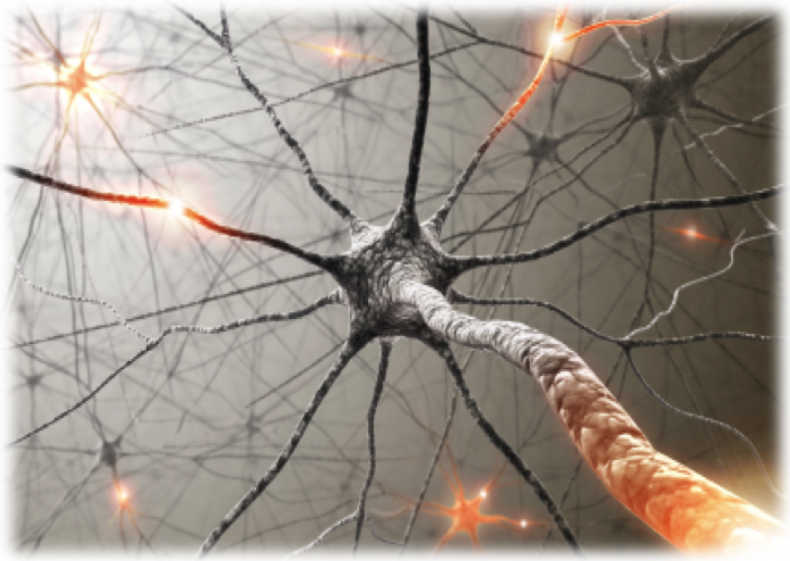
Lecture 18: Deep Learning



Reminders

- Projects due on Wednesday at noon PT (no late days allowed, to give time for grading).

- Exam regrade requests due Wednesday at midnight PT.

A brief history

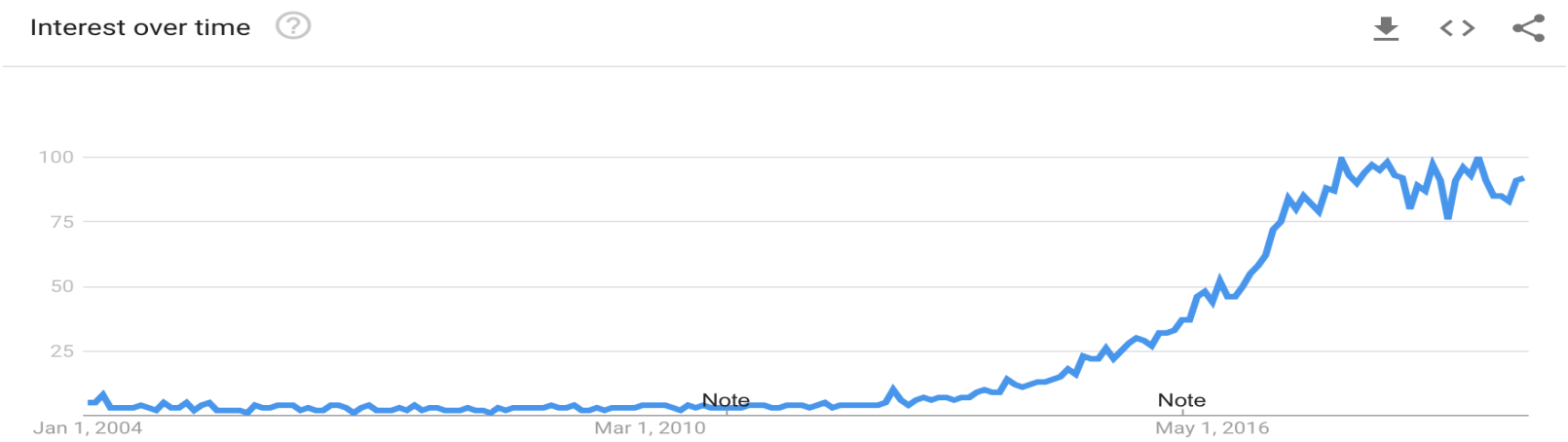
- 1943: neural networks $\Leftrightarrow$ logical circuits (McCulloch/Pitts)
- 1949: "cells that fire together wire together" learning rule (Hebb)
- 1969: theoretical limitations of neural networks (Minsky/Papert)
- 1974: backpropagation for training multi-layer networks (Werbos)
- 1986: popularization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

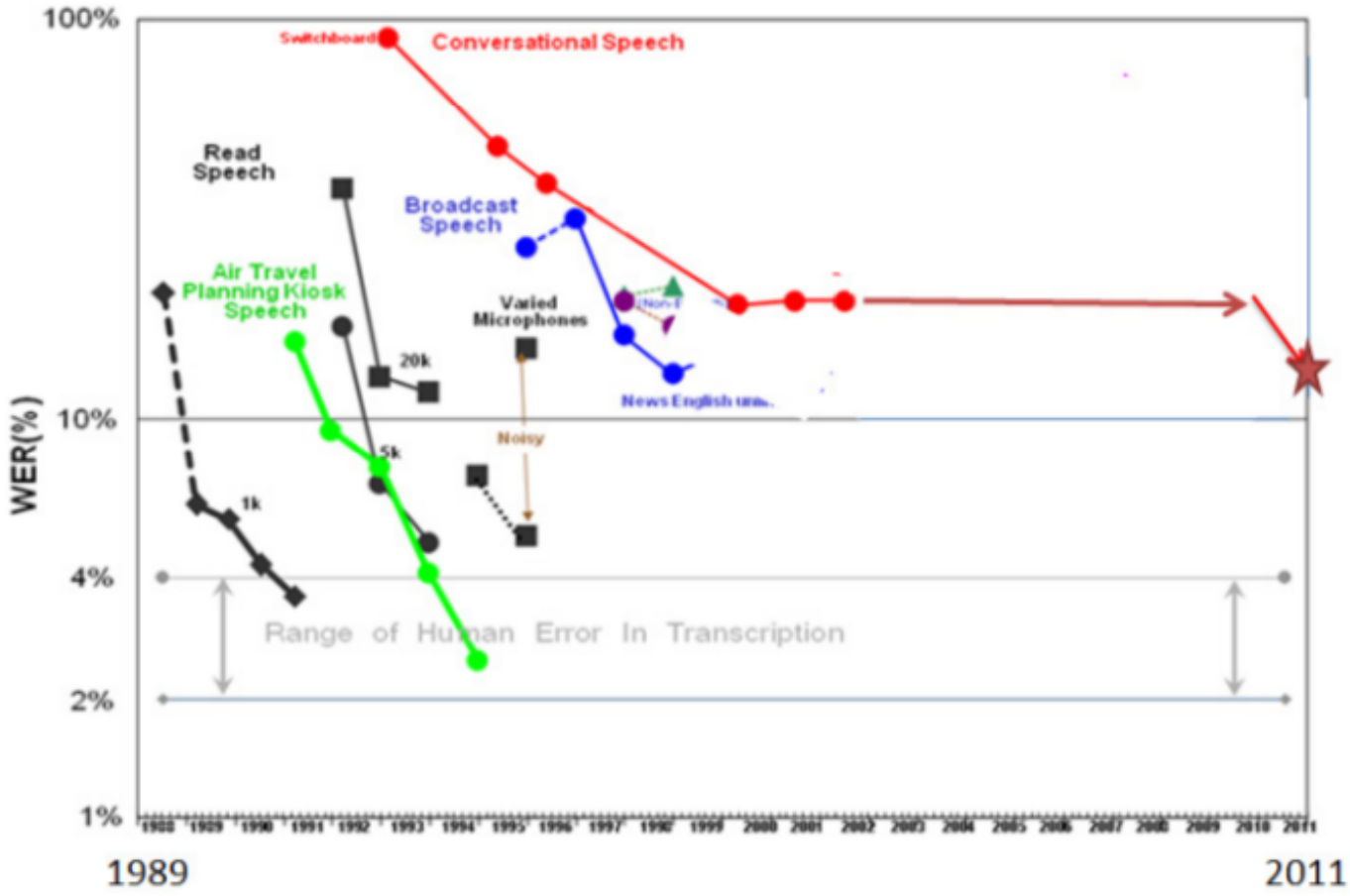
Google Trends

Query: deep learning



[figure from Li Deng]

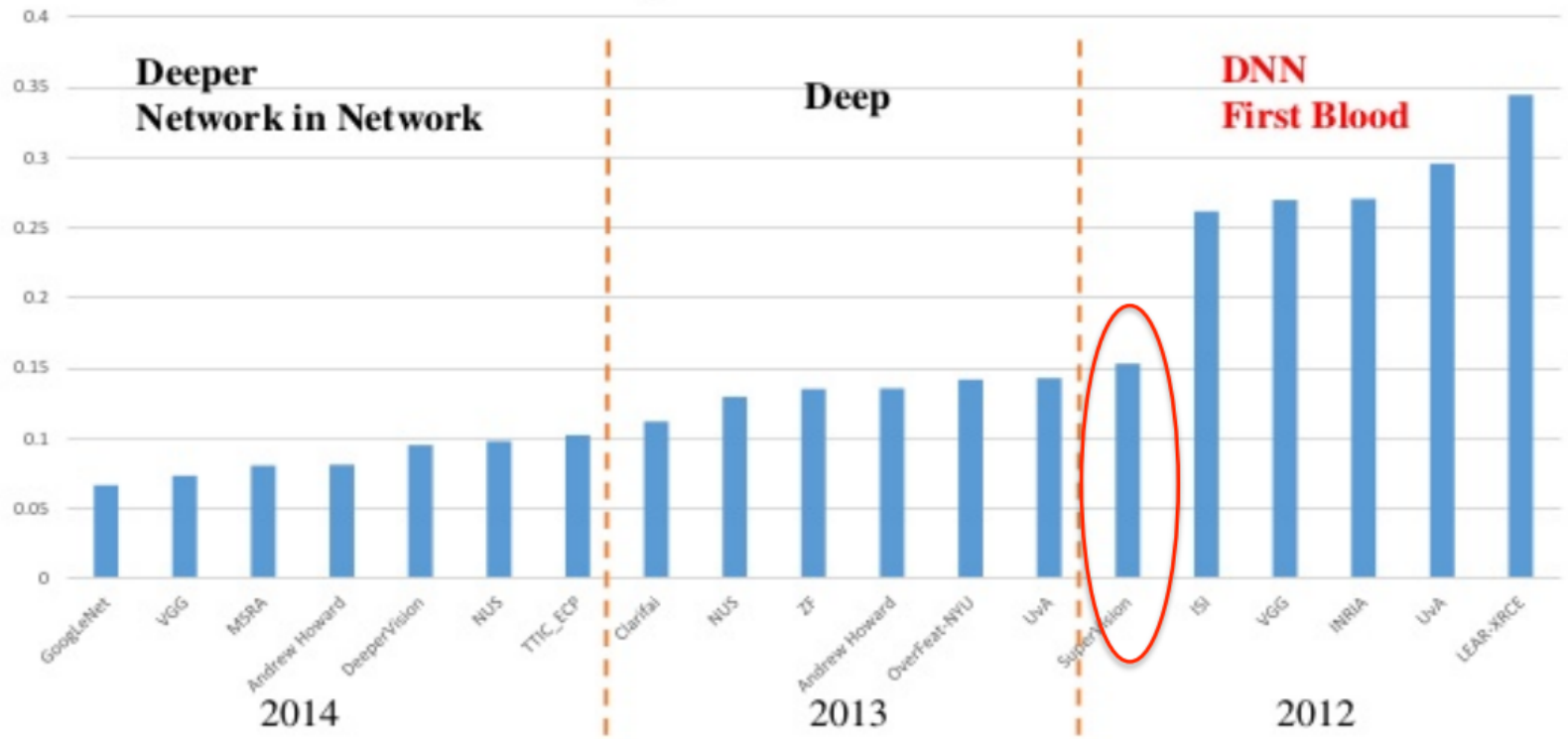
Speech recognition (2009-2011)



- Steep drop in WER due to deep learning
- IBM, Google, Microsoft all switched over from GMM-HMM

[Krizhevsky et al., 2012, a.k.a. AlexNet]

Object recognition (2012)



- Landslide win in ILSVRC object recognition competition
- Computer vision community switched to CNNs
- Simpler than hand-engineered features (SIFT)

- Before 2012, computer vision relied heavily on hand-engineered features like SIFT (Scale-Invariant Feature Transfo

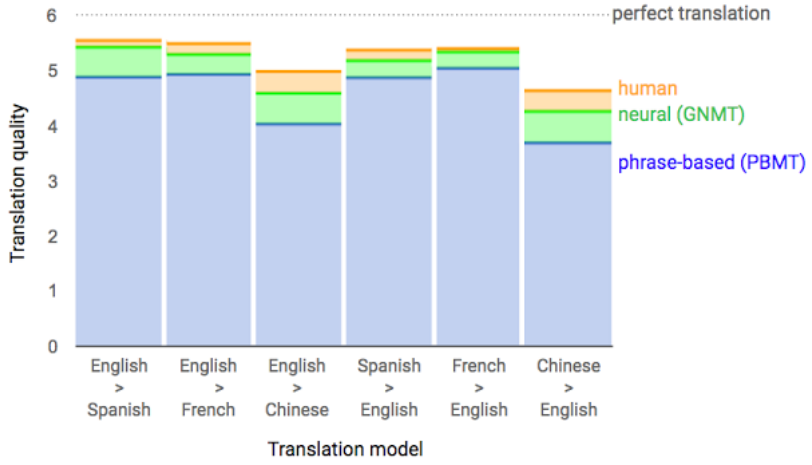
Machine translation (2016)

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

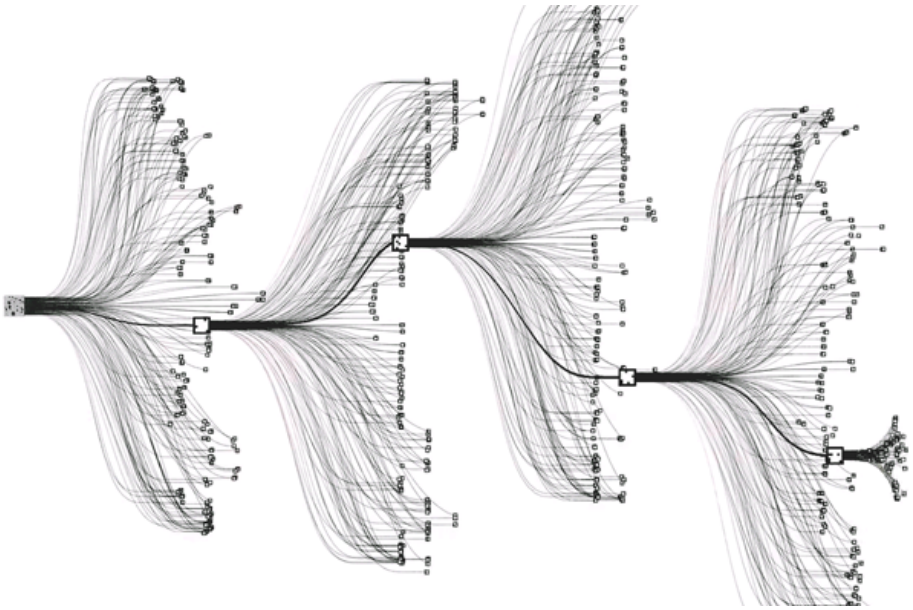
Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Input sentence:	*Translation (PBMT):*	*Translation (GNMT):*	*Translation (human):*
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.



