# Computer Vision Systems Programming VO

## Deep Learning

Christopher Pramerdorfer

Computer Vision Lab, Vienna University of Technology

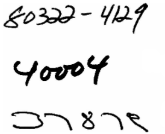
# Topics

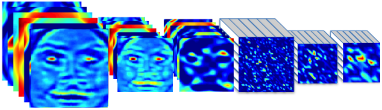
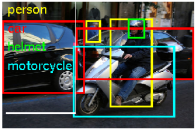
Deep learning motivation

Multilayer perceptrons

Convolutional neural networks

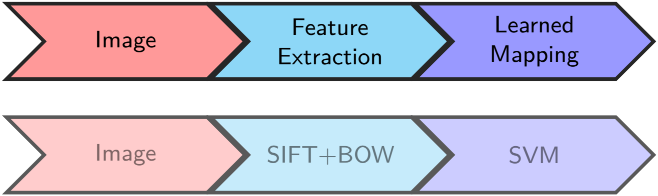
Deep learning applications




Images from LeCun et al. 1989, Taigman et al. 2013, `image-net.org`

# Object Recognition
## Traditional Approach



# Object Recognition
## Traditional Approach

Problem: how to choose the representation/features?

"General" features not optimal

- Not tuned to task at hand, low-level

Designing task-specific features is complex

- Virtually impossible to do optimally

Solution: learn representation as well

Learning high-level representations directly is difficult

Deep Learning (DL) solves this

- By learning a hierarchy of representations
- Layers in hierarchy build upon each other

# Object Recognition
## Deep Learning

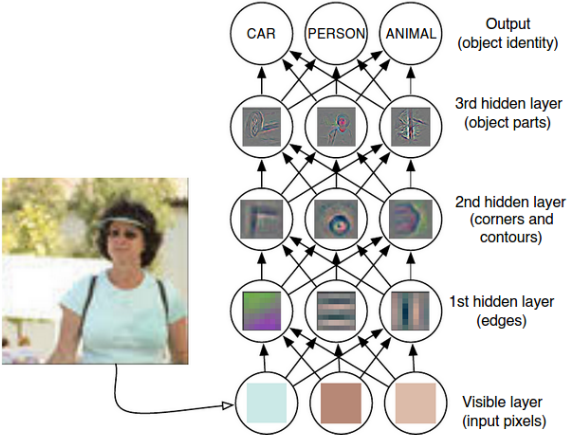
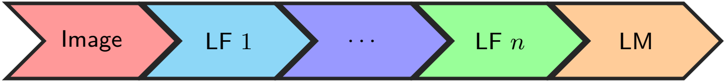



Image from Bengio, Goodfellow, and Courville 2014

$n$ levels of features/representations

Learned jointly with the output mapping



# Multilayer Perceptrons

DL is usually realized using MultiLayer Perceptrons (MLPs)

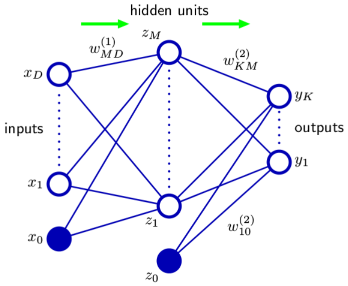



Image from Bishop 2006

Binary linear classifier

Feature vectors $\mathbf{x}$ classified as $f(\mathbf{w}^\top \mathbf{x} + b) \in \{-1, +1\}$

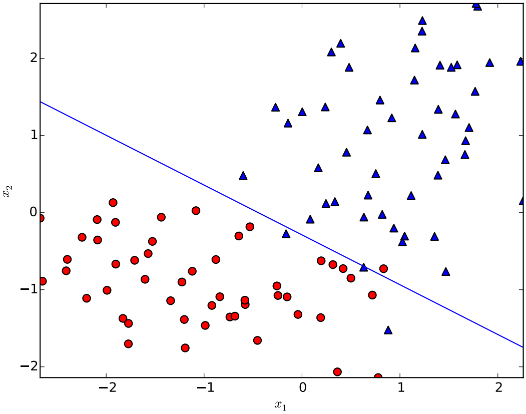
$f$ is a discontinuous step function

$$f(v) = \begin{cases} +1 & \text{if } v > 0 \\ -1 & \text{otherwise} \end{cases}$$

