Output Embedding for Large-Scale Visual Recognition

Florent Perronnin

Xerox Research Centre Europe

ERC Allegro workshop, July 2014

(based on CVPR'14 tutorial)

Visual recognition

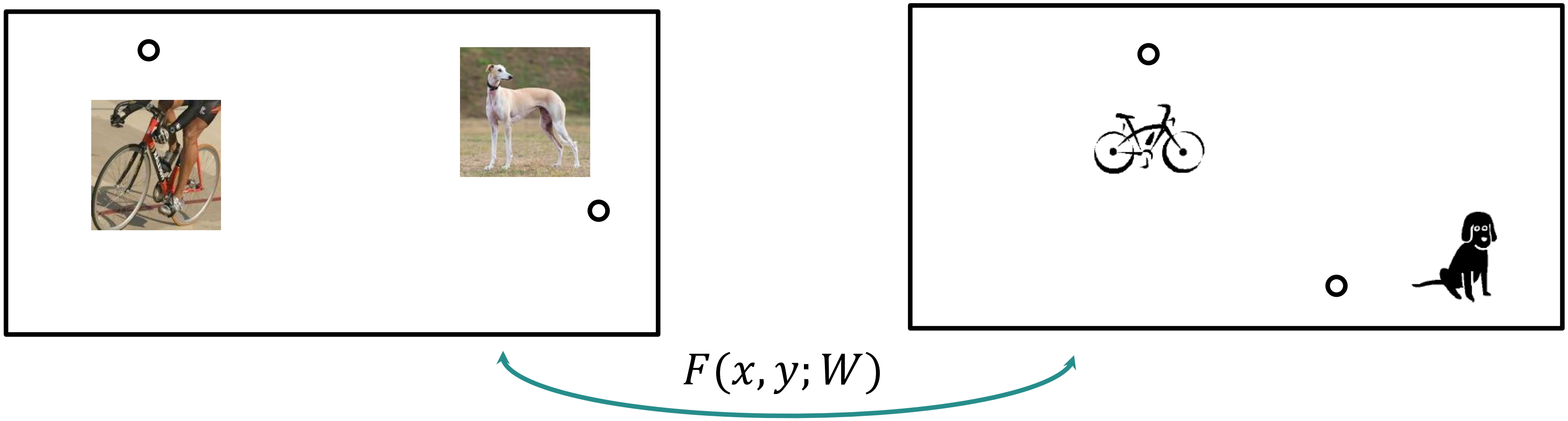
input =
image / video

output =
class name(s)

Input / output compatibility

Requires a **compatibility function** F between input x and output y:

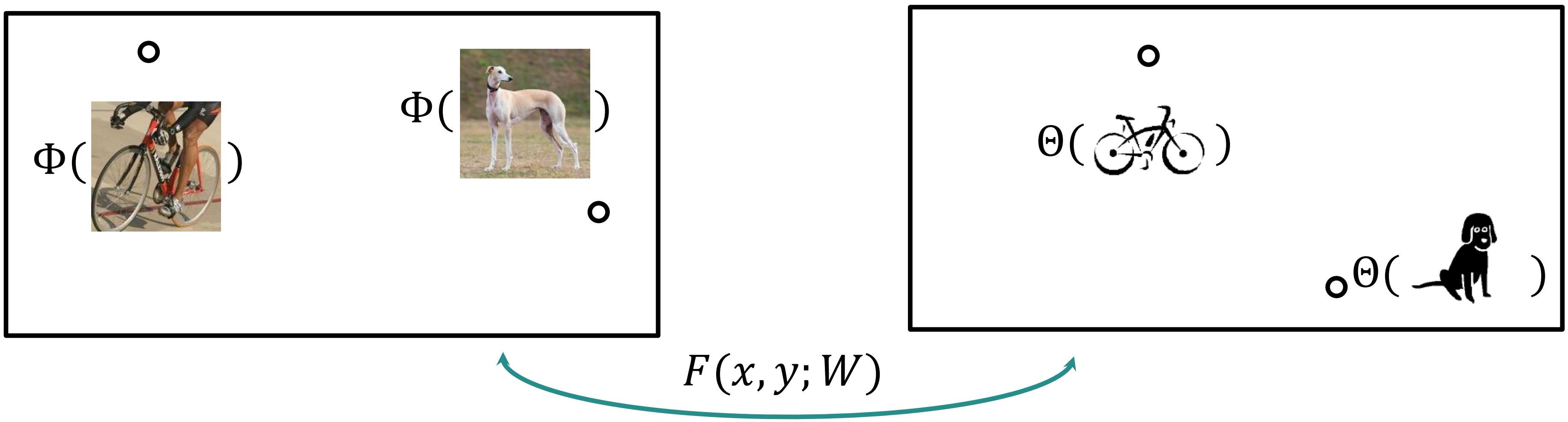
$$y^* = \arg\max_y F(x, y; W)$$



Input / output compatibility

Requires a **compatibility function** F between input x and output y:

$$y^* = \arg\max_y F(x, y; W)$$



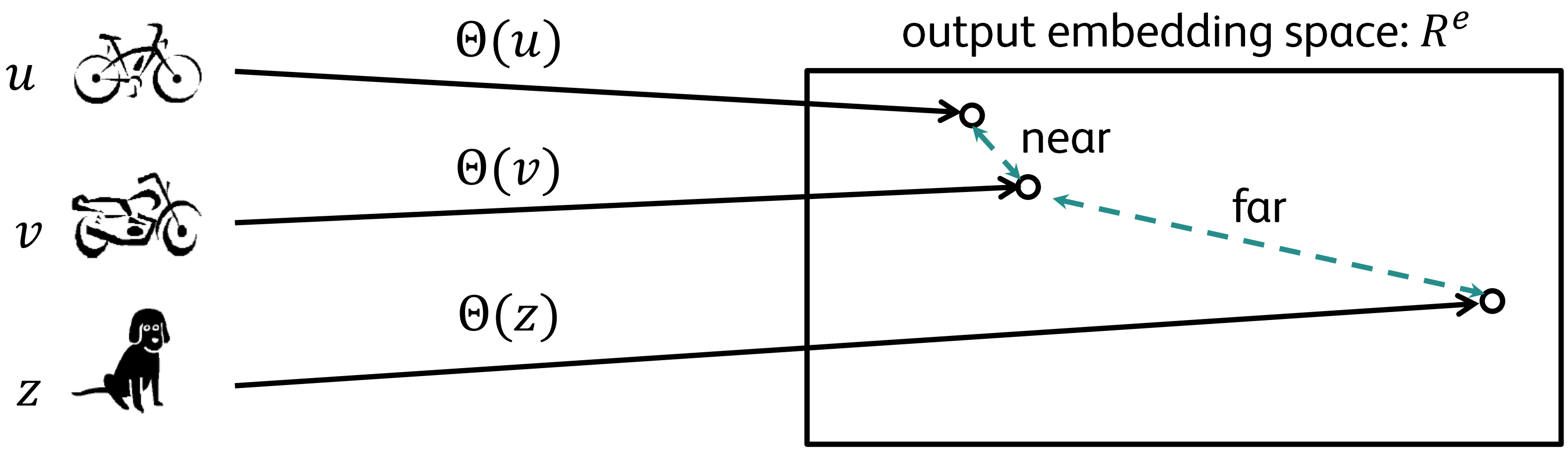
Directly measuring the image / class compatibility is challenging
→ first **embed input and output**

Output embedding / encoding

Encode output using a vectorial representation:

$y =$  $\longrightarrow \Theta(y)$

"Similar" classes u and v should be mapped to similar vectors $\Theta(u)$ and $\Theta(v)$:



Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Problems with a large number of classes

Web-scale face recognition:

LFW

Fine-grained recognition:

FGComp

Scene text recognition:

IIT-5K

xerox

Challenges of large number of classes

- Soft boundaries between classes
- Difficulty to collect labeled training data
- Computational and memory costs

Soft boundaries between classes

Standard assumption: **finite** number of **distinct** classes

Soft boundaries between classes

Standard assumption: **finite** number of **distinct** classes

As the number of concepts increases, we have to deal with:

- more and more complex concepts
- more and more unusual concepts
- overlapping concepts

Deng, Krause, Berg, Fei-Fei, “Hedging Your Bets: Optimizing Accuracy-Specificity Trade-offs in Large Scale Visual Recognition”, CVPR’12.

