.5
500 - val_precision: 0.7033 - val_recall: 0.3278
Epoch 51/100
33/33 [==============================] - 229s 7s/step - loss: 1.1863 - acc:
0.5455 - precision: 0.7376 - recall: 0.3428 - val_loss: 1.1499 - val_acc: 0.5
944 - val_precision: 0.7429 - val_recall: 0.3944
Epoch 52/100
33/33 [==============================] - 229s 7s/step - loss: 1.2010 - acc:
0.5426 - precision: 0.7379 - recall: 0.3835 - val_loss: 1.1335 - val_acc: 0.6
139 - val_precision: 0.7898 - val_recall: 0.4028
Epoch 53/100
33/33 [==============================] - 227s 7s/step - loss: 1.1882 - acc:
0.5348 - precision: 0.7511 - recall: 0.3555 - val_loss: 1.1809 - val_acc: 0.5
472 - val_precision: 0.7193 - val_recall: 0.4139
Epoch 54/100
33/33 [==============================] - 228s 7s/step - loss: 1.1522 - acc:
0.5710 - precision: 0.7204 - recall: 0.3873 - val_loss: 1.1394 - val_acc: 0.5
778 - val_precision: 0.7336 - val_recall: 0.4500
Epoch 55/100
33/33 [==============================] - 229s 7s/step - loss: 1.1530 - acc:
0.5631 - precision: 0.7253 - recall: 0.3851 - val_loss: 1.1280 - val_acc: 0.5
722 - val_precision: 0.7357 - val_recall: 0.4278
Epoch 56/100
33/33 [==============================] - 229s 7s/step - loss: 1.1439 - acc:
0.5625 - precision: 0.7553 - recall: 0.3842 - val_loss: 1.1533 - val_acc: 0.5
611 - val_precision: 0.6849 - val_recall: 0.4528
Epoch 57/100
33/33 [==============================] - 229s 7s/step - loss: 1.2102 - acc:
0.5451 - precision: 0.7189 - recall: 0.3813 - val_loss: 1.1674 - val_acc: 0.5
917 - val_precision: 0.7247 - val_recall: 0.4583
```

