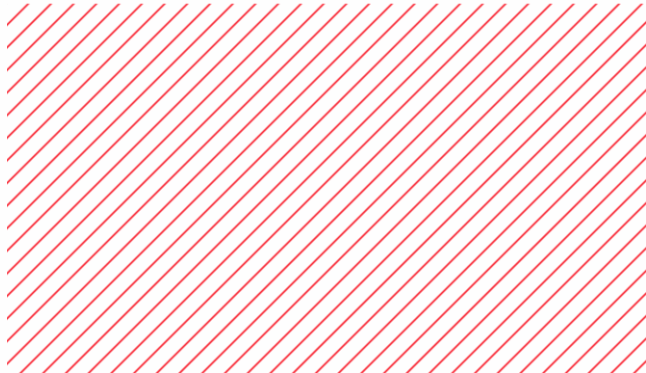
академия больших данных

mail.ru group

# Metric learning

Эдуард Тянтов

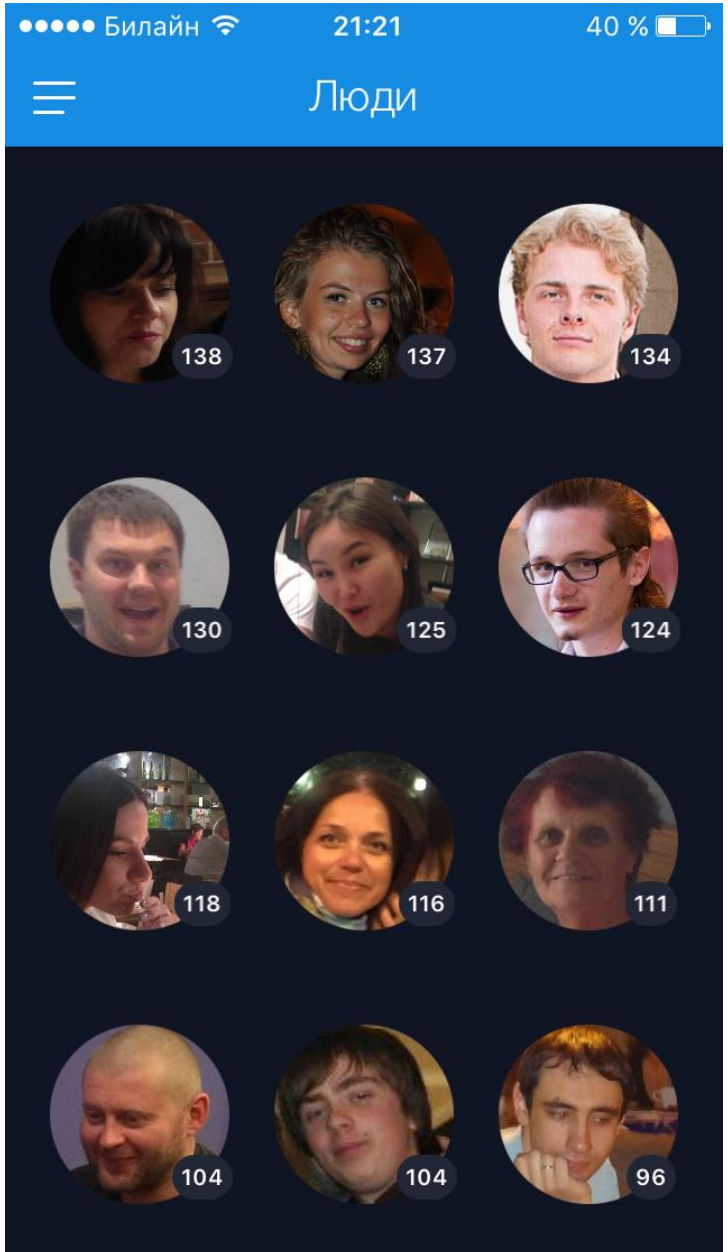
Руководитель направления машинного обучения
в Почте и Портале

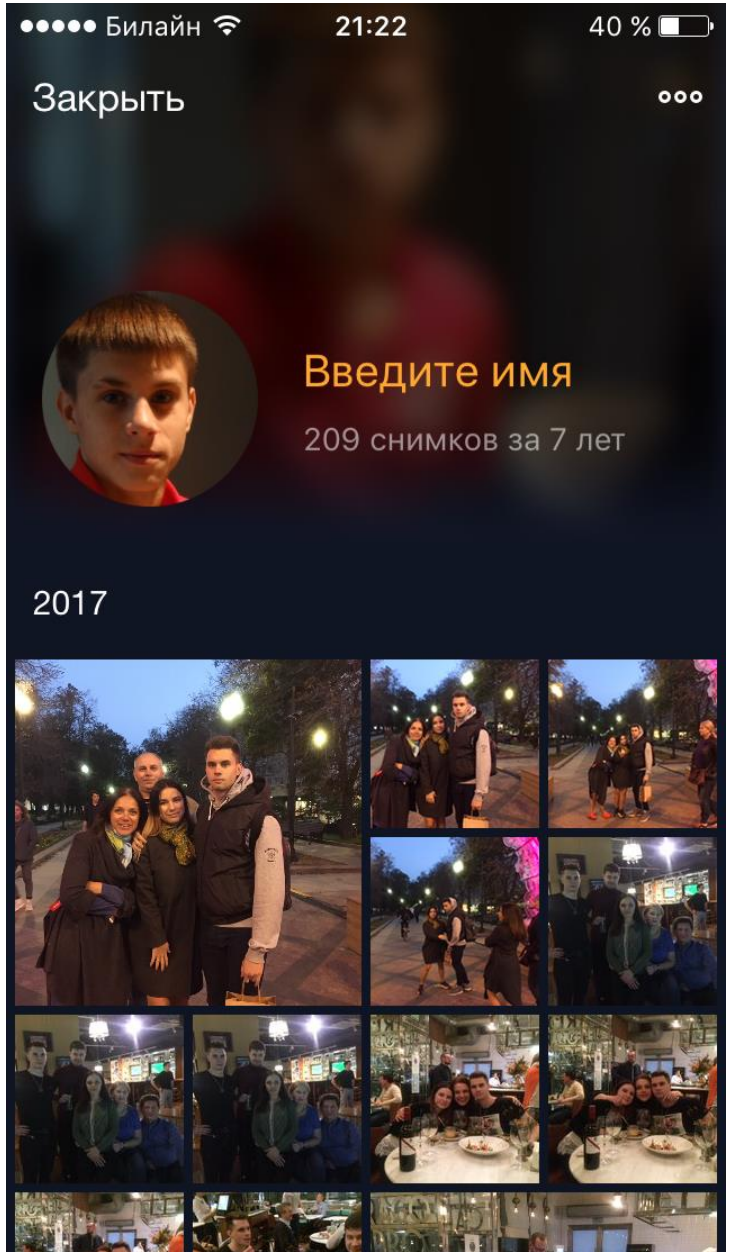


# Case study: FR at Cloud@Mail.ru

mail.ru group

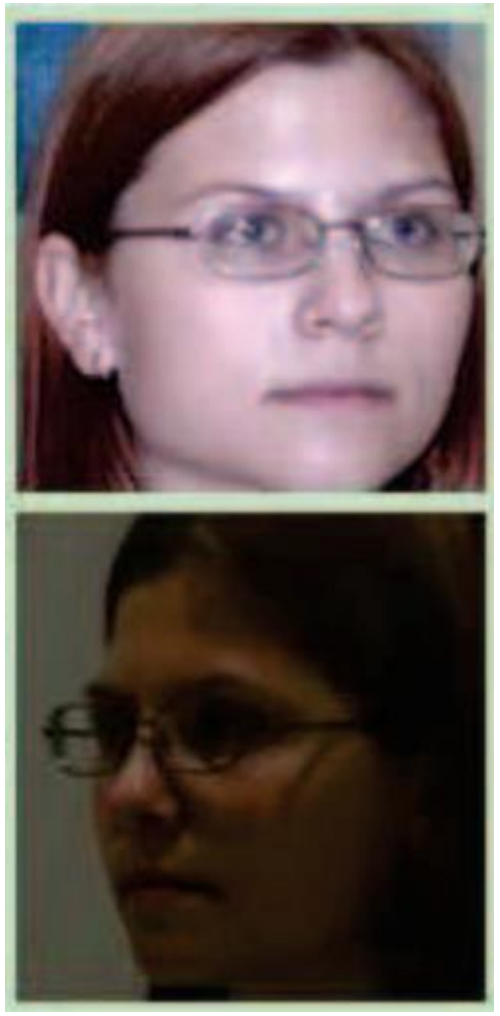
1. Users upload photos to Cloud
2. Backend identifies persons on photos, tags and show clusters



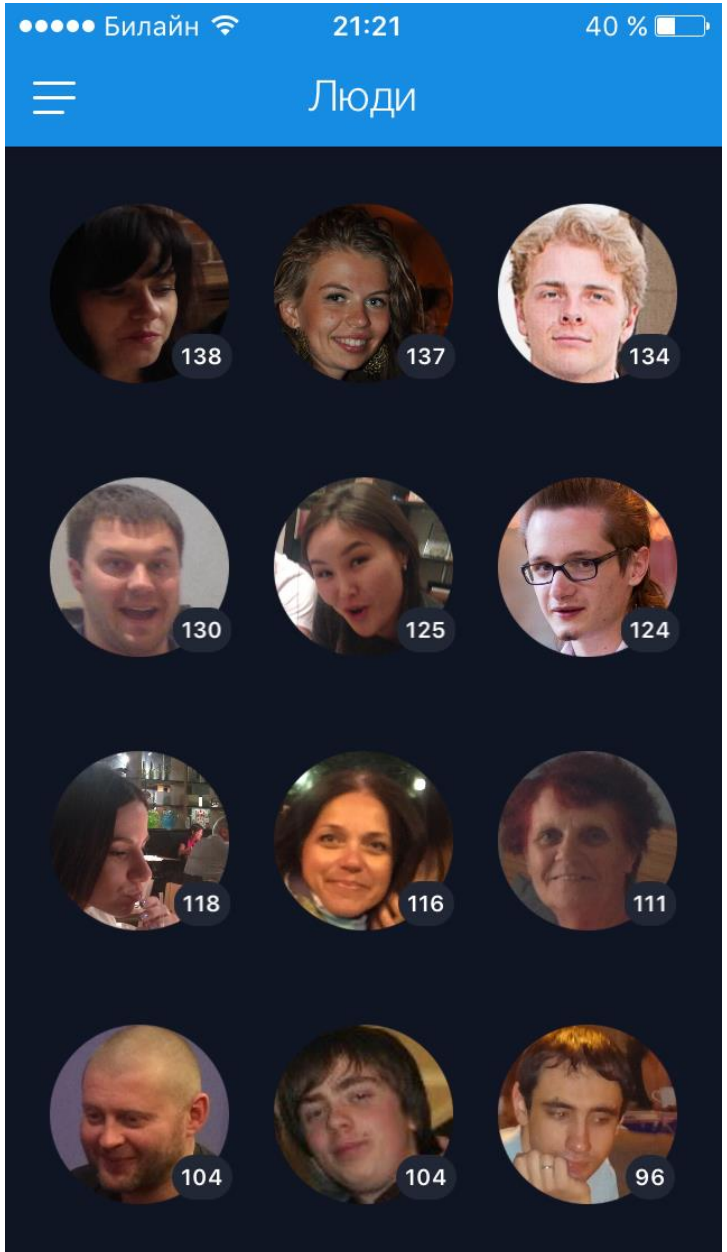
# На примере Face Recognition

Same person ?



# Components/Plan

1. Face detection
2. Face recognition
    1. Metric learning
    2. Data cleaning
    3. Edge cases
3. Search (AKNN)

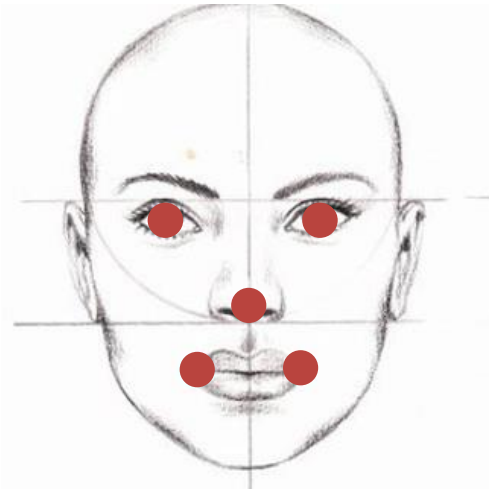


# Face Detection (briefly)

# Face detection

# Auxiliary task: facial landmarks

- Face alignment: rotation
- Goal: make it easier for Face Recognition



Rotate

# Train Datasets

## Wider

- 32k images
- 494k faces

## Celeba

- 200k images, 10k persons
- Landmarks, 40 binary attributes

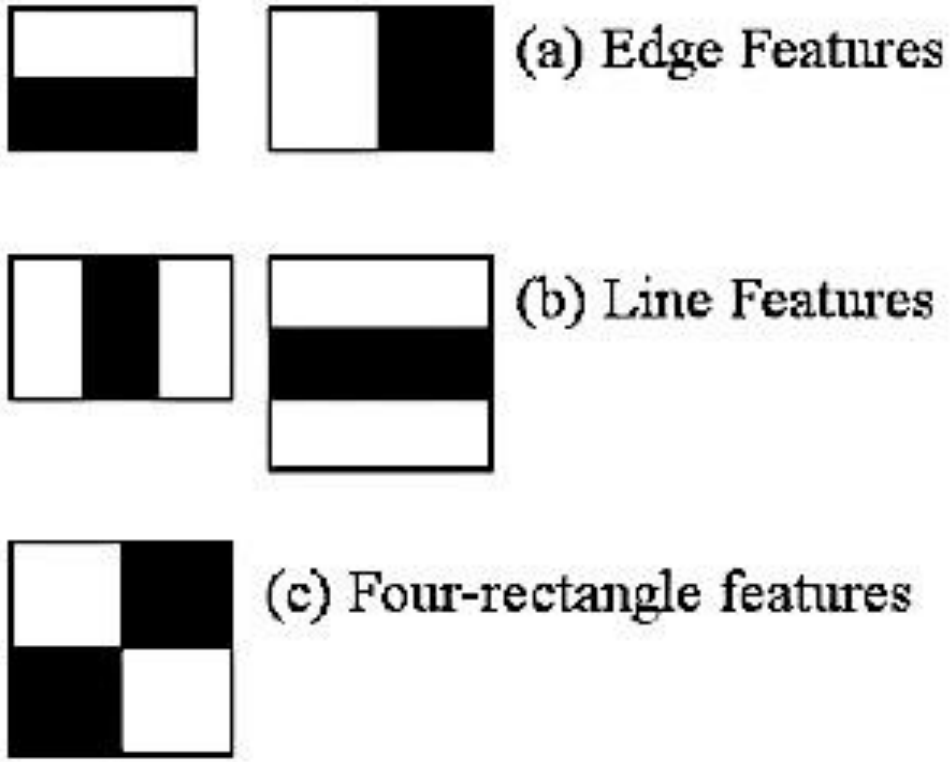
# Подходы к решению задач

1. Классический CV (**HOG**, SIFT, HAAR, …)
2. Convolutional neural networks
    1. Region-based (RCNN)
    2. Single-shot (SSD)
    3. Cascaded (MTCNN)

