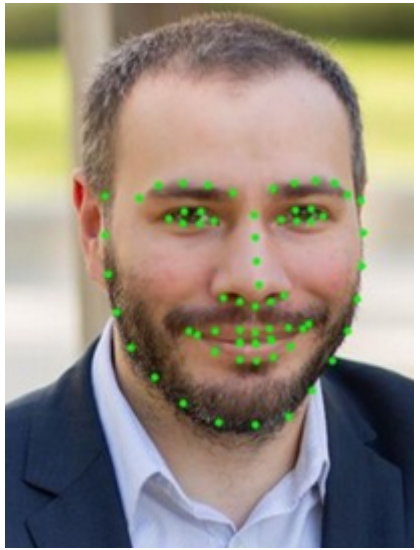
Sample Detecting Facial Features

https://towardsdatascience.com/detecting-face-features-with-python-30385aee4a8e



Point map
(67 points)

Jaw Points = 0–16

Right Brow Points = 17–21

Left Brow Points = 22–26

Nose Points = 27–35

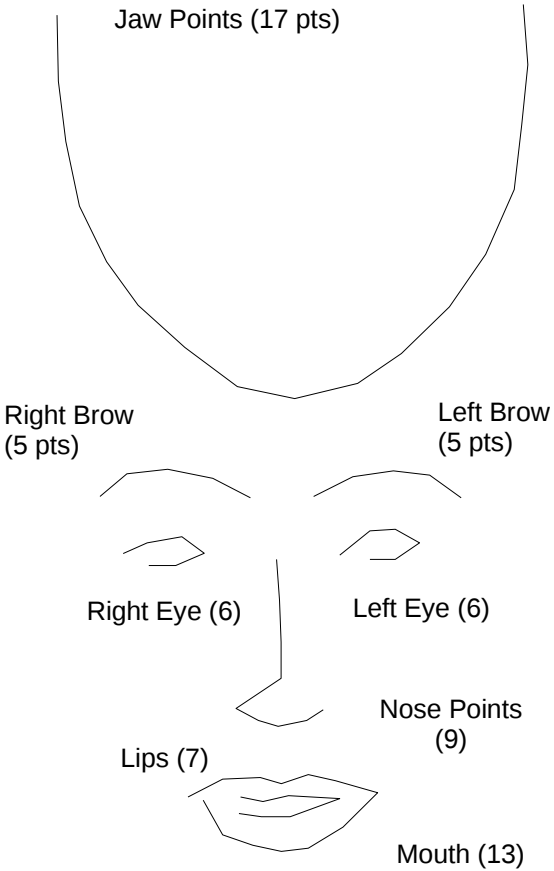
Right Eye Points = 36–41

Left Eye Points = 42–47

Mouth Points = 48–60

Lips Points = 61–67

Feature Vectors



$pip -V (to check your pip version)

Note this model is simplified version with no upper face

Harry Li, Ph.D.

Using http://dlib.net/ For Feature Extraction

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge.

1. Machine learning
2. Numerical algorithms
3. Graphical model inference algorithms
4. Image processing
5. Threading
6. Networking
7. GUI
8. Testing (Unit testing framework)
9. XML Parser etc.

