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

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

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

# Installing Keras
# pip install --upgrade keras

# Part 1 - Building the CNN

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

import warnings
warnings.filterwarnings('ignore')

# Initialising the CNN
classifier = Sequential()

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

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

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

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

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

# Step 4 - Full connection
```

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

classifier.summary()
```

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

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 64)	1792
max_pooling2d_1 (MaxPooling2	(None, 64, 64, 64)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 32, 32, 64)	0
conv2d_3 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 64)	0
flatten_1 (Flatten)	(None, 16384)	0
dropout_1 (Dropout)	(None, 16384)	0
reshape_1 (Reshape)	(None, 16, 1024)	0
lstm_1 (LSTM)	(None, 16, 50)	215000
lstm_2 (LSTM)	(None, 20)	5680
dense_1 (Dense)	(None, 7)	147

```
Total params: 296,475
Trainable params: 296,475
Non-trainable params: 0
```

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

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

from keras.preprocessing.image import ImageDataGenerator

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

test_datagen = ImageDataGenerator(rescale = 1./255)

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

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

```
Found 3960 images belonging to 7 classes.
Found 1320 images belonging to 7 classes.
```

In [4]:
```
results = classifier.fit_generator(training_set,
                         samples_per_epoch = 3960,
                         nb_epoch = 100,
                         validation_data = test_set,
                         nb_val_samples = 1320)
```

```
Epoch 1/100
123/123 [==============================] - 1666s 14s/step - loss: 1.8975 - ac
c: 0.2580 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 1.7336 - v
al_acc: 0.3042 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/100
123/123 [==============================] - 1670s 14s/step - loss: 1.7140 - ac
c: 0.3191 - precision: 0.4906 - recall: 0.0716 - val_loss: 1.5569 - val_acc:
0.3510 - val_precision: 0.6223 - val_recall: 0.1023
Epoch 3/100
123/123 [==============================] - 1672s 14s/step - loss: 1.6481 - ac
c: 0.3218 - precision: 0.5623 - recall: 0.1017 - val_loss: 1.5345 - val_acc:
0.3741 - val_precision: 0.5933 - val_recall: 0.0499
Epoch 4/100
123/123 [==============================] - 1669s 14s/step - loss: 1.6069 - ac
c: 0.3444 - precision: 0.5926 - recall: 0.1275 - val_loss: 1.4606 - val_acc:
0.3865 - val_precision: 0.6224 - val_recall: 0.1804
Epoch 5/100
123/123 [==============================] - 1677s 14s/step - loss: 1.5628 - ac
c: 0.3683 - precision: 0.6108 - recall: 0.1276 - val_loss: 1.4638 - val_acc:
0.3999 - val_precision: 0.6296 - val_recall: 0.1475
Epoch 6/100
123/123 [==============================] - 1670s 14s/step - loss: 1.5411 - ac
c: 0.3686 - precision: 0.6092 - recall: 0.1344 - val_loss: 1.3739 - val_acc:
0.4275 - val_precision: 0.6538 - val_recall: 0.1932
Epoch 7/100
123/123 [==============================] - 1674s 14s/step - loss: 1.5198 - ac
c: 0.3836 - precision: 0.6268 - recall: 0.1358 - val_loss: 1.4771 - val_acc:
0.3910 - val_precision: 0.6984 - val_recall: 0.1402
Epoch 8/100
123/123 [==============================] - 1678s 14s/step - loss: 1.4745 - ac
c: 0.4098 - precision: 0.6496 - recall: 0.1542 - val_loss: 1.4014 - val_acc:
0.4391 - val_precision: 0.6858 - val_recall: 0.1999
Epoch 9/100
123/123 [==============================] - 1682s 14s/step - loss: 1.4144 - ac
c: 0.4350 - precision: 0.6569 - recall: 0.1875 - val_loss: 1.2835 - val_acc:
0.4710 - val_precision: 0.7474 - val_recall: 0.1798
Epoch 10/100
123/123 [==============================] - 1669s 14s/step - loss: 1.3772 - ac
c: 0.4572 - precision: 0.6718 - recall: 0.2060 - val_loss: 1.1897 - val_acc:
0.5283 - val_precision: 0.6874 - val_recall: 0.2941
Epoch 11/100
123/123 [==============================] - 1673s 14s/step - loss: 1.3651 - ac
c: 0.4608 - precision: 0.6655 - recall: 0.2047 - val_loss: 1.1196 - val_acc:
0.5449 - val_precision: 0.7172 - val_recall: 0.3262
Epoch 12/100
123/123 [==============================] - 1715s 14s/step - loss: 1.3230 - ac
c: 0.4813 - precision: 0.6507 - recall: 0.2417 - val_loss: 1.0873 - val_acc:
0.5634 - val_precision: 0.7644 - val_recall: 0.3286
Epoch 13/100
123/123 [==============================] - 1681s 14s/step - loss: 1.2813 - ac
c: 0.5005 - precision: 0.6888 - recall: 0.2632 - val_loss: 1.0763 - val_acc:
0.5716 - val_precision: 0.7008 - val_recall: 0.3480
Epoch 14/100
123/123 [==============================] - 1676s 14s/step - loss: 1.2447 - ac
c: 0.5142 - precision: 0.6901 - recall: 0.2890 - val_loss: 1.0099 - val_acc:
0.5889 - val_precision: 0.7311 - val_recall: 0.3971
Epoch 15/100
```

```
123/123 [==============================] - 1676s 14s/step - loss: 1.2238 - ac
c: 0.5163 - precision: 0.6941 - recall: 0.2993 - val_loss: 1.0212 - val_acc:
0.5864 - val_precision: 0.7078 - val_recall: 0.4060
Epoch 16/100
123/123 [==============================] - 1675s 14s/step - loss: 1.2081 - ac
c: 0.5223 - precision: 0.6909 - recall: 0.3073 - val_loss: 1.0105 - val_acc:
0.5845 - val_precision: 0.7217 - val_recall: 0.4120
Epoch 17/100
123/123 [==============================] - 1679s 14s/step - loss: 1.1917 - ac
c: 0.5326 - precision: 0.6888 - recall: 0.3319 - val_loss: 0.9666 - val_acc:
0.6100 - val_precision: 0.7074 - val_recall: 0.4440
Epoch 18/100
123/123 [==============================] - 1672s 14s/step - loss: 1.1808 - ac
c: 0.5309 - precision: 0.6942 - recall: 0.3336 - val_loss: 0.9067 - val_acc:
0.6534 - val_precision: 0.7742 - val_recall: 0.4338
Epoch 19/100
123/123 [==============================] - 1683s 14s/step - loss: 1.1426 - ac
c: 0.5569 - precision: 0.7059 - recall: 0.3534 - val_loss: 0.9754 - val_acc:
0.6182 - val_precision: 0.6947 - val_recall: 0.4509
Epoch 20/100
123/123 [==============================] - 1676s 14s/step - loss: 1.0940 - ac
c: 0.5688 - precision: 0.7127 - recall: 0.3747 - val_loss: 0.8945 - val_acc:
0.6696 - val_precision: 0.7591 - val_recall: 0.4915
Epoch 21/100
123/123 [==============================] - 1678s 14s/step - loss: 1.1065 - ac
c: 0.5650 - precision: 0.7046 - recall: 0.3808 - val_loss: 0.8344 - val_acc:
0.6947 - val_precision: 0.7676 - val_recall: 0.5364
Epoch 22/100
123/123 [==============================] - 1677s 14s/step - loss: 1.0971 - ac
c: 0.5777 - precision: 0.6963 - recall: 0.3877 - val_loss: 0.9292 - val_acc:
0.6362 - val_precision: 0.7054 - val_recall: 0.5159
Epoch 23/100
123/123 [==============================] - 1677s 14s/step - loss: 1.0447 - ac
c: 0.5815 - precision: 0.7123 - recall: 0.4201 - val_loss: 0.8250 - val_acc:
0.6706 - val_precision: 0.7393 - val_recall: 0.5668
Epoch 24/100
123/123 [==============================] - 1675s 14s/step - loss: 1.0335 - ac
c: 0.6069 - precision: 0.7146 - recall: 0.4432 - val_loss: 0.7901 - val_acc:
0.6885 - val_precision: 0.7855 - val_recall: 0.5778
Epoch 25/100
123/123 [==============================] - 1673s 14s/step - loss: 1.0108 - ac
c: 0.6026 - precision: 0.7149 - recall: 0.4568 - val_loss: 0.8481 - val_acc:
0.6545 - val_precision: 0.7267 - val_recall: 0.5629
Epoch 26/100
123/123 [==============================] - 1675s 14s/step - loss: 1.0105 - ac
c: 0.6088 - precision: 0.7137 - recall: 0.4498 - val_loss: 0.7485 - val_acc:
0.7131 - val_precision: 0.7694 - val_recall: 0.6303
Epoch 27/100
123/123 [==============================] - 1673s 14s/step - loss: 0.9844 - ac
c: 0.6237 - precision: 0.7269 - recall: 0.4747 - val_loss: 0.7266 - val_acc:
0.7263 - val_precision: 0.7937 - val_recall: 0.6375
