Deep Learning

A journey from feature extraction and engineering to end-to-end pipelines

Part 1: Introduction, Computer Vision

Andrei Bursuc

With slides from A. Karpathy, F. Fleuret, J. Johnson, S. Yeung, G. Louppe, Y. Avrithis ...
Deep Learning - the hype?
Deep Learning - the hype?

- Evolution of ImageNet large scale visual recognition challenge
- 1.2 M training images with 1K object categories
Deep Learning - the hype?

- Evolution of ImageNet large scale visual recognition challenge
- 1.2 M training images with 1K object categories

ImageNet experiments
Deep Learning - the hype?

Conference attendance growth
Deep Learning - the hype?

Conference attendance growth

[Chart showing the growth of NIPS registrations compared to world population]
Deep Learning - the hype?

CVPR 2017 sponsors
Deep Learning - the hype?

Industry participation
Deep Learning - the hype?

1/2 parallel session at NIPS 2017
Deep Learning - the hype?

Poster session at NIPS 2017
Deep Learning - the hype?

Primary topic in submissions at NIPS 2017
Deep Learning - the hype?

Primary topic in submissions at NIPS 2017
Deep Learning - the hype?

Primary topic in submissions at NIPS 2017
Deep Learning - the hype?

Other remarkable changes

- Paper publishing is more intense: papers are released on arXiv right after submission deadline
- Results of papers can be already outperformed by the time of the conference
- Code and/or trained networks are released with paper most of the times
- High number of published datasets
- Contributions arrive also from non computer vision / machine learning classic domains: genomics, mechanics.
Domain applications of Deep Learning?

Speech-to-Text

[Baidu 2014]
Domain applications of Deep Learning?

Computer Vision

[Krizhevsky 2012]

[Ciresan et al. 2013]

[Faster R-CNN - Ren 2015]

[NVIDIA dev blog]
Domain applications of Deep Learning?

Computer Vision

[Stanford 2017]

[FaceNet - Google 2015]

[Nvidia Dev Blog 2017]

[Facial landmark detection CUHK 2014]
Domain applications of Deep Learning?

NLP
Domain applications of Deep Learning?

NLP

Hey, Wynton Marsalis is playing this weekend. Do you have a preference between Saturday and Sunday?

[Screenshot of conversation]

[Amazon Echo / Alexa]
Domain applications of Deep Learning?

Vision + NLP
Domain applications of Deep Learning?

Generative models

Sampled celebrities [Nvidia 2017]
Domain applications of Deep Learning?

Generative models

<table>
<thead>
<tr>
<th>Text description</th>
<th>Stage-I images</th>
<th>Stage-II images</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird is blue with white and has a very short beak</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>This bird has wings that are brown and has a yellow belly</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>A white bird with a black crown and yellow beak</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>This bird is white, black, and brown in color, with a brown beak</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>The bird has small beak, with reddish brown crown and gray belly</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>This is a small, black bird with a white breast and white on the wingbars.</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>This bird is white black and yellow in color, with a short black beak</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
</tr>
</tbody>
</table>

StackGAN v2 [Zhang 2017]
Domain applications of Deep Learning?

Image translation
Domain applications of Deep Learning?

Generative models

Sound generation with WaveNet [DeepMind 2017]
Domain applications of Deep Learning?

Generative models

Sound generation with WaveNet [DeepMind 2017]

Guess which one is generated?

Tacotron 2 Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions, 2017
DL in other sciences
DL in other sciences

[Deep Genomics 2017]
DL in other sciences

[Deep Genomics 2017]

MakeAGIF.com
DL for AI in games
DL for AI in games

AlphaGo/Zero: Monte Carlo Tree Search, Deep Reinforcement Learning, self-play
What is Deep Learning?

- Neural Networks with more layers/modules
- Non-linear, hierarchical, abstract representations of data
- Flexible models with any input/output size
- Differentiable functional programming
What is Deep Learning?