```
Epoch 58/100
33/33 [==============================] - 226s 7s/step - loss: 1.1328 - acc:
0.5650 - precision: 0.7334 - recall: 0.3895 - val_loss: 1.1624 - val_acc: 0.5
639 - val_precision: 0.7017 - val_recall: 0.4361
Epoch 59/100
33/33 [==============================] - 228s 7s/step - loss: 1.1339 - acc:
0.5616 - precision: 0.7231 - recall: 0.3920 - val_loss: 1.0889 - val_acc: 0.5
917 - val_precision: 0.7266 - val_recall: 0.4556
Epoch 60/100
33/33 [==============================] - 231s 7s/step - loss: 1.1522 - acc:
0.5717 - precision: 0.7262 - recall: 0.3936 - val_loss: 1.0353 - val_acc: 0.6
167 - val_precision: 0.7624 - val_recall: 0.4611
Epoch 61/100
33/33 [==============================] - 232s 7s/step - loss: 1.1455 - acc:
0.5716 - precision: 0.7261 - recall: 0.3768 - val_loss: 1.0507 - val_acc: 0.6
083 - val_precision: 0.7555 - val_recall: 0.4583
Epoch 62/100
33/33 [==============================] - 236s 7s/step - loss: 1.1224 - acc:
0.5846 - precision: 0.7463 - recall: 0.4309 - val_loss: 1.2554 - val_acc: 0.5
444 - val_precision: 0.6451 - val_recall: 0.4444
Epoch 63/100
33/33 [==============================] - 233s 7s/step - loss: 1.1582 - acc:
0.5720 - precision: 0.7411 - recall: 0.3807 - val_loss: 1.0909 - val_acc: 0.5
861 - val_precision: 0.7835 - val_recall: 0.4000
Epoch 64/100
33/33 [==============================] - 242s 7s/step - loss: 1.1183 - acc:
0.5795 - precision: 0.7290 - recall: 0.3895 - val_loss: 1.1696 - val_acc: 0.5
639 - val_precision: 0.7156 - val_recall: 0.4639
Epoch 65/100
33/33 [==============================] - 327s 10s/step - loss: 1.1379 - acc:
0.5679 - precision: 0.7611 - recall: 0.3943 - val_loss: 1.1610 - val_acc: 0.5
639 - val_precision: 0.7026 - val_recall: 0.4472
Epoch 66/100
33/33 [==============================] - 324s 10s/step - loss: 1.1056 - acc:
0.5931 - precision: 0.7565 - recall: 0.4217 - val_loss: 1.0528 - val_acc: 0.6
000 - val_precision: 0.7674 - val_recall: 0.4472
Epoch 67/100
33/33 [==============================] - 324s 10s/step - loss: 1.1077 - acc:
0.5859 - precision: 0.7318 - recall: 0.3835 - val_loss: 1.1155 - val_acc: 0.5
806 - val_precision: 0.7025 - val_recall: 0.4806
Epoch 68/100
33/33 [==============================] - 322s 10s/step - loss: 1.1503 - acc:
0.5537 - precision: 0.7215 - recall: 0.3920 - val_loss: 1.1620 - val_acc: 0.5
667 - val_precision: 0.6910 - val_recall: 0.4722
Epoch 69/100
33/33 [==============================] - 323s 10s/step - loss: 1.0980 - acc:
0.5811 - precision: 0.7468 - recall: 0.4082 - val_loss: 1.1029 - val_acc: 0.5
917 - val_precision: 0.7417 - val_recall: 0.4778
Epoch 70/100
33/33 [==============================] - 324s 10s/step - loss: 1.0833 - acc:
0.5969 - precision: 0.7451 - recall: 0.4173 - val_loss: 1.1073 - val_acc: 0.6
000 - val_precision: 0.7177 - val_recall: 0.4806
Epoch 71/100
33/33 [==============================] - 324s 10s/step - loss: 1.1071 - acc:
0.5758 - precision: 0.7284 - recall: 0.4034 - val_loss: 1.0840 - val_acc: 0.6
000 - val_precision: 0.7103 - val_recall: 0.4778
Epoch 72/100
```

