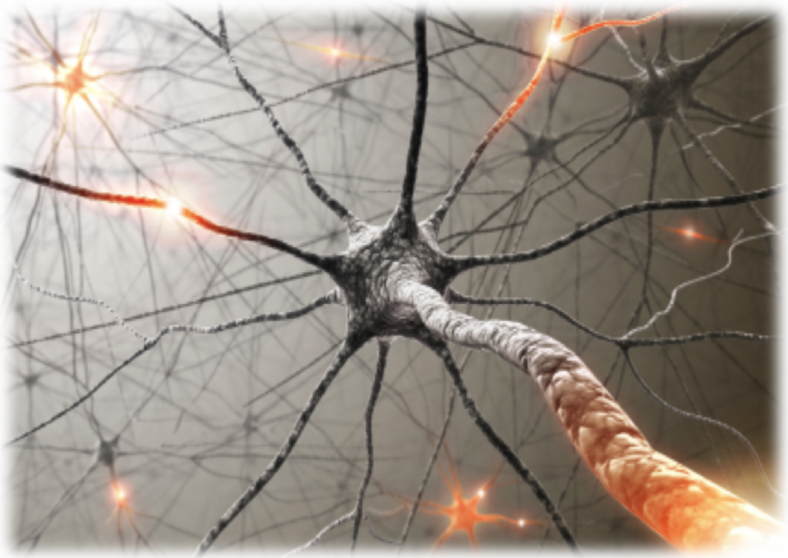
A Gentle Introduction to Deep Learning

Center for AI - Faculty of Engineering and Information Sciences



Part of the content is referred to Stanford AI Subject

A brief history

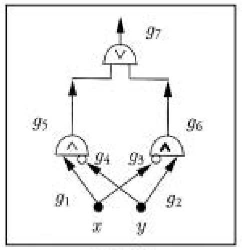


Fig. 2.1

Functional description:

$g_1 := x;$
$g_2 := y;$
$g_3 := \neg g_1;$
$g_4 := \neg g_2;$
$g_5 := g_1 \wedge g_4;$
$g_6 := g_2 \wedge g_3;$
$g_7 := g_5 \vee g_6;$

Straight-line program:

(1 READ x)
(2 READ y)
(3 NOT 1)
(4 NOT 2)
(5 AND 1 4)
(6 AND 3 2)
(7 OR 5 6)
(8 OUTPUT 5)
(9 OUTPUT 7)

Figure from Models of Computation

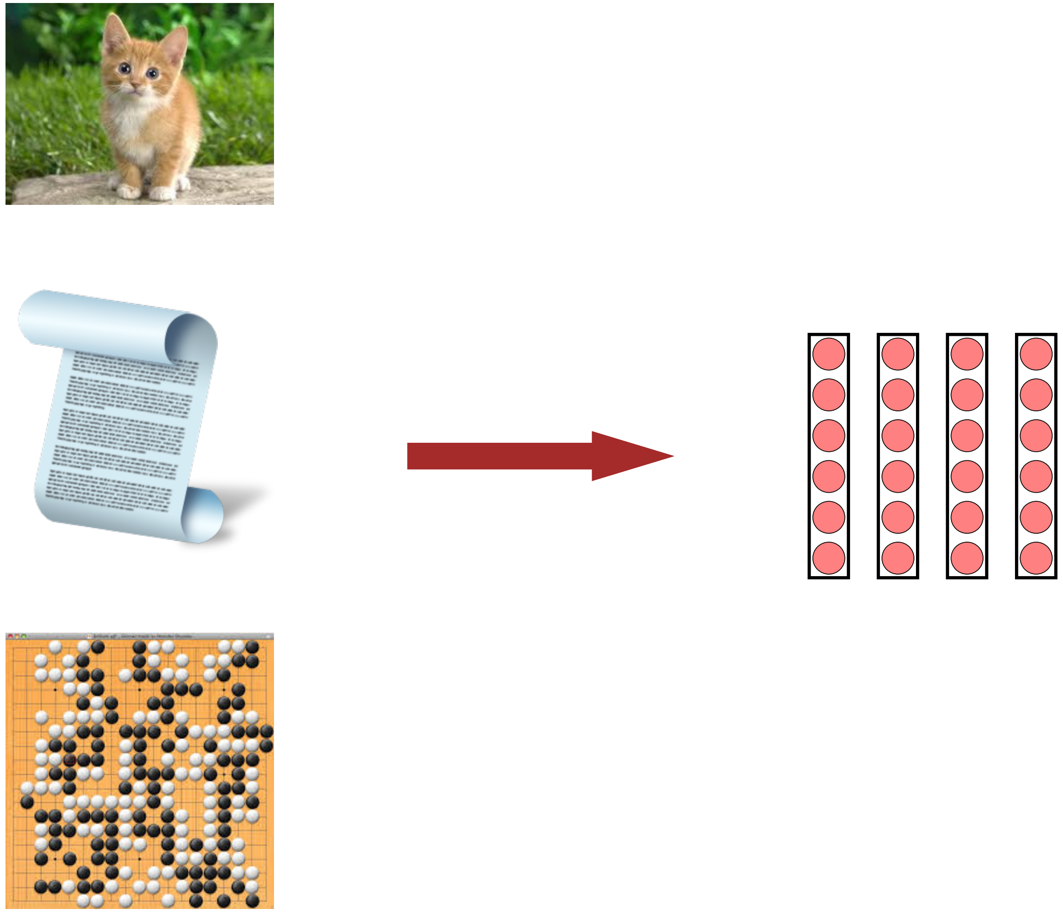
- 1943: neural networks $\Leftrightarrow$ logical circuits (McCulloch/Pitts)
- 1949: "cells that fire together wire together" learning rule (Hebb)
- 1969: theoretical limitations of neural networks (Minsky/Papert)
- 1974: backpropagation for training multi-layer networks (Werbos)
- 1986: popularization of backpropagation (Rumelhardt, Hinton, Williams)

A brief history

- 1980: Neocognitron, a.k.a. convolutional neural networks (Fukushima)
- 1989: backpropagation on convolutional neural networks (LeCun)
- 1990: recurrent neural networks (Elman)
- 1997: Long Short-Term Memory networks (Hochreiter/Schmidhuber)
- 2006: unsupervised layerwise training of deep networks (Hinton et al.)

What is deep learning?

A family of techniques for learning compositional vector representations of complex data.



Roadmap

Feedforward neural networks

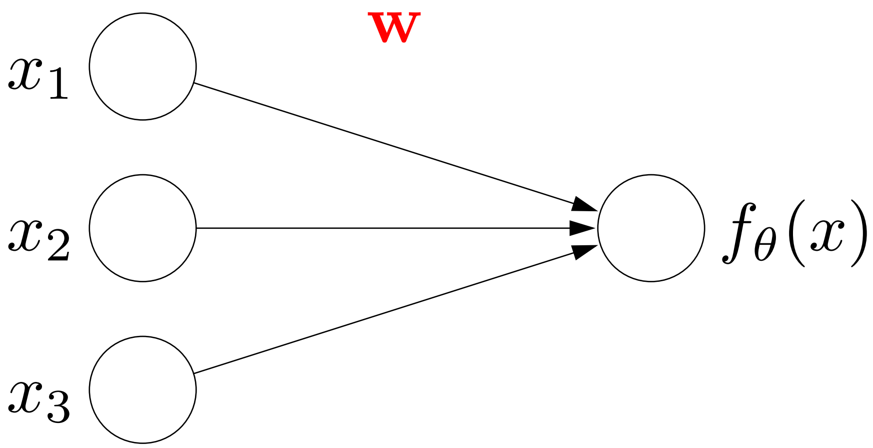
Convolutional neural networks

Recurrent neural networks

Unsupervised learning

Final remarks

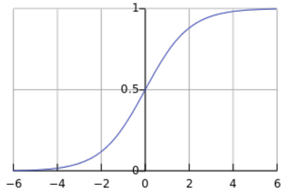
Review: linear predictors



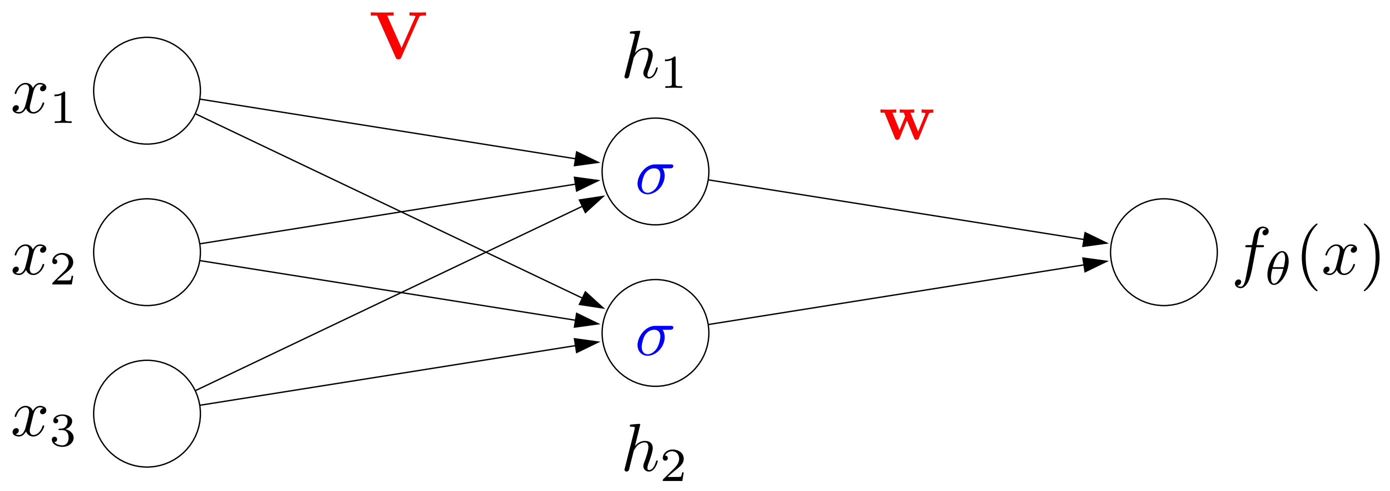
Output:

$$f_\theta(x) = \mathbf{w} \cdot x$$

Parameters: $\theta = \mathbf{w}$



Review: neural networks



Intermediate hidden units:

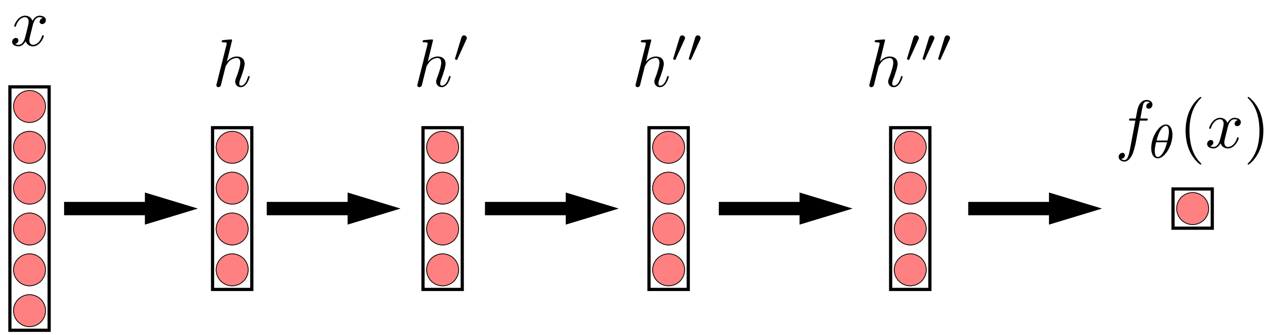
$$h_j(x) = \sigma(\mathbf{v}_j \cdot x) \qquad \sigma(z) = (1 + e^{-z})^{-1}$$

