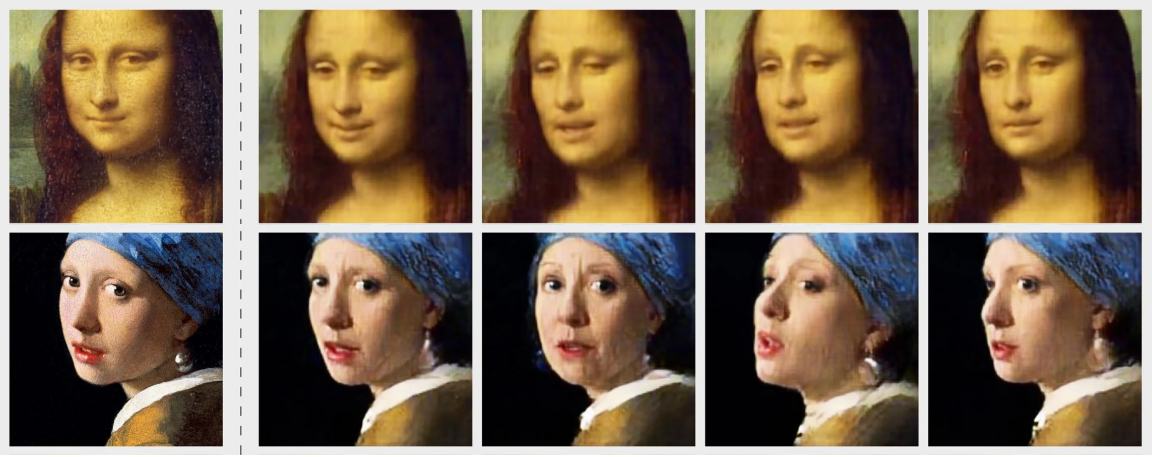
ICME Summer Workshops 2020

Introduction to Deep Learning

Session 1: 9:00—10:30 AM
Instructor: Sherrie Wang

icme-workshops.github.io/deep-learning



Zakharov *et al.* "Few-Shot Adversarial Learning of Realistic Neural Talking Head Models" (2019)

Workshop Schedule

Session 1 (9:00—10:30 AM)

- Introduction
- Current state-of-the-art in deep learning
- Math review
- Fully connected neural networks

Session 2 (10:45—12:00 PM)

- Loss functions
- Gradient descent
- Backpropagation
- Overfitting and underfitting

Lunch (12:00—2:00 PM)

Session 3 (2:00—3:15 PM)

- Convolutional neural networks
- Recurrent neural networks
- Other architectures
- Deep learning libraries
- Hands-on coding session in Tensorflow

Session 4 (3:30—4:45 PM)

- Hands-on coding session in Keras
- Hands-on coding session on transfer learning
- Failures of deep learning

About me

Hello
my name is

Sherrie Wang

@sherwang

- **PhD student** in ICME, Year 5
- **Research focus:** Machine learning methods for remote sensing and applications in sustainability
- **Relevant courses:**
  - CS 221 (Artificial Intelligence)
  - CS 229 (Machine Learning)
  - CS 228 (Probabilistic Graphical Models)
  - CS 230 (Deep Learning)
  - CS 231n (Convolutional Neural Networks)
  - CS 236 (Generative Adversarial Networks)
  - CS 330 (Deep Multi-Task and Meta Learning)
- **Relevant teaching:** CME 250 (Introduction to Machine Learning), Summer Workshop 2019

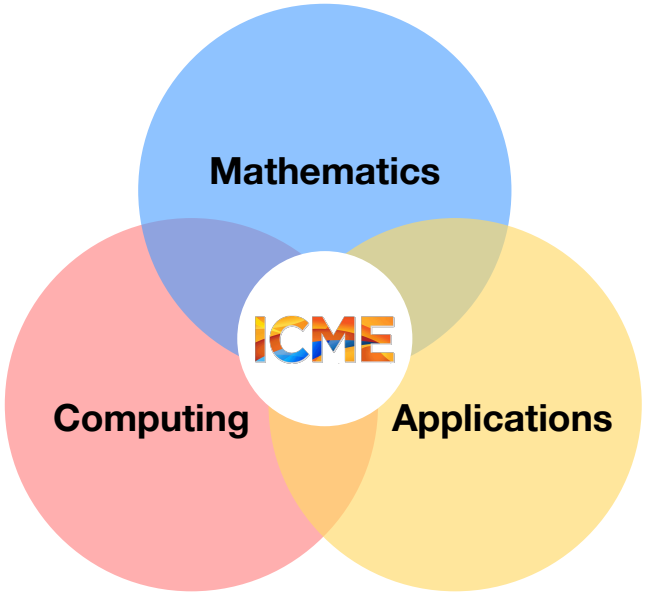
What is ICME?

Stanford Institute for Computational and Mathematical Engineering

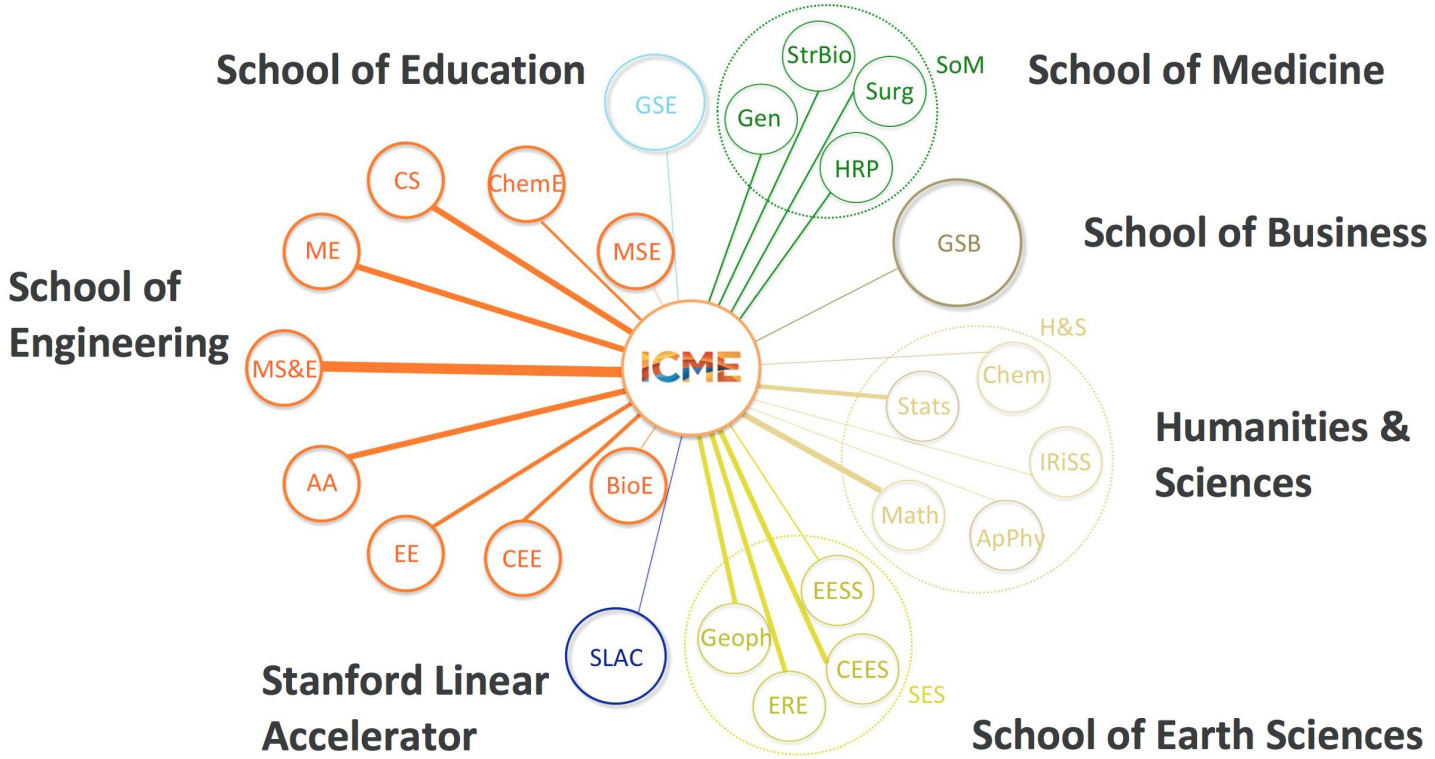
Hub at Stanford

Interdisciplinary field using advanced mathematics and computing to address complex problems

icme.stanford.edu



55+ affiliated faculty across Stanford



150+ MS and PhD students

ICME industry-academia ecosystem

WELLS FARGO

TOTAL

intuit

Microsoft

NVIDIA

Schlumberger

Tencent 腾讯

Google

MathWorks

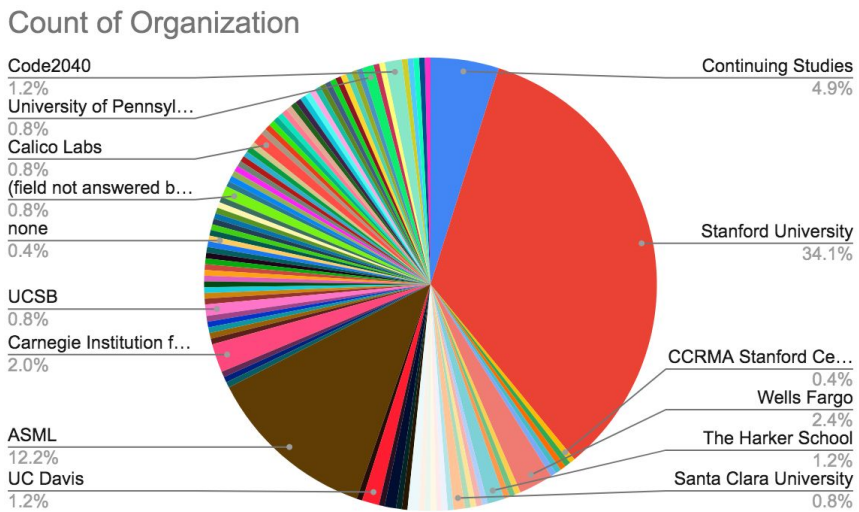
@WalmartLabs

Accenture Applied Intelligence

Who's here today

Go to **PollEv.com/dlworkshop2020**

250+ registered participants



Job title:

- 34% student
- 17% in research
- 5% continuing studies
- 4% software engineer
- 4% professor

Academia / industry:

- 66% academia
- 34% industry

The Deep Learning Revolution

What is “deep learning”?

Artificial intelligence

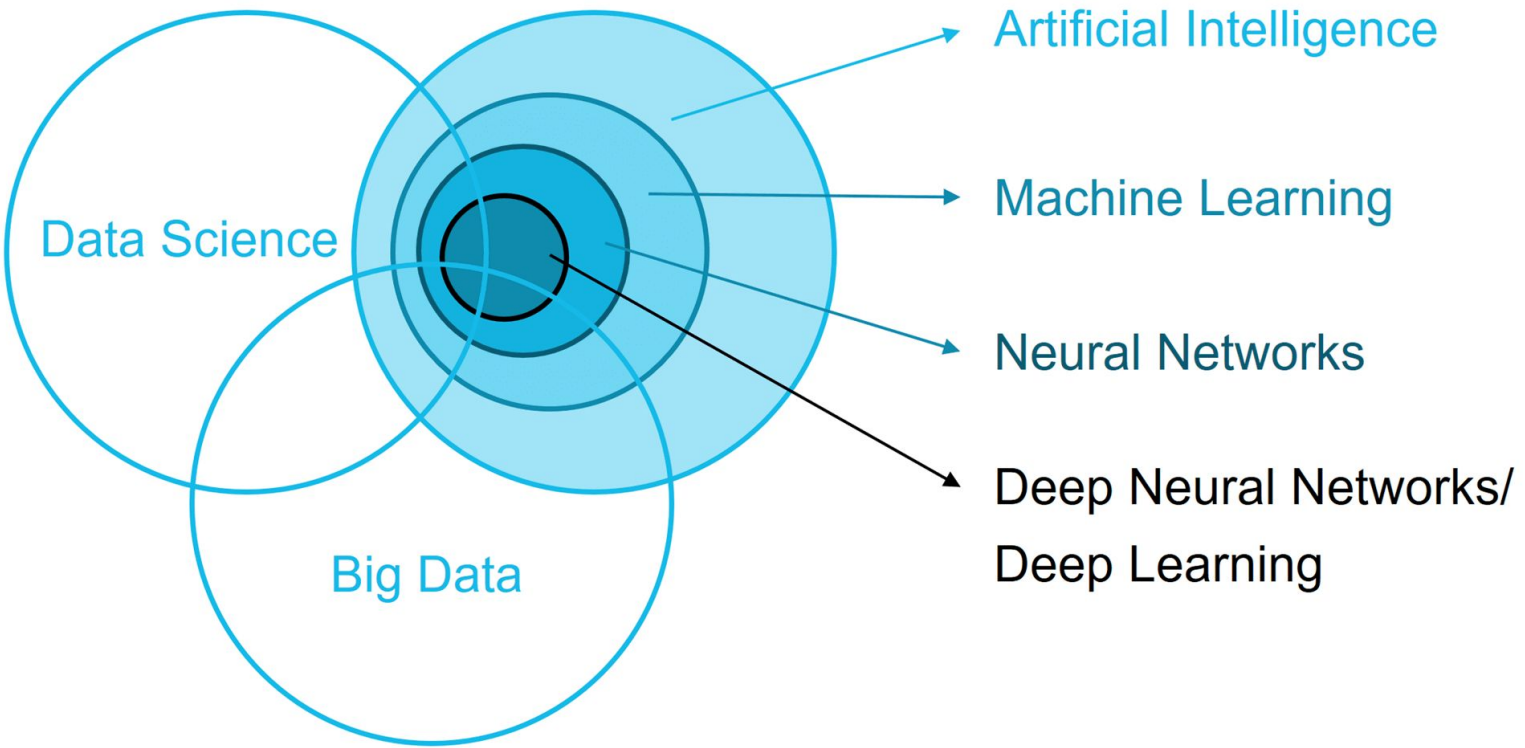
Big data

Machine learning

Deep learning

Data science

What is “deep learning”?



https://tamuk.quickstart.com/blog/artificial-intelligence-vs.-machine-learning-vs.-data-science/

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

1. Optical character recognition

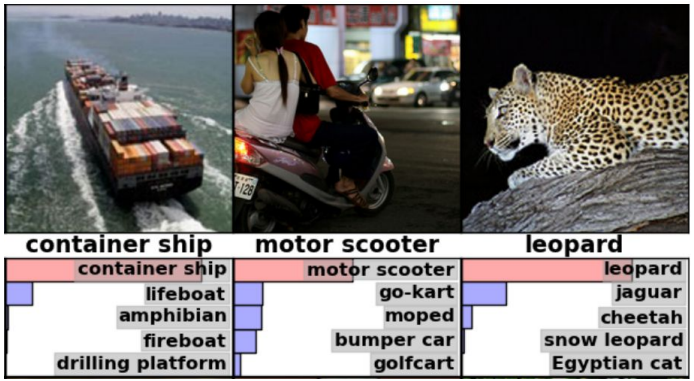
Computer vision is one of the first places where deep learning started to outshine other methods.

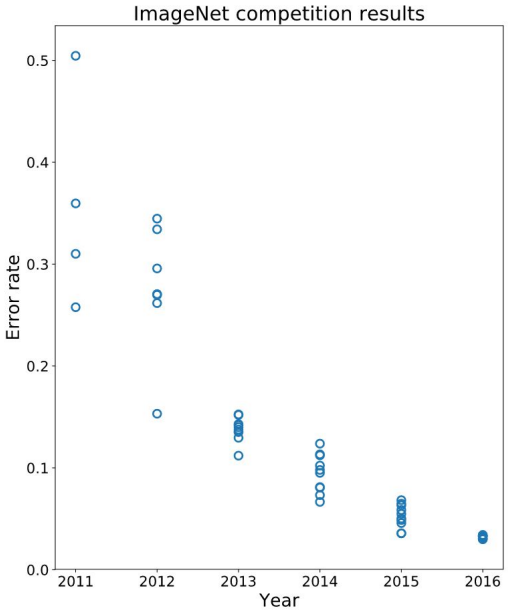
"The Street View House Numbers (SVHN) Dataset": http://ufldl.stanford.edu/housenumbers/

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

2. Image classification





Krizhevsky, Sutskever, and Hinton. "ImageNet Classification with Deep Convolutional Neural Networks" (2012)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

3. Text translation

Wu *et al.* "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation" (2016)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

4. Image generation

Progression in the capabilities of GANs

Brundage *et al*. "The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation" (2018)

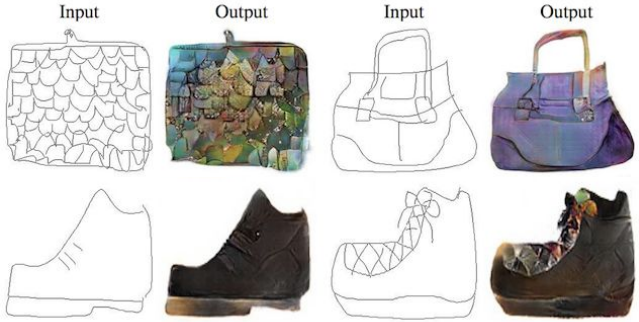
Brock *et al*. "Large Scale GAN Training for High Fidelity Natural Image Synthesis" (2018)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

4. Image generation

Image-to-image translation



Isola *et al.* "Image-to-Image Translation with Conditional Adversarial Networks" (2016)

Text-to-image translation

Reed *et al.* "Generative Adversarial Text to Image Synthesis" (2016)

Source: Instagram, FaceApp - AI Face Editor

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

5. Text generation

GPT-3-aided poetry!

Human prompt (bolded):

Here's a poem by Dr. Seuss about Elon Musk launching rockets with SpaceX, building electric cars at Tesla, selling flame throwers, dating Grimes, fighting the SEC, crushing short sellers, building brain chips to fight AI, and colonizing mars, and tweeting at all hours of day and night. Musk wants to make humans interplanetary to prevent extinction. The poem rhymes every other line with an ABAB structure. The rhyme structure is typical of Dr. Seuss nursery rhymes.

GPT-3 output (with "several hours" of trial and error):

Elon Musk
By Dr. Seuss

Once there was a man
who really was a Musk.
He liked to build robots
and rocket ships and such.

He said, "I'm building a car
that's electric and cool.
I'll bet it outsells those
Gasoline-burning clunkers soon!"

They sell cool cars
that are electric and clean.
And then he wanted to go
even farther, you see.

The first Tesla, Roadster,
Was just the start of a bigger dream.
He said, "I'm building a plane
that's electric and green.

That's not all!
That man is also a Musk.
He likes electric cars,
flamethrowers, tunnels, and such.

But then he thought, "If we make
a very small, small chip,
We'll implant it in our heads,
And then connect our minds to the Internet!"
...

Source: https://arr.am/2020/07/14/elon-musk-by-dr-seuss-gpt-3/ (2020)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

5. Text generation

GPT-3-generated text:

Human prompt is bolded.

Movie plot summaries (from Wikipedia):

The Matrix (1999): "A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers." (For the purposes of this post, we can set aside some of the specific details and just say that, basically, Neo learns that our world isn't real—that what we perceive isn't the real thing, but a simulation or "matrix" created to fool us.)

Inception (2010): "While competing with a rival thief to recover a potent artefact, Dom Cobb (Leonardo DiCaprio) falls into a dreamworld where he must 'extract' an idea from an influential businessman while being chased by the FBI."

