Ministério da Educação

Universidade Federal do Pará

Programa de Pós-Graduação em Computação Aplicada

- VISÃO COMPUTACIONAL

Prof. Daniel da Conceição Pinheiro

Monitores:

- Henrique Costa Fernandes

- Leon Matheus Oliveira Alves

DETECÇÃO E RECONHECIMENTO DE FACES COM PYTHON

DETECÇÃO VS RECONHECIMENTO

Detecção de faces

- Encontrar a localização das faces em imagens ou vídeos
- Não tem como objetivo dizer de quem é a face
- É um dos aspectos mais fundamentais da Visão Computacional

Reconhecimento facial

- Reconhece as faces em imagens ou vídeos
- Identifica e reconhece as pessoas
- Pode determinar se uma pessoa é “conhecida” checando se a face existe na base de dados

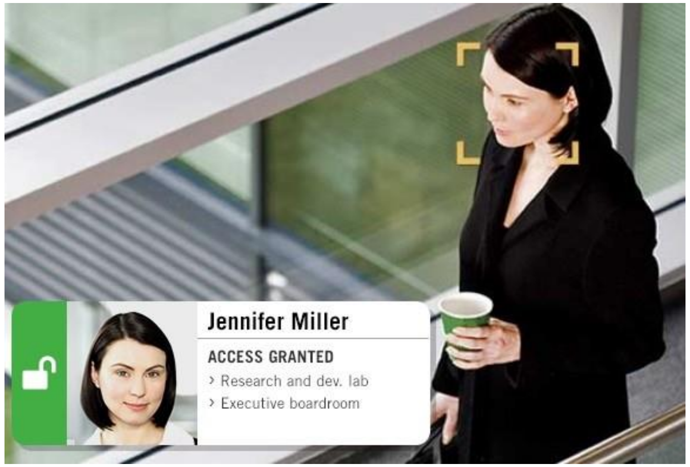
APLICAÇÕES DE DETECÇÃO DE FACES

- Contagem de pessoas
- Autofoco em câmeras (detecção de sorrisos)
- Alarmes
- Primeiro passo para rastreamento de objetos, reconhecimento de emoções e várias outras aplicações de Visão Computacional

Fonte: Pexels

APLICAÇÕES DE RECONHECIMENTO FACIAL

- Controle de acesso em áreas restritas
- Desbloqueio de celulares
- Identificação de pessoas
- Chamada eletrônica (virtual ou presencial)



Fonte: FaceFirst

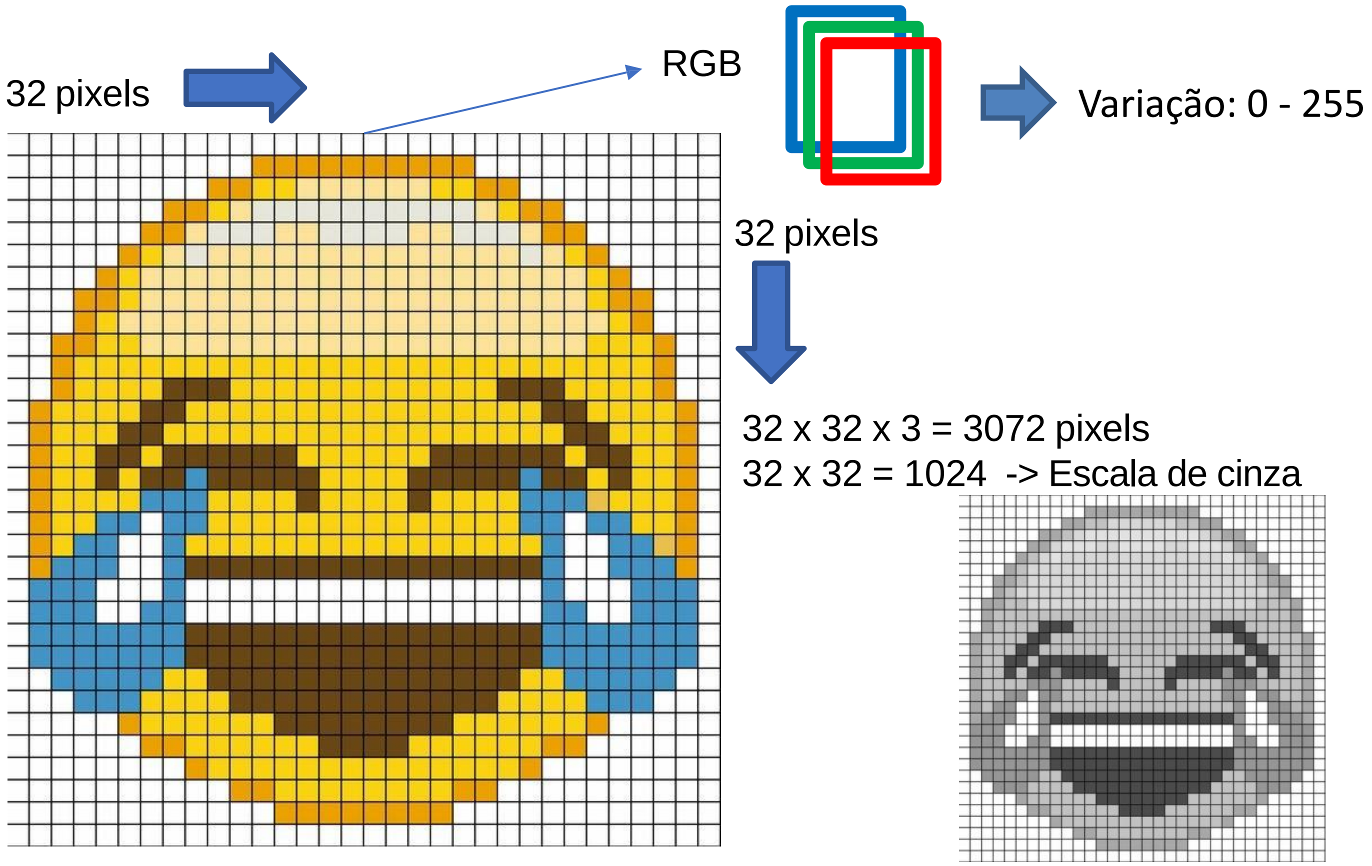
OPENCV

- OpenCV (Open Source Computer Vision Library)
- Escrito em C/C++
- Possui mais de 2500 algoritmos otimizados, incluindo algoritmos clássicos e outros do estado da arte
- Possui a licença BSD, o que torna fácil para empresas utilizarem e modificarem o código
- https://opencv.org/

DLIB

- Toolkit moderno escrito em C++ que possui algoritmos e algumas ferramentas
- Utilizado para criar softwares complexos que resolvem problemas do mundo real
- Ótima documentação
- Utilizado na indústria e academia em uma grande quantidade de domínios, como por exemplo: robótica, dispositivos embarcados, smartphones e ambientes computacionais que exigem alto desempenho
- Licença open source, permitindo utilizar em qualquer aplicação
- http://dlib.net/

PIXELS: Menor informação disponível na imagem



32 x 32 x 3 = 3072 pixels
32 x 32 = 1024 -> Escala de cinza

CLASSIFICADOR CASCADE

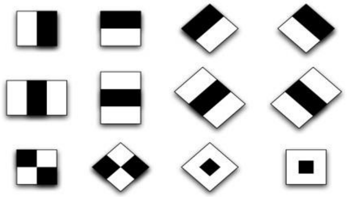
Faces (imagens positivas)

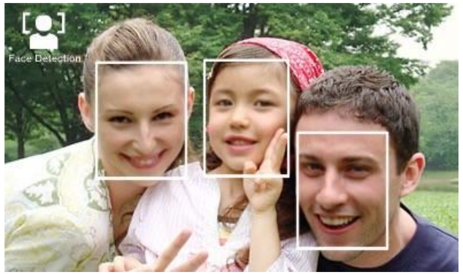
Não faces
(imagens não positivas)

AdaBoost
Training

Algoritmo de Machine Learning
(reconhecer padrões)

Feature
selection

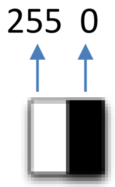




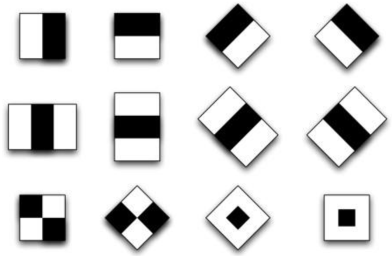
Aplicado para cada "sub-janela"

CLASSIFICADOR CASCADE

Soma dos pixels brancos
–
Soma dos pixels pretos

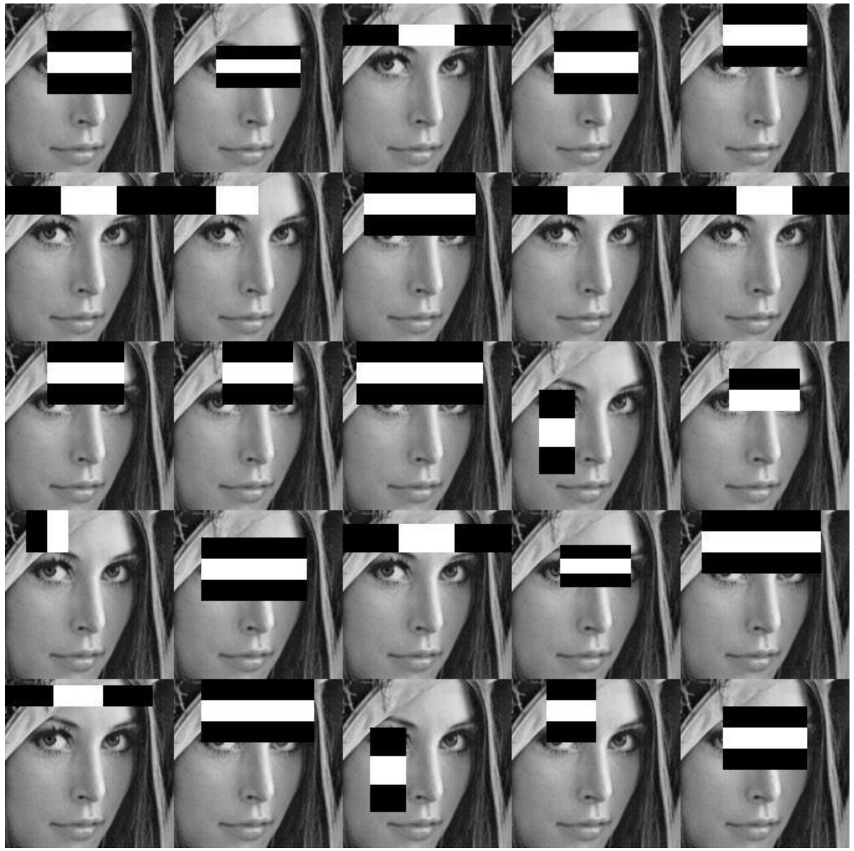


Mais de 160.000 combinações em uma imagem 24x24!