- Decisive wins have taken longer to achieve in NLP (words are meaningful in a way that pixels are not)
- Current state-of-the-art in machine translation

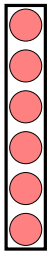
Go (2016)

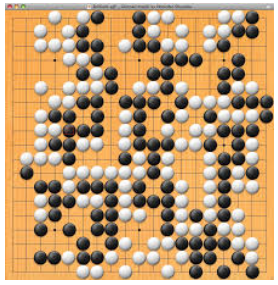


- Defeated world champion Le Sedol 4-1
- Simple architecture (in contrast, DeepBlue was search + hand-crafted heuristics)
- 2017: AlphaGoZero does not require human expert games as supervision

What is deep learning?

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





Roadmap

Feedforward neural networks

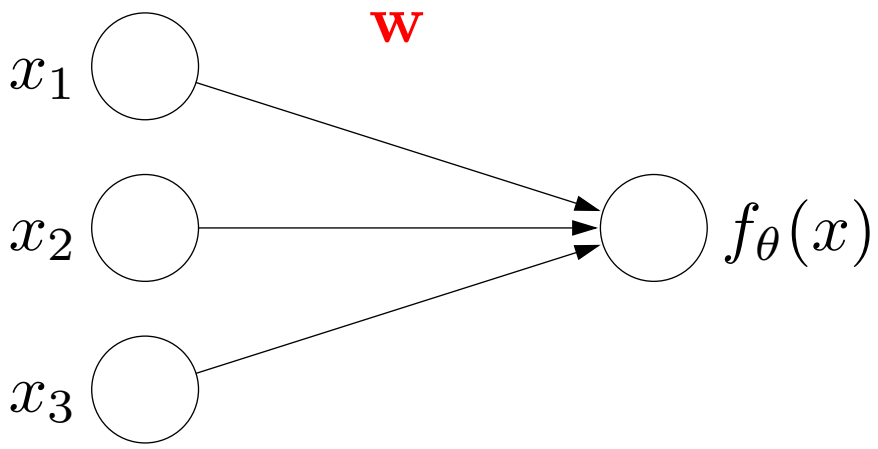
Convolutional networks

Sequence models

Unsupervised learning

Final remarks

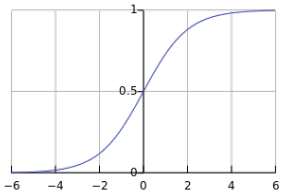
Review: linear predictors



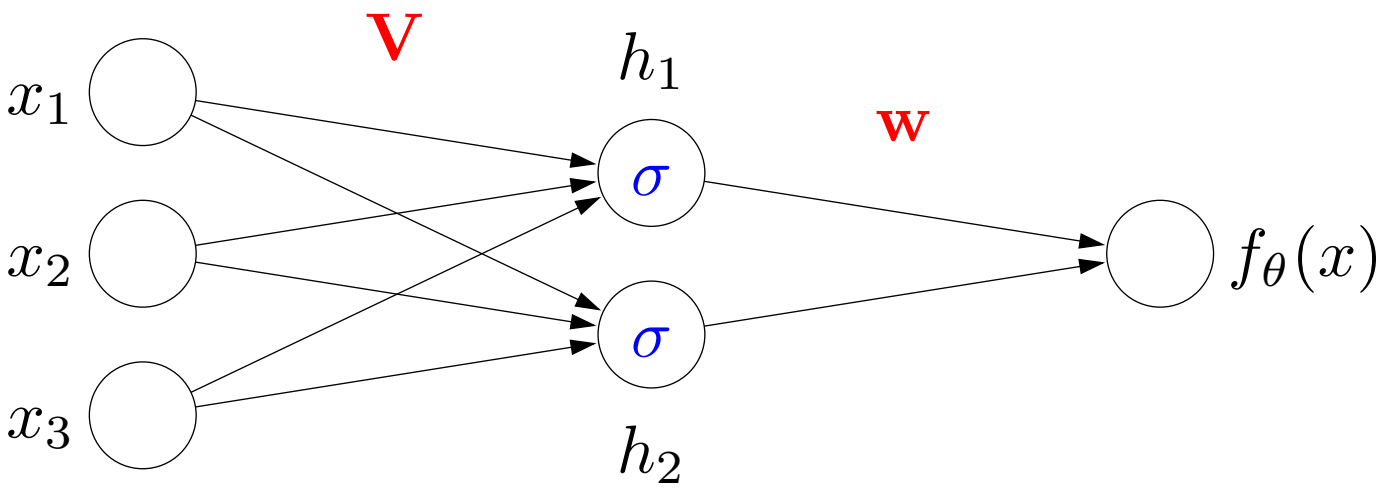
Output:

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

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



Review: neural networks



Intermediate hidden units:

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

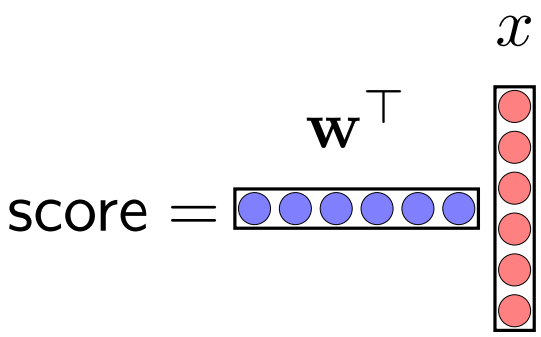
Output:

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

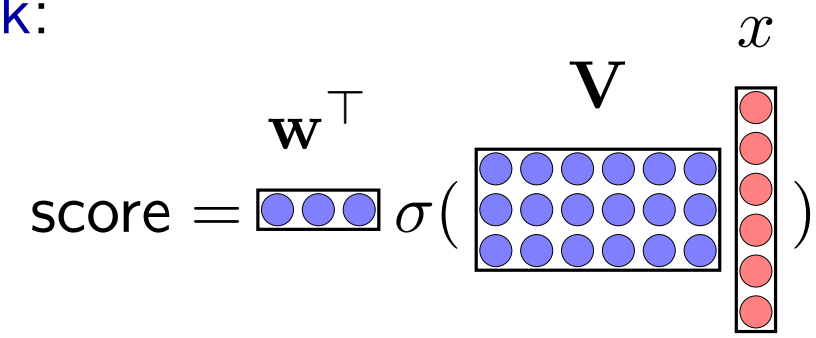
Parameters: $\theta = (\mathbf{V}, \mathbf{w})$

Deep neural networks

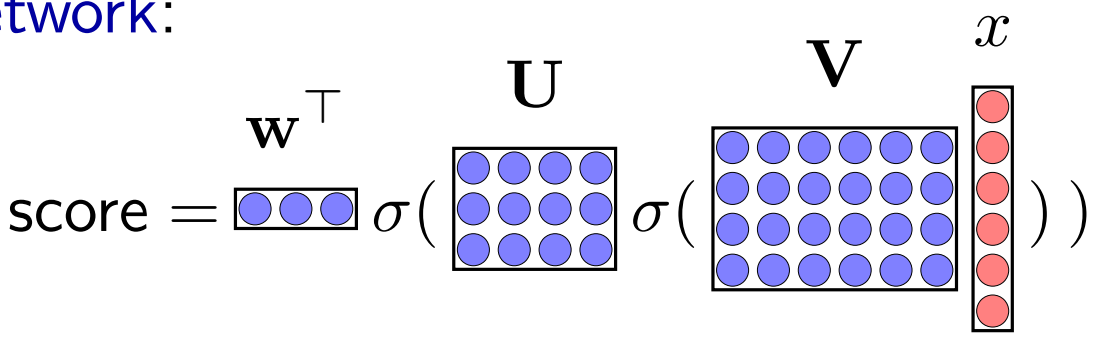
1-layer neural network:



2-layer neural network:

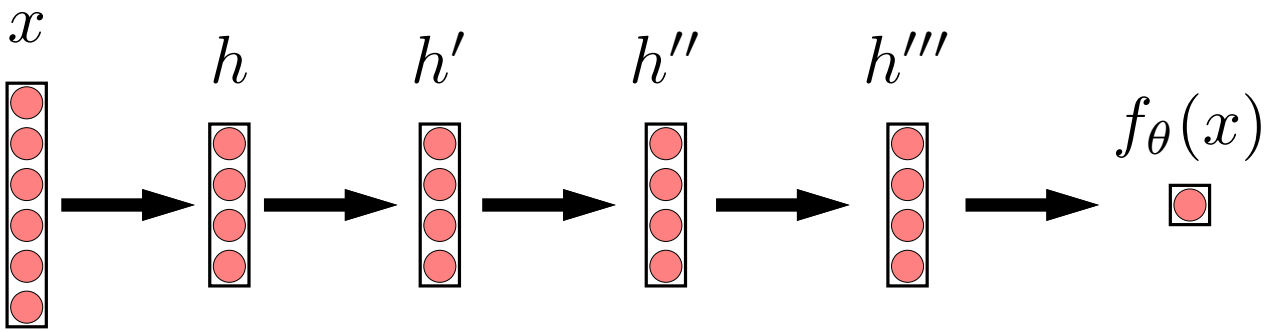


3-layer neural network:



...

Depth

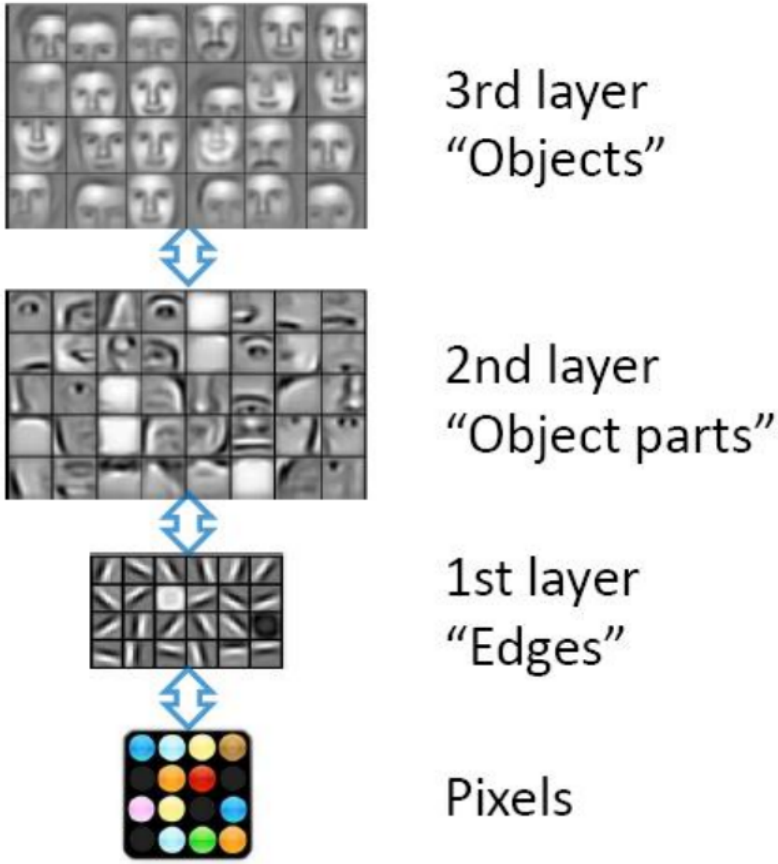


Intuitions:

- 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)
- Formal theory/understanding is still incomplete
- Some hypotheses emerging: double descent, lottery ticket hypothesis

[figure from Honglak Lee]

What's learned?



