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

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

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

# Installing Keras
# pip install --upgrade keras

# Part 1 - Building the CNN

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

import warnings
warnings.filterwarnings('ignore')

# Initialising the CNN
classifier = Sequential()

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

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

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

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

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

# Step 4 - Full connection
```

```
classifier.add(Dense(output_dim = 128, activation = 'relu'))
classifier.add(Dropout(rate = 0.5))
classifier.add(Dense(output_dim = 7, activation = 'softmax'))

classifier.summary()
```

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

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 64)      1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               2097280
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
=================================================================
Total params: 2,173,831
Trainable params: 2,173,831
Non-trainable params: 0
_________________________________________________________________
```

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

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

from keras.preprocessing.image import ImageDataGenerator

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

test_datagen = ImageDataGenerator(rescale = 1./255)

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

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

```
Found 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 [==============================] - 1657s 13s/step - loss: 1.8174 - ac
c: 0.2836 - precision: 0.3662 - recall: 0.0331 - val_loss: 1.5663 - val_acc:
0.3597 - val_precision: 0.6445 - val_recall: 0.1259
Epoch 2/100
123/123 [==============================] - 1657s 13s/step - loss: 1.6324 - ac
c: 0.3472 - precision: 0.6525 - recall: 0.1001 - val_loss: 1.5113 - val_acc:
0.3832 - val_precision: 0.7453 - val_recall: 0.0802
Epoch 3/100
123/123 [==============================] - 1651s 13s/step - loss: 1.5414 - ac
c: 0.3697 - precision: 0.6421 - recall: 0.1265 - val_loss: 1.3138 - val_acc:
0.4828 - val_precision: 0.7455 - val_recall: 0.1762
Epoch 4/100
123/123 [==============================] - 1646s 13s/step - loss: 1.4518 - ac
c: 0.4129 - precision: 0.6677 - recall: 0.1653 - val_loss: 1.1823 - val_acc:
0.5292 - val_precision: 0.7655 - val_recall: 0.2221
Epoch 5/100
123/123 [==============================] - 1644s 13s/step - loss: 1.4149 - ac
c: 0.4359 - precision: 0.6629 - recall: 0.1839 - val_loss: 1.1614 - val_acc:
0.5431 - val_precision: 0.7314 - val_recall: 0.2384
Epoch 6/100
123/123 [==============================] - 1644s 13s/step - loss: 1.3316 - ac
c: 0.4815 - precision: 0.6892 - recall: 0.2335 - val_loss: 1.1642 - val_acc:
0.5202 - val_precision: 0.7568 - val_recall: 0.2767
Epoch 7/100
123/123 [==============================] - 1645s 13s/step - loss: 1.3059 - ac
c: 0.4838 - precision: 0.6868 - recall: 0.2525 - val_loss: 1.0168 - val_acc:
0.5981 - val_precision: 0.7791 - val_recall: 0.3322
Epoch 8/100
123/123 [==============================] - 1657s 13s/step - loss: 1.2643 - ac
c: 0.4888 - precision: 0.6864 - recall: 0.2637 - val_loss: 0.9667 - val_acc:
0.6090 - val_precision: 0.7877 - val_recall: 0.3733
Epoch 9/100
123/123 [==============================] - 1659s 13s/step - loss: 1.2120 - ac
c: 0.5174 - precision: 0.7143 - recall: 0.3095 - val_loss: 0.9190 - val_acc:
0.6234 - val_precision: 0.7889 - val_recall: 0.4409
Epoch 10/100
123/123 [==============================] - 1640s 13s/step - loss: 1.1817 - ac
c: 0.5296 - precision: 0.7035 - recall: 0.3218 - val_loss: 0.8486 - val_acc:
0.6939 - val_precision: 0.8555 - val_recall: 0.4620
