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 11880 images belonging to 8 classes.
Found 3960 images belonging to 8 classes.
```

```
In [4]: results = classifier.fit_generator(training_set,
                                           samples_per_epoch = 11880,
                                           nb_epoch = 100,
                                           validation_data = test_set,
                                           nb_val_samples = 3960)
```

```
Epoch 1/100
371/371 [==============================] - 2651s 7s/step - loss: 1.9134 - ac
c: 0.2525 - precision: 0.2339 - recall: 0.0154 - val_loss: 1.6542 - val_acc:
0.3987 - val_precision: 0.4308 - val_recall: 0.0182
Epoch 2/100
371/371 [==============================] - 2641s 7s/step - loss: 1.7262 - ac
c: 0.3483 - precision: 0.5681 - recall: 0.0724 - val_loss: 1.4663 - val_acc:
0.4499 - val_precision: 0.6743 - val_recall: 0.1611
Epoch 3/100
371/371 [==============================] - 2634s 7s/step - loss: 1.6100 - ac
c: 0.3857 - precision: 0.6346 - recall: 0.1267 - val_loss: 1.4009 - val_acc:
0.4586 - val_precision: 0.7015 - val_recall: 0.1901
Epoch 4/100
371/371 [==============================] - 2646s 7s/step - loss: 1.5437 - ac
c: 0.4123 - precision: 0.6498 - recall: 0.1596 - val_loss: 1.2604 - val_acc:
0.5214 - val_precision: 0.7574 - val_recall: 0.2513
Epoch 5/100
371/371 [==============================] - 2639s 7s/step - loss: 1.4576 - ac
c: 0.4512 - precision: 0.6678 - recall: 0.2126 - val_loss: 1.2056 - val_acc:
0.5505 - val_precision: 0.7436 - val_recall: 0.3056
Epoch 6/100
371/371 [==============================] - 2633s 7s/step - loss: 1.3995 - ac
c: 0.4757 - precision: 0.6761 - recall: 0.2402 - val_loss: 1.0889 - val_acc:
0.6026 - val_precision: 0.7699 - val_recall: 0.3889
Epoch 7/100
371/371 [==============================] - 2630s 7s/step - loss: 1.3336 - ac
c: 0.5023 - precision: 0.7003 - recall: 0.2840 - val_loss: 1.0546 - val_acc:
0.5979 - val_precision: 0.7957 - val_recall: 0.3914
Epoch 8/100
371/371 [==============================] - 2635s 7s/step - loss: 1.3028 - ac
c: 0.5109 - precision: 0.6993 - recall: 0.3007 - val_loss: 0.9451 - val_acc:
0.6518 - val_precision: 0.8366 - val_recall: 0.4698
Epoch 9/100
371/371 [==============================] - 2626s 7s/step - loss: 1.2556 - ac
c: 0.5291 - precision: 0.7208 - recall: 0.3283 - val_loss: 0.9156 - val_acc:
0.6537 - val_precision: 0.8062 - val_recall: 0.5214
Epoch 10/100
371/371 [==============================] - 2628s 7s/step - loss: 1.2123 - ac
c: 0.5455 - precision: 0.7230 - recall: 0.3497 - val_loss: 0.8637 - val_acc:
0.6753 - val_precision: 0.8436 - val_recall: 0.5176
Epoch 11/100
371/371 [==============================] - 2625s 7s/step - loss: 1.1818 - ac
c: 0.5549 - precision: 0.7255 - recall: 0.3710 - val_loss: 0.7984 - val_acc:
0.6955 - val_precision: 0.8376 - val_recall: 0.5677
Epoch 12/100
371/371 [==============================] - 2634s 7s/step - loss: 1.1457 - ac
c: 0.5709 - precision: 0.7352 - recall: 0.3982 - val_loss: 0.8224 - val_acc:
0.7027 - val_precision: 0.8365 - val_recall: 0.5230
Epoch 13/100
371/371 [==============================] - 2630s 7s/step - loss: 1.1163 - ac
c: 0.5855 - precision: 0.7464 - recall: 0.4147 - val_loss: 0.7626 - val_acc:
0.7371 - val_precision: 0.8876 - val_recall: 0.5490
Epoch 14/100
371/371 [==============================] - 2627s 7s/step - loss: 1.0928 - ac
c: 0.5956 - precision: 0.7445 - recall: 0.4267 - val_loss: 0.7641 - val_acc:
0.7253 - val_precision: 0.8727 - val_recall: 0.5604
Epoch 15/100
```

```
371/371 [==============================] - 2623s 7s/step - loss: 1.0835 - ac
c: 0.5974 - precision: 0.7502 - recall: 0.4286 - val_loss: 0.7162 - val_acc:
0.7217 - val_precision: 0.8376 - val_recall: 0.6255
Epoch 16/100
371/371 [==============================] - 2624s 7s/step - loss: 1.0383 - ac
c: 0.6133 - precision: 0.7550 - recall: 0.4611 - val_loss: 0.6546 - val_acc:
0.7614 - val_precision: 0.8796 - val_recall: 0.6336
Epoch 17/100
371/371 [==============================] - 2617s 7s/step - loss: 1.0316 - ac
c: 0.6159 - precision: 0.7571 - recall: 0.4678 - val_loss: 0.6213 - val_acc:
0.7707 - val_precision: 0.8678 - val_recall: 0.6647
Epoch 18/100
371/371 [==============================] - 2615s 7s/step - loss: 1.0094 - ac
c: 0.6214 - precision: 0.7628 - recall: 0.4794 - val_loss: 0.6003 - val_acc:
0.7833 - val_precision: 0.8855 - val_recall: 0.6669
Epoch 19/100
371/371 [==============================] - 2614s 7s/step - loss: 1.0008 - ac
c: 0.6279 - precision: 0.7521 - recall: 0.4796 - val_loss: 0.6221 - val_acc:
0.7669 - val_precision: 0.8565 - val_recall: 0.6748
Epoch 20/100
371/371 [==============================] - 2621s 7s/step - loss: 0.9710 - ac
c: 0.6363 - precision: 0.7731 - recall: 0.5007 - val_loss: 0.6529 - val_acc:
0.7612 - val_precision: 0.8492 - val_recall: 0.6751
Epoch 21/100
371/371 [==============================] - 2666s 7s/step - loss: 0.9615 - ac
c: 0.6405 - precision: 0.7710 - recall: 0.5050 - val_loss: 0.5657 - val_acc:
0.8013 - val_precision: 0.8947 - val_recall: 0.6904
Epoch 22/100
371/371 [==============================] - 2730s 7s/step - loss: 0.9494 - ac
c: 0.6492 - precision: 0.7679 - recall: 0.5171 - val_loss: 0.5377 - val_acc:
0.8035 - val_precision: 0.8847 - val_recall: 0.7316
Epoch 23/100
371/371 [==============================] - 2708s 7s/step - loss: 0.9201 - ac
c: 0.6531 - precision: 0.7711 - recall: 0.5270 - val_loss: 0.5528 - val_acc:
0.8013 - val_precision: 0.8874 - val_recall: 0.7169
Epoch 24/100
371/371 [==============================] - 2699s 7s/step - loss: 0.9355 - ac
c: 0.6536 - precision: 0.7706 - recall: 0.5235 - val_loss: 0.5455 - val_acc:
0.8086 - val_precision: 0.8838 - val_recall: 0.7277
Epoch 25/100
371/371 [==============================] - 2669s 7s/step - loss: 0.8954 - ac
c: 0.6649 - precision: 0.7784 - recall: 0.5471 - val_loss: 0.4945 - val_acc:
0.8237 - val_precision: 0.8972 - val_recall: 0.7426
Epoch 26/100
371/371 [==============================] - 2629s 7s/step - loss: 0.9027 - ac
