graphics and vision gravis
University of Basel

Deep Neural Networks

Pattern Recognition

Fall 2018

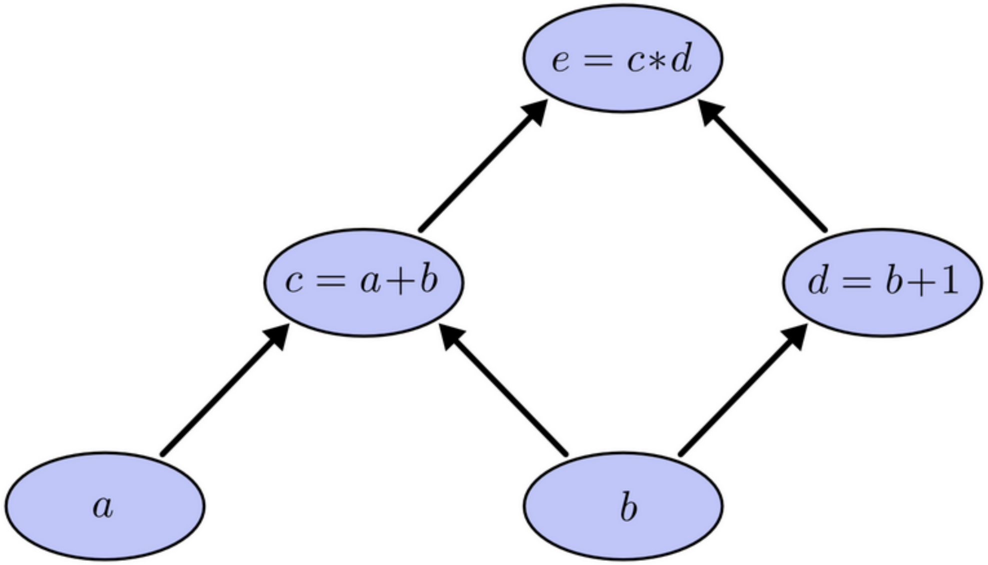
Adam Kortylewski

Overview

- **Backpropagation in Computational Graphs**
- Deep Neural Networks
  - From Perceptrons to Deep Neural Networks
  - High Level APIs
  - Optimization and Regularization
- Convolutional Neural Networks
  - Fundamental Properties of Images
  - Basic Architecture & Examples
- Applications
- Open Research Questions

Backpropagation in Computational Graphs

$$e = (a + b) * (b + 1)$$



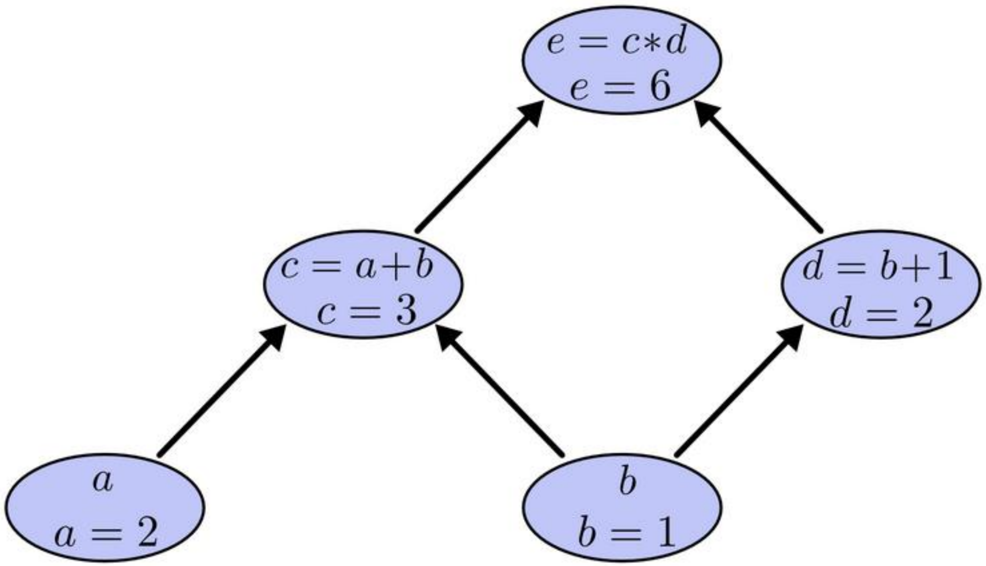
$$c = a + b$$

$$d = b + 1$$

$$e = c * d$$

Backpropagation in Computational Graphs

$$e = (a + b) * (b + 1)$$



$$c = a + b$$

$$d = b + 1$$

$$e = c * d$$

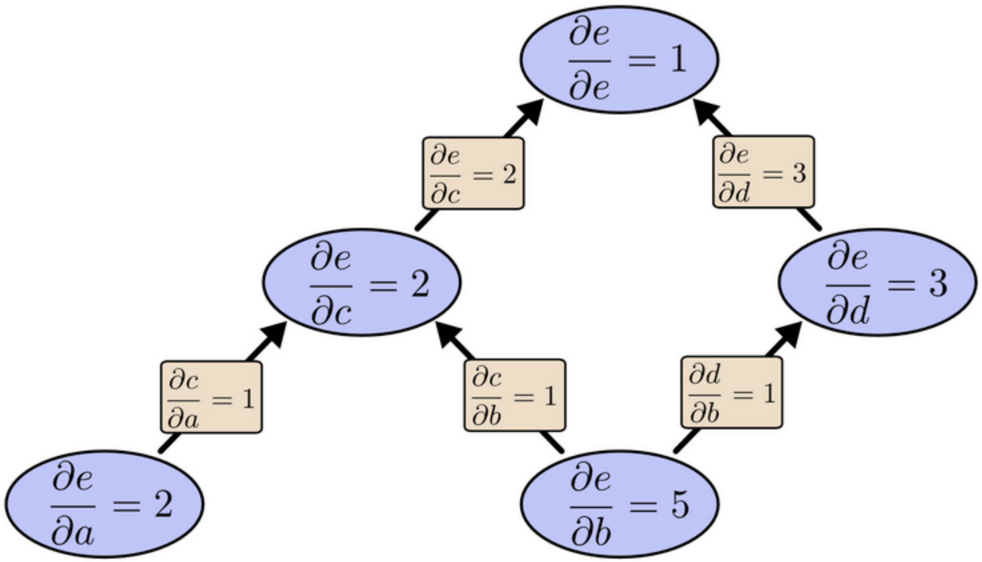
Backpropagation in Computational Graphs

$$e = (a+b) * (b+1)$$

$$c = a + b$$

$$d = b + 1$$

$$e = c * d$$



- Intermediate results need to be stored in order to compute the derivates

Automated differentiation with autograd

- Differentiating mathematical programs with *autograd* in numpy

```
import autograd.numpy as np   # Thinly-wrapped version of Numpy
from autograd import grad

def taylor_sine(x):  # Taylor approximation to sine function
    ans = currterm = x
    i = 0
    while np.abs(currterm) > 0.001:
        currterm = -currterm * x**2 / ((2 * i + 3) * (2 * i + 2))
        ans = ans + currterm
        i += 1
    return ans

grad_sine = grad(taylor_sine)
print "Gradient of sin(pi) is", grad_sine(np.pi)
```

- Automated differentiation is the basis for learning neural networks

Overview

- Backpropagation in Computational Graphs

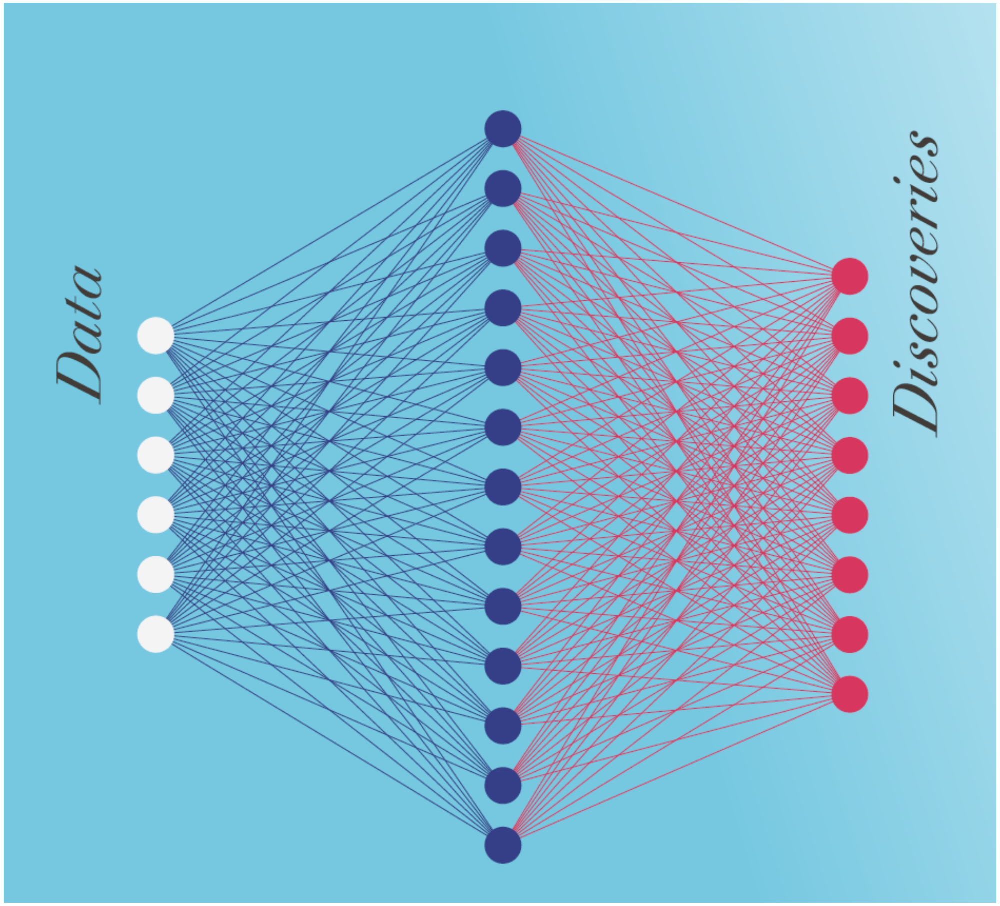
- **Deep Neural Networks**
  - From Perceptrons to Deep Neural Networks
  - High Level APIs
  - Optimization and Regularization

- Convolutional Neural Networks
  - Fundamental Properties of Images
  - Basic Architecture & Examples

