Bob's Biometric Recognition Framework

A Hands-on Tutorial with Face Recognition Examples

Dr. Manuel Günther

Vision and Security Technology
Department of Computer Science
University of Colorado Colorado Springs
United States of America – Colorado

http://vast.uccs.edu/~mgunther
mgunther@vast.uccs.edu

http://vast.uccs.edu/public-data/IJCB.html
http://www.idiap.ch/software/bob
http://www.idiap.ch/software/bob/docs/bob/bob.bio.base/stable

October 1, 2017 – IJCB in Denver

Vision And Security Technology
University of Colorado **Colorado Springs**

Biometric Recognition

What we will learn today

1. Which steps are required for biometric recognition?
2. How to run biometric recognition experiments using `bob.bio`?
   → Hands-On: Face recognition with eigenfaces
3. What are the advantages of GMM-based systems?
   → Hands-On: Face recognition with GMMs
4. How to implement your own biometric recognition algorithm?
   → Hands-On: Face recognition with DCNNs
5. How to provide reproducible research?

Contents

Advertisement Special Issue

Journal of *Imaging*

EMERGING SOURCES CITATION INDEX

Special Issue

Automatic Face Recognition

Guest Editor:

Dr.-Ing Manuel Günther
Vision and Security Technology Lab (VaST), University of Colorado Colorado Springs, P.O. Box 7150, 1420 Austin Bluffs Parkway, CO 80933-7150, USA

mgunther@vast.uccs.edu

Deadline for manuscript submissions:
31 March 2018

Message from the Guest Editor

Dear Colleagues,

Automatic face recognition has a long history in both industry and academia. While some automatic face systems work well in constraint environments with good illumination, high resolution images and cooperating subjects, face recognition in uncontrolled scenarios with unknown illumination, facial expression, face pose and/or occlusion still needs more research. Furthermore, it has been shown lately that open-set face recognition is very far from being deployable.

This special issue targets researchers in the area of automatic face recognition, including 2D and 3D algorithms. While new research in open-set face recognition would be favorable, any of the following topics are welcome:

- Open-set face recognition
- Closed-set face recognition
- Face recognition in the wild
- Face recognition at a distance
- Face clustering
- Face detection
- Facial landmark localization
- Face recognition with convolutional neural networks
- 3D algorithms for face recognition
- Face recognition datasets and evaluation protocols

Dr.-Ing. Manuel Günther

Author Benefits

Open Access: - free for readers, free publication for well-prepared manuscripts submitted in 2017.
High visibility: Indexed in the Emerging Sources Citation Index (ESCI - Web of Science), Inspec (IET) from Vol. 3 and in the DBLP Computer Science Bibliography.
Rapid publication: manuscripts are peer-reviewed and a first decision provided to authors approximately 36 days after submission; acceptance to publication is undertaken in 7 days (median values for papers published in this journal in first half of 2017).

MDPI
www.mdpi.com

J. Imaging Editorial Office, jimaging@mdpi.com
St. Alban-Anlage 66, 4052 Basel, Switzerland
Tel. +41 61 683 77 34, Fax: +41 61 302 89 18

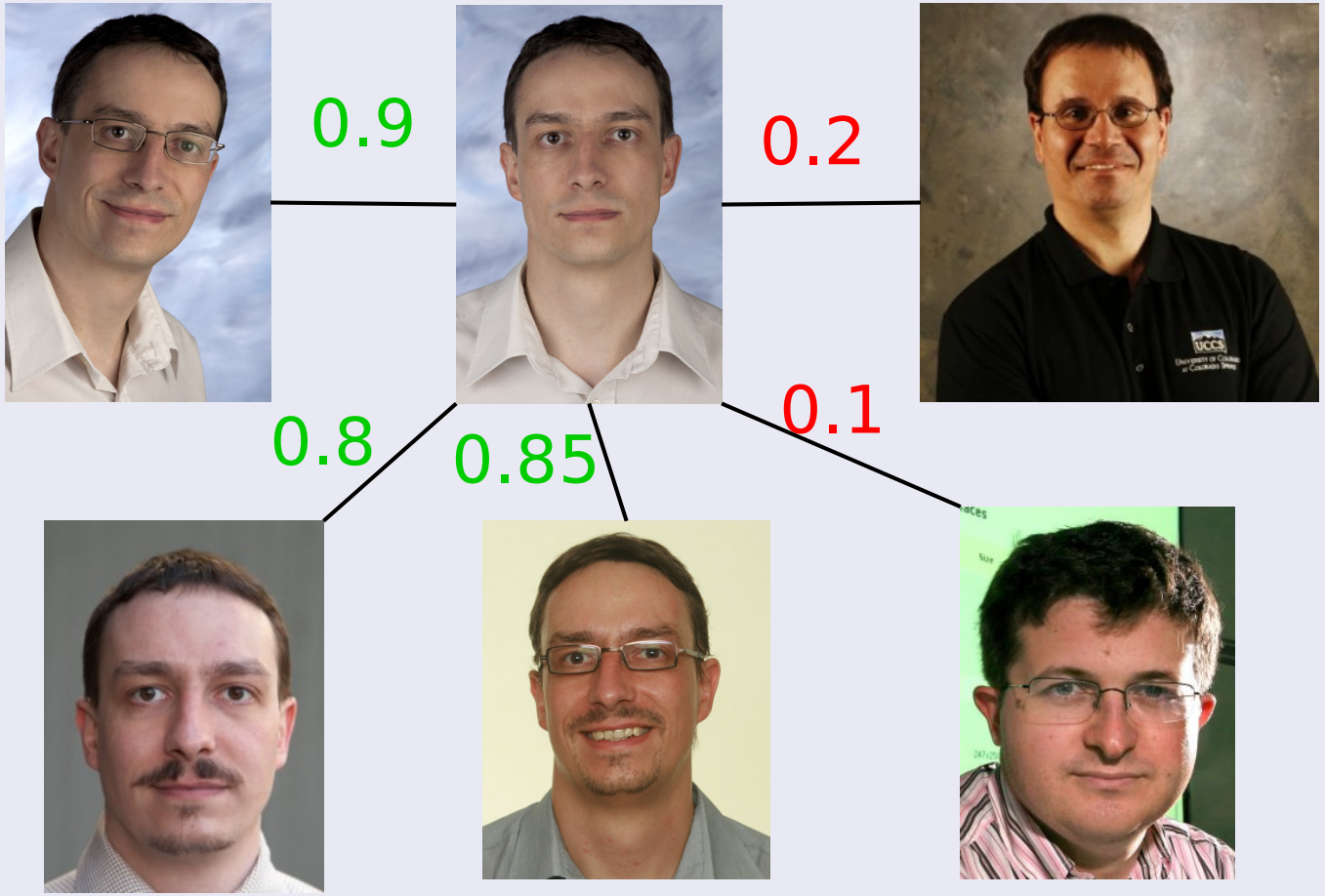
mdpi.com/si/11394

http://www.mdpi.com/journal/jimaging/special_issues/face_recognition

Outline

Goal of Biometric Recognition

Similarity Score for Pair of Samples



Biometric Recognition

Applications of Biometric Recognition

- Authentication: Unlock your phone
- Verification: Border control
- Identification: Who is in the data sample?
- Watch List: Unwanted persons
- Search: Give me all samples with XXX
- Clustering: Group samples of subjects
- Forensics: Who committed the crime?

Authentication and Verification

Token-based 1:1 Comparison

- Enroll a **model** (token) from one or more samples
- Take one or several **probe** sample(s)
- Compute **similarity** between model and probe(s)
- **Threshold** similarity to **accept** or **reject**

Characteristics

- Environment is **controlled**
- Subjects **cooperate** with the system

Identification

Gallery-based 1:N Identification

- Enroll models of different **clients** into a **gallery**
- Take one or several **probe** image(s)
- Compute **similarities** between models and the probe(s)
- Return one or several **most similar** clients

Variations

- Closed-set: probe always has a match in the gallery
- Open-set: probe might not be in gallery
  - $\Rightarrow$ Threshold similarities

Watch List

Gallery-based M:N Identification

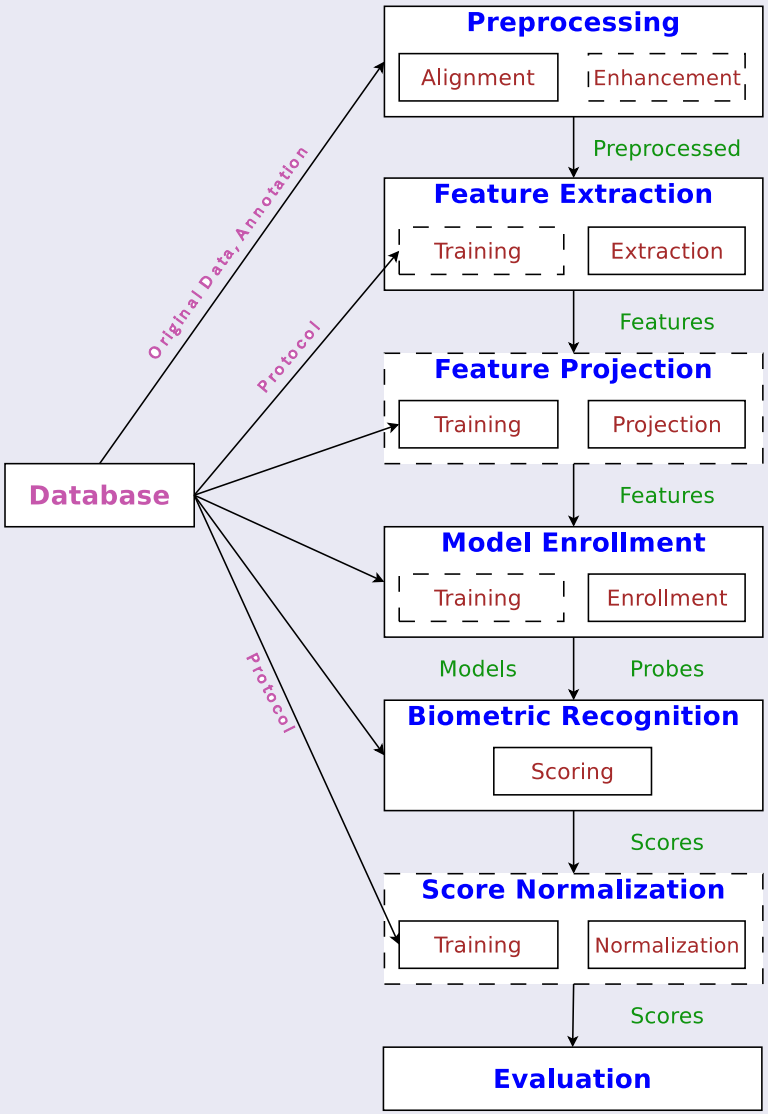
- Enroll models of different clients into a **watch list**
- **Detect** subjects in probe sample
- Compute **similarities** between models and all probes
- Decide, **which subjects** are on the watch list

Characteristics

- **Uncontrolled** samples with **uncooperative** subjects
- Many more **unknown** than **known** subjects
- Must handle **false alarms** of the detector

Biometric Recognition

Tool Chain

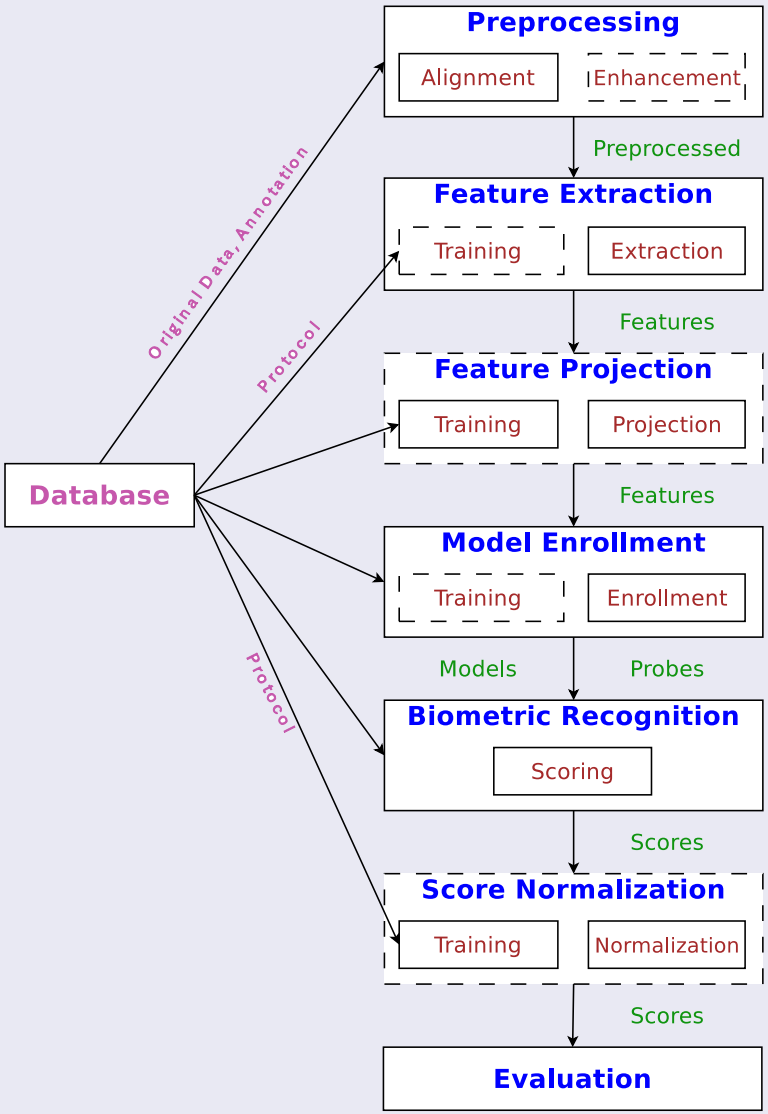


Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Biometric Recognition

Tool Chain

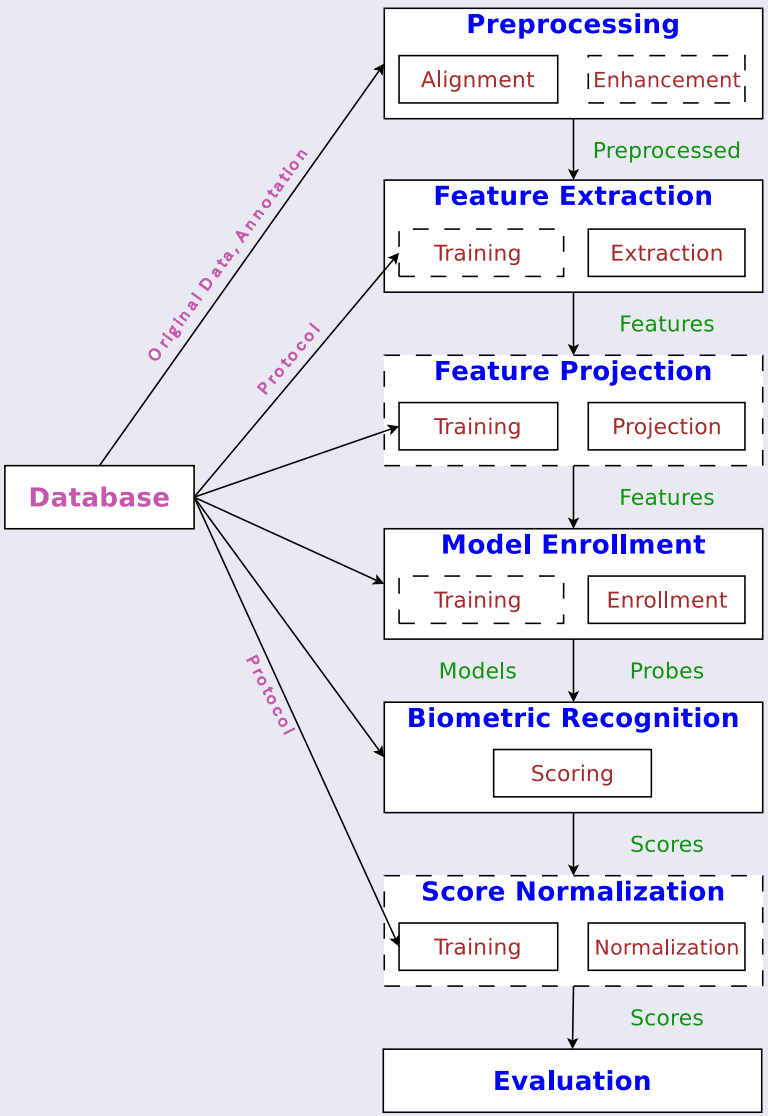


Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Database Interfaces

Tool Chain

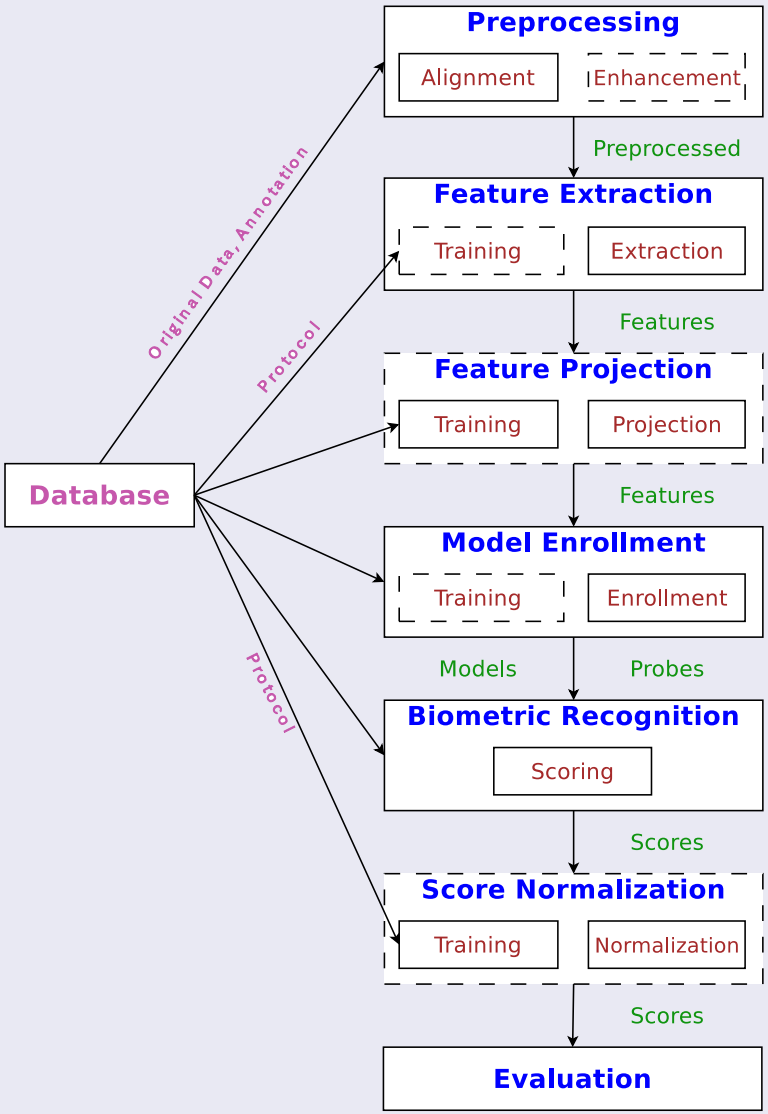


Database

- Provides data
  - → Face image, video
  - → Voice sample, fingerprint
  - → (Annotations)
- Protocol
  - → Training set
  - → Enrollment data
  - → Probe data (per model)
  - → Score normalization
- Protocol Types
  - → Biased / Unbiased
  - → Lausanne (development and evaluation set)

Biometric Recognition

Tool Chain



Steps

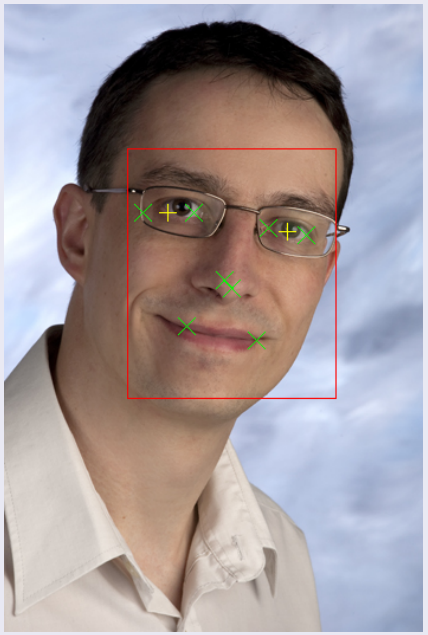
- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Face Alignment

Sources

- Face detector
  - → bounding box
- Landmark detector
  - → eyes
- Hand-labeled annotations
  - → eyes

Example



Photometric Enhancement

Color

Hist. Eq.

Tan&Triggs

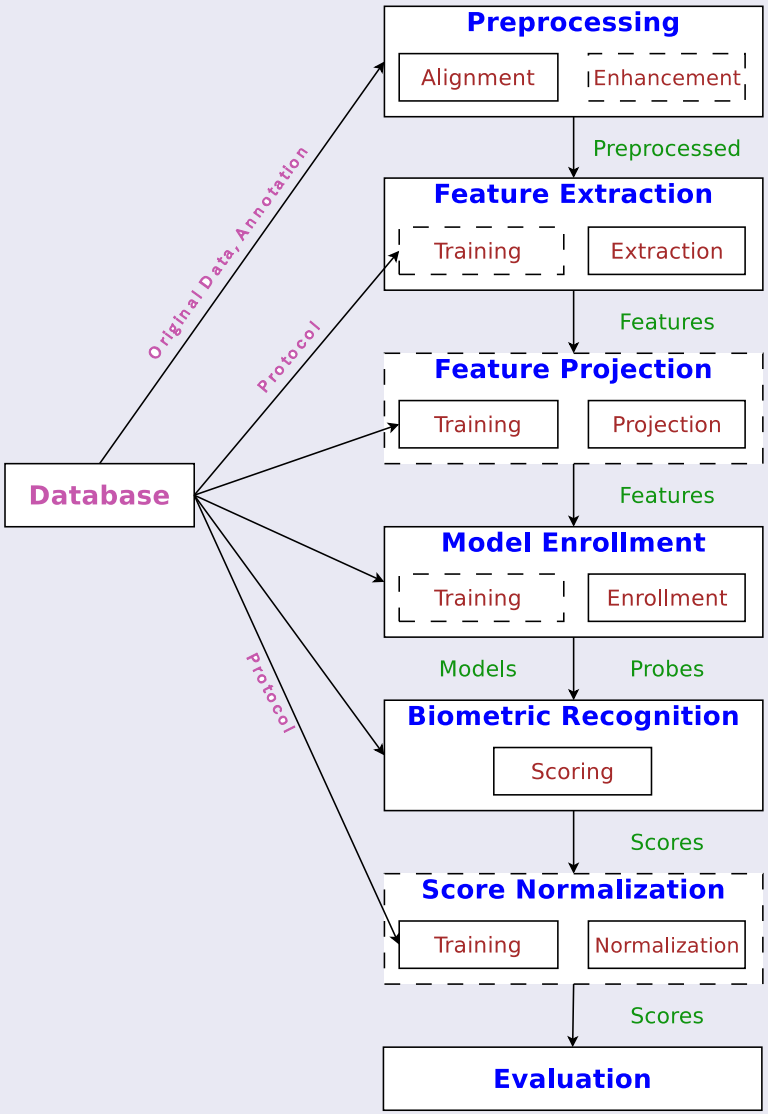
Grayscale

Self-Quot.

