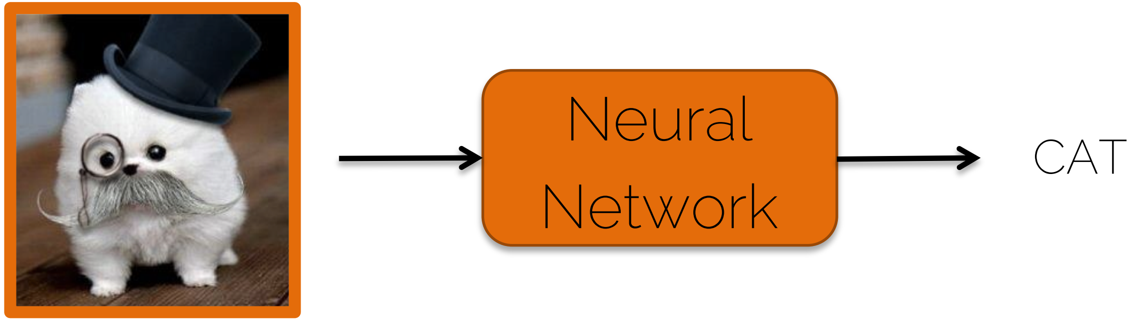
Siamese Neural Networks and Similarity Learning

What can ML do for us?

- Classification problem

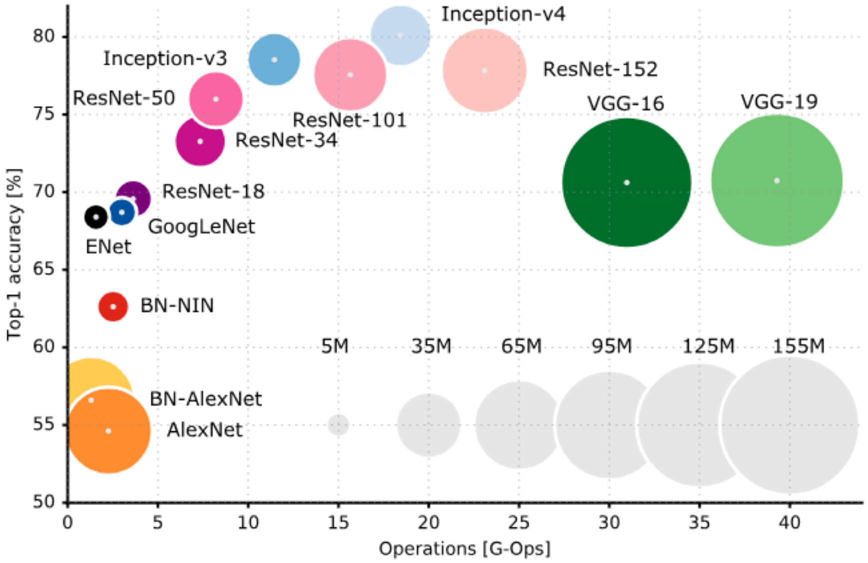


What can ML do for us?

- Classification problem on ImageNet with thousands of categories

What can ML do for us?

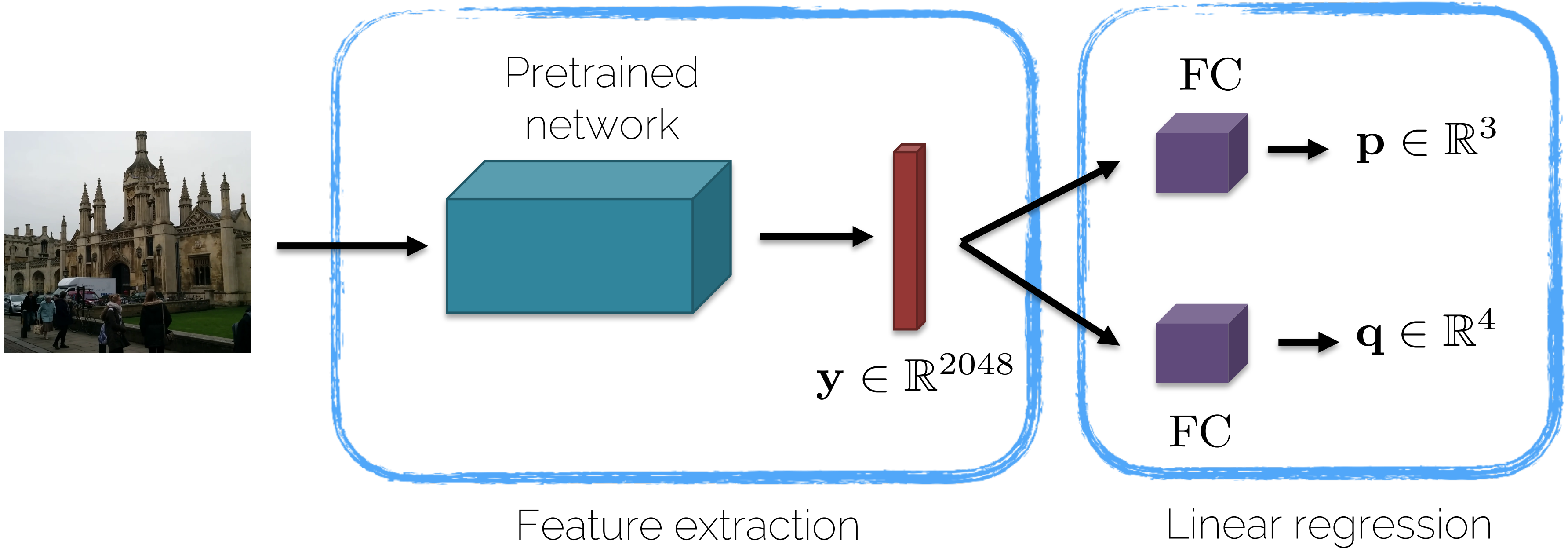
- Performance on ImageNet
  - Size of the blobs indicates the number of parameters



A. Canziani et al. „An Analysis of Deep Neural Network Models for Practical Applications". [arXiv:1605.07678](https://arxiv.org/abs/1605.07678) 2016

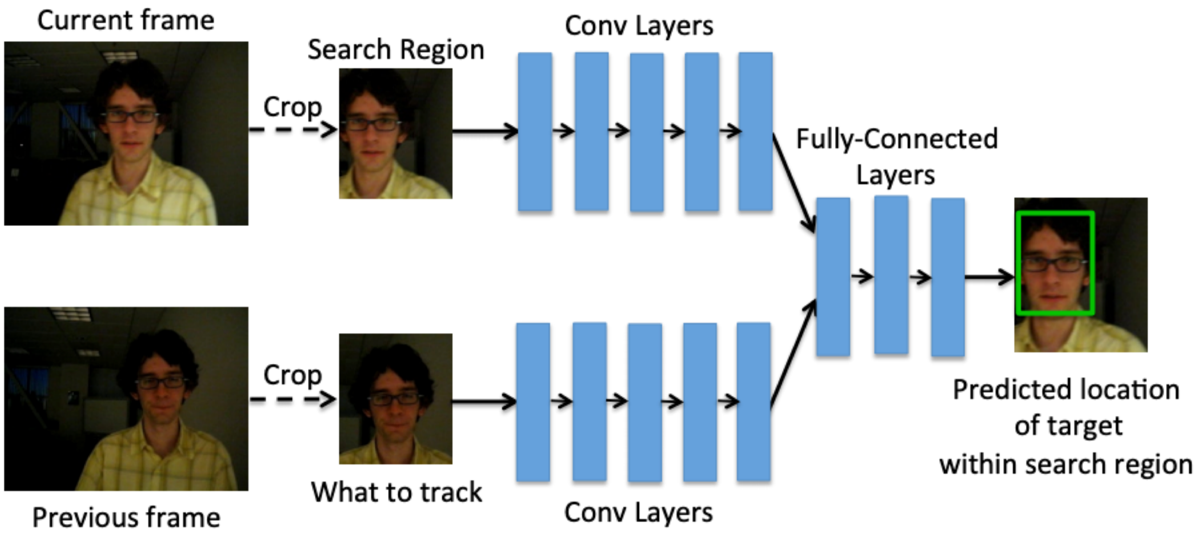
What can ML do for us?

- Regression problem: pose regression



What can ML do for us?

- Regression problem: bounding box regression



D. Held et al. „Learning to Track at 100 FPS with Deep Regression Networks". ECCV 2016

What can ML do for us?

- Third type of problems

A  Classification: person, face, female

B  Classification: person, face, male

What can ML do for us?

- Third type of problems

A

Is it the same person?

B

What can ML do for us?

- Third type of problems: Similarity Learning

A

B

– Comparison
– Ranking

Similarity Learning: when and why?

- Application: unlocking your iPhone with your face

Training

Similarity Learning: when and why?

- Application: unlocking your iPhone with your face

A  YES

B  NO

Testing 

Can be solved as a classification problem

Similarity Learning: when and why?

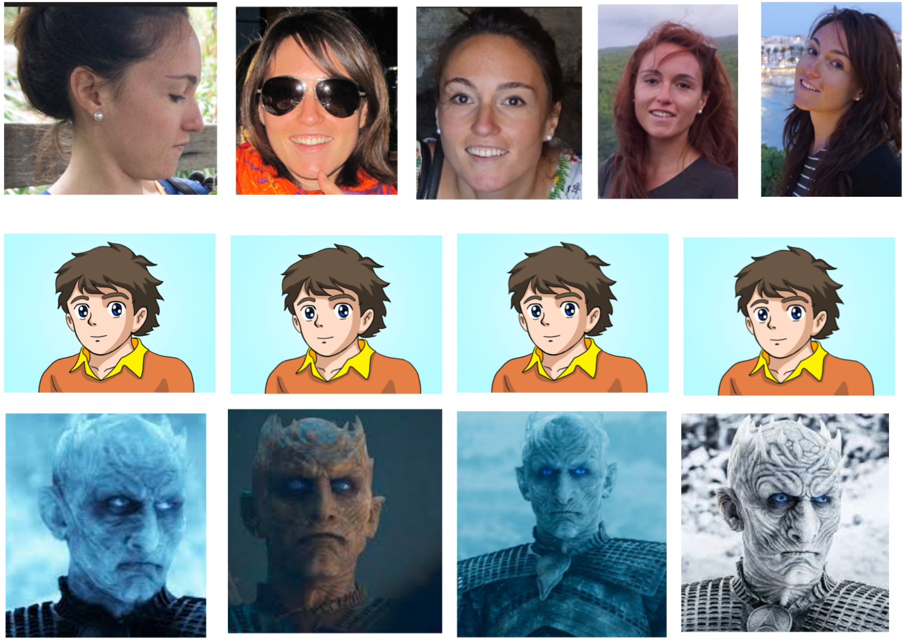
- Application: face recognition system so students can enter the exam room without the need for ID check

Training

Person 1

Person 2

Person 3



Similarity Learning: when and why?

