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 $y$:

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

$F(x, y; W)$
Input / output compatibility

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

$$y^* = \text{arg max}_y F(x, y; W)$$

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

Encode output using a vectorial representation:

\[ y = \begin{array}{c}
\text{bicycle}
\end{array} \rightarrow \Theta(y) \]

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

\[ u \rightarrow \Theta(u) \]
\[ v \rightarrow \Theta(v) \]
\[ z \rightarrow \Theta(z) \]

output embedding space: \( \mathbb{R}^e \)

\( u \) and \( v \) are near, \( z \) is far.
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:

Fine-grained recognition:

Scene text recognition:

LFW

FGComp

IIT-5K
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


⚠️ 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.

😊 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.

😊 trivially parallelizable

 риск training cost
 риск inference cost

 риск memory cost

→ linear cost might be prohibitive for very large # classes

\[ \text{in } O(# \text{ classes}) \]
Output embedding addresses all 3 challenges

• Soft boundaries between classes
  → from a discrete set of classes to a potentially $\infty$ set of continuous classes

• Difficulty to collect labeled training data
  → correct choice of embedding function $\Theta$ 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

Tschantaridis, 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

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
Lampert, Nickisch, Harmeling, “Learning to detect unseen object classes by between-class attribute transfer”, CVPR’09.

Use attribute-to-class associations to encode classes:

 зр visu ally similar categories are close
LookAndFeel requires expert knowledge

Ruby-throated Hummingbird

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

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

😊 semantically similar categories are close
ₓ no guarantee that visually similar categories are close
Embeddings derived from side information
Domain-specific embeddings

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

\[
1 = 1 \quad 2 = 2 \quad 3 = 3 \quad A = A \quad B = B
\]

Embeddings derived from side information
Domain-specific embeddings

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

\[
1 = 1 \quad 2 = 2 \quad 3 = 3 \quad A = A \quad B = B
\]


Can go one step further by:

- synthesizing the characters $\rightarrow$ 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:

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

**input embedding space:** $R^d$

**output embedding space:** $R^e$

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

General case where \( d \neq e \):

\[
F(x, y; W) = \begin{bmatrix} \Phi(x)^T \end{bmatrix} W \begin{bmatrix} \Theta(y) \end{bmatrix}
\]

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 $d \neq e$:

$$F(x, y; W) = \begin{bmatrix} \Phi(x)^T \end{bmatrix} W \begin{bmatrix} \Theta(y) \end{bmatrix}$$

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: $F(x, y; W) = -||\Phi(x) - W\Theta(y)||^2$
Low-rank compatibility function

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

Use a low-rank decomposition of $W: W = U^TV$ 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:

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

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 embedding enables performing the following operations:
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 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
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

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 \textbf{kernelized} version

Non-linear compatibility function
Solution 2: exploit the relationship with neural networks

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

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

$\Phi(x)$ $\rightarrow$ $z = W^T \Phi(x)$ $\rightarrow$ $\Theta^T z$

→ 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:

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

\[
W \xrightarrow{} \Theta
\]

\[
\Phi(x) \quad z = \sigma(W^T \Phi(x)) \quad \Theta^T z
\]

→ 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 \( \Theta = [\Theta(1), ..., \Theta(k)] \) the \( e \times k \) matrix of output embeddings:

\[
F(x, ; W) = \Theta^T (\Phi(x)^T W)
\]

\[\Phi(x) \quad z = \sigma(W^T \Phi(x)) \quad \Theta^T z\]

→ 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.
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
  → optimize over $W$ only

- The embedding is learned:
  → optimize over $W$ and $\Theta$
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 $\Theta$
Fixed $\Theta$, 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 training time:

[Diagram showing output embedding space with a bicycle and a dog]
Fixed $\Theta$, 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

Output embedding space

At inference time:

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

Fixed $\Theta$, learn $W$

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 $\Theta$, 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 $(x = \text{dog}, y^+ = \text{dog}, y^- = \text{bike})$ we want to enforce:

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

→ use a large-margin framework
Fixed $\Theta$, 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

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

+ some constraints on $W$ (regularization)
Fixed $\Theta$, learn $W$

Large-margin framework

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

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

Fixed $\Theta$, learn $W$

Large-margin framework

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

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


Can also be applied to ranking images from text queries:

Only difference: given a triplet $(y = \text{dog}, x^+ = \text{dog}, x^- = \text{bike})$
we want to enforce:

$$F(x^+, y; W) > F(x^-, y; W)$$
Fixed \( \Theta \), 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.

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

\[
\begin{bmatrix}
1 \\
1 \\
0 \\
\vdots
\end{bmatrix}
\]

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

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

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?

Results on Street View Text (SVT)
→ close to photoOCR at fraction of training cost

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBY [32]</td>
<td>35.00</td>
</tr>
<tr>
<td>Mishra et al. [16]</td>
<td>73.26</td>
</tr>
<tr>
<td>Goel et al. [32]</td>
<td>77.28</td>
</tr>
<tr>
<td>PhotoOCR [5]</td>
<td>90.39</td>
</tr>
<tr>
<td>Proposed (KCSR)</td>
<td>87.01</td>
</tr>
</tbody>
</table>
Conclusion

The three design choices output embedding:

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

\[\rightarrow \text{can be combined in an almost } \infty \text{ number of ways}\]
Conclusion

Output embedding enables handling a large number of classes:

- Soft boundaries between classes
  → from a discrete set of classes to a potentially $\infty$ 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