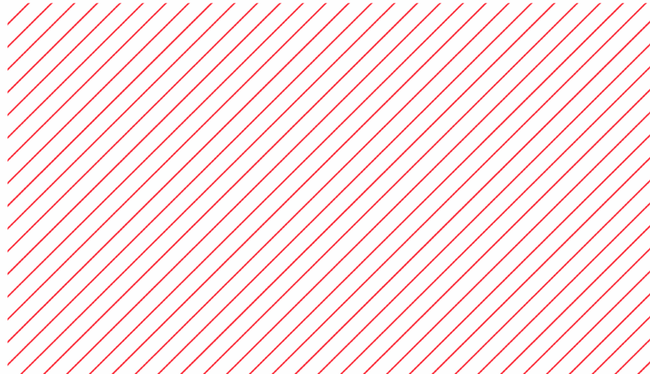
академия больших данных

mail.ru group

Metric learning

Андрей Бояров

Ведущий программист-исследователь в командах Машинного обучения почты и Машинного зрения Mail.ru, кандидат физ.-мат. наук



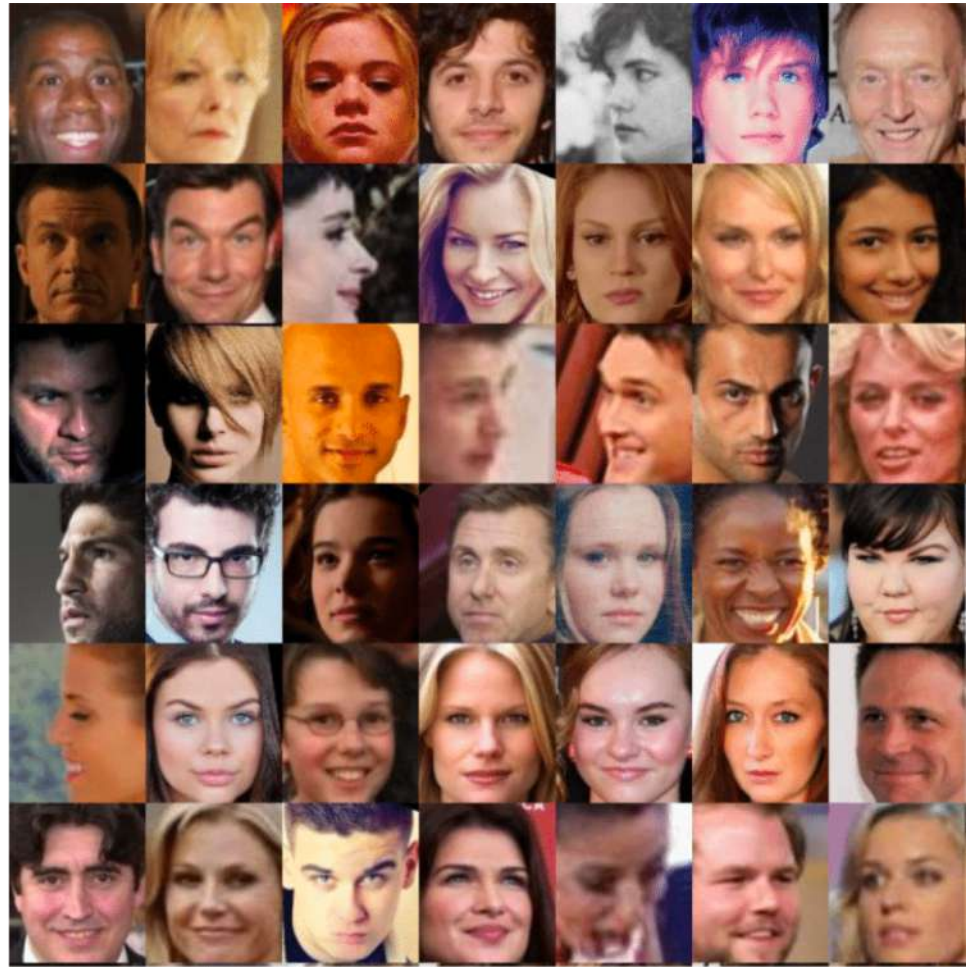
Lecture plan

1. Metric learning algorithms
    1. Siamese neural network
    2. Triplet loss
    3. Center loss
    4. ArcFace
2. Approximate K Nearest Neighbor Search (AKNN Search)
3. Face recognition & Landmarks recognition

Face recognition

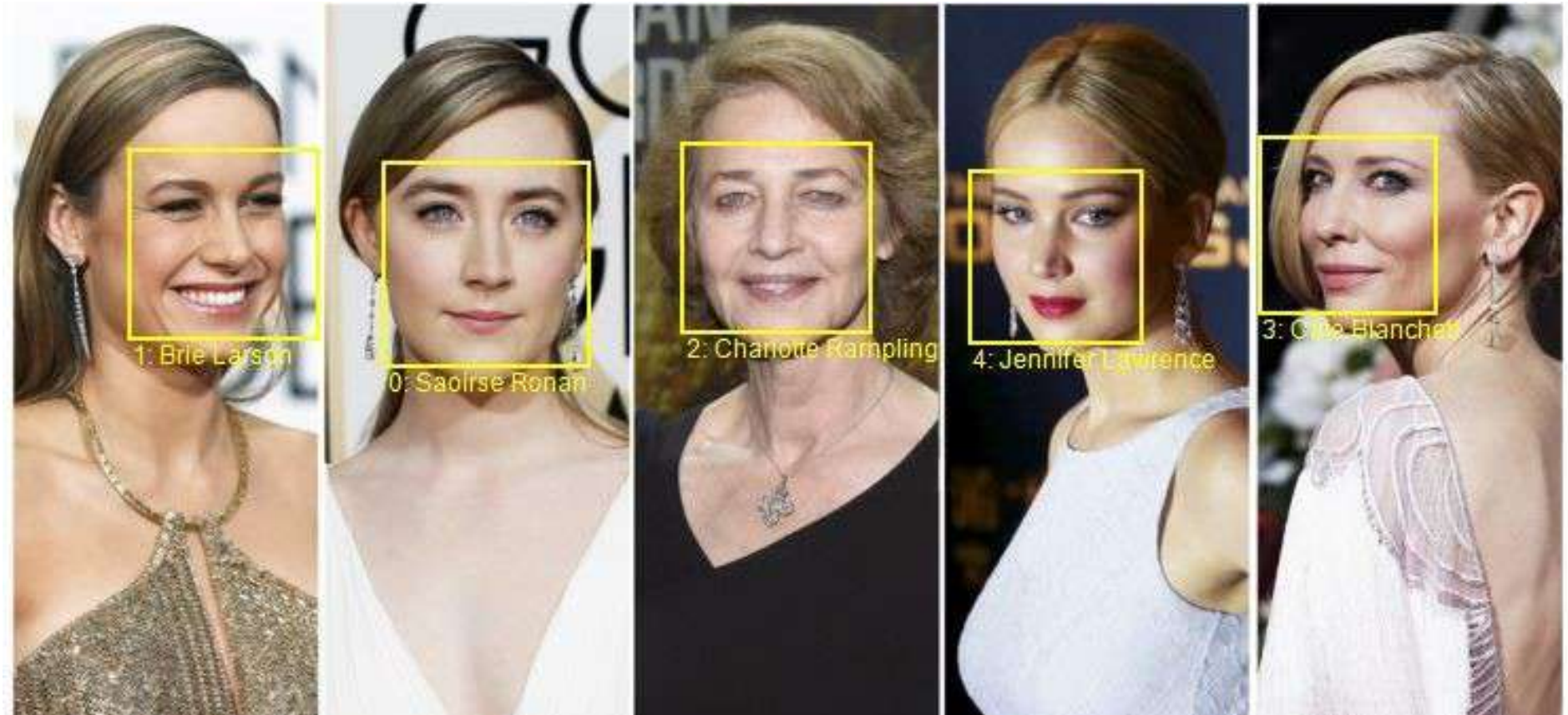
Face recognition task

- **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

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

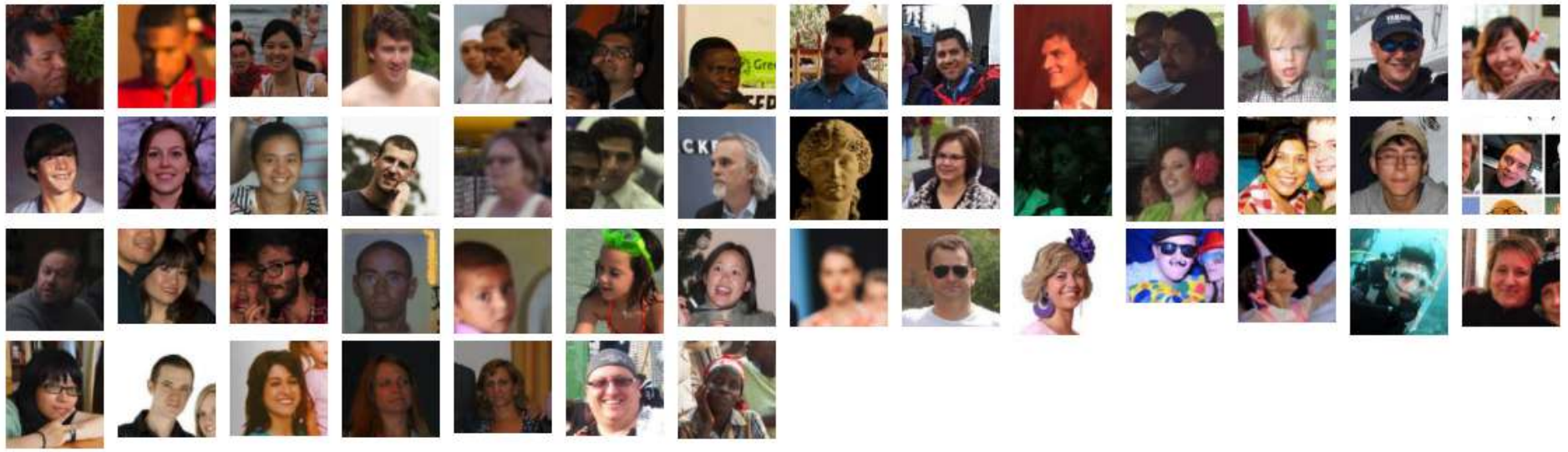
Explore Most Recent Public Results (last update 3/12/2017) ››

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

Participate and download Challenge 1 ››

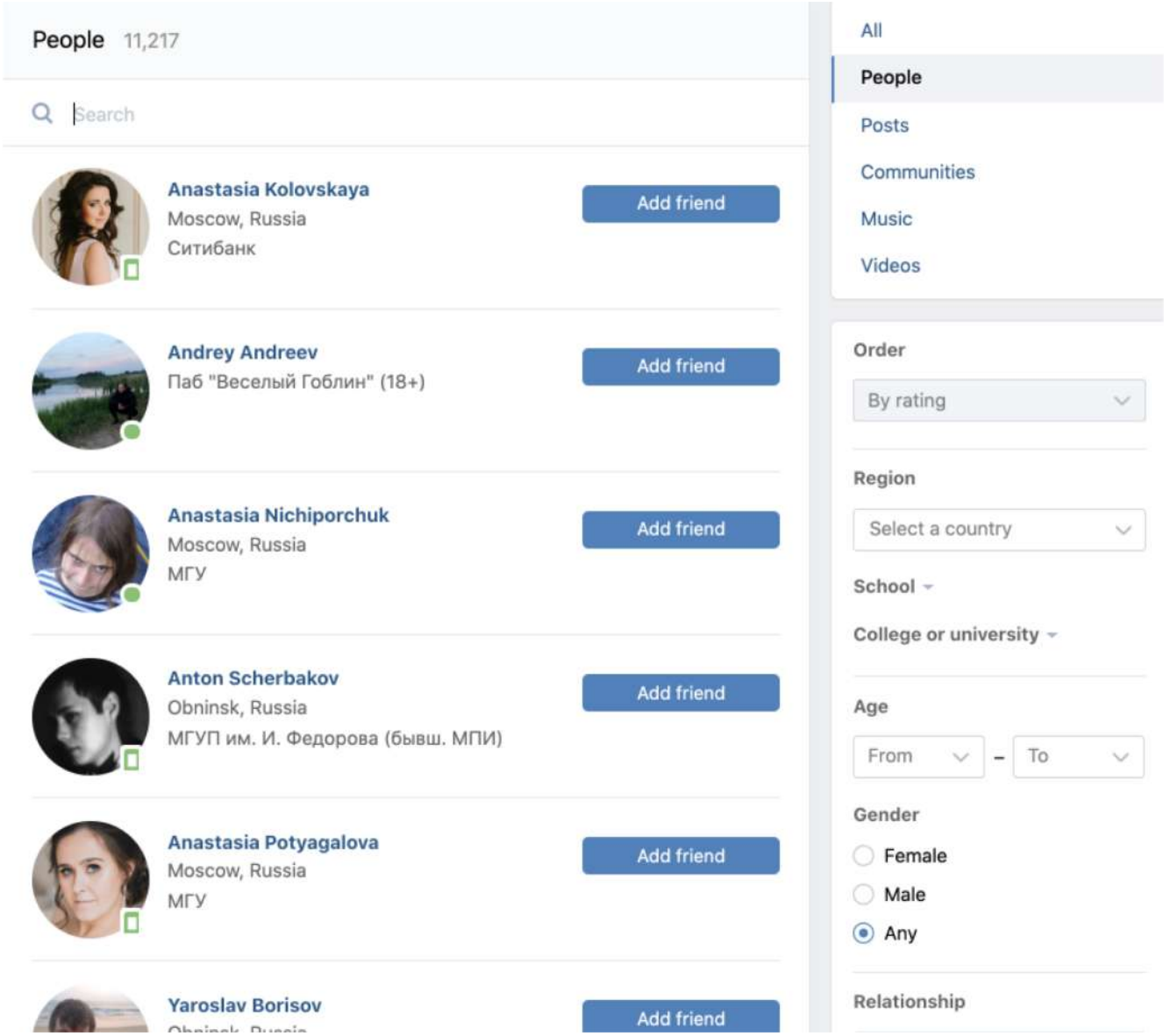
Challenge 2: Training on **672K** identities (4.7 Million photos), test at Million scale

Participate and download Challenge 2 ››



Our data

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



Metric learning

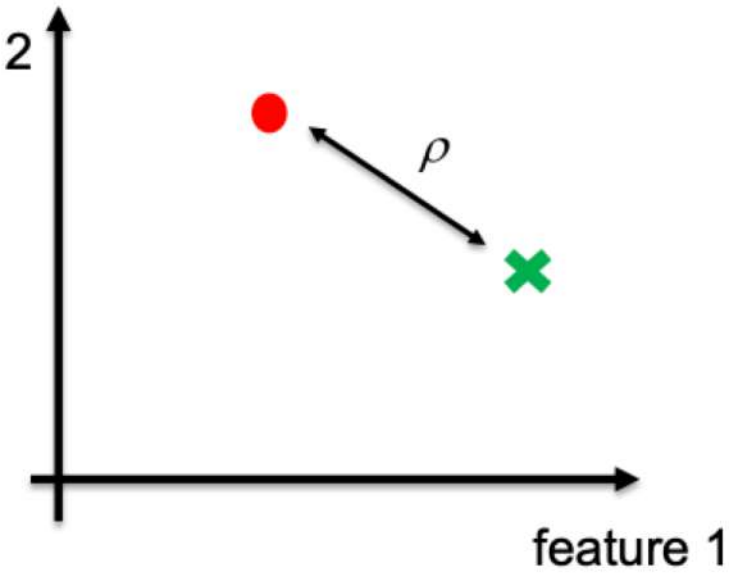
What's metric ?