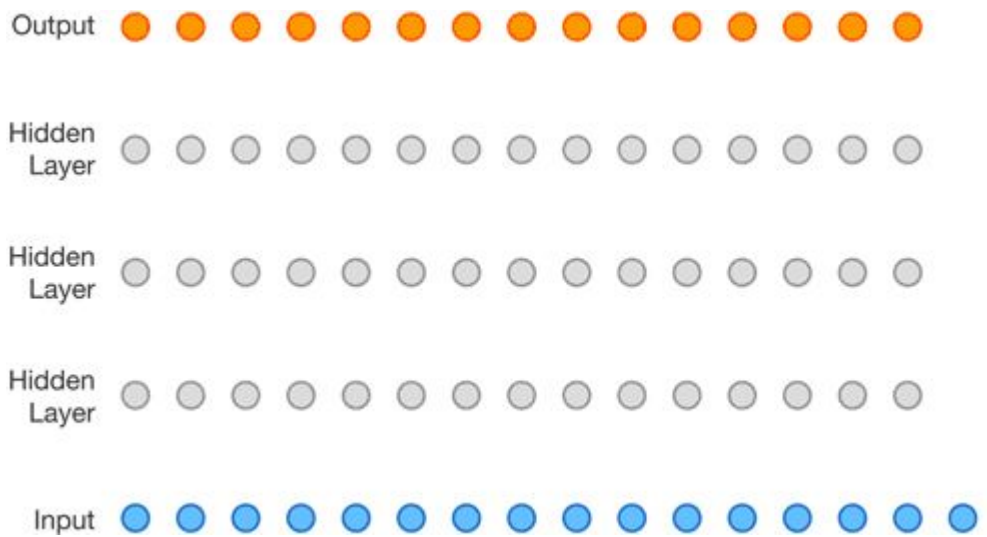
Source: https://www.gwern.net/GPT-3 (2020)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

6. Audio generation

DeepMind's WaveNet



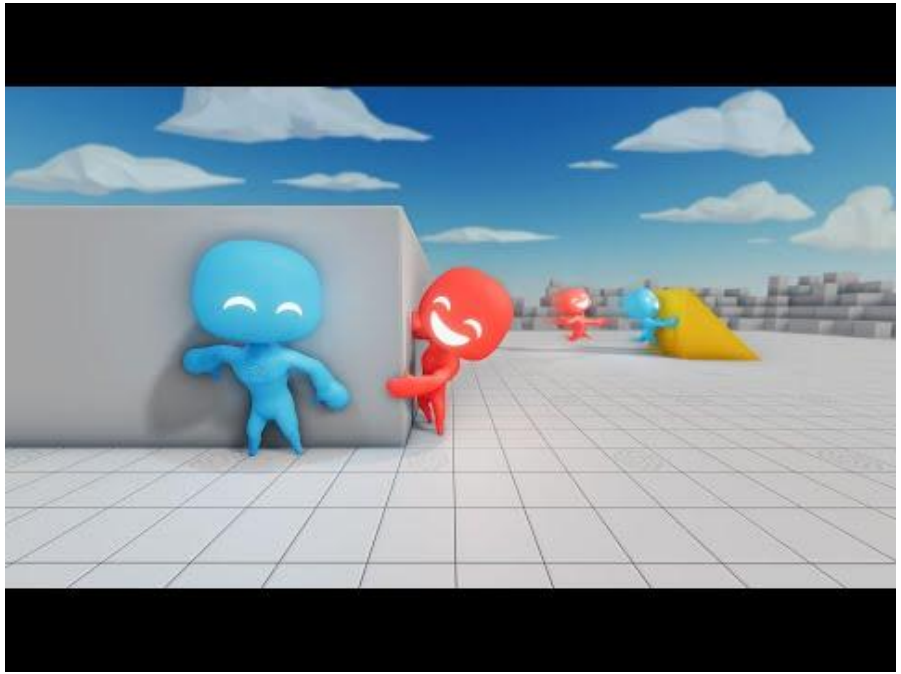
Source: https://deepmind.com/blog/article/wavenet-generative-model-raw-audio (2016)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

7. Reinforcement learning

OpenAI's "Hide and Seek" Game



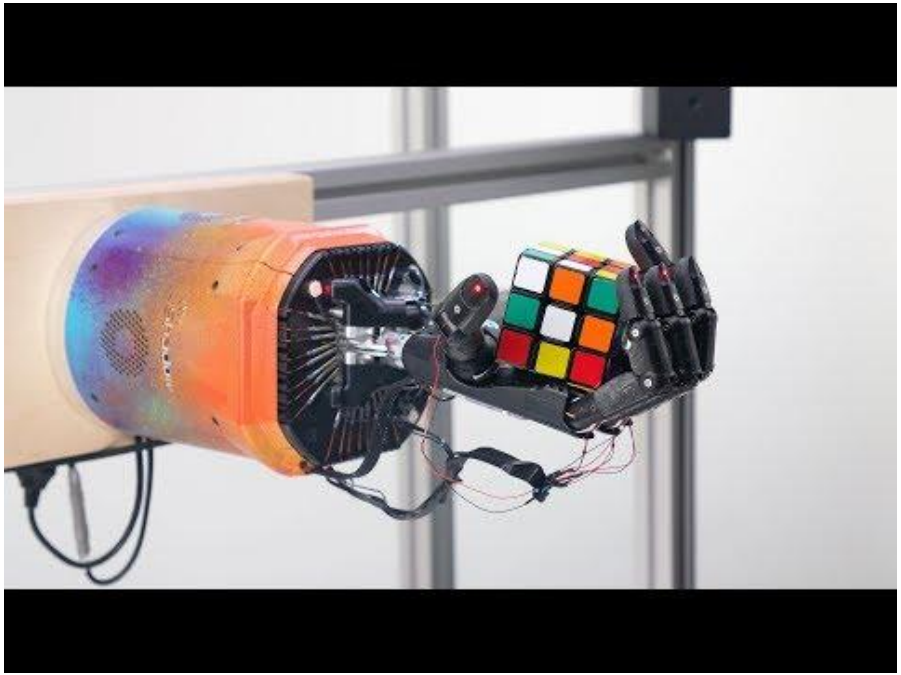
Source: https://openai.com/blog/emergent-tool-use/ (2019)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

7. Reinforcement learning

Solving a Rubik's cube with a robot hand



Source: https://openai.com/blog/solving-rubiks-cube/ (2019)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

7. Reinforcement learning

OpenAI vs. best human players in DOTA 2

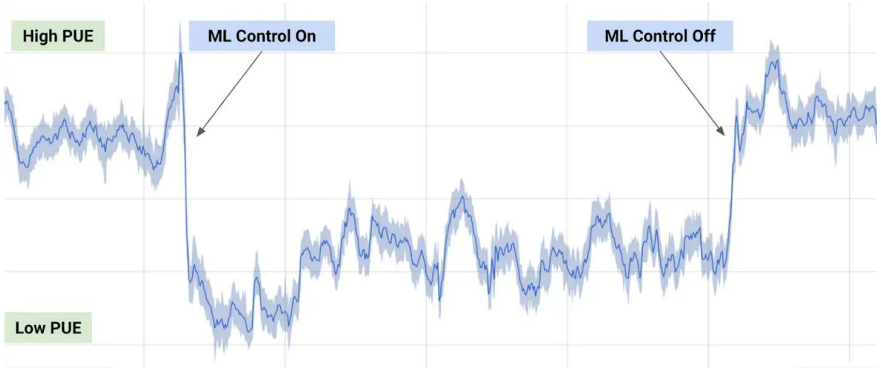
Source: https://openai.com/projects/five/ (2019)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

8. Energy

"DeepMind AI Reduces Google Data Centre Cooling Bill by 40%"



MENU nature

Article | Published: 19 February 2020

Closed-loop optimization of fast-charging protocols for batteries with machine learning

Peter M. Attia, Aditya Grover, Norman Jin, Kristen A. Severson, Todor M. Markov, Yang-Hung Liao, Michael H. Chen, Bryan Cheong, Nicholas Perkins, Zi Yang, Patrick K. Herring, Muratahan Aykol, Stephen J. Harris, Richard D. Braatz ✉, Stefano Ermon ✉ & William C. Chueh ✉

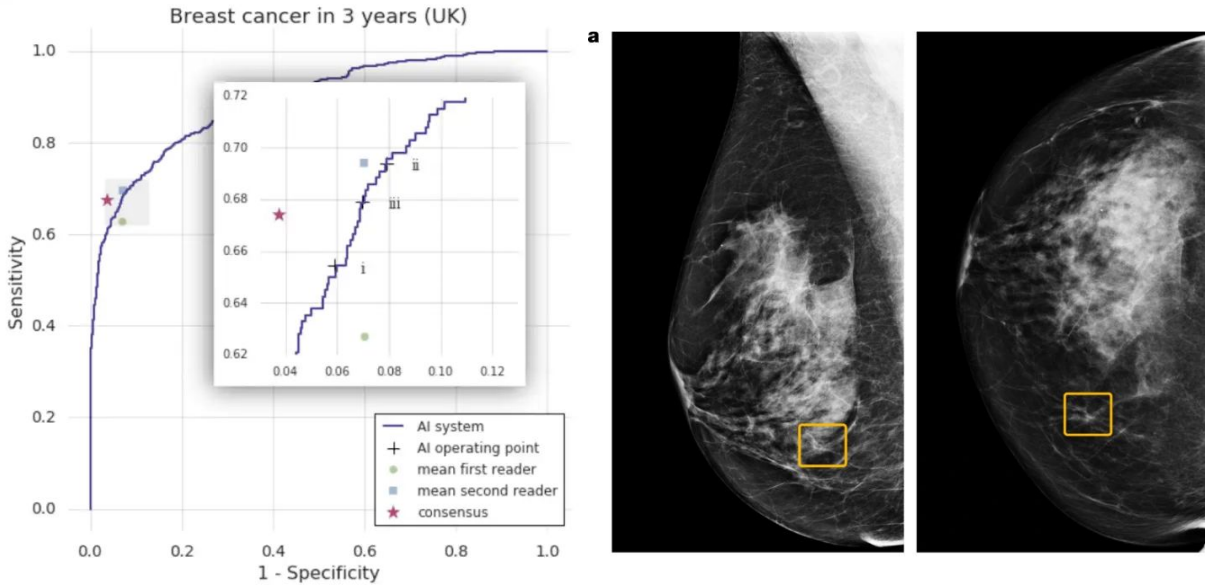
Source: (top) https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40 (2016)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

9. Healthcare

DeepMind's AI for breast cancer screening



Source: McKinney *et al.* "International evaluation of an AI system for breast cancer screening" (2020)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

10. Sustainable development

Estimating poverty using transfer learning and night lights

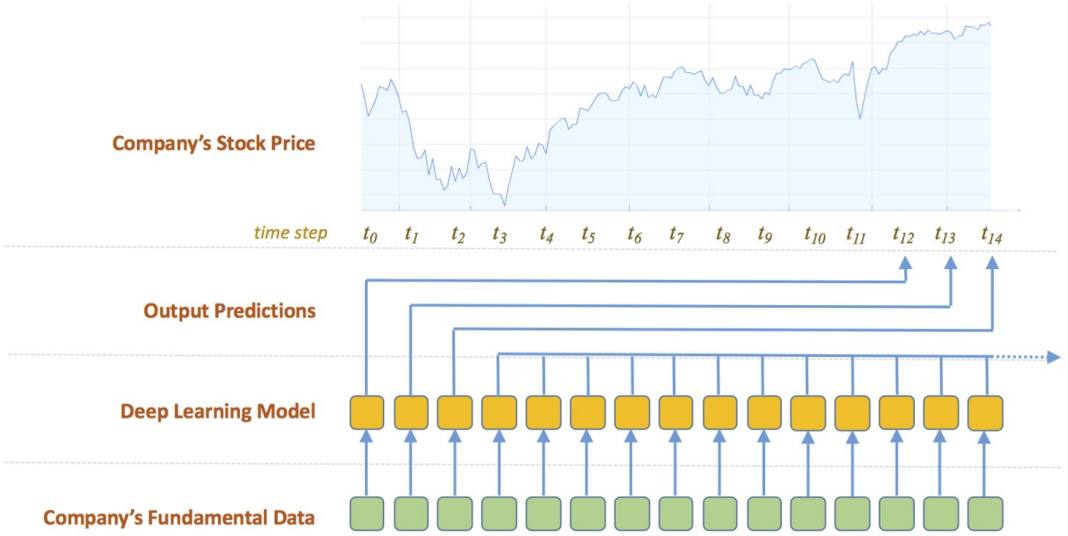
Source: Jean *et al.* "Combining satellite imagery and machine learning to predict poverty" (2016)

Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

11. Investing

Predicting stock price from company fundamentals



Source: Alberg and Lipton. "Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals" (2017)

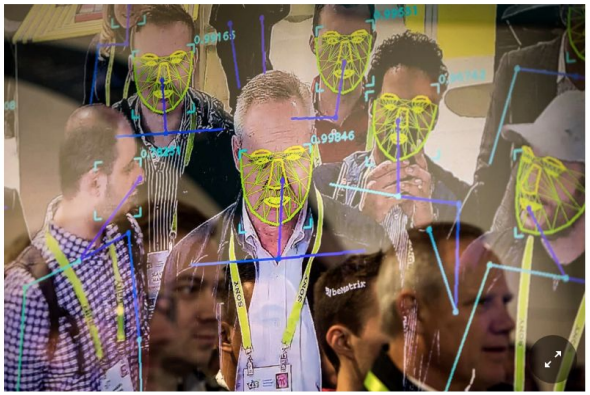
Deep learning examples

1. Optical character recognition (OCR)
2. Image classification
3. Text translation
4. Image generation
5. Text generation
6. Audio generation
7. Reinforcement learning
8. Energy
9. Healthcare
10. Sustainable development
11. Investing
12. Surveillance

12. Surveillance

The New York Times

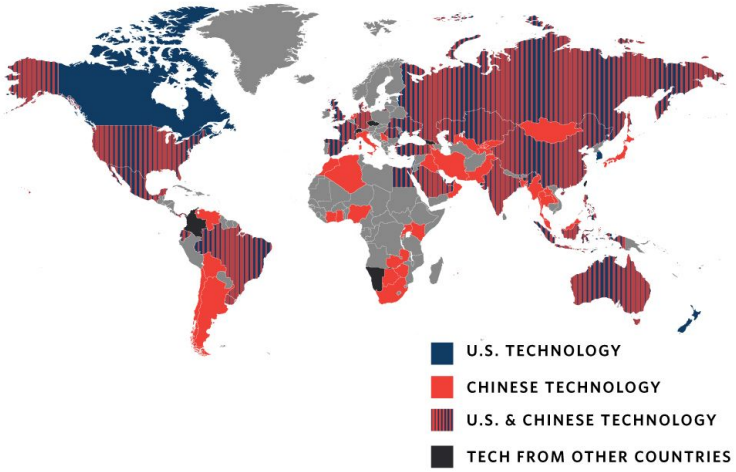
San Francisco Bans Facial Recognition Technology



Attendees interacting with a facial recognition demonstration at this year's CES in Las Vegas. Joe Buglewicz for The New York Times

MAP 1

AI Surveillance Technology Origin

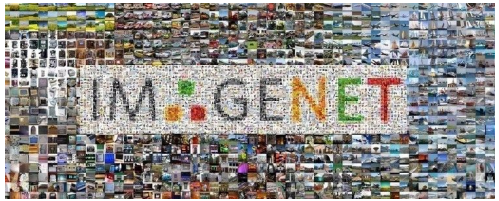


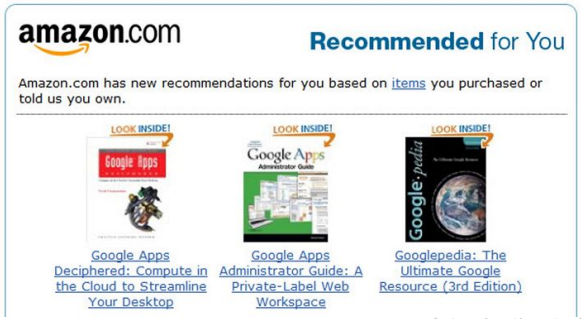
Sources: https://www.nytimes.com/2019/05/14/us/facial-recognition-ban-san-francisco.html (2019)
https://carnegieendowment.org/2019/09/17/global-expansion-of-ai-surveillance-pub-79847 (2019)

Why is deep learning at the forefront now?

Big data

Compute





Why is deep learning at the forefront now?

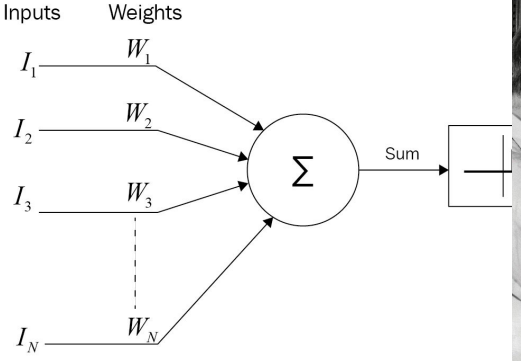
Algorithms

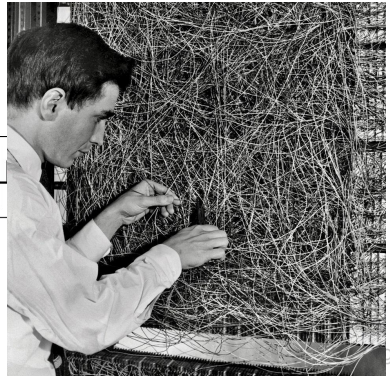
1943: McCulloch and Pitts develop a mathematical model of the neuron

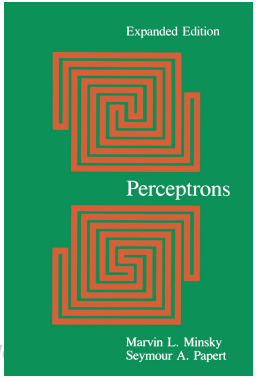
1957: Rosenblatt invents the perceptron algorithm and machine

1969: Minsky and Papert improve the perceptron and prove it cannot model nonlinearities like XOR

1974–1980: The First AI Winter



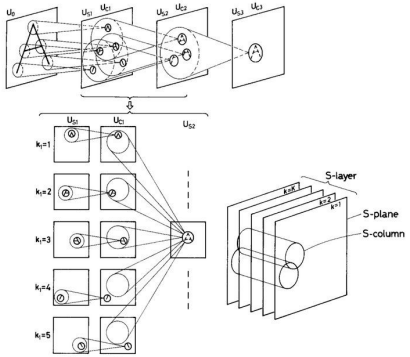




Why is deep learning at the forefront now?

Algorithms

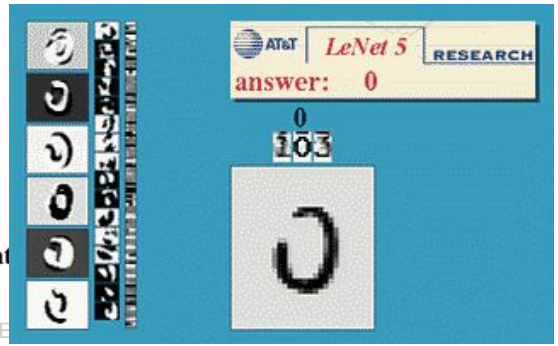
1980: Fukushima creates the “neocognitron”, introducing convolutional and downsampling layers



1986: Rumelhart, Hinton, and Williams apply backpropagation to train neural networks

D.E. Rumelhart, G.E. Hinton, R.J. Williams
Learning representation by back-propagat
errors. *Nature*, 323 (1986), pp. 533–536

1989: LeCun uses backprop to train convolutional neural networks to read digits



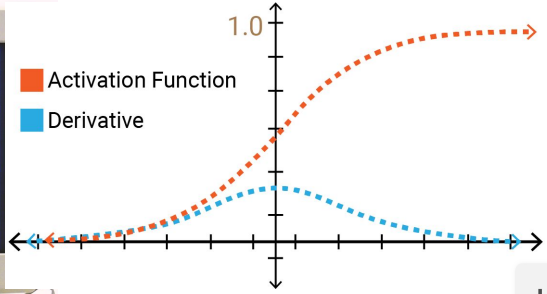
1987–1993: The Second AI Winter

Why is deep learning at the forefront now?