Review: optimization

Regression:

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

Key idea: minimize training loss

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

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

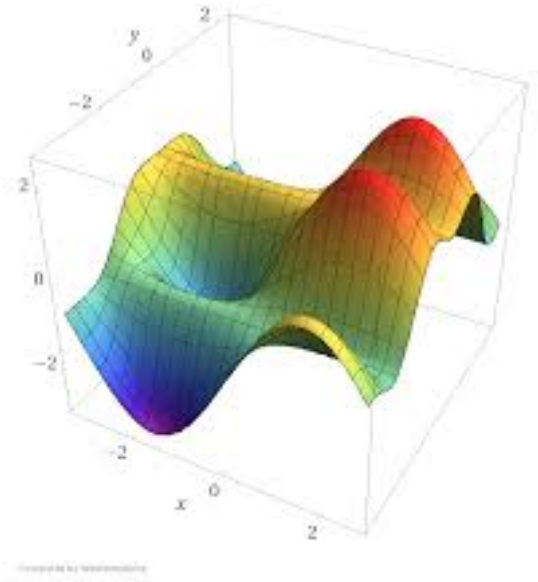
Algorithm: stochastic gradient descent

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



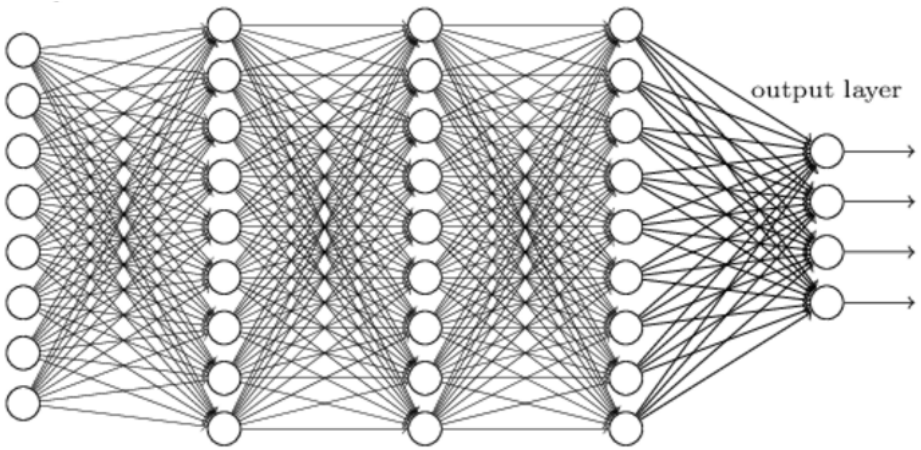
- Non-convex optimization
- No theoretical guarantees that it works
- Before 2000s, empirically very difficult to get working

What's different today

Computation (time/memory)

Information (data)

How to make it work



- More hidden units (over-parameterization)
- Adaptive step sizes (AdaGrad, Adam)
- Dropout to guard against overfitting
- Careful initialization (pre-training)
- Batch normalization

Model and optimization are tightly coupled

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

Roadmap

Feedforward neural networks

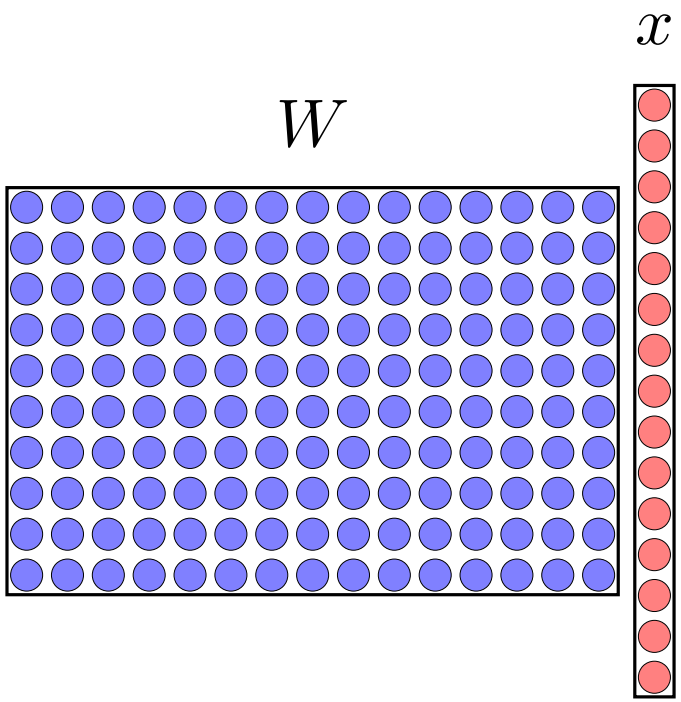
Convolutional networks

Sequence models

Unsupervised learning

Final remarks

Motivation

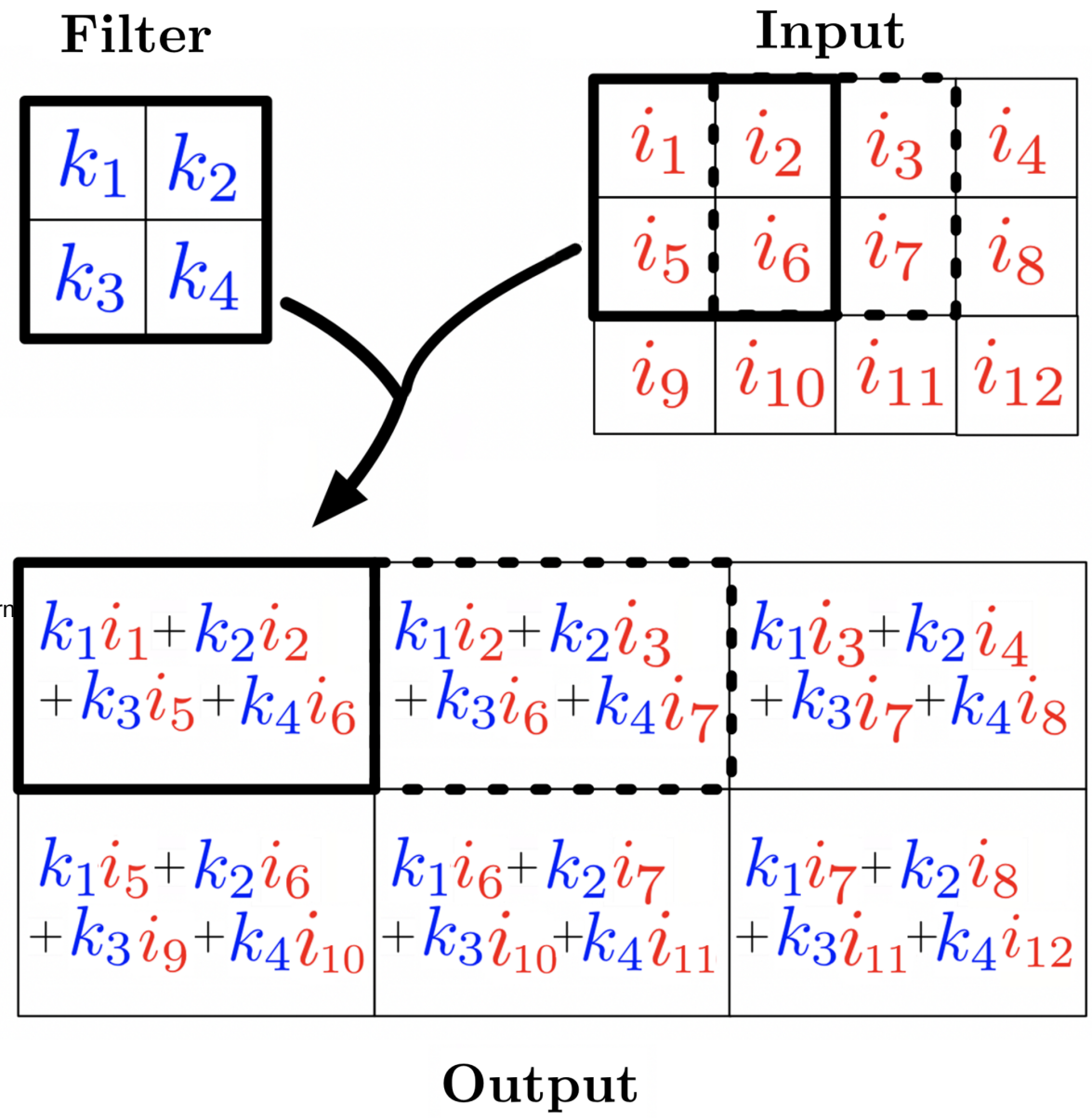


- Observation: images are not arbitrary vectors
- Goal: leverage spatial structure of images (translation equivariance)

Idea: Convolutions

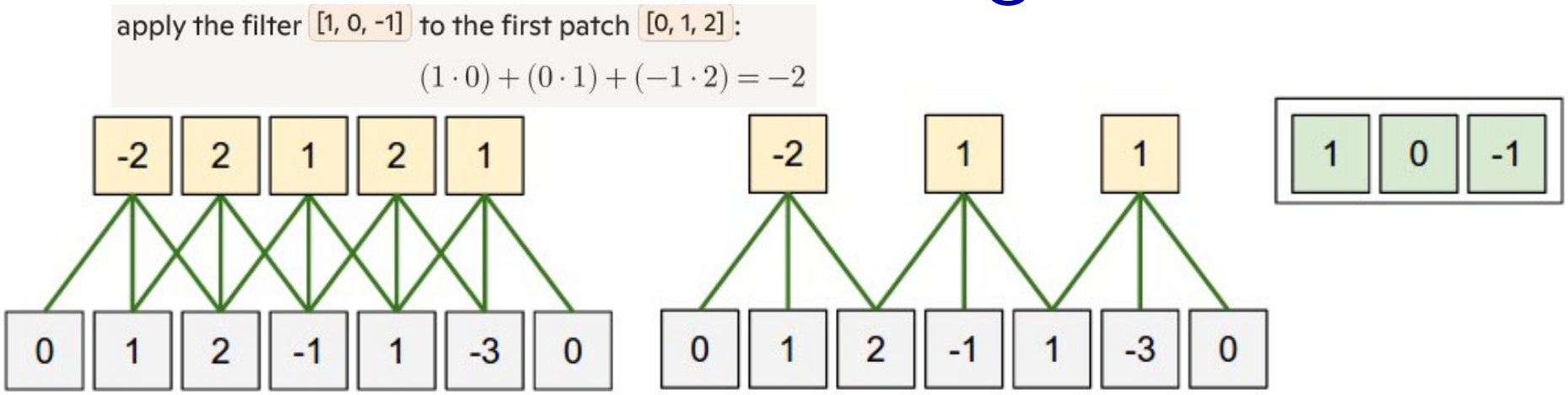
A convolution is an operation where a small filter (kernel) is slid across an input (like an image or sequence), computing dot products at each position. This produces a feature map that highlights where certain patterns occur.

-Filters act as detectors: edges, textures, shapes.-Early layers learn simple features (edges), deeper layers learn complex ones (object parts, whole objects).



[figure from Andrej Karpathy]

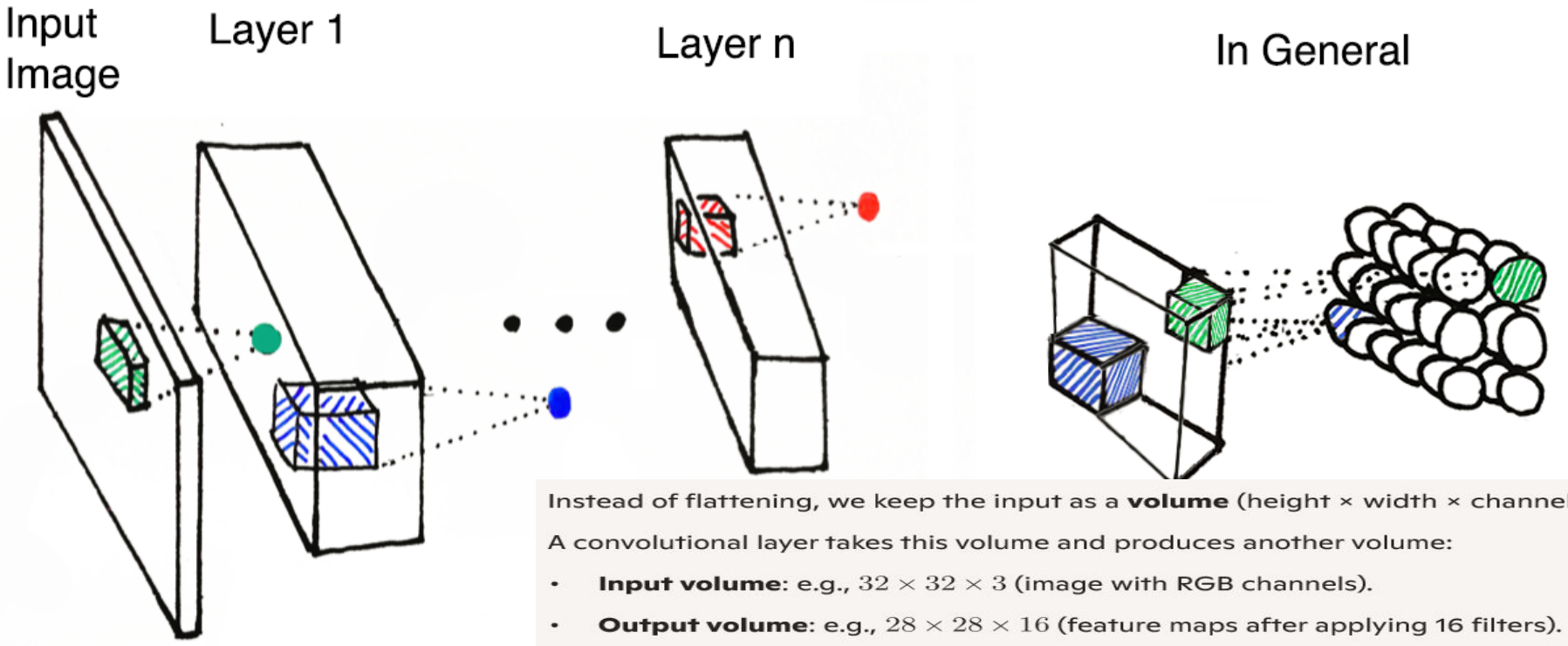
Prior knowledge



- 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 $3 \cdot 5$) (for 5 neurons each with 3 inputs, we learn just 3)
- Intuition: try to match a pattern in image

- The filter [1, 0, -1] acts like a **pattern detector** — it responds strongly when the input matches this pattern.
- This is the essence of convolution: **slide the filter** across the input and compute dot products to detect features.