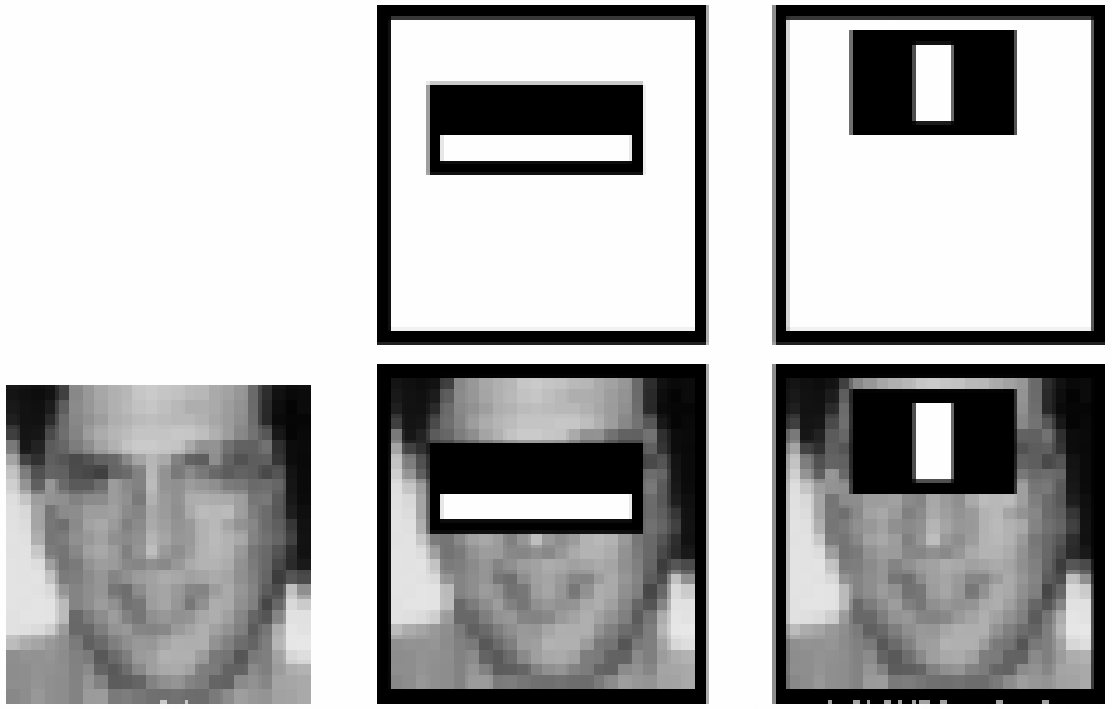
# Old school: Viola-Jones

Haar Feature-based Cascade Classifiers

Haar-like features




Examples



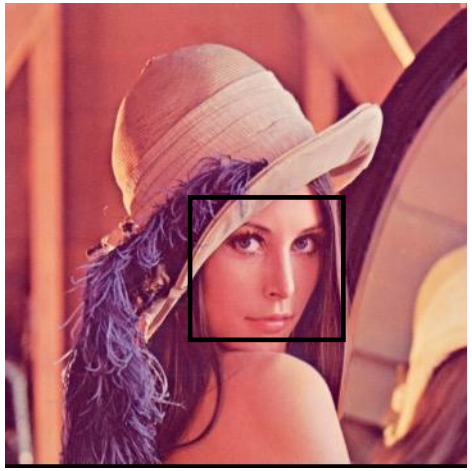
eyes darker

nose lighter

# Viola-Jones algorithm: training

features

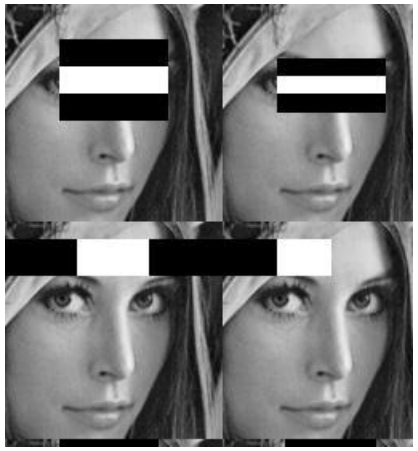
for each patch



160k

AdaBoost ensemble

valuable



6k

weighted sum

Dataset

Face or Not

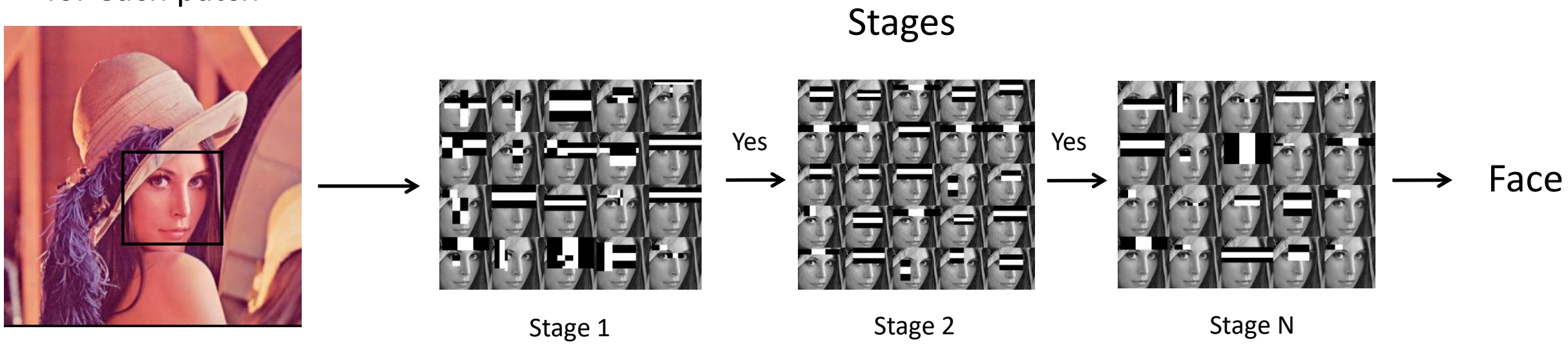
# Viola-Jones algorithm: inference

mail.ru group

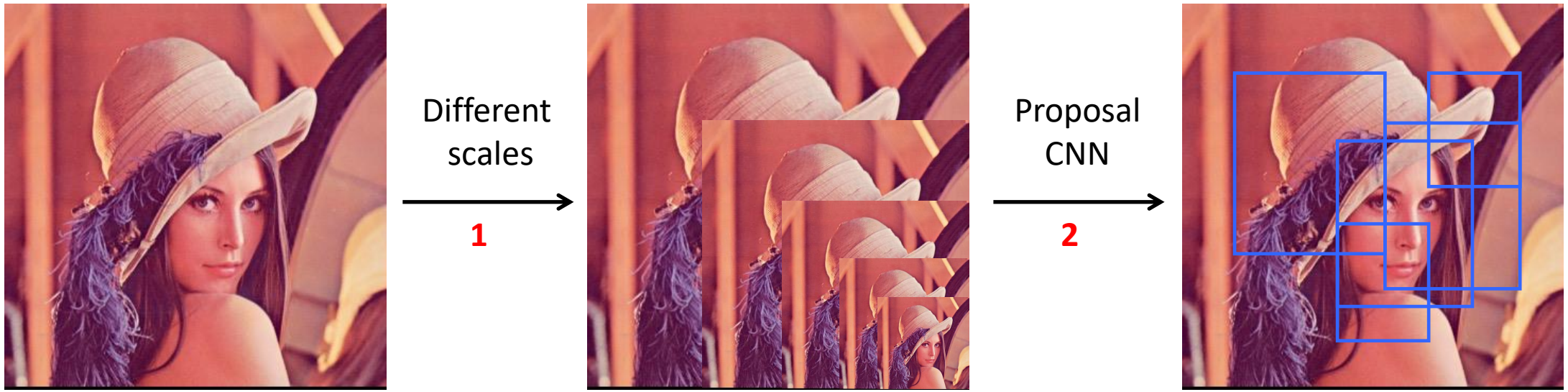
## Optimization

- Features are grouped into stages
- If a patch fails at any stage –> discard

for each patch



# MTCNN

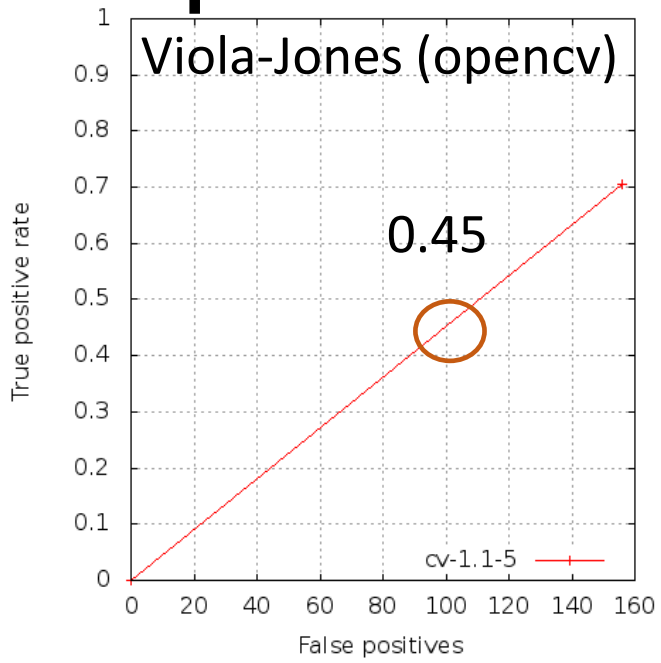


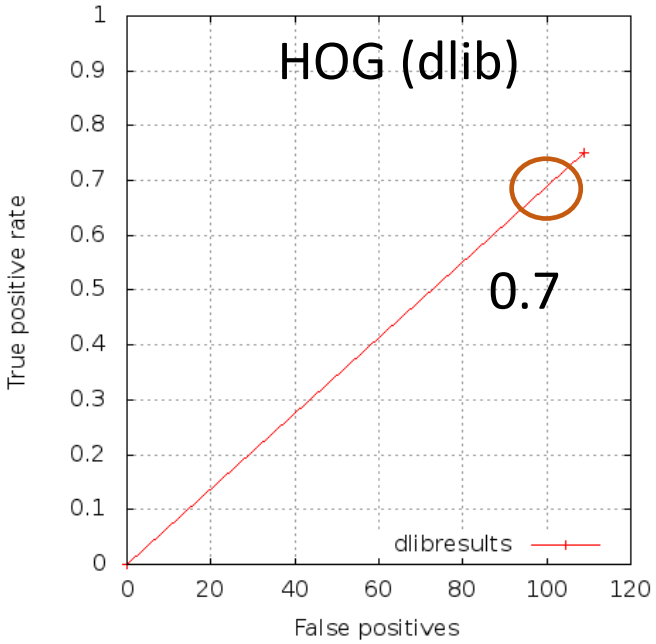

**Cascade of 3 CNN**

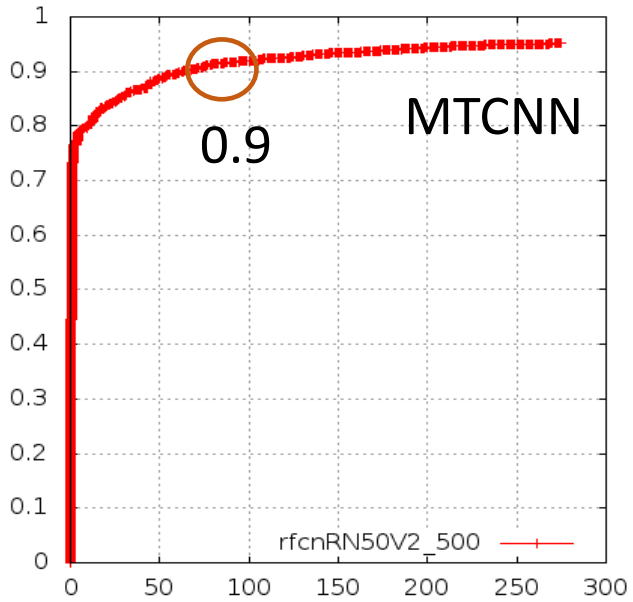
4. Output -> b-boxes + landmarks

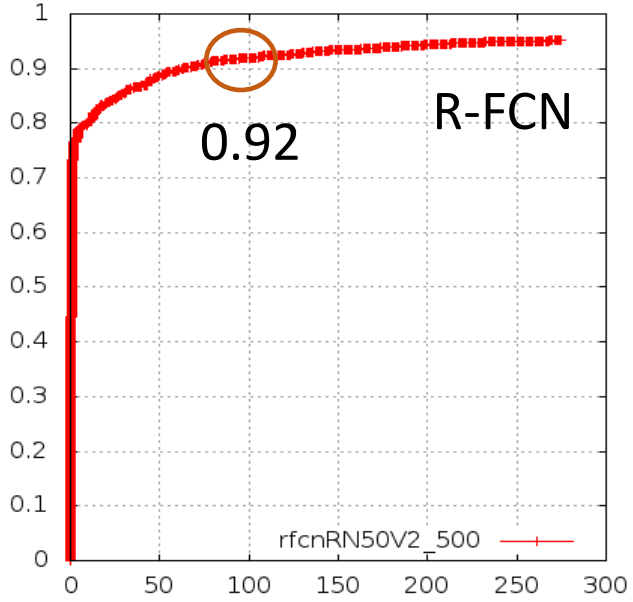
# Comparison

FDDB results



# Comparison: MTCNN vs R-FCN

mail.ru group

**MTCNN is the choice**

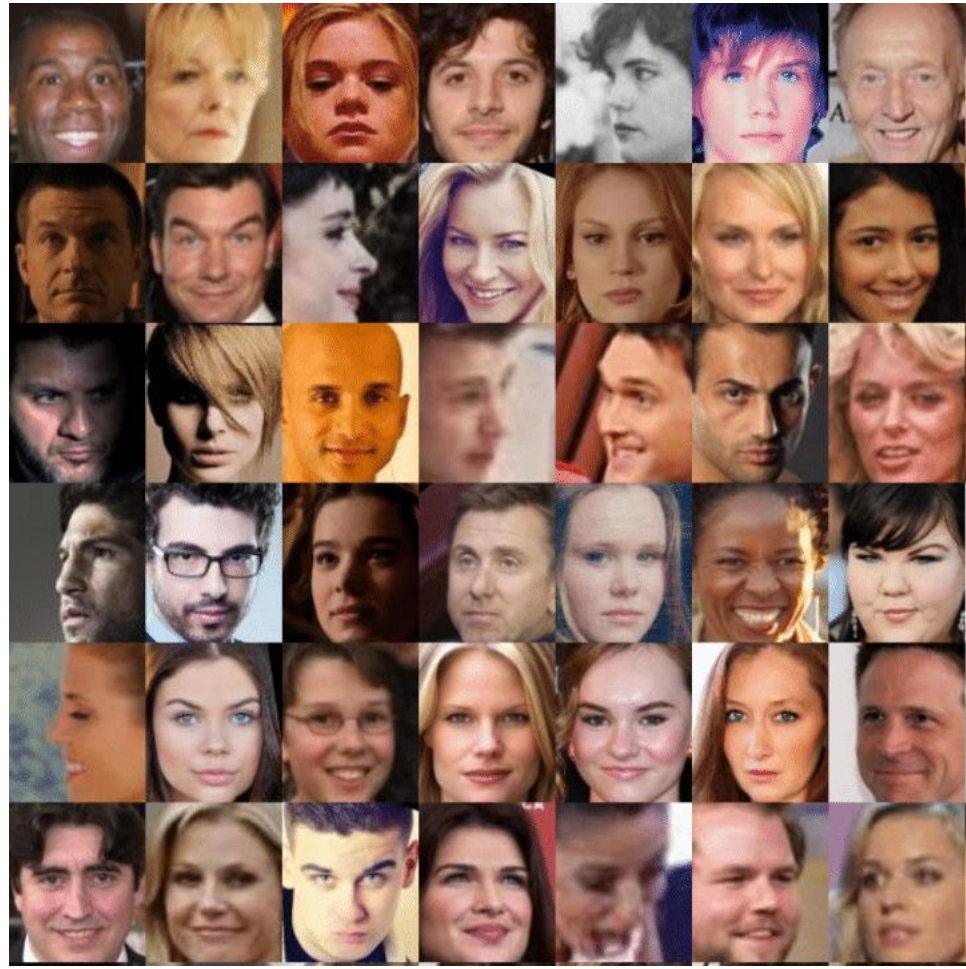
+ Faster
+ Landmarks
- Less accurate
- No batch processing

| Model | GPU Inference Titan X | FDDB Precision (100 errors) |
|---|---|---|
| R-FCN | 40 ms | **92%** |
| MTCNN | **17 ms** | 90% |

# Face recognition

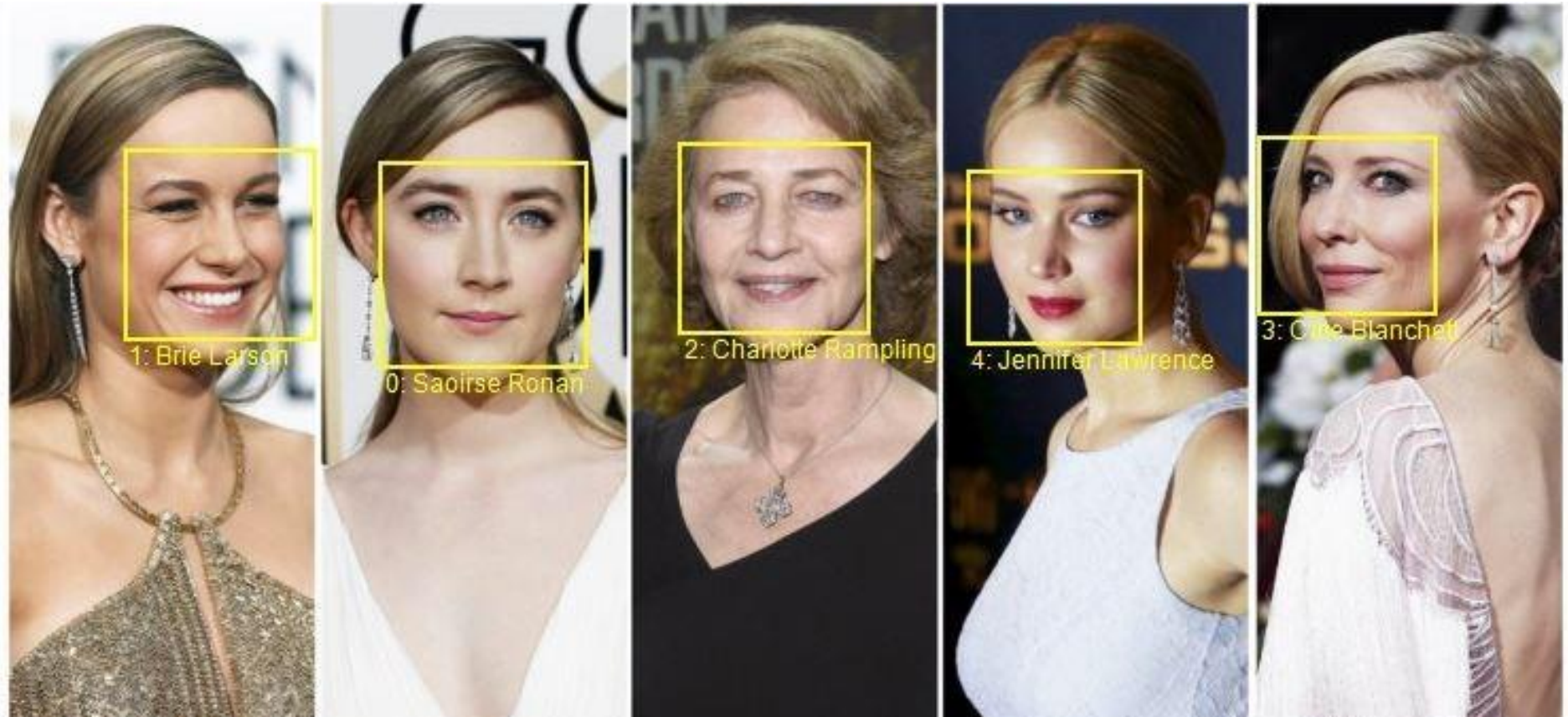
# Face recognition task

- **Verification**: validating a claimed identity based on the image of a face (one-to-one)
- **Identification**: to identify a person based on the image of a face/search (one-to-many)



# Training set: MSCeleb

- Top 100k celebrities
- 10 Million images, 100 per person
- Noisy: constructed by leveraging public search engines



# Small test dataset: LFW

mail.ru group

**Labeled Faces in the Wild Home**

- 13k images from the web
- 1680 persons have >= 2 photos



# Large test dataset: Megaface

- Identification under up to 1 million “distractors”
- 530 people to find

Explore Most Recent Public Results (last update 3/12/2017) ››

**Challenge 1:** Train on any dataset, test your method with **1 million** distractors

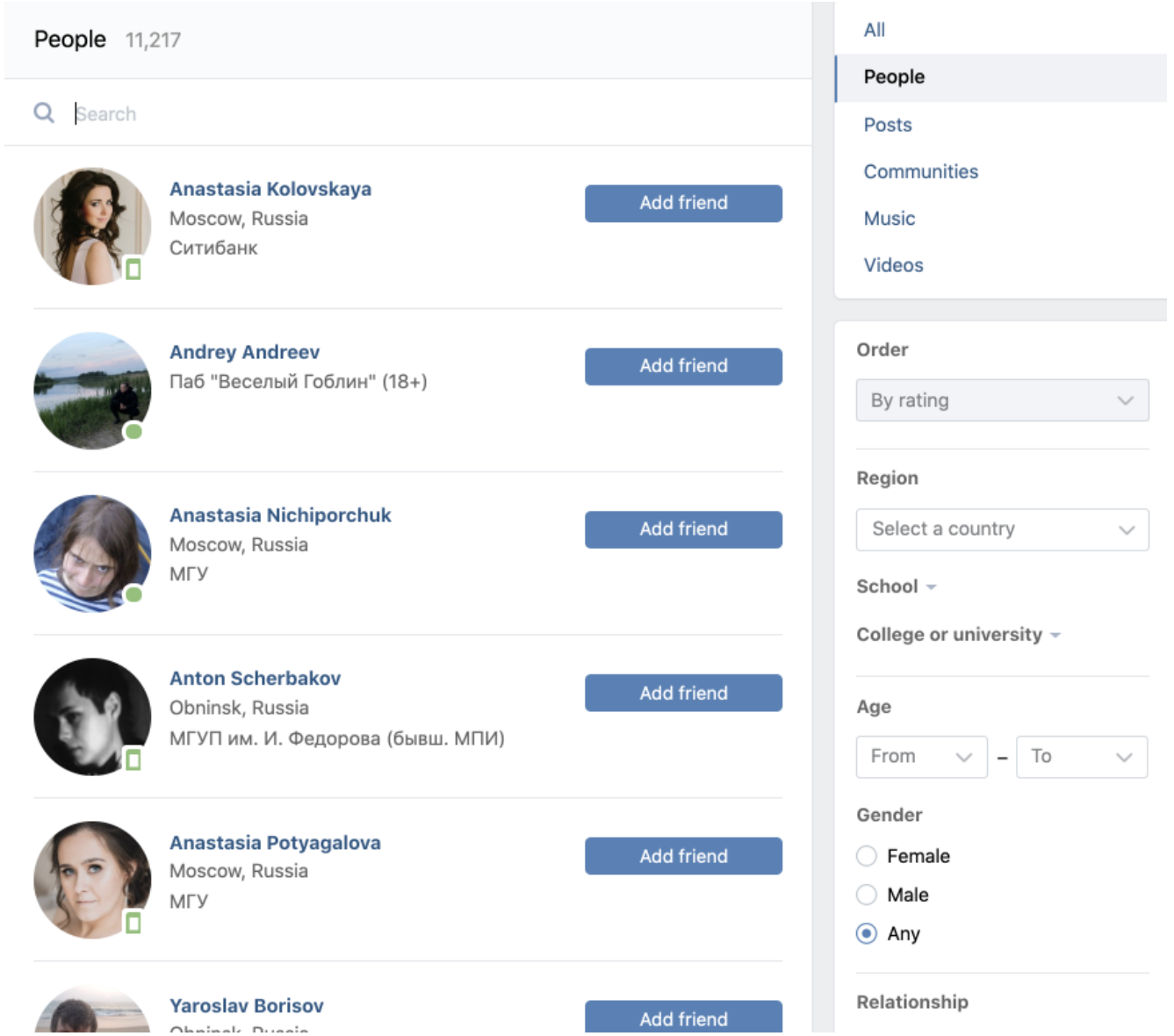
Participate and download Challenge 1 ››

**Challenge 2:** Training on **672K** identities (4.7 Million photos), test at Million scale

Participate and download Challenge 2 ››

# Our data

- Train/test: grabbed from VK
- Test like Megaface



# Metric learning

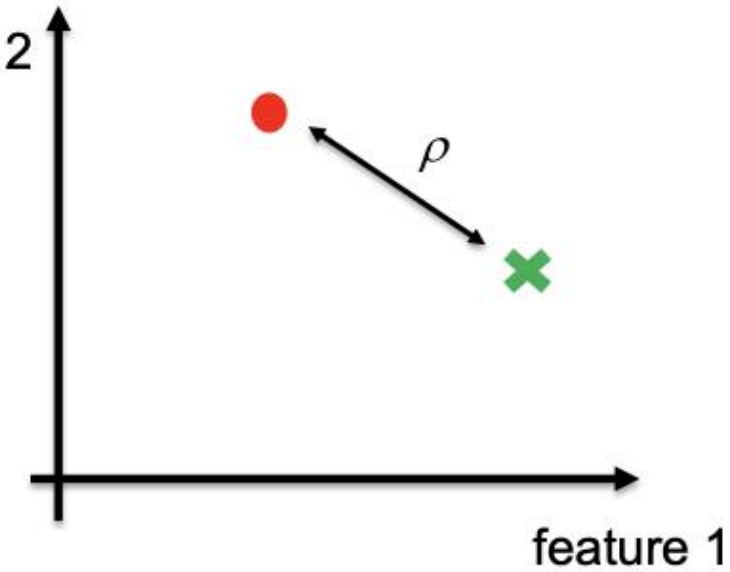
# What's metric ?

A type of mechanism to combine features to effectively compare observations

$$\rho(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2}$$

$$= \sqrt{\left[\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}\right] \cdot \left[\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}\right]}$$

$$= \sqrt{(x - y)^\mathsf{T}(x - y)}$$



# What's metric ?

It is a proper distance if and only if C satisfies:

1. $C_{ij} \geq 0$
2. $C_{ij} = C_{ji}$
3. $C_{ik} \leq C_{ij} + C_{jk}$, for any i,j,k

# Metrics

1. Minkowski distance

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

- Manhattan distance p=1
- Euclidean distance p=2

# Metrics

2. The Mahalanobis distance $D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}.$

- $\mu$ – mean, S – covariance matrix
- can be ”trained” on data
- $\mu$=0, S=1

# Metrics

2. Cosine similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Link to Euclidian:

$$||A||_2 = ||B||_2 = 1$$

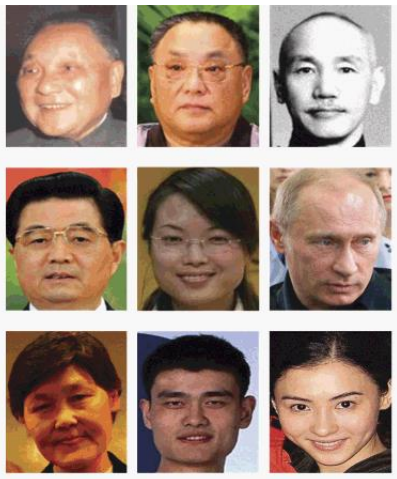
$$|A - B|_2^2 = (A - B)^T (A - B) =$$

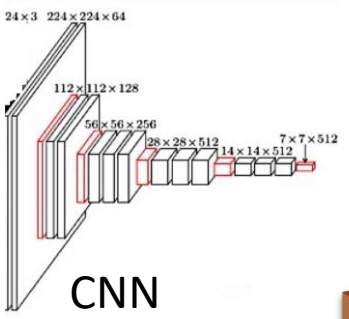
$$= A^T A - 2A^T B + B^T B = 2 - 2A^T B = \mathbf{2 - 2\cos\theta}$$

Good for high dimensions

# Metric learning for FR

- Goal – to compare faces
- How? To learn metric
- To enable **Zero-shot** learning