LBP

Biometric Recognition

Tool Chain

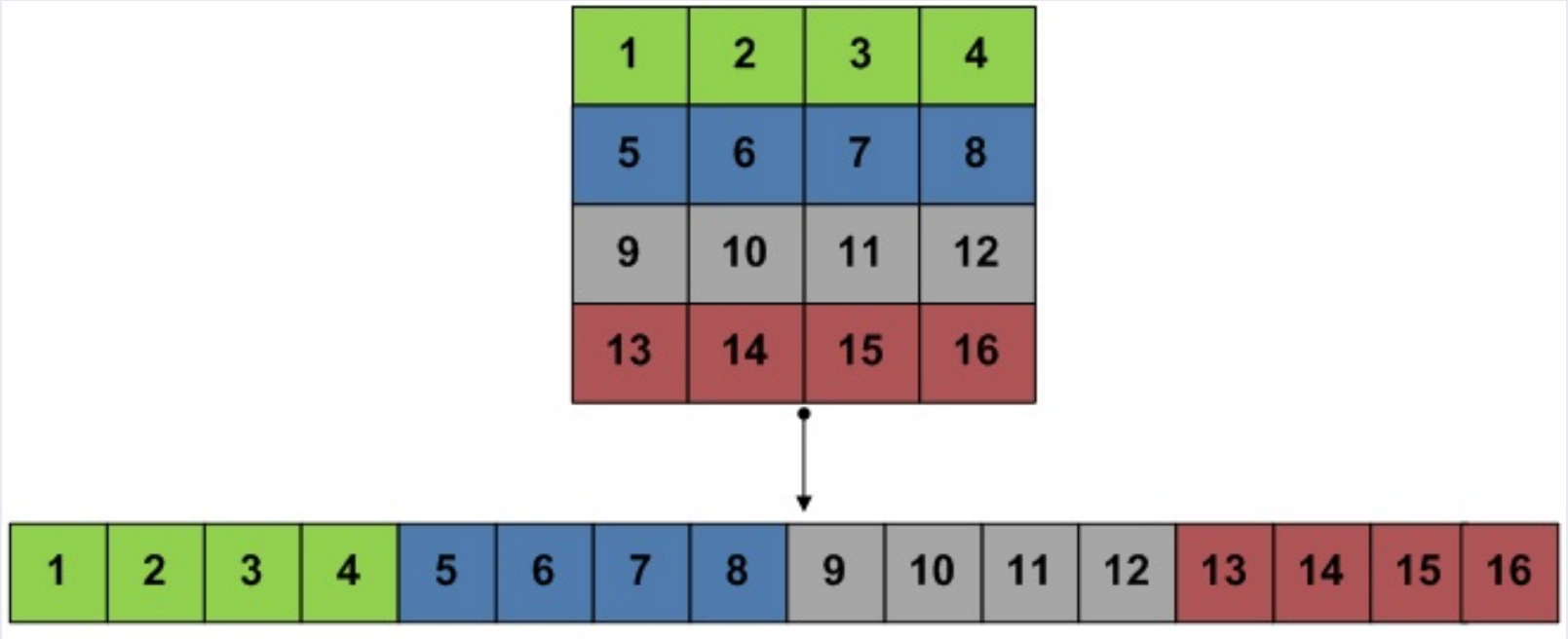


Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Global Features

Concatenate Pixels (link)



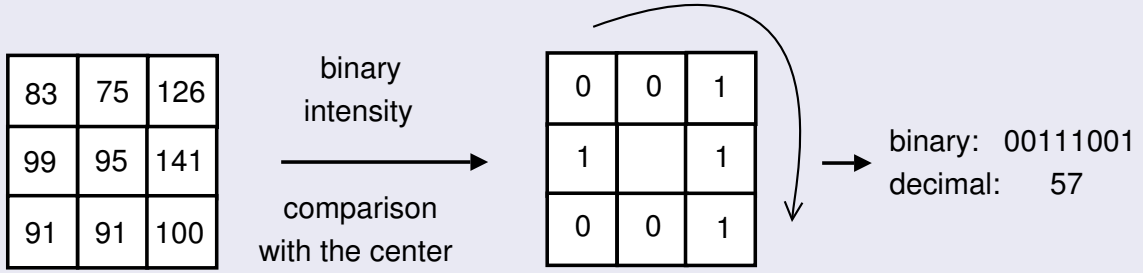
Global Features

LBP Equation

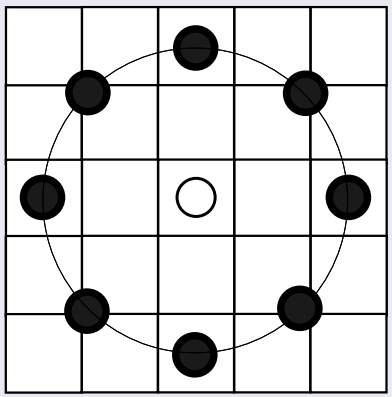
$$\mathrm{LBP} = \sum_{i=0}^{7} s(p_i - p_c)\, 2^i$$

$$s(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{else} \end{cases}$$

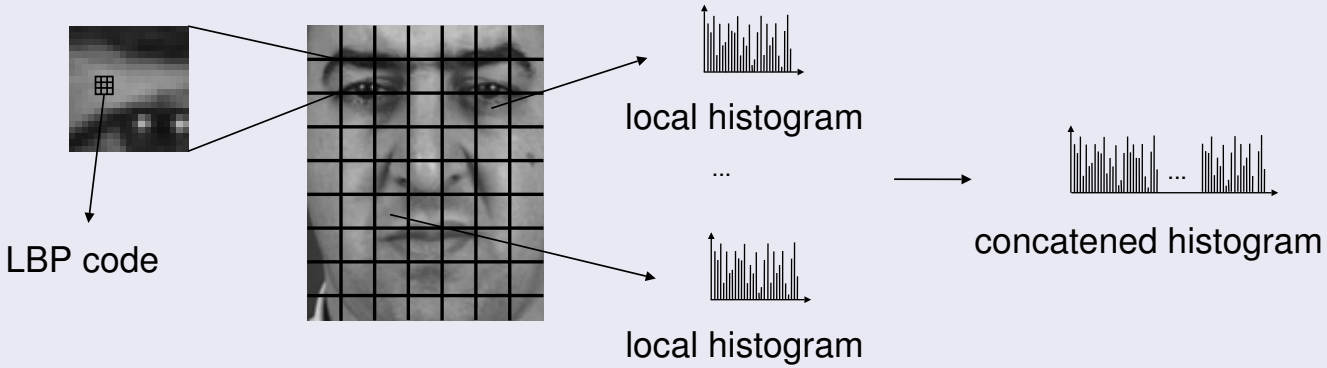
Binary Pattern Example



Round LBP

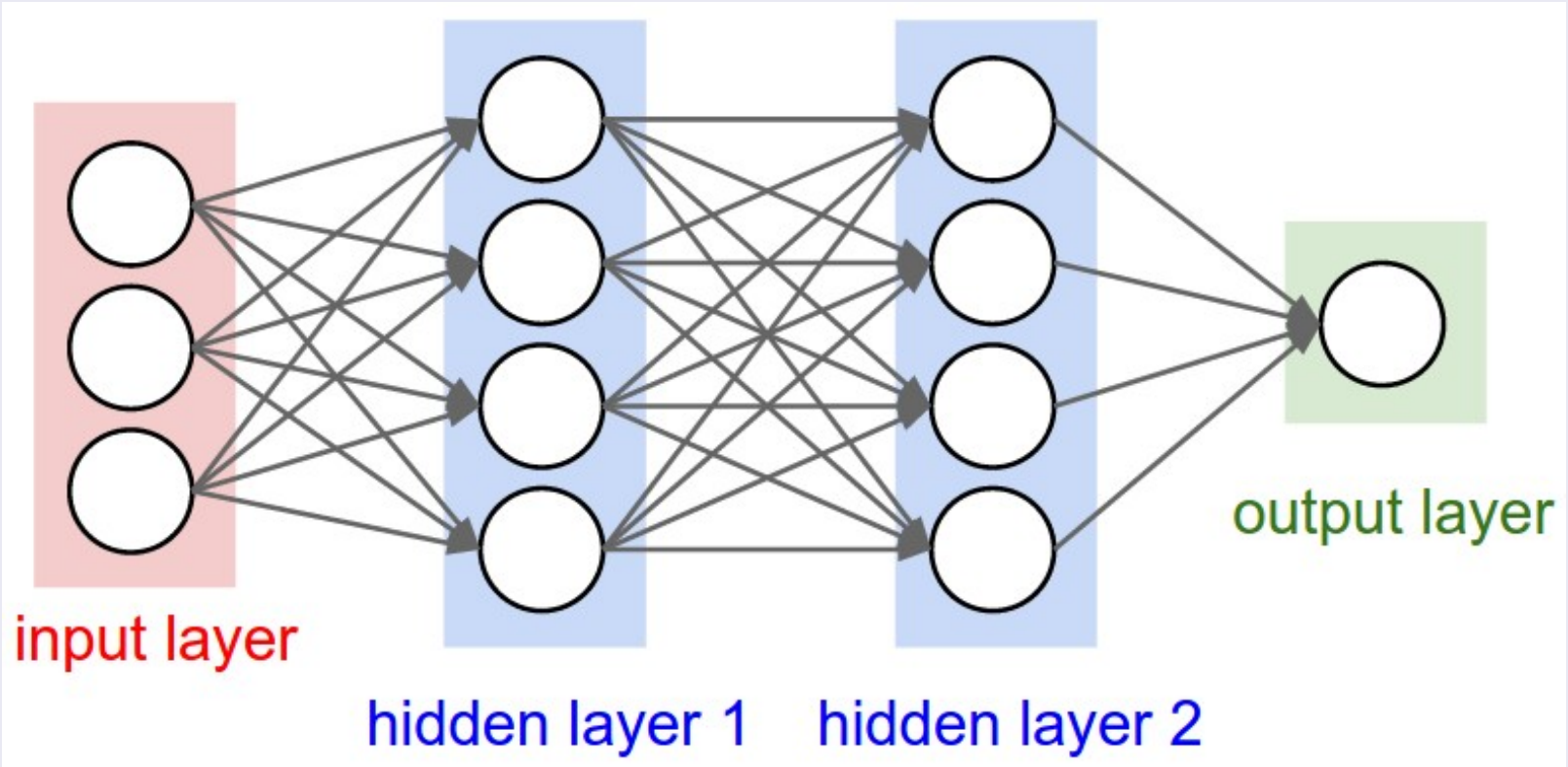


LBP Histogram Sequences



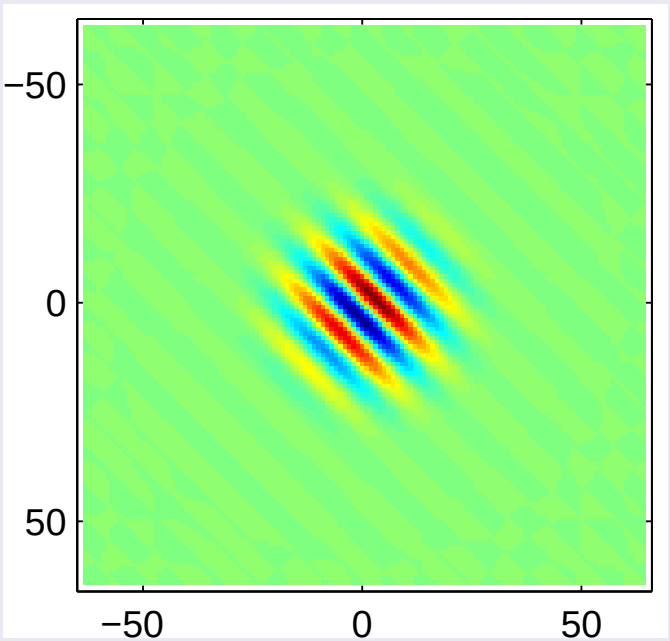
Global Features

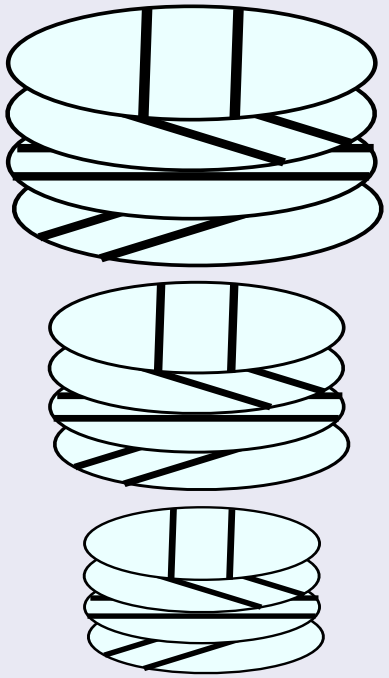
Neural Network (link)

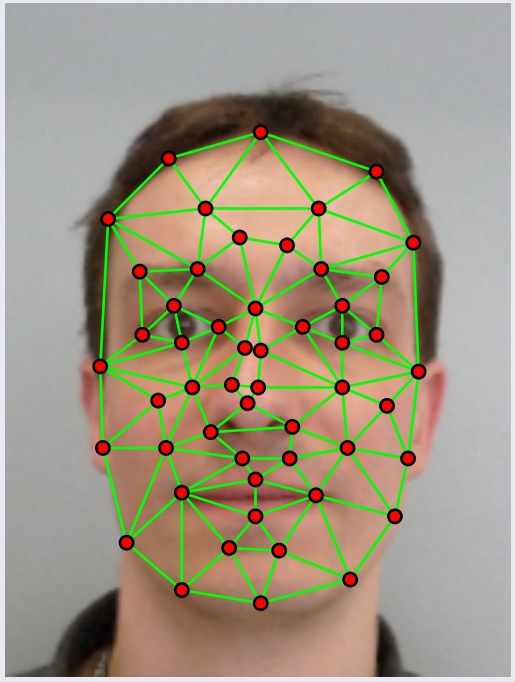


Local Features

Gabor Graphs







Gabor Wavelet

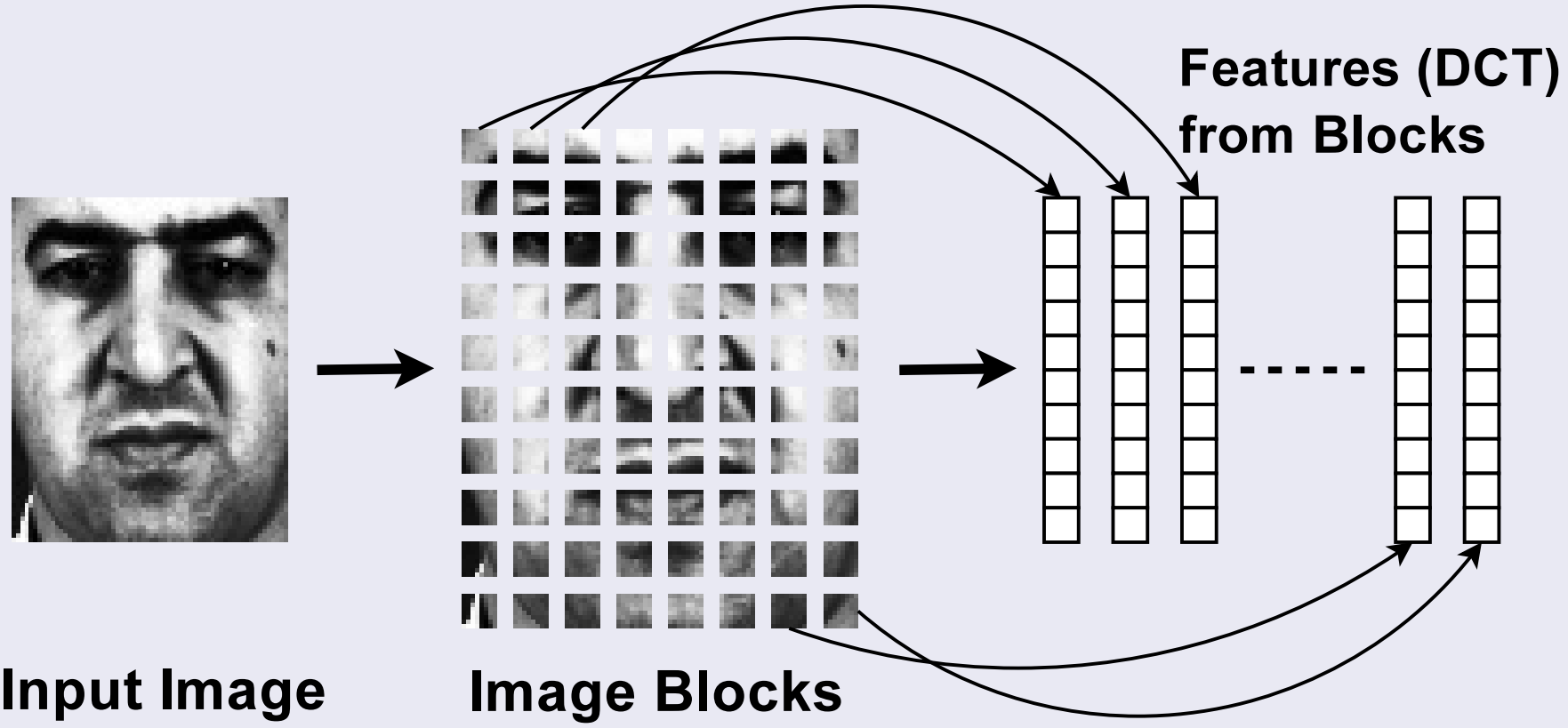
$$\psi_{\vec{k}_j}(\vec{t}) = \frac{\vec{k}_j^2}{\sigma^2} e^{-\frac{\vec{k}_j^2 \vec{t}^2}{2\sigma^2}} \left[e^{i \vec{k}_j^T \vec{t}} - e^{-\frac{\sigma^2}{2}} \right]$$

idiap RESEARCH INSTITUTE

Security Technology
olorado Colorado Springs

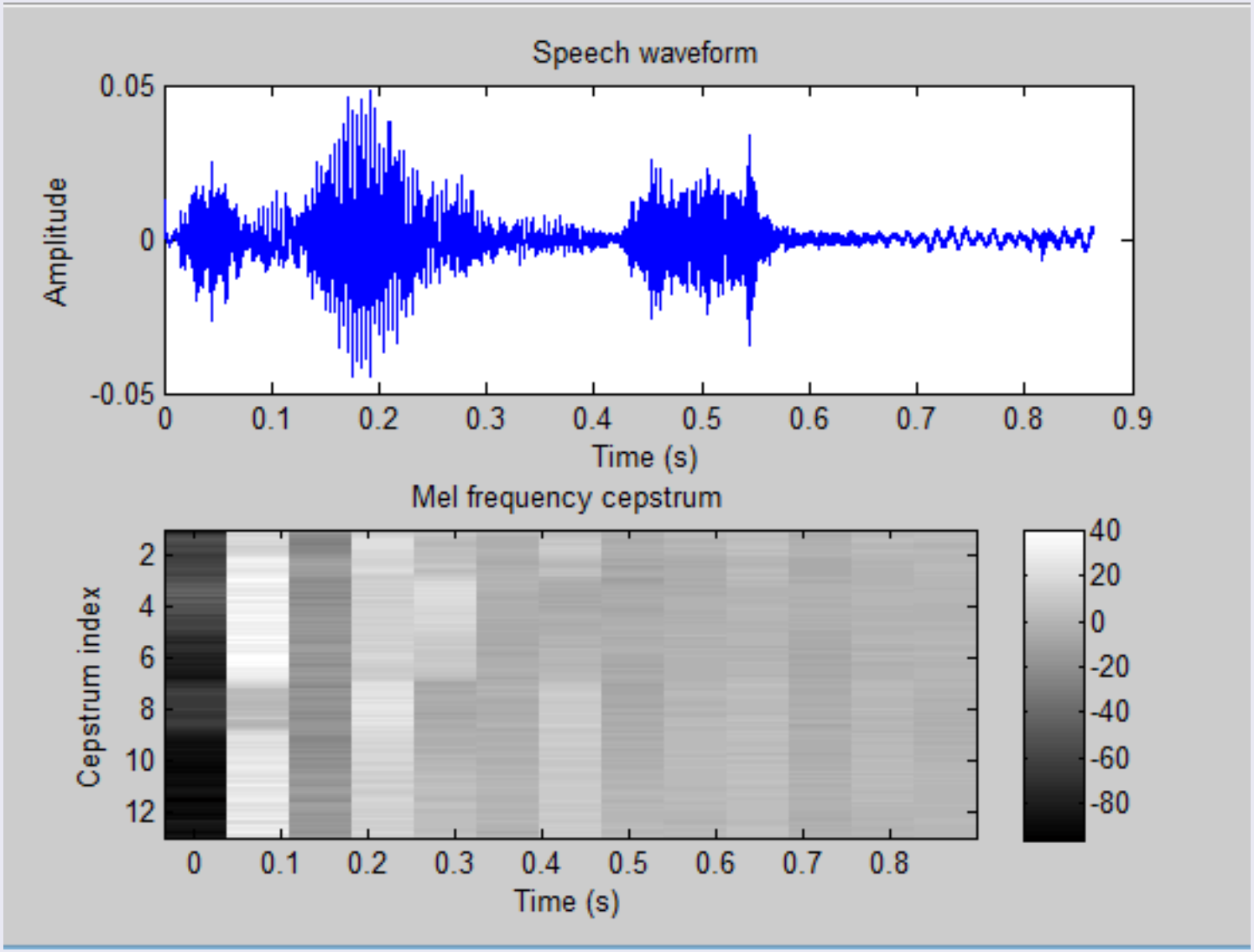
Local Features

DCT Blocks



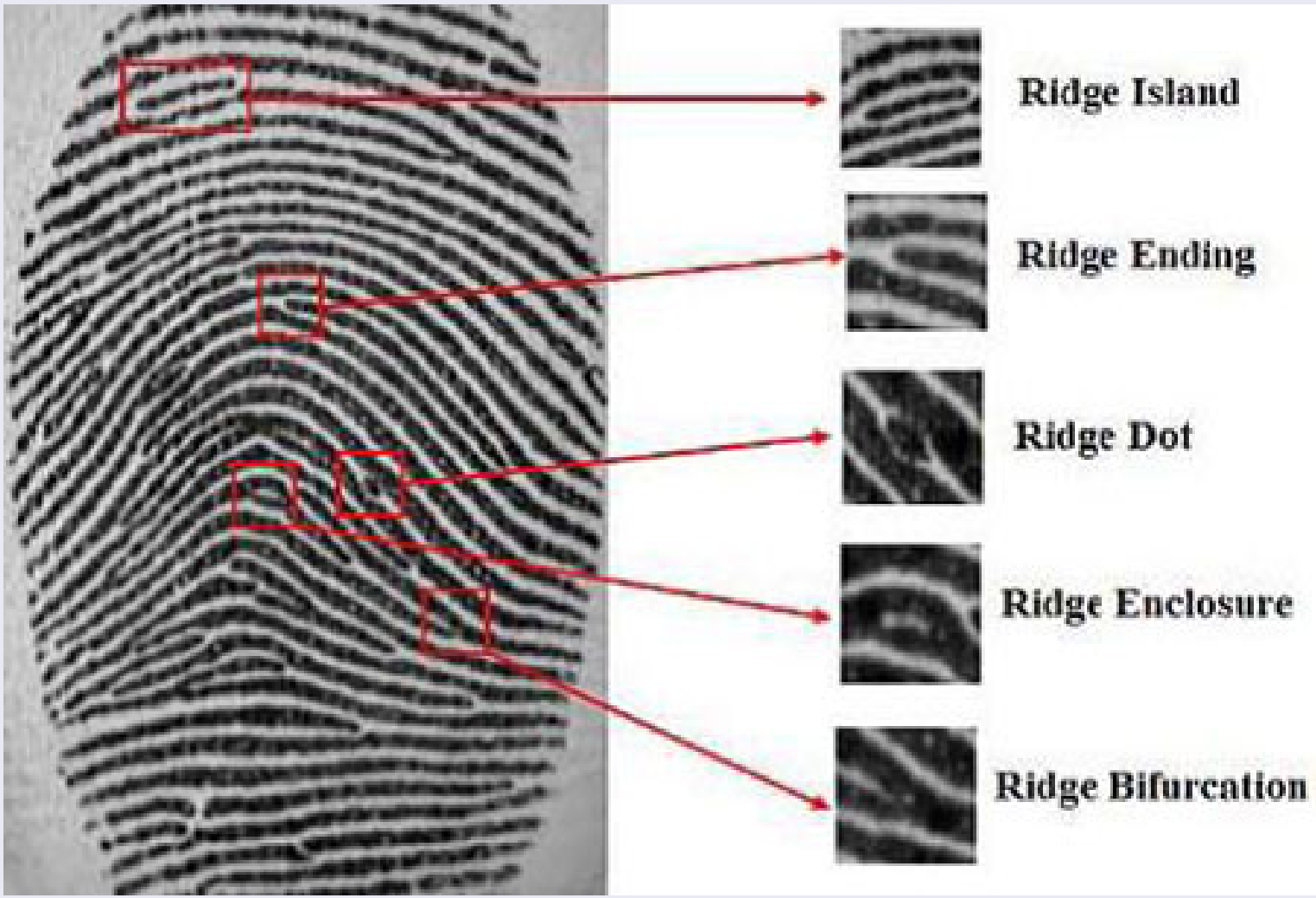
Local Features

Mel-Frequency Cepstral Coeff. (MFCC) (link)



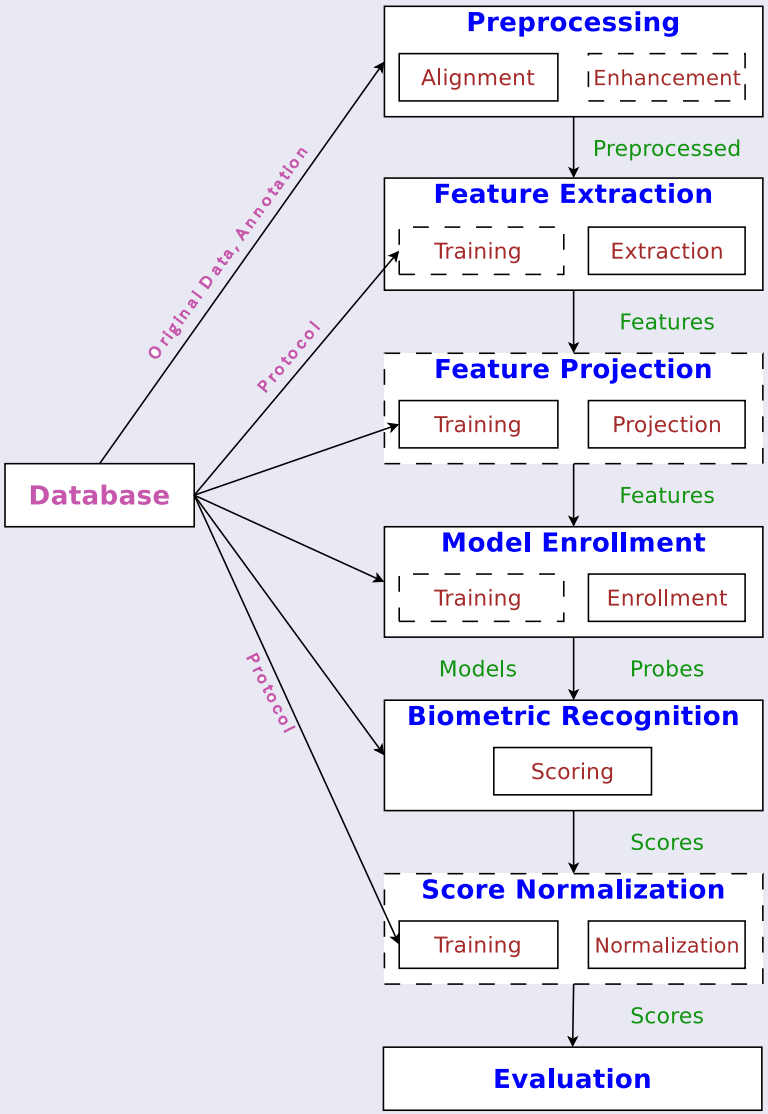
Local Features

Fingerprint Minutiae (link)



Biometric Recognition

Tool Chain



Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Feature Projection

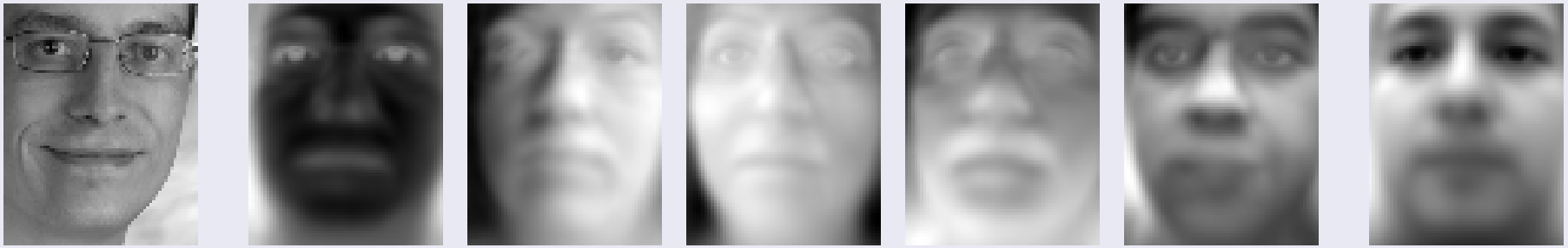
Linear Subspaces

$$\vec{z} = \mathbf{W}(\vec{x} - \vec{\mu})$$

with $\vec{x}, \vec{\mu} \in \mathbb{R}^D$, $\vec{z} \in \mathbb{R}^M$ and $\mathbf{W} \in \mathbb{R}^{M \times D}$

