Representation Learning

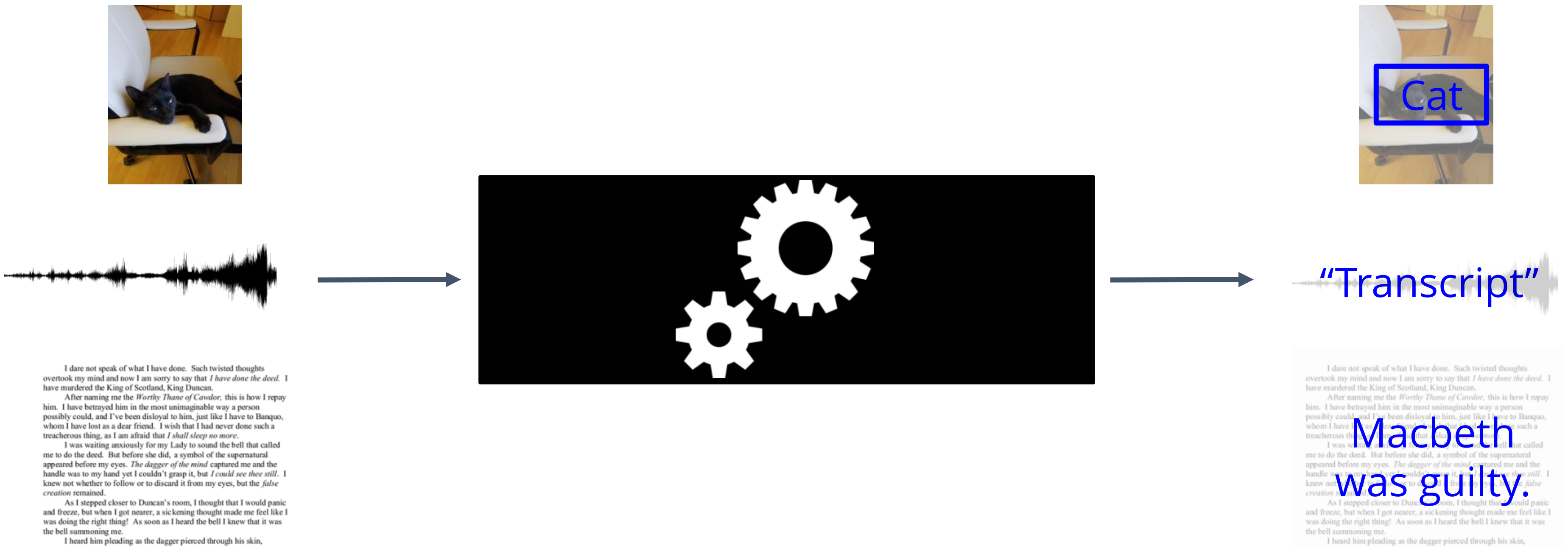
Computer Vision – Lecture 17

Further Reading

- Slides from S Savarese, A Zamir
- Slides from F Li
- Slides from A Geiger

So far: Task Learning

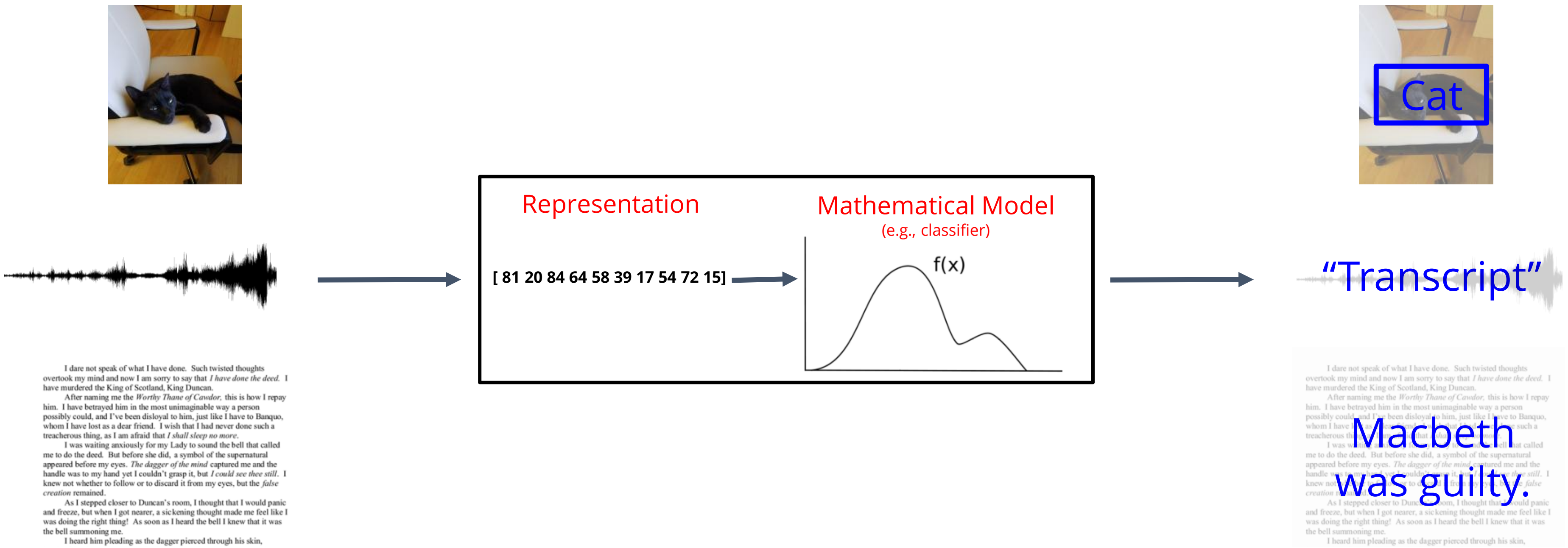
- Learn a function from input to task output.



Slide adapted from S Savarese

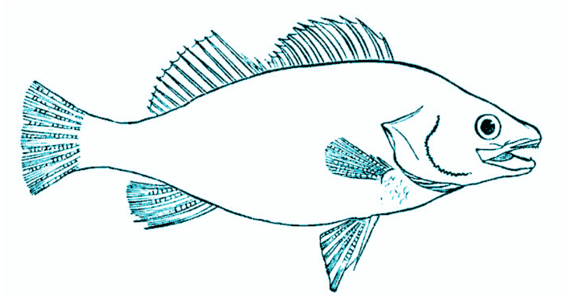
So far: Task Learning

- Learn a function from input to task output.
- Representation Learning: general representation + task head

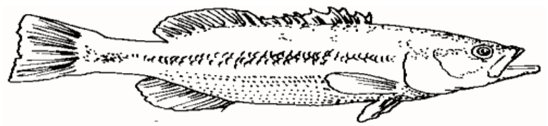


Slide adapted from S Savarese

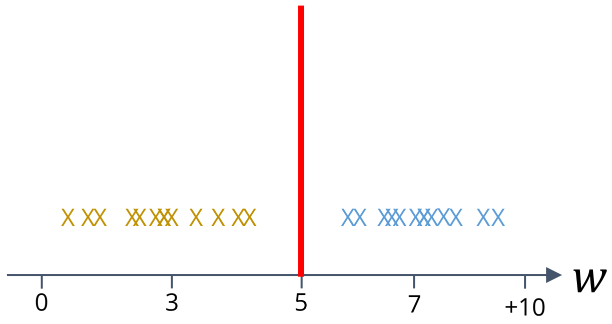
Representations

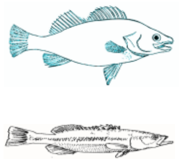


~5.4 kg



~3.6 kg





Representation

Weight (w)