Epoch 28/100
123/123 [==============================] - 1695s 14s/step - loss: 0.9823 - ac
c: 0.6249 - precision: 0.7201 - recall: 0.4853 - val_loss: 0.6933 - val_acc:
0.7385 - val_precision: 0.7955 - val_recall: 0.6506
Epoch 29/100
123/123 [==============================] - 1690s 14s/step - loss: 0.9346 - ac
```

```
c: 0.6456 - precision: 0.7389 - recall: 0.5159 - val_loss: 0.6852 - val_acc:
0.7446 - val_precision: 0.7992 - val_recall: 0.6762
Epoch 30/100
123/123 [==============================] - 1676s 14s/step - loss: 0.9200 - ac
c: 0.6413 - precision: 0.7431 - recall: 0.5288 - val_loss: 0.6326 - val_acc:
0.7771 - val_precision: 0.8355 - val_recall: 0.7088
Epoch 31/100
123/123 [==============================] - 1675s 14s/step - loss: 0.8975 - ac
c: 0.6625 - precision: 0.7534 - recall: 0.5407 - val_loss: 0.6162 - val_acc:
0.7855 - val_precision: 0.8283 - val_recall: 0.7128
Epoch 32/100
123/123 [==============================] - 1674s 14s/step - loss: 0.8680 - ac
c: 0.6761 - precision: 0.7501 - recall: 0.5617 - val_loss: 0.5964 - val_acc:
0.7870 - val_precision: 0.8270 - val_recall: 0.7386
Epoch 33/100
123/123 [==============================] - 1680s 14s/step - loss: 0.8885 - ac
c: 0.6585 - precision: 0.7393 - recall: 0.5567 - val_loss: 0.6459 - val_acc:
0.7696 - val_precision: 0.8205 - val_recall: 0.6968
Epoch 34/100
123/123 [==============================] - 1672s 14s/step - loss: 0.8477 - ac
c: 0.6850 - precision: 0.7515 - recall: 0.5741 - val_loss: 0.5449 - val_acc:
0.8025 - val_precision: 0.8507 - val_recall: 0.7502
Epoch 35/100
123/123 [==============================] - 1685s 14s/step - loss: 0.8479 - ac
c: 0.6773 - precision: 0.7548 - recall: 0.5823 - val_loss: 0.5331 - val_acc:
0.8082 - val_precision: 0.8576 - val_recall: 0.7627
Epoch 36/100
123/123 [==============================] - 1678s 14s/step - loss: 0.8072 - ac
c: 0.6988 - precision: 0.7735 - recall: 0.6129 - val_loss: 0.5356 - val_acc:
0.8024 - val_precision: 0.8367 - val_recall: 0.7674
Epoch 37/100
123/123 [==============================] - 1677s 14s/step - loss: 0.7833 - ac
c: 0.7089 - precision: 0.7728 - recall: 0.6231 - val_loss: 0.5720 - val_acc:
0.7850 - val_precision: 0.8145 - val_recall: 0.7448
Epoch 38/100
123/123 [==============================] - 1677s 14s/step - loss: 0.7954 - ac
c: 0.6961 - precision: 0.7659 - recall: 0.6143 - val_loss: 0.4872 - val_acc:
0.8220 - val_precision: 0.8513 - val_recall: 0.7756
Epoch 39/100
123/123 [==============================] - 1670s 14s/step - loss: 0.7701 - ac
c: 0.7051 - precision: 0.7691 - recall: 0.6306 - val_loss: 0.5657 - val_acc:
0.7870 - val_precision: 0.8198 - val_recall: 0.7537
Epoch 40/100
123/123 [==============================] - 1675s 14s/step - loss: 0.7414 - ac
c: 0.7225 - precision: 0.7898 - recall: 0.6542 - val_loss: 0.5262 - val_acc:
0.8136 - val_precision: 0.8371 - val_recall: 0.7818
Epoch 41/100
123/123 [==============================] - 1679s 14s/step - loss: 0.7378 - ac
c: 0.7261 - precision: 0.7846 - recall: 0.6552 - val_loss: 0.5114 - val_acc:
0.8157 - val_precision: 0.8440 - val_recall: 0.7838
Epoch 42/100
123/123 [==============================] - 1673s 14s/step - loss: 0.7470 - ac
c: 0.7264 - precision: 0.7838 - recall: 0.6498 - val_loss: 0.4207 - val_acc:
0.8490 - val_precision: 0.8806 - val_recall: 0.8262
Epoch 43/100
123/123 [==============================] - 1675s 14s/step - loss: 0.7001 - ac
c: 0.7381 - precision: 0.7900 - recall: 0.6803 - val_loss: 0.6045 - val_acc:
```

```
0.7741 - val_precision: 0.8177 - val_recall: 0.7452
Epoch 44/100
123/123 [==============================] - 1679s 14s/step - loss: 0.7222 - ac
c: 0.7371 - precision: 0.7917 - recall: 0.6709 - val_loss: 0.4157 - val_acc:
0.8470 - val_precision: 0.8794 - val_recall: 0.8227
Epoch 45/100
123/123 [==============================] - 1681s 14s/step - loss: 0.6859 - ac
c: 0.7491 - precision: 0.8002 - recall: 0.6878 - val_loss: 0.3928 - val_acc:
0.8554 - val_precision: 0.8717 - val_recall: 0.8274
Epoch 46/100