☹ the finite and distinct assumptions become less and less realistic

Difficulty to collect labeled training data

As the number of classes increases, the problem becomes **finer-grained**

As an example, ImageNet10K contains:

- 134 classes of fungus
- 183 classes of ungulates
- 262 classes of vehicles

Deng, Berg, Li, Fei-Fei, “What does classifying more than 10,000 image categories tell us?”, ECCV’10.

As the classes to be recognized become finer-grained:

- collecting data becomes harder and harder
- annotating the data requires expert knowledge

☹ only few training samples for some classes

Computational and memory costs

Standard approach to learning a large set of classifiers

→ learn a set of one-vs-rest classifiers independently

Deng, Berg, Li, Fei-Fei, “What does classifying more than 10,000 image categories tell us?”, ECCV’10.

Sánchez, Perronnin, “High-dimensional signature compression for large-scale image classification”, CVPR’11.

☺ trivially parallelizable

Computational and memory costs

Standard approach to learning a large set of classifiers

→ learn a set of one-vs-rest classifiers independently

Deng, Berg, Li, Fei-Fei, “What does classifying more than 10,000 image categories tell us?”, ECCV’10.

Sánchez, Perronnin, “High-dimensional signature compression for large-scale image classification”, CVPR’11.

☺ trivially parallelizable

$\left.\begin{array}{l}\text{☹ training cost}\\ \text{☹ inference cost}\\ \text{☹ memory cost}\end{array}\right\}$ in $O(\#\ classes)$

→ linear cost might be prohibitive for very large # classes

Output embedding addresses all 3 challenges

- Soft boundaries between classes

→ from a discrete set of classes to a potentially ∞ **set of continuous classes**

- Difficulty to collect labeled training data

→ correct choice of embedding function Θ enables **parameter sharing**

→ **side information** can be incorporated

- Computational and memory costs

→ number of "compatibility" parameters can be easily **parametrized**

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Outline

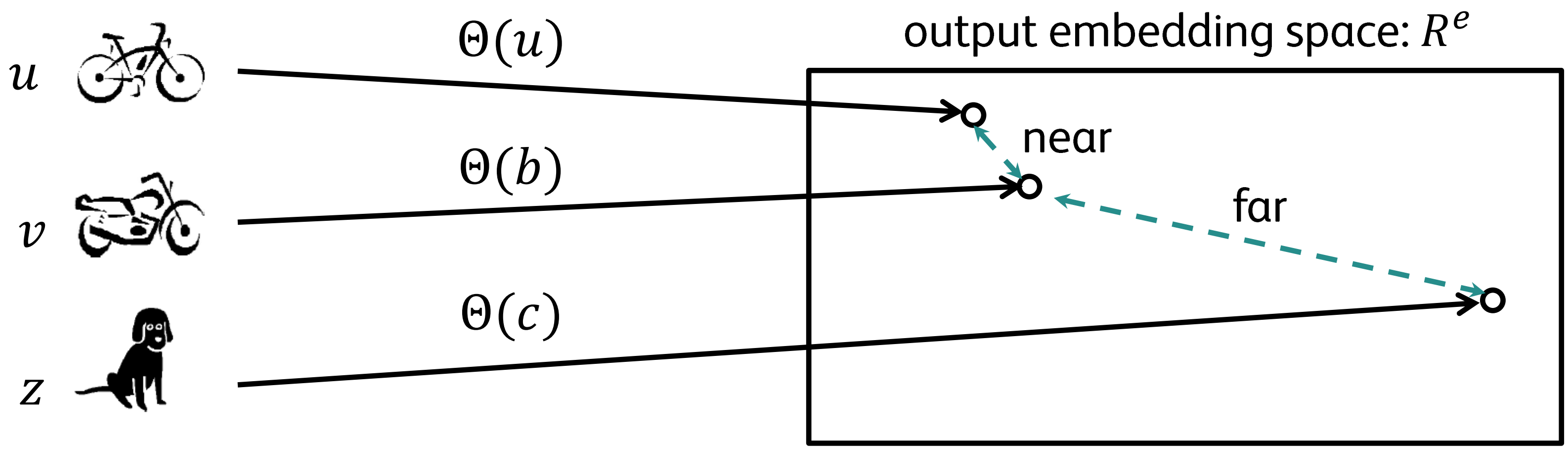
Challenges of a large # of classes

The three design choices of output embedding:

- **embedding function**
- input/output compatibility function
- learning objective function

Results

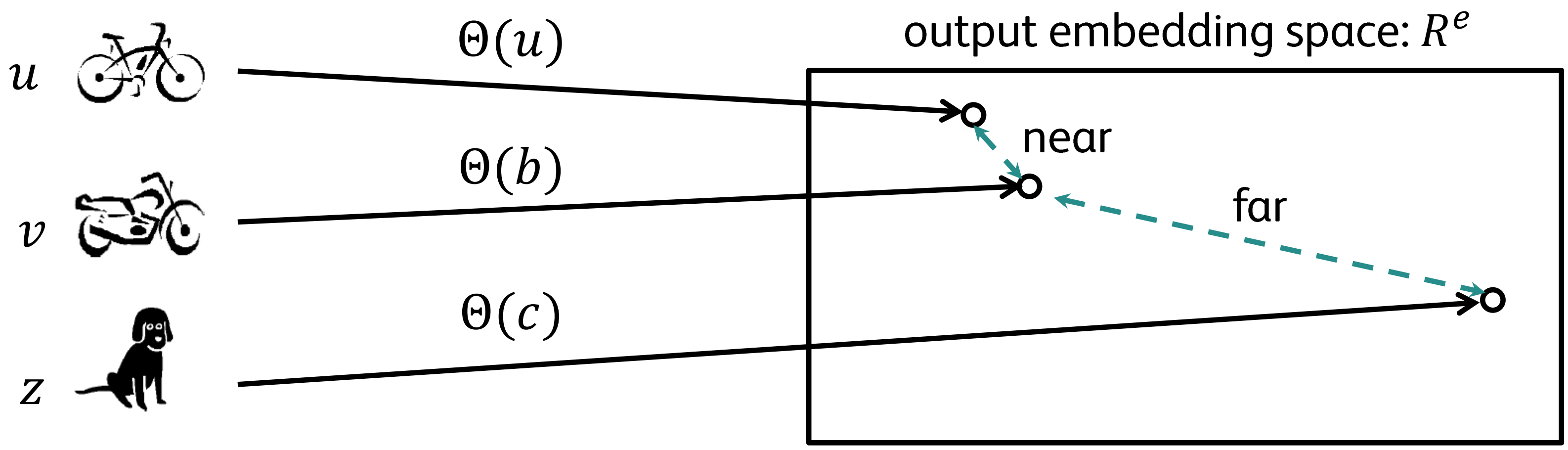
A taxonomy of embeddings



Three classes of embedding functions:

- Data-independent embeddings
- Learned embeddings (from training dataset)
- Embeddings derived from side information (external to training set)

A taxonomy of embeddings



Three classes of embedding functions:

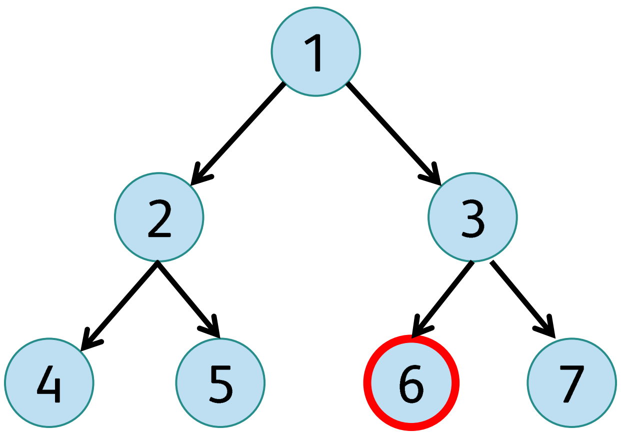
- Data-independent embeddings
- Learned embeddings (from training dataset)
- **Embeddings derived from side information (external to training set)**

Embeddings derived from side information

From a class hierarchy

Embed a class in a binary space:

- dimensionality is the number of classes in hierarchy
- a dimension is 1 if it corresponds to the considered class or one of its ancestors



$$\Theta(6) = [1, 0, 1, 0, 0, 1, 0]$$

→ classes in same path share parameters

Tsochantaridis, Joachims Hofmann, Altun,
"Large margin methods for structured and interdependent output variables", JMLR'05.

☺ simple and efficient

☹ which taxonomy to use? → learn tree structure, e.g. from confusion matrix

Bengio, Weston, Grangier, "Label embedding trees for large multi-class tasks", NIPS'10.

Deng, Satheesh, Berg, Fei-Fei, "Fast and Balanced: Efficient Label Tree Learning for Large Scale Object Recognition", NIPS'11.

Embeddings derived from side information

From attributes

Attributes: properties of an object which are shared across classes

Ferrari, Zisserman, "Learning visual attributes", NIPS'07.

Lampert, Nickisch, Harmeling, "Learning to detect unseen object classes by between-class attribute transfer", CVPR'09.

Ruby-throated Hummingbird

size = small
underparts color = olive
back color = grey
…

Use attribute-to-class associations to encode classes:

Akata, Perronnin, Harchaoui, Schmid, "Label-Embedding for Attribute-Based Classification", CVPR'13.

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ \cdots \end{bmatrix}$$

is small? → yes

olive underparts? → yes

white back? → no

☺ visually similar categories are close

☹ requires expert knowledge

Embeddings derived from side information
From textual resources

Exploit **co-occurrence of class names in a textual corpus:**

- at document level: factorize the word-document matrix with LSA, pLSA, etc.
- at local level: find word representation which is useful to predict surrounding words

Mikolov, Chen, Corrado, Dean, “Efficient estimation of word representations in vector space”, ICLR’13.

Embeddings derived from side information
From textual resources

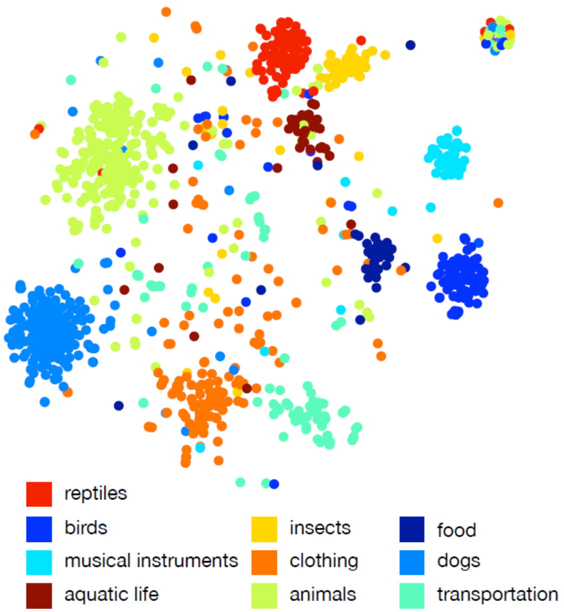
Exploit **co-occurrence of class names in a textual corpus:**

- at document level: factorize the word-document matrix with LSA, pLSA, etc.
- at local level: find word representation which is useful to predict surrounding words

Mikolov, Chen, Corrado, Dean, "Efficient estimation of word representations in vector space", ICLR'13.

Example embedding learned from wikipedia:

Frome, Corrado, Shlens, Bengio, Dean, Ranzato, Mikolov, "DeViSE: A deep visual-semantic embedding model", NIPS'13.



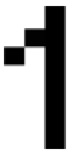
☺ semantically similar categories are close

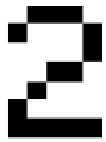
☹ no guarantee that visually similar categories are close

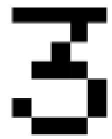
xerox

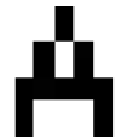
Embeddings derived from side information
Domain-specific embeddings

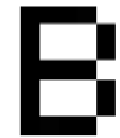
For the problem of **character recognition**, use a small (7×5 pixels) synthesized version of the character:

$1 =$ 

$2 =$ 

$3 =$ 

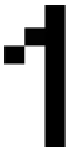
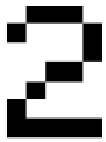
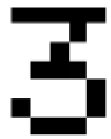
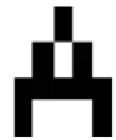
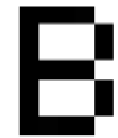
$A =$ 

$B =$ 

Larochelle, Erhan, Bengio, “Zero-data learning of new tasks”, AAAI’08.

Embeddings derived from side information
Domain-specific embeddings

For the problem of **character recognition**, use a small (7 × 5 pixels) synthesized version of the character:

$1 =$  $2 =$  $3 =$  $A =$  $B =$ 

Larochelle, Erhan, Bengio, "Zero-data learning of new tasks", AAAI'08.

Can go one step further by:

- synthesizing the characters → image
- embedding the synthesized image

$$\Theta(y) = \Phi(synthesis(y))$$

☺ input and output embeddings live in the same space

☹ synthesis + feature extraction comes at a cost

Rodríguez-Serrano, Sandhawalia, Bala, Perronnin, Saunders, "Data-driven vehicle identification by image matching", ECCV Workshops'12.