Epoch 11/100
123/123 [==============================] - 1643s 13s/step - loss: 1.1358 - ac
c: 0.5562 - precision: 0.7108 - recall: 0.3576 - val_loss: 0.8050 - val_acc:
0.6816 - val_precision: 0.7880 - val_recall: 0.5196
Epoch 12/100
123/123 [==============================] - 1664s 14s/step - loss: 1.1099 - ac
c: 0.5600 - precision: 0.7275 - recall: 0.3699 - val_loss: 0.7207 - val_acc:
0.7470 - val_precision: 0.8724 - val_recall: 0.5615
Epoch 13/100
123/123 [==============================] - 1670s 14s/step - loss: 1.0874 - ac
c: 0.5785 - precision: 0.7286 - recall: 0.3874 - val_loss: 0.7776 - val_acc:
0.7166 - val_precision: 0.8553 - val_recall: 0.5143
Epoch 14/100
123/123 [==============================] - 1677s 14s/step - loss: 1.0315 - ac
c: 0.5932 - precision: 0.7299 - recall: 0.4245 - val_loss: 0.6644 - val_acc:
0.7545 - val_precision: 0.8608 - val_recall: 0.5944
Epoch 15/100
```

```
123/123 [==============================] - 1647s 13s/step - loss: 1.0168 - ac
c: 0.6009 - precision: 0.7303 - recall: 0.4300 - val_loss: 0.5971 - val_acc:
0.7773 - val_precision: 0.8651 - val_recall: 0.6653
Epoch 16/100
123/123 [==============================] - 1652s 13s/step - loss: 1.0250 - ac
c: 0.6024 - precision: 0.7437 - recall: 0.4306 - val_loss: 0.6037 - val_acc:
0.7793 - val_precision: 0.8526 - val_recall: 0.6461
Epoch 17/100
123/123 [==============================] - 1653s 13s/step - loss: 0.9910 - ac
c: 0.6113 - precision: 0.7437 - recall: 0.4611 - val_loss: 0.5694 - val_acc:
0.8199 - val_precision: 0.8976 - val_recall: 0.6600
Epoch 18/100
123/123 [==============================] - 1653s 13s/step - loss: 0.9671 - ac
c: 0.6247 - precision: 0.7556 - recall: 0.4644 - val_loss: 0.5282 - val_acc:
0.8192 - val_precision: 0.8972 - val_recall: 0.7002
Epoch 19/100
123/123 [==============================] - 1662s 14s/step - loss: 0.9545 - ac
c: 0.6279 - precision: 0.7611 - recall: 0.4832 - val_loss: 0.5160 - val_acc:
0.8493 - val_precision: 0.9141 - val_recall: 0.7002
Epoch 20/100
123/123 [==============================] - 1657s 13s/step - loss: 0.9466 - ac
c: 0.6284 - precision: 0.7508 - recall: 0.4829 - val_loss: 0.5736 - val_acc:
0.7848 - val_precision: 0.8683 - val_recall: 0.6661
Epoch 21/100
123/123 [==============================] - 1653s 13s/step - loss: 0.9205 - ac
c: 0.6417 - precision: 0.7613 - recall: 0.5030 - val_loss: 0.5011 - val_acc:
0.8303 - val_precision: 0.9056 - val_recall: 0.7213
Epoch 22/100
123/123 [==============================] - 1644s 13s/step - loss: 0.8886 - ac
c: 0.6525 - precision: 0.7711 - recall: 0.5123 - val_loss: 0.4679 - val_acc:
0.8372 - val_precision: 0.8870 - val_recall: 0.7369
Epoch 23/100
123/123 [==============================] - 1650s 13s/step - loss: 0.8946 - ac
c: 0.6548 - precision: 0.7703 - recall: 0.5148 - val_loss: 0.4527 - val_acc:
0.8614 - val_precision: 0.9179 - val_recall: 0.7433
Epoch 24/100
123/123 [==============================] - 1656s 13s/step - loss: 0.8587 - ac
c: 0.6659 - precision: 0.7727 - recall: 0.5356 - val_loss: 0.4297 - val_acc:
0.8434 - val_precision: 0.8937 - val_recall: 0.7631
Epoch 25/100
123/123 [==============================] - 1649s 13s/step - loss: 0.8513 - ac
c: 0.6707 - precision: 0.7817 - recall: 0.5506 - val_loss: 0.4106 - val_acc:
0.8735 - val_precision: 0.9232 - val_recall: 0.7826
Epoch 26/100
123/123 [==============================] - 1653s 13s/step - loss: 0.8311 - ac
c: 0.6739 - precision: 0.7687 - recall: 0.5528 - val_loss: 0.4221 - val_acc:
0.8811 - val_precision: 0.9311 - val_recall: 0.7864
Epoch 27/100
123/123 [==============================] - 1656s 13s/step - loss: 0.8644 - ac
c: 0.6680 - precision: 0.7739 - recall: 0.5329 - val_loss: 0.3764 - val_acc:
0.8736 - val_precision: 0.9278 - val_recall: 0.8167
Epoch 28/100
123/123 [==============================] - 1652s 13s/step - loss: 0.7777 - ac
c: 0.6988 - precision: 0.7967 - recall: 0.5866 - val_loss: 0.3325 - val_acc:
0.8886 - val_precision: 0.9197 - val_recall: 0.8243