- Application: face recognition system so students can enter the exam room without the need for ID check

What is the problem
with this approach?

Scalability – we need to retrain our model every time a new student is registered to the course

Similarity Learning: when and why?

- Application: face recognition system so students can enter the exam room without the need for ID check

Can we train one model and use it every year?

Similarity Learning: when and why?

- Learn a similarity function

A

B

Low similarity score

A

B

High similarity score

Similarity Learning: when and why?

- Learn a similarity function: testing

A

$$d(A, B) > \tau$$

Not the same person

B

Similarity Learning: when and why?

- Learn a similarity function

A

Same person

$$d(A, B) < \tau$$

B

Similarity learning

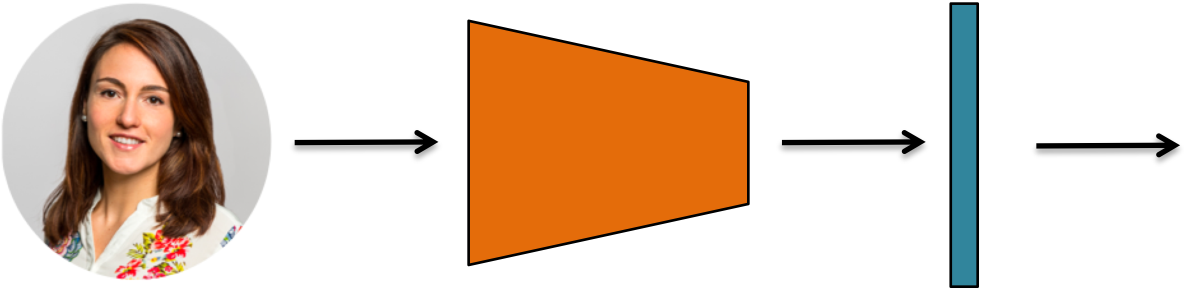
- How do we train a network to learn similarity?

Siamese Neural Networks

Similarity learning

- How do we train a network to learn similarity?

A



Representation of my face in 128 values

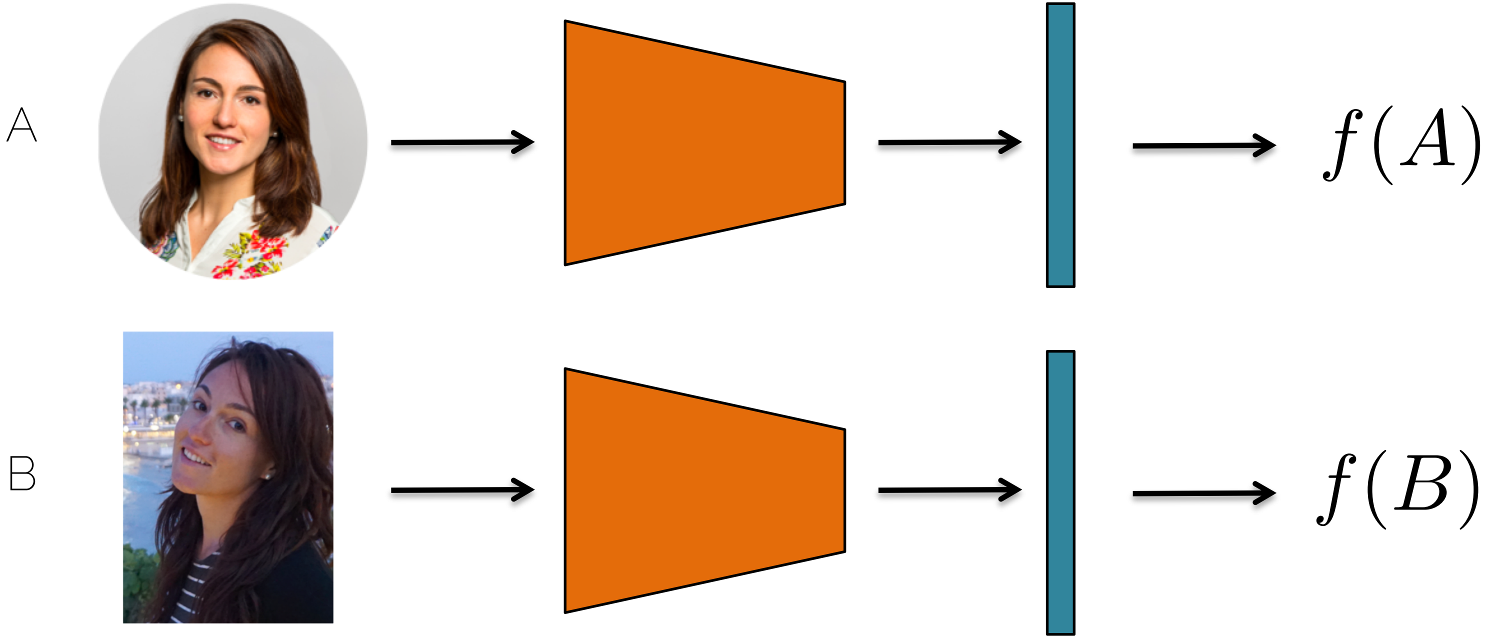
CNN

FC

Taigman et al. „DeepFace: closing the gap to human level performance". CVPR 2014

Similarity learning

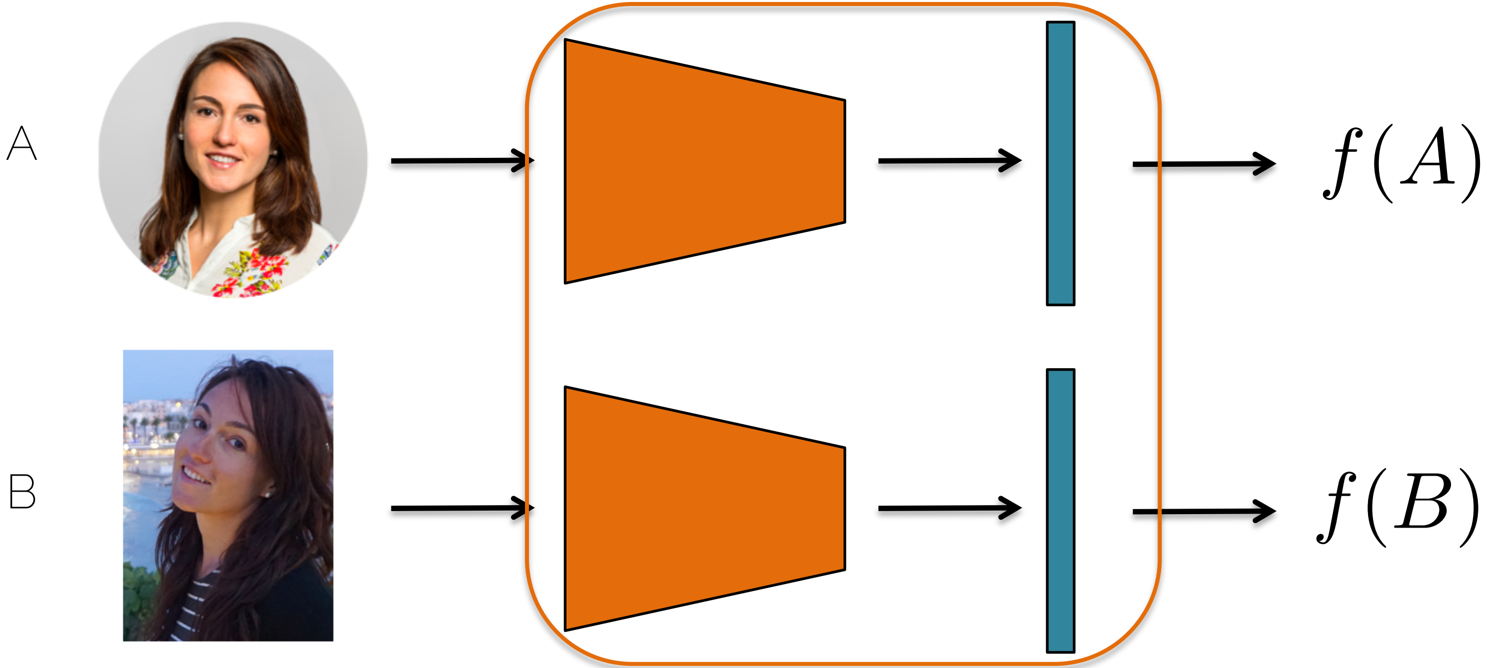
- How do we train a network to learn similarity?



Taigman et al. „DeepFace: closing the gap to human level performance“. CVPR 2014

Similarity learning

- Siamese network = shared weights



Taigman et al. „DeepFace: closing the gap to human level performance". CVPR 2014

Similarity learning

- Siamese network = shared weights
- We use the same network to obtain an encoding of the image $f(A)$
- To be done: compare the encodings

Taigman et al. „DeepFace: closing the gap to human level performance“. CVPR 2014

Similarity learning

- Distance function $d(A, B) = ||f(A) - f(B)||^2$
- Training: learn the parameter such that
  - If A and B depict the same person, $d(A, B)$ is small
  - If A and B depict a different person, $d(A, B)$ is large

Taigman et al. „DeepFace: closing the gap to human level performance". CVPR 2014

Similarity learning

- Loss function for a positive pair:
  - If A and B depict the same person, $d(A, B)$ is small

$$\mathcal{L}(A, B) = ||f(A) - f(B)||^2$$

Similarity learning

- Loss function for a negative pair:
  - If A and B depict a different person, $d(A, B)$ is large
  - Better use a Hinge loss:

$$\mathcal{L}(A, B) = \max(0, m^2 - ||f(A) - f(B)||^2)$$

If two elements are already far away, do not spend energy in pulling them even further away

Similarity learning

- Contrastive loss:

$$\mathcal{L}(A,B) = y^*||f(A) - f(B)||^2 + (1 - y^*)max(0, m^2 - ||f(A) - f(B)||^2)$$

Positive pair, reduce the distance between the elements

Negative pair, brings the elements further apart up to a margin

Similarity learning

- Training the siamese networks
  - You can update the weights for each channel independently and then average them

- This loss function allows us to learn to bring positive pairs together and negative pairs apart

Triplet Loss

Triplet loss

- Triplet loss allows us to learn a ranking

Anchor (A)

Positive (P)

Negative (N)

We want: $$||f(A) - f(P)||^2 < ||f(A) - f(N)||^2$$

Schroff et al „FaceNet: a unified embedding for face recognition and clustering". CVPR 2015

Triplet loss

- Triplet loss allows us to learn a ranking

$$||f(A) - f(P)||^2 < ||f(A) - f(N)||^2$$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 < 0$$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + m < 0$$

margin

Schroff et al „FaceNet: a unified embedding for face recognition and clustering". CVPR 2015