Output:

$$f_\theta(x) = \mathbf{w} \cdot \mathbf{h}(x)$$

Parameters: $\theta = (\mathbf{V}, \mathbf{w})$

Depth

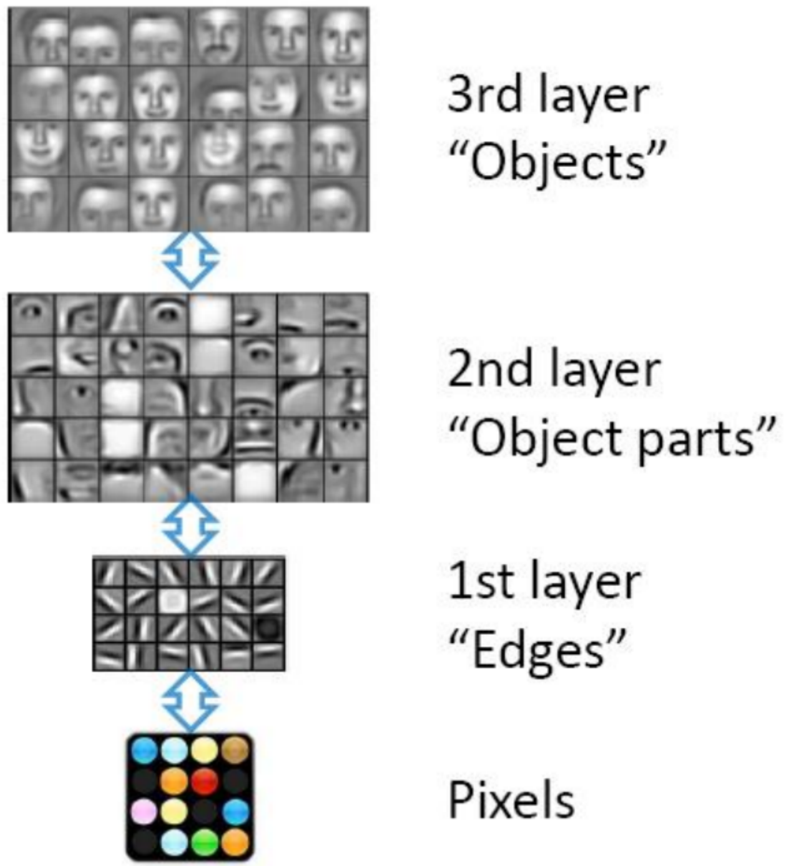


Intuitions:

- Hierarchical feature representations
- Can simulate a bounded computation logic circuit (original motivation from McCulloch/Pitts, 1943)
- Learn this computation (and potentially more because networks are real-valued)
- Depth $k+1$ logic circuits can represent more than depth k (counting argument)
- Formal theory/understanding is still incomplete

[figure from Honglak Lee]

What's learned?



Summary

- Deep networks learn hierarchical representations of data

- Train via SGD, use backpropagation to compute gradients

- Non-convex optimization, but works empirically given enough compute and data

Review: optimization

Regression:

$$\text{Loss}(x, y, \theta) = (f_\theta(x) - y)^2$$

Key idea: minimize training loss

$$\text{TrainLoss}(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \theta)$$

$$\min_{\theta \in \mathbb{R}^d} \text{TrainLoss}(\theta)$$

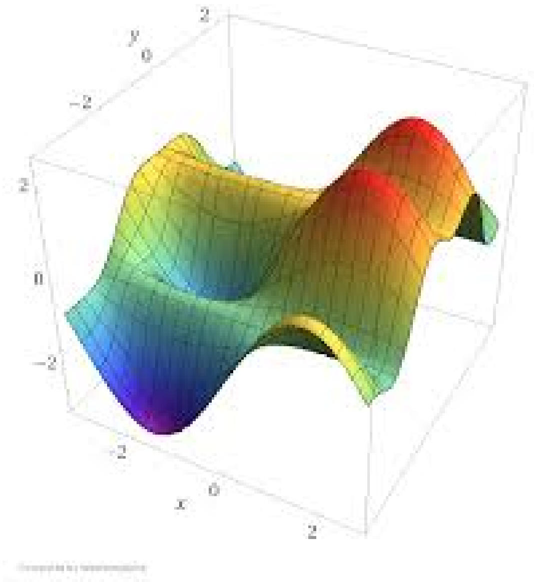
Algorithm: stochastic gradient descent

For $t = 1, \ldots, T$:

For $(x, y) \in \mathcal{D}_{\text{train}}$:

$\theta \leftarrow \theta - \eta_t \nabla_\theta \text{Loss}(x, y, \theta)$

Training



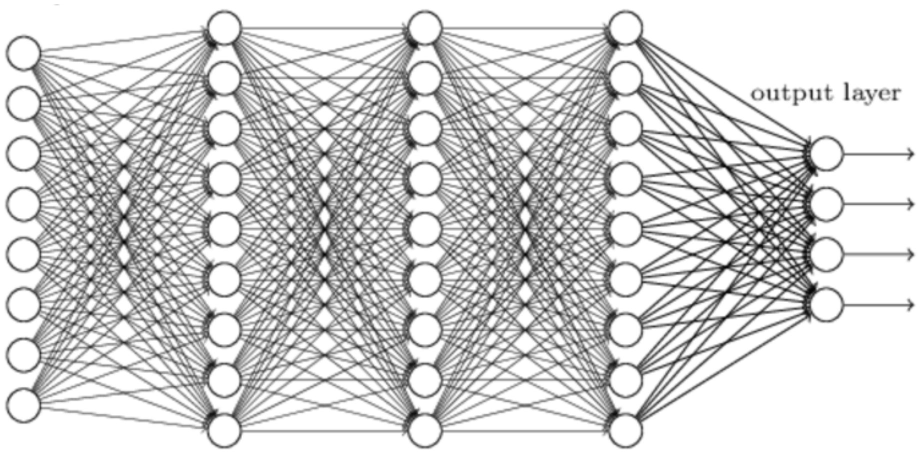
- Non-convex optimization
- No theoretical guarantees that it works
- Before 2000s, empirically very difficult to get working

What's different today

Computation (time/memory)

Information (data)

How to make it work



- More hidden units (over-provisioning)
- Adaptive step sizes (AdaGrad, ADAM)
- Dropout to guard against overfitting
- Careful initialization (pre-training)
- Batch normalization
- Model and optimization are tightly coupled