In other words: a graph of tensor operators taking advantage of:

- the chain rule (back-propagation),
- stochastic gradient descent,
- convolutions,
- parallel operations on GPU

We kind of had most of it in the networks from long ago
Why going deep?

- Traditional recognition: "shallow" architecture
  - Each block is designed and implemented individually

- "Deep" architecture (Convolutional Neural Network)
Why going deep?

Graph of tensors where blocks are trained and optimized jointly

- 1 - 140M trainable parameters
Why Deep Learning works now?

- Five decades of research in machine learning
- Computing and storage power
- Lots of (labelled) data from the internet
- Tools and culture of collaborative and reproducible science
- Resources and efforts from large companies
Why Deep Learning works now?

Five decades of research in ML provided:

- a taxonomy of ML concepts (classification, generative models, clustering, kernels, linear embeddings, etc.),
- a sound statistical formalization (Bayesian estimation, PAC),
- a clear picture of fundamental issues (bias/variance dilemma, VC dimension, generalization bounds, etc.),
- a good understanding of optimization issues,
- efficient large-scale algorithms.
Why Deep Learning works now?

From a practical perspective, deep learning:

- lessens the need for a deep mathematical grasp,
- makes the design of large learning architectures a system/software development task,
- allows to leverage modern hardware (clusters of GPUs),
- does not plateau when using more data,
- makes large trained networks a commodity.
Why Deep Learning works now?

Evolution in computer vision datasets

<table>
<thead>
<tr>
<th>Data-set</th>
<th>Year</th>
<th>Nb. images</th>
<th>Resolution</th>
<th>Nb. classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>1998</td>
<td>$6.0 \times 10^6$</td>
<td>$28 \times 28$</td>
<td>10</td>
</tr>
<tr>
<td>NORB</td>
<td>2004</td>
<td>$4.8 \times 10^4$</td>
<td>$96 \times 96$</td>
<td>5</td>
</tr>
<tr>
<td>Caltech 101</td>
<td>2003</td>
<td>$9.1 \times 10^7$</td>
<td>$\approx 300 \times 200$</td>
<td>101</td>
</tr>
<tr>
<td>Caltech 256</td>
<td>2007</td>
<td>$3.0 \times 10^4$</td>
<td>$\approx 640 \times 480$</td>
<td>256</td>
</tr>
<tr>
<td>LFW</td>
<td>2007</td>
<td>$1.3 \times 10^6$</td>
<td>$250 \times 250$</td>
<td>-</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>2009</td>
<td>$6.0 \times 10^4$</td>
<td>$32 \times 32$</td>
<td>10</td>
</tr>
<tr>
<td>PASCAL VOC</td>
<td>2012</td>
<td>$2.1 \times 10^6$</td>
<td>$\approx 500 \times 400$</td>
<td>20</td>
</tr>
<tr>
<td>MS-COCO</td>
<td>2015</td>
<td>$2.0 \times 10^5$</td>
<td>$\approx 640 \times 480$</td>
<td>91</td>
</tr>
<tr>
<td>ImageNet</td>
<td>2016</td>
<td>$14.2 \times 10^6$</td>
<td>$\approx 500 \times 400$</td>
<td>21,841</td>
</tr>
<tr>
<td>Cityscape</td>
<td>2016</td>
<td>$25 \times 10^4$</td>
<td>$2,000 \times 1000$</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure credit: F. Fleuret, EE-559 Deep learning
Why Deep Learning works now?

When more data is available
Why Deep Learning works now?

- Many deep learning frameworks freely available as open source
- Frequent changes and updates (every few weeks)
- Most frameworks supported by a large company
Deep Learning - the hype?