Triplet loss

- Triplet loss allows us to learn a ranking

$$||f(A) - f(P)||^2 < ||f(A) - f(N)||^2$$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 < 0$$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + m < 0$$

$$\mathcal{L}(A, P, N) = max(0, ||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + m)$$

Schroff et al „FaceNet: a unified embedding for face recognition and clustering". CVPR 2015

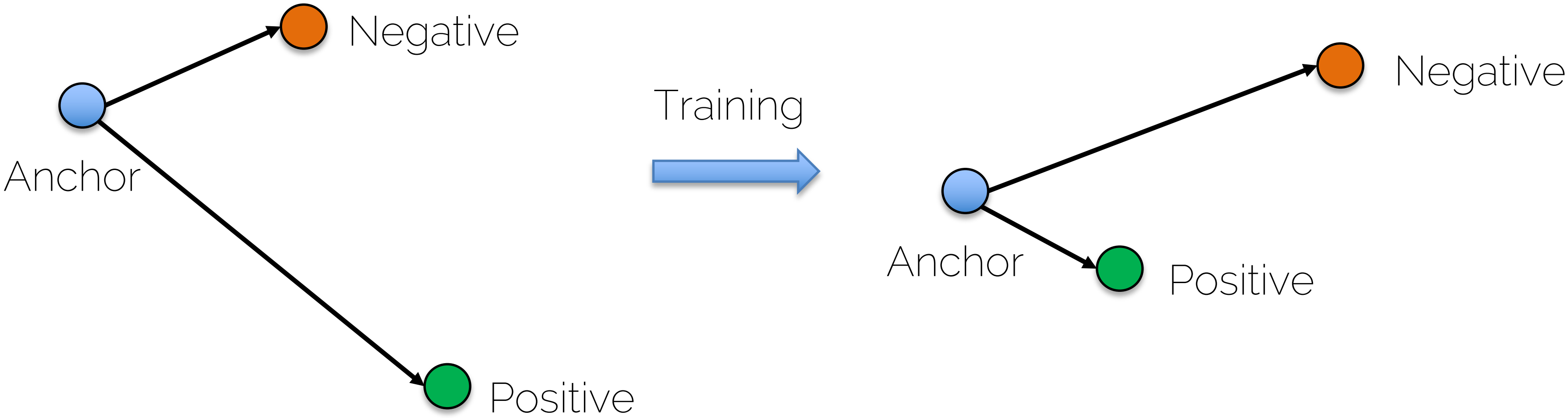
Triplet loss

- Hard negative mining: training with hard cases

$$\mathcal{L}(A, P, N) = max(0, ||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + m)$$

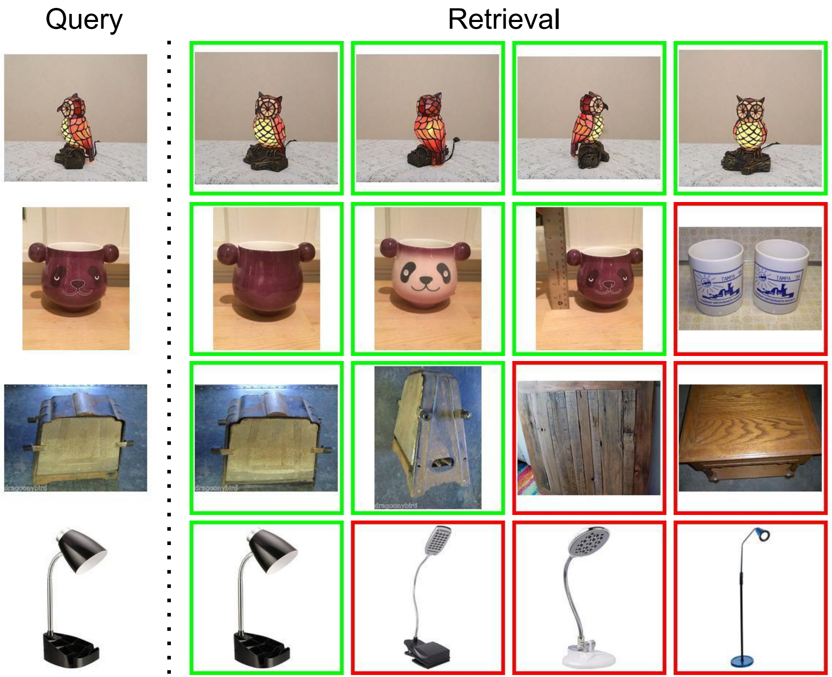
- Train for a few epochs
- Choose the hard cases where $d(A, P) \approx d(A, N)$
- Train with those to refine the distance learned

Triplet loss



Triplet loss: test time

- Just do nearest neighbor search!



Triplet Loss Challenges

- Random sampling does not work - the number of possible triplets is $O(n^3)$ so the network would need to be trained for a very long time.

- Even with hard negative mining, there is the risk of being stuck in local minima.

Several approaches to improve similarity learning

Improving similarity learning

- Loss:
  - Contrastive vs. triplet loss
- Sampling:
  - Choosing the best triplets to train with, sample the space wisely = diversity of classes + hard cases
- Ensembles:
  - Why not using several networks, each of them trained with a subset of triplets?
- Can we use a classification loss for similarity learning?

Losses: interesting works

- Wang et al., Deep metric learning with angular loss, (ICCV 2017)

- Yu et al., Correcting the triplet selection bias for triplet loss, (ECCV 2018)

Improving similarity learning

- Loss:
  - Contrastive vs. triplet loss

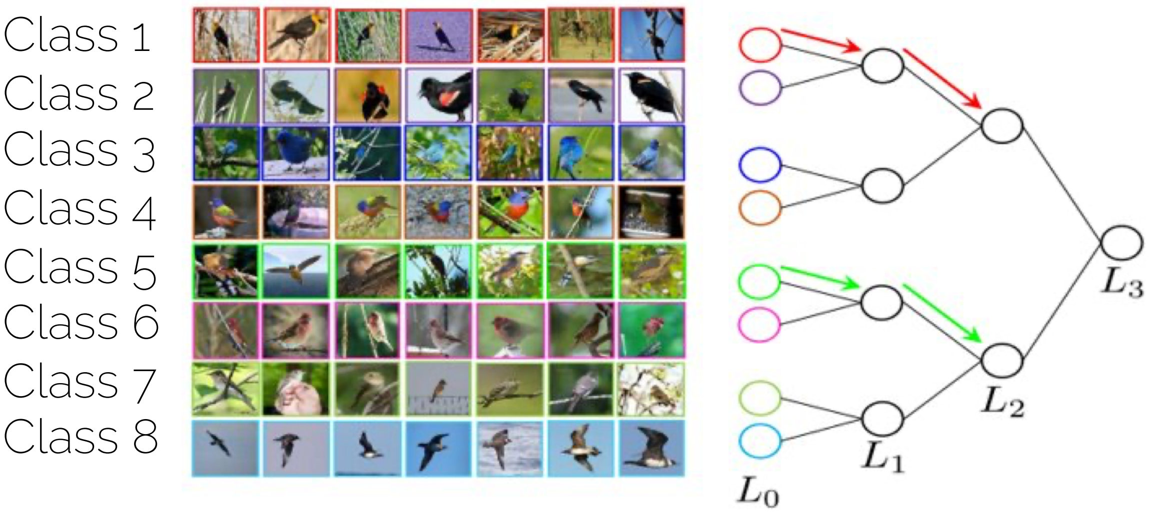
- Sampling:
  - Choosing the best triplets to train with, sample the space wisely = diversity of classes + hard cases

- Ensembles:
  - Why not using several networks, each of them trained with a subset of triplets?

- Can we use a classification loss for similarity learning?

Sampling: Hierarchical Triplet Loss

- Build a hierarchical tree where the leaves of the tree represent the image classes. Recursively merge them until you reach the root node



Ge et al., Deep Metric Learning with Hierarchical Triplet Loss, ECCV 2018

HTL: building the tree

- In order to create the tree, we first define a distance between classes. Intuition: if the distance is small, they will be merged in the next level of the tree.

$$d(p,q) = \frac{1}{n_p n_q} \sum_{i \in p, j \in q} \|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2$$

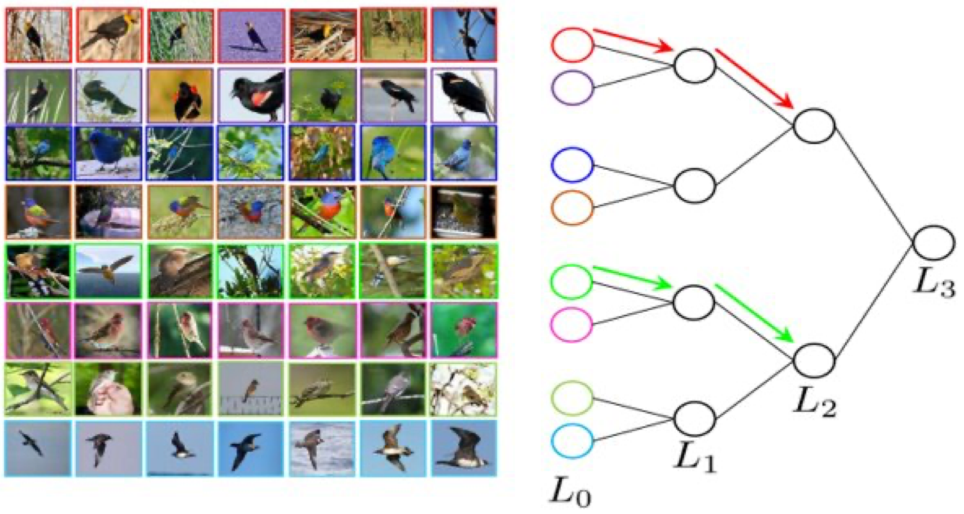
The cardinality of classes p and q (how many samples do we have for each class)

Deep features of images i and j

HTL: Finding the anchors

- Randomly select l' nodes at the 0th level
  - This is done to preserve class diversity in the mini-batch

Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8



HTL: Finding the anchors

- Randomly select l' nodes at the 0^{th} level
  - This is done to preserve class diversity in the mini-batch
- m-1 nearest classes at the 0^{th} level are selected for each of the l' nodes based on the distance in feature space.

HTL: Finding the anchors

- Randomly select l' nodes at the 0^{th} level
  - This is done to preserve class diversity in the mini-batch
- m-1 nearest classes at the 0^{th} level are selected for each of the l' nodes based on the distance in feature space:
  - We want to encourage the model to learn discriminative features from the visual similar classes.