Sample code:

```
import cv2
import numpy as np
import dlib   #for facial feature extraction
```

```
# Load the predictor
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

From dlib, additional features are possible

Pycharm IDE for Python

PyCharm is probably the only Python dedicated IDE that supports the vast expanse of features Python has. Sublime, Atom and Sypder do exist, but they only exist! The integrated development environment experience that PyCharm provides is way better than the others.

To install pycharm:

https://medium.com/@singh.shreya8/how-to-install-pycharm-in-ubuntu-16-04-ubuntu-14-04-ubuntu-18-04-linux-easiest-way-5ae19d052693

To start pycharm:
$sh pycharm.sh

https://www.youtube.com/watch?v=BPC-bGdBSM8&list=PLQ176FUIyIUZ1mwB-uImQE-gmkwzjNLjP

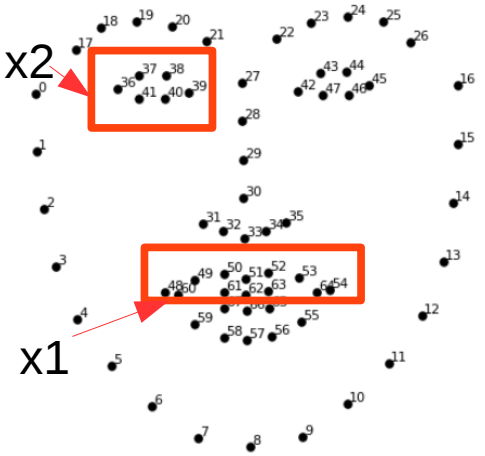
Pycharm hello the world

