Emotion Detection

By: Parth Doshi, Samiksha Emmaneni, Avni Kanodia, Sid Shastry, and Jenny Wang

- What is this project? Emotion Detection
  - In this project we seek to be able to discern the emotions of a person provided and image of that person.

This requires us to create an AI that cannot only distinguish expressions and give them a name, but create an AI that can differentiate a face from a background.

But…

What is this useful for?

According to Erin Digitale, the pediatrics science writer in the Office of Communications at Stanford University,

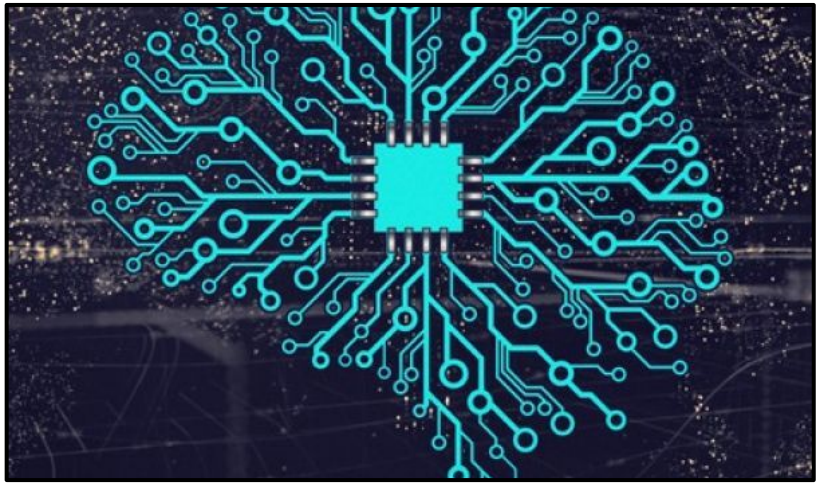
“children with autism were able to improve their social skills” (Digitale) thanks to an application and Google Glass.

“Alex took part in a pilot study in which a smartphone app paired with Google Glass was shown to help children with autism understand emotions conveyed in facial expressions.
Steve Fisch” (Digitale)

Google Glass essentially uses a Face Recognition/Emotion Detection AI. While the wearer is interacting with someone else it “identifies and names their emotions through the Google Glass speaker or screen” (Digitale).

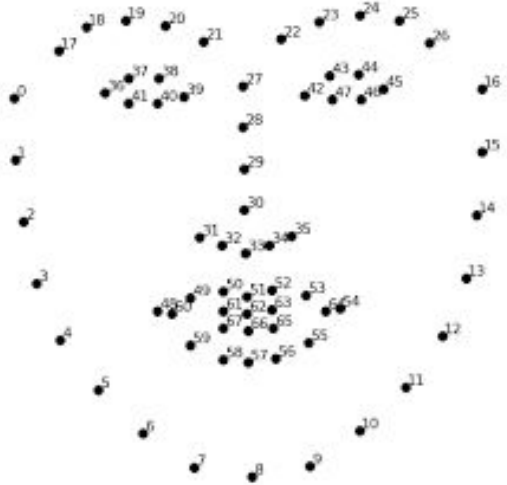
- What is the goal?
  - Classify the facial expressions into one of the following core emotions: Angry, Happy, Sad, Surprised, Neutral

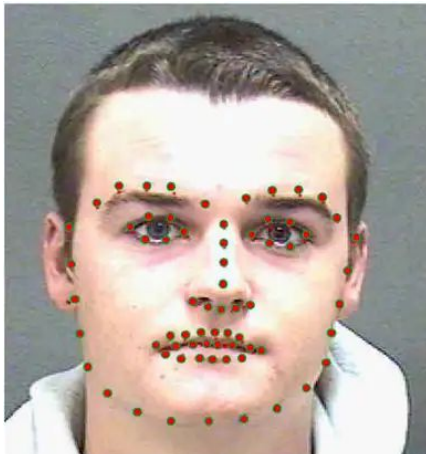
- How do we go about this?
  - We seek to find focal points on a person’s face. This will then be transferred to a convolutional neural network.