Convolutional layers



Instead of flattening, we keep the input as a **volume** (height × width × channels).

A convolutional layer takes this volume and produces another volume:

- **Input volume**: e.g., $32 \times 32 \times 3$ (image with RGB channels).
- **Output volume**: e.g., $28 \times 28 \times 16$ (feature maps after applying 16 filters).

Each filter slides across the input spatially, computing dot products, and produces a **feature map**.

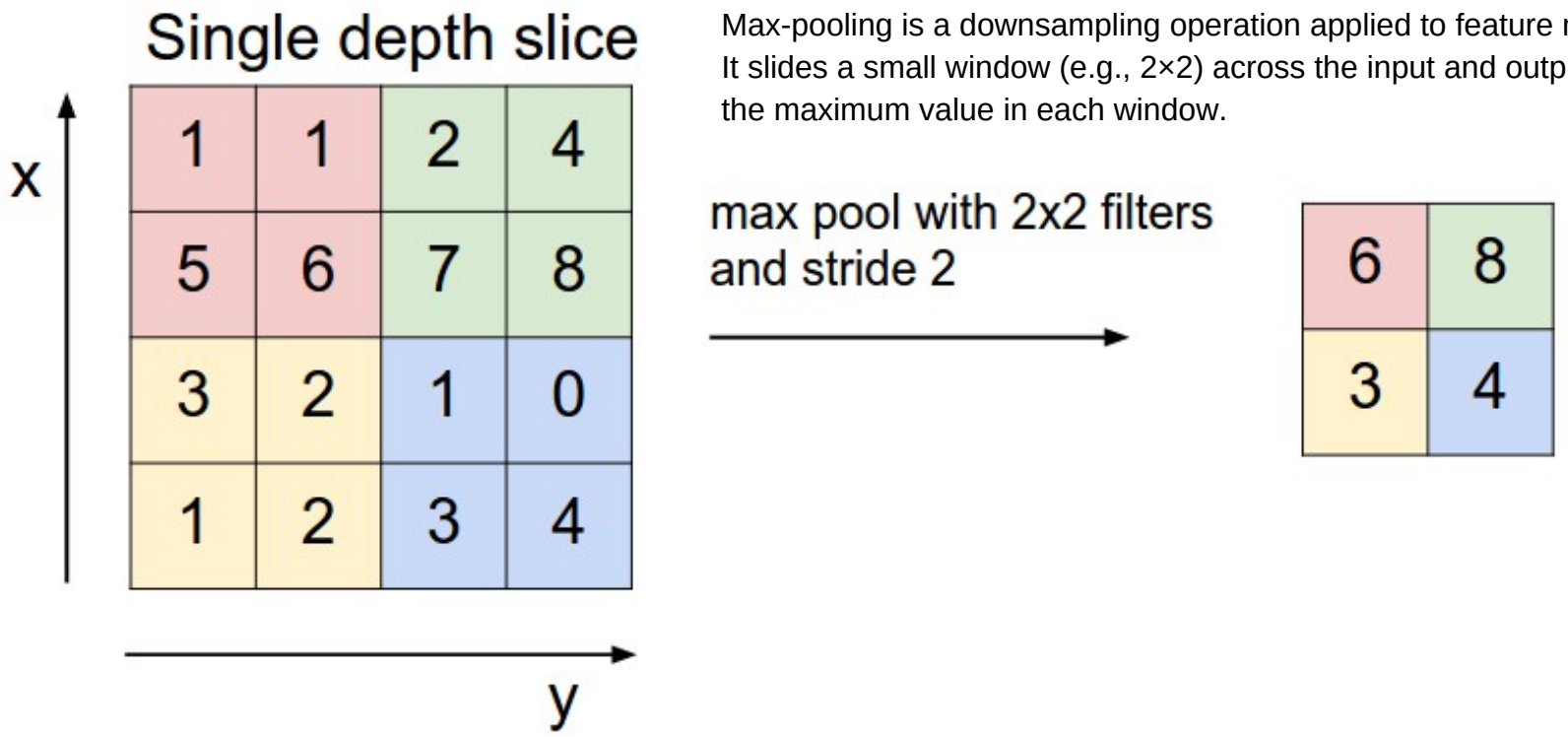
- Instead of vector to vector, we do volume to volume

[Andrej Karpathy's demo]

Convolutional layers are not "vector-to-vector" mappings but volume-to-volume transformations, preserving spatial locality and enabling deep hierarchical feature extraction.

[figure from Andrej Karpathy]

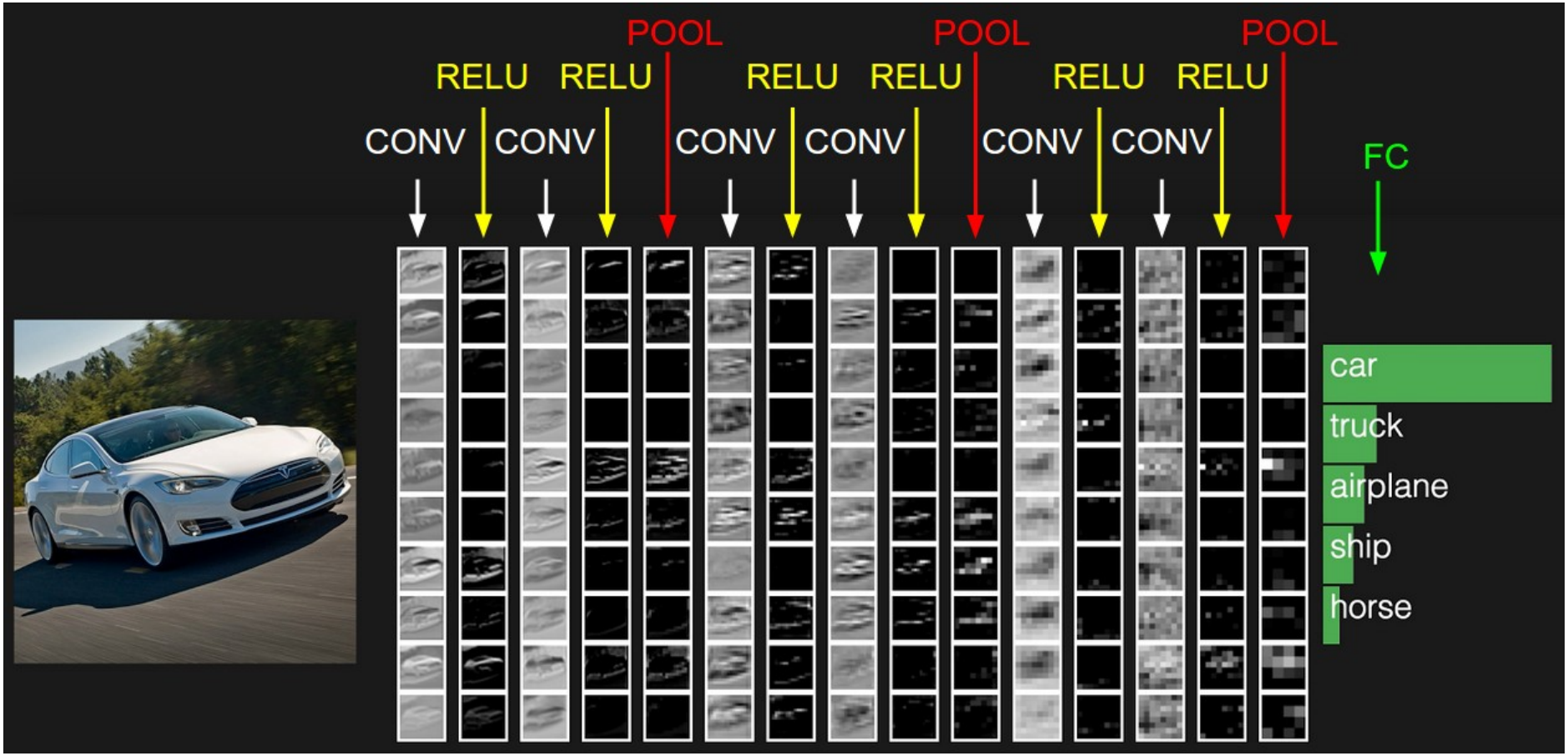
Max-pooling



Max-pooling is a downsampling operation applied to feature maps in CNNs. It slides a small window (e.g., 2×2) across the input and outputs the maximum value in each window.

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

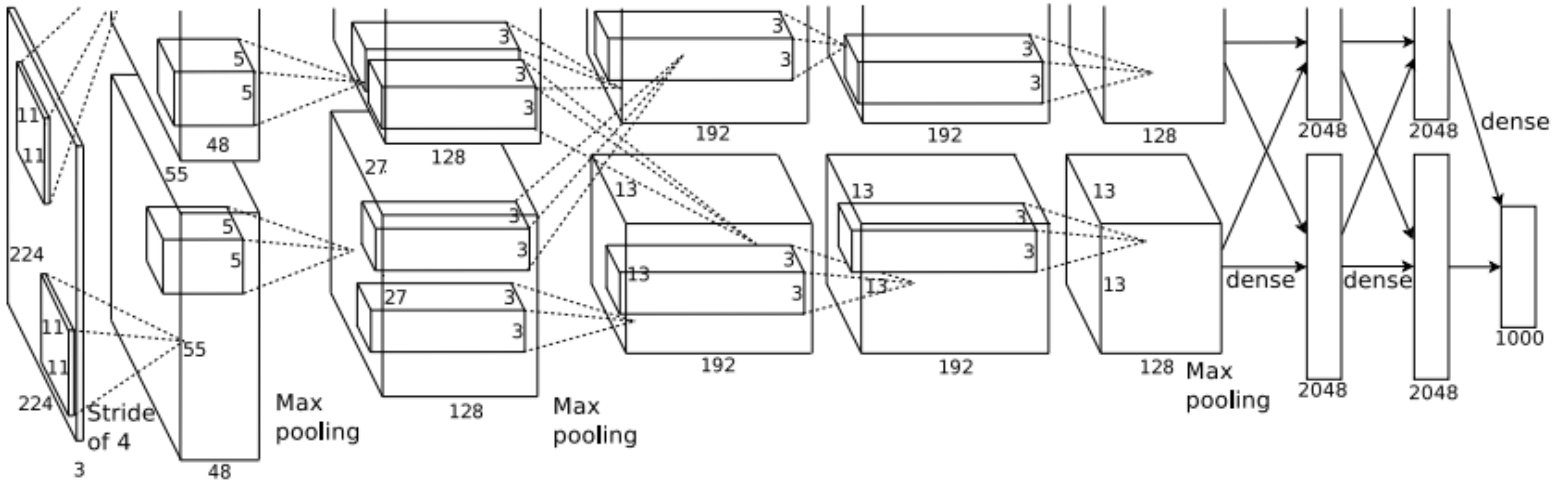
Example of function evaluation



[Andrej Karpathy's demo]

[Krizhevsky et al., 2012]

AlexNet

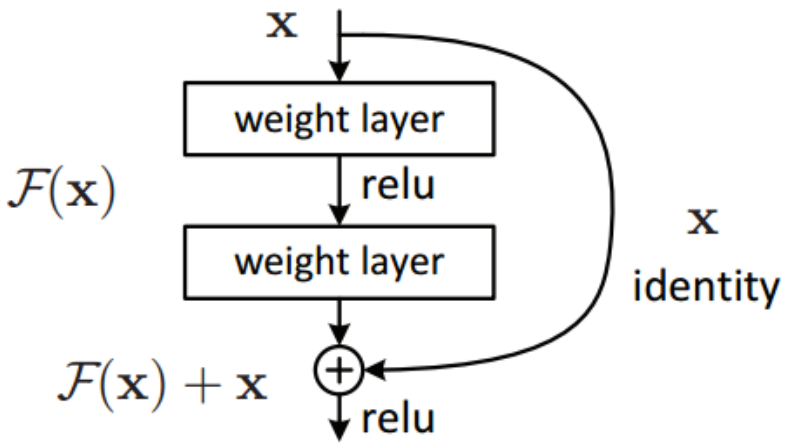


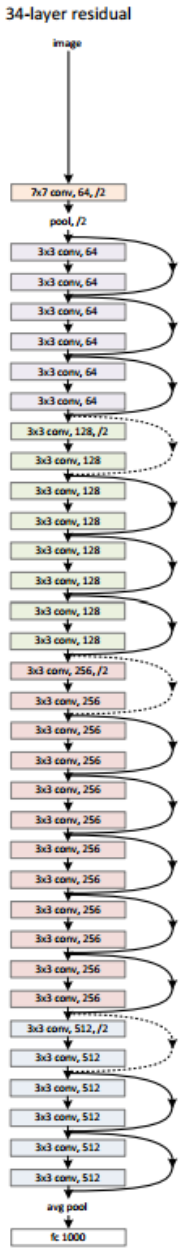
- Non-linearity: use ReIU ($\max(z, 0)$) instead of logistic
- Data augmentation: translate, horizontal reflection, vary intensity, dropout (guard against overfitting)
- Computation: parallelize across two GPUs (6 days)
- Results on ImageNet: 16.4% error (next best was 25.8%)

[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

- Residual networks introduce skip connections (or identity mappings).
- A skip connection bypasses one or more layers and adds the input directly to the output of those layers.
  Mathematically, if a block computes some transformation,
  the skip connection adds the original input: $y = F(x) + x$

- Deep networks often suffer from vanishing gradients and optimization difficulties.
- Skip connections allow gradients to flow more directly through the network, stabilizing training.
- ResNets enabled training of VERY DEEP NETWORKS (e.g., 152 layers in the original paper).
- Prior to ResNets, training networks beyond ~20–30 layers was extremely difficult.
- Think of residual blocks as saying: “If the deeper transformation isn’t useful, just keep the input as it is.”
- This flexibility makes optimization smoother and generalization stronger.