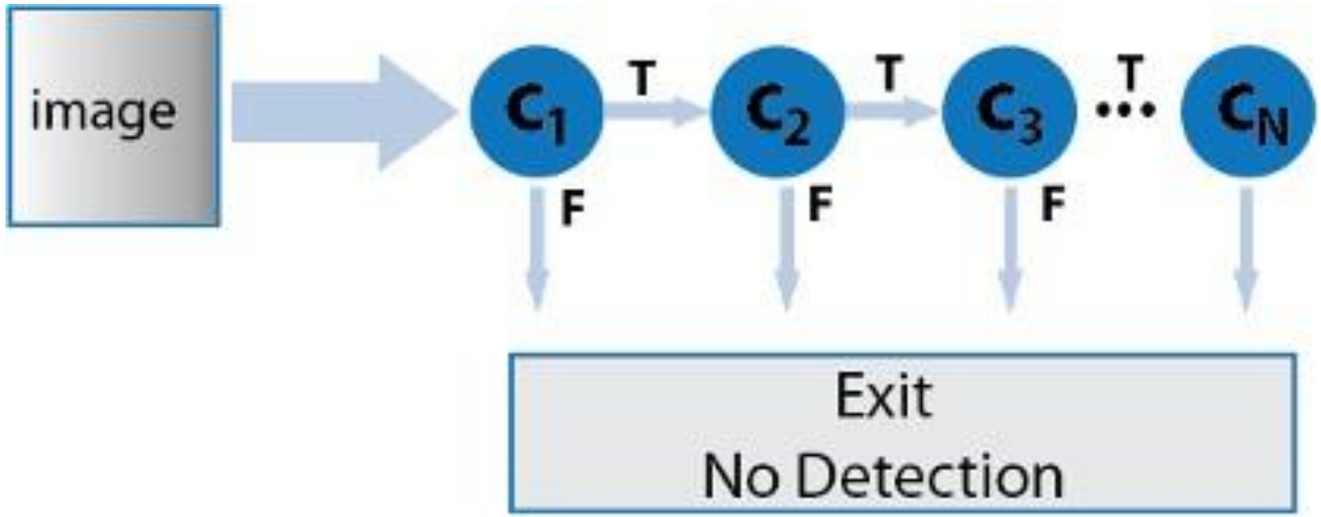


2 3 5 6
8 9 2 1

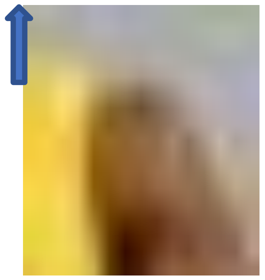
CLASSIFICADOR CASCADE

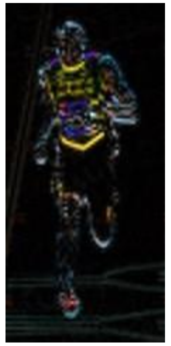


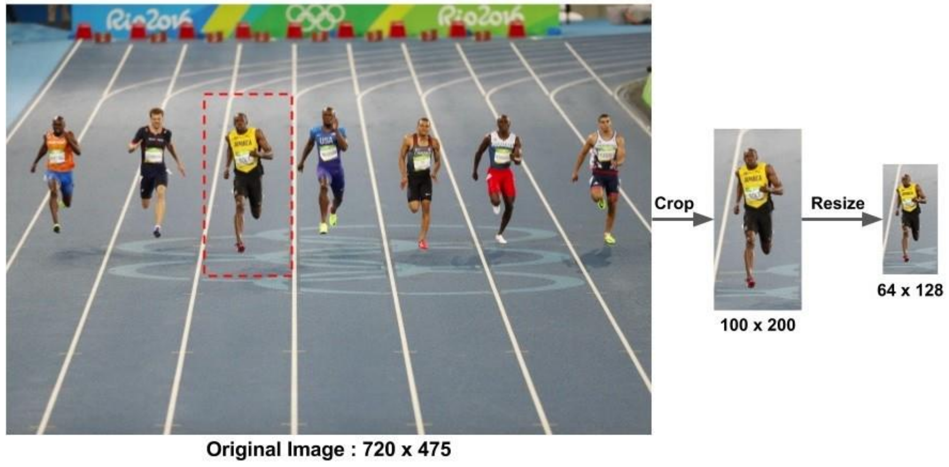
CLASSIFICADOR CASCADE

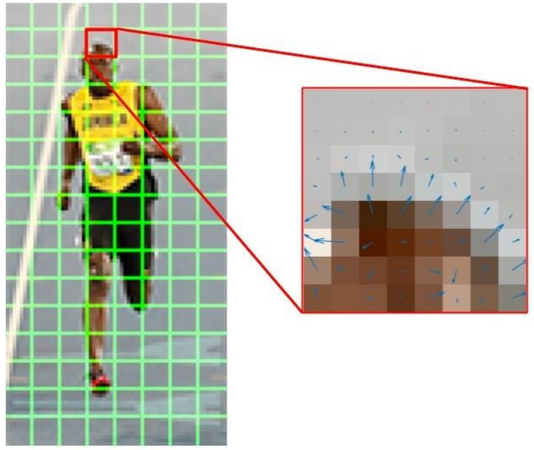


HOG – HISTOGRAMS OF ORIENTED GRADIENTS









2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

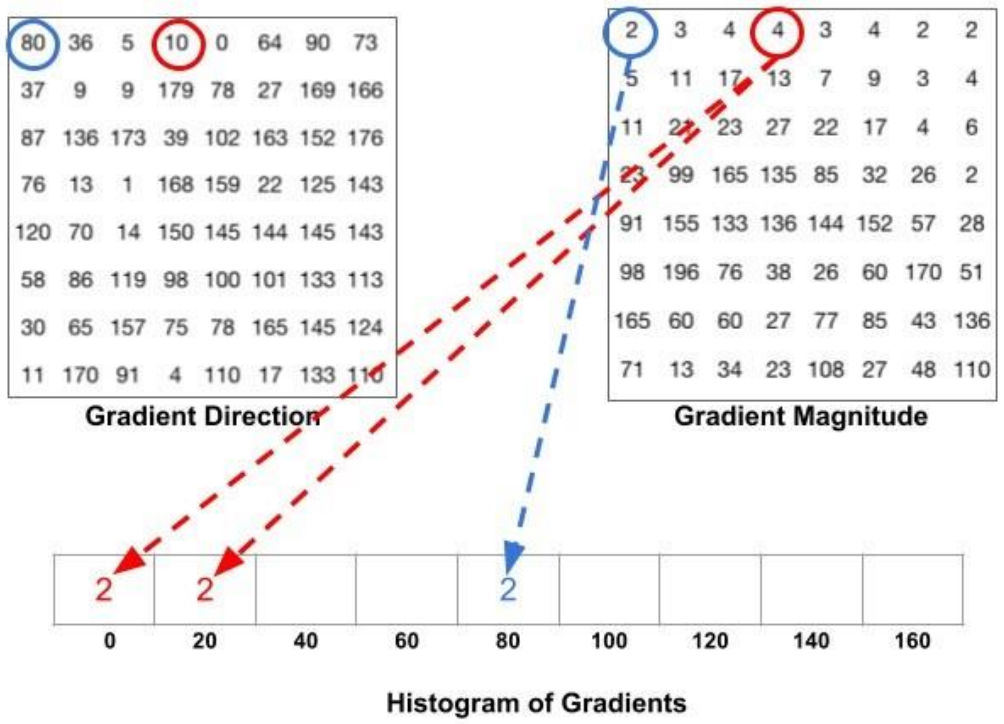
Gradient Direction

Derivada permite mensurar a taxa de mudança (derivada zero, derivada pequena e derivada alta)

Gradient vector (direção que o valor “aumenta”)

Fonte: https://www.learnopencv.com/histogram-of-oriented-gradients/

HOG - HISTOGRAMS OF ORIENTED GRADIENTS



80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

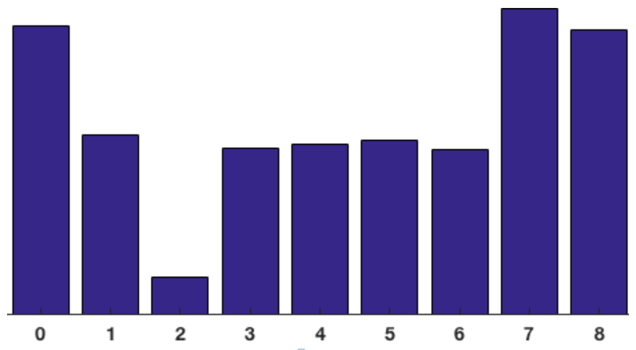
Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