Epoch 29/100
123/123 [==============================] - 1689s 14s/step - loss: 0.8309 - ac
```

```
c: 0.6838 - precision: 0.7802 - recall: 0.5634 - val_loss: 0.3462 - val_acc:
0.9008 - val_precision: 0.9332 - val_recall: 0.8266
Epoch 30/100
123/123 [==============================] - 1650s 13s/step - loss: 0.7777 - ac
c: 0.6974 - precision: 0.7827 - recall: 0.5821 - val_loss: 0.3338 - val_acc:
0.8750 - val_precision: 0.9035 - val_recall: 0.8309
Epoch 31/100
123/123 [==============================] - 1659s 13s/step - loss: 0.7523 - ac
c: 0.7073 - precision: 0.7885 - recall: 0.6137 - val_loss: 0.3503 - val_acc:
0.8841 - val_precision: 0.9163 - val_recall: 0.8130
Epoch 32/100
123/123 [==============================] - 1654s 13s/step - loss: 0.7643 - ac
c: 0.7033 - precision: 0.7974 - recall: 0.6073 - val_loss: 0.3034 - val_acc:
0.8977 - val_precision: 0.9267 - val_recall: 0.8628
Epoch 33/100
123/123 [==============================] - 1651s 13s/step - loss: 0.7652 - ac
c: 0.6974 - precision: 0.7882 - recall: 0.6029 - val_loss: 0.3425 - val_acc:
0.8758 - val_precision: 0.9145 - val_recall: 0.8259
Epoch 34/100
123/123 [==============================] - 1646s 13s/step - loss: 0.7687 - ac
c: 0.6994 - precision: 0.7917 - recall: 0.6028 - val_loss: 0.3181 - val_acc:
0.9112 - val_precision: 0.9504 - val_recall: 0.8575
Epoch 35/100
123/123 [==============================] - 1647s 13s/step - loss: 0.7411 - ac
c: 0.7101 - precision: 0.7926 - recall: 0.6092 - val_loss: 0.3049 - val_acc:
0.9070 - val_precision: 0.9379 - val_recall: 0.8554
Epoch 36/100
123/123 [==============================] - 1647s 13s/step - loss: 0.7356 - ac
c: 0.7237 - precision: 0.8039 - recall: 0.6315 - val_loss: 0.2912 - val_acc:
0.9009 - val_precision: 0.9291 - val_recall: 0.8637
Epoch 37/100
123/123 [==============================] - 1653s 13s/step - loss: 0.7529 - ac
c: 0.7103 - precision: 0.7954 - recall: 0.6188 - val_loss: 0.2712 - val_acc:
0.9144 - val_precision: 0.9370 - val_recall: 0.8789
Epoch 38/100
123/123 [==============================] - 1651s 13s/step - loss: 0.7031 - ac
c: 0.7157 - precision: 0.8005 - recall: 0.6325 - val_loss: 0.2672 - val_acc:
0.9258 - val_precision: 0.9566 - val_recall: 0.8818
Epoch 39/100
123/123 [==============================] - 1641s 13s/step - loss: 0.6862 - ac
c: 0.7399 - precision: 0.8120 - recall: 0.6585 - val_loss: 0.3404 - val_acc:
0.8880 - val_precision: 0.9320 - val_recall: 0.8418
Epoch 40/100
123/123 [==============================] - 1648s 13s/step - loss: 0.7081 - ac
c: 0.7265 - precision: 0.8015 - recall: 0.6358 - val_loss: 0.2550 - val_acc:
0.9333 - val_precision: 0.9529 - val_recall: 0.8916
Epoch 41/100
123/123 [==============================] - 1655s 13s/step - loss: 0.6474 - ac
c: 0.7530 - precision: 0.8225 - recall: 0.6766 - val_loss: 0.2541 - val_acc:
0.9234 - val_precision: 0.9541 - val_recall: 0.8970
Epoch 42/100
123/123 [==============================] - 1642s 13s/step - loss: 0.7083 - ac
c: 0.7232 - precision: 0.8093 - recall: 0.6342 - val_loss: 0.2618 - val_acc:
0.9234 - val_precision: 0.9558 - val_recall: 0.8855
Epoch 43/100
123/123 [==============================] - 1650s 13s/step - loss: 0.6502 - ac
c: 0.7485 - precision: 0.8138 - recall: 0.6753 - val_loss: 0.2077 - val_acc:
```

```
0.9380 - val_precision: 0.9566 - val_recall: 0.9153
Epoch 44/100
123/123 [==============================] - 1644s 13s/step - loss: 0.6665 - ac
c: 0.7454 - precision: 0.8172 - recall: 0.6651 - val_loss: 0.2407 - val_acc:
0.9271 - val_precision: 0.9448 - val_recall: 0.8954
Epoch 45/100
123/123 [==============================] - 1645s 13s/step - loss: 0.6731 - ac
c: 0.7457 - precision: 0.8117 - recall: 0.6560 - val_loss: 0.2473 - val_acc:
0.9289 - val_precision: 0.9504 - val_recall: 0.8993
Epoch 46/100
123/123 [==============================] - 1647s 13s/step - loss: 0.6660 - ac
c: 0.7475 - precision: 0.8165 - recall: 0.6690 - val_loss: 0.2114 - val_acc:
0.9470 - val_precision: 0.9686 - val_recall: 0.9120