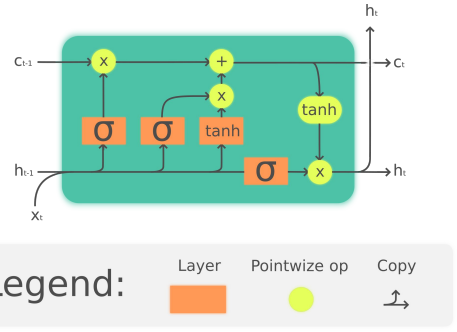
Algorithms

1990s: Computers become faster, GPUs developed

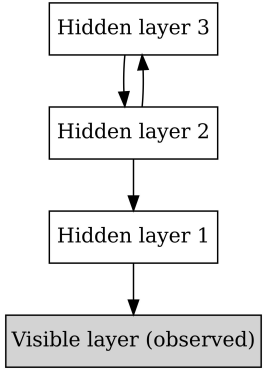
1991: The vanishing gradient problem is identified



1997: Hochreiter and Schmidhuber develop long short-term memory



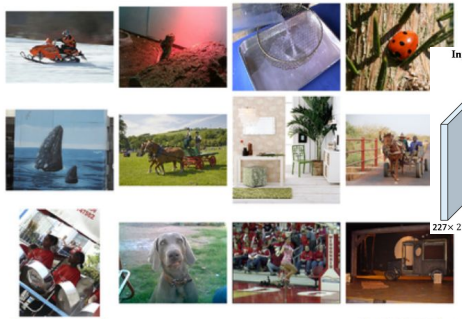
2006: Hinton and others show that deeper networks can be trained one layer at a time



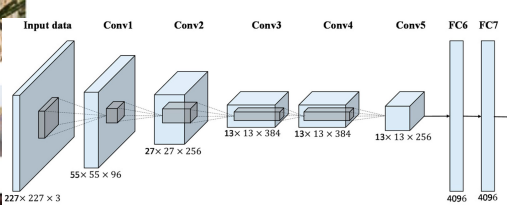
Why is deep learning at the forefront now?

Algorithms

2009: Fei-fei Li releases ImageNet, a classification dataset of 14 million labeled images



2012: AlexNet, a CNN, wins the Large Scale Visual Recognition Challenge — by a lot



2014: Goodfellow designs generative adversarial networks for data generation

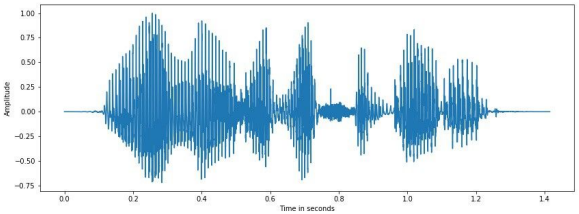
2015-2016: Deep learning frameworks like TensorFlow, Keras, and PyTorch are released

What's so special about deep learning?

Deep learning has made significant improvements in performance on tasks involving unstructured data

Images

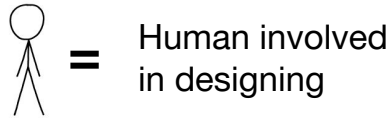
Speech

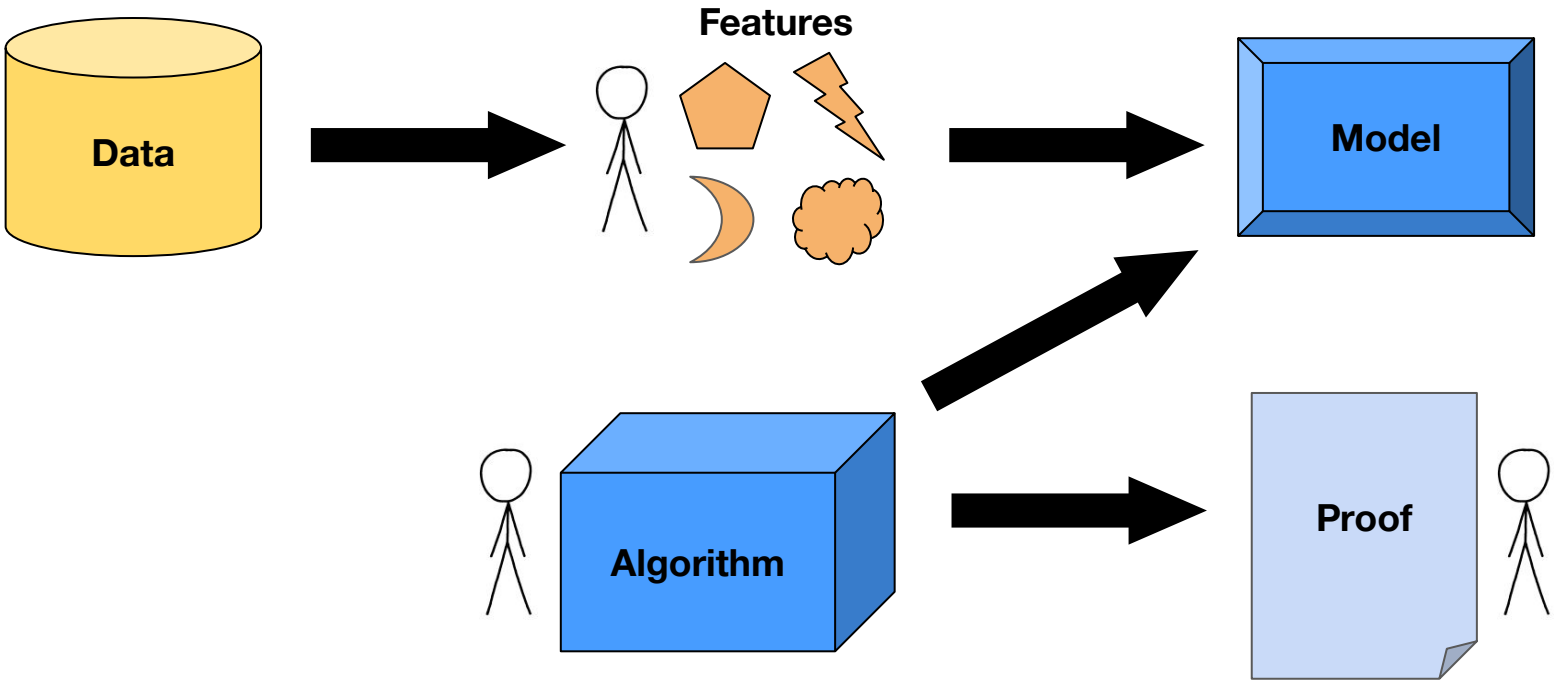


Text

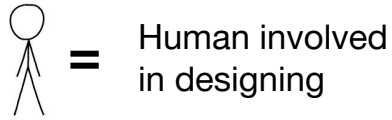
It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife. However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered ...

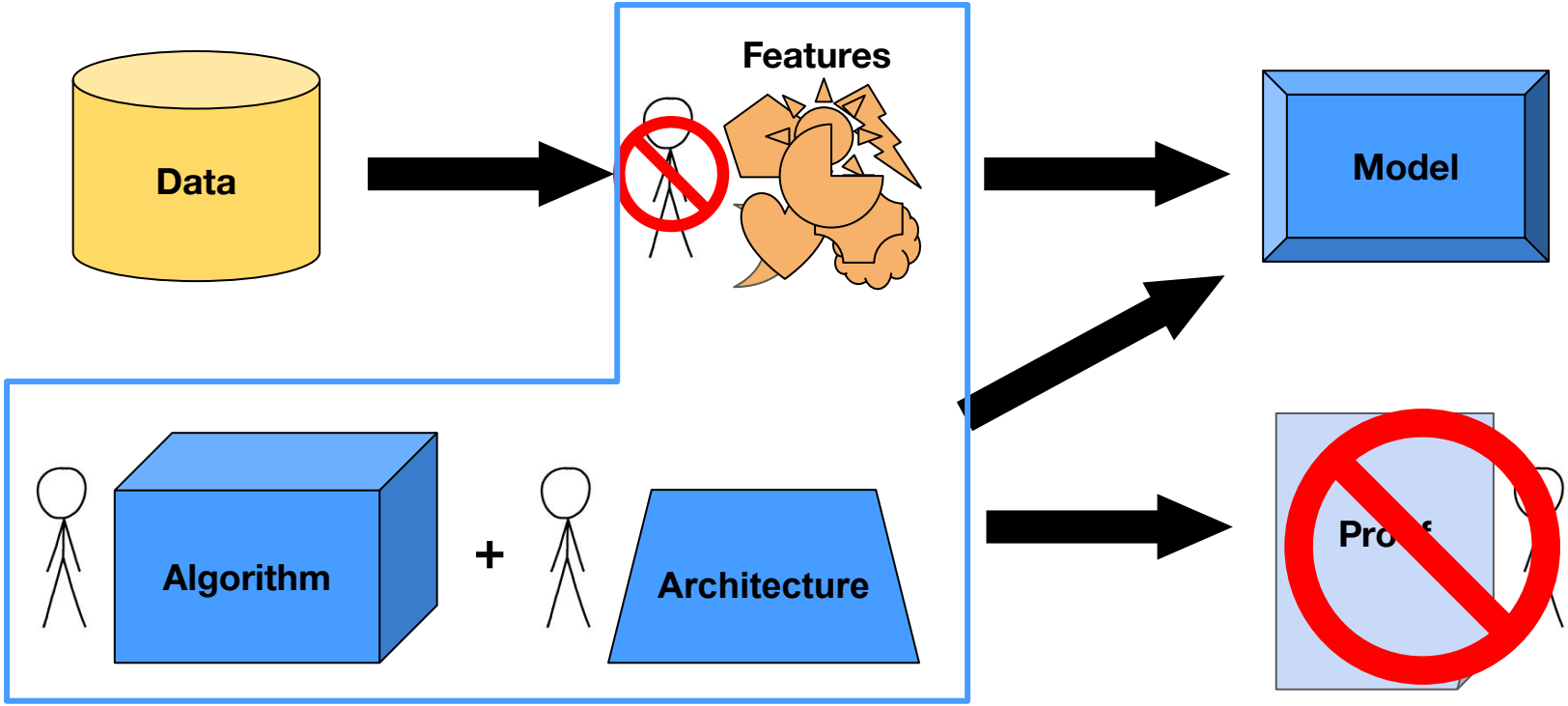
Traditional machine learning



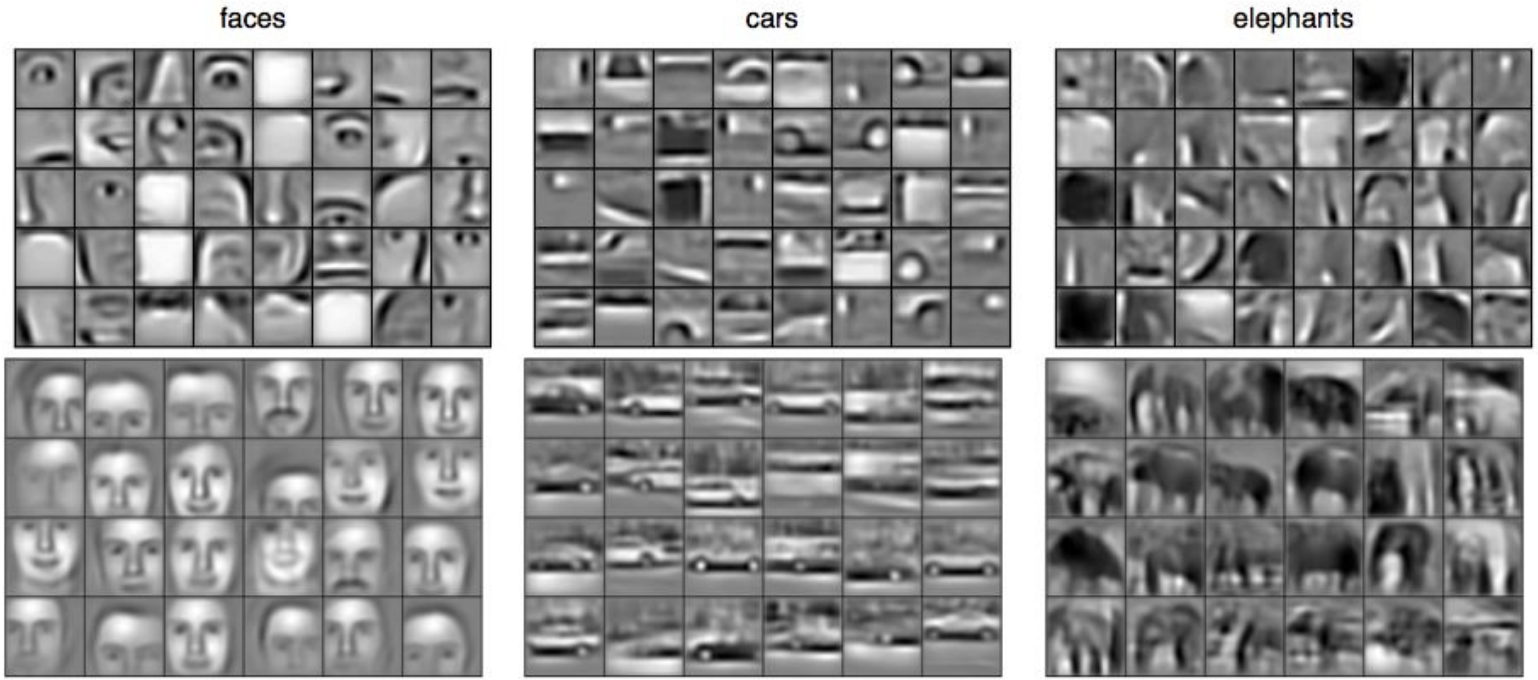


Deep learning



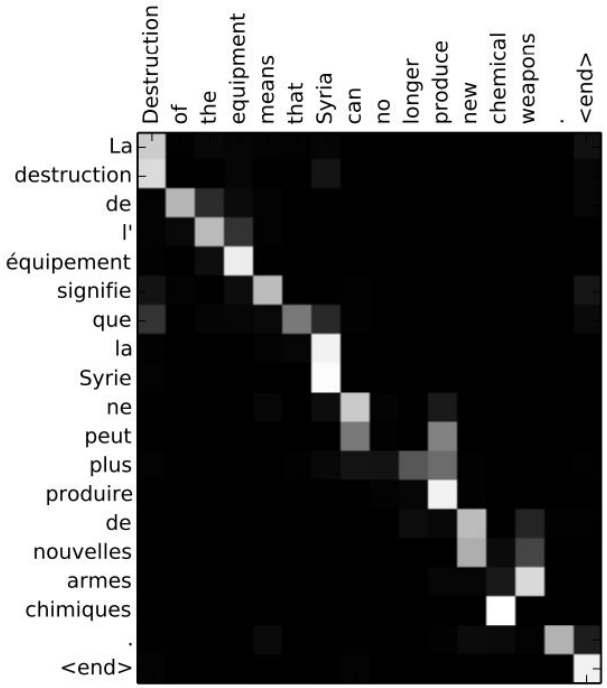


Deep learning-generated features



Lee *et al.* "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations" (2009)

Deep learning-generated features

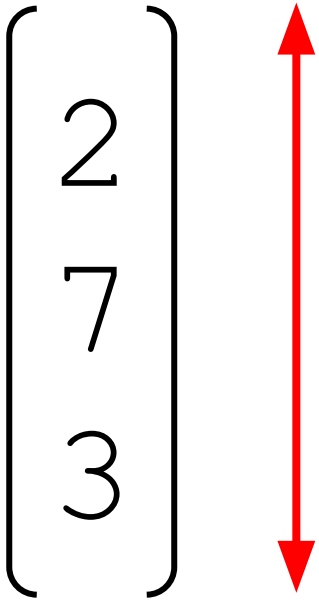


Bahdanau, Cho, and Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate” (2015)

Math Review

Vectors, matrices, and tensors

What is a vector?

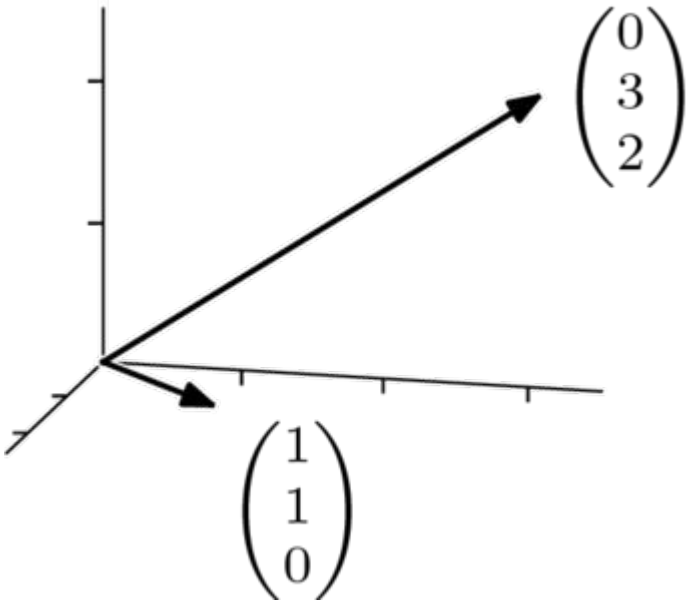


An array (or list) of numbers

Dimension = 3

$$\vec{v} \in \mathbb{R}^3$$

What is a vector?



In addition to linear algebra, you may have learned about vectors in physics courses as entities with direction and magnitude

Here are two more vectors of dimension 3, visualized in Euclidean space

Vector operations

1. Scalar multiplication

$$4 * \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} = \begin{pmatrix} 8 \\ 28 \\ 12 \end{pmatrix}$$

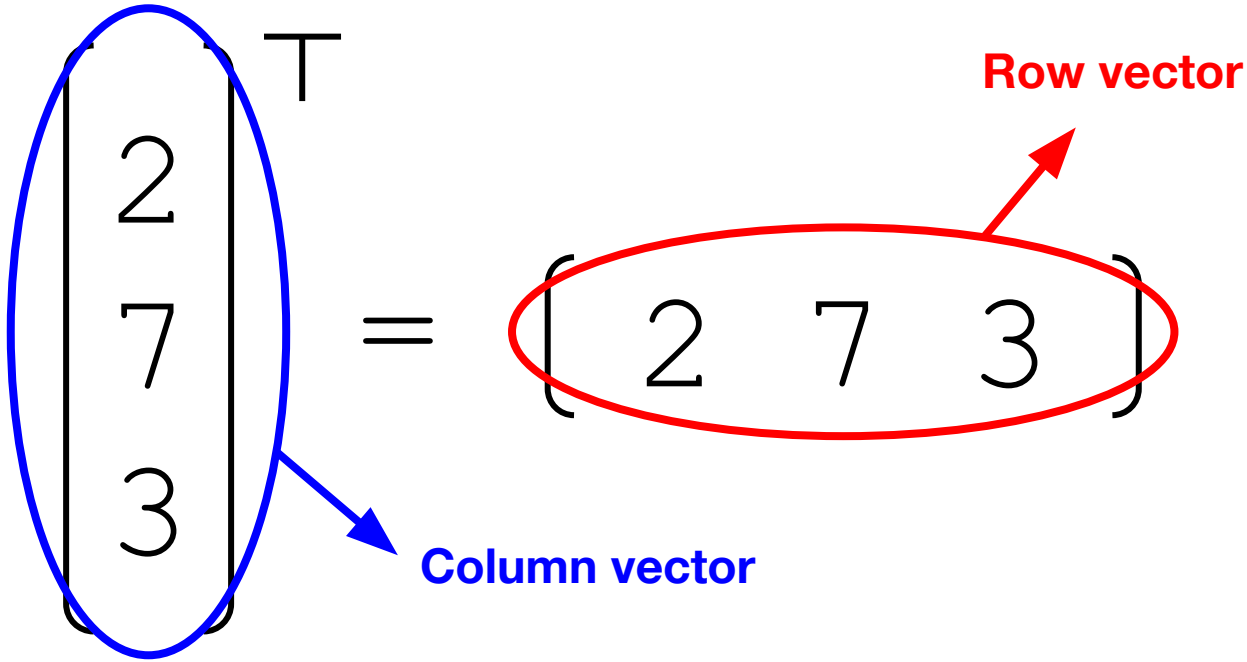
Vector operations

2. Addition

$$\begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \\ 6 \end{pmatrix} = \begin{pmatrix} 3 \\ 10 \\ 9 \end{pmatrix}$$

Vector operations

3. Transpose



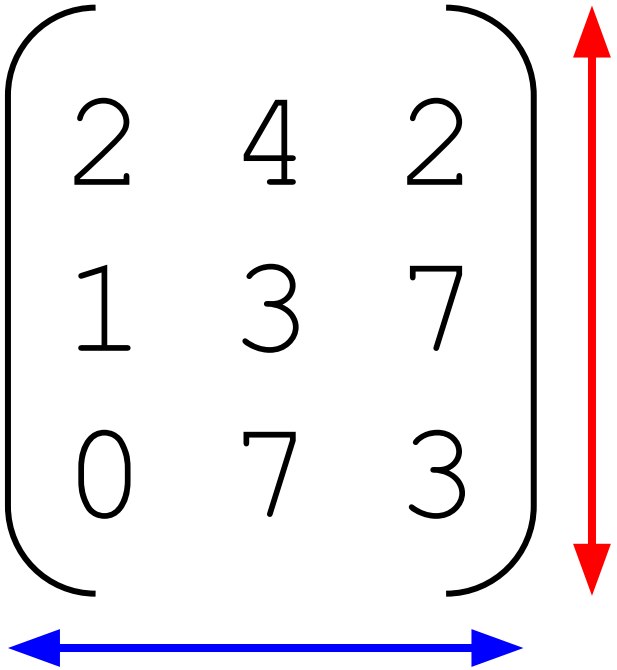
Vector operations

4. Inner product

$$\vec{u}^\top \vec{v} = w$$

$$\begin{pmatrix} 2 & 7 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 6 \end{pmatrix} = 2*1+7*3+3*6 = 41$$

What is a matrix?