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Outline

Challenges of a large # of classes

The three design choices of output embedding:

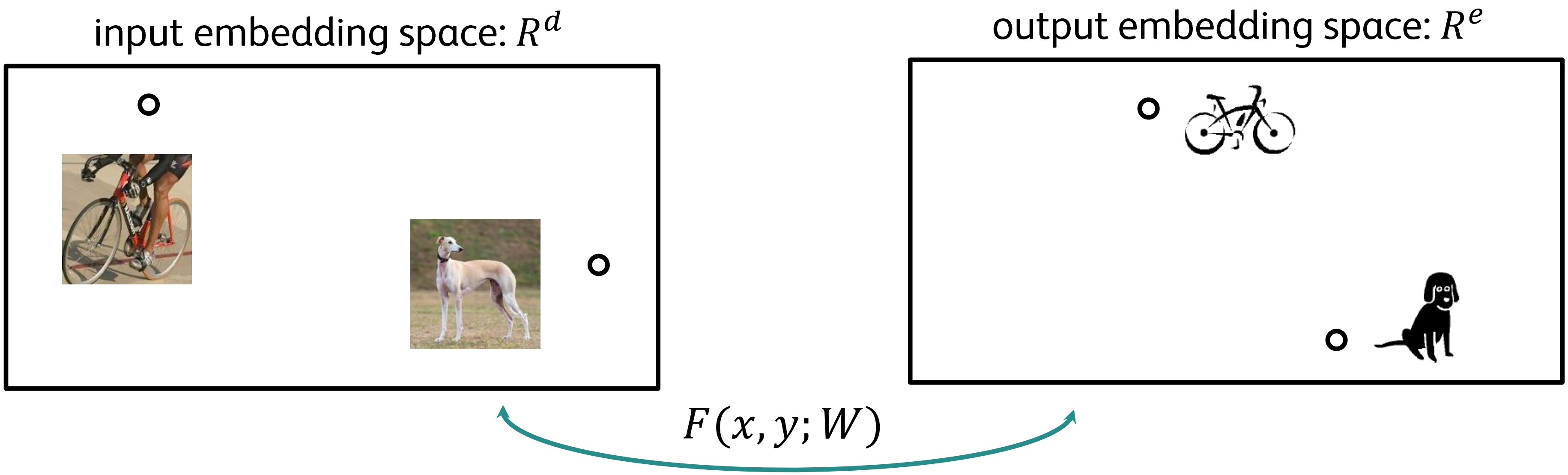
- embedding function
- **input/output compatibility function**
- learning objective function

Results

Compatibility function

Inference requires a **compatibility function** F between inputs and outputs:

$$y^* = \arg\max_y F(x, y; W)$$



How to measure the compatibility between $\Phi(x) \in R^d$ and $\Theta(y) \in R^e$, with $\boldsymbol{d} \neq \boldsymbol{e}$ in general?

xerox

Bilinear compatibility function

General case where $d \neq e$:

$$F(x, y; W) = \left[\boxed{\quad \Phi(x)^T \quad}\right] \left[\quad W \quad\right] \left[\boxed{\Theta(y)}\right]$$

where W is the $d \times e$ matrix that parametrizes the compatibility function

→ **input and output play symmetric** roles

→ related to **metric learning**: W encodes a metric between inputs/outputs

Bilinear compatibility function

General case where $d \neq e$:

$$F(x, y; W) = \left[\boxed{\quad \Phi(x)^T \quad}\right] \left[\quad W \quad\right] \left[\boxed{\Theta(y)}\right]$$

where W is the $d \times e$ matrix that parametrizes the compatibility function

Other possibilities:

- mapping input to output: $F(x, y; W) = -||W^T \Phi(x) - \Theta(y)||^2$
- mapping output to input: $F(x, y; W) = -||\Phi(x) - W\Theta(y)||^2$

xerox

Low-rank compatibility function

What if d and e are large? → high computational and memory costs

Use a low-rank decomposition of W: $W = U^T V$ with:

- U a $r \times d$ matrix
- V a $r \times e$ matrix

$r \ll d, e$

$F(x, y; W) = \Phi(x)^T W \Theta(y)$ rewrites as:

$$F(x, y; U, V) = \left(U\Phi(x)\right)^T \left(V\Theta(y)\right) = \Phi'(x)^T \Theta'(y)$$

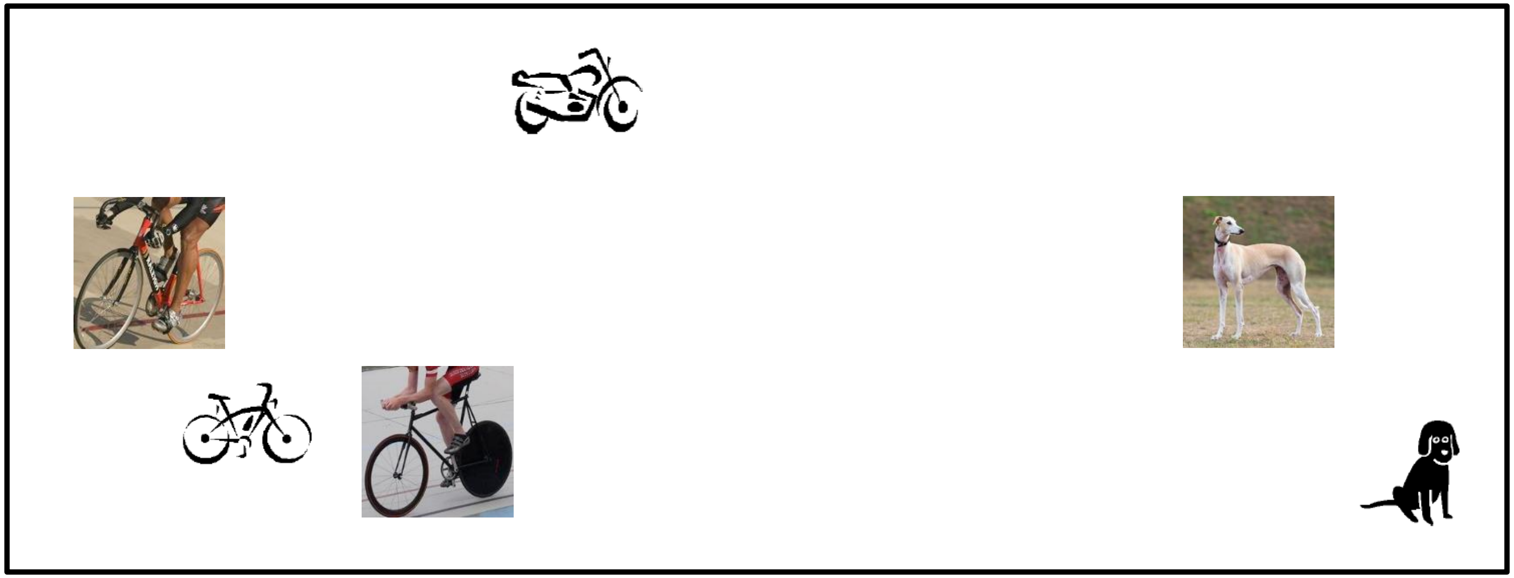
with $\Phi'(x) = U\Phi(x)$ and $\Theta'(y) V\Theta(y)$

