CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 2: Correspondence, Metric Learning

Virtual classrooms

- Virtual lectures on Zoom
  - Only host shares the screen
  - Keep video off and microphone muted
  - But please do speak up (remember to unmute!)
  - Slides uploaded on webpage just before class
- Virtual interactions on Zoom
  - Ask and answer plenty of questions
  - “Raise hand” feature on Zoom when you wish to speak
  - Post questions on chat window
  - Happy to try other suggestions!
- Lectures recorded and upload on Kaltura
  - Available under “My Media” on Canvas

Enrollment logistics

- To enroll if you are on the waitlist
  - Send **"Request to enroll"** email to instructor if on waitlist
  - Include CV, courses, project experience relevant to computer vision
- While on the waitlist
  - You are welcome to attend lectures even if on waitlist
  - To limit TA workload, we can grade only enrolled students
  - Most should be able to enroll eventually
- Canvas
  - All enrolled and waitlisted students should have access
- All announcements will be posted on Piazza
  - Send email to TA (CC instructor) if cannot access Piazza

Course details

- Class webpage:
  - http://cseweb.ucsd.edu/~mkchandraker/classes/CSE252D/Spring2021/
- Instructor email: mkchandraker@eng.ucsd.edu
- TA: Yu-Ying Yeh (Email: yuyeh@eng.ucsd.edu)
- Grading
  - 10% presentation
  - 60% assignments
  - 30% final exam
  - Ungraded quizzes
- Aim is to learn together, discuss and have fun!

Overall goals for the course

- Introduce the state-of-the-art in computer vision
- Study principles that make them possible
- Get understanding of tools that drive computer vision
- Enable one or all of several such outcomes
  - Pursue higher studies in computer vision
  - Join industry to do cutting-edge work in computer vision
  - Gain appreciation of modern computer vision technologies
- This is a great time to study computer vision!

Course details

- "Lightning" presentations
  - Provide a broad view of the field
  - An important skill to digest and present literature
  - Papers to be assigned by instructor
  - Order of presentation: alphabetic (Googledoc posted)
- Send recorded presentation video 3 days before class
  - Have shared a PPT template to be followed
  - Well-practiced and **fluent** presentation
  - **Time limit: 5 minutes**
  - Incorporate feedback from instructor or TA
  - Ask and answer questions

Course details

- Presentation format:
    1. Motivation and problem description
    2. Prior work
    3. Method overview
    4. Method analysis
    5. Experiments
    6. Future work and discussion

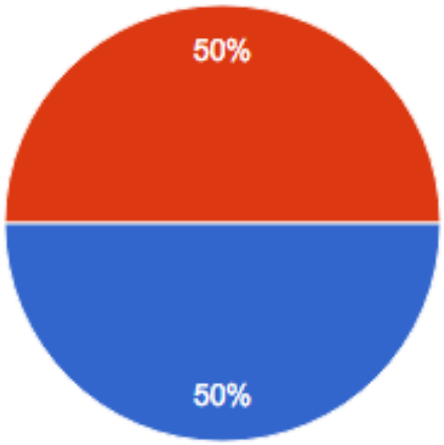
Papers for Wed, Apr 14

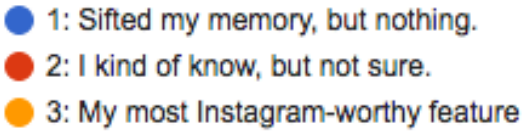
- SuperPoint: Self-Supervised Interest Point Detection and Description
  - https://arxiv.org/abs/1712.07629

- AnchorNet: A Weakly Supervised Network to Learn Geometry-Sensitive Features for Semantic Matching
  - https://arxiv.org/abs/1704.04749

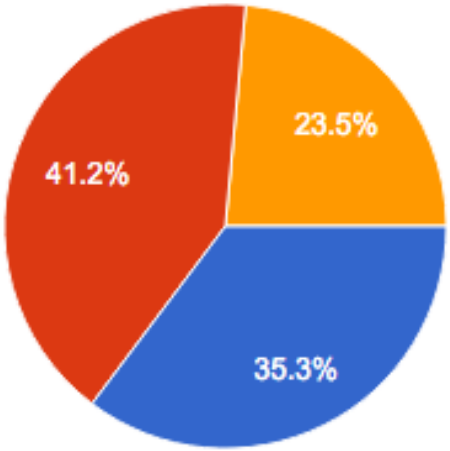
Module 1: Background

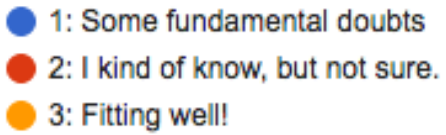
- Features





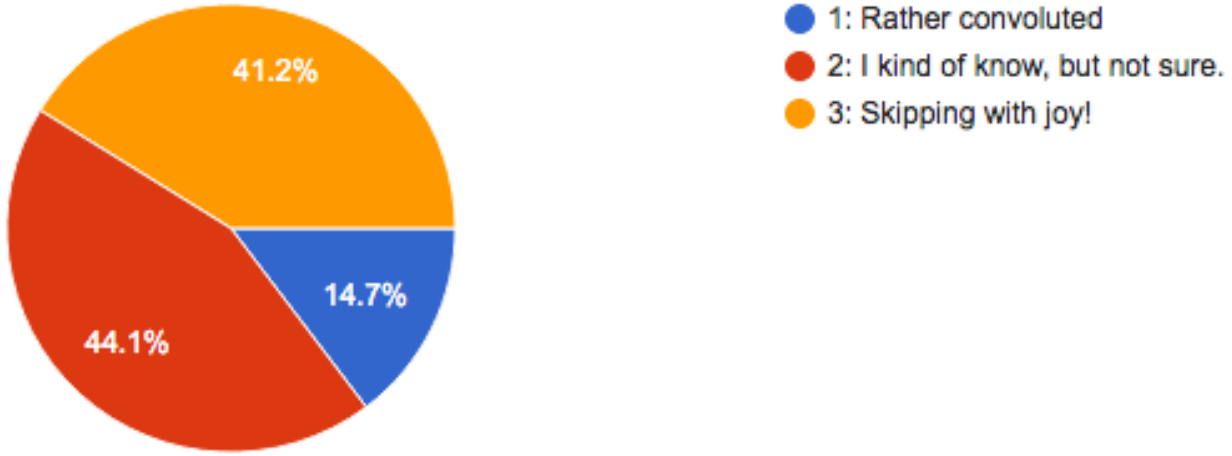
- Geometry





Module 1: Background

- Neural networks

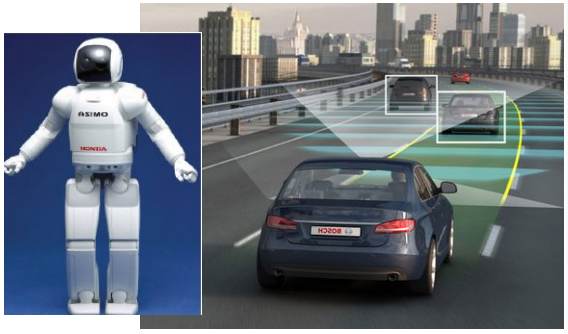


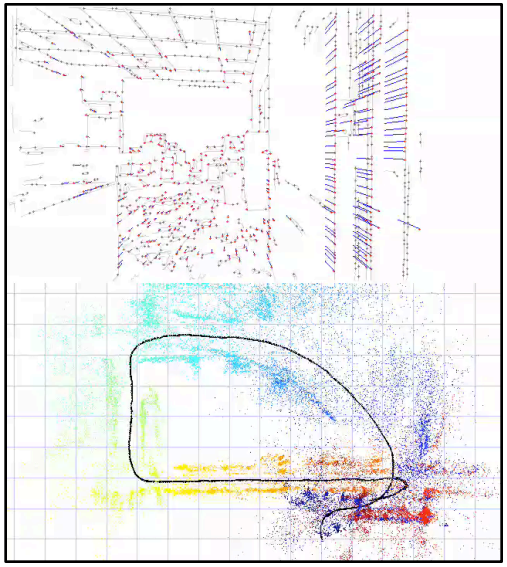
Correspondence

Visual correspondence aids 3D reconstruction

A key problem in 3D reconstruction: relate similar elements across a set of images

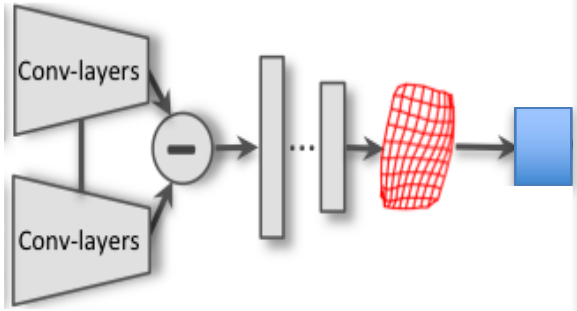
Geometric

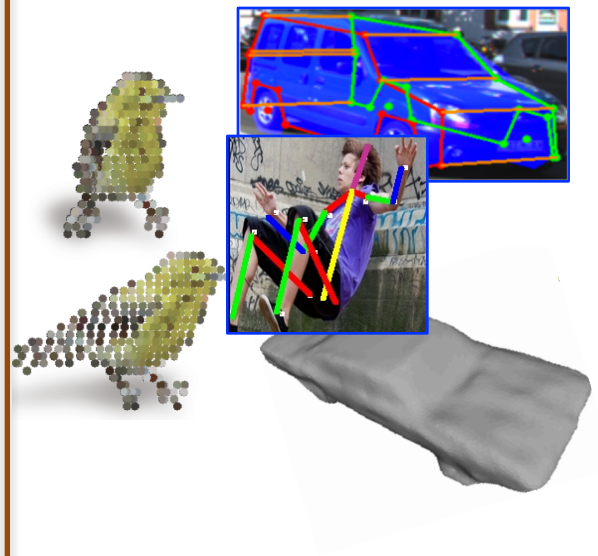




Photometric

Semantic





Simple matching methods

- SSD (Sum of Squared Differences)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$$

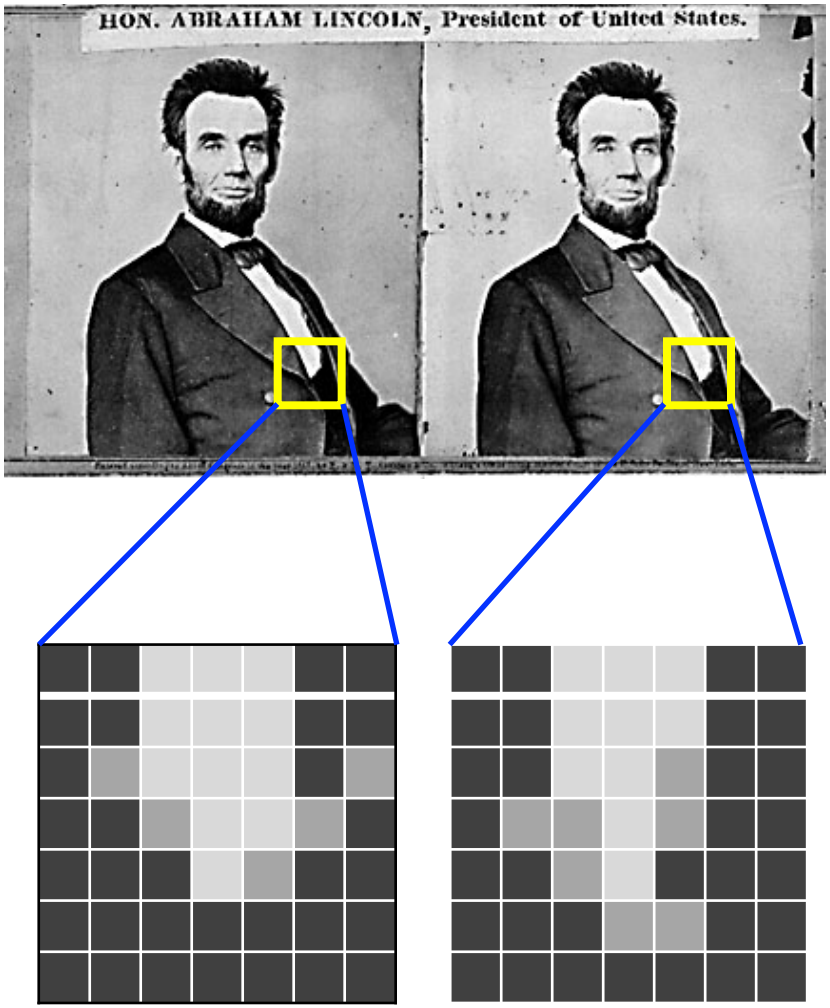
- NCC (Normalized Cross Correlation)

$$\sum_{x,y} \frac{(W_1(x,y) - \overline{W_1})(W_2(x,y) - \overline{W_2})}{\sigma_{W_1}\sigma_{W_2}}$$

$$\overline{W_i} = \frac{1}{n}\sum_{x,y} W_i \,, \quad \sigma_{W_i} = \sqrt{\frac{1}{n}\sum_{x,y}(W_i - \overline{W_i})^2}$$

(Mean) (Standard deviation)

- What advantages might NCC have over SSD?

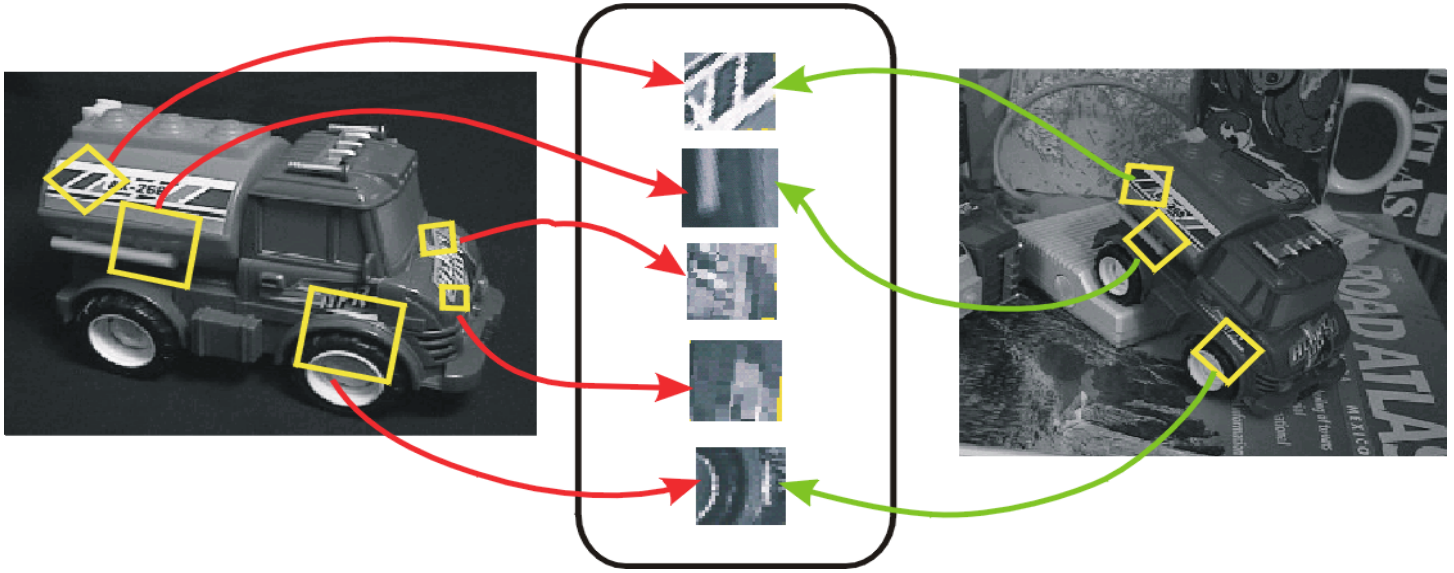


$W_1(x, y)$: k x k pixel patch in image 1

$W_2(x, y)$: k x k pixel patch in image 2

Idea of SIFT

- For better image matching, need to develop an interest operator invariant to scale and rotation.
- Also, need a descriptor robust to typical variations. The descriptor is the most-used part of SIFT.