- Applications

- Open Research Questions

Impact on Science



Target output y

Horizonte, Forschungsmagazin – June 2017

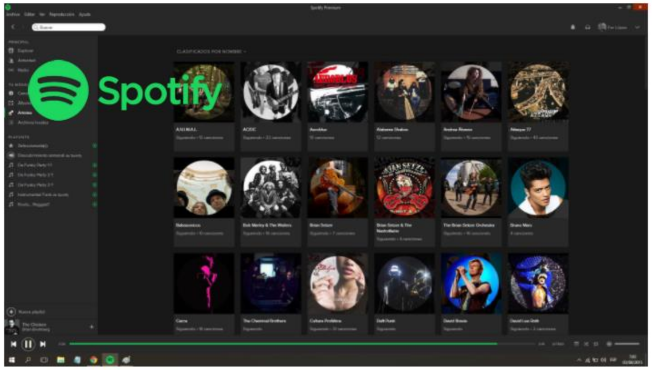
Speech and text analysis

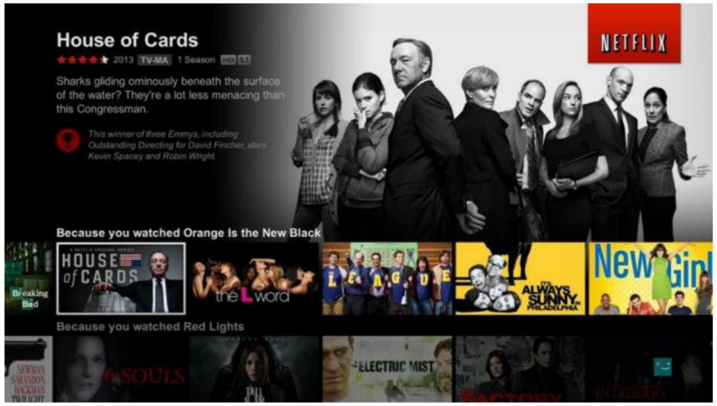
- Speech

Google Translate

- From images

Recommender Systems everywhere

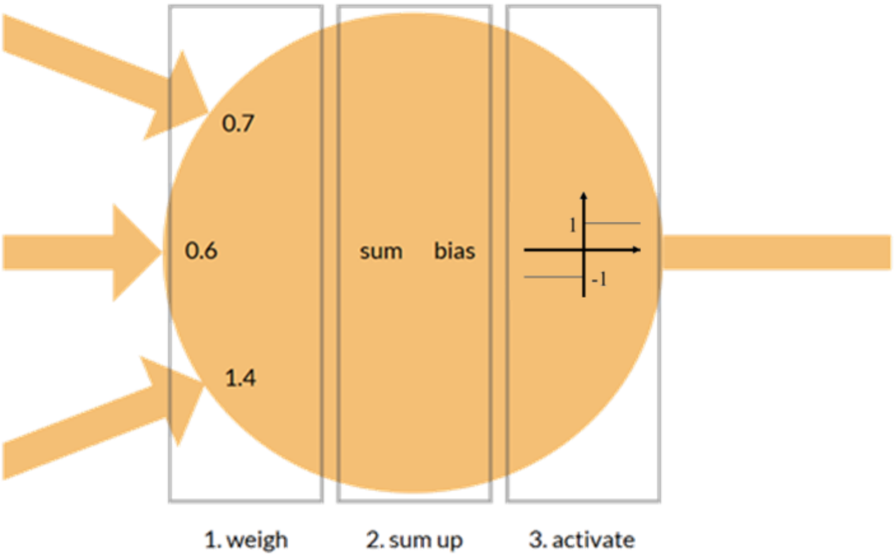




“Deep Learning for Recommender Systems: A Survey”, Ernesto Diaz-Aviles

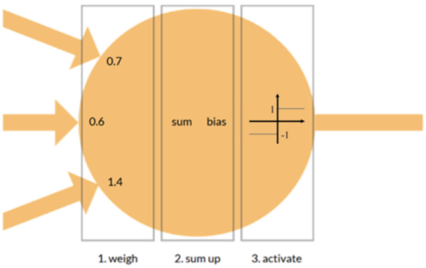
From Perceptrons to Deep Neural Networks

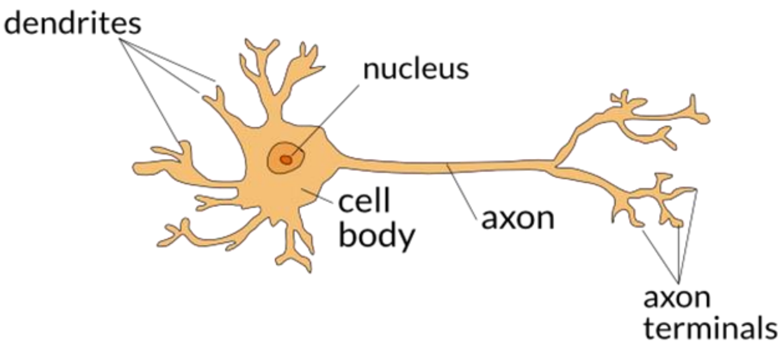
- Recap: The Perceptrons architecture



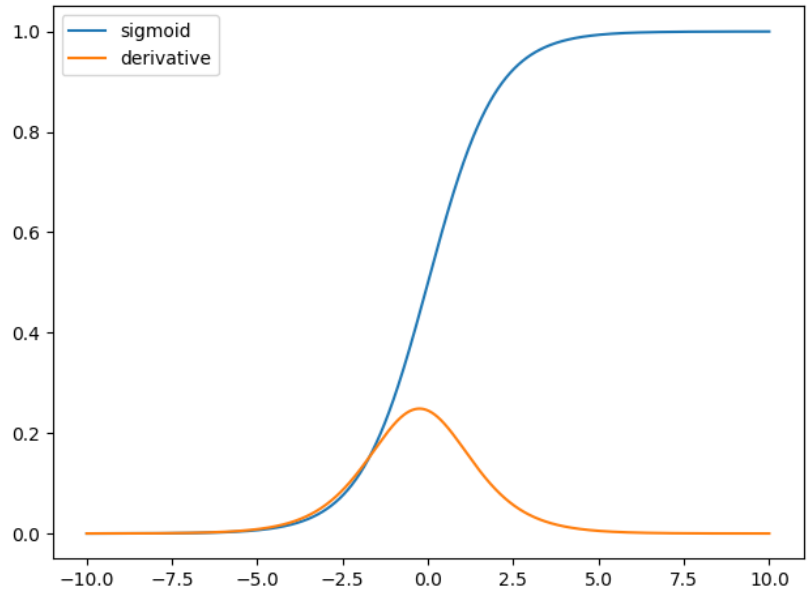
From Perceptrons to Deep Neural Networks

- Recap: The Perceptrons architecture
- Perceptrons are also referred to as "artificial neurons", highlighting the original inspiration from biological neurons



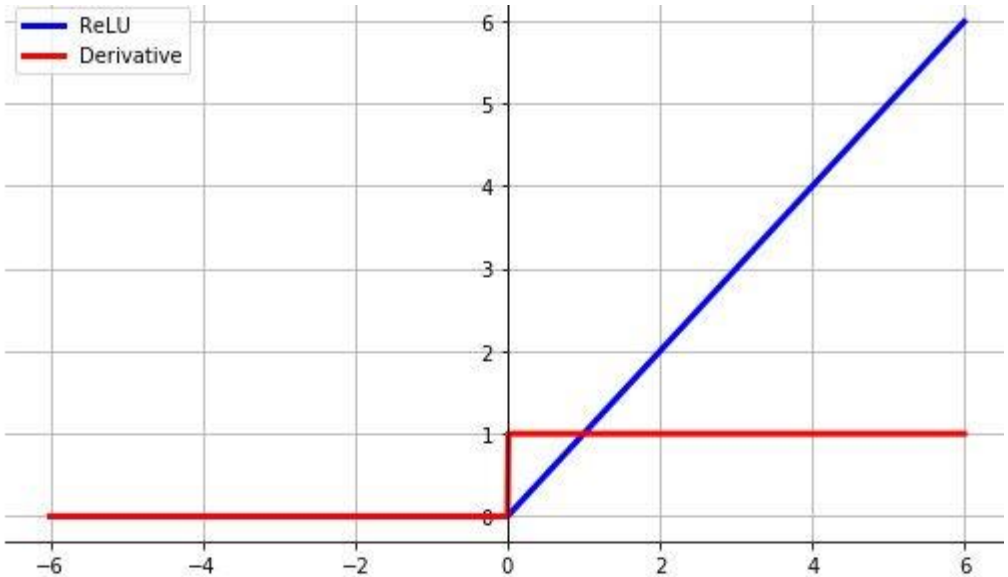


Activation functions: Sigmoid



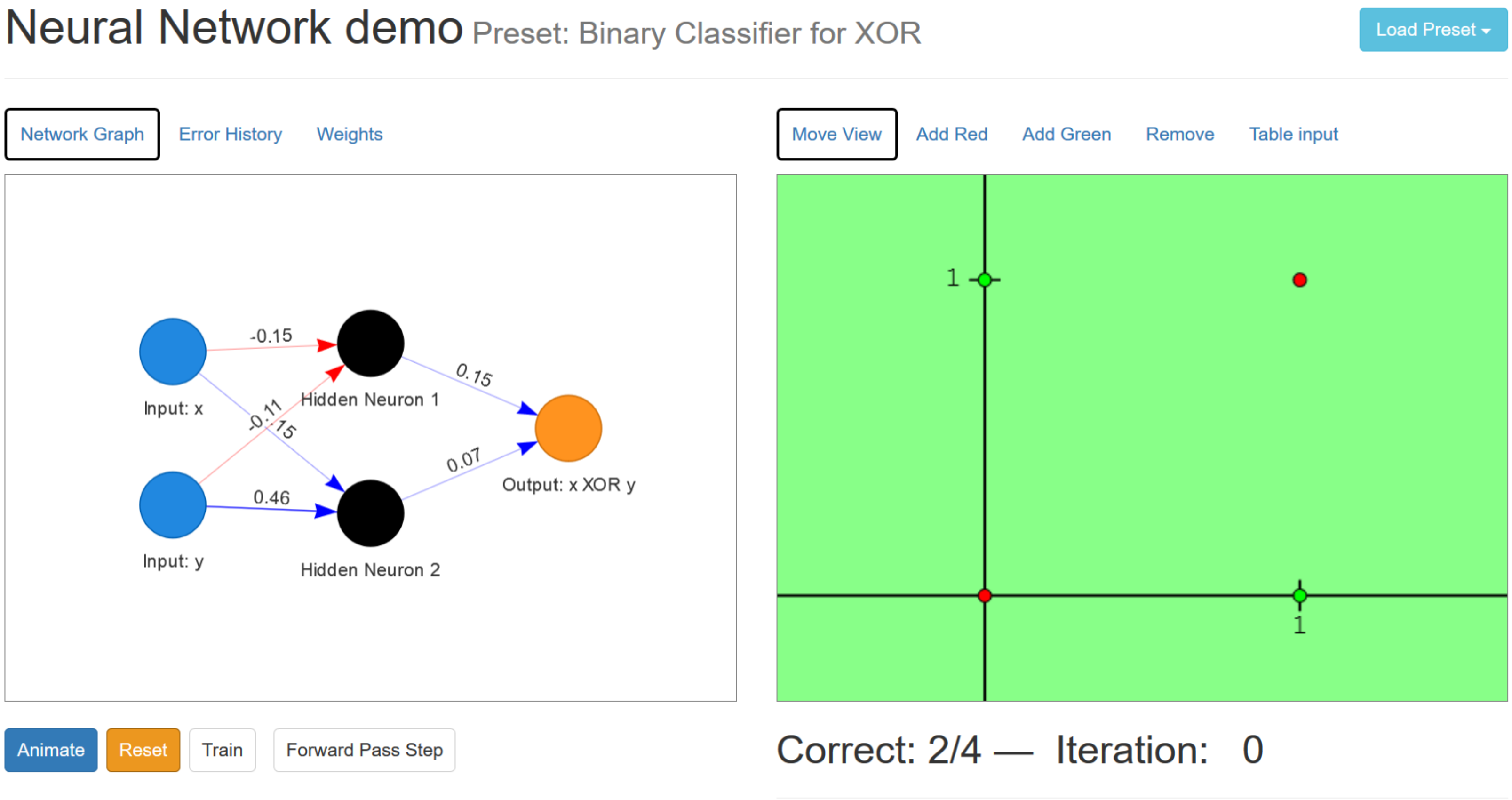
- Input is mapped into the range [0,1] -> probabilistic interpretation
- Reduces the gradient for large inputs -> vanishing gradients

Activation functions: ReLu



- “Rectified linear unit”
- Efficient to compute
- Smaller risk of vanishing gradients

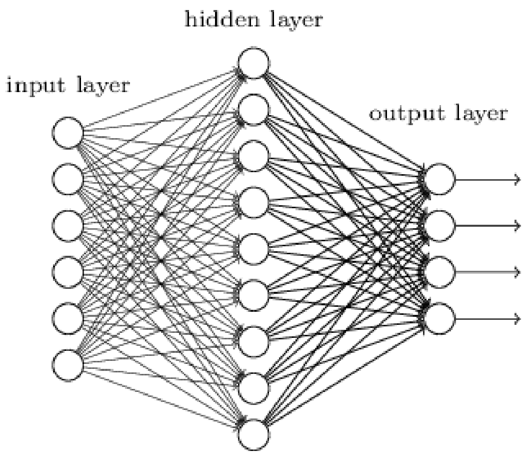
Example Training App



https://lecture-demo.ira.uka.de/neural-network-demo/

From Perceptrons to Deep Neural Networks

- 3-layer neural networks can be used to ***approximate*** any ***continuous function*** to any desired precision