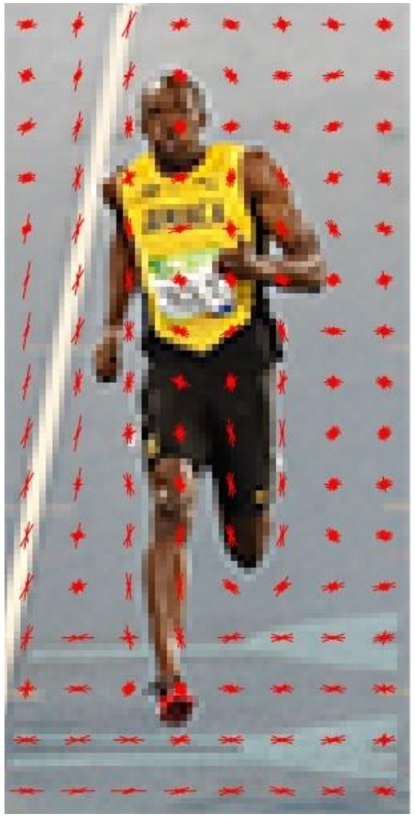
Gradient Magnitude

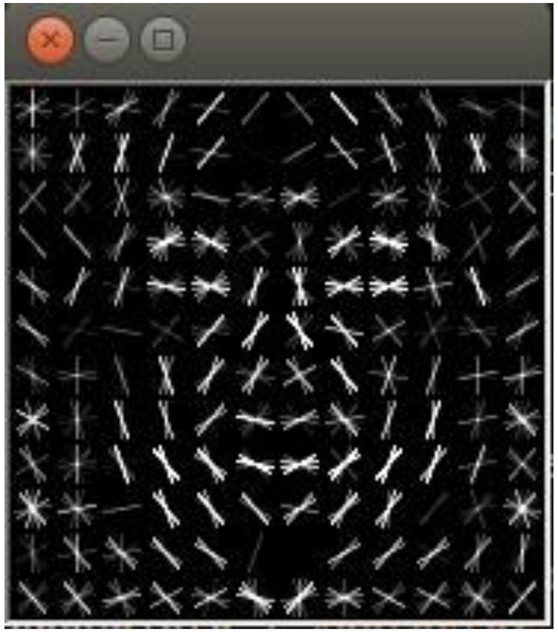
0	20	40	60	80	100	120	140	160
2	2			2				

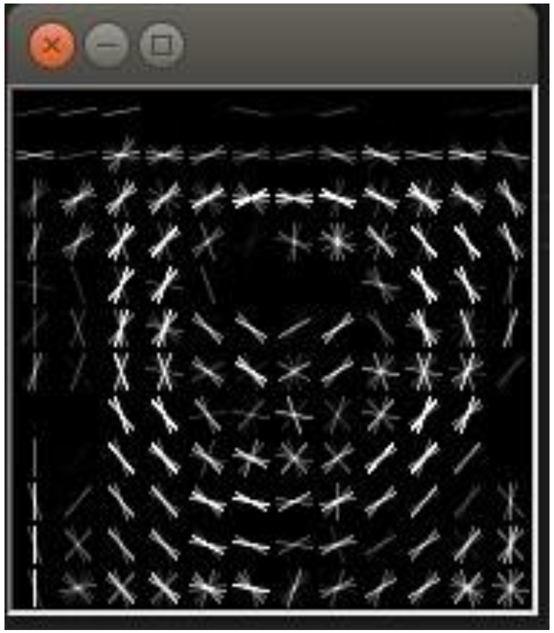
Histogram of Gradients



HOG - HISTOGRAMS OF ORIENTED GRADIENTS

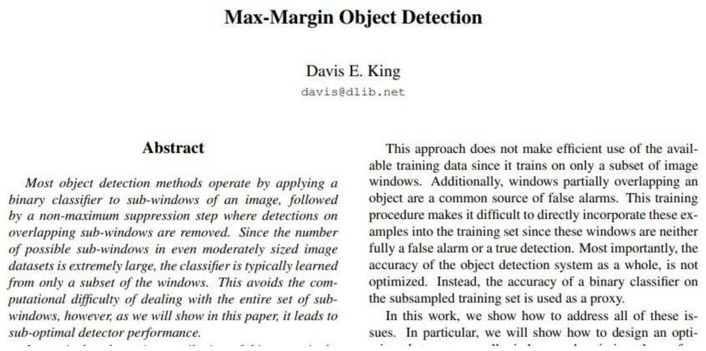






MMOD (CNN)

- MMOD (*Max-Margin Object Detection*) foi desenvolvido por Davis King, criador da biblioteca Dlib.
- É um método baseado em **Max-Margin Object Detection**.
- Muito preciso, porém requer GPU para executar com velocidade tolerável.
- Dlib model: *mmod_human_face_detector.dat*



Max-Margin Object Detection

Davis E. King
davis@dlib.net

Abstract

Most object detection methods operate by applying a binary classifier to sub-windows of an image, followed by a non-maximum suppression step where detections on overlapping sub-windows are removed. Since the number of possible sub-windows in even moderately sized image datasets is extremely large, the classifier is typically learned from only a subset of the windows. This avoids the computational difficulty of dealing with the entire set of sub-windows, however, as we will show in this paper, it leads to sub-optimal detector performance.

This approach does not make efficient use of the available training data since it trains on only a subset of image windows. Additionally, windows partially overlapping an object are a common source of false alarms. This training procedure makes it difficult to directly incorporate these examples into the training set since these windows are neither fully a false alarm or a true detection. Most importantly, the accuracy of the object detection system as a whole, is not optimized. Instead, the accuracy of a binary classifier on the subsampled training set is used as a proxy.

In this work, we show how to address all of these issues. In particular, we will show how to design an opti-

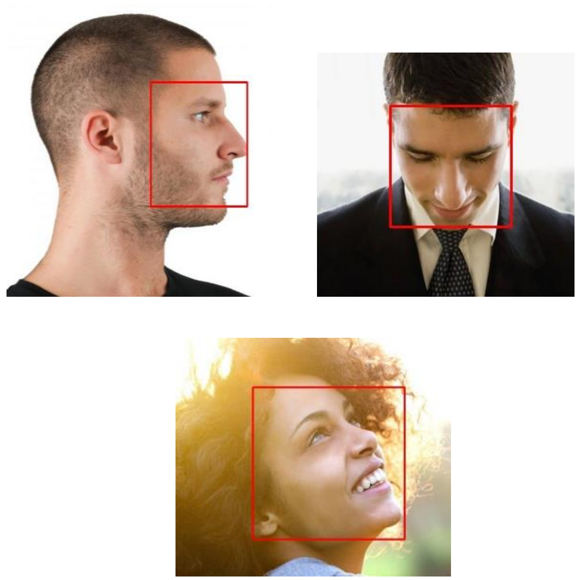
The Max-Margin Object Detection is well described Davis King's paper: https://arxiv.org/abs/1502.00046

MMOD (CNN)

- Enquanto outras técnicas tem dificuldades em detectar faces “de lado”, este detector baseado em CNN é capaz de detectar faces em praticamente todos os ângulos

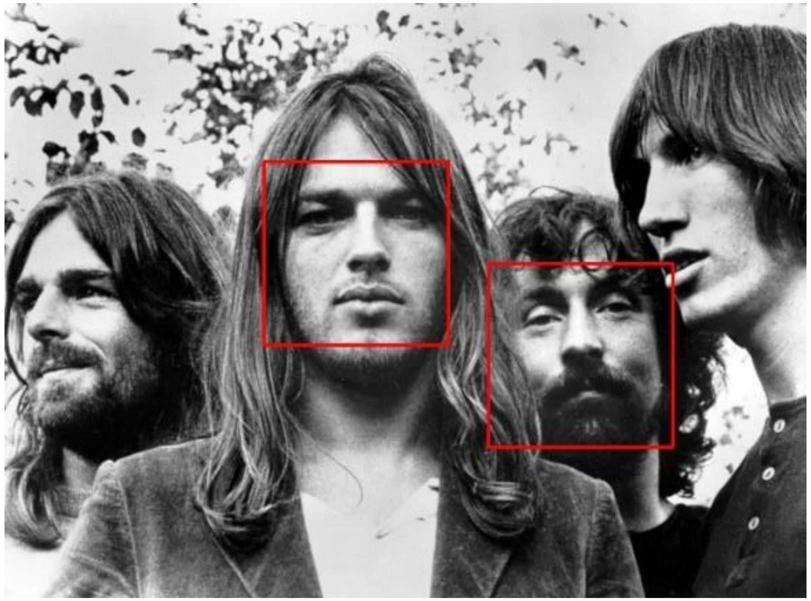
- MMOD pode detectar faces que não estão em posição frontal (que estão olhando diretamente para o lado)

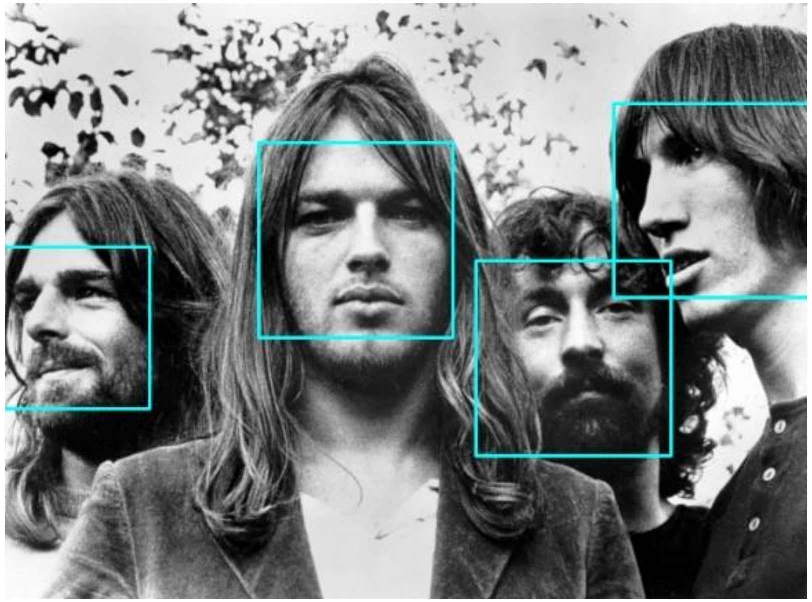
Alguns exemplos nos quais a técnica HOG falha, mas a CNN é capaz de detectar



MMOD (CNN)

HOG+SVM **vs.** **MMOD / CNN**





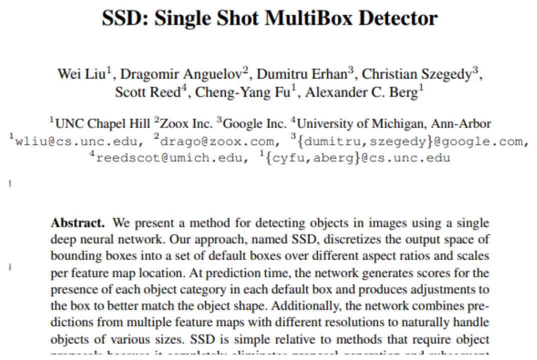
SSD DETECTOR

SSD: Single Shot MultiBox Detector

Artigo publicado em Novembro de 2016

Atingiu novos recordes em questões de desempenho e precisão para tarefas de detecção de objetos

- **Single Shot**: as tarefas de localização e classificação são feitas em um único passo da rede neural artificial
- **MultiBox**: nome de uma técnica de regressão para gerar os "bouding boxes" desenvolvido por W Liu et al.



SSD: Single Shot MultiBox Detector

Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[3], Christian Szegedy[3], Scott Reed[4], Cheng-Yang Fu[1], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]Zoox Inc. [3]Google Inc. [4]University of Michigan, Ann-Arbor
[1]wliu@cs.unc.edu, [2]drago@zoox.com, [3]{dumitru,szegedy}@google.com, [4]reedscot@umich.edu, [1]{cyfu,aberg}@cs.unc.edu

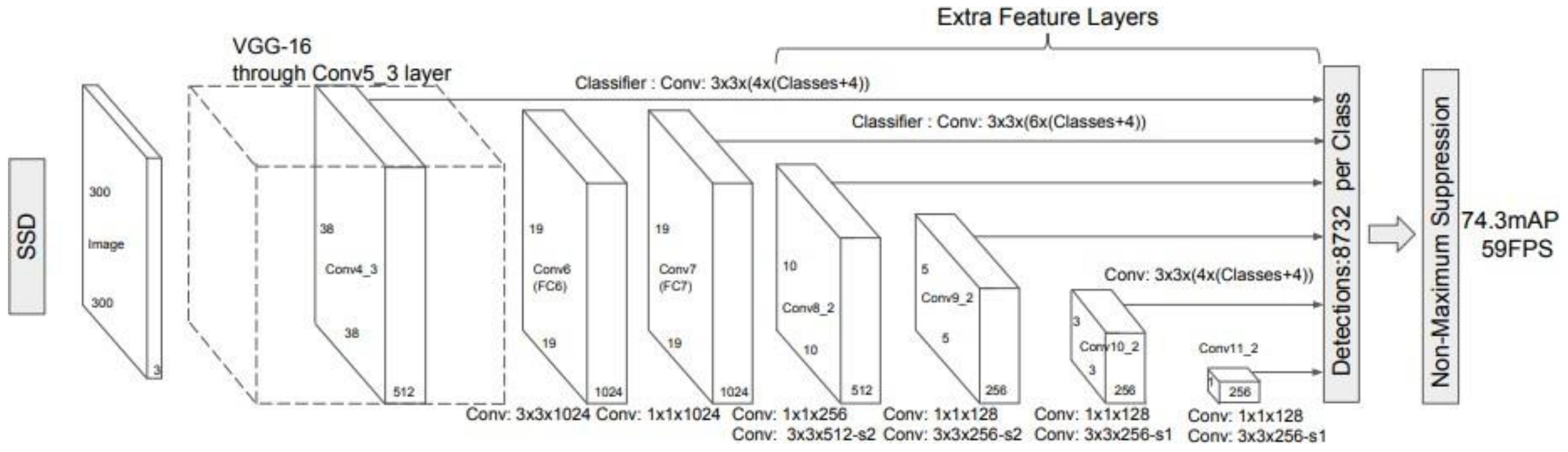
Abstract. We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simple relative to methods that require object

