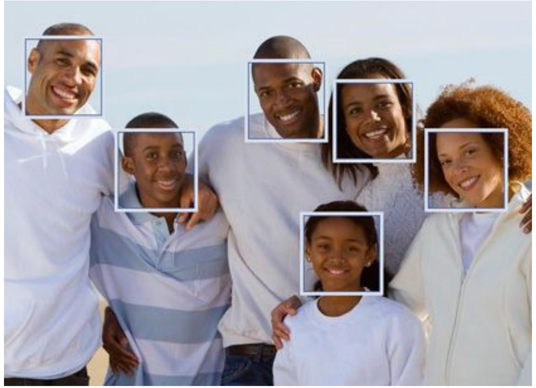
PROJECT NTDS
Face Emotion Recognition

Patryk OLENIUK
Carmen GALOTTA

INTRODUCTION - DATABASE



Aim of the project:

- Detect faces in an image
- Recognize and detect generic features of a human face
- Classify faces in discrete human emotions

Surprised

Sad

INTRODUCTION - DATABASE

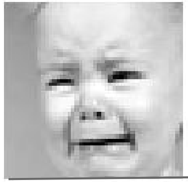
Database

- Kaggle competition
- 48x48 pixel grayscale images of labeled faces
- 35 000 face pictures

Classes:

0. Angry
1. Disgust
2. Fear
3. Happy
4. Sad
5. Surprise
6. Neutral

Images from the database:



Sad

Angry

Happy

Surprised

DATABASE - Data

Database

- Looks like taken from google images..
- Different facial perspective
- Some images corrupted

1

Data cleaning

1 Data cleaning

- Remove strange data
  - Pixelated images
  - Animation images
  - Black images

Looking at the image intensity histogram.

Important information removed as well.

- Merge class 0 and 1 (Angry and Disgust)
  - Very similar classes
  - Class “disgust” had a small amount of images.

Number of classes reduced to 6.
(Class “Angry” and “Disgust” merged)

1 Data cleaning

Image filtering - max(histogram) > 350 (discarded images example):

1 Data cleaning

Reduce the amount of “Happy” images

To have similar amount of data for each class, eliminated 3k “happy” images.

Number of 0 (angry) images	5 134	
Number of 1 (scared) images	4 829	
Number of 2 (happy) images	5 789	(was 9k -> reduced)
Number of 3 (sad) images	5 765	
Number of 4 (surprised) images	3 739	
Number of 5 (normal) images	5 876	

2

Exploring the models

2 Exploring the models

Model chosen: CONVOLUTIONAL NEURAL NETWORK

- Very good performances in image classification tasks
- Allows to extract many and very complicated features
- Models what do we(humans) do.

What we tried:

- 2 to 10 convolutional layers
- 1 to 4 fully connected layer in different positions
- Different size of patches (2 to 16)
- Different number of filters (10 to 64)
- Regularization and other techniques

2 Exploring the models - Techniques

- Hyperparameters
  - Size and number of filters
  - Number of layers
- Pooling layers
  - Reduce the dimensionality and thus the computational cost
- Dropout
  - Random deactivation of some unit in the NN with a defined probability