A rectangular array (or table) of numbers

What is a matrix?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \end{array}$$

$$\mathbf{A} \in \mathbb{R}^{m \times n}$$

What is a matrix?

$$\begin{pmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 4 \\ 3 \\ 7 \end{bmatrix} & \begin{bmatrix} 2 \\ 7 \\ 3 \end{bmatrix} \end{pmatrix}$$

You can think of a matrix as a row vector of column vectors

What is a matrix?

$$\begin{pmatrix} \begin{bmatrix} 2 & 4 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 3 & 7 \end{bmatrix} \\ \begin{bmatrix} 0 & 7 & 3 \end{bmatrix} \end{pmatrix}$$

… or as a column vector of row vectors

Matrix operations

1. Scalar multiplication
2. Addition

$$2 * \begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 8 & 4 \\ 2 & 6 & 14 \\ 0 & 14 & 6 \end{pmatrix}$$

Matrix operations

3. Transpose

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Source: https://en.wikipedia.org/wiki/Transpose

Matrix operations

3. Transpose

$$\left(\begin{bmatrix}2\\1\\0\end{bmatrix}\begin{bmatrix}4\\3\\7\end{bmatrix}\begin{bmatrix}2\\7\\3\end{bmatrix}\right)^{\top} = \begin{pmatrix}2 & 1 & 0\\4 & 3 & 7\\2 & 7 & 3\end{pmatrix}$$

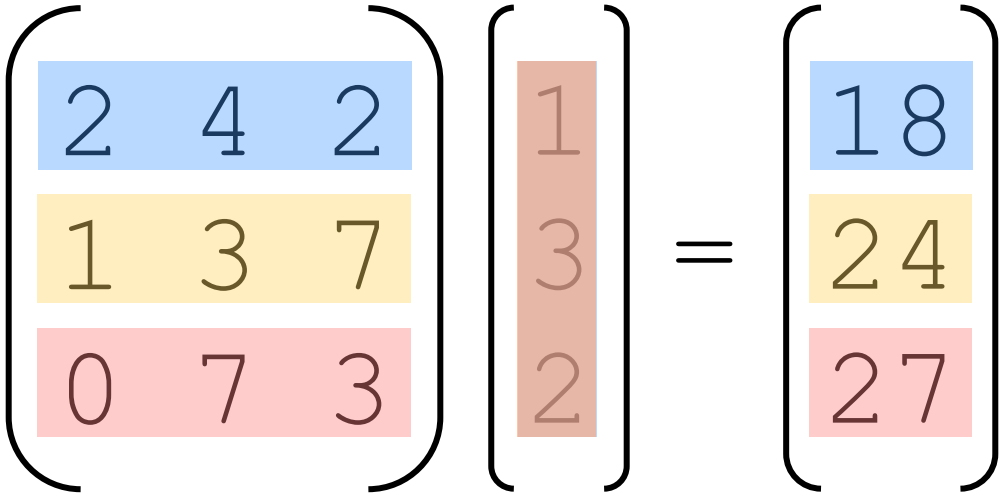
Matrix operations

3. Transpose

$$\left(\begin{bmatrix}2\\1\\0\end{bmatrix}\begin{bmatrix}4\\3\\7\end{bmatrix}\begin{bmatrix}2\\7\\3\end{bmatrix}\right)^{\top} = \begin{pmatrix}2 & 1 & 0\\4 & 3 & 7\\2 & 7 & 3\end{pmatrix}$$

Matrix operations

4. Multiplication with a vector



Matrix operations

4. Multiplication with a vector

$$\left(\begin{bmatrix}2\\1\\0\end{bmatrix}\begin{bmatrix}4\\3\\7\end{bmatrix}\begin{bmatrix}2\\7\\3\end{bmatrix}\right)\begin{bmatrix}1\\3\\2\end{bmatrix}=\begin{bmatrix}18\\24\\27\end{bmatrix}$$

Matrix operations

4. Multiplication with a vector

$$1\begin{pmatrix}2\\1\\0\end{pmatrix}+3\begin{pmatrix}4\\3\\7\end{pmatrix}+2\begin{pmatrix}2\\7\\3\end{pmatrix}=\begin{pmatrix}18\\24\\27\end{pmatrix}$$

Matrix operations

4. Multiplication with a vector

$$\begin{pmatrix} \begin{pmatrix} 2 & 4 & 2 \end{pmatrix} \\ \begin{pmatrix} 1 & 3 & 7 \end{pmatrix} \\ \begin{pmatrix} 0 & 7 & 3 \end{pmatrix} \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 18 \\ 24 \\ 27 \end{pmatrix}$$

Matrix operations

4. Multiplication with a vector

$$\begin{pmatrix} \begin{pmatrix} 2 & 4 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} \\ \\ \begin{pmatrix} 1 & 3 & 7 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} \\ \vdots \end{pmatrix} = \begin{pmatrix} 18 \\ 24 \\ 27 \end{pmatrix}$$

Matrix operations

4. Multiplication with a vector

$$\begin{pmatrix} \vec{a}_1^\top \\ \vec{a}_2^\top \\ \vec{a}_3^\top \end{pmatrix} \vec{b} = \begin{pmatrix} 18 \\ 24 \\ 27 \end{pmatrix}$$

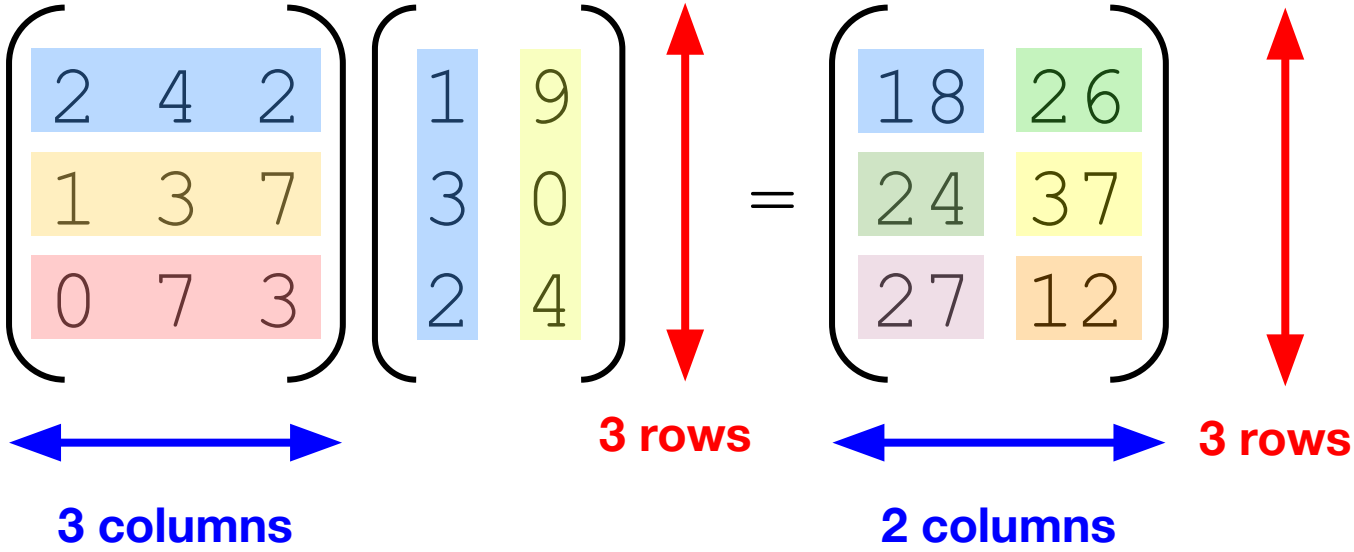
Matrix operations

4. Multiplication with a vector

$$\begin{pmatrix} \vec{a}_1^\top \vec{b} \\ \vec{a}_2^\top \vec{b} \\ \vec{a}_3^\top \vec{b} \end{pmatrix} = \begin{pmatrix} 18 \\ 24 \\ 27 \end{pmatrix}$$

Matrix operations

4. Multiplication with another matrix



Matrix operations

4. Multiplication with another matrix

$$\begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{pmatrix} \left(\begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} \begin{pmatrix} 9 \\ 0 \\ 4 \end{pmatrix} \right) = \begin{pmatrix} 18 & 26 \\ 24 & 37 \\ 27 & 12 \end{pmatrix}$$

Matrix operations

4. Multiplication with another matrix

$$\mathbf{A}\left(\begin{bmatrix}1\\3\\2\end{bmatrix}\begin{bmatrix}9\\0\\4\end{bmatrix}\right) = \left(\mathbf{A}\begin{bmatrix}1\\3\\2\end{bmatrix} \quad \mathbf{A}\begin{bmatrix}9\\0\\4\end{bmatrix}\right)$$

Matrix operations

4. Multiplication with another matrix

$$\begin{matrix} & & \vec{b_1} & \vec{b_2} \\ & & \downarrow & \downarrow \\ \vec{a_1}\rightarrow & \begin{bmatrix} 1 & 7 \\ 2 & 4 \end{bmatrix} & \cdot & \begin{bmatrix} 3 & 3 \\ 5 & 2 \end{bmatrix} \\ \vec{a_2}\rightarrow & & & \end{matrix} = \begin{bmatrix} \vec{a_1} \cdot \vec{b_1} & \vec{a_1} \cdot \vec{b_2} \\ \vec{a_2} \cdot \vec{b_1} & \vec{a_2} \cdot \vec{b_2} \end{bmatrix}$$

$A \qquad B \qquad C$

Source: https://ml-cheatsheet.readthedocs.io/en/latest/linear_algebra.html#matrix-multiplication

Poll: Matrix multiplication

Go to **PollEv.com/dlworkshop2020**

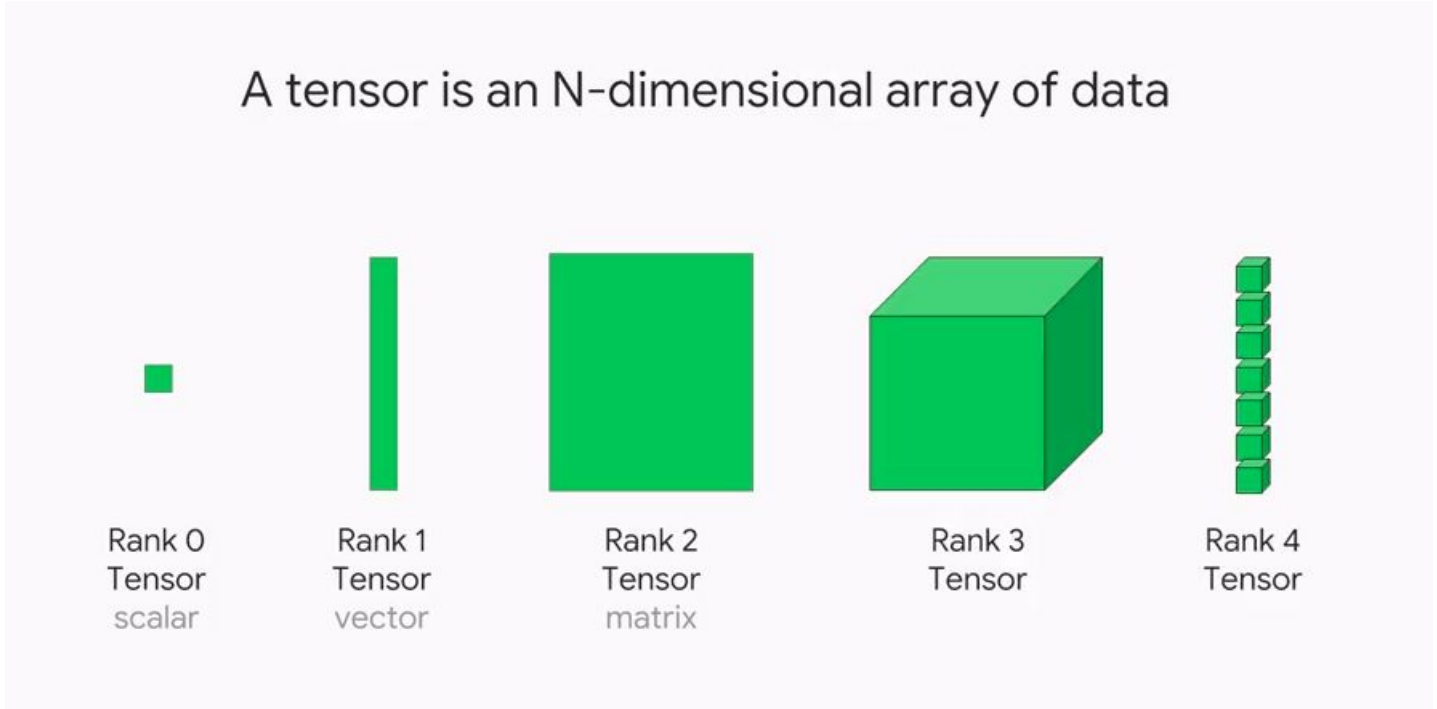
What are the dimensions of the matrix resulting from:

$$\begin{pmatrix} 2 & 4 & 2 & 8 & 4 & 1 & 5 & 0 \\ 1 & 3 & 7 & 9 & 3 & 2 & 6 & 8 \\ 0 & 7 & 3 & 1 & 0 & 4 & 9 & 3 \end{pmatrix} \begin{pmatrix} 3 & 4 \\ 2 & 9 \\ 0 & 1 \\ 5 & 5 \\ 3 & 7 \\ 6 & 4 \\ 2 & 1 \\ 4 & 0 \end{pmatrix}$$

A. 8 x 8 B. 8 x 2 C. 3 x 2 D. Not valid

Source: https://ml-cheatsheet.readthedocs.io/en/latest/linear_algebra.html#matrix-multiplication

What is a tensor?



Source: https://medium.com/mlait/tensors-representation-of-data-in-neural-networks-bbe8a711b93b

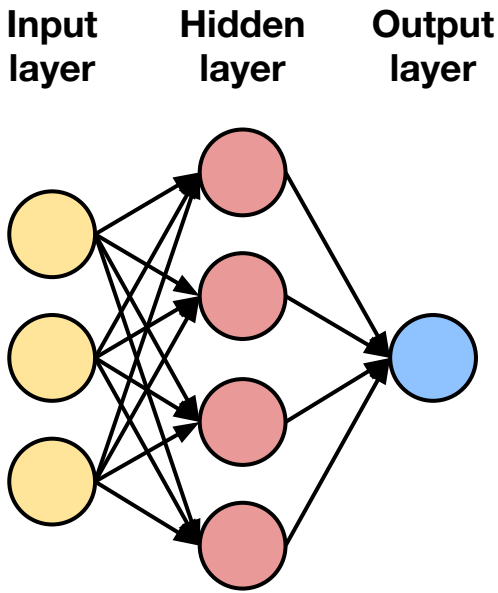
Neural Networks

Neural networks

We'll start with a *fully connected* neural network.

1. Single layer
2. Single neuron
3. Multiple layers
4. Input and output

Later, we'll cover other types of layers (convolution, max-pool, …)



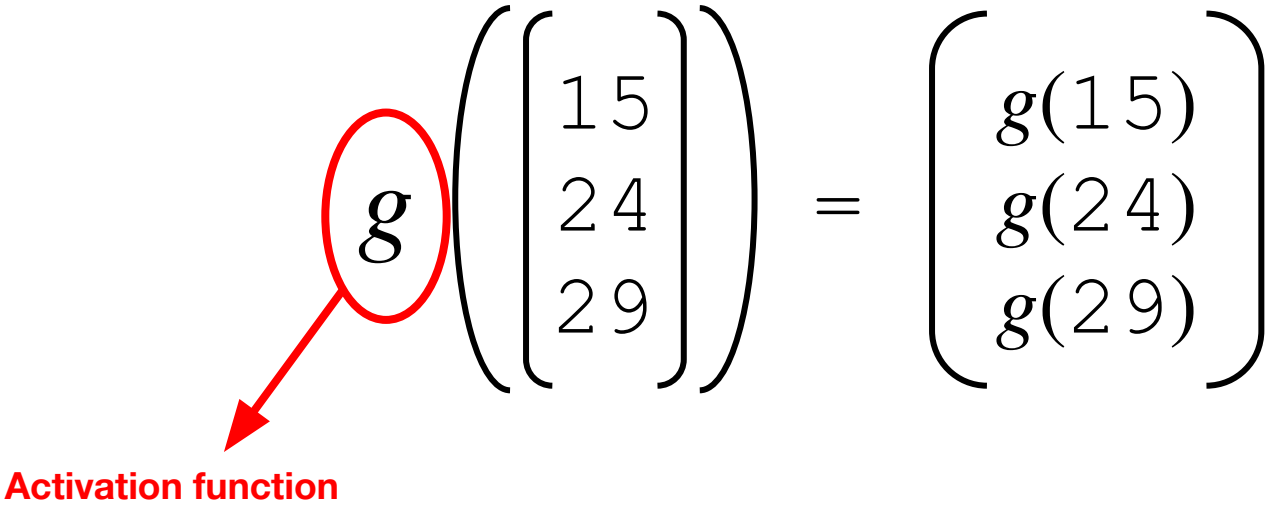
What does a single layer in a neural network do?

First, it involves some matrix operations...

$$\begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 15 \\ 24 \\ 29 \end{pmatrix}$$

What does a single layer in a neural network do?

Then, the intermediate values are passed through a nonlinear function...

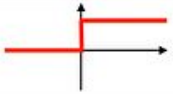
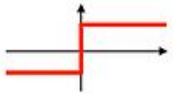
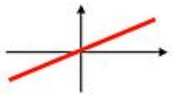
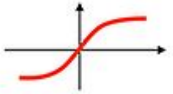
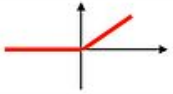
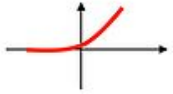


Activation functions

There exist a number of activation functions, some of which were designed with neurons in mind

What's the purpose?