Epoch 47/100
123/123 [==============================] - 1717s 14s/step - loss: 0.6440 - ac
c: 0.7590 - precision: 0.8278 - recall: 0.6762 - val_loss: 0.2263 - val_acc:
0.9373 - val_precision: 0.9555 - val_recall: 0.9070
Epoch 48/100
123/123 [==============================] - 1680s 14s/step - loss: 0.6599 - ac
c: 0.7485 - precision: 0.8185 - recall: 0.6740 - val_loss: 0.2371 - val_acc:
0.9304 - val_precision: 0.9463 - val_recall: 0.9060
Epoch 49/100
123/123 [==============================] - 1659s 13s/step - loss: 0.6401 - ac
c: 0.7535 - precision: 0.8222 - recall: 0.6806 - val_loss: 0.1955 - val_acc:
0.9531 - val_precision: 0.9707 - val_recall: 0.9281
Epoch 50/100
123/123 [==============================] - 1659s 13s/step - loss: 0.6440 - ac
c: 0.7555 - precision: 0.8266 - recall: 0.6795 - val_loss: 0.1925 - val_acc:
0.9506 - val_precision: 0.9658 - val_recall: 0.9226
Epoch 51/100
123/123 [==============================] - 1659s 13s/step - loss: 0.6231 - ac
c: 0.7711 - precision: 0.8403 - recall: 0.7010 - val_loss: 0.2047 - val_acc:
0.9259 - val_precision: 0.9414 - val_recall: 0.9108
Epoch 52/100
123/123 [==============================] - 1663s 14s/step - loss: 0.5925 - ac
c: 0.7733 - precision: 0.8313 - recall: 0.7047 - val_loss: 0.1954 - val_acc:
0.9354 - val_precision: 0.9474 - val_recall: 0.9188
Epoch 53/100
123/123 [==============================] - 1666s 14s/step - loss: 0.5973 - ac
c: 0.7738 - precision: 0.8353 - recall: 0.6968 - val_loss: 0.1848 - val_acc:
0.9471 - val_precision: 0.9607 - val_recall: 0.9258
Epoch 54/100
123/123 [==============================] - 1659s 13s/step - loss: 0.6139 - ac
c: 0.7632 - precision: 0.8266 - recall: 0.6999 - val_loss: 0.2154 - val_acc:
0.9378 - val_precision: 0.9585 - val_recall: 0.9121
Epoch 55/100
123/123 [==============================] - 1656s 13s/step - loss: 0.5917 - ac
c: 0.7803 - precision: 0.8415 - recall: 0.7155 - val_loss: 0.1716 - val_acc:
0.9538 - val_precision: 0.9709 - val_recall: 0.9356
Epoch 56/100
123/123 [==============================] - 1662s 14s/step - loss: 0.5936 - ac
c: 0.7756 - precision: 0.8338 - recall: 0.7105 - val_loss: 0.1545 - val_acc:
0.9612 - val_precision: 0.9703 - val_recall: 0.9460
Epoch 57/100
123/123 [==============================] - 1660s 13s/step - loss: 0.5722 - ac
c: 0.7840 - precision: 0.8403 - recall: 0.7271 - val_loss: 0.1552 - val_acc:
0.9599 - val_precision: 0.9674 - val_recall: 0.9448
```

```
Epoch 58/100
123/123 [==============================] - 1659s 13s/step - loss: 0.5895 - ac
c: 0.7818 - precision: 0.8360 - recall: 0.7167 - val_loss: 0.1606 - val_acc:
0.9568 - val_precision: 0.9665 - val_recall: 0.9408
Epoch 59/100
123/123 [==============================] - 1653s 13s/step - loss: 0.5545 - ac
c: 0.7919 - precision: 0.8458 - recall: 0.7307 - val_loss: 0.1652 - val_acc:
0.9539 - val_precision: 0.9635 - val_recall: 0.9373
Epoch 60/100
123/123 [==============================] - 1666s 14s/step - loss: 0.5548 - ac
c: 0.7886 - precision: 0.8432 - recall: 0.7310 - val_loss: 0.1786 - val_acc:
0.9537 - val_precision: 0.9610 - val_recall: 0.9371
Epoch 61/100
123/123 [==============================] - 1645s 13s/step - loss: 0.5562 - ac
c: 0.7859 - precision: 0.8460 - recall: 0.7305 - val_loss: 0.1614 - val_acc:
0.9517 - val_precision: 0.9621 - val_recall: 0.9389
Epoch 62/100
123/123 [==============================] - 1654s 13s/step - loss: 0.5569 - ac
c: 0.7884 - precision: 0.8437 - recall: 0.7302 - val_loss: 0.1495 - val_acc:
0.9575 - val_precision: 0.9665 - val_recall: 0.9408
Epoch 63/100
123/123 [==============================] - 1651s 13s/step - loss: 0.5372 - ac
c: 0.7923 - precision: 0.8488 - recall: 0.7392 - val_loss: 0.1223 - val_acc:
0.9635 - val_precision: 0.9753 - val_recall: 0.9583
Epoch 64/100
123/123 [==============================] - 1657s 13s/step - loss: 0.5375 - ac
c: 0.8036 - precision: 0.8586 - recall: 0.7406 - val_loss: 0.1238 - val_acc:
0.9674 - val_precision: 0.9760 - val_recall: 0.9560