Mathematical Model (Classifier)

w>5

Type A

Type B

Slide adapted from S Savarese

Representation Learning

Supervised

- Given a task, learn a representation for it.
- Representation is often constrained to task(s).
- today

Unsupervised

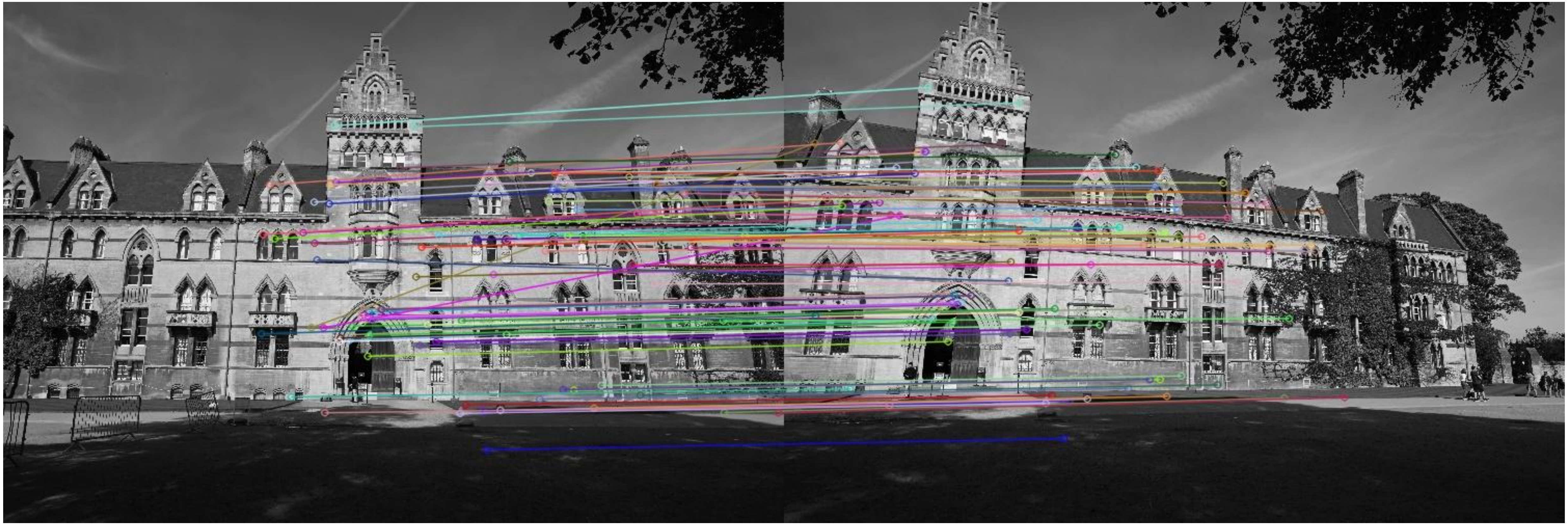
- Given only data, find a representation for it.
- Representation often does not align exactly with tasks.
- Part of Lecture 18

Handcrafting Representations

- Was the only way for a long time.
- (almost) Worked for many important applications:
  - Image Retrieval
  - Structure-from-motion
  - Face detection
  - etc.
- Why alternatives?
  - Can't quite find the discriminative signature for a problem.
  - Discriminative signature can be found, but hard to approach programmatically.
  - Too many contributing factors to the problem.
    - Fusion non-trivial. Rule-based fusion outruled.
    - Fusion of contributing factors itself a comparably complex representation problem.

Correspondences

Point correspondences estimated by a classic algorithm: SIFT



Recall: SIFT Descriptor

- Compute edge orientations and global orientation.
- Rotate all edges so that the global orientation is “up”.
- Split the local area around the keypoint into 4x4=16 regions.
- Compute edge histograms (8 directions) for each region.
- Concatenate histograms: descriptor 128 dimensional vector.

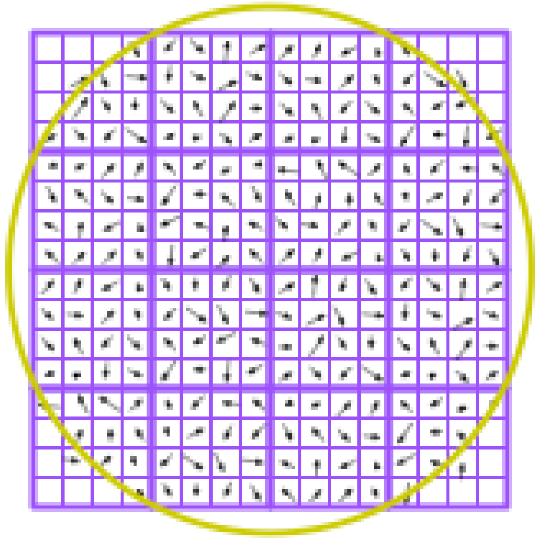
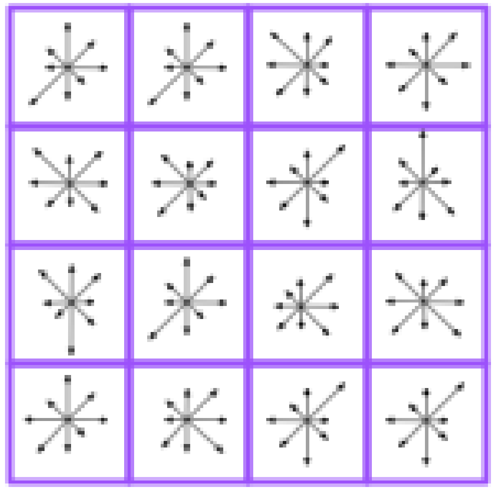


