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 = 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
_________________________________________________________________
reshape_1 (Reshape)          (None, 16, 1024)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 16, 50)            215000
_________________________________________________________________
lstm_2 (LSTM)                (None, 20)                5680
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 168
=================================================================
Total params: 296,496
Trainable params: 296,496
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 [==============================] - 542s 16s/step - loss: 2.0679 - acc: 0.1414 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 2.0458 - val_acc: 0.1806 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/100
33/33 [==============================] - 534s 16s/step - loss: 2.0656 - acc: 0.1462 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 2.0280 - val_acc: 0.2028 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 3/100
33/33 [==============================] - 539s 16s/step - loss: 2.0315 - acc: 0.1891 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9973 - val_acc: 0.1944 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 4/100
33/33 [==============================] - 544s 16s/step - loss: 2.0124 - acc: 0.1960 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9666 - val_acc: 0.2083 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 5/100
33/33 [==============================] - 537s 16s/step - loss: 1.9894 - acc: 0.1932 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9755 - val_acc: 0.2139 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 6/100
33/33 [==============================] - 536s 16s/step - loss: 2.0013 - acc: 0.2048 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9821 - val_acc: 0.1806 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 7/100
33/33 [==============================] - 533s 16s/step - loss: 1.9771 - acc: 0.2184 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9842 - val_acc: 0.1806 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 8/100
33/33 [==============================] - 533s 16s/step - loss: 1.9838 - acc: 0.2052 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9706 - val_acc: 0.1861 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 9/100
33/33 [==============================] - 535s 16s/step - loss: 1.9963 - acc: 0.2096 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9592 - val_acc: 0.2361 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 10/100
33/33 [==============================] - 450s 14s/step - loss: 1.9679 - acc: 0.2102 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9362 - val_acc: 0.2222 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 11/100
33/33 [==============================] - 438s 13s/step - loss: 1.9740 - acc: 0.2030 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9445 - val_acc: 0.2278 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 12/100