123/123 [==============================] - 1733s 14s/step - loss: 0.7016 - ac
c: 0.7416 - precision: 0.7947 - recall: 0.6855 - val_loss: 0.4112 - val_acc:
0.8561 - val_precision: 0.8742 - val_recall: 0.8213
Epoch 47/100
123/123 [==============================] - 1718s 14s/step - loss: 0.6765 - ac
c: 0.7455 - precision: 0.7986 - recall: 0.6915 - val_loss: 0.3821 - val_acc:
0.8672 - val_precision: 0.8870 - val_recall: 0.8520
Epoch 48/100
123/123 [==============================] - 1682s 14s/step - loss: 0.6390 - ac
c: 0.7708 - precision: 0.8211 - recall: 0.7231 - val_loss: 0.3519 - val_acc:
0.8824 - val_precision: 0.8938 - val_recall: 0.8627
Epoch 49/100
123/123 [==============================] - 1685s 14s/step - loss: 0.6364 - ac
c: 0.7675 - precision: 0.8091 - recall: 0.7149 - val_loss: 0.3583 - val_acc:
0.8828 - val_precision: 0.9007 - val_recall: 0.8637
Epoch 50/100
123/123 [==============================] - 1679s 14s/step - loss: 0.6540 - ac
c: 0.7588 - precision: 0.8085 - recall: 0.7108 - val_loss: 0.4984 - val_acc:
0.8260 - val_precision: 0.8435 - val_recall: 0.7996
Epoch 51/100
123/123 [==============================] - 1686s 14s/step - loss: 0.6113 - ac
c: 0.7767 - precision: 0.8206 - recall: 0.7264 - val_loss: 0.3744 - val_acc:
0.8615 - val_precision: 0.8827 - val_recall: 0.8471
Epoch 52/100
123/123 [==============================] - 1685s 14s/step - loss: 0.5902 - ac
c: 0.7917 - precision: 0.8335 - recall: 0.7447 - val_loss: 0.3151 - val_acc:
0.8997 - val_precision: 0.9128 - val_recall: 0.8837
Epoch 53/100
123/123 [==============================] - 1691s 14s/step - loss: 0.6148 - ac
c: 0.7797 - precision: 0.8196 - recall: 0.7284 - val_loss: 0.3271 - val_acc:
0.8895 - val_precision: 0.9062 - val_recall: 0.8781
Epoch 54/100
123/123 [==============================] - 1689s 14s/step - loss: 0.5530 - ac
c: 0.7982 - precision: 0.8376 - recall: 0.7595 - val_loss: 0.3351 - val_acc:
0.8932 - val_precision: 0.9020 - val_recall: 0.8780
Epoch 55/100
123/123 [==============================] - 1686s 14s/step - loss: 0.5757 - ac
c: 0.7916 - precision: 0.8287 - recall: 0.7511 - val_loss: 0.3037 - val_acc:
0.8986 - val_precision: 0.9129 - val_recall: 0.8812
Epoch 56/100
123/123 [==============================] - 1682s 14s/step - loss: 0.5849 - ac
c: 0.7843 - precision: 0.8260 - recall: 0.7416 - val_loss: 0.2821 - val_acc:
0.9060 - val_precision: 0.9187 - val_recall: 0.8909
Epoch 57/100
123/123 [==============================] - 1689s 14s/step - loss: 0.5486 - ac
c: 0.8014 - precision: 0.8382 - recall: 0.7653 - val_loss: 0.2798 - val_acc:
0.9051 - val_precision: 0.9150 - val_recall: 0.8908
```

```
Epoch 58/100
123/123 [==============================] - 1683s 14s/step - loss: 0.5425 - ac
c: 0.8095 - precision: 0.8425 - recall: 0.7717 - val_loss: 0.2770 - val_acc:
0.9061 - val_precision: 0.9168 - val_recall: 0.8939
Epoch 59/100
123/123 [==============================] - 1699s 14s/step - loss: 0.5336 - ac
c: 0.8057 - precision: 0.8482 - recall: 0.7727 - val_loss: 0.2178 - val_acc:
0.9318 - val_precision: 0.9449 - val_recall: 0.9211
Epoch 60/100
123/123 [==============================] - 1684s 14s/step - loss: 0.5189 - ac
c: 0.8150 - precision: 0.8462 - recall: 0.7782 - val_loss: 0.2568 - val_acc:
0.9137 - val_precision: 0.9191 - val_recall: 0.9024
Epoch 61/100
123/123 [==============================] - 1677s 14s/step - loss: 0.5114 - ac
c: 0.8209 - precision: 0.8528 - recall: 0.7834 - val_loss: 0.2567 - val_acc:
0.9141 - val_precision: 0.9265 - val_recall: 0.8997
Epoch 62/100
123/123 [==============================] - 1681s 14s/step - loss: 0.5034 - ac
c: 0.8196 - precision: 0.8562 - recall: 0.7908 - val_loss: 0.2628 - val_acc:
0.9070 - val_precision: 0.9222 - val_recall: 0.8964
Epoch 63/100
123/123 [==============================] - 1691s 14s/step - loss: 0.5045 - ac
c: 0.8226 - precision: 0.8532 - recall: 0.7892 - val_loss: 0.3107 - val_acc:
0.8970 - val_precision: 0.9146 - val_recall: 0.8834
Epoch 64/100
123/123 [==============================] - 1679s 14s/step - loss: 0.5119 - ac