Image gradients

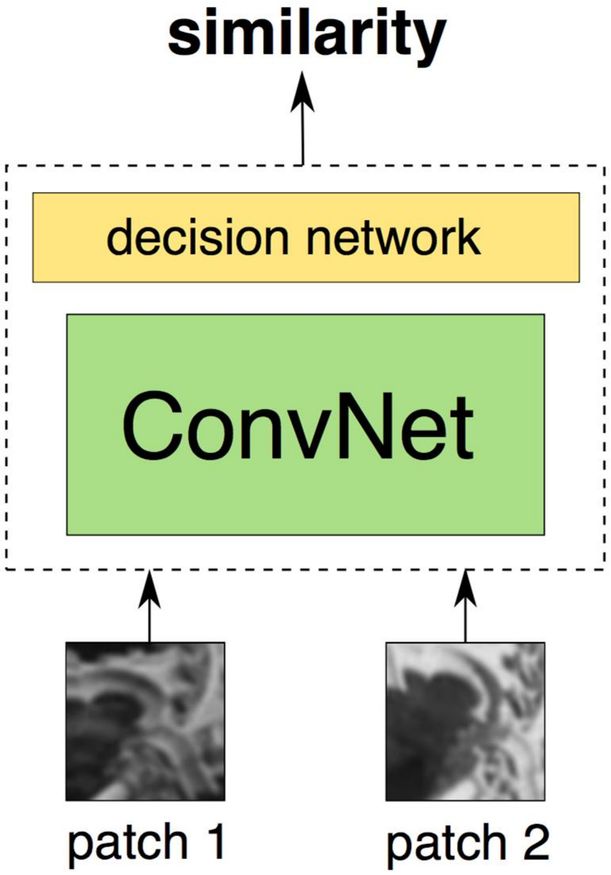


Keypoint descriptor

Learning a Keypoint Descriptor

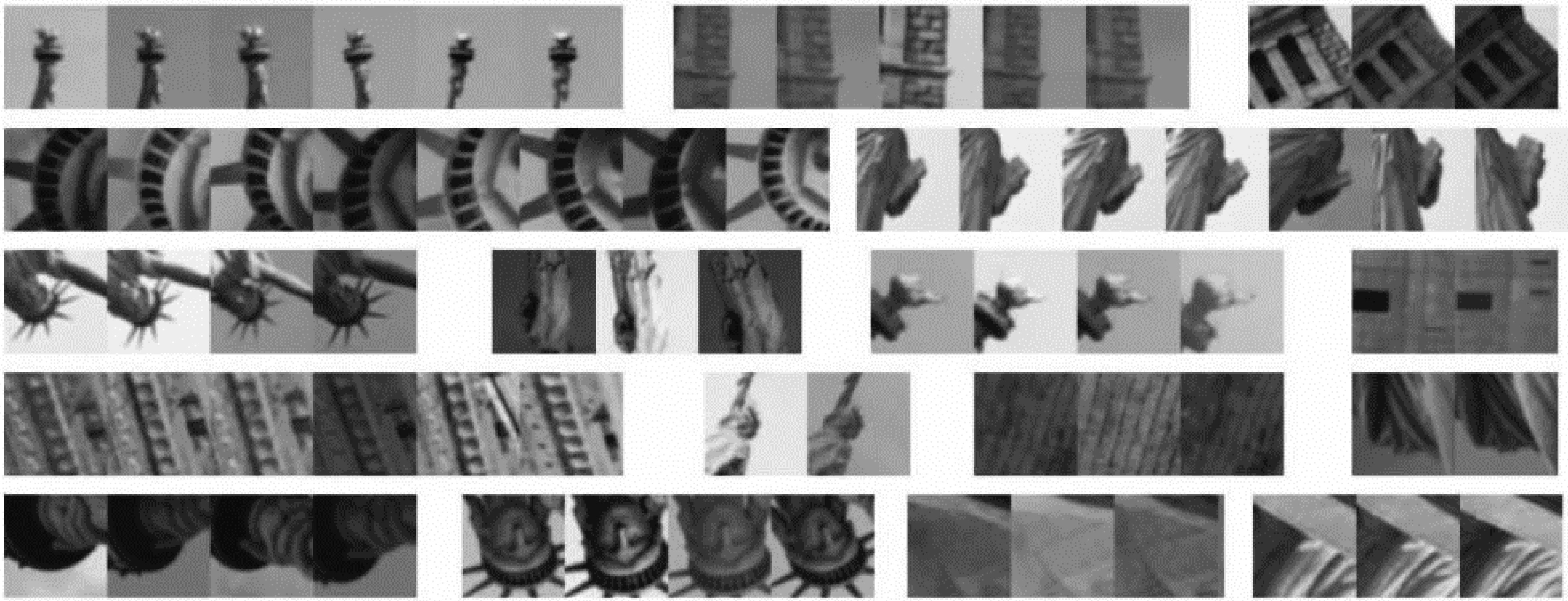
- Use a dataset with point correspondences.
- Extract patches around keypoints.
- Positives from matching keypoints.
- Negatives from random keypoints.

- Train a model to predict the similarity between two patches.



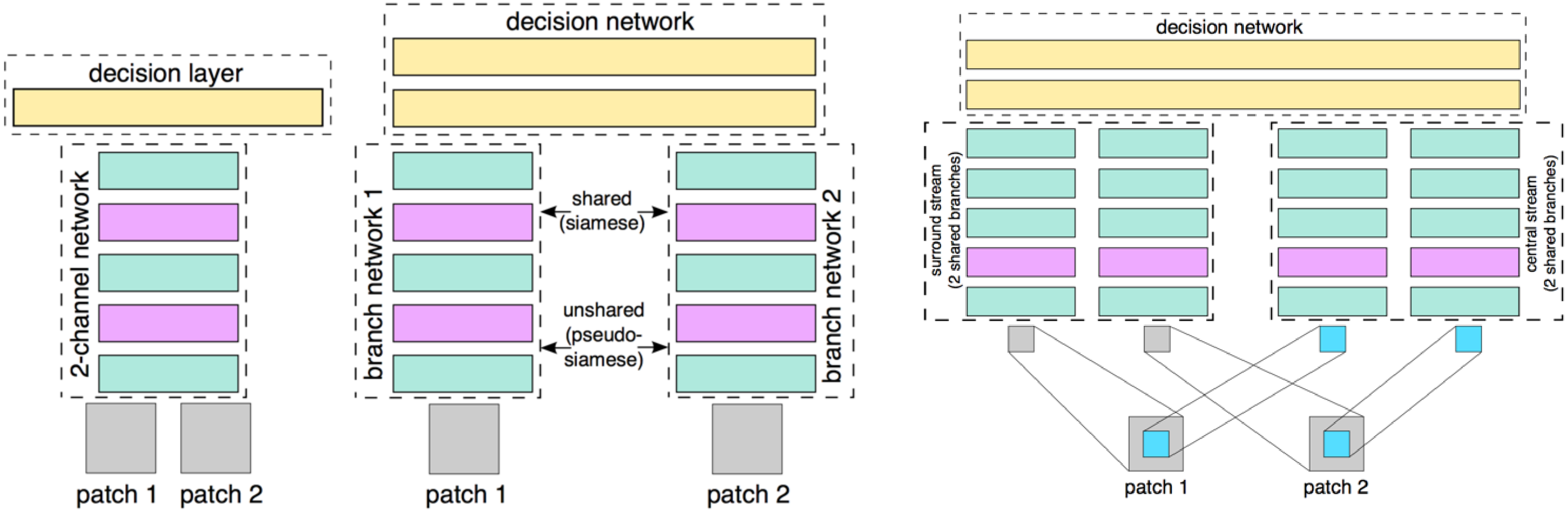
Zagoruyko & Komodakis. 2015.