Summary

- Key idea 1: locality of connections, capture spatial structure
- Key idea 2: Filters share parameters, capture translational equivariance
- Depth matters
- Applications to images, text, Go, drug design, etc.

Roadmap

Feedforward neural networks

Convolutional networks

Sequence models

Unsupervised learning

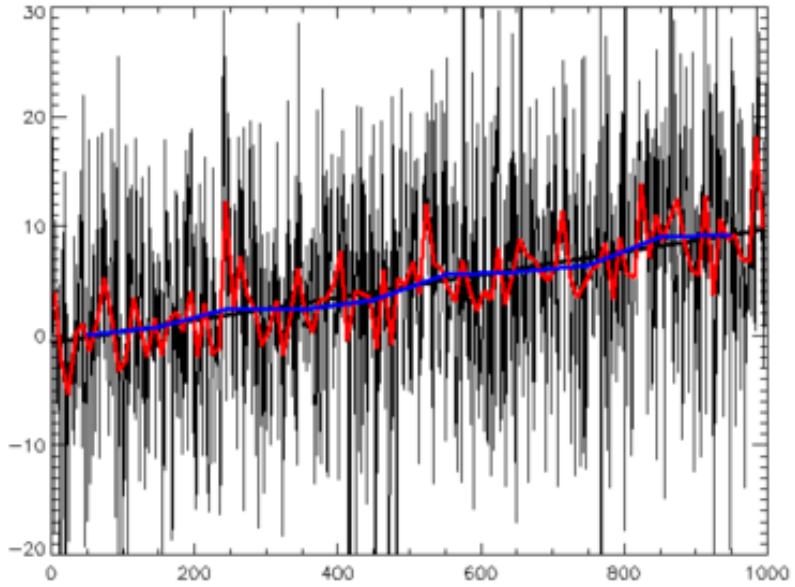
Final remarks

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:



Sequence models are designed to handle ordered data: text, speech, music, DNA, financial time series, etc.

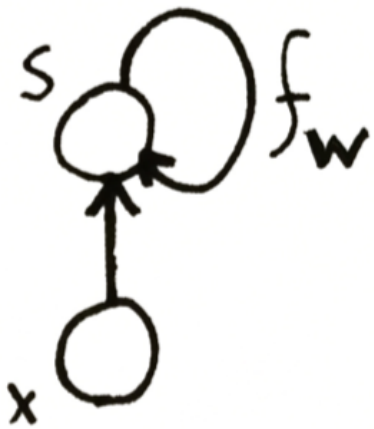
Sequence models

Sequence Models capture dependencies across time steps or positions in the sequence.

Formula

$$s_t = f_W(s_{t-1}, x_t)$$

Network

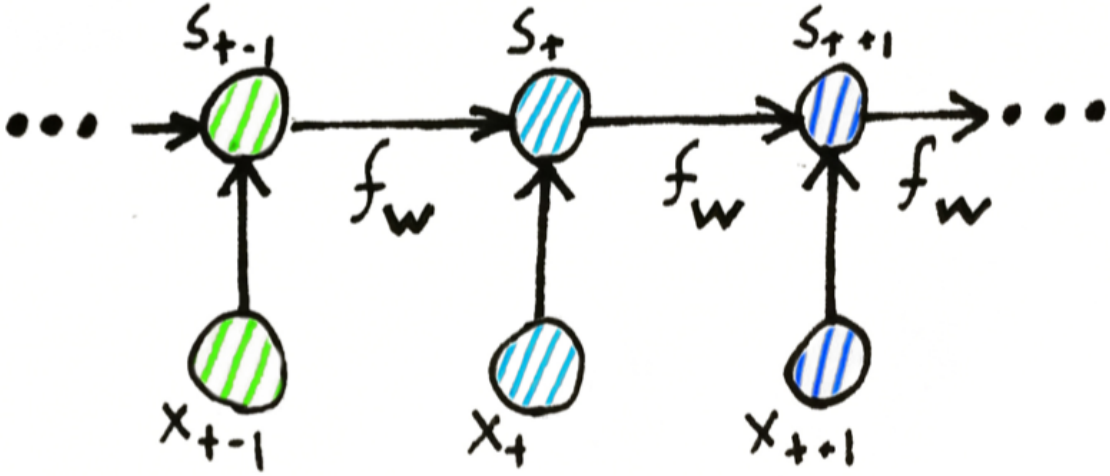


The circular nodes labeled s and x feeding into f_W represent:

- A **computational unit** that merges memory (state) and new input.
- This is where the model "decides" how to update its understanding of the sequence.

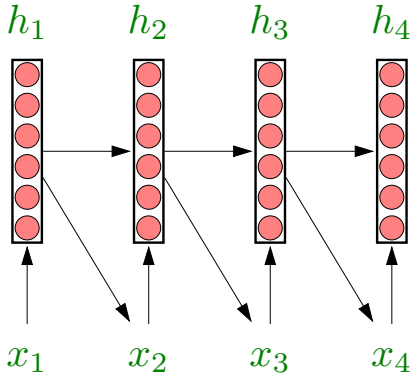
In graphical model terms, this is a **Markovian dependency**: s_t depends only on s_{t-1} and x_t, not on earlier states directly.

Computation Graph



Language models with RNNs

Recurrent Neural Networks (RNNs) are used for language modeling — predicting the next token (word, character, etc.) in a sequence.



h : hidden state at time,
a compressed summary of everything seen so far.
x : input at time (e.g., a character or word).

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

Update context vector:

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

Predict next character:

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

context h_t compresses $x_1, \ldots x_t$

This loop allows the model to generate sequences one token at a time, using its internal memory to guide predictions

[Elman, 1990]

Simple recurrent network

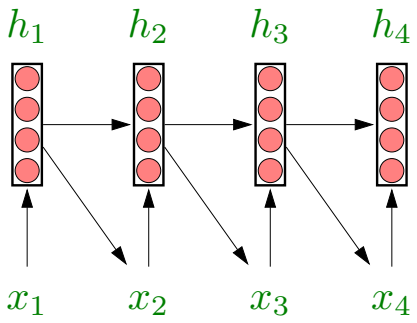
$$h_t = \sigma(Vh_{t-1} + Wx_t)$$

$$h_t = \sigma(Vh_{t-1} + Wx_t)$$

This equation updates the hidden state h_t at time t:

- h_{t-1}: previous hidden state (memory of the past)
- x_t: current input (e.g., a word or character)
- V: weight matrix for the hidden state
- W: weight matrix for the input
- σ: non-linear activation (e.g., tanh or ReLU)

- σ: non-linear activation (e.g., tanh or ReLU)



$$\text{Encode}(h_{t-1}, x_t) = \sigma(V \; h_{t-1} + W \; x_t) = h_t$$

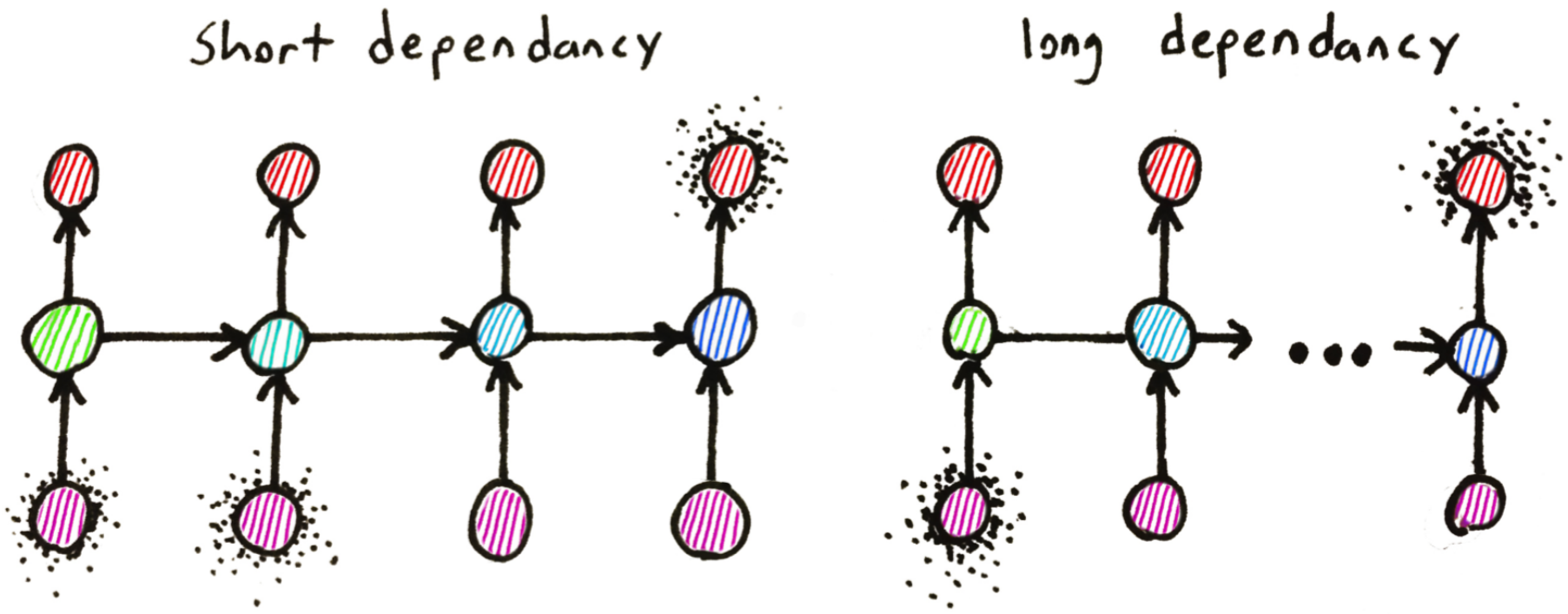
$$\text{Decode}(h_t) \sim \text{softmax}(W' h_t) = p(x_{t+1})$$

- h_t: the hidden state at time t, a vector summarizing the sequence so far.
- W': the **output weight matrix** that maps h_t to a vector of logits (unnormalized scores) over the vocabulary.
- **softmax**: converts these logits into a probability distribution over possible next tokens.

$$\text{Decode}(h_t) \sim \text{softmax}(W' \; h_t) = p(x_{t+1})$$

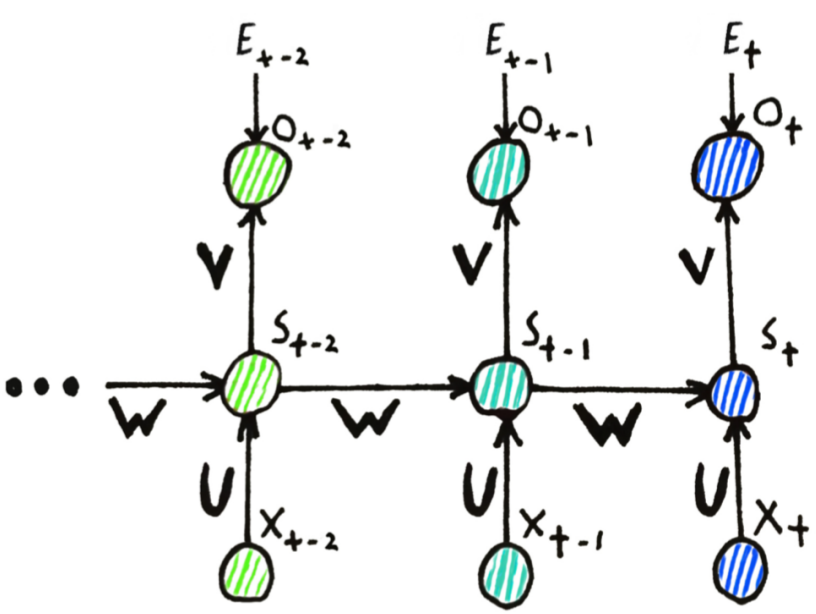
Think of h as a query and W' as a library of word embeddings:

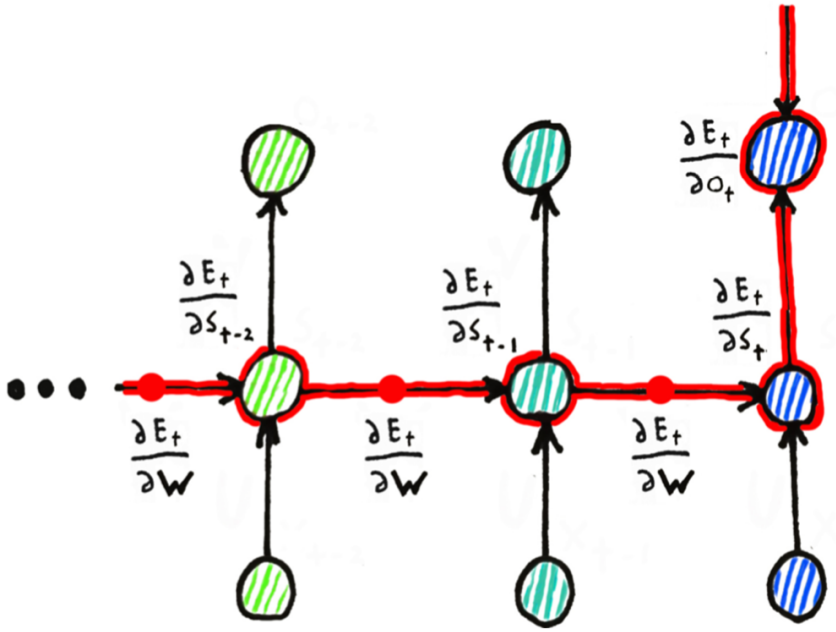
Vanishing/exploding gradient problem



- RNNs can have long or short dependencies
- When there are long dependencies, gradients have trouble back-propagating through