$\mathbf{w}, b$ learned from training data

# Multilayer Perceptrons

## The Perceptron



Only two classes

Linear decision boundaries

Learning never converges for non-separable data

# Multilayer Perceptrons

## Two-Layer Architecture

- Replace $f$ with continuous nonlinearity (e.g. $\tanh(\cdot)$)
- Introduce layer of $M$ such "Perceptrons" (hidden units)
- Hidden units connected to layer of $K$ output units

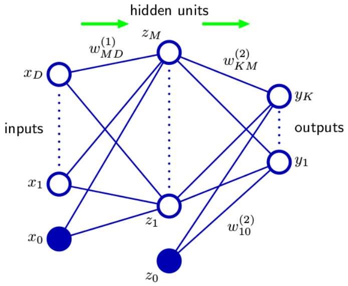



Image from Bishop 2006

# Multilayer Perceptrons
## Two-Layer Architecture

Output of $m$th hidden unit is $z_m(\mathbf{x}) = f(\mathbf{w}_m^\top \mathbf{x})$

Bias $b$ included in $\mathbf{w}$ and $\mathbf{x}$, $w_0 = b$, $x_0 = 1$

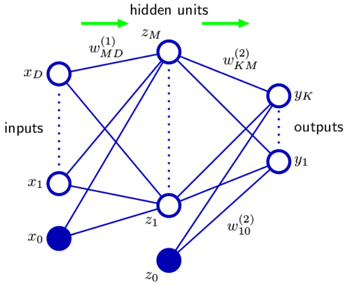



Image from Bishop 2006

# Multilayer Perceptrons
## Two-Layer Architecture

Output of $k$th output unit is $y_k(\mathbf{z}) = g(\mathbf{w}_k^\top \mathbf{z})$

Choice of $g$ depends on problem (regression, classification)

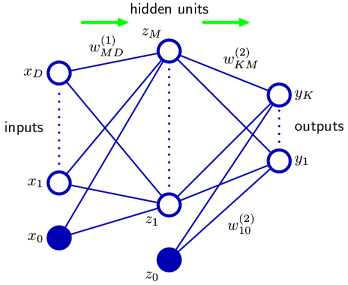



Image from Bishop 2006

Both $f$ and $g$ are differentiable

- Learn $\mathbf{w}$ using gradient descent
- Gradients evaluated via error backpropagation

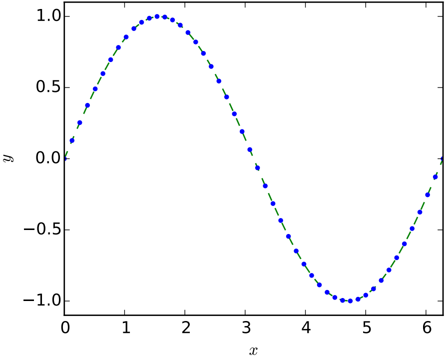
# The Pylearn2 Library

Machine learning library with focus on DL

Written in Python, but interaction mostly in YAML

Open-source: `https://github.com/lisa-lab/pylearn2`

Use pylearn2 to train a MLP for regression



# The Pylearn2 Library

## MLP Regression Example

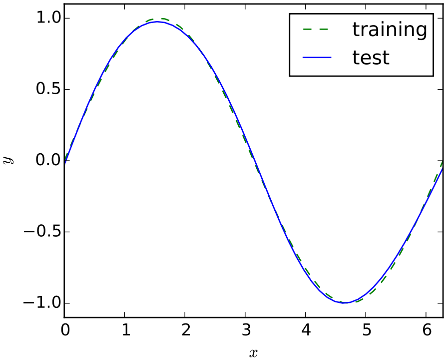
```
model: !obj:pylearn2.models.mlp.MLP {
    nvis: 1, # one input unit x
    layers: [ # two layers
        !obj:pylearn2.models.mlp.Tanh { # tanh activations for hidden units
            dim: 3, # use M=3 hidden units
            layer_name: 'hidden',
            irange: 0.1
        },
        !obj:pylearn2.models.mlp.Linear { # linear output layer for regression
            dim: 1, # one output unit, K=1
            layer_name: 'out',
            irange: 0.1
        }
    ]
}
```

