# Modelling (Final)

**CAPSTONE TERM - II**

*COURCE FACILITATOR: MARCOS BITTENCOURT*


| NAME | Reg. No |
|---|---|
| Mohamad Althaf | 100757321 |
| Karthik Ashok | 100695894 |
| Vinod Vijayaraghavan | 100772404 |
| Saravanan Balasubramanian | 100699139 |

## RATIONAL STATEMENT

This project start from the view that it is possible to identify the expressions of customers on different products by making use of Artificial intelligence and computer vision. Our primary focus is for clothing stores because a clothing store is an ideal place for implementing our system.

We are going to build a face recognition systems for expression extraction using computer vision with large sets of data on facial expressions. This will help the management to find the feedback from customers with ease and thereby make them make better decisions which will help them grow.

## KEY METRICS TO BENCHMARK BUSINESS VALUES

By analyzing the positive and negative attitudes towards a particular category of products using our system,

- Show room managerial could analyze the product selection and refinement using the results of analysis.
- This also helps to understand the ongoing trends of products in the showroom.
- This system will also help the management to understand the demonstration quality of aisle.

## PROBLEM STATEMENT

Our idea is to develop an application to detect real-time facial expressions using computer vision for clothing showrooms. This system can identify the customers' attitude towards the products and the arrangement mechanism in the showroom. This helps the businesses to understand the trends and interests of clothes and accessories plus the arrangement priorities of the showroom.

Given a customer and his expression towards the products in the store, an analysis about the feedback of the customers on the products is created using a dashboard. There are three problems that have to be handled during the development of the system: face recognition, expression extraction and creating an analysis based on the feedback of the customers.

# DATA REQUIREMENTS

Followings are the minimum requirements in dataset to build the model successfully.

1. Grayscale images:

   This will be our independent variable that'll be used to detect the sentiments of faces. These are basically 2-dimentional cropped images of faces. At least 7000 images need to be used in training to get a better model.

2. Facial key features:

   This will be our second set of independent features used in our model. These are basically pixel coordinates of facial key feature points. Each image data points should have these key feature points.

3. Sentiment labels:

   This will be the dependent/predictive variable in our model. In testing dataset, each grayscale image should have evenly distributed category of these labels to predict and analyze the sentiments of people faces. Followings will be the classes of labels

   - Angry
   - Disgust
   - Fear
   - Happy
   - Sad
   - Surprise
   - Neutral

## ASSUMPTIONS, LIMITATIONS AND CONSTRAINTS TO MOVE FORWARD THE SYSTEM

- The user's sentiments may not be related to the product. Maybe the customer is just having a bad day and the model will assume customer is not happy with the product.
- On rush days like black Fridays, the system might not be able to keep up because a lot of facial expressions has to be captured and processed in parallel.
- The system requires the clothes to be arranged in separate sections. I.e. if there are different types of clothes in different sections, system won't be able to predict what sentiment corresponds to which product.
- The system will need a cloud service to run continuously and uninterruptedly. Which is expensive at this point in time.
- Customers may have privacy concerns regarding capturing their sentiments for the purpose of making business more profitable.

## EXISTING METHODOLOGIES:

A plethora of methods for facial recognition have been developed due to its increased importance. These methods are categorized into geometric based and appearance-based methods based on feature extractions.

In these geometric based methods, information such as shape of the face and its components are used for feature extraction. The first main important step in the geometric based method is to initialize a set of facial points as the facial expression evolves over time. Many facial recognition systems in the early stages used elastic bunch graph matching algorithm for the initialization of the face points. The discriminative features are also selected from triangle and line features with multiclass Adaboost algorithm. An effective method for the selection of optimized active face regions was proposed in the later stages.

The Active shape model was employed to extract different facial expression recognitions. Zangeneh and Moradi first used the active appearance model (AAM) to reveal the important facial points and then different geometric features are extracted from those facial points. In the geometric based feature extraction techniques, it was difficult to track and initialize facial

feature points in the real time. If the error occurs during the facial point initialization process, then this error deteriorates the overall feature extraction process.

Recently, the work's that were proposed employed spatiotemporal convolution to extract the temporal dynamic and multilevel appearance feature of facial expressions. A method which reduced the influence of various distortions like occlusion and illumination by extracting the robust features from salient facial patches. Recent works also incorporated spatiotemporal convolution to jointly extract the temporal dynamic and multilevel appearance feature of facial expressions. The performance of the proposed method was tested on facial expression images which contains noise and partial occlusions. In order to normalize the illumination effects, we preprocessed the images by converting the data into arrays which is then fed into the neural networks and they are scaled between -1 to 1. The effectiveness of this proposed model was tested on wild environment images and lab-controlled data.