HTL: Finding the anchors

- Randomly select l' nodes at the 0th level
  - This is done to preserve class diversity in the mini-batch
- m-1 nearest classes at the 0th level are selected for each of the l' nodes based on the distance in feature space:
  - We want to encourage the model to learn discriminative features from the visual similar classes.
- t images per class are randomly collected

t*m*l' images in the mini-batch

HTL: Loss formulation

$$\mathcal{L}_{\mathcal{M}} = \frac{1}{2Z_{\mathcal{M}}} \sum_{\mathcal{T}^z \in \mathcal{T}^{\mathcal{M}}} \left[\| \boldsymbol{x}_a^z - \boldsymbol{x}_p^z \| - \| \boldsymbol{x}_a^z - \boldsymbol{x}_n^z \| + \alpha_z \right]_+$$

all the triplets

The margin actually depends on the distances computed on the hierachical tree. The idea is that it can adapt to class distributions and differences of the samples within the classes.

Sampling: interesting works

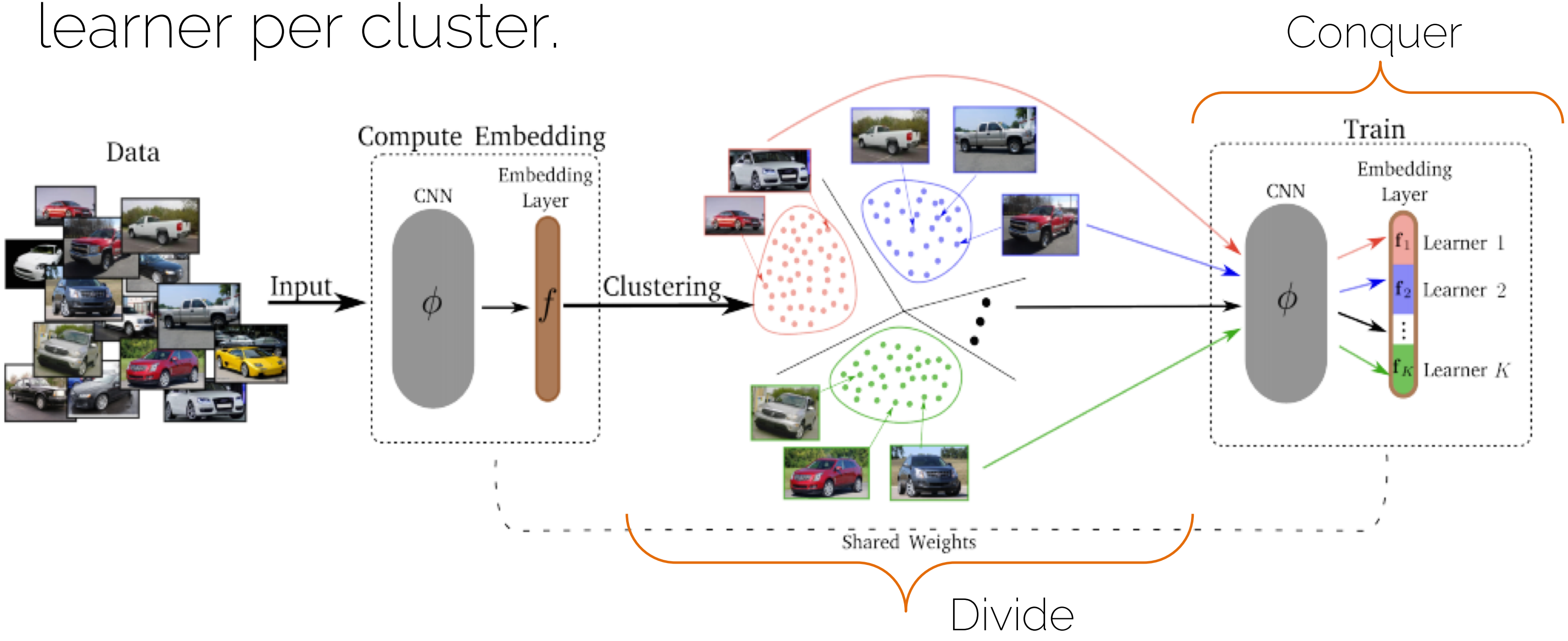
- Manmatha et al., Sampling matters for deep metric learning, (ICCV 2017) - original sampling method
- Xu et al., Deep asymmetric metric learning via rich relationship mining, (CVPR 2019)
- Duan et al., Deep embedding learning with discriminative sampling policy, (CVPR 2019)
- Wang et al., Ranked list loss for deep metric learning (CVPR 2019)
- Wang et al., Multi-similarity loss with general pair weighting for deep metric learning (CVPR 2019) - best performance

Improving similarity learning

- Loss:
  - Contrastive vs. triplet loss
- Sampling:
  - Choosing the best triplets to train with, sample the space wisely = diversity of classes + hard cases
- Ensembles:
  - Why not using several networks, each of them trained with a subset of triplets?
- Can we use a classification loss for similarity learning?

Ensembles

- Idea: divide the space into K clusters, and have one learner per cluster.



Sanakoyeu et al., Divide and Conquer the Embedding Space for Metric Learning, CVPR 2019

Ensembles: Divide and Conquer

1) Cluster the embedding space in K clusters using K-means.

2) Build K independent learners (fully connected layer) at the top of the CNN, where each learner corresponds to one cluster - DIVIDE

3) Until convergence, sample each mini-batch from one random cluster, and update only its corresponding learner.

4) After the network has converged finetune using all learners at the same time - CONQUER

5) Go back to (1) and repeat several times.

Ensembles: interesting works

- Opitz et al., BIER - Boosting Independent Embeddings Robustly, ICCV 2017 - train K independent networks.
- Elezi et al., The Group Loss for Metric Learning, arXiv 2020 - train K independent networks and concatenate their features.
- Yuan et al., Hard-Aware Deeply Cascaded Embedding, CVPR 2017 - concatenate features from different levels of the network.
- Wang et al., Ranked list loss for deep metric learning, CVPR 2019 - concatenate features from different levels of the network.
- Kim et al., Attention-based Ensemble for Deep Metric Learning, ECCV 2018 - use an attention mechanism such that each learner looks at different parts of the object.

Improving similarity learning

- Loss:
  - Contrastive vs. triplet loss
- Sampling:
  - Choosing the best triplets to train with, sample the space wisely = diversity of classes + hard cases
- Ensembles:
  - Why not using several networks, each of them trained with a subset of triplets?
- Can we use a classification loss for similarity learning?

Classification loss: interesting works

- Movshovitz-Attias et al., No Fuss Distance Metric Learning using Proxies, ICCV 2017 - learn "proxy" samples to keep as positives and negatives in the mini-batch).
- Teh et al., ProxyNCA++: Revisiting and Revitalizing Proxy Neighborhood Component Analysis, arXiv 2020 - a better way of using proxies, some of the best results in the field.
- Qian et al., SoftTriple Loss: Deep Metric Learning Without Triplet Sampling, ICCV 2019 - using multiple centers for class
- Elezi et al., The Group Loss for Deep Metric Learning, arXiv 2020 - refine the softmax probabilities via a dynamical system for better feature embedding.

Some results

	CUB-200-2011					CARS 196					Stanford Online Products			
Loss	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@10**	**R@100**	**NMI**
Triplet	42.5	55	66.4	77.2	55.3	51.5	63.8	73.5	82.4	53.4	66.7	82.4	91.9	89.5
Lifted Structure	43.5	56.5	68.5	79.6	56.5	53.0	65.7	76.0	84.3	56.9	62.5	80.8	91.9	88.7
Npairs	51.9	64.3	74.9	83.2	60.2	68.9	78.9	85.8	90.9	62.7	66.4	82.9	92.1	87.9
Facility Location	48.1	61.4	71.8	81.9	59.2	58.1	70.6	80.3	87.8	59.0	67.0	83.7	93.2	89.5
Angular Loss	54.7	66.3	76	83.9	61.1	71.4	81.4	87.5	92.1	63.2	70.9	85.0	93.5	88.6
Proxy-NCA	49.2	61.9	67.9	72.4	59.5	73.2	82.4	86.4	88.7	64.9	73.7	-	-	90.6
Deep Spectral	53.2	66.1	76.7	85.2	59.2	73.1	82.2	89.0	93.0	64.3	67.6	83.7	93.3	89.4
Classification	59.6	72	81.2	88.4	66.2	81.7	88.9	93.4	96	70.5	73.8	88.1	95	89.8
Bias Triplet	46.6	58.6	70.0	-	-	79.2	86.7	91.4	-	-	63.0	79.8	90.7	-
Group Loss	64.3	75.8	84.1	90.5	67.9	83.7	89.9	93.7	96.3	70.7	75.1	87.5	94.2	90.8
SoftTriple	65.4	76.4	84.5	90.4	69.3	84.5	90.7	94.5	96.9	70.1	78.3	90.3	95.9	92
HORDE	66.8	77.4	85.1	91	-	86.2	91.9	95.1	97.2	-	80.1	91.3	96.2	-

Jacob et al., Metric Learning With HORDE: High-Order Regularizer for Deep Embeddings, ICCV 2019

Some results

	CUB-200-2011					CARS 196					Stanford Online Products			
Loss+Sampling	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@10**	**R@100**	**NMI**
Samp. Matt.	63.6	74.4	83.1	90.0	69.0	79.6	86.5	91.9	95.1	69.1	72.7	86.2	93.8	90.7
Hier. triplet	57.1	68.8	78.7	86.5	-	81.4	88.0	92.7	95.7	-	74.8	88.3	94.8	-
DAMLRRM	55.1	66.5	76.8	85.3	61.7	73.5	82.6	89.1	93.5	64.2	69.7	85.2	93.2	88.2
DE-DSP	53.6	65.5	76.9	61.7	-	72.9	81.6	88.8	-	64.4	68.9	84.0	92.6	89.2
RLL 1	57.4	69.7	79.2	86.9	63.6	74	83.6	90.1	94.1	65.4	76.1	89.1	95.4	89.7
GPW	65.7	77.0	86.3	91.2	-	84.1	90.4	94.0	96.5	-	78.2	90.5	96.0	-
Teacher-Student														
RKD	61.4	73.0	81.9	89.0	-	82.3	89.8	94.2	96.6	-	75.1	88.3	95.2	-
Loss+Ensembles														
BIER 6	55.3	67.2	76.9	85.1	-	75.0	83.9	90.3	94.3	-	72.7	86.5	94.0	-
HDC 3	54.6	66.8	77.6	85.9	-	78.0	85.8	91.1	95.1	-	70.1	84.9	93.2	-
ABE 2	55.7	67.9	78.3	85.5	-	76.8	84.9	90.2	94.0	-	75.4	88.0	94.7	-
ABE 8	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	-
A-BIER 6	57.5	68.7	78.3	86.2	-	82.0	89.0	93.2	96.1	-	74.2	86.9	94.0	-
D and C 8	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	90.2
RLL 3 [45]	61.3	72.7	82.7	89.4	66.1	82.1	89.3	93.7	96.7	71.8	79.8	91.3	96.3	90.4
Group Loss	66.9	77.1	85.4	91.5	70.0	88.0	92.5	95.7	97.5	74.2	76.3	88.3	94.6	91.1
HORDE	63.9	75.7	84.4	91.2	-	88.0	93.2	96.0	97.9	-	80.1	91.3	96.2	-

So, which model to use?