Facial Landmarks

Examples of facial landmarks:





What are facial landmarks?

What's the purpose?

Three main components of Emotion Detection:

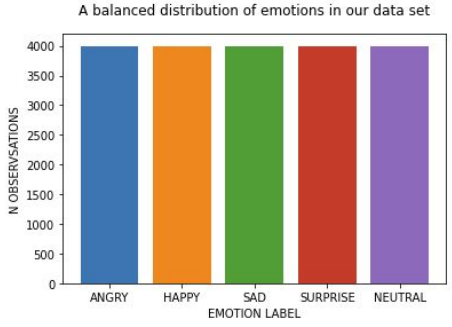
- Image processing
- Feature Extraction
- Feature Classification

Examples of data being collected:

```
[ ] #Integer to Label Mapping
    label_map = {0:"ANGRY",1:"HAPPY",2:"SAD",3:"SURPRISE",4:"NEUTRAL"}

    #Load the datala
    df = pd.read_csv("./ferdata.csv")
    df.head()
```

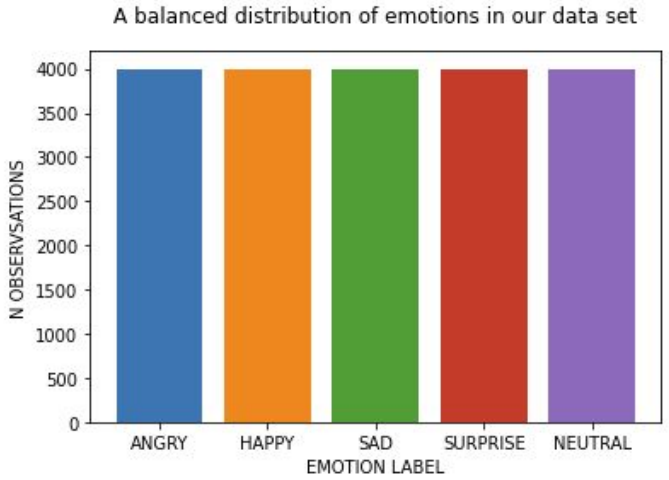
	emotion	pixels	Usage
0	0	215 216 215 215 215 216 216 216 214 178 81 30 ...	Training
1	4	244 244 244 244 243 244 242 190 132 93 81 73 7...	Training
2	4	255 255 255 255 255 255 255 255 255 255 255 25...	Training
3	4	38 56 60 52 58 65 53 44 35 48 59 60 36 30 21 3...	Training
4	4	77 40 27 21 22 25 20 31 27 17 27 42 47 55 51 4...	Training

Balanced and unbalanced data?



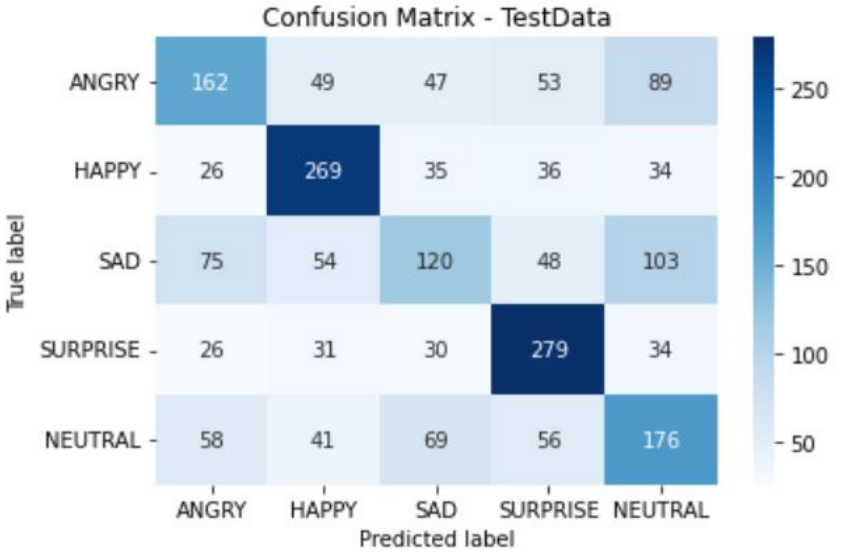
Bias

Uneven data leads to bias, have to ensure that all data is given in same amounts



If not, final data will be skewed towards the column with highest frequency

Data can also be skewed to give results with higher accuracy.