# The Pylearn2 Library
## MLP Regression Example

```
# dataset contains the (x,y) pairs from the previous figure
dataset: &train !pkl: 'mlp_data_regression.pkl',
# train using batch gradient descent
algorithm: !obj:pylearn2.training_algorithms.bgd.BGD {
    conjugate: 1,
    batch_size: 50,
    line_search_mode: 'exhaustive',
    termination_criterion: !obj:pylearn2.termination_criteria.EpochCounter {
        max_epochs: 100 # train for 100 epochs
    }
}
```

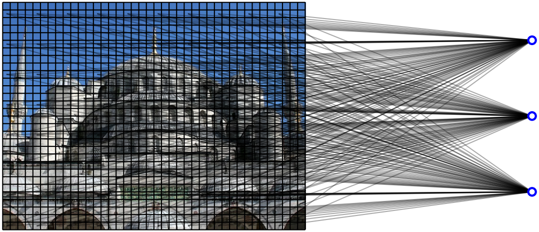
# The Pylearn2 Library
## MLP Regression Example

Full example: `https://github.com/cpra/cvsp-vo-slides`



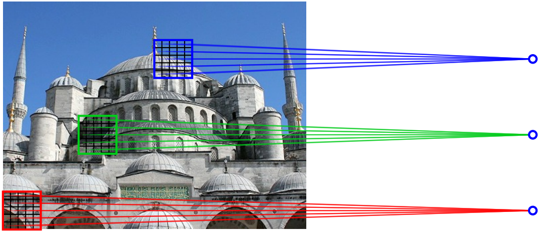
Above MLP architecture does not scale to images

- VGA resolution, $M = 1000$: $\sim 300$ million params



Nearby pixels are closely correlated

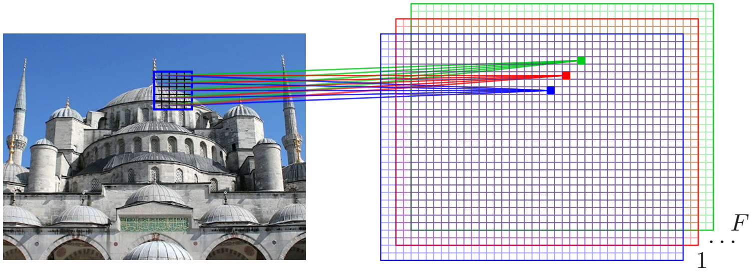
Exploit topology through locally connected layers



Convolutional layer consists of $F$ feature maps

Each map consists of $M$ hidden units with **shared** weights

Hidden units have a local receptive field



Number of params independent of image resolution and $M$

$F = 100$, $15 \times 15$ receptive field: $\sim 23,000$ params

Feature map evaluation equals

- Convolution with kernel $\mathbf{w}_f$ (hence the name)
- Followed by nonlinearity $f(\cdot) = \max(0, \cdot)$ (ReLU)

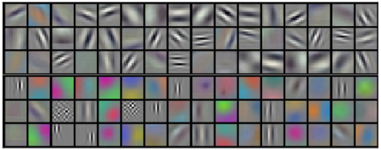
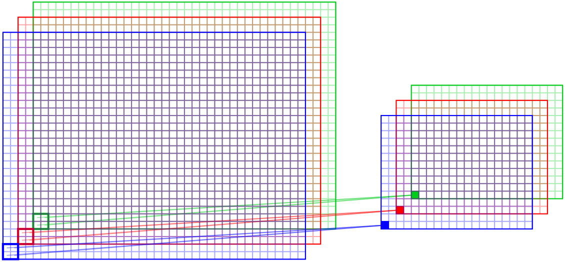


Image from Krizhevsky, Sutskever, and Hinton 2012

After convolutional layer

Hidden units pool information locally (e.g. max, avg)