Apart from the appearance based or geometric based feature extraction, fusion of the two-feature extraction method is also a promising trend. In this model, different from other models, we are a creating a dashboard which shows the results from the expressions of the customers which can be used for business insights.

## PROPOSED METHODOLOGY:

Step 01: Find the right dataset for training

Step 02: Pre-process the dataset after an exploratory data analysis

Step 03: Create a Convolutional Neural network

Step 04: Train the model with the training set

Step 05: Validate the model with testing set

Step 06: Save the model for the final system

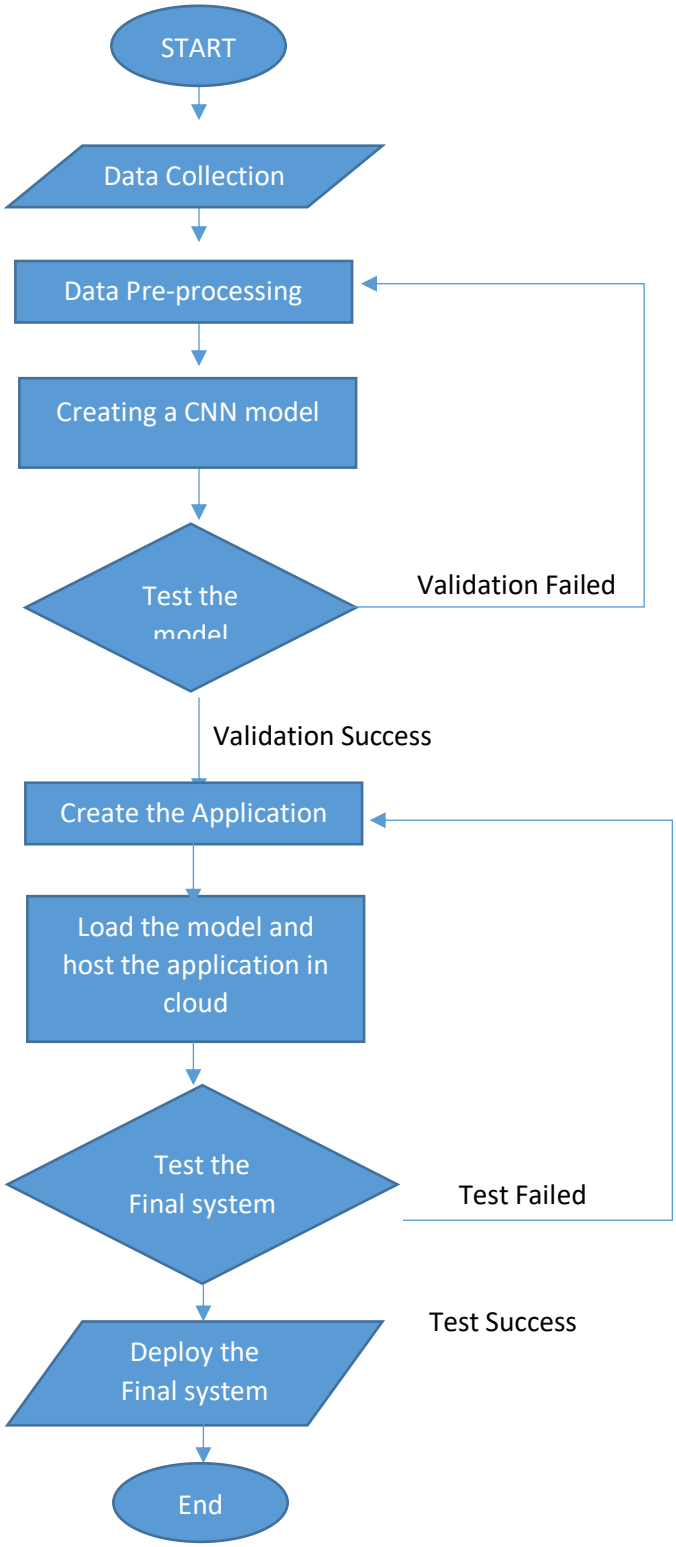
Step 07: Create the web app in cloud with a front end to show the analysis from each camera

Step 08: Use the saved model and test it for real time cameras

Step 09: Test the application for final

Step 10: Deploy the model

## Flow chart:


START
Data Collection
Data Pre-processing
Creating a CNN model
Test the model
Validation Failed
Validation Success
Create the Application
Load the model and host the application in cloud
Test the Final system
Test Failed
Test Success
Deploy the Final system
End

## DEPLOYMENT MODEL


---

Our system will capture real time video frames and send it to the model for analysis. The analysis results will be visible in the dashboard. Shop owner/ Management can make decisions based on the results and make changes to their clothing sections.