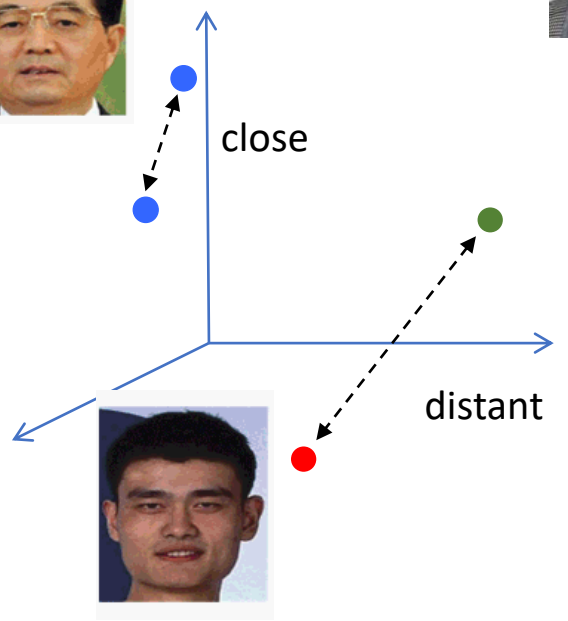



CNN

Embedding

128 floats

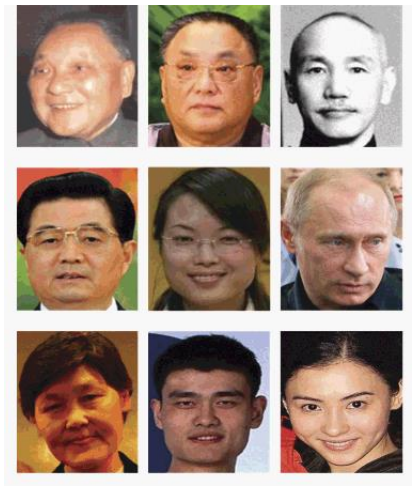
Latent Space

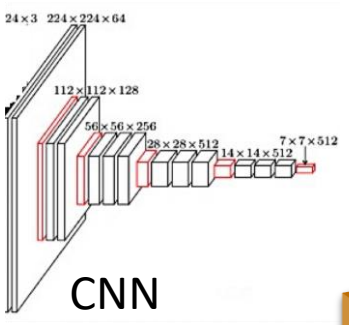



Unseen

# Classification

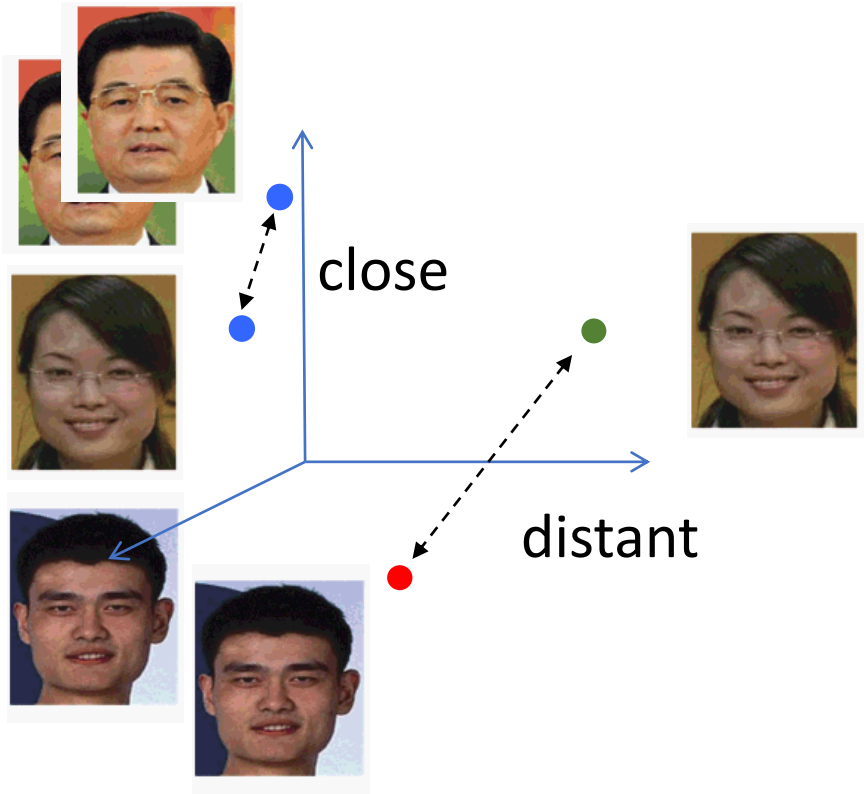
- Train CNN to predict classes
- Pray for good latent space





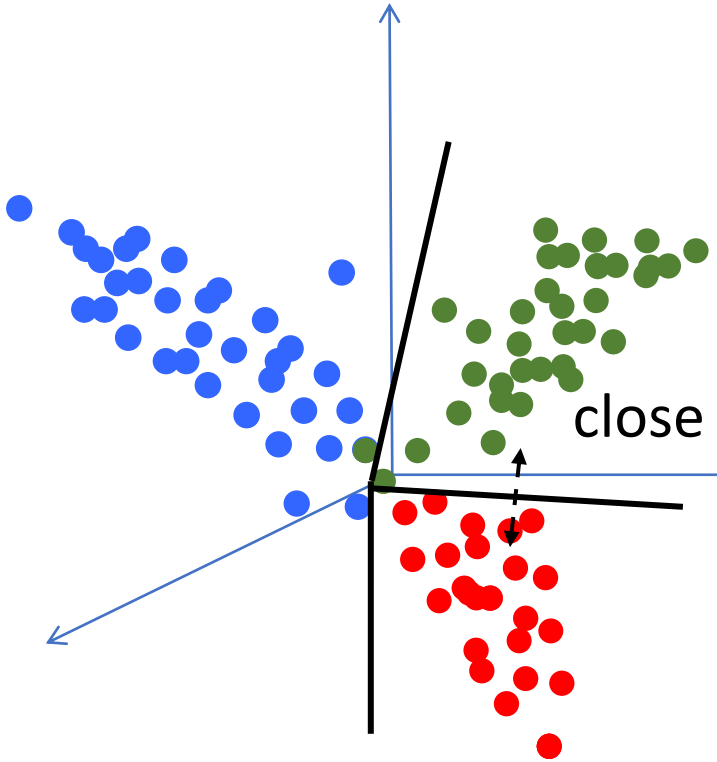

Embedding


Classify



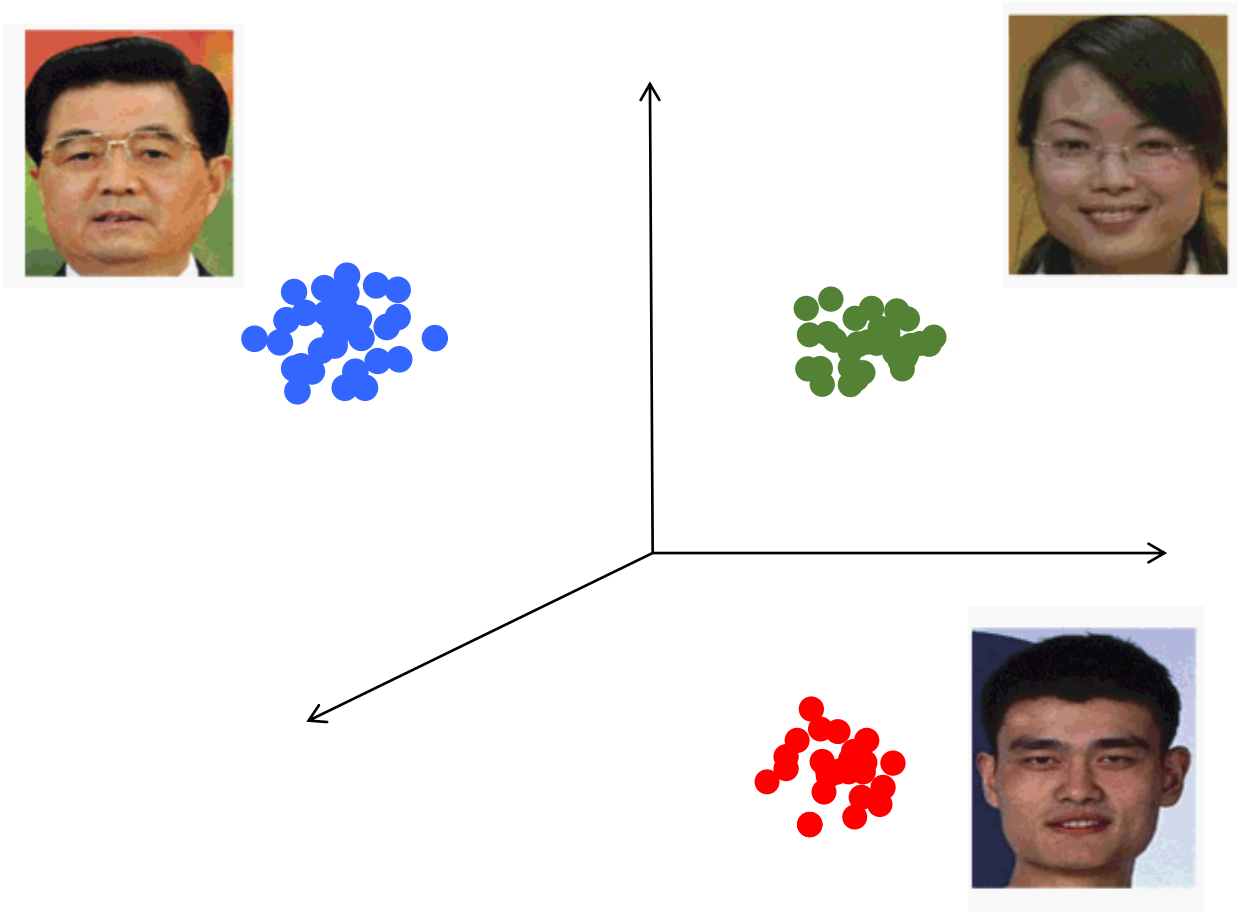
# Softmax

- Learned features only separable but not discriminative
- The resulting features are not sufficiently effective




# We need metric learning

- Tightness of the cluster
- Discriminative features



# Metric learning types

1. **Direct: similarity**
   - Classification similarity
   - Ranking similarity
2. **Indirect: classfication**
   - Softmax based

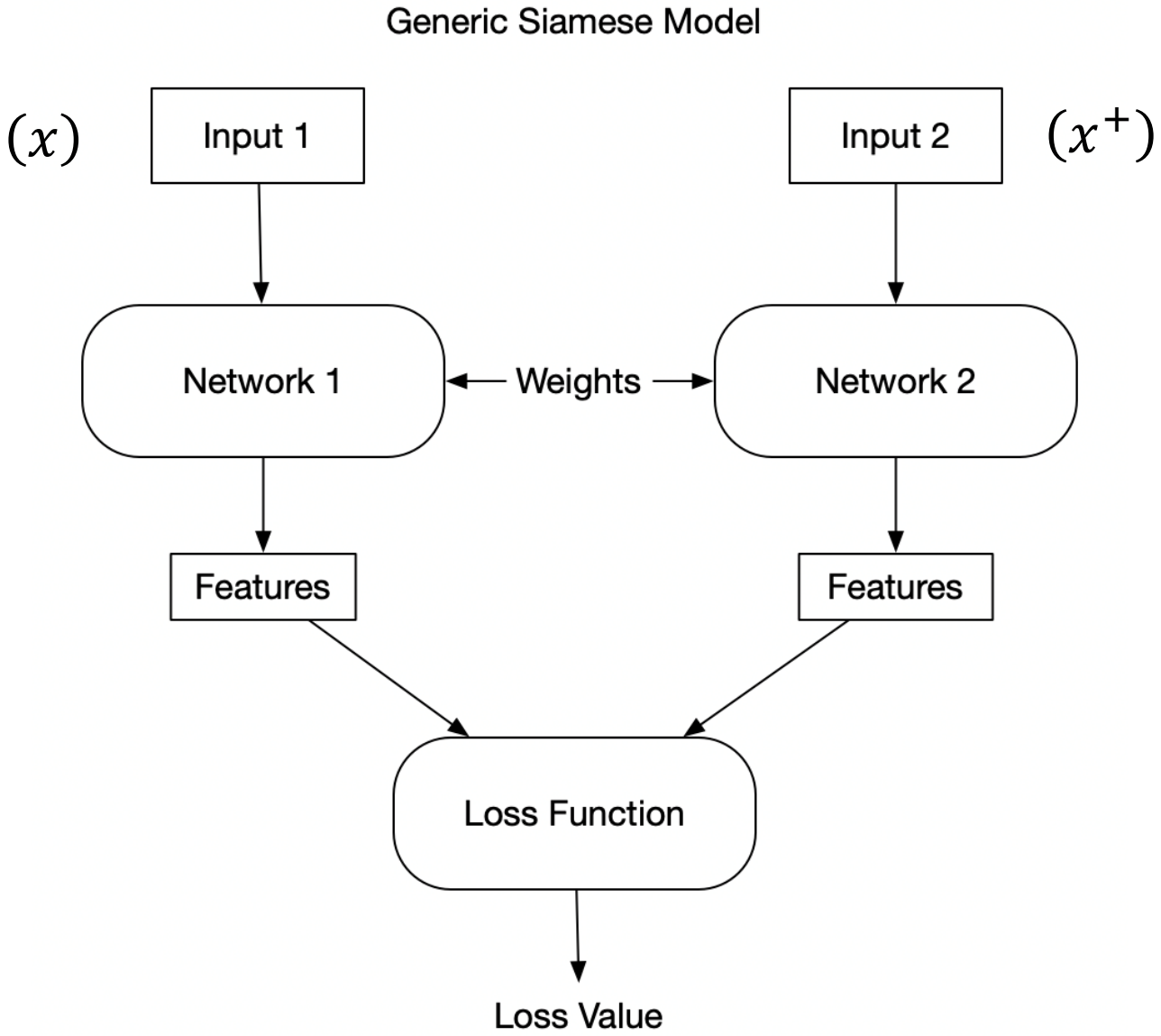
# Similarity based

# Classification similarity

Data: pairs of similar and not similar objects

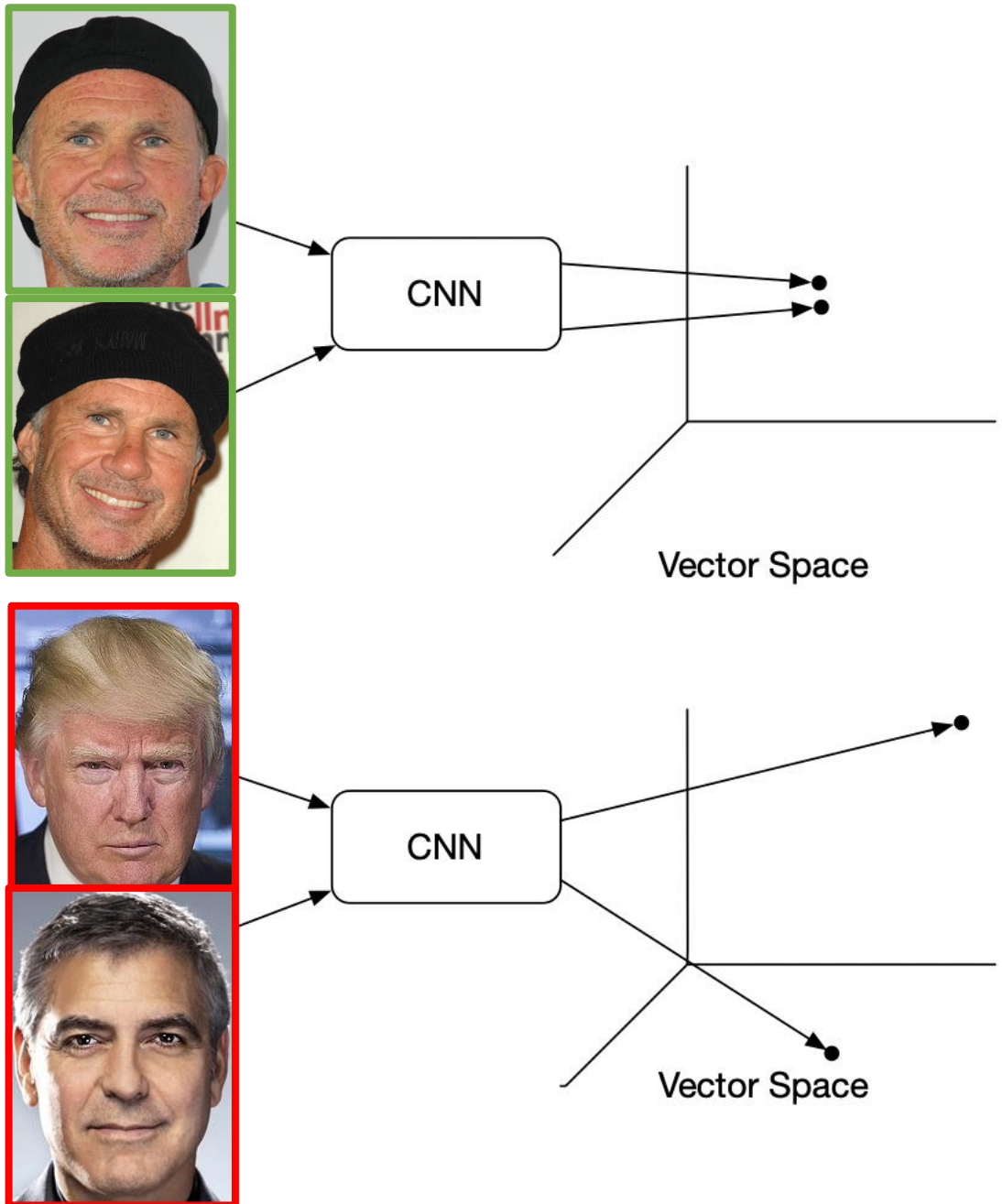
$$(x, x^+), (x, x^-)$$

# Siamese networks




Source: https://towardsdatascience.com/contrastive-loss-explaned-159f2d4a87ec

# Goal



# Classification similarity loss

- Cosine distance as prediction probabilities
- Softmax with temperature and vector similarities + CE
- Denominator: only pos to neg

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} {}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

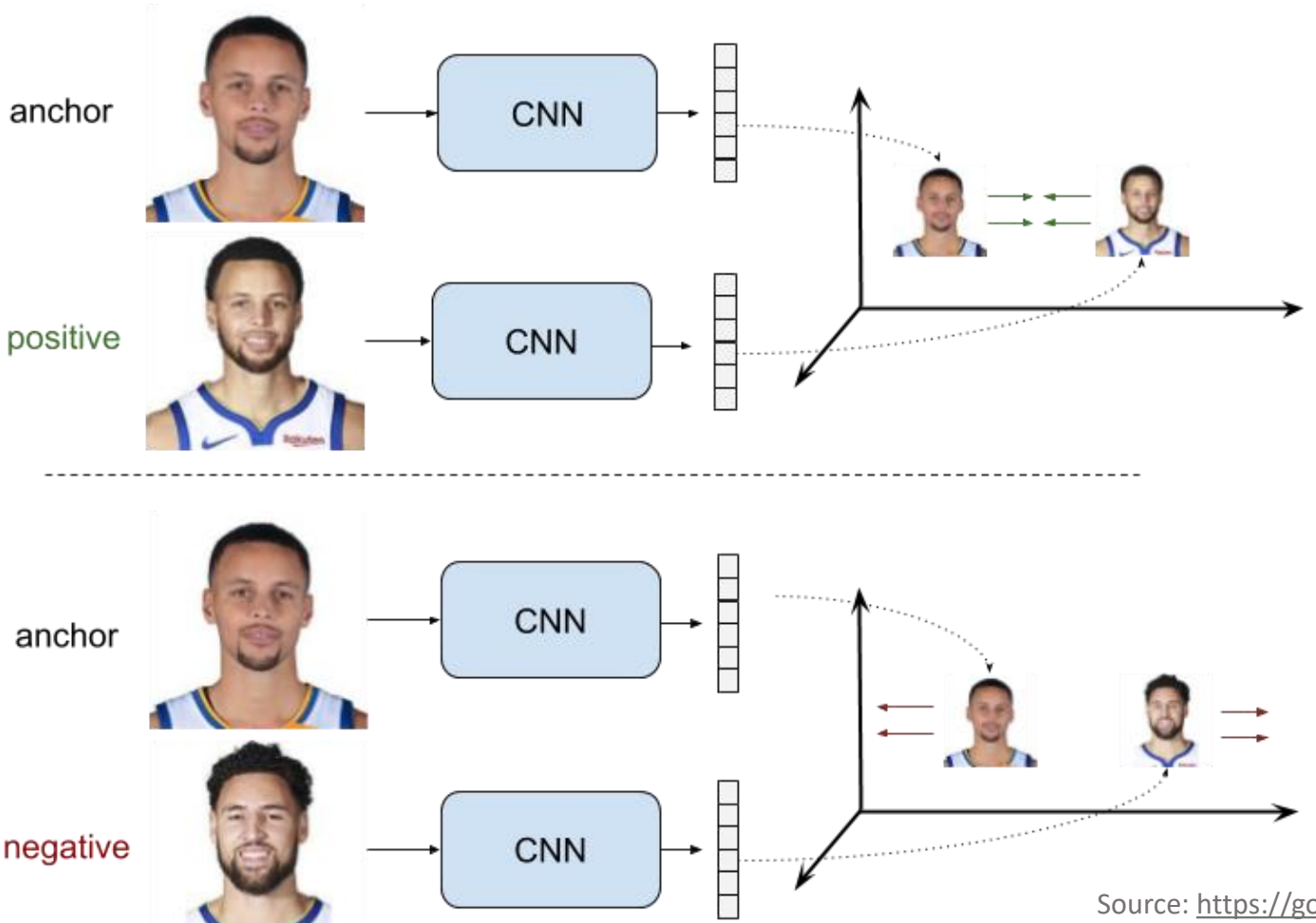
- $Optimal$: $\cos(x, x^+) = 1, \cos(x, x^-) = 0,\ l = -\log(1) = 0$

# Pairwise Ranking/Contrastive Loss

$$L(x_0, x_1, y) = y \left|\left| x_0 - x_1 \right|\right| + (1 - y) \max\left(0, m - \left|\left|x_0 - x_1\right|\right|\right),$$
$$y = 0|1$$

positive

negative



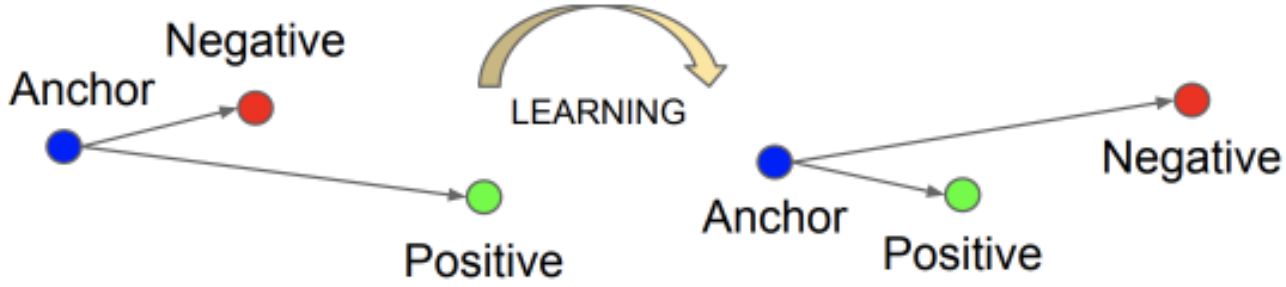

Source: https://gombru.github.io/2019/04/03/ranking_loss/

# Triplet loss

Data: triplet of anchor, positive and negative examples

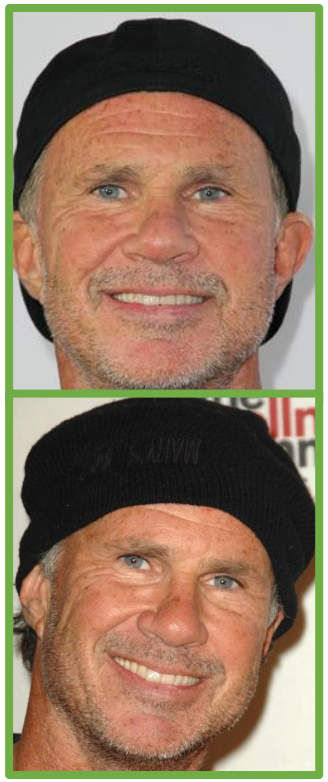
$$(x, x^-, x^+)$$

Introduces in FaceNet by Google




Paper: https://arxiv.org/pdf/1503.03832.pdf

# Triplet loss



minimize

Anchor

maximize

Positive

positive + α < negative

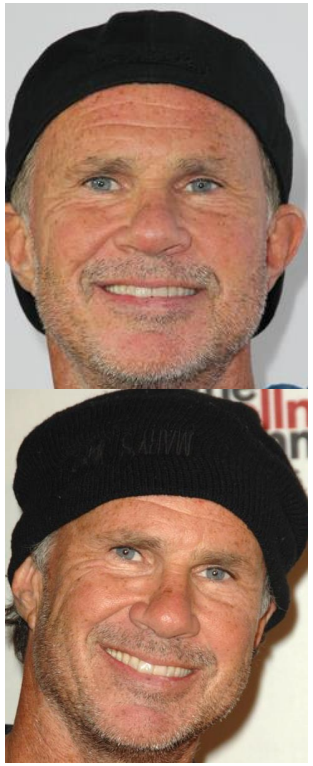
Negative

- $L(a, p, n) = \max(0, m + d(a, p) - d(a, n))$
- Enforces a margin between persons

