(Neural Net Blog)

Convolutional Neural Network Model For Emotion (Facial Expression) Identification

Parshu Rath

January 2021 update

Keywords: Emotion, MS cognitive service, Neural Networks (NN), Convolutional Neural Network (CNN), Graphics processing unit (GPU)

Software /Tools: Python 2.7.15, Jupyter Notebook 5.7.4, IPython 5.8.0, Theano, Tensor flow, Docker, Scikitlearn, Scipy, OpenCV, Numpy, Data Augmentation, Keras ImageDataGenerator

Objectives

The goal of this work was to build a neural network model to accurately predict the emotions of an image. All the available emotions are classified into 8 categories. Neural network models were built based on a given set of images. One of the objectives is to optimize the model for the best (lowest) generalization error. Following is a summary of the work.

Table of Contents

Objectives 1
1. Introduction 2
2. Challenges 2
Processing Time 3
GPU Processing 3
Image Preprocessing 3
Setting up Environments in Docker 3
Dataset Size 3
3. Data and Methods 3
Images 3
Image Processing 4
Data Augmentation 5
Model Building 6
Model Performance 8
4. Results and Performance Comparison 8
5. Lessons Learned 9
6. Appendix 10

1. Introduction

Images, whether single or in 3 channels (color images), are large arrays of data that pose a tremendous challenge in a standard regression model processing. It becomes computationally impracticable to build a supervised model based on linear regression. Neural networks (NN), can be used to model the images and a model can be built for classification of the images. Standard fully connected NN models compute large number of matrix multiplications and can suffer from extremely large cpu time and poor classification accuracy. On the other hand, convolutional models connected to fully connected NN layers offer a solution where image processing could be faster with a better outcome of classification.

Images can be classified for several emotions. In this work eight different classes of emotions were used to classify a set of approximately 13,000 images. A convolutional neural network (CNN) was built and optimized to predict emotion of a given image. Several factors such as, balancing the data set, image size processing, NN hyperparameter adjustments were considered while building this model. The model performs approximately 84% accurately in predicting all of the eight emotions, but to a better likelihood of all emotion prediction with an accuracy of 78% using a balanced dataset .

2. Challenges

There are several challenges that needed to be addressed in successfully building a CNN. Some of them are discussed below.

Processing Time

Neural networks processing are cpu intensive. It takes hours to build a model of certain size. The processing time depends on size of the NN, size of the dataset, and also image size. Each of these needed to be evaluated independently to determine the optimal condition in building an accurate NN model.

GPU Processing

As stated above, NN models need a tremendous amount of computing power. In the absence of parallel computing architecture, the GPU processor (graphics processing unit) can be utilized to perform the processing task. To access the GPU, certain processing environment needed to be set up properly. For example, Theano, or Tensorflow could be set up to use GPU. Therefore, environment for either Theano or Tensorflow needed to be setup to harvest the processing power of GPU.

Image Preprocessing

Images come in different sizes, shapes, and color. They need to be processed to harmonize (same array size) their size and color before inputting into the NN for model building. Color images may need to be changed to grey to speed up model building. Significant amount of time may be needed to transform all the images to an uniform format for faster model building.

Setting up Environments in Docker

One of the most challenging part is setting up of the docker environment. The GPU compilers Theano or Tensorflow are built to run in an unix environment. Therefore a virtual machine needs to be created in an windows machine for these compilers to run under Windows operating system. One of the preferred setup was to have Theano/Tensorflow running in a docker container built in a virtual machine. Since docker is an opensource software, there is continuous modifications being made to it which made it difficult to be setup in a given desktop.

Dataset Size

Another challenge is that the size of the dataset could be large occupying gigabytes of disk space (depending on image size). Docker has a limit of 30 MB file size in its container. Therefore it is essential to have volume mapping set up for docker to store images directly to the computer hard drive rather than loading them all into the memory. This proved to be very difficult as not all stable versions of docker offered volume mapping.