CUB

	Concatenated (512-dim)		
	P@1	**RP**	**MAP@R**
Pretrained	51.05	24.85	14.21
Contrastive	67.21 ± 0.49	36.92 ± 0.28	26.19 ± 0.28
Triplet	64.40 ± 0.38	34.63 ± 0.36	23.79 ± 0.36
ProxyNCA	66.14 ± 0.32	35.48 ± 0.18	24.56 ± 0.18
Margin	65.48 ± 0.50	35.04 ± 0.24	24.10 ± 0.26
N. Softmax	65.43 ± 0.23	35.98 ± 0.22	25.20 ± 0.21
CosFace	67.19 ± 0.37	$\mathbf{37.36 \pm 0.23}$	$\mathbf{26.53 \pm 0.23}$
ArcFace	67.06 ± 0.31	37.23 ± 0.17	26.35 ± 0.17
FastAP	63.64 ± 0.24	34.45 ± 0.21	23.71 ± 0.20
SNR	$\mathbf{67.26 \pm 0.46}$	36.86 ± 0.20	26.10 ± 0.22
MS	65.93 ± 0.16	35.91 ± 0.11	25.16 ± 0.10
MS+Miner	65.75 ± 0.34	35.95 ± 0.21	25.21 ± 0.22
SoftTriple	66.20 ± 0.37	36.46 ± 0.20	25.64 ± 0.21

CARS

	Concatenated (512-dim)		
	P@1	**RP**	**MAP@R**
Pretrained	46.89	13.77	5.91
Contrastive	81.57 ± 0.36	35.72 ± 0.35	25.49 ± 0.41
Triplet	77.48 ± 0.57	32.85 ± 0.45	22.13 ± 0.45
ProxyNCA	83.25 ± 0.37	36.63 ± 0.34	26.39 ± 0.41
Margin	82.08 ± 2.41	34.71 ± 2.17	24.14 ± 2.25
N. Softmax	83.58 ± 0.29	36.56 ± 0.19	26.36 ± 0.21
CosFace	85.27 ± 0.23	36.72 ± 0.20	26.86 ± 0.22
ArcFace	83.95 ± 0.23	35.44 ± 0.26	25.24 ± 0.27
FastAP	78.20 ± 0.74	33.39 ± 0.67	22.90 ± 0.69
SNR	81.87 ± 0.35	35.40 ± 0.44	25.14 ± 0.49
MS	$\mathbf{85.29 \pm 0.31}$	$\mathbf{37.96 \pm 0.63}$	$\mathbf{27.84 \pm 0.77}$
MS+Miner	84.59 ± 0.29	37.70 ± 0.37	27.59 ± 0.43
SoftTriple	83.66 ± 0.22	36.31 ± 0.16	26.06 ± 0.19

When trained correctly (and using the same backbone, same embedding space and no extra-tricks to boost the results) the difference in accuracy between different models is not that large…

Musgrave et al., A Metric Learning Reality Check, arXiv 2020

Tips and tricks

- Simple baselines (contrastive loss, triplet loss and classification loss) actually perform well when trained correctly.

- Sampling is as important as the choice of loss function. Every method can be boosted by devising an intelligent sampling strategy.

- Some tricks may further improve the results (temperature for softmax, freezing batch-norm layers, using multiple centers per class, etc).

Tips and tricks

- Even naive ensembles may (significantly) boost performance.

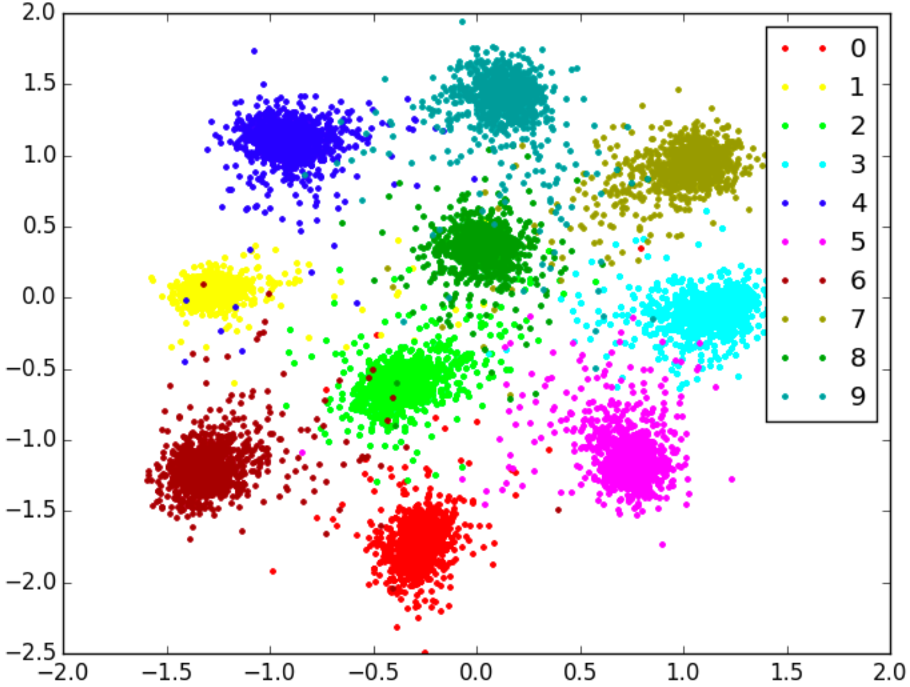
- Good out-of-box choices: Proxy-NCA and SoftTriple Loss → they perform well, and do not require a massive hyperparameter search (and have code online!).

- Contrastive loss and triplet loss give a similarity score in addition to the feature embedding.

- Stronger backbone choices (densenet) further improve the results.

Applications in vision

Siamese network on MNIST



Establishing image correspondences

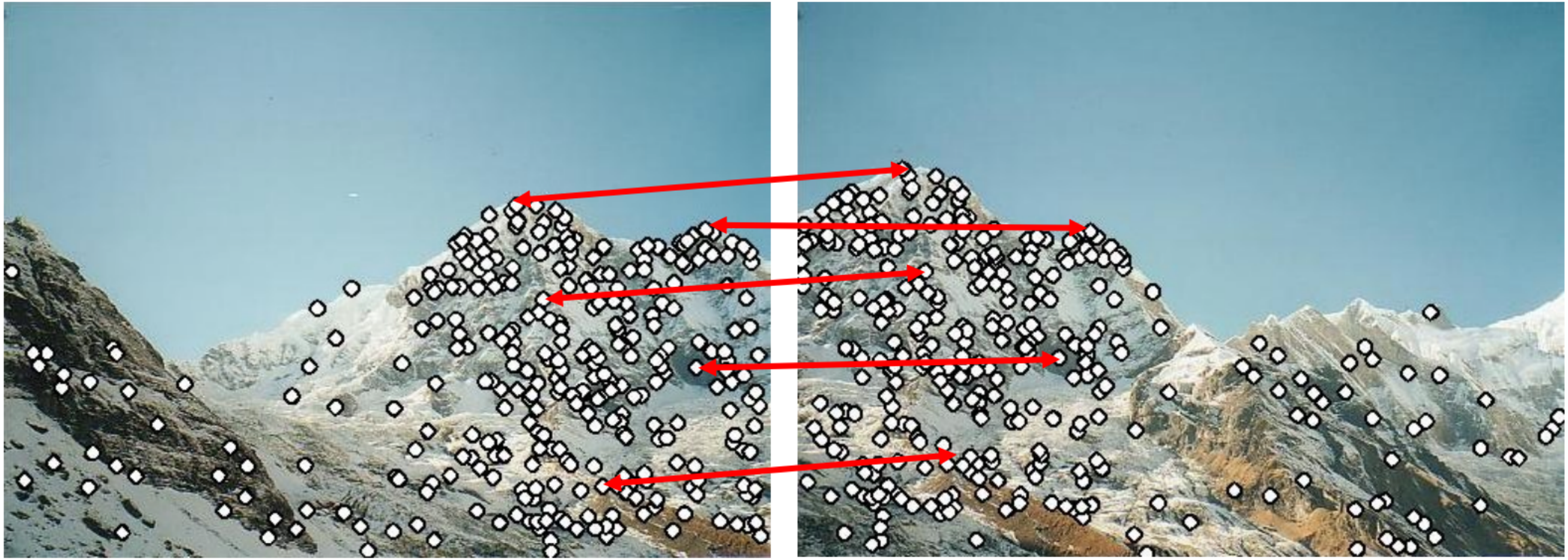


Image from University of Washington

Establishing image correspondences

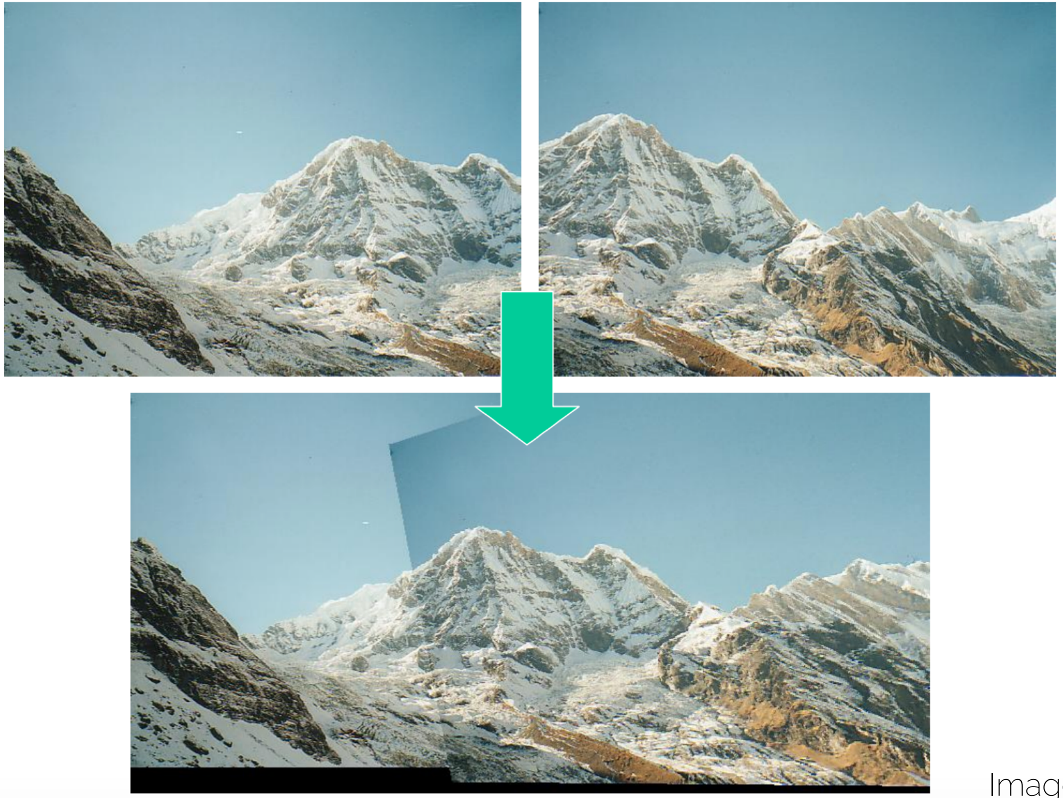


Image from University of Washington

Establishing image correspondences

- Used in a wide range of Computer Vision applications
  – Image stitching or image alignment
  – Object recognition
  – 3D reconstruction
  – Object tracking
  – Image retrieval