→ **no clear cut between compatibility function and input/output embedding**

→ **joint embedding of input/output** in a common r-dim space

Advantages of a joint embedding

joint input / output embedding space

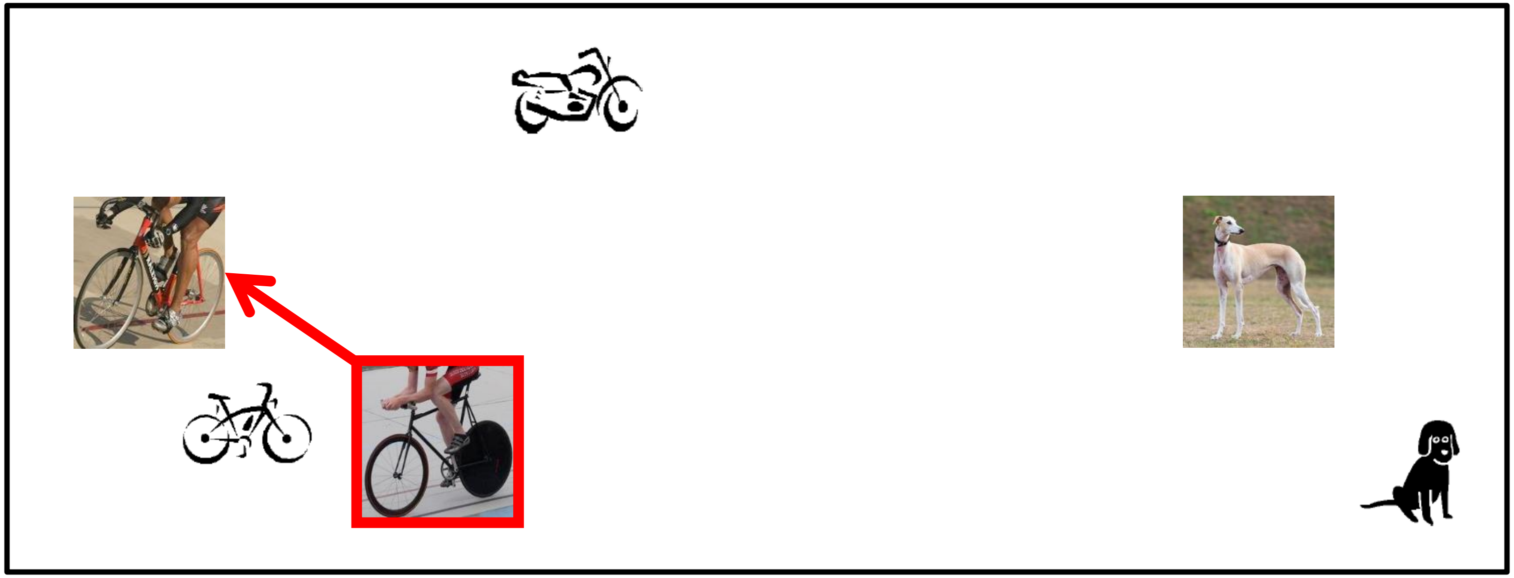


Joint embedding enables performing the following operations:

xerox

Advantages of a joint embedding

joint input / output embedding space

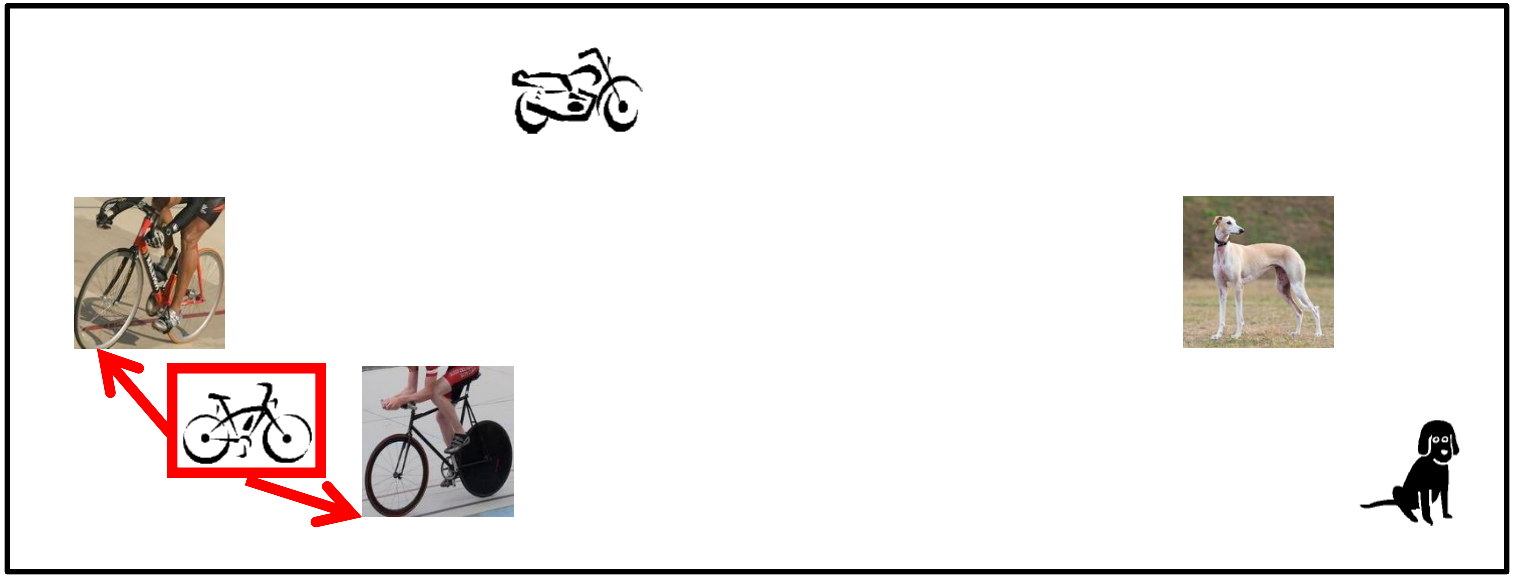


Joint embedding enables performing the following operations:

- image-to-image matching: search by example

Advantages of a joint embedding

joint input / output embedding space

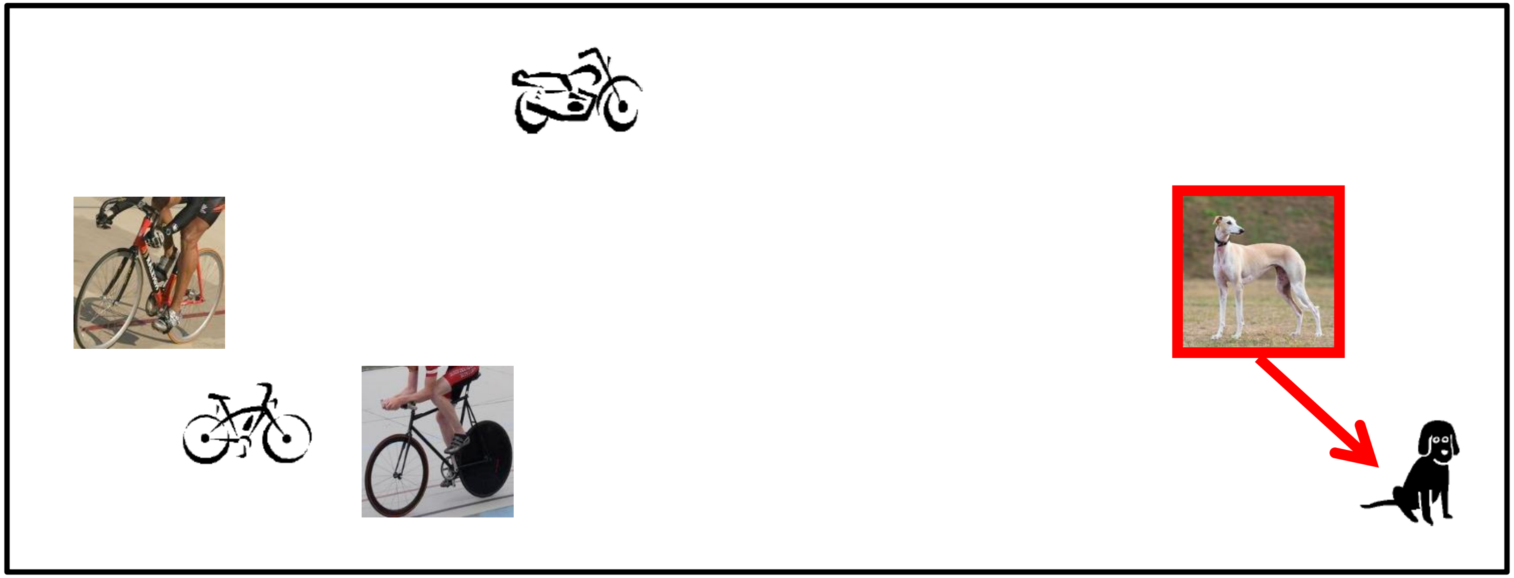


Joint embedding enables performing the following operations:

- image-to-image matching: search by example
- class-to-image matching: search by text query

Advantages of a joint embedding

joint input / output embedding space



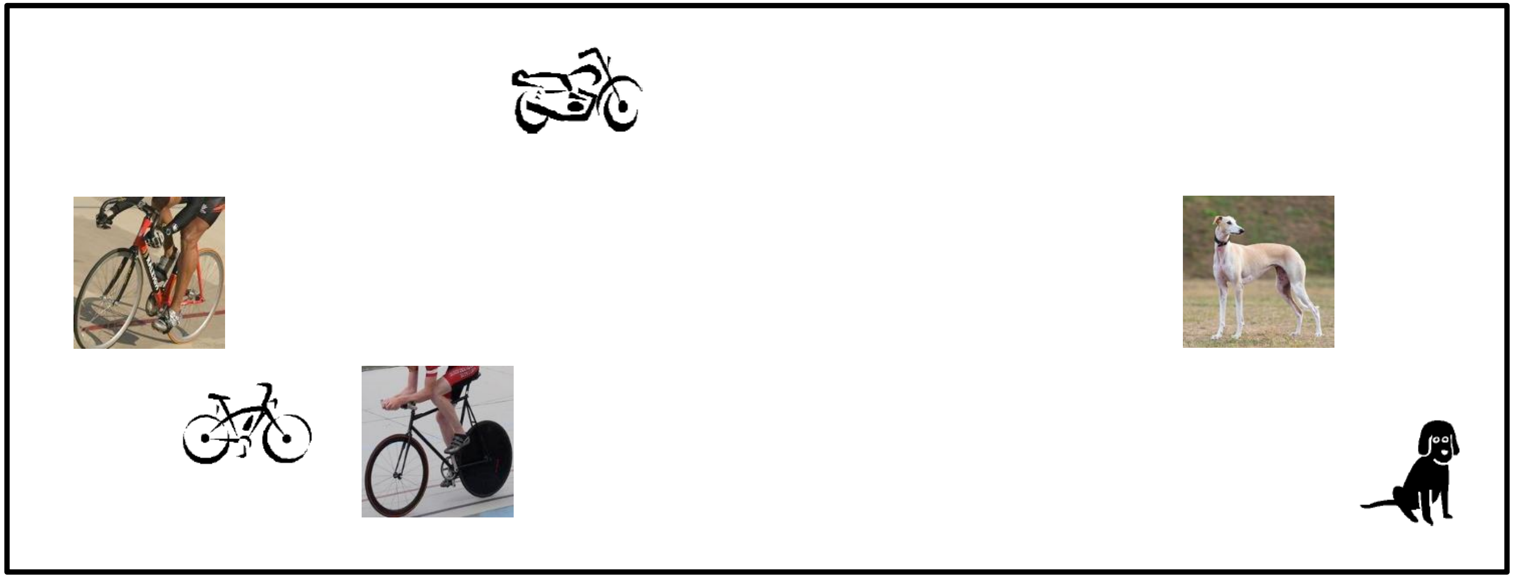
Joint embedding enables performing the following operations:

- image-to-image matching: search by example
- class-to-image matching: search by text query
- image-to-class matching: annotation

xerox

Advantages of a joint embedding

joint input / output embedding space



Joint embedding enables performing the following operations:

- image-to-image matching: search by example
- class-to-image matching: search by text query
- image-to-class matching: annotation

→ **bridges the gap between search and classification**

Gordo, Rodriguez-Serrano, Perronnin, Valveny, “Leveraging category-level labels for instance-level image retrieval”, CVPR’12

Non-linear compatibility function

Bi-linear compatibility function might be sufficient for very high-dimensional linearly separable inputs / outputs

But how to introduce non-linearity? Two solutions:

- solution 1: exploit relationship with **structured output learning**
- solution 2: exploit the relationship with **neural networks** and **deep learning**

Non-linear compatibility function

Solution 1: exploit the relationship with structured learning

Given:

- $\Psi(x, y) = \Phi(x) \otimes \Theta(y)$ a de-dim vector (joint input/output embedding)
- w the de-dim linearization of W

we can rewrite the compatibility function as:

$$F(x, y; W) = \Phi(x)^T W \Theta(y) = w^T \Psi(x, y)$$

→ standard structured output learning formalism

→ use **kernelized** version

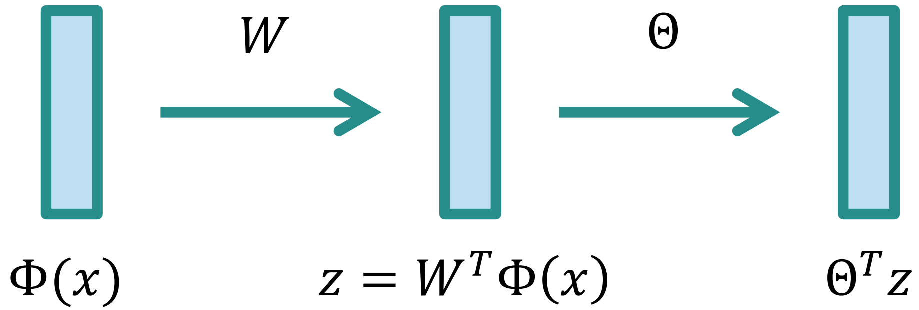
Tsochantaridis, Joachims Hofmann, Altun, “Large Margin Methods for Structured and Interdependent Output Variables”, JMLR’05.

Non-linear compatibility function

Solution 2: exploit the relationship with neural networks

Introducing $\Theta = [\Theta(1), \dots, \Theta(k)]$ the $e \times k$ matrix of output embeddings:

$$F(x, .; W) = \Theta^T (\Phi(x)^T W)$$



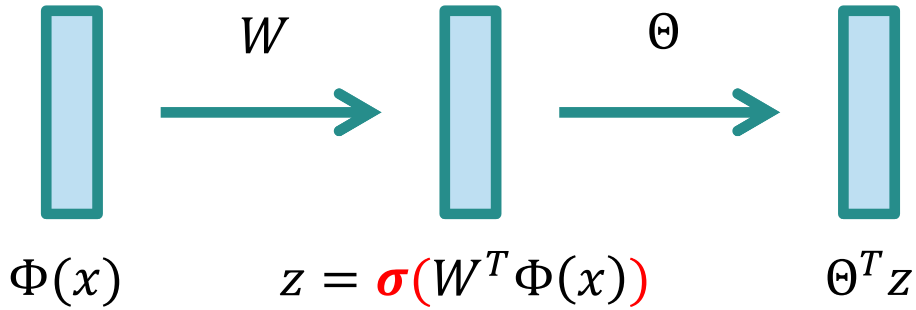
→ fully-connected neural network with 1 hidden layer and no non-linearity

Non-linear compatibility function

Solution 2: exploit the relationship with neural networks

Introducing $\Theta = [\Theta(1), \ldots, \Theta(k)]$ the $e \times k$ matrix of output embeddings:

$$F(x, .; W) = \Theta^T (\Phi(x)^T W)$$



$\rightarrow$ fully-connected neural network with 1 hidden layer and no non-linearity