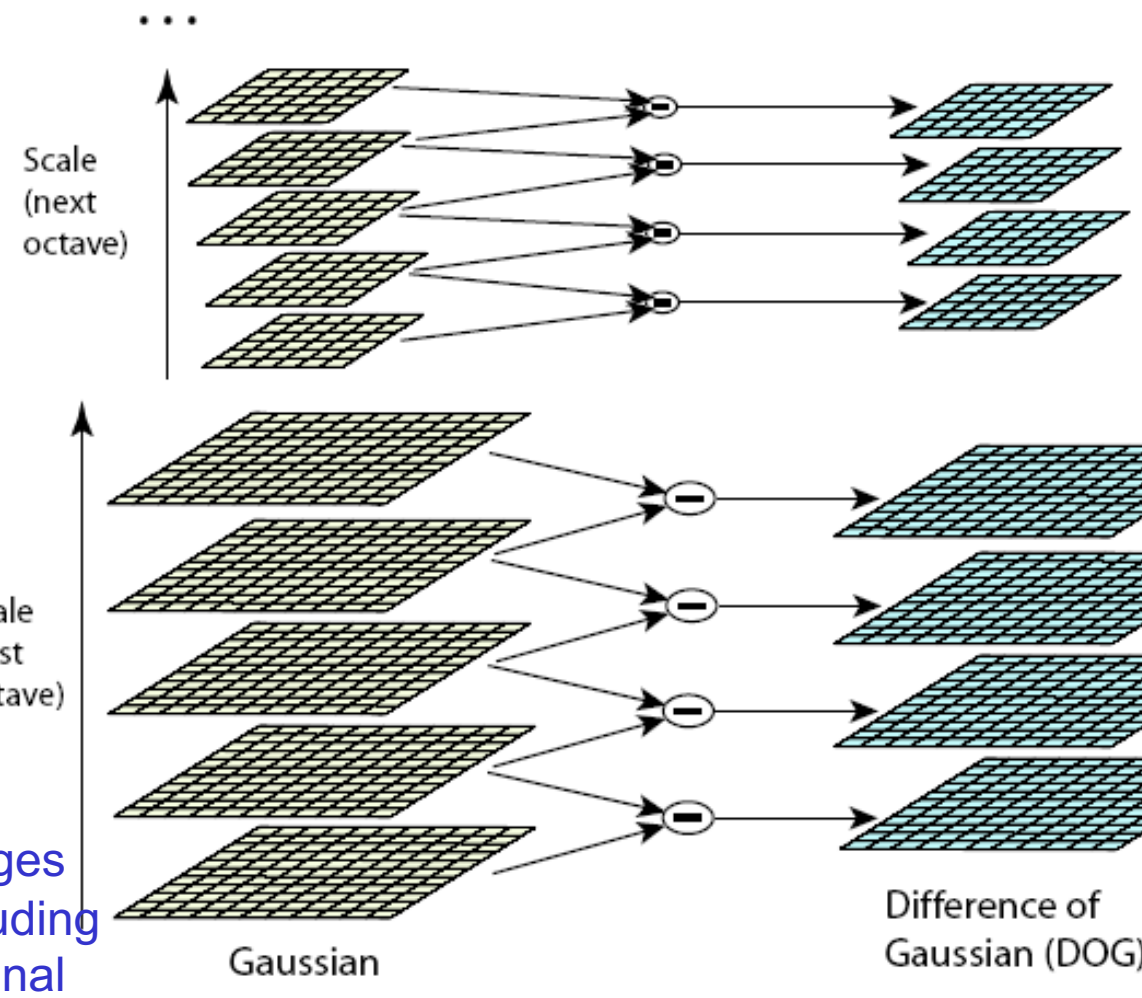


[Adapted from: D. Lowe]

1. Scale-space construction

- Scale space axioms: linearity, shift invariant, rotation invariant, no spurious extrema
- Gaussian filter uniquely satisfies axioms

Scale Space Pyramid



s+2 filters

$\sigma_{s+1}=2^{(s+1)/s}\sigma_0$

.

.

$\sigma_i=2^{i/s}\sigma_0$

.

.

$\sigma_2=2^{2/s}\sigma_0$

$\sigma_1=2^{1/s}\sigma_0$

σ_0

s+3 images including original

s+2 difference images

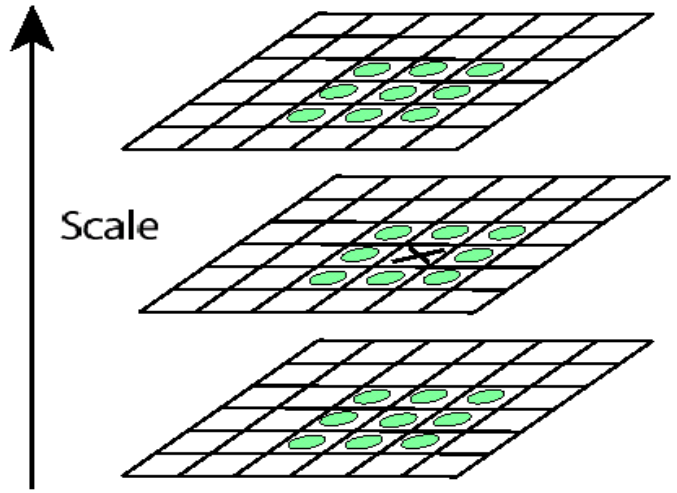
The parameter **s** determines the number of images per octave.

[Adapted from: D. Lowe]

2. Key point localization

s+2 difference images.
top and bottom ignored.
s planes searched.

- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

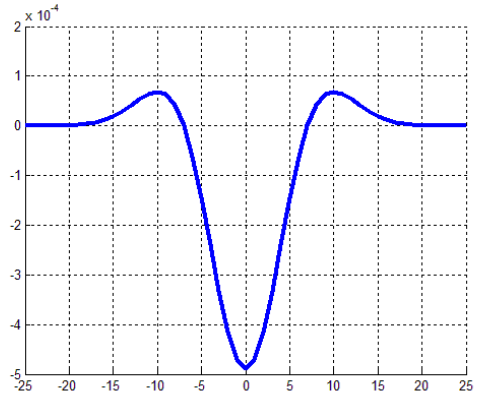


For each max or min found, output is the **location** and the **scale**.

[Adapted from: D. Lowe]

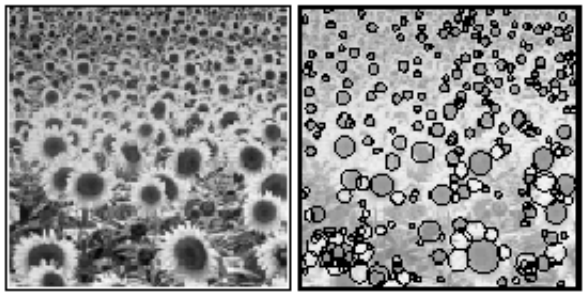
Difference of Gaussians

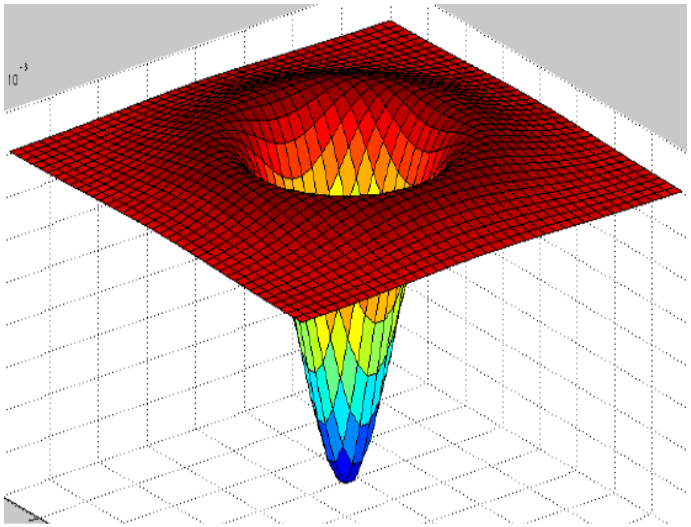
- Scale-space detection
  - Find local maxima across scale/space
  - A good "blob" detector



$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

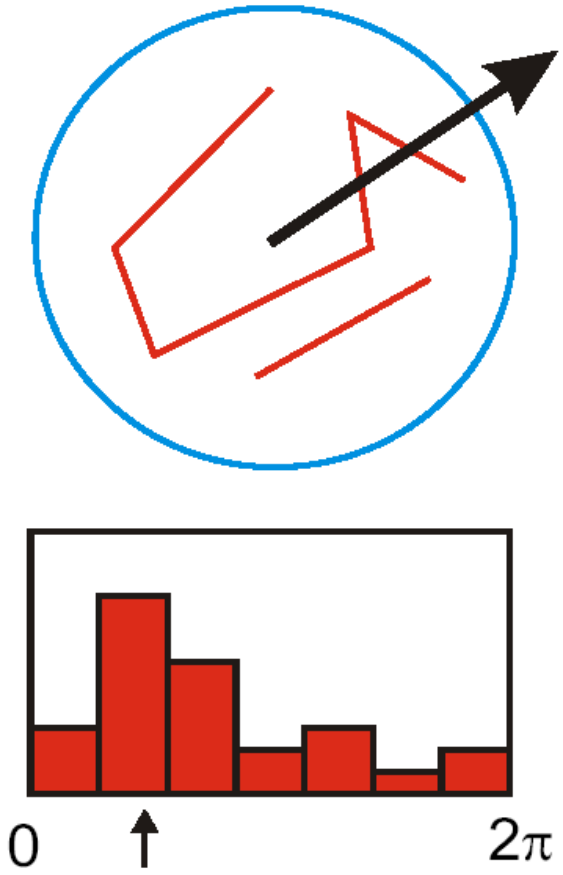
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G.$$





[T. Lindeberg IJCV 1998]

3. Orientation assignment



- Create histogram of local gradient directions at selected scale
- Assign canonical orientation at peak of smoothed histogram

If 2 major orientations, use both.

[Adapted from: D. Lowe]

4. Keypoint Descriptors

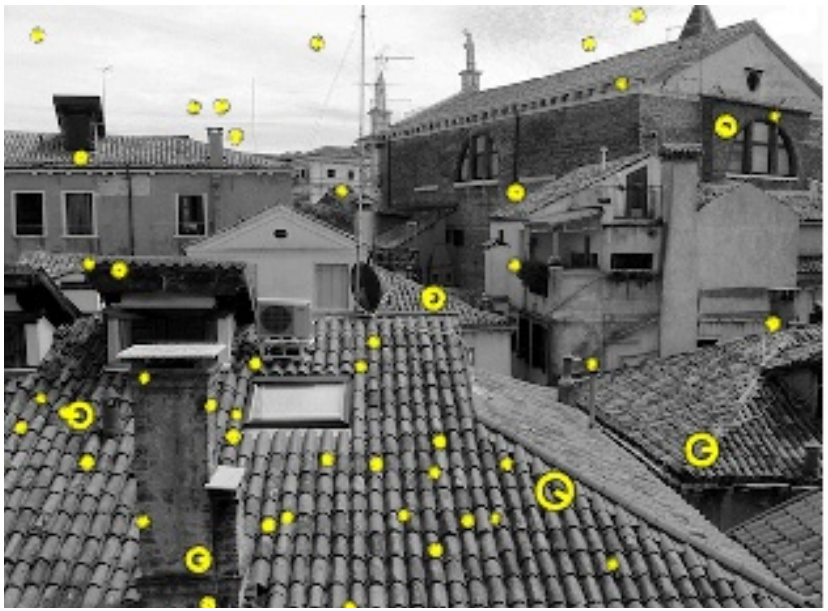
- At this point, each keypoint has
  - location
  - scale
  - orientation

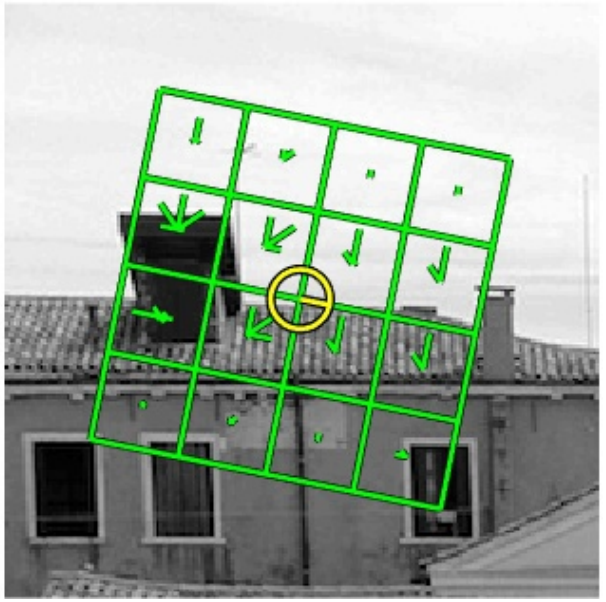
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - as invariant as possible to variations such as changes in viewpoint and illumination

[Adapted from: D. Lowe]

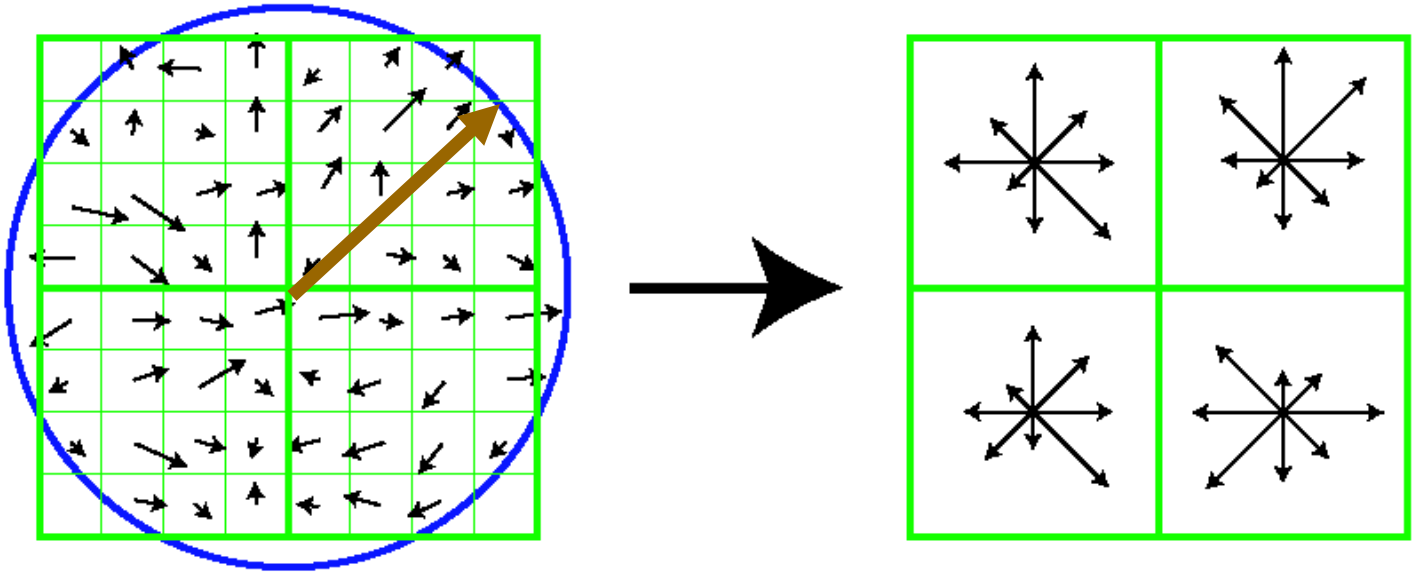
Normalization

- Rotate the window to standard orientation
- Scale the window size based on the scale at which the point was found.





SIFT Keypoint Descriptor (shown with 2 X 2 descriptors)

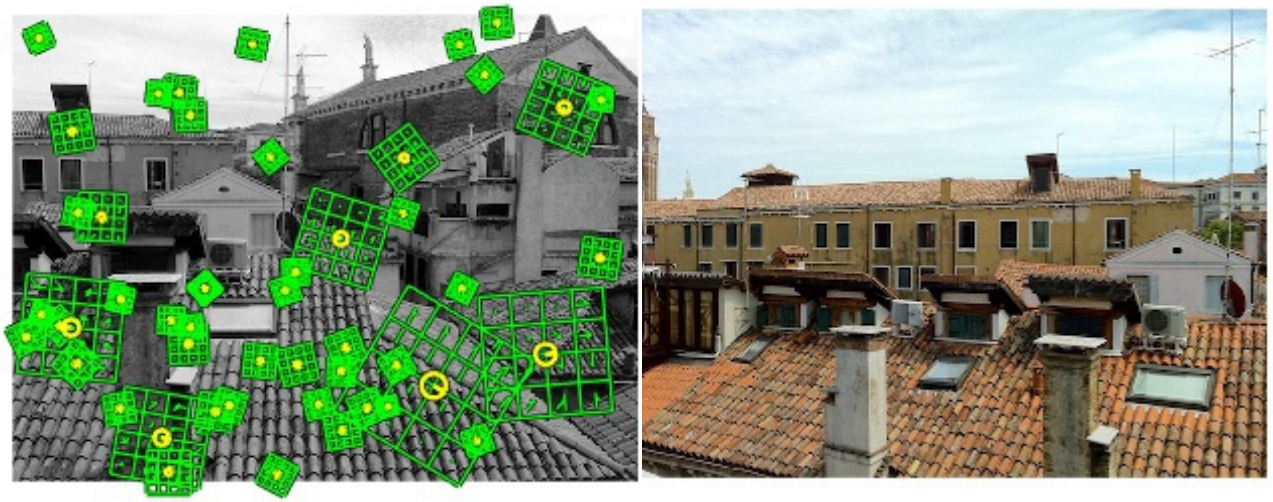


In implementation, 4x4 arrays of 8 bin histogram are used, a total of 128 features for one keypoint.