3

Our neural network

3 Our convolutional neural network

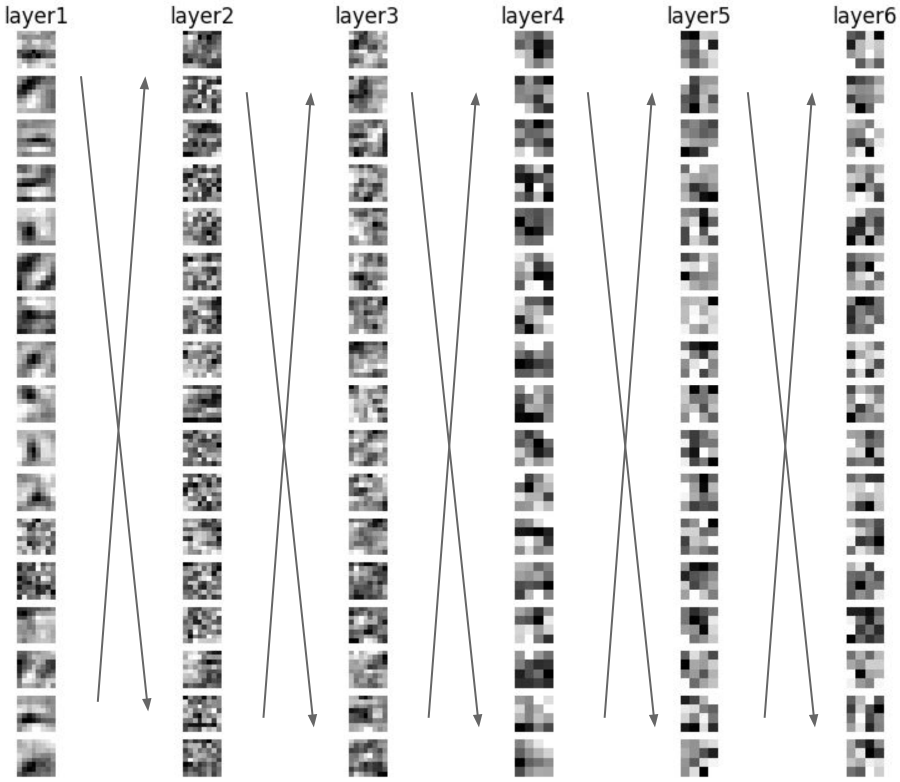
Our network consists of <u>6 convolutional</u> layers and a <u>final fully connected</u>

1. Convolutional layer
   a. Filter number: 22
   b. Filter size: 8x8
   c. ReLU activation
2. Convolutional layer
   a. Filter number: 22
   b. Filter size: 8x8
   c. Pool size: 2
   d. ReLU activation
3. Convolutional layer
   a. Filter number:22
   b. Filter size: 8x8
   c. ReLU activation
4. Convolutional layer
   a. Filter number: 22
   b. Filter size: 4x4
   c. Pool size: 4
   d. ReLU activation

3 Our convolutional neural network

5. Conv. layer
   a. Filter size: 4x4 – 22 filters
   b. ReLU activation
6. Conv. layer
   a. Filter size: 4x4 – 22 filters
   b. Dropout
   c. ReLU activation
7. Fully connected
   a. Size: (792, 6)
   b. Softmax

Adam Optimizer: learning rate 0.001



4

Performances and results

4 Performances and results

- Test accuracy: 56%
- Iterations: 20k
- Loss: 0.38
- Train accuracy: 84%

```
Iteration i= 19400 , train accuracy= 0.875 , loss= 0.398623
test accuracy= 0.4375

Iteration i= 19500 , train accuracy= 0.875 , loss= 0.293031
test accuracy= 0.4375

Iteration i= 19600 , train accuracy= 0.875 , loss= 0.39247
test accuracy= 0.484375

Iteration i= 19700 , train accuracy= 0.875 , loss= 0.430336
test accuracy= 0.453125

Iteration i= 19800 , train accuracy= 0.8125 , loss= 0.450704
test accuracy= 0.53125

Iteration i= 19900 , train accuracy= 0.84375 , loss= 0.373512
test accuracy= 0.4375

Iteration i= 20000 , train accuracy= 0.84375 , loss= 0.388213
test accuracy= 0.5625
```

4 Performances and results

20k Iterations to train the model

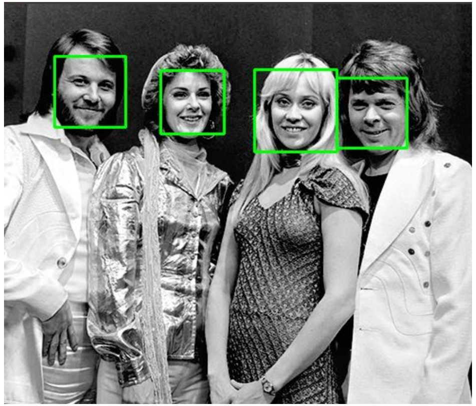
- Train accuracy= 84.4%
- Loss= 0.38
- Test accuracy= 56.25%