The paper about **SSD: Single Shot MultiBox Detector** (by W. Liu et al.) reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO.

https://arxiv.org/abs/1512.02325

SSD DETECTOR

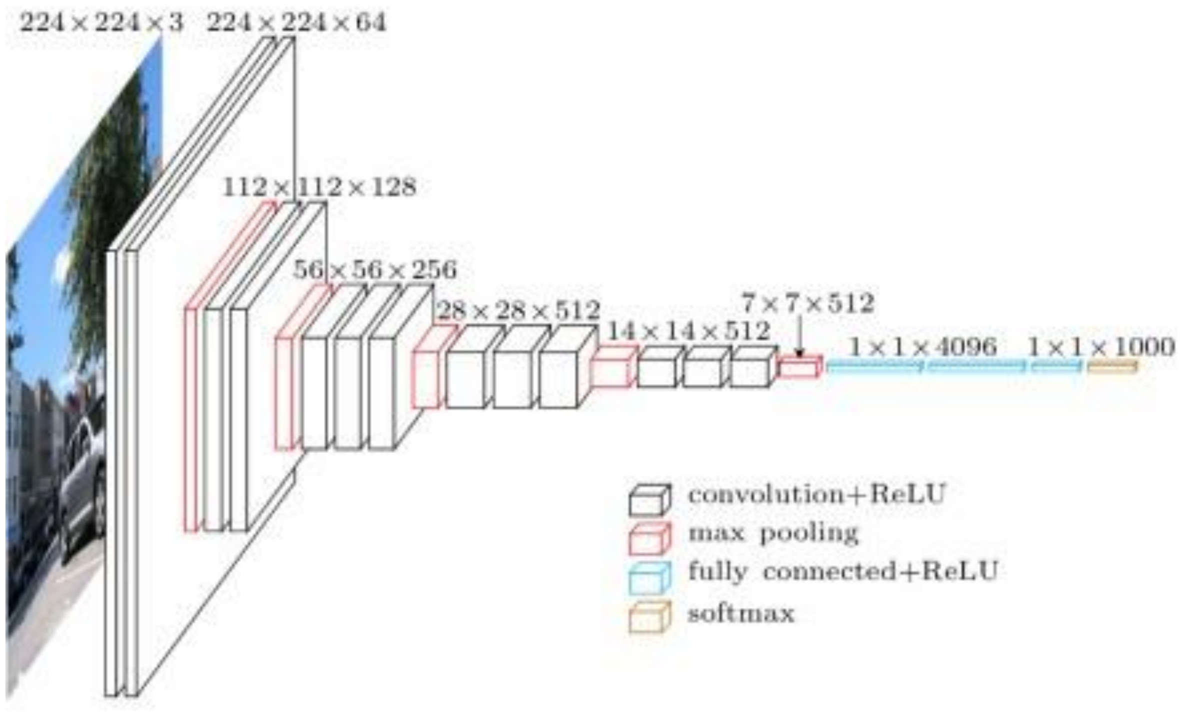
Architecture of Single Shot MultiBox detector (input is 300x300x3)



Fonte: SSD paper

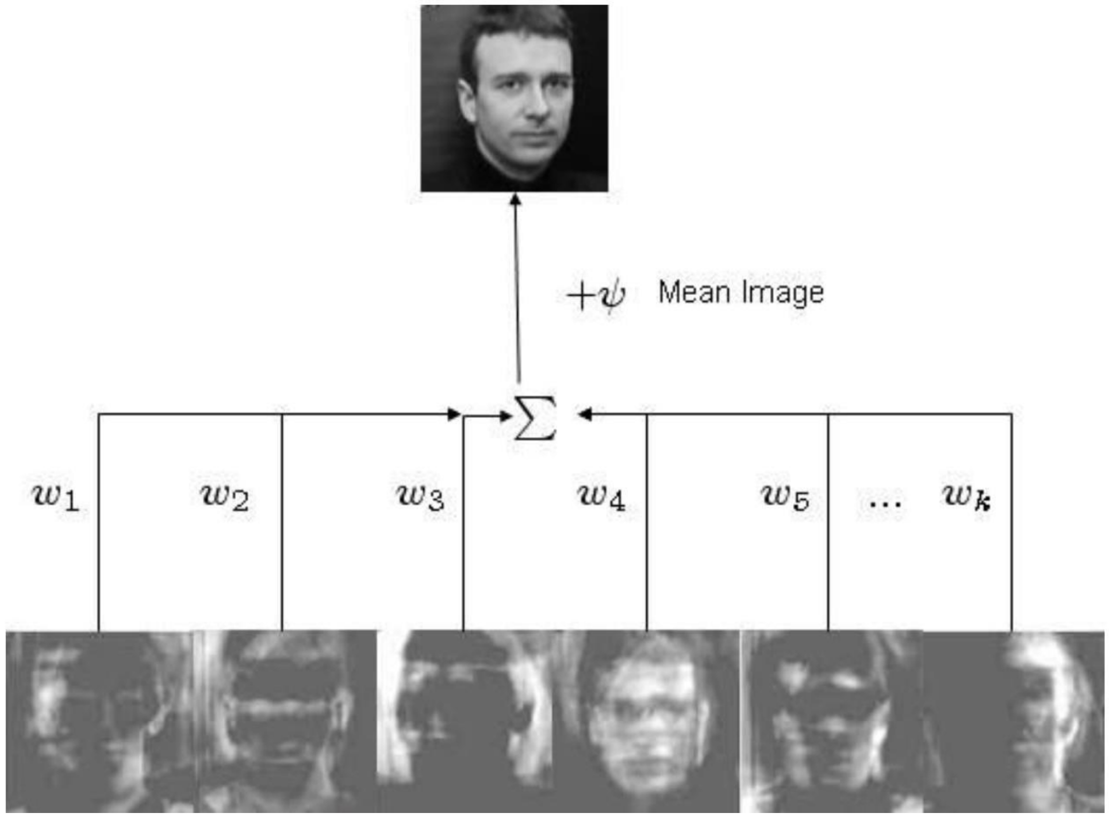
SSD DETECTOR

VGG architecture (input is 224x224x3)



Fonte: [SSD paper]

EIGENFACES



EIGENFACES

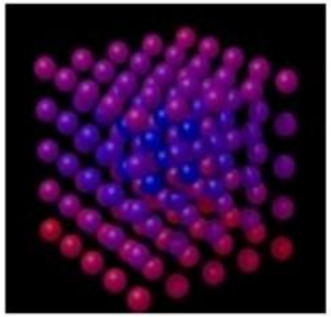
$$\begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & -3 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ -3 \end{bmatrix}$$

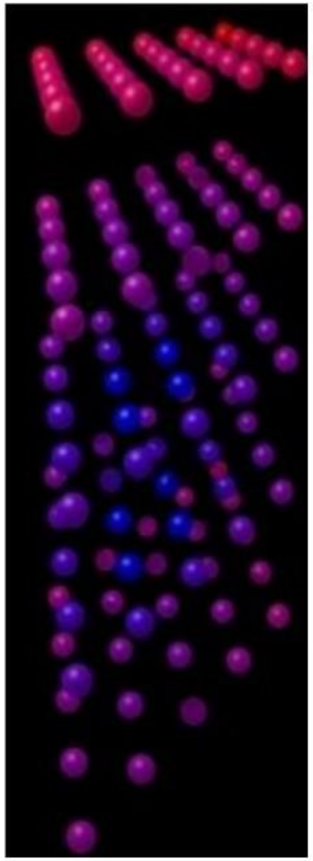
1 * 1 + 2 * 1 + 0 * 1
2 * 1 + 1 * 1 + 0 * 1
0 * 1 + 0 * 1 + -3 * 1

Eigen vector

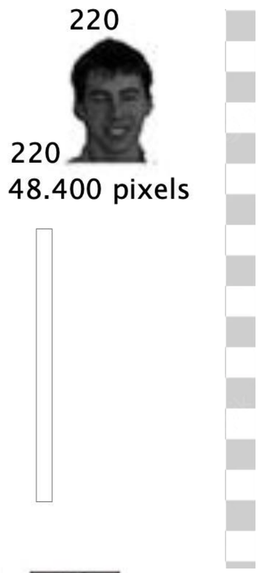
$$\begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & -3 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} = \underbrace{3}_{\text{Eigen value}} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

1 * 1 + 2 * 1 + 0 * 0
2 * 1 + 1 * 1 + 0 * 0
0 * 1 + 0 * 1 + -3 * 0

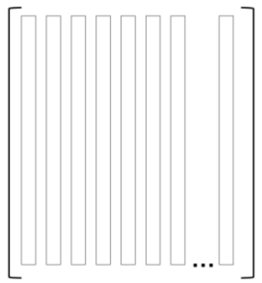




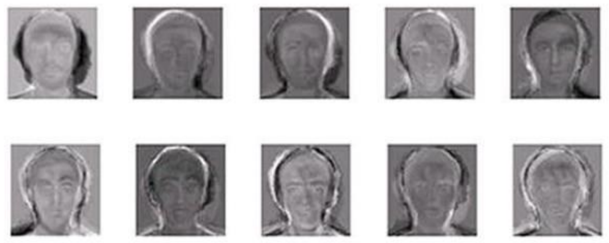
EIGENFACES



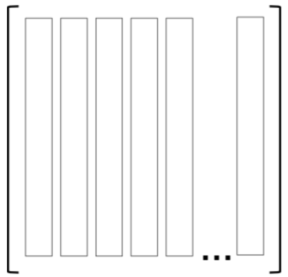
Cada face é uma coluna



PCA (Principal Component Analysis)



Eigen vector/Eigen faces



Redução de dimensionalidade

PARÂMETROS EIGENFACES

- **num_components –** número de componentes (eigenfaces) serão gerados (Principal Component Analysis)
- **threshold** = confiança para as previsões. Quanto menor o parâmetro da distância, melhor é a confiança.