Epoch 65/100
123/123 [==============================] - 1667s 14s/step - loss: 0.5573 - ac
c: 0.7865 - precision: 0.8423 - recall: 0.7324 - val_loss: 0.1489 - val_acc:
0.9582 - val_precision: 0.9667 - val_recall: 0.9484
Epoch 66/100
123/123 [==============================] - 1660s 13s/step - loss: 0.5348 - ac
c: 0.8023 - precision: 0.8521 - recall: 0.7524 - val_loss: 0.1434 - val_acc:
0.9622 - val_precision: 0.9752 - val_recall: 0.9531
Epoch 67/100
123/123 [==============================] - 1662s 14s/step - loss: 0.5295 - ac
c: 0.8040 - precision: 0.8504 - recall: 0.7559 - val_loss: 0.1401 - val_acc:
0.9598 - val_precision: 0.9713 - val_recall: 0.9506
Epoch 68/100
123/123 [==============================] - 1668s 14s/step - loss: 0.5330 - ac
c: 0.7955 - precision: 0.8468 - recall: 0.7370 - val_loss: 0.1200 - val_acc:
0.9727 - val_precision: 0.9777 - val_recall: 0.9636
Epoch 69/100
123/123 [==============================] - 1656s 13s/step - loss: 0.5066 - ac
c: 0.8111 - precision: 0.8585 - recall: 0.7643 - val_loss: 0.1259 - val_acc:
0.9689 - val_precision: 0.9707 - val_recall: 0.9546
Epoch 70/100
123/123 [==============================] - 1654s 13s/step - loss: 0.5064 - ac
c: 0.8107 - precision: 0.8580 - recall: 0.7513 - val_loss: 0.1124 - val_acc:
0.9658 - val_precision: 0.9767 - val_recall: 0.9590
Epoch 71/100
123/123 [==============================] - 1656s 13s/step - loss: 0.5208 - ac
c: 0.8080 - precision: 0.8598 - recall: 0.7565 - val_loss: 0.1067 - val_acc:
0.9666 - val_precision: 0.9693 - val_recall: 0.9605
Epoch 72/100
```

```
123/123 [==============================] - 1662s 14s/step - loss: 0.4723 - ac
c: 0.8195 - precision: 0.8621 - recall: 0.7738 - val_loss: 0.1132 - val_acc:
0.9651 - val_precision: 0.9769 - val_recall: 0.9635
Epoch 73/100
123/123 [==============================] - 1661s 14s/step - loss: 0.5121 - ac
c: 0.8149 - precision: 0.8624 - recall: 0.7689 - val_loss: 0.1144 - val_acc:
0.9712 - val_precision: 0.9784 - val_recall: 0.9644
Epoch 74/100
123/123 [==============================] - 1654s 13s/step - loss: 0.4913 - ac
c: 0.8178 - precision: 0.8623 - recall: 0.7666 - val_loss: 0.0967 - val_acc:
0.9705 - val_precision: 0.9770 - val_recall: 0.9636
Epoch 75/100
123/123 [==============================] - 1661s 14s/step - loss: 0.5010 - ac
c: 0.8176 - precision: 0.8560 - recall: 0.7721 - val_loss: 0.1092 - val_acc:
0.9713 - val_precision: 0.9816 - val_recall: 0.9652
Epoch 76/100
123/123 [==============================] - 1664s 14s/step - loss: 0.4912 - ac
c: 0.8172 - precision: 0.8626 - recall: 0.7702 - val_loss: 0.1229 - val_acc:
0.9583 - val_precision: 0.9720 - val_recall: 0.9522
Epoch 77/100
123/123 [==============================] - 1655s 13s/step - loss: 0.4896 - ac
c: 0.8133 - precision: 0.8614 - recall: 0.7677 - val_loss: 0.0859 - val_acc:
0.9773 - val_precision: 0.9823 - val_recall: 0.9689
Epoch 78/100
123/123 [==============================] - 1660s 13s/step - loss: 0.4831 - ac
c: 0.8196 - precision: 0.8660 - recall: 0.7738 - val_loss: 0.1171 - val_acc:
0.9628 - val_precision: 0.9736 - val_recall: 0.9508
Epoch 79/100
123/123 [==============================] - 1653s 13s/step - loss: 0.4837 - ac
c: 0.8214 - precision: 0.8653 - recall: 0.7754 - val_loss: 0.0961 - val_acc:
0.9759 - val_precision: 0.9832 - val_recall: 0.9706
Epoch 80/100
123/123 [==============================] - 1665s 14s/step - loss: 0.4900 - ac
c: 0.8194 - precision: 0.8598 - recall: 0.7777 - val_loss: 0.0876 - val_acc:
0.9750 - val_precision: 0.9823 - val_recall: 0.9697
Epoch 81/100
123/123 [==============================] - 1664s 14s/step - loss: 0.4612 - ac
c: 0.8232 - precision: 0.8626 - recall: 0.7795 - val_loss: 0.0971 - val_acc:
0.9697 - val_precision: 0.9762 - val_recall: 0.9659
Epoch 82/100
123/123 [==============================] - 1673s 14s/step - loss: 0.4705 - ac
c: 0.8237 - precision: 0.8651 - recall: 0.7832 - val_loss: 0.0900 - val_acc:
0.9766 - val_precision: 0.9794 - val_recall: 0.9691
Epoch 83/100
