Multiview Feature Learning

Roland Memisevic

Frankfurt, Montreal

Tutorial at IPAM 2012
Outline

1 Introduction
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning

2 Learning relational features
   - Encoding relations
   - Learning

3 Factorization, eigen-spaces and complex cells
   - Factorization
   - Eigen-spaces, energy models, complex cells

4 Applications and extensions
   - Applications and extensions
   - Conclusions
What this is about

- Extend feature learning to model *relations*.

- Feature learning beyond object recognition
Extend feature learning to model *relations*.

“mapping units”, “bi-linear models”, “energy-models”, “complex cells”, “spatio-temporal features”, “covariance features”, “bi-linear classification”, “quadrature features”, “gated Boltzmann machine”, “mcrbm”, ...

**Feature learning beyond object recognition**
Object recognition started to work very well.
The main reason is the use of local features.
Local features for recognition

- Object recognition started to work very well.
- The main reason is the use of **local features**.
Find **interest points**.

2. Crop patches around interest points.

3. Represent each patch with a sparse local descriptor ("features").

4. Add all local descriptors to obtain a global descriptor for the image.
Bag-of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** ("features").
4. Add all local descriptors to obtain a global descriptor for the image.
Bag-Of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** ("features").
4. Add all local descriptors to obtain a global descriptor for the image.
Bag-Of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** ("features").
4. **Add** all local descriptors to obtain a global descriptor for the image.
Bag-Of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** ("features").
4. Add all local descriptors to obtain a global descriptor for the image.
1. Crop patches along a regular grid (dense or not).
2. Represent each patch with a local descriptor.
3. Concatenate all descriptors in a very large vector.
Convolutional

1. Crop patches along a regular grid (dense or not).
2. Represent each patch with a local descriptor.
3. Concatenate all descriptors in a very large vector.
Convolutional

1. Crop patches along a regular grid (dense or not).
2. Represent each patch with a local descriptor.
3. **Concatenate** all descriptors in a very large vector.
After computing representations, use logistic regression, SVM, NN, ...

There are various extensions, like fancy pooling, etc.
How to extract local features.

- Engineer them. SIFT, HOG, LBP, etc.
- Learn them from image data → deep learning
How to extract local features.

Engineer them. SIFT, HOG, LBP, etc.

Learn them from image data → deep learning
Extracting local features

- How to extract local features.
- Engineer them. SIFT, HOG, LBP, etc.
- *Learn* them from image data → **deep learning**
Feature learning

\[ z(y) = \text{sigmoid}(W^T y) \]

\[ y(z) = W z \]

\[ W = \arg \min_W \sum_{\alpha} \| y^\alpha - y(z(y^\alpha)) \|^2 \]
Feature learning models

Restricted Boltzmann machine (RBM)

\[ p(\mathbf{y}, \mathbf{z}) = \frac{1}{Z} \exp \left( \sum_{jk} w_{jk} y_j z_k \right) \]

Learning: Maximum likelihood/contrastive divergence.
Feature learning models

\[
z_k = \text{sigmoid} \left( \sum_j a_{jk} y_j \right)
\]

\[
y_j = \sum_k w_{jk} z_k
\]

Autoencoder

- Add **inference parameters**.
- Learning: Minimize reconstruction error.
- Add a sparsity penalty or **corrupt inputs during training** (Vincent et al., 2008).
Feature learning models

\[ y_j = \sum_k w_{jk} z_k \]

Independent Components Analysis (ICA)

- Learning: Make responses sparse, while keeping filters sensible

\[
\min_W \| W^T y \|_1 \\
\text{s.t. } W^T W = I
\]
Manifold perspective
Outline

1 Introduction
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning

2 Learning relational features
   - Encoding relations
   - Learning

3 Factorization, eigen-spaces and complex cells
   - Factorization
   - Eigen-spaces, energy models, complex cells

4 Applications and extensions
   - Applications and extensions
   - Conclusions
Beyond object recognition

Can we do more with Feature Learning than recognize *things*?

- Brains can do much more than recognize objects.
- Many vision tasks go beyond object recognition.
- In surprisingly many of them, the relationship *between* images carries the relevant information.
Beyond object recognition

Can we do more with Feature Learning than recognize *things*?

- Brains can do much more than recognize objects.
- Many vision tasks go beyond object recognition.
- In surprisingly many of them, the relationship between images carries the relevant information.
Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

**Some correspondence tasks in Vision**
- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure
Correspondence is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
  - Geometry
  - Optical Flow
  - Invariant Recognition
  - Odometry
  - Action Recognition
  - Contours, Within-image structure
Correspondence in Computer Vision

Correspondence is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
  - Optical Flow
  - Invariant Recognition
  - Odometry
  - Action Recognition
  - Contours, Within-image structure
Correspondence is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
  - Invariant Recognition
  - Odometry
  - Action Recognition
  - Contours, Within-image structure
Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure
Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- **Odometry**
- Action Recognition
- Contours, Within-image structure
Correspondence is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure
Correspondence is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure
**Correspondence** is one of the most ubiquitous problems in Computer Vision.

### Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- **Action Recognition**
- Contours, Within-image structure
Adding frames is not just about adding proportionally more information.

The relationships between frames contain additional information, that is not present in any single frame.

See *Heider and Simmel, 1944*: Any single frame shows a bunch of geometric figures. The motions reveal the story.
Random dot stereograms

You can see objects even when images contain *no* features.
If *correspondences* matter in vision, **can we learn them?**
Learning features to model correspondences

- We can, if we let latent variables act like gates, that dynamically change the connections between fellow variables.
Learning and inference (slightly) different from learning without.

We can set things up, such that inference is almost unchanged. Yet, the meaning of the latent variables will be entirely different.
Multiplicative interactions allow hidden variables to blend in a whole “sub”-network.

This leads to a qualitatively quite different behaviour from the common, bi-partite feature learning models.
Multiplicative interactions

Brief history of gating

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...
Multiplicative interactions

Brief history of gating

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...
Multiplicative interactions

**Brief history of gating**

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- **Higher-order neural nets, “Sigma-Pi-units”**
  - Routing circuits (Olshausen; 1994)
  - Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
  - Subspace SOM (Kohonen, 1996)
  - ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
  - (2006 –) GBM, mcRBM, GAE, convISA, applications...
Multiplicative interactions

Brief history of gating

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...
Multiplicative interactions

Brief history of gating

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...
Multiplicative interactions

Brief history of gating

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
  - ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
  - (2006 –) GBM, mcRBM, GAE, convISA, applications...
Multiplicative interactions

**Brief history of gating**

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...
**Multiplicative interactions**

**Brief history of gating**

- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...
Mapping units 1981

(Hinton, 1981)
Mapping units 1981

(Hinton, 1981)
Example application: Action recognition

(Hollywood 2)

(Marszałek et al., 2009)

- Convolutional GBM (Taylor et al., 2010)
- Hierarchical ISA (Le, et al., 2011)
Mocap

(Taylor, Hinton; 2009), (Taylor, et al.; 2010)
Gated MRFs

(Ranzato et al., 2010)
Analogy making
Invariance

aperture feature similarities

image similarities

Roland Memisevic (Frankfurt, Montreal)  Multiview Feature Learning  Tutorial at IPAM 2012
1. Introduction
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning

2. Learning relational features
   - Encoding relations
   - Learning

3. Factorization, eigen-spaces and complex cells
   - Factorization
   - Eigen-spaces, energy models, complex cells

4. Applications and extensions
   - Applications and extensions
   - Conclusions
Outline

1 Introduction
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning

2 Learning relational features
   - Encoding relations
   - Learning

3 Factorization, eigen-spaces and complex cells
   - Factorization
   - Eigen-spaces, energy models, complex cells

4 Applications and extensions
   - Applications and extensions
   - Conclusions
Sparse coding of images pairs?

How to extend sparse coding to model relations?

Sparse coding on the concatenation?
Sparse coding of images pairs?

How to extend sparse coding to model relations?
Sparse coding on the *concatenation*?
Sparse coding on the concatenation?

- A case study: Translations of binary, one-d images.
- Suppose images are random and can change in one of three ways:

Example Image $\mathbf{x}$:

```
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
```

Possible Image $\mathbf{y}$:

```
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
```

Roland Memisevic (Frankfurt, Montreal)  Multiview Feature Learning  Tutorial at IPAM 2012  35 / 163
Sparse coding on the concatenation?

- A hidden variable that collects evidence for a shift to the right.
- What if the images are random or noisy?
- Can we pool over more than one pixel?
Sparse coding on the concatenation?

- A hidden variable that collects evidence for a shift to the right.
- What if the images are random or noisy?
- Can we pool over more than one pixel?
A hidden variable that collects evidence for a shift to the right.
What if the images are random or noisy?
Can we pool over more than one pixel?
Sparse coding on the concatenation?

- Obviously not, because now the hidden unit would get equally happy if it would see the non-shift (second pixel from the left).
- The problem: Hidden variables act like OR-gates, that accumulate evidence.
Cross-products

- Intuitively, what we need instead are logical ANDs, which can represent *coincidences* (e.g., Zetzsche et al., 2003, 2005).
- This amounts to using the outer product $L := \text{outer}(x, y)$:

\[
\begin{array}{ccc}
\times \\
\hline
\end{array}
\]

- We can unroll this matrix, and let this be the data:
Each component $L_{ij}$ of the outer-product matrix will constitute evidence for exactly \textit{one} type of shift.

Hiddens pool over products of pixels.
Cross-products

- Each component $L_{ij}$ of the outer-product matrix will constitute evidence for exactly one type of shift.
- Hidden pools over products of pixels.
Each component $L_{ij}$ of the outer-product matrix will constitute evidence for exactly one type of shift.

Hiddens pool over products of pixels.
A different view: Families of manifolds

Feature learning reveals the (local) manifold structure in data.

When $y$ is a transformed version of $x$, we can still think of $y$ as being confined to a manifold, but it will be a conditional manifold.

Idea: Learn a model for $y$, but let parameters be a function of $x$. 