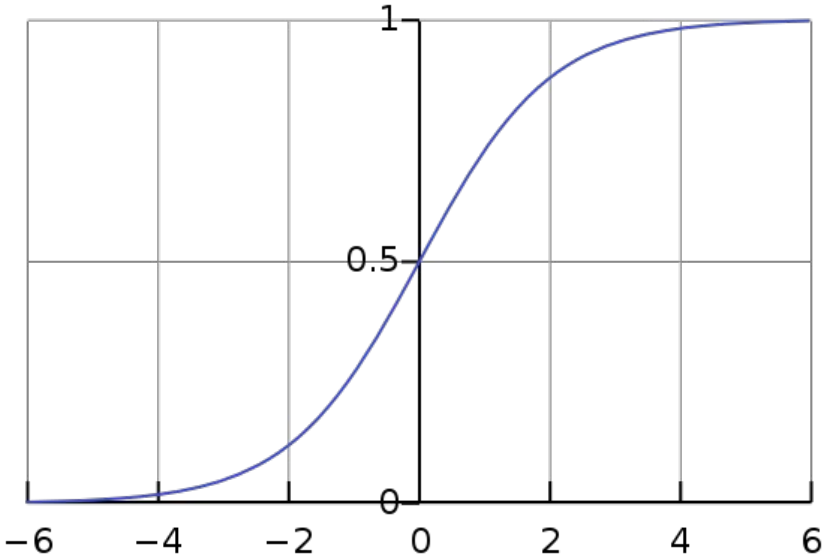
- Most phenomena are not linear, so we introduce some nonlinearities to the network

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

"A visual proof that neural nets can compute any function": http://neuralnetworksanddeeplearning.com/chap4.html

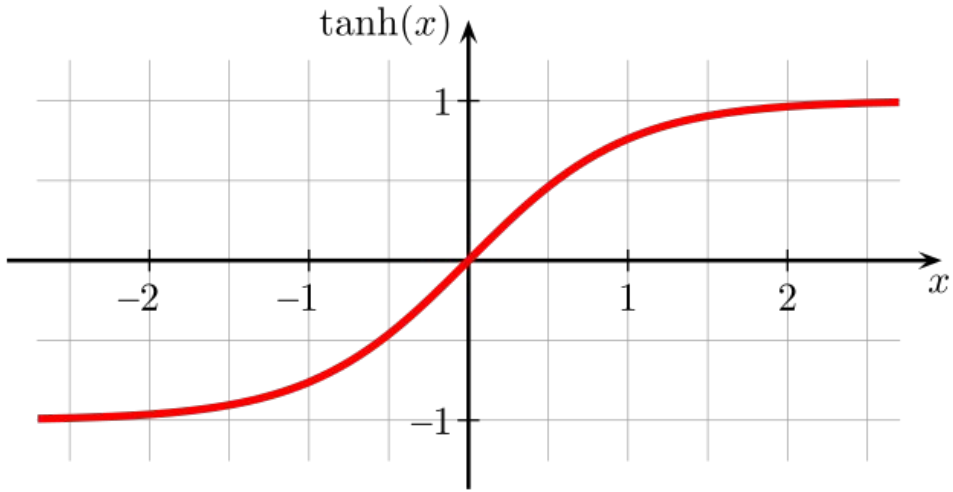
Sigmoid (logistic) activation function

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



Hyperbolic tangent activation function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



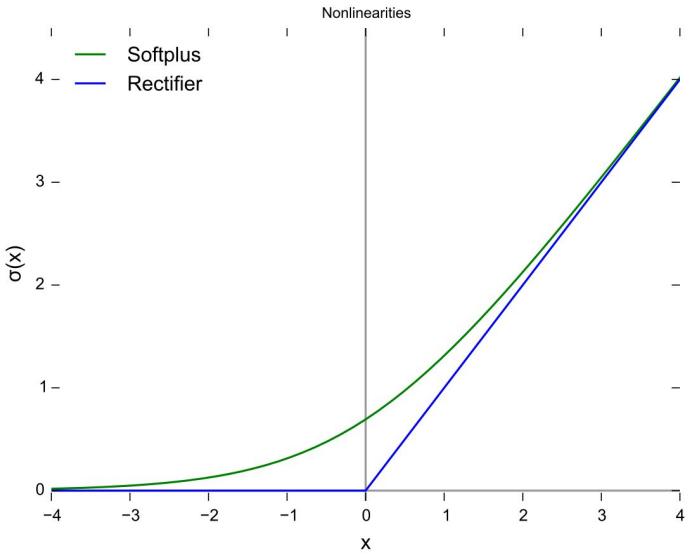
ReLU activation function

Stands for **Re**ctified **L**inear **U**nit

$$\mathrm{ReLU}(x) = \max(0, x)$$

A smooth version of ReLU is softplus

$$\mathrm{softplus}(x) = \ln(1 + e^x)$$



Poll: Activation functions

Go to **PollEv.com/dlworkshop2020**

As its input goes to infinity, the output of the logistic function goes to...

$$\lim_{x \to \infty} \frac{1}{1 + e^{-x}} =$$

A. 0 B. 0.5 C. 1 D. Infinity

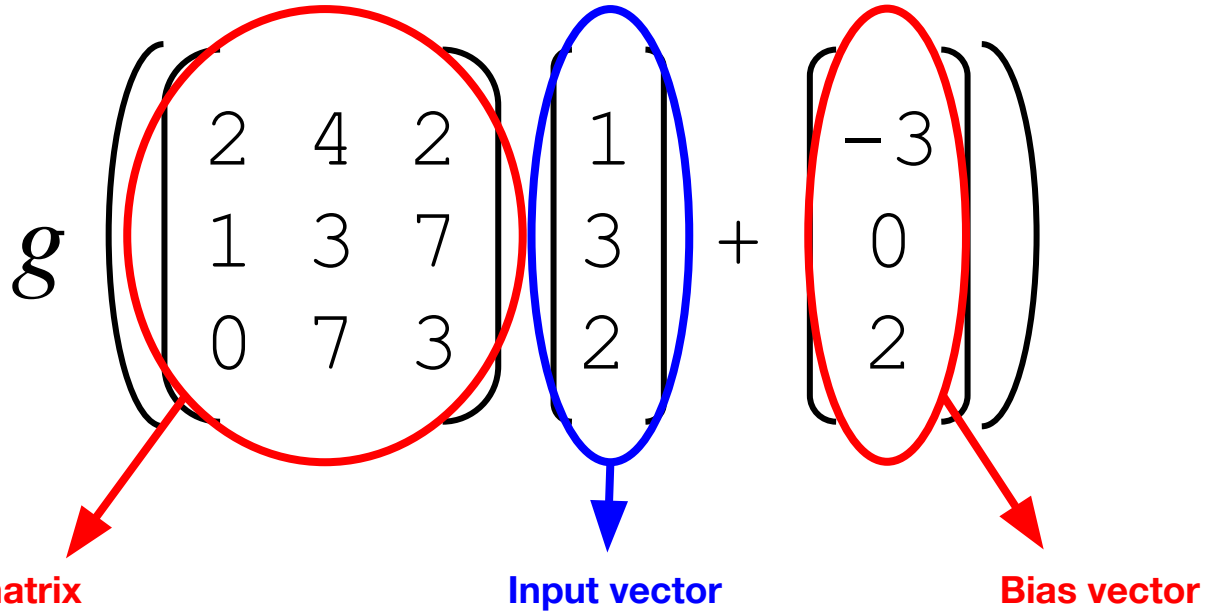
Activation functions FAQ

1. Why these activation functions? Why not x^2 or other nonlinear functions?

2. How does one choose among the various options for activation functions?

What are the matrix and vectors?

Putting the linear operations and nonlinearity together, we get:



What are the matrix and vectors?

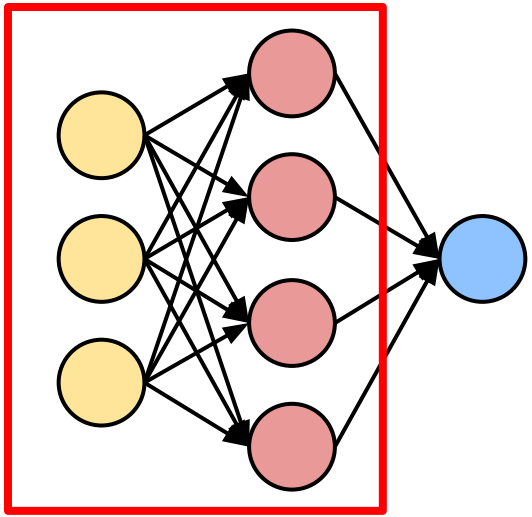
Putting the linear operations and nonlinearity together, we get:

$$g\left(\begin{bmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ 0 \\ 2 \end{bmatrix}\right)$$

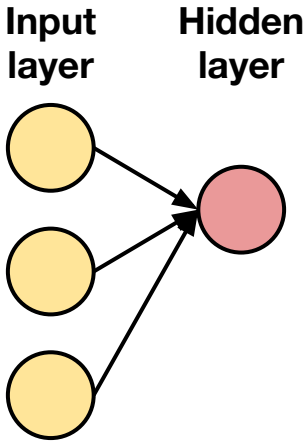
Trainable parameters

A more compact notation

$$g(\mathbf{W}\vec{x} + \vec{b})$$



What does a single neuron do?



Each circle is one element of an input or hidden layer

Each edge is a weight

$$g(\vec{w}_1^{\top}\vec{x} + b_1)$$

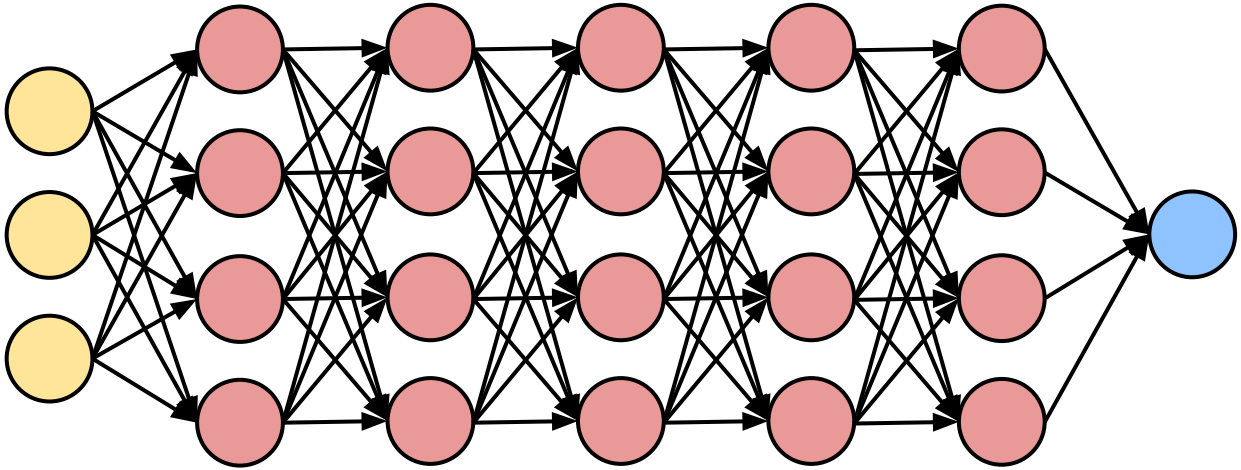
Weights for each neuron

$$\mathbf{W} = \begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{pmatrix} = \begin{pmatrix} w_1^\top \\ w_2^\top \\ w_3^\top \end{pmatrix}$$

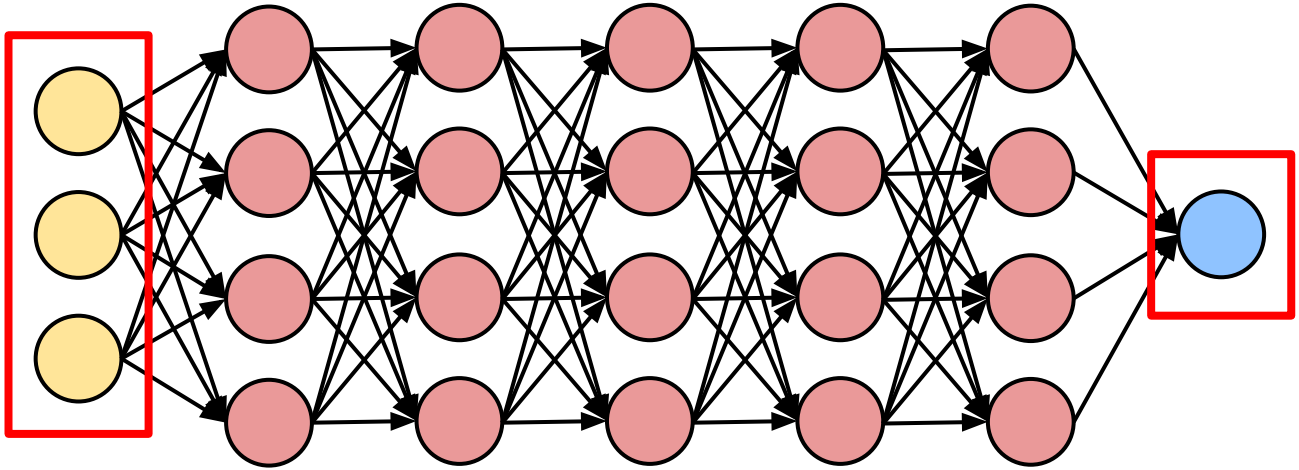
A fully connected layer with weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$

Stacking multiple layers

It becomes a *deep* neural network when you use many layers



What are the inputs and outputs?



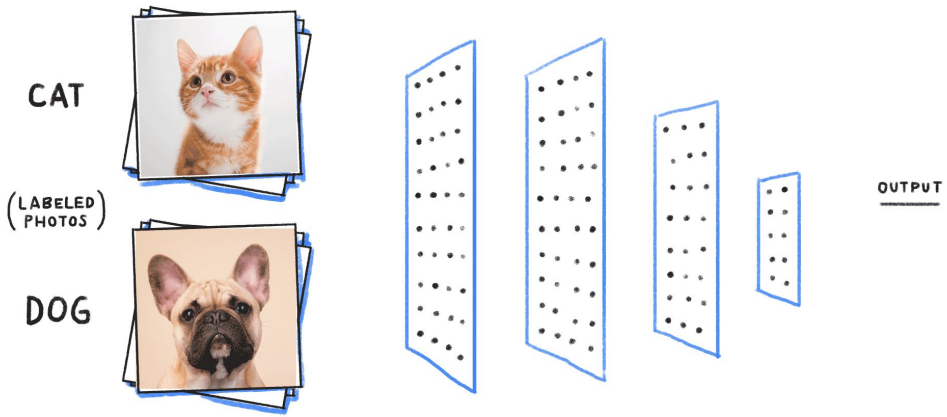
Defining the problem

- You want to predict some output **y** given input **x**
- If **y** is a discrete output (e.g. image classes), then it's a *classification problem*
- If **y** is a continuous variable (e.g. price), then it's a *regression problem*

$$y = \mathrm{DNN}(\vec{x})$$

Defining the problem

- If **y** is a discrete output, you can use one-hot encoding
- If **y** is a continuous variable, it can be output by the network directly



Classes = {cat, dog}

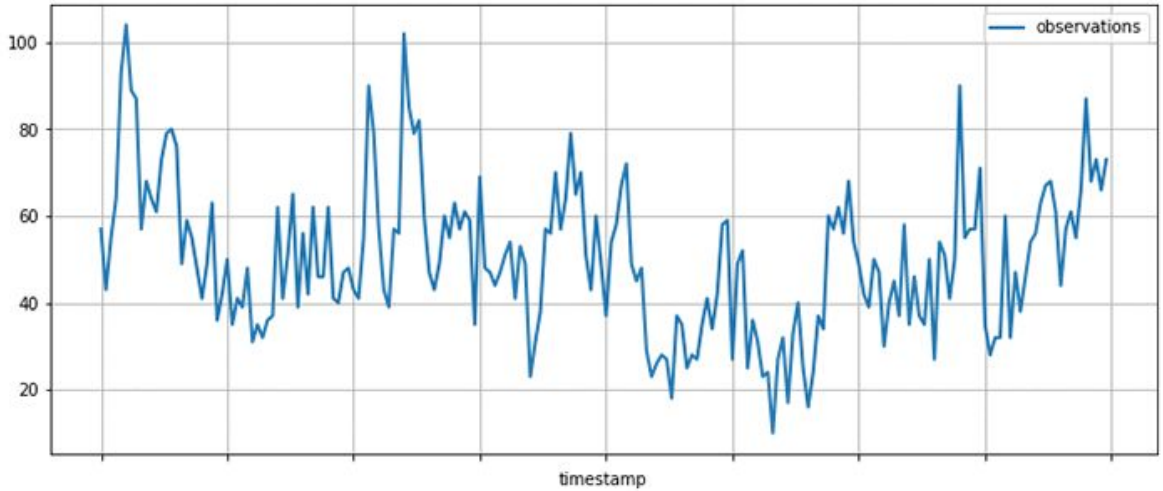
$$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$ "cat"

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$ "dog"

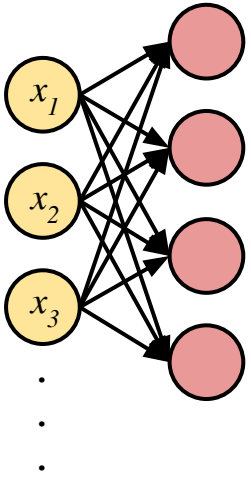
https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

Defining the problem

- If **x** is a discrete output, you can use one-hot encoding
- If **x** is a continuous variable, it can be input to the network directly



$$x = \begin{pmatrix} 59 \\ 42 \\ 63 \\ 93 \\ 102 \\ 88 \\ 86 \\ 58 \\ \dots \end{pmatrix}$$

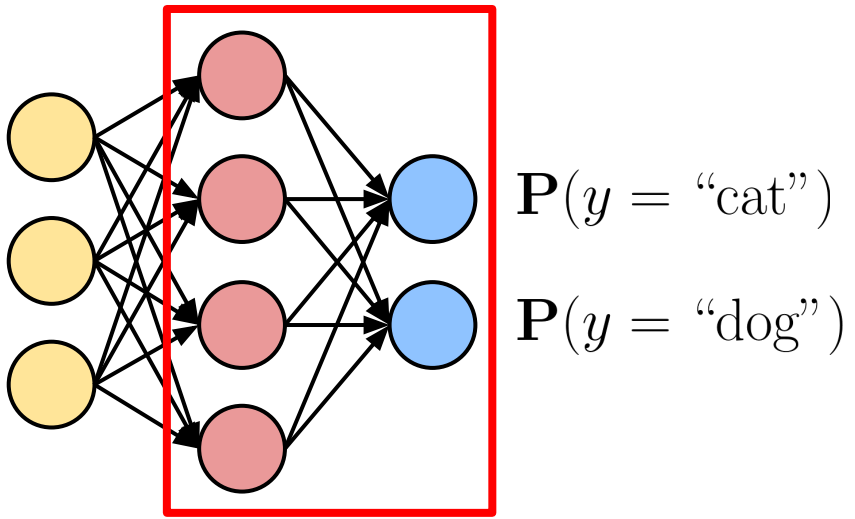


Last layer activation function

For classification, each neuron in the output represents one output class

You want the output to represent the probability that the input belongs to each class

What constraints must the output satisfy?



Last layer activation function

What constraints must the output satisfy?