c: 0.6629 - precision: 0.7811 - recall: 0.5436 - val_loss: 0.5733 - val_acc:
0.7874 - val_precision: 0.8738 - val_recall: 0.7255
Epoch 27/100
371/371 [==============================] - 2646s 7s/step - loss: 0.8885 - ac
c: 0.6720 - precision: 0.7812 - recall: 0.5505 - val_loss: 0.4915 - val_acc:
0.8339 - val_precision: 0.8897 - val_recall: 0.7472
Epoch 28/100
371/371 [==============================] - 2614s 7s/step - loss: 0.8848 - ac
c: 0.6683 - precision: 0.7818 - recall: 0.5488 - val_loss: 0.5542 - val_acc:
0.8016 - val_precision: 0.8829 - val_recall: 0.7273
Epoch 29/100
371/371 [==============================] - 2610s 7s/step - loss: 0.8667 - ac
```

```
c: 0.6789 - precision: 0.7849 - recall: 0.5619 - val_loss: 0.4751 - val_acc:
0.8339 - val_precision: 0.9065 - val_recall: 0.7662
Epoch 30/100
371/371 [==============================] - 2612s 7s/step - loss: 0.8685 - ac
c: 0.6798 - precision: 0.7905 - recall: 0.5624 - val_loss: 0.4397 - val_acc:
0.8417 - val_precision: 0.9006 - val_recall: 0.7784
Epoch 31/100
371/371 [==============================] - 2610s 7s/step - loss: 0.8522 - ac
c: 0.6859 - precision: 0.7894 - recall: 0.5767 - val_loss: 0.4563 - val_acc:
0.8498 - val_precision: 0.9057 - val_recall: 0.7621
Epoch 32/100
371/371 [==============================] - 2614s 7s/step - loss: 0.8349 - ac
c: 0.6882 - precision: 0.7920 - recall: 0.5831 - val_loss: 0.4458 - val_acc:
0.8379 - val_precision: 0.8935 - val_recall: 0.7838
Epoch 33/100
371/371 [==============================] - 2611s 7s/step - loss: 0.8372 - ac
c: 0.6907 - precision: 0.7922 - recall: 0.5840 - val_loss: 0.4142 - val_acc:
0.8505 - val_precision: 0.9063 - val_recall: 0.7967
Epoch 34/100
371/371 [==============================] - 2641s 7s/step - loss: 0.8276 - ac
c: 0.6928 - precision: 0.7963 - recall: 0.5877 - val_loss: 0.4366 - val_acc:
0.8531 - val_precision: 0.9123 - val_recall: 0.7841
Epoch 35/100
371/371 [==============================] - 2642s 7s/step - loss: 0.8117 - ac
c: 0.7006 - precision: 0.7974 - recall: 0.5972 - val_loss: 0.4176 - val_acc:
0.8574 - val_precision: 0.9131 - val_recall: 0.7934
Epoch 36/100
371/371 [==============================] - 2638s 7s/step - loss: 0.8132 - ac
c: 0.6998 - precision: 0.8004 - recall: 0.5973 - val_loss: 0.4200 - val_acc:
0.8505 - val_precision: 0.9058 - val_recall: 0.8008
Epoch 37/100
371/371 [==============================] - 2634s 7s/step - loss: 0.8054 - ac
c: 0.7013 - precision: 0.7975 - recall: 0.6007 - val_loss: 0.4306 - val_acc:
0.8442 - val_precision: 0.8968 - val_recall: 0.7970
Epoch 38/100
371/371 [==============================] - 2632s 7s/step - loss: 0.7746 - ac
c: 0.7150 - precision: 0.8062 - recall: 0.6199 - val_loss: 0.4413 - val_acc:
0.8425 - val_precision: 0.8875 - val_recall: 0.7980
Epoch 39/100
371/371 [==============================] - 2646s 7s/step - loss: 0.7867 - ac
c: 0.7129 - precision: 0.8019 - recall: 0.6127 - val_loss: 0.3391 - val_acc:
0.8874 - val_precision: 0.9253 - val_recall: 0.8311
Epoch 40/100
371/371 [==============================] - 2636s 7s/step - loss: 0.7775 - ac