Vanishing/exploding gradient problem





Chain rule => multiplications

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

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:

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

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 f_t 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:

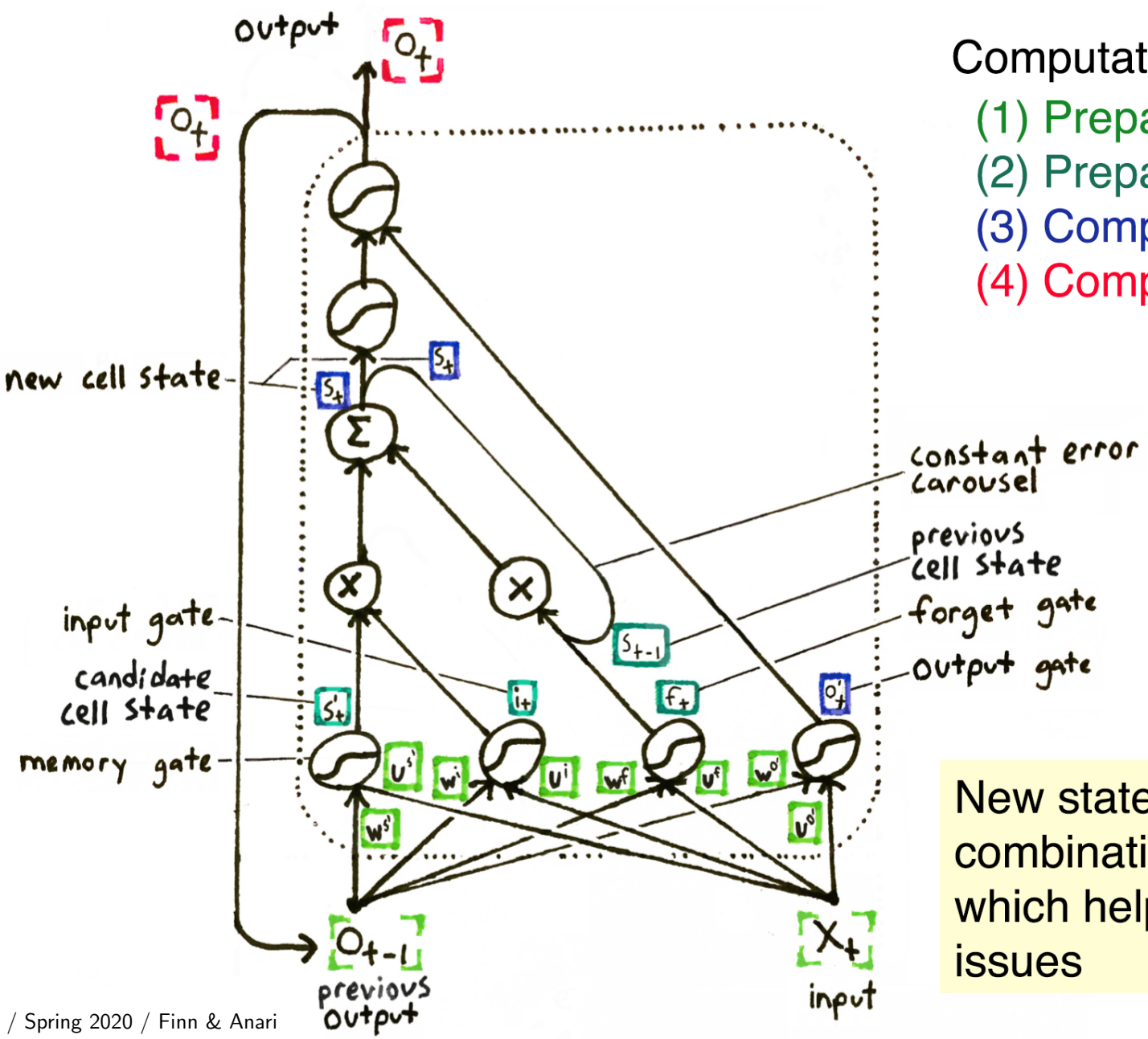
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o)$$

Hidden state:

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

[Schmidhuber & Hochreiter, 1997]

Long Short Term Memory (LSTM)



Computation:

(1) Prepare cell inputs

(2) Prepare new state inputs

(3) Compute new state

(4) Compute output

New state is additive combination with old state, which helps avoid gradient issues

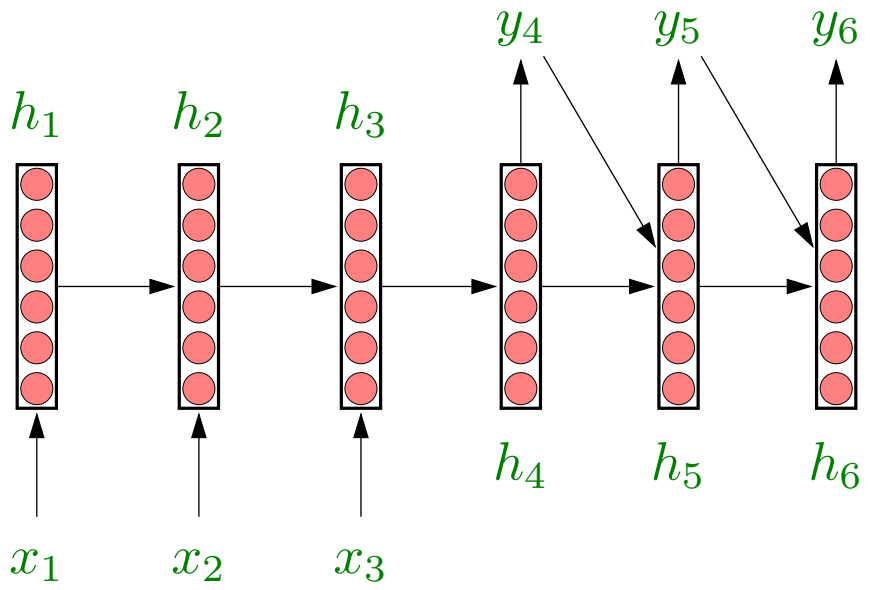
[Sutskever et al., 2014]

Sequence-to-sequence model

Motivation: machine translation

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

y: *Fear the man of one book.*



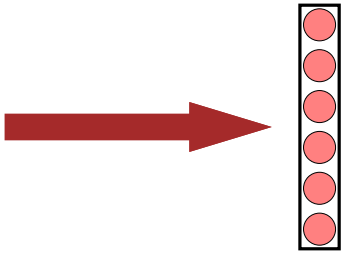
Read in a sentence first, output according to RNN:

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

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

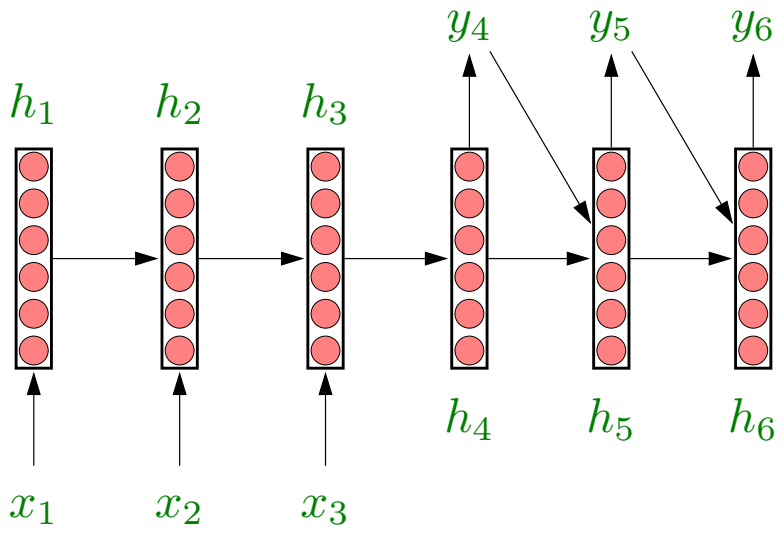
Learn to look back at your notes.

Think of attention as:

- "Learning to look back at your notes."
- Instead of memorizing everything in one shot, the model dynamically retrieves the relevant parts of the input when needed.

[Bahdanau et al., 2015]

Attention-based models



Distribution over input positions:

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

The model decides
where to look in
the input sequence
when generating the next output

Generate with attended input:

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

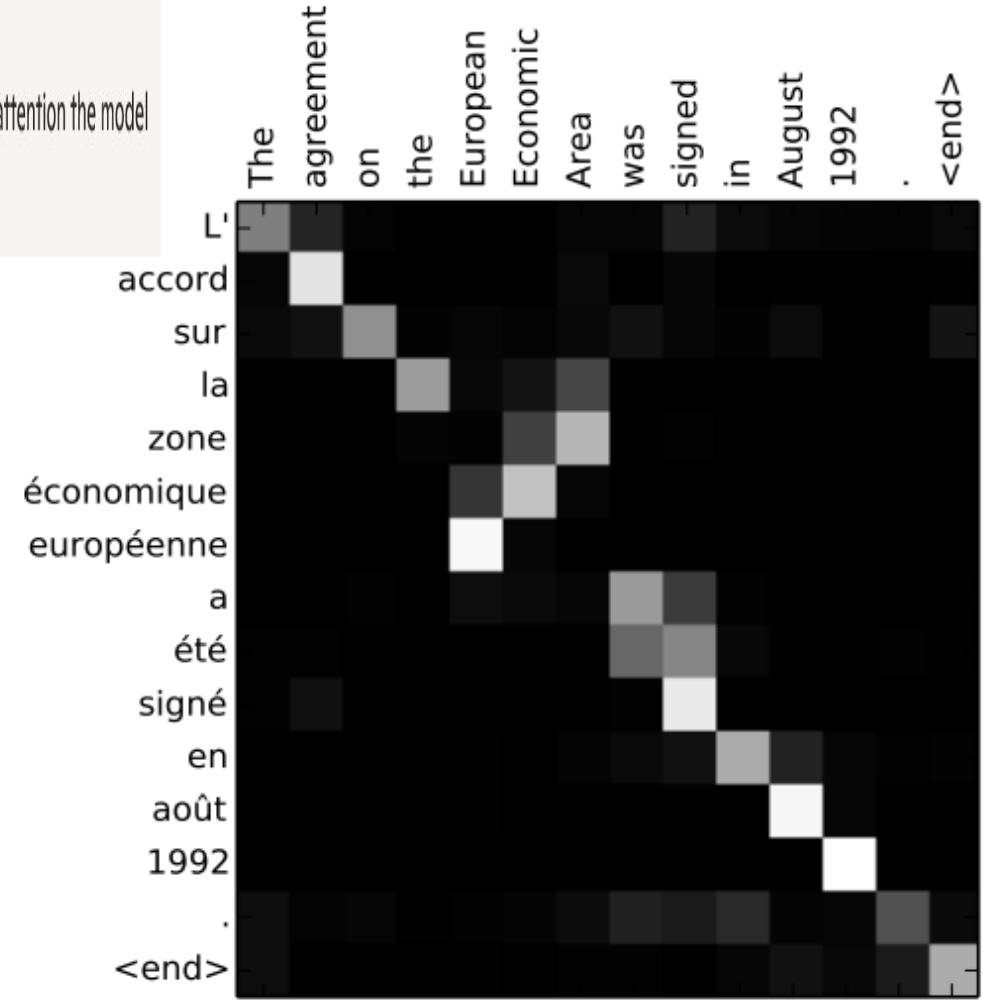
A weighted sum of input hidden states,
weighted by attention scores
This is the context vector — a dynamic summary of
the input, tailored to what the model needs at time .

Transformer models: attention only – no RNN!

[Bahdanau et al., 2015]

Machine translation

- **Rows**: French words (source sentence)
- **Columns**: English words (target sentence)
- **Cells**: Brightness indicates **alignment strength** – how much attention the model pays to a French word when generating an English word.



When generating each English word, the model:

1. Computes attention scores over all French words.
2. Uses these scores to form a **context vector** – a weighted sum of French word embeddings.
3. Combines this context with the current decoder state to predict the next English word.

This allows the model to:

- Focus on relevant parts of the source sentence.
- Handle word reordering and multi-word expressions.
- Translate phrases more fluidly than rigid word-by-word systems.

[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 networks

Sequence models

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

An **autoencoder** is a neural network trained to:

- **Compress** input data into a lower-dimensional representation (encoding).
- **Reconstruct** the original input from that representation (decoding).

It learns to capture the **essential structure** of the data.