3. Data and Methods

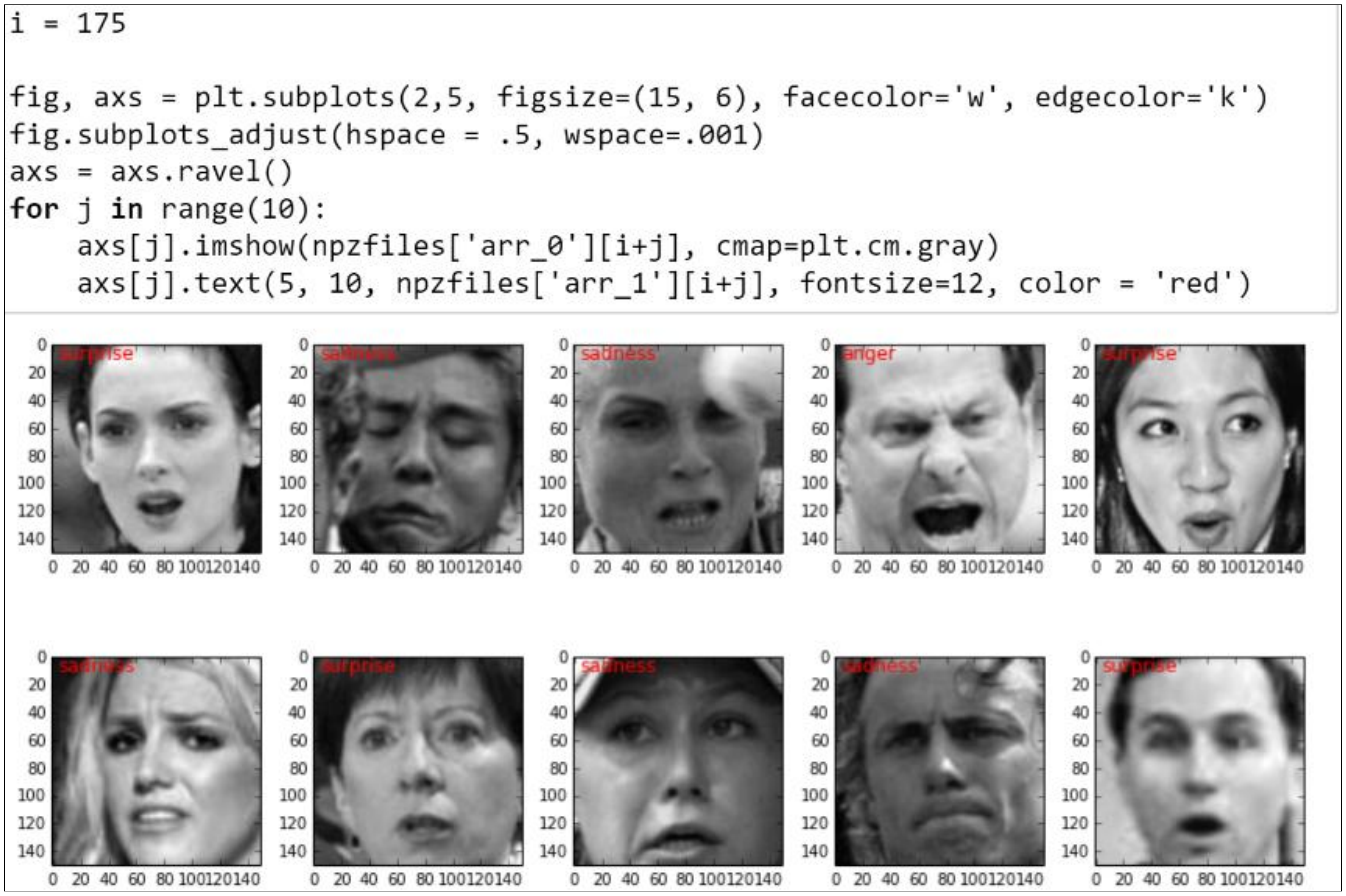
Images

Images for building the models were obtained from http://vis-www.cs.umass.edu/lfw/. The database contains approximately 13,000 images. Microsoft cognitive services API calls (https://www.microsoft.com/cognitive-services/en-us/emotion-api) were made to identify emotion in each of the images. There are total of eight emotions. They are:

1. Anger
2. Contempt
3. Disgust
4. Fear
5. Happiness
6. Neutral
7. Sadness
8. Surprise

Image Processing
Images were of different sizes and dimensions. Some were in color and others in grey scale. All the images were transformed into grey scale and of fixed size (50x50 or 150x150x pixels) prior to building the NN model. If the image was bigger than required size, then it was cropped. Similarly if the image was smaller than the required size, black padding (pixel value = 0) was used. For this part, python's scipy, OpenCV, Numpy packages were used.