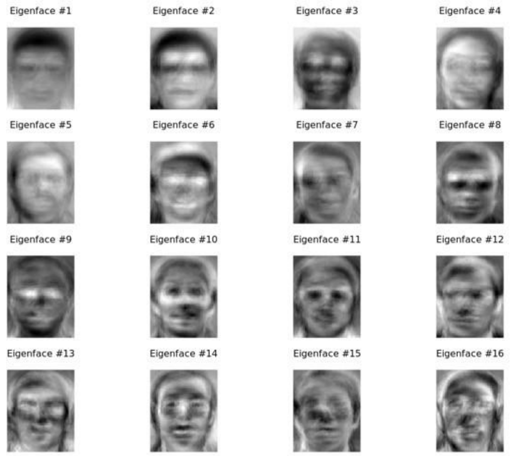
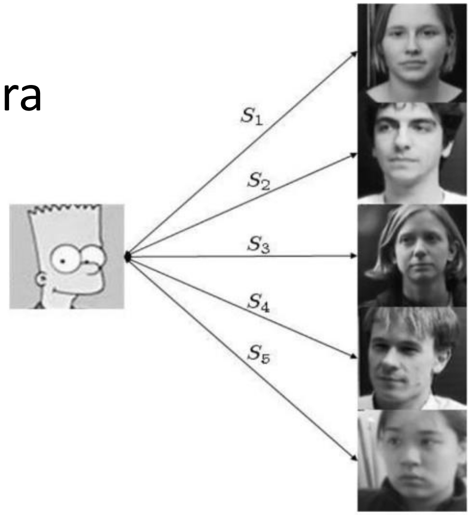
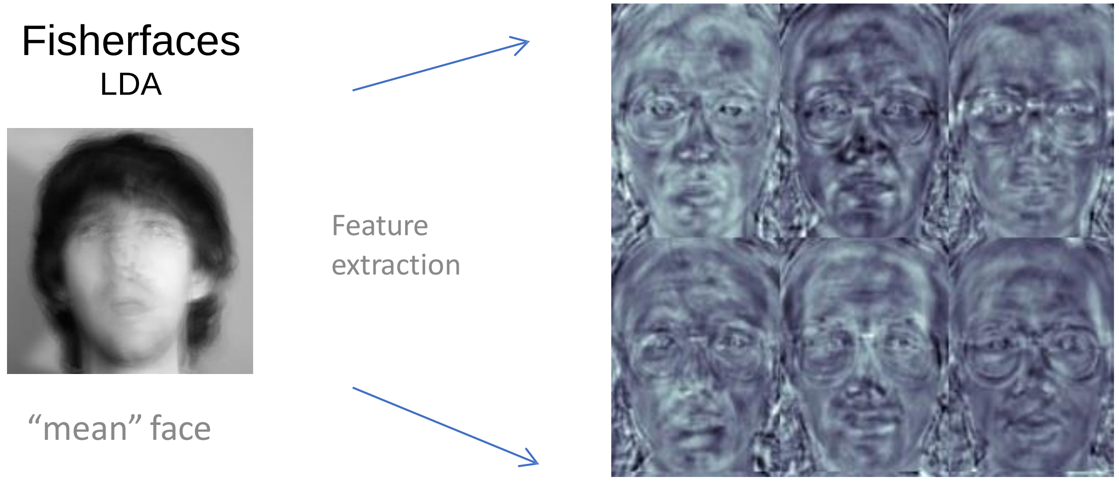


Image source: https://www.bytefish.de/blog/eigenfaces.html



FISHERFACES

- Alternativa melhorada do Eigenfaces
- Utiliza Linear Discriminant Analysis (LDA) (*Ronald A.* ***Fisher***)
- Atingiu menos taxas de erro porque o algoritmo é menos sensível à grandes variações na iluminação



FISHERFACES

- LDA também reduz a dimensionalidade
- É um classificador melhor porque LDA não está tão interessado em grandes variações, mas na maximização da separação entre as classes
- As características são extraídas separadamente, por exemplo, a iluminação de uma face não afetará as outras
- Extrai os componentes principais que diferenciam uma pessoa de outra. Os componentes de uma pessoa não se tornam mais importantes do que os outros

PARÂMETROS FISHERFACES

- **num_components –** número de componentes principais
- **threshold** - confiança

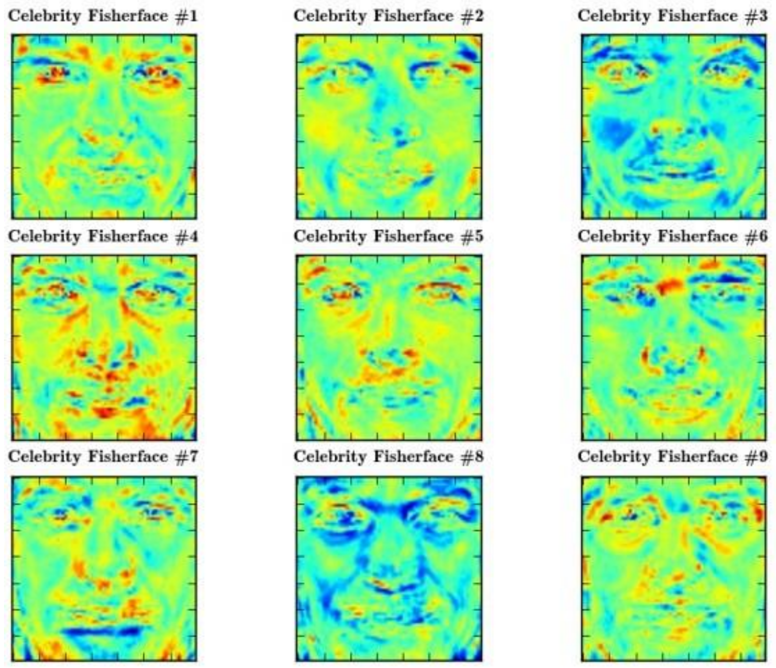
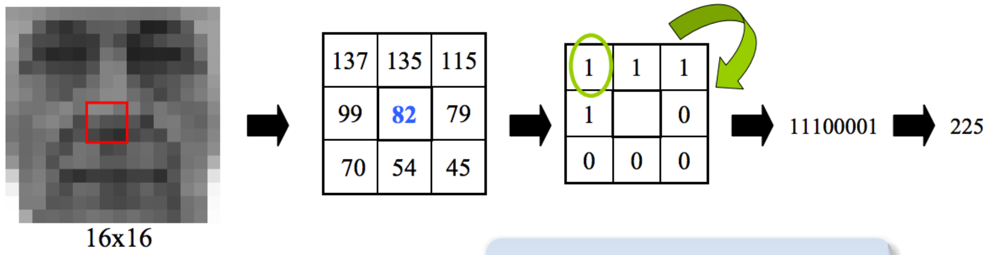
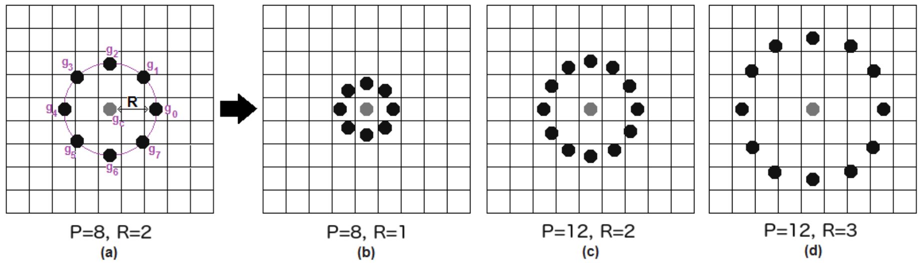


Image source: https://www.bytefish.de/blog/fisherfaces.html

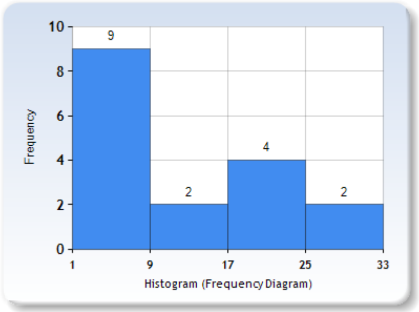
LBPH - LOCAL BINARY PATTERN HISTOGRAM

- Local Binary Pattern (LBP) – simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel
- It considers the result as a binary number
- Using the LBP combined with Histograms we can represent the face images with a simple data vector.





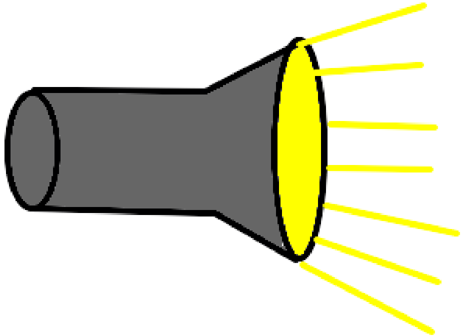
Source: https://en.wikipedia.org/wiki/Local_binary_patterns