c: 0.7171 - precision: 0.8004 - recall: 0.6178 - val_loss: 0.3661 - val_acc:
0.8667 - val_precision: 0.9120 - val_recall: 0.8240
Epoch 41/100
371/371 [==============================] - 2631s 7s/step - loss: 0.7803 - ac
c: 0.7148 - precision: 0.8043 - recall: 0.6210 - val_loss: 0.3458 - val_acc:
0.8770 - val_precision: 0.9233 - val_recall: 0.8323
Epoch 42/100
371/371 [==============================] - 2638s 7s/step - loss: 0.7609 - ac
c: 0.7224 - precision: 0.8111 - recall: 0.6280 - val_loss: 0.4035 - val_acc:
0.8560 - val_precision: 0.9102 - val_recall: 0.8058
Epoch 43/100
371/371 [==============================] - 2652s 7s/step - loss: 0.7511 - ac
c: 0.7237 - precision: 0.8120 - recall: 0.6355 - val_loss: 0.3270 - val_acc:
```

```
0.8887 - val_precision: 0.9326 - val_recall: 0.8420
Epoch 44/100
371/371 [==============================] - 2631s 7s/step - loss: 0.7658 - ac
c: 0.7208 - precision: 0.8077 - recall: 0.6293 - val_loss: 0.3292 - val_acc:
0.8942 - val_precision: 0.9392 - val_recall: 0.8379
Epoch 45/100
371/371 [==============================] - 2633s 7s/step - loss: 0.7456 - ac
c: 0.7312 - precision: 0.8155 - recall: 0.6369 - val_loss: 0.3389 - val_acc:
0.8810 - val_precision: 0.9222 - val_recall: 0.8464
Epoch 46/100
371/371 [==============================] - 2637s 7s/step - loss: 0.7290 - ac
c: 0.7284 - precision: 0.8111 - recall: 0.6488 - val_loss: 0.3390 - val_acc:
0.8863 - val_precision: 0.9273 - val_recall: 0.8447
Epoch 47/100
371/371 [==============================] - 2635s 7s/step - loss: 0.7412 - ac
c: 0.7290 - precision: 0.8124 - recall: 0.6445 - val_loss: 0.3693 - val_acc:
0.8735 - val_precision: 0.9222 - val_recall: 0.8223
Epoch 48/100
371/371 [==============================] - 2648s 7s/step - loss: 0.7189 - ac
c: 0.7358 - precision: 0.8207 - recall: 0.6527 - val_loss: 0.3493 - val_acc:
0.8788 - val_precision: 0.9250 - val_recall: 0.8304
Epoch 49/100
371/371 [==============================] - 2633s 7s/step - loss: 0.7093 - ac
c: 0.7416 - precision: 0.8216 - recall: 0.6557 - val_loss: 0.3350 - val_acc:
0.8937 - val_precision: 0.9368 - val_recall: 0.8396
Epoch 50/100
371/371 [==============================] - 2638s 7s/step - loss: 0.7068 - ac
c: 0.7386 - precision: 0.8250 - recall: 0.6589 - val_loss: 0.2758 - val_acc:
0.9086 - val_precision: 0.9417 - val_recall: 0.8727
Epoch 51/100
371/371 [==============================] - 2630s 7s/step - loss: 0.7103 - ac
c: 0.7397 - precision: 0.8187 - recall: 0.6582 - val_loss: 0.2731 - val_acc:
0.9126 - val_precision: 0.9441 - val_recall: 0.8770
Epoch 52/100
371/371 [==============================] - 2633s 7s/step - loss: 0.6992 - ac
c: 0.7445 - precision: 0.8224 - recall: 0.6627 - val_loss: 0.2585 - val_acc:
0.9174 - val_precision: 0.9468 - val_recall: 0.8816
Epoch 53/100
371/371 [==============================] - 2632s 7s/step - loss: 0.7035 - ac
c: 0.7404 - precision: 0.8243 - recall: 0.6608 - val_loss: 0.2857 - val_acc:
0.9043 - val_precision: 0.9434 - val_recall: 0.8697
Epoch 54/100
371/371 [==============================] - 2632s 7s/step - loss: 0.6952 - ac