```
33/33 [==============================] - 323s 10s/step - loss: 1.0433 - acc:
0.6080 - precision: 0.7520 - recall: 0.4407 - val_loss: 1.1172 - val_acc: 0.5
944 - val_precision: 0.6987 - val_recall: 0.4778
Epoch 73/100
33/33 [==============================] - 322s 10s/step - loss: 1.0427 - acc:
0.6013 - precision: 0.7571 - recall: 0.4659 - val_loss: 1.0620 - val_acc: 0.5
917 - val_precision: 0.7586 - val_recall: 0.4889
Epoch 74/100
33/33 [==============================] - 322s 10s/step - loss: 1.0721 - acc:
0.6029 - precision: 0.7442 - recall: 0.4302 - val_loss: 1.0364 - val_acc: 0.5
972 - val_precision: 0.7475 - val_recall: 0.4694
Epoch 75/100
33/33 [==============================] - 324s 10s/step - loss: 1.0591 - acc:
0.6247 - precision: 0.7774 - recall: 0.4619 - val_loss: 1.1120 - val_acc: 0.5
889 - val_precision: 0.7068 - val_recall: 0.4889
Epoch 76/100
33/33 [==============================] - 323s 10s/step - loss: 1.0515 - acc:
0.5852 - precision: 0.7342 - recall: 0.4391 - val_loss: 1.0578 - val_acc: 0.5
861 - val_precision: 0.7227 - val_recall: 0.4778
Epoch 77/100
33/33 [==============================] - 324s 10s/step - loss: 1.0440 - acc:
0.6142 - precision: 0.7723 - recall: 0.4520 - val_loss: 0.9874 - val_acc: 0.6
222 - val_precision: 0.7352 - val_recall: 0.4889
Epoch 78/100
33/33 [==============================] - 324s 10s/step - loss: 1.0195 - acc:
0.6278 - precision: 0.7841 - recall: 0.4583 - val_loss: 1.0891 - val_acc: 0.6
028 - val_precision: 0.6929 - val_recall: 0.5056
Epoch 79/100
33/33 [==============================] - 323s 10s/step - loss: 1.0041 - acc:
0.6206 - precision: 0.7567 - recall: 0.4719 - val_loss: 1.0146 - val_acc: 0.6
083 - val_precision: 0.7073 - val_recall: 0.4917
Epoch 80/100
33/33 [==============================] - 323s 10s/step - loss: 0.9673 - acc:
0.6342 - precision: 0.7722 - recall: 0.4858 - val_loss: 0.9864 - val_acc: 0.6
000 - val_precision: 0.7270 - val_recall: 0.4833
Epoch 81/100
33/33 [==============================] - 326s 10s/step - loss: 1.0494 - acc:
0.6136 - precision: 0.7412 - recall: 0.4384 - val_loss: 1.0054 - val_acc: 0.6
167 - val_precision: 0.7356 - val_recall: 0.4944
Epoch 82/100
33/33 [==============================] - 325s 10s/step - loss: 1.0492 - acc:
0.6256 - precision: 0.7670 - recall: 0.4580 - val_loss: 1.1780 - val_acc: 0.6
028 - val_precision: 0.7009 - val_recall: 0.4806
Epoch 83/100
33/33 [==============================] - 325s 10s/step - loss: 1.0590 - acc:
0.5959 - precision: 0.7381 - recall: 0.4539 - val_loss: 0.9959 - val_acc: 0.6
111 - val_precision: 0.7415 - val_recall: 0.4639
Epoch 84/100
33/33 [==============================] - 323s 10s/step - loss: 1.0469 - acc:
0.6098 - precision: 0.7546 - recall: 0.4366 - val_loss: 0.9698 - val_acc: 0.6
222 - val_precision: 0.7411 - val_recall: 0.5083
Epoch 85/100
33/33 [==============================] - 325s 10s/step - loss: 1.0408 - acc:
0.5988 - precision: 0.7707 - recall: 0.4561 - val_loss: 1.0643 - val_acc: 0.5
861 - val_precision: 0.6943 - val_recall: 0.4667
Epoch 86/100
33/33 [==============================] - 326s 10s/step - loss: 0.9703 - acc:
```