LBPH - LOCAL BINARY PATTERN HISTOGRAM

12	15	18
5	8	3
8	1	2

If >= 8: 1
If < 8: 0



42	55	48
35	38	33
38	30	32

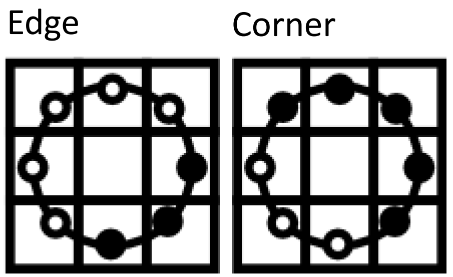
Binary = 11100010

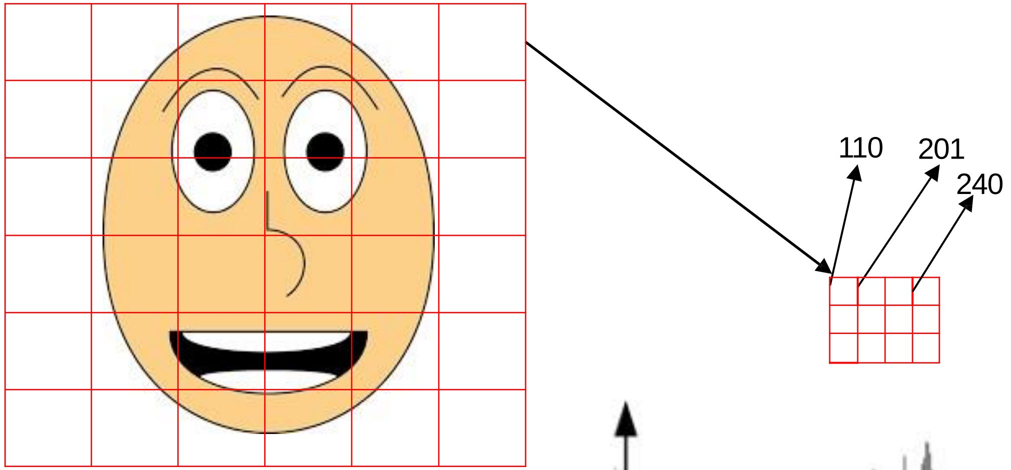
1	1	1
0	8	0
1	0	0

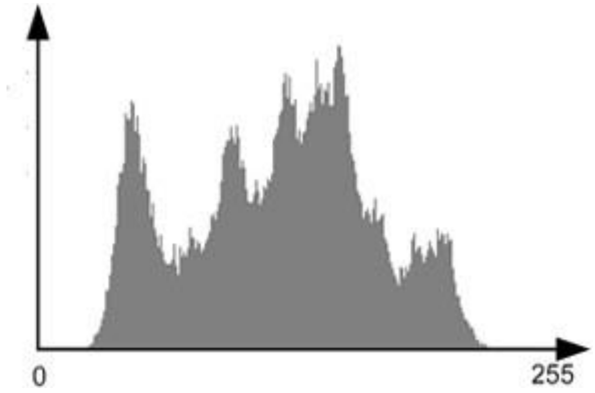
Binary = 11100010

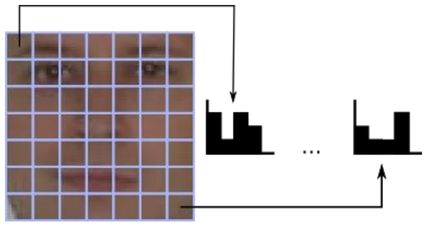
Decimal = 226

LBPH - LOCAL BINARY PATTERN HISTOGRAM







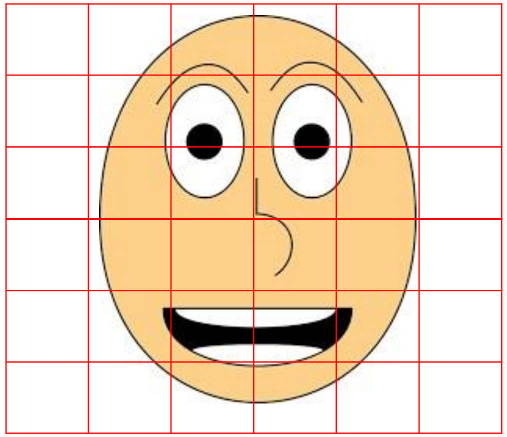


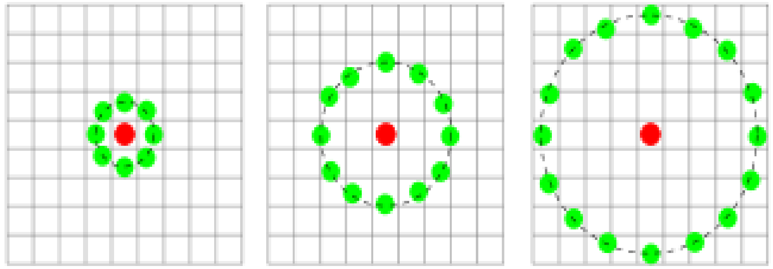
Source: https://bytefish.de/blog/local_binary_patterns/

LBPH - LOCAL BINARY PATTERN HISTOGRAM

1. Radius
2. Neighbors
3. grid_x and grid_y
4. Threshold

12	15	18
5	8	3
8	1	2





Source: https://en.wikipedia.org/wiki/Local_binary_patterns

CONTEÚDO ADICIONAL

FACE RECOGNITION - DEEP LEARNING METHOD

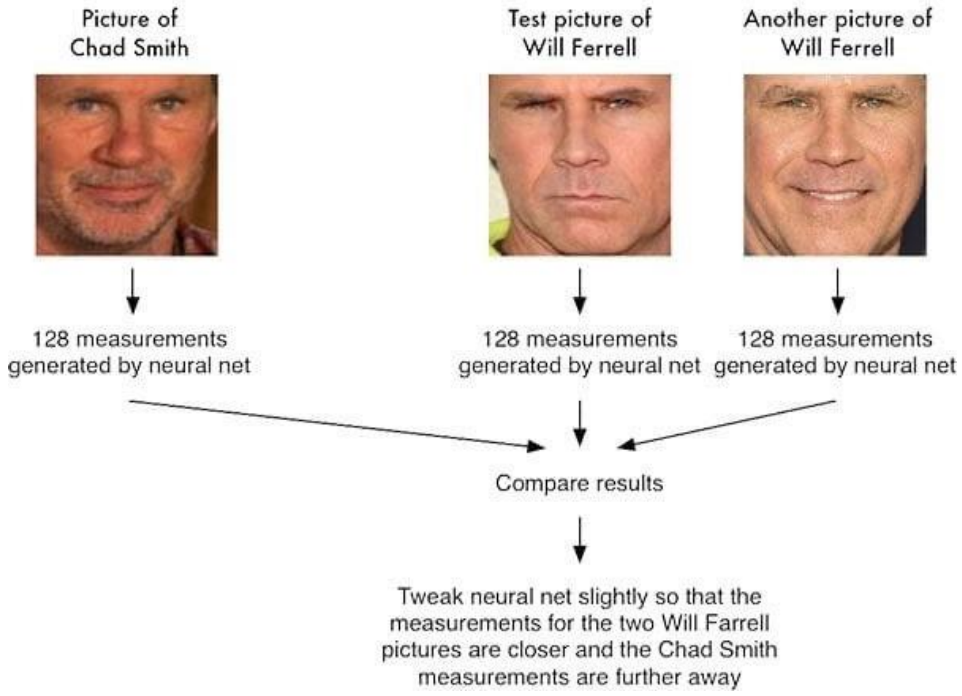
How face recognition with deep learning works?

The network output a real-valued feature vector.

Instead of returning a single label/classification (or even the bounding box coordinates, used in object detection), the neural network output is a real-valued feature vector with 128 measurements.

This method is known as ***deep metric learning.***

You can learn more about it on Davis King's article: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html



Picture of Chad Smith
Test picture of Will Ferrell
Another picture of Will Ferrell
128 measurements generated by neural net
128 measurements generated by neural net
128 measurements generated by neural net
Compare results
Tweak neural net slightly so that the measurements for the two Will Farrell pictures are closer and the Chad Smith measurements are further away

*Original image: [**"Machine Learning is Fun" blog**]()*

FACE RECOGNITION - DEEP LEARNING METHOD

A basic pipeline for face recognition using deep learning:

Training:
1. Detect face and its facial points landmarks
2. Detecting facial descriptors
3. Store the descriptors of all faces

Testing:
1. Detect face and its facial points landmarks
2. Calculating the distance between faces and comparing them
3.Finding the person's name from the descriptor by returning the smaller distance

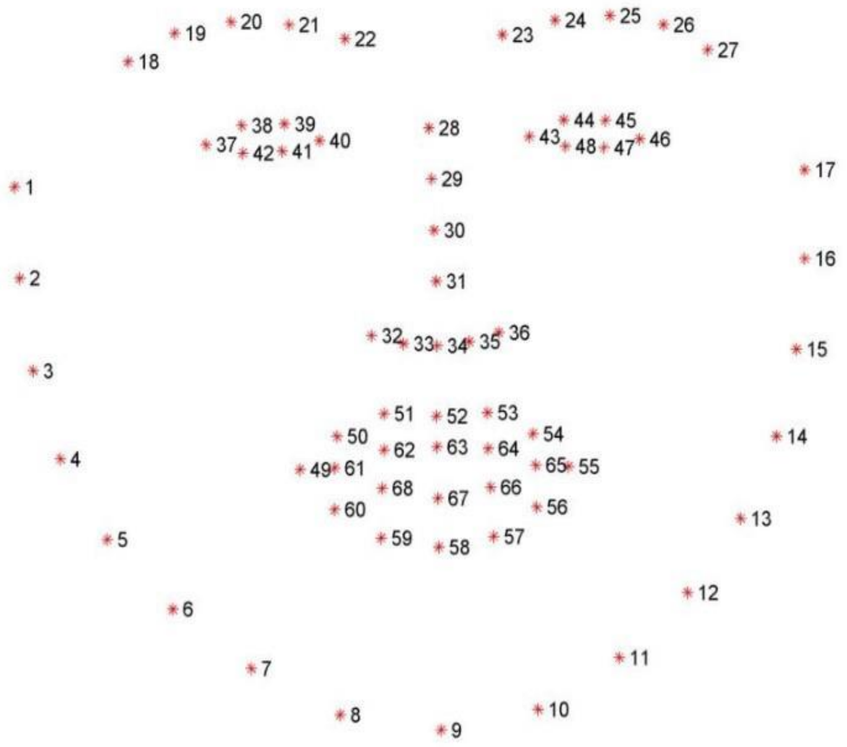
To implement this algorithm we are going to use **Dlib** functions and models trained by its author, Davis King.

FACIAL LANDMARKS

Detecting Facial Points (Landmarks)

- Jawline points: 1 – 17
- Right eyebrow points: 18 – 22
- Left eyebrow points: 23 – 27
- Nose bridge points: 28 – 31
- Lower nose points: 32 – 36
- Right eye points: 37 – 42
- Left eye points: 43 – 48
- Mouth points: 49 – 68

These 68 point annotations come after the iBUG 300-W dataset, which the dlib facial landmark predictor model was trained on. (*shape_predictor_68_face_landmarks.dat*)



Original image: *https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/*

FACIAL DESCRIPTORS

Detecting the Facial Descriptors

We need is a way to extract a few basic measurements from each face.

Input image

128 measurements generated from image

```
[[-0.23665184,  0.10333353, -0.02587086, -0.1013638 , -0.18408863,
  -0.00470767, -0.06494965, -0.06674606,  0.0450636 , -0.0476653 ,
   0.2058301 , -0.06543196, -0.34334478, -0.10437407, -0.0111365 ,
   0.17762549, -0.06317157, -0.07488471, -0.19065717, -0.09802263,
  -0.0379511 , -0.00093259,  0.05174568,  0.00783502, -0.07496783,
  -0.27508509, -0.12616815, -0.00694909,  0.01427757, -0.10069921,
   0.02106138, -0.00872875, -0.21809772, -0.09083748, -0.00660014,
   0.10903806, -0.16486223, -0.11058567,  0.17901379,  0.01369788,
  -0.13453469,  0.00441023,  0.09405778,  0.24202225,  0.20026542,
   0.0270569 ,  0.0038138 , -0.03924897,  0.08264139, -0.2861248 ,
   0.03378662,  0.16994755,  0.13304466,  0.04844667,  0.08608264,
  -0.05094043,  0.0495569 ,  0.21069752, -0.17605905, -0.01423565,
   0.03637291, -0.03286019, -0.13554637, -0.14761461,  0.21790203,
   0.19233903, -0.13524455, -0.13831057,  0.23347993, -0.0996014 ,
  -0.07110672,  0.06214657, -0.1037801 , -0.19015412, -0.30450463,
   0.10212639,  0.32803264,  0.19465008, -0.12812477, -0.0006907 ,
  -0.05840821, -0.03366487, -0.07097461,  0.00269504, -0.15650333,
  -0.05677082, -0.07269751,  0.0295887 ,  0.1743864 , -0.06831874,
  -0.01068573,  0.22309065,  0.1144831 ,  0.01271382,  0.04215482,
   0.02550944, -0.08972572, -0.12291306, -0.15439837, -0.04159033,
   0.11771025, -0.18411945,  0.02407164,  0.08229396, -0.09862326,
   0.17255184,  0.02950443, -0.02336782, -0.04844049, -0.01287997,
  -0.11040097,  0.0115043 ,  0.13323261, -0.20383728,  0.22639935,
   0.16896744,  0.00852558,  0.07681741,  0.21908414,  0.08944321,
  -0.0104048 ,  0.01055228, -0.07418033, -0.18171579,  0.07391257,
  -0.00257582,  0.06582376,  0.08675323]])
```

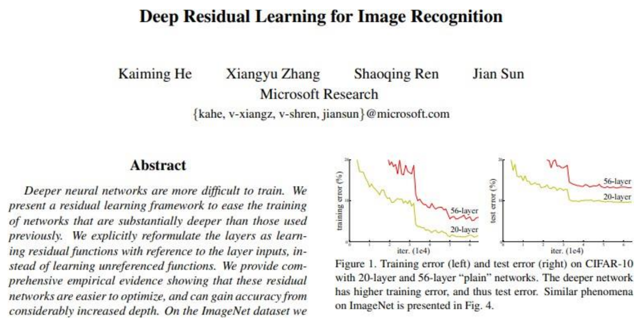
In the testing phase, we need to measure the unknown face the same way so we can find the known face with the closest measurements.