33/33 [==============================] - 446s 14s/step - loss: 1.9677 - acc: 0.2317 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9283 - val_acc: 0.2583 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 13/100
33/33 [==============================] - 449s 14s/step - loss: 1.9731 - acc: 0.2207 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9278 - val_acc: 0.2667 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 14/100
33/33 [==============================] - 442s 13s/step - loss: 1.9471 - acc: 0.2377 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9001 - val_acc: 0.2778 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 15/100
```

```
33/33 [==============================] - 443s 13s/step - loss: 1.9354 - acc:
0.2421 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.9296 - val_
acc: 0.2194 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 16/100
33/33 [==============================] - 438s 13s/step - loss: 1.9220 - acc:
0.2465 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8845 - val_
acc: 0.2722 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 17/100
33/33 [==============================] - 429s 13s/step - loss: 1.9324 - acc:
0.2348 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.8961 - val_
acc: 0.2556 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 18/100
33/33 [==============================] - 238s 7s/step - loss: 1.8821 - acc:
0.2519 - precision: 0.0152 - recall: 9.4697e-04 - val_loss: 1.8265 - val_acc:
0.2722 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 19/100
33/33 [==============================] - 234s 7s/step - loss: 1.8651 - acc:
0.2882 - precision: 0.0606 - recall: 0.0019 - val_loss: 1.8202 - val_acc: 0.2
833 - val_precision: 0.1778 - val_recall: 0.0056
Epoch 20/100
33/33 [==============================] - 239s 7s/step - loss: 1.8816 - acc:
0.2487 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.7916 - val_
acc: 0.3222 - val_precision: 0.2430 - val_recall: 0.0083
Epoch 21/100
33/33 [==============================] - 242s 7s/step - loss: 1.8494 - acc:
0.2926 - precision: 0.1667 - recall: 0.0057 - val_loss: 1.8067 - val_acc: 0.3
028 - val_precision: 0.2422 - val_recall: 0.0083
Epoch 22/100
33/33 [==============================] - 238s 7s/step - loss: 1.7882 - acc:
0.3040 - precision: 0.2879 - recall: 0.0095 - val_loss: 1.7332 - val_acc: 0.3
417 - val_precision: 0.5096 - val_recall: 0.0250
Epoch 23/100
33/33 [==============================] - 231s 7s/step - loss: 1.7947 - acc:
0.3049 - precision: 0.4596 - recall: 0.0189 - val_loss: 1.7583 - val_acc: 0.2
972 - val_precision: 0.3007 - val_recall: 0.0111
Epoch 24/100
33/33 [==============================] - 229s 7s/step - loss: 1.8061 - acc:
0.2885 - precision: 0.3789 - recall: 0.0189 - val_loss: 1.8288 - val_acc: 0.2
833 - val_precision: 0.3531 - val_recall: 0.0167
Epoch 25/100
33/33 [==============================] - 232s 7s/step - loss: 1.7703 - acc:
0.3128 - precision: 0.5204 - recall: 0.0294 - val_loss: 1.6786 - val_acc: 0.3