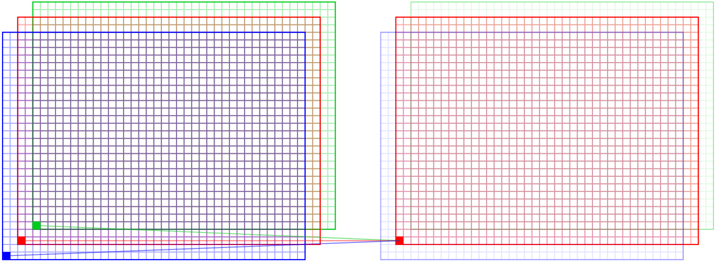
Data reduction, robustness to small translations



Between convolution and pooling layers

Normalize responses spatially or over adjacent feature maps

Produce more expressive, robust features



MLPs with above layers are Convolutional Neural Networks (CNNs)

Usually several Conv $\Rightarrow$ Norm $\Rightarrow$ Pooling blocks

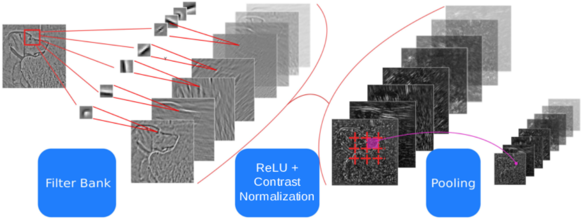



Image adapted from Kavukcuoglu 2011

Blocks followed by a traditional MLP

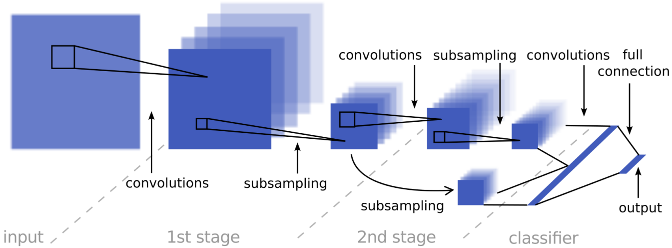



Image adapted from Sermanet et al. 2012

Zip code recognition from images

Among first applications of CNNs (1989!)

- DL and CNNs are "old" concepts
- Now successful due to available data & processing power

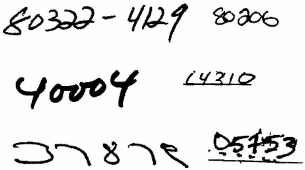



Image from LeCun et al. 1989

# Deep Learning Applications
## Zip Code Recognition

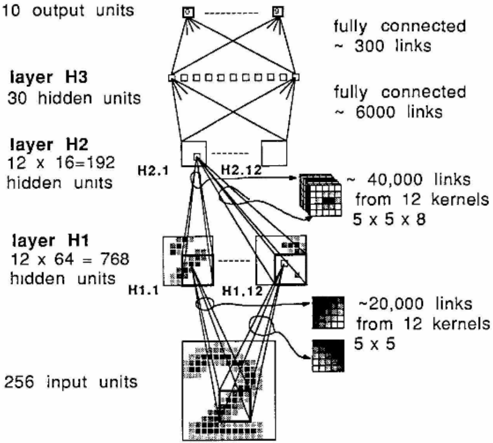



Image from LeCun et al. 1989

Let's try this ourselves

`http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html`

Image from `convnetjs.com`

# Deep Learning Applications
## Face Recognition

Large market (expected $\$6.5$B by 2018)

- Security, law enforcement, HCI, ...

Complicated task

- Pose, occlusions, aging, expressions, accessories

Image adapted from `http://vis-www.cs.umass.edu/lfw/`

Face recognition via 3D face frontalization and DL

- Conv ⇒ Pooling ⇒ Conv for low-level features
- Locally connected layers for high-level features
- $\sim 120$ million parameters, $\sim 4$ million training images