123/123 [==============================] - 1670s 14s/step - loss: 0.4791 - ac
c: 0.8252 - precision: 0.8638 - recall: 0.7858 - val_loss: 0.1127 - val_acc:
0.9675 - val_precision: 0.9709 - val_recall: 0.9584
Epoch 84/100
123/123 [==============================] - 1670s 14s/step - loss: 0.4529 - ac
c: 0.8282 - precision: 0.8678 - recall: 0.7895 - val_loss: 0.0970 - val_acc:
0.9713 - val_precision: 0.9808 - val_recall: 0.9645
Epoch 85/100
123/123 [==============================] - 1670s 14s/step - loss: 0.4734 - ac
c: 0.8270 - precision: 0.8680 - recall: 0.7850 - val_loss: 0.0992 - val_acc:
0.9726 - val_precision: 0.9785 - val_recall: 0.9689
Epoch 86/100
123/123 [==============================] - 1674s 14s/step - loss: 0.4407 - ac
```

```
c: 0.8310 - precision: 0.8704 - recall: 0.7876 - val_loss: 0.0898 - val_acc:
0.9743 - val_precision: 0.9794 - val_recall: 0.9682
Epoch 87/100
123/123 [==============================] - 1654s 13s/step - loss: 0.4449 - ac
c: 0.8327 - precision: 0.8731 - recall: 0.7976 - val_loss: 0.0906 - val_acc:
0.9734 - val_precision: 0.9770 - val_recall: 0.9688
Epoch 88/100
123/123 [==============================] - 1662s 14s/step - loss: 0.4685 - ac
c: 0.8234 - precision: 0.8658 - recall: 0.7823 - val_loss: 0.0878 - val_acc:
0.9758 - val_precision: 0.9808 - val_recall: 0.9697
Epoch 89/100
123/123 [==============================] - 1667s 14s/step - loss: 0.4426 - ac
c: 0.8405 - precision: 0.8744 - recall: 0.8070 - val_loss: 0.1036 - val_acc:
0.9675 - val_precision: 0.9753 - val_recall: 0.9569
Epoch 90/100
123/123 [==============================] - 1705s 14s/step - loss: 0.4530 - ac
c: 0.8317 - precision: 0.8774 - recall: 0.7925 - val_loss: 0.1044 - val_acc:
0.9682 - val_precision: 0.9739 - val_recall: 0.9622
Epoch 91/100
123/123 [==============================] - 1684s 14s/step - loss: 0.4310 - ac
c: 0.8398 - precision: 0.8763 - recall: 0.8017 - val_loss: 0.0902 - val_acc:
0.9788 - val_precision: 0.9839 - val_recall: 0.9735
Epoch 92/100
123/123 [==============================] - 1672s 14s/step - loss: 0.4170 - ac
c: 0.8429 - precision: 0.8801 - recall: 0.8100 - val_loss: 0.0861 - val_acc:
0.9749 - val_precision: 0.9815 - val_recall: 0.9704
Epoch 93/100
123/123 [==============================] - 1667s 14s/step - loss: 0.4425 - ac
c: 0.8349 - precision: 0.8750 - recall: 0.7983 - val_loss: 0.0924 - val_acc:
0.9773 - val_precision: 0.9809 - val_recall: 0.9690
Epoch 94/100
123/123 [==============================] - 1663s 14s/step - loss: 0.4625 - ac
c: 0.8290 - precision: 0.8684 - recall: 0.7952 - val_loss: 0.0871 - val_acc:
0.9788 - val_precision: 0.9846 - val_recall: 0.9704
Epoch 95/100
123/123 [==============================] - 1662s 14s/step - loss: 0.4422 - ac
c: 0.8312 - precision: 0.8724 - recall: 0.7879 - val_loss: 0.0978 - val_acc:
0.9766 - val_precision: 0.9868 - val_recall: 0.9659
Epoch 96/100
123/123 [==============================] - 1664s 14s/step - loss: 0.4415 - ac
c: 0.8365 - precision: 0.8689 - recall: 0.7963 - val_loss: 0.0989 - val_acc:
0.9751 - val_precision: 0.9869 - val_recall: 0.9660
Epoch 97/100
123/123 [==============================] - 1668s 14s/step - loss: 0.4182 - ac
c: 0.8438 - precision: 0.8789 - recall: 0.8045 - val_loss: 0.0726 - val_acc:
0.9780 - val_precision: 0.9831 - val_recall: 0.9705
Epoch 98/100
123/123 [==============================] - 1682s 14s/step - loss: 0.4169 - ac
c: 0.8491 - precision: 0.8822 - recall: 0.8173 - val_loss: 0.0913 - val_acc:
0.9796 - val_precision: 0.9832 - val_recall: 0.9706
Epoch 99/100
123/123 [==============================] - 1654s 13s/step - loss: 0.4267 - ac
c: 0.8376 - precision: 0.8742 - recall: 0.8007 - val_loss: 0.0859 - val_acc:
0.9790 - val_precision: 0.9848 - val_recall: 0.9722
Epoch 100/100
123/123 [==============================] - 1664s 14s/step - loss: 0.4039 - ac
```

```
c: 0.8542 - precision: 0.8840 - recall: 0.8204 - val_loss: 0.0891 - val_acc:
0.9710 - val_precision: 0.9746 - val_recall: 0.9673
```