Local Feature Learning Dataset



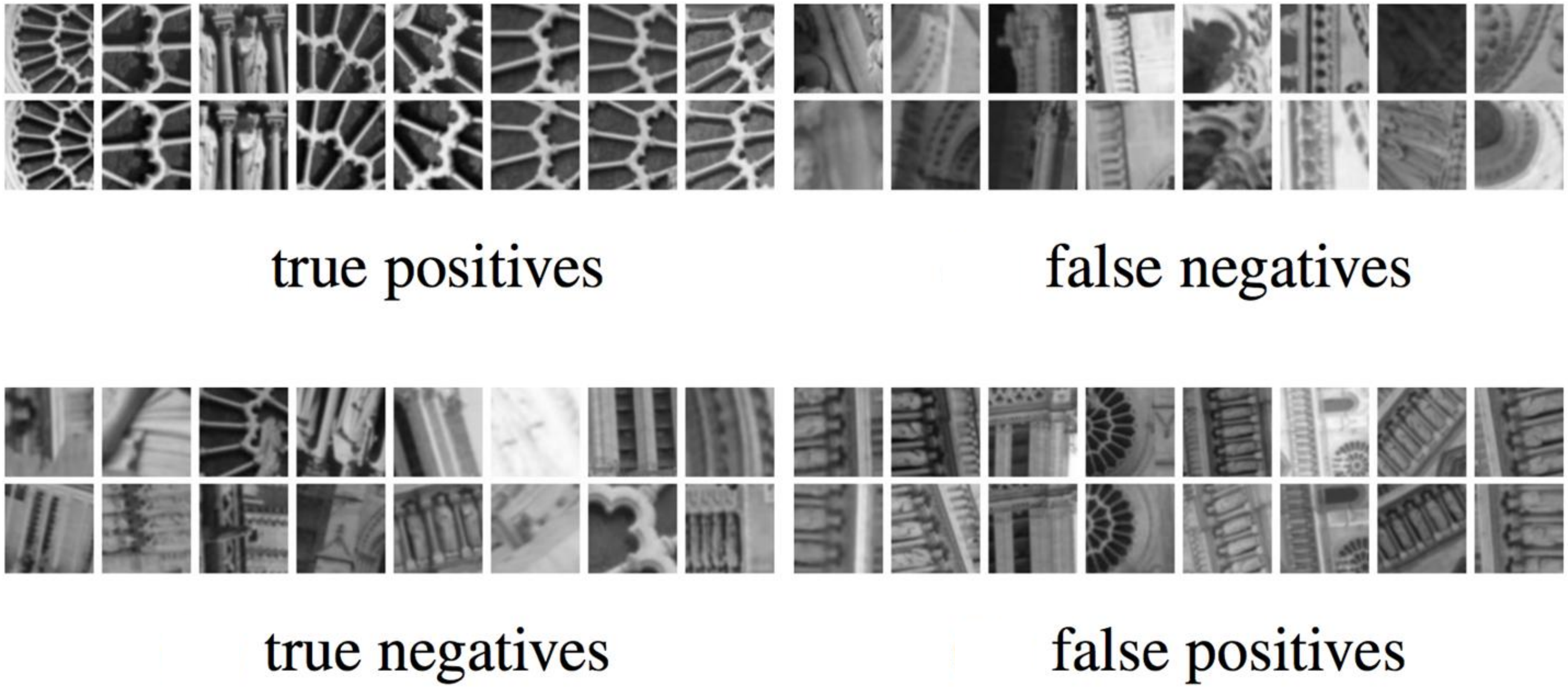
Brown, Hua, Winder, Discriminative Learning of Local Features, 2011

Low-level matching architectures



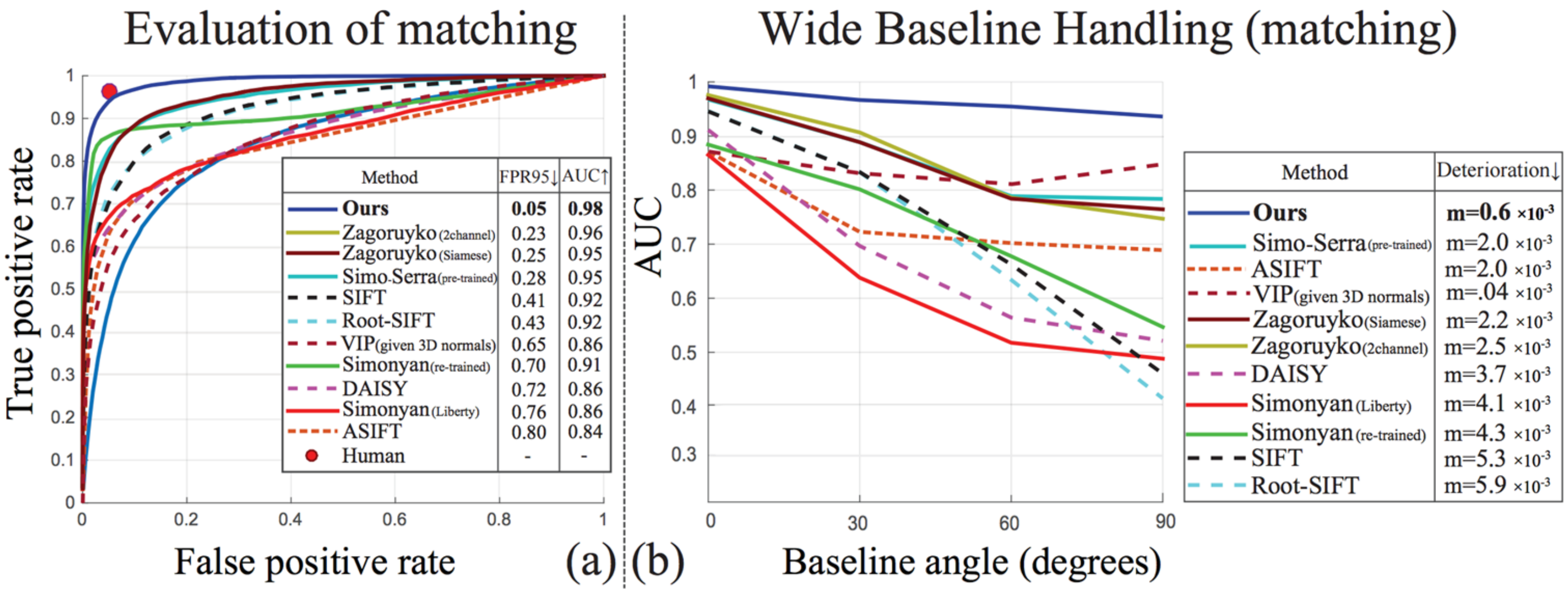
Zagoruyko & Komodakis. 2015.

Low-level matching: qualitative results



Zagoruyko & Komodakis. 2015.

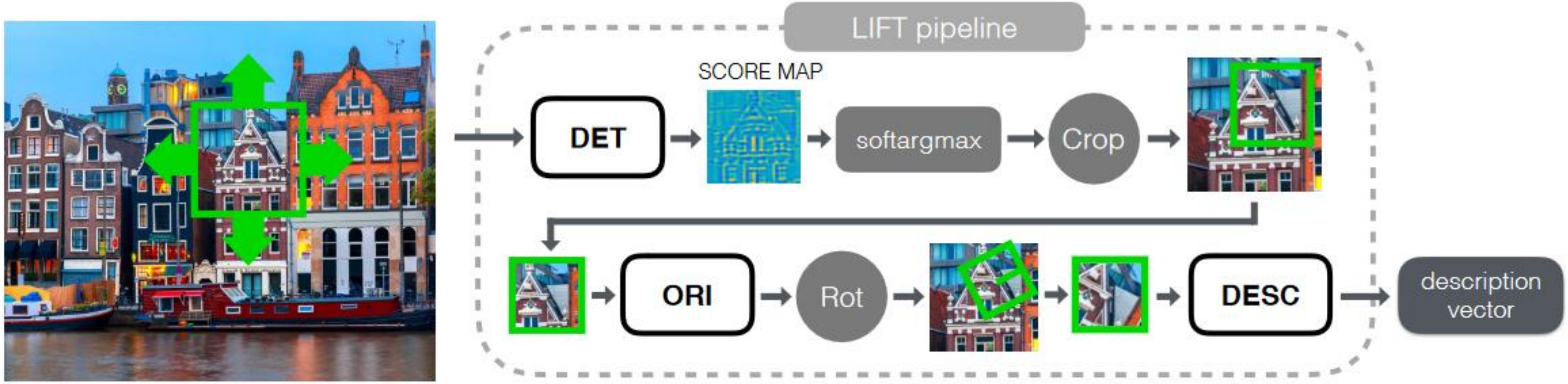
Handcrafted vs Learned features



Zamir et al. 2016.