Dimensionality Reduction to 5 Dimensions

$\vec{x}$ $\vec{w}_1$ $\vec{w}_2$ $\vec{w}_3$ $\vec{w}_4$ $\vec{w}_5$ $\vec{\mu}$



$$\vec{z} = (-6.4, \quad -0.9, \quad -6.8, \quad +0.9, \quad +2.4) \quad (1)$$

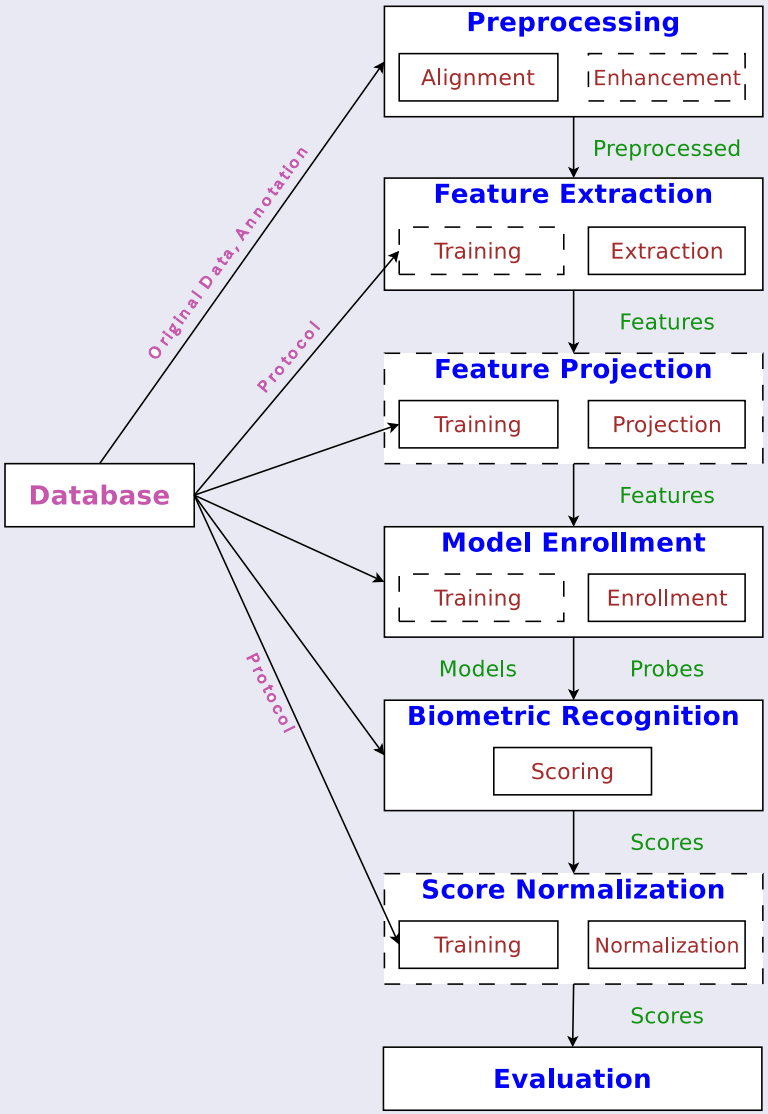
Feature Projection Examples

Eigenfaces from PCA

Fisherfaces from PCA+LDA

Biometric Recognition

Tool Chain



Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Model Enrollment

Idea

Given one or more features of the same person (**client**, subject, identity),
combine them into a representation of the person (**model**, template, token)

Simple Solutions

- Store all features
- Compute average feature

Complex Solutions

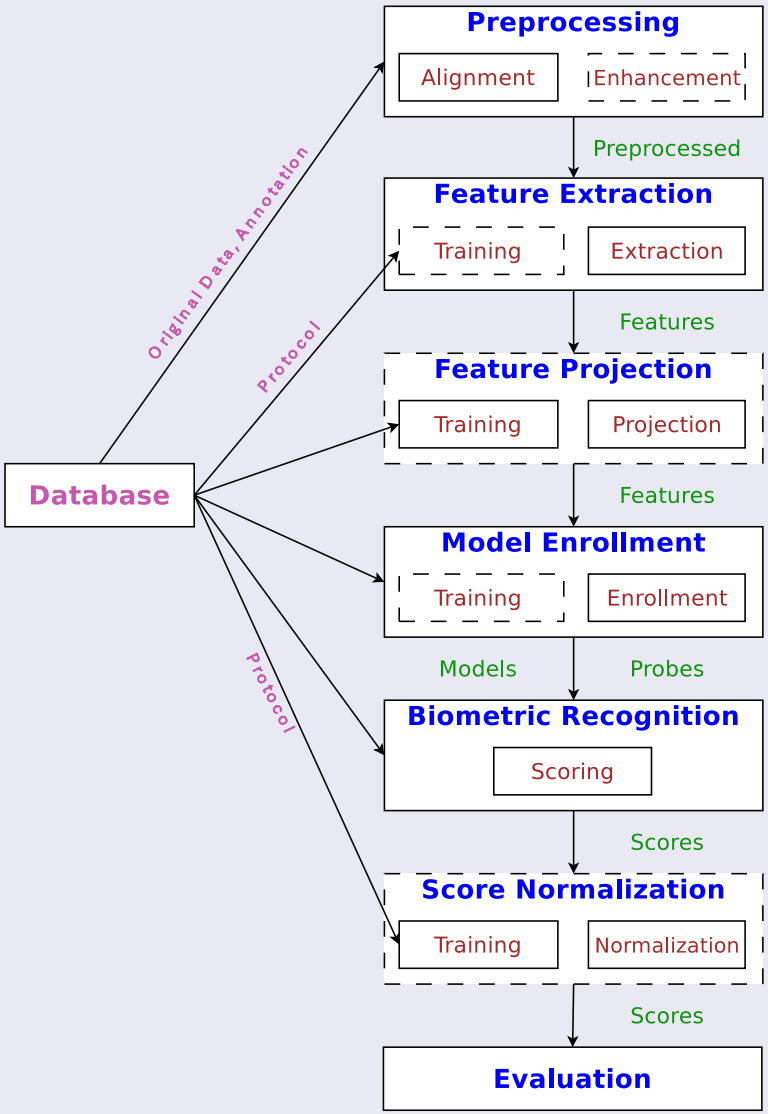
- Train classifier
  → SVM
- MAP adaptation
  → GMM

idiap RESEARCH INSTITUTE

University of Colorado **Colorado Springs**

Biometric Recognition

Tool Chain



Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Probing/Scoring

Idea

Given one **model** and one **probe** feature, compute a similarity **score**

Feature Comparisons

- Cosine similarity
- Histogram similarity
- Gabor Jet similarity

Complex Solutions

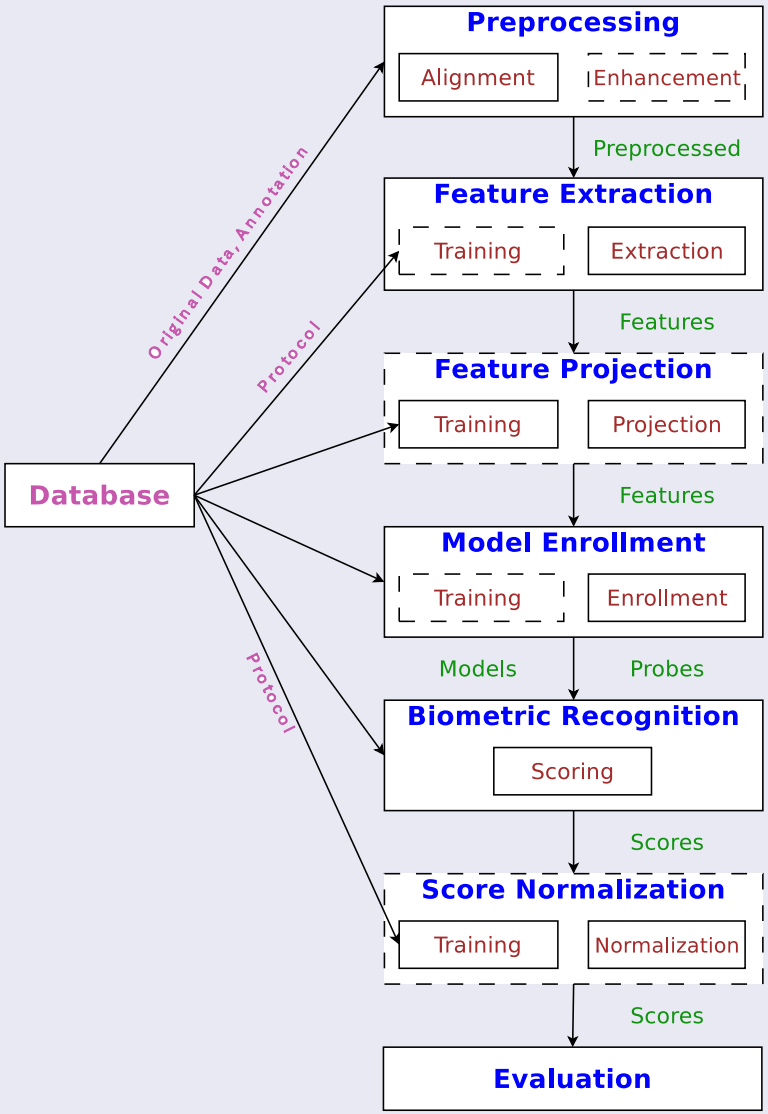
- Classification result
  - → SVM
- Log-likelihood
  - → GMM

Variation

Several probe features for **one** comparison

Biometric Recognition

Tool Chain



Steps

- Database
  - → Assures comparability
- Preprocessing
  - → Detection
  - → Alignment
  - → Enhancement
- Features
  - → Local vs. global
  - → Feature projection
- Algorithms
  - → Model enrollment
  - → Scoring
- Evaluation

Closed-Set Identification

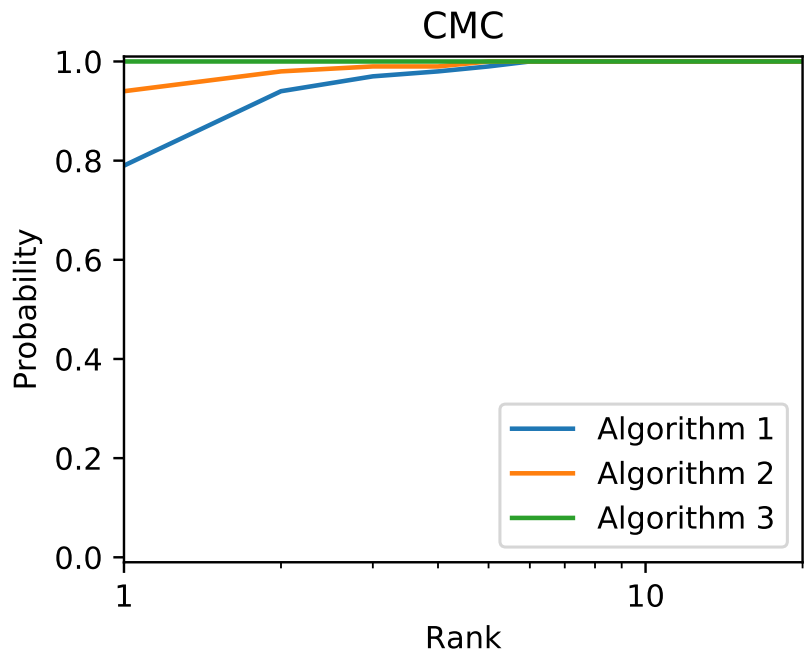
Rank rk for Probe $f \in P$

$$\mathrm{rk}(f) = \Big|\{m \mid \mathrm{s}(m,f) \geq \mathrm{s}(m^*,f), m \in G\}\Big|$$

Rank r Recognition Rate

$$\mathrm{RR}(r) = \frac{\Big|\{f \mid \mathrm{rk}(f) \leq r\}\Big|}{|P|}$$

Cumulative Match Char.



idiap RESEARCH INSTITUTE

bob

UCCS Vision And Security Technology
University of Colorado Colorado Springs

Open-Set Identification – Watch List

Detection & Identification

Known probes P_K:

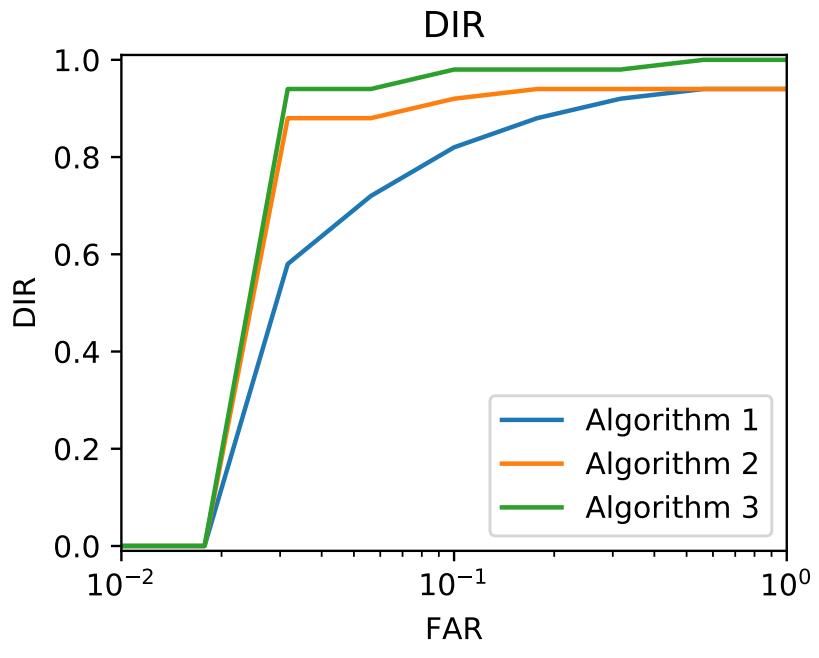
$$\mathrm{DIR}_r(\theta) = \frac{\left|\{f \mid \mathrm{rk}(f) \leq r \wedge \mathrm{s}(m^*, f) \geq \theta\}\right|}{|P_K|}$$

False Alarm Rate

Unknown probes P_U:

$$\mathrm{FAR}(\theta) = \frac{\left|\{f \mid \max_{m \in G} \mathrm{s}(m, f) > \theta\}\right|}{|P_U|}$$

Detection & Identification



idiap RESEARCH INSTITUTE

UCCS Vision And Security Technology University of Colorado Colorado Springs

Verification – Single Test Set

False Non-Match Rate

Genuine scores s_g:

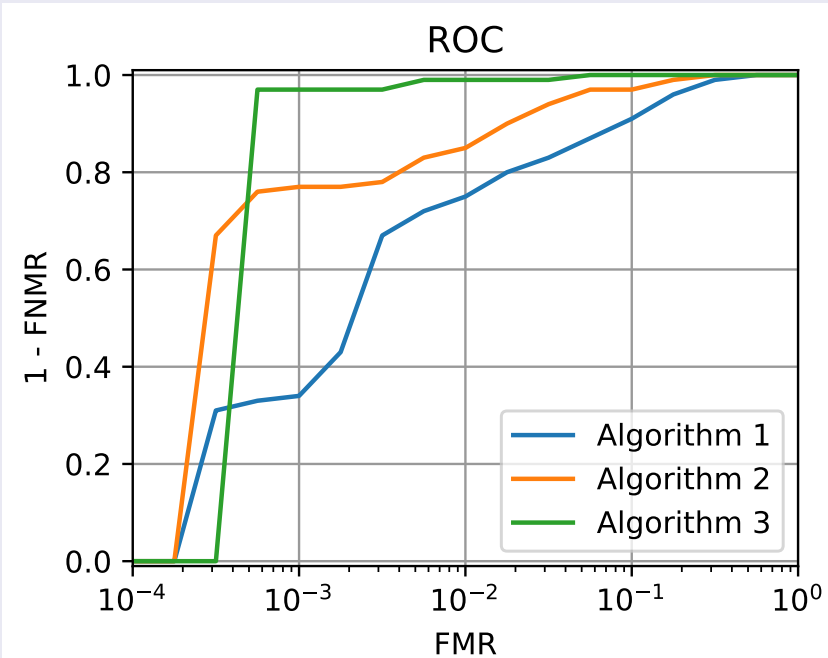
$$\mathrm{FNMR}(\theta) = \frac{|\{s_g | s_g < \theta\}|}{|\{s_g\}|}$$

False Match Rate

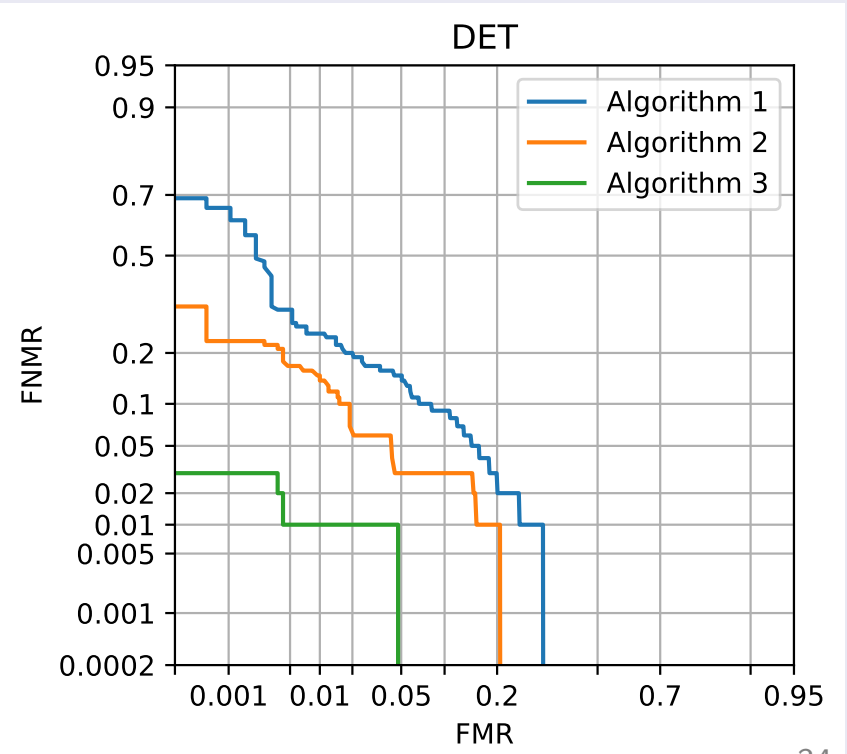
Impostor scores s_i:

$$\mathrm{FMR}(\theta) = \frac{|\{s_i | s_i \geq \theta\}|}{|\{s_i\}|}$$

Receiver Operating Char.



Detection Error Tradeoff



Verification – Lausanne Protocol

Expected Performance

$$\theta^* = \arg\min_{\theta} \alpha \cdot \mathrm{FMR}_{dev}(\theta) + (1-\alpha) \cdot \mathrm{FNMR}_{dev}(\theta)$$

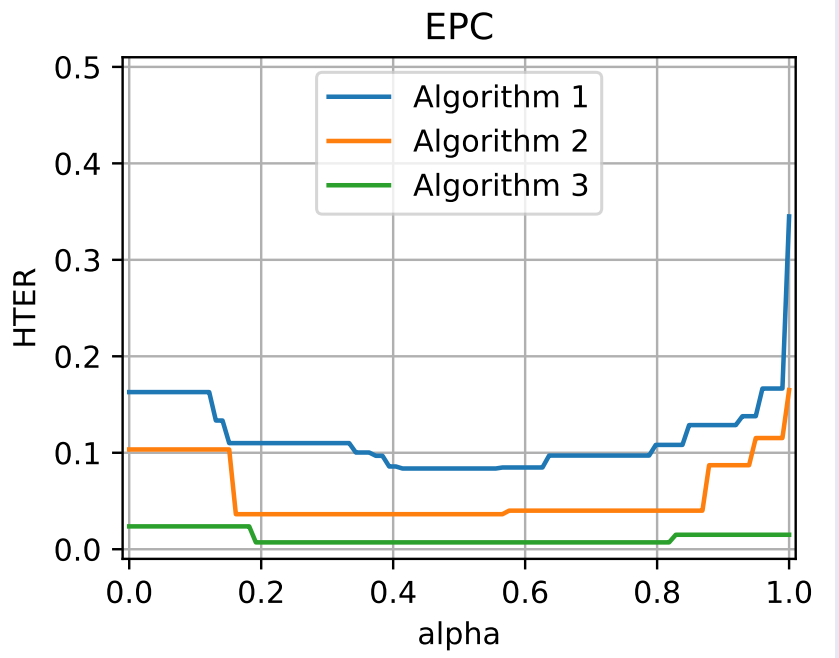
Equal Error Rate

$$\mathrm{EER} = \frac{\mathrm{FMR}_{dev}(\theta^*) + \mathrm{FNMR}_{dev}(\theta^*)}{2}$$

Half Total Error Rate

$$\mathrm{HTER} = \frac{\mathrm{FMR}_{eval}(\theta^*) + \mathrm{FNMR}_{eval}(\theta^*)}{2}$$

Expected Performance Curve



Outline

Open Source Biometric Recognition

OpenBR: Application-Oriented

- C++ only (thin Python wrapper)
- Only a few algorithms
- (Relatively) complicated command line

CSU Face Recognition Resources: Example-Oriented

- Python only $\Rightarrow$ slow
- Two face recognition algorithms, one dataset
- Not extensible

BEAT Platform Comparison-Oriented

- Evaluates algorithms online
- Not designed for algorithm development

Bob

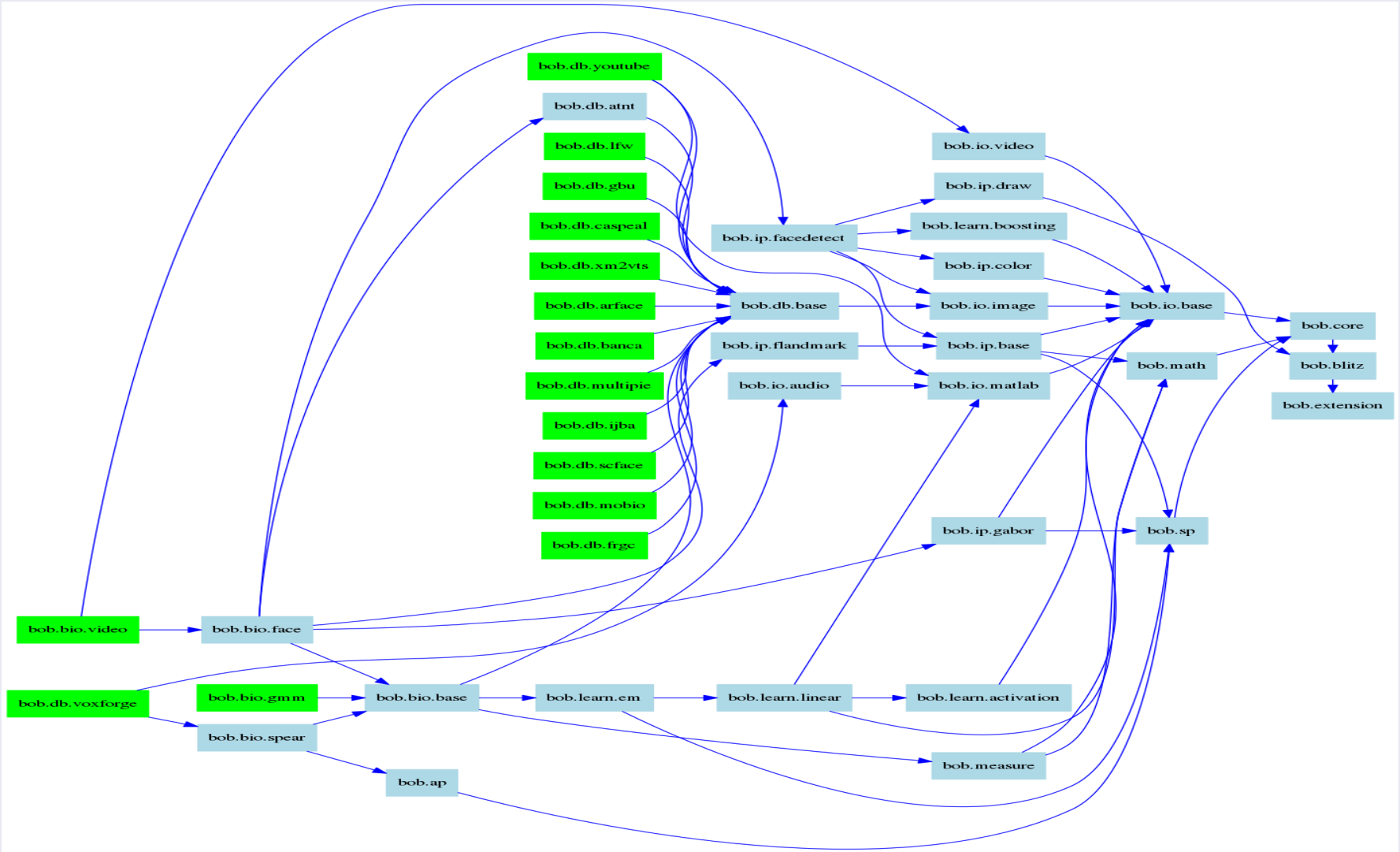
Signal-Processing and Machine Learning toolbox

http://www.idiap.ch/software/bob

- Mixture of C++ (Blitz++) and Python (NumPy)
- Integrates well into NumPy, SciPy, scikit-learn, ...
- Large signal and image processing library
- Machine learning (Machines and Trainers) $\Rightarrow$ GMM
- Image, video and audio I/O (most formats)
- Binary I/O with HDF5
- Simple interface for running biometric experiments
- Unified interfaces for biometric databases