SIFT for Correspondence

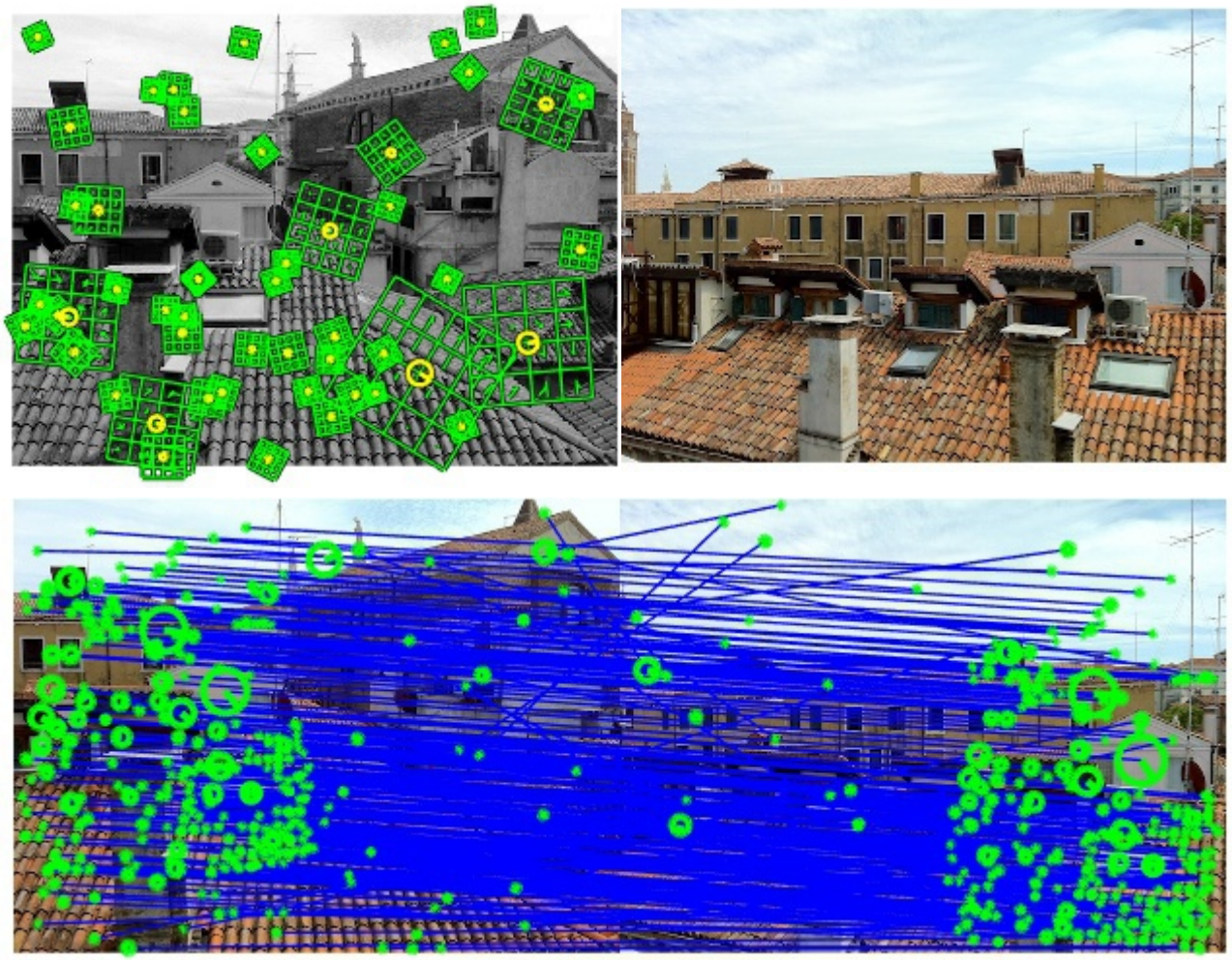
[Code and tutorial: https://www.vlfeat.org/overview/sift.html]

SIFT for Correspondence



[Code and tutorial: https://www.vlfeat.org/overview/sift.html]

SIFT for Correspondence



[Code and tutorial: https://www.vlfeat.org/overview/sift.html]

Cases where SIFT does not work

- Strong illumination changes
- Large out-of-plane rotations
- Non-rigid deformations or articulations
- Semantic correspondence

Learning Correspondence

Measuring patch similarity

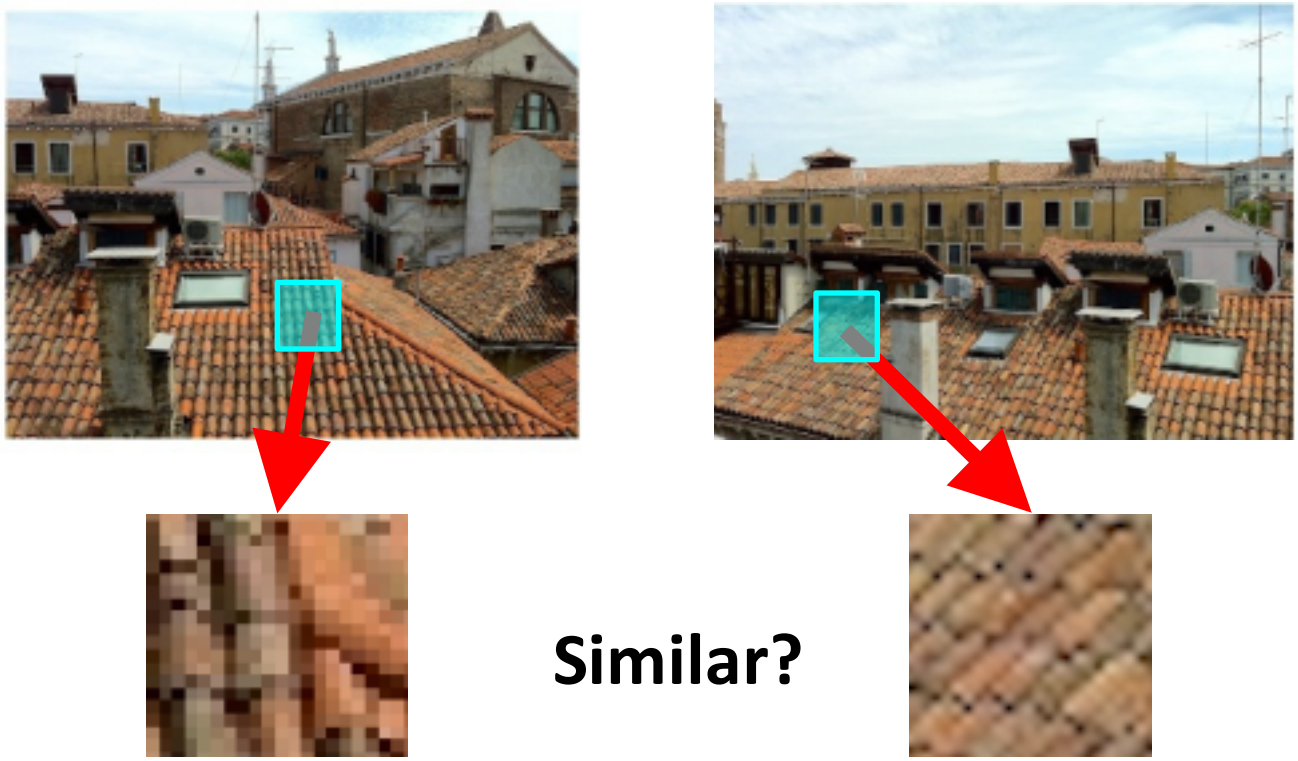


Image courtesy of [Vedaldi et al., VLFeat]

Measuring patch similarity

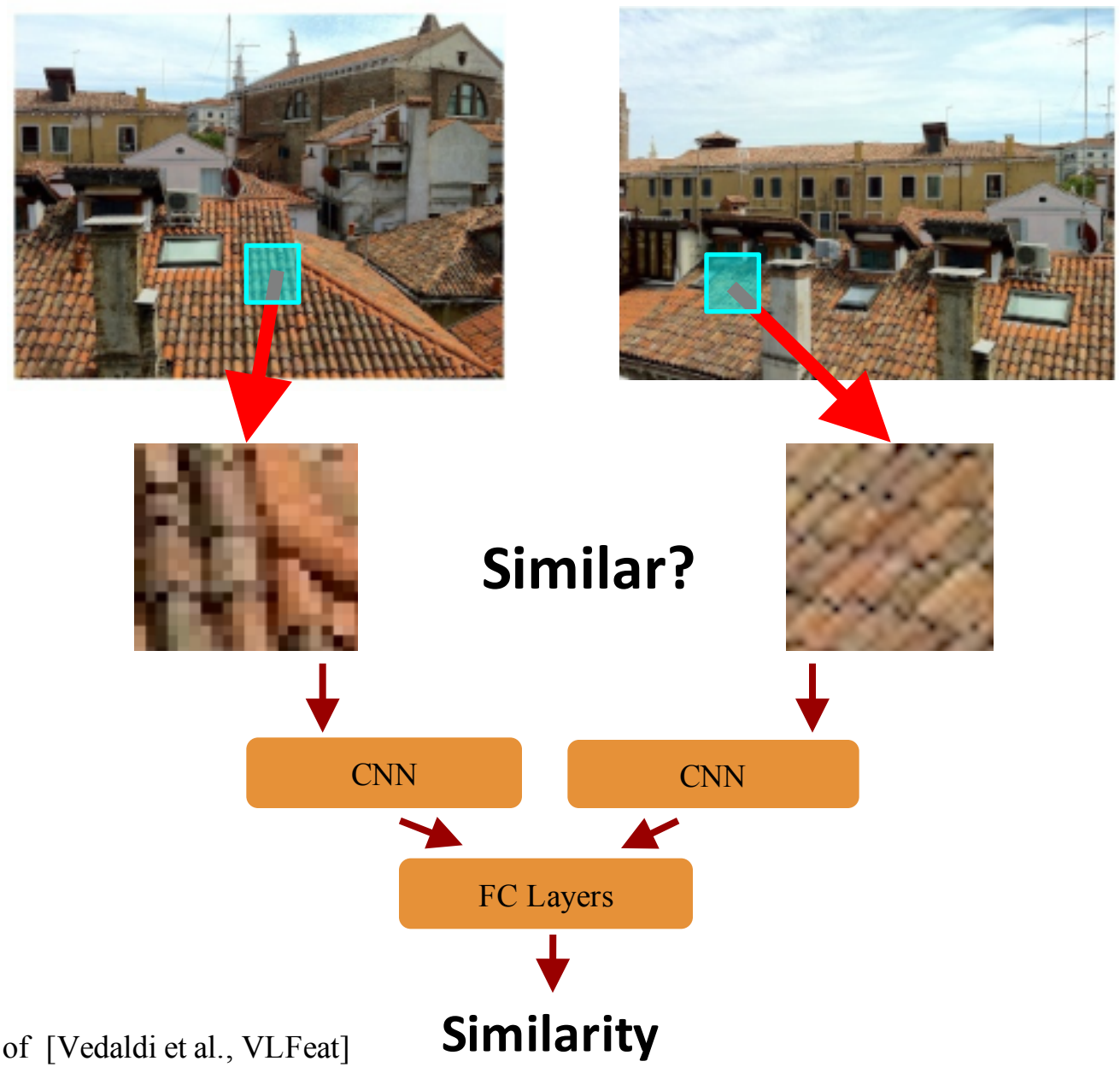


Image courtesy of [Vedaldi et al., VLFeat]

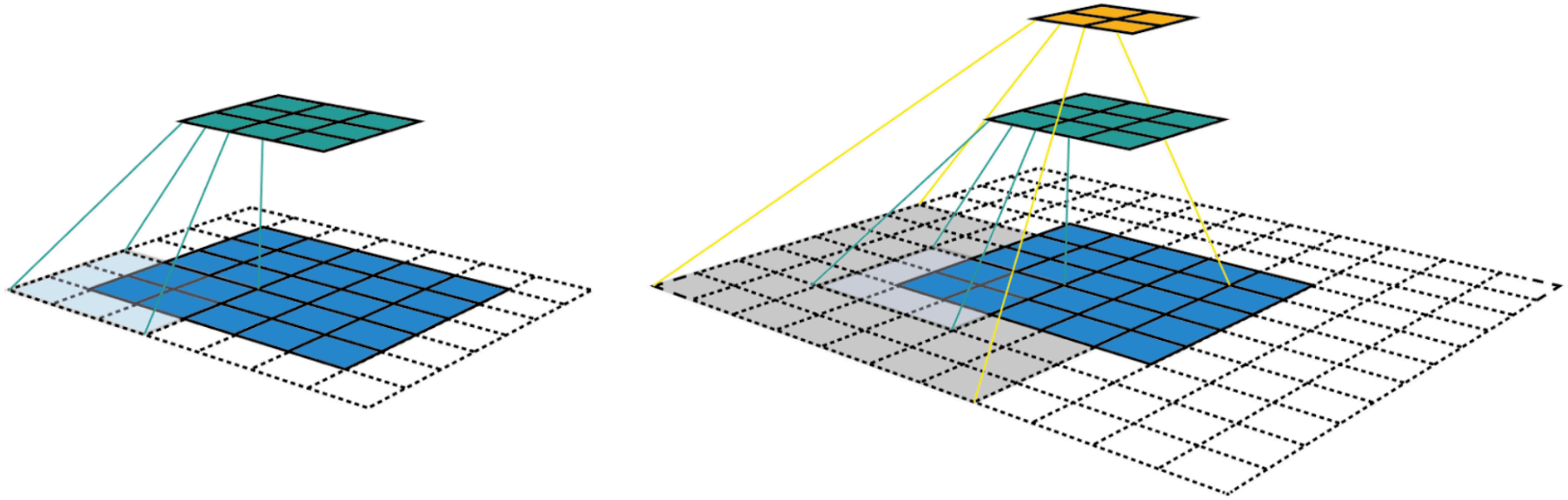
Spatial localization in classification CNNs

- Do Convnet features have the spatial specificity necessary for localization?
  - Trained from whole-image labels.
  - Convnet features correspond to large overlapping input windows.
  - Pooling may destroy local information in these features.

[Long et al., “Do Convnets Learn Correspondence?”, NeurIPS 2014]

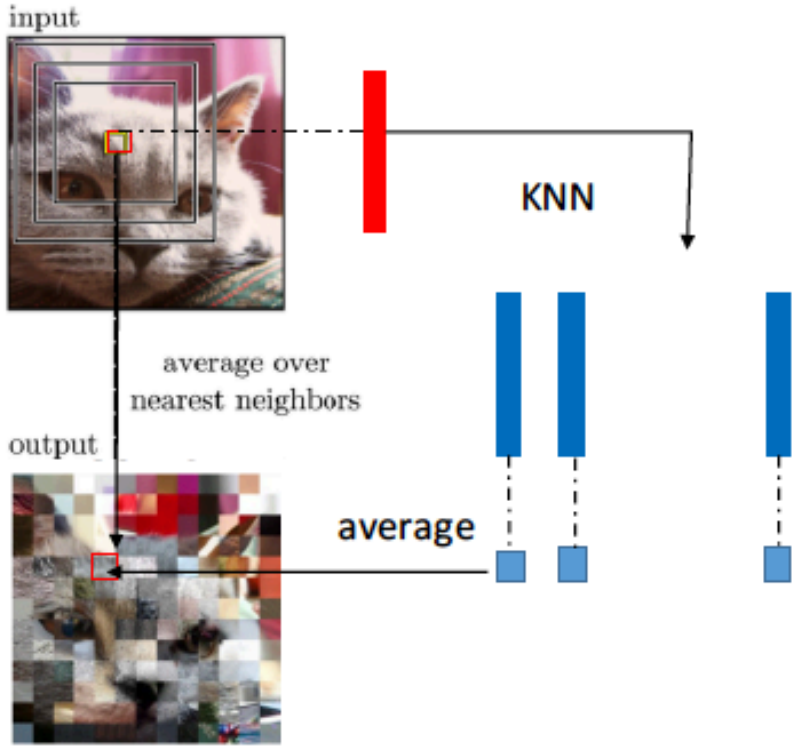
Receptive fields in CNNs

- The area in the input image "seen" by a unit in a CNN
- Units in deeper layers will have wider receptive fields



Spatial localization in classification CNNs

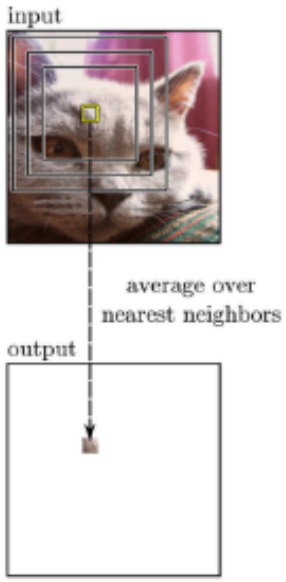
- Reconstruct image by replacing patches with averages of their top-k nearest neighbors in a Convnet feature space.
  - Associate a feature with a patch in the original image centered at the receptive field with size equal to the stride
  - Replace each patch with an average over k nearest neighbor patches.