# Choosing triplets

**Crucial problem:** too many trivial triplets

How to choose triplets ? Useful triplets = hardest errors



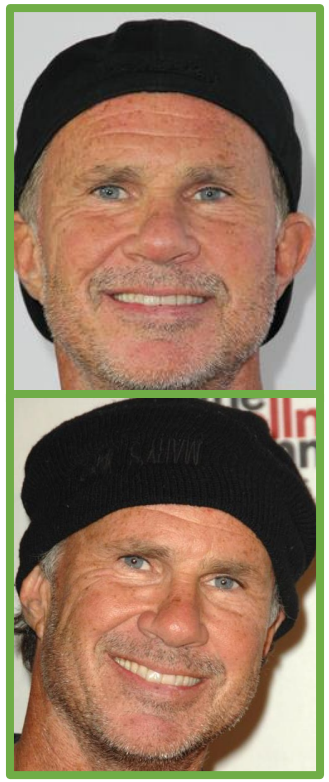
Pick all positive

Hard enough

Too easy

# Choosing triplets: trap

# Choosing triplets: trap

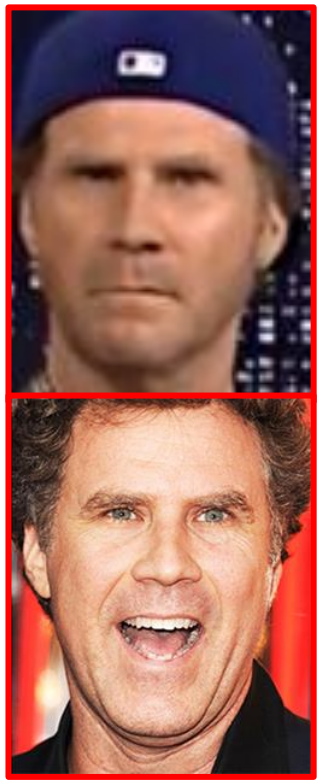


Positive

minimize

Anchor

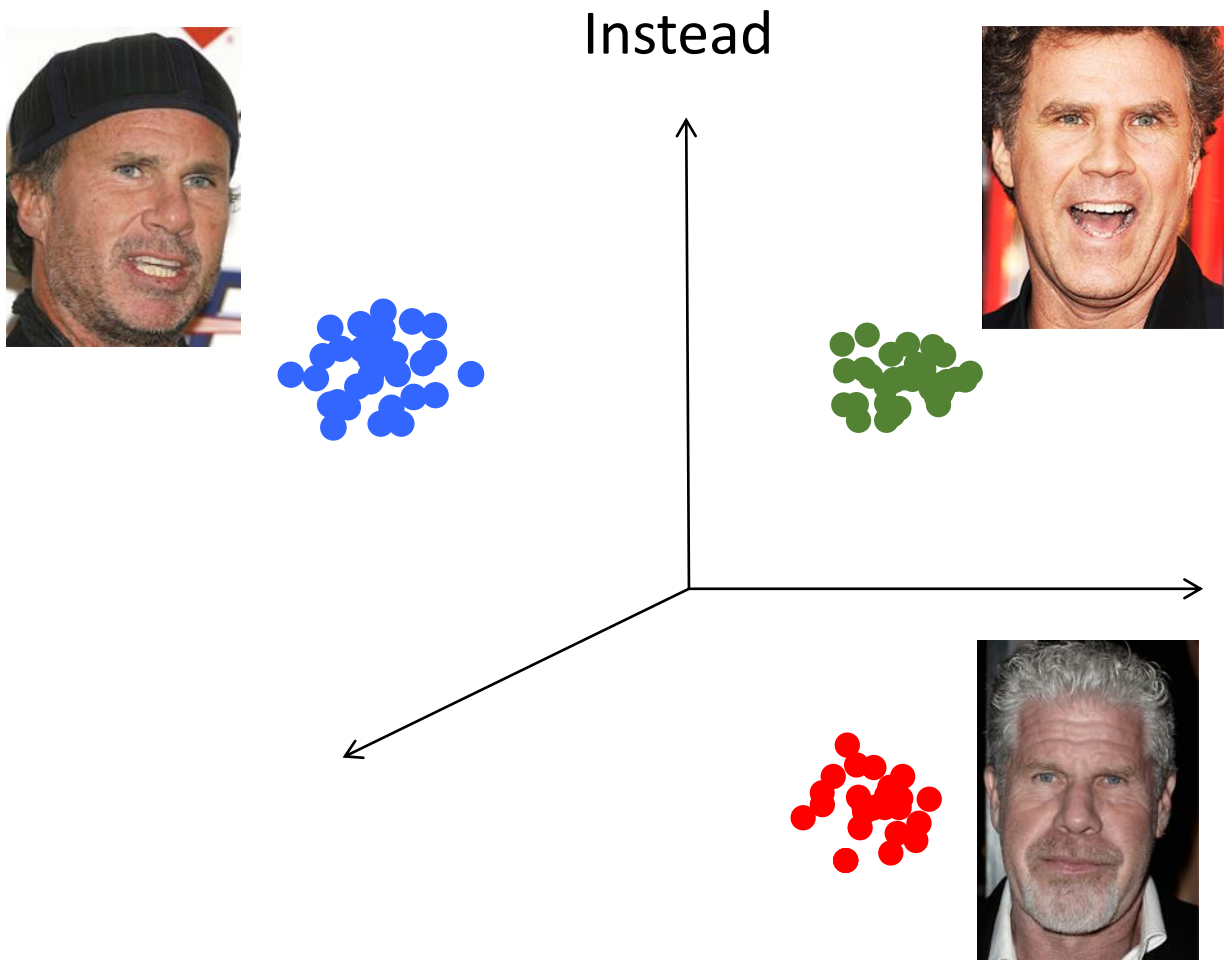
maximize



Negative

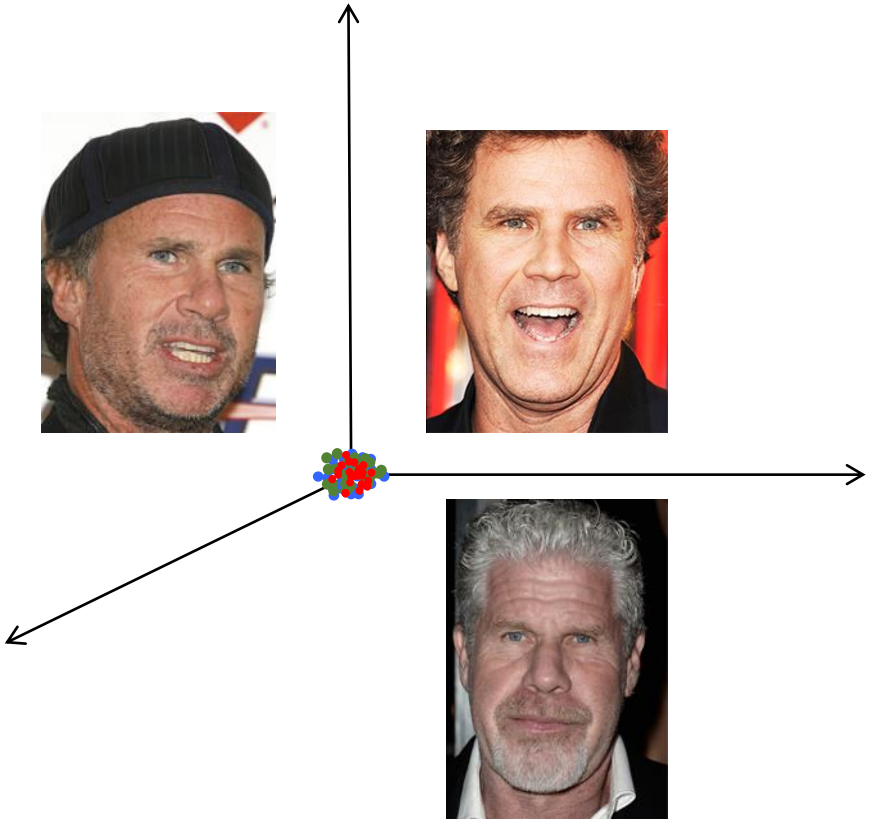
positive ~ negative

# Choosing triplets: trap

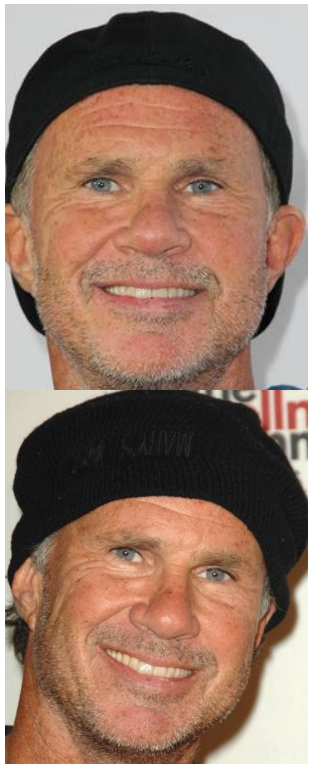


# Choosing triplets: trap

Selecting hardest negative may lead to the collapse early in the training



# Choosing triplets: semi-hard



Pick all positive

Too hard

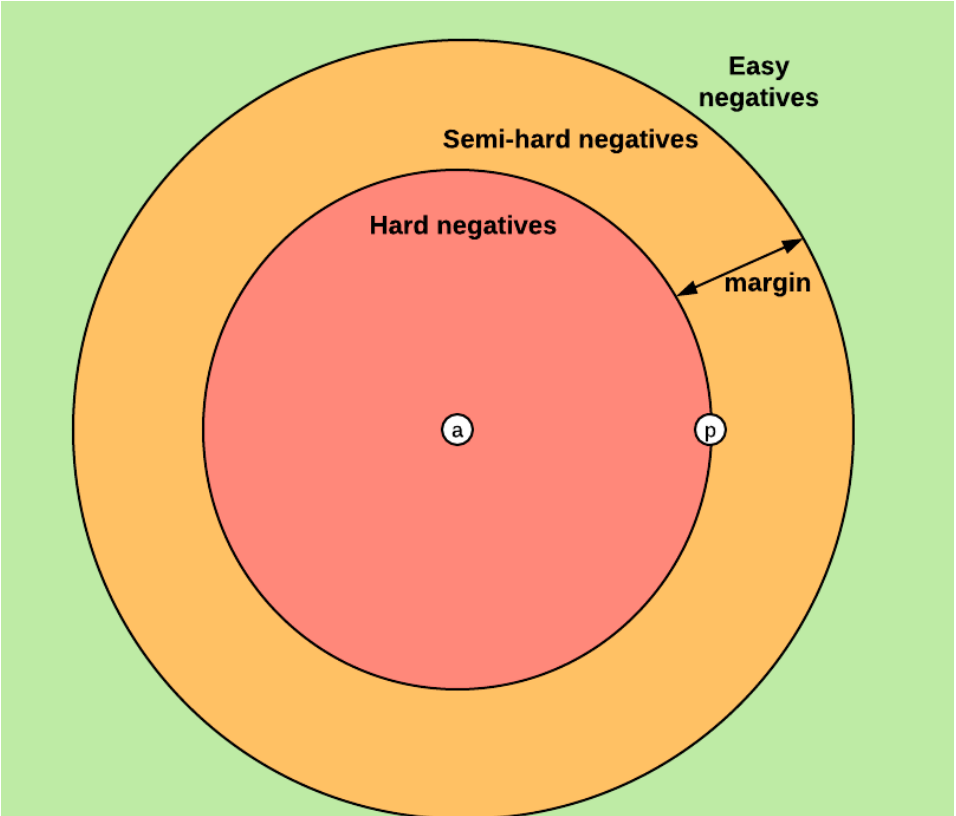
Semi-hard

Too easy

positive < negative < positive + α

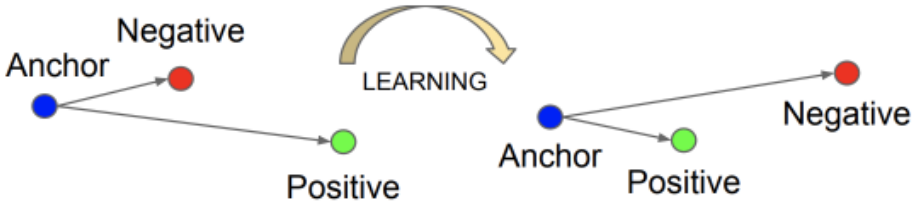
# Online hard-negative mining

- Problem: too many triplets
  - scanning the dataset is impractical
- Solution: online within a large mini-batch (>1000)



# Triplet loss: summary

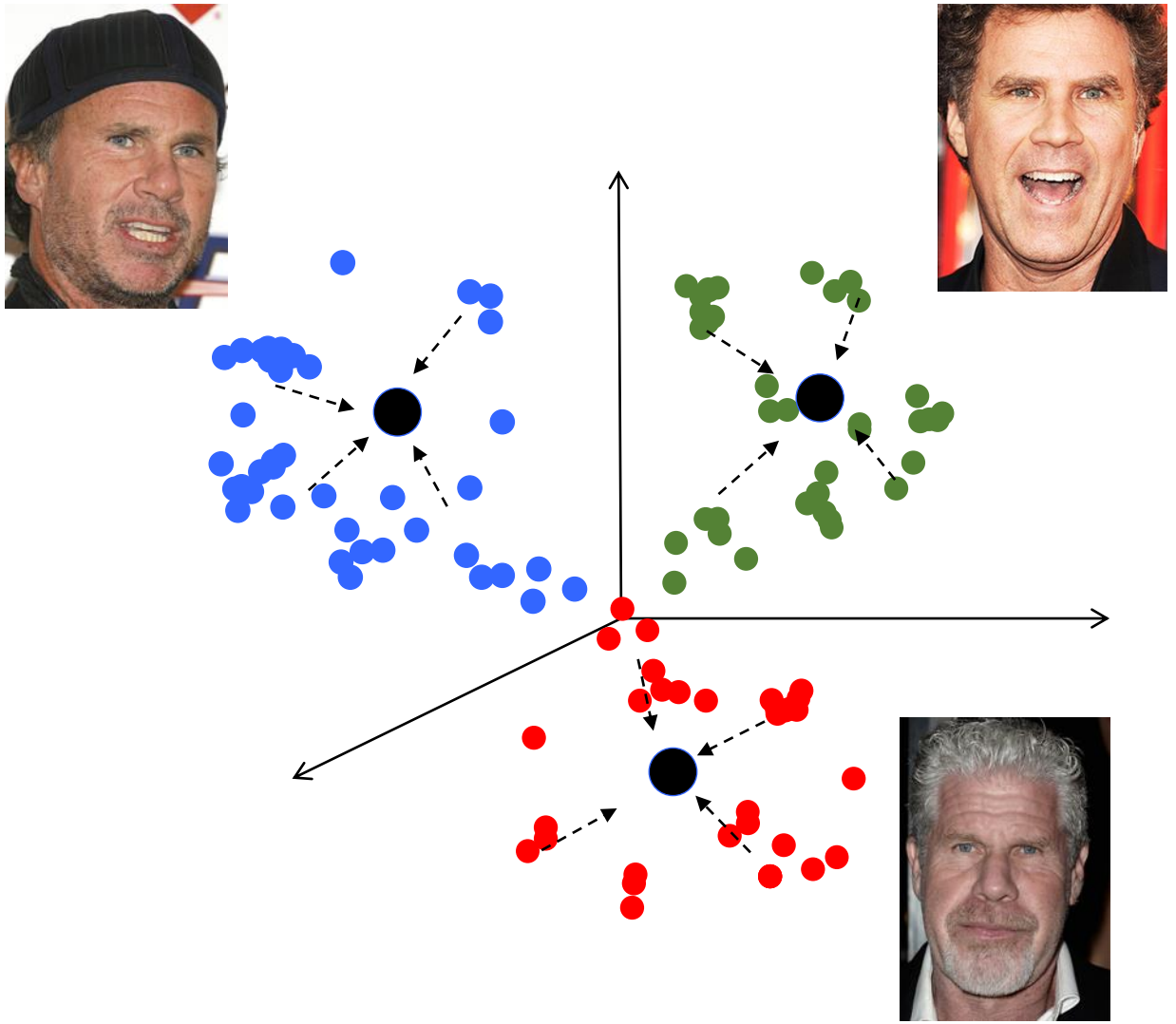
+ Supports large datasets
– Requires large batches
– Slow convergence



# Softmax-based

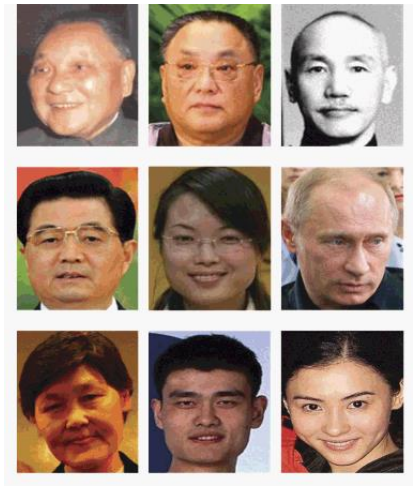
# Center loss

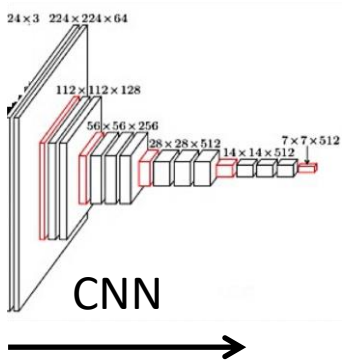
Idea: pull points to class **centroids**



# Center loss: structure

- Without classification loss – collapses
- Final loss = Cross Entropy + λ Center loss






Embedding

128 floats

Classify

Pull

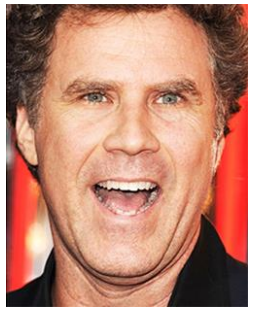
Softmax & CE

λ

Center Loss

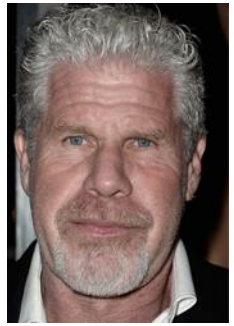
$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

# Center Loss: different lambdas

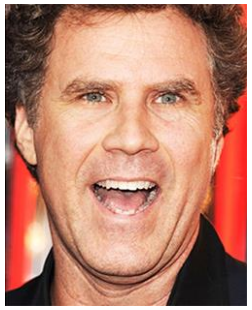


$\lambda = 10^{-7}$

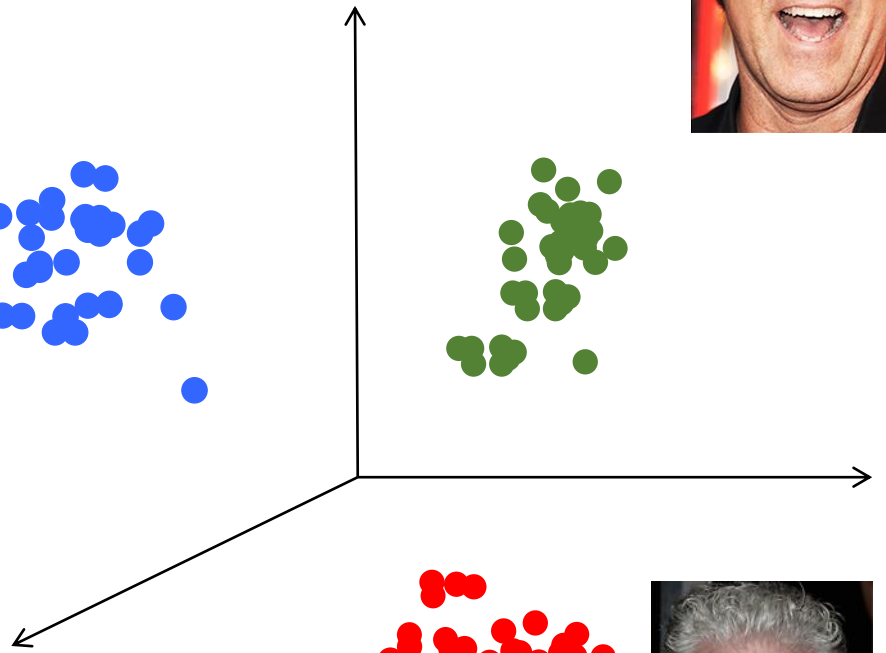


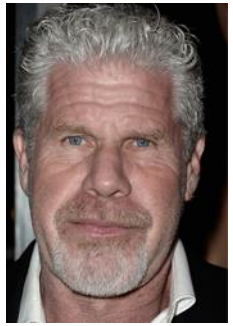


# Center Loss: different lambdas

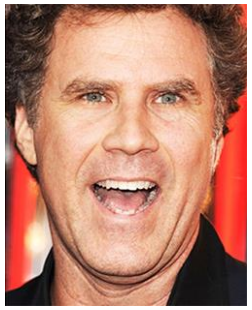


$\lambda = 10^{-6}$



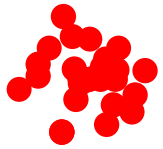


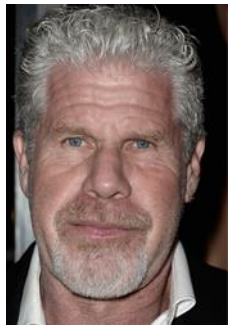
# Center Loss: different lambdas



$\lambda = 10^{-5}$







# Centres-loss gradients

**Problem**: we have to compute centers over whole dataset

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

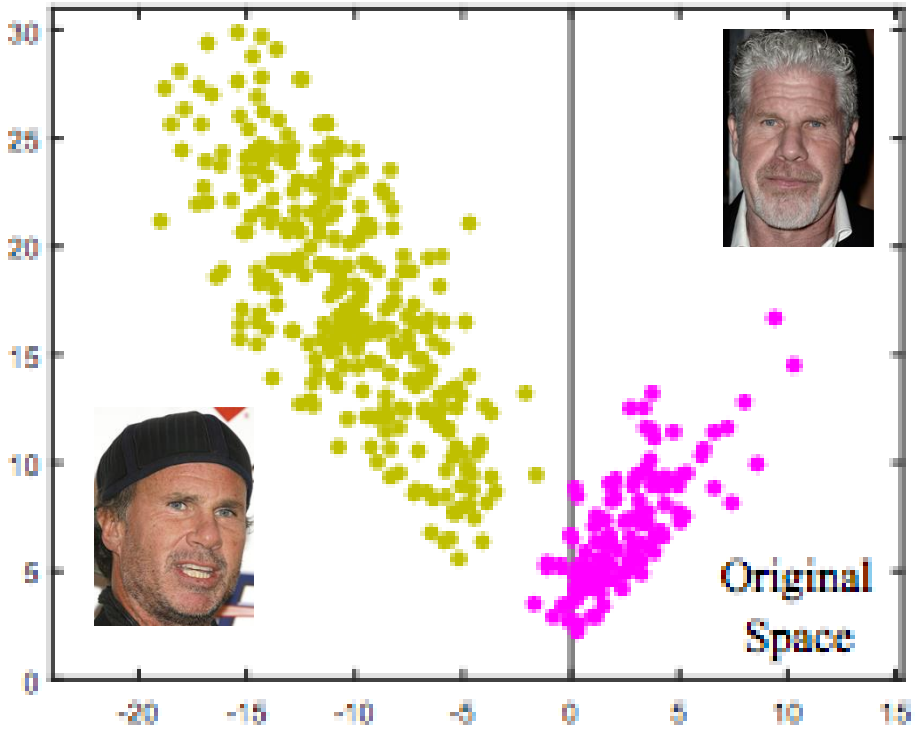
**Solution**:

we train centers as parameters & update on mini-batch

# Center loss: summary

+ Simple
+ Intra-class compactness and inter-class separability
+ Improve classification tasks
– Not SOTA