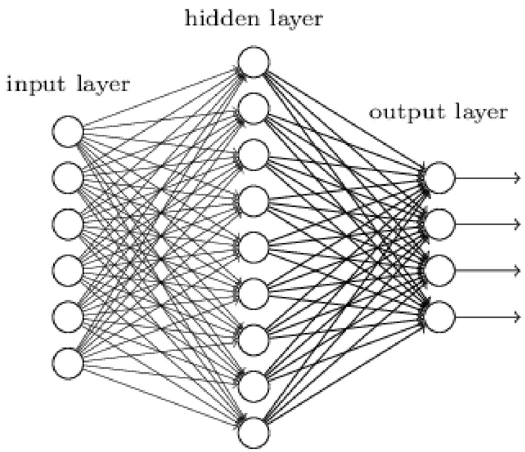
Artificial Neural Network

MNIST – ZIP code data

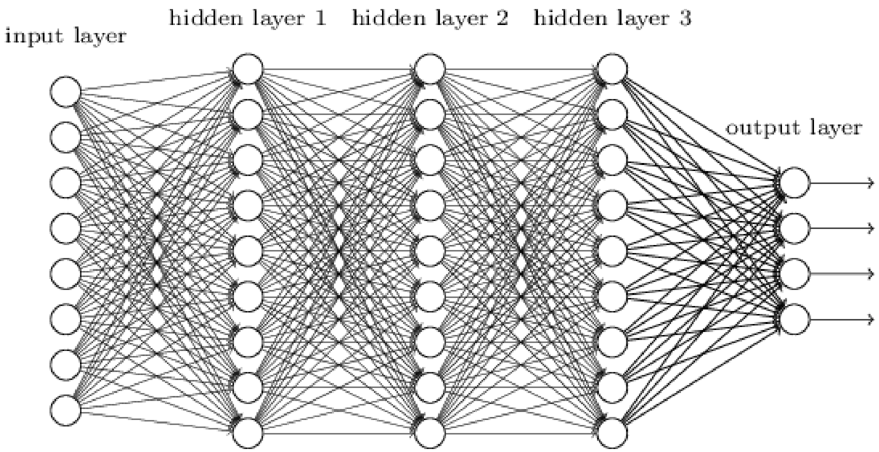
See “Neural Networks and Deep Learning, Michael Nielsen” for an intuitive discussion on that topic.

From Perceptrons to Deep Neural Networks

- Multi-layer networks are preferable over 3-layered networks because they often generalize better



Artificial Neural Network



Deep ANN

See "Neural Networks and Deep Learning, Michael Nielsen" for an intuitive discussion on that topic.

Deep Learning APIs

- Provide a high level API for learning neural networks (define models, load data, automated differentiation)
- Mostly python libraries (caffe is c++)
- For “standard” users these APIs have little difference in terms of what you can do with them

Linear regression in PyTorch

```python
import torch
from torch.autograd import Variable

x = torch.Tensor(range(-5,5))
y = 3*x + 4

w = Variable(torch.Tensor([1.0]), requires_grad=True)
b = Variable(torch.Tensor([1.0]), requires_grad=True)

lr = 0.01

for i in range(25):
    y_hat = w*x + b

    error = torch.sum( torch.pow(y - y_hat,2) )
    error.backward()

    # update parameters
    with torch.no_grad():
        w -= lr * w.grad
        b -= lr * b.grad
        w.grad.zero_()
        b.grad.zero_()
    print("Error: {:.4f}".format(error))

print("w_pred = %.2f, b_pred = %.2f" % (w, b))
```

- In case y were class labels, how could we change this code to perform logistic regression?

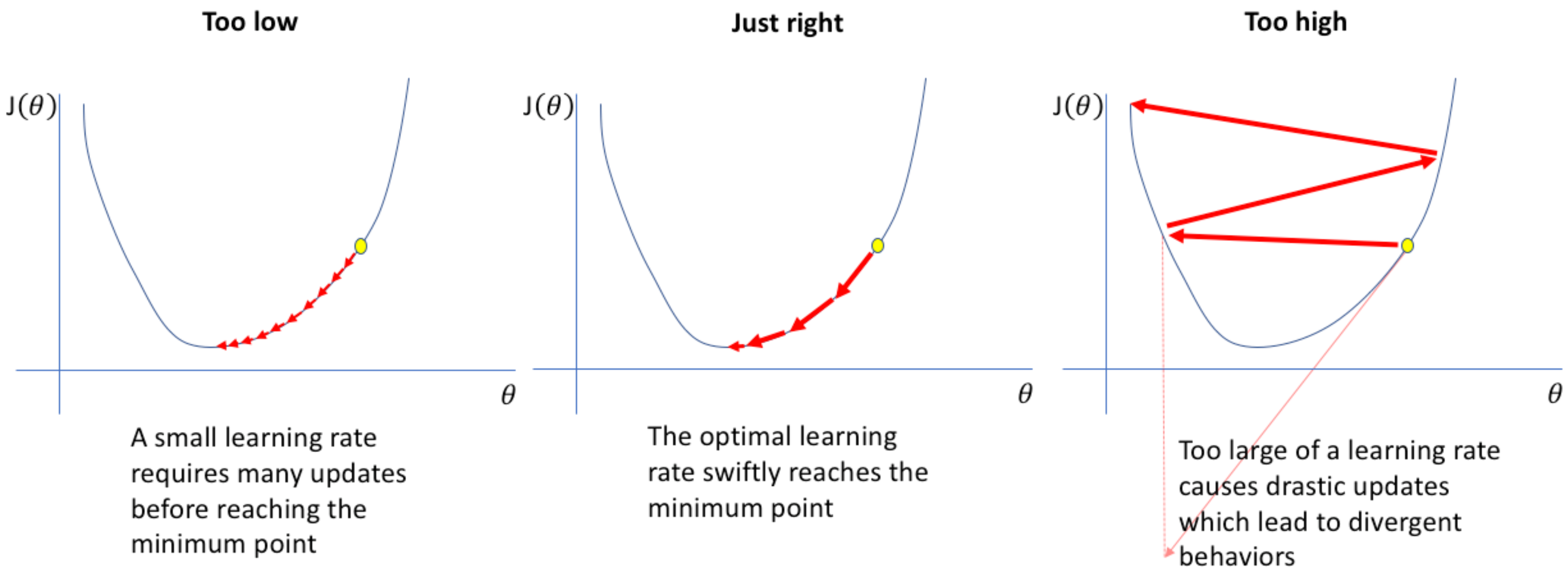
Stochastic Gradient Descent

- Gradient is not accumulated over the whole dataset but over *random* subsets of the training data ("mini-batches")

$$
\begin{aligned}
w_{t+1} &= w_t + \lambda \frac{\partial}{\partial w_t} \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \\
&\approx w_t + \lambda \frac{\partial}{\partial w_t} \frac{1}{\color{red}{M}} \sum_{i=1}^{\color{red}{M}} (y_i - \hat{y}_i)^2, \qquad M \ll N
\end{aligned}
$$

- More efficient in terms of memory consumption and computational cost

Learning rate annealing



- When GD nears a minima in the cost surface, the parameter values can oscillate back and forth around the minima.
- Slow down the parameter updates by decreasing the learning rate
- This could be done manually, however automated techniques are preferable

Learning rate annealing: Adagrad

$$w_{t+1,i} = w_{t,i} + \frac{\lambda}{\sqrt{S_{t,i} + \epsilon}} \frac{\partial L}{\partial w_{t,i}}$$

$$S_{t,i} = S_{t-1,i} + \left[\frac{\partial L}{\partial w_{t,i}} \right]^2$$

- Adapt learning rate by dividing with the cumulative sum of current and past squared gradients *for each feature independently*
- This is beneficial for training since the scale of the gradients in each layer is often different by several orders of magnitude

Variants of gradient descent

Optimiser	**Year**	**Learning Rate**	**Gradient**
Nesterov	1983		✓
Momentum	1999		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓

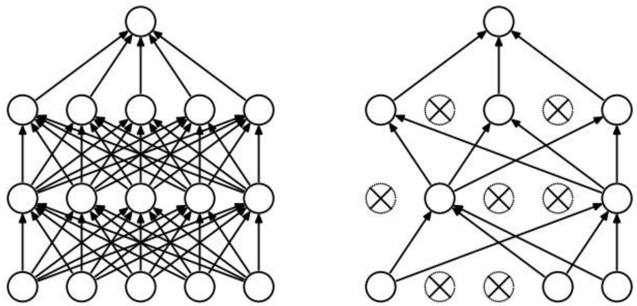
- Variants of gradient descent act either on the learning rate or the gradient itself
- Typically search for the method which is best suited for your problem via trial and error

Regularization

- Weight regularization (weight decay)

$$L(y, \hat{y}) = (y_i - \hat{y}_i)^2 + \alpha ||W||_2$$

- Dropout – drop random neurons along with their connections



(a) Standard Neural Net (b) After applying dropout.

- Early stopping

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

Overview

- Backpropagation in Computational Graphs

- Deep Neural Networks
    - From Perceptrons to Deep Neural Networks
    - High Level APIs
    - Optimization and Regularization

- **Convolutional Neural Networks**
    - Fundamental Properties of Images
    - Basic Architecture & Examples

- Applications

- Open Research Questions

Image Classification

- In Computer Vision, a very popular application scenario is image classification

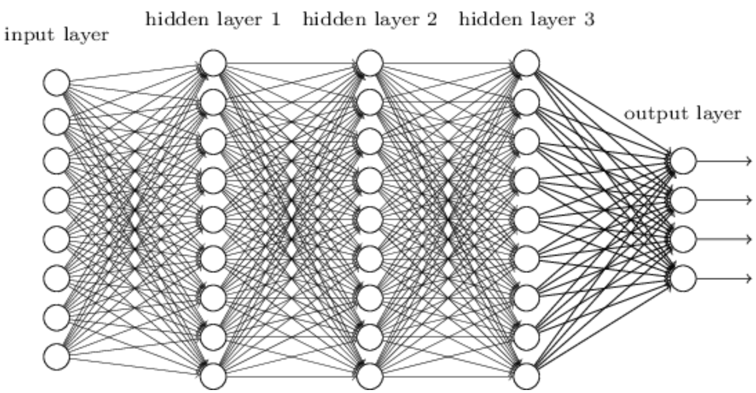
IMAGENET

1000 object classes
1.2m training images
100k testing images

From Perceptrons to Deep Neural Networks

- However, when the input- and output layer are very high dimensional, the number of free parameters becomes huge:

- 5-layer fully connected network
- Hidden layers have the same number of nodes Z
- Number of free parameters: $N_F = N_I Z + Z^2 + Z^2 + Z N_O$



Convolutional Neural Networks

- Key Idea: *Constrain* the networks *architecture* to reduce the amount of network parameters.

- The network is constrained such that:
  - Hidden units are locally connected
  - Weights share shared among hidden units
  - Hidden layers are subsampled

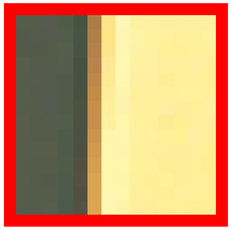
- These changes to the network architecture reflect properties which are specific to images.