417 - val_precision: 0.5290 - val_recall: 0.0472
Epoch 26/100
33/33 [==============================] - 240s 7s/step - loss: 1.7460 - acc:
0.3182 - precision: 0.7469 - recall: 0.0625 - val_loss: 1.7038 - val_acc: 0.3
417 - val_precision: 0.5625 - val_recall: 0.0389
Epoch 27/100
33/33 [==============================] - 232s 7s/step - loss: 1.7250 - acc:
0.3424 - precision: 0.5853 - recall: 0.0429 - val_loss: 1.6873 - val_acc: 0.3
639 - val_precision: 0.6017 - val_recall: 0.0389
Epoch 28/100
33/33 [==============================] - 231s 7s/step - loss: 1.7460 - acc:
0.3220 - precision: 0.4066 - recall: 0.0341 - val_loss: 1.7322 - val_acc: 0.3
167 - val_precision: 0.5853 - val_recall: 0.0333
Epoch 29/100
33/33 [==============================] - 230s 7s/step - loss: 1.7294 - acc:
```

```
0.3286 - precision: 0.5611 - recall: 0.0335 - val_loss: 1.7426 - val_acc: 0.3
306 - val_precision: 0.6259 - val_recall: 0.0333
Epoch 30/100
33/33 [==============================] - 237s 7s/step - loss: 1.6938 - acc:
0.3415 - precision: 0.6091 - recall: 0.0404 - val_loss: 1.7481 - val_acc: 0.3
694 - val_precision: 0.4835 - val_recall: 0.0278
Epoch 31/100
33/33 [==============================] - 230s 7s/step - loss: 1.6859 - acc:
0.3441 - precision: 0.4781 - recall: 0.0398 - val_loss: 1.6810 - val_acc: 0.3
722 - val_precision: 0.6773 - val_recall: 0.0500
Epoch 32/100
33/33 [==============================] - 232s 7s/step - loss: 1.6682 - acc:
0.3535 - precision: 0.6985 - recall: 0.0562 - val_loss: 1.6736 - val_acc: 0.3
750 - val_precision: 0.5147 - val_recall: 0.0611
Epoch 33/100
33/33 [==============================] - 232s 7s/step - loss: 1.6868 - acc:
0.3495 - precision: 0.4541 - recall: 0.0417 - val_loss: 1.6427 - val_acc: 0.3
667 - val_precision: 0.5524 - val_recall: 0.0389
Epoch 34/100
33/33 [==============================] - 234s 7s/step - loss: 1.6500 - acc:
0.3481 - precision: 0.5216 - recall: 0.0429 - val_loss: 1.6845 - val_acc: 0.3
583 - val_precision: 0.6278 - val_recall: 0.0361
Epoch 35/100
33/33 [==============================] - 233s 7s/step - loss: 1.7249 - acc:
0.3358 - precision: 0.4882 - recall: 0.0555 - val_loss: 1.6687 - val_acc: 0.3
722 - val_precision: 0.4985 - val_recall: 0.0278
Epoch 36/100
33/33 [==============================] - 236s 7s/step - loss: 1.6760 - acc:
0.3147 - precision: 0.6109 - recall: 0.0555 - val_loss: 1.6750 - val_acc: 0.3
722 - val_precision: 0.5651 - val_recall: 0.0417
Epoch 37/100
33/33 [==============================] - 238s 7s/step - loss: 1.6206 - acc:
0.3671 - precision: 0.6086 - recall: 0.0685 - val_loss: 1.7298 - val_acc: 0.3
500 - val_precision: 0.6607 - val_recall: 0.0639
Epoch 38/100
33/33 [==============================] - 236s 7s/step - loss: 1.6767 - acc:
0.3580 - precision: 0.6469 - recall: 0.0638 - val_loss: 1.6876 - val_acc: 0.3
833 - val_precision: 0.6808 - val_recall: 0.0667
Epoch 39/100
33/33 [==============================] - 234s 7s/step - loss: 1.6594 - acc:
0.3618 - precision: 0.6083 - recall: 0.0666 - val_loss: 1.6090 - val_acc: 0.4