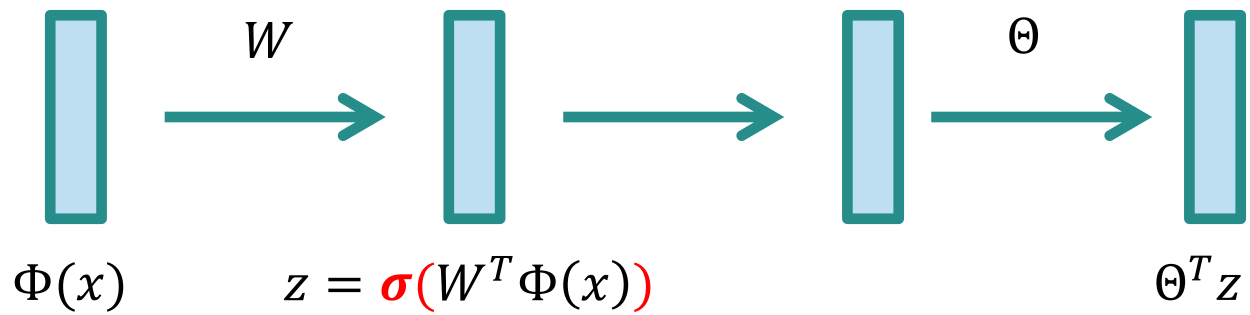
- add non-linearities

Non-linear compatibility function

Solution 2: exploit the relationship with neural networks

Introducing $\Theta = [\Theta(1), \ldots, \Theta(k)]$ the $e \times k$ matrix of output embeddings:

$$F(x,.;W) = \Theta^T(\Phi(x)^T W)$$



→ fully-connected neural network with 1 hidden layer and no non-linearity

- add non-linearities
- add more hidden layers

→ **deep learning** of the compatibility function

Hadsell, Chopra, LeCun, “Dimensionality reduction by learning an invariant mapping, CVPR’06.
Frome, Corrado, Shlens, Bengio, Dean, Ranzato, Mikolov, “DeViSE: A deep visual-semantic embedding model”, NIPS’13.

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- **learning objective function**

Results

The learning objective function

Two cases:

- The embedding is known and fixed a priori

$\rightarrow$ optimize over W only

- The embedding is learned:

$\rightarrow$ optimize over W and Θ

The learning objective function

Two cases:

- **The embedding is known and fixed a priori**

→ optimize over W only

- The embedding is learned:

→ optimize over W and Θ

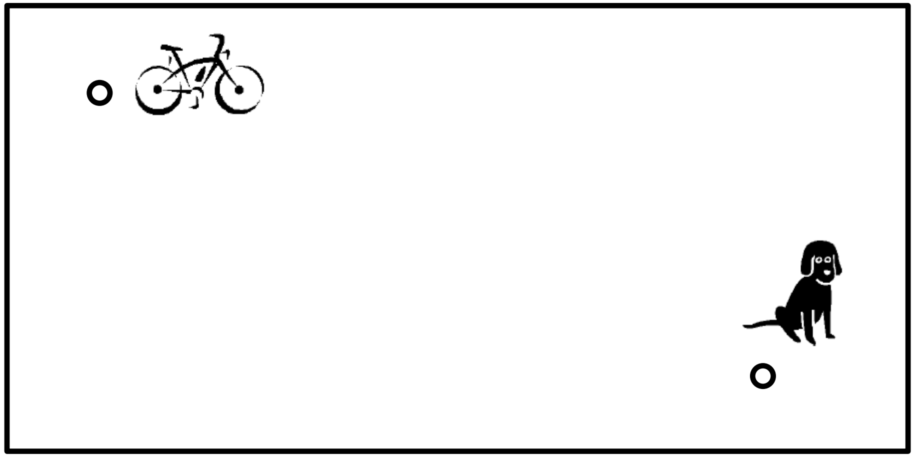
Fixed Θ, learn W

W can be learned from a set of classes and extrapolated to new classes for which we have no labeled training data: **zero-shot recognition**

$\rightarrow$ replacing labeled training data with descriptions

at training time:

output embedding space

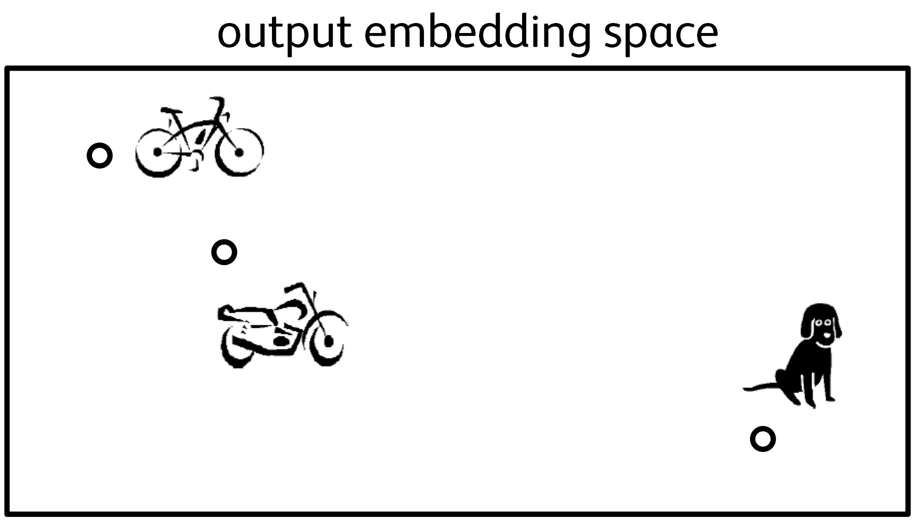


Fixed Θ, learn W

W can be learned from a set of classes and extrapolated to new classes for which we have no labeled training data: **zero-shot recognition**

→ replacing labeled training data with descriptions

at inference time:



Generalization ability depends on the distance of “new” classes to existing ones

Palatucci, Pomerleau, Hinton, Mitchell, “Zero-shot learning with semantic output codes”, NIPS’09.

Fixed Θ, learn W

Maximizing compatibility

Maximizing compatibility:

$$\arg\max_W \frac{1}{n}\sum_{i=1}^{n} F(x_i, y_i; W)$$

+ constraints on W (regularization)

If $F(x, y; W) = -||W^T\Phi(x) - \Theta(y)||^2$ or $F(x, y; W) = -||\Phi(x) - W\Theta(y)||^2$

→ **regression**

If $F(x, y; U, V) = -||U\Phi(x) - V\Theta(y)||^2$

→ **Canonical Correlation Analysis (CCA)**

☺ simple optimization

☹ does not optimize the end-goal

Fixed Θ, learn W

Large-margin framework

Let us assume an **image annotation** task: given an image, rank the correct labels higher than the incorrect ones

Given a triplet $\Big(x =$  $, y^+ =$  $, y^- =$  $\Big)$ we want to enforce:

$$F(x, y^+; W) > F(x, y^-; W)$$

$\rightarrow$ use a **large-margin** framework

Fixed Θ, learn W

Large-margin framework

Multi-class loss (mono-label problems):

$$\ell(x, y; W) = \max_j \{\Delta(y, y_j) - F(x, y; W) + F(x, y_j; W)\}$$

Crammer, Singer, "On the algorithmic implementation of multi-class kernel-based vector machines", MLR'01.

where $\Delta(y, y_j)$ quantifies the loss of misclassifying y and y_j.