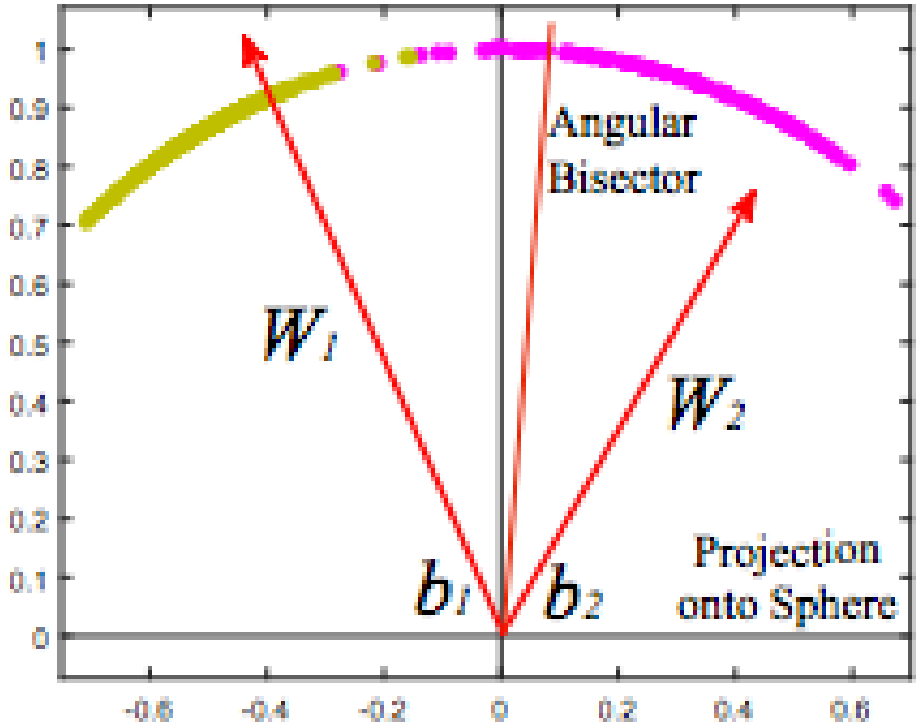
# Angular Softmax




$$(\boldsymbol{W}_1-\boldsymbol{W}_2)\boldsymbol{x}+b_1-b_2=0$$

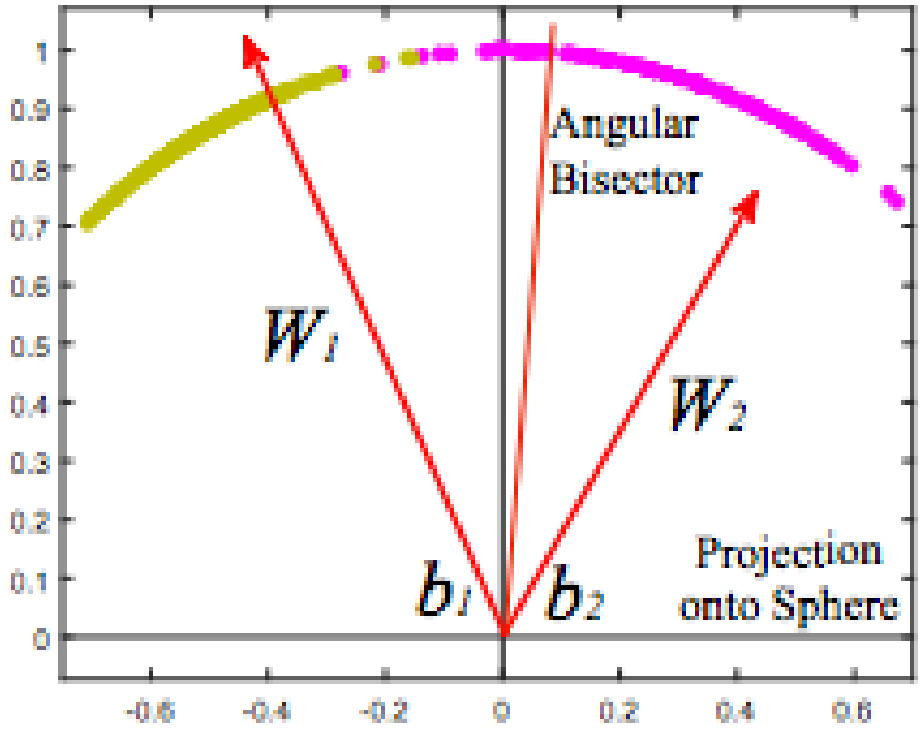
$\|X\| = 1$

$\|W\| = 1$
$b=0$

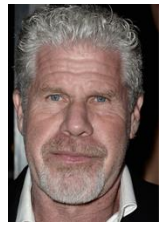


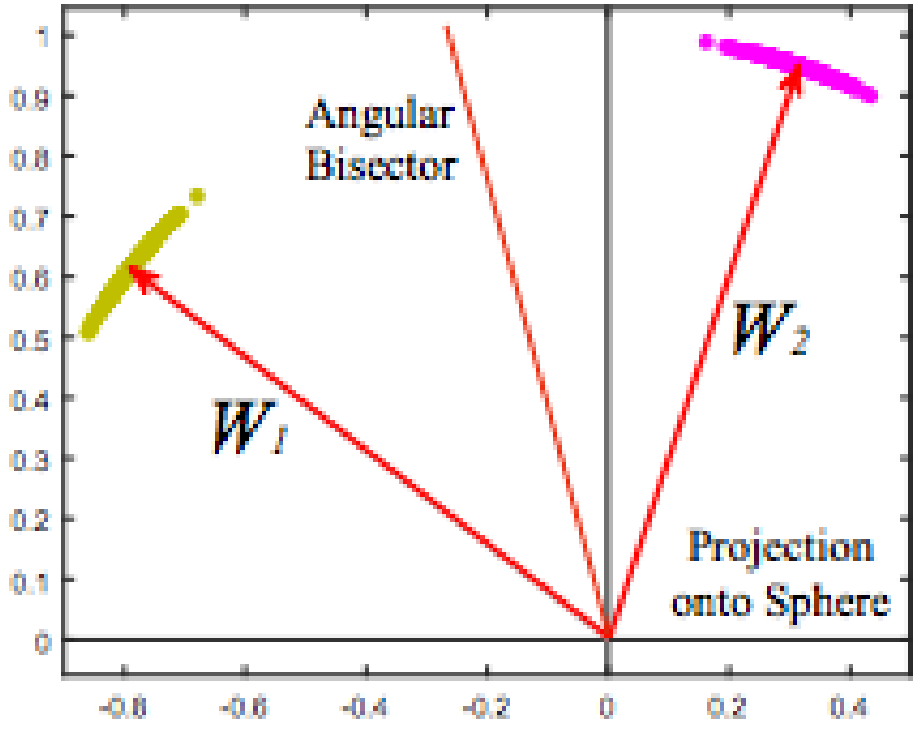

$$\|\boldsymbol{x}\|(\cos(\theta_1)-\cos(\theta_2))=0$$

# Angular Softmax


To enforce larger angle

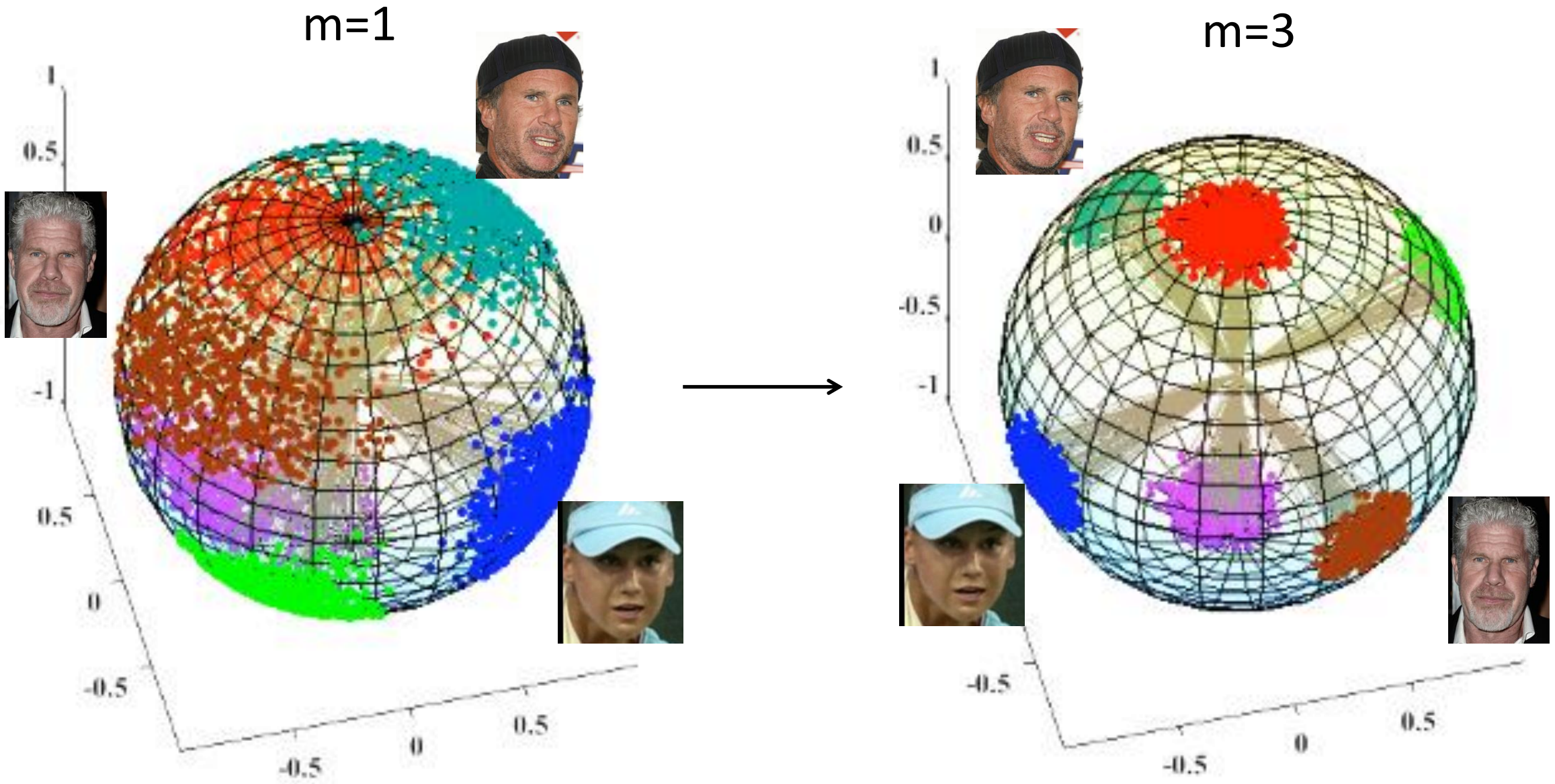






$$\|\boldsymbol{x}\|(\cos(\theta_1)-\cos(\theta_2))=0$$

$$\|\boldsymbol{x}\|(\cos(m\theta_1)-\cos(\theta_2))=0$$

# Angular Softmax: different «m»



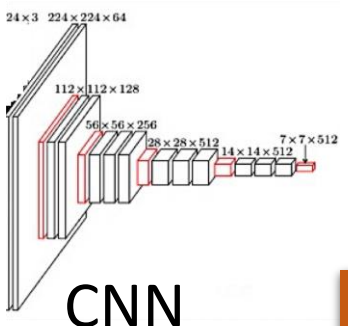
# Angular Softmax family

Variations

- CosFace
- ArcFace (SOTA)

$$f_j^{arcface} = \begin{cases} s\cos(\theta_j + m) & , j = y \\ s\cos\theta_j & , j \neq y \end{cases}$$

# Metric learning: Arcface




CNN →

Embedding

**128 floats**

Classify →

Scores

10

9

~~20~~ 13

Softmax →

Cross Entropy

# Softmax-based summary

+ ArcFace – SOTA
+ Easy integration with Softmax
+ Improve classification tasks
– But … Memory/Compute limitations

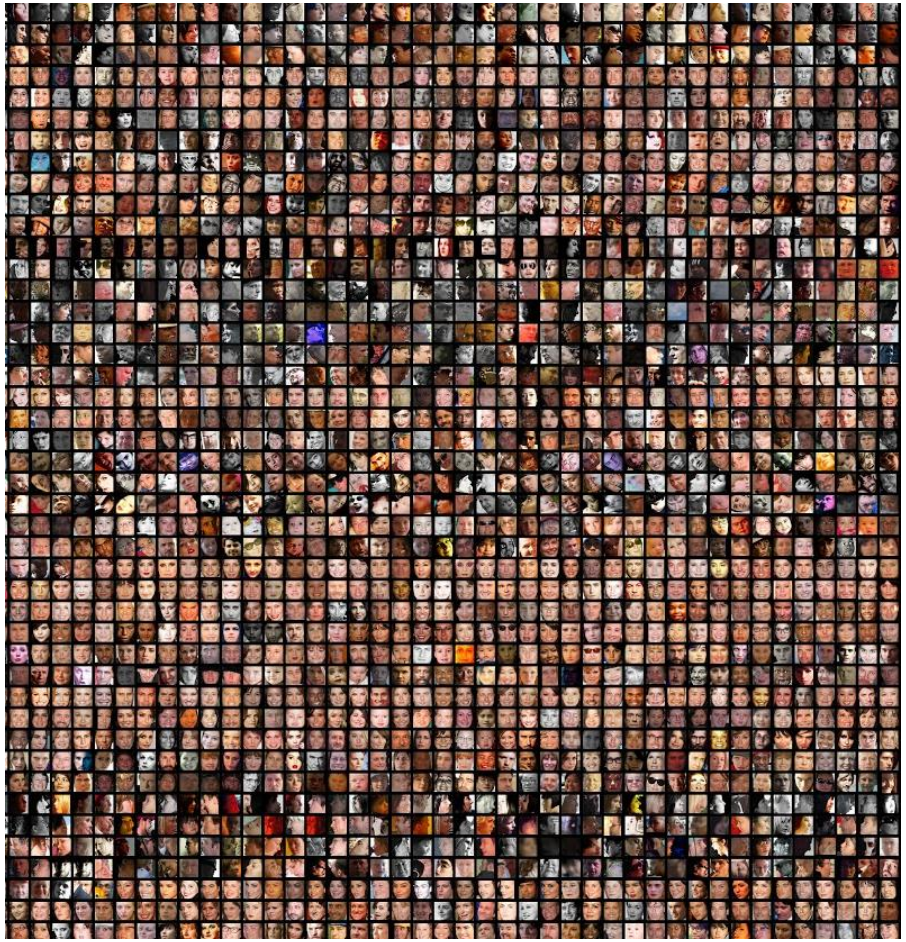
# How to handle big dataset

It seems we can add more data infinitely, but no.

Linear layer depends on the **number of classes**

**Problems**

- Memory consumption (Softmax)
- Computational costs
- A lot of noise in gradients



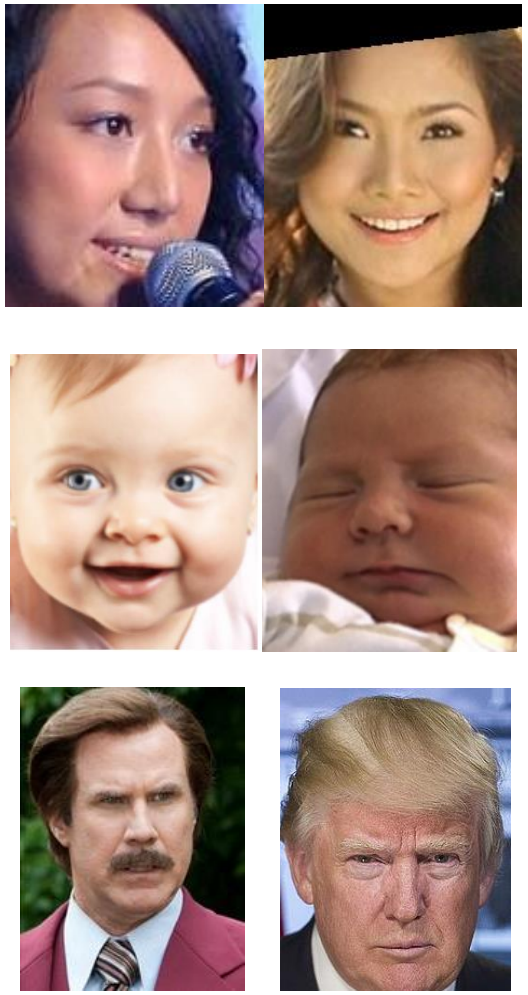
# Softmax Approximation

**Algorithm**

1. Perform K-Means clustering using current FR model

Dataset

K-Means



Women

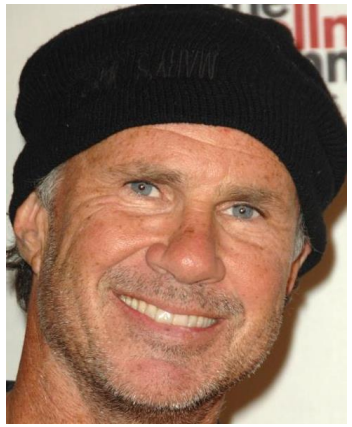
Children

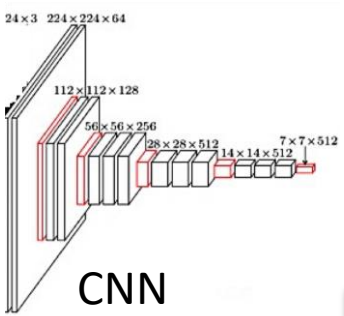
Men

Smaller sets

# Softmax Approximation

2. Two Softmax heads:
   1. Predicts cluster label
   2. Class within the true cluster





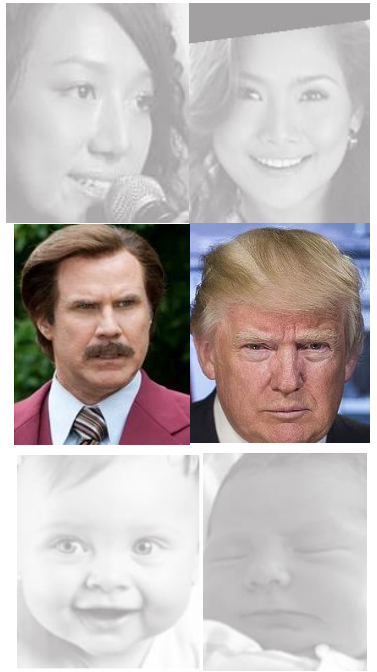

CNN

Embedding

128 floats

Predict cluster

Cluster Softmax



Men

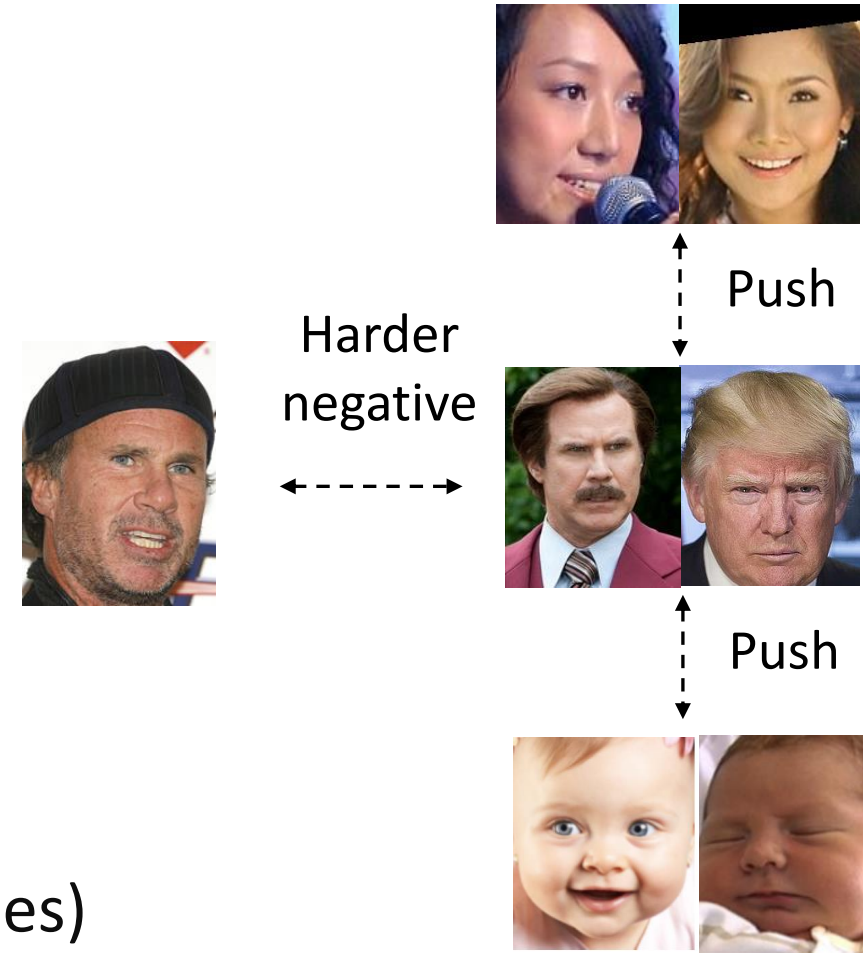
Predict person

Person Softmax

# Softmax Approximation

**Pros**

1. Prevents fusing of the clusters
2. Does hard-negative mining
3. Clusters can be specified
   - Children
   - Asian




**Results**

- Doesn’t improve accuracy
- Decreases memory consumption (**K** times)