139 - val_precision: 0.6597 - val_recall: 0.0639
Epoch 40/100
33/33 [==============================] - 231s 7s/step - loss: 1.6172 - acc:
0.3797 - precision: 0.5943 - recall: 0.0729 - val_loss: 1.6606 - val_acc: 0.3
889 - val_precision: 0.5556 - val_recall: 0.0667
Epoch 41/100
33/33 [==============================] - 232s 7s/step - loss: 1.6256 - acc:
0.3763 - precision: 0.6590 - recall: 0.0704 - val_loss: 1.5658 - val_acc: 0.4
222 - val_precision: 0.6682 - val_recall: 0.0722
Epoch 42/100
33/33 [==============================] - 232s 7s/step - loss: 1.6154 - acc:
0.3712 - precision: 0.5721 - recall: 0.0742 - val_loss: 1.5653 - val_acc: 0.4
028 - val_precision: 0.6222 - val_recall: 0.0861
Epoch 43/100
33/33 [==============================] - 231s 7s/step - loss: 1.6049 - acc:
0.3683 - precision: 0.6177 - recall: 0.0846 - val_loss: 1.5620 - val_acc: 0.4
```

```
167 - val_precision: 0.6555 - val_recall: 0.0889
Epoch 44/100
33/33 [==============================] - 231s 7s/step - loss: 1.6038 - acc:
0.3797 - precision: 0.5598 - recall: 0.0817 - val_loss: 1.5526 - val_acc: 0.4
194 - val_precision: 0.6436 - val_recall: 0.0972
Epoch 45/100
33/33 [==============================] - 230s 7s/step - loss: 1.5808 - acc:
0.3987 - precision: 0.6543 - recall: 0.0985 - val_loss: 1.6423 - val_acc: 0.3
806 - val_precision: 0.6406 - val_recall: 0.0889
Epoch 46/100
33/33 [==============================] - 228s 7s/step - loss: 1.5723 - acc:
0.3861 - precision: 0.6029 - recall: 0.1080 - val_loss: 1.5960 - val_acc: 0.3
972 - val_precision: 0.6044 - val_recall: 0.1667
Epoch 47/100
33/33 [==============================] - 229s 7s/step - loss: 1.5891 - acc:
0.3930 - precision: 0.6255 - recall: 0.1155 - val_loss: 1.6096 - val_acc: 0.3
833 - val_precision: 0.5809 - val_recall: 0.1472
Epoch 48/100
33/33 [==============================] - 230s 7s/step - loss: 1.5492 - acc:
0.4151 - precision: 0.6256 - recall: 0.1133 - val_loss: 1.5497 - val_acc: 0.4
194 - val_precision: 0.6499 - val_recall: 0.1389
Epoch 49/100
33/33 [==============================] - 232s 7s/step - loss: 1.5735 - acc:
0.3778 - precision: 0.6530 - recall: 0.1250 - val_loss: 1.5709 - val_acc: 0.4
306 - val_precision: 0.6906 - val_recall: 0.1444
Epoch 50/100
33/33 [==============================] - 234s 7s/step - loss: 1.5138 - acc:
0.4214 - precision: 0.6725 - recall: 0.1341 - val_loss: 1.5481 - val_acc: 0.4
028 - val_precision: 0.6080 - val_recall: 0.1667
Epoch 51/100
33/33 [==============================] - 229s 7s/step - loss: 1.5686 - acc:
0.4037 - precision: 0.6721 - recall: 0.1342 - val_loss: 1.5197 - val_acc: 0.4
306 - val_precision: 0.7141 - val_recall: 0.1528
Epoch 52/100
33/33 [==============================] - 231s 7s/step - loss: 1.5637 - acc:
0.4056 - precision: 0.5748 - recall: 0.1206 - val_loss: 1.5849 - val_acc: 0.4
222 - val_precision: 0.6824 - val_recall: 0.1222
Epoch 53/100
33/33 [==============================] - 230s 7s/step - loss: 1.5215 - acc:
0.4246 - precision: 0.6246 - recall: 0.1383 - val_loss: 1.5263 - val_acc: 0.4