https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html#summary

use Cmake to build and install OpenCV and Extra Modules from source and configure your Pycharm IDE

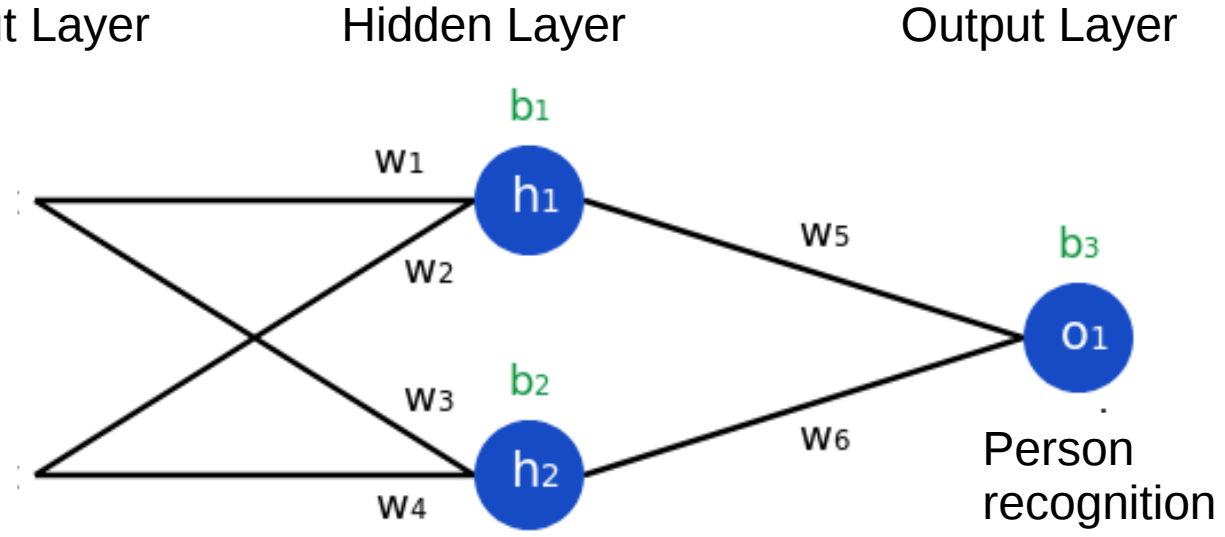
https://towardsdatascience.com/how-to-install-opencv-and-extra-modules-from-source-using-cmake-and-then-set-it-up-in-your-pycharm-7e6ae25dbac5

Simple Feed Forward Neural Networks



x1 (lip)

x2 (right eye brow)



To do: (1) add h3 hidden layer, (2) add o2 at output layer, modify the code

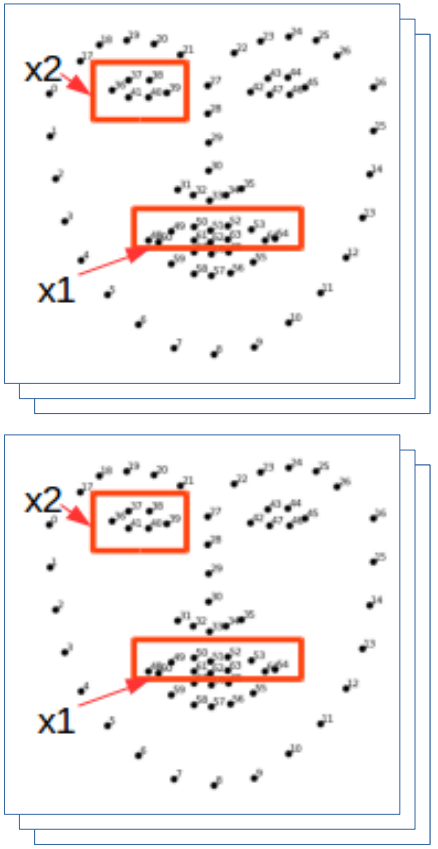
```
def feedforward(self, x):
  # x is a numpy array with 2 elements.
  h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
  h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
  o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
  return o1
```

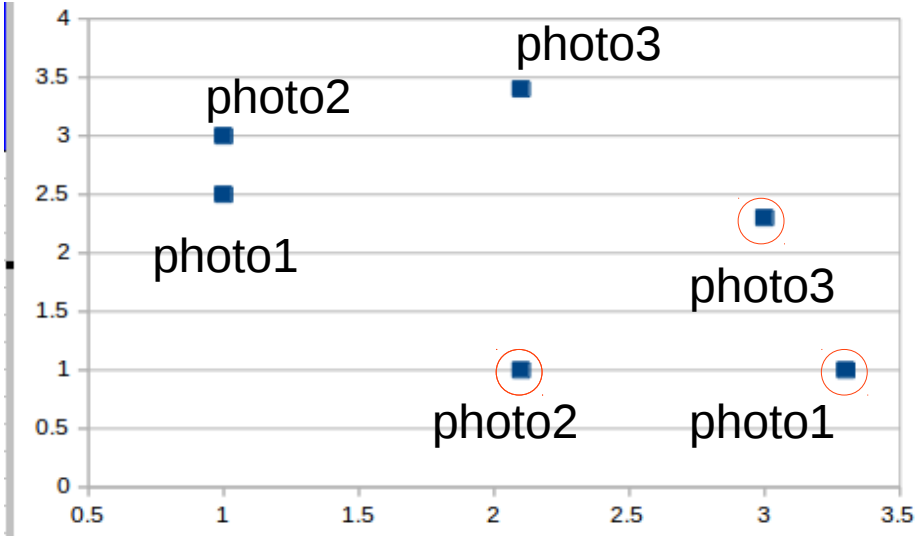
Prepare the Data Set

For 2 persons