c: 0.7459 - precision: 0.8215 - recall: 0.6698 - val_loss: 0.2836 - val_acc:
0.9003 - val_precision: 0.9389 - val_recall: 0.8652
Epoch 55/100
371/371 [==============================] - 2629s 7s/step - loss: 0.6911 - ac
c: 0.7455 - precision: 0.8249 - recall: 0.6648 - val_loss: 0.2772 - val_acc:
0.9065 - val_precision: 0.9425 - val_recall: 0.8707
Epoch 56/100
371/371 [==============================] - 2633s 7s/step - loss: 0.6953 - ac
c: 0.7445 - precision: 0.8220 - recall: 0.6669 - val_loss: 0.2864 - val_acc:
0.9007 - val_precision: 0.9328 - val_recall: 0.8669
Epoch 57/100
371/371 [==============================] - 2635s 7s/step - loss: 0.6764 - ac
c: 0.7550 - precision: 0.8302 - recall: 0.6803 - val_loss: 0.2712 - val_acc:
0.9121 - val_precision: 0.9433 - val_recall: 0.8773
```

```
Epoch 58/100
371/371 [==============================] - 2648s 7s/step - loss: 0.6795 - ac
c: 0.7519 - precision: 0.8261 - recall: 0.6787 - val_loss: 0.2713 - val_acc:
0.9180 - val_precision: 0.9464 - val_recall: 0.8778
Epoch 59/100
371/371 [==============================] - 2640s 7s/step - loss: 0.6762 - ac
c: 0.7556 - precision: 0.8272 - recall: 0.6801 - val_loss: 0.3017 - val_acc:
0.8931 - val_precision: 0.9288 - val_recall: 0.8553
Epoch 60/100
371/371 [==============================] - 2647s 7s/step - loss: 0.6748 - ac
c: 0.7533 - precision: 0.8248 - recall: 0.6771 - val_loss: 0.2608 - val_acc:
0.9199 - val_precision: 0.9543 - val_recall: 0.8863
Epoch 61/100
371/371 [==============================] - 2638s 7s/step - loss: 0.6778 - ac
c: 0.7551 - precision: 0.8320 - recall: 0.6839 - val_loss: 0.2575 - val_acc:
0.9121 - val_precision: 0.9434 - val_recall: 0.8833
Epoch 62/100
371/371 [==============================] - 2622s 7s/step - loss: 0.6604 - ac
c: 0.7598 - precision: 0.8289 - recall: 0.6894 - val_loss: 0.2435 - val_acc:
0.9269 - val_precision: 0.9572 - val_recall: 0.8908
Epoch 63/100
371/371 [==============================] - 2609s 7s/step - loss: 0.6568 - ac
c: 0.7654 - precision: 0.8381 - recall: 0.6906 - val_loss: 0.2599 - val_acc:
0.9192 - val_precision: 0.9494 - val_recall: 0.8818
Epoch 64/100
371/371 [==============================] - 2607s 7s/step - loss: 0.6376 - ac
c: 0.7654 - precision: 0.8327 - recall: 0.7001 - val_loss: 0.2315 - val_acc:
0.9242 - val_precision: 0.9493 - val_recall: 0.8937
Epoch 65/100
371/371 [==============================] - 2616s 7s/step - loss: 0.6376 - ac
c: 0.7679 - precision: 0.8348 - recall: 0.6990 - val_loss: 0.2365 - val_acc:
0.9217 - val_precision: 0.9468 - val_recall: 0.8884
Epoch 66/100
371/371 [==============================] - 2615s 7s/step - loss: 0.6440 - ac
c: 0.7654 - precision: 0.8328 - recall: 0.6963 - val_loss: 0.2495 - val_acc:
0.9172 - val_precision: 0.9421 - val_recall: 0.8866
Epoch 67/100
371/371 [==============================] - 2608s 7s/step - loss: 0.6415 - ac
c: 0.7641 - precision: 0.8312 - recall: 0.6962 - val_loss: 0.2171 - val_acc:
0.9349 - val_precision: 0.9575 - val_recall: 0.9089
Epoch 68/100