For example, more sad faces can be issued as it is the most confused on sad faces.

Understanding Euclidean Distance

Recognizing facial features

Emotions are able to be detected based on distances between facial features

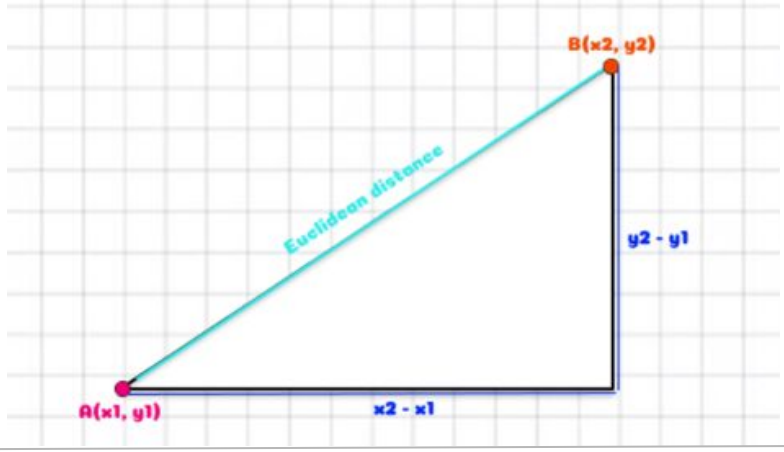
Done using euclidean distances.

Euclidean distance found using formula:

$$a = (X_1 - X_2),\ b = (Y_1 - Y_2)$$

$$a^2 + b^2 = c^2$$

$$\sqrt{C^2} = \text{Euclidean distance}$$

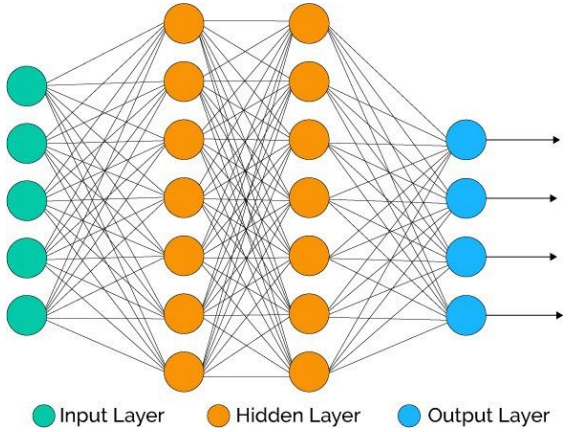


Euclidean distance

Using 68 face points, 2238 different connections possible without including any repeats (68+67+66+65…+1)