LIFT: Learned Invariant Feature Transform

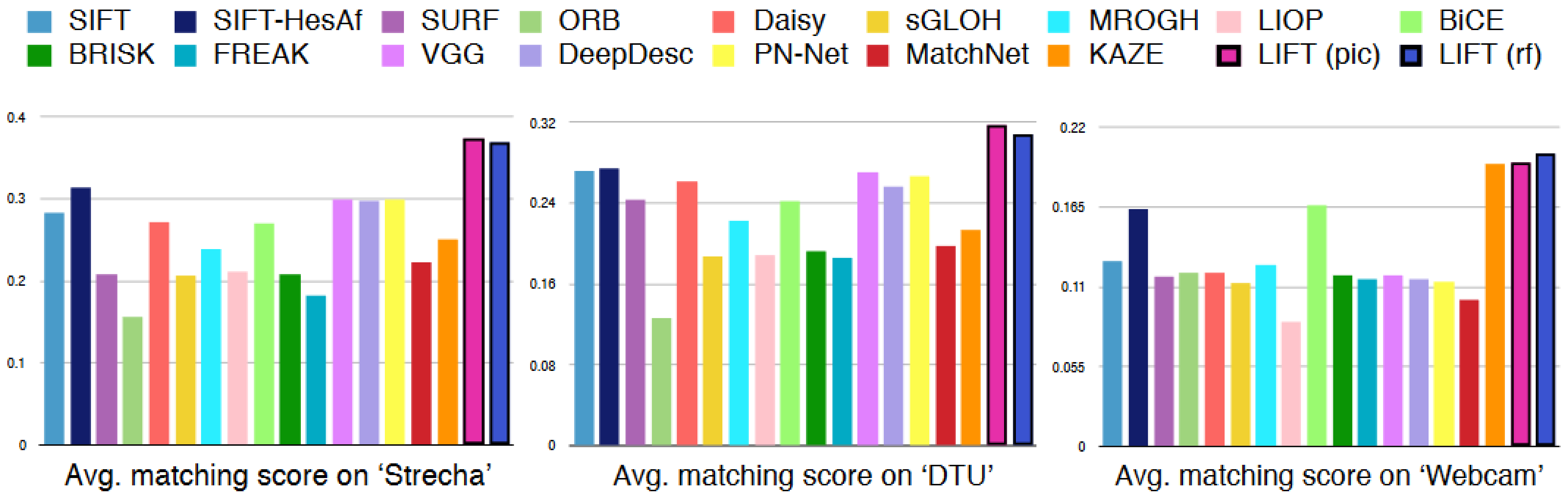
- Learn a keypoint detector.
- Learn a keypoint orientation predictor.
- Learn a keypoint descriptor.



K. M. Yi, E. Trulls, V. Lepetit, P. Fua, LIFT, 2016

LIFT

- Works well in domains close to the training data.
- Slower than SIFT.



Representation Learning Losses

- Given a set of samples x_i we learn a function that maps each sample into an embedding space $f(x_i) = \phi_i \in \mathbb{R}^d$.
- To learn f we need a loss function that acts on representations.
- Training data comes with positive $x_{i,j}^+$ and negative examples $x_{i,k}^-$ for each sample x_i (or you can construct them easily).

x_i

$x_{i,j}^+$

$x_{i,k}^-$

Cosine Similarity

- Cosine similarity: cosine of the angle between embedding vectors.
- Cosine is 1 if the embeddings align, 0 if orthogonal, -1 if opposite.

$$f(x) = \phi$$

$$\hat{\phi} = \frac{\phi}{\|\phi\|}$$

$$\mathcal{S}_{\cos}(\phi_i, \phi_j) = \hat{\phi}_i^T \, \hat{\phi}_j$$

Cosine Similarity

- For each sample, we maximise the similarity with its positives, and minimise similarity with its negatives.

$$\mathcal{L}_{\cos}(\phi_i) = -\frac{1}{J_i}\sum_{j=1}^{J_i} \mathcal{S}_{\cos}\left(\phi_i, \phi_{i,j}^{+}\right) + \frac{1}{K_i}\sum_{k=1}^{K_i} \mathcal{S}_{\cos}\left(\phi_i, \phi_{i,k}^{-}\right)$$

- Often select one random positive and one random negative for faster loss approximation.

$$\mathcal{L}_{\cos}(\phi_i) = -\mathcal{S}_{\cos}(\phi_i, \phi_i^{+}) + \mathcal{S}_{\cos}(\phi_i, \phi_i^{-})$$

Negatives

- Why do we need negatives?

- Degenerate solution: $f(x) = c$ predicts a constant for all x.

$$\mathcal{L}_{\cos}\big(f(x_i), f(x_j)\big) = -\mathcal{S}_{\cos}(c, c) = -1$$

- This minimises the loss for all training samples
- But: the representation is useless.

Triplet Loss

- Cosine similarity forces positive pairs to be almost identical (colinear) and negatives to be fully dissimilar.
- Often: select one random positive and one random negative for faster loss approximation.
- Triplet: anchor, positive, negative (ϕ, ϕ^+, ϕ^-)
- Idea: relative loss.
- Similarity between positive pair should be greater than similarity of negative pair.

Triplet Loss

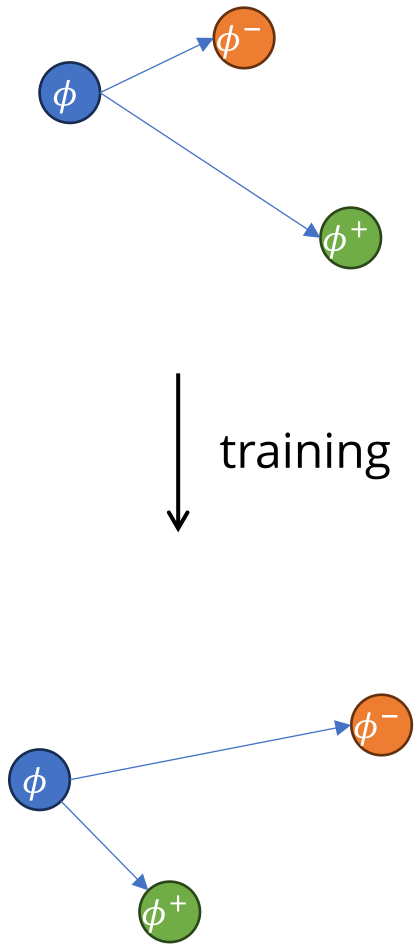
Similarity between positive pair should be greater than similarity of negative pair.

$$\mathcal{L}_{\text{triplet}}(\phi, \phi^+, \phi^-) = \max(0, \mathcal{S}(\phi, \phi^-) - \mathcal{S}(\phi, \phi^+) + \epsilon)$$

Minimum: $\mathcal{L}_{\text{triplet}} = 0$
if $\mathcal{S}(\phi, \phi^+) > \mathcal{S}(\phi, \phi^-) + \epsilon$ for a margin $\epsilon > 0$.