- Accuracy for each class

  - Angry: 69.6%
  - Scared: 57.1%
  - Happy: 57.1%
  - Sad: 31.6%
  - Surprised: 72.7%
  - Normal: 45.5%

4 Face Extraction from an image OpenCV + examples

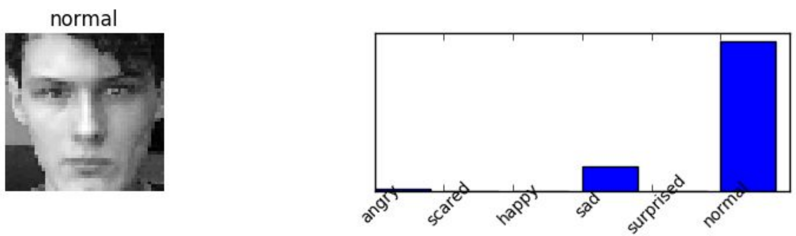
- Used OpenCV 3.0
- Face model and extraction based on the online tutorial: https://realpython.com/blog/python/face-recognition-with-python/
-

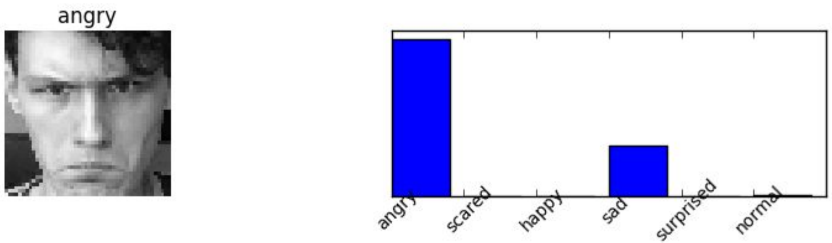


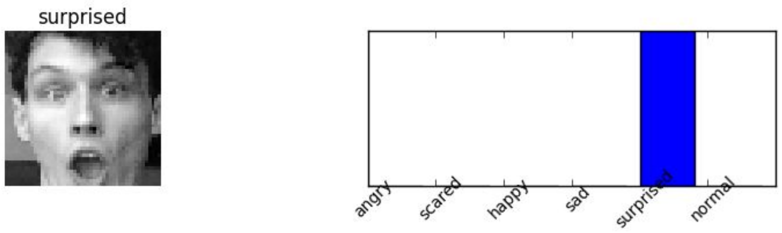
```
faces = faceCascade.detectMultiScale(
    image,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

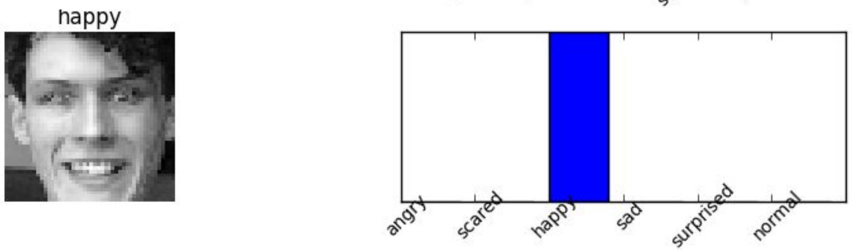
4 Feeding new (unlabeled) data

E.G. From the camera, file -->>









Conclusions and comment

- High computational power to train the network (~20h)
- Overall accuracy of about 56%
- Very good accuracy for ”happy” and “surprised” class
- Noise due to images labeled wrong or not centered in the picture
- Complicated features to extract in face emotions

Possible improvements:

- Randomly flip the images
- Deeper neural network to extract more features
- More sophisticated image pre-processing (e.g. straighten faces).
- Run CNN on GPU / s.

THANKS!

Questions?

Time for a demo ! →