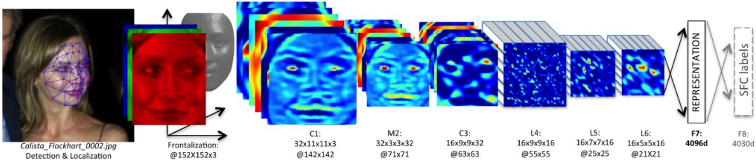



Image from Taigman et al. 2013

Human-level face verification performance

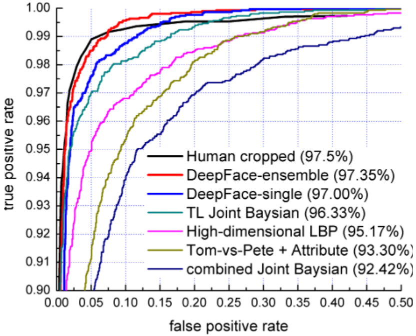
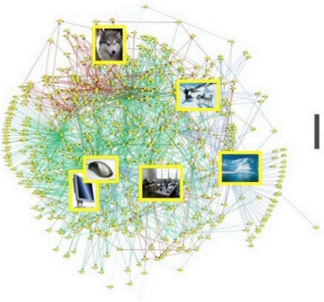



Image from Taigman et al. 2013

Object recognition on the ImageNet database

- $\sim 14$ million images categorized hierarchically
- Annual object recognition challenges



IMAGENET

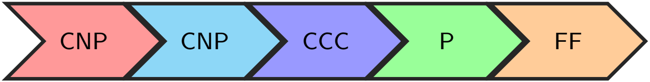
Image from `http://web.eecs.umich.edu/~jiadeng/`

# Deep Learning Applications
## Object Recognition – Krizhevsky, Sutskever, and Hinton 2012

Object classification and localization via DL

Won the 2012 challenge by a large margin

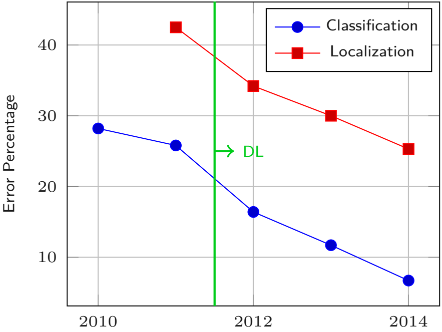
$\sim 60$ million parameters, $\sim 1.2$ million training images



Implemented using cuda-convnet

- `https://code.google.com/p/cuda-convnet/`
- Open-source C++/CUDA implementation of CNNs
- Network structure definitions available on the webpage

DL has lead to significant performance gains on ImageNet

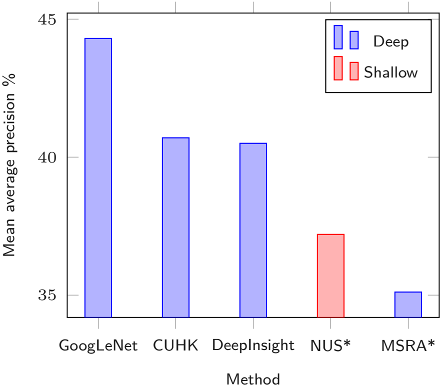


Online demo of 2013 winner of classification challenge

`http://www.clarifai.com/`

# Deep Learning Applications
## Object Recognition

Results of 2014 object detection challenge (excerpt)



Scene labeling

Action recognition

Speech recognition

...

# Remarks

DL and CNNs are very powerful concepts

- State of the art results in many areas of application

Training CNNs is complex

- Some CNNs are trained on GPUs for weeks
- Large training datasets required
- Reason why DL became popular only recently

Several free implementations available

- pylearn2, convnetjs, cuda-convnet, caffe, torch7, ...

# Bibliography I


Bengio, Yoshua, Ian Goodfellow, and Aaron Courville (2014). **Deep Learning (draft)**. MIT Press.

Bishop, Christopher (2006). **Pattern recognition and machine learning**. Springer.

Kavukcuoglu, Koray (2011). **Learning feature hierarchies for object recognition**. PhD thesis.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). **Imagenet classification with deep convolutional neural networks**. NIPS.

LeCun, Yann et al. (1989). **Backpropagation applied to handwritten zip code recognition**. Neural computation.

# Bibliography II


Sermanet, Pierre et al. (2012). **Pedestrian Detection with Unsupervised Multi-Stage Feature Learning**. CoRR.

Taigman, Yaniv et al. (2013). **Deepface: Closing the gap to human-level performance in face verification**. CVPR.