- Many of these applications are now targeted directly with Neural Networks as we will see in the course

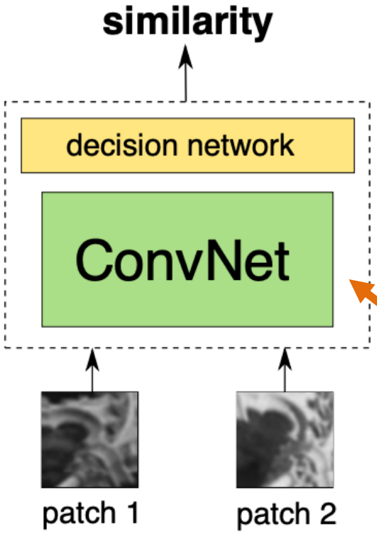
Establishing image correspondences

- Classic method pipeline
  - Extract manually designed feature descriptors
    - Harris, SIFT, SURF: most are based on image gradients
    - They suffer under extreme illumination or viewpoint changes
    - Slow to extract dense features
  - Match descriptors from the two images
    - Many descriptors are similar, one needs to filter out possible double matches and keep only reliable ones.

Sameer Agarwal et al. „Building Rome in a Day". ICCV 2009

Establishing image correspondences

- End-to-end learning for patch similarity



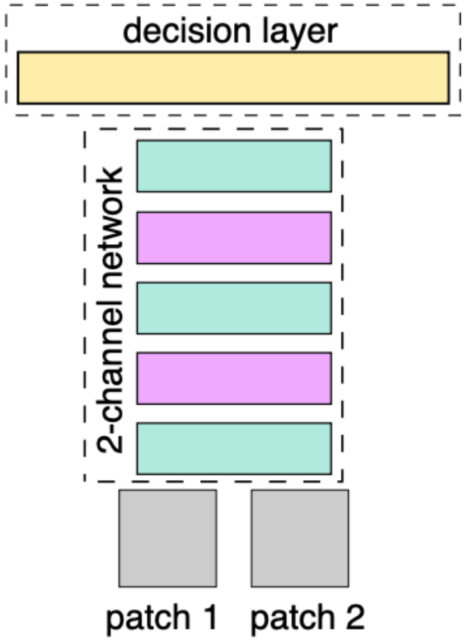
- Fast to allow dense extraction
- Invariant to a wide array of transformations (illumination, viewpoint)

Siamese network

S. Zagoruyko and N. Komodakis. „Learning to Compare Image Patches via Convolutional Neural Networks". CVPR 2015

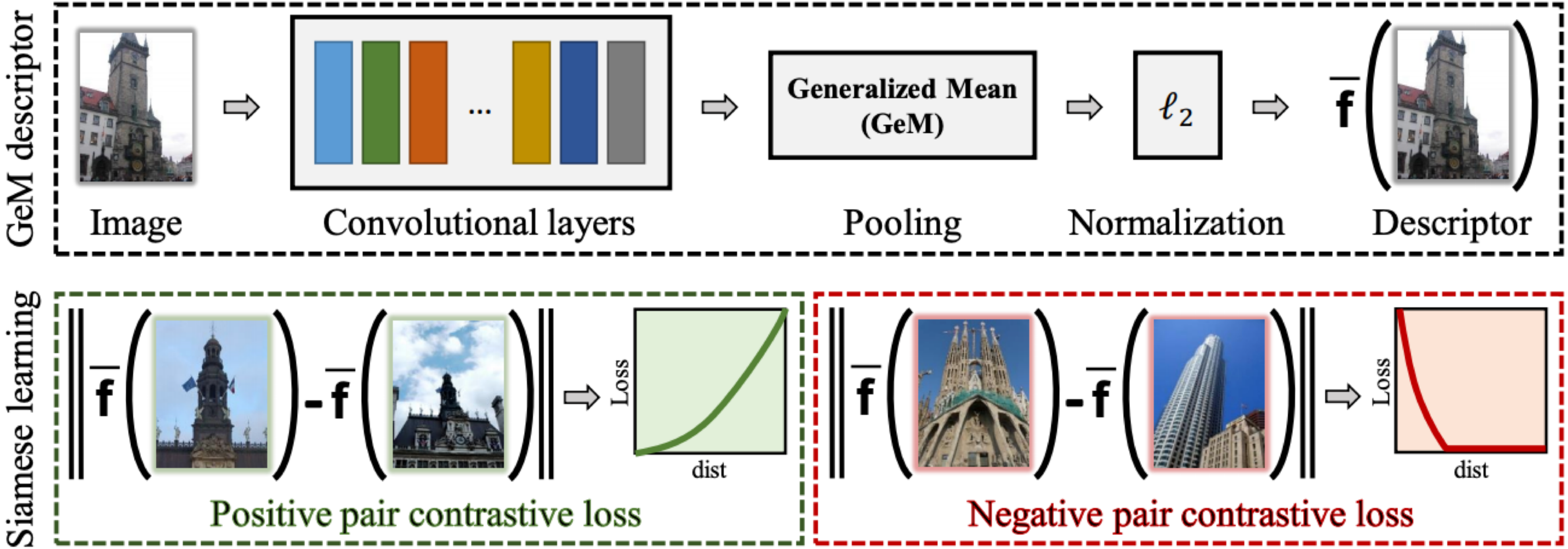
Establishing image correspondences

- Classic Siamese architecture
  - Shared layers
    - Simulated feature extraction
  - One decision layer
    - Simulates the matching



S. Zagoruyko and N. Komodakis. „Learning to Compare Image Patches via Convolutional Neural Networks". CVPR 2015

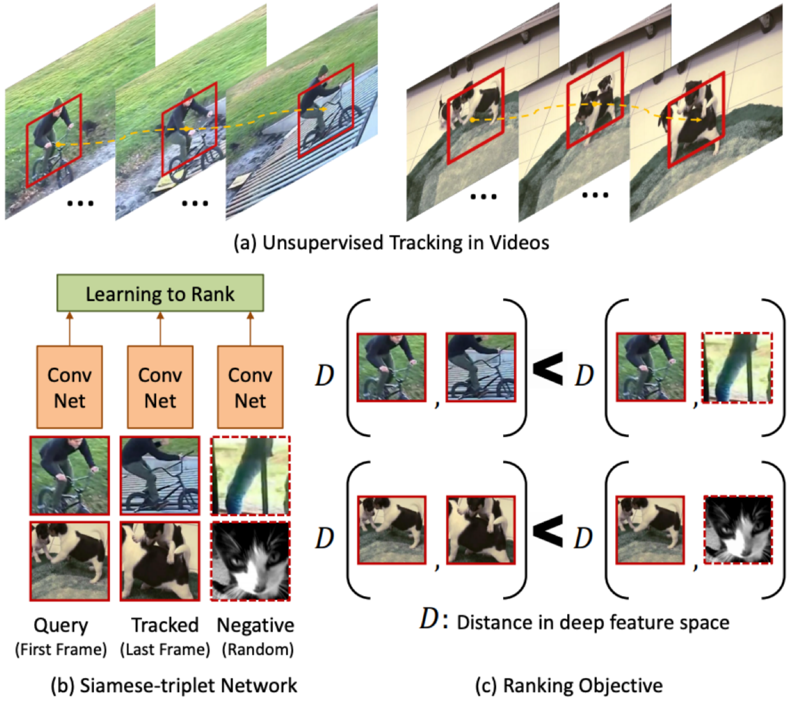
Image retrieval



Radenovic et al.. „Fine-tuning CNN Image Retrieval with No Human Annotation“. TPAMI 2018

Unsupervised learning

- Learning from videos
  - Tracking provides the supervision
  - Use those as positive samples
  - Extract random patches as negative samples



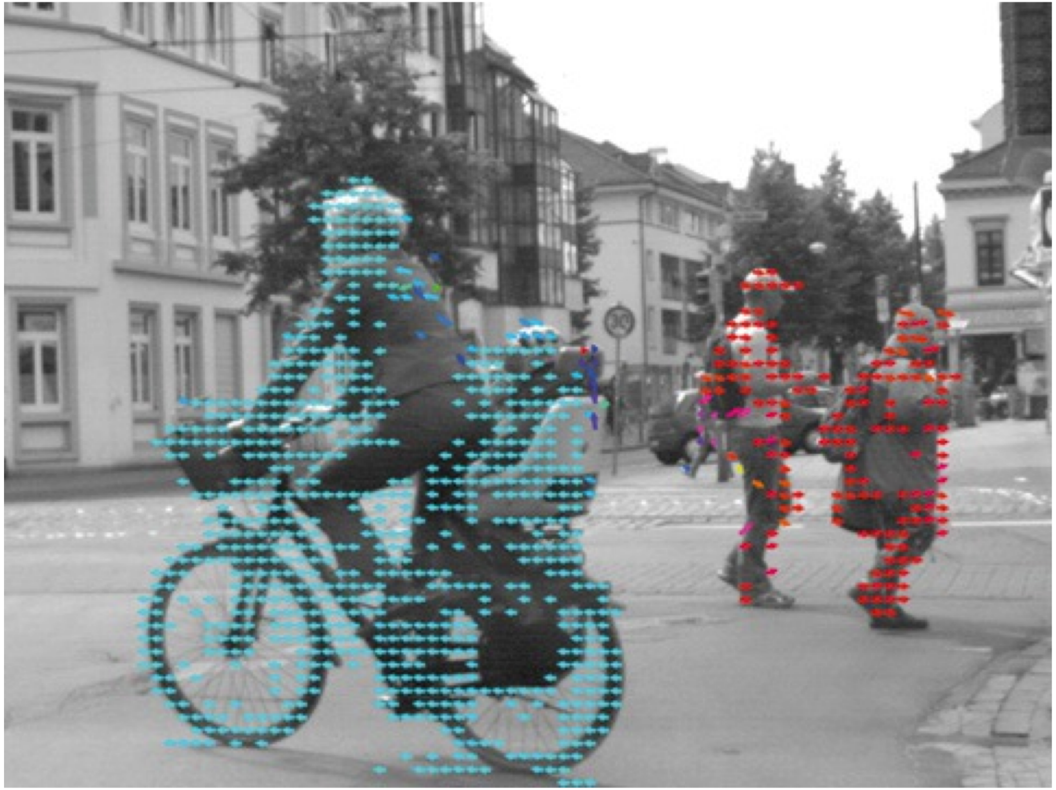
Wang and Gupta. „Unsupervised Learning of Visual Representations using Videos". ICCV 2015