Bob

Bob Package Dependencies



Bob's Biometric Recognition Tools

Bob's Biometrics Packages: Research-Oriented

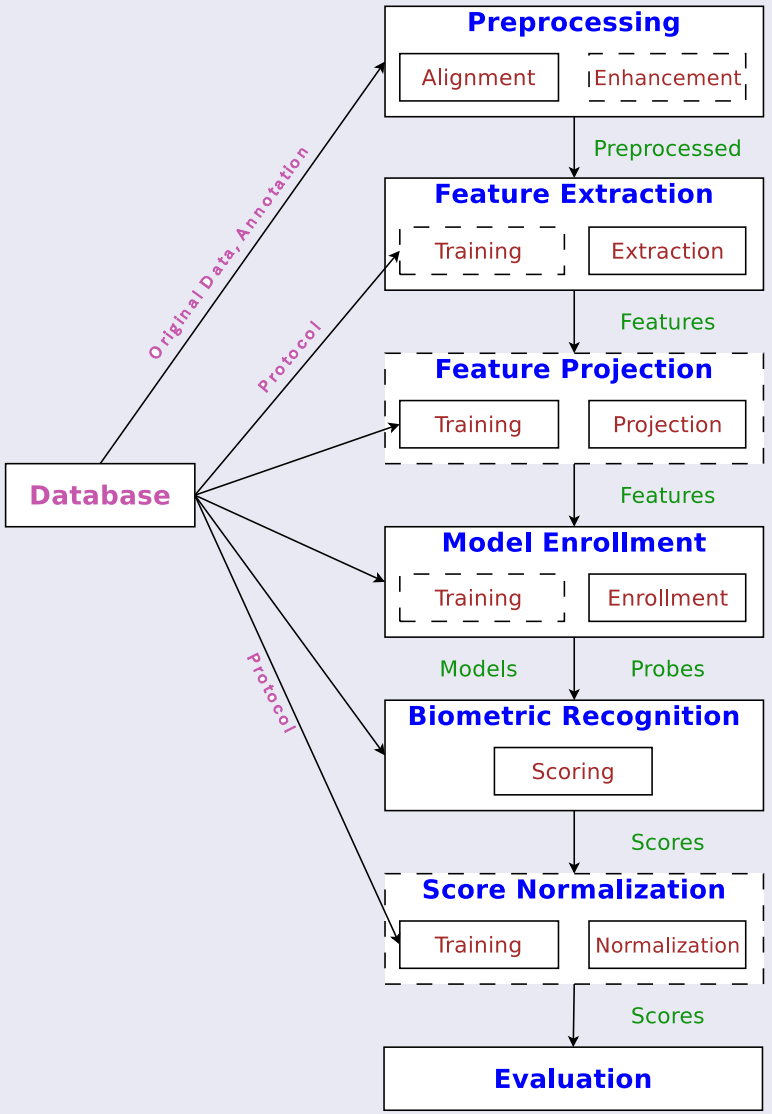
- Designed to be modular, extensible and configurable
- Supports different biometrics (face, speech, ...)
- Enables more than 10 different biometric databases
- Implements several data preprocessors
- Implements several types of features
- Implements several different biometric algorithms
- Handles several features for enrollment and probing

Characteristics

- Python interface to C++ implementations
- Parallelism by design (through GridTK)

Bob's Biometric Recognition Tools

Tool Chain



Steps File-based

- Database (`database`)
  - → Assures comparability
- Preproc. (`preprocessor`)
  - → Face image, video
  - → Voice sample, fingerprint
  - → (Annotations)
- Features (`extractor`)
  - → Local vs. global
- Algorithms (`algorithm`)
  - → Feature projection
  - → Model enrollment
  - → Scoring
- Evaluation (`evaluate.py`)

Bob's Biometric Recognition Tools

Database

```
class bob.bio.base.database.BioFile:
    __init__(id, client_id, path)

class bob.bio.base.database.BioDatabase:
    __init__(original_directory, protocol, ...)

    all_files() -> [BioFile]
    training_files() -> [BioFile]
    model_ids() -> [model_id]
    enroll_files(model_id) -> [BioFile]
    probe_files(model_id) -> [BioFile]
    client_id_from_model_id(model_id) -> client_id

    original_file_names([BioFile]) -> [filename]
    annotations(BioFile) -> annotations
```

Bob's Biometric Recognition Tools

Preprocessor

```
class bob.bio.base.preprocessor.Preprocessor:
    __init__(parameters)

    read_original_data(BioFile, dir, ext) -> data

    __call__(data, annotations) -> preprocessed
    write_data(preprocessed, filename)
    read_data(filename) -> preprocessed
```

Bob's Biometric Recognition Tools

Implemented Preprocessors

```
bob.bio.face.preprocessor.Base
bob.bio.face.preprocessor.FaceCrop
bob.bio.face.preprocessor.FaceDetect

bob.bio.face.preprocessor.HistogramEqualization
bob.bio.face.preprocessor.SelfQuotientImage
bob.bio.face.preprocessor.TanTriggs
bob.bio.face.preprocessor.INormLBP

bob.bio.spear.preprocessor.Mod_4Hz
bob.bio.spear.preprocessor.Energy_2Gauss
bob.bio.spear.preprocessor.Energy_Thr
```

Photometric Enhancement

FaceCrop

HistogramEquali

TanTriggs

FaceCrop

SelfQuotientImag

INormLBP

Bob's Biometric Recognition Tools

Extractor

```
class bob.bio.base.extractor.Extractor:
    __init__(parameters)

    train(training_features, extractor_file)
    load(extractor_file)

    __call__(preprocessed) -> feature
    write_feature(feature, filename)
    read_feature(filename) -> feature
```

Bob's Biometric Recognition Tools

Implemented Extractors

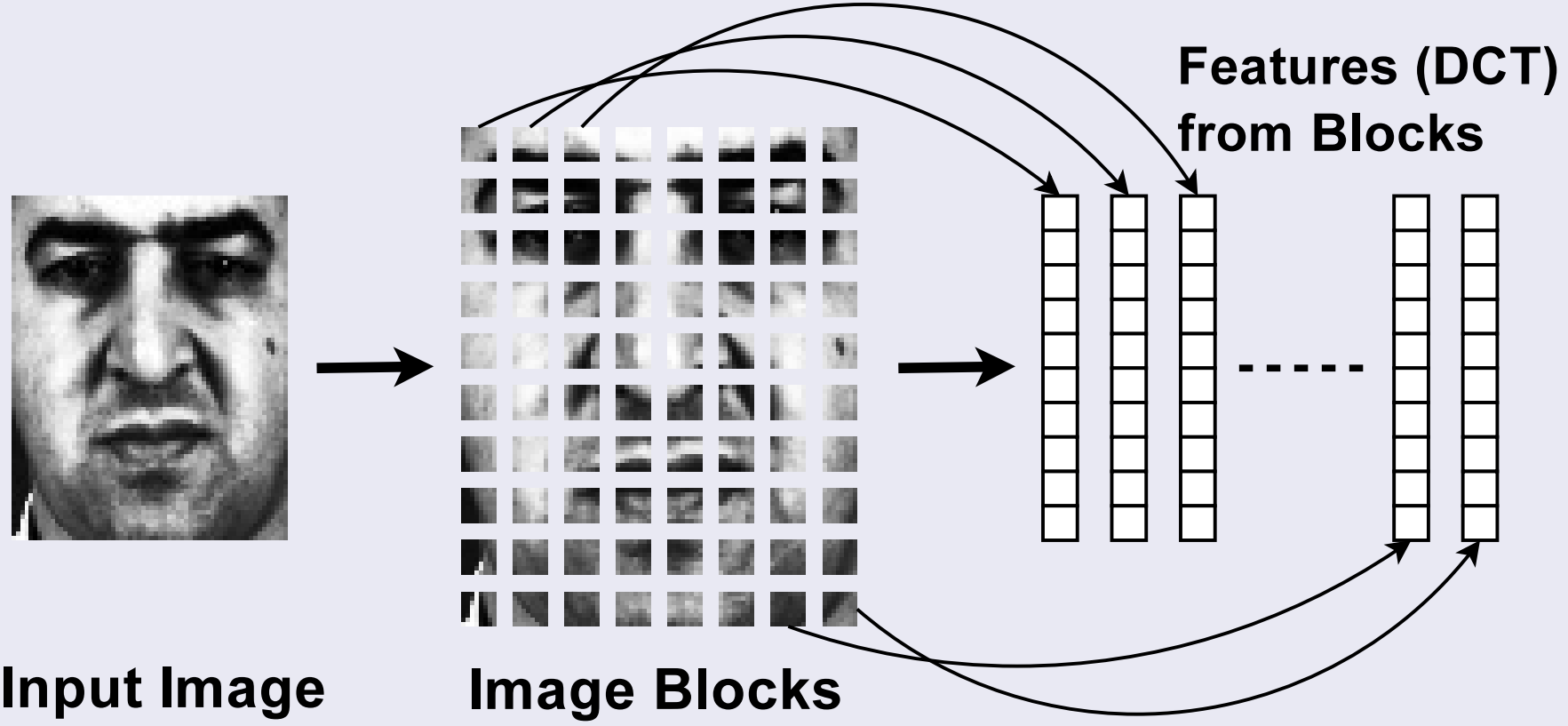
```
bob.bio.base.extractor.Linearize

bob.bio.face.extractor.GridGraph
bob.bio.face.extractor.LGBPHS
bob.bio.face.extractor.DCTBlocks
bob.bio.face.extractor.Eigenface

bob.bio.spear.extractor.Cepstral
bob.bio.spear.extractor.HTKFeatures
bob.bio.spear.extractor.SPROFeatures
```

Local Features

DCT Blocks



Bob's Biometric Recognition Tools

Algorithm

```
class bob.bio.base.algorithm.Algorithm:
    __init__(parameters)

    train_projector(training_features, projector_file)
    load_projector(projector_file)
    project(feature) -> projected
    write_feature(projected, filename)
    read_feature(filename) -> projected

    train_enroller(training_features, enroller_file)
    load_enroller(enroller_file)
    enroll(features) -> model
    write_model(model, model_file)
    read_model(model_file) -> model

    score(model, probe) -> similarity
    score_for_multiple_probes(model, probes) -> similarity
```

Bob's Biometric Recognition Tools

Implemented Algorithms

```
bob.bio.base.algorithm.Distance
bob.bio.base.algorithm.PCA
bob.bio.base.algorithm.LDA
bob.bio.base.algorithm.PLDA
bob.bio.base.algorithm.BIC

bob.bio.face.algorithm.GaborJet
bob.bio.face.algorithm.Histogram

bob.bio.gmm.algorithm.GMM
bob.bio.gmm.algorithm.JFA
bob.bio.gmm.algorithm.ISV
bob.bio.gmm.algorithm.IVector
```

idiap RESEARCH INSTITUTE

d Security Technology
Colorado **Colorado Springs**

Outline

Excursion: conda

Installation Through conda

- Local environments using `conda`
- Provides different environments
- Bob is on conda!
- Environment defined in `bob.yml`

Installation via conda

```
# create new environment
$ conda env create -f bob.yml
# activate the environment
$ source activate bob
# test the installation
$ python -c "import caffe; import bob.bio.base"
```

Biometric Recognition with Bob

One Command Line to Run Experiments

```
verify.py
```

Biometric Recognition with Bob

One Command Line to Run Experiments

```
verify.py
```

And one More to Evaluate

```
evaluate.py
```

Biometric Recognition with bob.bio

Configuration File (see Documentation)

```
$ verify.py --create-config-file experiment.py
```

Required

```
database
preprocessor
extractor
algorithm
sub_directory
```

Optional

```
protocol
groups
parallel
verbose
dry_run
force
skip_...
```

Files and Directories

```
temp_directory
result_directory
extractor_file
projector_file
enroller_file
extracted_directory
...
```

Running an Experiment

```
$ verify.py experiment.py
```

idiap RESEARCH INSTITUTE

And Security Technology
y of Colorado Colorado Springs

Biometric Recognition with bob.bio

Writing Configuration Files

- Instantiating a class

```
import bob.bio.base
import scipy.spatial
algorithm = bob.bio.base.algorithm.PCA(
    subspace_dimension=10,
    distance_function=scipy.spatial.distance.cosine
)
```

- Including other configuration files

```
preprocessor = "path/to/other/config.py"
```

- Using registered resources

```
database = "atnt"
```

Biometric Recognition with bob.bio

Resources

- Defined in certain `bob.bio` package
- Links to configuration file
- Specifies **default** configuration
- Can be listed with `resources.py`

Listing Resources

```
$ resources.py --details --types algorithm
List of registered algorithms:
...
- bob.bio.base 2.0.9 @ /opt/conda/envs/bob/lib/python2.7/site-packages:
...
  + pca                    --> bob.bio.base.config.algorithm.pca: algorithm
    ==> bob.bio.base.algorithm.PCA.PCA(multiple_probe_scoring='average', multiple_model_scoring='average',
            subspace_dimension=0.95, is_distance_function=True, uses_variances=False,
            distance_function='<function euclidean at 0x7f649a4e55f0>')
```

Actual Configuration File

```
/opt/conda/envs/bob/lib/python2.7/site-packages/bob/bio/base/config/algorithm/pca.py
```

Evaluation

Information about Experiment: `Experiment.info`

- Inside `results/[sub_directory]`
- Contains (complete) configuration of all tools
- Contains (exact) command line to repeat experiment

Score file: `scores-dev`

- Inside `results/[sub_directory]/[protocol]/nonorm`
- Contains 4 values per line
  - → `client_id` of model
  - → `client_id` of probe
  - → probe path
  - → score

Evaluation

Evaluation Command Line

```
$ evaluate.py -vv --dev-files \
      results/[sub_directory1]/[protocol]/nonorm/scores-dev \
      results/[sub_directory2]/[protocol]/nonorm/scores-dev \
    --legends legend1 legend2 \
    ...
```

Different Ways to Evaluate

- Recognition rate `--rr`
- Equal error rate `--criterion EER`
- CMC curve `--cmc cmc.pdf`
- ROC curve `--roc roc.pdf`
- DET plot `--det det.pdf`
- DIR curve `--dir dir.pdf`

Outline

2 Biometric Recognition using Bob
- Open Source Toolboxes
- bob.bio
  - Bob's Biometric Recognition Tools
  - Biometric Recognition with bob.bio
  - Evaluation
- Hands on: Eigenfaces/Fisherfaces
  - Dataset
  - Experiment
- Bob's tools

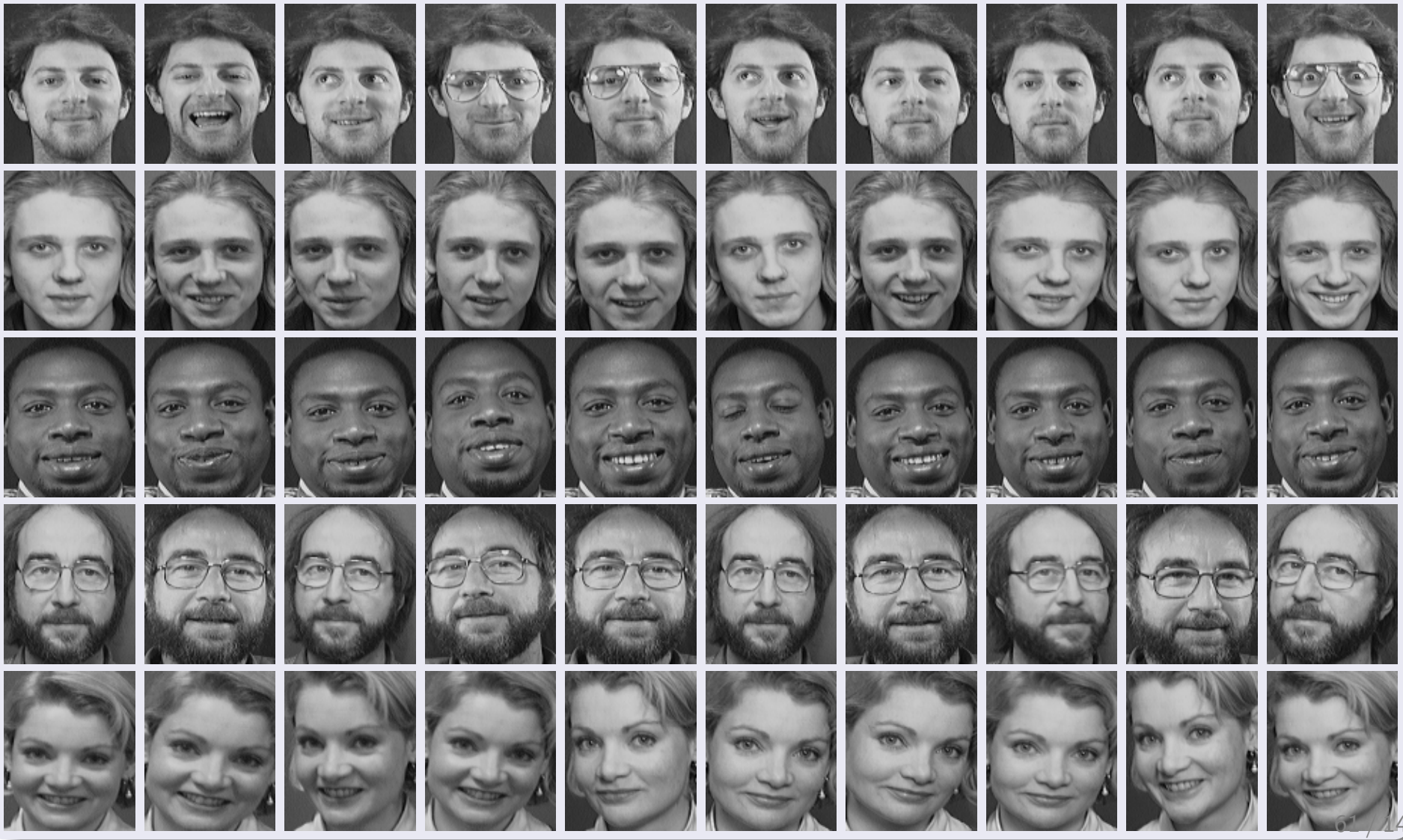
Hands on: Eigenfaces

Example Dataset: AT&T Database of Faces (ORL)

- 40 identities, 10 images each
- Training set: 20 identities
- Test set: 20 identities
  - Enrollment: 1 model from 5 images per identity
  - Probes: 5 images per identity $\rightarrow$ 5 probes
- Gray images, roughly aligned, 92×112
- Small dataset, **do not use for publications**

Hands on: Eigenfaces

Examples of the AT&T Dataset



Hands on: Eigenfaces

First Face Recognition Experiment

- Database: AT&T database of faces `"atnt"`
- Preprocessor: None (faces are aligned) `"base"`
- Extractor: Concatenate pixels `"linearize"`
- Algorithm: Principal component analysis `"pca"`

Experiment Preparation

- Write configuration file `atnt_pca.py`
- Open terminal in same directory
- Active conda environment: `$ source activate bob`

Running the Experiment

```
$ verify.py atnt_pca.py
```

idiap RESEARCH INSTITUTE

on And Security Technology
ersity of Colorado **Colorado Springs**

Hands on: Eigenfaces

Basic Experiments

- Use a face detector `"face-detect"`
- Change photometric enhancement `"tan-triggs"`
- Compare PCA with PCA+LDA `"pca+lda"`

Advanced Experiments

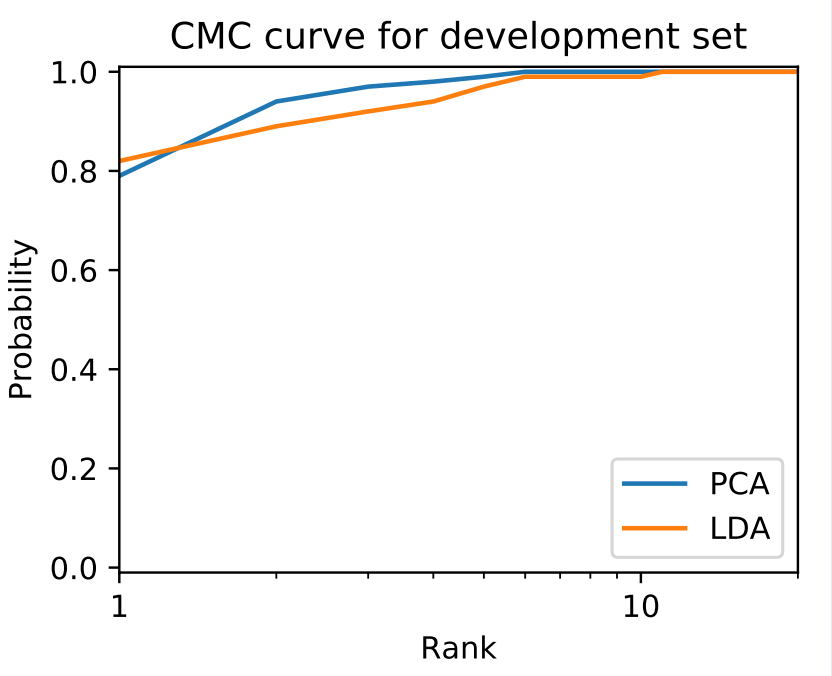
- Select different subspace dimensions for PCA
- Change distance function to cosine
- Change image resolution

Note

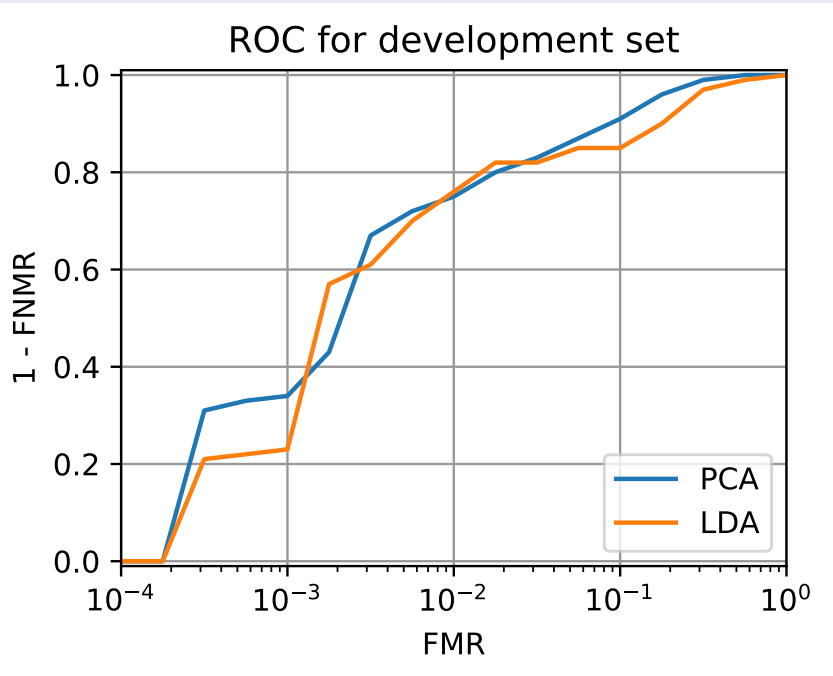
Each experiment needs a separate `sub_directory`

Hands on: Eigenfaces

Expected CMC



Expected ROC



idiap RESEARCH INSTITUTE

Vision And Security Technology
University of Colorado Colorado Springs

Bob's Biometric Recognition Tools

bob.bio.base

- Experiment infrastructure
  - → `verify.py`
  - → `evaluate.py`
  - → Toolchain
  - → Parallelism
  - → Command line
  - → Resource handling
  - → Interfacing databases
  - → Base classes
- Implemented tools
  - → Filelist database
  - → Basic algorithms

bob.bio.face

- FaceRec databases
- Face detection
- Facial features
- FaceRec algorithms

bob.bio.spear

- Speaker databases
- Voice activity detection
- Speech features (MFCC)

bob.bio.gmm

- GMM-based algorithms

How it Works in the Background

Connecting Parts of the Experiments

- Generic I/O interface `bob.io.base.load`
- Binary I/O support `bob.io.base.HDF5File`
- Image I/O support `bob.io.image`
- Audio I/O support `bob.io.audio`
- Video I/O support `bob.io.video`

Image Processing

- Face detection `bob.ip.facedetect`
- Face alignment `bob.ip.base.FaceEyesNorm`
- Photometric enhancement `bob.ip.base.TanTriggs`
- Linearization `numpy.ndarray.flatten`

How it works in the background

Learning

- PCA `bob.learn.linear.PCATrainer`
- LDA `bob.learn.linear.FisherLDATrainer`

Database Interface

- Base interface `bob.bio.base.database`
- Implemented `List of all interfaces`
- AT&T database interface `bob.db.atnt`

Evaluation

- Loading score files `bob.measure.load`
- Curves `bob.measure`
- Plots `matplotlib.pyplot.plot`

Outline

Biometric Recognition with GMMs

GMMs for Biometric Recognition?