528 - val_precision: 0.7059 - val_recall: 0.1611
Epoch 54/100
33/33 [==============================] - 230s 7s/step - loss: 1.4825 - acc:
0.4324 - precision: 0.6612 - recall: 0.1291 - val_loss: 1.4820 - val_acc: 0.4
417 - val_precision: 0.6790 - val_recall: 0.1944
Epoch 55/100
33/33 [==============================] - 230s 7s/step - loss: 1.5569 - acc:
0.4091 - precision: 0.5331 - recall: 0.1203 - val_loss: 1.5158 - val_acc: 0.4
306 - val_precision: 0.6624 - val_recall: 0.1500
Epoch 56/100
33/33 [==============================] - 229s 7s/step - loss: 1.5259 - acc:
0.4277 - precision: 0.6260 - recall: 0.1537 - val_loss: 1.6005 - val_acc: 0.4
083 - val_precision: 0.6250 - val_recall: 0.2028
Epoch 57/100
33/33 [==============================] - 231s 7s/step - loss: 1.5346 - acc:
0.4031 - precision: 0.6158 - recall: 0.1468 - val_loss: 1.4643 - val_acc: 0.4
667 - val_precision: 0.6934 - val_recall: 0.1833
```

```
Epoch 58/100
33/33 [==============================] - 231s 7s/step - loss: 1.4722 - acc:
0.4400 - precision: 0.6496 - recall: 0.1679 - val_loss: 1.4589 - val_acc: 0.4
417 - val_precision: 0.7001 - val_recall: 0.2194
Epoch 59/100
33/33 [==============================] - 234s 7s/step - loss: 1.5002 - acc:
0.4316 - precision: 0.6695 - recall: 0.1689 - val_loss: 1.5119 - val_acc: 0.4
222 - val_precision: 0.6671 - val_recall: 0.2056
Epoch 60/100
33/33 [==============================] - 236s 7s/step - loss: 1.4931 - acc:
0.4261 - precision: 0.5791 - recall: 0.1600 - val_loss: 1.5400 - val_acc: 0.4
333 - val_precision: 0.6404 - val_recall: 0.2000
Epoch 61/100
33/33 [==============================] - 236s 7s/step - loss: 1.5153 - acc:
0.4271 - precision: 0.6071 - recall: 0.1702 - val_loss: 1.5218 - val_acc: 0.4
389 - val_precision: 0.5711 - val_recall: 0.2306
Epoch 62/100
33/33 [==============================] - 238s 7s/step - loss: 1.4943 - acc:
0.4293 - precision: 0.6335 - recall: 0.1926 - val_loss: 1.5789 - val_acc: 0.4
139 - val_precision: 0.6086 - val_recall: 0.1972
Epoch 63/100
33/33 [==============================] - 234s 7s/step - loss: 1.4723 - acc:
0.4444 - precision: 0.6932 - recall: 0.2036 - val_loss: 1.5019 - val_acc: 0.4
444 - val_precision: 0.5843 - val_recall: 0.2361
Epoch 64/100
33/33 [==============================] - 319s 10s/step - loss: 1.4824 - acc:
0.4593 - precision: 0.6216 - recall: 0.1705 - val_loss: 1.4508 - val_acc: 0.4
472 - val_precision: 0.6941 - val_recall: 0.2444
Epoch 65/100
33/33 [==============================] - 329s 10s/step - loss: 1.4687 - acc:
0.4388 - precision: 0.6510 - recall: 0.1989 - val_loss: 1.5141 - val_acc: 0.4
333 - val_precision: 0.6834 - val_recall: 0.2139
Epoch 66/100
33/33 [==============================] - 327s 10s/step - loss: 1.4709 - acc:
0.4211 - precision: 0.6310 - recall: 0.1865 - val_loss: 1.4317 - val_acc: 0.4
639 - val_precision: 0.6921 - val_recall: 0.2222
Epoch 67/100
33/33 [==============================] - 328s 10s/step - loss: 1.4604 - acc:
0.4325 - precision: 0.6408 - recall: 0.1900 - val_loss: 1.4334 - val_acc: 0.4