Overview

- Backpropagation in Computational Graphs
- Deep Neural Networks
  - From Perceptrons to Deep Neural Networks
  - High Level APIs
  - Optimization and Regularization
- Convolutional Neural Networks
  - **Fundamental Properties of Images**
  - Basic Architecture & Examples
- Applications
- Open Research Questions

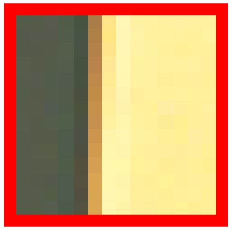
Fundamental Properties of Images

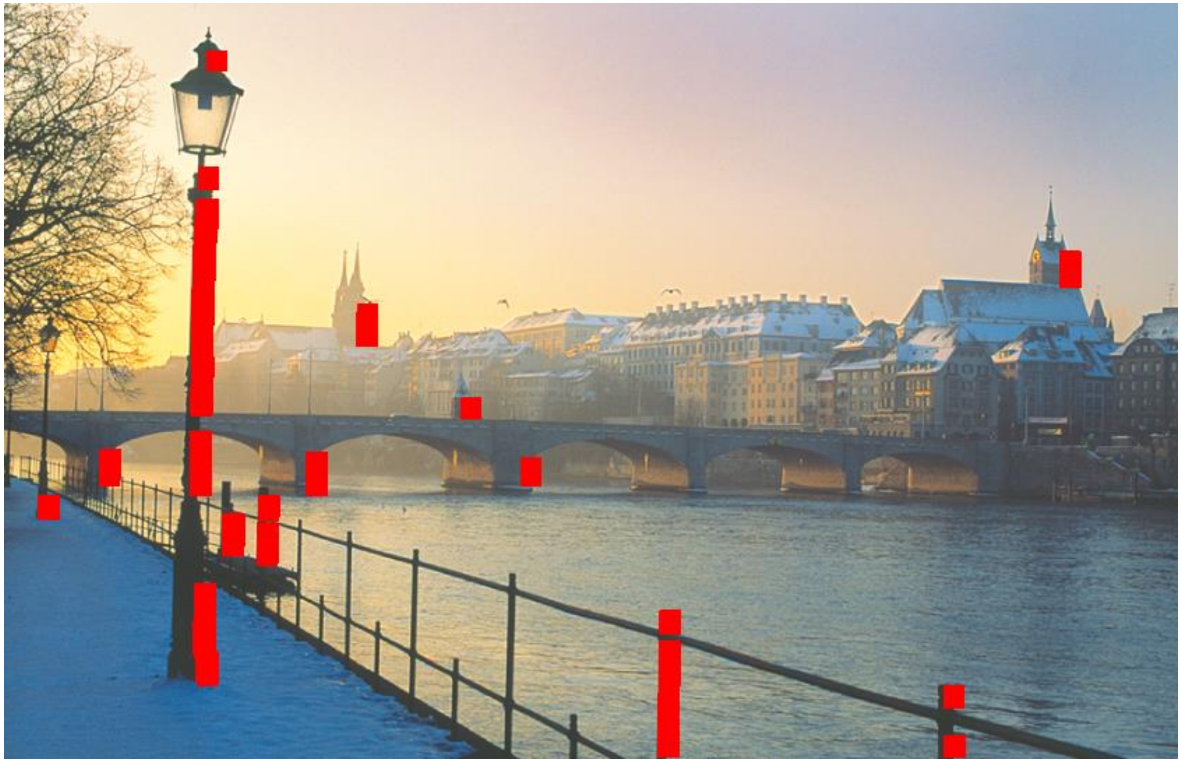
- Property 1: Image statistics are locally correlated (“structured”)



Fundamental Properties of Images

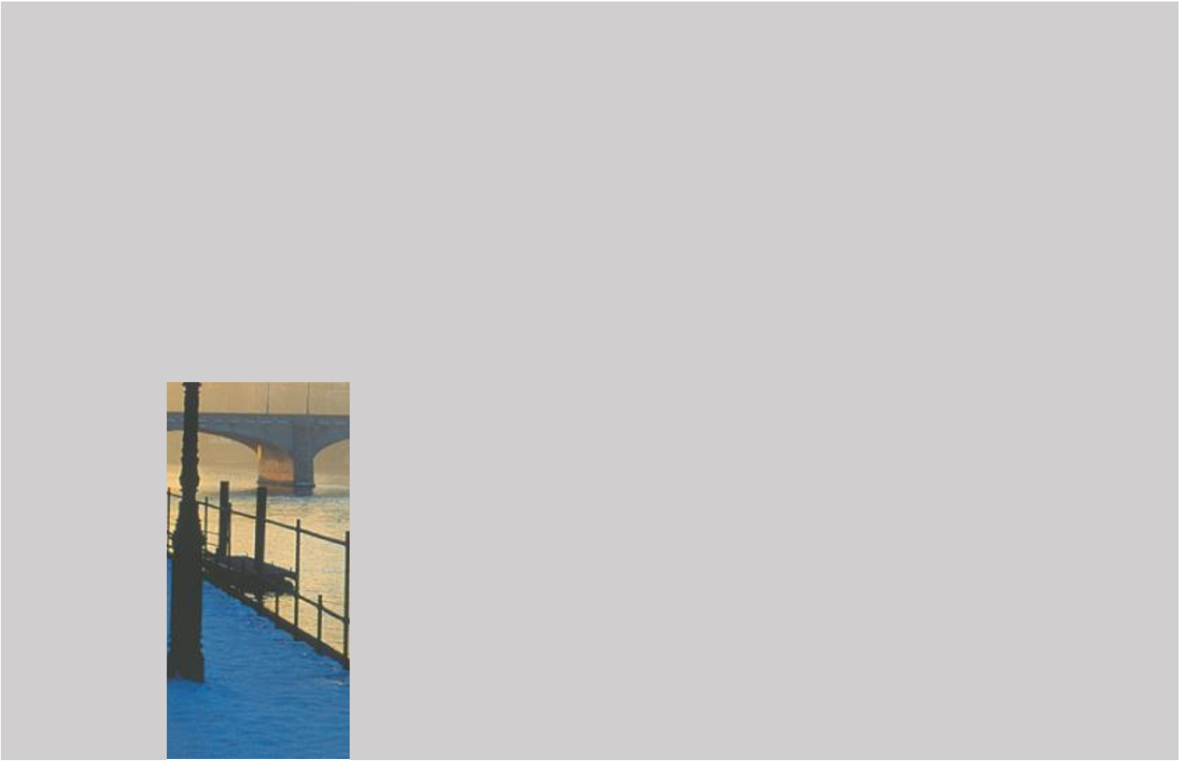
- Property 2: Redundancy





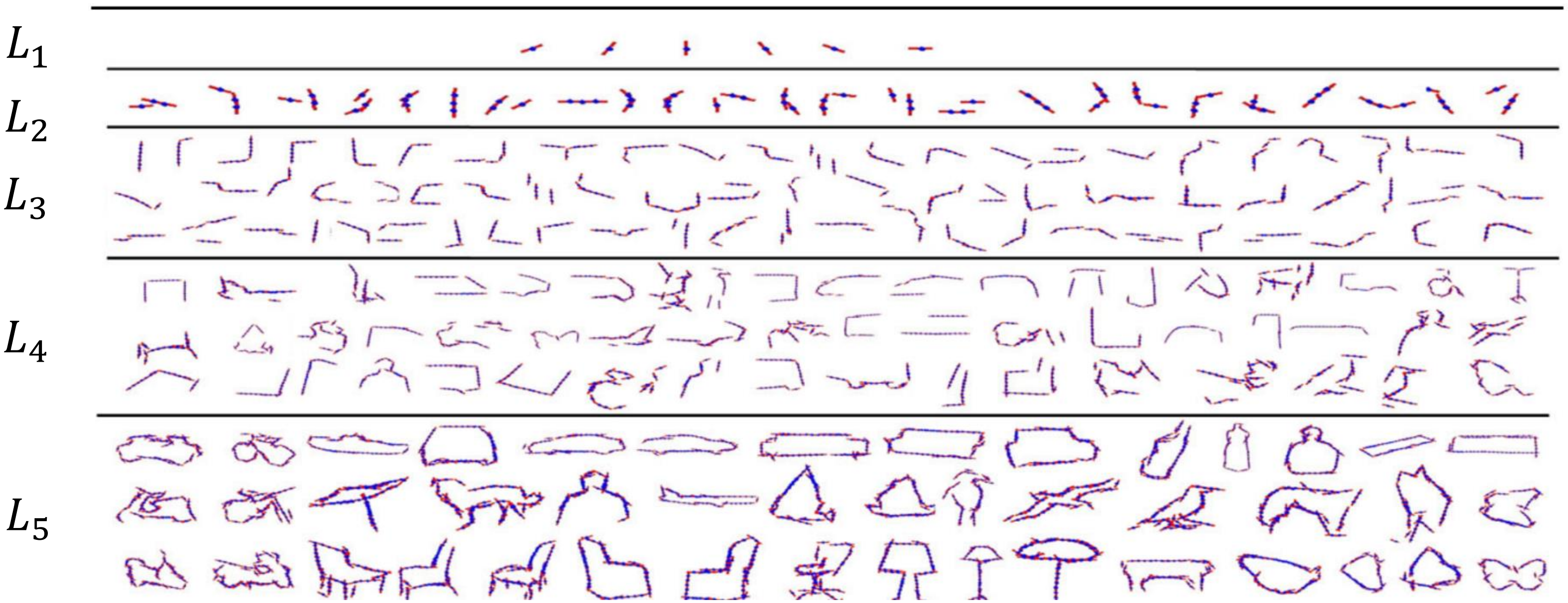
Fundamental Properties of Images

- Property 3: Global Correlations



Fundamental Properties of Images

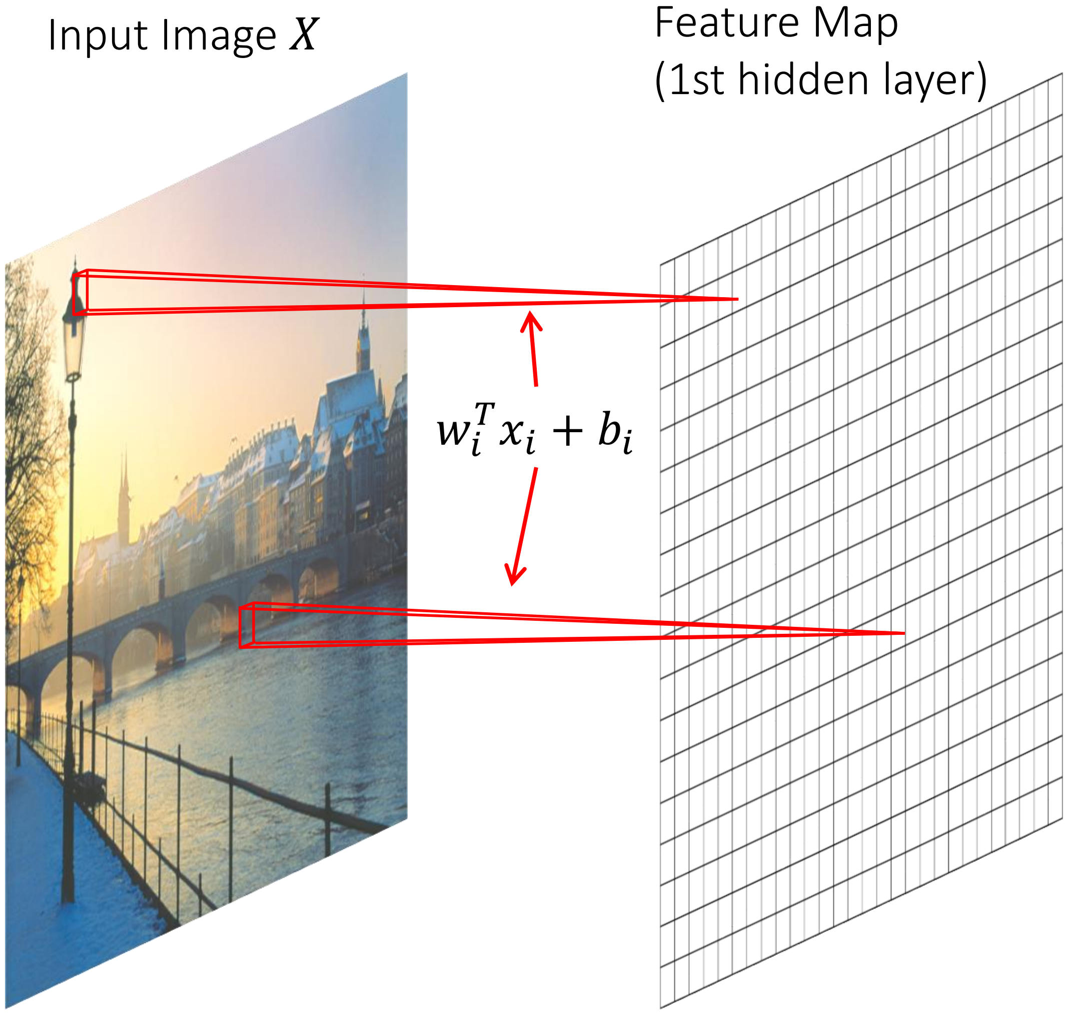
- Property 4: Compositionality of Objects – A small set of building blocks (L_1) is enough to build complex object shapes (L_5) via recursive composition