Examples of some of the expressions are shown in the figure below.



```
i = 175

fig, axs = plt.subplots(2,5, figsize=(15, 6), facecolor='w', edgecolor='k')
fig.subplots_adjust(hspace = .5, wspace=.001)
axs = axs.ravel()
for j in range(10):
    axs[j].imshow(npzfiles['arr_0'][i+j], cmap=plt.cm.gray)
    axs[j].text(5, 10, npzfiles['arr_1'][i+j], fontsize=12, color = 'red')
```

Figure 1: Examples of the emotions in the training dataset. Emotion labels are shown in red.

Data Augmentation
Keras ImageDataGenerator() was used for generating augmented images inline during the model building step. Generative Adversarial Network for generating adversarial images proved to be too cpu intensive an time consuming to generate meaningful images for augmentation.

The original dataset has 12,758 images and is highly skewed towards two emotions, 'happiness' and 'neutral'. This can be seen from Figure 2 below.

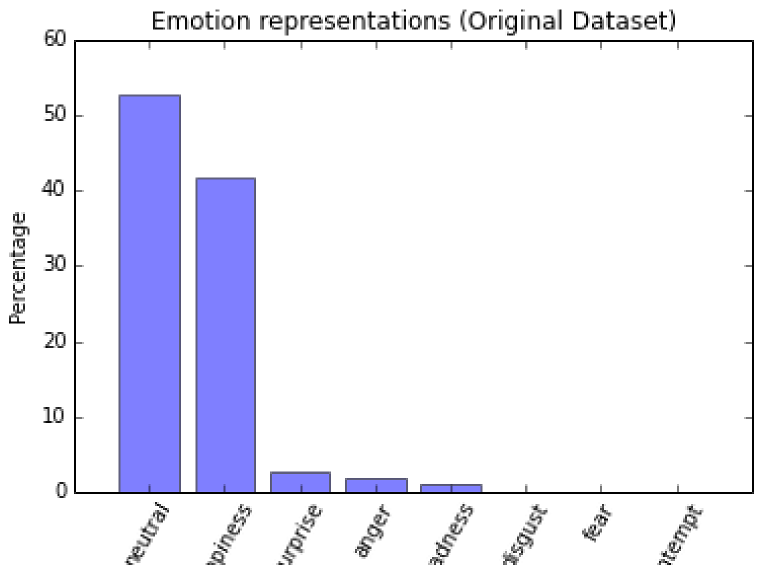


Figure 2: Percentage of all emotions in the original dataset.

Because of the skewed dataset, low frequency emotions may not be detected (prone to high prediction error) even when model accuracy is high. For example even with a very high accuracy of 98%, the likelihood that the emotion contempt (occurs only 0.07% of the time) will be detected is only 2.5% (see the appendix, ***Probability of detecting an emotion:***).

Therefore it is essential that the training data is balanced. To accomplish this, the images in the low frequency occurring emotions were augmented and deed to the dataset. The new dataset (augmented dataset) is quite balanced as seen in Figure 3.