Paper: https://arxiv.org/pdf/1609.04309.pdf

# Metric learning: bottom line

- ArcFace > Center loss, Triplet, …
- Triplet’s best for very large datasets

# Metric learning for Data cleaning

# MSCeleb dataset's errors

MSCeleb is constructed by leveraging search engines

Joe Eszterhas

=

Mel Gibson

Joe Eszterhas and Mel Gibson public confrontation leads to the error

# MSCeleb dataset's errors

Female
+
Male

# MSCeleb dataset's errors

Asia
Mix

# MSCeleb dataset's errors

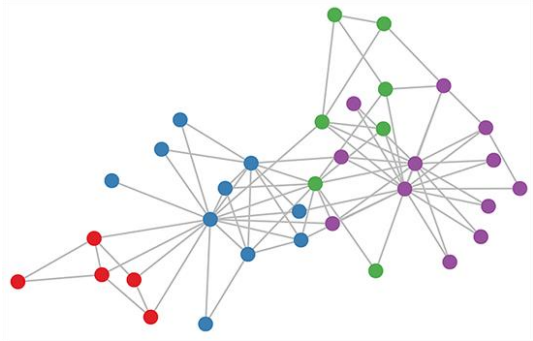
Random search engine

# How to make it clean

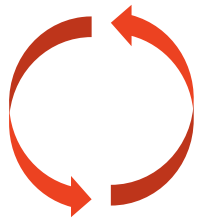
# HowTo: clean

Face Detection →

FR+ Clustering →



← Pick largest



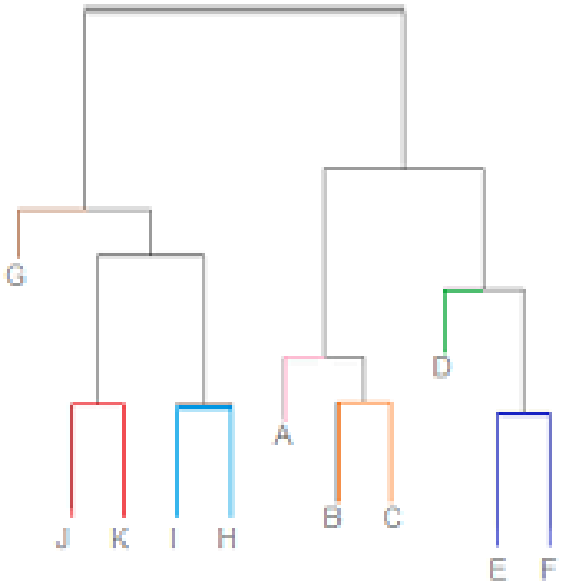
**Cleaning algorithm**

Iterate after each model improvement

# Clustering Algorithms

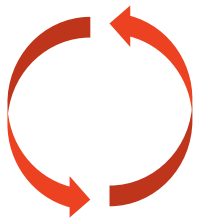
Hierarchical clustering: **CLink**

- Define distance threshold
- Spherical clusters
- $O(n^2)$



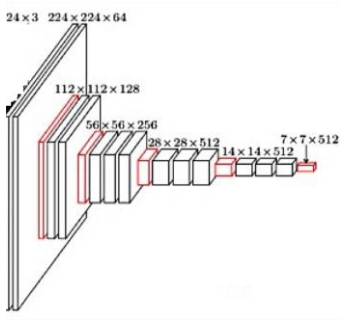
# Data cleaning: bottom line

Good embedding space provides means for cleaning your data

Loop: model  Data

# Tricks

# Tricks: augmentation




Embedding

128 floats

Flipped Embedding

128 floats

Average

Final Embedding

128 floats

**Test time augmentation**

- Flip image
- Compute 2 embeddings
- Average embeddings

# Errors: blur

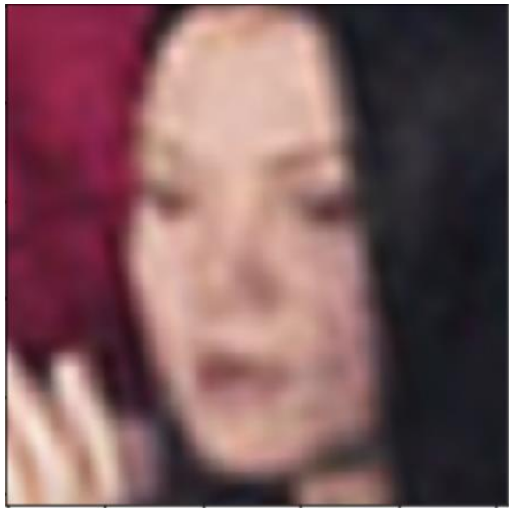
**Problem**

- Detector yields blurry photos
- Recognition forms «blurry clusters»

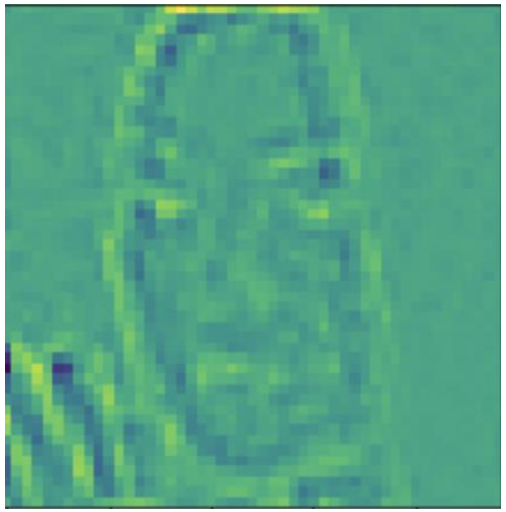
**Solution**

Laplacian – 2nd order derivative of the image

# Laplacian in action



Low variance

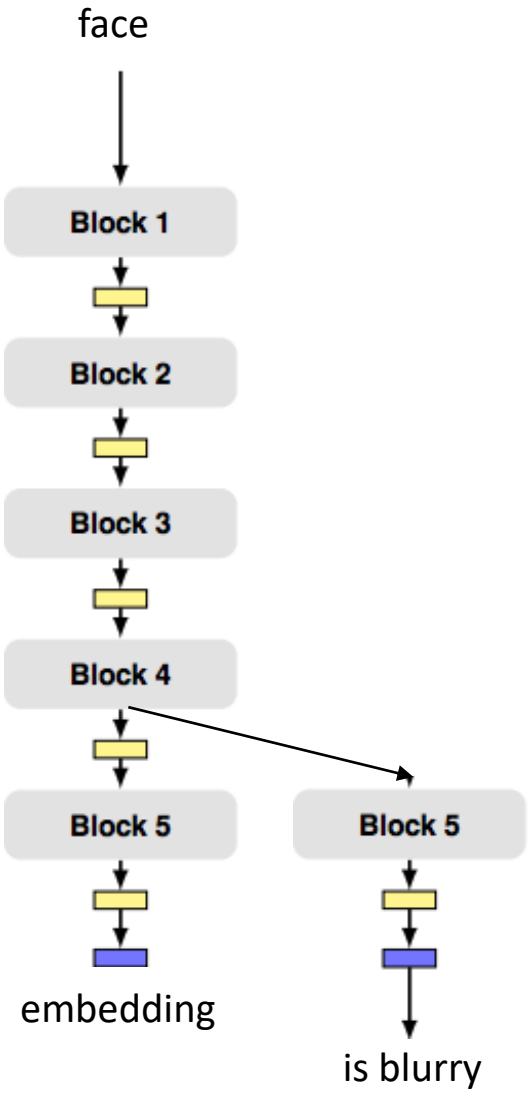


High variance

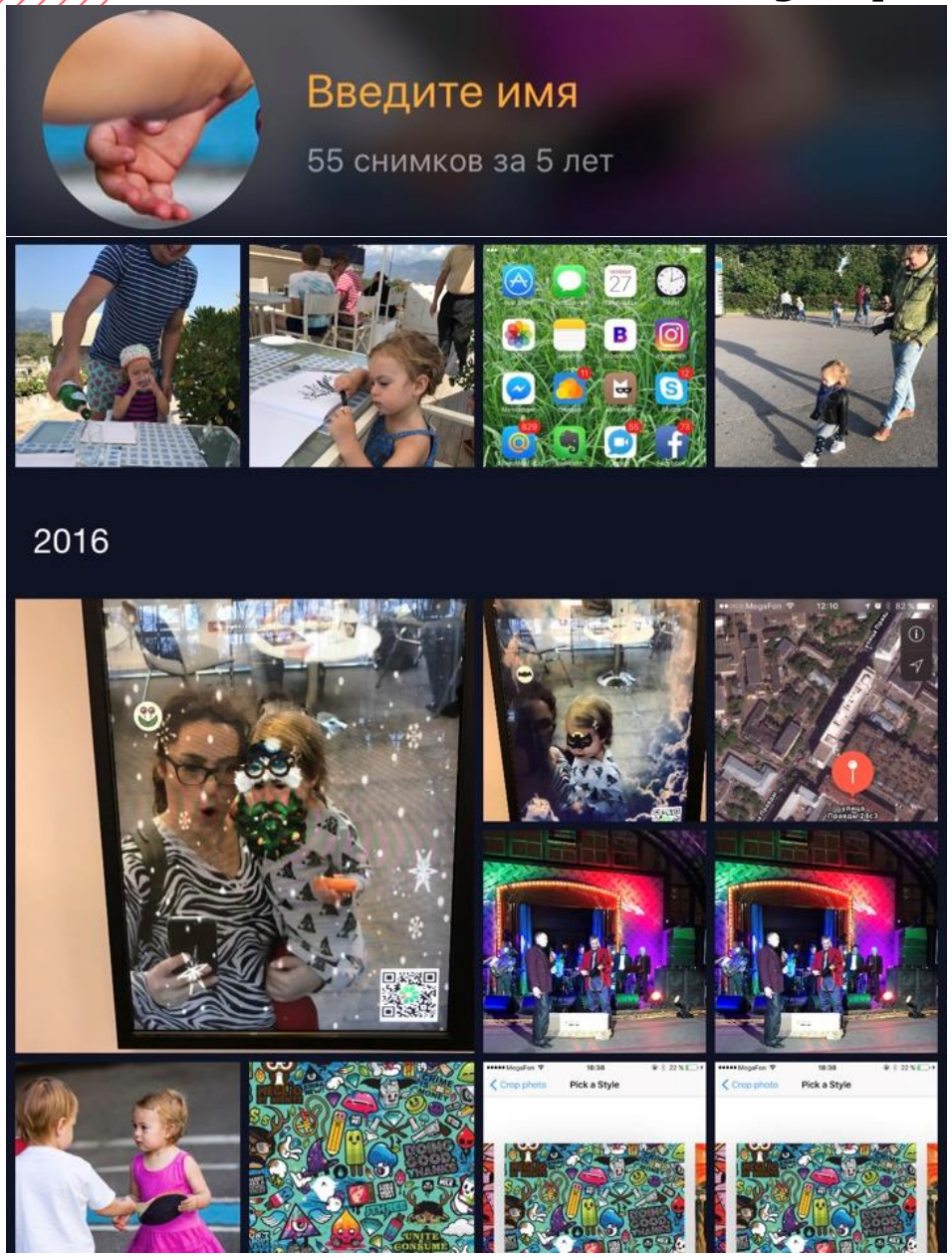
# Blur classification

Training classification is always an option

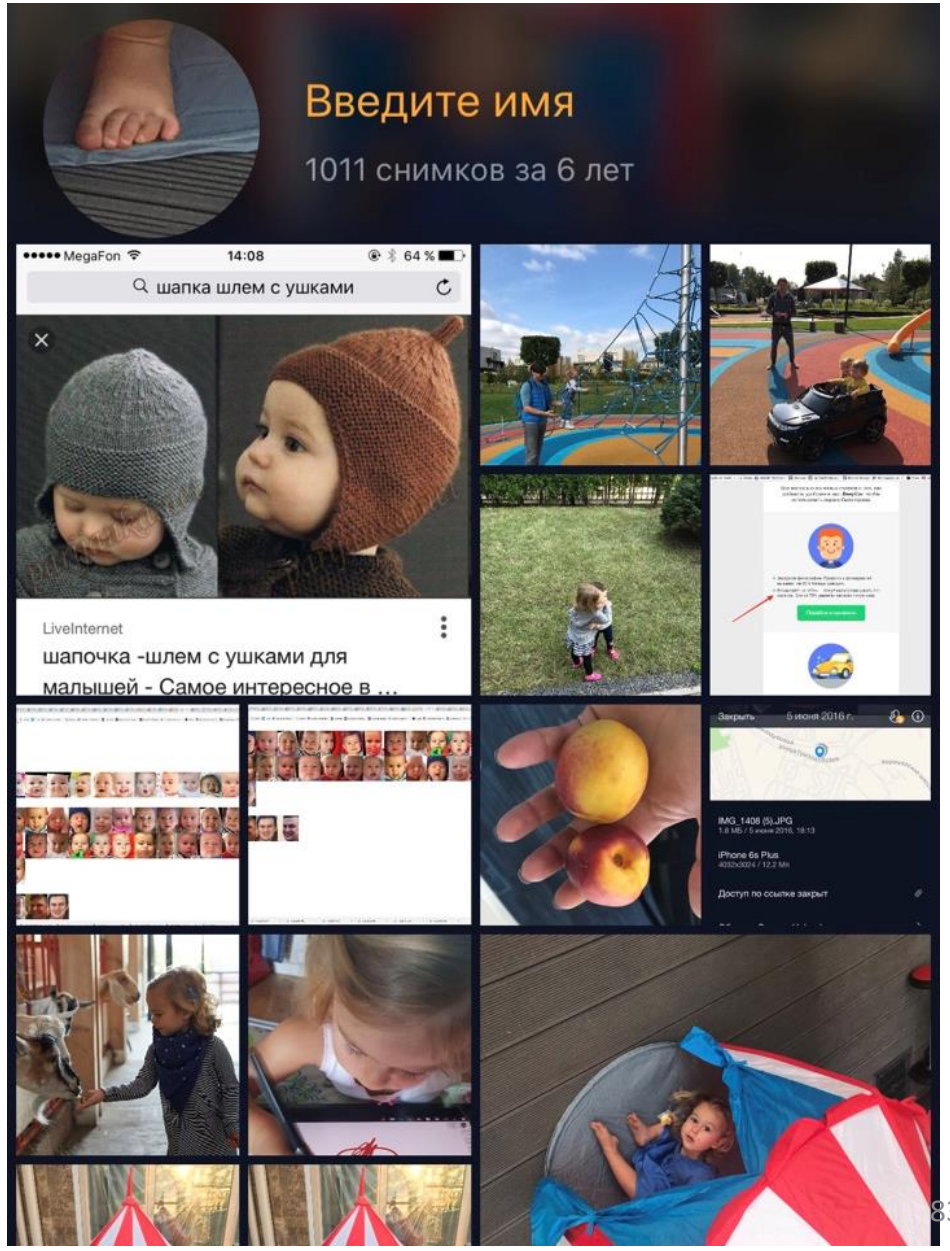
- One more branch in Face Recognition CNN
- Small overhead



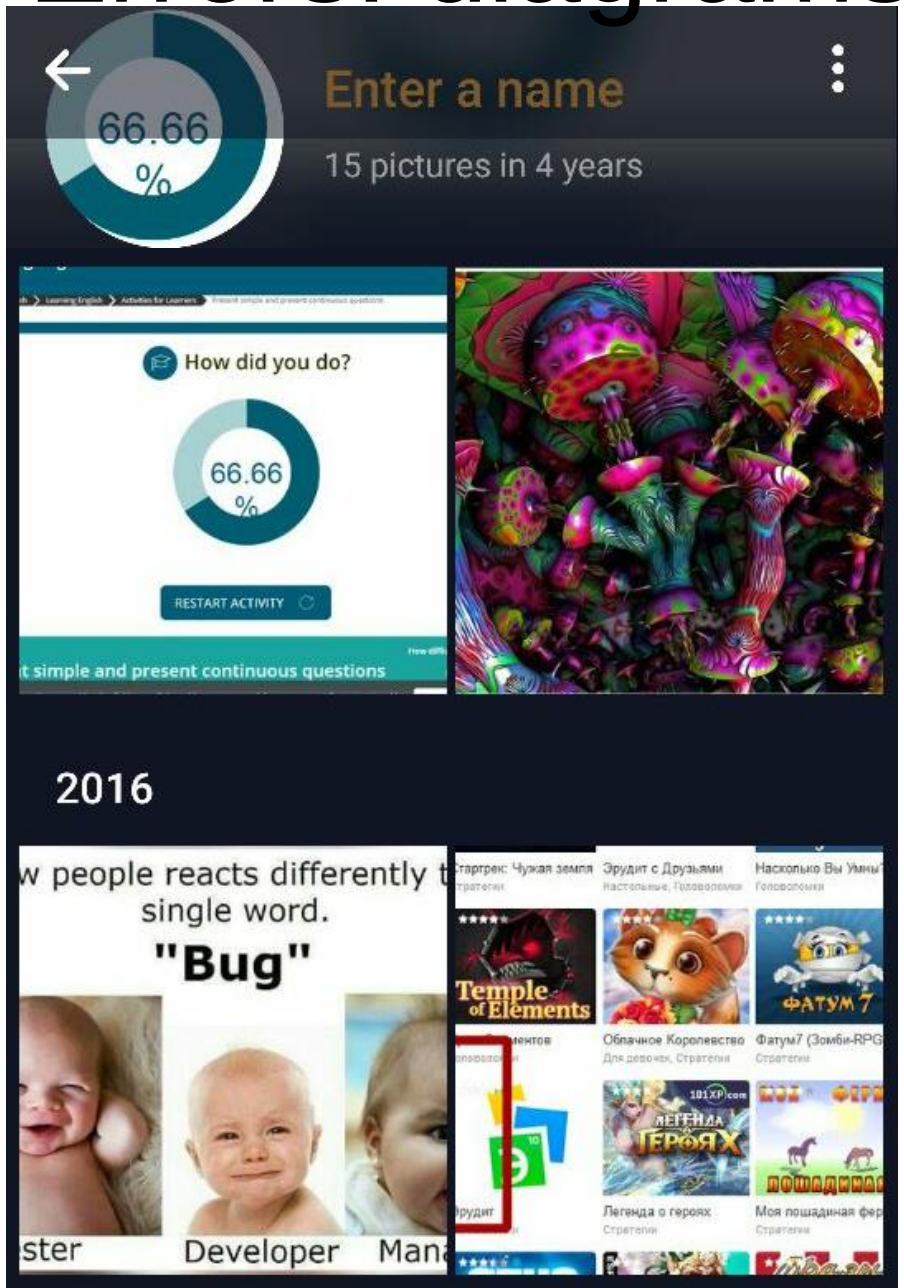
# Errors: body parts

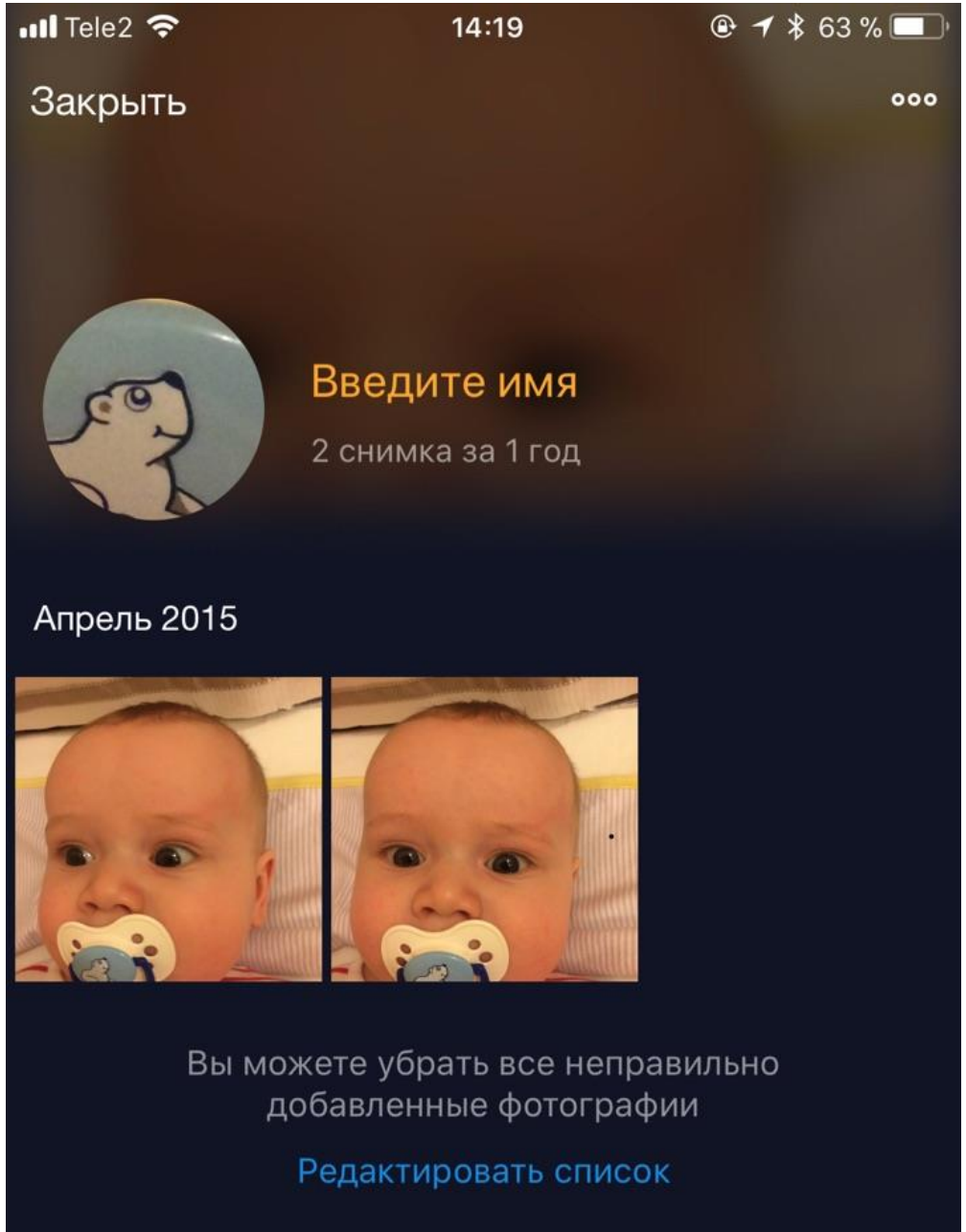



Detection mistakes form clusters



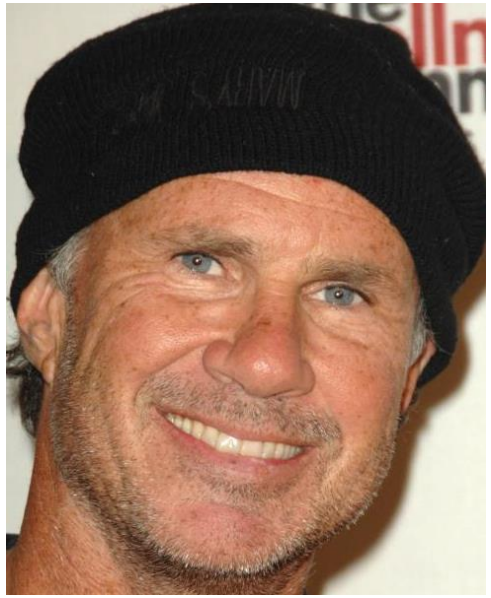
# Errors: diagrams & mushrooms

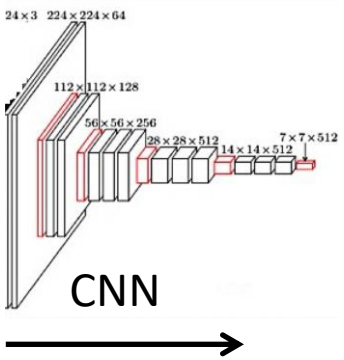


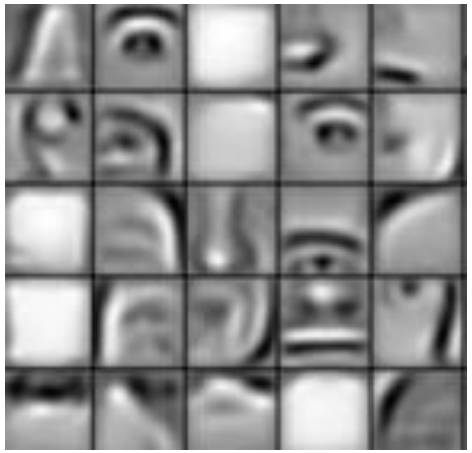
# Fixing trash clusters

There is similarity between “no faces”!