c: 0.8116 - precision: 0.8467 - recall: 0.7798 - val_loss: 0.1882 - val_acc:
0.9394 - val_precision: 0.9520 - val_recall: 0.9311
Epoch 65/100
123/123 [==============================] - 1678s 14s/step - loss: 0.4866 - ac
c: 0.8298 - precision: 0.8593 - recall: 0.7975 - val_loss: 0.2079 - val_acc:
0.9425 - val_precision: 0.9479 - val_recall: 0.9365
Epoch 66/100
123/123 [==============================] - 1691s 14s/step - loss: 0.4794 - ac
c: 0.8238 - precision: 0.8524 - recall: 0.7939 - val_loss: 0.2677 - val_acc:
0.9061 - val_precision: 0.9144 - val_recall: 0.8971
Epoch 67/100
123/123 [==============================] - 1691s 14s/step - loss: 0.4515 - ac
c: 0.8357 - precision: 0.8628 - recall: 0.8142 - val_loss: 0.2130 - val_acc:
0.9342 - val_precision: 0.9397 - val_recall: 0.9311
Epoch 68/100
123/123 [==============================] - 1689s 14s/step - loss: 0.4654 - ac
c: 0.8387 - precision: 0.8673 - recall: 0.8100 - val_loss: 0.1977 - val_acc:
0.9439 - val_precision: 0.9516 - val_recall: 0.9385
Epoch 69/100
123/123 [==============================] - 1684s 14s/step - loss: 0.4689 - ac
c: 0.8368 - precision: 0.8662 - recall: 0.8058 - val_loss: 0.1936 - val_acc:
0.9394 - val_precision: 0.9492 - val_recall: 0.9326
Epoch 70/100
123/123 [==============================] - 1679s 14s/step - loss: 0.4635 - ac
c: 0.8328 - precision: 0.8621 - recall: 0.8041 - val_loss: 0.1619 - val_acc:
0.9516 - val_precision: 0.9607 - val_recall: 0.9447
Epoch 71/100
123/123 [==============================] - 1691s 14s/step - loss: 0.4709 - ac
c: 0.8357 - precision: 0.8651 - recall: 0.8074 - val_loss: 0.1757 - val_acc:
0.9454 - val_precision: 0.9531 - val_recall: 0.9393
Epoch 72/100
```

```
123/123 [==============================] - 1690s 14s/step - loss: 0.4305 - ac
c: 0.8486 - precision: 0.8759 - recall: 0.8229 - val_loss: 0.1584 - val_acc:
0.9562 - val_precision: 0.9652 - val_recall: 0.9463
Epoch 73/100
123/123 [==============================] - 1688s 14s/step - loss: 0.4378 - ac
c: 0.8466 - precision: 0.8720 - recall: 0.8225 - val_loss: 0.1627 - val_acc:
0.9575 - val_precision: 0.9618 - val_recall: 0.9552
Epoch 74/100
123/123 [==============================] - 1681s 14s/step - loss: 0.4352 - ac
c: 0.8442 - precision: 0.8692 - recall: 0.8173 - val_loss: 0.2960 - val_acc:
0.8969 - val_precision: 0.9078 - val_recall: 0.8878
Epoch 75/100
123/123 [==============================] - 1689s 14s/step - loss: 0.4308 - ac
c: 0.8479 - precision: 0.8693 - recall: 0.8258 - val_loss: 0.1320 - val_acc:
0.9652 - val_precision: 0.9709 - val_recall: 0.9621
Epoch 76/100
123/123 [==============================] - 1692s 14s/step - loss: 0.4168 - ac
c: 0.8540 - precision: 0.8809 - recall: 0.8314 - val_loss: 0.2131 - val_acc:
0.9368 - val_precision: 0.9438 - val_recall: 0.9331
Epoch 77/100
123/123 [==============================] - 1685s 14s/step - loss: 0.4271 - ac
c: 0.8497 - precision: 0.8756 - recall: 0.8263 - val_loss: 0.1846 - val_acc:
0.9447 - val_precision: 0.9480 - val_recall: 0.9372
Epoch 78/100
123/123 [==============================] - 1686s 14s/step - loss: 0.3865 - ac
c: 0.8695 - precision: 0.8881 - recall: 0.8492 - val_loss: 0.1598 - val_acc:
0.9476 - val_precision: 0.9548 - val_recall: 0.9468
Epoch 79/100
123/123 [==============================] - 1687s 14s/step - loss: 0.3946 - ac
c: 0.8685 - precision: 0.8852 - recall: 0.8460 - val_loss: 0.1359 - val_acc:
0.9622 - val_precision: 0.9665 - val_recall: 0.9584
Epoch 80/100
123/123 [==============================] - 1685s 14s/step - loss: 0.3827 - ac
c: 0.8684 - precision: 0.8901 - recall: 0.8514 - val_loss: 0.1578 - val_acc:
0.9545 - val_precision: 0.9588 - val_recall: 0.9522
Epoch 81/100
123/123 [==============================] - 1699s 14s/step - loss: 0.3935 - ac
c: 0.8653 - precision: 0.8896 - recall: 0.8448 - val_loss: 0.1533 - val_acc:
0.9576 - val_precision: 0.9670 - val_recall: 0.9546
Epoch 82/100
123/123 [==============================] - 1701s 14s/step - loss: 0.3568 - ac
c: 0.8775 - precision: 0.8986 - recall: 0.8618 - val_loss: 0.1468 - val_acc:
0.9560 - val_precision: 0.9616 - val_recall: 0.9485
Epoch 83/100
123/123 [==============================] - 1698s 14s/step - loss: 0.3716 - ac
c: 0.8717 - precision: 0.8923 - recall: 0.8528 - val_loss: 0.1150 - val_acc:
0.9659 - val_precision: 0.9694 - val_recall: 0.9613
Epoch 84/100
123/123 [==============================] - 1695s 14s/step - loss: 0.3844 - ac
c: 0.8712 - precision: 0.8930 - recall: 0.8542 - val_loss: 0.1057 - val_acc:
0.9719 - val_precision: 0.9741 - val_recall: 0.9704
Epoch 85/100
123/123 [==============================] - 1695s 14s/step - loss: 0.3383 - ac
c: 0.8836 - precision: 0.9015 - recall: 0.8653 - val_loss: 0.1041 - val_acc:
0.9705 - val_precision: 0.9727 - val_recall: 0.9705
Epoch 86/100
123/123 [==============================] - 1687s 14s/step - loss: 0.3334 - ac
```

```
c: 0.8887 - precision: 0.9050 - recall: 0.8742 - val_loss: 0.0998 - val_acc:
0.9758 - val_precision: 0.9772 - val_recall: 0.9735
Epoch 87/100
123/123 [==============================] - 1701s 14s/step - loss: 0.3576 - ac
c: 0.8780 - precision: 0.8913 - recall: 0.8645 - val_loss: 0.0911 - val_acc:
0.9773 - val_precision: 0.9802 - val_recall: 0.9735
Epoch 88/100
123/123 [==============================] - 1710s 14s/step - loss: 0.3693 - ac
c: 0.8743 - precision: 0.8920 - recall: 0.8583 - val_loss: 0.1013 - val_acc:
0.9727 - val_precision: 0.9771 - val_recall: 0.9705
Epoch 89/100
123/123 [==============================] - 1715s 14s/step - loss: 0.3542 - ac
c: 0.8852 - precision: 0.9012 - recall: 0.8689 - val_loss: 0.1415 - val_acc:
0.9507 - val_precision: 0.9563 - val_recall: 0.9455
Epoch 90/100
123/123 [==============================] - 1701s 14s/step - loss: 0.3326 - ac
c: 0.8893 - precision: 0.9044 - recall: 0.8725 - val_loss: 0.1034 - val_acc:
0.9713 - val_precision: 0.9734 - val_recall: 0.9690
Epoch 91/100
123/123 [==============================] - 1699s 14s/step - loss: 0.3490 - ac
c: 0.8789 - precision: 0.8953 - recall: 0.8620 - val_loss: 0.0990 - val_acc:
0.9713 - val_precision: 0.9727 - val_recall: 0.9675
Epoch 92/100
123/123 [==============================] - 1694s 14s/step - loss: 0.3505 - ac
c: 0.8817 - precision: 0.9007 - recall: 0.8667 - val_loss: 0.1108 - val_acc:
0.9658 - val_precision: 0.9701 - val_recall: 0.9636
Epoch 93/100
123/123 [==============================] - 1694s 14s/step - loss: 0.3436 - ac
c: 0.8854 - precision: 0.9032 - recall: 0.8684 - val_loss: 0.1275 - val_acc:
0.9628 - val_precision: 0.9657 - val_recall: 0.9598
Epoch 94/100
123/123 [==============================] - 1691s 14s/step - loss: 0.3087 - ac
c: 0.8984 - precision: 0.9138 - recall: 0.8814 - val_loss: 0.1100 - val_acc:
0.9696 - val_precision: 0.9725 - val_recall: 0.9666
Epoch 95/100
123/123 [==============================] - 1702s 14s/step - loss: 0.3210 - ac
c: 0.8877 - precision: 0.9053 - recall: 0.8703 - val_loss: 0.0928 - val_acc:
0.9742 - val_precision: 0.9757 - val_recall: 0.9727
Epoch 96/100
123/123 [==============================] - 1697s 14s/step - loss: 0.3296 - ac
c: 0.8890 - precision: 0.9023 - recall: 0.8725 - val_loss: 0.0899 - val_acc:
0.9750 - val_precision: 0.9772 - val_recall: 0.9735
Epoch 97/100
123/123 [==============================] - 1699s 14s/step - loss: 0.3179 - ac
c: 0.8930 - precision: 0.9106 - recall: 0.8787 - val_loss: 0.1057 - val_acc:
0.9696 - val_precision: 0.9726 - val_recall: 0.9674
Epoch 98/100
123/123 [==============================] - 1682s 14s/step - loss: 0.3051 - ac
c: 0.8956 - precision: 0.9089 - recall: 0.8801 - val_loss: 0.1046 - val_acc:
0.9720 - val_precision: 0.9741 - val_recall: 0.9705
Epoch 99/100
123/123 [==============================] - 1477s 12s/step - loss: 0.3008 - ac
c: 0.8941 - precision: 0.9073 - recall: 0.8852 - val_loss: 0.1150 - val_acc:
0.9628 - val_precision: 0.9664 - val_recall: 0.9598
Epoch 100/100
123/123 [==============================] - 1281s 10s/step - loss: 0.2955 - ac
```

```
c: 0.8996 - precision: 0.9140 - recall: 0.8877 - val_loss: 0.0721 - val_acc:
0.9810 - val_precision: 0.9832 - val_recall: 0.9788
```