```
0.6326 - precision: 0.7588 - recall: 0.4744 - val_loss: 1.0511 - val_acc: 0.6
028 - val_precision: 0.7144 - val_recall: 0.4750
Epoch 87/100
33/33 [==============================] - 321s 10s/step - loss: 0.9977 - acc:
0.6237 - precision: 0.7591 - recall: 0.4795 - val_loss: 1.0328 - val_acc: 0.6
083 - val_precision: 0.7246 - val_recall: 0.5167
Epoch 88/100
33/33 [==============================] - 323s 10s/step - loss: 0.9918 - acc:
0.6364 - precision: 0.7606 - recall: 0.4706 - val_loss: 1.0141 - val_acc: 0.6
194 - val_precision: 0.7184 - val_recall: 0.5194
Epoch 89/100
33/33 [==============================] - 324s 10s/step - loss: 0.9773 - acc:
0.6446 - precision: 0.7760 - recall: 0.4874 - val_loss: 1.1130 - val_acc: 0.6
333 - val_precision: 0.7151 - val_recall: 0.5417
Epoch 90/100
33/33 [==============================] - 322s 10s/step - loss: 0.9804 - acc:
0.6354 - precision: 0.7547 - recall: 0.4905 - val_loss: 1.0453 - val_acc: 0.6
139 - val_precision: 0.7213 - val_recall: 0.5417
Epoch 91/100
33/33 [==============================] - 323s 10s/step - loss: 1.0227 - acc:
0.6234 - precision: 0.7449 - recall: 0.4653 - val_loss: 0.9994 - val_acc: 0.6
056 - val_precision: 0.7326 - val_recall: 0.4972
Epoch 92/100
33/33 [==============================] - 322s 10s/step - loss: 0.9571 - acc:
0.6380 - precision: 0.7922 - recall: 0.5041 - val_loss: 0.9960 - val_acc: 0.6
139 - val_precision: 0.6943 - val_recall: 0.5056
Epoch 93/100
33/33 [==============================] - 323s 10s/step - loss: 0.9837 - acc:
0.6354 - precision: 0.7781 - recall: 0.4662 - val_loss: 1.0371 - val_acc: 0.6
028 - val_precision: 0.6917 - val_recall: 0.5056
Epoch 94/100
33/33 [==============================] - 324s 10s/step - loss: 0.9725 - acc:
0.6203 - precision: 0.7645 - recall: 0.4909 - val_loss: 0.9638 - val_acc: 0.6
250 - val_precision: 0.7368 - val_recall: 0.5222
Epoch 95/100
33/33 [==============================] - 323s 10s/step - loss: 0.9657 - acc:
0.6443 - precision: 0.7862 - recall: 0.5057 - val_loss: 1.0314 - val_acc: 0.6
250 - val_precision: 0.7614 - val_recall: 0.5500
Epoch 96/100
33/33 [==============================] - 322s 10s/step - loss: 0.9893 - acc:
0.6376 - precision: 0.7617 - recall: 0.4997 - val_loss: 1.0277 - val_acc: 0.6
250 - val_precision: 0.7462 - val_recall: 0.5639
Epoch 97/100
33/33 [==============================] - 323s 10s/step - loss: 0.9903 - acc:
0.6184 - precision: 0.7458 - recall: 0.4792 - val_loss: 0.9090 - val_acc: 0.6
528 - val_precision: 0.7773 - val_recall: 0.5306
Epoch 98/100
33/33 [==============================] - 326s 10s/step - loss: 0.9729 - acc:
0.6430 - precision: 0.7509 - recall: 0.4883 - val_loss: 0.9722 - val_acc: 0.6
389 - val_precision: 0.7487 - val_recall: 0.5472
Epoch 99/100
33/33 [==============================] - 325s 10s/step - loss: 1.0040 - acc:
0.6370 - precision: 0.7596 - recall: 0.4852 - val_loss: 0.9822 - val_acc: 0.6
250 - val_precision: 0.7291 - val_recall: 0.5222
Epoch 100/100
33/33 [==============================] - 316s 10s/step - loss: 0.9528 - acc:
```

```
0.6531 - precision: 0.7678 - recall: 0.4997 - val_loss: 0.9558 - val_acc: 0.6
444 - val_precision: 0.7762 - val_recall: 0.5583
```

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)
```

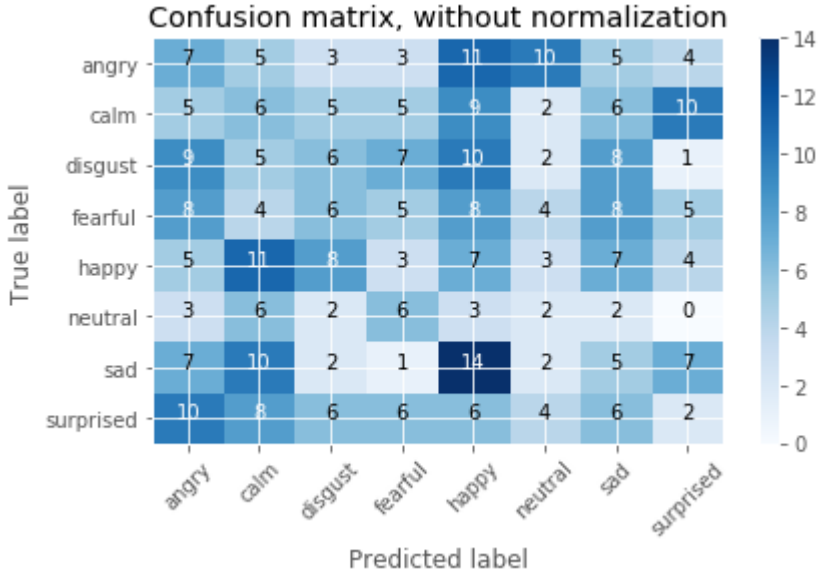
In [6]:
```
true_classes = test_set.classes
class_labels = list(test_set.class_indices.keys())
```