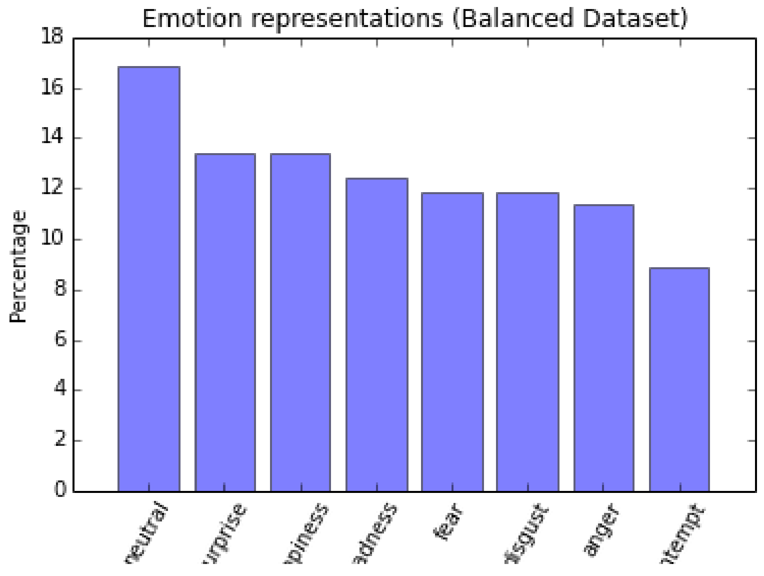


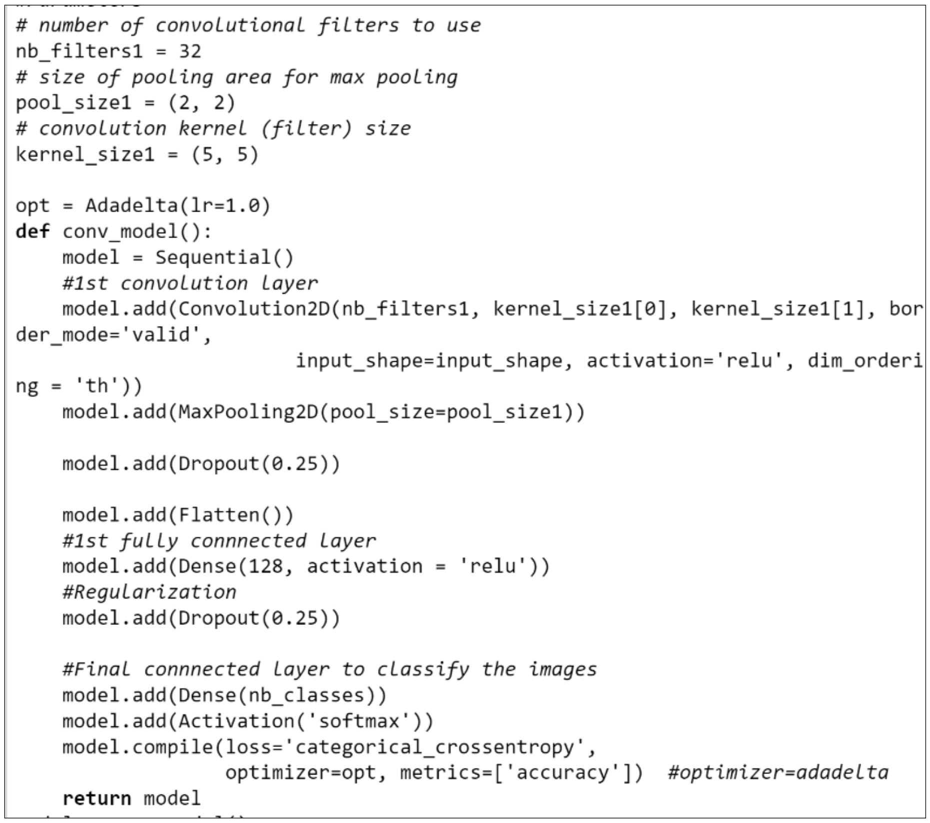
Figure 3: Percentage of all emotions in the augmented dataset.

The augmented balanced dataset has 39,749 images that includes all of the original images for 'happiness' and 'neutral' emotions. It also includes images for other emotions and their augmented images..

Model Building
Model consisting of two convolution layers and two dense layers in addition to the final dense layer was built and the hyperparameters adjusted for optimal performance. The model consisted of the following features:

1. 2D convolution model
2. Two convolution layers
3. Total filters 32 (truncate at borders) in first layer and 16 in the second convolution layer
4. Filter size (kernel) is 5x5, and 3x3
5. Pooling window size 2x2
6. Activaton is ReLU
7. Dropout rate 0.25
8. Two fully connected dense layers with 128 and 32 hidden nodes (ReLu activation)
9. Output layer with softmax activation and 8 dimensional output vector.