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)
```

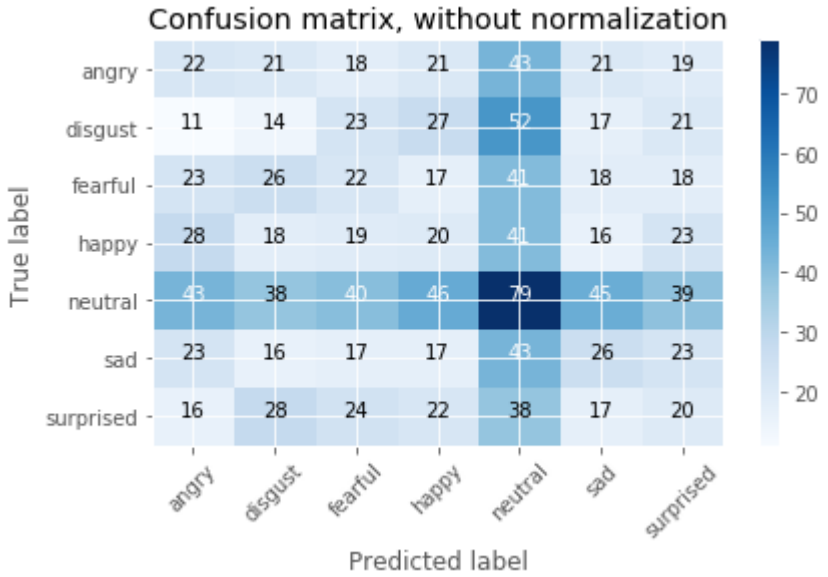
In [6]:
```
true_classes = test_set.classes
class_labels = list(test_set.class_indices.keys())
```

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

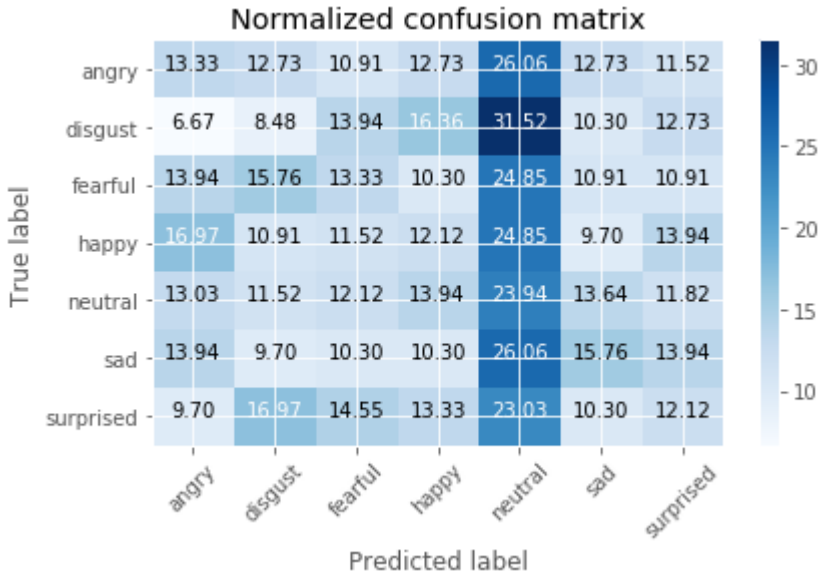
```
             precision    recall  f1-score   support