806 - val_precision: 0.6725 - val_recall: 0.2250
Epoch 68/100
33/33 [==============================] - 328s 10s/step - loss: 1.4631 - acc:
0.4441 - precision: 0.6410 - recall: 0.1932 - val_loss: 1.4559 - val_acc: 0.4
722 - val_precision: 0.6356 - val_recall: 0.2583
Epoch 69/100
33/33 [==============================] - 329s 10s/step - loss: 1.4383 - acc:
0.4671 - precision: 0.6671 - recall: 0.2146 - val_loss: 1.4154 - val_acc: 0.4
917 - val_precision: 0.6677 - val_recall: 0.2639
Epoch 70/100
33/33 [==============================] - 329s 10s/step - loss: 1.4057 - acc:
0.4540 - precision: 0.6292 - recall: 0.2093 - val_loss: 1.5144 - val_acc: 0.4
139 - val_precision: 0.6663 - val_recall: 0.2389
Epoch 71/100
33/33 [==============================] - 326s 10s/step - loss: 1.4240 - acc:
0.4413 - precision: 0.6581 - recall: 0.2159 - val_loss: 1.5449 - val_acc: 0.4
417 - val_precision: 0.6075 - val_recall: 0.2500
Epoch 72/100
```

```
33/33 [==============================] - 330s 10s/step - loss: 1.4379 - acc:
0.4334 - precision: 0.6038 - recall: 0.2121 - val_loss: 1.4282 - val_acc: 0.4
528 - val_precision: 0.6407 - val_recall: 0.2472
Epoch 73/100
33/33 [==============================] - 329s 10s/step - loss: 1.3673 - acc:
0.4905 - precision: 0.6797 - recall: 0.2487 - val_loss: 1.4356 - val_acc: 0.4
611 - val_precision: 0.6393 - val_recall: 0.2500
Epoch 74/100
33/33 [==============================] - 329s 10s/step - loss: 1.3976 - acc:
0.4716 - precision: 0.6338 - recall: 0.2263 - val_loss: 1.4115 - val_acc: 0.4
639 - val_precision: 0.6910 - val_recall: 0.2500
Epoch 75/100
33/33 [==============================] - 328s 10s/step - loss: 1.3780 - acc:
0.4792 - precision: 0.6691 - recall: 0.2585 - val_loss: 1.4719 - val_acc: 0.4
361 - val_precision: 0.6122 - val_recall: 0.2472
Epoch 76/100
33/33 [==============================] - 327s 10s/step - loss: 1.3960 - acc:
0.4622 - precision: 0.6660 - recall: 0.2308 - val_loss: 1.3889 - val_acc: 0.5
111 - val_precision: 0.6872 - val_recall: 0.2694
Epoch 77/100
33/33 [==============================] - 328s 10s/step - loss: 1.4111 - acc:
0.4659 - precision: 0.6571 - recall: 0.2304 - val_loss: 1.3966 - val_acc: 0.4
694 - val_precision: 0.7089 - val_recall: 0.2611
Epoch 78/100
33/33 [==============================] - 327s 10s/step - loss: 1.3799 - acc:
0.4912 - precision: 0.6689 - recall: 0.2418 - val_loss: 1.4694 - val_acc: 0.4
278 - val_precision: 0.6318 - val_recall: 0.2667
Epoch 79/100
33/33 [==============================] - 329s 10s/step - loss: 1.3812 - acc:
0.5009 - precision: 0.6617 - recall: 0.2570 - val_loss: 1.4981 - val_acc: 0.4
500 - val_precision: 0.6302 - val_recall: 0.2694
Epoch 80/100
33/33 [==============================] - 328s 10s/step - loss: 1.3303 - acc:
0.4934 - precision: 0.7079 - recall: 0.2677 - val_loss: 1.3595 - val_acc: 0.4
972 - val_precision: 0.6541 - val_recall: 0.3083
Epoch 81/100
33/33 [==============================] - 333s 10s/step - loss: 1.3177 - acc:
0.5038 - precision: 0.6468 - recall: 0.2689 - val_loss: 1.5421 - val_acc: 0.4
083 - val_precision: 0.6024 - val_recall: 0.2639
Epoch 82/100
33/33 [==============================] - 327s 10s/step - loss: 1.4211 - acc:
0.4413 - precision: 0.6206 - recall: 0.2339 - val_loss: 1.4119 - val_acc: 0.4
556 - val_precision: 0.7165 - val_recall: 0.2444
Epoch 83/100
33/33 [==============================] - 329s 10s/step - loss: 1.3823 - acc:
0.4848 - precision: 0.6674 - recall: 0.2648 - val_loss: 1.4520 - val_acc: 0.4
639 - val_precision: 0.6492 - val_recall: 0.2333
Epoch 84/100
33/33 [==============================] - 330s 10s/step - loss: 1.3430 - acc:
0.5095 - precision: 0.7110 - recall: 0.2882 - val_loss: 1.4208 - val_acc: 0.4
472 - val_precision: 0.6217 - val_recall: 0.3056
Epoch 85/100
33/33 [==============================] - 329s 10s/step - loss: 1.3718 - acc:
0.4830 - precision: 0.6561 - recall: 0.2860 - val_loss: 1.3545 - val_acc: 0.4
972 - val_precision: 0.6931 - val_recall: 0.2972
Epoch 86/100
33/33 [==============================] - 328s 10s/step - loss: 1.3373 - acc:
```

```
0.4994 - precision: 0.6983 - recall: 0.2876 - val_loss: 1.3414 - val_acc: 0.4
861 - val_precision: 0.6361 - val_recall: 0.3111
Epoch 87/100
33/33 [==============================] - 325s 10s/step - loss: 1.3486 - acc:
0.5070 - precision: 0.6787 - recall: 0.2705 - val_loss: 1.3921 - val_acc: 0.4
583 - val_precision: 0.6289 - val_recall: 0.2667
Epoch 88/100
33/33 [==============================] - 331s 10s/step - loss: 1.2543 - acc:
0.5394 - precision: 0.7077 - recall: 0.3257 - val_loss: 1.3978 - val_acc: 0.4
861 - val_precision: 0.6379 - val_recall: 0.3250
Epoch 89/100
33/33 [==============================] - 328s 10s/step - loss: 1.3742 - acc:
0.4791 - precision: 0.6687 - recall: 0.2866 - val_loss: 1.3948 - val_acc: 0.4
583 - val_precision: 0.6355 - val_recall: 0.3083
Epoch 90/100
33/33 [==============================] - 327s 10s/step - loss: 1.3299 - acc:
0.5095 - precision: 0.6978 - recall: 0.2850 - val_loss: 1.3604 - val_acc: 0.4
583 - val_precision: 0.6452 - val_recall: 0.3083
Epoch 91/100
33/33 [==============================] - 325s 10s/step - loss: 1.3066 - acc:
0.5304 - precision: 0.6559 - recall: 0.3151 - val_loss: 1.4130 - val_acc: 0.4
444 - val_precision: 0.5868 - val_recall: 0.2861
Epoch 92/100
33/33 [==============================] - 329s 10s/step - loss: 1.3540 - acc:
0.4704 - precision: 0.6401 - recall: 0.2749 - val_loss: 1.3740 - val_acc: 0.4
972 - val_precision: 0.6062 - val_recall: 0.3083
Epoch 93/100
33/33 [==============================] - 326s 10s/step - loss: 1.3352 - acc:
0.5038 - precision: 0.6999 - recall: 0.2951 - val_loss: 1.3433 - val_acc: 0.4
806 - val_precision: 0.6343 - val_recall: 0.3167
Epoch 94/100
33/33 [==============================] - 328s 10s/step - loss: 1.2573 - acc:
0.5455 - precision: 0.7036 - recall: 0.3135 - val_loss: 1.3440 - val_acc: 0.4
889 - val_precision: 0.6336 - val_recall: 0.3389
Epoch 95/100
33/33 [==============================] - 329s 10s/step - loss: 1.3143 - acc:
0.5120 - precision: 0.6895 - recall: 0.3207 - val_loss: 1.3578 - val_acc: 0.5
167 - val_precision: 0.6401 - val_recall: 0.3250
Epoch 96/100
33/33 [==============================] - 326s 10s/step - loss: 1.2821 - acc:
0.5158 - precision: 0.7132 - recall: 0.3116 - val_loss: 1.3280 - val_acc: 0.5
083 - val_precision: 0.6489 - val_recall: 0.3306
Epoch 97/100
33/33 [==============================] - 327s 10s/step - loss: 1.2739 - acc:
0.5205 - precision: 0.7107 - recall: 0.3147 - val_loss: 1.3751 - val_acc: 0.4
889 - val_precision: 0.6400 - val_recall: 0.3278
Epoch 98/100
33/33 [==============================] - 325s 10s/step - loss: 1.2661 - acc:
0.5322 - precision: 0.7035 - recall: 0.3201 - val_loss: 1.3122 - val_acc: 0.5
111 - val_precision: 0.6540 - val_recall: 0.3278
Epoch 99/100
33/33 [==============================] - 275s 8s/step - loss: 1.2511 - acc:
0.5496 - precision: 0.7099 - recall: 0.3188 - val_loss: 1.3949 - val_acc: 0.4
750 - val_precision: 0.6051 - val_recall: 0.3028
Epoch 100/100
33/33 [==============================] - 134s 4s/step - loss: 1.2787 - acc:
```

```
0.5335 - precision: 0.7109 - recall: 0.3106 - val_loss: 1.5509 - val_acc: 0.4
722 - val_precision: 0.5568 - val_recall: 0.3194
```

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())
```

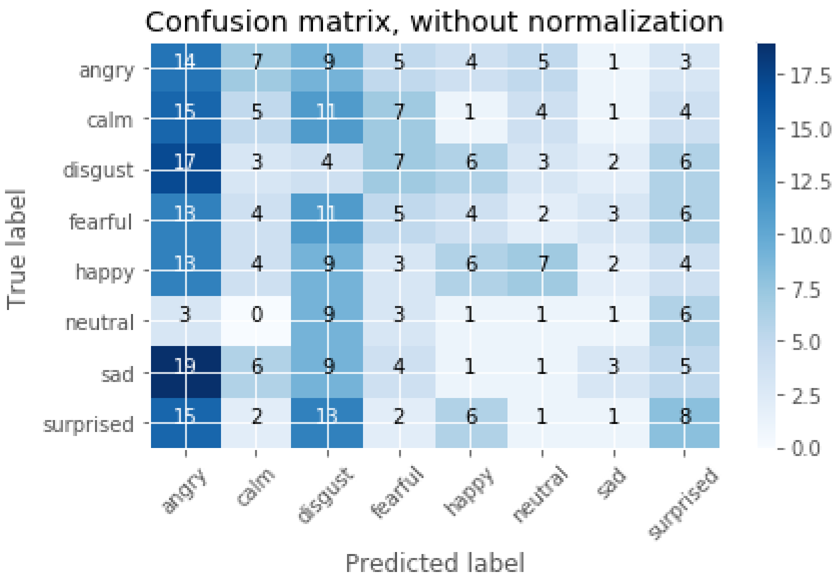
In [7]:

```
import sklearn.metrics as metrics
report = metrics.classification_report(true_classes, predicted_classes, target
_names=class_labels)
print(report)
```

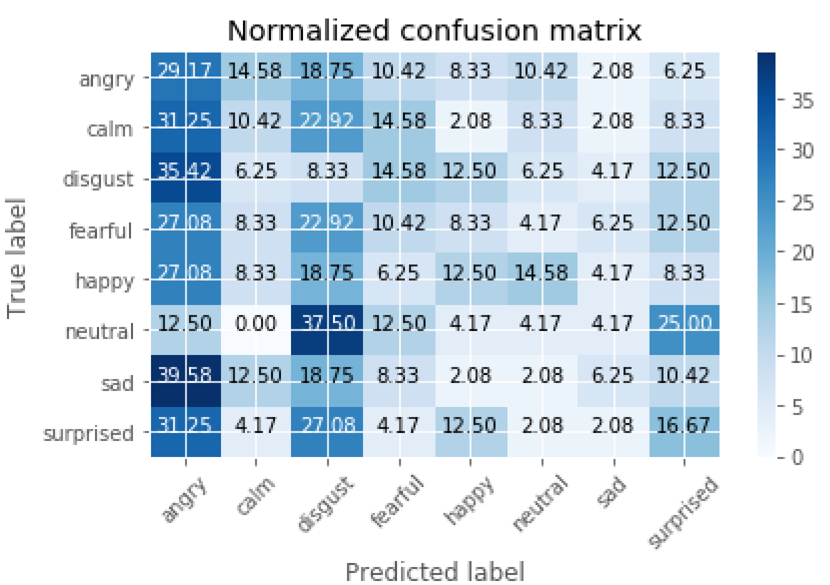
```
             precision    recall  f1-score   support

      angry       0.13      0.29      0.18        48
       calm       0.16      0.10      0.13        48
    disgust       0.05      0.08      0.07        48
    fearful       0.14      0.10      0.12        48
      happy       0.21      0.12      0.16        48
    neutral       0.04      0.04      0.04        24
        sad       0.21      0.06      0.10        48
  surprised       0.19      0.17      0.18        48

avg / total       0.15      0.13      0.13       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_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
[[14  7  9  5  4  5  1  3]
 [15  5 11  7  1  4  1  4]
 [17  3  4  7  6  3  2  6]
 [13  4 11  5  4  2  3  6]
 [13  4  9  3  6  7  2  4]
 [ 3  0  9  3  1  1  1  6]
 [19  6  9  4  1  1  3  5]
 [15  2 13  2  6  1  1  8]]
```




```
Normalized confusion matrix
[[29.1667 14.5833 18.75   10.4167  8.3333 10.4167  2.0833  6.25   ]
 [31.25   10.4167 22.9167 14.5833  2.0833  8.3333  2.0833  8.3333]
 [35.4167  6.25    8.3333 14.5833 12.5     6.25    4.1667 12.5    ]
 [27.0833  8.3333 22.9167 10.4167  8.3333  4.1667  6.25   12.5    ]
 [27.0833  8.3333 18.75    6.25   12.5    14.5833  4.1667  8.3333]
 [12.5     0.     37.5    12.5     4.1667  4.1667  4.1667 25.     ]
 [39.5833 12.5    18.75    8.3333  2.0833  2.0833  6.25   10.4167]
 [31.25    4.1667 27.0833  4.1667 12.5     2.0833  2.0833 16.6667]]
```



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_lstm.png")
```