Roadmap

Feedforward neural networks

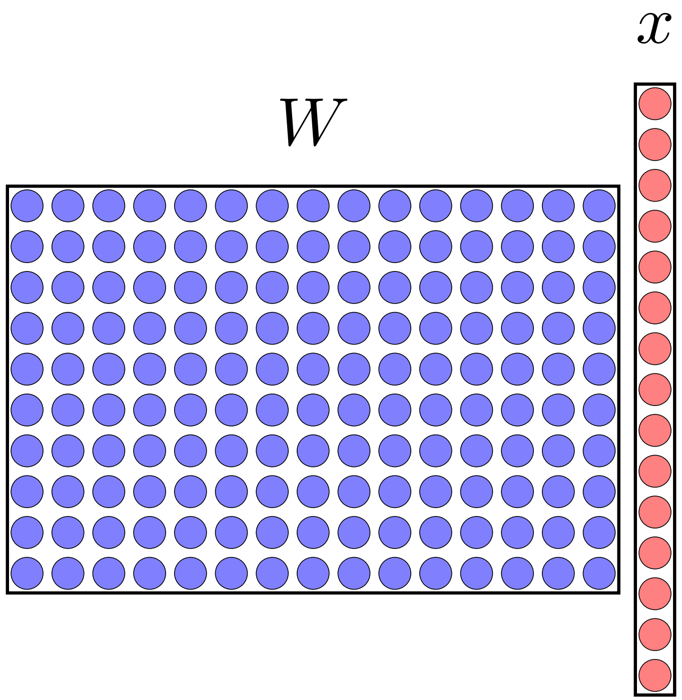
Convolutional neural networks

Recurrent neural networks

Unsupervised learning

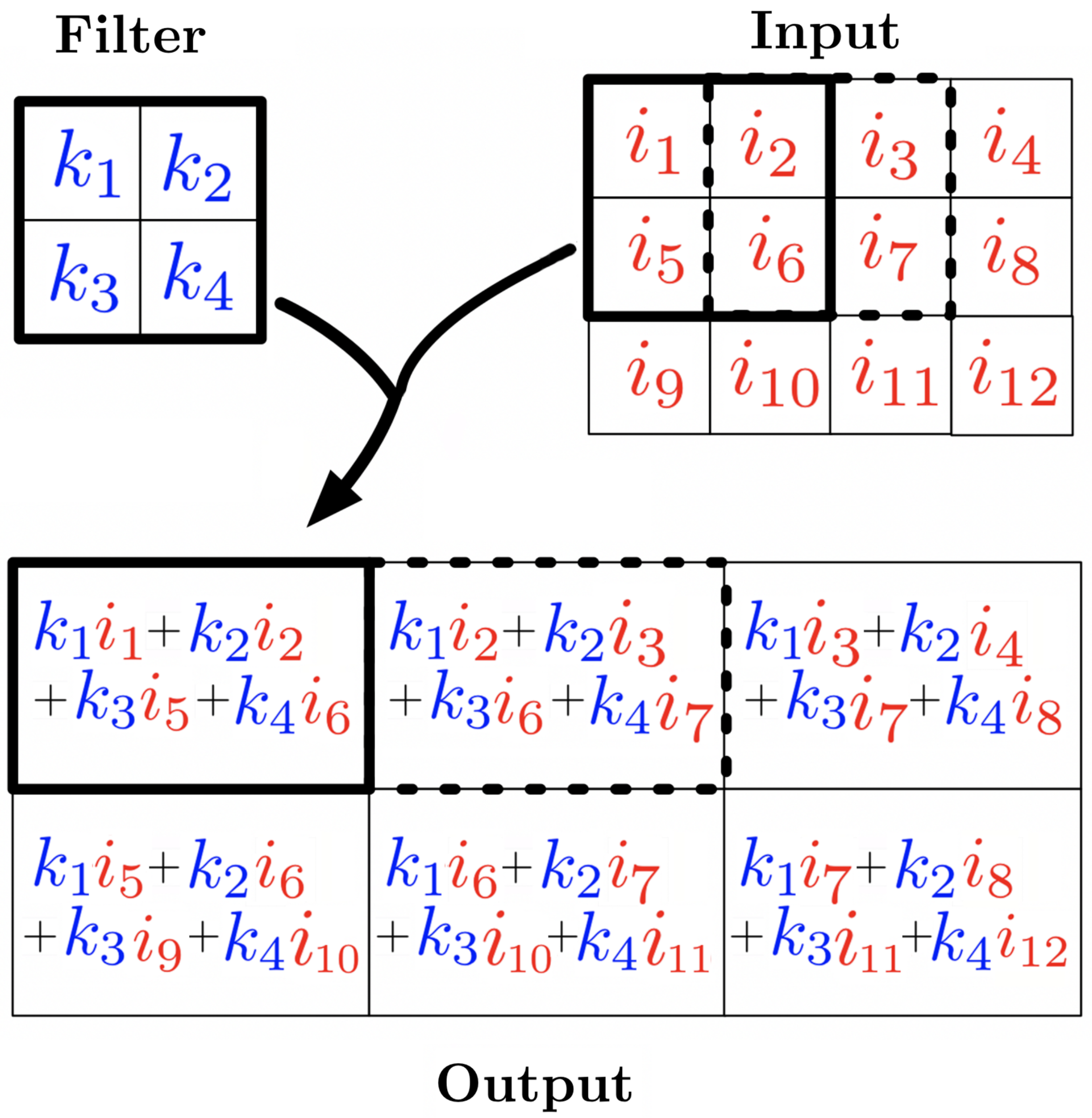
Final remarks

Motivation



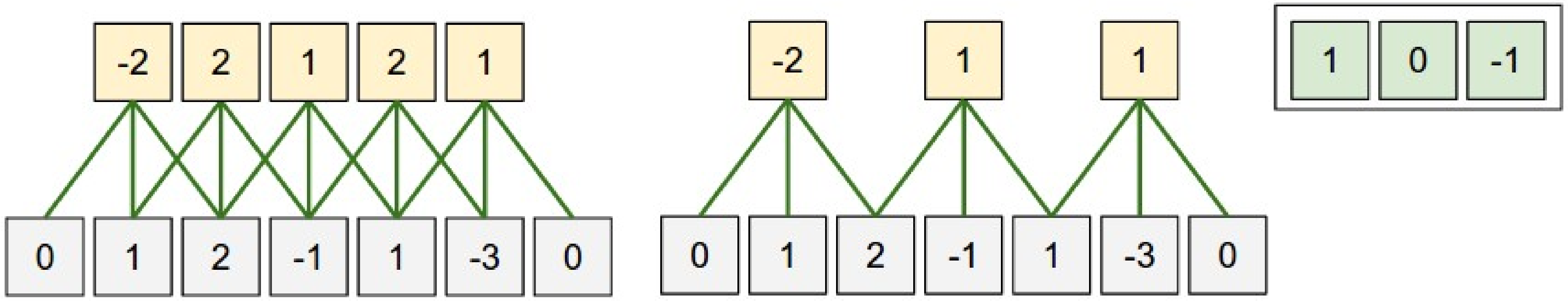
- Observation: images are not arbitrary vectors
- Goal: leverage spatial structure of images (translation invariance)

Idea: Convolutions



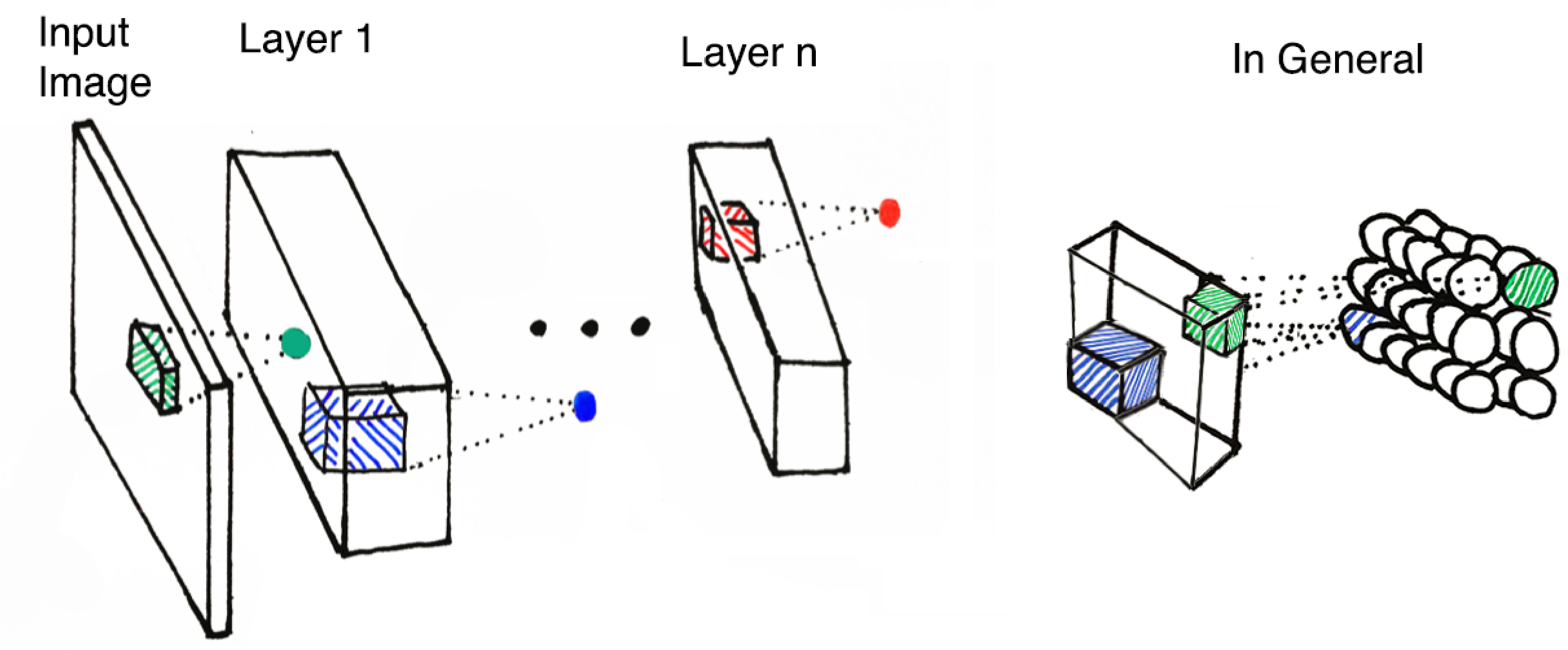
[figure from Andrej Karpathy]

Prior knowledge



- Local connectivity: each hidden unit operates on a local image patch (3 instead of 7 connections per hidden unit)
- Parameter sharing: processing of each image patch is same (3 parameters instead of $3 \cdot 5$)
- Intuition: try to match a pattern in image