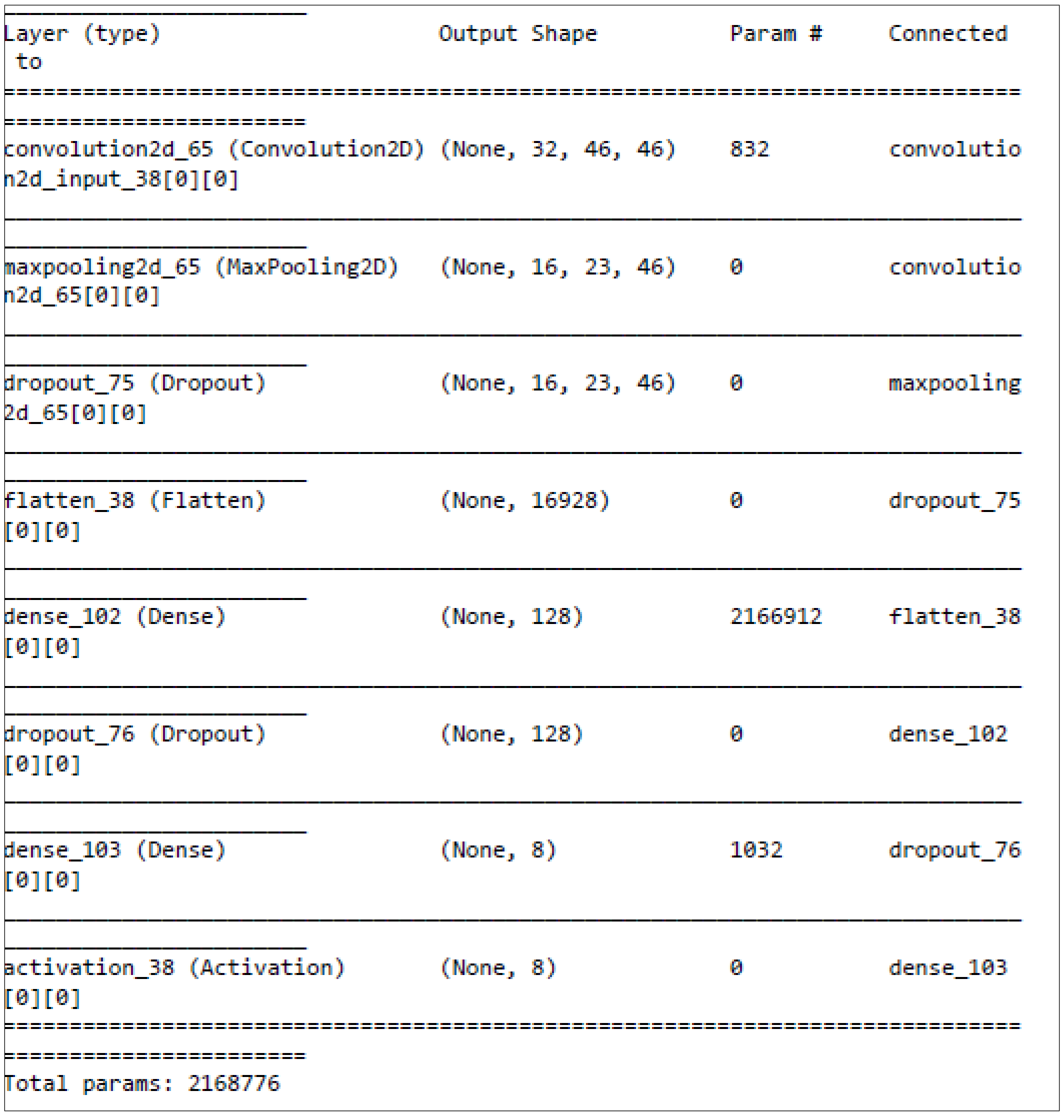
A 2D convolutional model was defined (python script) as shown below, with options to vary the number of filters, kernel size, activation type, dropout rates etc.



```
# number of convolutional filters to use
nb_filters1 = 32
# size of pooling area for max pooling
pool_size1 = (2, 2)
# convolution kernel (filter) size
kernel_size1 = (5, 5)

opt = Adadelta(lr=1.0)
def conv_model():
    model = Sequential()
    #1st convolution layer
    model.add(Convolution2D(nb_filters1, kernel_size1[0], kernel_size1[1], bor
der_mode='valid',
                        input_shape=input_shape, activation='relu', dim_orderi
ng = 'th'))
    model.add(MaxPooling2D(pool_size=pool_size1))

    model.add(Dropout(0.25))

    model.add(Flatten())
    #1st fully connnected layer
    model.add(Dense(128, activation = 'relu'))
    #Regularization
    model.add(Dropout(0.25))

    #Final connnected layer to classify the images
    model.add(Dense(nb_classes))
    model.add(Activation('softmax'))
    model.compile(loss='categorical_crossentropy',
                  optimizer=opt, metrics=['accuracy'])  #optimizer=adadelta
    return model
```

Figure 4: Convolutional Neural Network model python script.