Optical flow

- Input: 2 consecutive images (e.g. from a video)
- Output: displacement of every pixel from image A to image B

- Results in the "perceived" 2D motion, not the real motion of the object

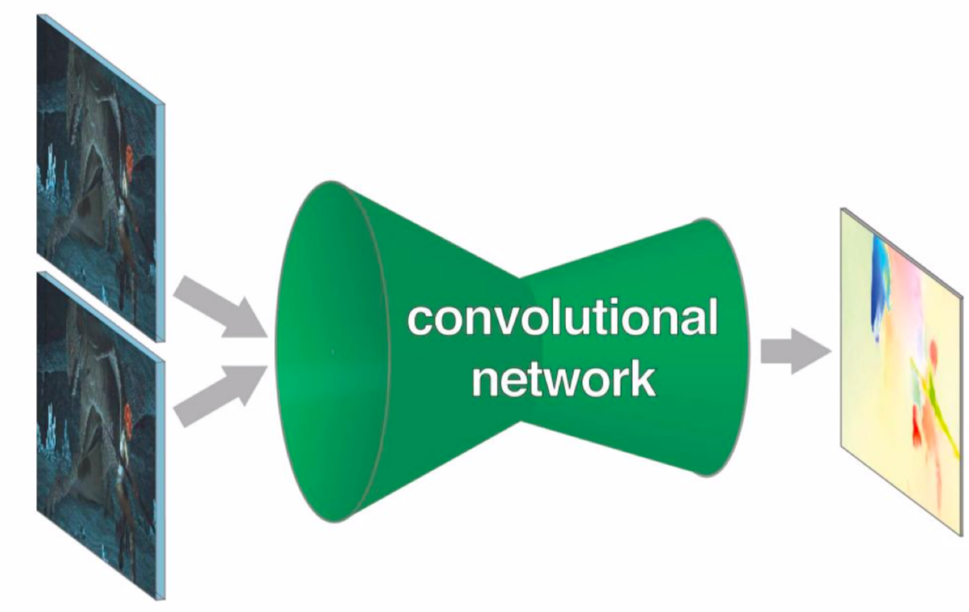
Optical flow



Optical flow

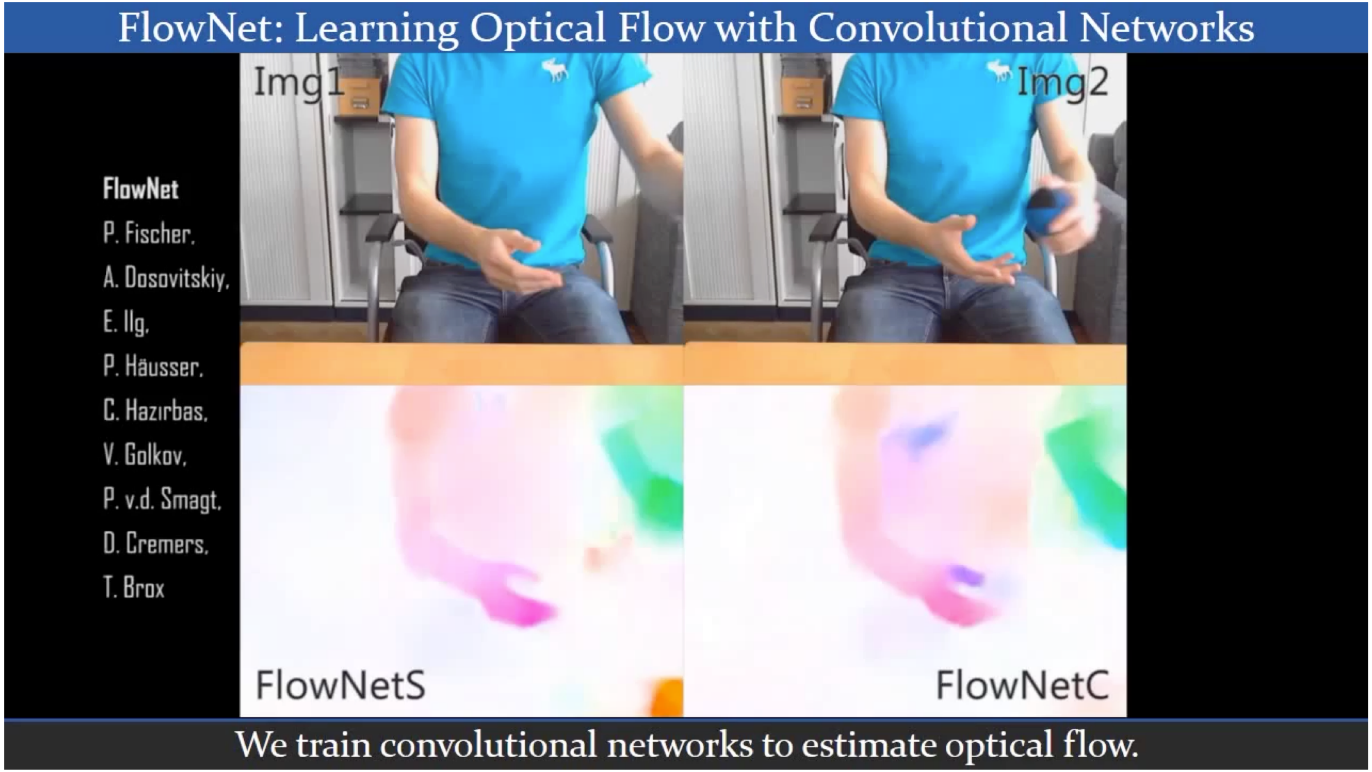
Optical flow with CNNs

- End-to-end supervised learning of optical flow



P. Fischer et al. „FlowNet: Learning Optical Flow With Convolutional Networks". ICCV 2015

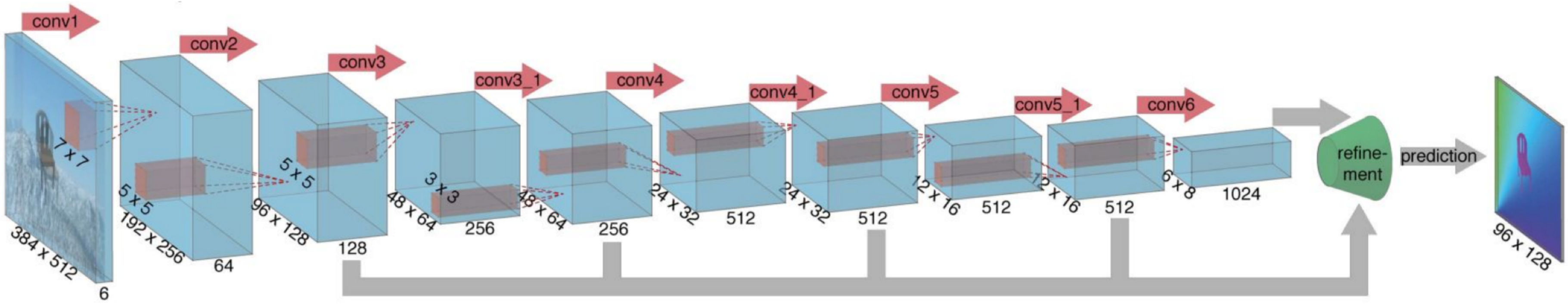
Optical flow with CNNs



P. Fischer et al. „FlowNet: Learning Optical Flow With Convolutional Networks". ICCV 2015

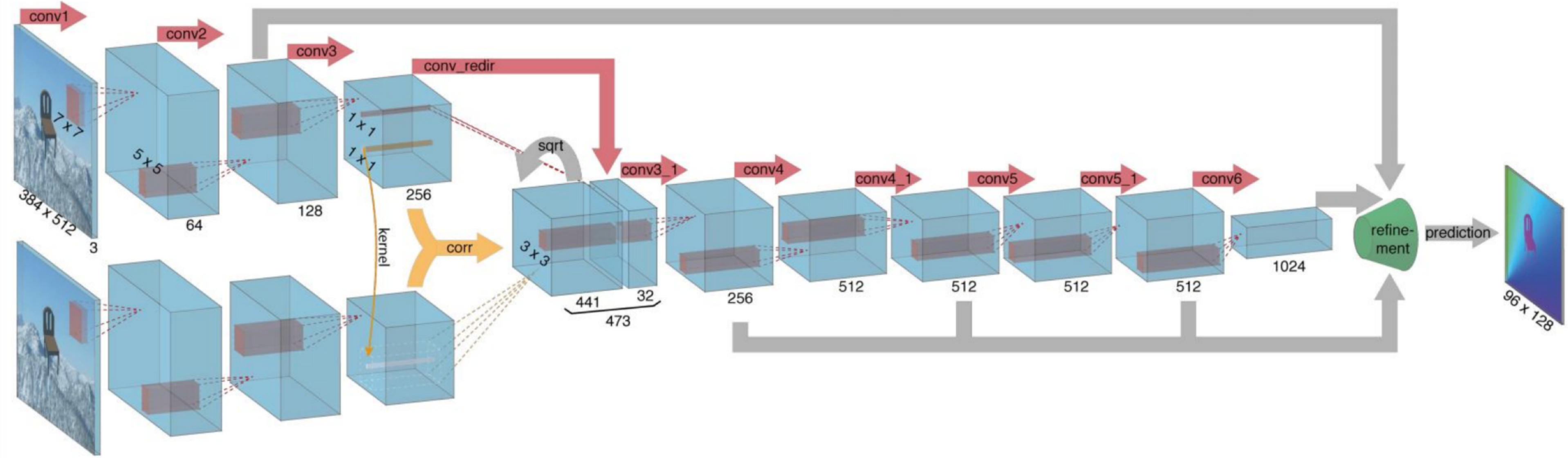
FlowNet: architecture 1

- Stack both images → input is now 2 x RGB = 6 channels



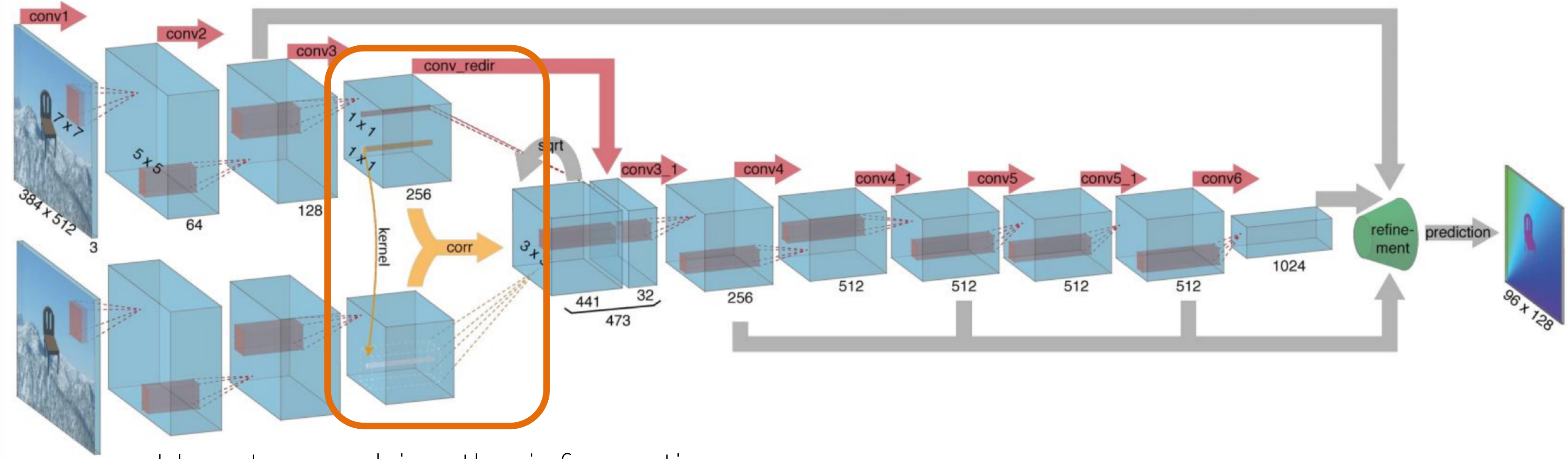
FlowNet: architecture 2

- Siamese architecture



FlowNet : architecture 2

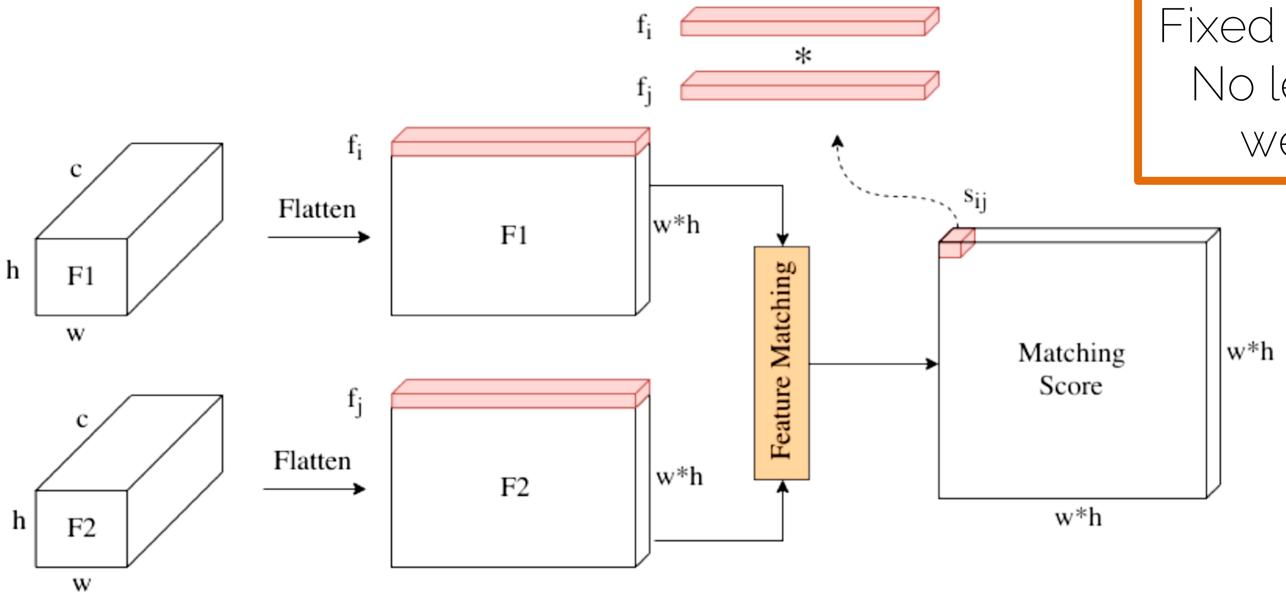
- Two key design choices



How to combine the information
from both images?