371/371 [==============================] - 2611s 7s/step - loss: 0.6553 - ac
c: 0.7689 - precision: 0.8345 - recall: 0.7022 - val_loss: 0.2398 - val_acc:
0.9301 - val_precision: 0.9532 - val_recall: 0.8944
Epoch 69/100
371/371 [==============================] - 2609s 7s/step - loss: 0.6256 - ac
c: 0.7738 - precision: 0.8404 - recall: 0.7065 - val_loss: 0.2199 - val_acc:
0.9303 - val_precision: 0.9590 - val_recall: 0.9084
Epoch 70/100
371/371 [==============================] - 2613s 7s/step - loss: 0.6344 - ac
c: 0.7679 - precision: 0.8333 - recall: 0.7063 - val_loss: 0.2054 - val_acc:
0.9381 - val_precision: 0.9611 - val_recall: 0.9111
Epoch 71/100
371/371 [==============================] - 2616s 7s/step - loss: 0.6264 - ac
c: 0.7711 - precision: 0.8395 - recall: 0.7083 - val_loss: 0.2471 - val_acc:
0.9180 - val_precision: 0.9452 - val_recall: 0.8917
Epoch 72/100
```

```
371/371 [==============================] - 2613s 7s/step - loss: 0.6218 - ac
c: 0.7740 - precision: 0.8365 - recall: 0.7095 - val_loss: 0.2209 - val_acc:
0.9323 - val_precision: 0.9517 - val_recall: 0.9058
Epoch 73/100
371/371 [==============================] - 2613s 7s/step - loss: 0.6278 - ac
c: 0.7734 - precision: 0.8394 - recall: 0.7090 - val_loss: 0.2086 - val_acc:
0.9351 - val_precision: 0.9607 - val_recall: 0.9093
Epoch 74/100
371/371 [==============================] - 2627s 7s/step - loss: 0.6037 - ac
c: 0.7844 - precision: 0.8451 - recall: 0.7183 - val_loss: 0.2064 - val_acc:
0.9394 - val_precision: 0.9552 - val_recall: 0.9111
Epoch 75/100
371/371 [==============================] - 2616s 7s/step - loss: 0.6055 - ac
c: 0.7791 - precision: 0.8414 - recall: 0.7219 - val_loss: 0.2242 - val_acc:
0.9270 - val_precision: 0.9521 - val_recall: 0.9023
Epoch 76/100
371/371 [==============================] - 2629s 7s/step - loss: 0.6146 - ac
c: 0.7728 - precision: 0.8366 - recall: 0.7102 - val_loss: 0.2333 - val_acc:
0.9215 - val_precision: 0.9507 - val_recall: 0.8910
Epoch 77/100
371/371 [==============================] - 2615s 7s/step - loss: 0.6020 - ac
c: 0.7780 - precision: 0.8406 - recall: 0.7189 - val_loss: 0.2215 - val_acc:
0.9313 - val_precision: 0.9563 - val_recall: 0.9051
Epoch 78/100
371/371 [==============================] - 2615s 7s/step - loss: 0.5970 - ac
c: 0.7804 - precision: 0.8449 - recall: 0.7211 - val_loss: 0.1925 - val_acc:
0.9432 - val_precision: 0.9644 - val_recall: 0.9169
Epoch 79/100
371/371 [==============================] - 2627s 7s/step - loss: 0.5924 - ac
c: 0.7848 - precision: 0.8461 - recall: 0.7226 - val_loss: 0.2033 - val_acc:
0.9341 - val_precision: 0.9552 - val_recall: 0.9101
Epoch 80/100
371/371 [==============================] - 2622s 7s/step - loss: 0.6062 - ac
c: 0.7813 - precision: 0.8446 - recall: 0.7206 - val_loss: 0.2240 - val_acc:
0.9344 - val_precision: 0.9635 - val_recall: 0.9073
Epoch 81/100
371/371 [==============================] - 2620s 7s/step - loss: 0.6043 - ac
c: 0.7800 - precision: 0.8448 - recall: 0.7194 - val_loss: 0.2786 - val_acc:
0.9038 - val_precision: 0.9339 - val_recall: 0.8801