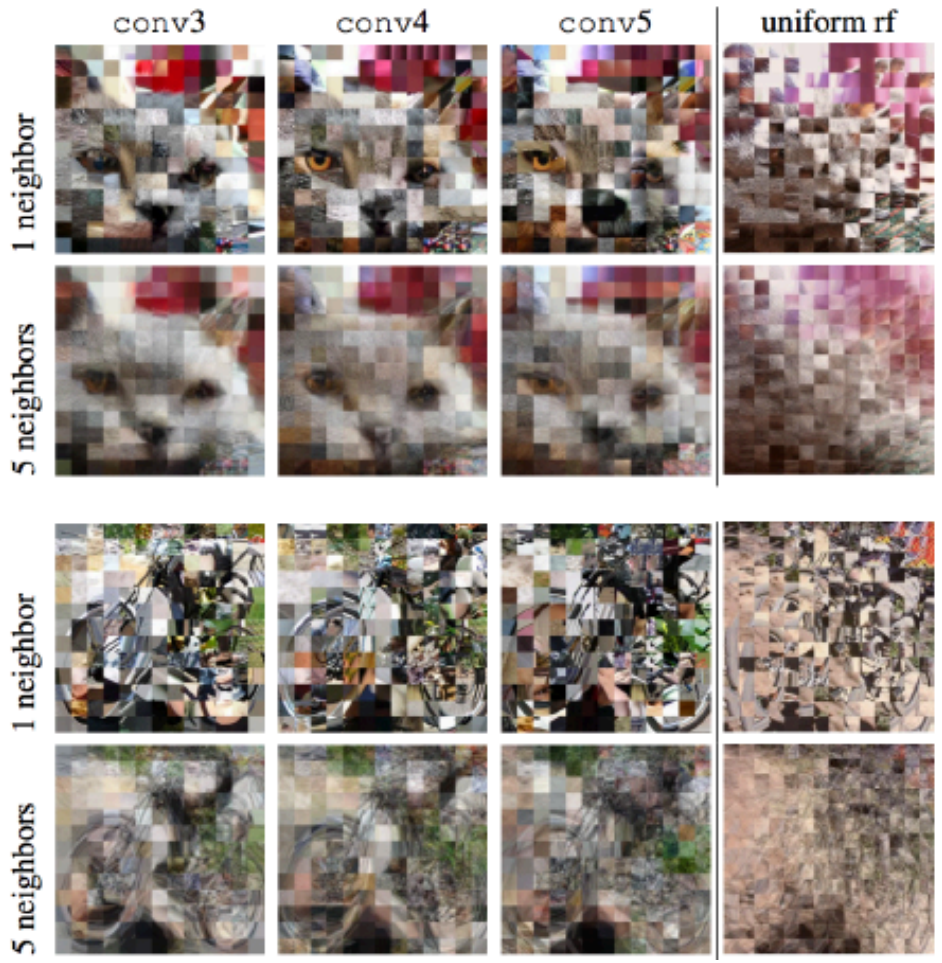


layer	rf size	stride
conv1	11 × 11	4 × 4
conv2	51 × 51	8 × 8
conv3	99 × 99	16 × 16
conv4	131 × 131	16 × 16
conv5	163 × 163	16 × 16
pool5	195 × 195	32 × 32

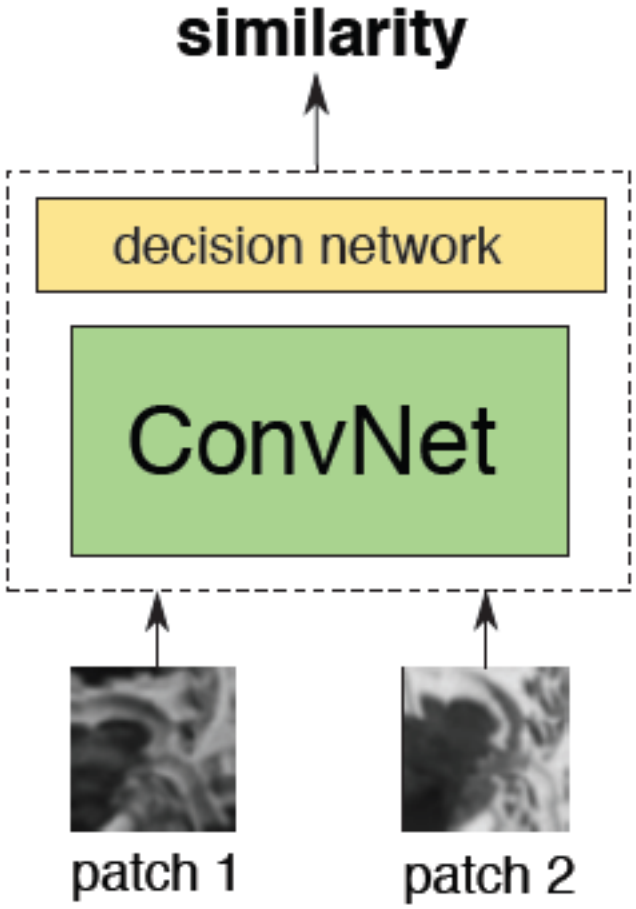
Spatial localization in classification CNNs

- The features retain useful correspondence to their input centers.
  - Notable features are replaced in their corresponding locations.
  - The replacement appears more semantic and less visually specific as the layer deepens





CNNs for learning correspondence



Idea:

- Siamese network to decide patch similarity
- Use intermediate activations as features.

Advantages:

- Easy to train

Issues:

- Inefficient: extract features for overlapping regions within patches
- $O(n^2)$ feed-forward passes to compare n patches in each image.

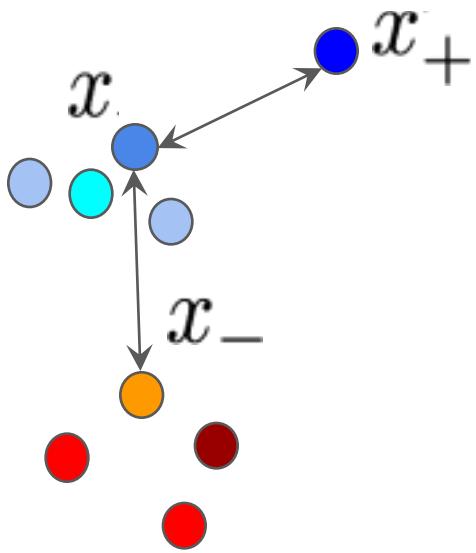
[Zagoruyko and Komodakis, CVPR 2015]

Need for a metric in correspondence learning

- Learn a feature space directly optimized for correspondence
- Intermediate activations of patch similarity are surrogate features
- Mapping an image to a metric space
  - Metric Space: Distance relationship = Class membership

$$\|f(x) - f(x_+)\| \to 0$$

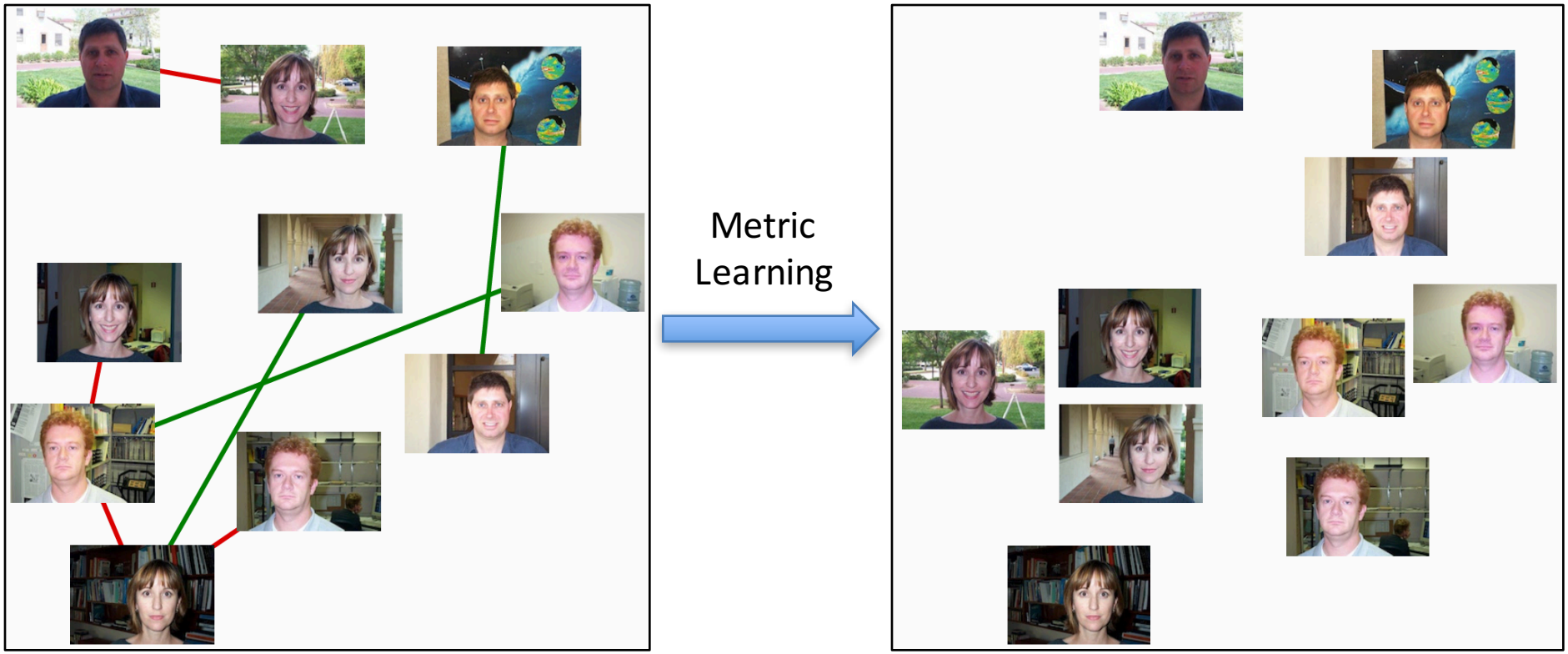
$$\|f(x) - f(x_-)\| \geq m$$



Metric Learning

Metric

- A metric quantifies distance between any two members of a set
- Induces a similarity measure

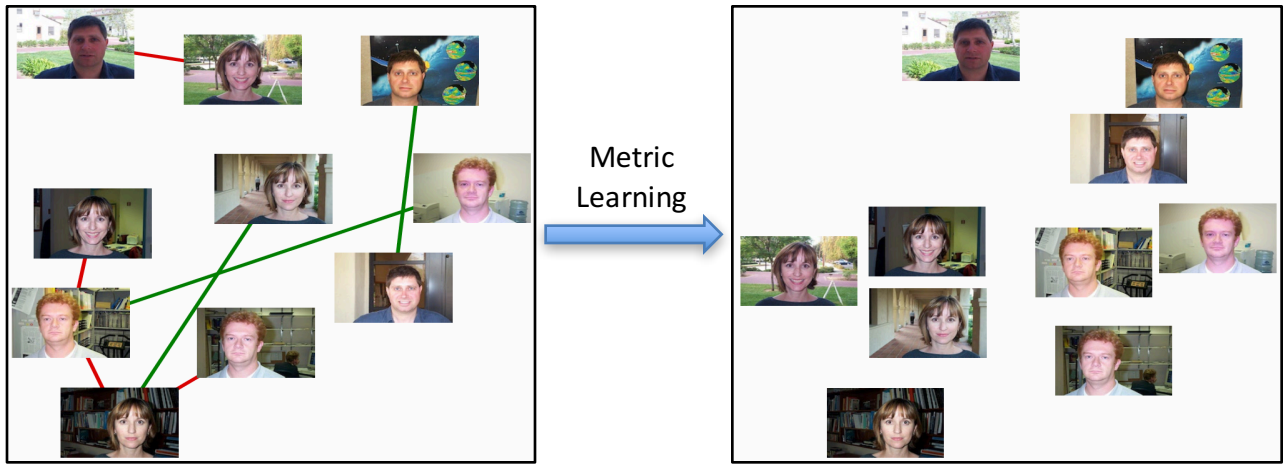


Metric

- A metric quantifies distance between any two members of a set
- Induces a similarity measure

- Properties of a metric
  - Non-negativity: $f(x, y) \geq 0$
  - Identity: $f(x, y) = 0$ if and only if $x = y$
  - Symmetry: $f(x, y) = f(y, x)$
  - Triangle inequality: $f(x, y) + f(y, z) \geq f(x, z)$

- Example of metric : $f(x, y) = \| x - y \|$
- Example of non-metric : $f(x, y) = x^2 y^2$

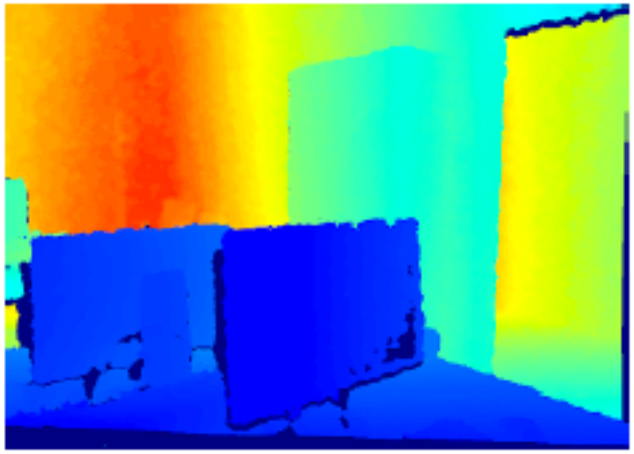


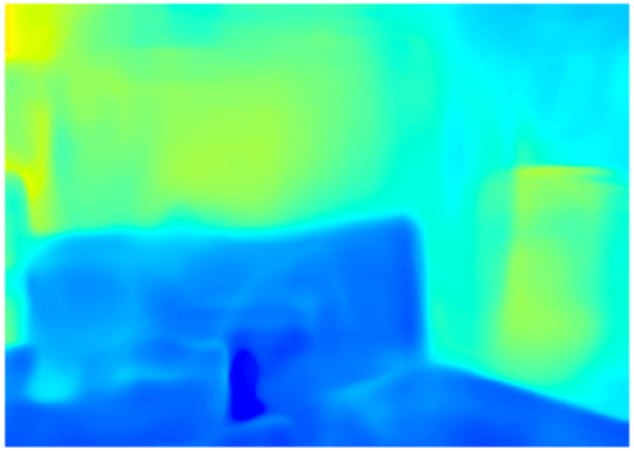
Metric

- A metric quantifies distance between any two members of a set
- Induces a similarity measure

- Properties of a metric
  - Non-negativity: $f(x, y) \geq 0$
  - Identity: $f(x, y) = 0$ if and only if $x = y$
  - Symmetry: $f(x, y) = f(y, x)$
  - Triangle inequality: $f(x, y) + f(y, z) \geq f(x, z)$

- Sometimes, the choice of metric is trivial
  - Depth estimation: distances in 3D, so Euclidean distance a good metric





Metric

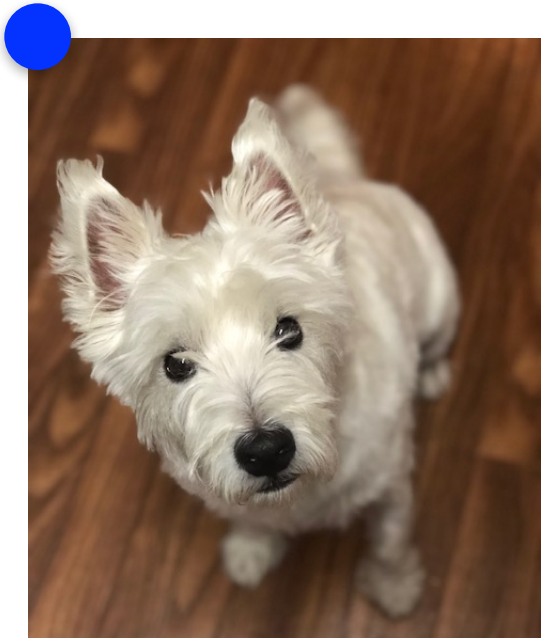
- A metric quantifies distance between any two members of a set
- Induces a similarity measure