Overview

- Backpropagation in Computational Graphs
- Deep Neural Networks
  - From Perceptrons to Deep Neural Networks
  - High Level APIs
  - Optimization and Regularization
- Convolutional Neural Networks
  - Fundamental Properties of Images
  - **Basic Architecture & Examples**
- Applications
- Open Research Questions

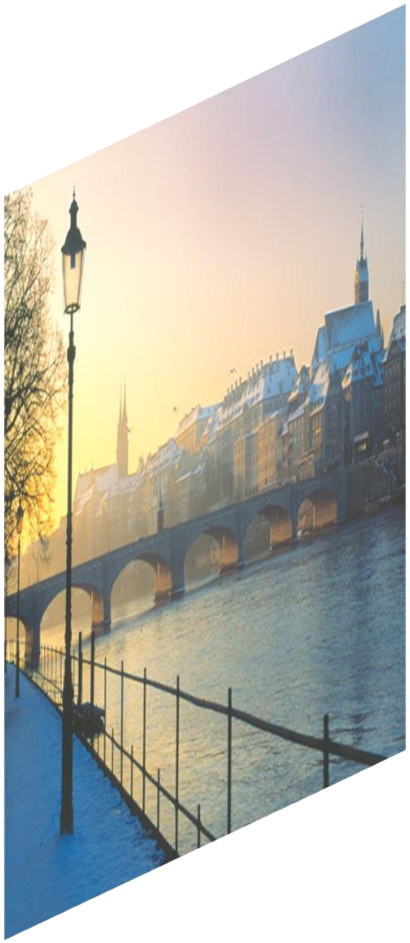
Convolutional Layer



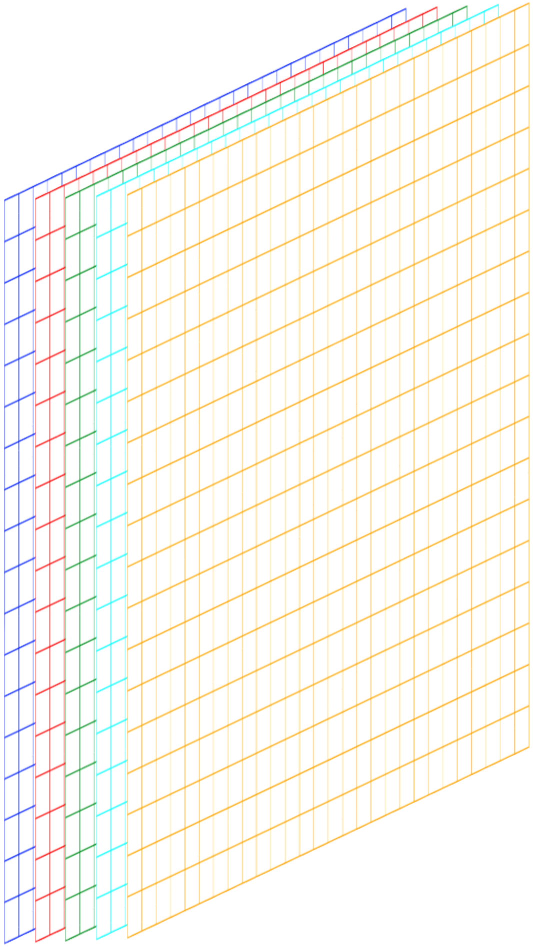
- Preserve the 2D structure of X (no vectorization)
- Hidden units in the *feature map* are connected to small image patches x_i of size $z \times z$ (Property 1)
- Weights w_i are shared across the hidden units in the same feature map (Property 2)

Convolutional Layer

Input Image X



Feature Maps



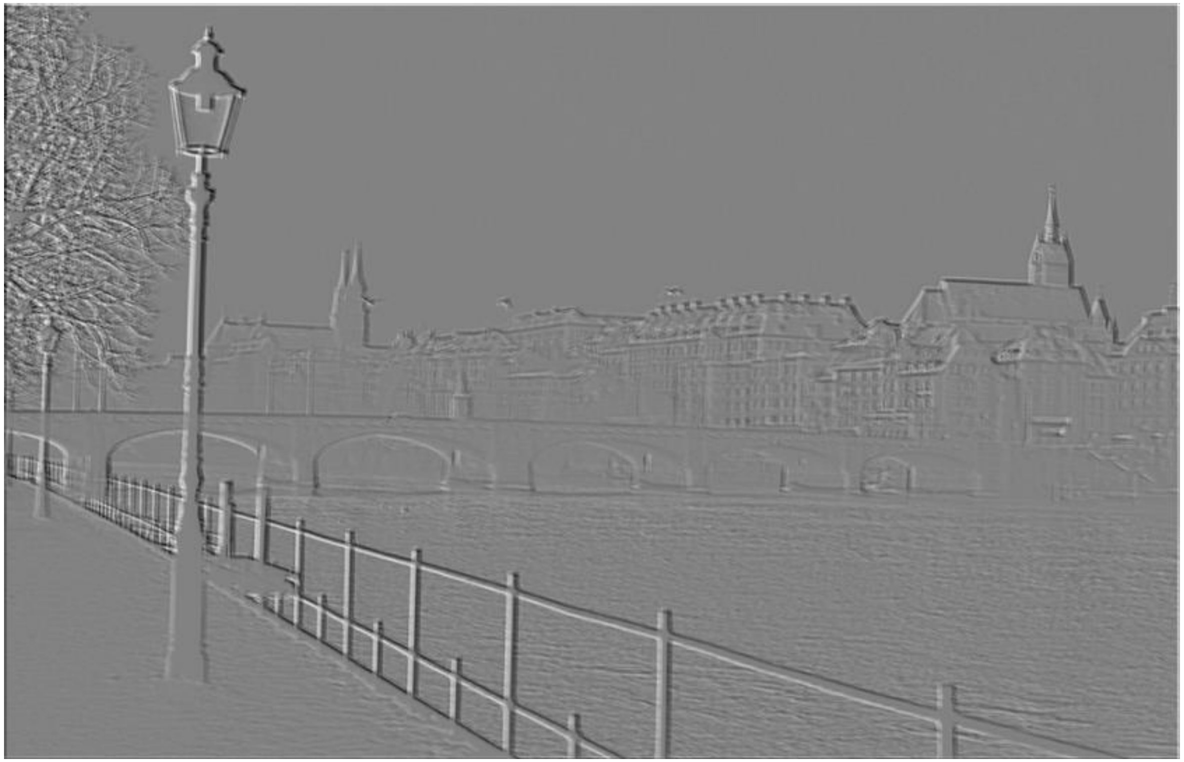
- Preserve the 2D structure of X (no vectorization)
- Hidden units in the *feature map* are connected to small image patches x_i of size $z \times z$ (Property 1)
- Weights w_i are shared across the hidden units (Property 2) -> $w_i = w \quad \forall \quad x_i$
- Multiple (N) feature maps are learned per conv-layer
- This reduces the number of learnable parameters to $\mathrm{N} * z^2$ (e.g. $N = 64, z = 3$)

Convolution

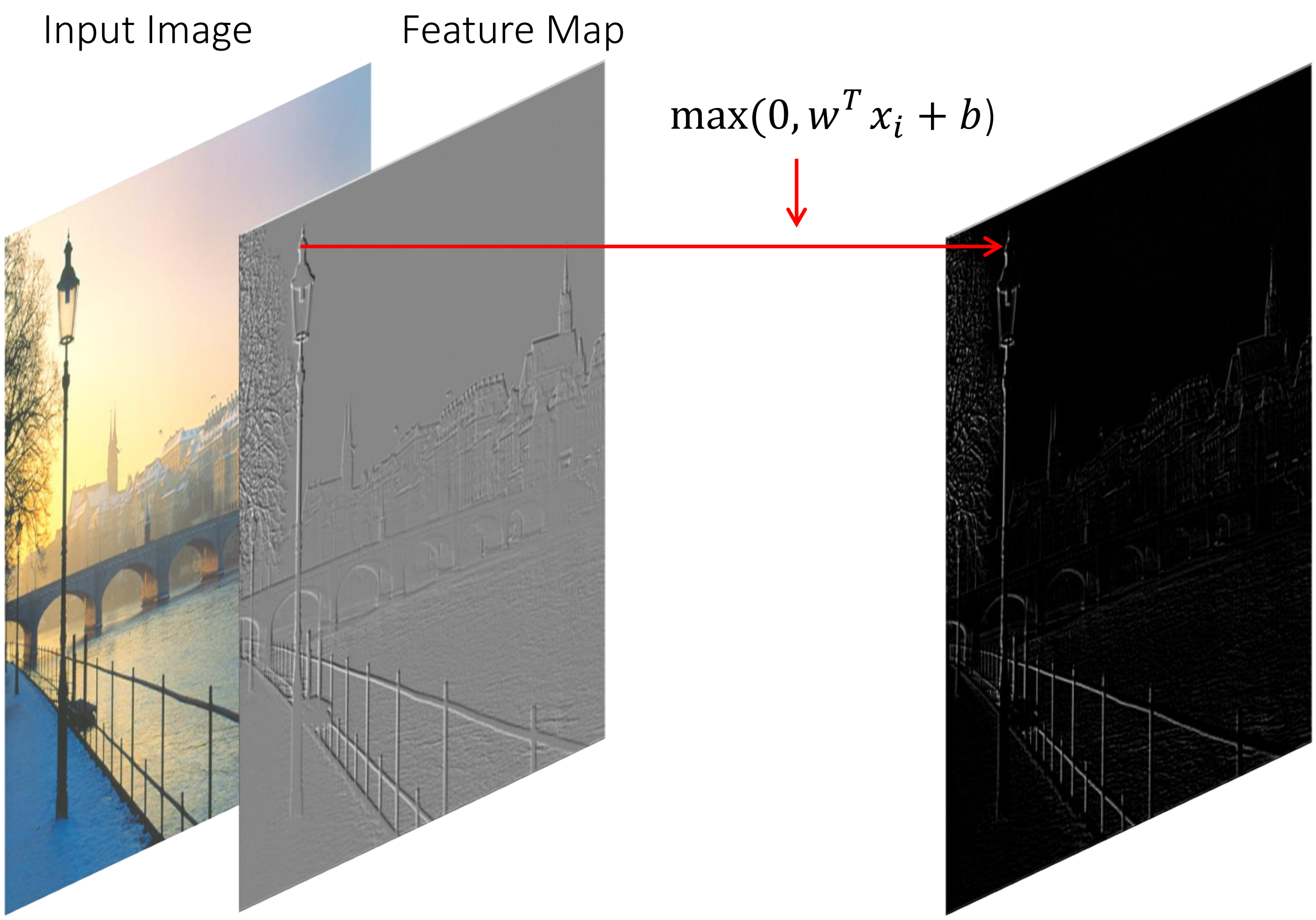
Random weights:

$$w = \begin{matrix} -0.12 & -0.12 & -0.18 & -0.39 & -0.34 \\ -0.27 & 0.36 & 0.29 & -0.42 & 0.10 \\ -0.22 & 0.11 & 0.28 & 0.06 & -0.00 \\ 0.15 & 0.08 & -0.09 & 0.31 & -0.46 \\ 0.00 & 0.45 & 0.10 & 0.46 & -0.13 \end{matrix}$$

Feature Map:



ReLu Activation Function



Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together

9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together

9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

$\longrightarrow$

9		

Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together

9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

9	2	

Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together

9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

9	2	9

Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together

9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

$\longrightarrow$

9	2	9
3		

Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together

9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	1	1
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