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 $D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}.$
   - μ – mean, S – covariance matrix
   - can be ”trained” on data
   - μ=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:

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

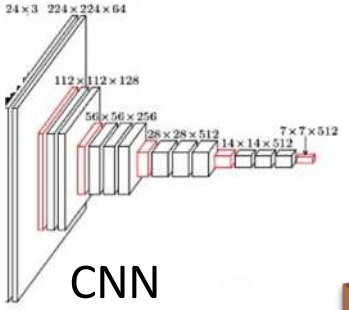
$$|A - B|_2^2 = (A - B)^T (A - B) =$$

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

Good for high dimensions

Metric learning for FR

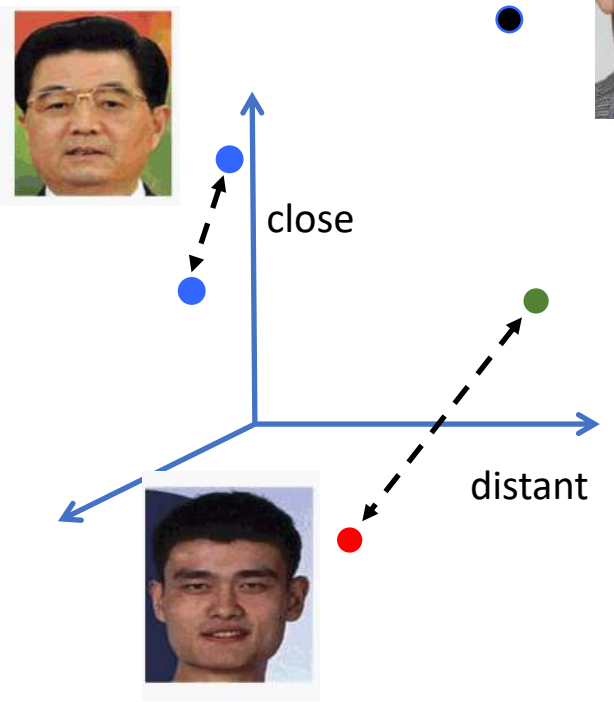
- Goal – to compare faces
- How? To learn metric
- To enable **Zero-shot** learning



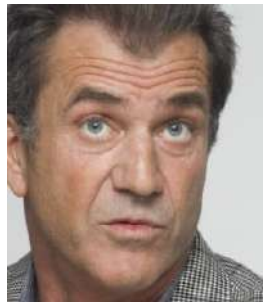
CNN

Embedding

Latent Space

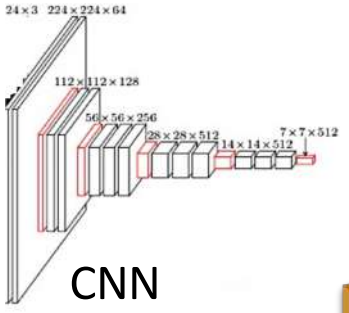


Unseen



Classification

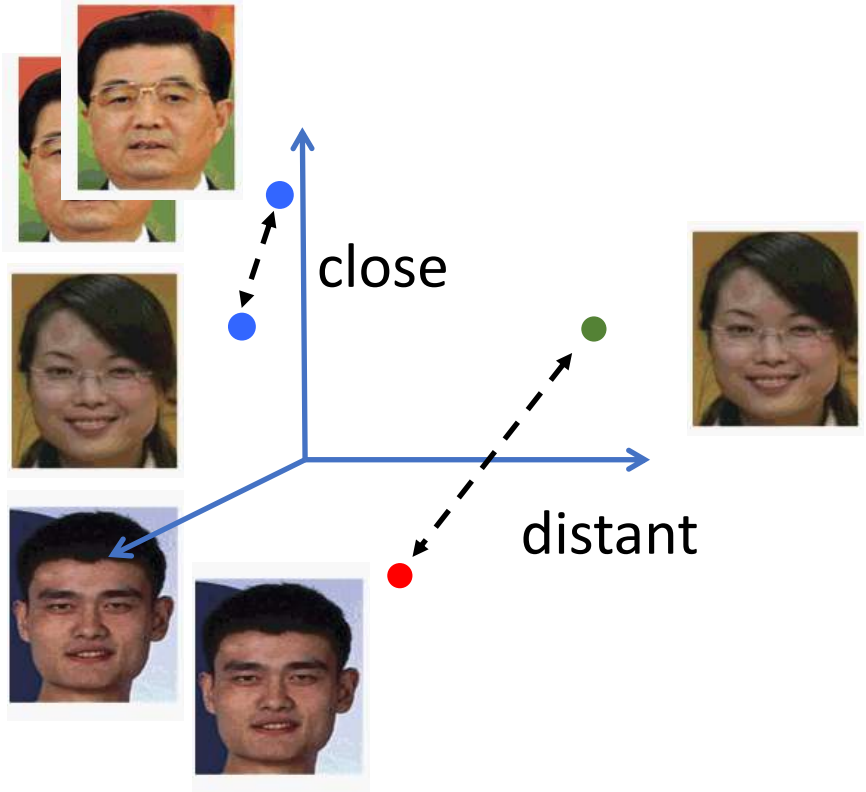
- Train CNN to predict classes
- Pray for good latent space



CNN

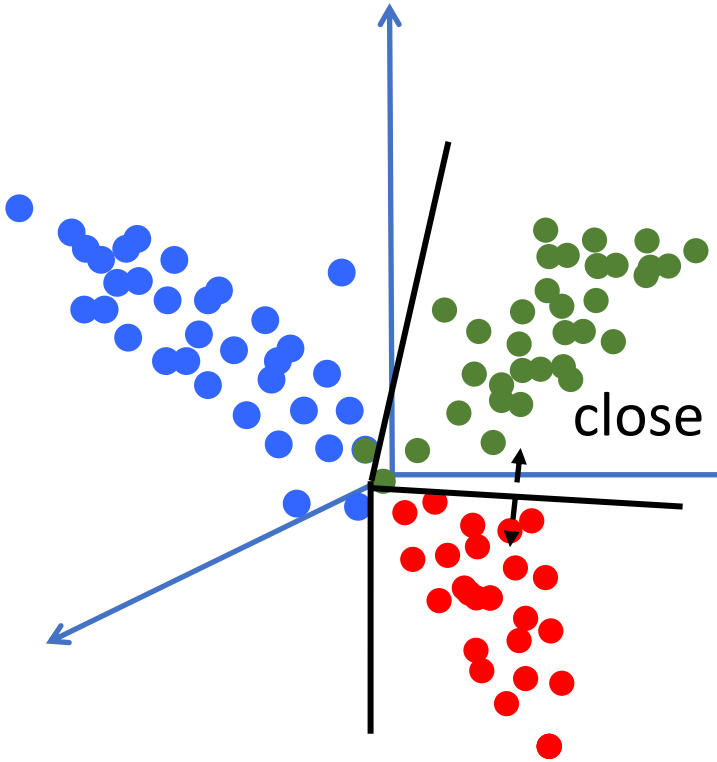
Embedding

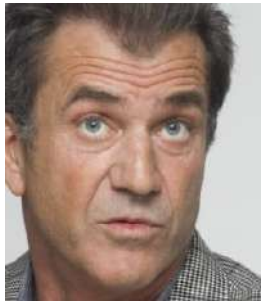
Classify



Softmax

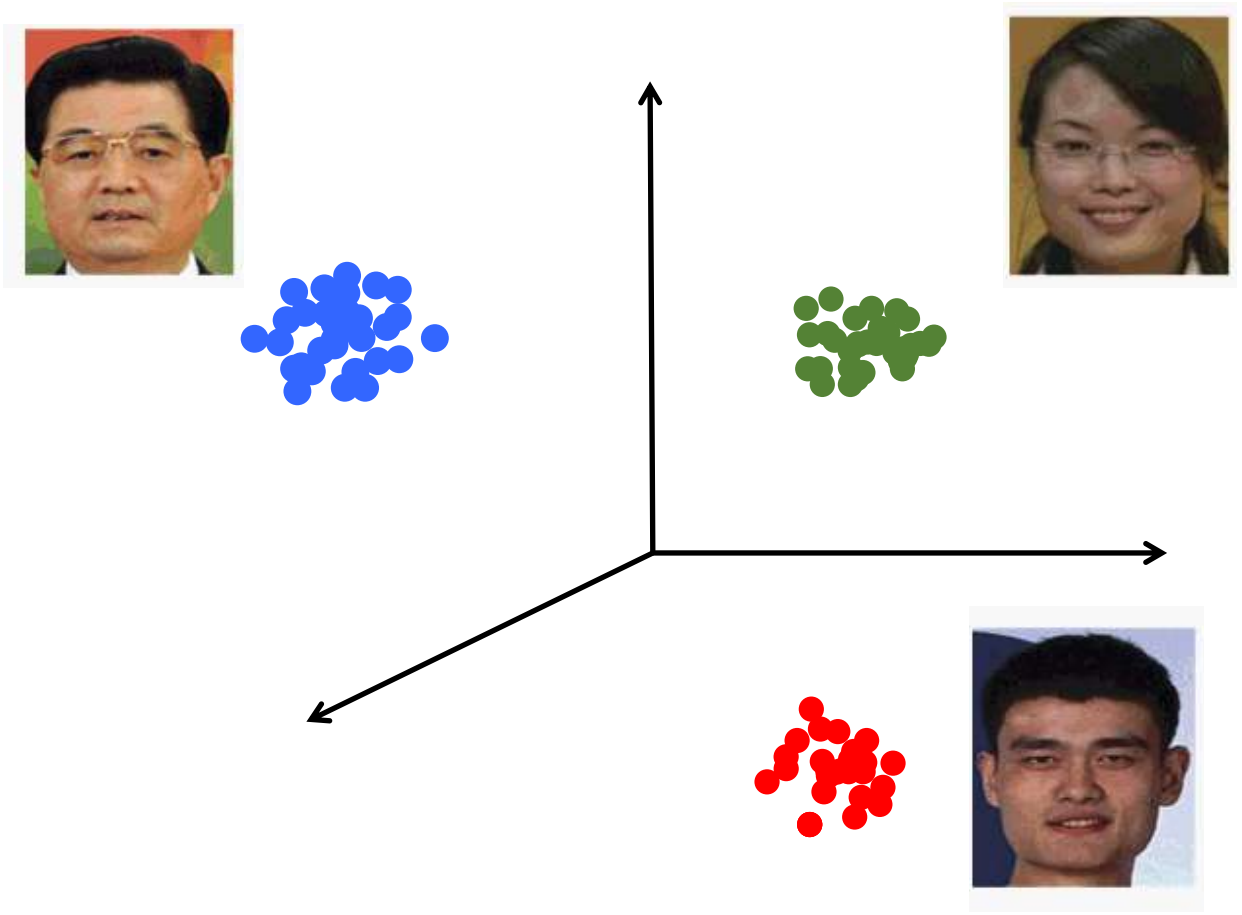
- Learned features only separable but not discriminative
- The resulting features are not sufficiently effective