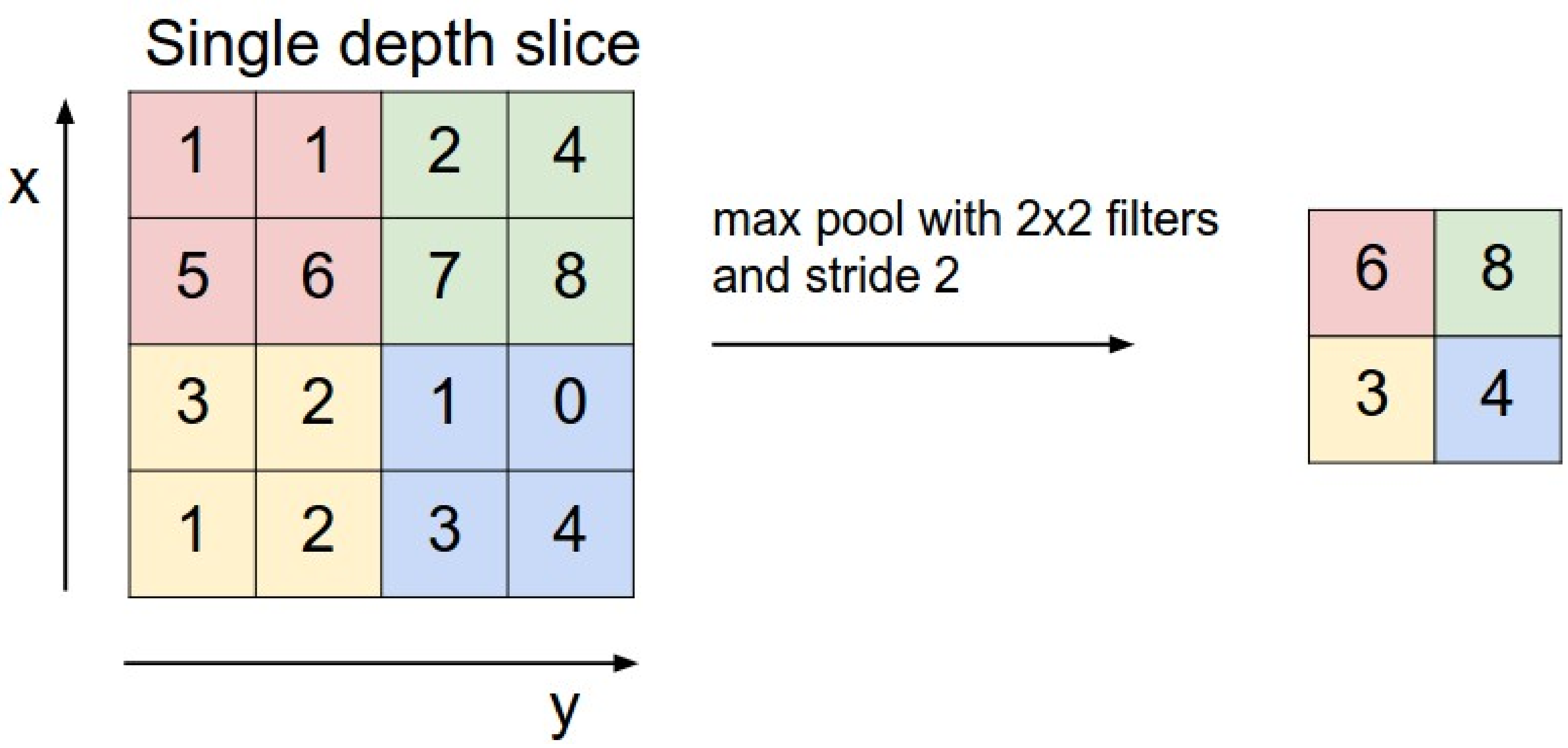
Convolutional layers



- Instead of vector to vector, we do volume to volume

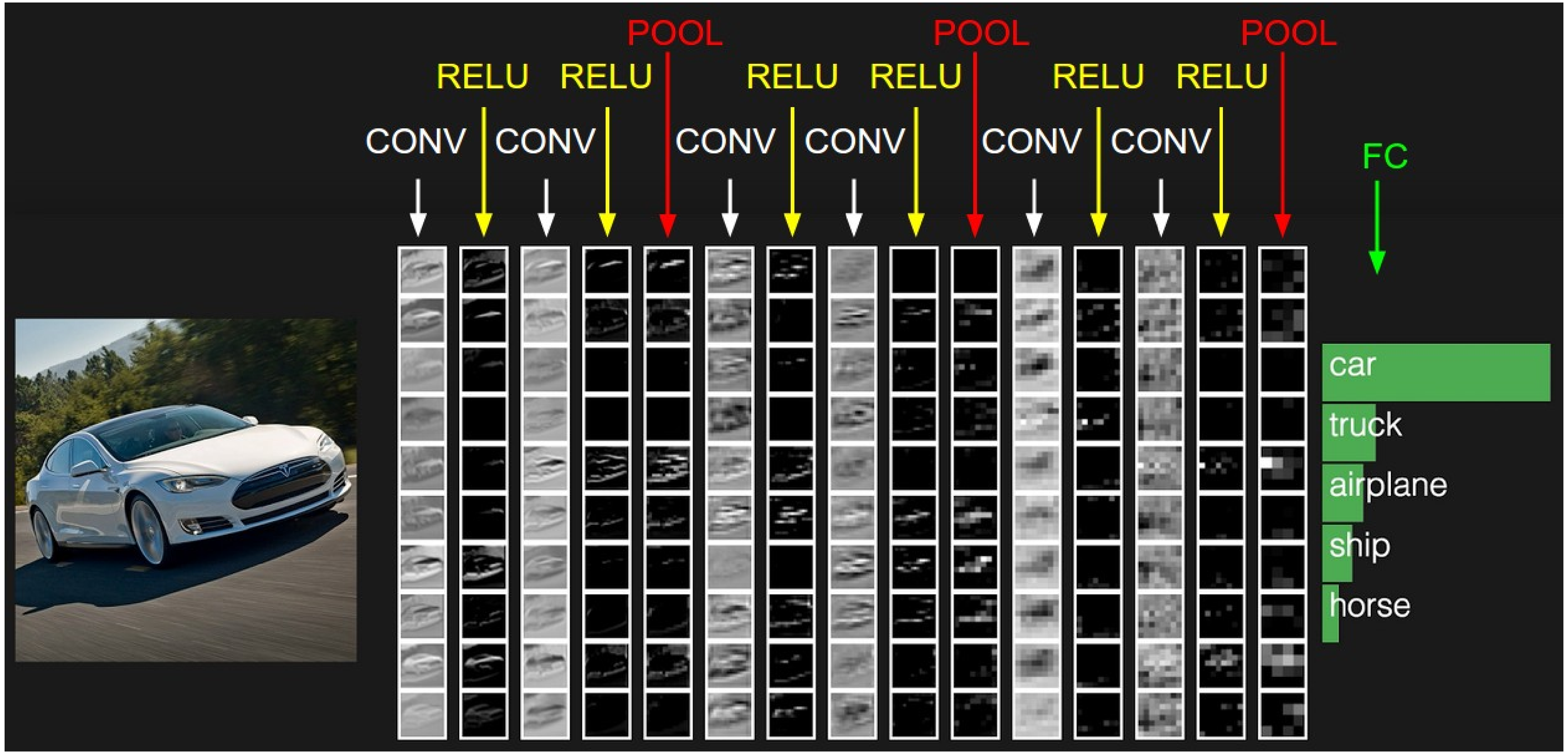
[figure from Andrej Karpathy]

Max-pooling



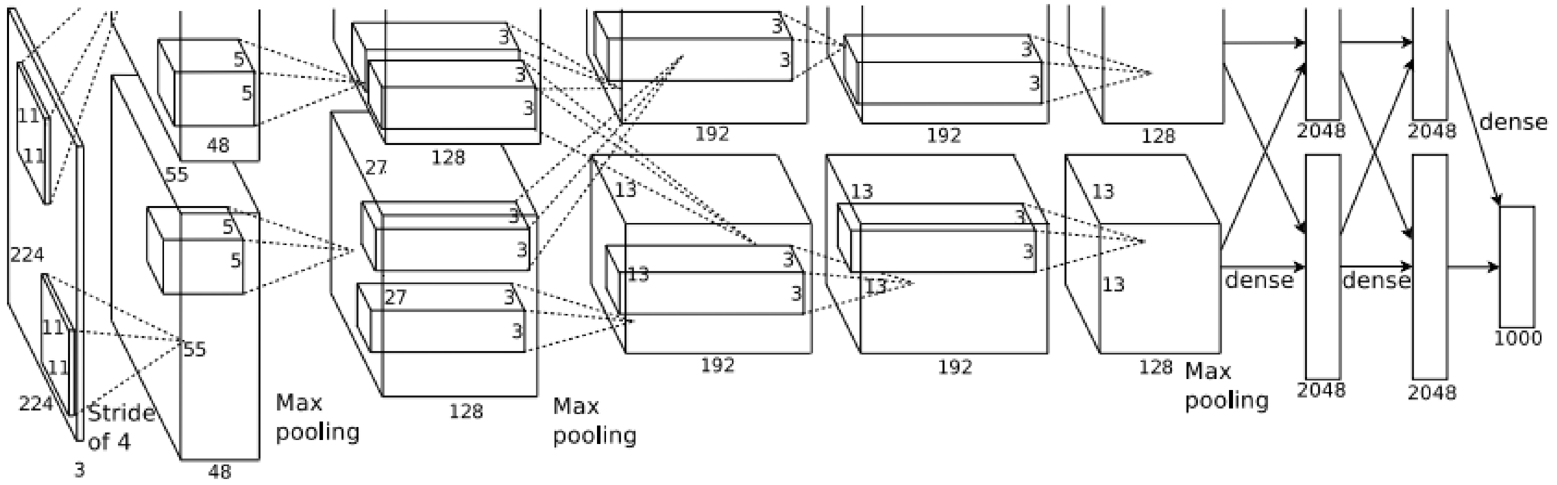
- Intuition: test if there exists a pattern in neighborhood
- Reduce computation, prevent overfitting

Example of function evaluation



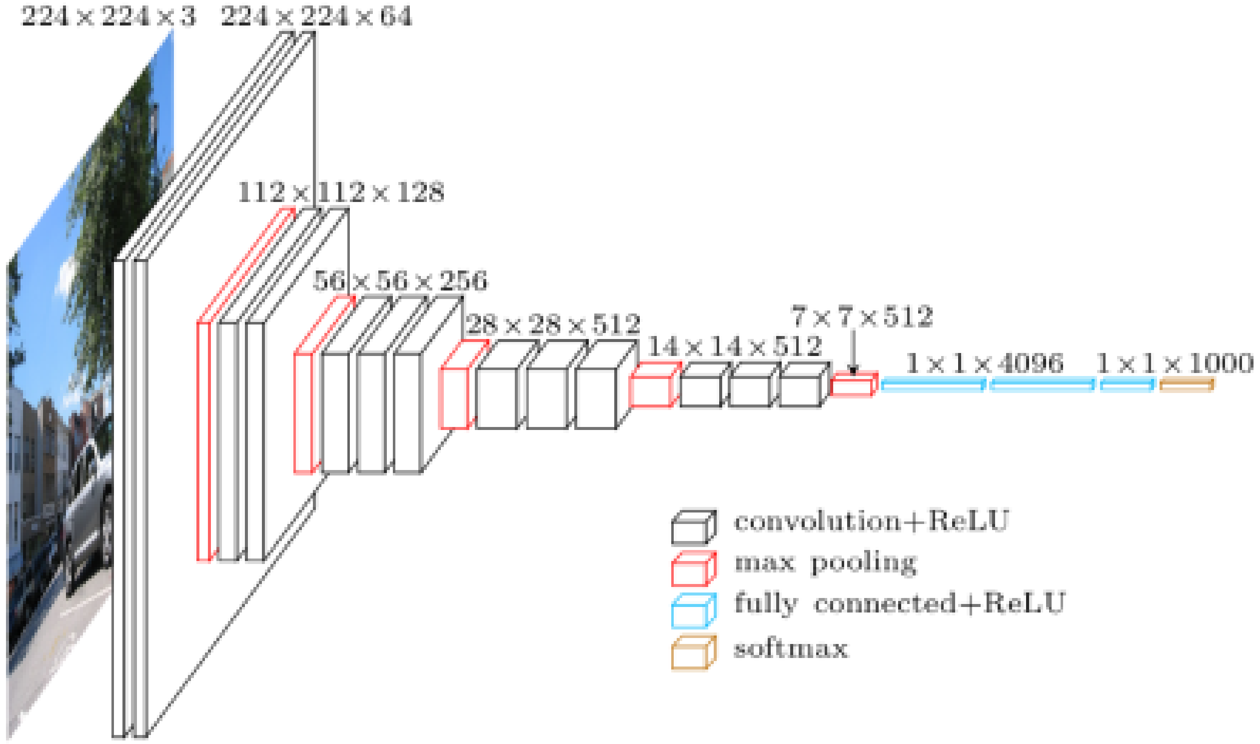
[Krizhevsky et al., 2012]

AlexNet



- Non-linearity: use RelU ($\max(z, 0)$) instead of logistic
- Data augmentation: translate, horizontal reflection, vary intensity, dropout (guard against overfitting)
- Computation: parallelize across two GPUs (6 days)
- Results on ImageNet: 16.4% error (next best was 25.8%)

VGGNet

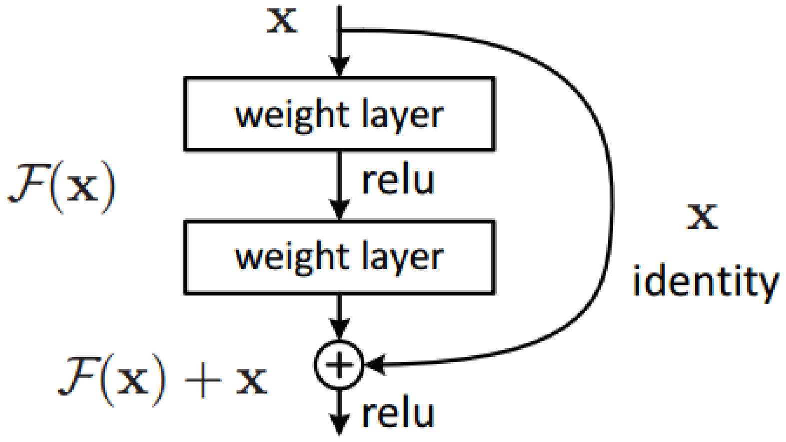


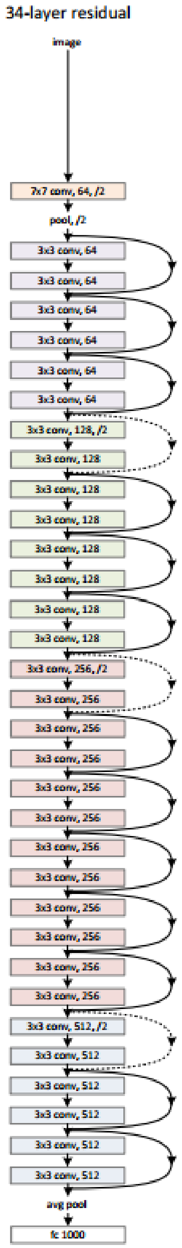
- Architecture: deeper but smaller filters; uniform
- Computation: 4 GPUs for 2-3 weeks
- Results on ImageNet: 7.3% error (AlexNet: 16.4%)

[He et al. 2015]

Residual networks

$$x \mapsto \sigma(Wx) + x$$





- Key idea: make it easy to learn the identity (good inductive bias)
- Enables training 152 layer networks
- Results on ImageNet: 3.6% error

Summary

- Key idea: locality of connections, capture spatial structure

- Filters have parameter sharing; most parameters in last fully connected layers

- Depth really matters

- Applications to text, Go, drug design, etc.