Embedding

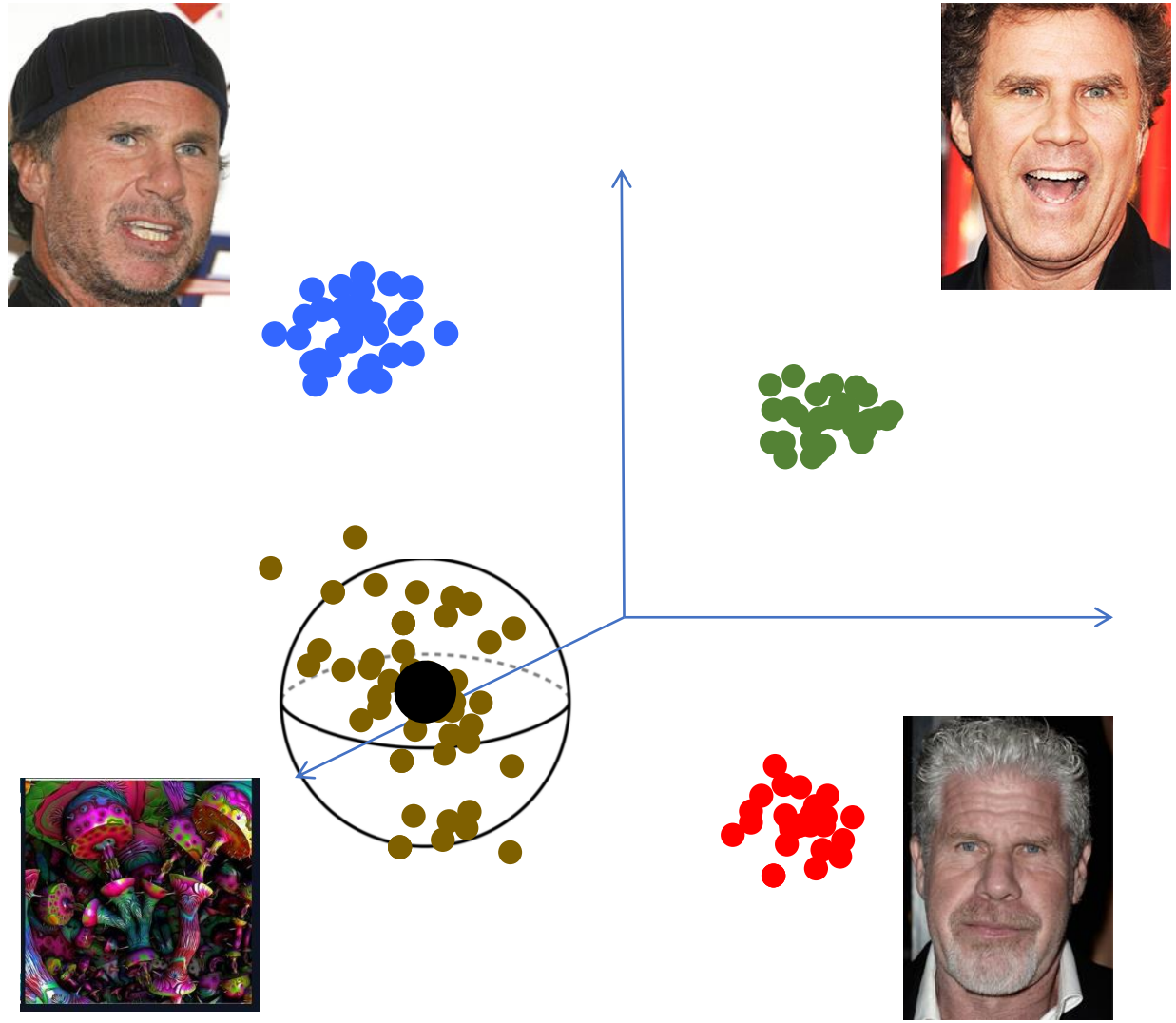


Specific activations

# Workaround

**Algorithm**

1. Construct «trash» dataset
2. Compute average embedding
3. Every point inside the sphere – trash



**Results**

- ROC AUC 97%

# AKNN

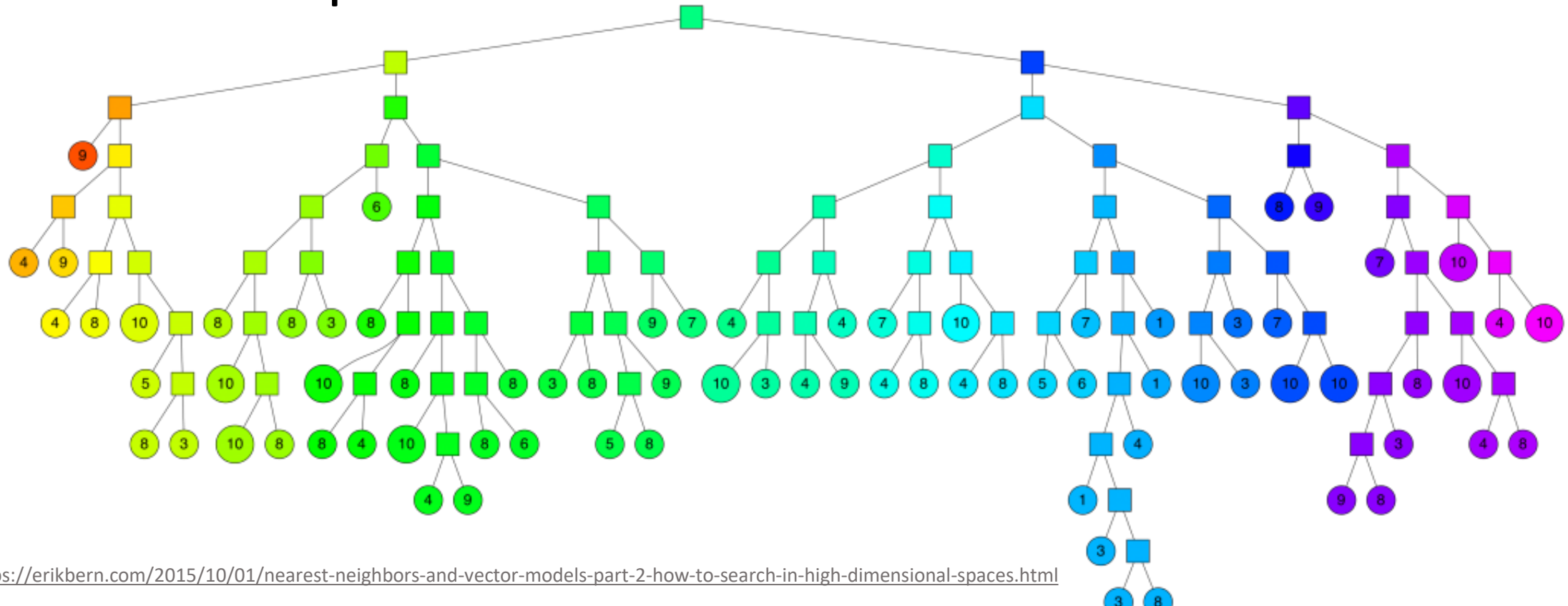
# Problem formulation

**What if you've got 100M persons in the database, how to efficiently search it ?**

# AKNN: Annoy

Forest of Binary Trees

1. Node: Pick 2-points at random –> split space
2. Until subspace isn’t numerous

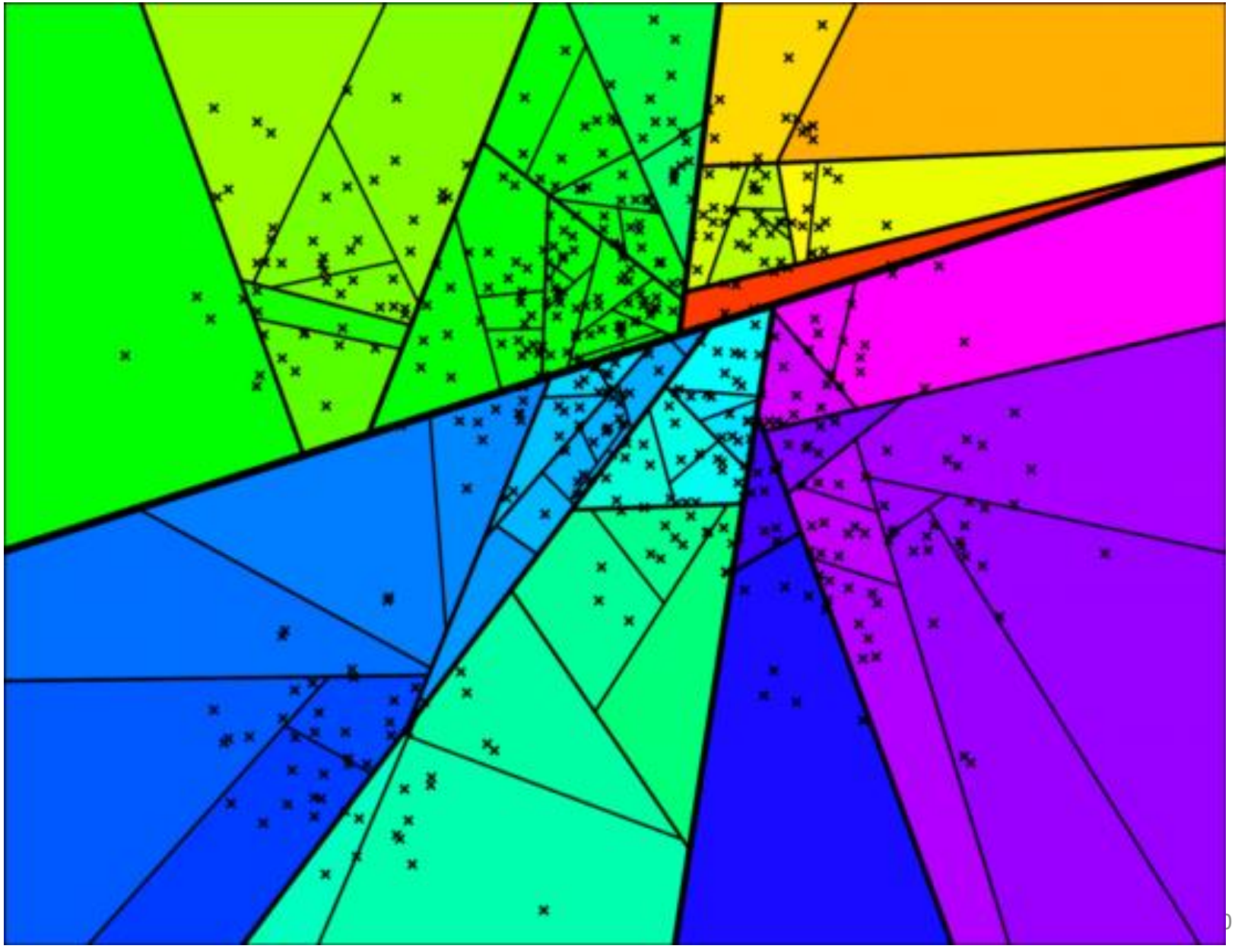


Source: https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

# Annoy: pros & cons

mail.ru group

+ Simple
- No GPU support
- High Memory Consumption
- Slow



# Hierarchical Navigable Small World

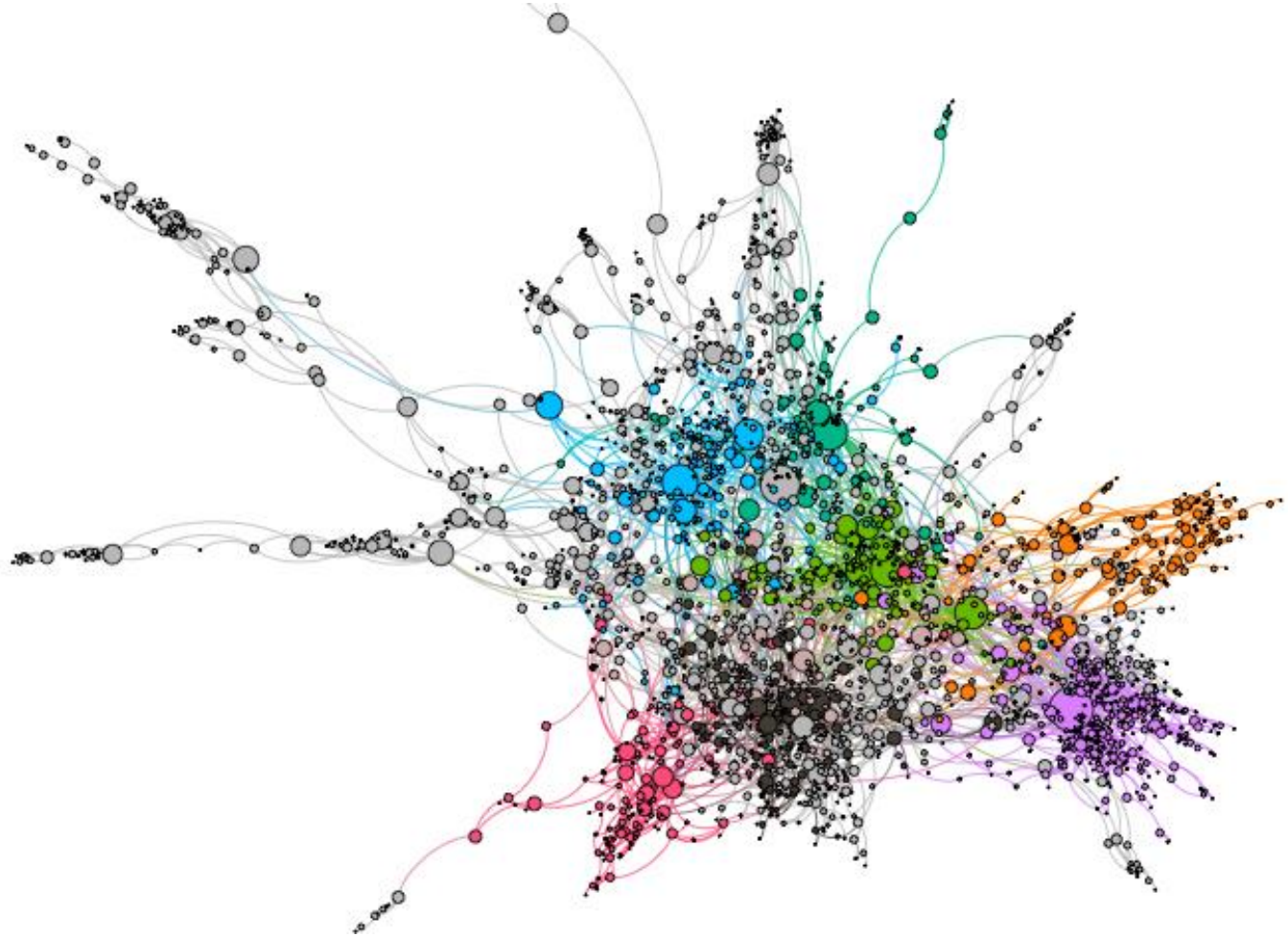
mail.ru group

## Small World:

1. highly transitive
2. small average distance

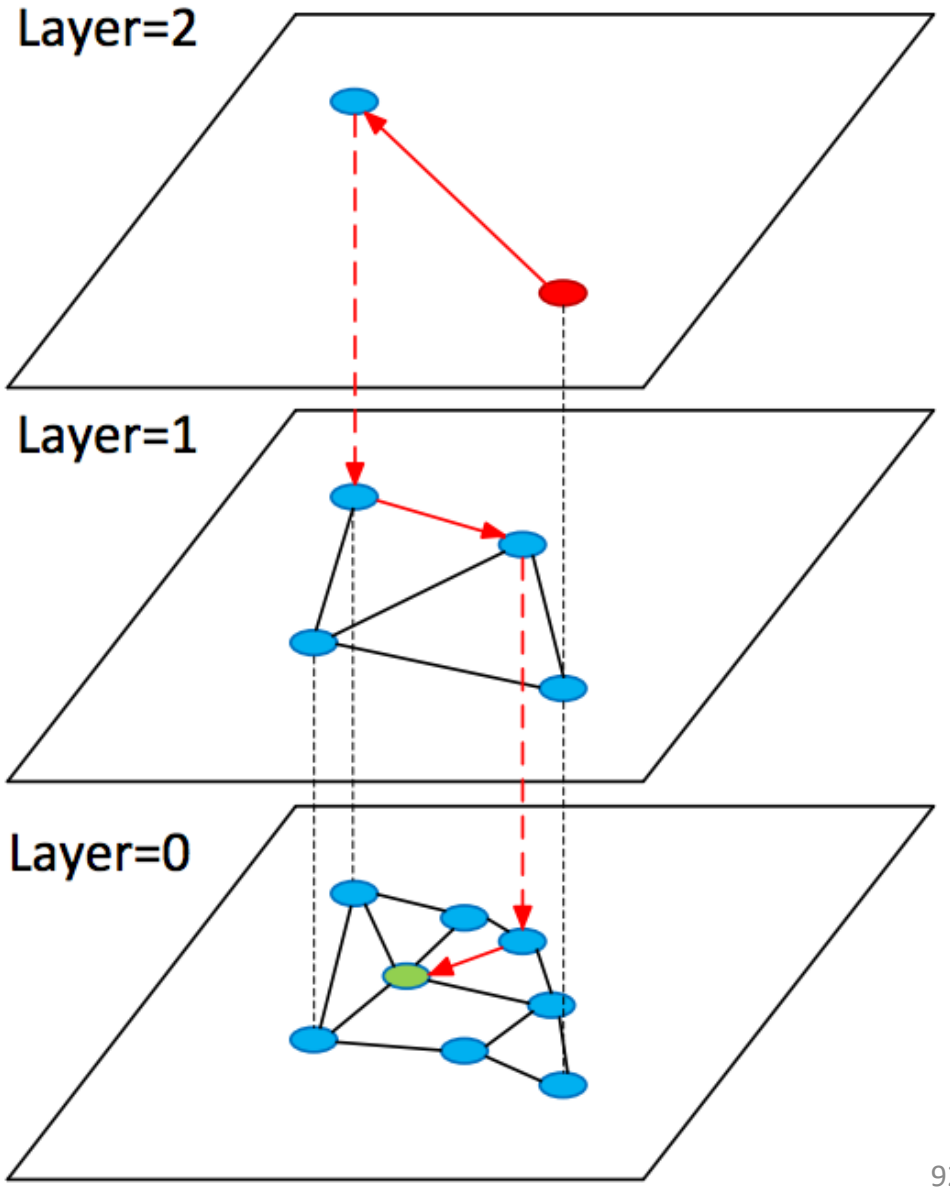
## Traversal Algorithm

1. Start from a random node
2. Pick smallest edge
3. Repeat until convergence



# HNSW

1. Hierarchy:
    1. top –> few points
    2. bottom –> all points

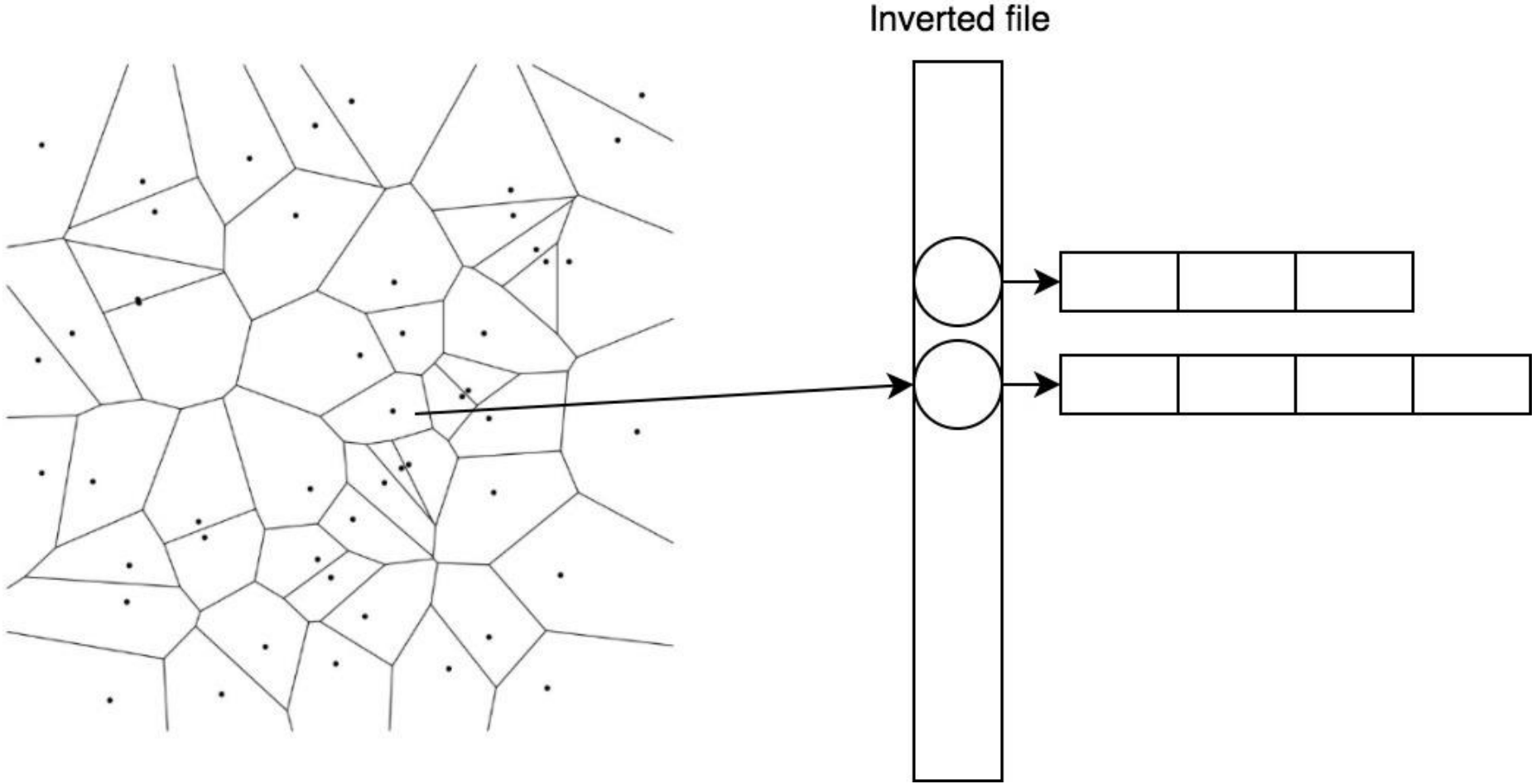


# HNSW: pros & cons

+ High accuracy
+ Fastest
+ Batching
– High memory consumptions

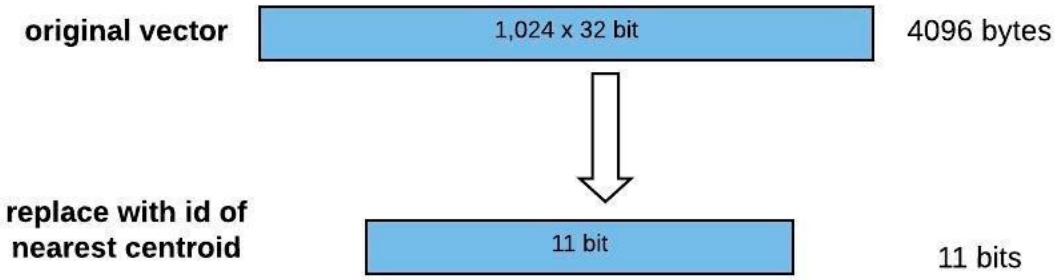
# FAISS: inverted file

- Split space by K-Means
- On inference: pick several closest Centroids



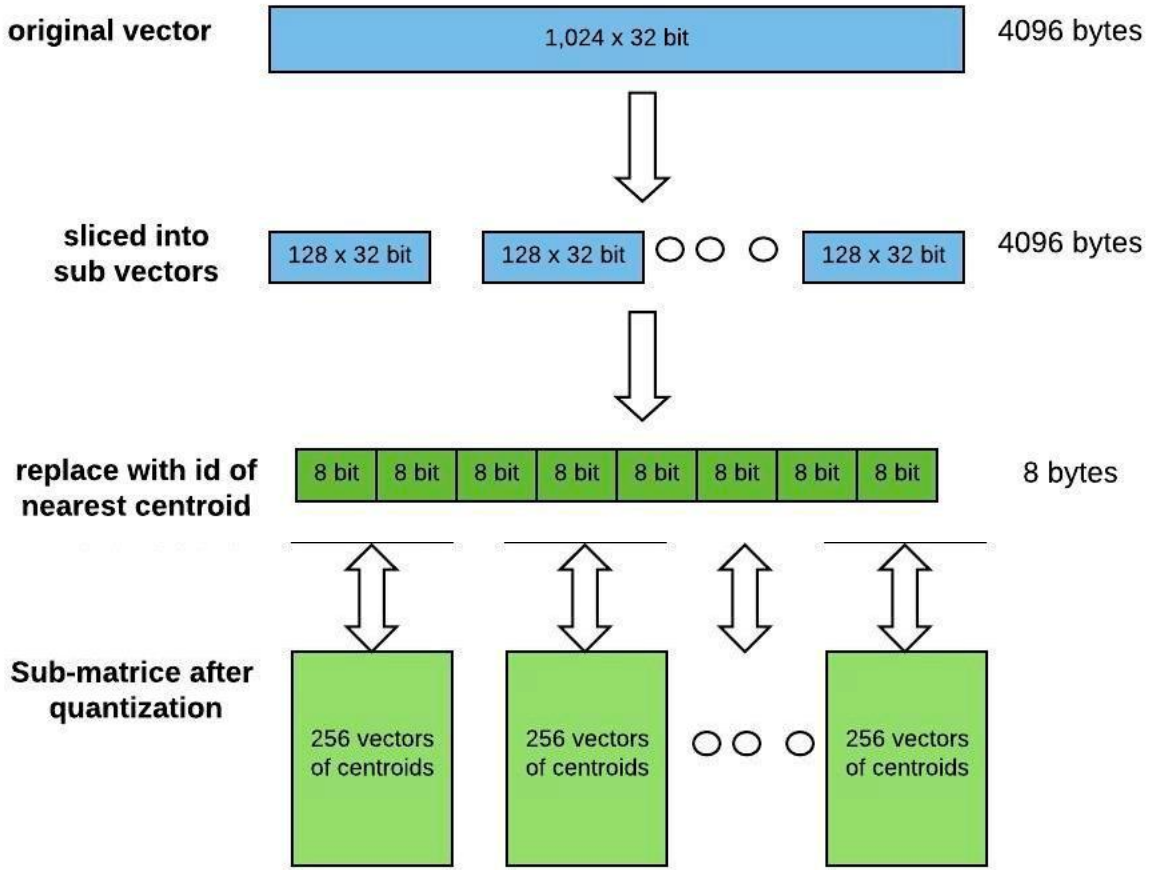
# FAISS: Product Quantization

## Goal: compress memory




2042 centroids can represent each vector with 11 bits

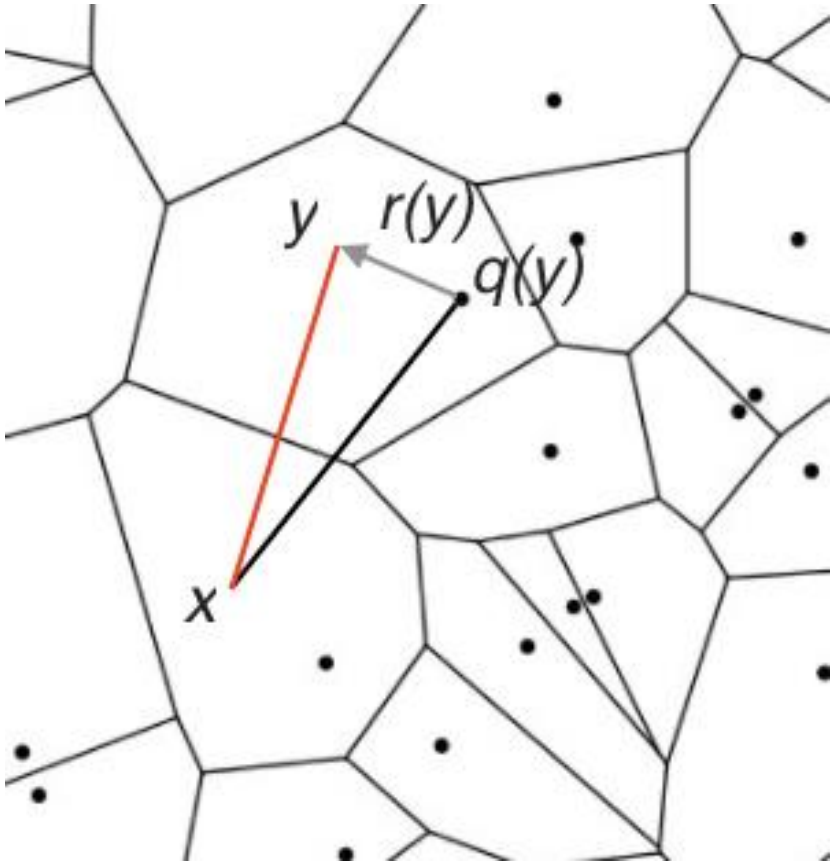
**To increase accuracy slice vectors**

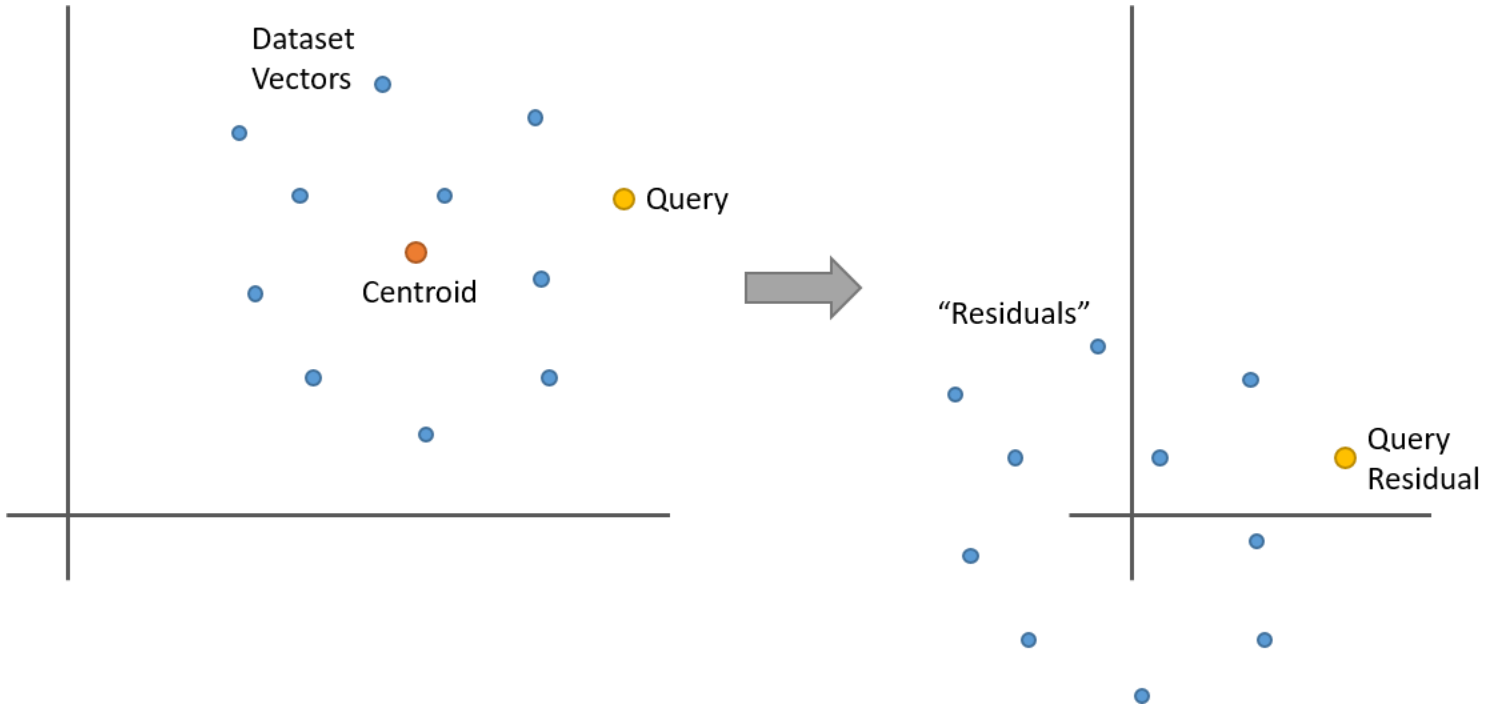