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

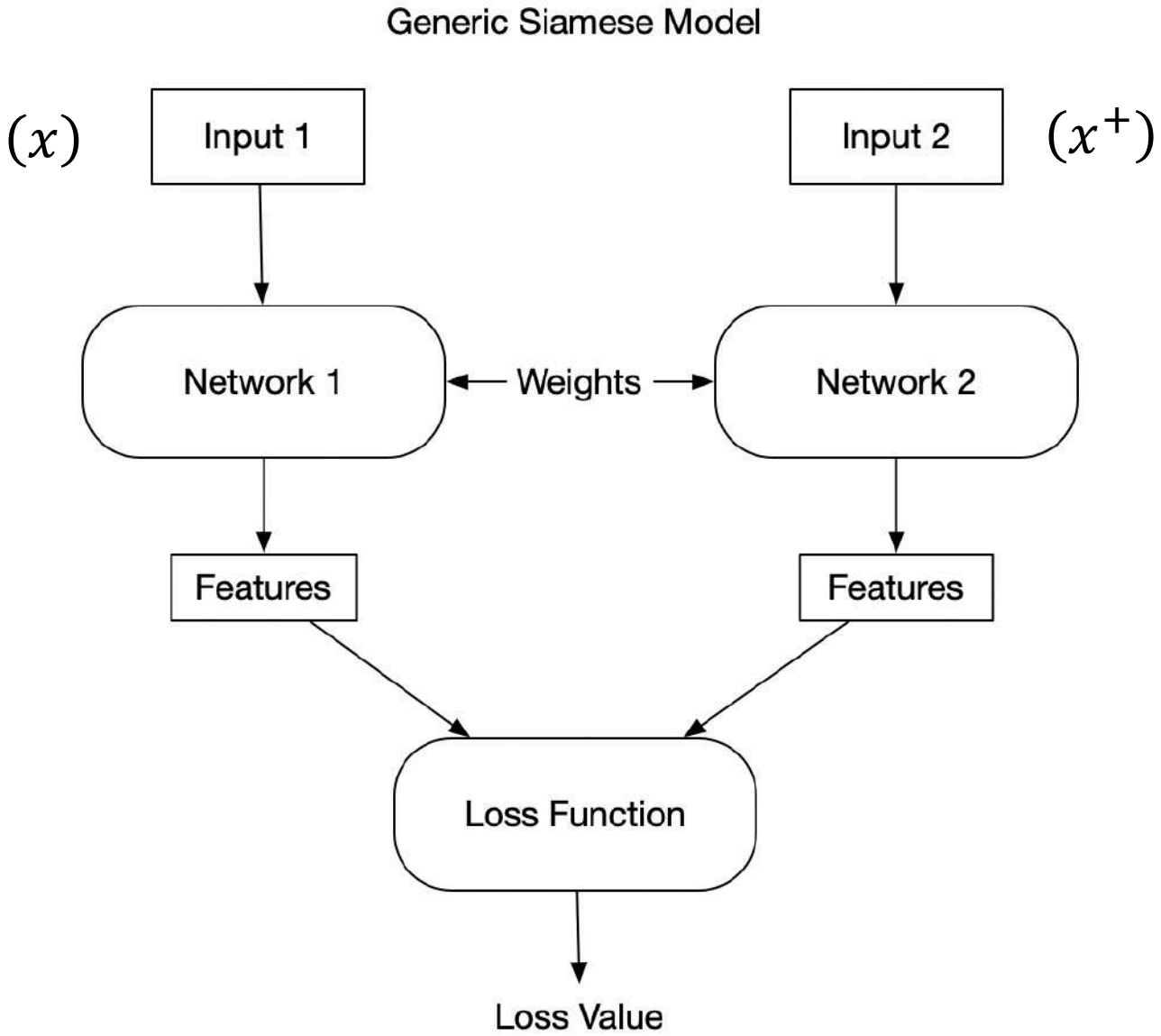
Similarity based

Classification similarity

Data: pairs of similar and not similar objects

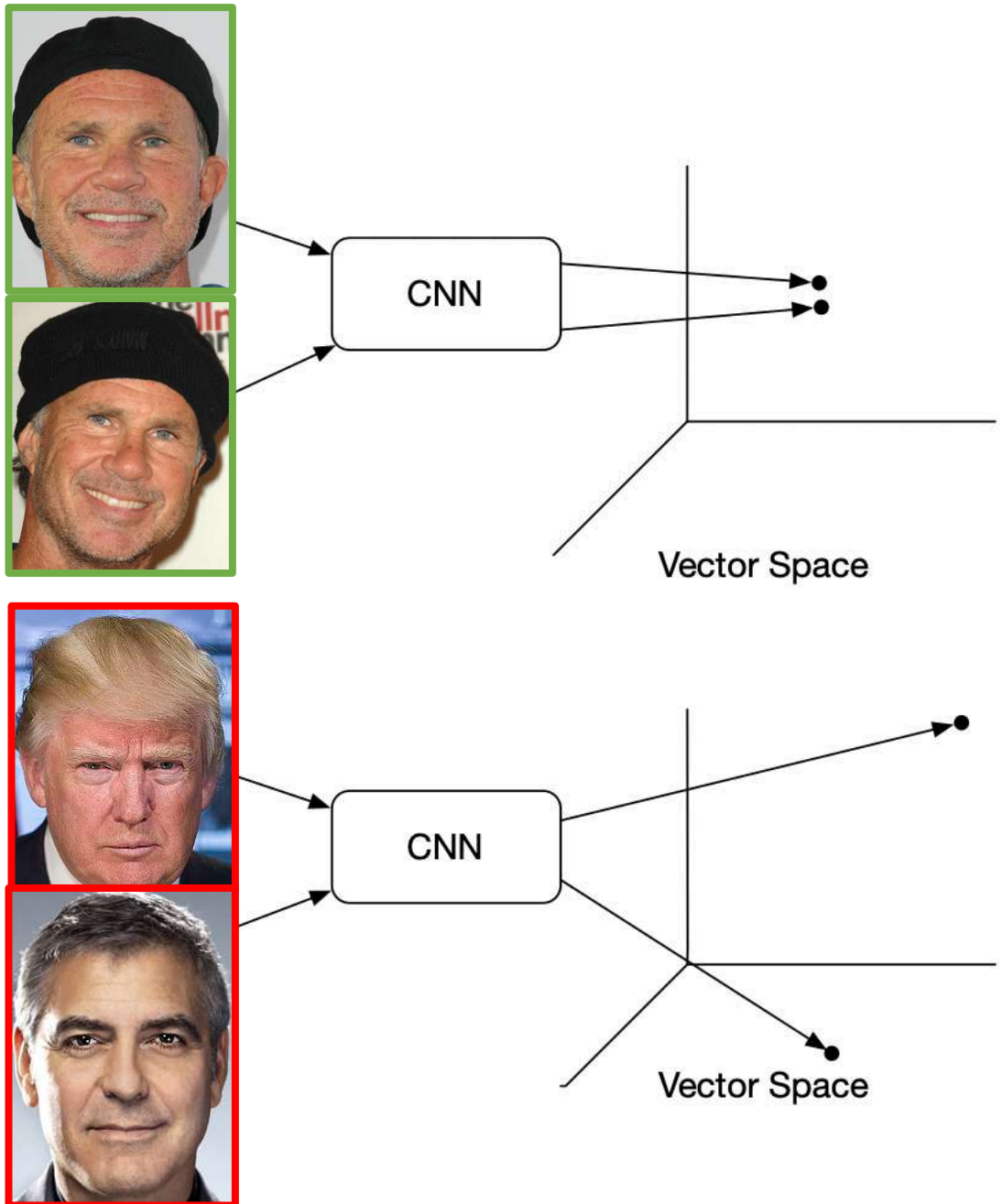
$$(x, x^+), (x, x^-)$$

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

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

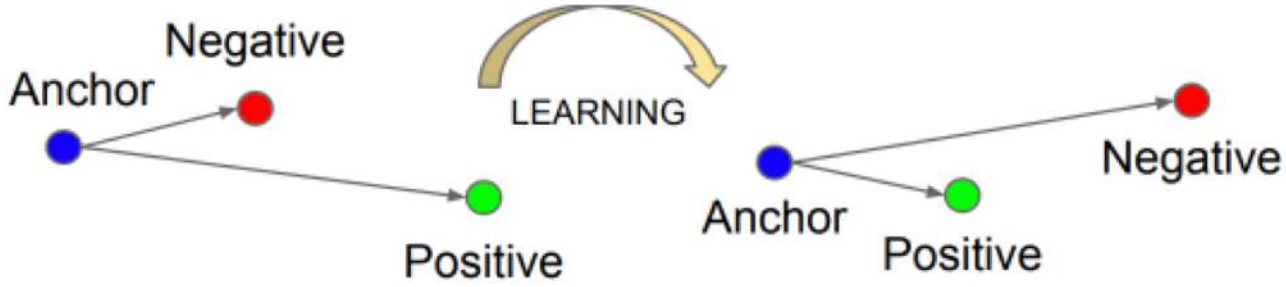
- $Optimal$: $\cos(x, x^+) = 1, \cos(x, x^-) = 0,\ l = -\log(1) = 0$