Correlation layer

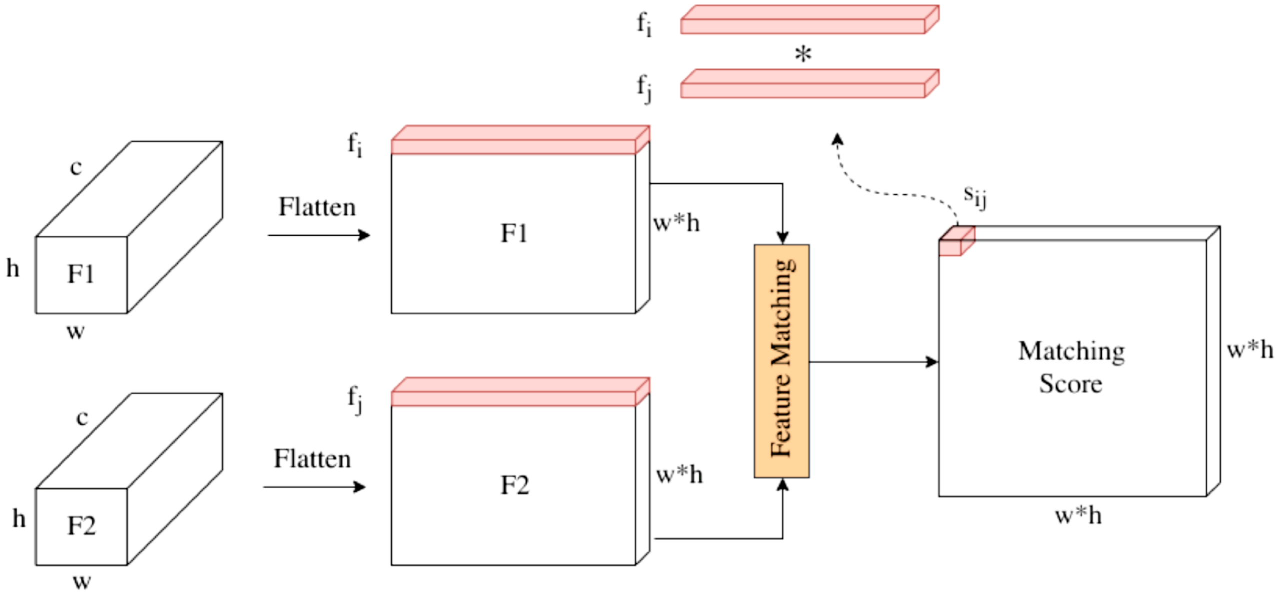
- Multiplies a feature vector with another feature vector



Fixed operation. No learnable weights!

Correlation layer

- The matching score represents how correlated these two feature vectors are



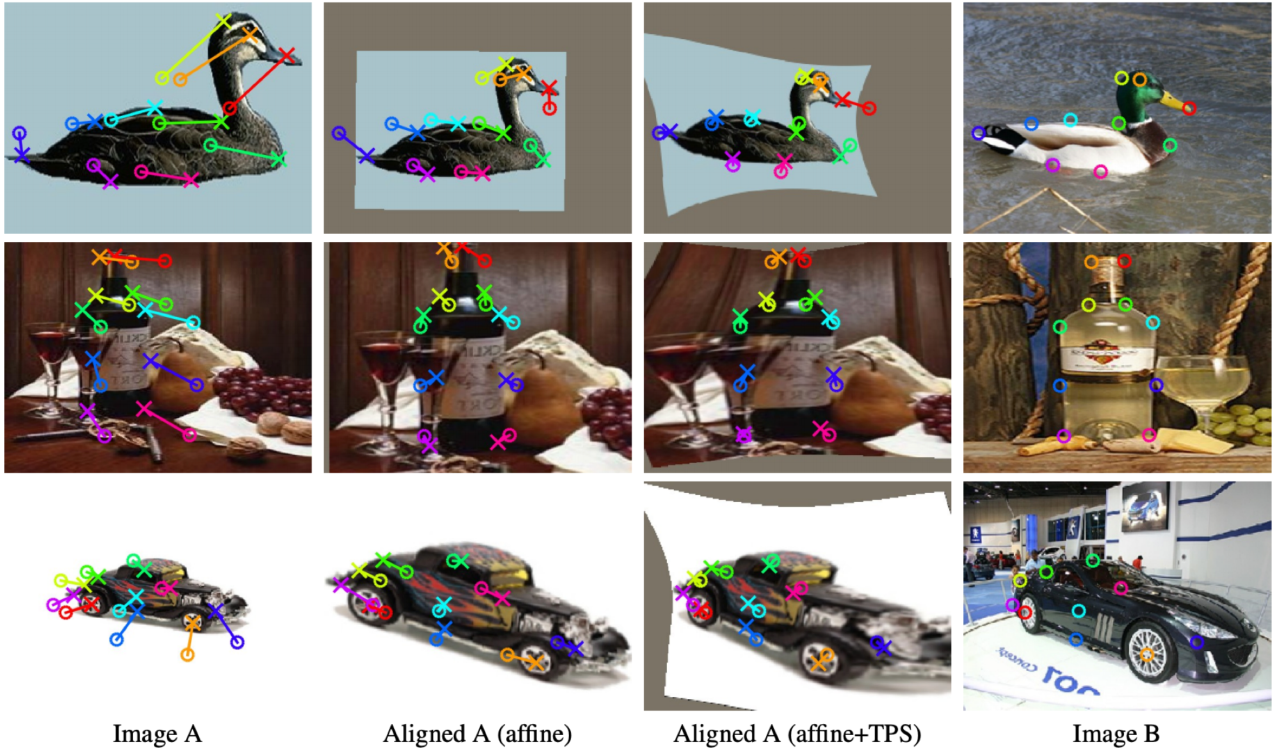
Correlation layer

- Useful for finding image correspondences

Find a transformation from image A to image B

I. Rocco et al. "Convolutional neural network architecture for geometric matching. CVPR 2017.

Correlation layer



I. Rocco et al. "Convolutional neural network architecture for geometric matching. CVPR 2017.

Siamese Neural Networks and Similarity Learning

Further references

- Savinov et al. „Quad-networks: unsupervised learning to rank for interest point detection“. CVPR 2017
- Ristani & Tomasi. „Features for Multi-Target Multi-Camera Tracking and Re-Identification“. CVPR 2018
- Chen et al. „Beyond triplet loss: a deep quadruplet network for person re-identification“. CVPR 2017