- Many deep learning frameworks freely available as open source
- Frequent changes and updates (every few weeks)
- Most frameworks supported by a GAFA company
"PyTorch is a python package that provides two high-level features:

- Tensor computation (like numpy) with strong GPU acceleration
- Deep Neural Networks built on a tape-based autograd system

You can reuse your favorite python packages such as numpy, scipy and Cython to extend PyTorch when needed."
PyTorch

MNIST dataset

28 × 28 grayscale images, 60k train samples, 10k test samples
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=3))
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=2))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std()
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_target.cuda()

for e in range(10):
    for b in range(0, nb_train_samples, bs):
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=3))
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=2))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std()
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_target.cuda()

for e in range(10):
    for b in range(0, nb_train_samples, bs):
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=3))
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=2))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std()
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_target.cuda()

for e in range(10):
    for b in range(0, nb_train_samples, bs):
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=3))
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=2))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std()
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_target.cuda()

for e in range(10):
    for b in range(0, nb_train_samples, bs):
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=3))
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=2))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std()
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_target.cuda()

for e in range(10):
    for b in range(0, nb_train_samples, bs):
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), kernel_size=3))
        x = F.relu(F.max_pool2d(self.conv2(x), kernel_size=2))
        x = x.view(-1, 256)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = Net()

mu, std = train_input.data.mean(), train_input.data.std()
train_input.data.sub_(mu).div_(std)
optimizer = optim.SGD(model.parameters, lr=1e-1)
criterion, batch_size = nn.CrossEntropyLoss(), 100

model.cuda()
criterion.cuda()
train_input, train_target = train_input.cuda(), train_target.cuda()

for e in range(10):
    for b in range(0, nb_train_samples, bs):
Outline

1. Computer Vision
   ○ before things went deep: handcrafted features

2. Neural networks]
   ○ empirical risk minimization, stochastic gradient descent, backpropagation
   ○ multilayer perceptrons

3. Going deeper
   ○ convolutional layers, deep regularization, deep architectures

4. Under the hood
   ○ understanding and visualizing CNNs, adversarial attacks
   ○ CPU vs GPU, using CNNs in practice

5. Unsupervised and self-supervised learning
   ○ generative models: autoencoders, variational autoencoders, generative adversarial networks
   ○ self-supervised learning
Outline

1. Computer Vision
   - before things went deep: handcrafted features

2. Neural networks
   - empirical risk minimization, stochastic gradient descent, backpropagation
   - multilayer perceptrons

3. Going deeper
   - convolutional layers, deep regularization, deep architectures

4. Under the hood
   - understanding and visualizing CNNs, adversarial attacks
   - CPU vs GPU, using CNNs in practice

5. Unsupervised and self-supervised learning
   - generative models: autoencoders, variational autoencoders, generative adversarial networks
   - self-supervised learning
Computer Vision

The old-school way
The problem with Computer Vision

The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:

This is a mushroom.
This is a mushroom.
This is a mushroom.
This is known as the **semantic gap**. Extracting semantic information requires models of high complexity.
BIG collections of images and videos

1-100 million images
Thousands hours of video

Figure credit: H. Jégou
Find all images that depict visual content similar to the query image. (same scene, location, etc.)
Find all images that depict visual content similar to the query (same scene, location, etc)

→ image retrieval
Find some particular object for which you have an example

→ particular object retrieval
Find all images where an object of a "class" appear:

Find all "trees"

→ image classification
Image retrieval challenges
Image retrieval challenges

- scale
- viewpoint
- occlusion
- clutter
- lighting
- distinctiveness
- distractors
Image classification challenges

- scale
- viewpoint
- occlusion
- clutter
- lighting
- number of instances
- texture/color
- pose
- deformability
- intra-class variability
Visual descriptors
Visual descriptors

Pre-deep pipeline
Visual descriptors

Concatenation of pixels into 1D descriptors
Global descriptors

Concatenation of pixels into 1D descriptors

- face recognition

- digit recognition
Global descriptors

Tiny images

- resize images to $32 \times 32$ pixels ($3072d$ vectors)

- high speed, limited accuracy
- used for scene recognition
Global descriptors

Color histogram

- Histogram is a summary of the data describing image statistics (here color)
Global descriptors

Color histogram

- Histogram is a summary of the data describing image statistics (here color)
Global descriptors

Color histogram

- The problem with histograms
Global descriptors

Color histogram

- The problem with histograms
Global descriptors

- sampling scheme adapted to power spectrum statistics
- filtering and global pooling in frequency domain
Global descriptors