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)
```

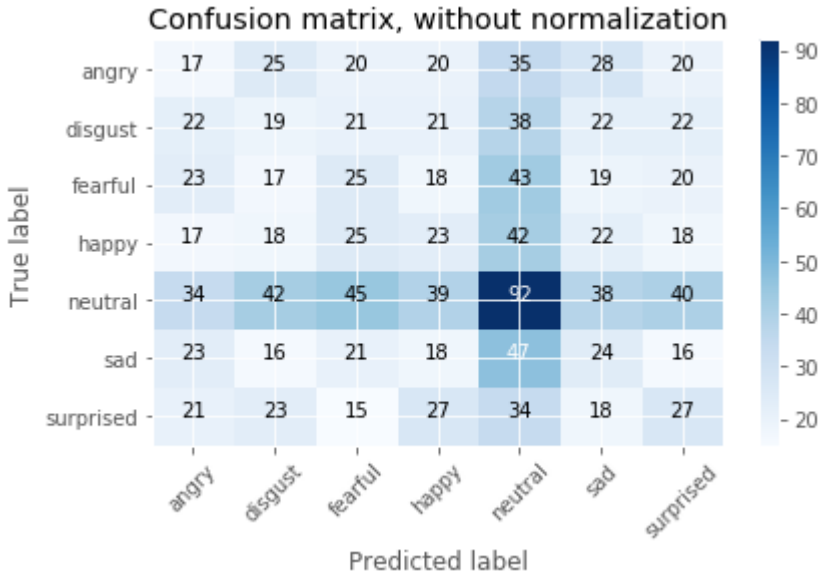
In [6]:
```
true_classes = test_set.classes
class_labels = list(test_set.class_indices.keys())
```

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

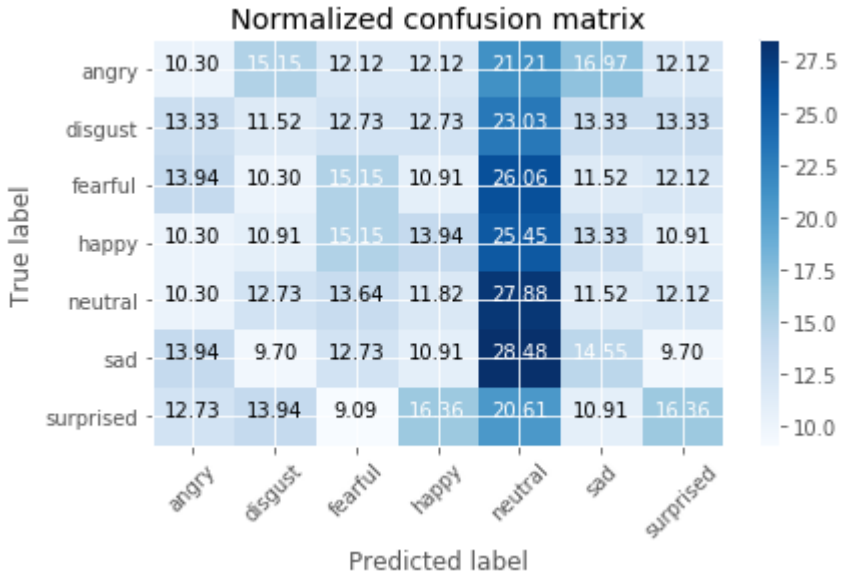
```
             precision    recall  f1-score   support