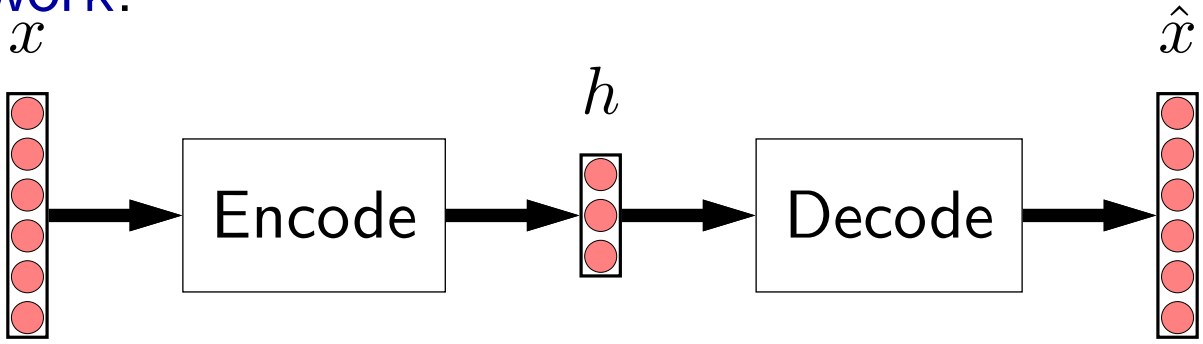
Analogy:

A A A A B B B B B ⟶ 4 A's, 5 B's ⟶ 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:

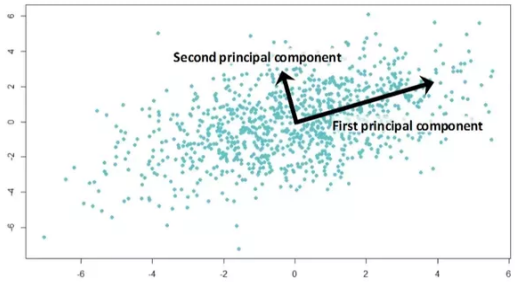


minimize $\|x - \hat{x}\|^2$

The goal is to minimize reconstruction error:
This means the network learns
to preserve as much information as possible
in the compressed representation.

Principal component analysis

Input: points $x_1, \ldots, x_n$



$$\text{Encode}(x) = \underset{U^\top}{\text{[matrix]}} \; \underset{x}{\text{[vector]}} \qquad \text{Decode}(h) = \underset{U}{\text{[matrix]}} \; \underset{h}{\text{[vector]}}$$

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

PCA objective:

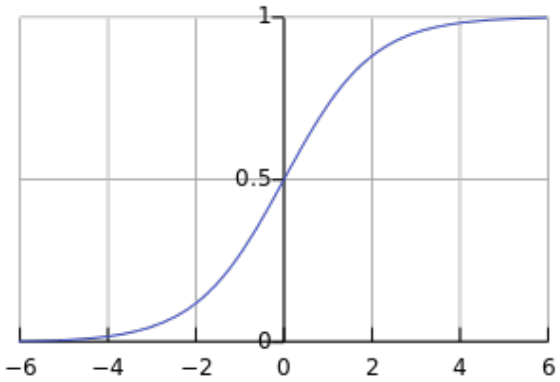
$$\text{minimize} \sum_{i=1}^{n} \|x_i - \text{Decode}(\text{Encode}(x_i))\|^2$$

Linear autoencoders are equivalent to PCA.

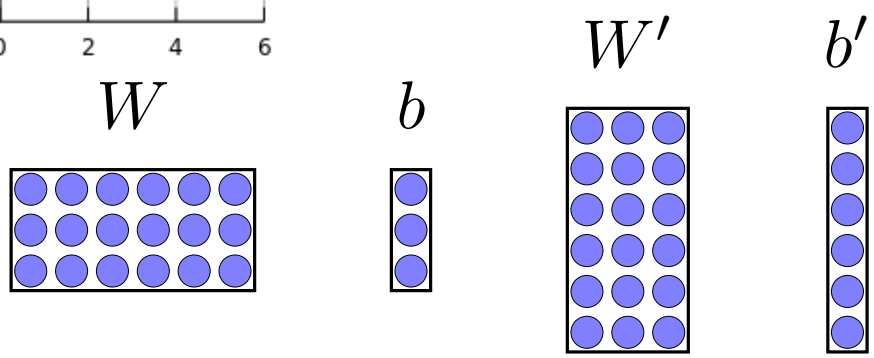
Non-linear autoencoders

Non-Linear Transformations :
Instead of using simple linear mappings, this model uses non-linear activations:
— specifically the logistic function
to increase expressiveness.

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



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

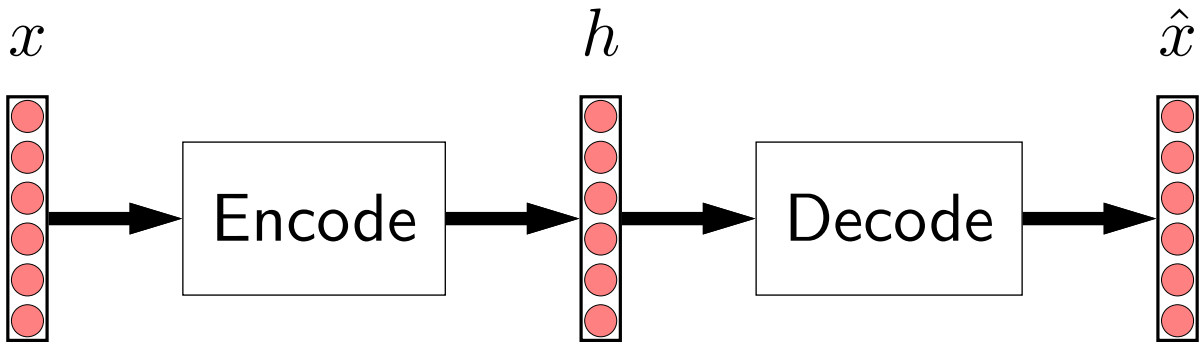


Loss function:

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

Autoencoders

Increase dimensionality of hidden dimension:

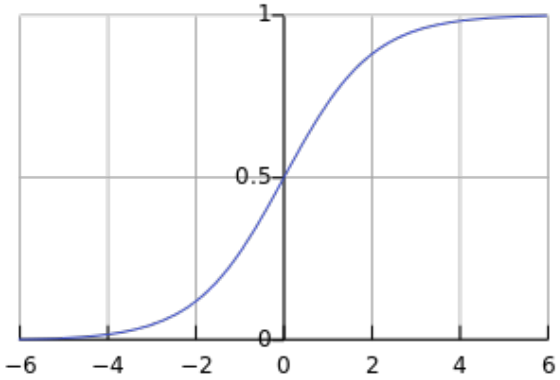


- Problem: learning nothing — just set Encode, Decode to identity function!

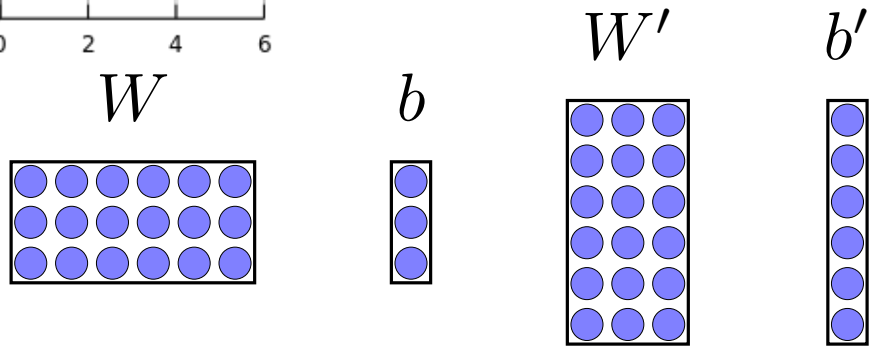
Result: zero reconstruction error, but no meaningful compression or abstraction
This defeats the purpose of representation learning — the model learns nothing useful.

Non-linear autoencoders

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



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



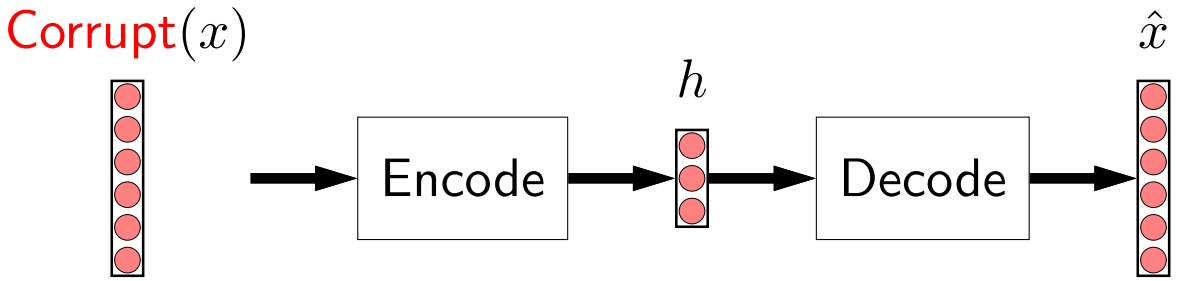
Loss function:

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

Key: Compressing h (e.g. low-dim, sparse) prevents identity

Denoising autoencoders

Corrupt input x slightly, then train to reconstruct the clean version.



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, identity function no longer good

By compressing h (low-dimensional bottleneck, sparsity, or noise), we prevent the autoencoder from collapsing into the identity function and instead force it to learn meaningful representations.

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

Denoising autoencoders

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

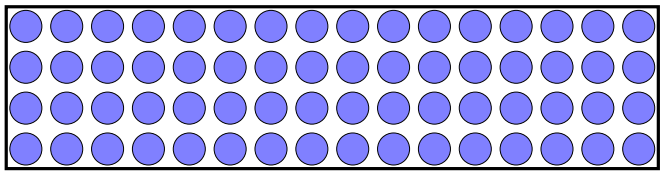
-Instead of learning to copy the input , the model learns to reconstruct the original input from a noisy version.
-This forces the network to capture essential structure rather than memorizing pixel-level details.

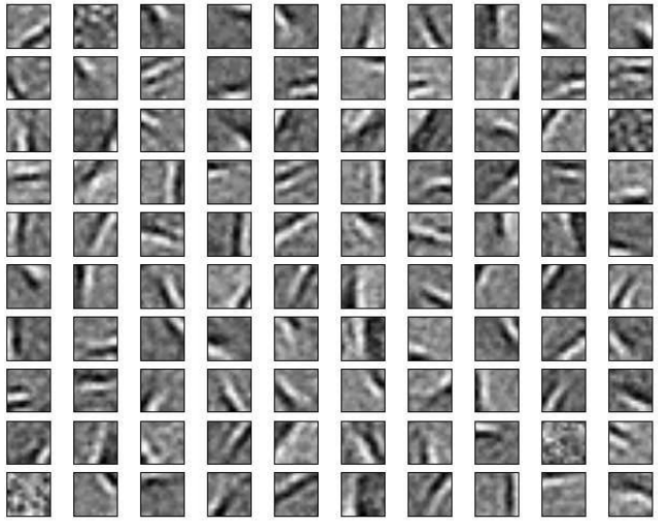
- Input: 60,000 handwritten digit images, each flattened to 784 dimensions.
- Hidden layer: 200 neurons → each row of W becomes a **learned filter**.
- Visualization: the grid of 200 grayscale patches shows what each neuron "looks for" – strokes, curves, digit fragments.

These filters resemble **edge detectors**, **stroke patterns**, and **digit prototypes** – learned automatically from data.

200 learned filters (rows of W):

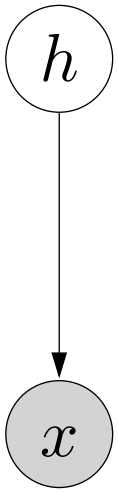
W





Variational autoencoders

Motivation: learn a latent-variable model



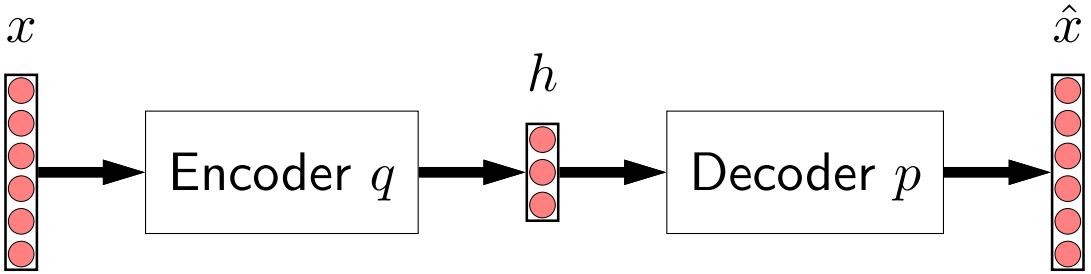
$$p(h, x) = p(h)p(x \mid h)$$

E-step in EM: computing $p(h \mid x)$ is intractable

Solution: approximate using a neural network $q(h \mid x)$

$$q(h \mid x) \approx p(h \mid x)$$

Variational autoencoders



Objective: maximize

$$\log p(x) \geq \mathbb{E}_{q(h|x)}[\log p(x \mid h)] - \mathsf{KL}(q(h \mid x)||p(h))$$

Algorithm:

- Sample h from encoder q, gradient update on q and p
- Reparametrization trick [Kingma/Welling, 2014]

Variational Autoencoders (VAEs) are widely used for generative modeling, anomaly detection, denoising, and representation learning, with applications spanning computer vision, natural language processing, healthcare, and IoT.

They are especially powerful because they learn structured latent spaces that allow smooth interpolation and sampling of new data.
A latent space is a compressed, hidden representation where the model organizes data into a more meaningful, lower-dimensional form.

[Rajpurkar+ 2016]

Reading comprehension (SQuAD)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

A large-scale dataset (100K+ examples) introduced by Rajpurkar et al. (2016).

- Each example consists of:
- A passage (typically from Wikipedia)
- A set of questions about the passage
- Answers that are spans of text directly extracted from the passage

Reading Comprehension:
The model must:

1. Read the passage.
2. Understand its meaning.
3. Locate and extract the correct answer span for each question.

This is a span-based QA task,
not multiple choice or generative.