      angry       0.13      0.13      0.13       165
    disgust       0.09      0.08      0.09       165
    fearful       0.13      0.13      0.13       165
      happy       0.12      0.12      0.12       165
    neutral       0.23      0.24      0.24       330
        sad       0.16      0.16      0.16       165
  surprised       0.12      0.12      0.12       165

avg / total       0.15      0.15      0.15      1320
```

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
[[22 21 18 21 43 21 19]
 [11 14 23 27 52 17 21]
 [23 26 22 17 41 18 18]
 [28 18 19 20 41 16 23]
 [43 38 40 46 79 45 39]
 [23 16 17 17 43 26 23]
 [16 28 24 22 38 17 20]]
```




```
Normalized confusion matrix
[[13.3333 12.7273 10.9091 12.7273 26.0606 12.7273 11.5152]
 [ 6.6667  8.4848 13.9394 16.3636 31.5152 10.303  12.7273]
 [13.9394 15.7576 13.3333 10.303  24.8485 10.9091 10.9091]
 [16.9697 10.9091 11.5152 12.1212 24.8485  9.697  13.9394]
 [13.0303 11.5152 12.1212 13.9394 23.9394 13.6364 11.8182]
 [13.9394  9.697  10.303  10.303  26.0606 15.7576 13.9394]
 [ 9.697  16.9697 14.5455 13.3333 23.0303 10.303  12.1212]]
```



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")
```