- Probabilities are positive
- Probabilities sum to 1

$$\mathbf{P}(y = \text{“cat”}) \geq 0$$
$$\mathbf{P}(y = \text{“dog”}) \geq 0$$

$$\mathbf{P}(y = \text{“cat”}) + \mathbf{P}(y = \text{“dog”}) = 1$$

Softmax activation function

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Ensures positivity

Normalizes sum to 1

$$\mathbf{z} = \begin{pmatrix} 1.1 \\ 0.4 \\ -8.2 \\ -0.9 \\ 3.7 \end{pmatrix} \quad e^{\mathbf{z}} = \begin{pmatrix} e^{1.1} \\ e^{0.4} \\ e^{-8.2} \\ e^{-0.9} \\ e^{3.7} \end{pmatrix} \quad \frac{e^{\mathbf{z}}}{\sum_{j=1}^{n} e^{z_j}} = \begin{pmatrix} e^{1.1}/\sum_j e^{\mathbf{z}_j} \\ e^{0.4}/\sum_j e^{\mathbf{z}_j} \\ e^{-8.2}/\sum_j e^{\mathbf{z}_j} \\ e^{-0.9}/\sum_j e^{\mathbf{z}_j} \\ e^{3.7}/\sum_j e^{\mathbf{z}_j} \end{pmatrix} = \begin{pmatrix} 0.07 \\ 0.03 \\ 0.00 \\ 0.01 \\ 0.89 \end{pmatrix}$$

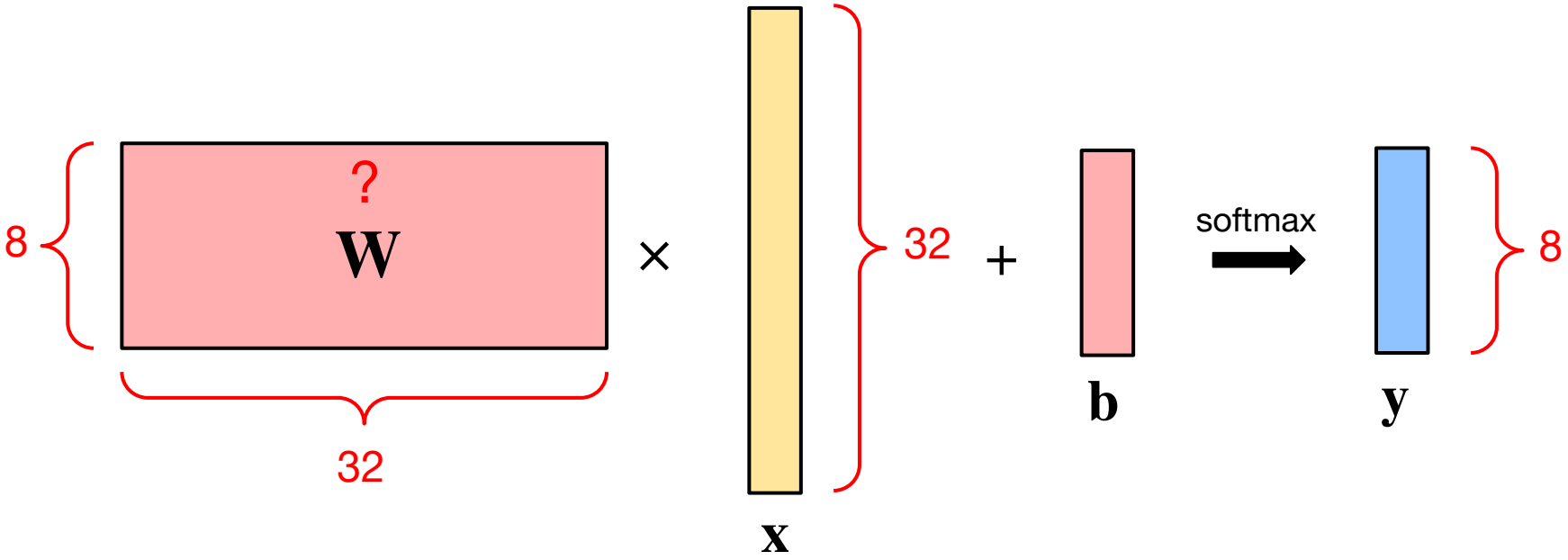
Poll: Weight matrix

Go to **PollEv.com/dlworkshop2020**

Suppose we have a neural network with 1 fully connected hidden layer. The input is a 32-dimensional vector. The output is a probability distribution across 8 classes. The weight matrix of the hidden layer should have dimension...

A. 32x1 B. 8x32 C. 32x8 D. 32x32

Poll: Weight matrix

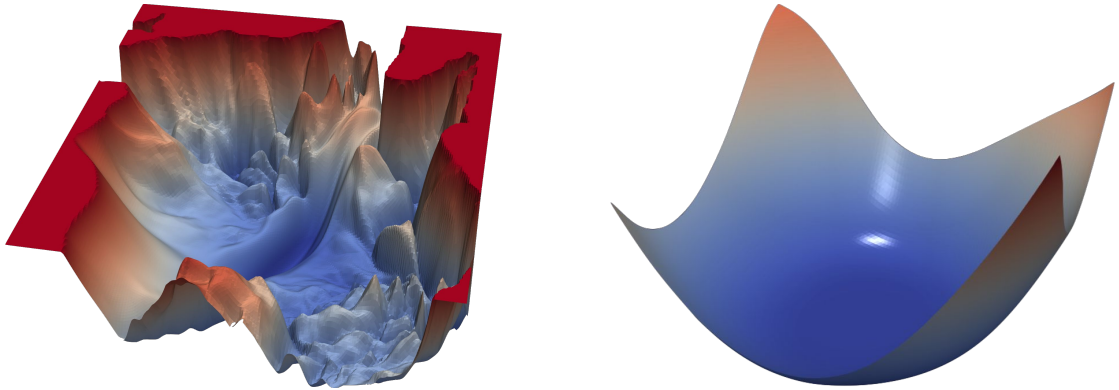


ICME Summer Workshops 2020

Introduction to Deep Learning

Session 2: 10:45—12:00 PM
Instructor: Sherrie Wang

icme-workshops.github.io/deep-learning



Li *et al.* "Visualizing the Loss Landscape of Neural Nets" (2018)

Workshop Schedule

Session 1 (9:00—10:30 AM)
- Introduction
- Current state-of-the-art in deep learning
- Math review
- Fully connected neural networks

Session 2 (10:45—12:00 PM)
- Loss functions
- Gradient descent
- Backpropagation
- Overfitting and underfitting

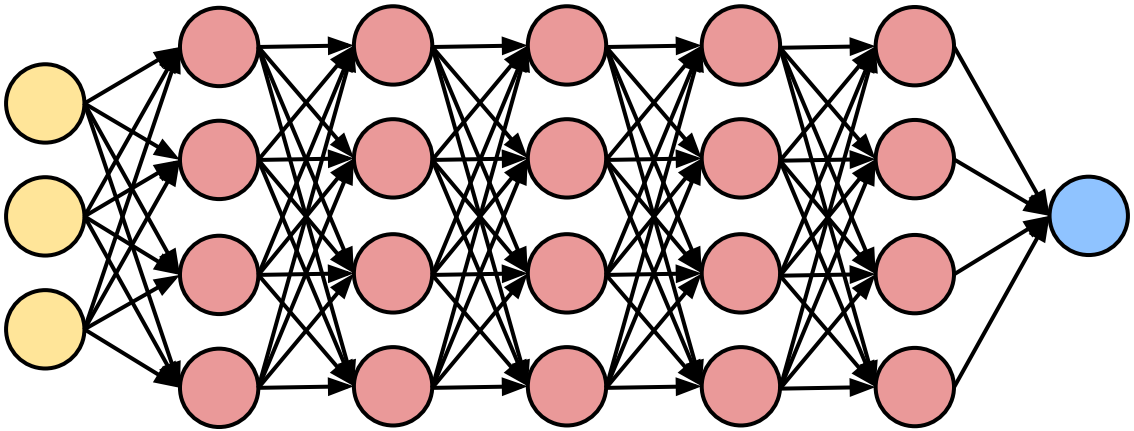
Lunch (12:00—2:00 PM)

Session 3 (2:00—3:15 PM)
- Convolutional neural networks
- Recurrent neural networks
- Other architectures
- Deep learning libraries
- Hands-on coding session in Tensorflow

Session 4 (3:30—4:45 PM)
- Hands-on coding session in Keras
- Hands-on coding session on transfer learning
- Failures of deep learning

Loss Functions

Training neural networks



$$g\left(\begin{bmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ 0 \\ 2 \end{bmatrix}\right)$$

Trainable parameters

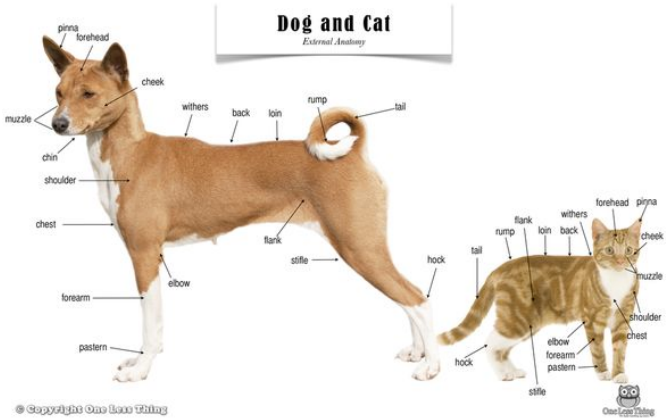
Trained how?

Training neural networks

Machine learning: builds a model based on “training data” in order to make predictions without being explicitly programmed to do so

Not machine learning

Human says: “dogs are like this, and cats are like this…”



Machine learning

“Here’s a bunch of pictures of cats and dogs. Figure out how to sort them.”

https://mc.ai/deep-learning-vs-machine-learning-a-simple-explanation/

Training neural networks

Dataset

training

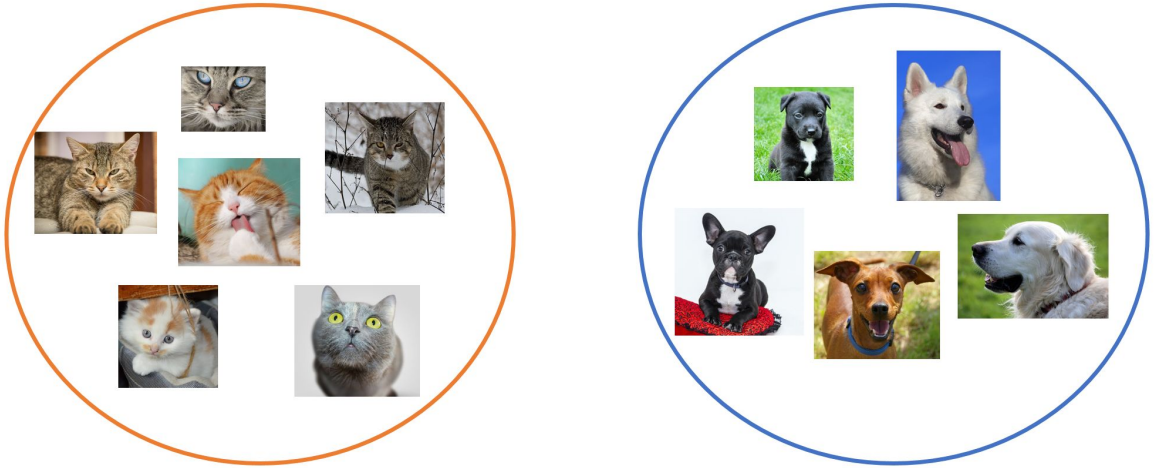
Weights & biases

$$\begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \end{pmatrix} \quad \begin{pmatrix} -3 \\ 0 \\ 2 \end{pmatrix}$$

https://mc.ai/deep-learning-vs-machine-learning-a-simple-explanation/

Supervised learning

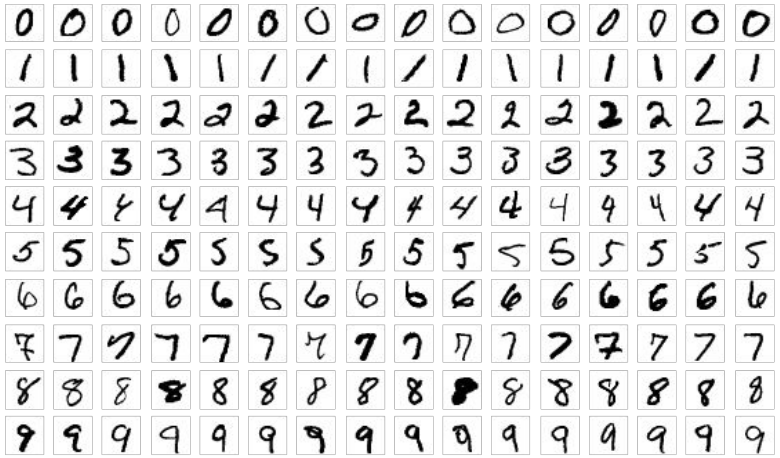
Dataset has labels



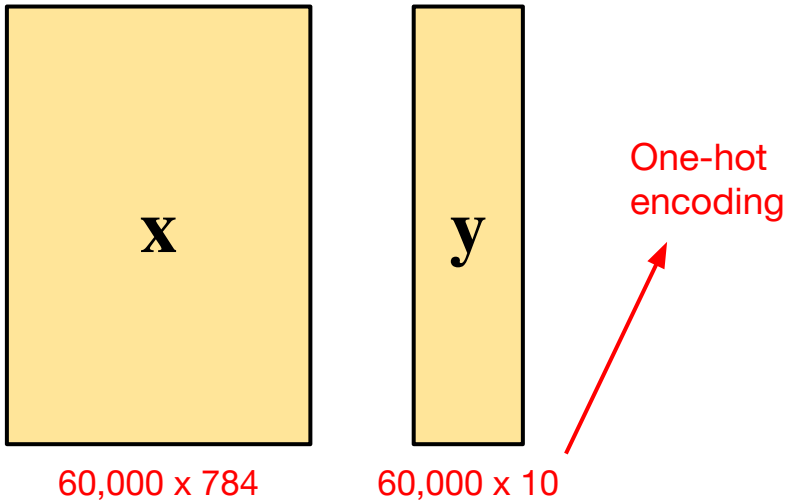
https://blogs.sas.com/content/subconsciousmusings/2017/09/25/machine-learning-concepts-styles-machine-learning/

Supervised learning

MNIST Dataset (Handwritten digits)



Each image is 28 x 28 pixels



The goal of training

Intuitively, we want to tweak the weights of the neural network until the network predicts the correct output for most of the examples in the training set.

Neural network #1

$$\text{softmax}\left(\begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0.98 \\ 0.02 \end{bmatrix}$$

Weights Input Prediction

Neural network #2

$$\text{softmax}\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0.12 \\ 0.88 \end{bmatrix}$$

Weights Input Prediction

True label $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

We prefer network #1 to #2

Optimization formulation

We will frame this as an optimization problem:

$$\max_{\mathbf{W}_i, \mathbf{b}_i} \text{accuracy}(\text{DNN}(\mathbf{W}_i, \mathbf{b}_i))$$

Accuracy is the fraction of training examples that the model predicts correctly

Optimization formulation

Accuracy is not a continuous function, so it is hard to directly optimize for maximum accuracy.

So we use **loss functions**.

Think of losses as continuous proxies to accuracy, so that we can formulate it as an optimization problem and solve it.

Loss functions

True label $\mathbf{y}$ = $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ Predicted probabilities = $\begin{pmatrix} 0.2 \\ 0.7 \\ 0.1 \end{pmatrix}$

In English, we want to **reward the neural network** for predicting class 2 with higher probability than classes 1 and 3, while incentivizing it to become even better at making this distinction

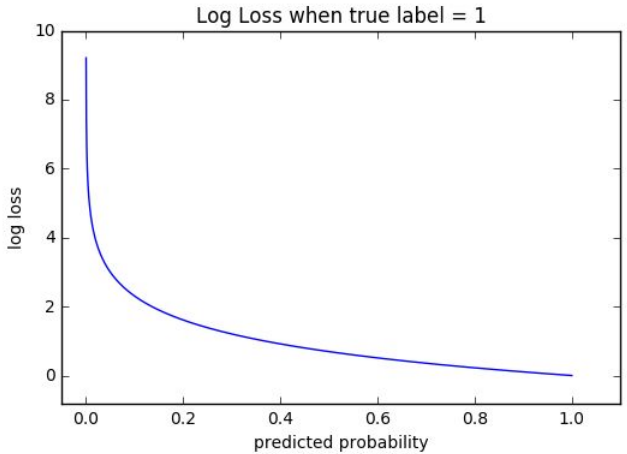
Cross entropy loss

Predicted probabilities = $\begin{pmatrix} 0.2 \\ 0.7 \\ 0.1 \end{pmatrix}$ $-\log(0.7) = 0.35$

Predicted probabilities = $\begin{pmatrix} 0.6 \\ 0.1 \\ 0.3 \end{pmatrix}$ $-\log(0.1) = 2.3$

Cross entropy loss

$$\mathrm{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \log \hat{y}_i$$



Cost function

Cost = Average loss over training examples

$$\min_{\mathbf{W}_i, \mathbf{b}_i} \text{cost}(\text{DNN}(\mathbf{W}_i, \mathbf{b}_i))$$

Often written $\min_{\mathbf{W}_i, \mathbf{b}_i} \mathcal{J}(\mathbf{W}_i, \mathbf{b}_i)$

The objective function is now a continuous function of the weights and biases

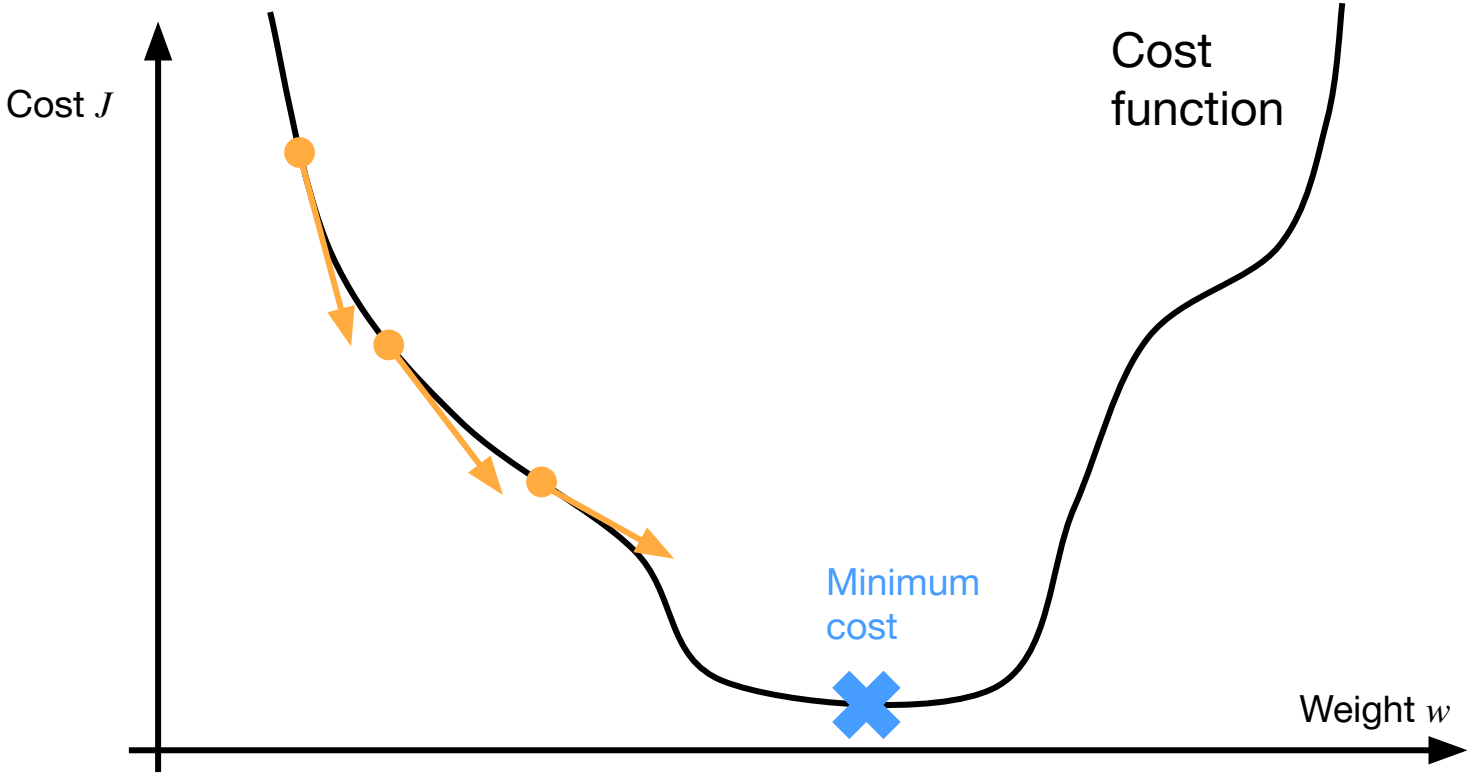
Gradient Descent

How do we solve the optimization problem?