Roadmap

Feedforward neural networks

Convolutional neural networks

Recurrent neural networks

Unsupervised learning

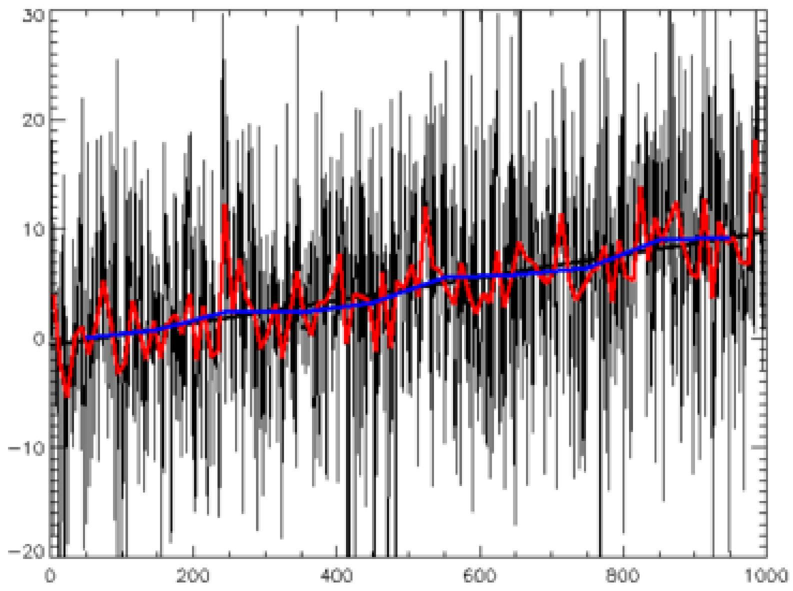
Final remarks

Motivation: modeling sequences

Sentences:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
Paris	*Talks*	*Set*	*Stage*	*for*	*Action*	*as*	*Risks*	*to*	*the*	*Climate*	*Rise*

Time series:

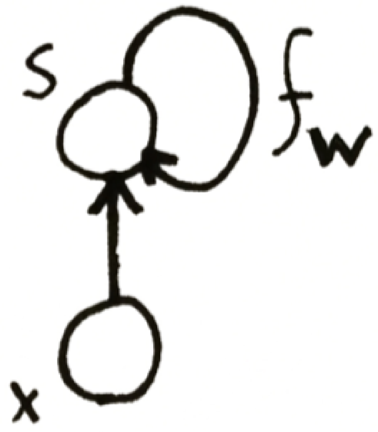


Recurrent neural networks

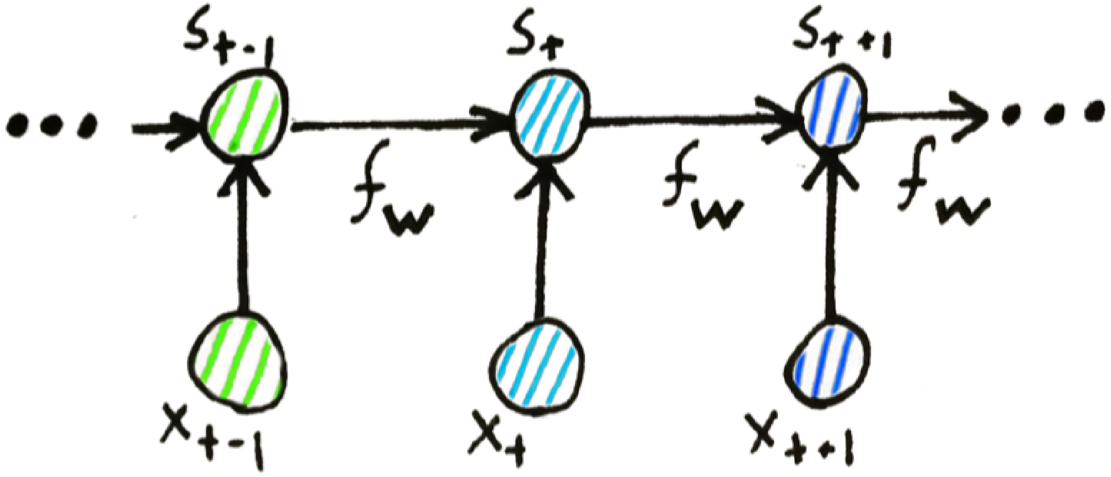
Formula

$$\boldsymbol{s}_t = f_{\boldsymbol{W}}(\boldsymbol{s}_{t-1}, \boldsymbol{x}_t)$$

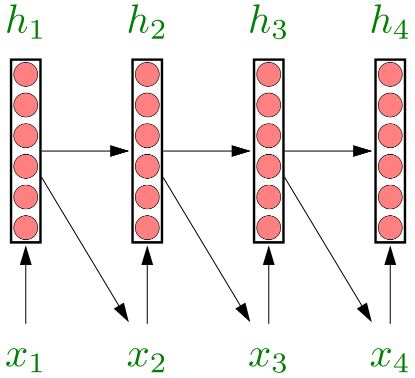
Network



Computation Graph



Recurrent neural networks



$$h_1 = \text{Encode}(x_1)$$
$$x_2 \sim \text{Decode}(h_1)$$
$$h_2 = \text{Encode}(h_1, x_2)$$
$$x_3 \sim \text{Decode}(h_2)$$
$$h_3 = \text{Encode}(h_2, x_3)$$
$$x_4 \sim \text{Decode}(h_3)$$
$$h_4 = \text{Encode}(h_3, x_4)$$

Update context vector:

$$h_t = \text{Encode}(h_{t-1}, x_t)$$

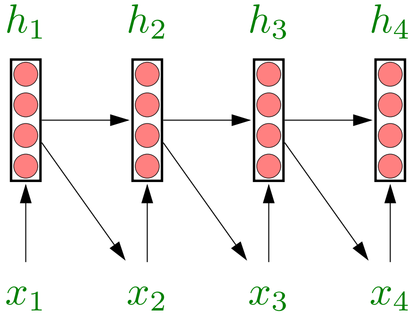
Predict next character:

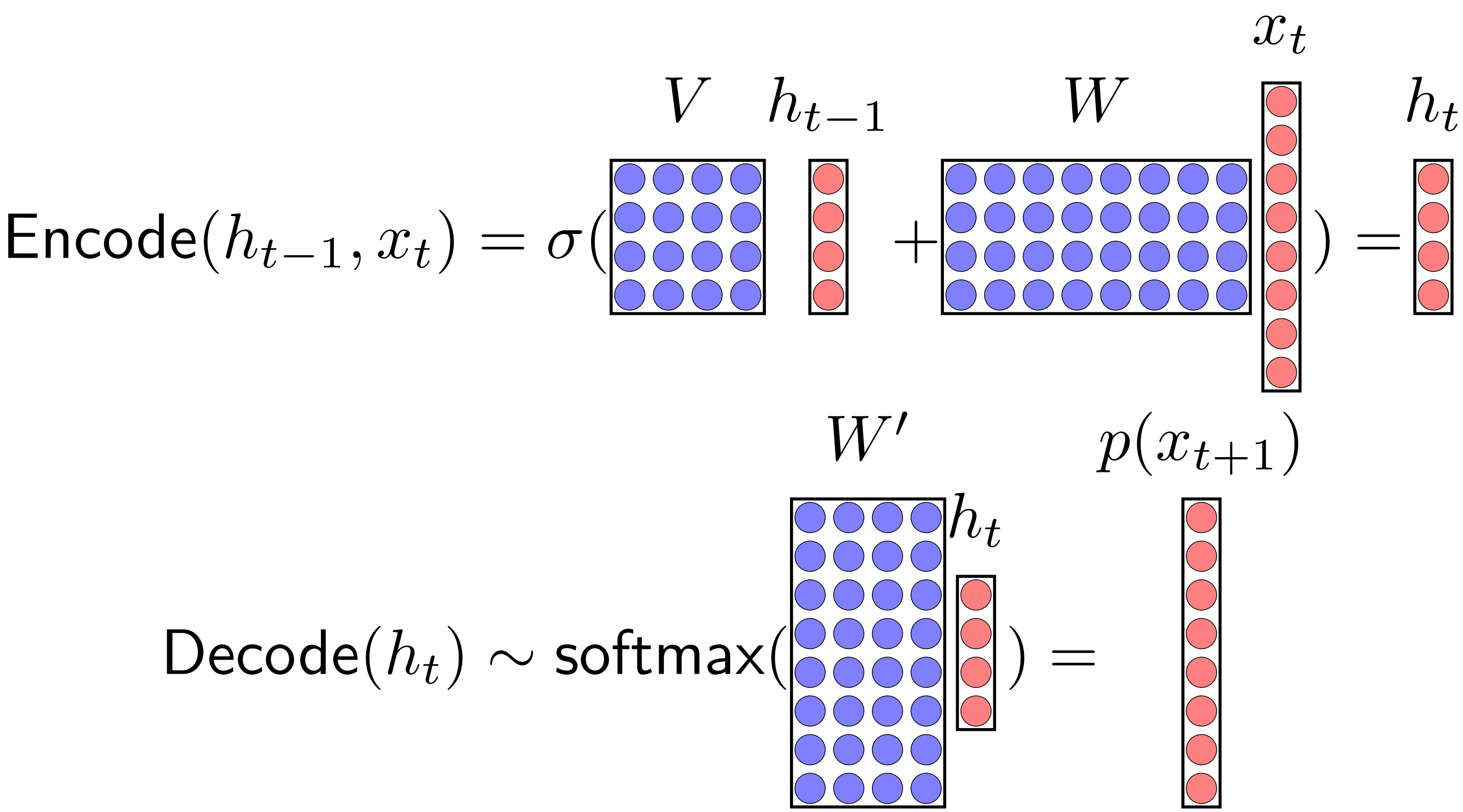
$$x_{t+1} = \text{Decode}(h_t)$$

$$\boxed{\text{context } h_t \text{ compresses } x_1, \ldots x_t}$$

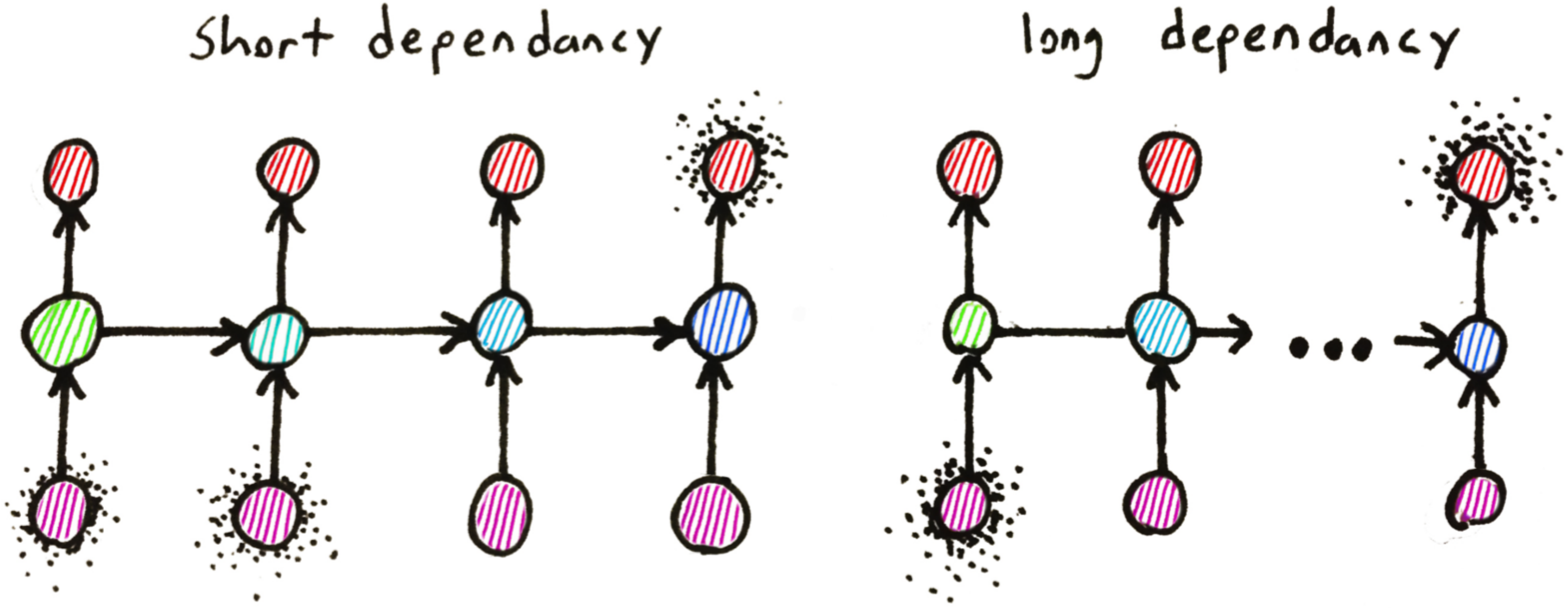
[Elman, 1990]

Simple recurrent network



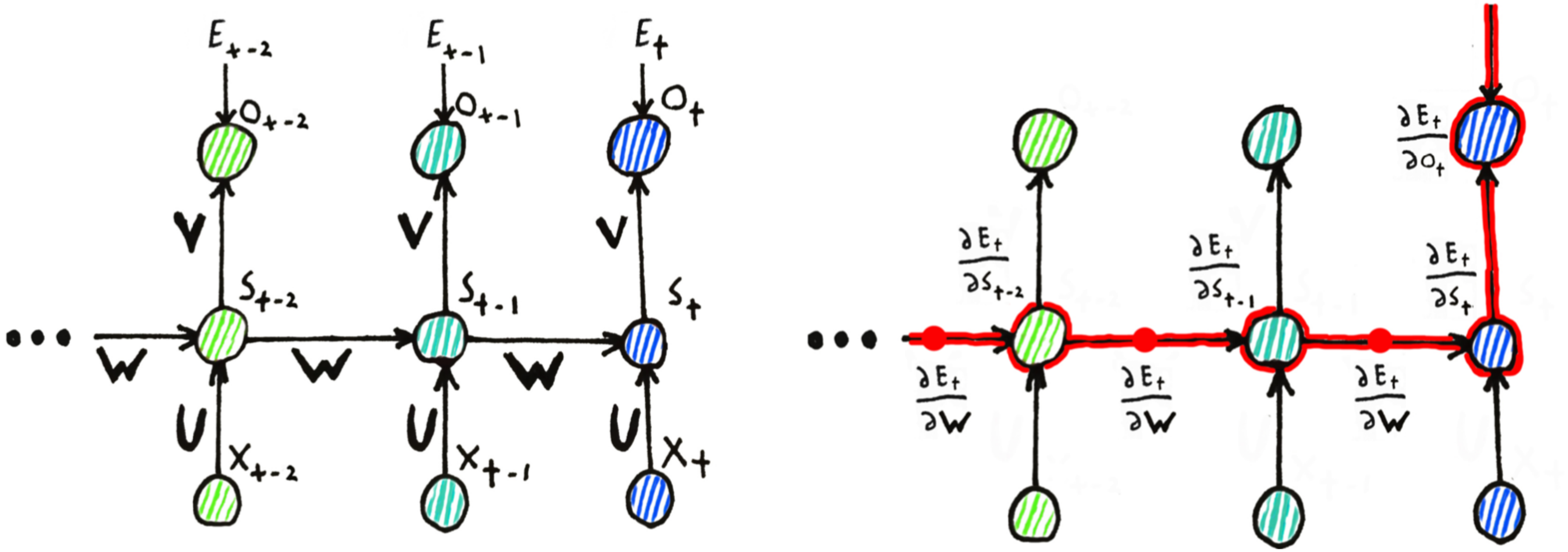


Vanishing gradient problem



- RNNs can have long or short dependancies
- When there are long dependancies, gradients have trouble backpropagating through

Vanishing gradient problem



$$\frac{\partial E_t}{\partial \boldsymbol{W}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \boldsymbol{o}_t} \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{s}_t} \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_k} \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{W}}$$

Chain rule => multiplications

Can explode or shrink!

$$\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_k} = \prod_{j=k+1}^{t} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}}$$

[Schmidhuber & Hochreiter, 1997]

Long Short Term Memory (LSTM)

API:

$$(h_t, c_t) = \mathsf{LSTM}(h_{t-1}, c_{t-1}, x_t)$$

Input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i)$$

Forget gate (initialize with b_f large, so close to 1):

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f)$$

Cell: additive combination of RNN update with previous cell

$$c_t = i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) + f_t \odot c_{t-1}$$

Output gate:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o)$$

Hidden state:

$$h_t = o_t \odot \tanh(c_t)$$

Compared with RNN, LSTM can handle the information in memory for the long period of time---remembering information for long period of time.

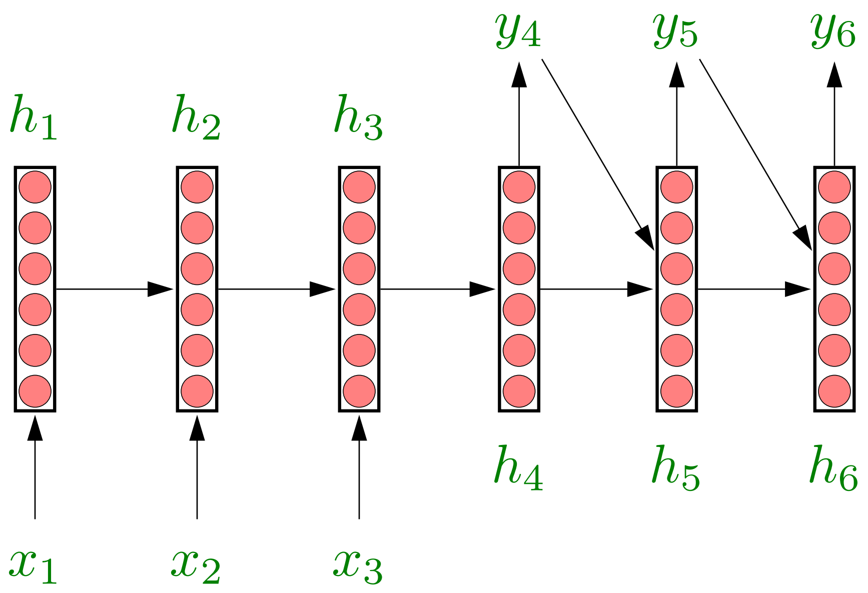
[Sutskever et al., 2014]

Sequence-to-sequence model

Motivation: machine translation

x: *Je crains l'homme de un seul livre.*

y: *Fear the man of one book.*



Sequence-to-sequence models are not a type of neural network (like RNN or LSTM), but rather a framework for solving sequence transduction problems by using RNN or LSTM.

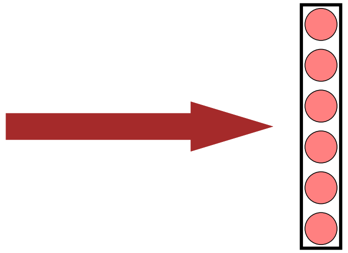
Read in a sentence first, output according to RNN:

$$h_t = \text{Encode}(h_{t-1}, x_t \text{ or } y_{t-1}), \quad y_t = \text{Decode}(h_t)$$

Attention-based models

Motivation: long sentences — compress to finite dimensional vector?

Eine Folge von Ereignissen bewirkte, dass aus Beethovens Studienreise nach Wien ein dauerhafter und endgültiger Aufenthalt wurde. Kurz nach Beethovens Ankunft, am 18. Dezember 1792, starb sein Vater. 1794 besetzten französische Truppen das Rheinland, und der kurfürstliche Hof musste fliehen.

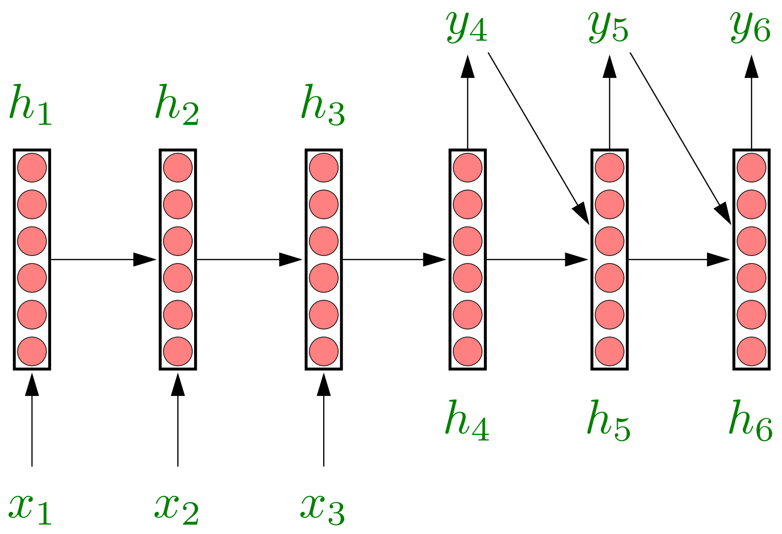


Key idea: attention

Learn to look back at your notes.