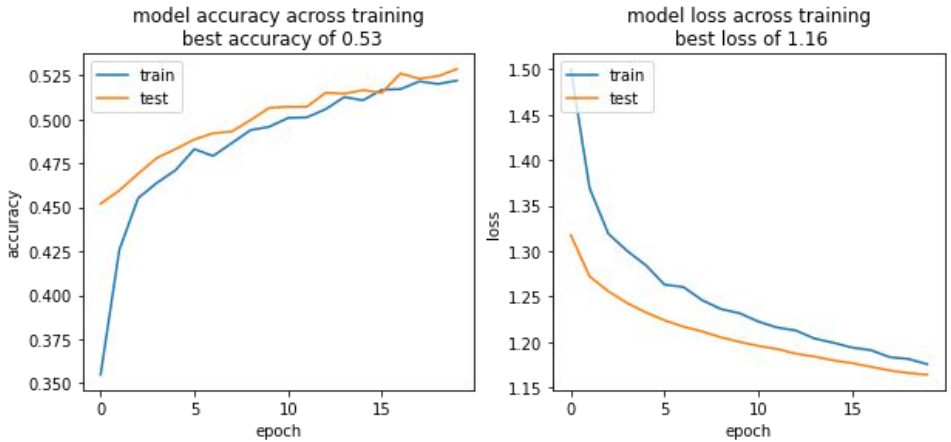
Multilayer Perceptrons

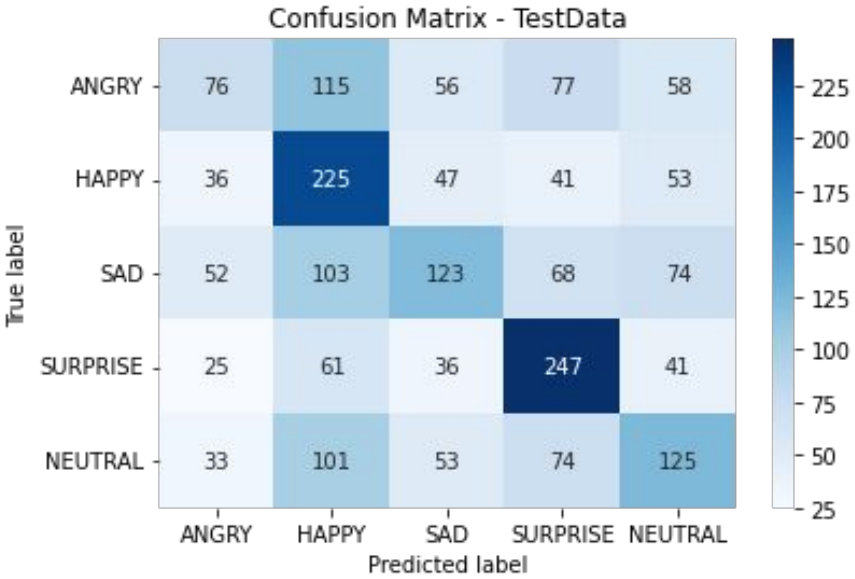
1. The number of neurons
2. The activation of the neurons
3. The losses and metrics




```
# First, we initialize our model
tmp_model = Sequential()

# then we add a "Dense" (i.e. fully connected) layer
tmp_model.add(Dense(7, input_shape=(5,), activation = 'relu')) # for the first layer we specify the input dimensions

# then we have to add another layer
tmp_model.add(Dense(7, activation = 'relu'))

# we end by defining the output layer, which has the number of dimensions of the predictions we're making
tmp_model.add(Dense(4, activation = 'softmax'))

# we finalize the model by "compiling" it and defining some other hyperparameters
tmp_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Detecting Emotions

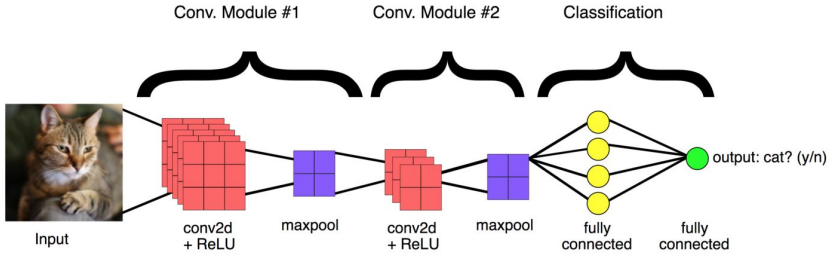