- Widely used for speech processing
- Mostly unknown in other biometrics

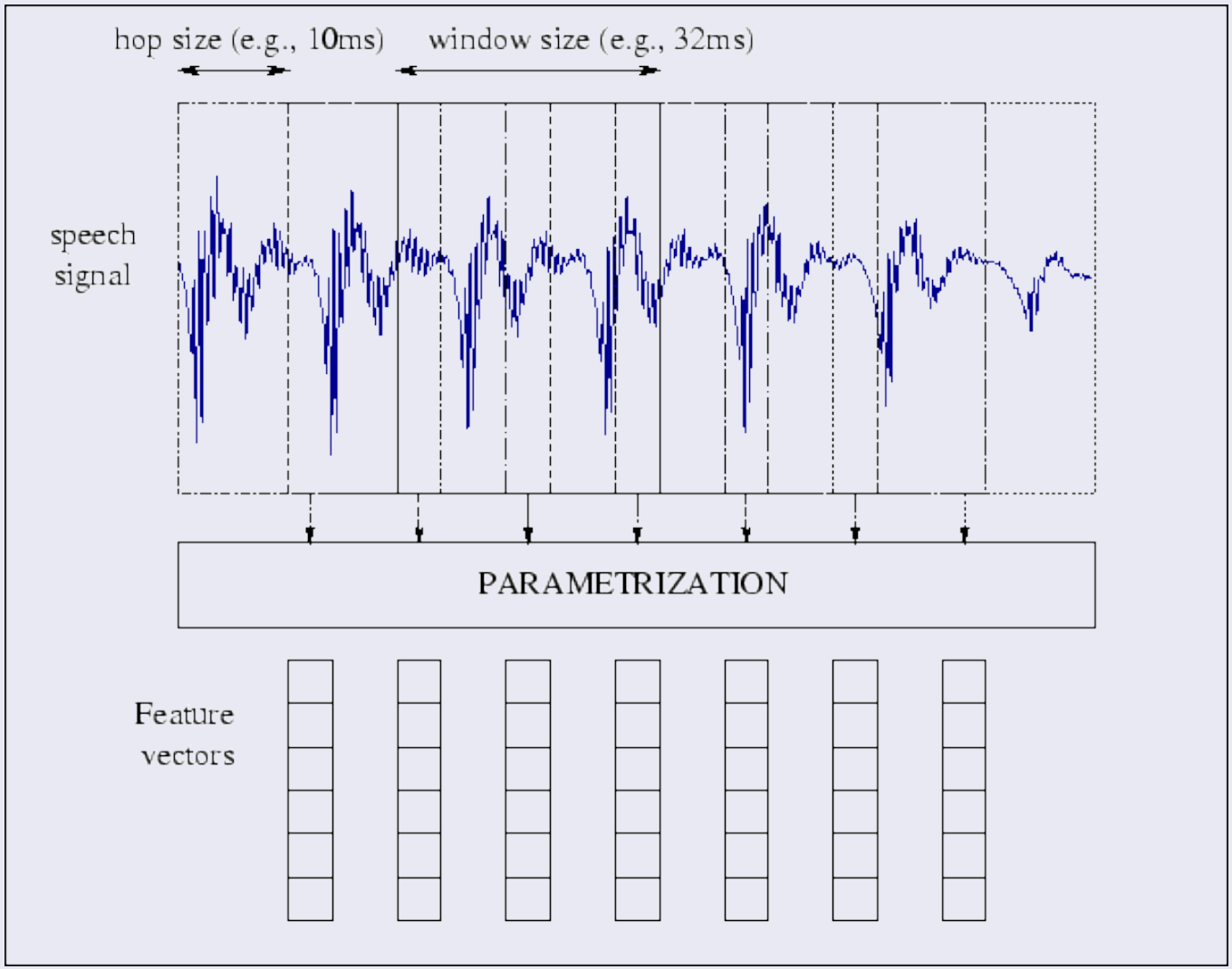
Basic Idea

- Extract several **local** features from data sample
- Treat features as **independent** observations
- Enrollment: Model **distribution** of observations from enrollment samples
- Scoring: Compute probability that probe observations were **generated** by model

gy
gs

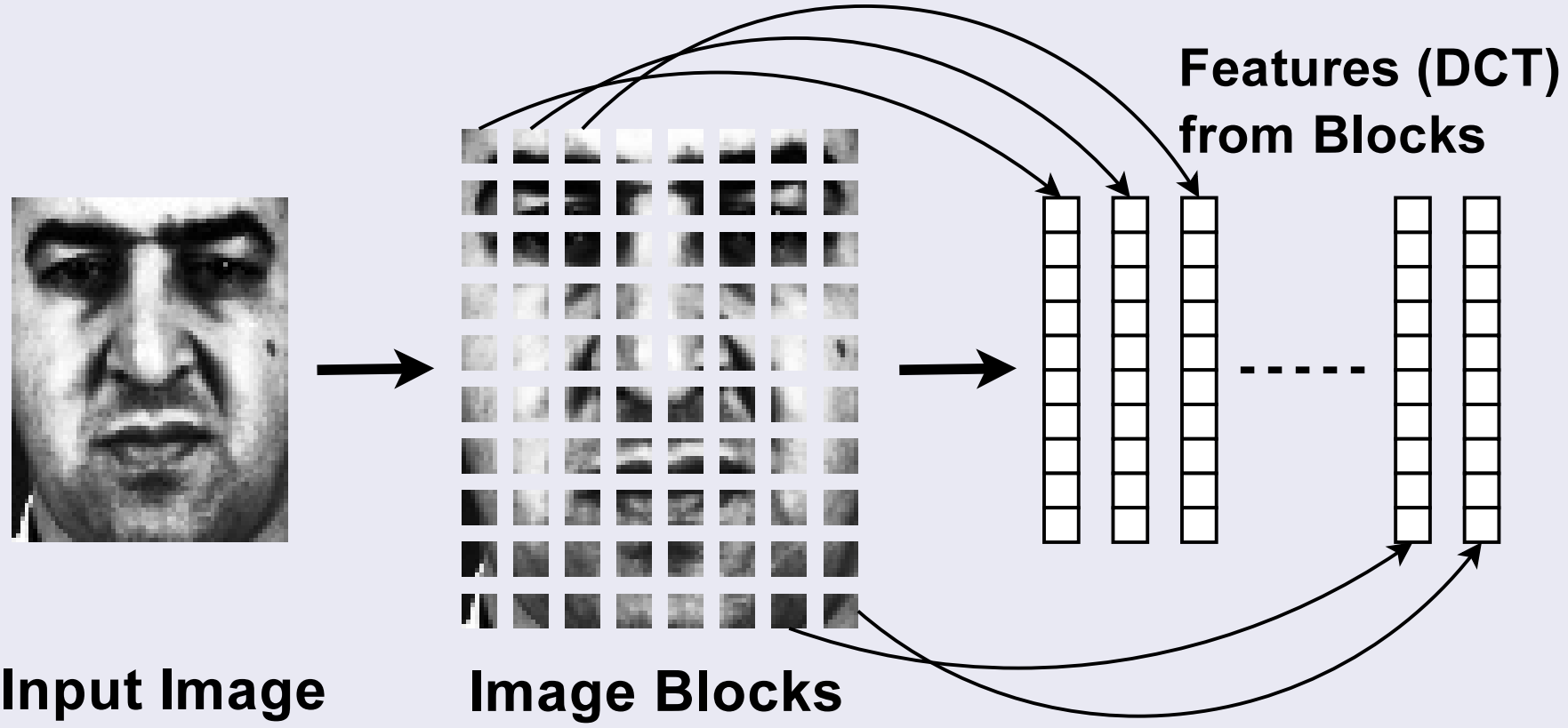
Local Features

Speech Signal Extraction (link)



Local Features

DCT Blocks



Outline

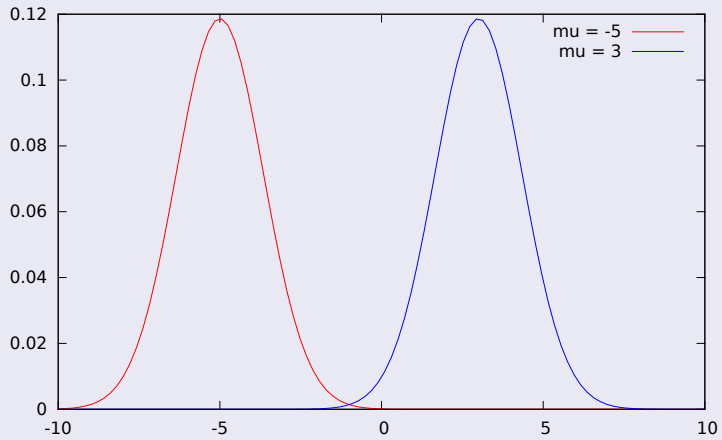
Gaussian Distributions

Gaussian Distribution (Normal Distribution)

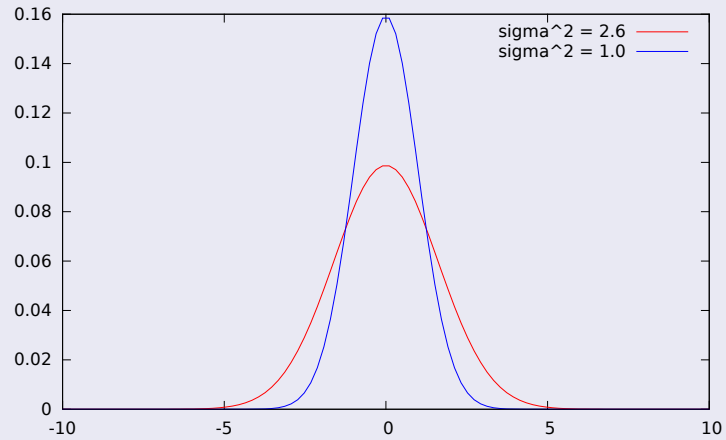
$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- x: the input variable $\in \mathbb{R}$
- μ: the mean $\in \mathbb{R}$
- σ^2: the variance $\in \mathbb{R}$

Mean



Variance



Gaussian Distributions

Estimating Gaussian Parameters from Data

- Given: dataset $\mathcal{D} = \{x_1, \ldots, x_N\}; x_n \in \mathbb{R}$
- Goal: Compute optimal μ and σ^2

Mean

$$\mu = \frac{1}{N} \sum_{n=1}^{N} x_n$$

Variance

$$\sigma^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_n - \mu)^2$$

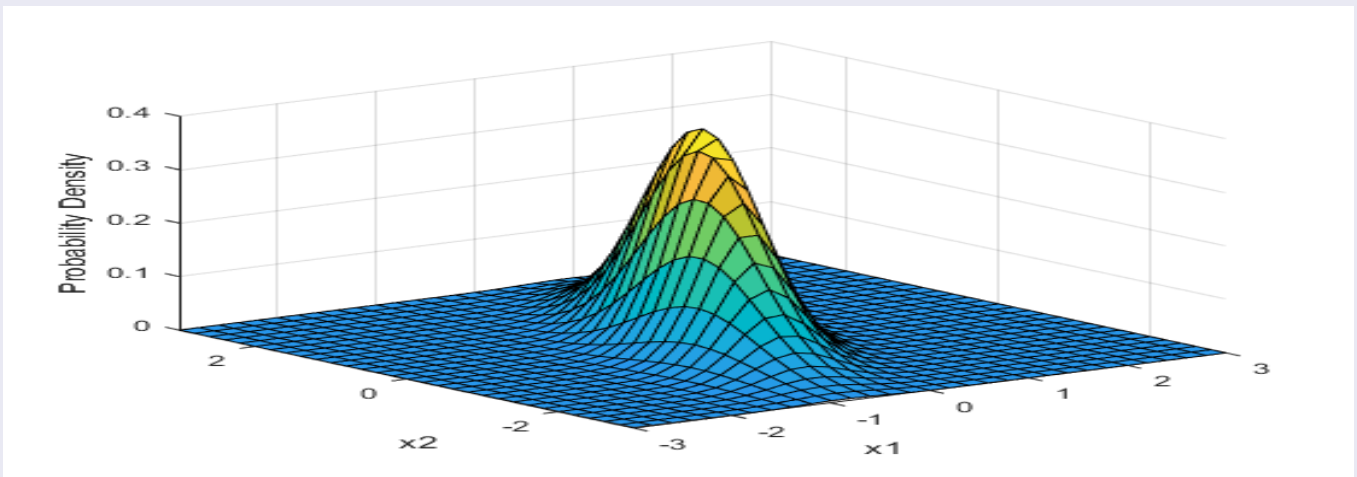
Multivatiate Gaussians

Multivariate Gaussian (Normal) Distribution

$$\mathcal{N}_{\vec{\mu},\Sigma}(\vec{x}) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \mathrm{e}^{-\frac{1}{2}(\vec{x}-\vec{\mu})^{\mathrm{T}}\Sigma^{-1}(\vec{x}-\vec{\mu})}$$

- $\vec{x}$: the multivariate input $\in \mathbb{R}^d$
- $\vec{\mu}$: the mean $\in \mathbb{R}^d$
- Σ: the covariance matrix $\in \mathbb{R}^{d\times d}$

2D Gaussian (link)



Multivatiate Gaussians

Estimating Multivariate Parameters from Data

- Given: dataset $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_N\}; \vec{x}_n \in \mathbb{R}^d$
- Goal: Compute optimal $\vec{\mu}$ and Σ

Mean

$$\vec{\mu} = \frac{1}{N} \sum_{n=1}^{N} \vec{x}_n$$

Covariance

$$\Sigma = \frac{1}{N-1} \sum_{n=1}^{N} (\vec{x}_n - \vec{\mu})(\vec{x}_n - \vec{\mu})^T$$

Gaussian Mixture Models

Multivariate Gaussian Mixture Models

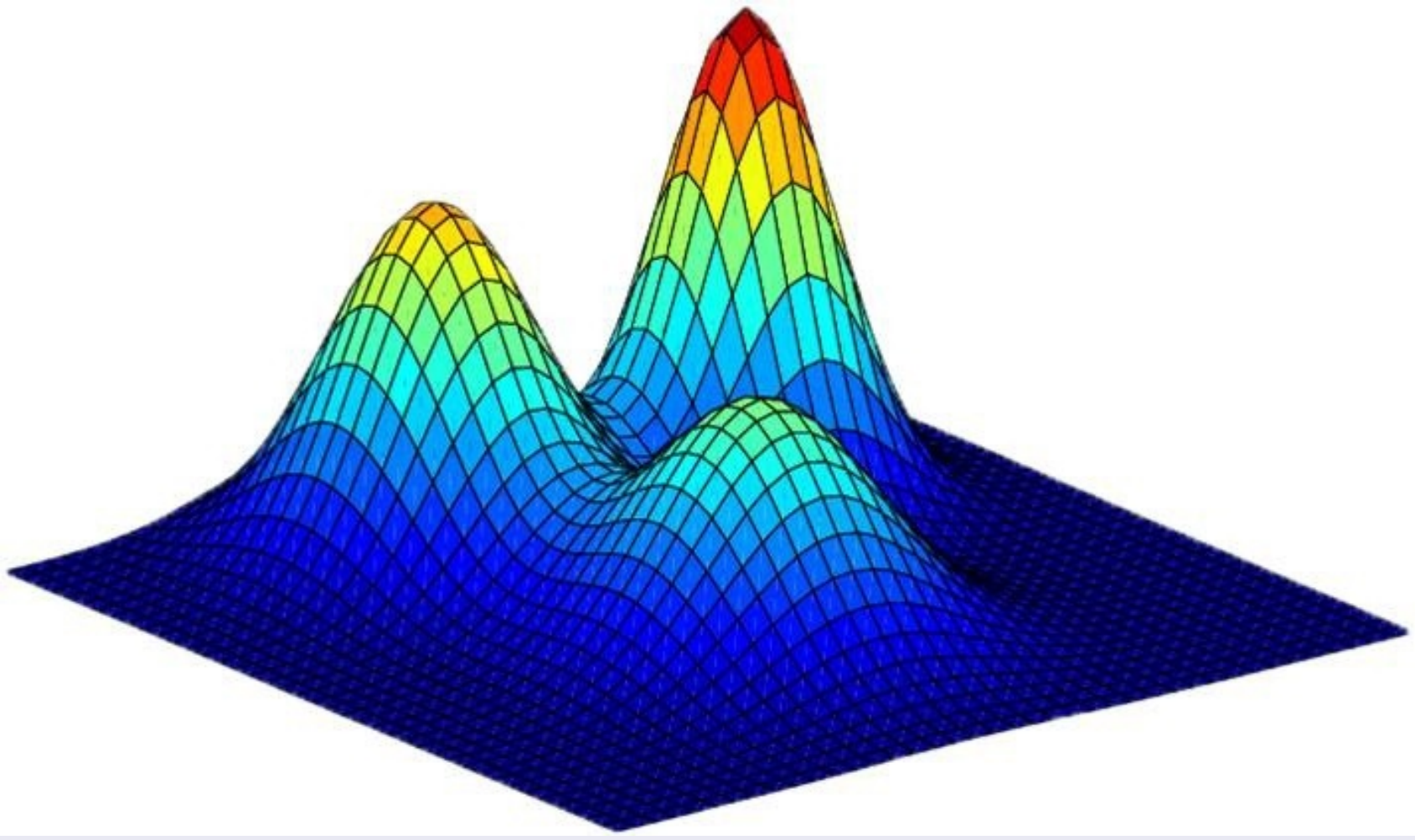
Collection of M Gaussian components:

$$\mathcal{P}_\Theta(\vec{x}) = \sum_{i=1}^{M} w_i \, \mathcal{N}_{\vec{\mu}_i, \Sigma_i}(\vec{x})$$

- $\vec{x}$: the multivariate input $\in \mathbb{R}^d$
- $\Theta = \{w_i, \vec{\mu}_i, \Sigma_i \mid \forall i = 1, \ldots, M\}$
  - w_i: mixing coefficient (weight) $\in \mathbb{R}$
    - $\rightarrow \sum_{i=1}^{M} w_i = 1$
  - $\vec{\mu}_i$: mean $\in \mathbb{R}^d$ of component i
  - Σ_i: covariance matrix of component i
    - $\rightarrow$ full rank $\in \mathbb{R}^{d \times d}$
    - $\rightarrow$ or diagonal $\in \mathbb{R}^d$ with elements $\vec{\sigma}_i$

Gaussian Mixture Models

Example GMM with $d = 2$ and $M = 3$ (link)



Outline

The EM Algorithm for GMMs

How to Obtain the GMM Parameters Θ?

- Given: dataset $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_N\}; \vec{x}_n \in \mathbb{R}^d$
- Goal: Θ^* that maximizes likelihood $\mathcal{L}(\Theta \mid \mathcal{D})$
- Assuming **independence** of training examples $\vec{x}_n$

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} \mathcal{L}(\Theta \mid \mathcal{D}) \\ &= \arg\max_{\Theta} \prod_{n=1}^{N} \mathcal{P}_{\Theta}(\vec{x}_n) \\ &= \arg\max_{\Theta} \sum_{n=1}^{N} \log \mathcal{P}_{\Theta}(\vec{x}_n)\end{aligned}$$

The EM Algorithm for GMMs

Log Likelihood

$$\log \mathcal{L}(\Theta \mid \mathcal{D}) = \sum_{n=1}^{N} \log \left\{ \sum_{i=1}^{M} w_i \mathcal{N}_{\vec{\mu}_i, \Sigma_i}(\vec{x}_n) \right\}$$

Idea of Expectation Maximization for GMMs

- Initialize parameters $\Theta^{(0)}$
- Iterate (here: round r):
  1. E-Step: estimate relevance $P_{\Theta^{(r)}}(i \mid \vec{x}_n)$ of each component i for each data point $\vec{x}_n$
  2. M-Step: estimate new Gaussian parameters $\Theta^{(r+1)}$ using new relevances
- Until $\log \mathcal{L}(\Theta^{(r+1)} \mid \mathcal{D}) - \log \mathcal{L}(\Theta^{(r)} \mid \mathcal{D}) < \epsilon$

The EM Algorithm for GMMs

E-step: Relevance of Component i

$$P_\Theta(i \mid \vec{x}_n) = \frac{w_i \, \mathcal{N}_{\vec{\mu}_i, \Sigma_i}(\vec{x}_n)}{\sum_{j=1}^{M} w_j \, \mathcal{N}_{\vec{\mu}_j, \Sigma_j}(\vec{x}_n)}$$

M-step: Weights

$$w_i = \frac{1}{N} \sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n)$$

M-step: Means

$$\vec{\mu}_i = \frac{\sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n) \, \vec{x}_n}{\sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n)}$$

M-step: Variances (Dull or Diagonal)

$$\Sigma_i = \frac{\sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n)(\vec{x}_n - \vec{\mu}_i)(\vec{x}_n - \vec{\mu}_i)^T}{\sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n)} \quad \text{or} \quad \vec{\sigma}_i^2 = \frac{\sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n) \, (\vec{x}_n - \vec{\mu}_i)^2}{\sum_{n=1}^{N} P_\Theta(i \mid \vec{x}_n)}$$

The EM Algorithm for GMMs

Parallelization of the E-step over Training set $\mathcal{D}$

1. Load GMM model from previous step
2. Compute relevances (posterior probabilities)
3. Pre-compute weights, means and variances
4. Store weights, means and variances into `GMMStats`

M-step (not Parallelized)

1. Load all `GMMStats`
2. Aggregate weights, means and variances
3. Store new GMM model

Outline

Biometric Recognition with GMMs

Practical Considerations

- Select initial $\Theta^{(0)}$
  - $\rightarrow$ Randomly $\Rightarrow$ slow and unreliable
  - $\rightarrow$ Based on initial K-Means clustering on $\mathcal{D}$
- Small sample size problem during enrollment
  - $\rightarrow$ Ignore $\Rightarrow$ models overfit
  - $\rightarrow$ Compute Universal Background Model (UBM) + Adapt model to enrollment data
- How to compute the score for model and probe
  - $\rightarrow$ Log likelihood using only model
  - $\rightarrow$ Log likelihood ratio using model and UBM

Initialization of $\Theta^{(0)}$

K-Means

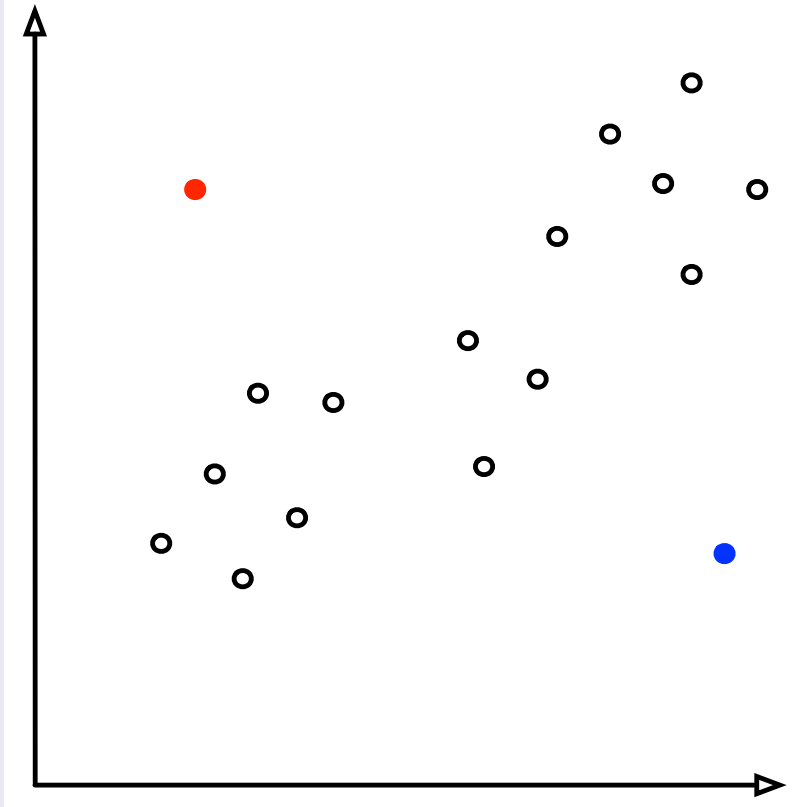
- Given: dataset $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_N\}; \vec{x}_n \in \mathbb{R}^d$
- Goal: Cluster input data into $K = M$ clusters
- Implementation: EM-Algorithm

EM for K-Means

- Initialize M means $\vec{\mu}_i$ randomly or at data points $\vec{x}_n$
- Iterate:
  1. Assign each data point to the nearest cluster
  2. Compute new mean $\vec{\mu}_i$ from cluster
- Until clusters do not change any more