Epoch 82/100
371/371 [==============================] - 2620s 7s/step - loss: 0.5941 - ac
c: 0.7839 - precision: 0.8443 - recall: 0.7225 - val_loss: 0.1803 - val_acc:
0.9470 - val_precision: 0.9633 - val_recall: 0.9280
Epoch 83/100
371/371 [==============================] - 2621s 7s/step - loss: 0.5883 - ac
c: 0.7856 - precision: 0.8462 - recall: 0.7302 - val_loss: 0.2611 - val_acc:
0.9107 - val_precision: 0.9345 - val_recall: 0.8900
Epoch 84/100
371/371 [==============================] - 2628s 7s/step - loss: 0.6015 - ac
c: 0.7807 - precision: 0.8456 - recall: 0.7203 - val_loss: 0.2201 - val_acc:
0.9381 - val_precision: 0.9606 - val_recall: 0.9066
Epoch 85/100
371/371 [==============================] - 2627s 7s/step - loss: 0.5763 - ac
c: 0.7880 - precision: 0.8472 - recall: 0.7296 - val_loss: 0.2247 - val_acc:
0.9243 - val_precision: 0.9469 - val_recall: 0.9048
Epoch 86/100
371/371 [==============================] - 2623s 7s/step - loss: 0.5794 - ac
```

```
c: 0.7940 - precision: 0.8530 - recall: 0.7358 - val_loss: 0.1950 - val_acc:
0.9368 - val_precision: 0.9542 - val_recall: 0.9154
Epoch 87/100
371/371 [==============================] - 2625s 7s/step - loss: 0.5797 - ac
c: 0.7926 - precision: 0.8528 - recall: 0.7353 - val_loss: 0.1881 - val_acc:
0.9399 - val_precision: 0.9590 - val_recall: 0.9214
Epoch 88/100
371/371 [==============================] - 2644s 7s/step - loss: 0.5755 - ac
c: 0.7903 - precision: 0.8500 - recall: 0.7372 - val_loss: 0.2072 - val_acc:
0.9369 - val_precision: 0.9542 - val_recall: 0.9081
Epoch 89/100
371/371 [==============================] - 2638s 7s/step - loss: 0.5710 - ac
c: 0.7920 - precision: 0.8524 - recall: 0.7362 - val_loss: 0.1737 - val_acc:
0.9455 - val_precision: 0.9588 - val_recall: 0.9278
Epoch 90/100
371/371 [==============================] - 2667s 7s/step - loss: 0.5677 - ac
c: 0.7928 - precision: 0.8493 - recall: 0.7368 - val_loss: 0.1816 - val_acc:
0.9449 - val_precision: 0.9612 - val_recall: 0.9262
Epoch 91/100
371/371 [==============================] - 2666s 7s/step - loss: 0.5572 - ac
c: 0.7990 - precision: 0.8558 - recall: 0.7442 - val_loss: 0.1821 - val_acc:
0.9467 - val_precision: 0.9644 - val_recall: 0.9242
Epoch 92/100
371/371 [==============================] - 2676s 7s/step - loss: 0.5707 - ac
c: 0.7889 - precision: 0.8487 - recall: 0.7364 - val_loss: 0.1869 - val_acc:
0.9414 - val_precision: 0.9615 - val_recall: 0.9205
Epoch 93/100
371/371 [==============================] - 2660s 7s/step - loss: 0.5552 - ac
c: 0.7983 - precision: 0.8545 - recall: 0.7430 - val_loss: 0.1632 - val_acc:
0.9505 - val_precision: 0.9656 - val_recall: 0.9346
Epoch 94/100
371/371 [==============================] - 2624s 7s/step - loss: 0.5761 - ac
c: 0.7913 - precision: 0.8461 - recall: 0.7350 - val_loss: 0.1769 - val_acc:
0.9419 - val_precision: 0.9595 - val_recall: 0.9215
Epoch 95/100
371/371 [==============================] - 2634s 7s/step - loss: 0.5504 - ac
c: 0.8005 - precision: 0.8565 - recall: 0.7460 - val_loss: 0.1774 - val_acc:
0.9472 - val_precision: 0.9633 - val_recall: 0.9278
Epoch 96/100
371/371 [==============================] - 2697s 7s/step - loss: 0.5605 - ac
c: 0.7968 - precision: 0.8521 - recall: 0.7420 - val_loss: 0.1778 - val_acc:
0.9434 - val_precision: 0.9572 - val_recall: 0.9260
Epoch 97/100
371/371 [==============================] - 2685s 7s/step - loss: 0.5485 - ac
c: 0.8016 - precision: 0.8541 - recall: 0.7494 - val_loss: 0.1757 - val_acc:
0.9421 - val_precision: 0.9603 - val_recall: 0.9232
Epoch 98/100
371/371 [==============================] - 2663s 7s/step - loss: 0.5456 - ac
c: 0.8037 - precision: 0.8567 - recall: 0.7529 - val_loss: 0.1859 - val_acc:
0.9459 - val_precision: 0.9626 - val_recall: 0.9229
Epoch 99/100
371/371 [==============================] - 2626s 7s/step - loss: 0.5495 - ac
c: 0.8016 - precision: 0.8554 - recall: 0.7511 - val_loss: 0.1640 - val_acc:
0.9472 - val_precision: 0.9635 - val_recall: 0.9323
Epoch 100/100
371/371 [==============================] - 2618s 7s/step - loss: 0.5523 - ac
```

```
c: 0.7970 - precision: 0.8543 - recall: 0.7460 - val_loss: 0.1601 - val_acc:
0.9495 - val_precision: 0.9631 - val_recall: 0.9346
```

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())
```

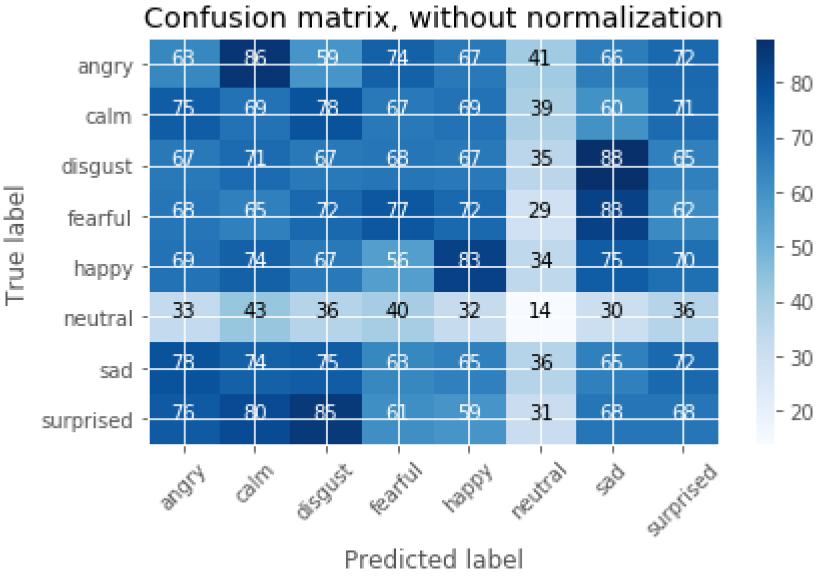
In [7]:
```
import sklearn.metrics as metrics
report = metrics.classification_report(true_classes, predicted_classes, target
_names=class_labels)
print(report)
```

```
             precision    recall  f1-score   support

      angry       0.12      0.12      0.12       528
       calm       0.12      0.13      0.13       528
    disgust       0.12      0.13      0.13       528
    fearful       0.15      0.15      0.15       528
      happy       0.16      0.16      0.16       528
    neutral       0.05      0.05      0.05       264
        sad       0.12      0.12      0.12       528
  surprised       0.13      0.13      0.13       528

avg / total       0.13      0.13      0.13      3960
```

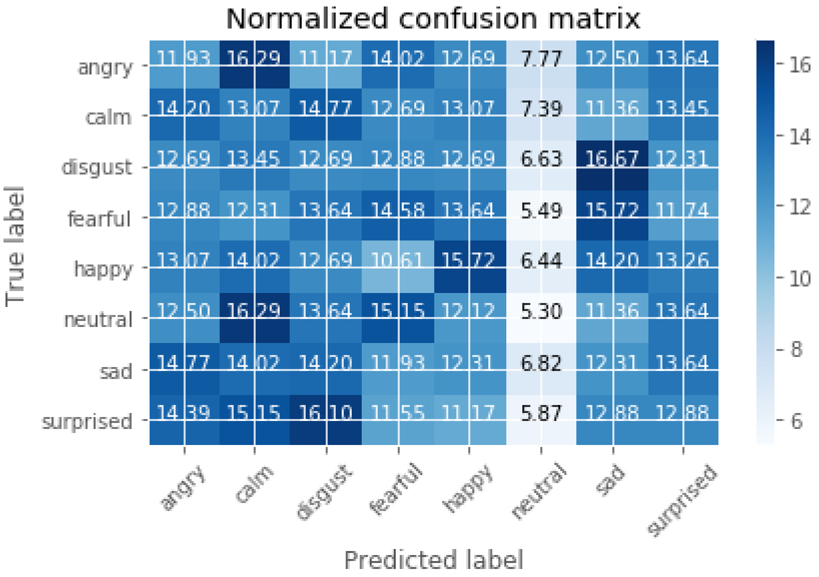
In [10]:

```
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
[[63 86 59 74 67 41 66 72]
 [75 69 78 67 69 39 60 71]
 [67 71 67 68 67 35 88 65]
 [68 65 72 77 72 29 83 62]
 [69 74 67 56 83 34 75 70]
 [33 43 36 40 32 14 30 36]
 [78 74 75 63 65 36 65 72]
 [76 80 85 61 59 31 68 68]]
```

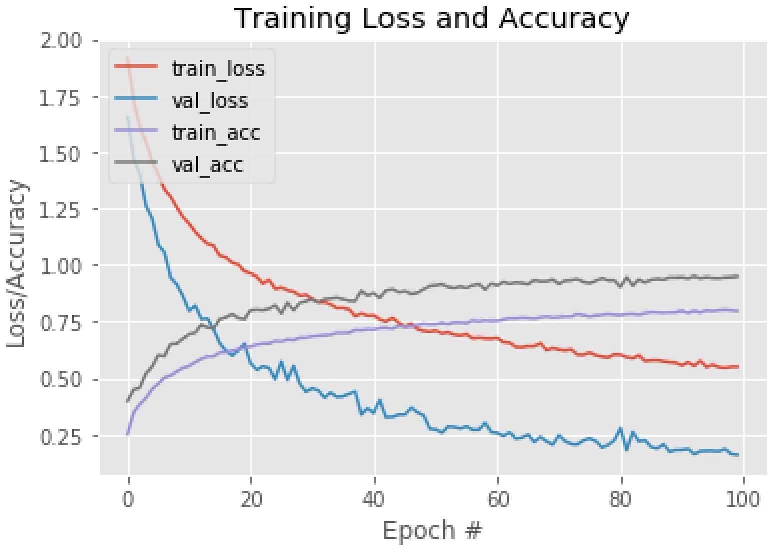


```
Normalized confusion matrix
[[11.9318 16.2879 11.1742 14.0152 12.6894  7.7652 12.5     13.6364]
 [14.2045 13.0682 14.7727 12.6894 13.0682  7.3864 11.3636 13.447 ]
 [12.6894 13.447  12.6894 12.8788 12.6894  6.6288 16.6667 12.3106]
 [12.8788 12.3106 13.6364 14.5833 13.6364  5.4924 15.7197 11.7424]
 [13.0682 14.0152 12.6894 10.6061 15.7197  6.4394 14.2045 13.2576]
 [12.5    16.2879 13.6364 15.1515 12.1212  5.303  11.3636 13.6364]
 [14.7727 14.0152 14.2045 11.9318 12.3106  6.8182 12.3106 13.6364]
 [14.3939 15.1515 16.0985 11.553  11.1742  5.8712 12.8788 12.8788]]
```



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")
```


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