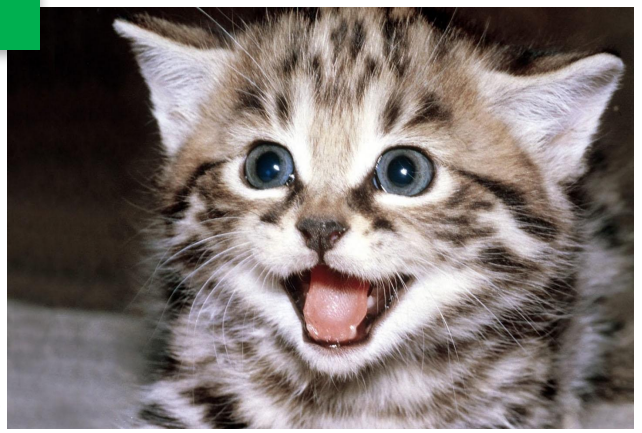
- Properties of a metric
  - Non-negativity: $f(x, y) \geq 0$
  - Identity: $f(x, y) = 0$ if and only if $x = y$
  - Symmetry: $f(x, y) = f(y, x)$
  - Triangle inequality: $f(x, y) + f(y, z) \geq f(x, z)$

- Sometimes, the choice of metric is trivial
- Other times, it is not immediately clear how to define a metric
  - Find the two faces that are closest to each other

Metric: distances in feature space

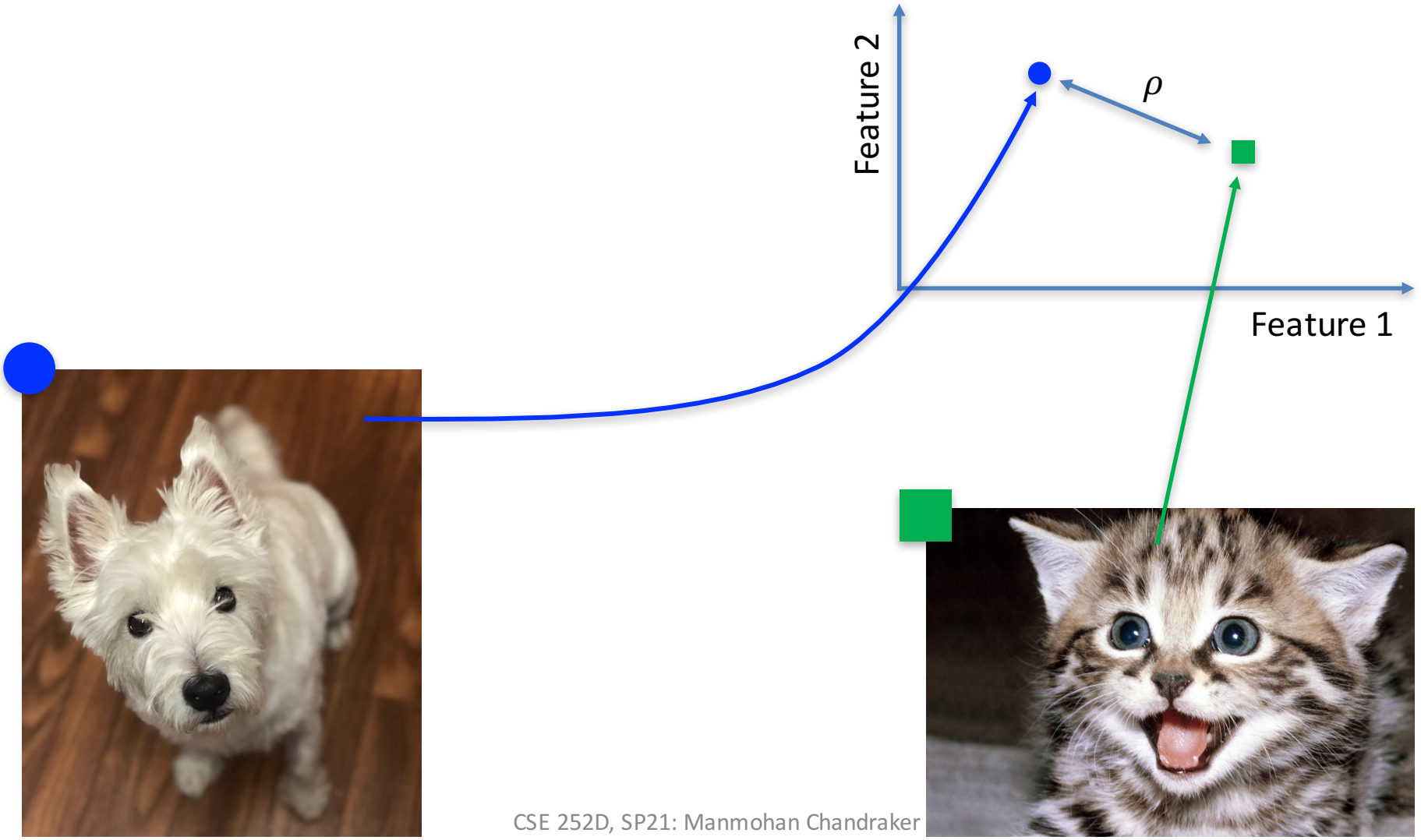
- Training data composed of (image, label) pairs
- (Image_1, dog), (Image_2, cat), (Image_3, dog), ...
- Classify test image as dog or cat





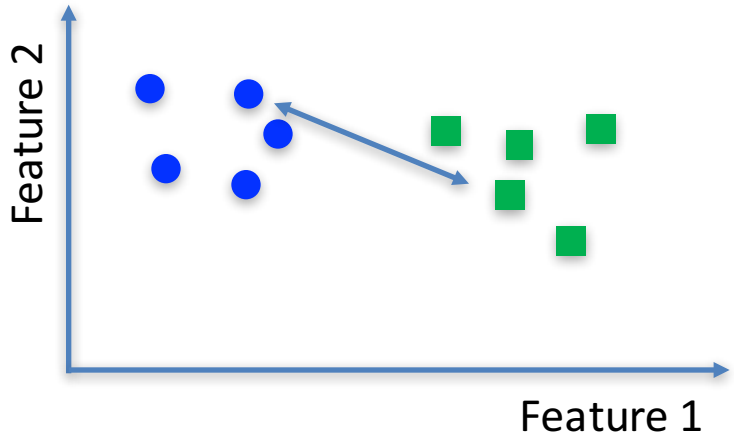
Metric: distances in feature space

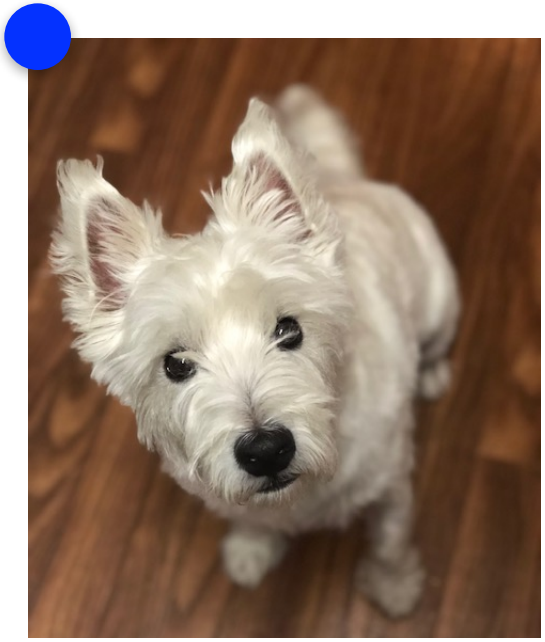
- Represent images as features, can find distance between features



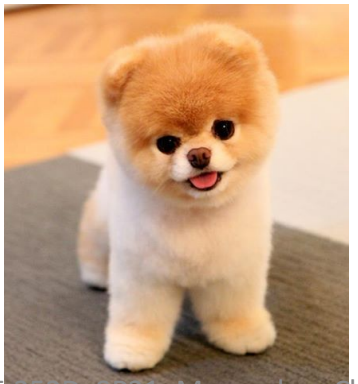
Metric: distances in feature space

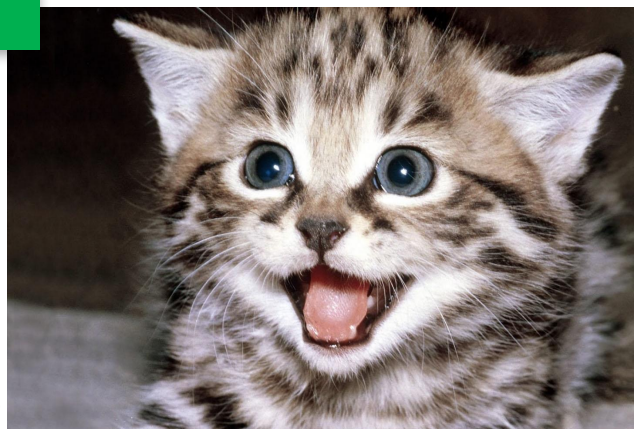
- Represent images as features, can find distance between features
- Map new image to feature space, find distances





??





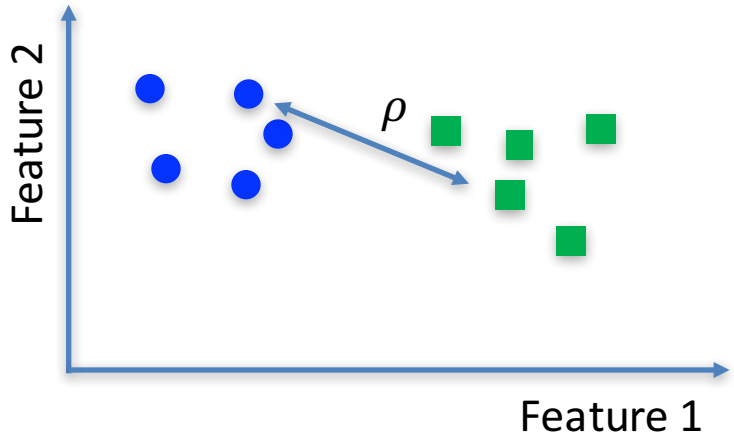
Metric: distances in feature space

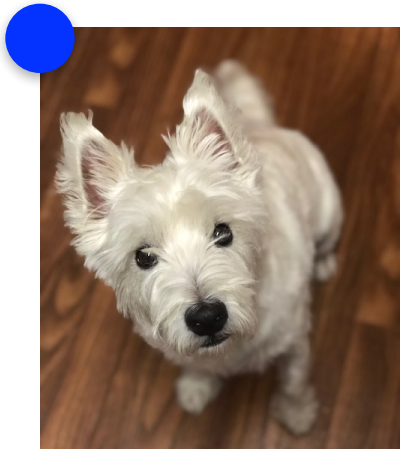
- Represent images as features, can find distance between features
- Map new image to feature space, find distances

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

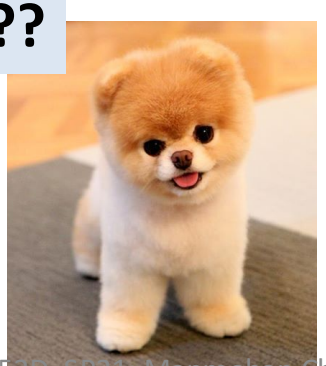
$$= \sqrt{\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}\right) \cdot \left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}\right)}$$

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



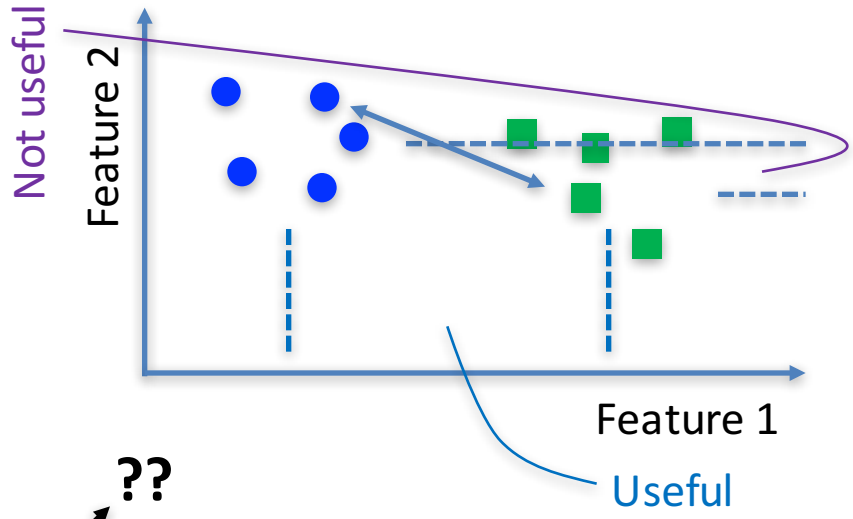


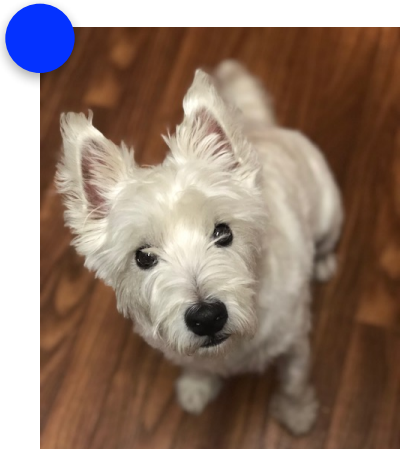
??



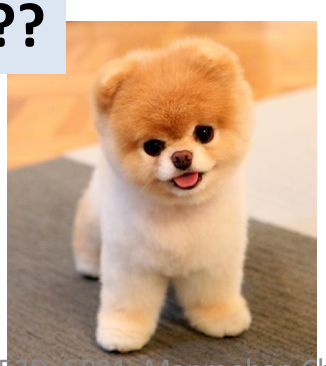
Metric: distances in feature space

- Represent images as features, can find distance between features
- Map new image to feature space, find distances
- Not all dimensions are equally useful!



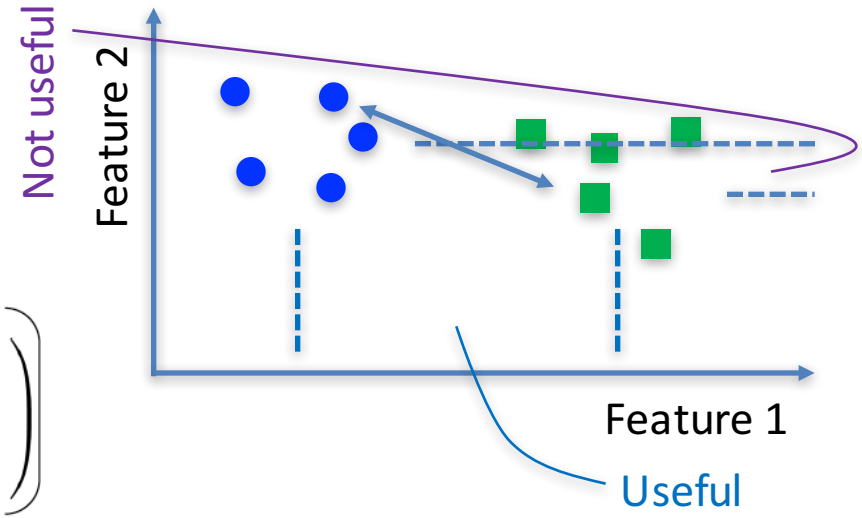


??