      angry       0.11      0.10      0.11       165
    disgust       0.12      0.12      0.12       165
    fearful       0.15      0.15      0.15       165
      happy       0.14      0.14      0.14       165
    neutral       0.28      0.28      0.28       330
        sad       0.14      0.15      0.14       165
  surprised       0.17      0.16      0.16       165

avg / total       0.17      0.17      0.17      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.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
[[17 25 20 20 35 28 20]
 [22 19 21 21 38 22 22]
 [23 17 25 18 43 19 20]
 [17 18 25 23 42 22 18]
 [34 42 45 39 92 38 40]
 [23 16 21 18 47 24 16]
 [21 23 15 27 34 18 27]]
```




```
Normalized confusion matrix
[[10.303  15.1515 12.1212 12.1212 21.2121 16.9697 12.1212]
 [13.3333 11.5152 12.7273 12.7273 23.0303 13.3333 13.3333]
 [13.9394 10.303  15.1515 10.9091 26.0606 11.5152 12.1212]
 [10.303  10.9091 15.1515 13.9394 25.4545 13.3333 10.9091]
 [10.303  12.7273 13.6364 11.8182 27.8788 11.5152 12.1212]
 [13.9394  9.697  12.7273 10.9091 28.4848 14.5455  9.697 ]
 [12.7273 13.9394  9.0909 16.3636 20.6061 10.9091 16.3636]]
```



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")
```