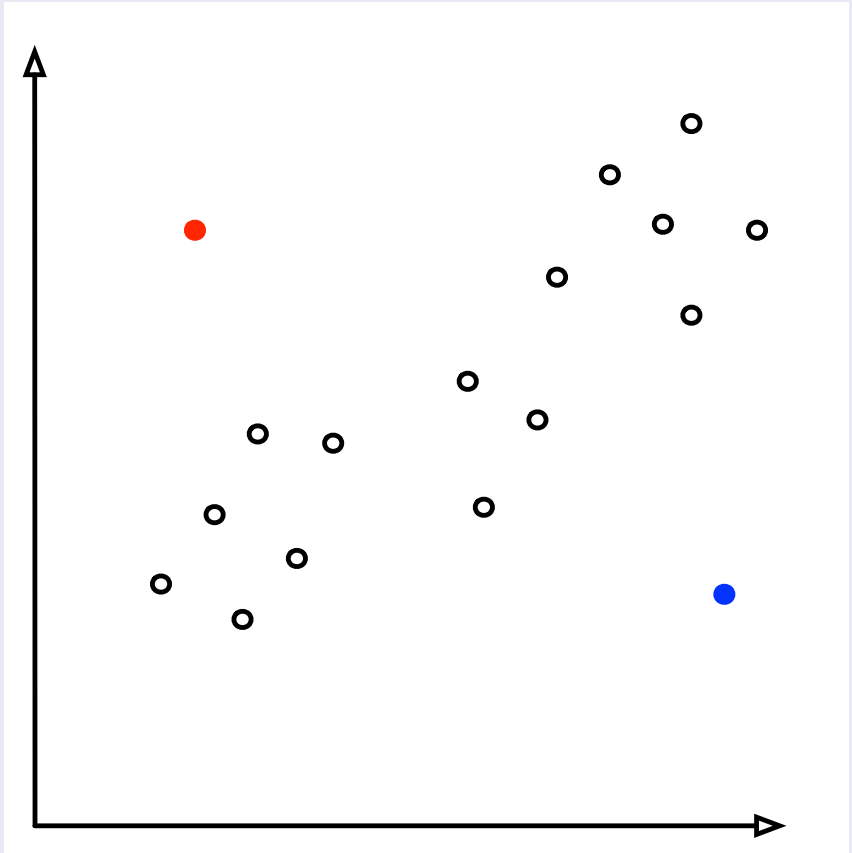
K-Means

Random Initialization

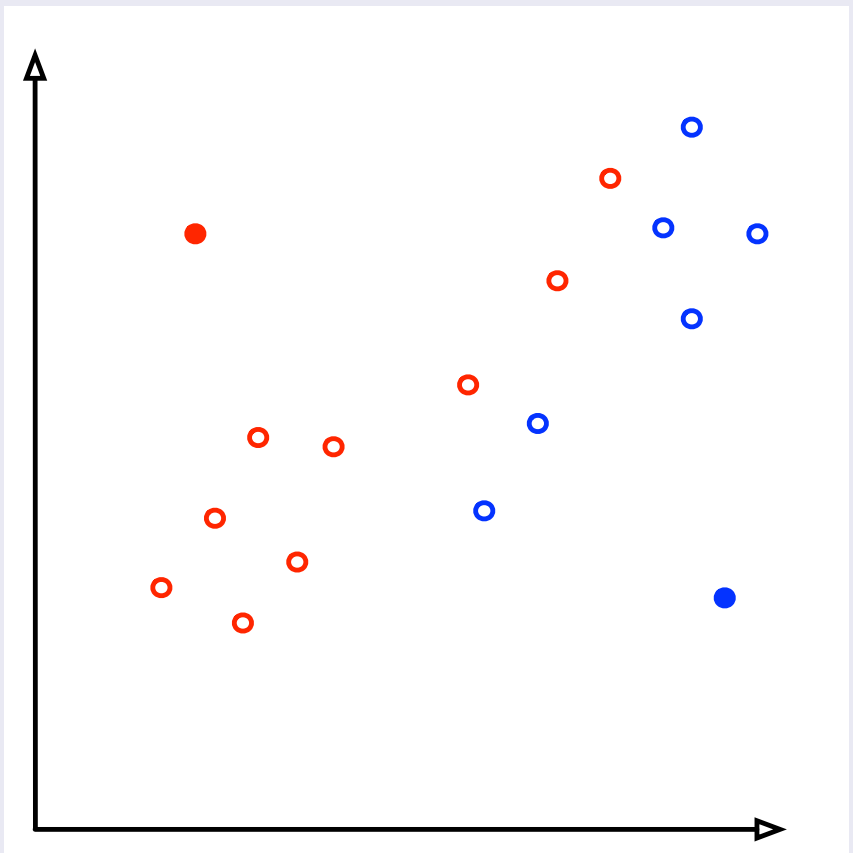


K-Means

Random Initialization

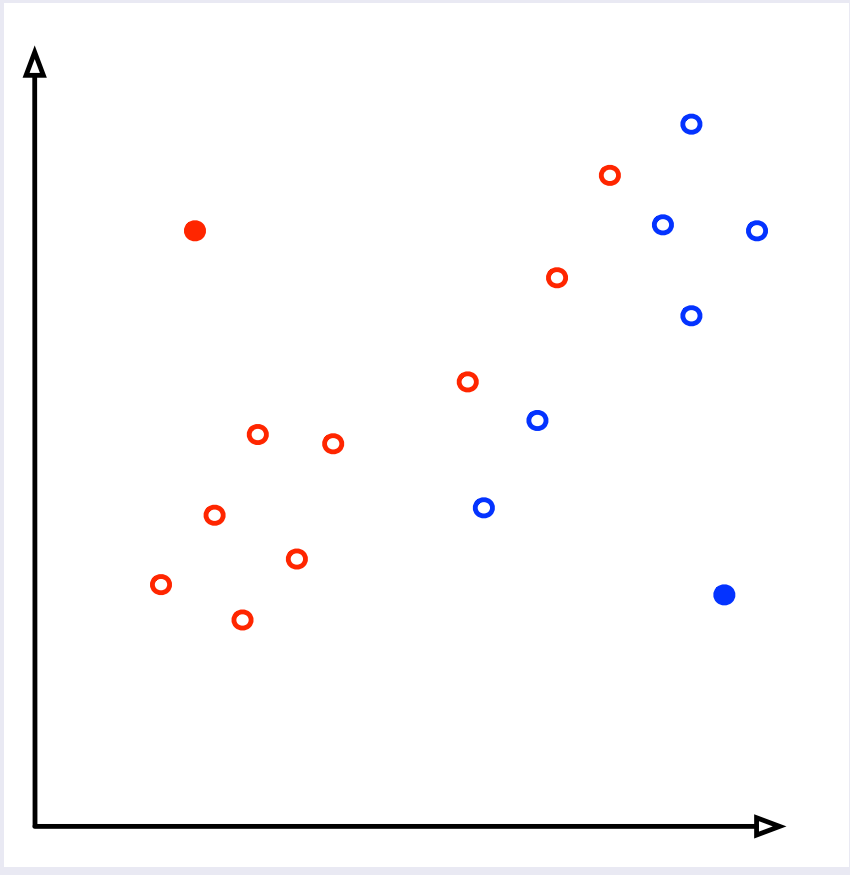


E-Step 1: Assign Clusters

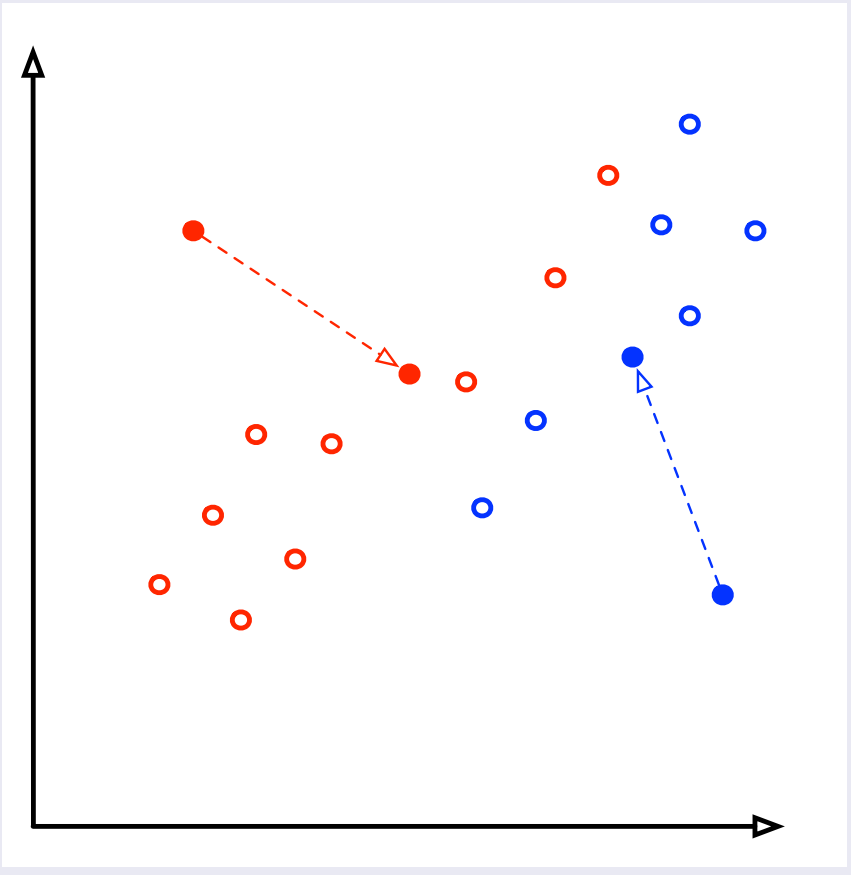


K-Means

E-Step 1: Assign Clusters

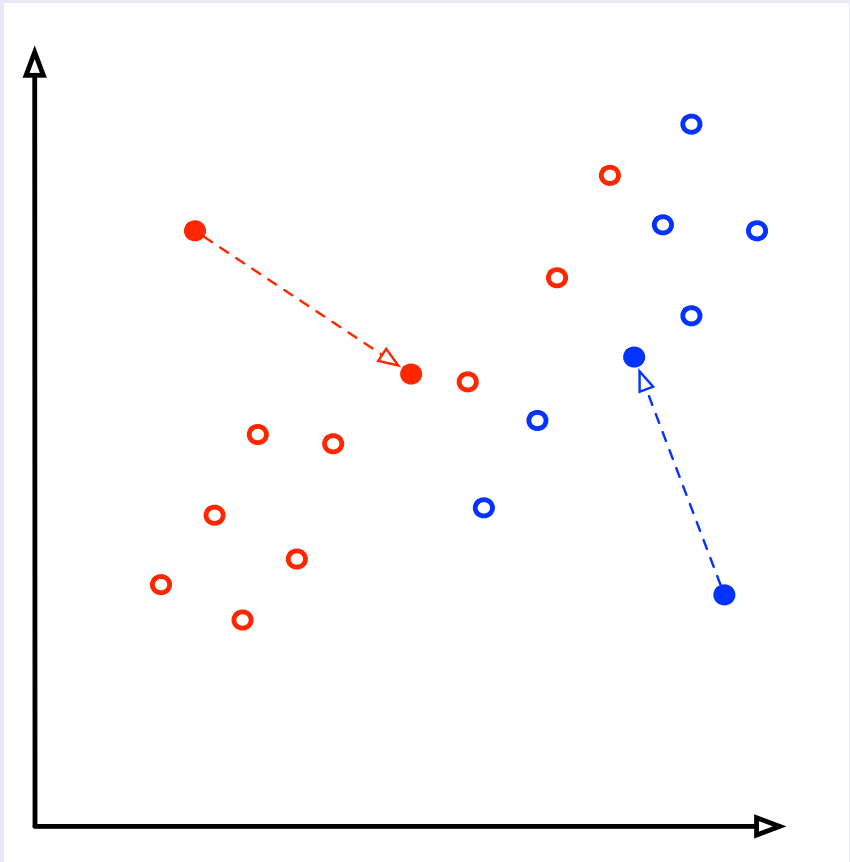


M-Step 1: Update Means

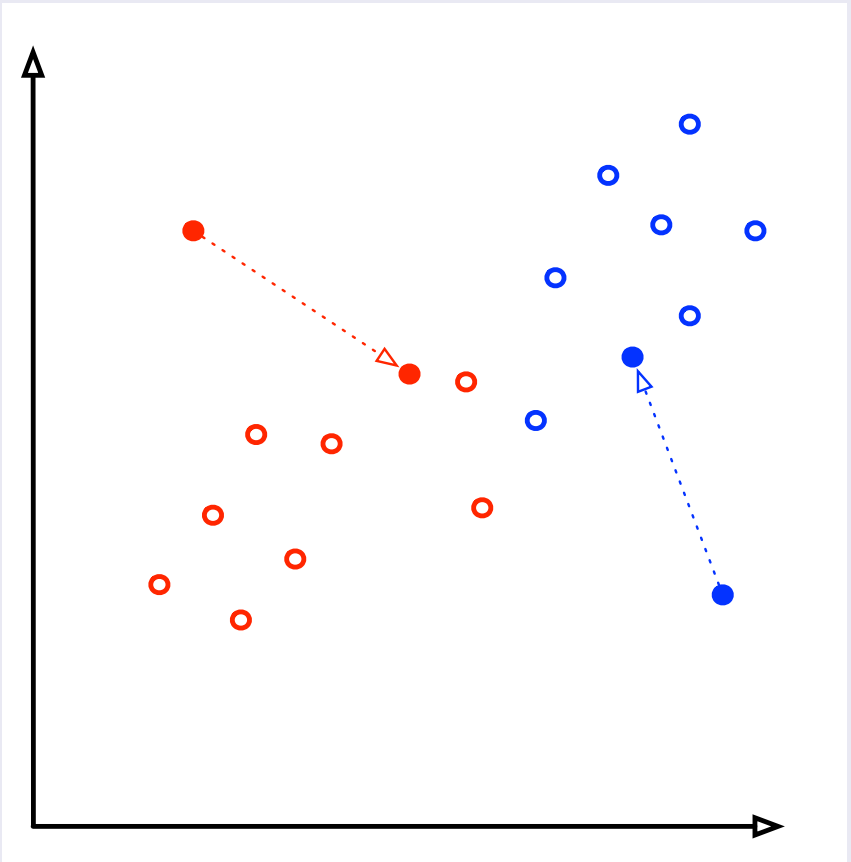


K-Means

M-Step 1: Update Means

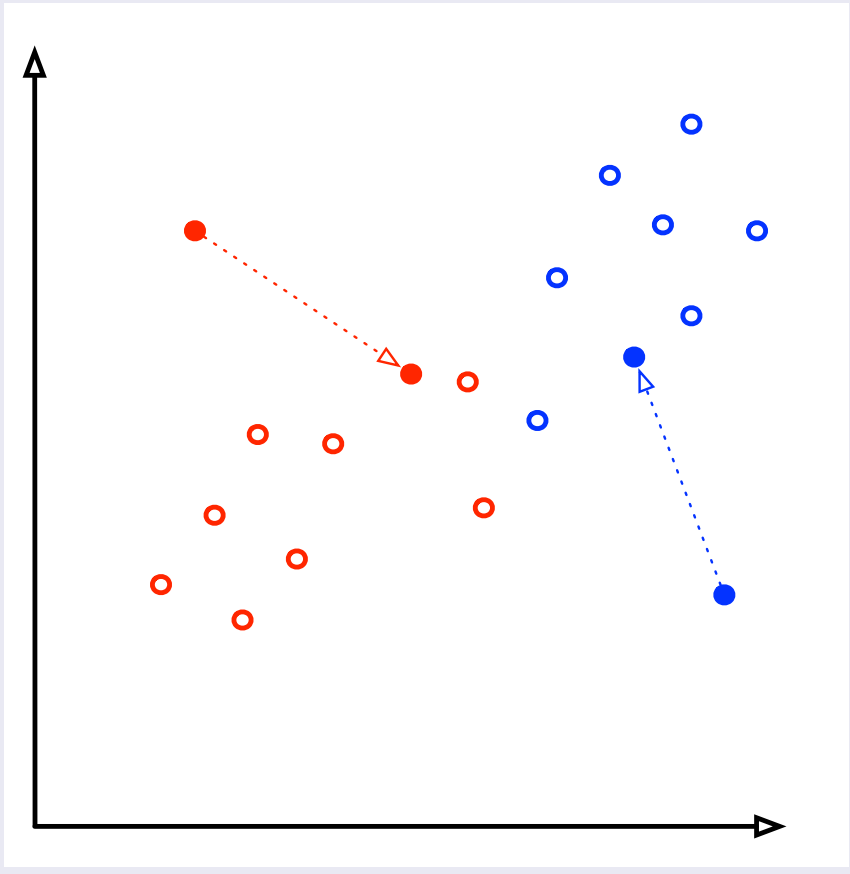


E-Step 2: Assign Clusters

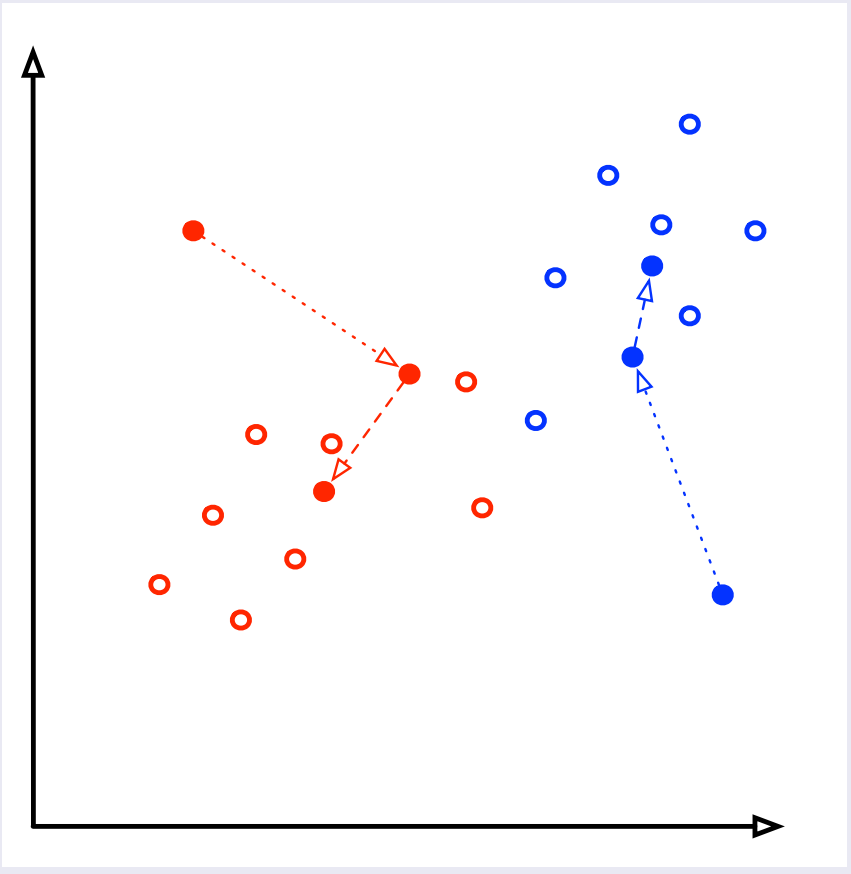


K-Means

E-Step 2: Assign Clusters

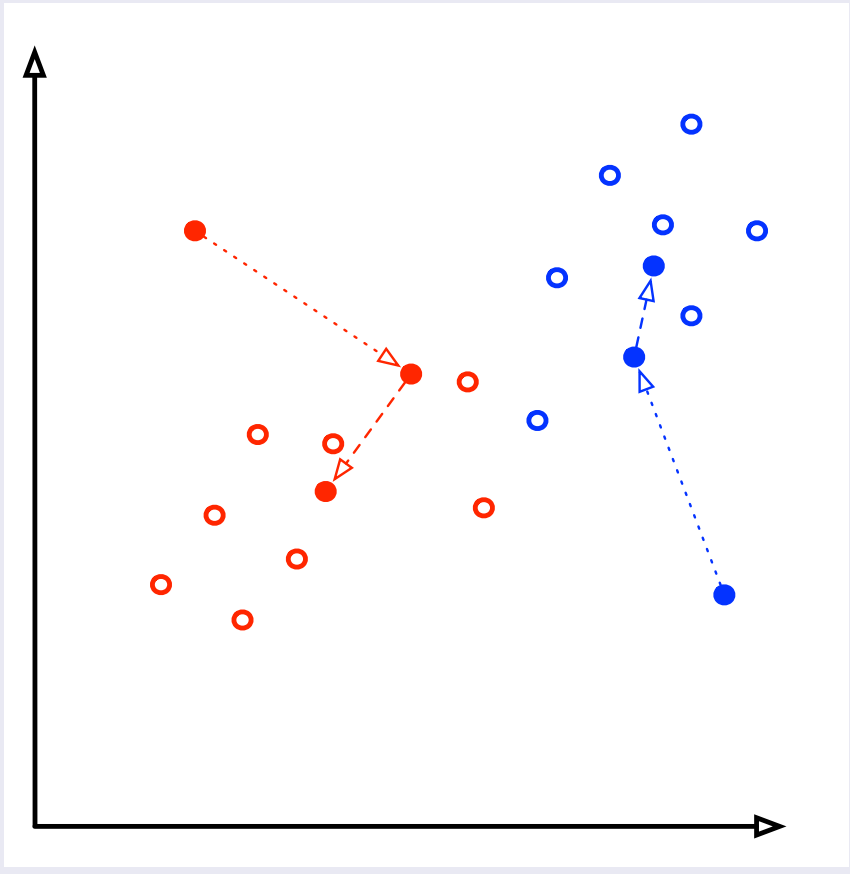


M-Step 2: Update Means

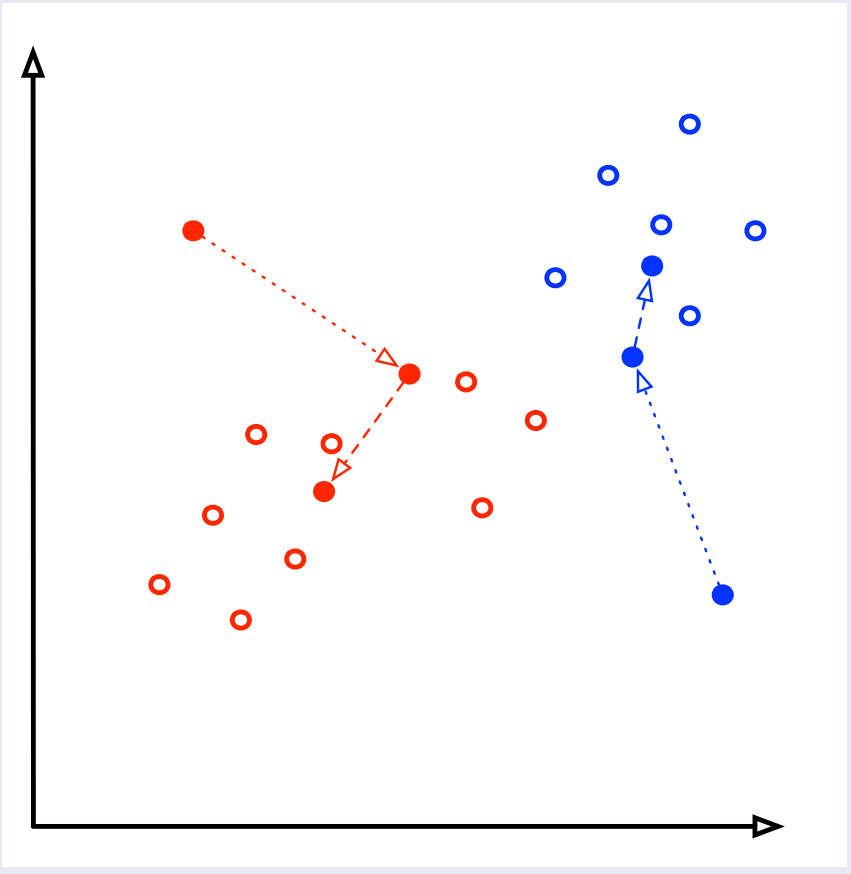


K-Means

M-Step 2: Update Means

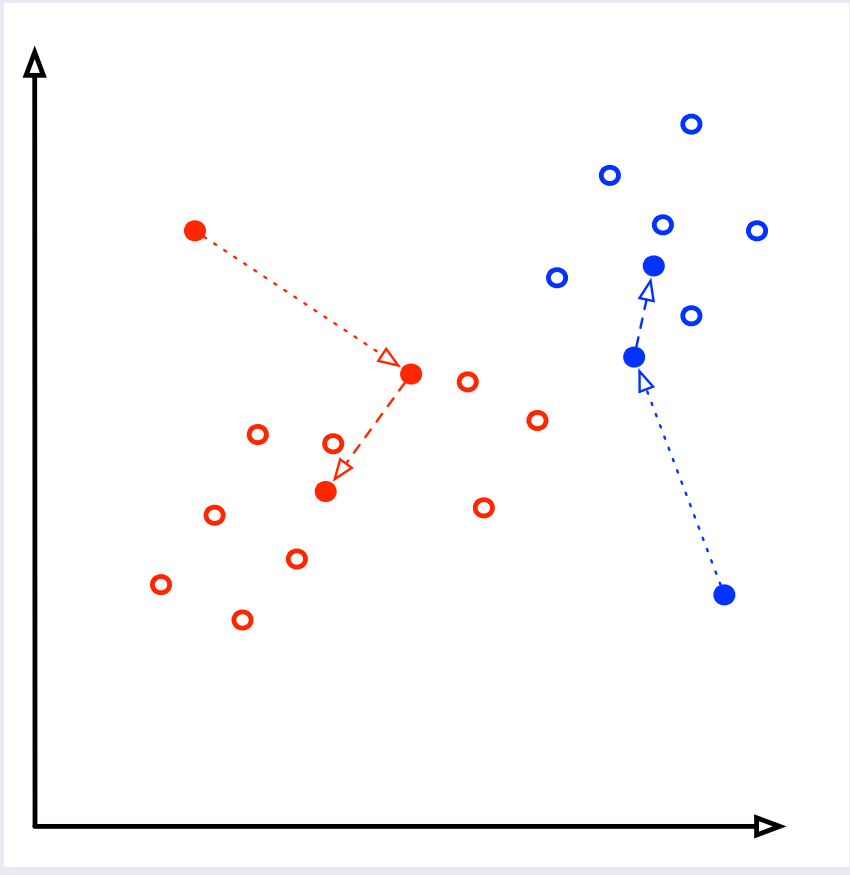


E-Step 3: Assign Clusters

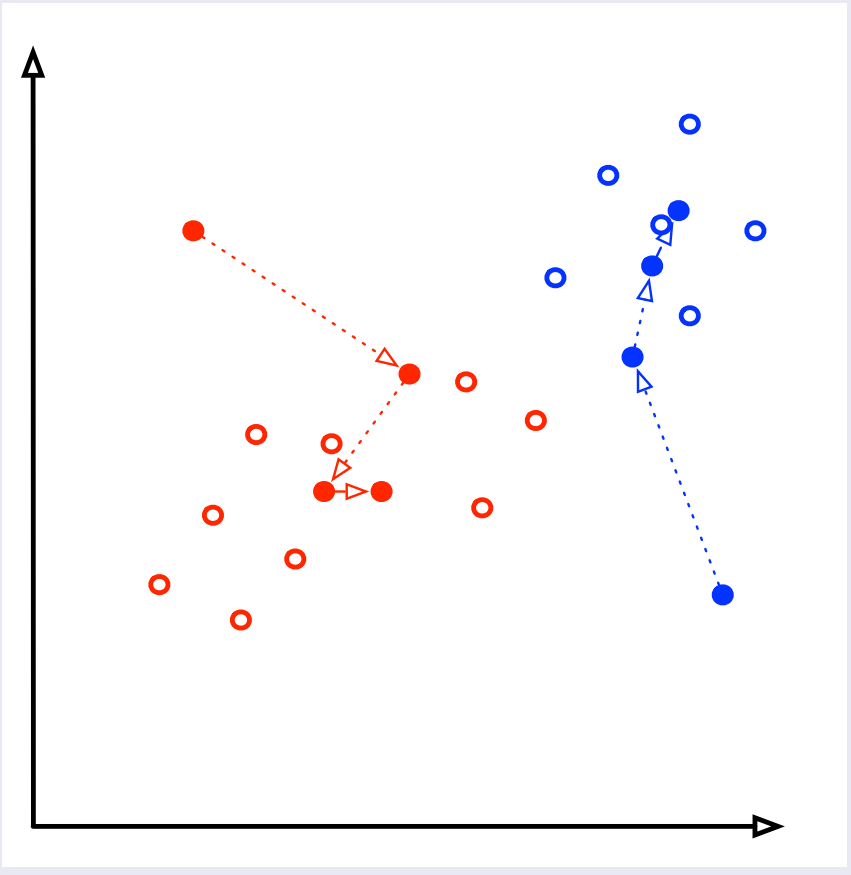


K-Means

E-Step 3: Assign Clusters



M-Step 3: Update Means



Biometric Recognition with GMMs

Practical Considerations

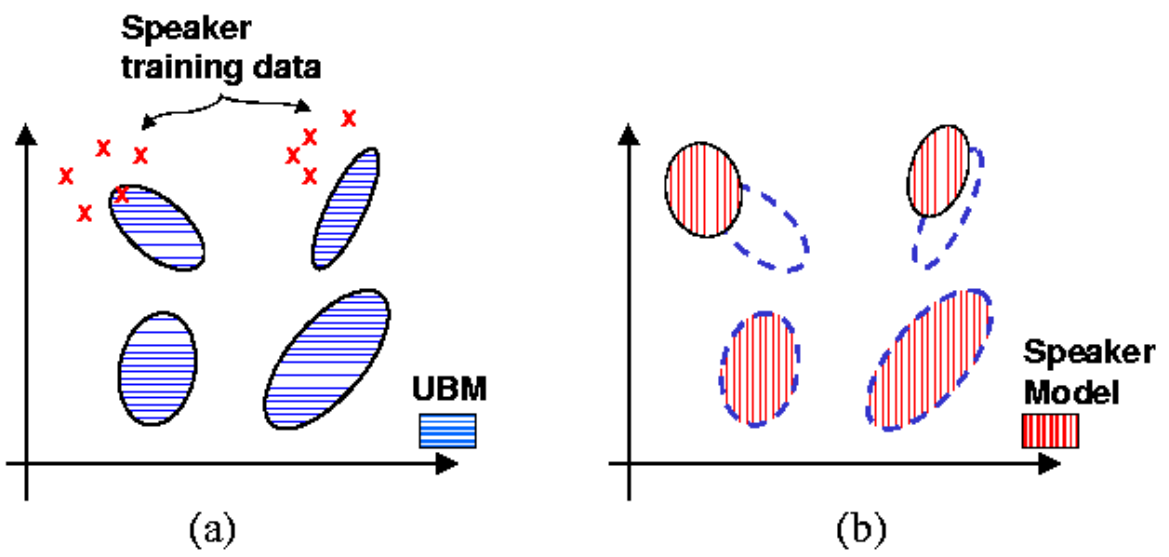
- Select initial $\Theta^{(0)}$
  - → Randomly ⇒ slow and unreliable
  - → Based on initial K-Means clustering on $\mathcal{D}$
- Small sample size problem during enrollment
  - → Ignore ⇒ models overfit
  - → Compute Universal Background Model (UBM) + Adapt model to enrollment data
- How to compute the score for model and probe
  - → Log likelihood using only model
  - → Log likelihood ratio using model and UBM