- $\{0,1\}$ loss if $y = y_j$ or $y \neq y_j$
- more complex distances in Euclidean space are possible

→ optimize: $\arg\max_W \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i\,; W)$

+ some constraints on W (regularization)

Fixed Θ, learn W

Large-margin framework

Ranking loss (mono- and multi-label problems):

$$\ell(x, y; W) = \sum_{j=1}^{k} \max\{0, \Delta(y, y_j) - F(x, y; W) + F(x, y_j; W)\}$$

T. Joachims, “Optimizing search engines using clickthrough data”, SIGKDD’02.

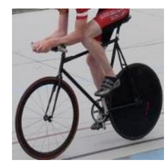
Fixed Θ, learn W
Large-margin framework

Ranking loss (mono- and multi-label problems):

$$\ell(x, y; W) = \sum_{j=1}^{k} \max\{0, \Delta(y, y_j) - F(x, y; W) + F(x, y_j; W)\}$$

T. Joachims, "Optimizing search engines using clickthrough data", SIGKDD'02.

Can also be applied to ranking images from text queries:

 $\longrightarrow$ ( ,  , )

Only difference: given a triplet ($y =$  , $x^+ =$  , $x^- =$ )
we want to enforce:

$$F(x^+, y; W) > F(x^-, y; W)$$

Fixed Θ, learn W
Large-margin framework

In the bilinear compatibility case:

$$\Delta(y^+, y^-) - F(x, y^+; W) + F(x, y^-; W) = \Delta(y^+, y^-) - x^T W (y^+ - y^-)$$

→ closely related to large-margin metric learning

Weinberger, Saul, "Distance metric learning for large margin nearest neighbor classification', JMLR'09.
Chechik, Shalit, Sharma, Bengio, "An online algorithm for large scale image similarity learning", NIPS'09.

See also: Kulis, "Metric learning: a survey", FTML'13.

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Outline

Challenges of a large # of classes

The three design choices of output embedding:

- embedding function
- input/output compatibility function
- learning objective function

Results

Example applications

Zero-shot recognition

Ruby-throated Hummingbird

→

size = small
underparts color = olive
back color = grey
…

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ \dots \end{bmatrix}$$

is small? → yes
olive underparts? → yes
white back? → no

Standard approach → **Direct Attribute Prediction (DAP):**

Lampert, Nickisch, Harmeling, "Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer", CVPR'09

- predict absence / presence of each attribute + combine probabilities

Approach based on output embedding → **Attribute Label Embedding (ALE):**

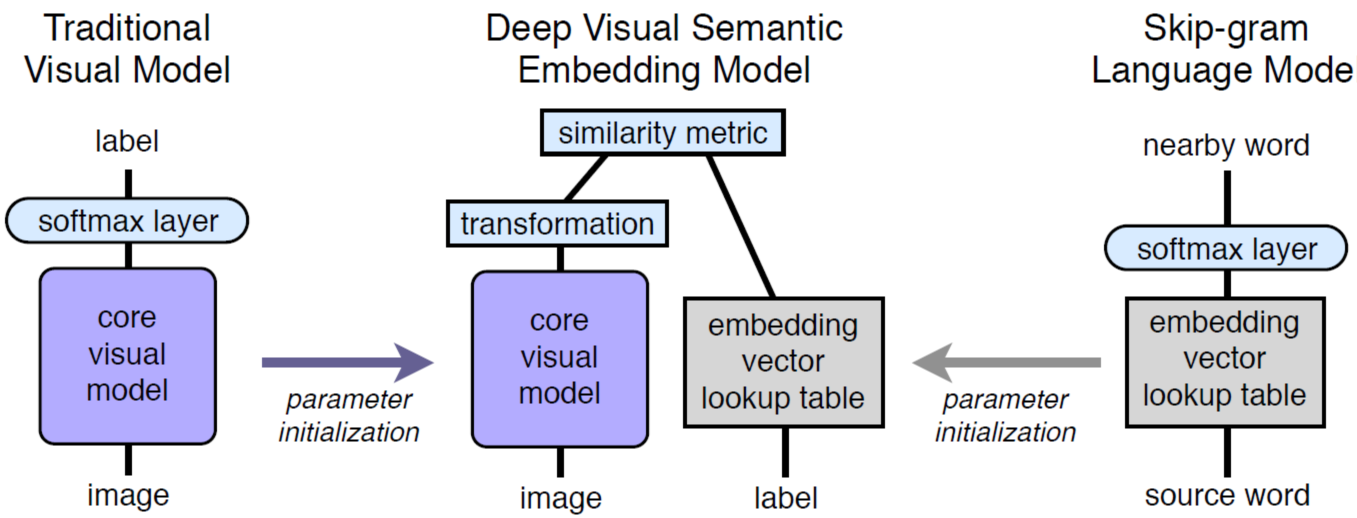
Akata, Perronnin, Harchaoui, Schmid, "Label-embedding for attribute-based classification", CVPR'13

- encode classes using attributes + map input / output with bilinear function

→ **ALE outperforms DAP**, e.g. on 200 birds dataset: 18 % accuracy vs 10.5 %

xerox

Example applications
Large-scale recognition (with abundant training data)



Frome et al., “DeViSE: A Deep visual-semantic embedding model”, NIPS’13

Comparison on ImageNet’12:

- flat loss: traditional visual model performs best
- hierarchical loss: model based on output embedding performs best

→ **system based on output embedding makes more plausible errors**

Example applications

Scene text recognition

Standard OCR approach:

- detect characters + combine character predictions

Bissacco, Cummins, Netzer, Neven, “PhotoOCR: Reading Text in Uncontrolled Conditions”, ICCV’13

Example applications
Scene text recognition

Standard OCR approach:

- detect characters + combine character predictions

Bissacco, Cummins, Netzer, Neven, “PhotoOCR: Reading Text in Uncontrolled Conditions”, ICCV’13

Approach based on output embedding:

- encode output words to respect **lexicographic similarity**

→ is a character present and where?

encoding “CVPR” $\begin{bmatrix}1\\0\\1\\\cdots\end{bmatrix}$

“C” in first half? → yes

“P” in first half? → no

“P” in second half? → yes

Rodríguez-Serrano and Perronnin, “Label embedding for text recognition”, BMVC’13.

Almazán, Gordo, Fornés, Valveny, “Handwritten word spotting with corrected attributes”, ICCV’13.

Example applications

Scene text recognition

Standard OCR approach:

- detect characters + combine character predictions

Bissacco, Cummins, Netzer, Neven, “PhotoOCR: Reading Text in Uncontrolled Conditions”, ICCV’13

Approach based on output embedding:

- encode output words to respect **lexicographic similarity**

→ is a character present and where?

encoding “CVPR”

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \cdots \end{bmatrix}$$

“C” in first half? → yes

“P” in first half? → no

“P” in second half? → yes

Rodríguez-Serrano and Perronnin, “Label embedding for text recognition”, BMVC’13.

Almazán, Gordo, Fornés, Valveny, “Handwritten word spotting with corrected attributes”, ICCV’13.

Results on Street View Text (SVT)

→ close to photoOCR at fraction of training cost

Almazán, Gordo, Fornés, Valveny, “Word Spotting and Recognition with Embedded Attributes”, TPAMI, to appear.

ABBY [32]	35.00
Mishra *et al.* [16]	73.26
Goel *et al.* [32]	77.28
PhotoOCR [5]	**90.39**
Proposed (KCSR)	87.01

Conclusion

The three design choices output embedding:

- The embedding function
- The input/output embedding function
- The learning objective function

→ can be combined in an almost ∞ number of ways

Conclusion

Output embedding enables handling a large number of classes:

- Soft boundaries between classes

→ from a discrete set of classes to a potentially ∞ set of continuous classes

- Difficulty to collect labeled training data

→ correct choice of embedding function enables parameter sharing

→ side information might be incorporated

- Computational and memory costs

→ number of model parameters can be easily parametrized

Conclusion

Output embedding is related to many other machine learning and computer vision concepts:

multi-task learning

structured learning   ECOC

deep learning   metric learning

transfer learning   attributes

zero-shot recognition