- Gabor filters: inspired from neuroscience and related to tiny salient pattern for the human vision system
- filters are pre-defined and manually computed
Global descriptors

The gist descriptor

- apply filter bank to entire image in frequency domain
- partition image in $4 \times 4$ cells
- average pooling of filter responses per cell

*Building the Gist of a Scene: the Role of Global Image Features in Recognition; Oliva and Torralba; VP 2006*
Global descriptors

gist pipeline

- 3-channel RGB input $\rightarrow$ 1-channel gray-scale
Global descriptors

gist pipeline

- 3-channel RGB input $\rightarrow$ 1-channel gray-scale
- apply filters at 4 scales $\times$ 8 orientations

*Building the Gist of a Scene: the Role of Global Image Features in Recognition; Oliva and Torralba; VP 2006*

Slide credit: Y. Avrithis, SIF Deep Learning for Vision
Global descriptors

gist pipeline

- 3-channel RGB input $\rightarrow$ 1-channel gray-scale
- apply filters at 4 scales $\times$ 8 orientations
- average pooling on $4 \times 4$ cells $\rightarrow$ descriptor of length 512

*Building the Gist of a Scene: the Role of Global Image Features in Recognition; Oliva and Torralba; VP 2006*

Slide credit: Y. Avrithis, SIF Deep Learning for Vision
Local descriptors

scale-invariant feature transform (SIFT)

- detect a sparse set of "stable" features (rectangular patches) equivariant to translation, scale and rotation
- for each patch:
  - normalize with respect to scale and orientation
  - construct a histogram of gradient orientations

Object recognition from local scale-invariant features.; Lowe; ICCV 1999
Local descriptors

scale-invariant feature transform (SIFT)

- votes in 8-bin orientation histograms weighted by magnitude and by weighted by a Gaussian window,
- histograms pooled over $4 \times 4$ cells,
- 128-dimensional descriptor, normalized, clipped at 0.2, normalized

*Object recognition from local scale-invariant features*; Lowe; ICCV 1999
Local descriptors

Histogram of Oriented Gradients (HoG)

- applied to person detection by sliding window and SVM classifier learns positive and negative weights on positions and orientations
- switch focus back to dense features for classification

Histogram of Oriented Gradients for Human Detection; Dalal and Triggs; CVPR 2005
Local descriptors

HOG descriptor

- applied densely to adjacent cells of $8 \times 8$ pixels
- no scale or orientation normalization; only single-scale
- normalized by overlapping blocks of $3 \times 3$ cells -- redundant

*Histogram of Oriented Gradients for Human Detection; Dalal and Triggs; CVPR 2005*
Local descriptors

SIFT/HOG pipeline

- 3-channel patch (image) RGB input $\rightarrow$ 1-channel gray-scale
Local descriptors

SIFT/HOG pipeline

- 3-channel patch (image) RGB input $\rightarrow$ 1-channel gray-scale
- compute gradient magnitude and orientation
Local descriptors

SIFT/HOG pipeline

- 3-channel patch (image) RGB input → 1-channel gray-scale
- compute gradient magnitude and orientation
- encode into $b = 8$ orientation bins
Local descriptors

SIFT/HOG pipeline

- 3-channel patch (image) RGB input $\rightarrow$ 1-channel gray-scale
- compute gradient magnitude and orientation
- encode into $b = 8(9)$ orientation bins
- average pooling on $c = 4 \times 4$ cells
- descriptor of length $c \times b = 128$

Slide credit: Y. Avrithis, SIF Deep Learning for Vision
Local descriptors

- matching everything with everything

Figure credit: A. Vedaldi
Local descriptors

Exhaustive matching

Step 0: get an image pair
Local descriptors

Exhaustive matching

*Step 1: detect local features $f$ and extract descriptors $d$*
Local descriptors