Can use any similarity/distance metric :

$$\max(0, \|\phi - \phi^+\| - \|\phi - \phi^-\| + \epsilon)$$



Similarity vs. Distance

- Similarity
  - Cosine similarity, correlation
  - Intersection over Union
  - …
- Distance
  - Euclidian distance (L1, L2, …)
  - Manhattan distance
- Remember: minimise distance, maximise similarity.

Efficiency Considerations

- The triplet loss uses three function evaluations for each loss $f(x), f(x^+), f(x^-)$.
- Can we do better?
- $f(x) = \phi$ is slow, while $\mathcal{L}(\cdot)$ is much faster in comparison.
- Find a way to use each ϕ multiple times.
- Idea: in a training batch, we can use most other samples as negatives too.

Efficiency

- Construct a training batch such that
  - each sample x_i has exactly one positive pair x_i^+
  - all other samples x_j (and x_j^+) are negatives to x_i (and x_i^+)
  - Example: include exactly two images of each class.

- Now we can reuse the computed embeddings for every sample

$$\sum_{j \neq i} \mathcal{L}_{\text{triplet}}\left(\phi_i, \phi_i^+, \phi_j\right) + \mathcal{L}_{\text{triplet}}\left(\phi_i, \phi_i^+, \phi_j^+\right)$$

- Batch of $6N$ samples. Before: $2N$ loss evals . Now: $4N(2N-1)$

Contrastive Loss

- Simplify notation: i^+ is the index of the positive pair to i.

$$-\log \frac{\exp\left(\mathcal{S}(\phi_i, \phi_{i^+})\right)}{\sum_{k=1}^{B} \exp\left(\mathcal{S}(\phi_i, \phi_k)\right)}$$

- Minimised by large numerator and small denominator.

- Same ides: maximise $\mathcal{S}(\phi_i, \phi_i^+)$ and minimise all other similarities.

Recall: Softmax Cross-entropy loss

Soft-max classifier for K classes C_k:

$$p(C_k|x) = \mathrm{softmax}_k\, f(x) = \frac{\exp f_k(x)}{\sum_j \exp f_j(x)}$$

Cross-entropy:

$$-\sum_{k}^{K} p_{GT}(C_k, x) \log\big(p(C_k|x)\big)$$

Since all $p_{GT}(C_k|x)$ are zero, except the target class C_{GT},i.e. $p_{GT}(C_{GT}, x) = 1$, this simplifies to

$$-\log\big(p(C_{GT}|x)\big) = -\log \frac{\exp f_{GT}(x)}{\sum_j \exp f_j(x)}$$

Contrastive Loss

$$-\log \frac{\exp\left(\mathcal{S}(\phi_i, \phi_{i^+})\right)}{\sum_{k=1}^{B} \exp\left(\mathcal{S}(\phi_i, \phi_k)\right)}$$

- Classifier where the logits are replaced by similarities.
- B-way classifier with one positive target per sample.
- Find the positive sample among all others.

Contrastive Loss

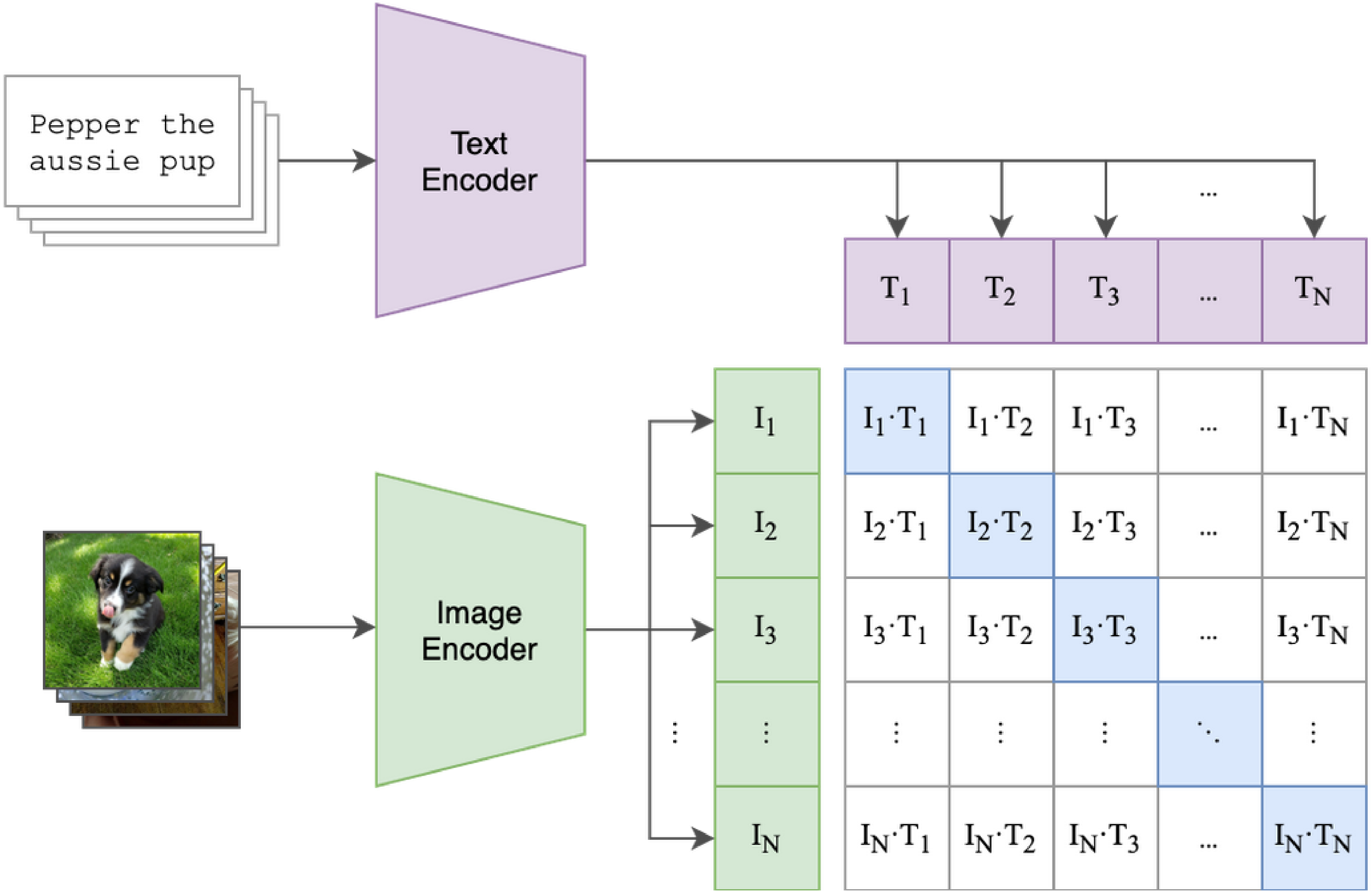
- Naming is confusing and inconsistent.
- Noise Contrastive Estimation (Gutmann, Hyvarinen, 2010)
  - Learn to separate data and noise with logistic regression.
- Proper name: InfoNCE (“CPC”, van den Oord, et al., 2018)
  - Use soft-max crossentropy to find positive sample within the batch.
- Popular loss: now often simply called contrastive loss.

Multi-Modal Representation Learning

- Goal: learn a common embedding space for different modalities.
- Typical setup:
  - one encoder per modality.
  - Train contrastively using matching pairs across modalities.
- Strong representations as they relate information from multiple sources.