Metric: distances in feature space

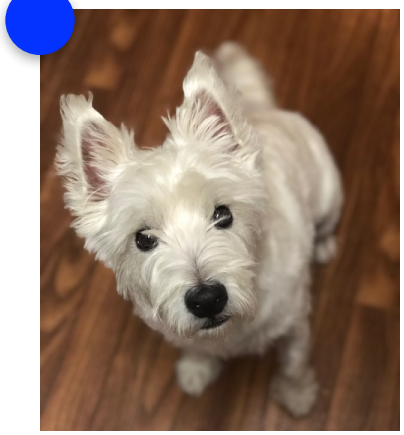
- Represent images as features, can find distance between features
- Map new image to feature space, find distances
- Not all dimensions are equally useful!
- ***Idea:*** *change how we measure distance*

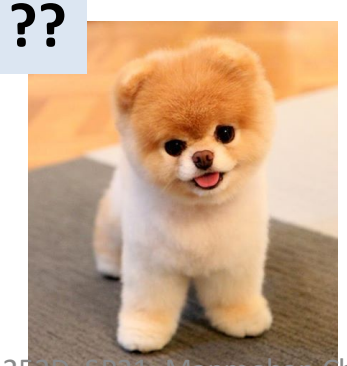


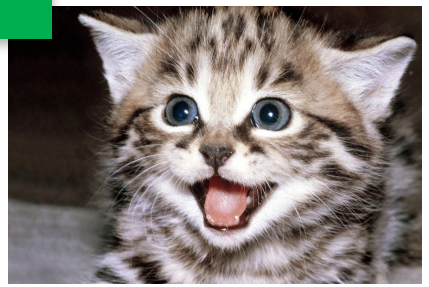
$$\rho(x,y) = \sqrt{w_1(x_1 - y_1) + w_2(x_2 - y_2)}$$

$$= \sqrt{W\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix} \cdot W\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}}$$

$$= \sqrt{[W(x-y)]^\mathsf{T}[W(x-y)]} = \sqrt{(x-y)^\mathsf{T}[W^\mathsf{T}W](x-y)} = \sqrt{(x-y)^\mathsf{T}M(x-y)}$$







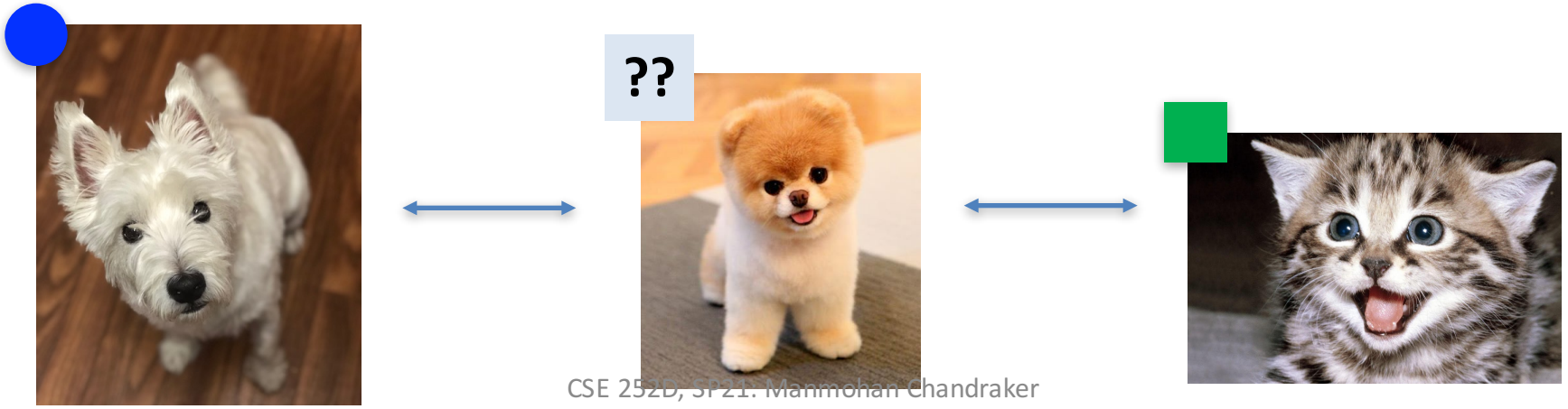
Metric learning

- Given data, learn a metric *M* that helps prediction using a distance function

$$\rho_M(x, y) = \sqrt{(x-y)^\top M (x-y)}$$

- **Goal:** Find a metric *M* such that
  - Samples from same class (positives) have low ρ_M
  - Samples from different classes (negatives) have high ρ_M

- **Approach:** Pose as an optimization problem



Metric learning

- Given data, learn a metric *M* that helps prediction using a distance function

$$\rho_M(x, y) = \sqrt{(x - y)^\top M (x - y)}$$

- Create two sets, similar pairs *S* and dissimilar pairs *D*

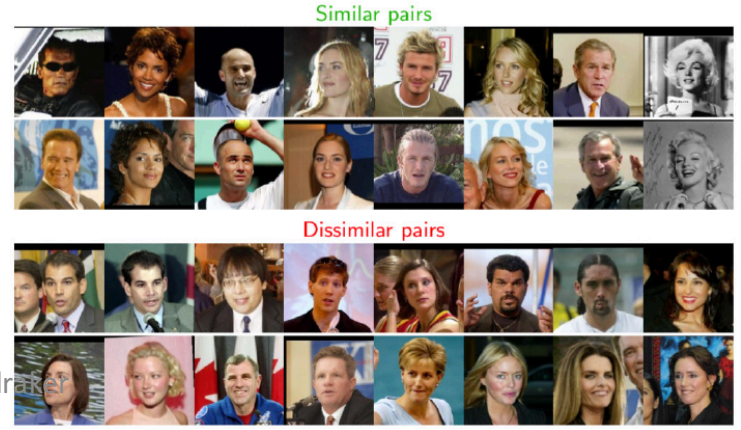
Metric learning

- Given data, learn a metric *M* that helps prediction using a distance function

$$\rho_M(x, y) = \sqrt{(x - y)^\top M (x - y)}$$

- Create two sets, similar pairs *S* and dissimilar pairs *D*
- Find *M* such that

$$\rho_M(x, x') \quad \text{large, for } (x, x') \in D$$
$$\rho_M(x, x') \quad \text{small, for } (x, x') \in S$$



Metric learning

- Given data, learn a metric *M* that helps prediction using a distance function

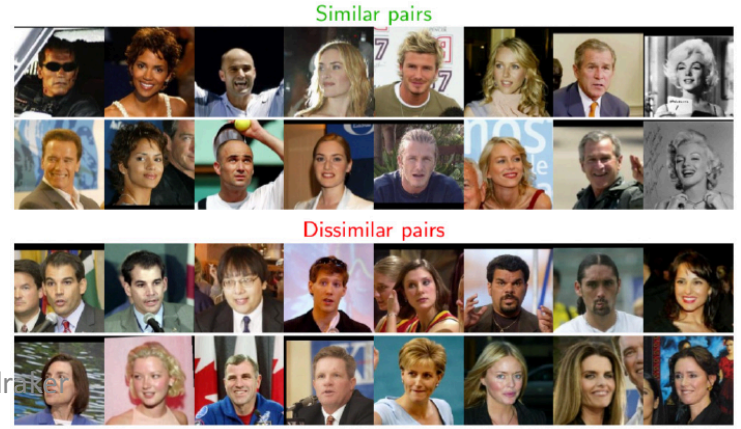
$$\rho_M(x, y) = \sqrt{(x - y)^\top M (x - y)}$$

- Create two sets, similar pairs *S* and dissimilar pairs *D*
- Find *M* such that

$$\rho_M(x, x') \quad \text{large, for } (x, x') \in D$$
$$\rho_M(x, x') \quad \text{small, for } (x, x') \in S$$

- Minimize an objective function with respect to *M*:

$$\Psi(M) = \lambda \sum_{(x,x') \in S} \rho_M^2(x, x') - (1 - \lambda) \sum_{(x,x') \in D} \rho_M^2(x, x')$$



Metric learning

- Given data, learn a metric *M* that helps prediction using a distance function

$$\rho_M(x, y) = \sqrt{(x - y)^\top M (x - y)}$$

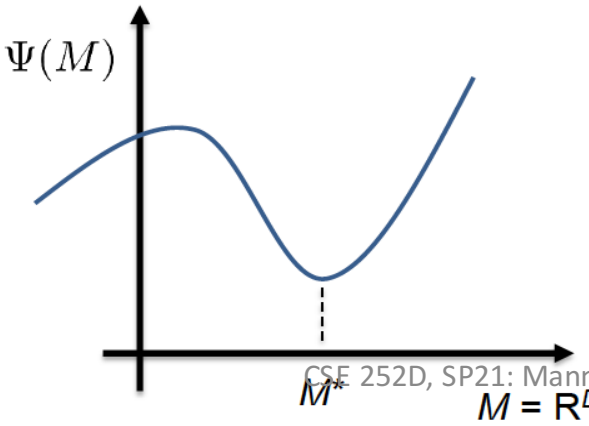
- Create two sets, similar pairs *S* and dissimilar pairs *D*
- Find *M* such that

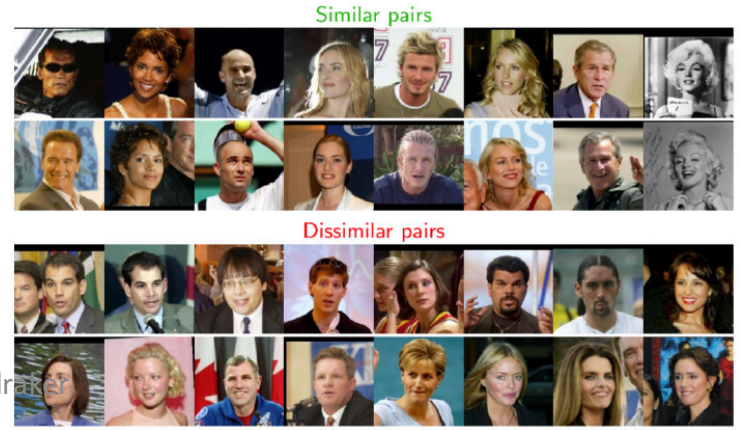
$$\rho_M(x, x') \quad \text{large, for } (x, x') \in D$$
$$\rho_M(x, x') \quad \text{small, for } (x, x') \in S$$

- Minimize an objective function with respect to *M*:

$$\Psi(M) = \lambda \sum_{(x,x') \in S} \rho_M^2(x, x') - (1 - \lambda) \sum_{(x,x') \in D} \rho_M^2(x, x')$$

- An optimization problem: find stationary points!





Metric learning

- Given data, learn a metric *M* that helps prediction using a distance function

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

- Create two sets, similar pairs *S* and dissimilar pairs *D*
- Find *M* such that

$$\rho_M(x, x') \quad \text{large, for } (x, x') \in D$$
$$\rho_M(x, x') \quad \text{small, for } (x, x') \in S$$

- **Margin maximization** variant:

$$\text{maximize}_M \quad \sum_{(x,x') \in D} \rho_M^2(x, x')$$

$$\text{constraint:} \quad \sum_{(x,x') \in S} \rho_M^2(x, x') \leq 1$$

$$M \in \text{PSD}$$

Recall:

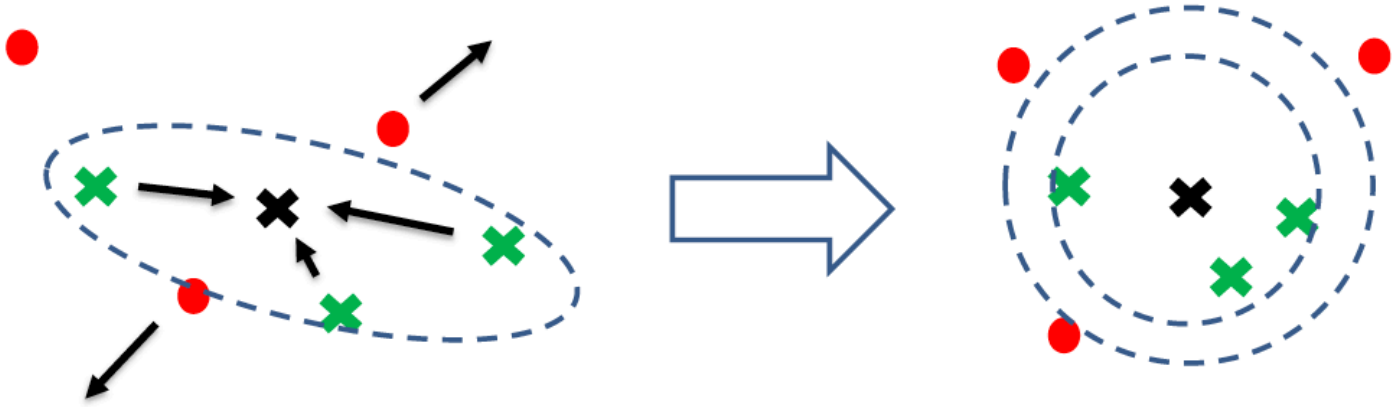
$$M = W^\mathsf{T} W$$

- Convex optimization (semidefinite program), "efficiently" solvable!

Large margin nearest neighbors

- **Target neighbors** of point x_i: points x_j with similar labels
- **Impostors** are points x_l closer to x_i than x_j, but with different labels
- **Goal:** *pull* targets closer, *push* impostors farther away

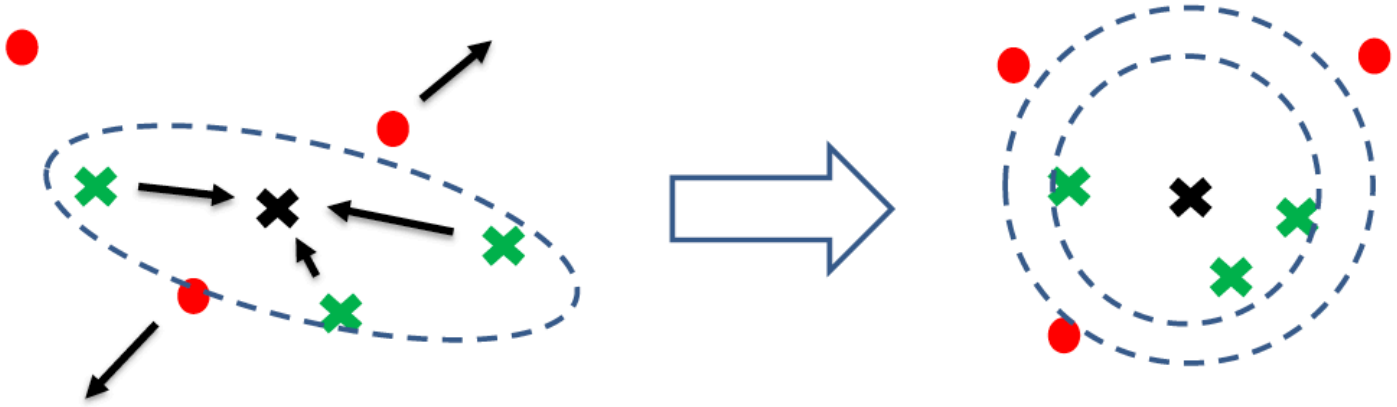
$$\Psi(M) = \lambda\ \Psi_{\text{pull}}(M)\ + (1-\lambda)\ \Psi_{\text{push}}(M)$$



Large margin nearest neighbors

- **Target neighbors** of point x_i: points x_j with similar labels
- **Impostors** are points x_l closer to x_i than x_j, but with different labels
- **Goal:** *pull* targets closer, *push* impostors farther away

$$\Psi(M) = \lambda\ \Psi_{\text{pull}}(M)\ + (1-\lambda)\ \Psi_{\text{push}}(M)$$

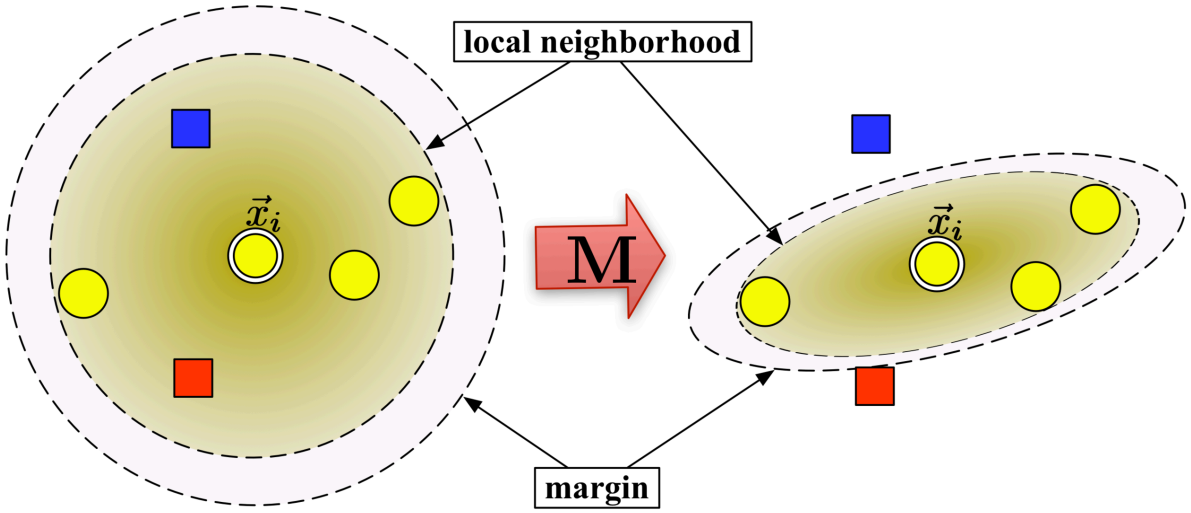


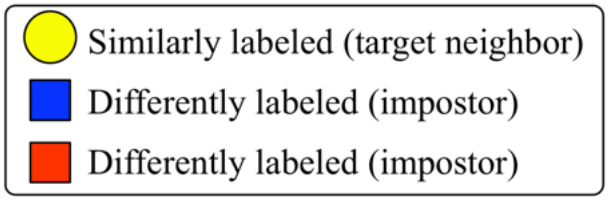
- Given point *i*, similar **<u>neighboring</u>** points *j*, impostor points *l* defined by *i* and *j*:

$$\Psi_{\text{pull}}(M) = \sum_{i,j(i)} \rho_M^2(x_i, x_j)$$

$$\Psi_{\text{push}}(M) = \sum_{i,j(i),l(i,j)} 1 + \rho_M^2(x_i, x_j) - \rho_M^2(x_i, x_l)$$

Large margin nearest neighbors





- Pulling target neighbors together

$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{i,j \rightsquigarrow i} ||\mathbf{L}(\vec{x}_i - \vec{x}_j)||^2.$$