## DATA SET

To experiment with the recommendation algorithms, we will need data that consists of a images which are labeled with emotion data. The dataset that we have chosen is the Facial Expression Recognition (FED) Dataset from Kaggle. The dataset consists of 48 x 48 pixel images of faces. The images are gray-scaled. The images in the dataset are so aligned in such a way that the face are more or less centered and captures the same amount of distance with rest of the images.

The dataset is labeled with seven categories:

- Angry
- Disgust
- Fear
- Happy
- Sad
- Surprise
- Neutral

The dataset file comprises of three major files, one FED.csv file containing the train and test data and the train.csv and test.csv which are actually split for model training and evaluation. The train.csv files consists of 35,888 entries of two selection of columns, “emotion” and “pixels”. The emotion columns consists of values ranging from 0 to 6, each representing an emotion according to the label codes above. The pixel columns consists of pixel values written in a single string for each of the image.

Requirements to run the code:

- Python
- Jupyter notebook
- Necessary libraries:
- Pandas
- Numpy
- Matplotlob
- Sklearn
- Tensorflow
- Keras
- Opencv2

Kindly find attached the link to my Exploratory Data Analysis Jupiter notebook.

https://github.com/saravanan21030/RealTimeFacialRecognition

# Execution:

## Modelling Part 1:

## Data Pre-processing Pipeline:

In [36]:
```
# Looking for number of data points

with open("fer2013.csv") as f:
    content = f.readlines()

    lines = np.array(content)

    num_of_instances = lines.size
    print("Number of instances: ",num_of_instances)
```

```
Number of instances:  35888
```

In [37]:
```
# Importing necessary packages

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import cv2

dataset_path = 'fer2013.csv'
image_size=(48,48)
```

In [38]:
```
# Converting the data to arrays that can be fed into neural networks

def load_fer2013():
    data = pd.read_csv(dataset_path)
    pixels = data['pixels'].tolist()
    width, height = 48, 48
    faces = []
    for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        face = np.asarray(face).reshape(width, height)
        face = cv2.resize(face.astype('uint8'),image_size)
        faces.append(face.astype('float32'))
    faces = np.asarray(faces)
    faces = np.expand_dims(faces, -1)
    emotions = pd.get_dummies(data['emotion']).as_matrix()
    return faces, emotions
```

In [39]:
```
faces, emotions = load_fer2013()
```

In [40]:
```
print(faces)
```

```
[[[[ 70.]
   [ 80.]
   [ 82.]
   ...
```

In [42]:
```
# It is a standard way to pre-process images by scaling them between -1 to 1.
# Images is scaled to [0,1] by dividing it by 255. Further, subtraction by 0.5 and multiplication by 2
# changes the range to [-1,1].
# [-1,1] has been found a better range for neural network models in computer vision problems.

def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.5
        x = x * 2.0
    return x


faces = preprocess_input(faces)
```

In [44]:
```
# Storing them using numpy

np.save('fdataX1', faces)
np.save('flabels1', emotions)
```

In [45]:
```
print("Preprocessing Done")
print("Number of Features: "+str(len(faces[0])))
print("Number of Labels: "+ str(len(emotions[0])))

print("faces, emotions stored in fdataX1.npy and flabels1.npy respectively")
```

```
Preprocessing Done
Number of Features: 48
Number of Labels: 7
faces, emotions stored in fdataX1.npy and flabels1.npy respectively
```

---

## Modelling part 2:

# Model Library selection:

Our model will be built using the CNN and computer vision libraries like PIL, scikit-image, PythonMagick, OpenCV-Python etc

- **PIL**

PIL (Python Imaging Library) supports opening, manipulating and saving the images in many file formats. It supports various image manipulations like filtering, enhancing, masking, handling transparency, additions and the like.

- **Scikit-image**

Scikit-image library includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection in images. It's mostly written in python except for the parts written in Cython for the sake of performance. It can be interoperated with SciPy and NumPy

- **PythonMagick**

PythonMagick is the Python binding of the ImageMagick which is a free software. It supports cropping, changing colors, applying various effects, adding text and geometrical figures etc. It supports reading, modifying and creating images in over 200 file formats.

- **OpenCV-Python**

OpenCV-Python is a Python wrapper for the OpenCV C++ implementation. OpenCV-Python makes use of Numpy. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

## Model Selection

Our plan is to build a model using Convolutional neural network (CNN) which we believe will produce the best result. In case if the CNN didn't work well, we will go with the traditional Artificial neural network (ANN).