CLIP: Contrastive Language-Image Pre-Training

- Learn a joint embedding space of images and text
- Double Contrastive loss: across text and images
- Trained on 400M Image-Text pairs.



Radford et al., Learning Transferable Visual Models From Natural Language Supervision, 2021

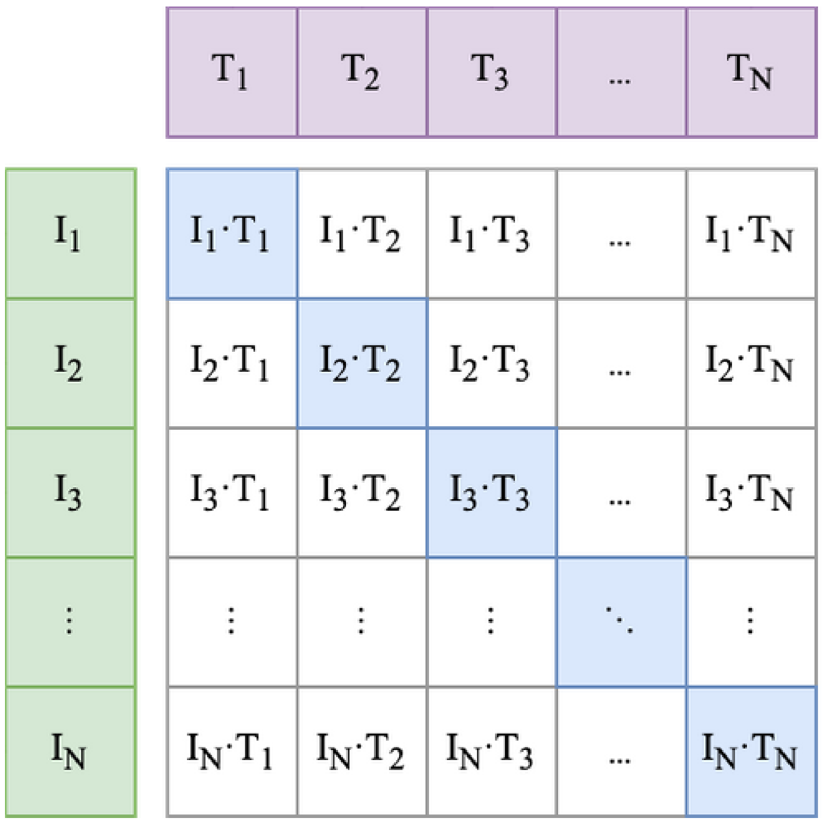
CLIP Implementation

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_e]
T_f = text_encoder(T)  #[n, d_e]


# scaled pairwise cosine similarities [n, n]
logits = dot(I_e, T_e.T) * exp(t)

# symmetric loss function
labels = arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



SigLIP

- Zhai et al., ICCV’23 (SigLIP2, arxiv’25)
- Possible improvement over CLIP (hard to tell)
- Focused on efficiency, multi-GPU training
- Back to negatives and positives instead of contrastive training
- Each entry in the matrix is a binary classifier

SigLIP implementation

```
# img_emb : image model embedding [n, dim]
# txt_emb : text model embedding [n, dim]
# t_prime, b : learnable temperature and bias
# n : mini-batch size

t = exp(t_prime)
zimg = l2_normalize(img_emb)
ztxt = l2_normalize(txt_emb)
logits = dot(zimg, ztxt.T) * t + b
labels = 2 * eye(n) - ones(n) # -1 with diagonal 1
l = -sum(log_sigmoid(labels * logits)) / n
```

Models learn to read

- Many examples of images containing text in the training data
- Models “learn to read”
- Easy adversarial examples

42%	78%	an apple
56%	91%	a picture of an apple
60%	0%	an ipod
99%	0%	an apple with a note saying "ipod"

from SigLIP2 colab notebook

Video and Sound

A Owens et al., Visually Indicated Sounds, 2015

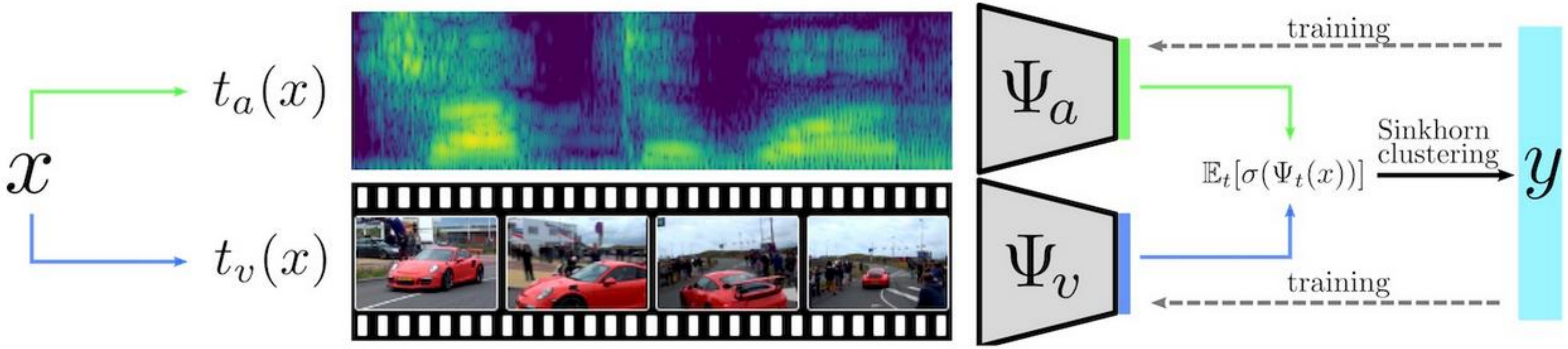
Video source

Vision and Sound

A Owens et al., Ambient Sound Provides Supervision for Visual Learning, 2016

Multi-Modal Retrieval

- Combine multi-modal representation learning with clustering.
- Learn an audio-visual embedding from videos with sound.



[Demo link](#)

Asano et al., Labelling unlabelled videos from scratch with multi-modal self-supervision, 2020

Multi-Modal Retrieval

Table 4: **Retrieval** via various number of nearest neighbors.