Universal Background Model

UBM/GMM enrollment

- Compute Universal Background Model (UBM) Θ
  - $\rightarrow$ Utilize features from training set
- Adapt UBM to client-specific model Θ^c
  - $\rightarrow$ Use enrollment data $\{\vec{x}_1^c, \ldots, \vec{x}_{N^c}^c\}$ of client c

From Reynolds et al. (2000)



Mean-Only MAP Adaptation

Relevance for Component i (E-Step)

$$P_\Theta(i \mid \vec{x}_n^c) = \frac{w_i \, \mathcal{N}_{\vec{\mu}_i, \Sigma_i}(\vec{x}_n^c)}{\sum_{j=1}^{M} w_j \, \mathcal{N}_{\vec{\mu}_j, \Sigma_j}(\vec{x}_n^c)}$$

Mean Adaptation (M-Step)

$$\Delta\vec{\mu}_i^c = \frac{\sum_{n=1}^{N^c} P_\Theta(i \mid \vec{x}_n^c) \, \vec{x}_n^c}{\sum_{n=1}^{N^c} P_\Theta(i \mid \vec{x}_n^c)}$$

Relevance Factor τ

$$\alpha_i = \frac{\sum_{n=1}^{N^c} P_\Theta(i \mid \vec{x}_n^c)}{\tau + \sum_{n=1}^{N^c} P_\Theta(i \mid \vec{x}_n^c)}$$

Weighted Mean Adaptation

$$\vec{\mu}_i^c = \alpha_i \, \Delta\vec{\mu}_i^c + (1 - \alpha_i) \, \vec{\mu}_i$$

idiap RESEARCH INSTITUTE

Security Technology
olorado Colorado Springs

Biometric Recognition with GMMs

Practical Considerations

- Select initial $\Theta^{(0)}$
  - → Randomly ⇒ slow and unreliable
  - → Based on initial K-Means clustering on $\mathcal{D}$
- Small sample size problem during enrollment
  - → Ignore ⇒ models overfit
  - → Compute Universal Background Model (UBM) + Adapt model to enrollment data
- How to compute the score for model and probe
  - → Log likelihood using only model
  - → Log likelihood ratio using model and UBM

Log Likelihood Ratio Scoring

Test Probe Features $\{\vec{x}_1^p, \ldots, \vec{x}_{N^p}^p\}$ with Model Θ^c

$$s = \sum_{n=1}^{N^p} \log \frac{P_{\Theta^c}(\vec{x}_n^p)}{P_\Theta(\vec{x}_n^p)}$$

Advantages

- Concentrates on client-specific features (e.g. scars)
  - $\rightarrow$ Can be explained by Θ^c **but not** by Θ
- Ignores features that are common between identities
  - $\rightarrow$ Can be explained by Θ^c **and** by Θ
- Ignores features from background or occlusions
  - $\rightarrow$ Can**not** be explained by **either** Θ^c or Θ
- Denominator can be precomputed (projection)

Implementation in bob.bio.gmm

Algorithm bob.bio.gmm.algorithm.GMM

- Define parameters `__init__`
- Compute UBM Θ `train_projector`
- Pre-compute $P_\Theta(\vec{x}_n^p)$ `project`
- Client-specific model Θ^c `enroll`
- Log-likelihood ratio `score`

Parallelizing UBM Training

- K-Means for GMM initialization (E-Step)
- UBM training (E-Step)
- Use specific command line `verify_gmm.py`

Outline

Excursion GridTK

bob.bio Speaks GridTK

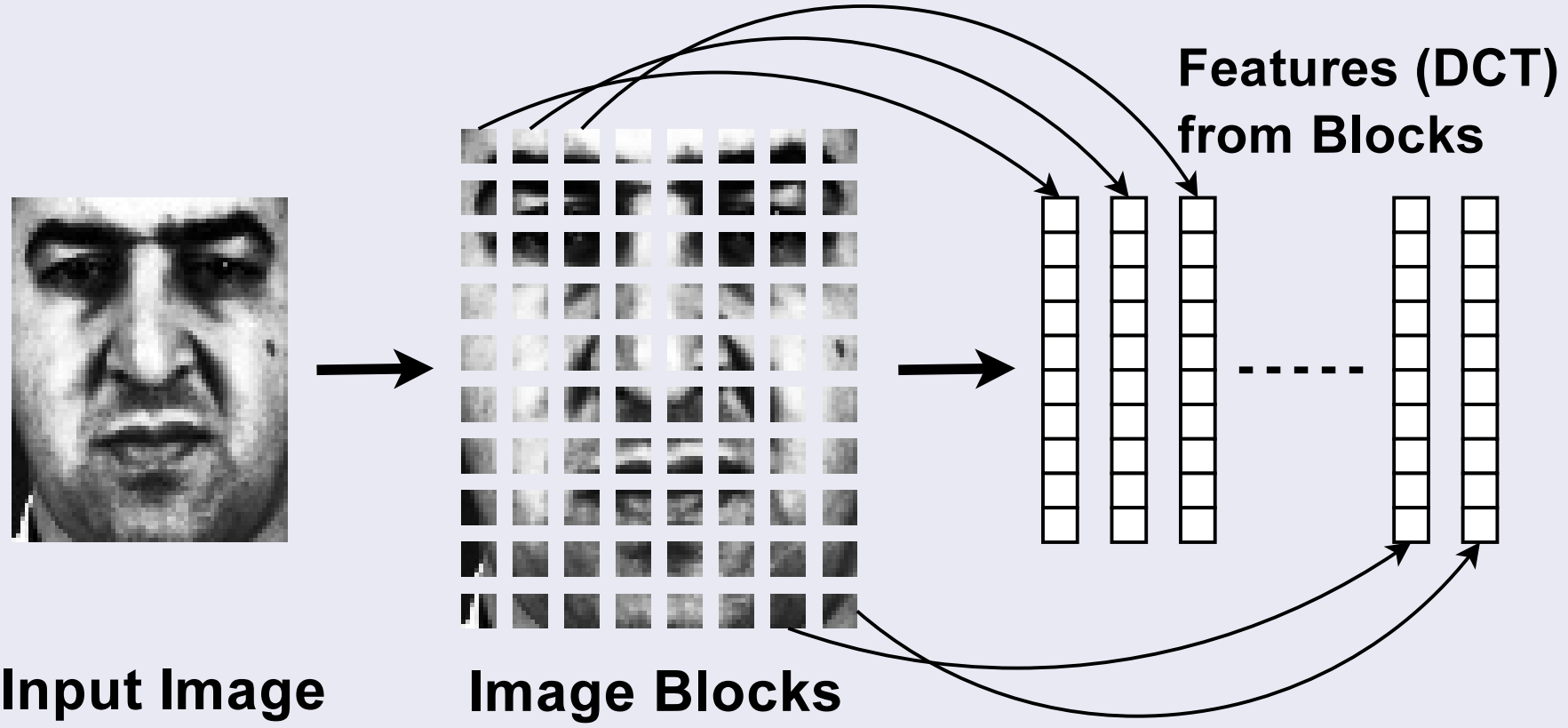
- Developed to submit and monitor jobs with SGE
- Implements a local scheduler with dependencies
- `verify.py` will submit and execute jobs locally

Monitoring Jobs with the Job Manager

- List current jobs and their dependencies: `jman list -xl`
- List parallel jobs `jman list -j 1-4 -a`
- Check logs of processes `jman report -j 1-2`
- Delete jobs `jman del -j 1-20`

Local Features

DCT Blocks



Feature Selection

GMM Configuration

```
import bob.bio.gmm
algorithm = bob.bio.gmm.algorithm.GMM(
    number_of_gaussians = 20,
    kmeans_training_iterations = 5,
    gmm_training_iterations = 5
)
```

Resource (**not Recommended in this Tutorial**)

```
algorithm = "gmm"
```

Hands on: Gaussian Mixture Models

GMM-based Face Recognition

- Database: AT&T database of faces `"atnt"`
- Preprocessor: Tan&Triggs `"tan-triggs"`
- Features: DCT Blocks `"dct-blocks"`
- Algorithm: Gaussian Mixture Models (see above)

Experiment Preparation

- Write configuration file `atnt_gmm.py`
- Open terminal in `Desktop/Tutorial`
- Active conda environment: `$ source activate bob`

Running the Experiment in Parallel

```
$ verify_gmm.py atnt_gmm.py --parallel 4
```

Hands on: Gaussian Mixture Models

Basic Experiments

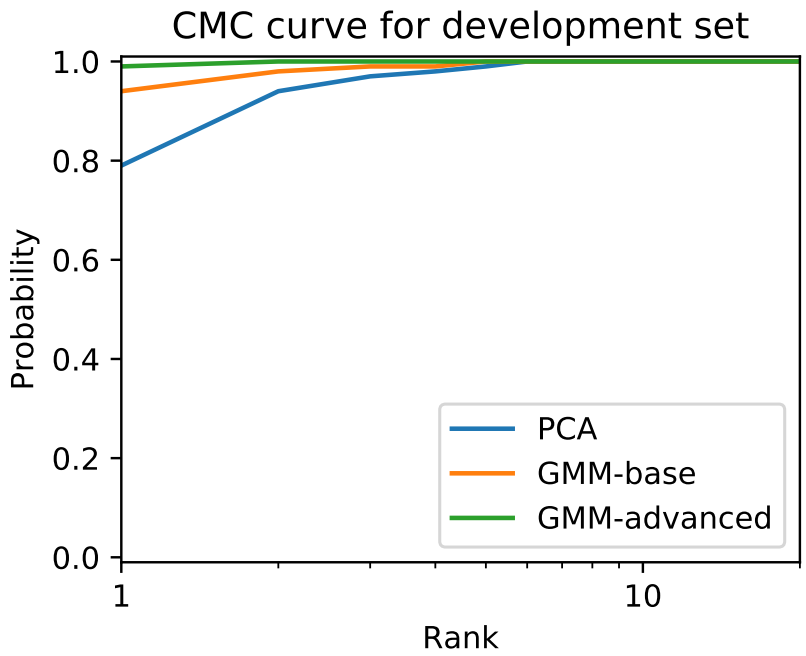
- Compare different numbers of Gaussians
  → Warning! More Gaussians require more memory
- Compare different numbers of training iterations
  → Warning! More iterations require more time

Advanced Experiments

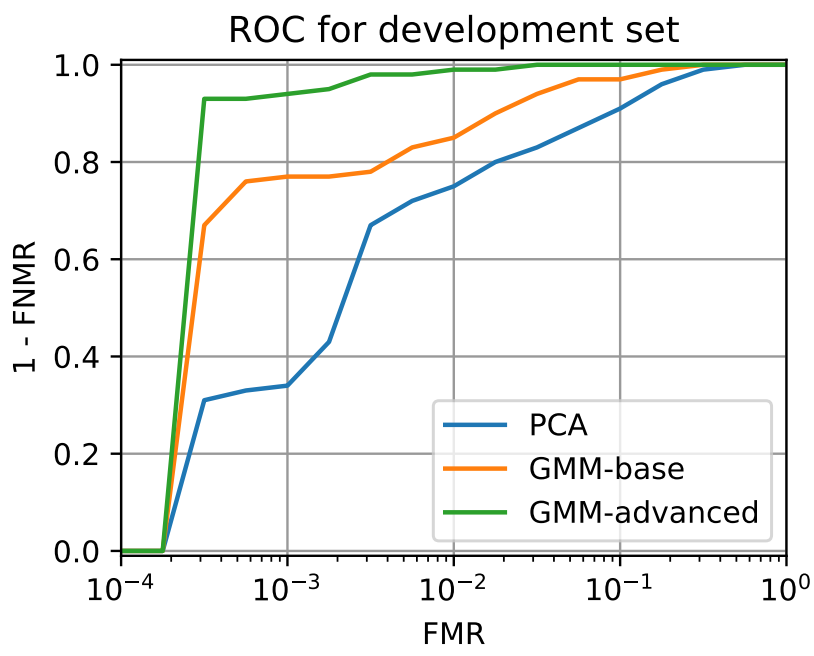
- Compare different feature parameters
- Change photometric enhancement
- Change image resolution
- Try out ISV or JFA extensions

Hands on: Gaussian Mixture Models

Possible CMC



Possible ROC



idiap RESEARCH INSTITUTE

bob

UCCS Vision And Security Technology
University of Colorado Colorado Springs

Outline

Session Variability Modeling

GMM Mean Super-Vector Notation

$$\vec{m} = \begin{bmatrix} \vec{\mu}_1 \\ \vec{\mu}_2 \\ \vdots \\ \vec{\mu}_M \end{bmatrix} \in \mathbb{R}^{M \cdot d} \qquad \Sigma = \begin{bmatrix} \Sigma_1 & 0 & \cdots & 0 \\ 0 & \Sigma_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_M \end{bmatrix} \in \mathbb{R}^{M \cdot d \times M \cdot d}$$

Mean-only MAP Adaptation

$$\vec{m}^c = \vec{m} + \vec{d}^c \in \mathbb{R}^{M \cdot d}$$

Session Variability Modeling

Inter-Session Variability Modeling (ISV)

Suppress session variations (illumination, channels, ...)

Estimate Session Offset per Sample j

$$\vec{m}^c = \vec{m} + \vec{u}_j^c + \vec{d}^c = \vec{m} + \mathrm{U}\vec{h}_j^c + \vec{d}^c$$

Training of Session Variability Matrix U using Class Labels

Expectation Maximization (EM)

Test Probe Features $\{\vec{x}_1^p, \ldots, \vec{x}_{N^p}^p\}$ with Model $\vec{m}^c$

Estimate latent variables $\vec{h}_n^c$ and $\vec{h}_n$ and:

$$s = \sum_{n=1}^{N^p} \log \frac{P(\vec{x}_n^p \mid \vec{m}^c + \mathrm{U}\vec{h}_n^c)}{P(\vec{x}_n^p \mid \vec{m} + \mathrm{U}\vec{h}_n)}$$

Session Variability Modeling

Total Variability Modeling (TV)

- Estimate GMM ν that best describes $\{\vec{x}_1, \ldots, \vec{x}_N\}$:

$$\nu = \vec{m} + \mathrm{T}\vec{w}$$

- TV-Matrix $\mathrm{T} \in \mathbb{R}^{M \cdot d \times r}$: Linear subspace of variations
- i-vector $\vec{w} \in \mathbb{R}^r$: Estimated variation

Training of TV-Matrix T w/o Class Labels

Expectation Maximization (EM)

TV-Matrix as Feature Extractor

Distance (e.g. cosine) between i-vectors $\vec{w}^c$ and $\vec{w}^p$

Outline

4 Extensions of `bob.bio`

- Writing new Tools
  - CSU Face Recognition Resources
  - Video Face Recognition
- Deep Convolutional Neural Networks
  - Training DCNNs
  - Using a Pretrained DCNN in `bob.bio`
- Hands on: DCNN Features
  - Command Line
  - Evaluation

More Algorithms

Writing new Tools

- Database
  `bob.bio.base.BioDatabase`
- FileList database
  `bob.bio.base.database.FileListBioDatabase`
- Preprocessor
  `bob.bio.base.preprocessor.Preprocessor`
- Extractor
  `bob.bio.base.extractor.Extractor`
- Algorithm
  `bob.bio.base.algorithm.Algorithm`

LRPCA and LDA-IR

CSU Face Recognition Resources

- Provides two baseline algorithms
  - → Local Region PCA (LRPCA)
  - → LDA on color channels (LDA-IR) aka. CohortLDA
- Evaluated on Good, Bad & Ugly (GBU) database

bob.bio.csu

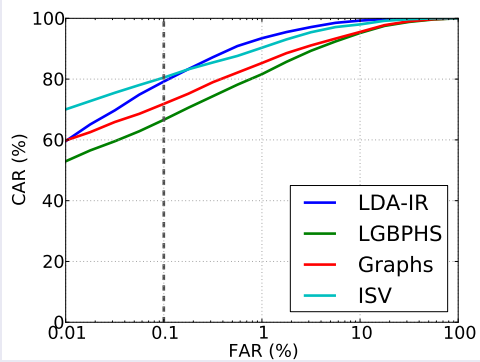
- Installation instructions `bob.bio.csu`
- Implements wrappers for all tools
  - → Preprocessor `bob.bio.csu.preprocessor`
  - → Extractor `bob.bio.csu.extractor`
  - → Algorithm `bob.bio.csu.algorithm`
- Applicable to **all implemented** databases

CSU Face Recognition Resources

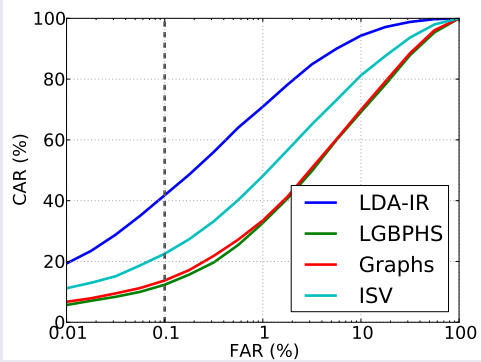
Günther et al. (2012)

"An Open Source Framework for Standardized Comparisons of Face Recognition Algorithms"

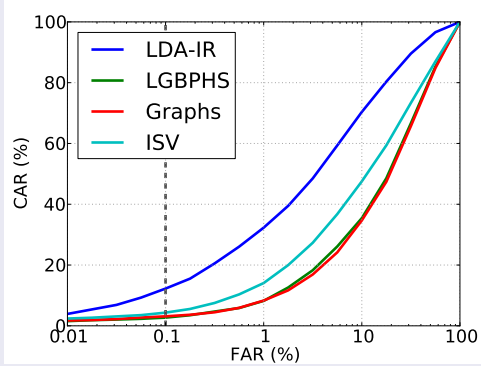
GBU – Good



GBU – Bad



GBU – Ugly



BANCA

	LDA-IR	LGBPHS	Graphs	ISV
EER	26.2%	13.2%	11.7%	10.0%
HTER	27.2%	16.1%	12.4%	10.9%

Outline

Video Face Recognition

Use Face Recognition Algorithms in Videos

- `bob.bio.video.FrameSelector`
  - → Extract frames from videos
- `bob.bio.video.FrameContainer`
  - → Data I/O using HDF5
- `bob.bio.video.preproecssor.Wrapper`
  - → Preprocess frames individually
- `bob.bio.video.extractor.Wrapper`
  - → Extract features for each frame independently
- `bob.bio.video.algorithm.Wrapper`
  - → Enroll from several frames (of several videos)
  - → Probe with several frames of one video

Video Face Recognition

Example Configuration File video_gmm.py

```
import bob.bio.face
import bob.bio.video

database = "mobio-video"
protocol = "male"

preprocessor = bob.bio.video.preprocessor.Wrapper(
    preprocessor = 'face-detect',
    frame_selector = bob.bio.video.FrameSelector(20, 'spread')
)

extractor = bob.bio.video.extractor.Wrapper(
    extractor = bob.bio.face.extractor.DCTBlocks(12,0,45),
    frame_selector = bob.bio.video.FrameSelector(selection_style='all')
)

algorithm = bob.bio.video.algorithm.Wrapper(
    algorithm = 'gmm',
    frame_selector = bob.bio.video.FrameSelector(selection_style='all')
)

sub_directory = "gmm_MOBIO_video"
```

Example Command Line

```
$ verify_gmm.py video_gmm.py --parallel 4 -vv
```

idiap RESEARCH INSTITUTE

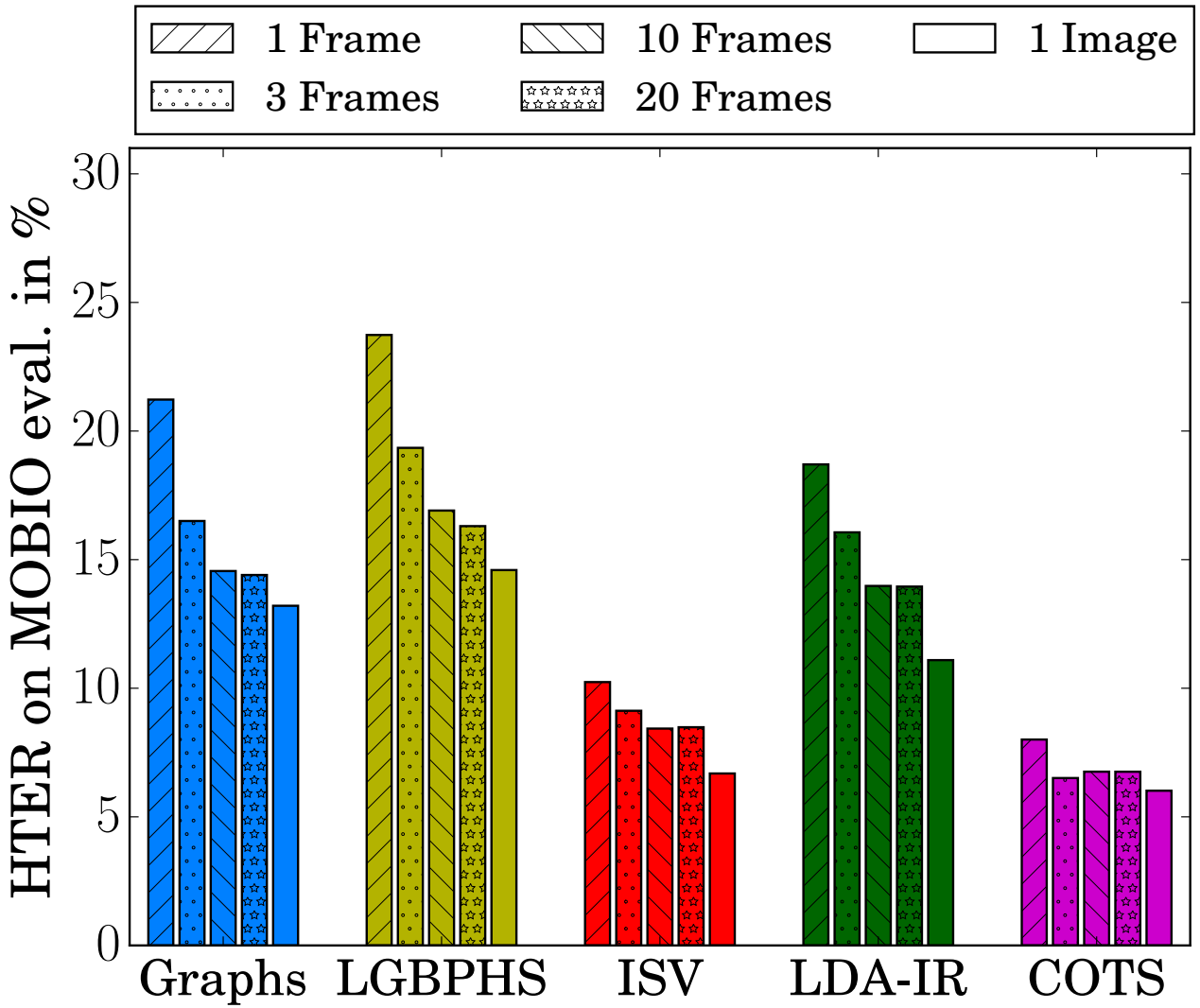
rity Technology
o Colorado Springs

The MOBIO Dataset

Examples of the MOBIO Dataset

Video FR (Günther et al., 2016)

Algorithm Performance with Frames



idiap RESEARCH INSTITUTE

rity Technology
o Colorado Springs

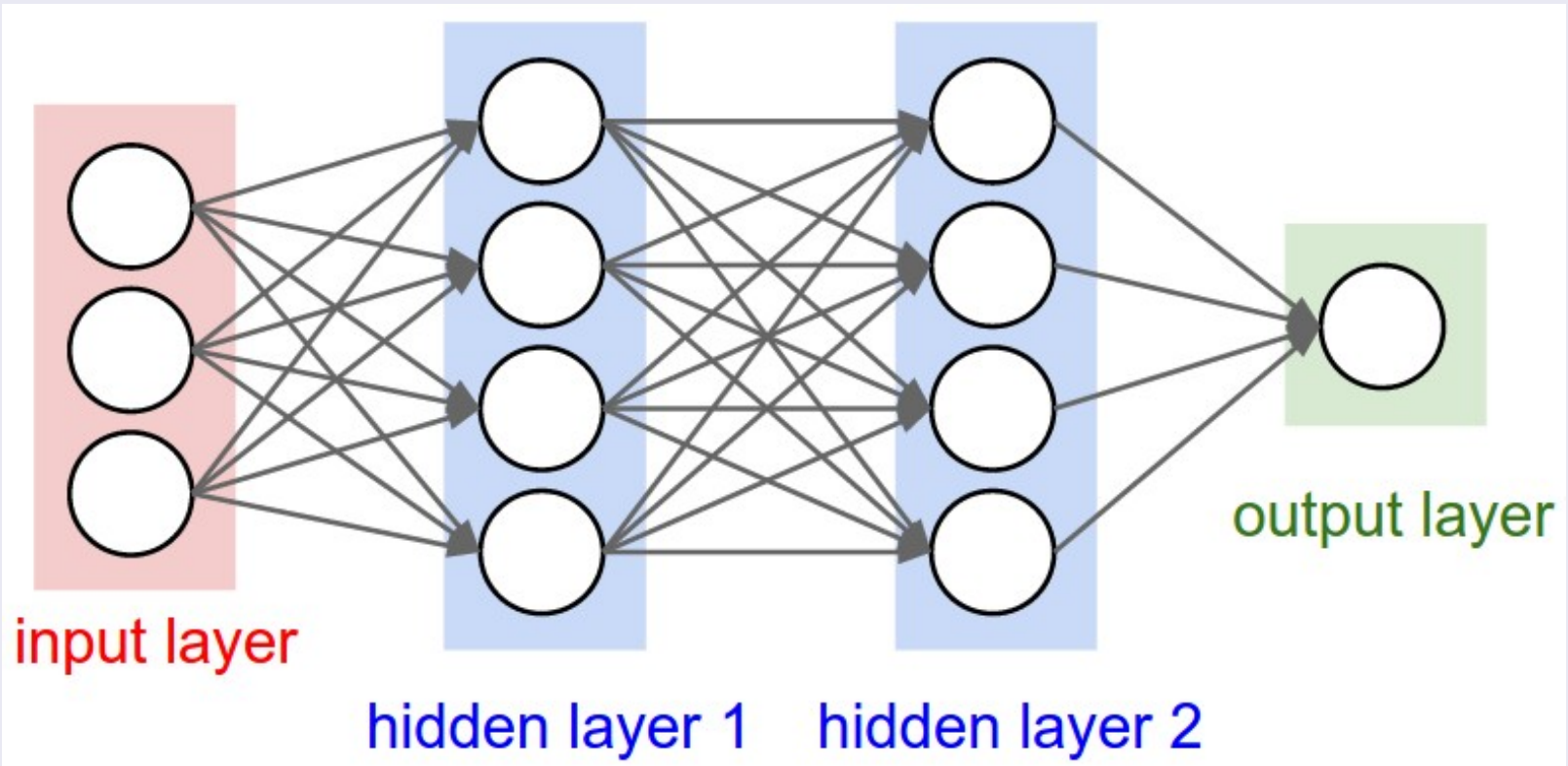
Outline

Deep Convolutional Neural Networks

Layered Network Topology

$$\mathcal{M}(\mathcal{I}) = \mathcal{M}^{(L)}(\mathcal{M}^{(L-1)}(\cdots(\mathcal{M}^{(2)}(\mathcal{M}^{(1)}(\mathcal{I})))\cdots))$$