9	2	9
3	9	1
9	2	9

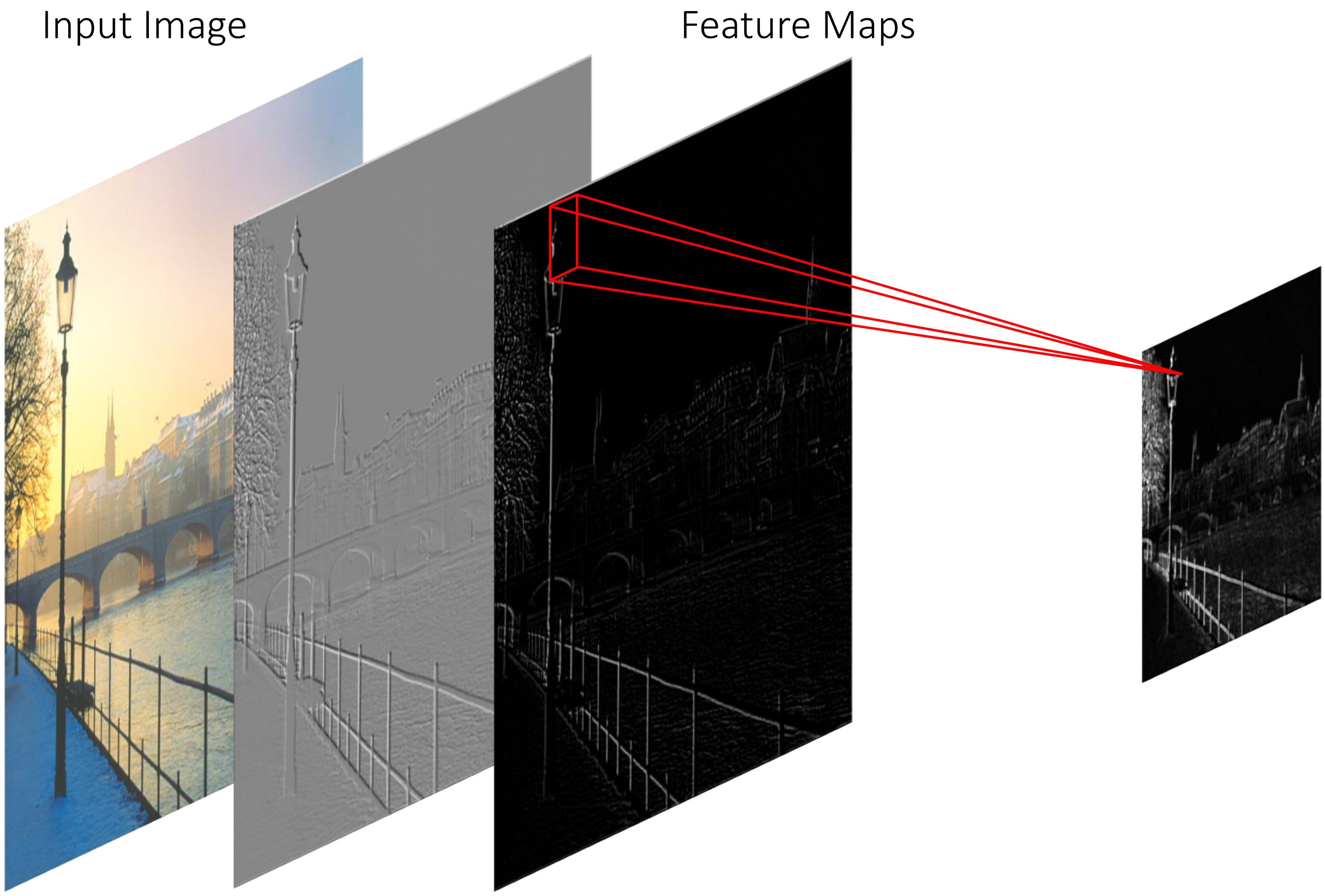
Max Pooling

- Max pooling is a down-sampling process, that locally pools feature responses together. Its main benefits are:
  1. Dimensionality reduction
     - Reduces the number of parameters
     - Simplifies discovery of global patterns
  2. Invariance to small changes of the input signal

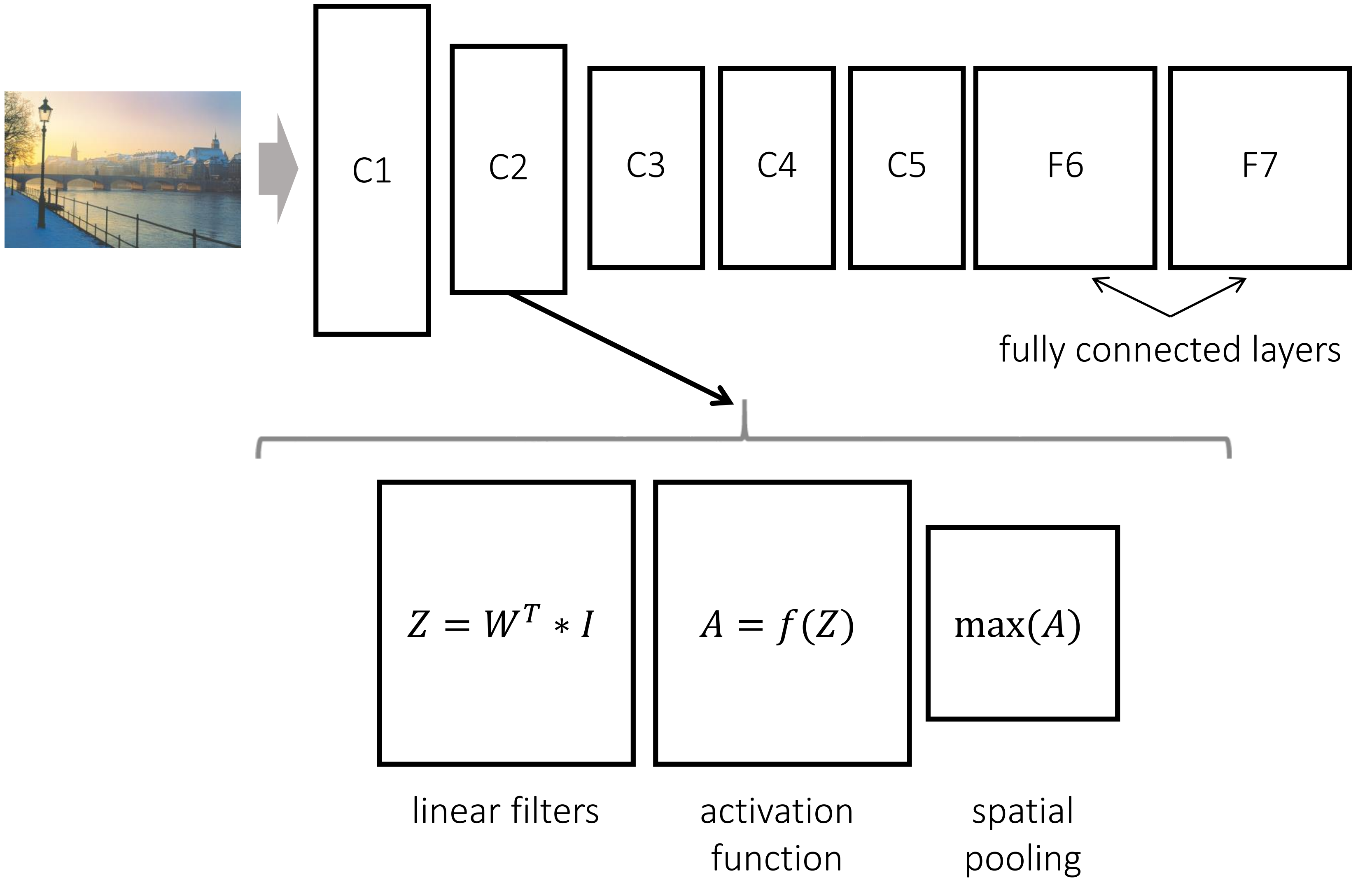
9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	1	1
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

9	2	9
3	9	1
9	2	9

Pooling Layer

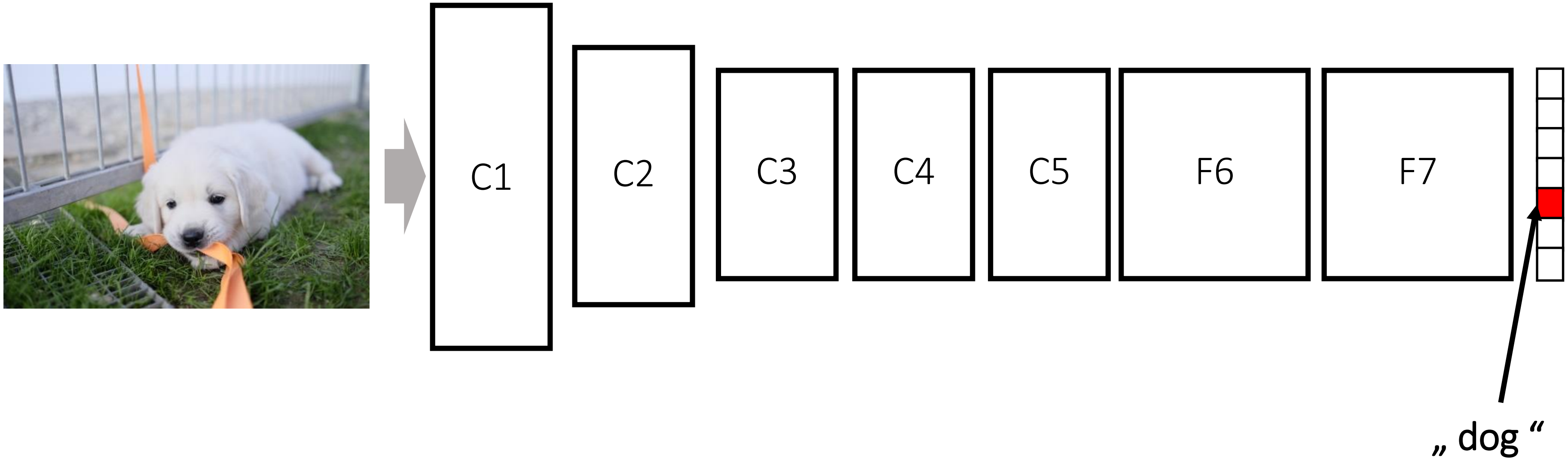


Layered Architecture (Property 3 & 4)



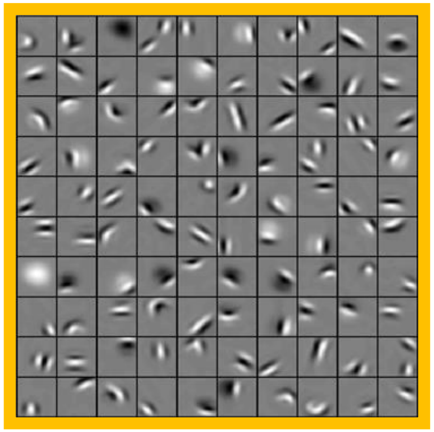
Classification

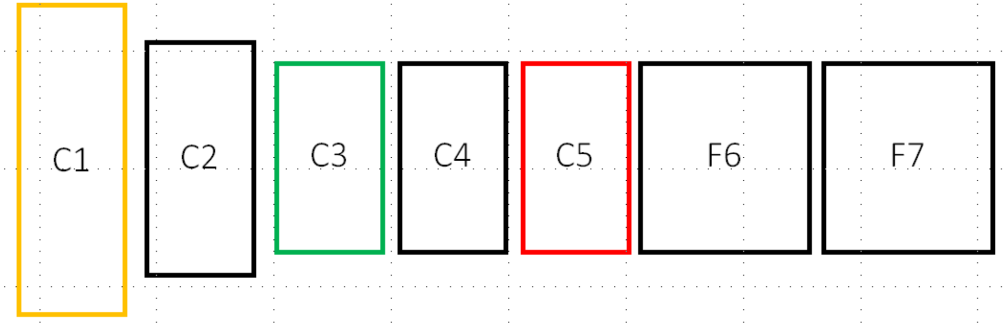
- Add an output layer and train the weights via backpropagation