Local constraints: directly improve neighbor quality

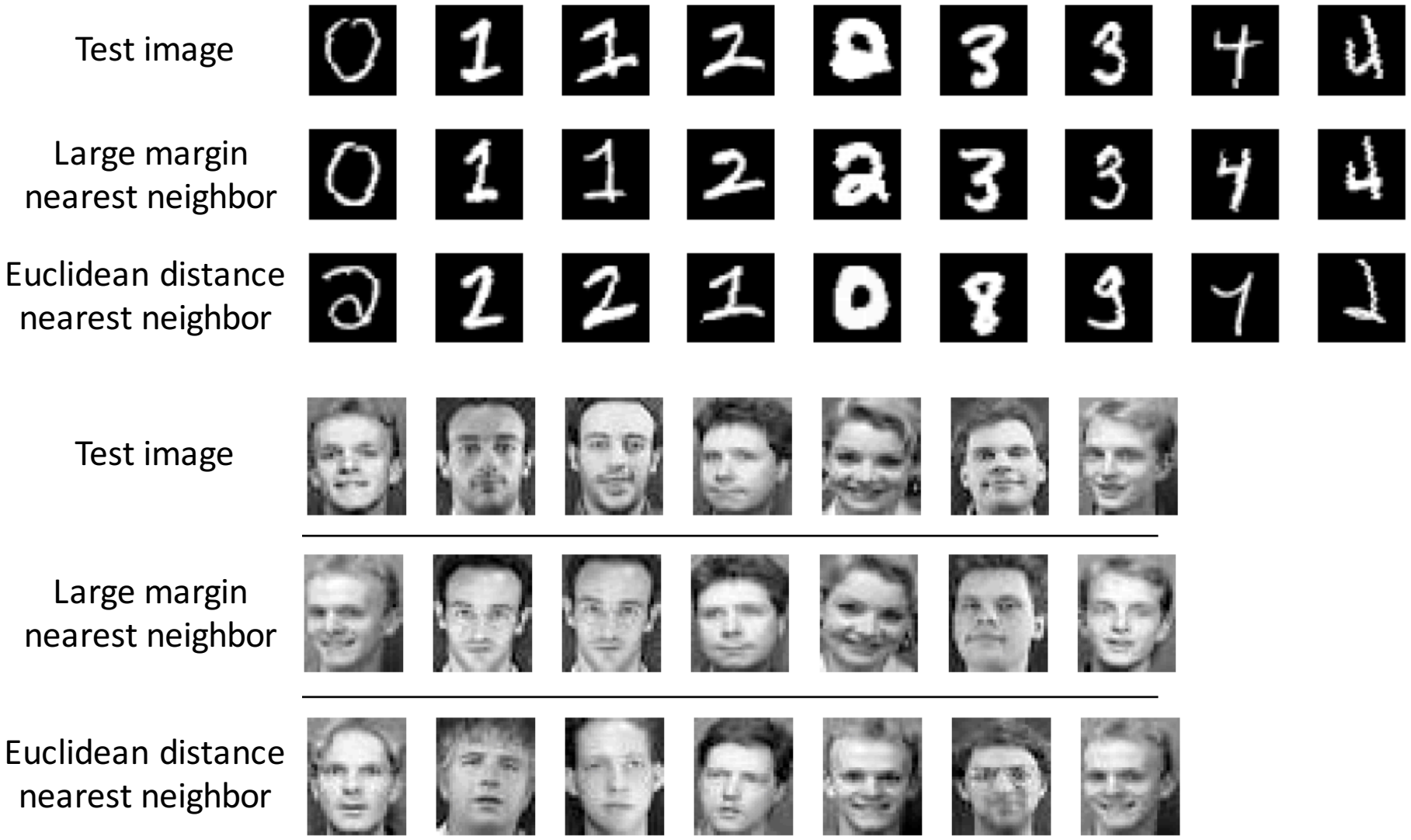
- Pushing impostors away

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - y_{il})[1 + ||\mathbf{L}(\vec{x}_i - \vec{x}_j)||^2 - ||\mathbf{L}(\vec{x}_i - \vec{x}_l)||^2]_+$$

- Convex combination

$$\varepsilon(\mathbf{L}) = \mu\varepsilon_{\text{pull}}(\mathbf{L}) + (1 - \mu)\varepsilon_{\text{push}}(\mathbf{L}), \quad \mu \in [0, 1]$$

Large margin nearest neighbors



[Weinberger et al., 2009]

General recipe for metric learning

1. Pick a parametric distance or similarity function
   - Say, a distance $D_M(x, x')$ function parameterized by M
2. Collect similarity judgments on data pairs/triplets
   - $\mathcal{S} = \{(x_i, x_j) : x_i \text{ and } x_j \text{ are similar}\}$
   - $\mathcal{D} = \{(x_i, x_j) : x_i \text{ and } x_j \text{ are dissimilar}\}$
   - $\mathcal{R} = \{(x_i, x_j, x_k) : x_i \text{ is more similar to } x_j \text{ than to } x_k\}$
3. Estimate parameters s.t. metric best agrees with judgments
   - Solve an optimization problem of the form

$$\hat{M} = \arg\min_M \left[\underbrace{\ell(M, \mathcal{S}, \mathcal{D}, \mathcal{R})}_{\text{loss function}} + \underbrace{\lambda reg(M)}_{\text{regularization}} \right]$$

Siamese CNNs for distance metric learning

- Advantage of LMNN: convex problem!
- Disadvantages: feature representation is learned independent of the metric
- Advantage of CNNs: can optimize features conditioned on similarity measure

Siamese CNNs for distance metric learning

- Advantage of LMNN: convex problem!
- Disadvantages: feature representation is learned independent of the metric
- Advantage of CNNs: can optimize features conditioned on similarity measure

- **Input**: A pair of input signatures.
- **Output (Target)**: A label, 0 for **similar,** 1 **else.**

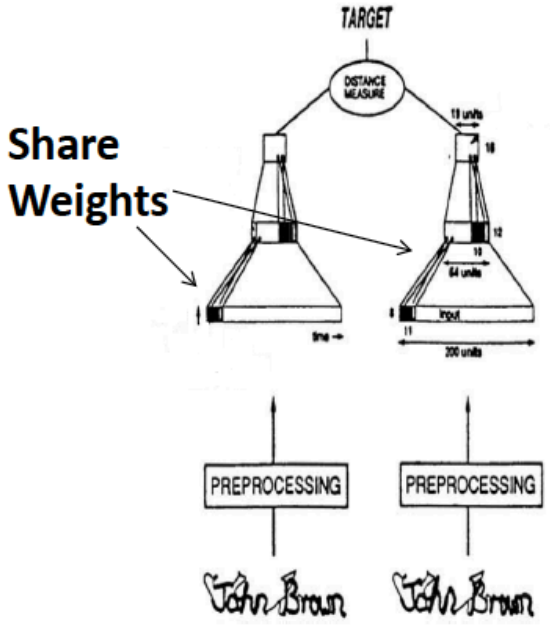
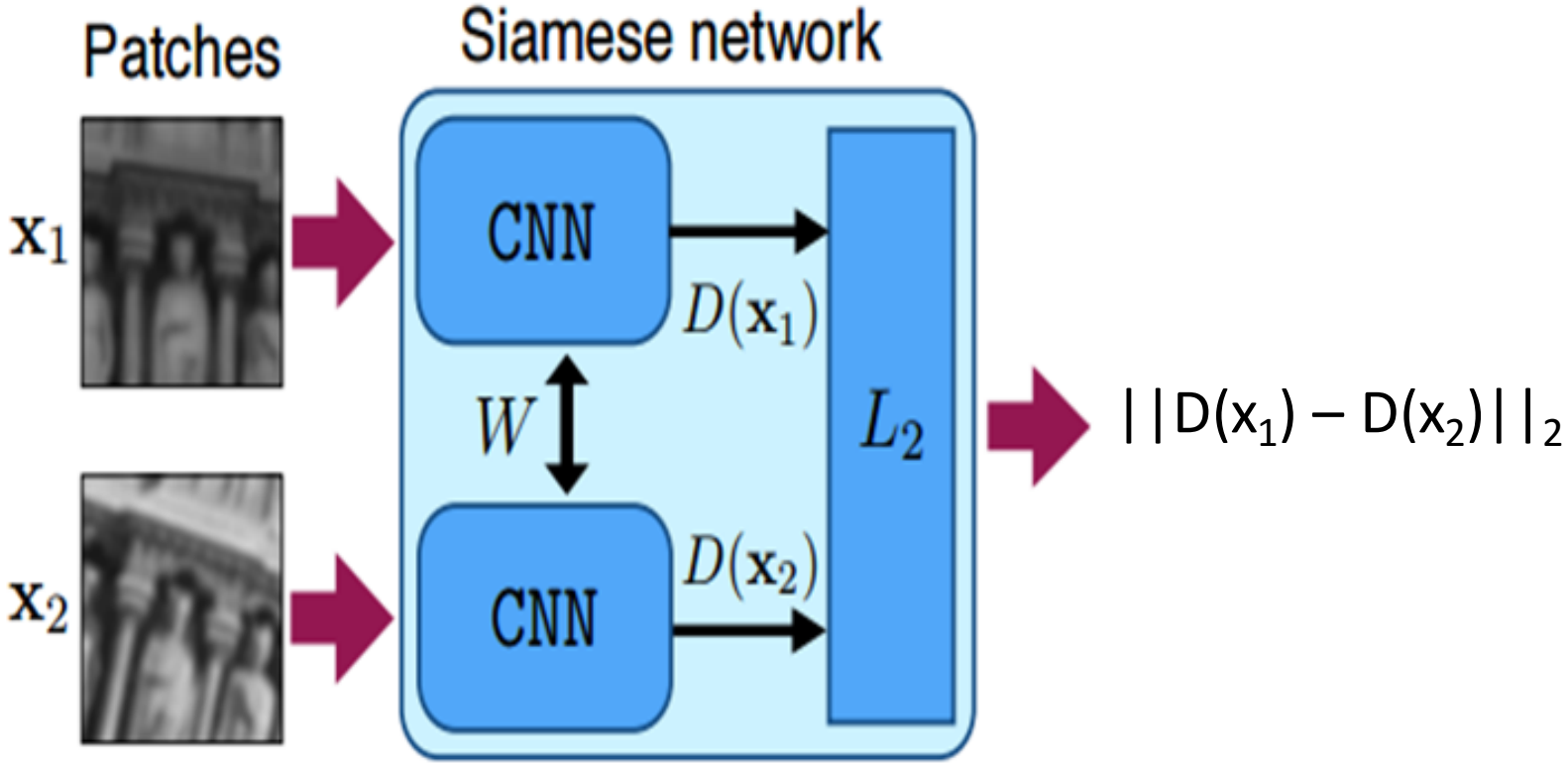


Image Source: Google

Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E. and Shah, R., 1993. Signature Verification Using A "Siamese" Time Delay Neural Network. *IJPRAI*, *7*(4), pp.669-688.

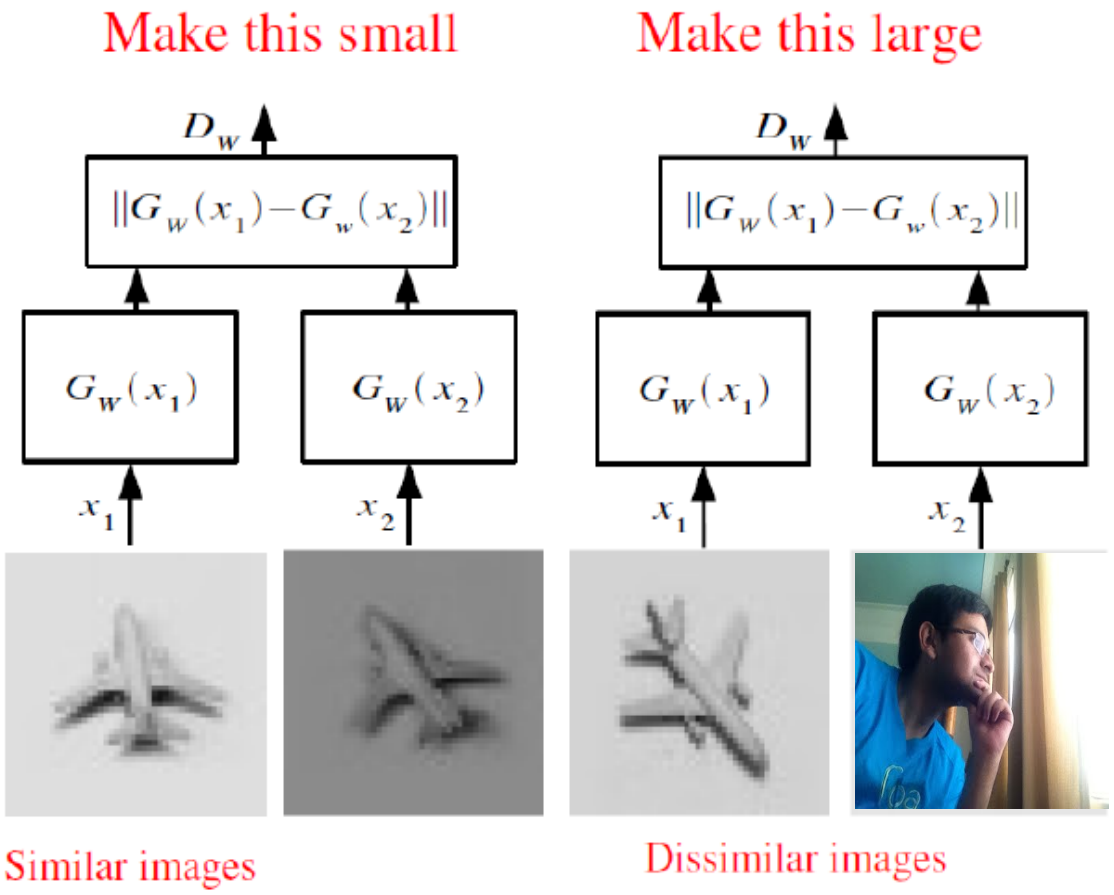
Siamese CNN for patch similarity



Issue with using this loss for training?

Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P. and Moreno-Noguer, F., 2015. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 118-126).

Loss function for Siamese CNNs



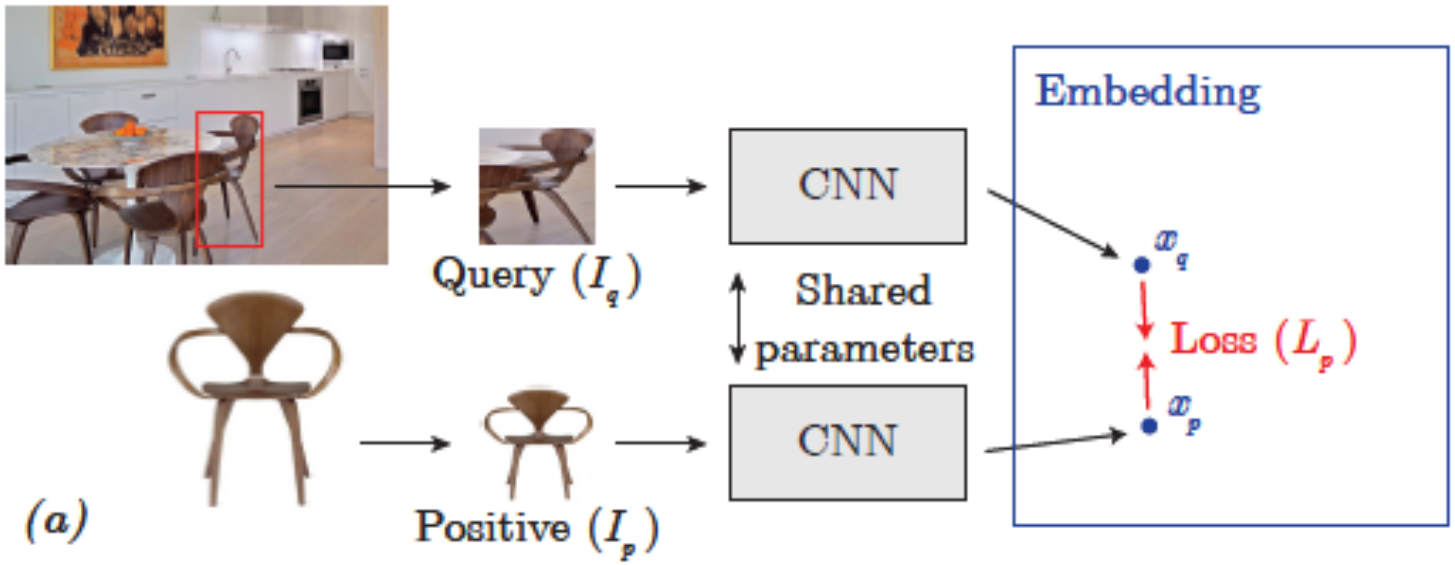
The final loss is defined as :

L = ∑loss of positive pairs + ∑ loss of negative pairs

Loss function for Siamese CNNs

We can use different loss functions for the two types of input pairs.