The purpose of reformulating the problem in terms of a cost function is so that we can use gradient-based methods

1. Start with random weights
2. Compute the gradient with respect to the cost function
3. Update the weights so that the cost function decreases
4. Repeat steps 2-3

Gradient descent

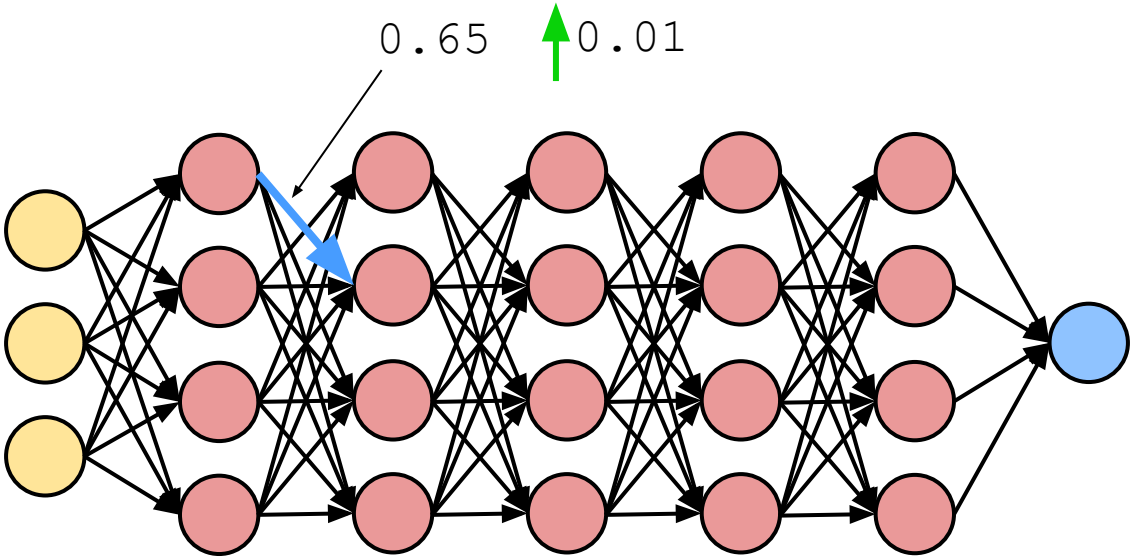


What are gradients?

1. It is a vector whose elements are the partial derivatives associated with every weight and bias in the neural network
2. The partial derivative on a weight = how many units the cost function will change by if the weight is changed by one unit
3. This assumes that the unit of change is small (technically infinitesimally small)

$$\nabla f(x, y, z) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

What are gradients?



J = 4.50

0.05

Gradient = $\frac{\partial \mathcal{J}}{\partial w_i} = \frac{-0.05}{0.01} = -5$

What are gradients?

Suppose you have a function $\mathrm{F}(x,y,z)$

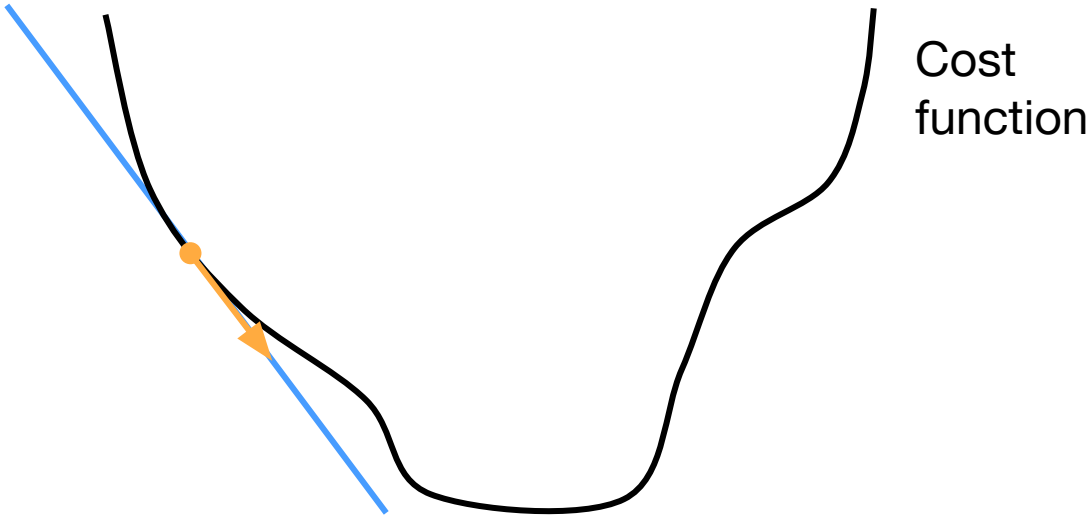
Then we have partial derivatives F_x, F_y, and F_z

$$F(x+\Delta x, y, z) \approx F(x, y, z) + F_x \Delta x$$
$$F(x, y+\Delta y, z) \approx F(x, y, z) + F_y \Delta y$$
$$F(x, y, z+\Delta z) \approx F(x, y, z) + F_z \Delta z$$

Linear approximation to the cost function

We can build a linear approximation to the function $\mathrm{F}(x,y,z)$ using the gradients

$$F(x+\Delta x, y+\Delta y, z+\Delta z) \approx F(x, y, z)+F_x\Delta x+F_y\Delta y+F_z\Delta z$$



Gradient descent

1. Go downhill according to the linear approximation of the cost function
2. But not by too much, as the approximation may not be accurate far away
3. This means updating each weight in proportion to the gradient:

$$w' = w - \alpha g$$

Learning rate

Backpropagation

How to compute gradients

- Numerically approximating gradients is too slow and computationally expensive
- The neural network is a well-defined mathematical function, so we should be able to differentiate it and calculate the derivatives using the **Chain Rule**

...like this?

$$
\begin{aligned}
&\frac{d}{dx}\sqrt{\sqrt{(x-1)(x+2)}+1} \\
&= \frac{d}{dx}\left\{[(x-1)(x+2)]^{1/2}+1\right\}^{1/2} \\
&= \frac{1}{2}\left\{[(x-1)(x+2)]^{1/2}+1\right\}^{1/2-1}\frac{d}{dx}\left\{[(x-1)(x+2)]^{1/2}+1\right\} \\
&= \frac{1}{2}\left\{[(x-1)(x+2)]^{1/2}+1\right\}^{1/2-1}\frac{1}{2}[(x-1)(x+2)]^{1/2-1}\frac{d}{dx}[(x-1)(x+2)] \\
&= \frac{1}{2}\left\{[(x-1)(x+2)]^{1/2}+1\right\}^{1/2-1}\frac{1}{2}[(x-1)(x+2)]^{1/2-1}\left\{\left[\frac{d}{dx}(x-1)\right](x+2)+(x-1)\left[\frac{d}{dx}(x+2)\right]\right\}
\end{aligned}
$$

- This also too tedious, so we use an algorithm called **backpropagation**

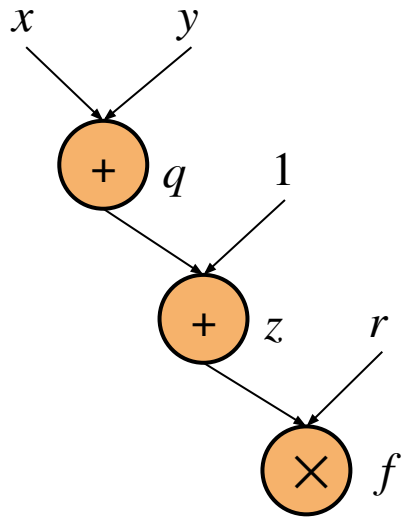
Computational graph

A way to organize mathematical expressions

Mathematical expressions

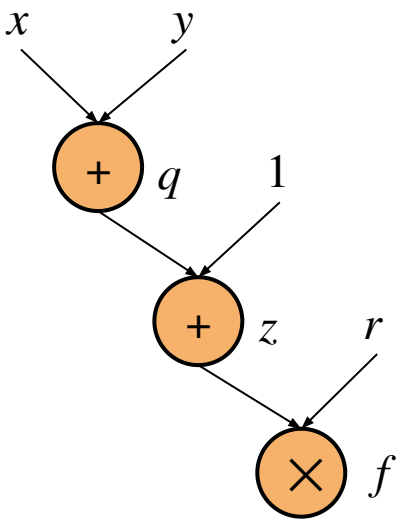
$$x + y = q$$
$$q + 1 = z$$
$$z \times r = f$$

Computational graph



Each circle denotes an **operation**

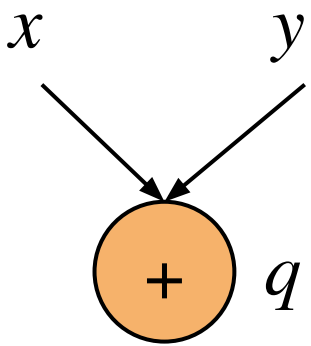
Gradients on computational graphs



Suppose we want to know $\frac{\partial f}{\partial x}$.

We can find this by chaining together derivatives at each operation.

Gradients on computational graphs

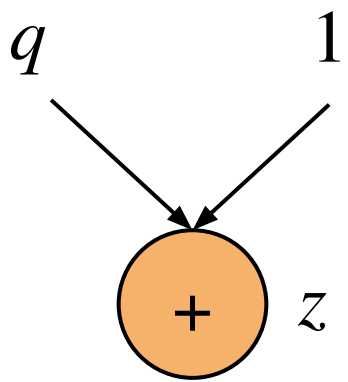


Let's take one piece of the computational graph at a time.

$$x + y = q$$

$$\frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

Gradients on computational graphs



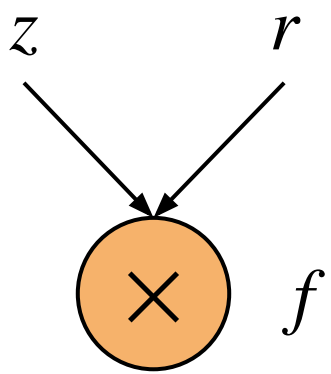
$$q + 1 = z$$

$$\frac{dz}{dq} = 1$$

Chain Rule

$$\frac{\partial z}{\partial x} = \frac{dz}{dq}\frac{\partial q}{\partial x} = 1$$

Gradients on computational graphs



$$z \times r = f$$

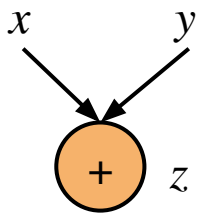
$$\frac{\partial f}{\partial z} = r \quad \frac{\partial f}{\partial r} = z$$

Chain Rule

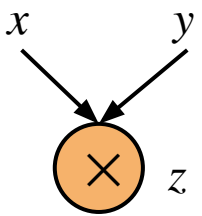
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z}\frac{dz}{dq}\frac{\partial q}{\partial x} = r$$

In backprop, gradients are highly local

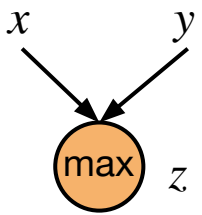
Three common operations in neural networks:



$$\frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$



$$\frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x$$

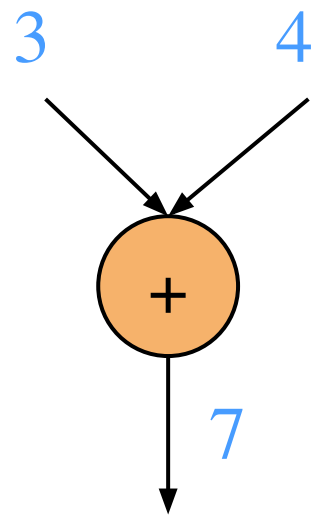


$$\frac{\partial f}{\partial x} = \mathbf{1}\{x \geq y\}$$

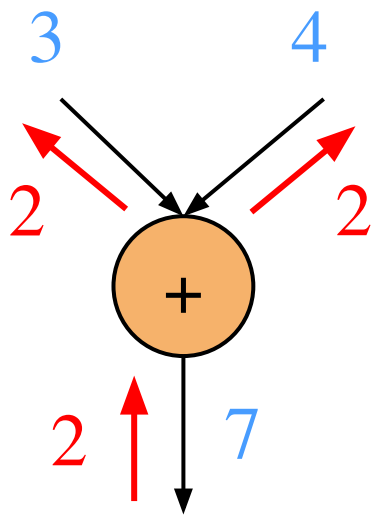
$$\frac{\partial f}{\partial y} = \mathbf{1}\{y \geq x\}$$

A small example with numbers

Forward pass

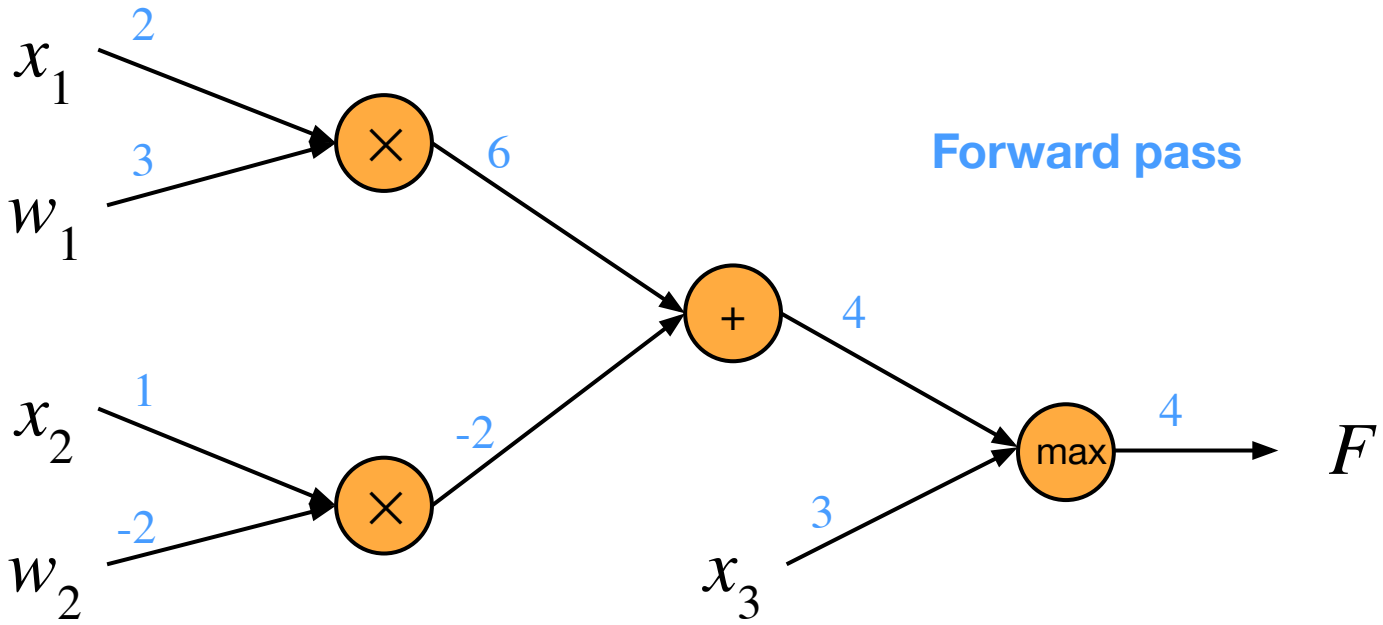


Backward pass



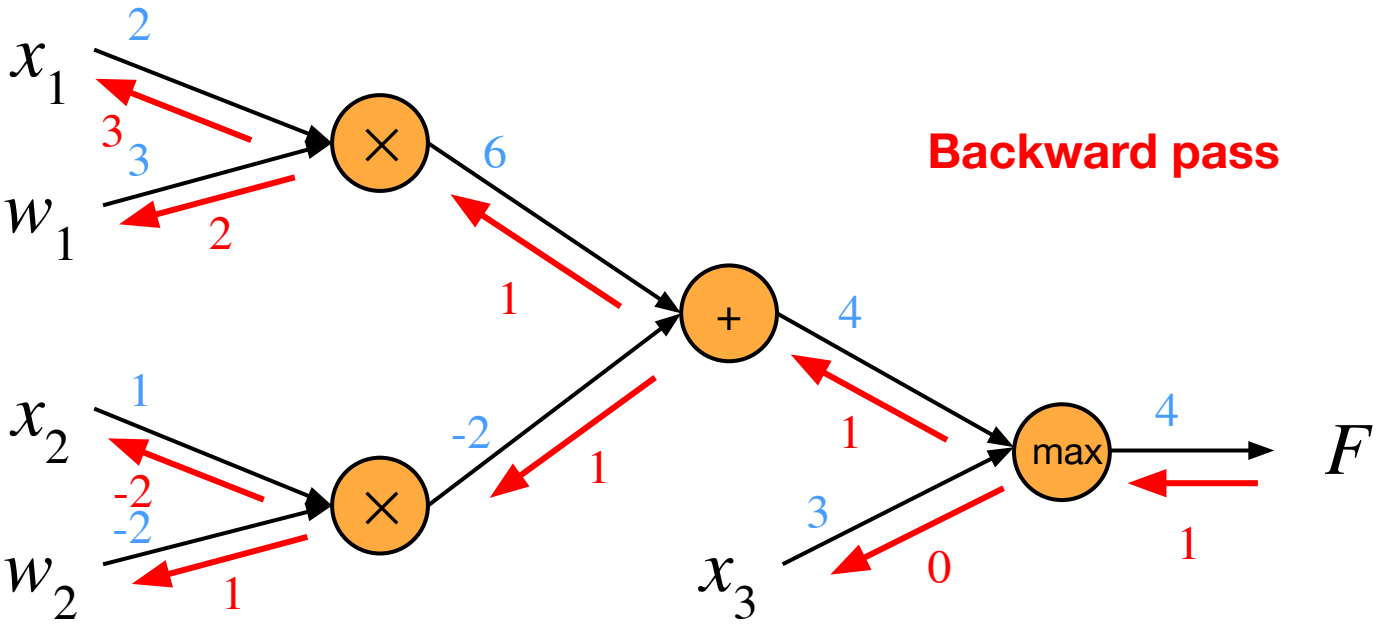
A larger example with numbers

$$F(x_1, x_2, x_3) = \max(w_1 x_1 + w_2 x_2, x_3)$$



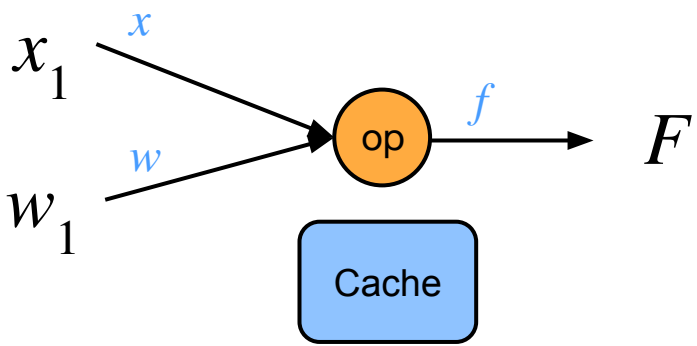
A larger example with numbers

$$F(x_1, x_2, x_3) = \max(w_1 x_1 + w_2 x_2, x_3)$$

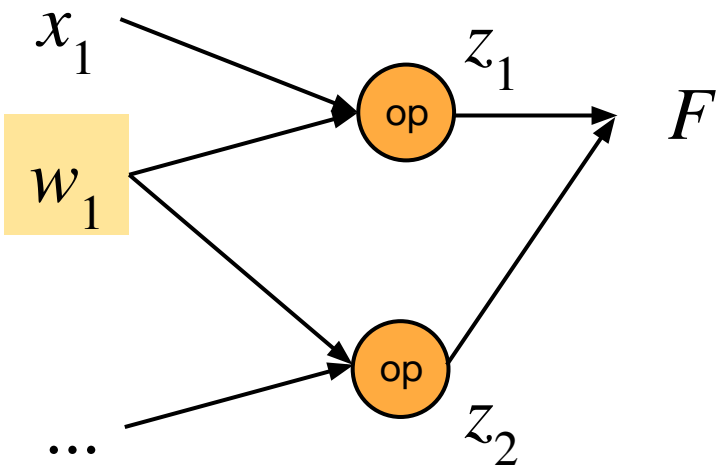


Note 1: Cache forward pass variables

During the backward pass, variable values from the forward pass are often needed



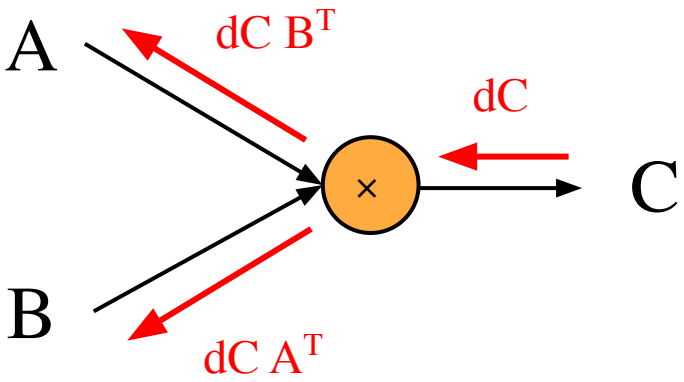
Note 2: Gradients add when a variable appears multiple times



$$\frac{\partial F}{\partial w_1} = \frac{\partial F}{\partial z_1}\frac{\partial z_1}{\partial w_1} + \frac{\partial F}{\partial z_2}\frac{\partial z_2}{\partial w_1}$$