This process can also be known as **face encoding**.

FACIAL DESCRIPTORS

To detect the facial descriptors we are going to use the model *dlib_face_recognition_resnet_model_v1.dat*, which is a **ResNet** network with 29 conv layers.

It's essentially a version of the ResNet-34 network from the paper **Deep Residual Learning for Image Recognition,** but with a few layers removed and the number of filters per layer reduced by half.



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Paper: https://arxiv.org/abs/1512.03385

The network was trained from scratch on a dataset of about 3 million faces. After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person.

The final dataset is derived from a number of datasets:

- Face scrub dataset - http://vintage.winklerbros.net/facescrub.html
- VGG dataset - http://www.robots.ox.ac.uk/~vgg/data/vgg_face/
- And a large number of images Davis King (Dlib author) scraped from the internet.

FACIAL RECOGNITION

The last step is technically the easiest in the whole process.

All we have to do is find the person in our database of known people who has the closest measurements to our test image.

This can be done by using any basic machine learning classification algorithm. No sophisticated deep learning tricks are needed.

We can use **k-NN** to make the final face classification. Other traditional machine learning models could be used here as well.

- if two face descriptor vectors have a Euclidean distance between them less than the tolerance threshold (default=0.6) then they are from the same person, otherwise they are from different people.

K-NN

Example of k-NN classification

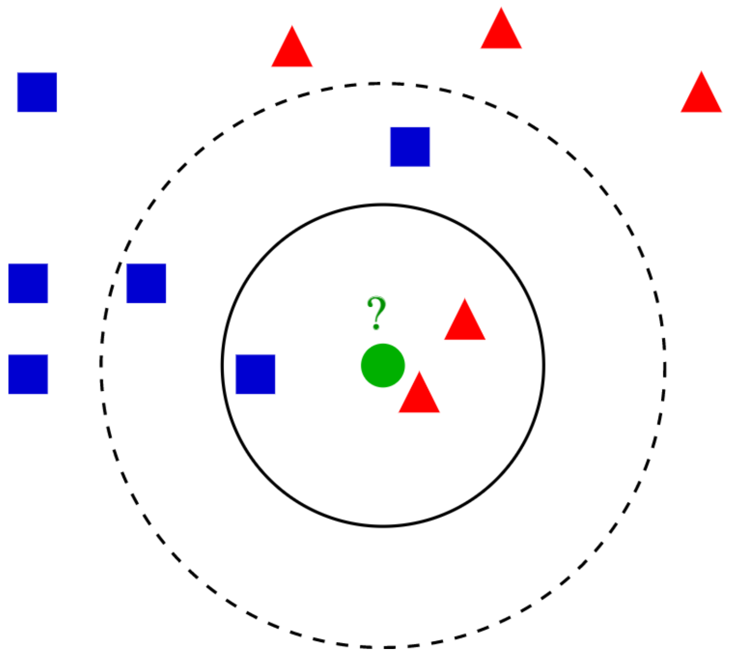


Image author: Antti Ajanki

In order to apply the k-nearest Neighbor classification, we need to define a distance metric or similarity function.
A common choices includes the **Euclidean distance**:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

FACIAL RECOGNITION

Some clarifications:

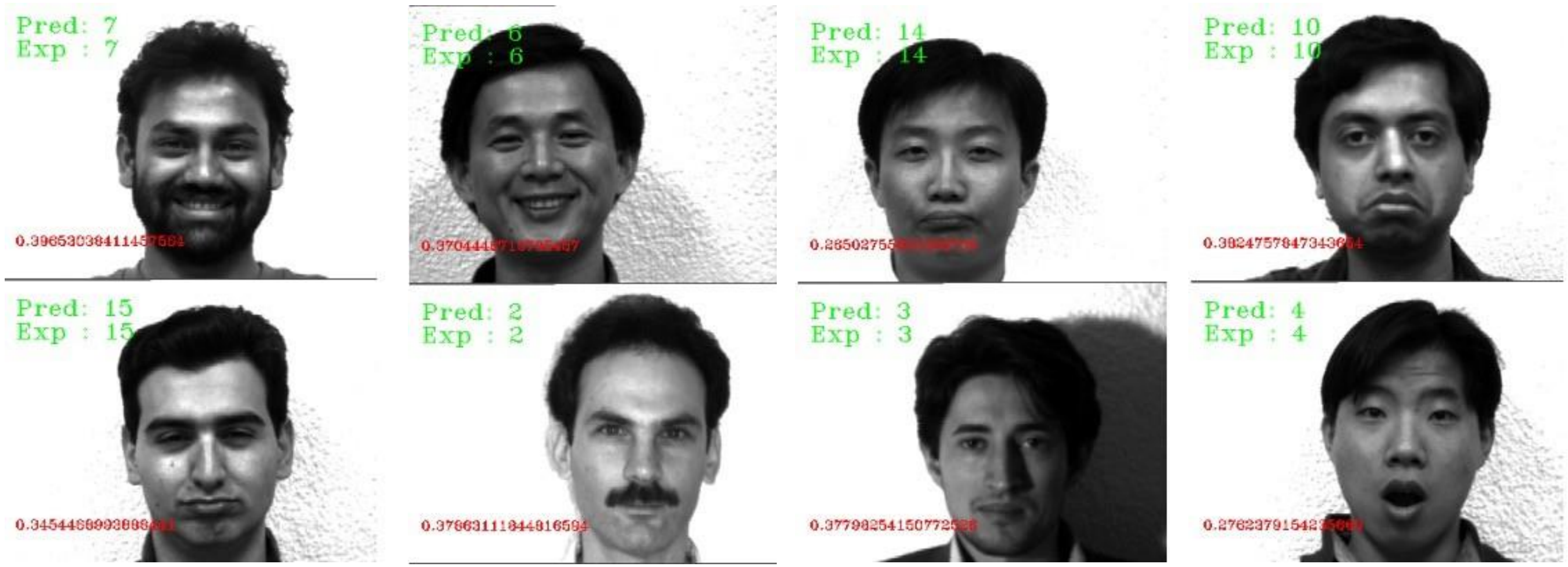
- Although the word training is used in this context, actually there isn't any "training" going on, like it was done for traditional face recognition methods (Eigenfaces, Fisherfaces and LBPH).

- So for example if you need to add a new person to the dataset then you just need to extract the 128-d face embeddings for the new faces and update the list (and the pickle file, if you saved them into disk).

- The confidence returned is not a true percentage, but a distance.

This means -> lesser the value greater is the confidence of the prediction.

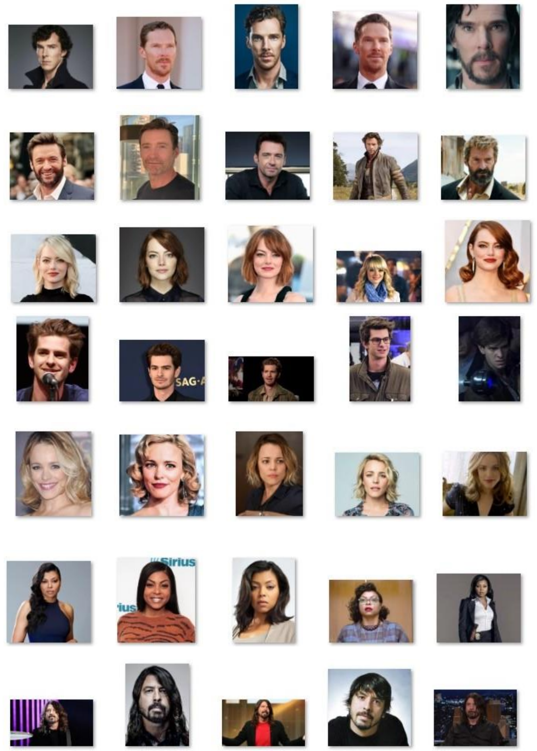
RECOGNITION RESULTS WITH DEEP LEARNING



Accuracy on YALE Faces Database: 1.0 (100%)

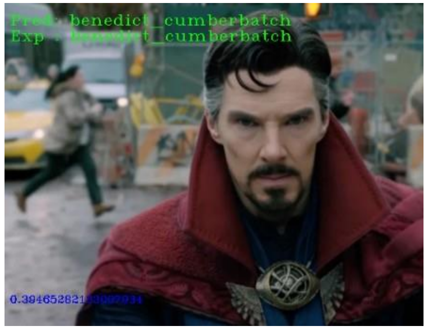
MORE CHALLENGING SITUATIONS - NATURAL SCENES

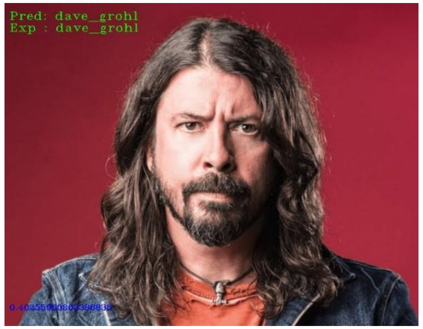
- The dataset we have used so far is considered "controlled“, because the photos were taken in a controlled environment.
- Now that we guaranteed we can achieve 100% accuracy with deep learning on such a scenario, let's increase the level of difficulty. Although the facial expression vary a lot, it was not "difficult enough".
- So for now we are going to load another dataset developed for this course, which contains photos of celebrities.
- These photos were taken in the real world, some were taken on a studio with controlled illumination and others on natural scenes.

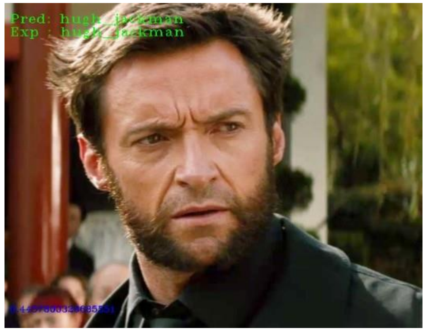


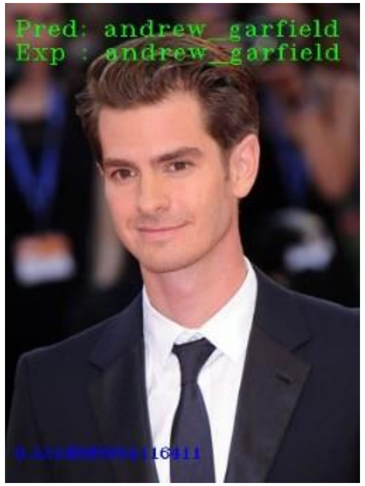
Some samples from celeb_dataset (before face cropping)

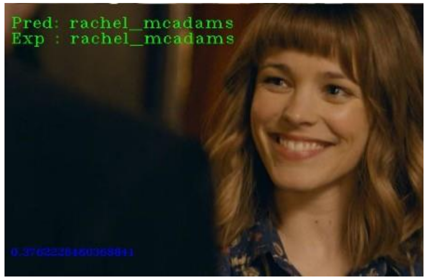
RECOGNITION RESULTS WITH DEEP LEARNING

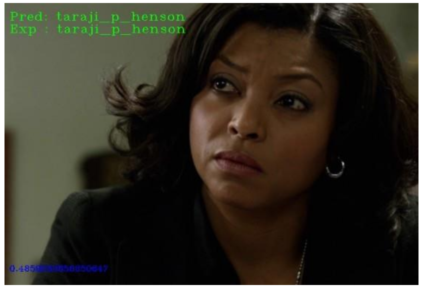


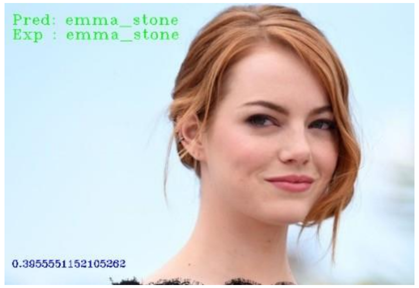












Accuracy with 0.5 tolerance => 0.714 - Accuracy with 0.6 tolerance => **1.0**

IMPROVED FACE RECOGNITION WITH DEEP LEARNING

The **face_recognition** library wraps around the Dlib's facial recognition functionality we've seen, making it easier and more practical to work with.