Below are the reasons why we prefer CNN over ANN

**Disadvantages of ANN on image processing**

- ANN Need substantial number of training samples
- ANN is Slow learning (convergence times)
- Inadequate parameter selection techniques that lead to poor minima
- Network should exhibit invariance to translation, scaling and elastic deformations
- It ignores a key property of images

Information can be merged at later stages to get higher order

**Advantages of CNN on image processing**

- Convolutional neural networks revolutionized the industry, due to the ability to handle large, unstructured data.
- CNNs are extremely successful in areas where large, unstructured data is involved, such as image classification, speech recognition, natural language processing.
- CNNs are more powerful than machine learning algorithms and are also computationally efficient.

## Model Creation:

```
In [ ]: # Importing necessary packages

import tensorflow as tf

import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from keras.layers import Dense, Activation, Dropout, Flatten

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt
```

- Importing necessary packages.

### Model Creation

```
In [7]: #construct CNN structure
model = Sequential()

#1st convolution layer
model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(5,5), strides=(2, 2)))

#2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

#3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

model.add(Flatten())

#fully connected neural networks
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation='softmax'))

WARNING:tensorflow:From /Users/Saravanan/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:40
70: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /Users/Saravanan/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:40
74: The name tf.nn.avg_pool is deprecated. Please use tf.nn.avg_pool2d instead.
```

- Model creation and it's summary.

In [8]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 44, 44, 64)        1664
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 20, 20, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 18, 18, 64)        36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
average_pooling2d_1 (Average (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 128)         73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 128)         147584
_________________________________________________________________
average_pooling2d_2 (Average (None, 1, 1, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              132096
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 7175
=================================================================
Total params: 1,485,831
Trainable params: 1,485,831
Non-trainable params: 0
```

## Model Traning

In [9]:
```
#batch process

gen = ImageDataGenerator()
train_generator = gen.flow(x_train, y_train, batch_size=batch_size)
```

In [10]:
```
model.compile(loss='categorical_crossentropy'
    , optimizer=keras.optimizers.Adam()
    , metrics=['accuracy'])
```

In [11]:
```
fit = True

if fit == True:
    #model.fit_generator(x_train, y_train, epochs=epochs) #train for all trainset
    model.fit_generator(train_generator, steps_per_epoch=batch_size, epochs=5) #train for randomly selected one
else:
    model.load_weights('facial_expression_model_weights.h5') #load weights
```

```
WARNING:tensorflow:From /Users/Saravanan/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:42
2: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/5
256/256 [==============================] - 143s 557ms/step - loss: 1.7835 - accuracy: 0.2654
Epoch 2/5
256/256 [==============================] - 151s 590ms/step - loss: 1.5500 - accuracy: 0.3899
Epoch 3/5
256/256 [==============================] - 150s 587ms/step - loss: 1.3967 - accuracy: 0.4621
Epoch 4/5
256/256 [==============================] - 148s 580ms/step - loss: 1.2813 - accuracy: 0.5095
Epoch 5/5
256/256 [==============================] - 155s 606ms/step - loss: 1.1892 - accuracy: 0.5485
```

## Saving the model and testing

```
In [12]: model.save('facial_expression_model_weightsepo5.h5')
```

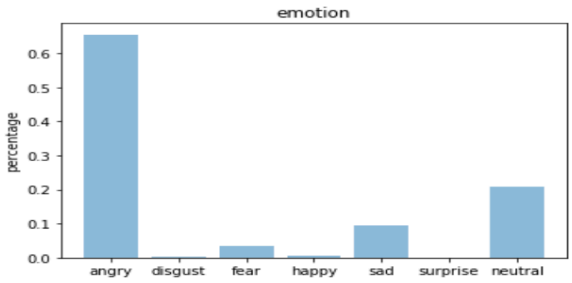
```
In [13]: #overall evaluation
         score = model.evaluate(x_test, y_test)
         print('Test loss:', score[0])
         print('Test accuracy:', 100*score[1])
```

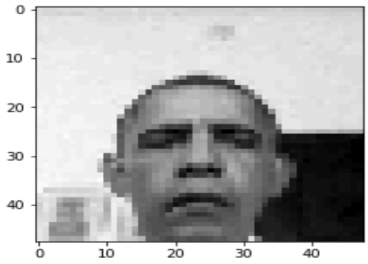
```
3589/3589 [==============================] - 2s 431us/step
Test loss: 1.2457964211727257
Test accuracy: 52.46586799621582
```

---

## Testing the model on the custom image:

```
In [27]: #make prediction for custom image out of test set

         img = image.load_img("img3.jpeg", grayscale=True, target_size=(48, 48))

         x = image.img_to_array(img)
         x = np.expand_dims(x, axis = 0)

         x /= 255

         custom = model.predict(x)
         emotion_analysis(custom[0])

         x = np.array(x, 'float32')
         x = x.reshape([48, 48]);

         plt.gray()
         plt.imshow(x)
         plt.show()
```




---