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

Positive

minimize

Anchor

maximize

Negative

positive + α < 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

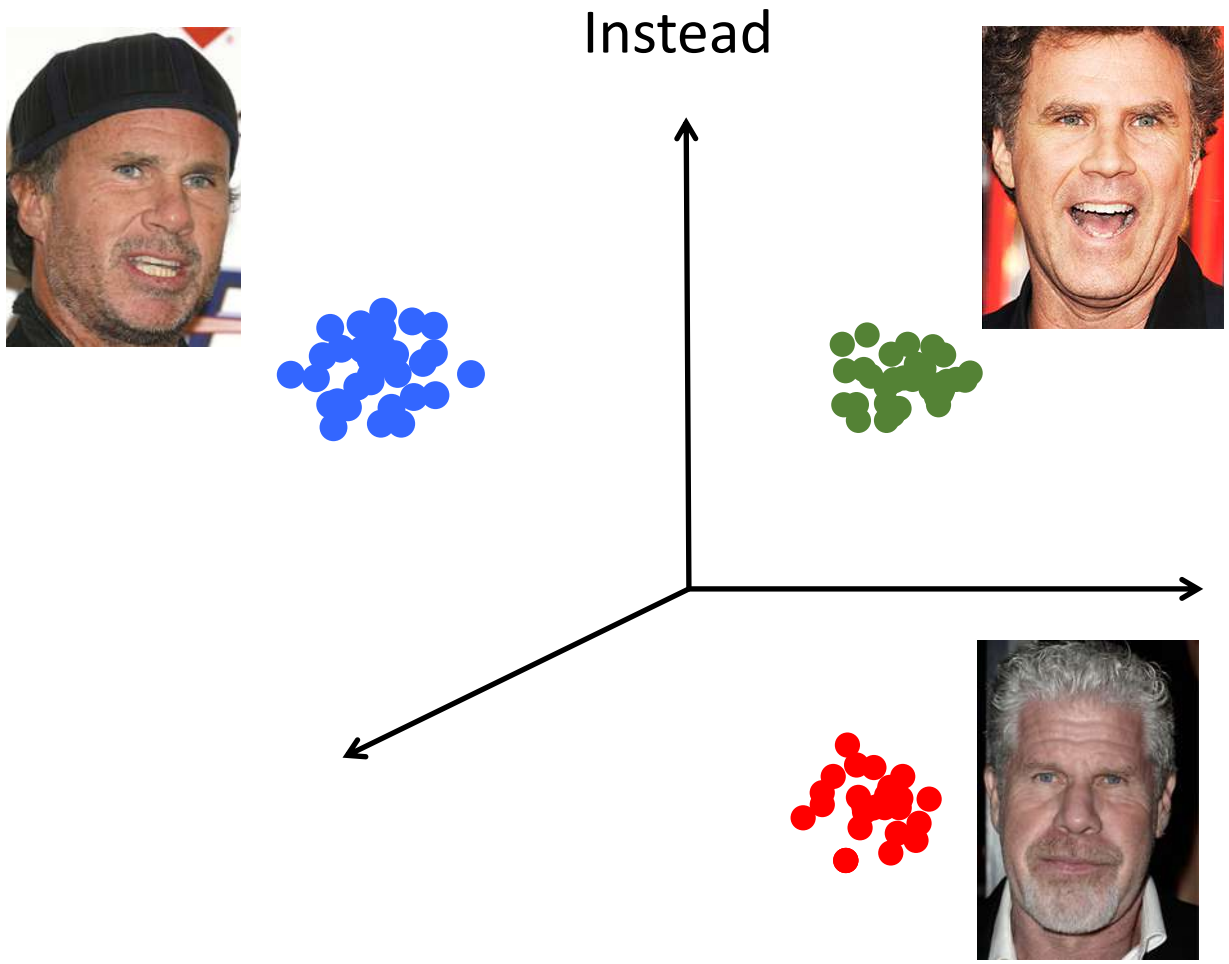
Anchor

maximize

Negative

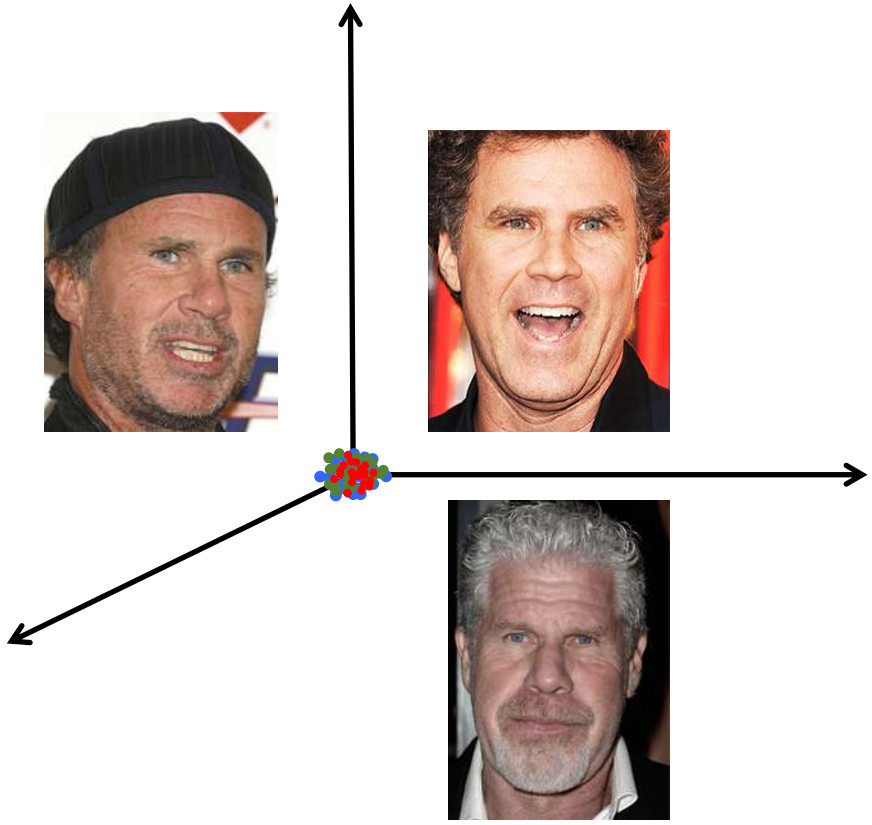
positive ~ negative

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

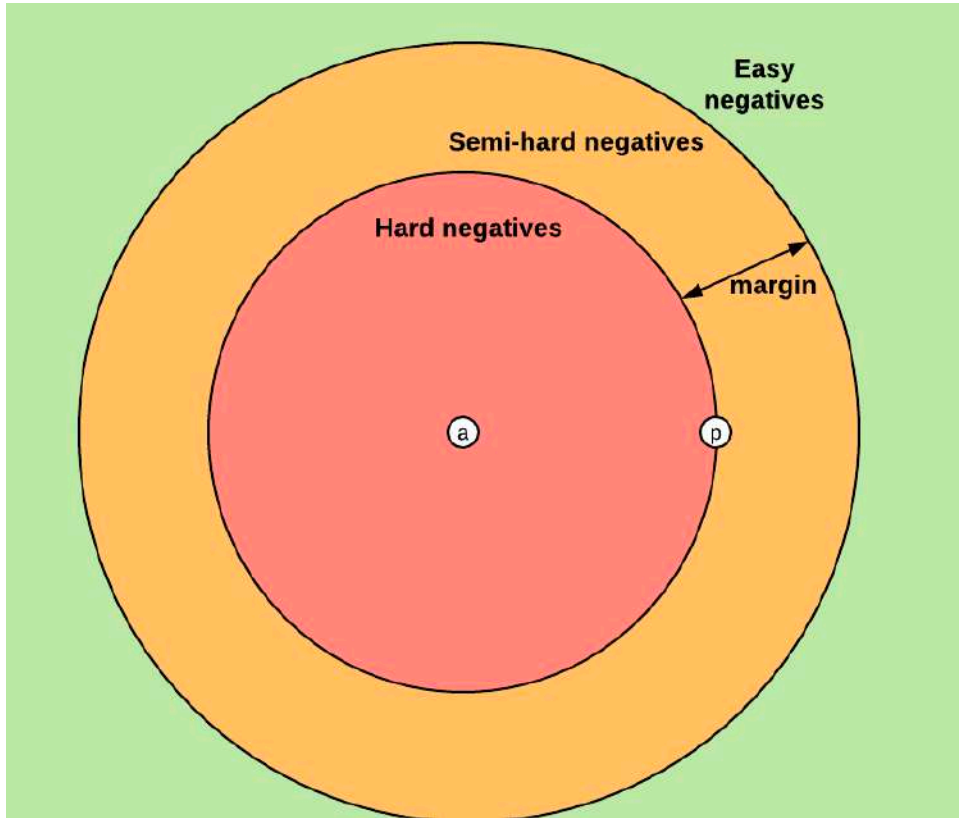
Semi-hard

Too easy

positive < negative < positive + α

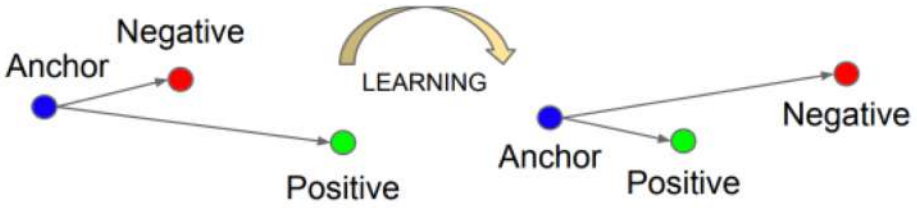
Online hard-negative mining

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



Triplet loss: summary

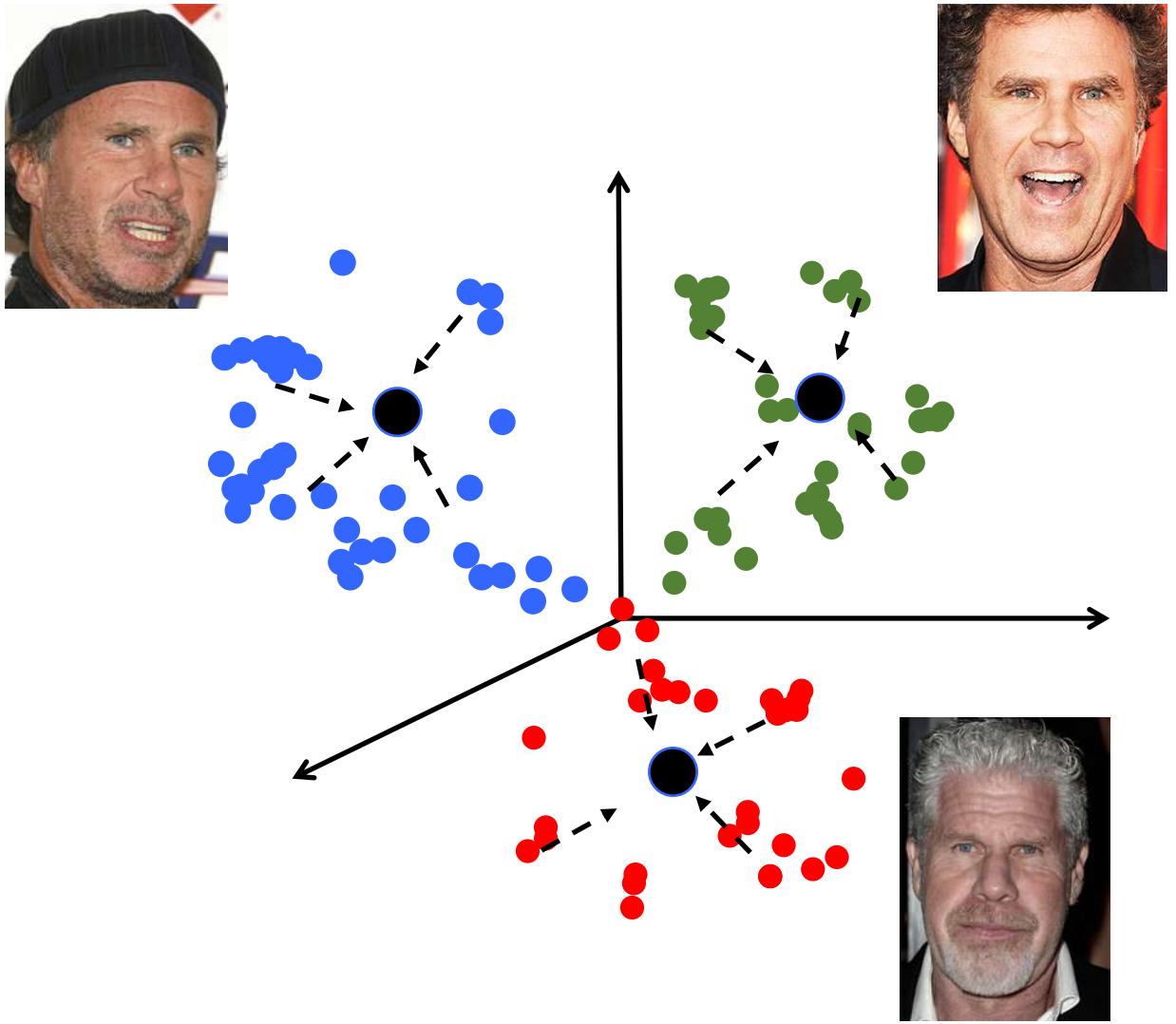
+ Supports large datasets
– Requires large batches
– Slow convergence



Softmax-based

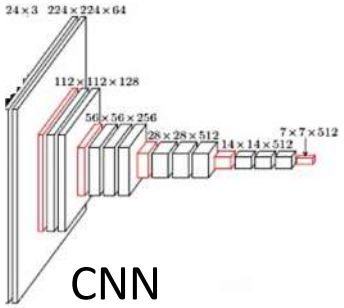
Center loss

Idea: pull points to class **centroids**



Center loss: structure

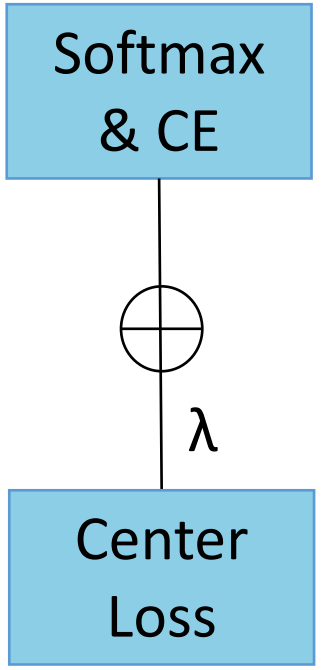
- Without classification loss – collapses
- Final loss = Cross Entropy + λ Center loss



Embedding

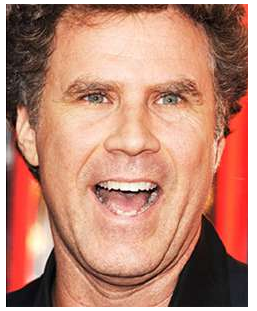
Classify

Pull

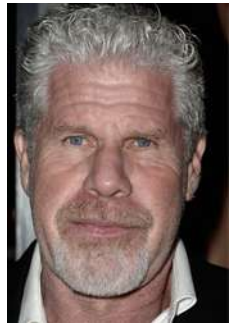


$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

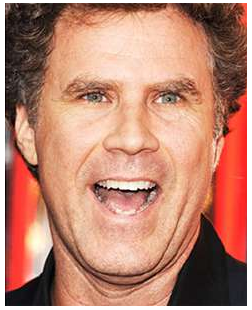
Center Loss: different lambdas



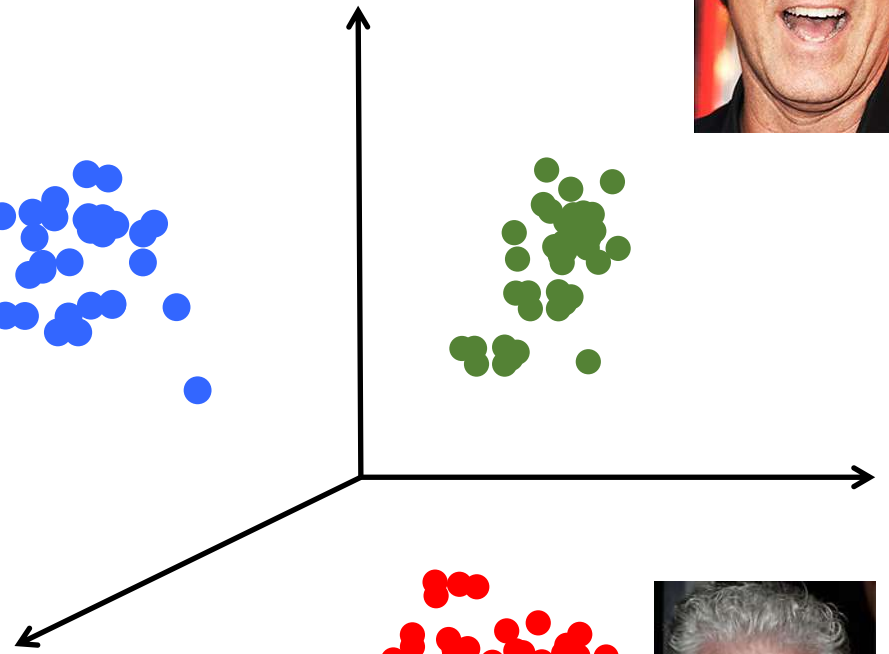
$\lambda = 10^{-7}$

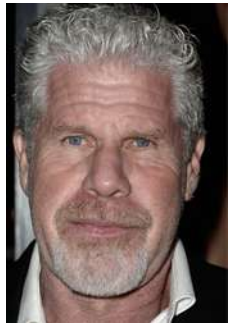


Center Loss: different lambdas

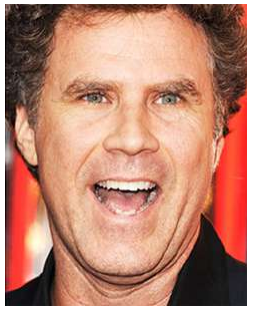


$\lambda = 10^{-6}$

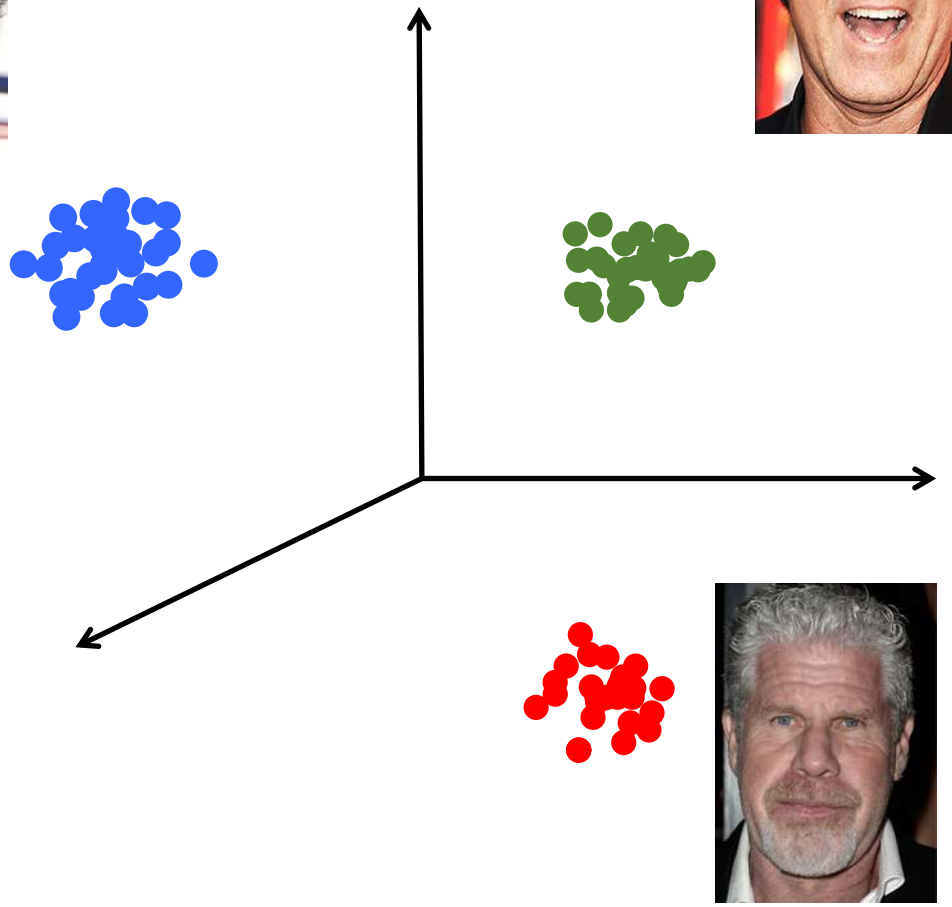




Center Loss: different lambdas



$\lambda = 10^{-5}$



Centres-loss gradients

Problem: we have to compute centers over whole dataset

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

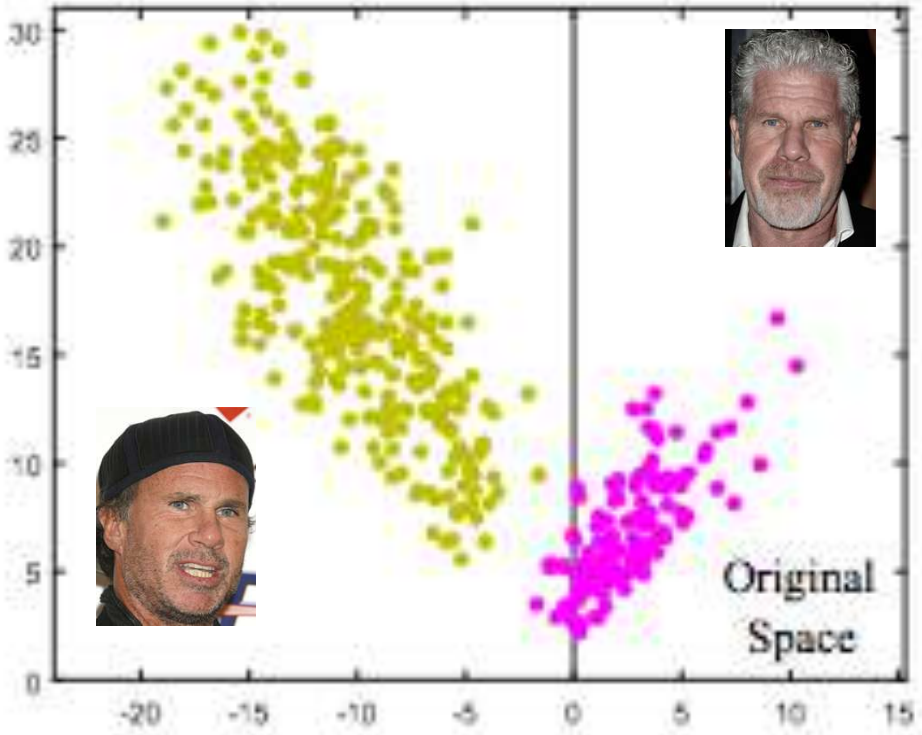
Solution:

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

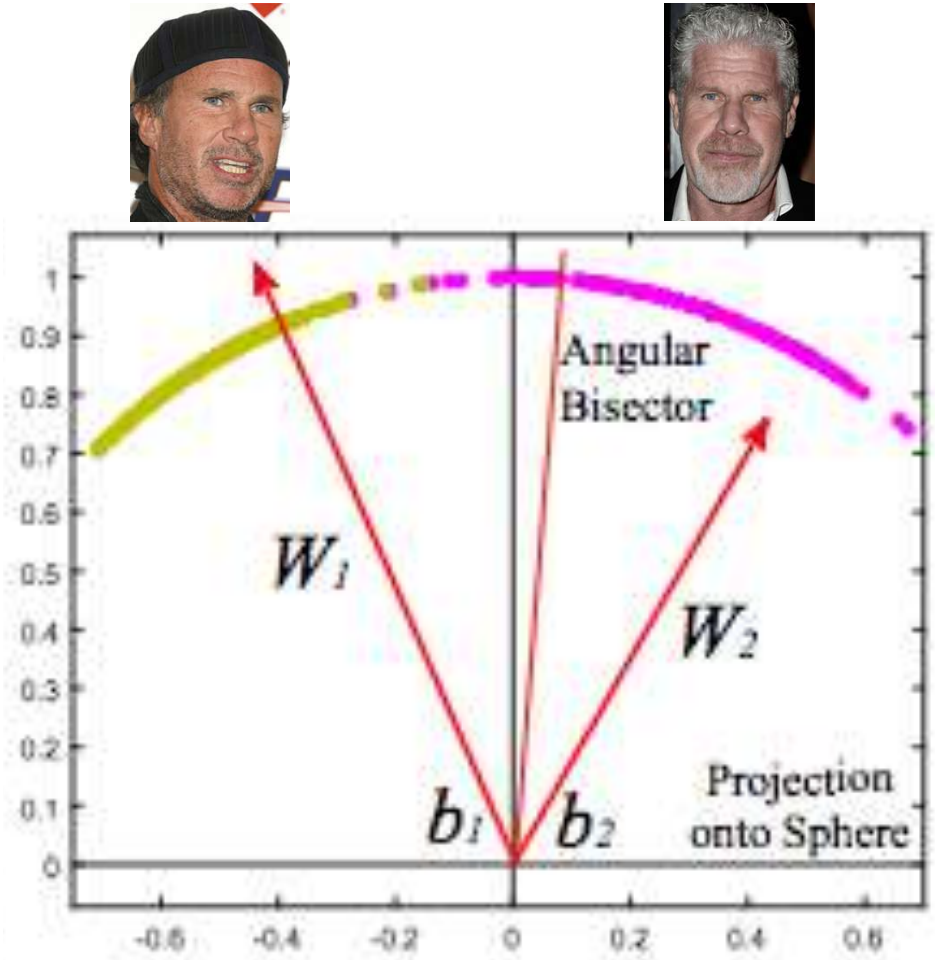
Angular Softmax



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

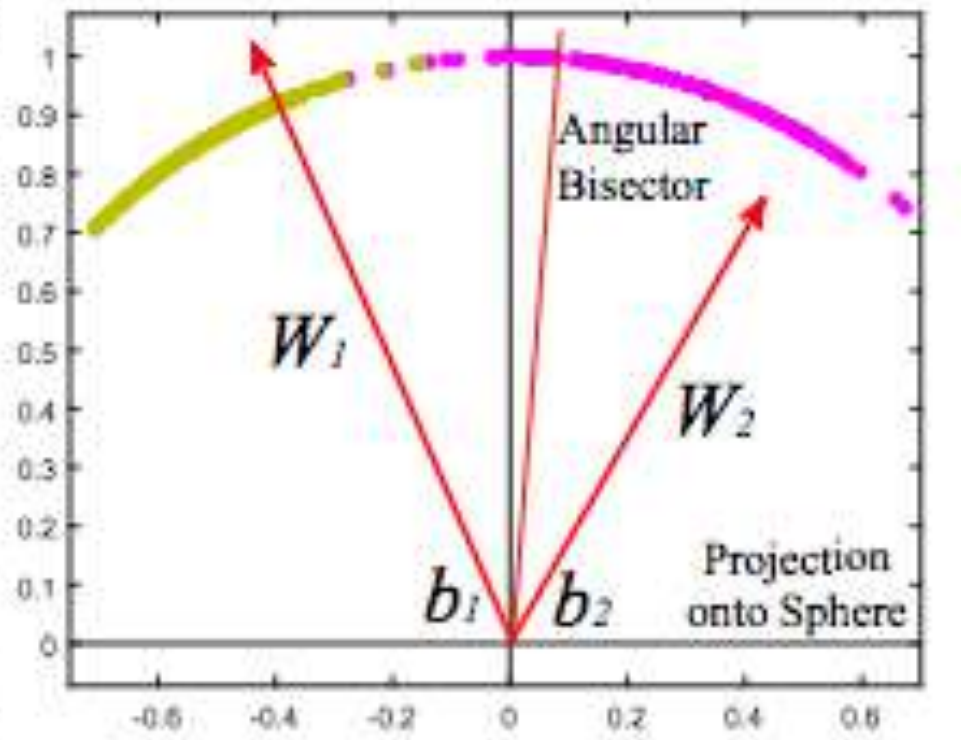
$\|X\| = 1$

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



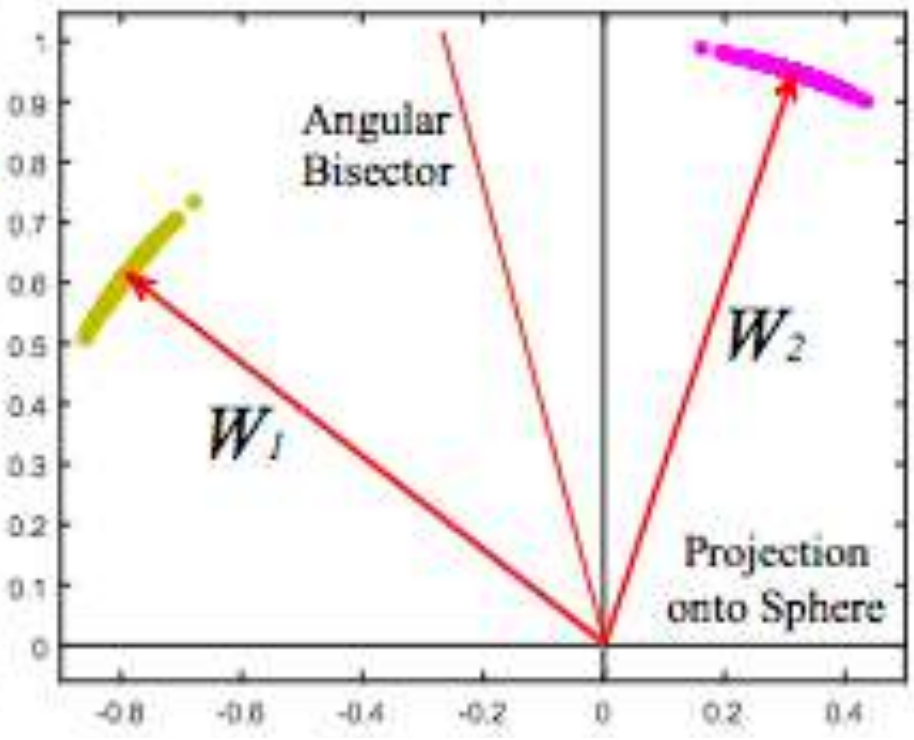
$$\|\boldsymbol{x}\|(\cos(\theta_1) - \cos(\theta_2)) = 0$$

Angular Softmax



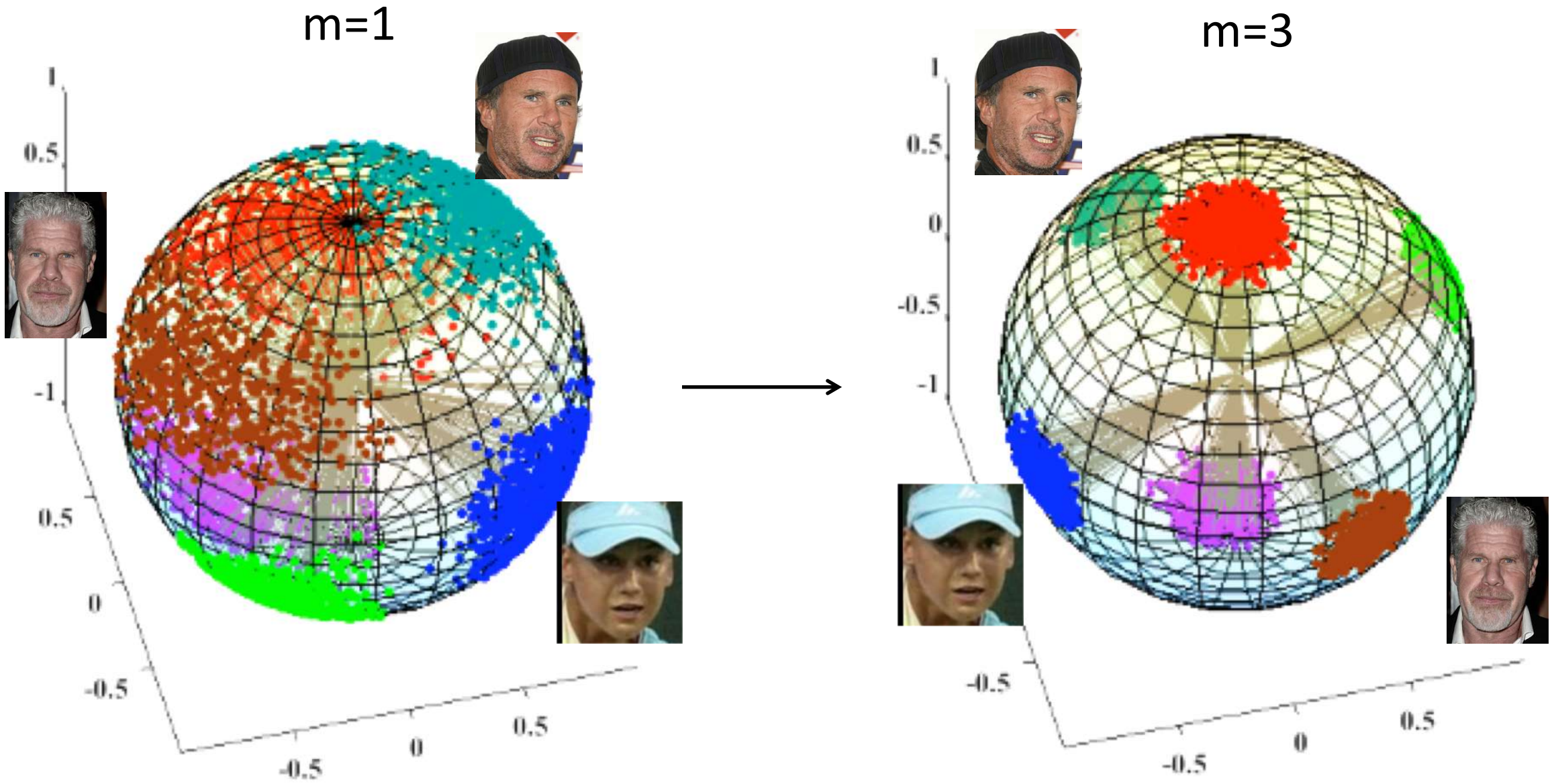
$$\|\boldsymbol{x}\|(\cos(\theta_1)-\cos(\theta_2))=0$$

To enforce larger angle

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

Angular Softmax: different «m»