- Typical positive pair (x_p, x_q) loss: $L(x_p, x_q) = ||x_p - x_q||^2$
  (Euclidean Loss)

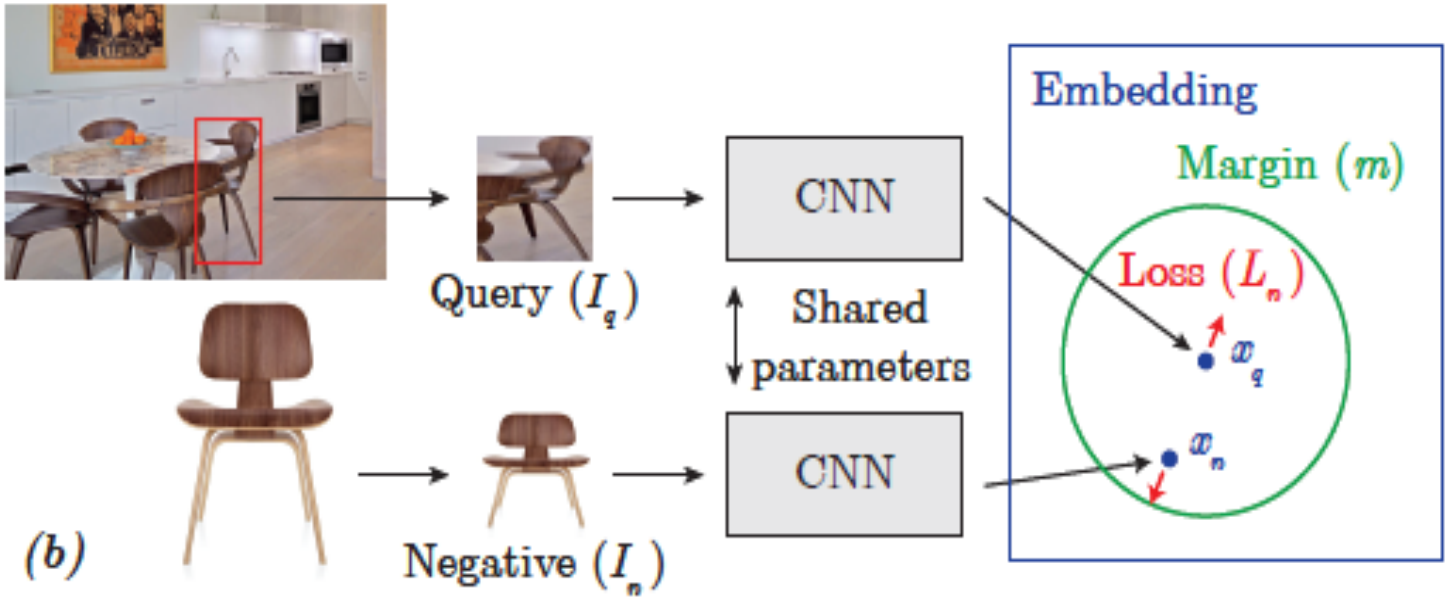


Bell, S. and Bala, K., 2015. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, *34*(4), p.98.

Loss function for Siamese CNNs

- Typical negative pair (x_n, x_q) loss :

  $L(x_n, x_q) = \max(0, m^2 - ||x_n - x_q||^2)$ (Hinge Loss)



Bell, S. and Bala, K., 2015. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, *34*(4), p.98.

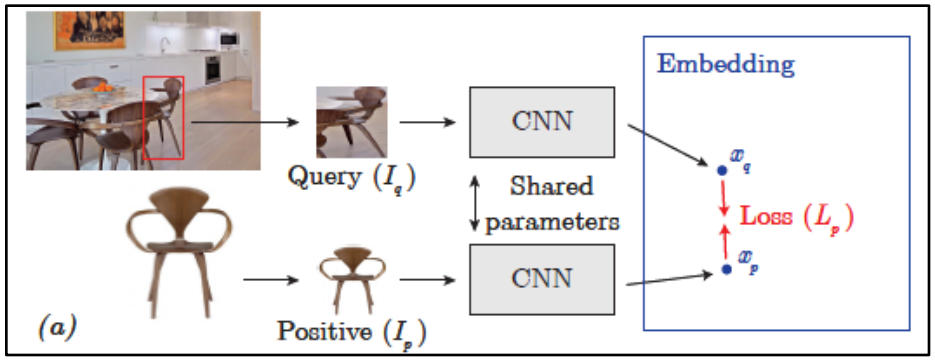
Loss function for Siamese CNNs

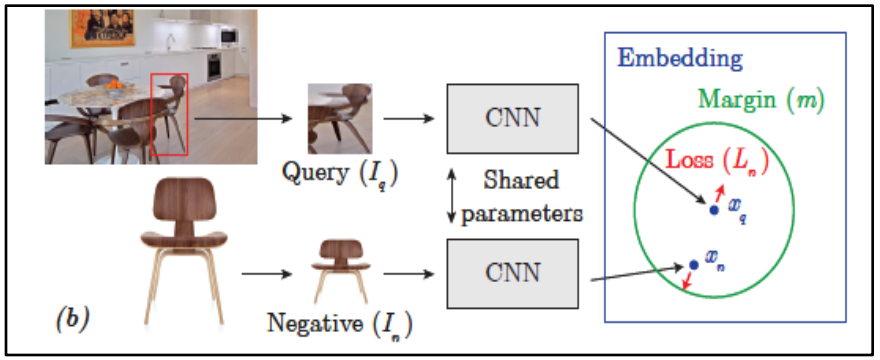
- Combined into a contrastive loss
- For pair of training examples x_1 and x_2 (with labels y_1, y_2):

$$L(x_1, x_2) = s_{12}||x_1 - x_2||^2 + (1 - s_{12}) \max(0, m^2 - ||x_1 - x_2||^2),$$

$s_{12} = 1$ when $y_1 = y_2$,

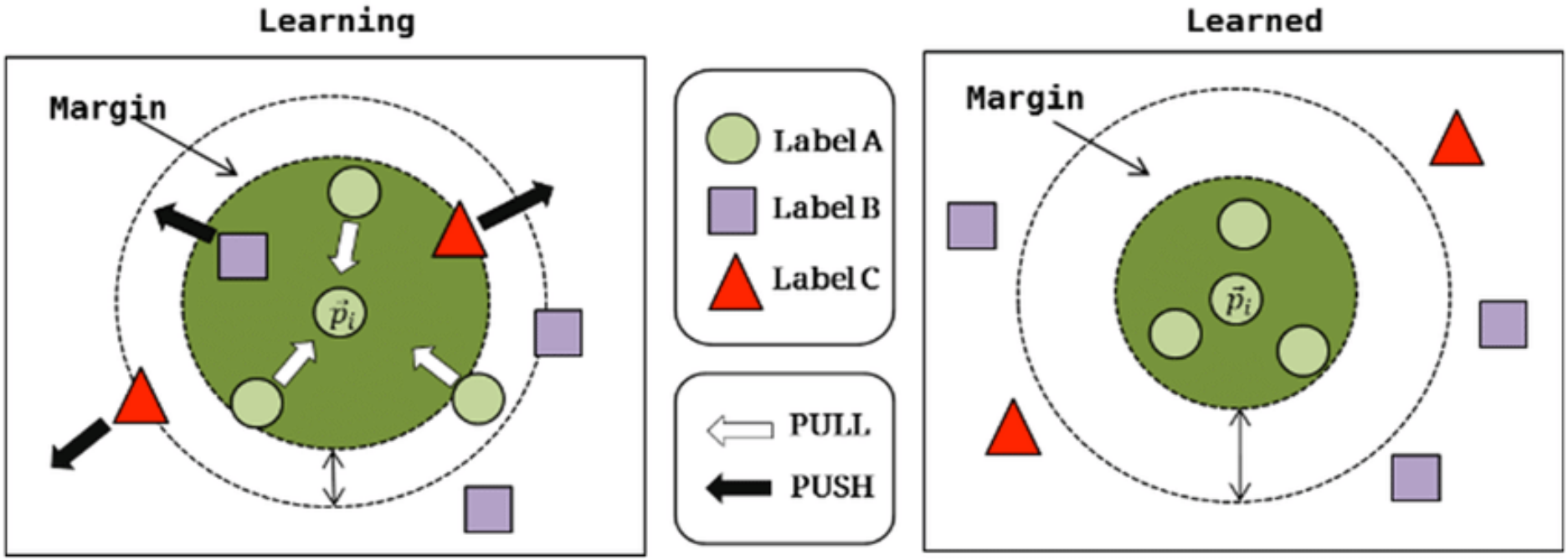
otherwise $s_{12} = 0$.





Triplet Loss for Metric Learning

- Goal: get positives close together, negatives far away



Triplet Loss for Metric Learning

- At each training iteration, sample a set of triplets

$$\mathcal{T} = (\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n) \qquad y_a = y_p \neq y_n$$

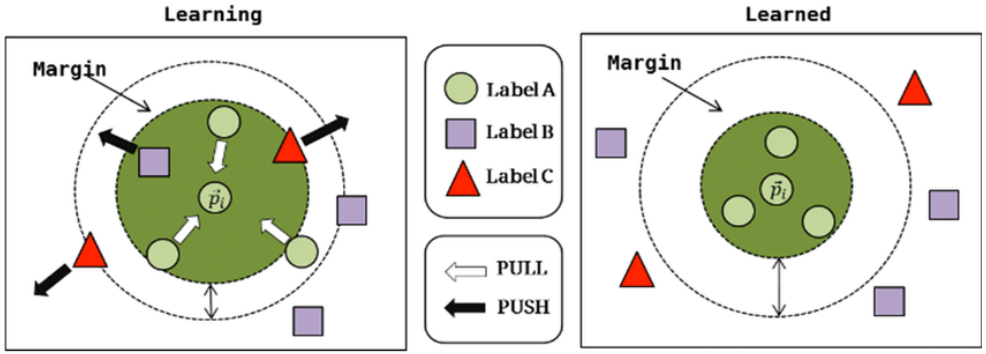
Anchor Positive Negative

- Goal: push negative a margin further from anchor than positive

$$\|\mathbf{x}_a - \mathbf{x}_p\|^2 + m \leq \|\mathbf{x}_a - \mathbf{x}_n\|^2$$

- Triplet loss:

$$l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$$

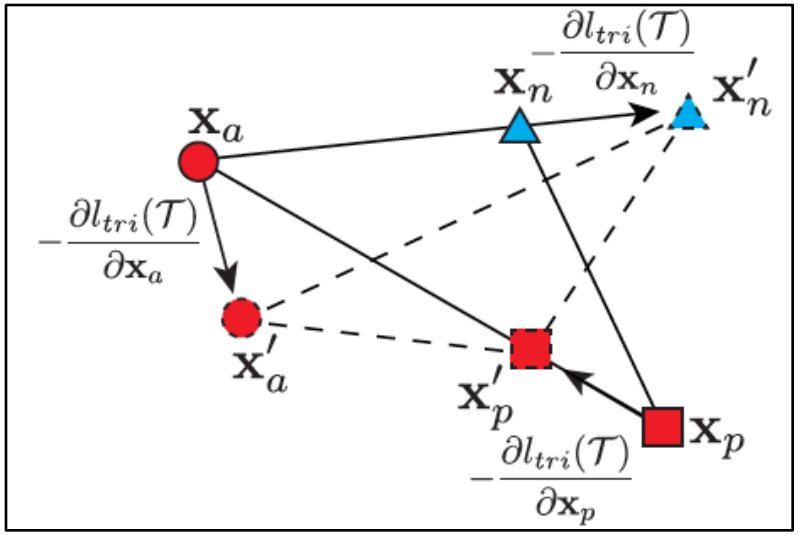


Triplet Loss for Metric Learning

- Triplet loss: $l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$

- Gradients:

$$\frac{\partial l_{tri}(\mathcal{T})}{\partial \mathbf{x}_n} = 2(\mathbf{x}_a - \mathbf{x}_n)$$
$$\frac{\partial l_{tri}(\mathcal{T})}{\partial \mathbf{x}_p} = 2(\mathbf{x}_p - \mathbf{x}_a)$$
$$\frac{\partial l_{tri}(\mathcal{T})}{\partial \mathbf{x}_a} = 2(\mathbf{x}_n - \mathbf{x}_p)$$



Recall: gradient descent

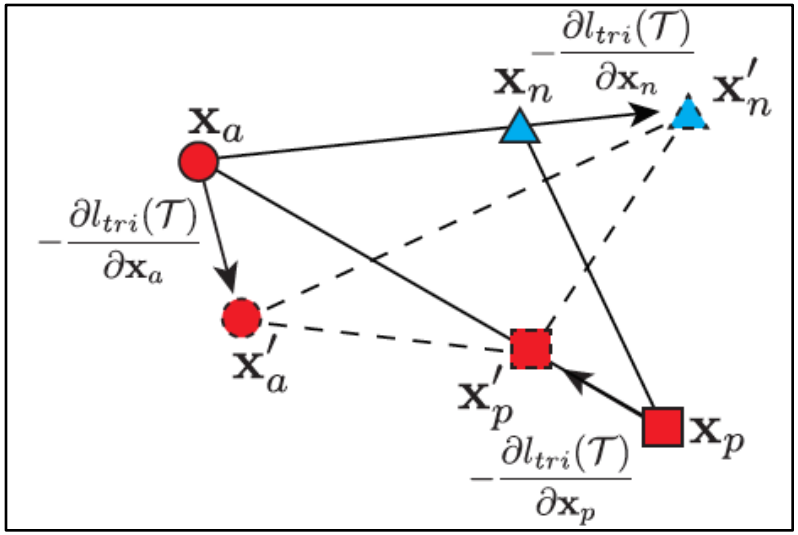
$$x_{t+1} = x_t - \eta \nabla l$$

Triplet Loss for Metric Learning

- Triplet loss: $l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$

- Gradients:

$$\frac{\partial l_{tri}(\mathcal{T})}{\partial \mathbf{x}_n} = 2(\mathbf{x}_a - \mathbf{x}_n)$$
$$\frac{\partial l_{tri}(\mathcal{T})}{\partial \mathbf{x}_p} = 2(\mathbf{x}_p - \mathbf{x}_a)$$
$$\frac{\partial l_{tri}(\mathcal{T})}{\partial \mathbf{x}_a} = 2(\mathbf{x}_n - \mathbf{x}_p)$$

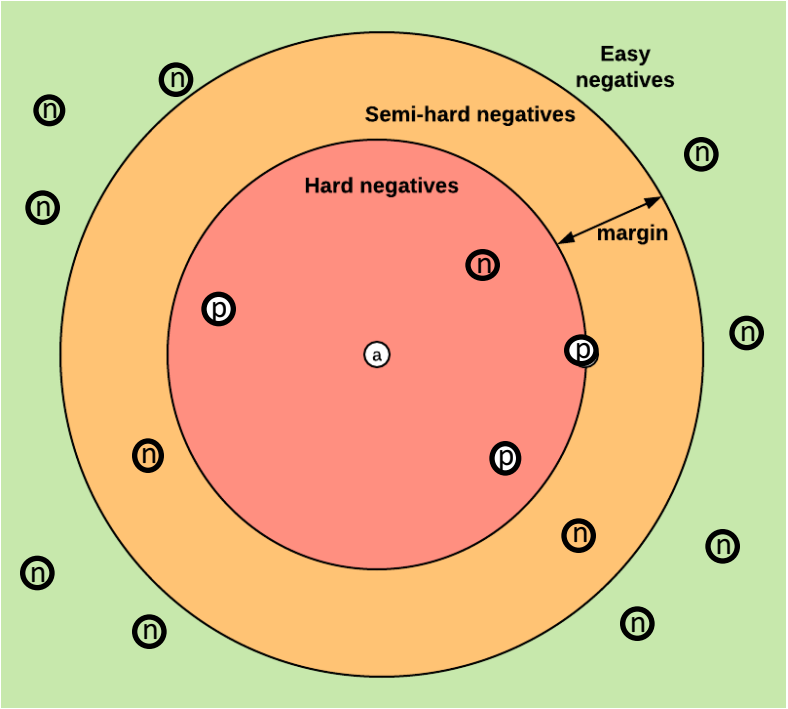


- Issues:
  - Fixed parameter m, while intra-class distances can vary
  - Inefficient to consider all triplets

Hard Negative Mining

- Triplet loss: $l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$

Loss is 0 for easy negatives



- Metric learning is driven by hard negatives
- Most negatives are easy
- Need strategy to find hard negatives among samples

Hard Negative Mining

- Easy negatives are abundant
- Decision boundary trained with easy negatives will not be “sharp”
- Select those negatives which classifier currently thinks are positives