[Bahdanau et al., 2015]

Attention-based models



Distribution over input positions:

$$\alpha_t = \text{softmax}([\text{Attend}(h_1, h_{t-1}), \ldots, \text{Attend}(h_L, h_{t-1})])$$

Generate with attended input:

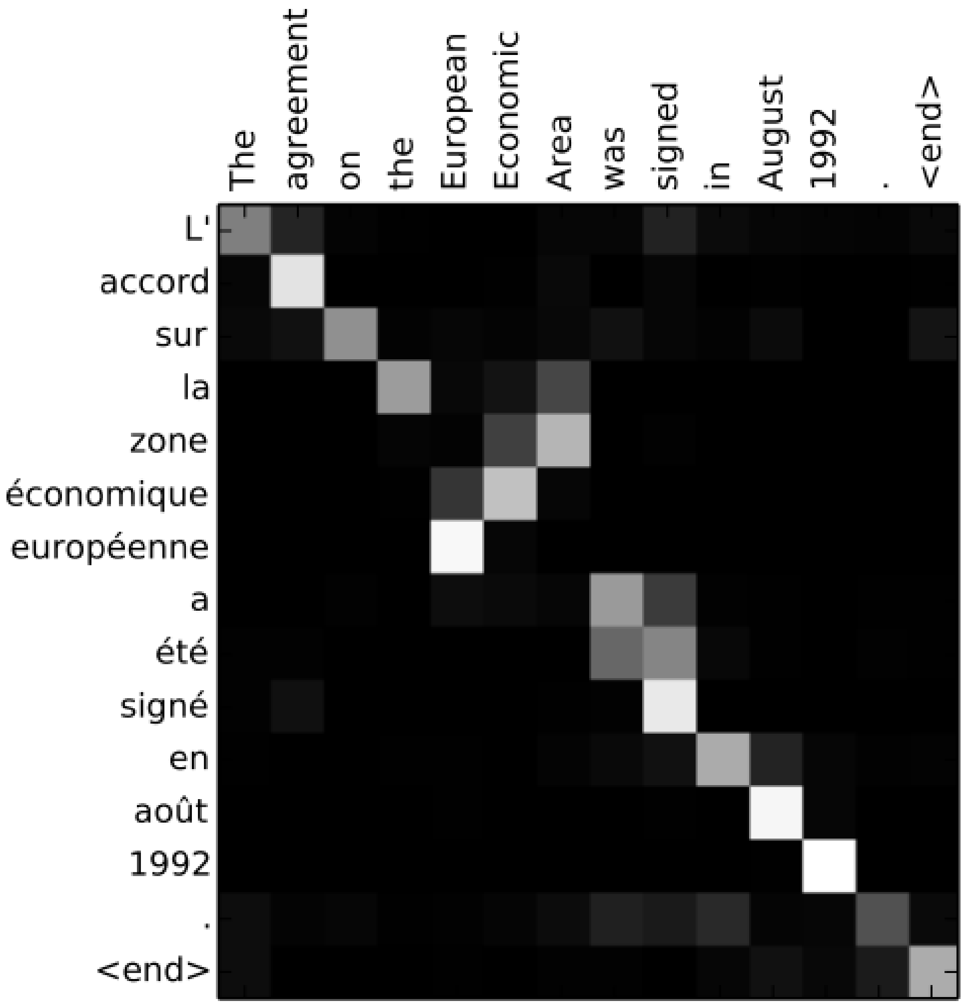
$$h_t = \text{Encode}(h_{t-1}, y_{t-1}, \sum_{j=1}^{L} \alpha_t h_j)$$

Transformer models: attention only – no RNN!

The Transformer model:

1. An encoder-decoder model
2. Uses self-attention
3. Parallel processing (Not sequential)
4. The "T" in ChatGPT

Machine translation



[Xu et al., 2015]

Image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

Summary

- Recurrent neural networks: model sequences (non-linear version of Kalman filter or HMM)
- Logic intuition: learning a program with a for loop (reduce)
- LSTMs mitigate the vanishing gradient problem
- Attention-based models: when only part of input is relevant at a time
- Newer models with "external memory": memory networks, neural Turing machines

Roadmap

Feedforward neural networks

Convolutional neural networks

Recurrent neural networks

Unsupervised learning

Final remarks

Motivation

- Deep neural networks require lot of data
- Sometimes not very much labeled data, but plenty of unlabeled data (text, images, videos)
- Humans rarely get direct supervision; can learn from raw sensory information?

Autoencoders

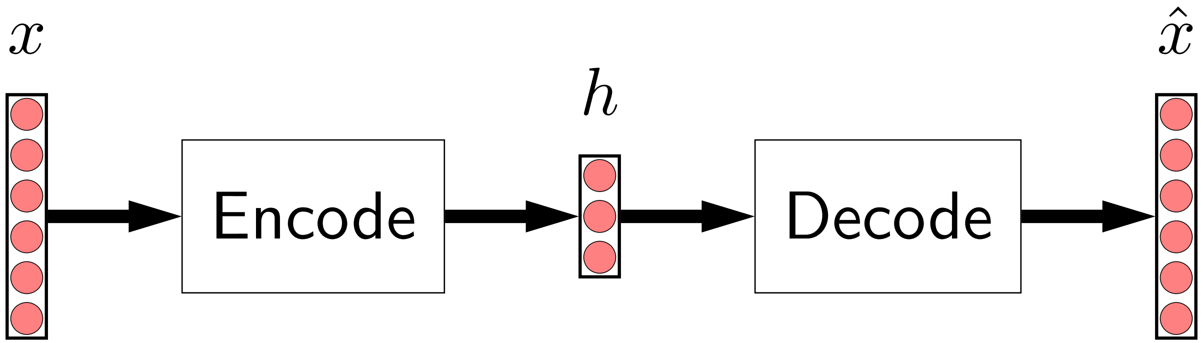
Analogy:

A A A A B B B B B $\longrightarrow$ 4 A's, 5 B's $\longrightarrow$ A A A A B B B B B

Key idea: autoencoders

If we can compress a data point and still reconstruct it, then we have learned something generally useful.

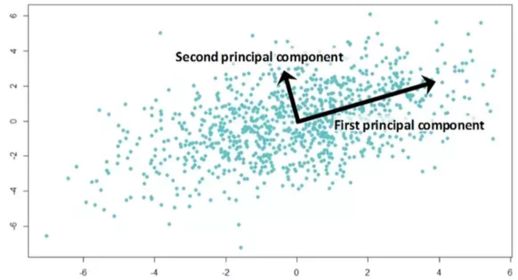
General framework:

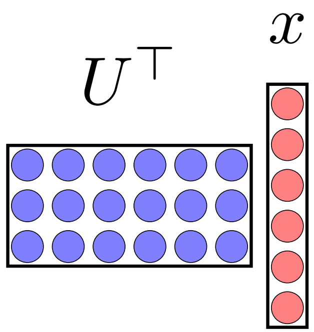


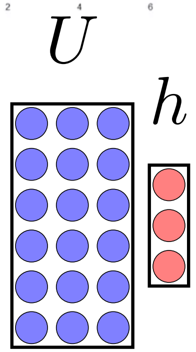
minimize $\|x - \hat{x}\|^2$

Principal component analysis

Input: points $x_1, \ldots, x_n$



Encode(x) = 

Decode(h) = 

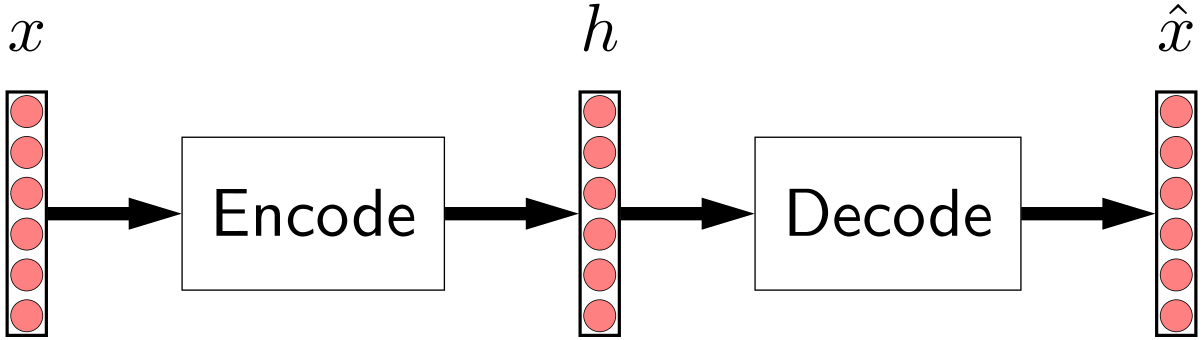
(assume x_i's are mean zero and U is orthogonal)

PCA objective:

$$\text{minimize} \sum_{i=1}^{n} \|x_i - \text{Decode}(\text{Encode}(x_i))\|^2$$

Autoencoders

Increase dimensionality of hidden dimension:

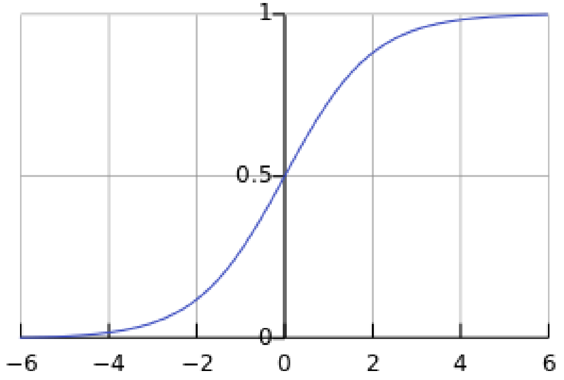


- Problem: learning nothing — just set Encode, Decode to identity function!

- Need to control complexity of Encode and Decode somehow...

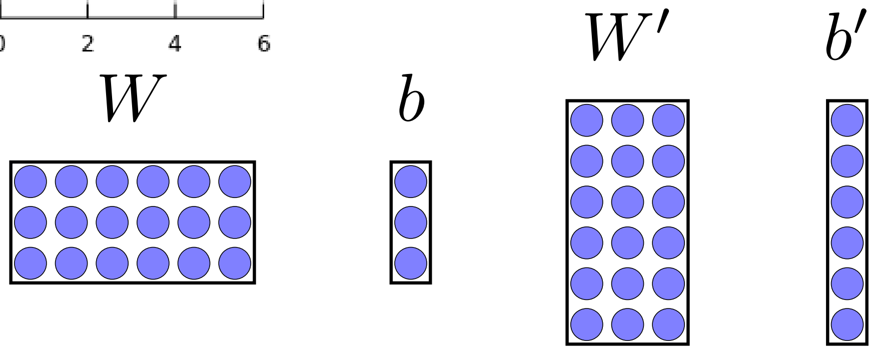
Non-linear autoencoders

Non-linear transformation (e.g., logistic function):



$$\text{Encode}(x) = \sigma(Wx + b)$$

$$\text{Decode}(h) = \sigma(W'h + b')$$

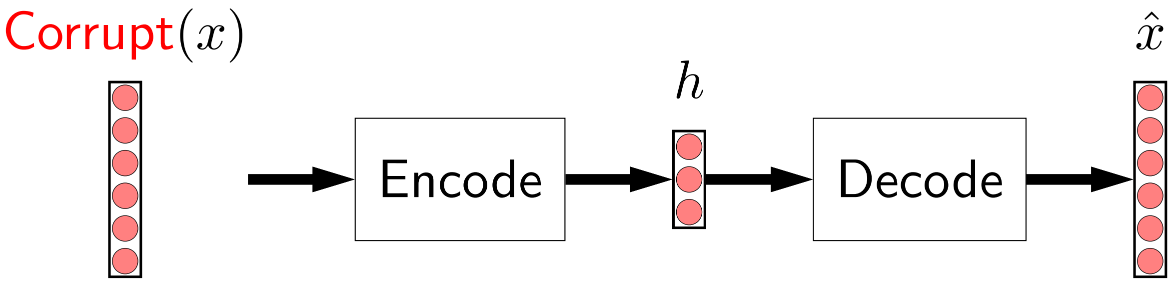


Loss function:

$$\text{minimize } \|x - \text{Decode}(\text{Encode}(x))\|^2$$

Key: non-linearity makes life harder, prevents degeneracy

Denoising autoencoders



Types of noise:

- Blankout: Corrupt$([1, 2, 3, 4]) = [0, 2, 3, 0]$
- Gaussian: Corrupt$([1, 2, 3, 4]) = [1.1, 1.9, 3.3, 4.2]$

Objective:

$$\text{minimize } \|x - \text{Decode}(\text{Encode}(\text{Corrupt}(x)))\|^2$$

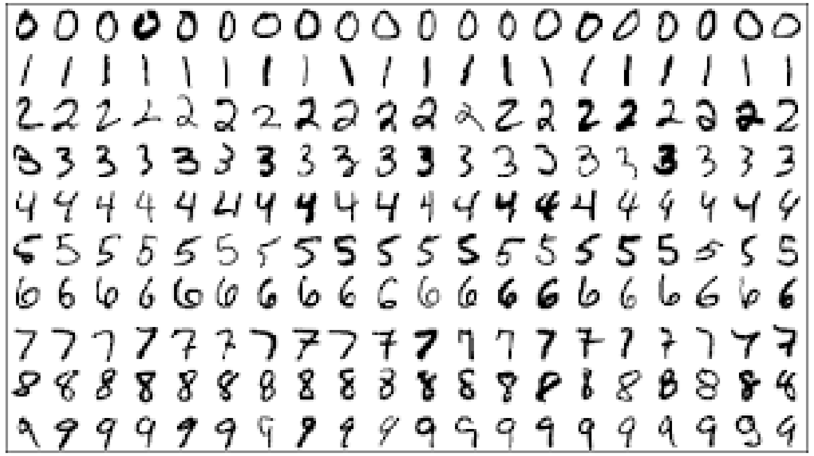
Algorithm: pick example, add fresh noise, SGD update

Key: noise makes life harder, prevents degeneracy

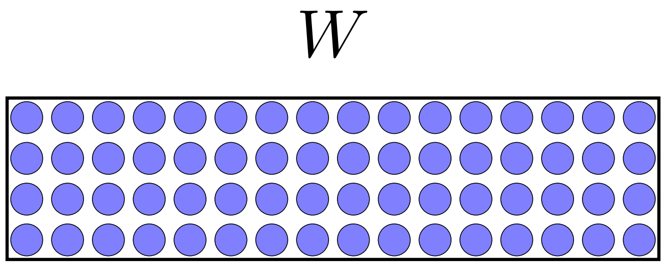
[Figure 7 of Vincent et al. (2010)]

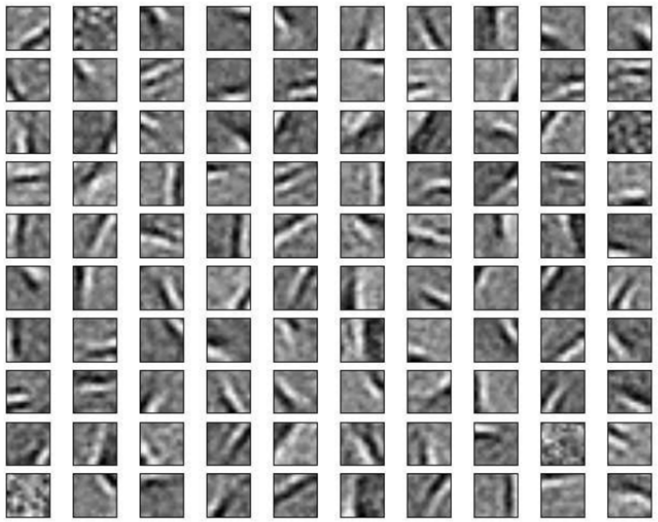
Denoising autoencoders

MNIST: 60,000 images of digits (784 dimensions)

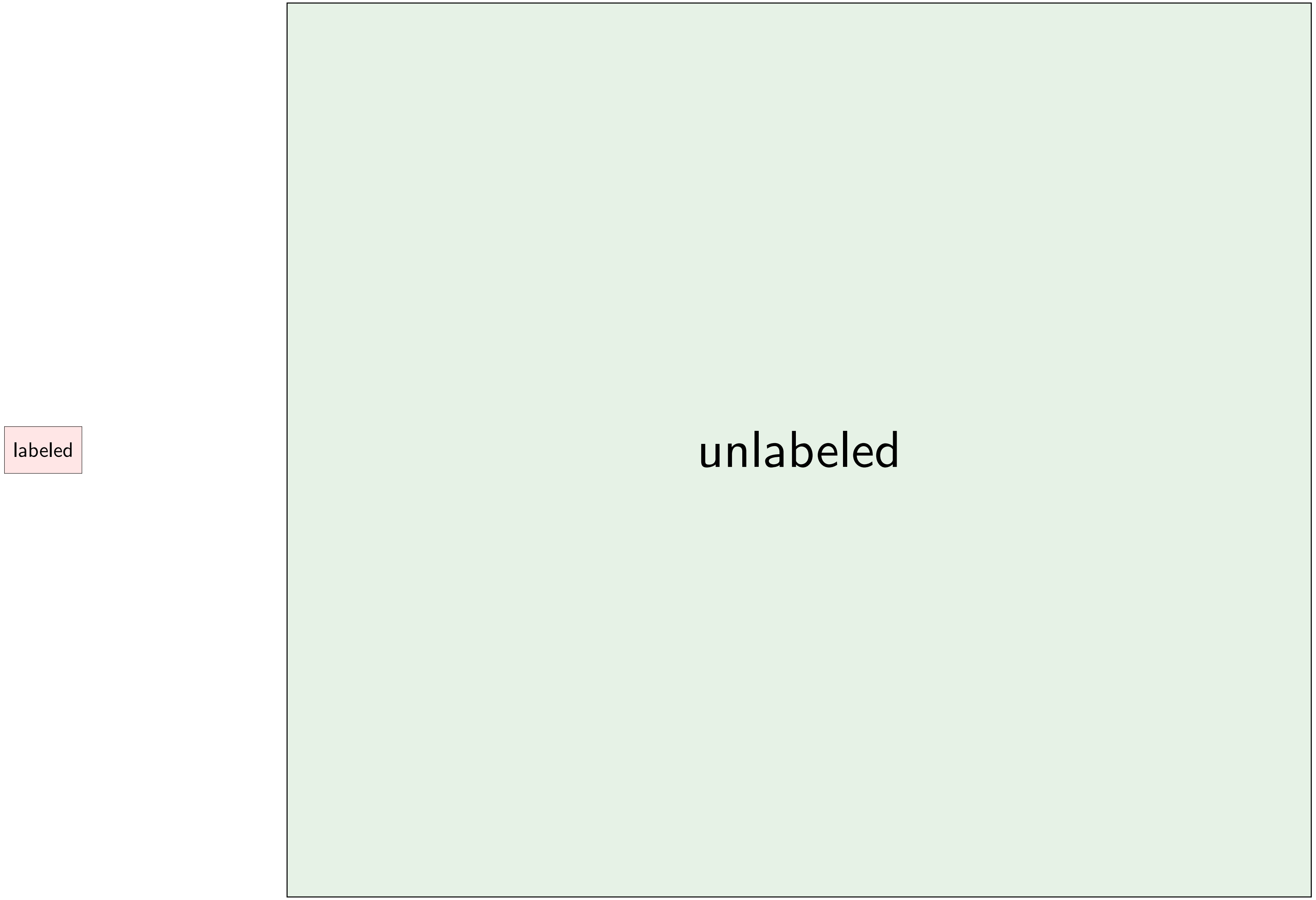


200 learned filters (rows of W):





Unsupervised pre-training



[Devlin+ 2018]

BERT

(Bidirectional Encoder Representations from Transformers, Google 2018)

Paris Talks ___ Stage for _____ as Risks to ___ Climate Rise

↓

Paris Talks Set Stage for Action as Risks to the Climate Rise

- Tasks: fill in words, predict whether is next sentence
- Trained on 3.3B words, 4 days on 64 TPUs

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google A.I.*	**87.433**	**93.160**
2 Oct 05, 2018	BERT (single model) *Google A.I.*	85.083	91.835
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
2 Sep 26, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.954	91.677
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490
4 Jul 08, 2018	r-net (ensemble) *Microsoft Research Asia*	84.003	90.147
5 Mar 19, 2018	QANet (ensemble) *Google Brain & CMU*	83.877	89.737
5 Sep 09, 2018	nlnet (single model) *Microsoft Research Asia*	83.468	90.133
5 Jun 20, 2018	MARS (ensemble) *YUANFUDAO research NLP*	83.982	89.796
6 Sep 01, 2018	MARS (single model) *YUANFUDAO research NLP*	83.185	89.547

Unsupervised learning

- Principle: make up prediction tasks (e.g., x given x or context)
- Hard task $\rightarrow$ pressure to learn something
- Loss minimzation using SGD
- Discriminatively fine tune: initialize feedforward neural network and backpropagate to optimize task accuracy
- How far can one push this?

Roadmap

Feedforward neural networks

Convolutional neural networks

Recurrent neural networks

Unsupervised learning

Final remarks

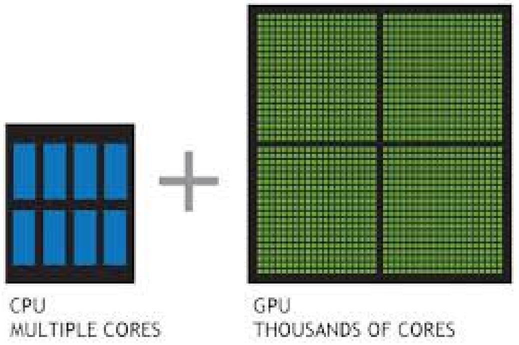
Getting things to work

Better optimization algorithms: SGD, SGD+momentum, AdaGrad, AdaDelta, momentum, Nesterov, Adam

Tricks: initialization, gradient clipping, batch normalization, dropout

More hyperparameter tuning: step sizes, architectures

Better hardware: GPUs, TPUs



...wait for a long time...

Theory: why does it work?

Two questions:

- Approximation: why are neural networks good hypothesis classes?
- Optimization: why can SGD optimize a high-dimensional non-convex problem?

Partial answers:

- 1-layer neural networks can approximate any continuous function on compact set [Cybenko, 1989; Barron, 1993]
- Generate random features works too [Rahimi/Recht, 2009; Andoni et. al, 2014]
- Use statistical physics to analyze loss surfaces [Choromanska et al., 2014]

Summary

Phenomena	Ideas
Fixed vectors	Feedforward NNs
Spatial structure	convolutional NNs
Sequence	recurrent NNs LSTMs
Sequence-to-sequence	encoder-decoder attention-based models
Unsupervised	autoencoders any auxiliary task

Outlook

Extensibility: able to compose modules

LSTM Attend Encode

Learning programs: think about analogy with a computer

$$x \longrightarrow \boxed{f_\theta} \longrightarrow y$$

We cannot simply depend on GPU and data to achieve AGI.
New science and technology are needed.