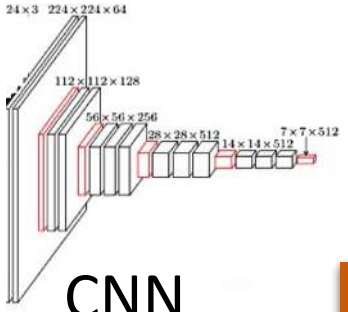
Angular Softmax family

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

AKNN

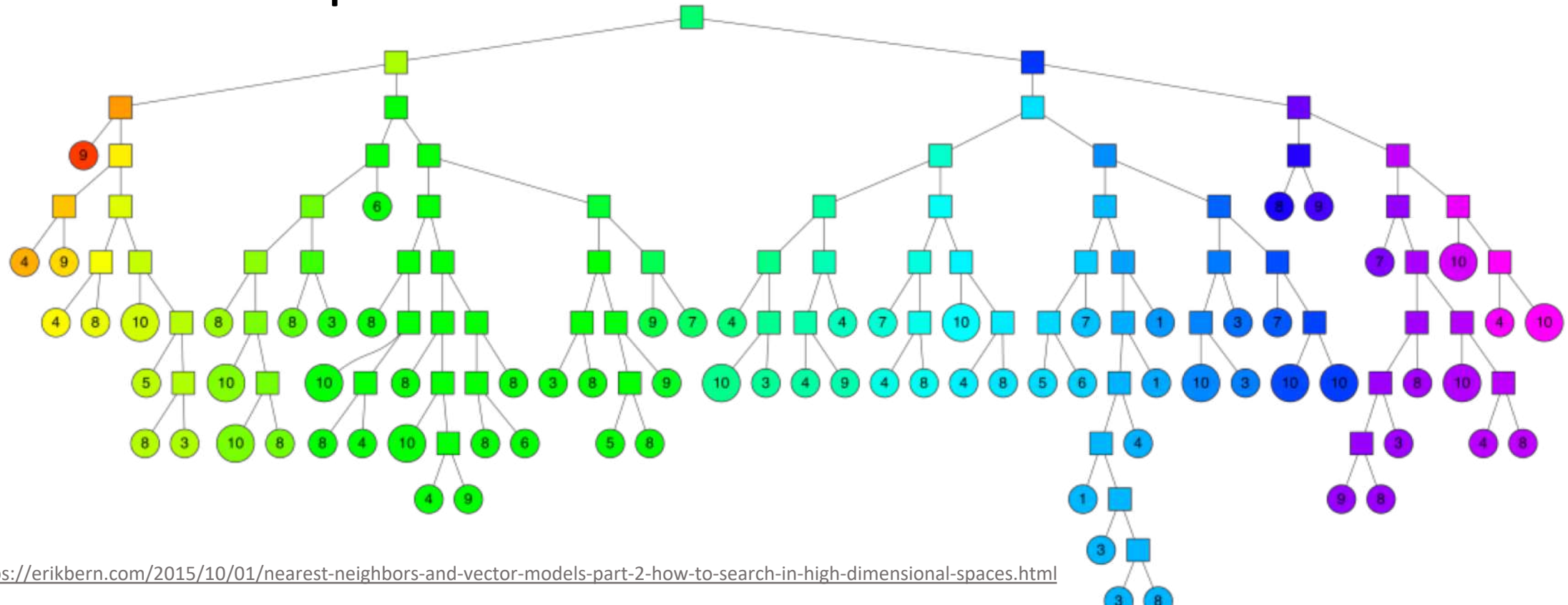
Problem formulation

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

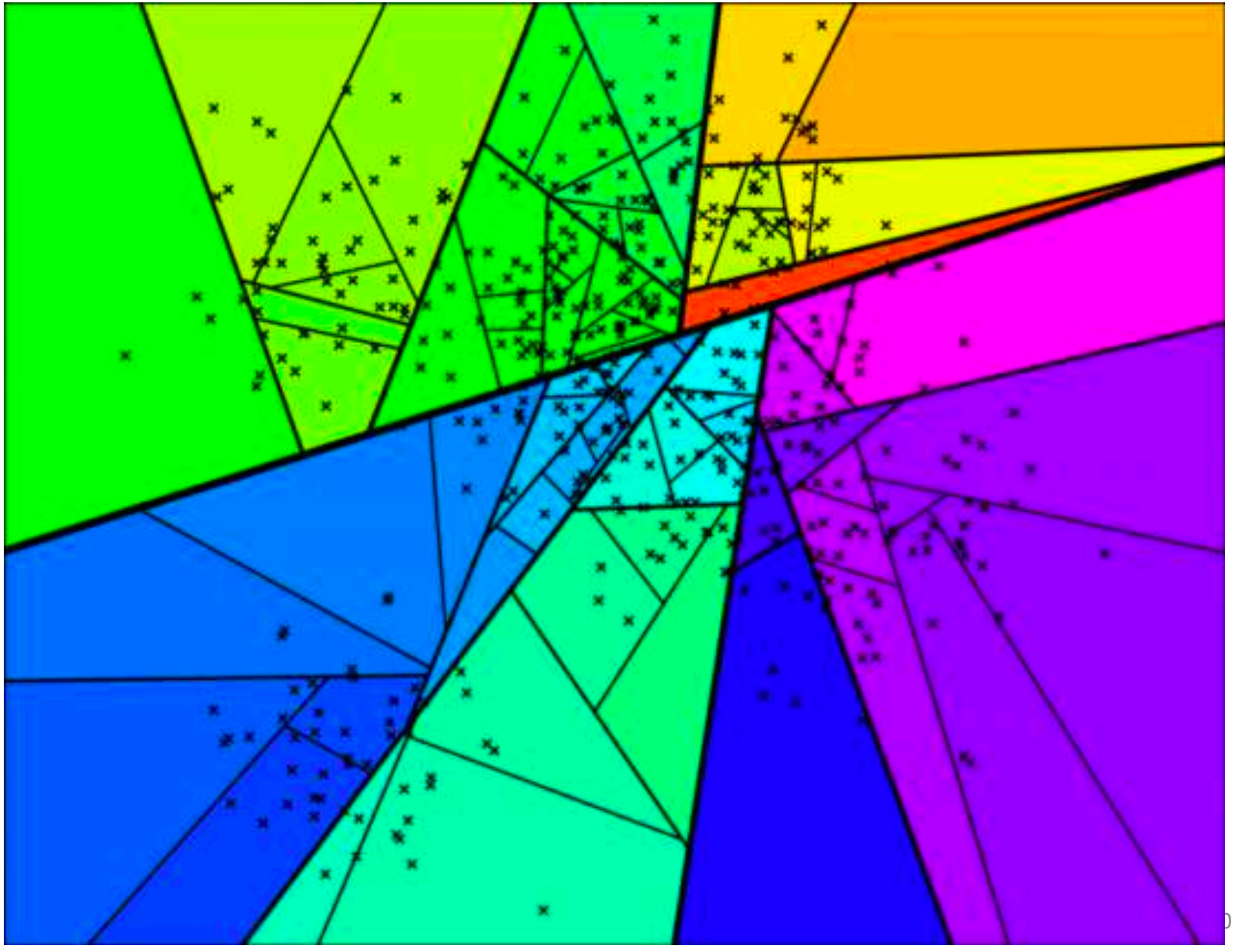


Source: https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

Annoy: pros & cons

mail.ru group

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



Hierarchical Navigable Small World

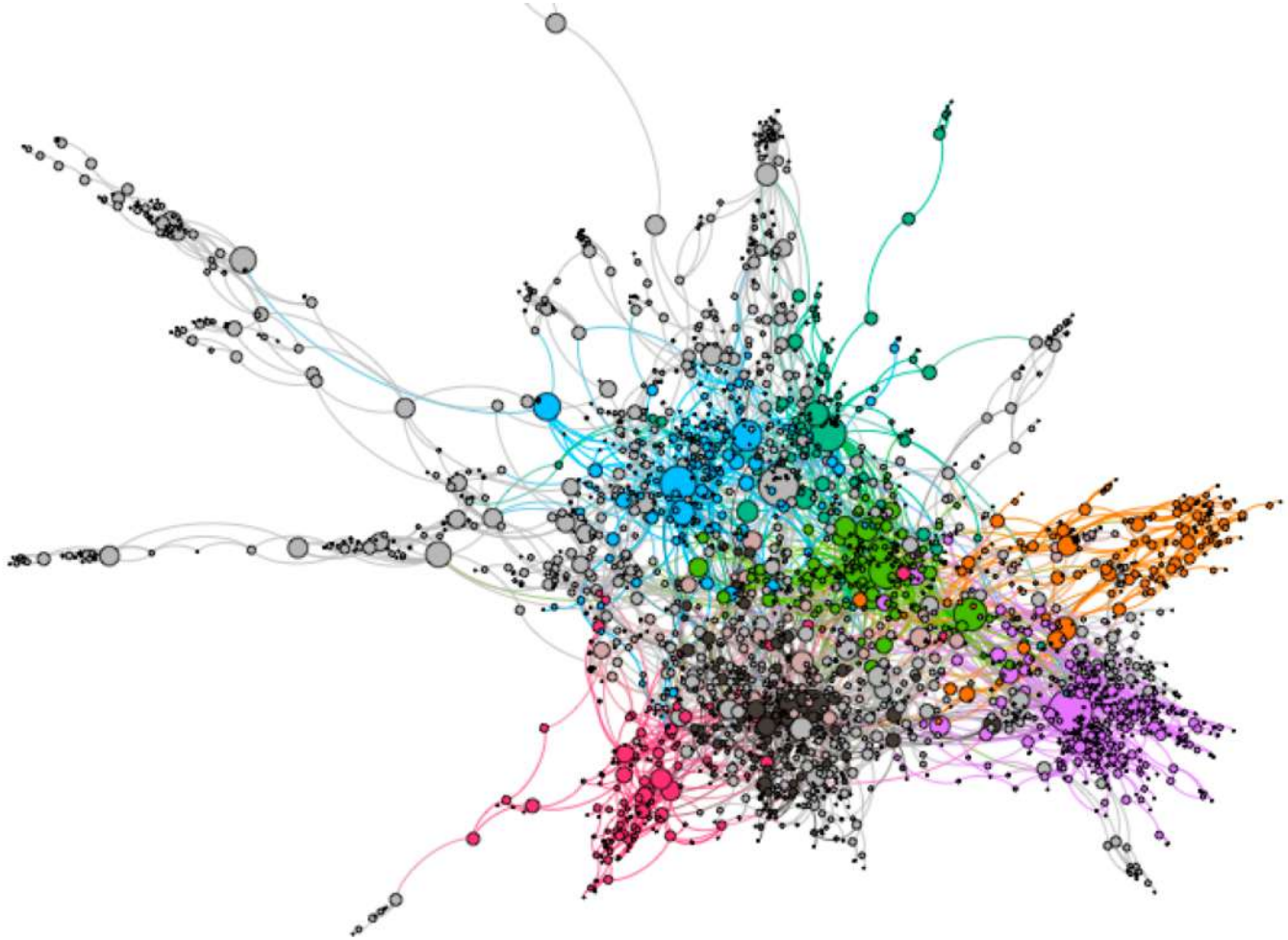
mail.ru group

Small World:

1. highly transitive
2. small average distance

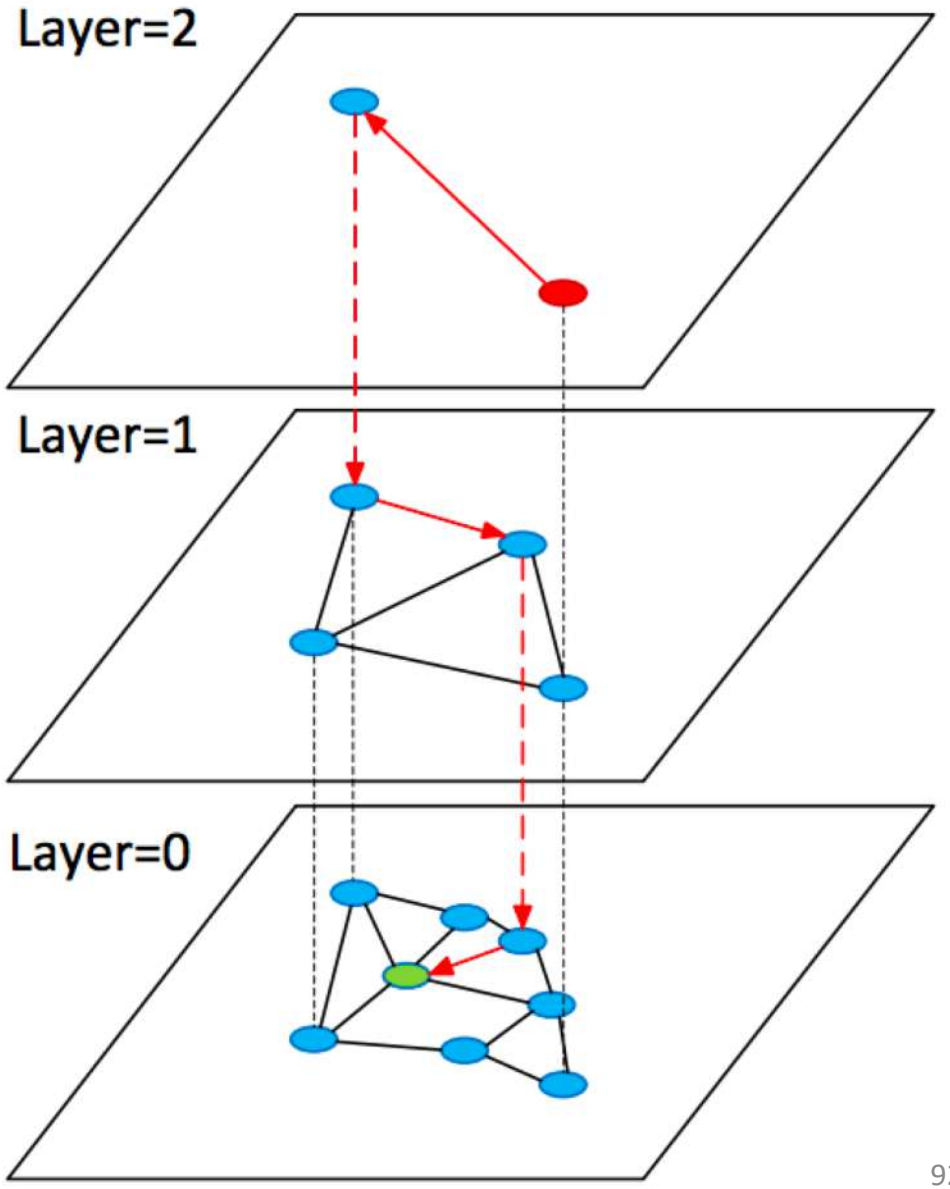
Traversal Algorithm

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



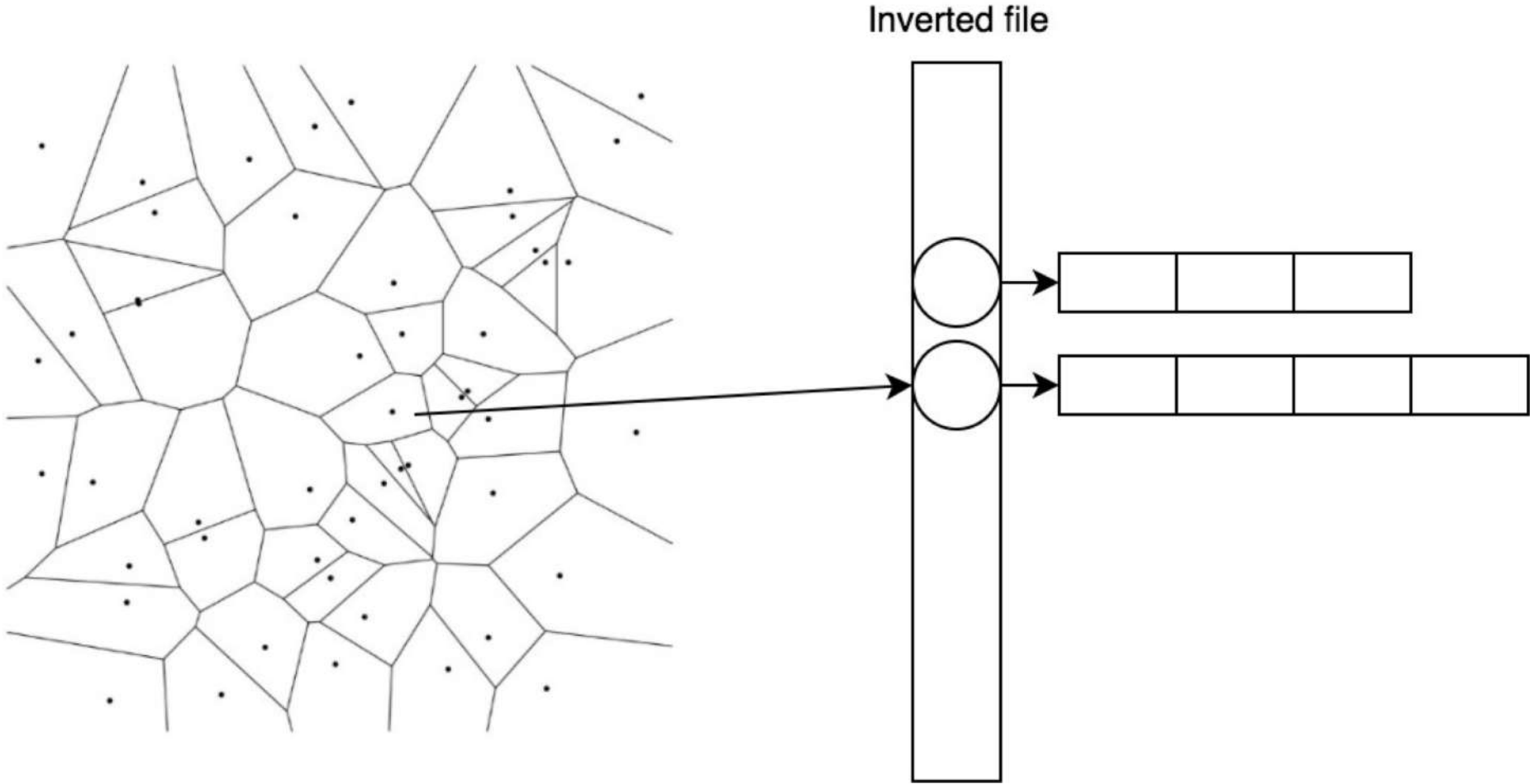
HNSW: pros & cons

mail.ru group

+ High accuracy
+ Fastest
+ Batching
– High memory consumptions

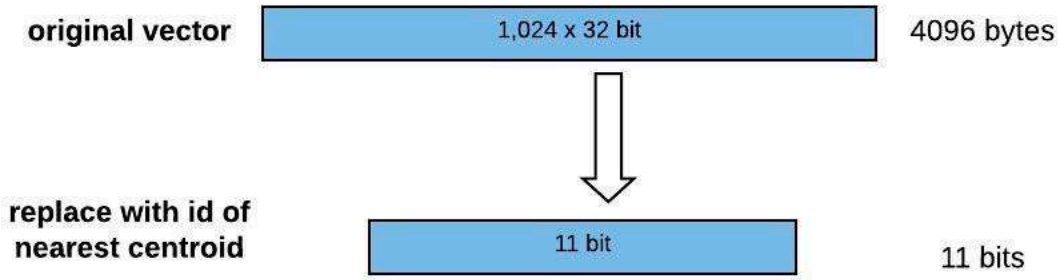
FAISS: inverted file

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



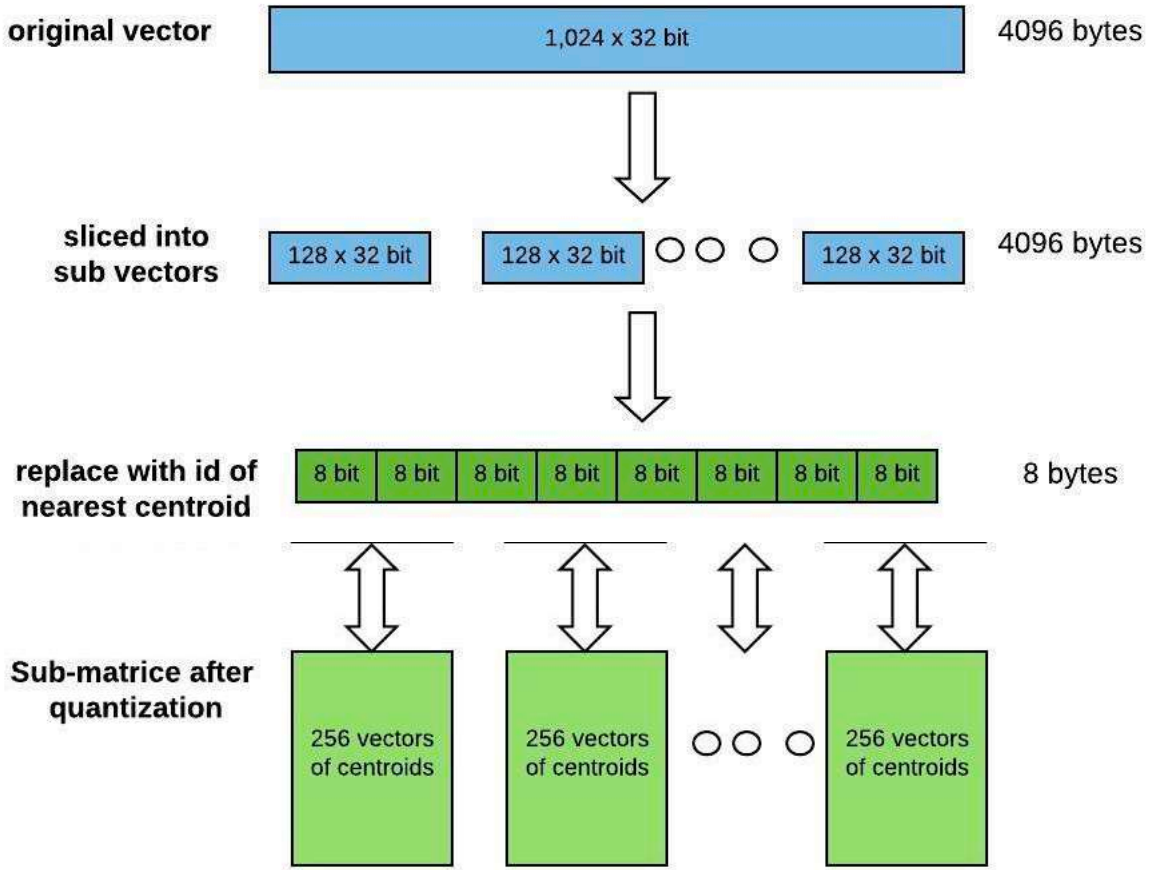
FAISS: Product Quantization

Goal: compress memory



2042 centroids can represent each vector with 11 bits

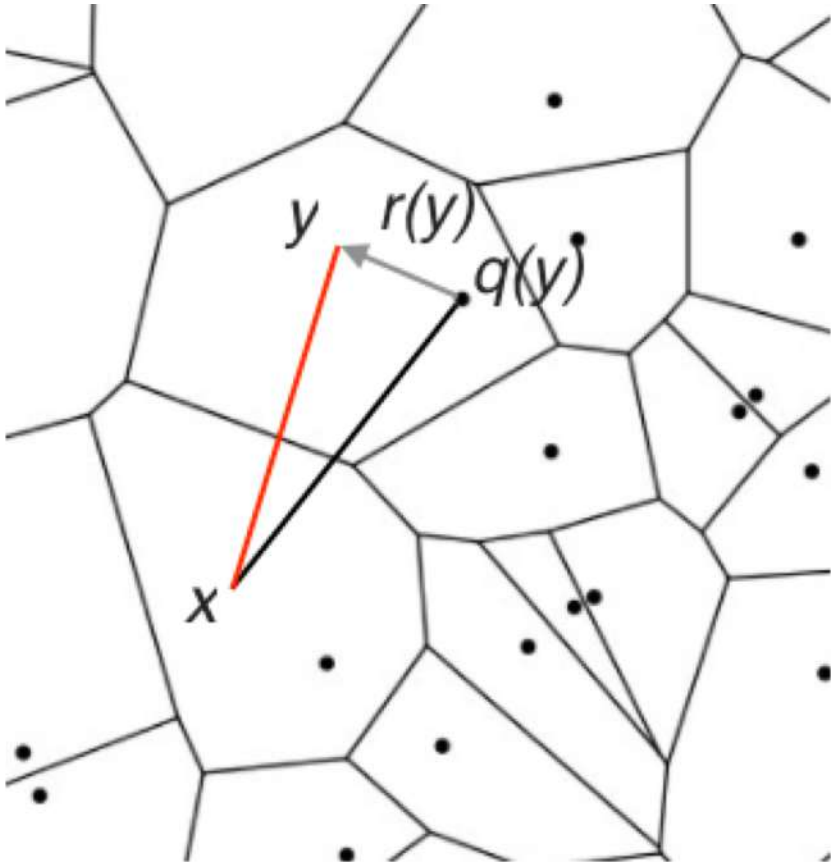
To increase accuracy slice vectors

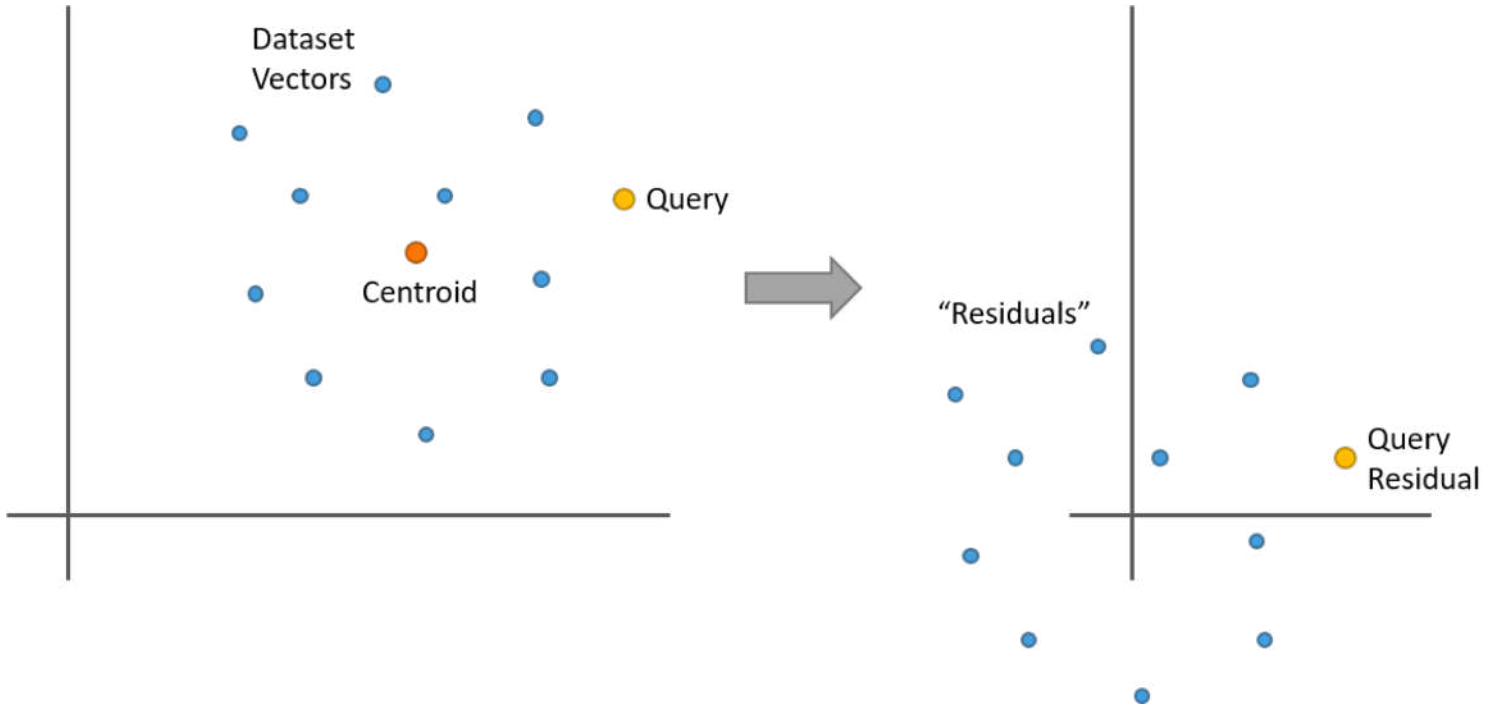


Source: https://towardsdatascience.com/comprehensive-guide-to-approximate-nearest-neighbors-algorithms-8b94f057d6b6