Source: https://towardsdatascience.com/comprehensive-guide-to-approximate-nearest-neighbors-algorithms-8b94f057d6b6

# FAISS: residual

- To increase accuracy: reduce the variation inside clusters
- $r(y) = y - q_c(y), q_c - quantized\ centroid$



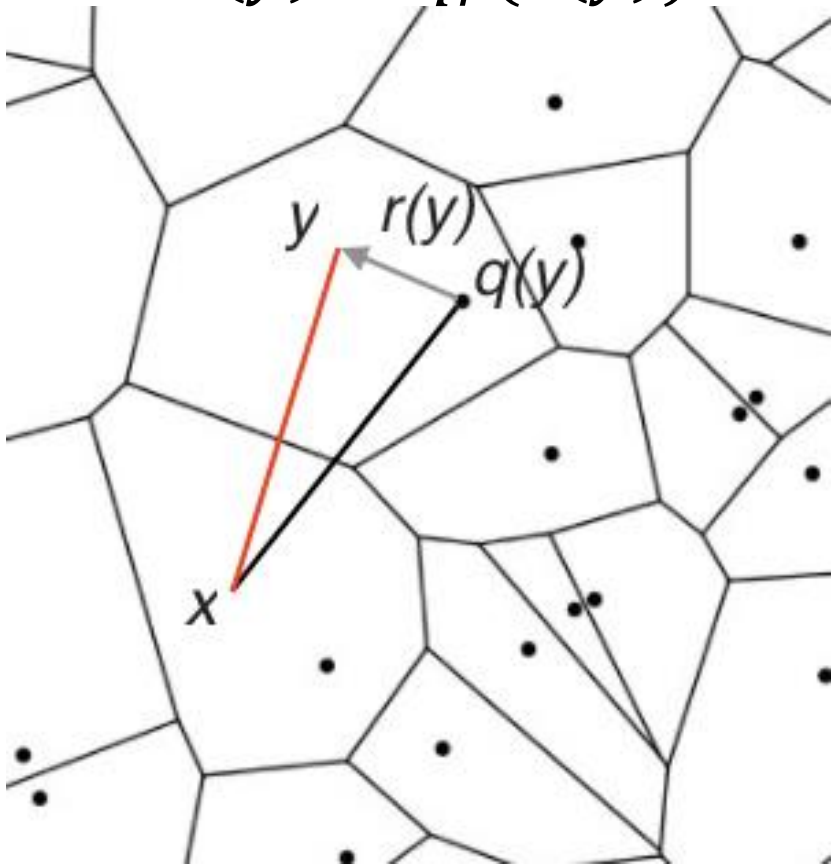



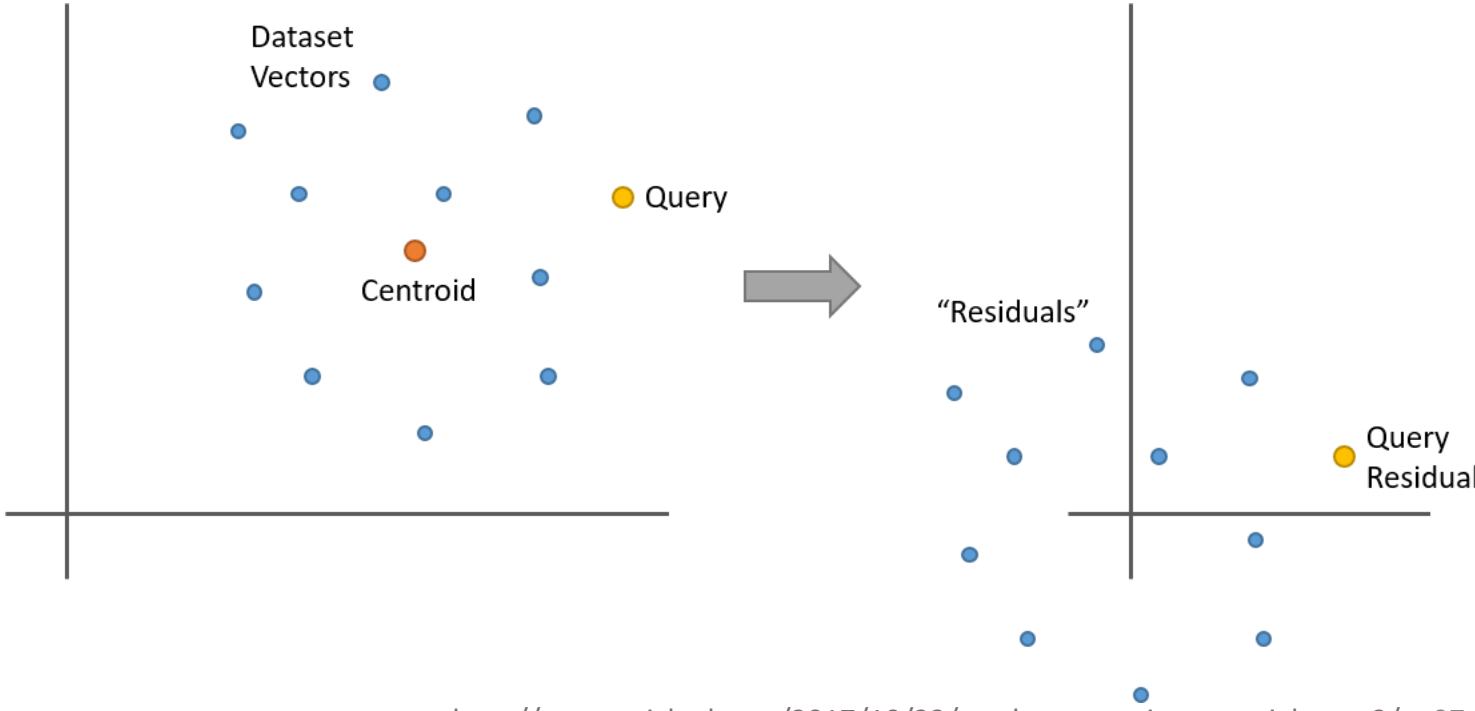

http://mccormickml.com/2017/10/22/product-quantizer-tutorial-part-2/

# FAISS: residual

— points are centered around 0

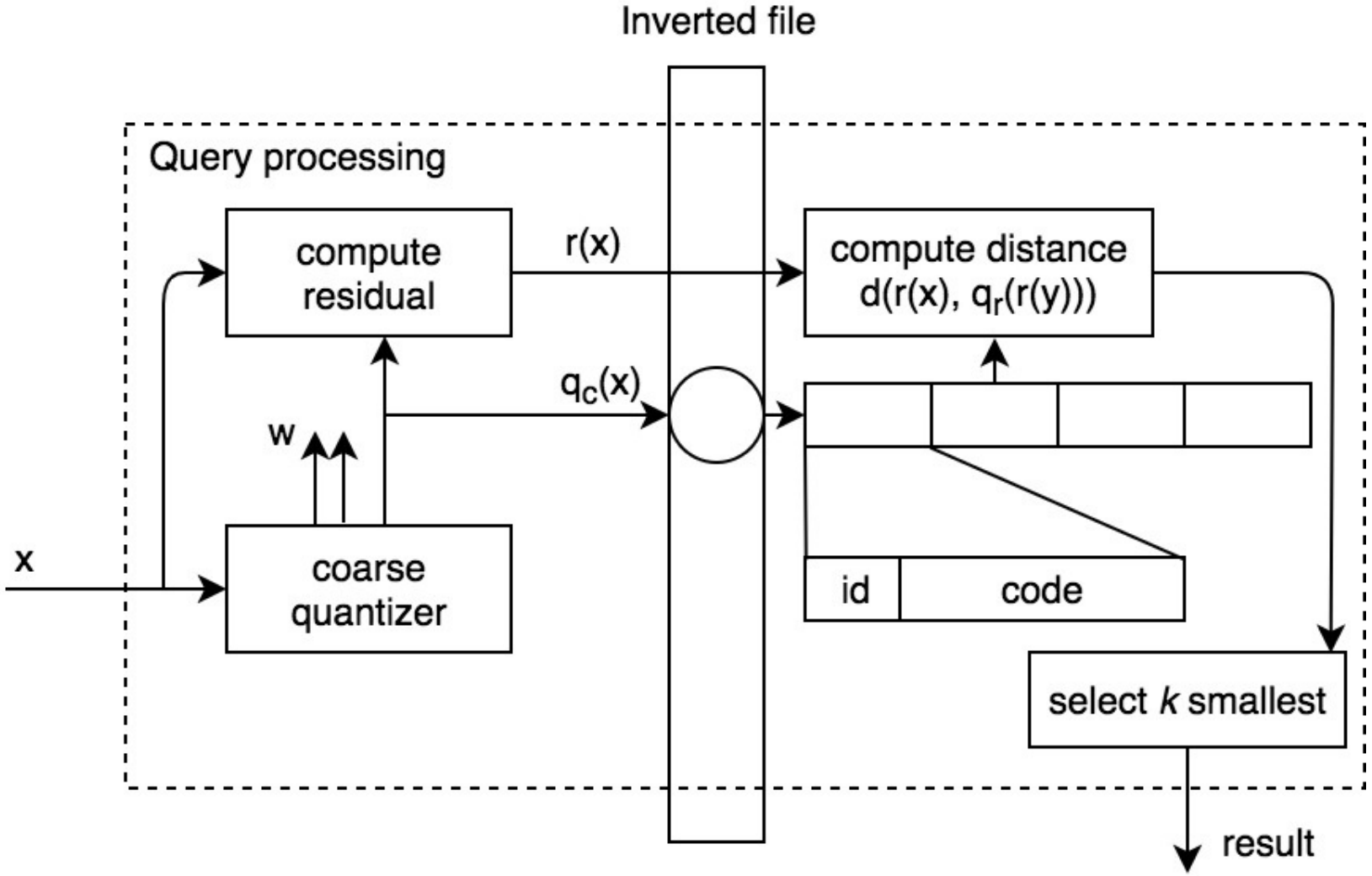
- By reducing the variety in the dataset, it takes fewer prototypes to represent the vectors effectively!
- $r(y) \cong q_r(r(y))$




http://mccormickml.com/2017/10/22/product-quantizer-tutorial-part-2/

# FAISS: all together



# FAISS: pros & cons

+ Great compression ratio
+ Batch
+ GPU support
– The exact nearest neighbor might be across the boundary to one of the neighboring cells

# Comparison

The only candidates: HNSW & Faiss (faiss library)

– We trade *compute* vs *memory + accuracy*

| Method | search time | 1-R@1 | index size | index build time |
|---|---|---|---|---|
| Flat-CPU | 9.100 s | 1.0000 | 512 MB | 0 s |
| nmslib (hnsw) | 0.081 s | 0.8195 | 512 + 796 MB | 173 s |
| IVF16384,Flat | 0.538 s | 0.8980 | 512 + 8 MB | 240 s |
| IVF16384,Flat (Titan X) | 0.059 s | 0.8145 | 512 + 8 MB | 5 s |

**1M vectors**

# Recap

# Recap

1. ArcFace > Center loss, Triplet, …
2. Triplet can handle arbitrary large datasets
3. Good embedding space provides means for cleaning your data
4. AKNN: Faiss vs HNSW

# Контакты

tg: @ed_tyantov

tyantov@corp.mail.ru

академия больших данных

mail.ru group

made