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

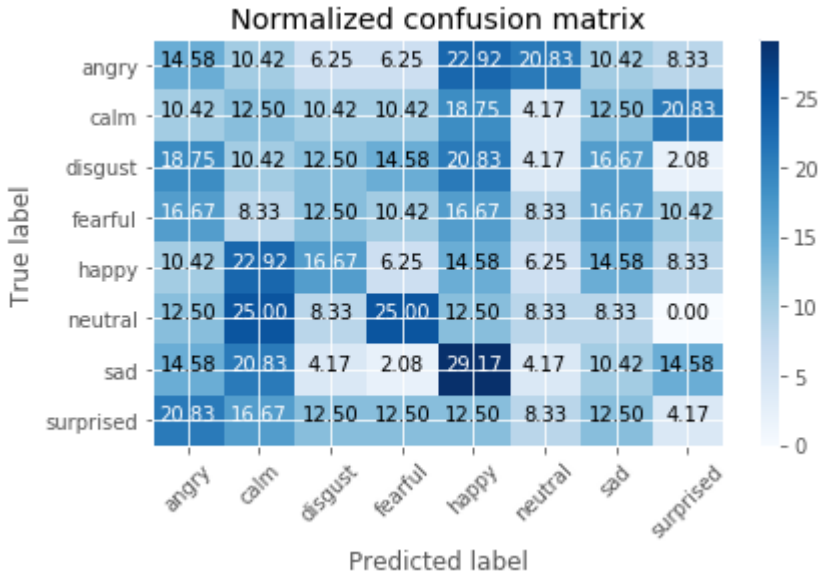
```
             precision    recall  f1-score   support

      angry       0.13      0.15      0.14        48
       calm       0.11      0.12      0.12        48
    disgust       0.16      0.12      0.14        48
    fearful       0.14      0.10      0.12        48
      happy       0.10      0.15      0.12        48
    neutral       0.07      0.08      0.08        24
        sad       0.11      0.10      0.11        48
  surprised       0.06      0.04      0.05        48

avg / total       0.11      0.11      0.11       360
```

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
[[ 7  5  3  3 11 10  5  4]
 [ 5  6  5  5  9  2  6 10]
 [ 9  5  6  7 10  2  8  1]
 [ 8  4  6  5  8  4  8  5]
 [ 5 11  8  3  7  3  7  4]
 [ 3  6  2  6  3  2  2  0]
 [ 7 10  2  1 14  2  5  7]
 [10  8  6  6  6  4  6  2]]
```




```
Normalized confusion matrix
[[14.5833 10.4167  6.25    6.25   22.9167 20.8333 10.4167  8.3333]
 [10.4167 12.5    10.4167 10.4167 18.75    4.1667 12.5    20.8333]
 [18.75   10.4167 12.5    14.5833 20.8333  4.1667 16.6667  2.0833]
 [16.6667  8.3333 12.5    10.4167 16.6667  8.3333 16.6667 10.4167]
 [10.4167 22.9167 16.6667  6.25   14.5833  6.25   14.5833  8.3333]
 [12.5    25.      8.3333 25.     12.5     8.3333  8.3333  0.    ]
 [14.5833 20.8333  4.1667  2.0833 29.1667  4.1667 10.4167 14.5833]
 [20.8333 16.6667 12.5    12.5    12.5     8.3333 12.5     4.1667]]
```



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")
```