Visualization of the learned weights

- When trained for face detection:

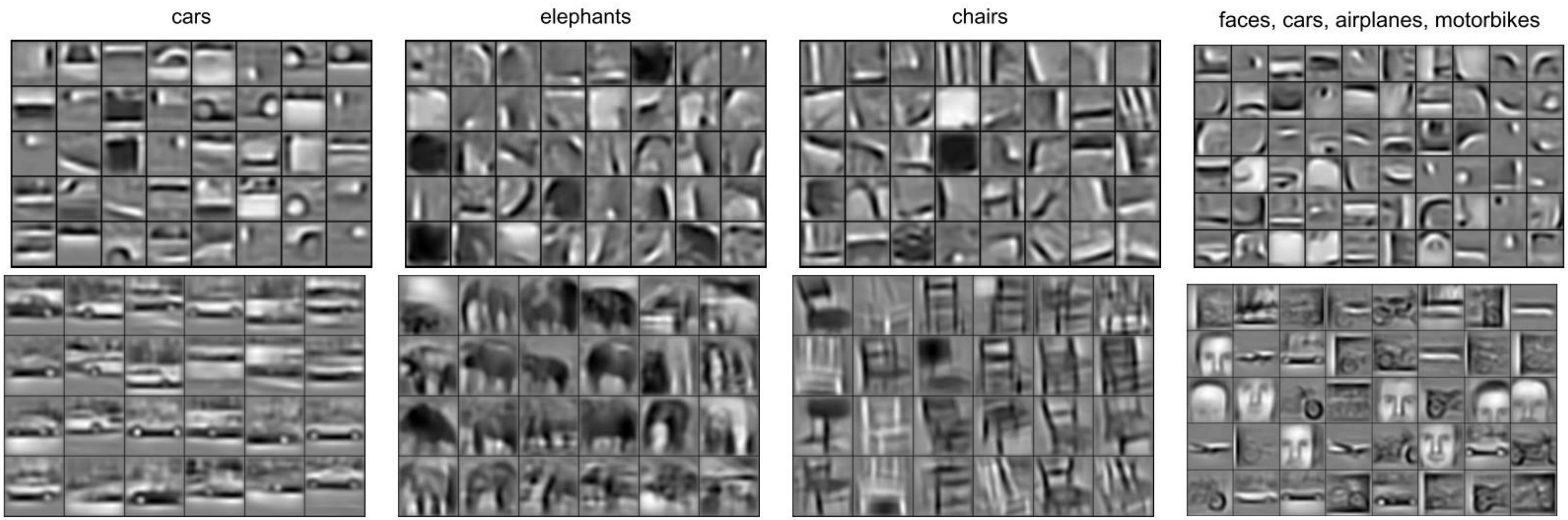




Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Lee, Honglak, et al. 2009

Visualization of the learned weights

- When trained for different object classes:

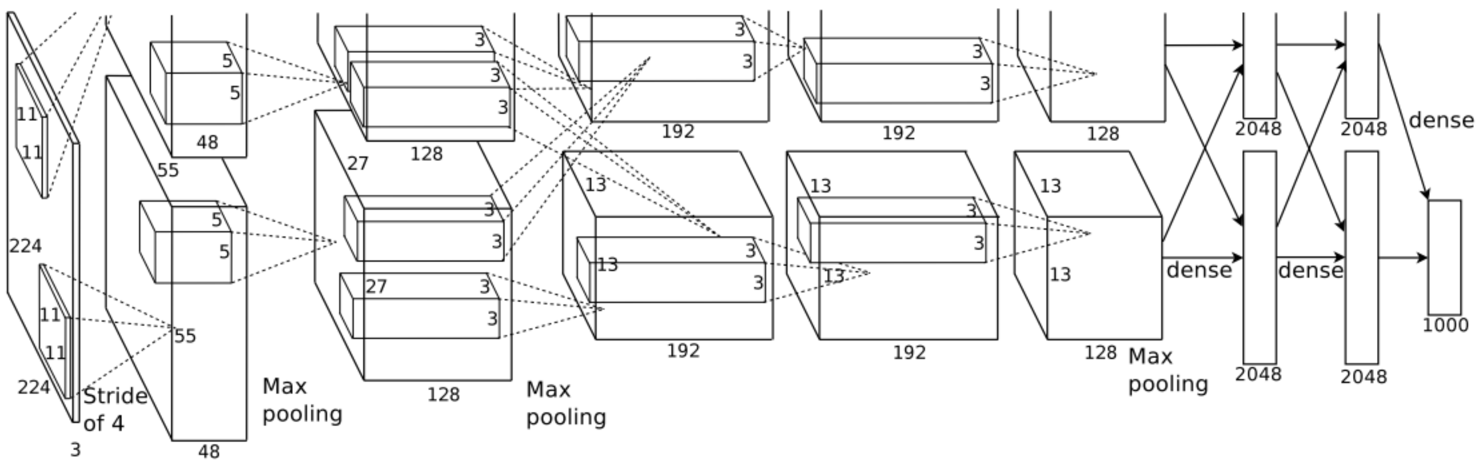


Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Lee, Honglak, et al. 2009

Hyper-Parameters

- **Architecture**
  - Number of layers
  - Order of layers
- **Convolutional Layer**
  - Number of features
  - Size of features
- **Pooling Layer**
  - Window size
  - Window stride
- **Fully Connected Layer**
  - Number of hidden units

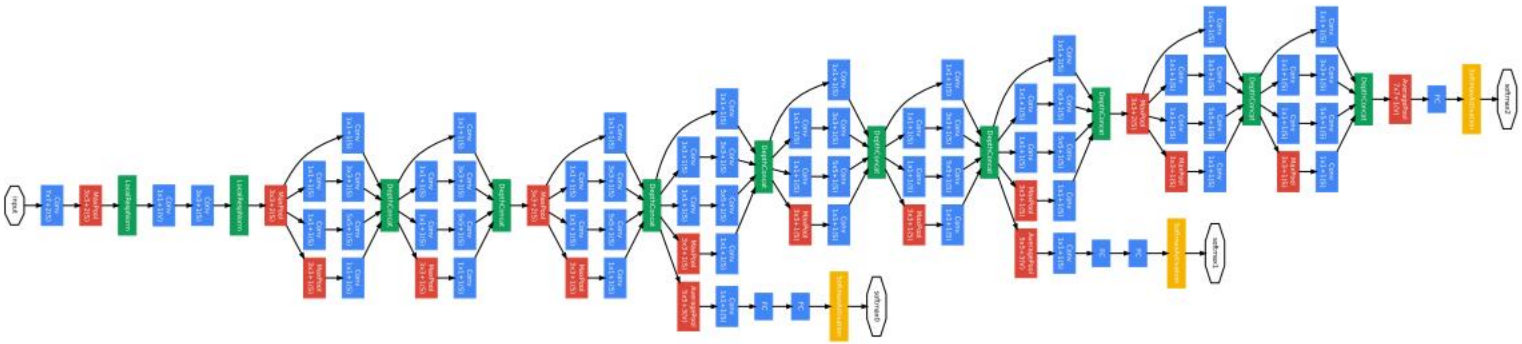
Practical Example



- The winner of the ImageNet Challenge 2012 (84.7%)
  - ~ 60 million parameters, 8 layers
- Choosing the hyper-parameters needs a lot of expert knowledge

Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Lee, Honglak, et al. 2009

Practical Example



- This CNN was the winner of the ImageNet Challenge 2012 (**84.7%**)
  - ~ 60 million parameters, 8 layers
- Choosing the hyper-parameters needs a lot of expert knowledge
- 2014: GoogLeNet – **93.33%**, 22 layers

Going deeper with convolutions." Szegedy, Christian, et al. 2015.

Overview

- Backpropagation in Computational Graphs
- Deep Neural Networks
  - From Perceptrons to Deep Neural Networks
  - High Level APIs
  - Optimization and Regularization
- Convolutional Neural Networks
  - Fundamental Properties of Images
  - Basic Architecture & Examples
- **Applications**
- Open Research Questions

Application: Scene Classification

Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** bridge:0.40, lighthouse:0.09, viaduct:0.08, river:0.08, tower:0.07

http://places.csail.mit.edu/demo.html

Applications beyond Classification

A Neural Algorithm of Artistic Style - Gatys, Ecker, Bethge. 2015

Beyond CNNs: Speech Recognition

- Microsoft performs on par with human performance in speech recognition

THE MICROSOFT 2016 CONVERSATIONAL SPEECH RECOGNITION SYSTEM

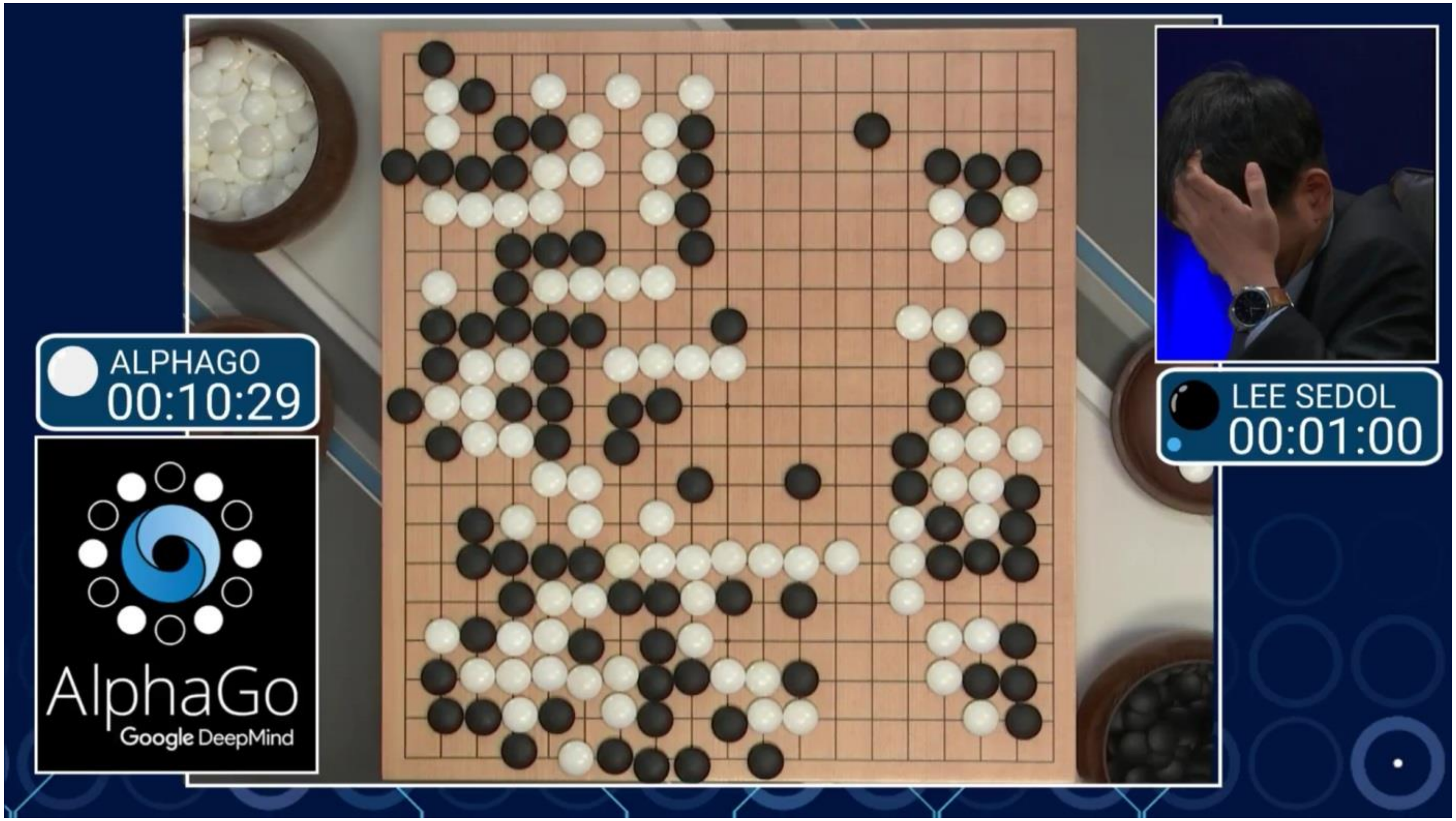
W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu and G. Zweig

Microsoft Research

Table 5. Word error rates (%) on the eval 2000 set with different acoustic models. Unless otherwise noted, models are trained on the full 2000 hrs. of data and have 9k senones.