Setup of a 2D convolutional NN model with a single convolution layer is shown in a model summary below.



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected
 to
====================================================================================================
convolution2d_65 (Convolution2D) (None, 32, 46, 46)    832         convolutio
n2d_input_38[0][0]
____________________________________________________________________________________________________
maxpooling2d_65 (MaxPooling2D)   (None, 16, 23, 46)    0           convolutio
n2d_65[0][0]
____________________________________________________________________________________________________
dropout_75 (Dropout)             (None, 16, 23, 46)    0           maxpooling
2d_65[0][0]
____________________________________________________________________________________________________
flatten_38 (Flatten)             (None, 16928)         0           dropout_75
[0][0]
____________________________________________________________________________________________________
dense_102 (Dense)                (None, 128)           2166912     flatten_38
[0][0]
____________________________________________________________________________________________________
dropout_76 (Dropout)             (None, 128)           0           dense_102
[0][0]
____________________________________________________________________________________________________
dense_103 (Dense)                (None, 8)             1032        dropout_76
[0][0]
____________________________________________________________________________________________________
activation_38 (Activation)       (None, 8)             0           dense_103
[0][0]
====================================================================================================
Total params: 2168776
```

Figure 5: Single convolutional layer Neural Network model setup.

Model Performance

Input dataset was split into 80:20 divisions with 80% in the training set and 20% of the images in the test set. Classification and confusion matrix were used to evaluate model accuracy. Various options from Keras, and Scikitlearn (sklearn) packages were used in this step.

4. Results and Performance Comparison

As discussed above, the dataset is highly skewed. Using the original dataset an accuracy of 85% could be achieved. However, this could be misleading as the emotions in low frequency groups were completely misclassified. This can be seen in the confusion matrix below (Table 1).

Table 1: Classification Matrix with original dataset

```
             precision    recall  f1-score   support

      anger       0.18      0.08      0.11        36
   contempt       0.00      0.00      0.00         2
    disgust       0.00      0.00      0.00         3
       fear       0.00      0.00      0.00         3
  happiness       0.86      0.90      0.88      1097
    neutral       0.85      0.89      0.87      1314
    sadness       0.00      0.00      0.00        20
   surprise       0.52      0.14      0.22        77

avg / total       0.83      0.85      0.83      2552
```

When the augmented balanced dataset was used for training, an accuracy of 78% (20x augmentation, 5 epochs, 115 min cpu time) was achieved. Much higher classification rates for the low frequency emotions were achieved as shown in Table 2 below.

Table 2: Classification Matrix with augmented balanced dataset

```
             precision    recall  f1-score   support

      anger       0.60      0.58      0.59       880
   contempt       0.95      0.99      0.97       695
    disgust       0.93      0.99      0.96       953
       fear       0.96      0.97      0.97       972
  happiness       0.77      0.81      0.79      1059
    neutral       0.77      0.76      0.76      1309
    sadness       0.62      0.68      0.65      1006
   surprise       0.70      0.56      0.62      1076

avg / total       0.78      0.78      0.78      7950
```

5. Lessons Learned

There are several lessons to be learned from this project.

Real world data are messy. Need a significant amount of time and patience in cleaning up and processing the data to be model ready.

Setting up software environments and dependencies are not straight forward, particularly so when it is a freeware (no customer support). Need to spend a lot of time on the internet searching for a solution.

Deep learning models can be very useful in modeling multidimensional data but have several loose ends that need to be tied up before it can perform a task satisfactorily.

In classification problem, there must be adequate representation in the dataset for the model to be robust. A skewed dataset may be very good in predicting few classes but can totally fail for other

6. Appendix

Probability of detecting an emotion:

A: $event\ that\ an\ emotion\ is\ correctly\ identified$
B: $event\ that\ the\ emotion\ actually\ occurs$

$$P(A|B) = \frac{P(B|A)P(A)}{\mathrm{P(B|A)P(A) + P(B|A^C)P(A^C)}}$$

Example:

Accuracy = 0.98, emotion **'contempt'** occurs only 0.07% of times.

Probability of correctly identifying 'contempt' in the dataset is given by
$P(\text{contempt will be detected }|\text{emotion is contempt}) =$

$$\frac{0.98*0.007}{0.98*0.007 + 0.02 * 0.9993} = 0.025$$