Feature Extraction using DCNN (link)



Deep Convolutional Neural Networks

Network Layer Types

- Convolution layers: 2D $\Rightarrow$ 2D

$$\mathcal{M}^{(l)}(x,y) = \sum_{u,v} w_{uv}^{(l)} \cdot \mathcal{M}^{(l-1)}(u-x, v-y)$$

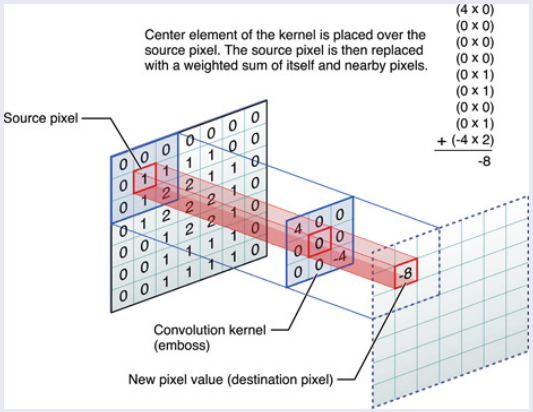
- Fully connected layer: 1D $\Rightarrow$ 1D

$$\mathcal{M}^{(l)}(k) = \sum_{j} w_{kj}^{(l)} \cdot \mathcal{M}^{(l-1)}(j)$$

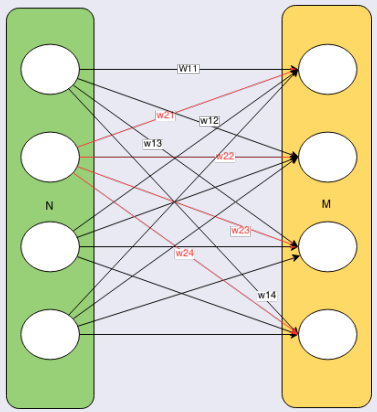
- Activation function: 1D $\Rightarrow$ 1D; no weights

$$\mathcal{M}^{(l)}(k) = \phi(\mathcal{M}^{(l-1)}(k))$$

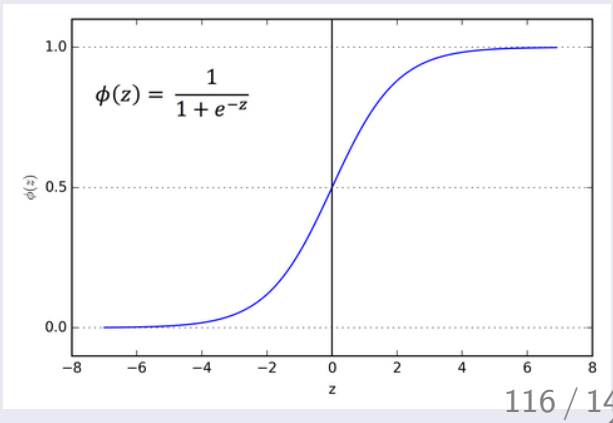
Conv. (link)



Fully (link)

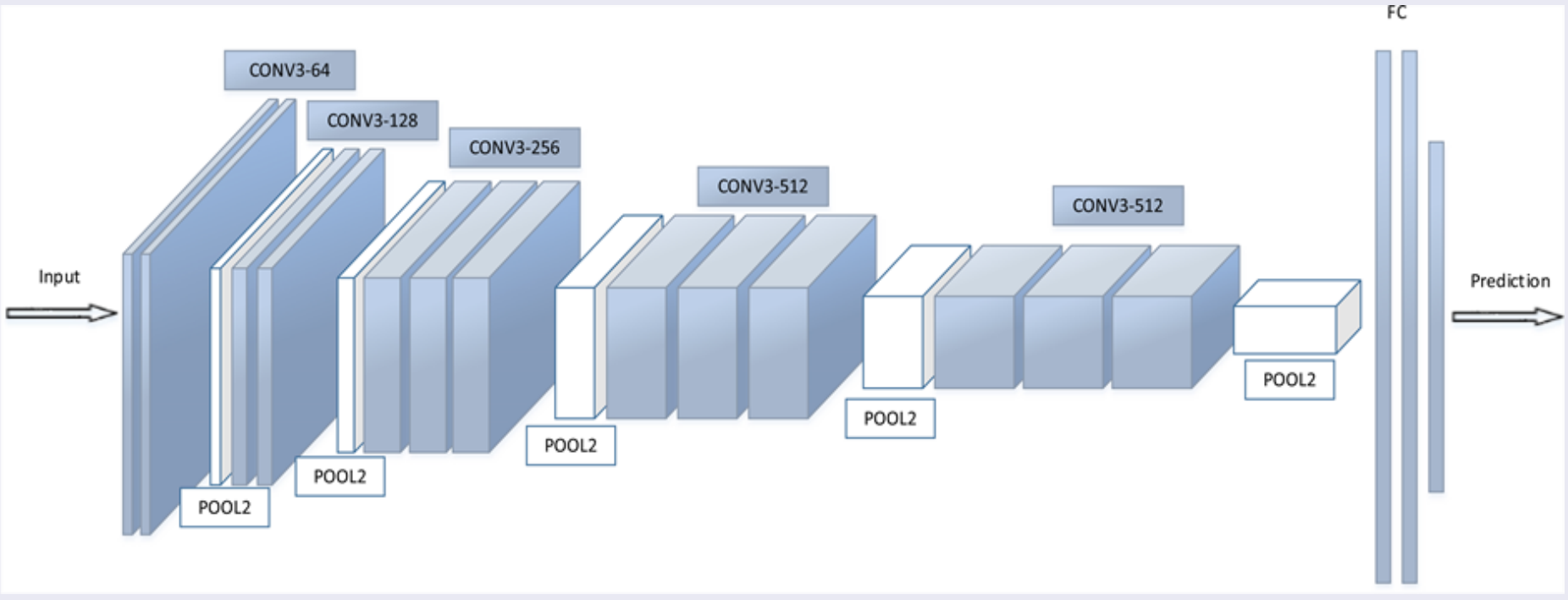


Activation (link)



Deep Convolutional Neural Networks

Feature Extraction: VGG Network (Parkhi et al., 2015)



Deep Convolutional Neural Networks

Network Training

- Training images: $\mathcal{D} = \{\mathcal{I}_1, \ldots, \mathcal{I}_N\}; \mathcal{I}_n \in \mathbb{R}^{W \times H}$
- According labels: $\mathcal{T} = \{t_1, \ldots, t_N\}; t_n \in \{1, \ldots, C\}$
- Loss function, e.g., $J = \sum_{n=1}^{N} (\mathcal{M}(\mathcal{I}_n) - t_n)^2$

Unary Label Representation

- Last layer $\mathcal{M}^{(L)}$ with C outputs
- Represent each class by $C - 1$ zeros and 1 one

Deep Convolutional Neural Networks

Network Training via Gradient Descent

- Compute gradient $\nabla_{kj}^{(L)}$ for image/label pair
- Chain rule (Backpropagation)

$$\nabla_{kj}^{(L)} = \frac{\partial J}{\partial w_{kj}^{(L)}} = \frac{\partial J}{\partial \mathcal{M}^{(L)}} \cdot \frac{\partial \mathcal{M}^{(L)}}{\partial w_{kj}^{(L)}}$$

$$\nabla_{kj}^{(L-1)} = \frac{\partial J}{\partial w_{kj}^{(L-1)}} = \frac{\partial J}{\partial \mathcal{M}^{(L)}} \cdot \frac{\partial \mathcal{M}^{(L)}}{\partial \mathcal{M}^{(L-1)}} \cdot \frac{\partial \mathcal{M}^{(L-1)}}{\partial w_{kj}^{(L-1)}}$$

- Example: Fully connected

$$\frac{\partial \mathcal{M}^{(l)}}{\partial w_{kj}^{(l)}} = \frac{\sum_{j'} w_{kj'}^{(l)} \cdot \mathcal{M}^{(l-1)}(j')}{\partial w_{kj}^{(l)}} = \mathcal{M}^{(l-1)}(j)$$

- Update weights: $w_{kj}^{(l)} \mathrel{-}= \eta \nabla_{kj}^{(l)}$

Deep Convolutional Neural Networks

How to Train a Deep Network

1. Design network topology
2. Label **tons** of training data
3. Buy 1-100 **powerful** GPUs
4. Prepare training data
5. Start network training
6. Go for a trip to **Hawaii**

Outline

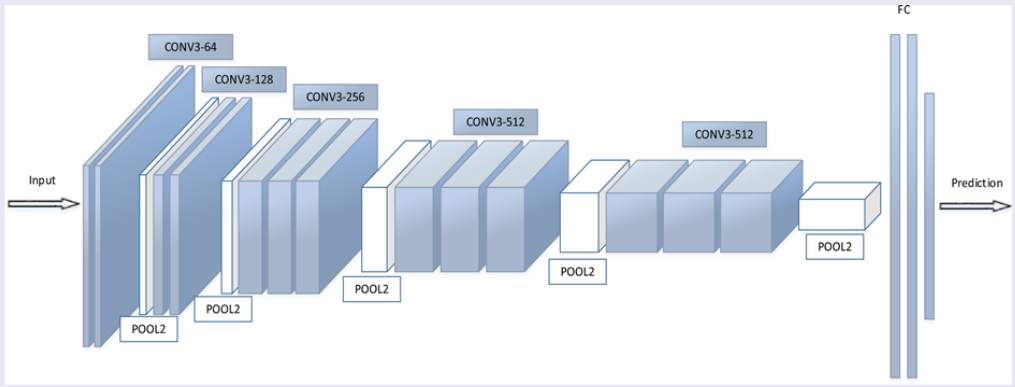
Deep Convolutional Neural Networks

Characteristics

Parkhi et al. (2015)

- Image resolution: 224×224 RGB
- Feature vector `@fc7`: 4096 `float`
- Training set: 982,803 images of 2,622 identities

VGG Network



Triplet Loss

$$\min_{W} \sum_{(a,p,n)\in T} \max\{0, \alpha - \|\vec{z}_a - \vec{z}_n\|^2 + \|\vec{z}_a - \vec{z}_p\|^2\}$$

$$\vec{z} = W \frac{\mathrm{VGG}(\mathcal{I})}{\|\mathrm{VGG}(\mathcal{I})\|}$$

idiap RESEARCH INSTITUTE bob

UCCS Vision And Security Technology University of Colorado Colorado Springs

Using a Pretrained DCNN in `bob.bio`

Implementation Details

- Framework: Caffe
  `http://caffe.berkeleyvision.org`
- Can use GPU (here CPU only)
- Two files:
  - → `.prototxt`: Network description (text)
  - → `.caffemodel`: Network parameters (binary)
- Provides Python interface:
  `caffe.Net(".caffemodel", ".prototxt", caffe.TEST)`
  - → Not very well documented

Using a Pretrained DCNN in bob.bio

Shallow Network Wrapper Class (VGG.py)

```
class Network (bob.bio.base.extractor.Extractor):
    __init__(
        network_file,
        network_proto,
        output_layer,
        input_layer = 'data',
        mean_values = None
    )

    load()
    __call__(data) -> feature
```

Outline

Hands on: DCNN Features

VGG Configuration (atnt_vgg.py)

```
import bob.bio.face
from .VGG import Network
import numpy

preprocessor = bob.bio.face.preprocessor.FaceCrop(
    cropped_image_size = (224,224),
    cropped_positions  = {"reye" : (112, 86), "leye" : (112, 138)},
    fixed_positions    = {"reye" : (50, 26),  "leye" : (50, 64)},
    dtype              = numpy.float32,
    color_channel      = "rgb"
)

extractor = Network(
    network_caffemodel = "./vgg_face_caffe/VGG_FACE.caffemodel",
    network_prototxt   = "./vgg_face_caffe/VGG_FACE_deploy.prototxt",
    output_layer       = "fc7",
    mean_values        = [129.1863, 104.7624, 93.5940]
)
```

Hands on: DCNN Features

Face Recognition with DCNN features

- Database: AT&T database of faces `"atnt"`
- Preprocessor: Color images in 224×224 (see above)
- Features: VGG deep features 4096-D (see above)
- Algorithm: Distance `"distance-euclidean"`

Experiment Command Line

```
$ verify.py atnt_vgg.py --parallel 4
```

Hands on: DCNN Features

Basic Experiments

- Test different distance functions
- Try different output layers (see `.prototxt`)

Advanced Experiments

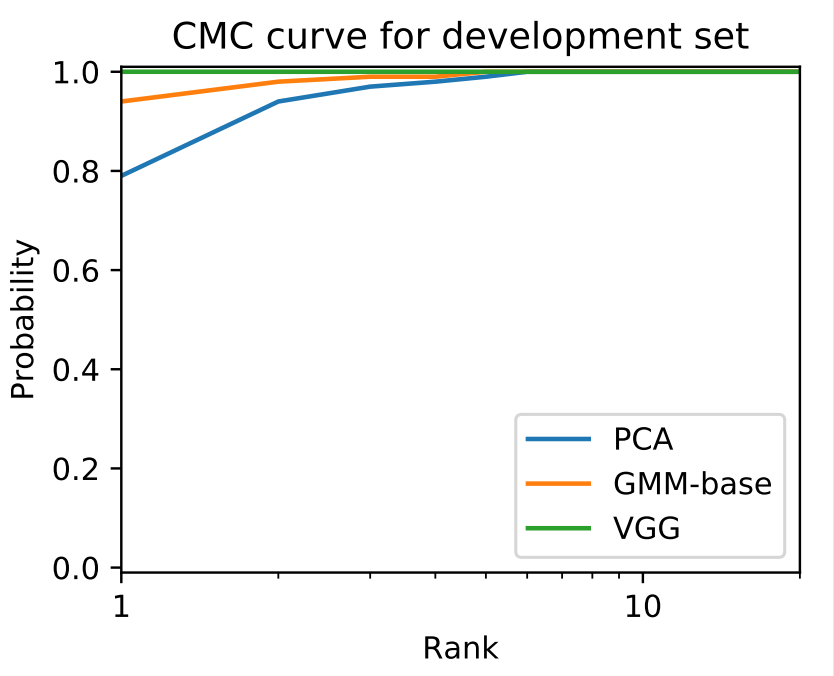
- Use different algorithm (e.g., PCA)
  - → Re-use extracted features from previous experiments
- Optimize face alignment (see `atnt_vgg.py`)

Expert Experiments

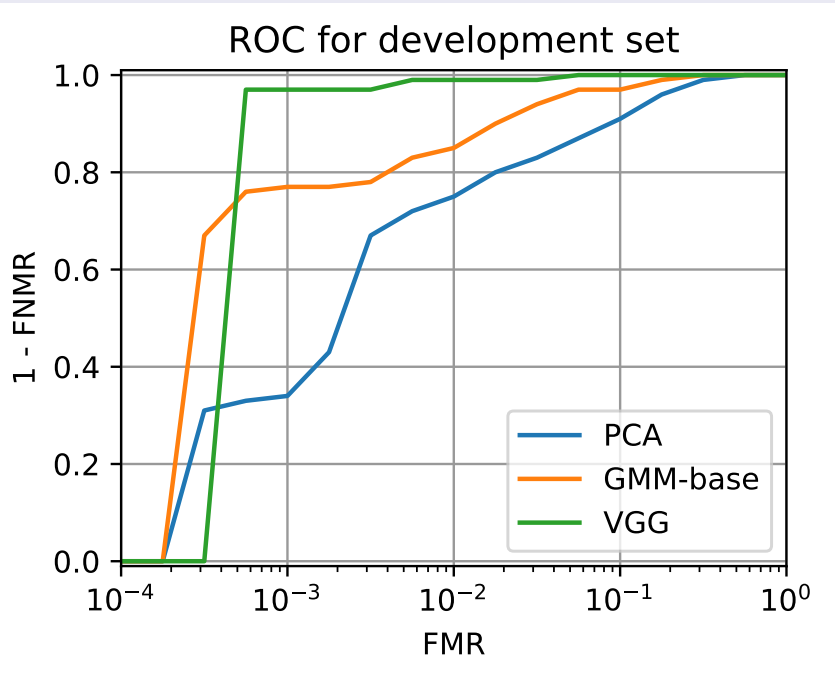
- Use facial landmark detector for alignment
- Implement your own algorithm
  - → Enroll by averaging deep features

Hands on: DCNN Features

Expected CMC



Expected ROC



idiap RESEARCH INSTITUTE

bob

UCCS Vision And Security Technology
University of Colorado Colorado Springs

Parenthesis: Classification of the Results

Amount of Training Data and Training Time

- PCA and GMM trained on AT&T training set
  - → 200 images of 20 identities, some minutes
- VGG trained on external dataset
  - → 982,803 images of 2,622 identities, several days

Dataset and Evaluation Criteria

- AT&T database small, outdated (Jain and Li, 2005)
  - → Easy recognition of 20 identities, no low FMR
- Only small frontal images with little variations
  - → Frontal face recognition is considered solved
- Enrollment from 5 images, scoring with one image
  - → No image/image or template/template comparison
- Closed-set recognition, pre-cropped faces

Outline

Reproducible Research

What is Required for Reproducible Research

- A **paper** that describes your work **in all relevant details**
- Some **code** to reproduce all results
- The **data** required to reproduce the results
- Detailed **instructions** on how to **apply** the *code* on the *data* to **replicate** the results of the *paper*

Reproducible Research

Levels of Reproducibility (Vandewalle 2009)

0. Irreproducible
1. Cannot seem to reproduce
2. Reproducible, with extreme effort ($>$ 1 month)
3. Reproducible, with considerable effort ($>$ 1 week)
4. Easily reproducible ($\sim$ 15 min.), but requires proprietary software (e.g. Matlab)
5. **Easily reproducible ($\sim$ 15 min.), only free software**

Reproducible Research

Why Reproducible Research?

Providing source code and data requires some time . . .

Boost your **Research Impact**! (Vandewalle 2012)

- **Lower entrance barrier** to your publications
- Stand out current trends:
  - → Only **10 % of TIP** papers provide source code (reproducibility level ≥ 2)
- Statistically, RR work is **more valuable**:
  - → **13 / 15 most cited** TPAMI or TIP papers provide code
  - → Multiply number of citations **by a factor of 7**

Why Reproducible Research?

Productivity Boost of your Research Team

- Start from the **state-of-the-art**
- Start to **work fast**
- **Collaborate** with external partners
- Create something **durable**
- **Avoid** re-doing things
- You or your team members:
  - → Improve source code
  - → Test it
  - → Document it
  - → Keep it operational
- For the **benefit** of **all team members**

How to Provide Reproducible Research

Requirements of Reproducible Research (Anjos et al., 2017)

1. Repeatable: Re-run experiments, obtain same results
   → Configuration file
2. Shareable: Possible to share data and code
   → Open-source code and data
3. Extensible: Easy to implement new research
   → Bob's biometric recognition framework
4. Stable: Guaranteed repeatability through time
   → Local environments with conda and pin-pointed versions

$\Rightarrow$ Mostly fulfilled by Bob and `bob.bio`

How to Publish Reproducible Research

Steps to Publish your Research (Anjos et al., 2017)

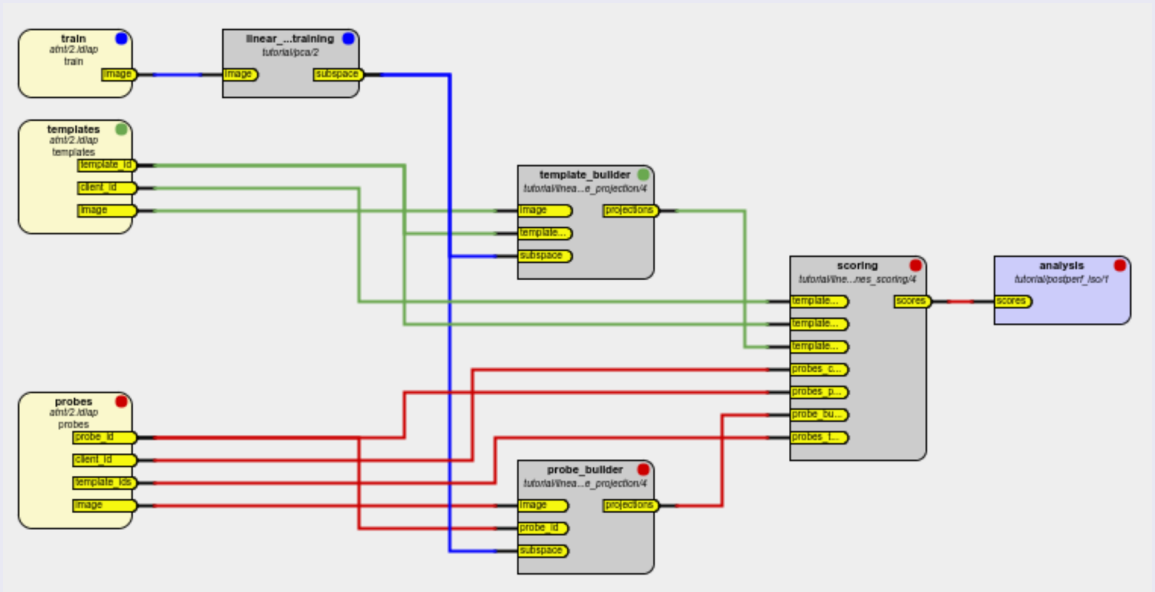
1. **Design** your experiment reproducible
   - → Avoid proprietary software (Matlab), use free libraries
   - → One experiment = one environment = one code package
   - → Keep track of required software, see `bob.yml`
2. Use a **version control system** (SVN, git, . . .)
   - → Add **tags**, e.g., `submitted_to_conference_X`
   - → Track issues and share with your team
3. Implement **unit tests**, run them for each change
4. Provide **documentation**
   - → Installation instructions
   - → Copy-and-paste command lines
5. Version, package and publish
   - → The Python Package Index (PyPI)

How to Publish Reproducible Research

Biometric Evaluation And Testing (BEAT)

- Web-based platform to evaluate biometric algorithms
- Open-source toolchains and algorithms
- Attestation of results
  - → Link paper with implementation

Toolchain Interface



Implementation

```
import bob
import numpy


class Algorithm:

    def __init__(self):
        self.data = []


    def setup(self, parameters):
        self.number_of_components = parameters['number-of-components']
        return True


    def process(self, inputs, outputs):
        self.data.append(inputs["image"].data.value.astype('float64').flatten())

        if not(inputs.hasMoreData()):
            self.data = numpy.vstack(self.data)

            trainer = bob.trainer.PCATrainer()

            pca_machine, eigen_values = trainer.train(self.data)

            pca_machine.resize(pca_machine.shape[0], int(self.number_of_components))

            # outputs data
            outputs["subspace"].write({
                'input_subtract': pca_machine.input_subtract,
                'input_divide':   pca_machine.input_divide,
                'weights':        pca_machine.weights,
                'biases':         pca_machine.biases,
            })

        return True
```

Escape the Vicious Circle! (link)

The End

Topics

- PCA:
  Turk and Pentland (1991)
- GMM, EM:
  Bishop (2006)
- UBM-GMM:
  Reynolds et al. (2000)
- ISV:
  Vogt and Sridharan (2008)
- DCNN:
  Bishop (2006); Parkhi et al. (2015)
- Evaluation:
  Jain and Li (2005)
- Bob:
  Anjos et al. (2012)
- bob.bio:
  Günther et al. (2012, 2016)
- Reproducible Research:
  Vandewalle et al. (2009); Vandewalle (2012); Anjos et al. (2017)

References

Anjos, A., El Shafey, L., Wallace, R., Günther, M., McCool, C., and Marcel, S. (2012). Bob: a free signal processing and machine learning toolbox for researchers. In *ACMMM*.

Anjos, A., Günther, M., de Freitas Pereira, T., Korshunov, P., Mohammadi, A., and Marcel, S. (2017). Continuously reproducing toolchains in pattern recognition and machine learning experiments. In *ICML*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Günther, M., El Shafey, L., and Marcel, S. (2016). Face recognition in challenging environments: An experimental and reproducible research survey. In *Face Recognition Across the Imaging Spectrum*. Springer.

Günther, M., Wallace, R., and Marcel, S. (2012). An open source framework for standardized comparisons of face recognition algorithms. In *ECCV Workshops and Demonstrations*.

Jain, A. K. and Li, S. Z. (2005). *Handbook of Face Recognition*. Springer.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In *BMVC*.

Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing*.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*.

Vandewalle, P. (2012). Code sharing is associated with research impact in image processing. *Computing in Science Engineering*.

Vandewalle, P., Kovacevic, J., and Vetterli, M. (2009). Reproducible Research in Signal Processing - What, why, and how. *IEEE Signal Processing Magazine*.

Vogt, R. and Sridharan, S. (2008). Explicit modelling of session variability for speaker verification. *Computer Speech and Language*.