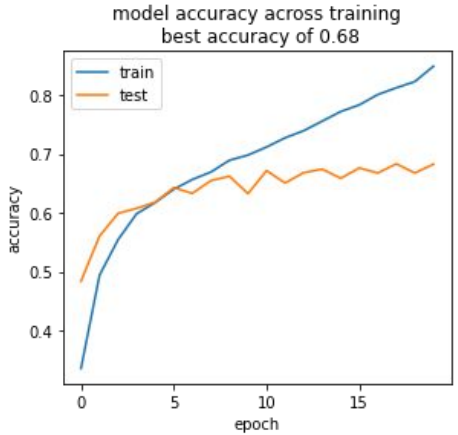


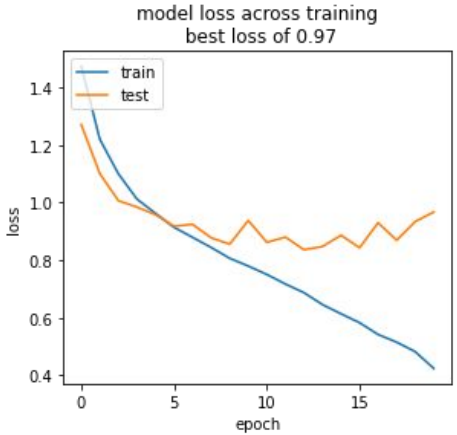
Here, we are training our model with distances between facial features to make predictions

Convolutional Neural Networks (CNN)

CNN models use “kernels” to make an image’s output more and more detailed

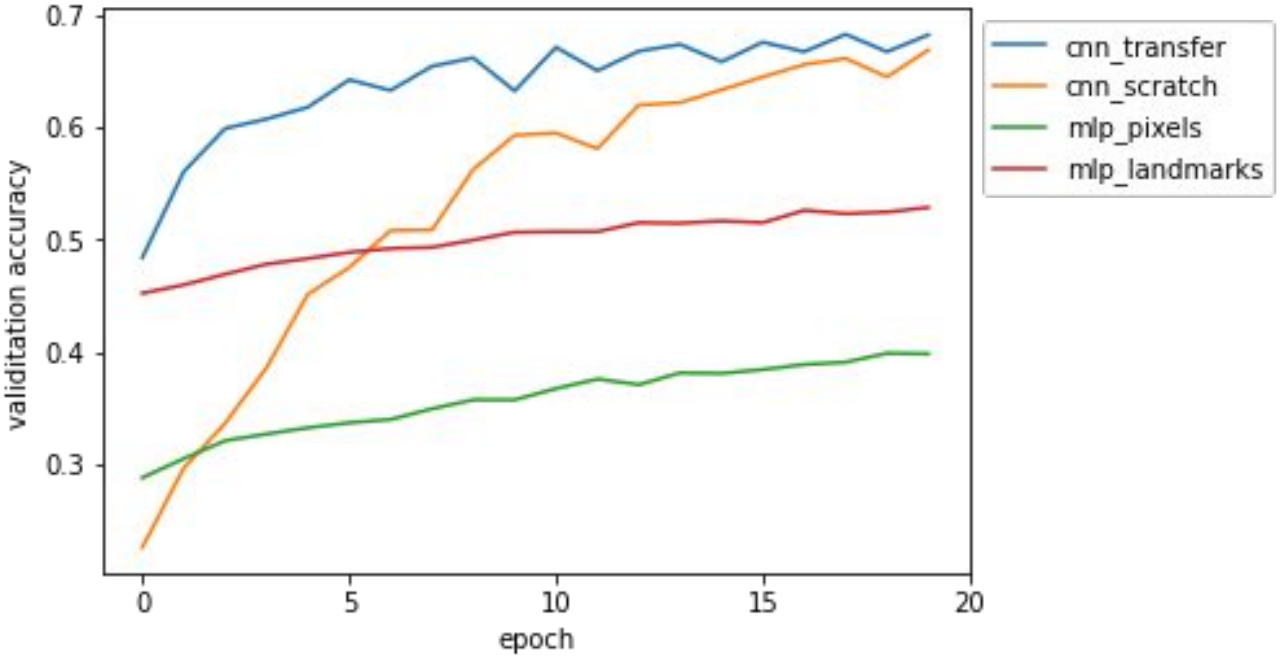







```
cnn_model = Sequential()

cnn_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(width, height, 1), kernel_regularizer=l2(0.01)))
cnn_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
cnn_model.add(Dropout(0.5))

cnn_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
cnn_model.add(BatchNormalization())
cnn_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
```

CNN vs. MLP


0.7
0.6
0.5
0.4
0.3
validitation accuracy
0
5
10
15
20
epoch
cnn_transfer
cnn_scratch
mlp_pixels
mlp_landmarks

Thanks for listening