FAISS: residual

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

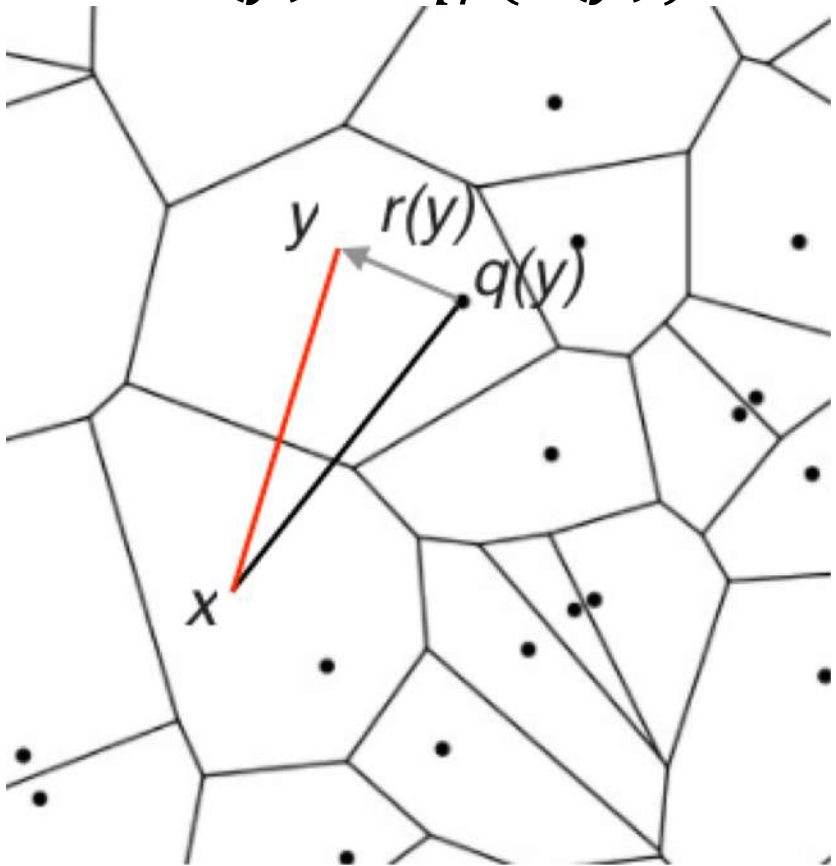


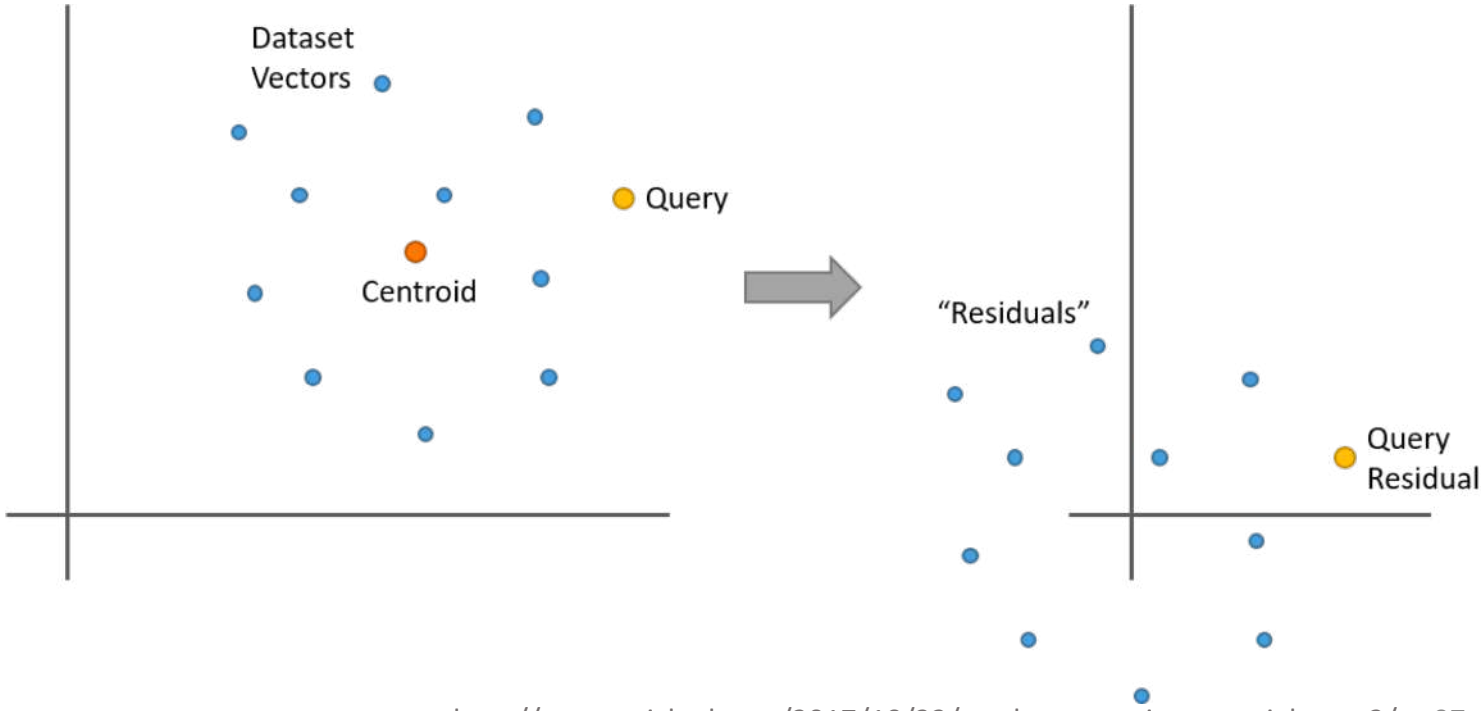


http://mccormickml.com/2017/10/22/product-quantizer-tutorial-part-2/

FAISS: residual

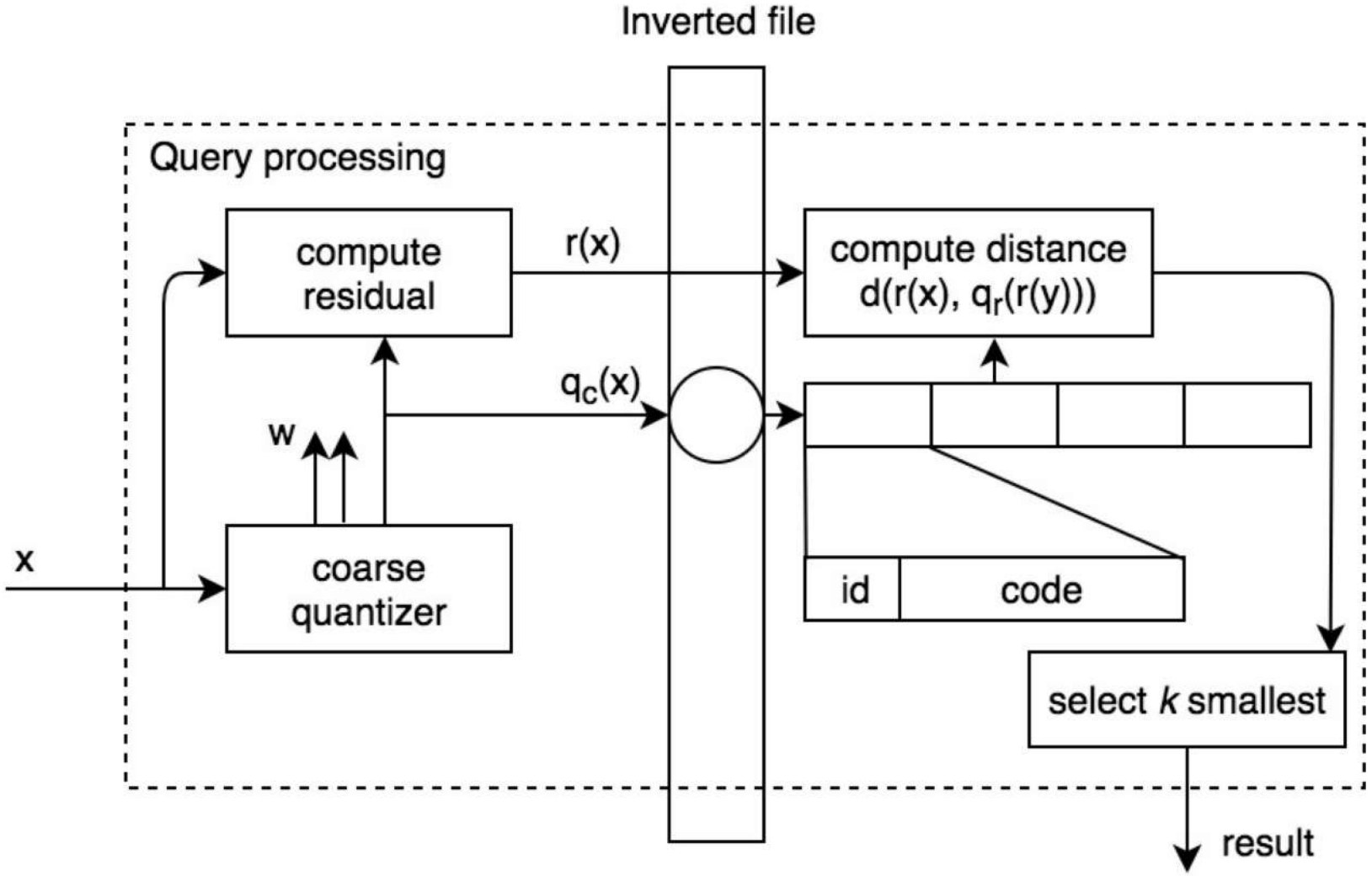
- points are centered around 0
  - By reducing the variety in the dataset, it takes fewer prototypes to represent the vectors effectively!
  - $r(y) \cong q_r(r(y))$





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)

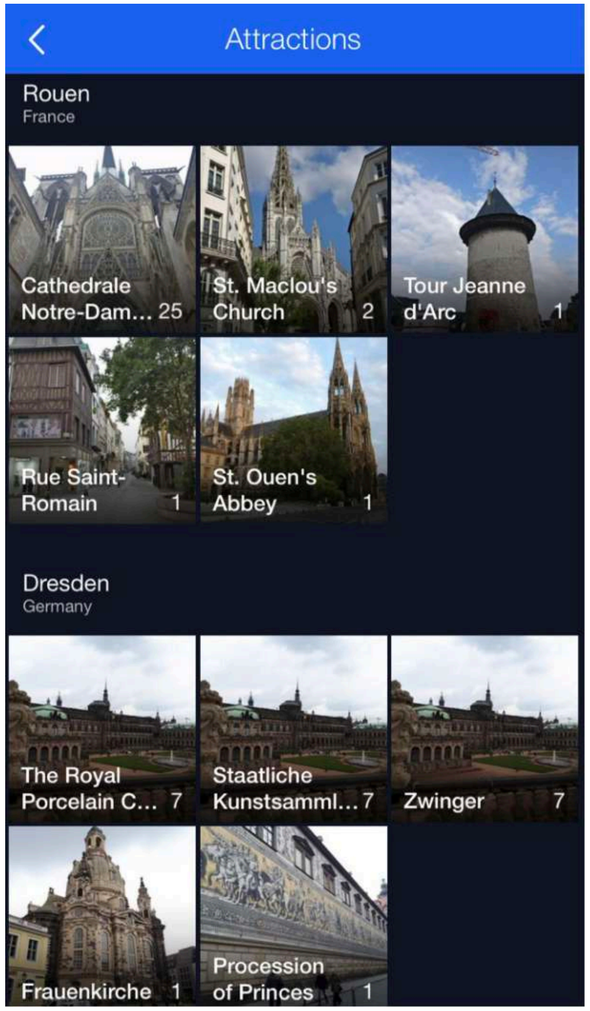
– We trade *compute* vs *memory + accuracy*

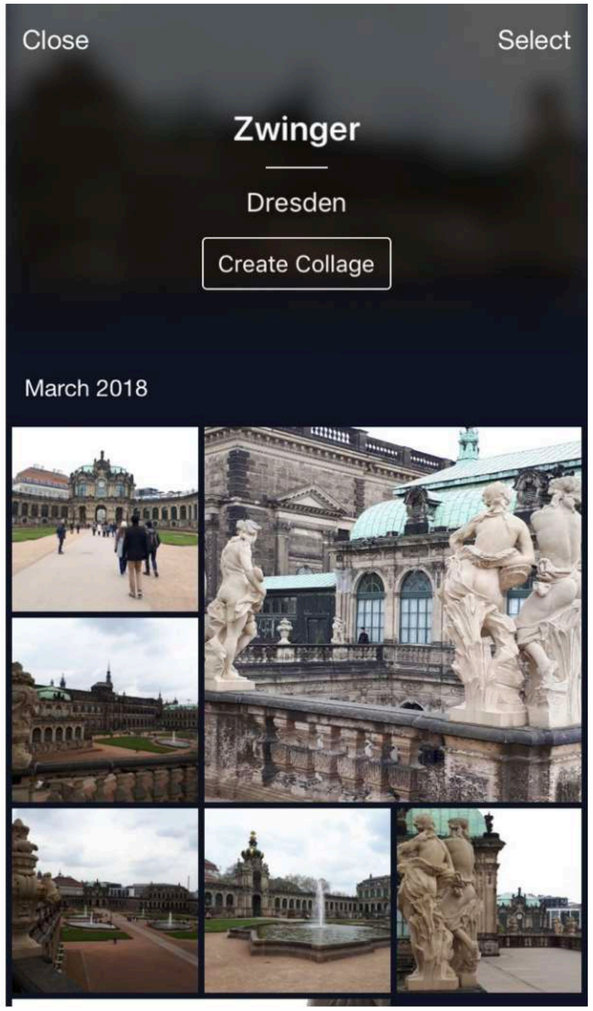
Method	search time	1-R@1	index size	index build time
Flat-CPU	9.100 s	1.0000	512 MB	0 s
nmslib (hnsw)	0.081 s	0.8195	512 + 796 MB	173 s
IVF16384,Flat	0.538 s	0.8980	512 + 8 MB	240 s
IVF16384,Flat (Titan X)	0.059 s	0.8145	512 + 8 MB	5 s

1M vectors

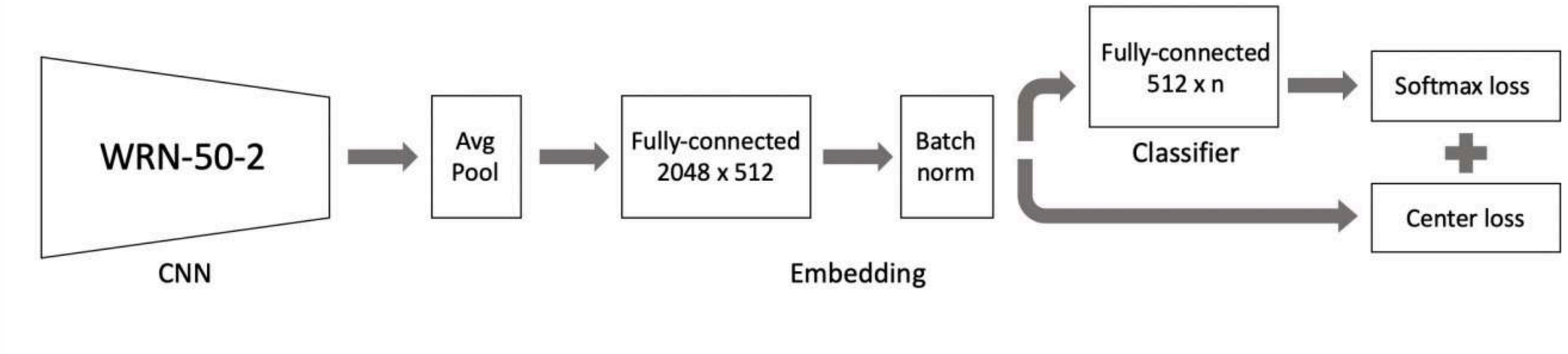
Landmarks recognition

Landmarks recognition task

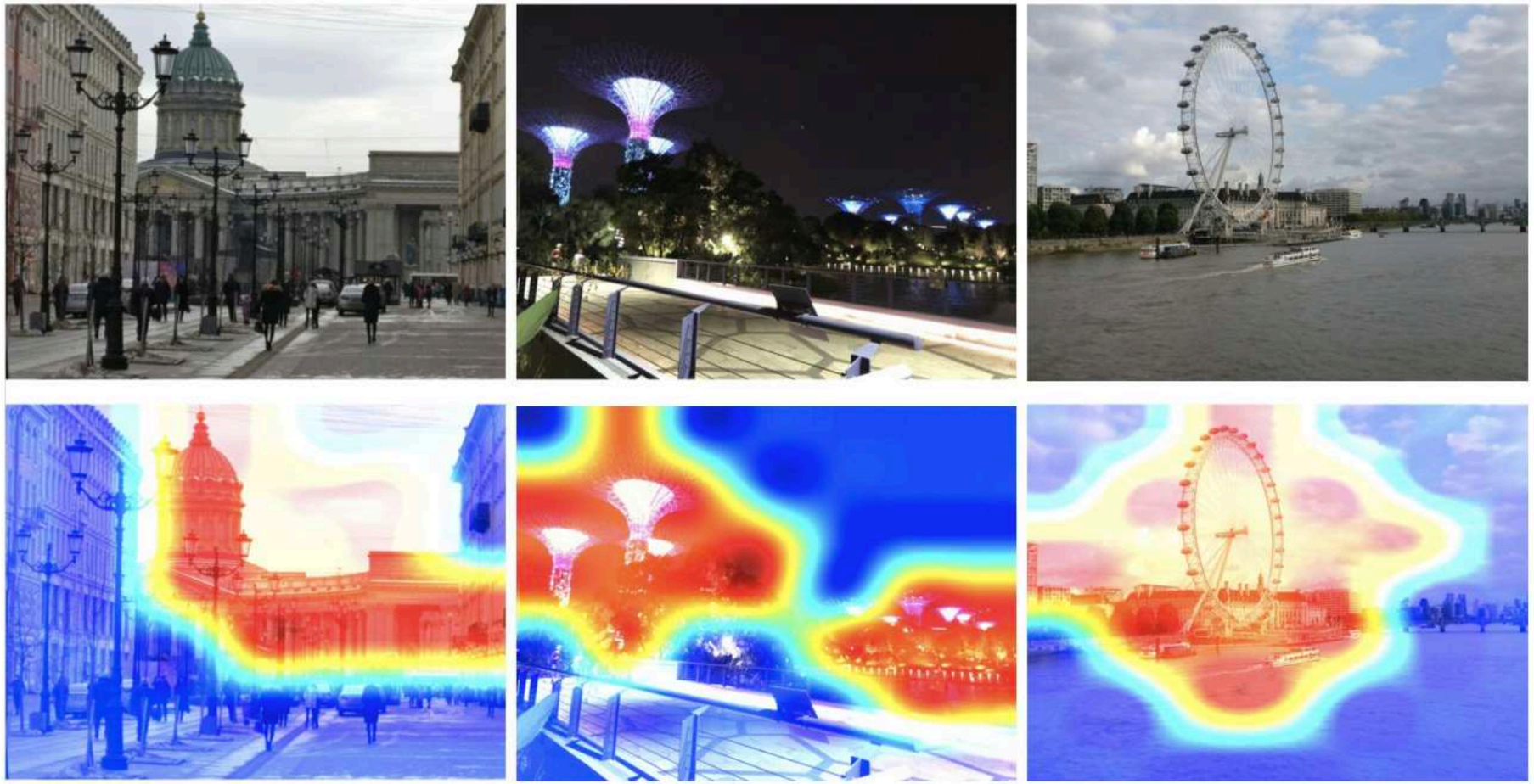




CNN architecture



Class activation map



Offline test results

Model	Sensitivity %	Specificity %
Softmax loss	17	99
Softmax + Center loss	31	99
Ours (without curriculum learning)	55	98
Ours (1 centroid per class)	62	98.8
Ours ($\eta = 250$)	**80**	**99**
DELF [17] (5 candidates)	77.8	99
DELF [17] (20 candidates)	**80.1**	**99**

Revisited Paris test

Method	$\mathcal{R}$Par		$\mathcal{R}$Par + $\mathcal{R}$1M	
	mAP	mP@10	mAP	mP@10
AlexNet-GeM [20]	29.7	67.6	8.4	39.6
VGG16-GeM [20]	44.3	83.7	19.1	64.9
ResNet101-GeM [20]	56.3	89.1	24.7	73.3
ResNet101-R-MAC [6]	59.4	86.1	28.0	70.0
HesAff (best) [25]	35.0	81.7	16.8	65.3
DELF [17]	55.4	93.4	26.4	75.7
DELF—GLD (best) [24]	58.6	91.0	29.4	83.9
Ours (standard setting)	50.2	70.9	34.9	59.7
Ours (our classes)	54.5	75.9	40.2	68.1
Ours (revised junk)	68.2	89.0	43.3	73.1
Ours (revised junk + our classes)	74.0	95.4	50.1	83.7

Timing & memory comparison with SOTA

- 15 times faster
- 133 times memory savings
- Large scale landmark recognition via deep metric learning, Boiarov et al., 2019 (https://arxiv.org/abs/1908.10192)

Recap

Metric learning is everywhere

- Recognition
- Few-shot learning
  - Prototypical networks
  - Matching networks
- Self-supervised learning
  - SimCLR
- Attention
  - Transformer (GPT, BERT, …)

Recap

- Triplet, Center loss, ArcFace
- Triplet can handle arbitrary large datasets
- AKNN: Faiss vs HNSW