100K examples

- **Benchmark for QA**: SQuAD became the standard dataset for machine reading comprehension.
- **Catalyst for Model Development**: Inspired architectures like BiDAF, DrQA, BERT, RoBERTa, and later Transformer-based QA systems.
- **Extensions**:
  - **SQuAD 2.0 (2018)**: Added unanswerable questions to test robustness.
  - Influenced other span-based QA datasets and open-domain QA research.

Raw text

Stanford University (officially **Leland Stanford Junior University**,[10] colloquially "the Farm") is a private research university in Stanford, California. Stanford is known for its academic strength, wealth, proximity to Silicon Valley, and ranking as one of the world's top universities.[11][12][13][14][15]

The university was founded in 1885 by Leland and Jane Stanford in memory of their only child, Leland Stanford Jr., who had died of typhoid fever at age 15 the previous year. Stanford was a U.S. Senator and former Governor of California who made his fortune as a railroad tycoon. The school admitted its first students on October 1, 1891,[2][3] as a coeducational and non-denominational institution.

Stanford University struggled financially after the death of Leland Stanford in 1893 and again after much of the campus was damaged by the 1906 San Francisco earthquake.[16] Following World War II, Provost Frederick Terman supported faculty and graduates' entrepreneurialism to build self-sufficient local industry in what would later be known as Silicon Valley.[17] The university is also one of the top fundraising institutions in the country, becoming the first school to raise more than a billion dollars in a year.[18]

The university is organized around three traditional schools consisting of 40 academic departments at the undergraduate and graduate level and four professional schools that focus on graduate programs in Law, Medicine, Education and Business. Stanford's undergraduate program is one of the top three most selective in the United States by acceptance rate.[19][20][21][22][23] Students compete in 36 varsity sports, and the university is one of two private institutions in the Division I FBS Pac-12 Conference. It has gained 117 NCAA team championships,[24] the most for a university. Stanford athletes have won 512 individual championships,[25] and Stanford has won the NACDA Directors' Cup for 23 consecutive years, beginning in 1994–1995.[26] In addition, Stanford students and alumni have won 270 Olympic medals including 139 gold medals.[27]

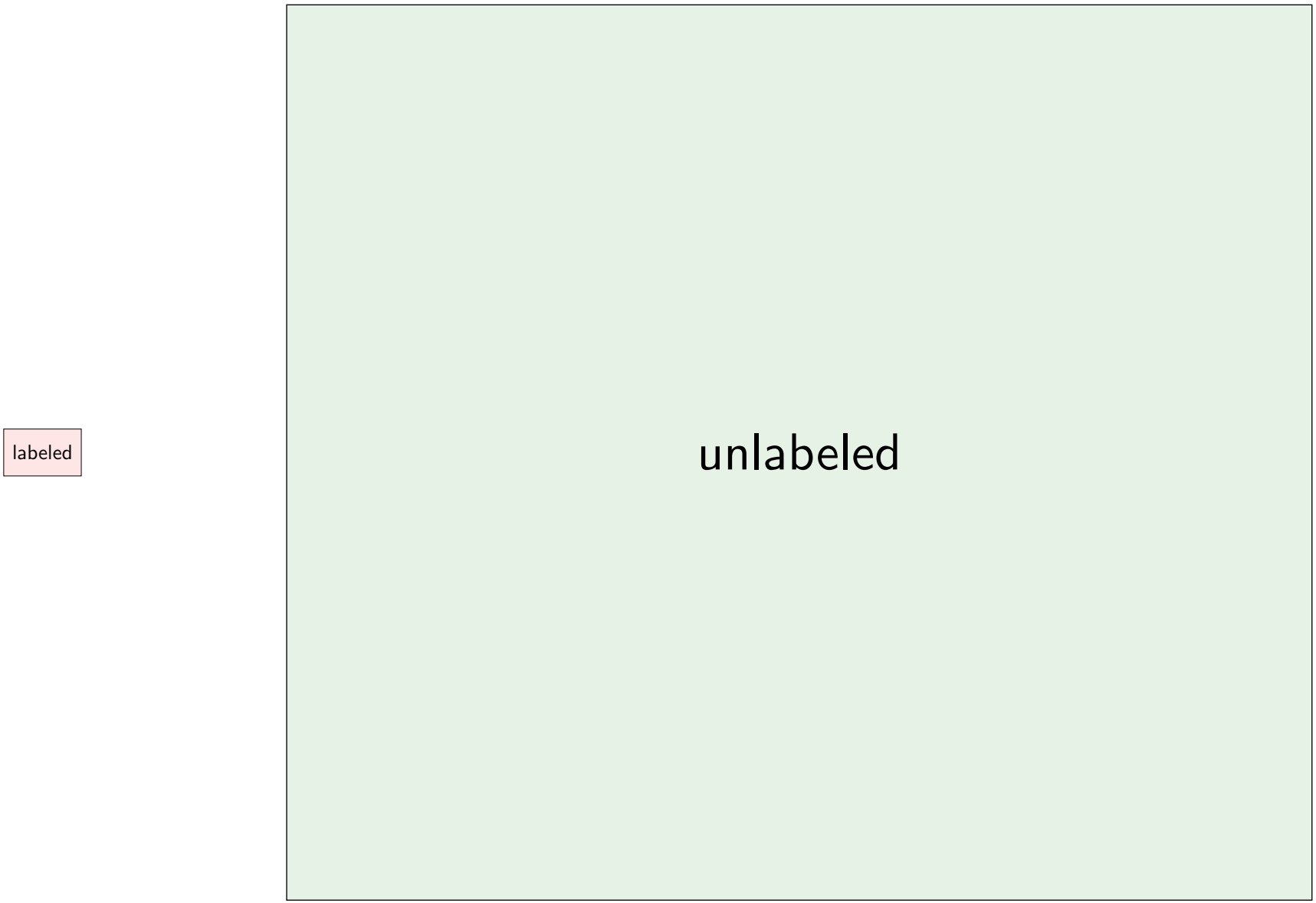
...

3.3 billion words

The raw text input used in large-scale language modeling: Example: — specifically showing a Wikipedia-style article about Stanford University.
There are about 3.3 billion words in Wikipedia.
This is "Raw text", meaning it's unprocessed — no annotations, no labels, just plain natural language.

Language models are trained to predict words or fill in blanks based on massive corpora like this.

Unsupervised pre-training



[Devlin+ 2018]

BERT

- A Transformer-based model trained to deeply understand language by looking both left and right of a word — hence bidirectional.
- Unlike traditional models that read text left-to-right (like GPT) or right-to-left, BERT reads in both directions simultaneously, capturing richer context.
- Masked Language Modeling (MLM): Randomly mask words in a sentence and train the model to predict them.
- Next Sentence Prediction (NSP) • Given two sentences, predict whether the second logically follows the first. • Helps BERT learn relationships between sentences — useful for tasks like QA and summarization.

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

Why BERT important:

- Achieved state-of-the-art on 11 NLP tasks upon release.
- Became the foundation for models like RoBERTa, ALBERT, DistilBERT, and many multilingual variants.
- Powers applications in: Question answering (e.g., SQuAD), Sentiment analysis, Named entity recognition, Text classification

BERT

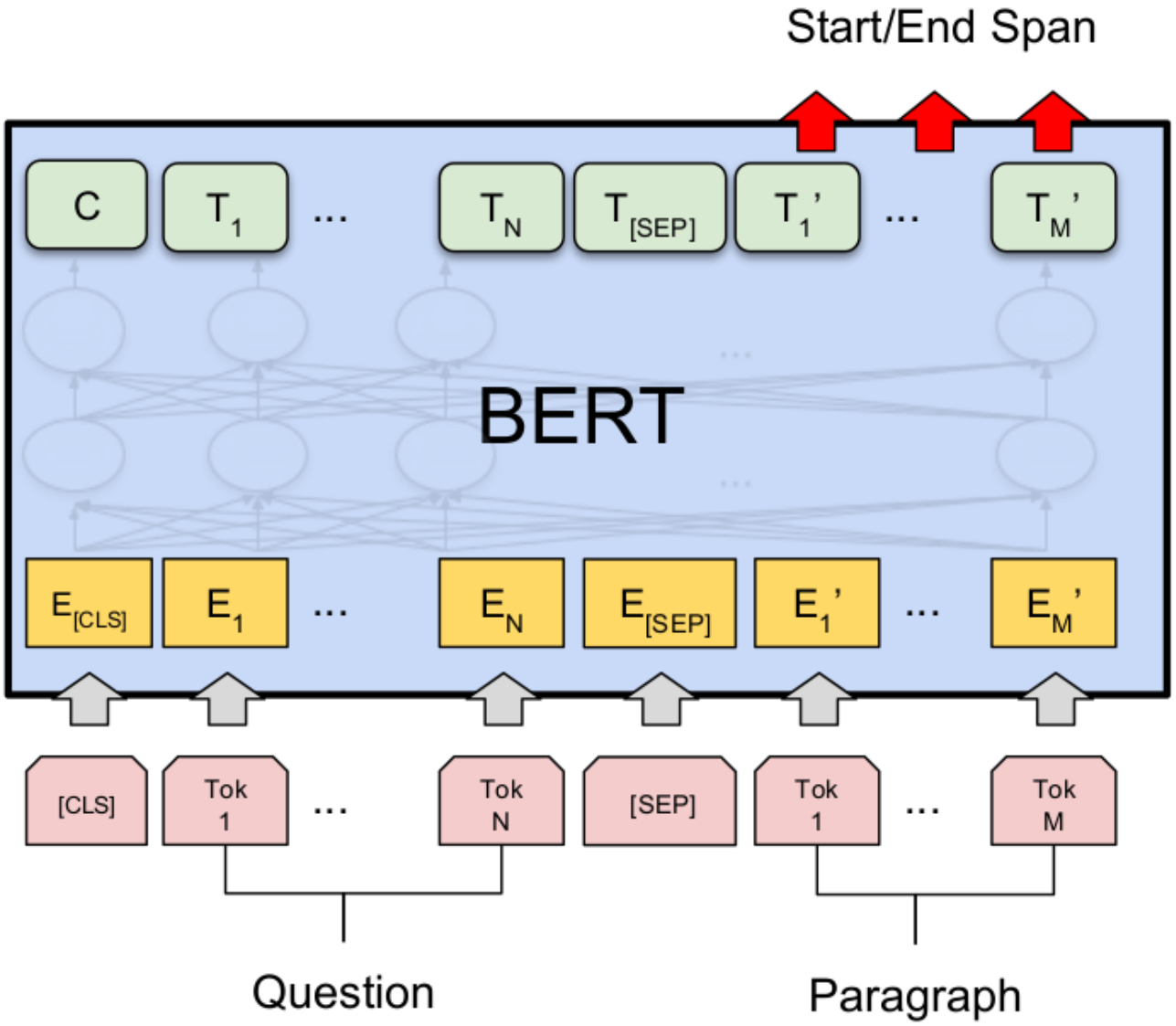
how is BERT used for question answering (QA)?

Given:

- A question (e.g., “Where was Stanford founded?”)
- A paragraph (e.g., a Wikipedia article about Stanford)

BERT must predict:

- The start and end positions of the answer within the paragraph.



-This leaderboard showcases the performance of various models on a machine reading comprehension benchmark.

-What's Being Measured?Two key metrics:• EM (Exact Match): Percentage of predictions that exactly match the ground truth answer.• F1 Score: Measures overlap between predicted and true answers (precision + recall).

-BERT (ensemble) outperforms human baseline on both EM and F1 — a major milestone in NLP.- Ensemble models combine multiple BERTs for stronger predictions.- The leaderboard reflects rapid progress in deep learning for language understanding, especially post-Transformer era.

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 networks

Sequence models

Unsupervised learning

Final remarks

Getting things to work

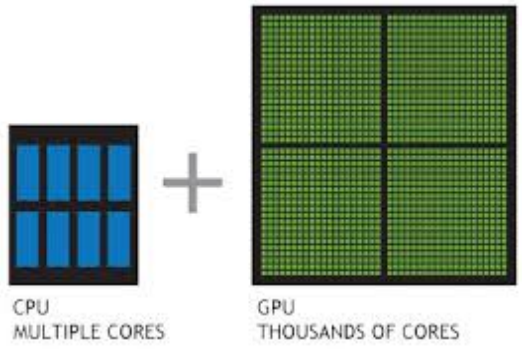
Better optimization algorithms: SGD, SGD+momentum, AdaGrad, AdaDelta, momentum, Nesterov, Adam

Tricks: initialization, gradient clipping, batch normalization, dropout

More hyperparameter tuning: step sizes, architectures

More data: larger, broader datasets

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

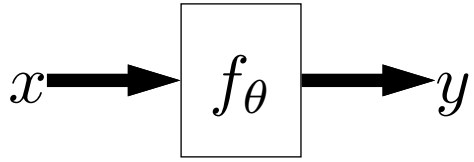
Phenomena	Ideas
Fixed vectors	Feedforward NNs
Spatial structure	convolutional NNs
Sequence	recurrent NNs LSTMs
Sequence-to-sequence	encoder-decoder attention-based models
Unsupervised	autoencoders variational autoencoders any auxiliary task

Outlook

Extensibility: able to compose modules

(You can combine these modules flexibly — like building blocks — to create more powerful systems.)

Learning programs: think about analogy with a computer



- Like a traditional computer program, a neural network:
  - Receives input
  - Executes a sequence of operations
  - Produces output
- But instead of hard-coded logic, it learns the operations via gradient descent.
- f is a function mapping input x to output y, parameterized by θ