- It was developed by Adam Geitgey, a machine learning researcher.
- The process developed by Adam has some improvements, but the pipeline is almost the same:

1. Find all the faces on image
Encode a picture using the HOG algorithm to create a simplified version of the image. Using this simplified image, find the part of the image that most looks like a generic HOG encoding of a face.

2. Posing and Projecting Faces
Figure out the pose of the face by finding the main landmarks in the face. Once we find those landmarks, use them to warp the image so that the eyes and mouth are centered.

3. Encoding Faces
Pass the centered face image through a neural network that knows how to measure features of the face. Save those 128 measurements.

4. Finding the person's name from the encoding
Looking at all the faces we've measured in the past, see which person has the closest measurements to our face's measurements.

IMPROVED FACE RECOGNITION WITH DEEP LEARNING

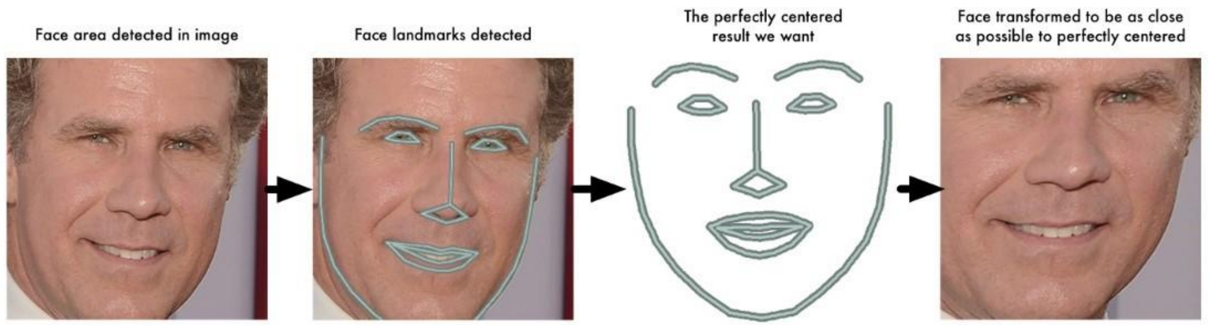
Posing and Projecting Faces

We need to deal with the problem that faces turned different
With the detdirections look totally different to a computer

Solution: warp each picture so that the eyes and lips are always in the sample place in the image (or, very close at least)

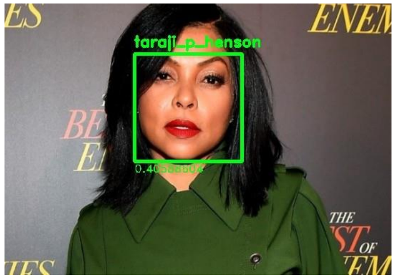
- Detect the 68 landmarks => rotate, scale and shear the image so that the eyes and mouth are centered as best as possible

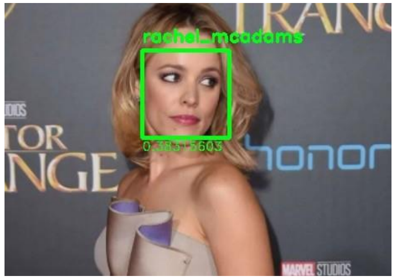
While for humans it's easy, computers would see these pictures as two completely different people.

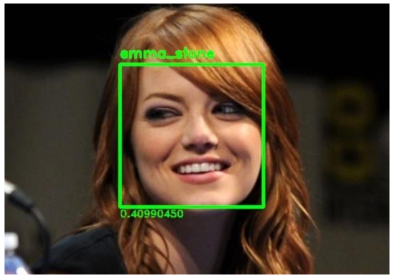

Face area detected in image
Face landmarks detected
The perfectly centered result we want
Face transformed to be as close as possible to perfectly centered

Original image: Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning

IMPROVED FACE RECOGNITION WITH DEEP LEARNING

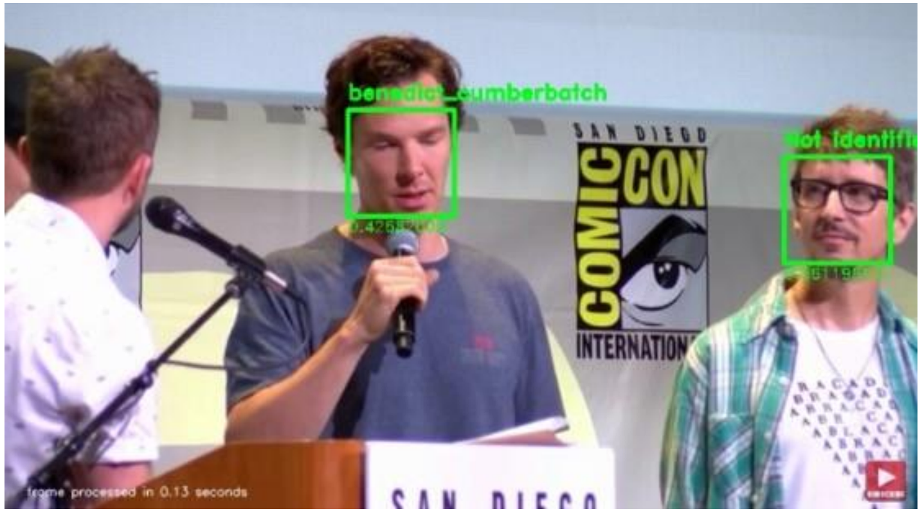


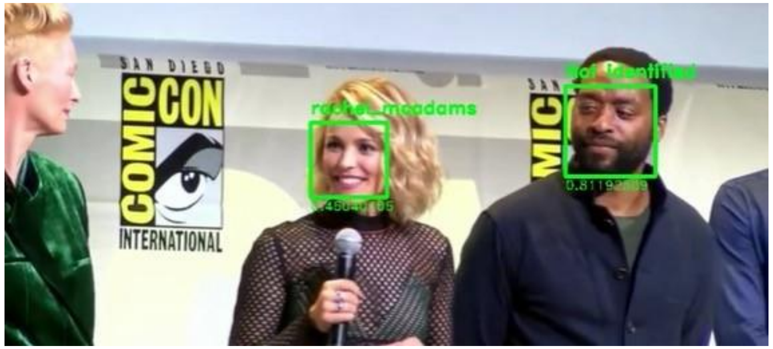


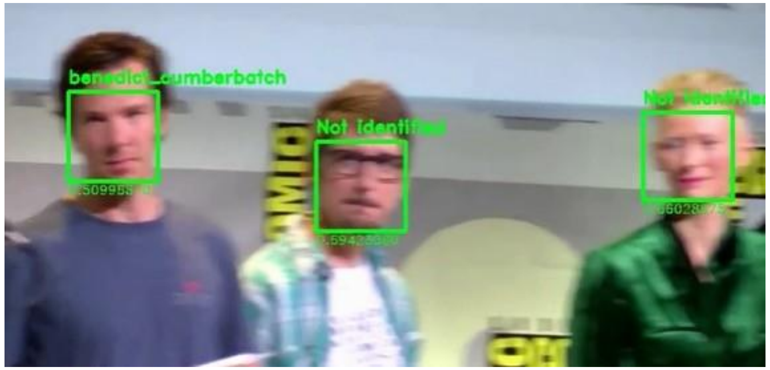


Accuracy:1.0 on celeb_dataset (*default tolerance*)

FACE RECOGNITION ON VIDEOS







CREATING YOUR OWN FACE DATASET

There are mainly 3 options:

1. Saving photos manually
2. Downloading the photos programmatically (with a Python script or API)
3. Take new photos with your webcam

CREATING YOUR OWN FACE DATASET

Recommendations when taking the photos:

- Different **lighting conditions** – in intensity or angle (whether on the left, right or in front of the face)
- Different **Facial expressions** - happiness, sadness, surprise, etc.
- Different **angles** (relative to the camera)

OTHER FACE DATASETS

Other datasets that could be used for testing:

- AT&T Database of Faces - https://www.kaggle.com/kasikrit/att-database-of-faces
- Unconstrained Facial Images - http://ufi.kiv.zcu.cz
- Flickr Faces - https://github.com/NVlabs/ffhq-dataset
- Google Facial Expression Comparison - https://research.google/tools/datasets/google-facial-expression/
- Youtube with Facial Keypoints - https://www.kaggle.com/selfishgene/youtube-faces-with-facial-keypoints

OTHER FACE RECOGNITION SOLUTIONS

OpenFace - a Python and Torch implementation of face recognition with deep neural networks and is based on the paper *FaceNet*, from Google Inc.

- https://cmusatyalab.github.io/openface/

Deepface - wraps several state-of-the-art models: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace, Dlib and SFace.

- https://github.com/serengil/deepface

InsightFace - an open source 2D & 3D deep face analysis toolbox, mainly based on PyTorch and MXNet.

- https://github.com/deepinsight/insightface

FaceNet - TensorFlow implementation of the face recognizer with deep neural networks described in FaceNet paper.

- https://github.com/davidsandberg/facenet

REFERENCES

https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c

https://trio.dev/blog/facial-recognition-applications

https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

https://foundationsofdl.com/2020/11/07/ssd300-implementation/

https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06

https://medium.com/sciforce/face-detection-explained-state-of-the-art-methods-and-best-tools-f730fca16294

https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c

https://arxiv.org/pdf/1811.00116.pdf - Face Recognition: From Traditional to Deep Learning Methods

https://arxiv.org/abs/1502.00046

REFERENCES

https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html

https://www.bytefish.de/blog/eigenfaces.html

https://www.bytefish.de/blog/fisherfaces.html

http://www.scholarpedia.org/article/Local_Binary_Patterns

Dlib models: https://github.com/davisking/dlib-models

http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

https://github.com/ageitgey/face_recognition

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

https://github.com/ageitgey/face_recognition/blob/master/face_recognition/api.py#L213