Roland Memisevic (Frankfurt, Montreal)
Conditional inference

Inferring $z$

- If we use a linear function, $w_{jk}(\mathbf{x}) = \sum_i w_{ijk}x_i$, we get

$$z_k = \sum_j w_{jk}y_j = \sum_j \left( \sum_i w_{ijk}x_i \right)y_j = \sum_{ij} w_{ijk}x_i y_j$$

- Inference via \textbf{bilinear} function of the inputs.
Conditional inference

To infer $y$:

$$y_j = \sum_k w_{jk} z_k = \sum_k \left( \sum_i w_{ijk} x_i \right) z_k = \sum_{ik} w_{ijk} x_i z_k$$

Inference via **bilinear** function of $x$, $z$. 
This is feature learning with input-dependent weights.
Input pixels can vote for features in the output image.
A hidden can blend in one slice $W_{xk}$ of the parameter tensor.

A slice does linear regression in “pixel space”.

So for binary hiddens, this is a mixture of $2^K$ image warps.
1. Introduction
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning

2. Learning relational features
   - Encoding relations
   - Learning

3. Factorization, eigen-spaces and complex cells
   - Factorization
   - Eigen-spaces, energy models, complex cells

4. Applications and extensions
   - Applications and extensions
   - Conclusions
Learning is predictive coding

Predictive sparse coding

- The cost for a training pair \((x, y)\) is:

\[
\sum_j (y_j - \sum_{ik} w_{ijk} x_i z_k)^2
\]

- Training as usual: Infer \(z\), update \(W\). (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007), (Memisevic, Hinton; 2007)
Example: Gated Boltzmann machine

Three-way RBM (Memisevic, Hinton; 2007)

\[
E(x, y, z) = \sum_{ijk} w_{ijk} x_i y_j z_k
\]

\[
p(y, z|x) = \frac{1}{Z(x)} \exp \left( E(x, y, z) \right),
Z(x) = \sum_{y,z} \exp \left( E(x, y, z) \right)
\]
Example: Gated Boltzmann machine

Three-way RBM (Memisevic, Hinton; 2007)

\[ E(x, y, z) = \sum_{ijk} w_{ijk} x_i y_j z_k \]

\[ p(y, z|x) = \frac{1}{Z(x)} \exp \left( E(x, y, z) \right), \quad Z(x) = \sum_{y,z} \exp \left( E(x, y, z) \right) \]
Example: Gated Boltzmann machine

Three-way RBM (Memisevic, Hinton; 2007)

\[
p(z_k|x, y) = \text{sigmoid}(\sum_{ij} W_{ijk} x_i y_j)
\]

\[
p(y_j|x, z) = \text{sigmoid}(\sum_{ik} W_{ijk} x_i z_k)
\]
Gated autoencoders

- Turn encoder and decoder weights into functions of \( x \).
- Learning the same as in a standard auto-encoder for \( y \).
- The model is still a DAG, so back-prop works exactly like in a standard autoencoder. (Memisevic, 2011)
Toy example: Conditionally trained “Hidden flow-fields”
Toy example: Conditionally trained “Hidden flow-fields”, inhibitory connections
Toy example: Learning optical flow

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>x&lt;sup&gt;test&lt;/sup&gt;</th>
<th>y(x&lt;sup&gt;test&lt;/sup&gt;, z)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
<td><img src="image25.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Combinatorial flowfields

\[ x \quad y \quad z \quad x^{\text{test}} \quad y(x^{\text{test}}, z) \]
Conditional training makes it hard to answer questions like:

“How likely are the given images transforms of one another?”

To answer questions like these, we require a joint image model, $p(x, y | z)$, given mapping units.
Joint training

\[ E(x, y, z) = \sum_{ijk} w_{ijk} x_i y_j z_k \]

\[ p(x, y, z) = \frac{1}{Z} \exp \left( E(x, y, z) \right) \]

\[ Z = \sum_{x,y,z} \exp \left( E(x, y, z) \right) \]

- Use three-way sampling in a Gated Boltzmann Machine (Susskind et al., 2011).
- Can apply this to matching tasks (more later).
Joint training

$$E(x, y, z) = \sum_{ijk} w_{ijk} x_i y_j z_k$$

$$p(x, y, z) = \frac{1}{Z} \exp \left( E(x, y, z) \right)$$

$$Z = \sum_{x, y, z} \exp \left( E(x, y, z) \right)$$

- Use three-way sampling in a Gated Boltzmann Machine (Susskind et al., 2011).
- Can apply this to matching tasks (more later).
For the autoencoder we can use a simple hack:

Add up two conditional costs:

\[
\sum_j (y_j - \sum_{ik} w_{ijk} x_i z_k)^2 + \sum_i (x_i - \sum_{jk} w_{ijk} y_j z_k)^2
\]

Force parameters to transform in both directions.
For the autoencoder we can use a simple hack:

Add up two conditional costs:

$$\sum_j (y_j - \sum_{ik} w_{ijk} x_i z_k)^2 + \sum_i (x_i - \sum_{jk} w_{ijk} y_j z_k)^2$$

Force parameters to transform in both directions.
Pool over products

Take-home message

To gather evidence for a transformation, let hidden units compute the sum over products of input components.