Step 2: match each descriptor to its closest one

number of matches: 2048
Local descriptors

Step 3: reject ambiguous matches using the 2nd-nn test

number of matches: 293
Local descriptors

- the final step is to test whether matches are consistent with an overall image transformation
- inconsistent matches are rejected
From image matching to image search

- This matching strategy can be used to search a few images exhaustively
- However this is far too slow to search a large database
- Example:

  - $L$ images in the database
  - $N$ features per image (incl. query)
  - $D$ dimensional feature descriptor
  - Exhaustive search cost: $O(N^2 L D)$
  - Memory footprint: $O(NLD)$

  - e.g. $10^6 - 10^{10}$ (FaceBook)
  - e.g. $10^3$ (~ SIFT detector)
  - e.g. $10^2$ (~ SIFT descriptor)

  - $10^{11} - 10^{15}$ ops = 100 days - 300 years
  - 1TB - 1PB
Visual words

- Visual descriptors
  - descriptor $d$
  - $k = \pi(d)$

- Visual words
  - visual word $k$
  - visual dictionary
    - $\in \{ \text{red, blue, purple, ...} \}$

- Continuous space
  - E.g. 128D for SIFT

- Discrete space
  - $K$ elements

Figure credit: A. Vedaldi
Visual words

- Dictionary is typically learned using k-means
- Value of $k$ depends on the task: from 8 to $16M$
Visual words

- Visual word examples: each row is an equivalence class of patches mapped to the same cluster by k-means
- Visual words = iconic image fragments

Figure credit: A. Vedaldi
Visual words

Quantization

Figure credit: A. Vedaldi
Visual words

- Two steps:
  - **Extraction**: extract local features and compute corresponding descriptors
  - **Quantization**: map the descriptors to k-means cluster centroids to obtain the corresponding visual words

Figure credit: A. Vedaldi
Histogram of visual words

- A simple but efficient global image descriptor
- Vector of the number of occurrences of the $k$ visual words in the image
- If there are $k$ visual words, then $h \in \mathbb{R}^k$
- The vector $h$ is a global image descriptor
Histogram of visual words

- This is also called a **bag of (visual) words** - BOW because it does not remember the relative positions of the features, just the number of occurrences
- *h* discards spatial information
- **Pros**: more invariant to viewpoint changes and other nuisance factors
- **Cons**: less discriminative
Histogram of visual words

Intuition
Global descriptors

Bag-of-Words pipeline

- 3-channel patch RGB input $\rightarrow$ 1-channel gray-scale
Global descriptors

Bag-of-Words pipeline

- 3-channel patch RGB input → 1-channel gray-scale
- set of ~1000 features × 128-dim SIFT descriptors
Global descriptors

Bag-of-Words pipeline

- 3-channel patch RGB input $\rightarrow$ 1-channel gray-scale
- set of $\sim$1000 features $\times$ 128-dim SIFT descriptors
- element-wise encoding of $k = 10^4$ visual words
Global descriptors

Bag-of-Words pipeline

- 3-channel patch RGB input → 1-channel gray-scale
- set of $\sim 1000$ features $\times$ 128-dim SIFT descriptors
- element-wise encoding of $k = 10^4$ visual words
- global sum pooling, $\ell^2$ normalization

Slide credit: Y. Avrithis, SIF Deep Learning for Vision
Linear predictor

\[ F(x) = \langle w, x \rangle \]

- e.g. \( F(x) \) can be linear SVM
Data representations

A linear predictor can be used to classify vector data. The question is how such a predictor can be applied to images, text, videos, or sounds.

This is solved by an encoder, which maps the data to a vectorial representation

\[ F(x) = \langle w, \Phi(x) \rangle \]
Evolution of representation learning across years
The Deep Learning promise: end-to-end trainable models

- input: raw image | output: prediction
- feature extraction, feature representations and classifiers are jointly learned

Figure credit: A. Vedaldi
Recap

- Huge variety of human-engineered features for visual representation and recognition
- Global descriptors: pixel vectors, color histogram, gist
- Local descriptors: SIFT, HoG
- Matching local descriptors
- *Bag of Visual Words*
Up next:

Neural Networks