```
#---------------------------------
# Define dataset and all_y_trues
#---------------------------------
data = np.array([
  [1, 2.5],     # person A
  [1, 3],       # person A
  [2.1, 3.4],   # Person A
  [2.1, 1], # person B
  [3.3, 1], # person B
  [3, 2.3], # person B
])
all_y_trues = np.array([
  1, # person A
  1, # person A
  1, # person A
  0, # person B
  0, # person B
  0, # person B
])
```

For 2 persons





Prepare the Output Layer

For 2 persons

```
#---------------------------------
# Define dataset and all_y_trues
#---------------------------------
data = np.array([
  [1, 2.5],     # person A
  [1, 3],       # person A
  [2.1, 3.4],   # Person A
  [2.1, 1], # person B
  [3.3, 1], # person B
  [3, 2.3], # person B
])
all_y_trues = np.array([
  1, # person A
  1, # person A
  1, # person A
  0, # person B
  0, # person B
  0, # person B
])
```

For 4 persons

(1) Output layer with 2 neurons,

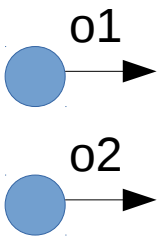
o1 o2

0 0

0 1

1 0

1 1



(2) output layer with 4 neurons

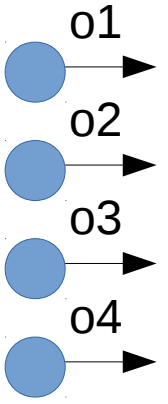
o1 o2 o3 o4

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1 t



Output Layer from ImageNet

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-18.html

ImageNet Training in Minutes

Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel and Kurt Keutzer

EECS Department

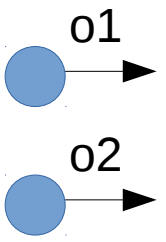
University of California, Berkeley

Technical Report No. UCB/EECS-2020-18

January 25, 2020

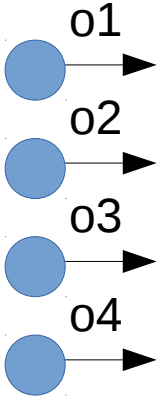
(1) Output layer with 2 neurons,

o1	o2
0	0
0	1
1	0
1	1



(2) output layer with 4 neurons

o1	o2	o3	o4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1 t



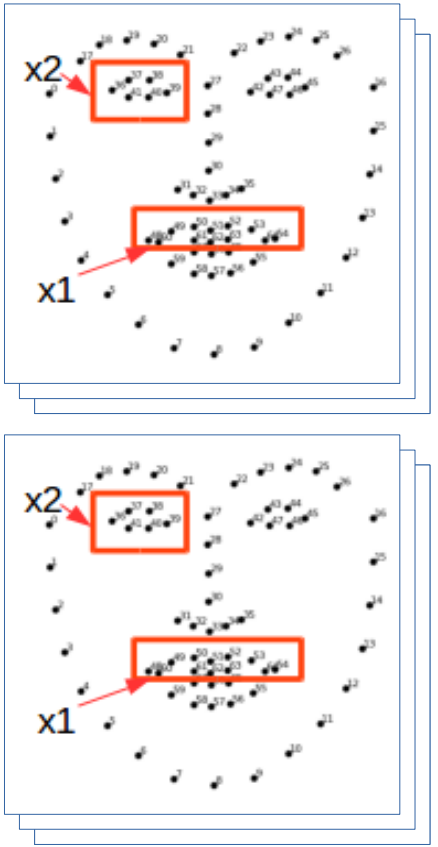
For 4 persons

Prepare the Data Set

For 2 persons

```
#---------------------------------
# Define dataset and all_y_trues
#---------------------------------
data = np.array([
  [1, 2.5],     # person A
  [1, 3],       # person A
  [2.1, 3.4],   # Person A
  [2.1, 1], # person B
  [3.3, 1], # person B
  [3, 2.3], # person B
])
all_y_trues = np.array([
  1, # person A
  1, # person A
  1, # person A
  0, # person B
  0, # person B
  0, # person B
])
```

For 2 persons



For 4 persons

4 Person Data Set

For 4 persons

```
#---------------------------------
# Define dataset and all_y_trues
#---------------------------------
data = np.array([
  [1, 2.5],     # person A
  [1, 3],       # person A
  [2.1, 3.4],   # Person A
  [2.1, 1], # person B
  [3.3, 1], # person B
  [3, 2.3], # person B
  [3, 2.5],       # person C
  [3, 3.6],       # person C
  [4.1, 5.4],    # Person C
  [4.5, 3.1], # person D
  [3.3, 1.9], # person D
  [5.9, 2.3], # person D
])
```

```
all_y_trues = np.array([
  01, # person A
  01, # person A
  01, # person A
  00, # person B
  00, # person B
  00, # person B
  11, # person C
  11, # person C
  11, # person C
  10, # person D
  10, # person D
  10, # person D
])
```

For 4 persons