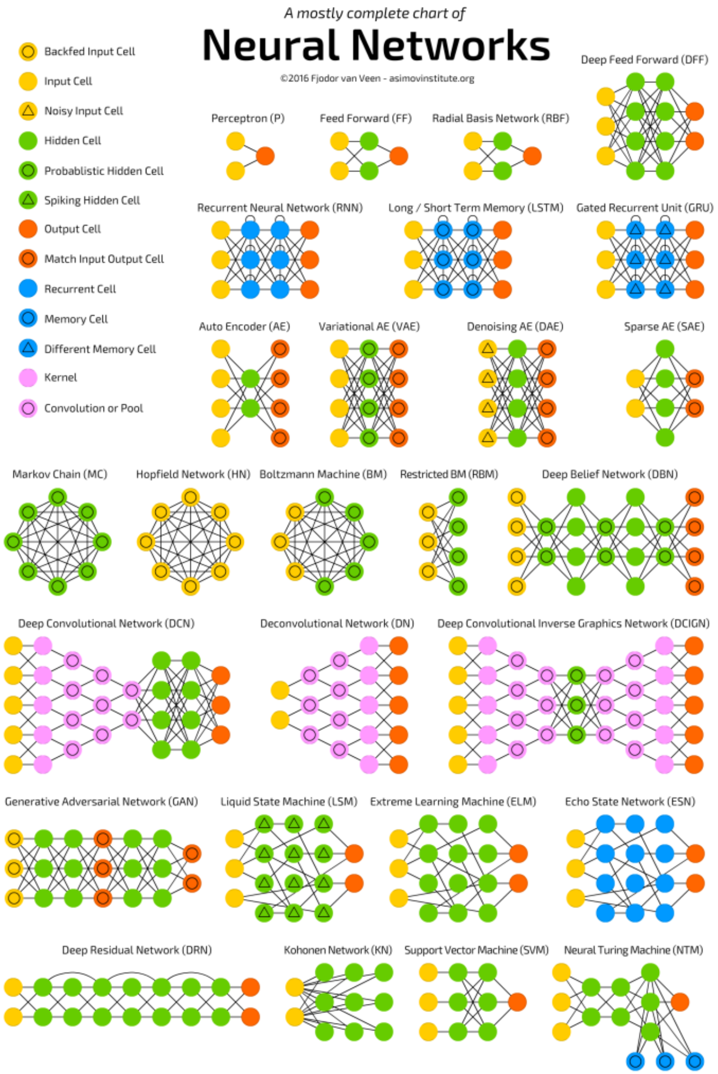
Model	N-gram LM		RNN LM	
	CH	SWB	CH	SWB
Saon et al. [28] LSTM	15.1	9.0	-	-
Povey et al. [19] LSTM	15.3	8.5	-	-
Saon et al. [28] Combination	-	-	12.2	6.6
300h ResNet	19.2	10.0	17.7	8.2
ResNet GMM alignment	15.3	8.8	13.7	7.3
ResNet	14.8	8.6	13.2	6.9
VGG	15.7	9.1	14.1	7.6
LACE	14.8	8.3	13.5	7.1
BLSTM	16.7	9.0	15.3	7.8
27k Senone BLSTM	16.2	8.7	14.6	7.5
Combination	13.4	7.4	**11.9**	**6.3**

Beyond CNNs: Playing Go



Mastering the game of Go with deep neural networks and tree search – David Silver et al. 2015

Prototypical Network Architectures



Overview

- Backpropagation in Computational Graphs

- Deep Neural Networks
  - From Perceptrons to Deep Neural Networks
  - High Level APIs
  - Optimization and Regularization

- Convolutional Neural Networks
  - Fundamental Properties of Images
  - Basic Architecture & Examples

- Applications

- **Open Research Questions**

Learning from Failure Cases

Predictions:

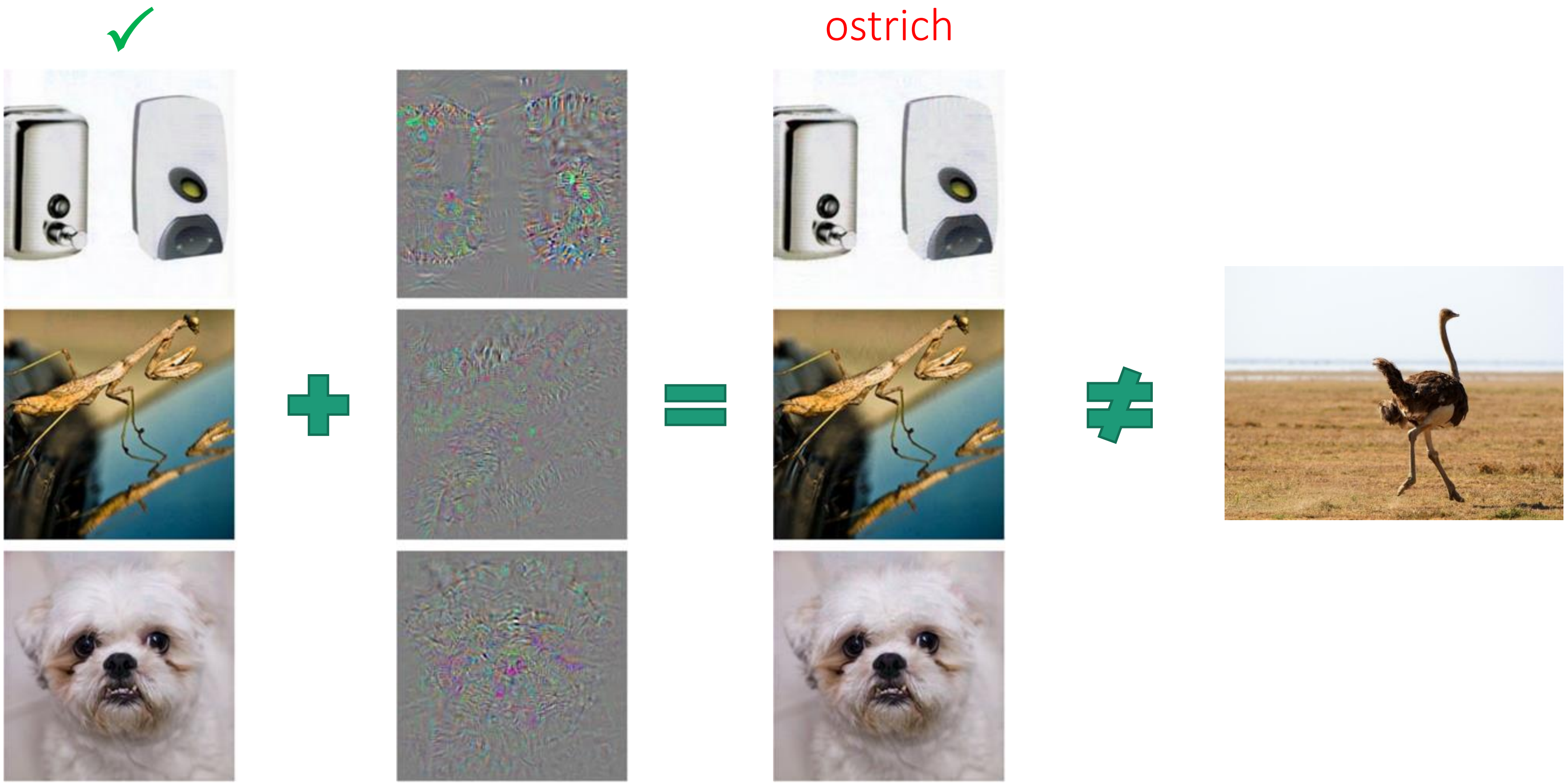
- **Type of environment:** outdoor
- **Semantic categories:** arch:0.29, amphitheater:0.13, viaduct:0.11, stadium/football:0.11, bridge:0.08

How do we resolve these errors?

http://places.csail.mit.edu/demo.html

Learning from Failure Cases

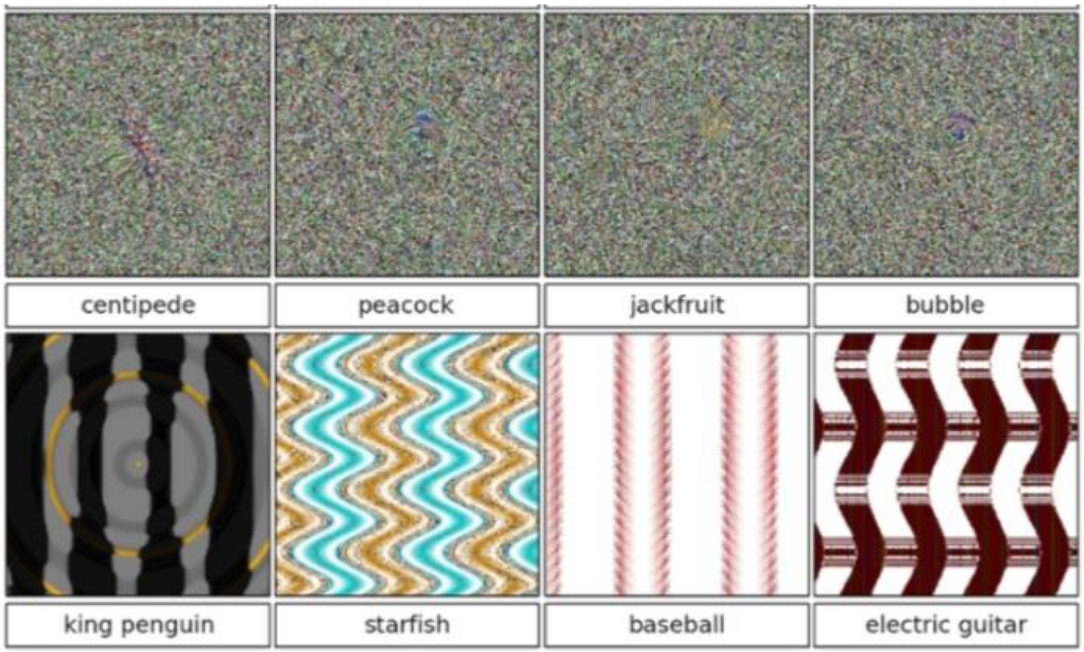
- Adding the “right” noise induces miss-classification



Szegedy, Christian, et al. "Intriguing properties of neural networks." 2013

Learning from Failure Cases

- Generating "adversarial" examples – classification confidence > 99%



Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. Nguyen, Anh, Jason Yosinski, and Jeff Clune. *2015*

Open Questions

- Transfer learning

  Reuse learning results from other datasets

- How can the Hyper-Parameters be learned?

- Vanishing Gradients

  Different activation functions
  Adding momentum to the gradient

- How to apply these networks to problems with few data

  Data Augmentation

- Better theoretical understanding

  Why and when do more hidden layers help?

- How to integrate reasoning capabilities (context, human expert knowledge)

Summary

- Automated differentiation on computational graphs allows for differentiation of complex mathematical programs
- Deep Neural Networks are a powerful tool and the driving force of recent developments in artificial intelligence
- Deep learning is currently rather an engineering science than a theoretical science (comparable to early alchemy)
- Many open questions left that must be addressed

Credits

neuralnetworksanddeeplearning.com

class.coursera.org/neuralnets-2012-001

cs231n.github.io

appliedgo.net

brohrer.github.io

Presentations @ CVSS15 of

- Raquel Urtasun
- Andrew Zisserman