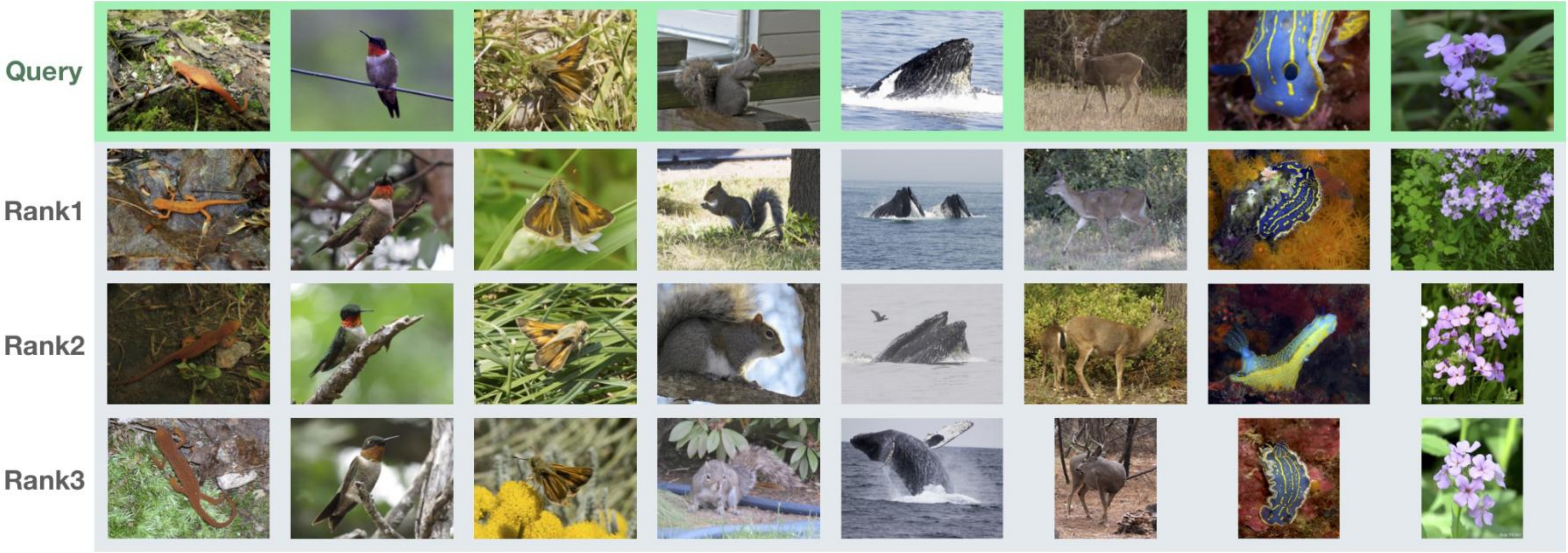
	HMDB			UCF		
Recall @	1	5	20	1	5	20
3D-Puzzle [42]	–	–	–	19.7	28.5	40.0
OPN [46]	–	–	–	19.9	28.7	40.6
ST Order [12]	–	–	–	25.7	36.2	49.2
ClipOrder [78]	7.6	22.9	48.8	14.1	30.3	51.1
SpeedNet [11]	–	–	–	13.0	28.1	49.5
VCP [51]	7.6	24.4	53.6	18.6	33.6	53.5
VSP [19]	10.3	26.6	54.6	24.6	41.9	76.9
SeLaVi	**24.8**	**47.6**	**75.5**	**52.0**	**68.6**	**84.5**

Asano et al., Labelling unlabelled videos from scratch with multi-modal self-supervision, 2020

Ranking Loss

- What if we have multiple positives?
- We want the similarities of all positives to be greater than to all negatives.
- Sometimes, there is a ranking between positives.
- Ranking losses can be built from pairwise losses (e.g. triplet).
- Ranking useful to learn retrieval problems.

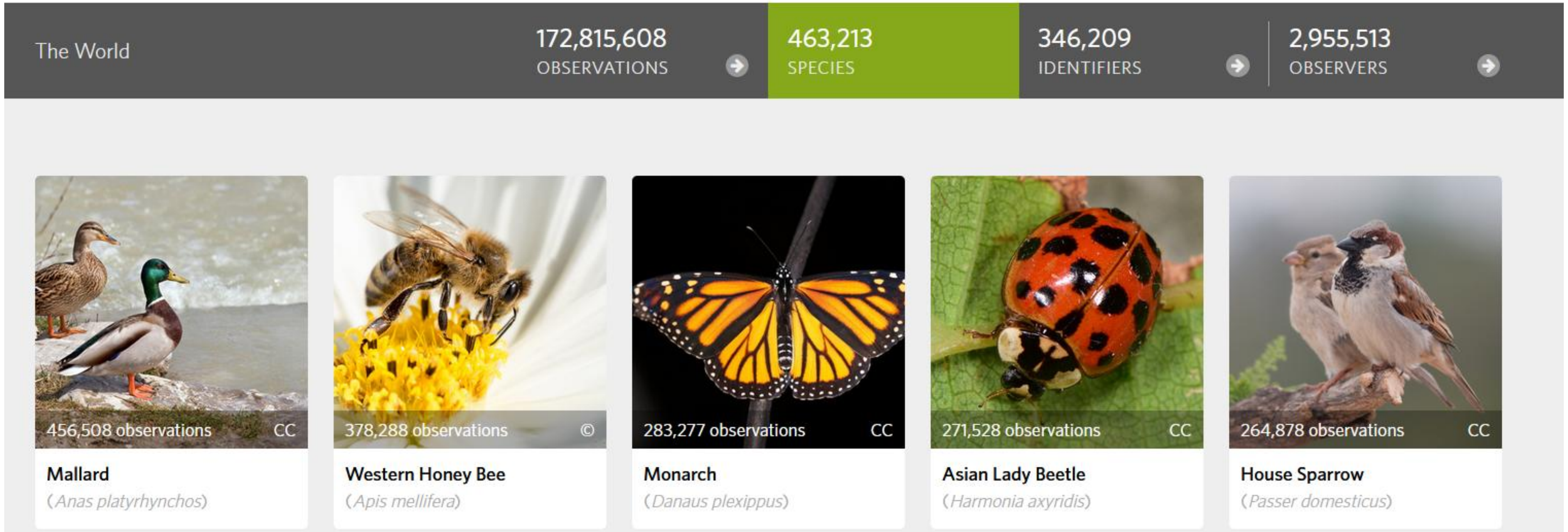
Representation Learning for Retrieval



Brown, et al., Smooth-AP: Smoothing the Path Towards Large-Scale Image Retrieval, 2020

Representation Learning for Retrieval

- Representation learning is very useful for retrieval problems.
- This is because there is usually not a predefined set of classes.
- Even if there is, the set of classes is too large to train a classifier.



www.inaturalist.org

↳ Beetles (Order Coleoptera)

↳ Water, Rove, Scarab, Long-horned, Leaf, and Snout Beetles (Suborder Polyphaga)

↳ Cucujiform Beetles (Infraorder Cucujiformia)

↳ Lady, Fungus, Scavenger, and Bark Beetles (Superfamily Coccinelloidea)

↳ Lady Beetles (Family Coccinellidae)

↳ Common Lady Beetles (Subfamily Coccinellinae)

↳ Black-spotted Lady Beetles (Tribe Coccinellini)

↳ **Greater Lady Beetles** (Genus *Harmonia*)	Observations
↳ Antipodean Ladybird (*Harmonia antipoda*)	57
↳ *Harmonia areolata*	0
↳ Asian Lady Beetle (*Harmonia axyridis*)	274,755
↳ *Harmonia bicolor*	6
↳ Large Spotted Ladybird (*Harmonia conformis*)	6,660
↳ *Harmonia decussata*	0
↳ Greater Asian Lady Beetle (*Harmonia dimidiata*)	1,193
↳ *Harmonia eucharis*	140
↳ *Harmonia manillana*	1
↳ *Harmonia nigromarginata*	0
↳ Maculate Ladybird (*Harmonia octomaculata*)	837
↳ Leopard Lady Beetle (*Harmonia pardalina*)	31
↳ Cream-streaked Ladybird (*Harmonia quadripunctata*)	2,515
↳ Sixteen-spotted Ladybird (*Harmonia sedecimnotata*)	283
↳ *Harmonia shoichii*	2
↳ Tortoise-shelled Ladybird (*Harmonia testudinaria*)	1,786
↳ Chequered Lady Beetle (*Harmonia vigintiduomaculata*)	51
↳ Yedo Lady Beetle (*Harmonia yedoensis*)	133