Multivariable Chain Rule

Note 3: Backprop through matrix multiplication



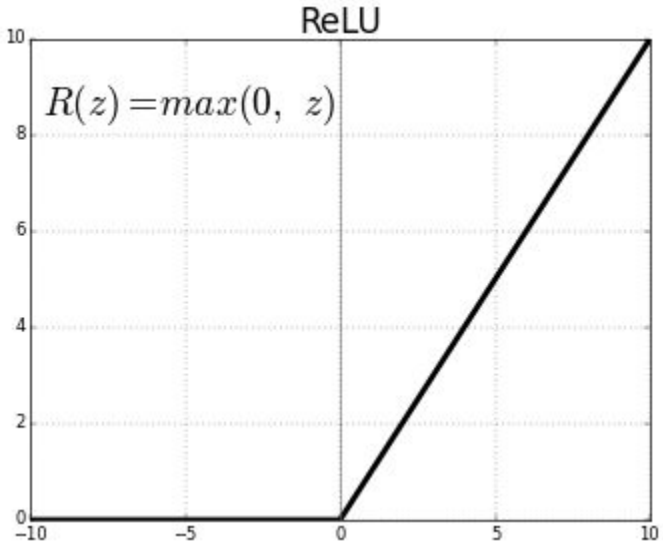
$$AB = C$$

$$dA = dC \times B^\top$$

$$dB = dC \times A^\top$$

Check matrix dimensions to verify

Exercise: **What happens when the gradient passes through the ReLU function?**



To answer, please type into Zoom chat

Gradient descent

$$w' = w - \alpha g$$

The cost function is the average of the loss function over all training examples.

This means we have to forward and backward pass through all training examples before we take one step of gradient descent.

Stochastic gradient descent

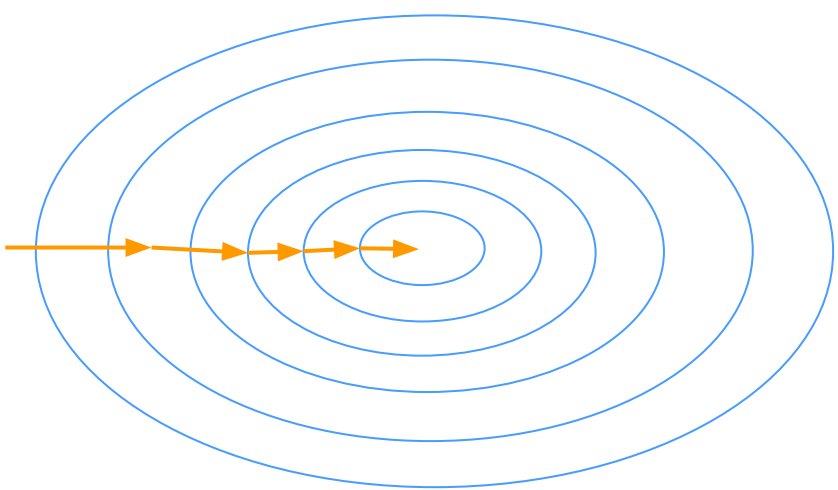
In SGD, we estimate the gradient by evaluating it over only a small subset of the training examples.

The size of this subsample is the **batch size**, which becomes another hyperparameter.

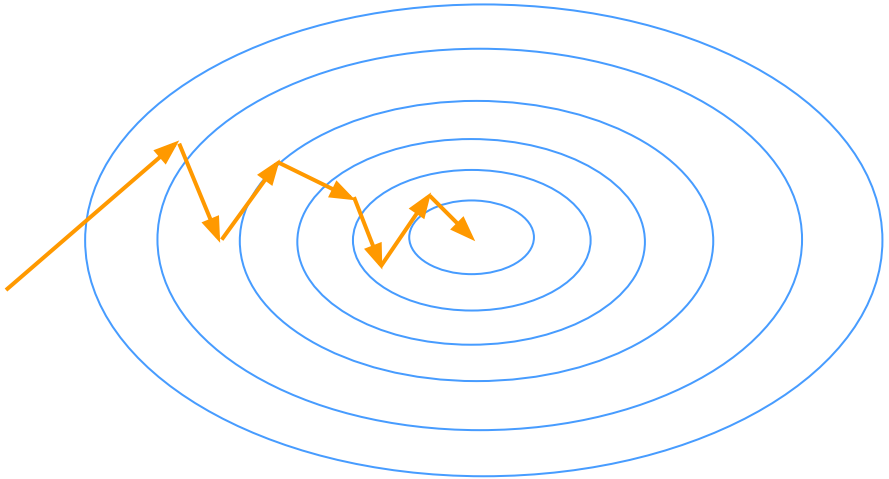
We use a different subsample for each gradient step, so as to go through all the training examples. Going through the entire training set once is called one **epoch** of training.

Stochastic gradient descent

Gradient descent



Stochastic gradient descent



Training, validation, and test sets

To evaluate the performance of the model, set aside a portion of the dataset.

Training set: used to train the weights of the neural network.

Validation set: used to tune the network hyperparameters and perform architecture search.

Test set: reserved until the end, used only once to evaluate the final weights and hyperparameters of the model.

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Poll: Data splits

Go to **PollEv.com/dlworkshop2020**

Suppose I want to try training my neural network with different learning rates {0.1, 0.01, 0.001, 0.0001}. Which split should I use to evaluate which learning rate is best?

A. Train

B. Validation

C. Test

Training loop

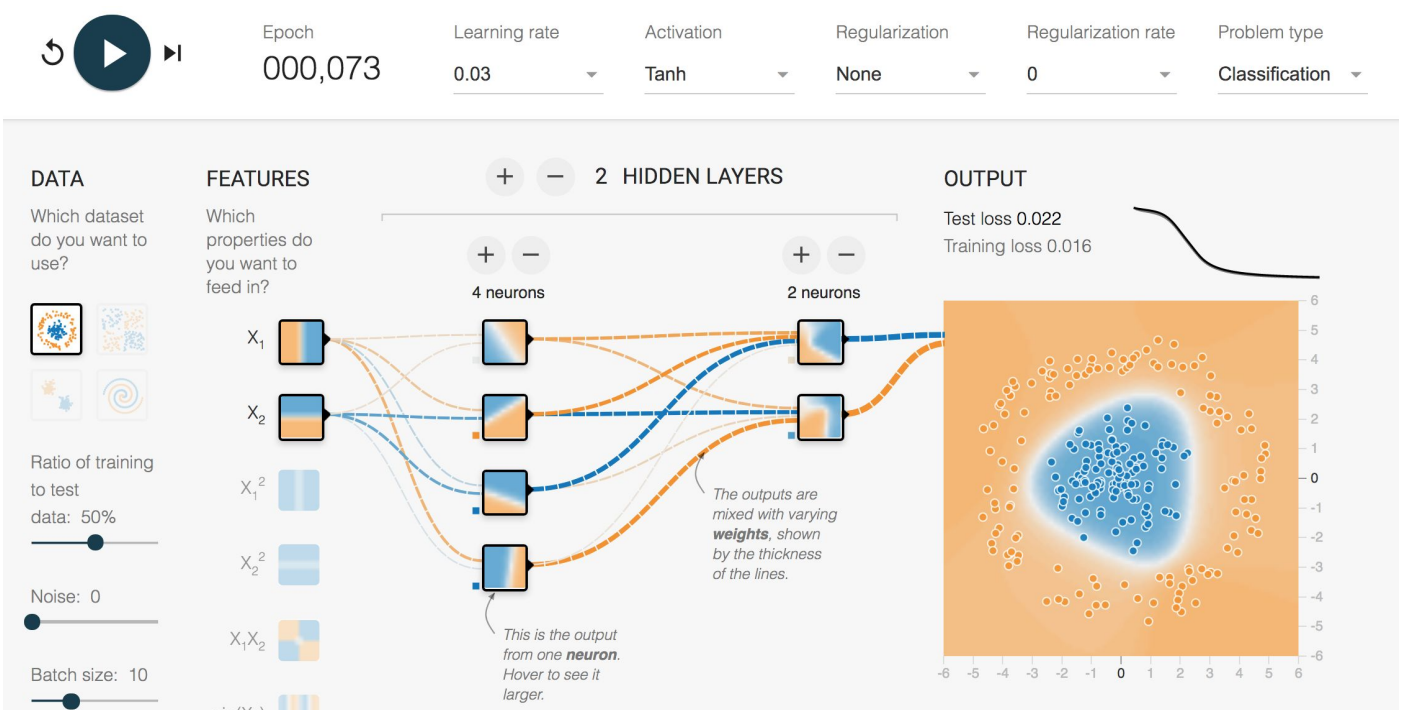
1. Sample a batch of data
2. Forward pass through all examples in the batch
3. Compute loss
4. Backpropagate through all examples
5. Compute final gradient on batch and update weights
6. Repeat

Workflow

1. Split your data into train/val/test
2. Choose architecture
3. Choose hyperparameters: learning rate, batch size, etc
4. Train neural network until convergence, evaluating on validation set each epoch
5. Check if training is overfitting or underfitting
6. Modify architecture and hyperparameters as needed, repeat

TensorFlow playground

https://playground.tensorflow.org



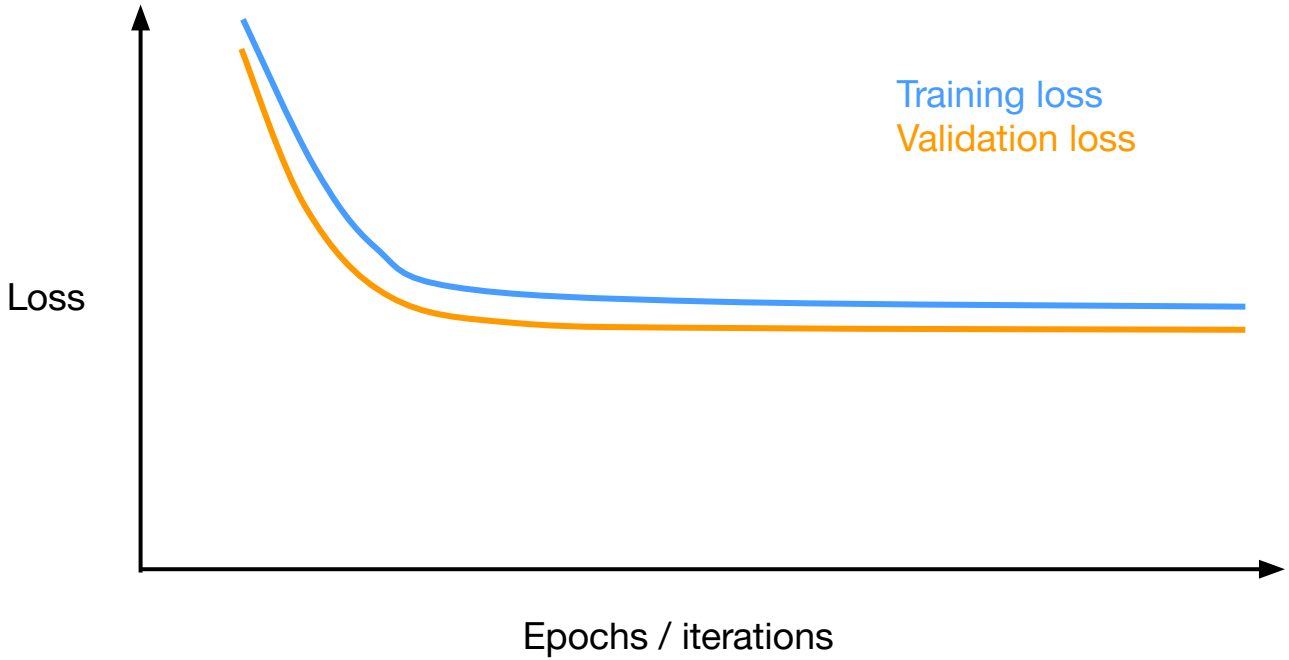
Overfitting and Underfitting

Overfitting and underfitting

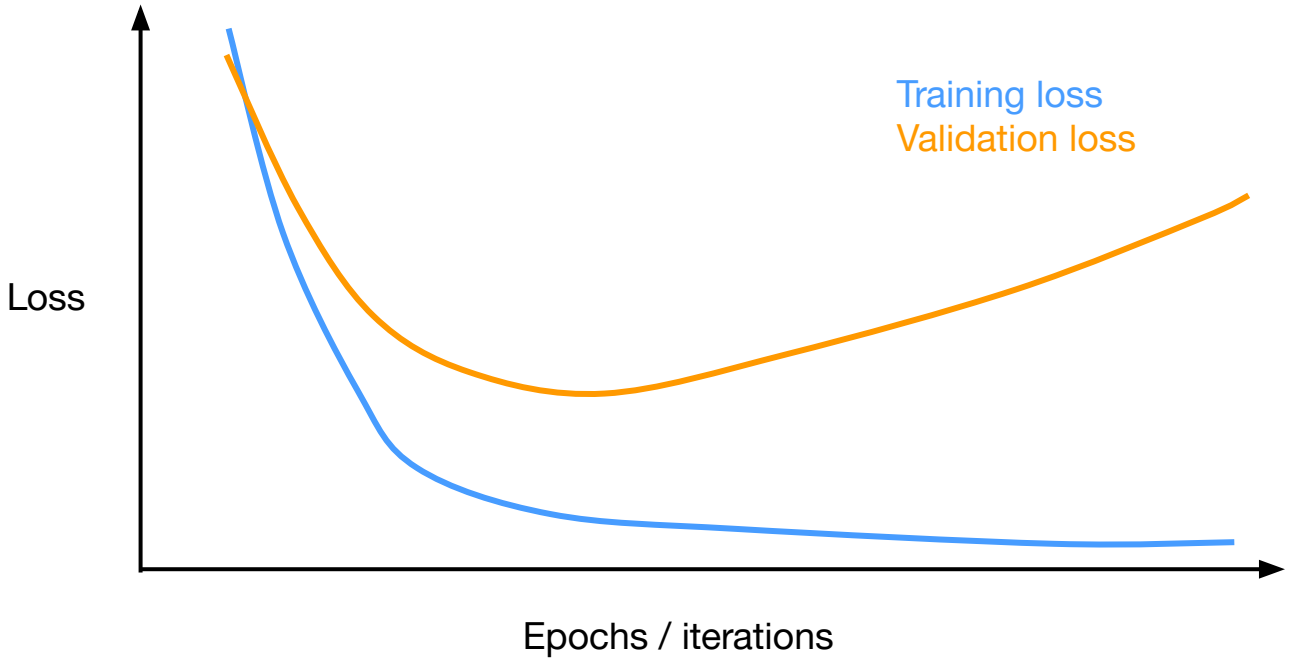
- **Underfitting:** when your model is not complex enough to capture the structure in the data. The capacity of the model is insufficient. Both training loss and validation loss are high.

- **Overfitting:** when your model has so much capacity that it memorizes your training data. It usually indicates insufficient training data relative to your model complexity. Training loss is low, but validation loss is high.

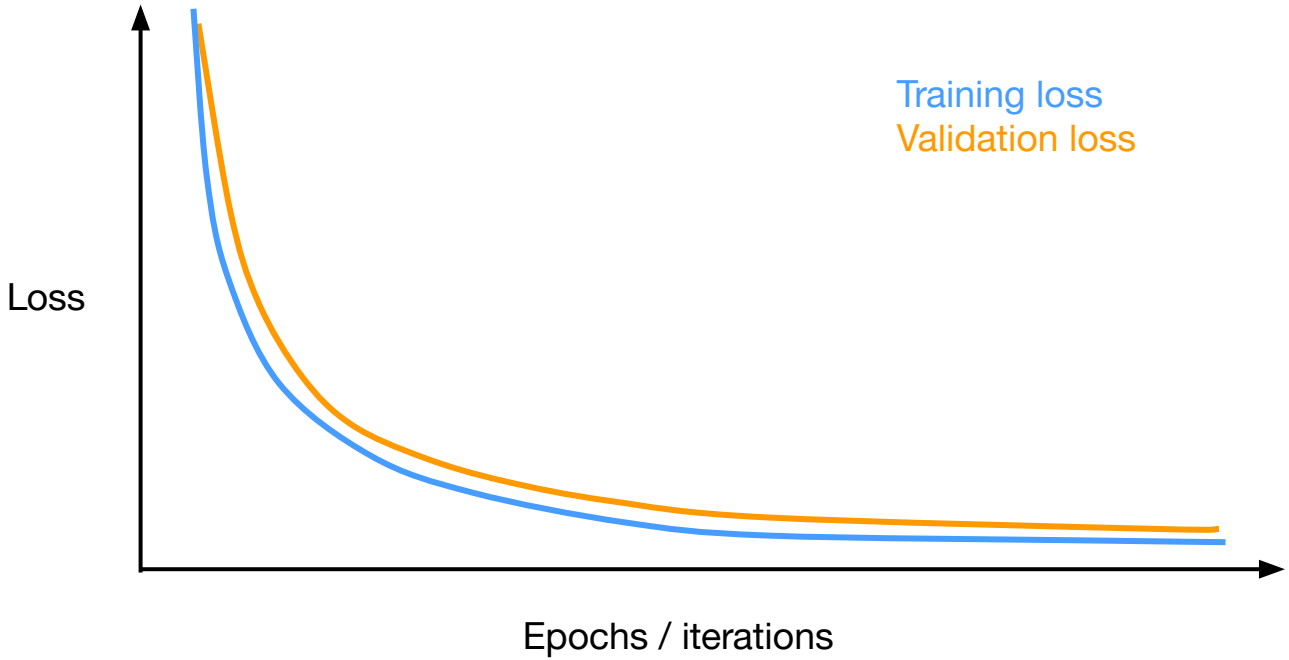
Underfitting



Overfitting

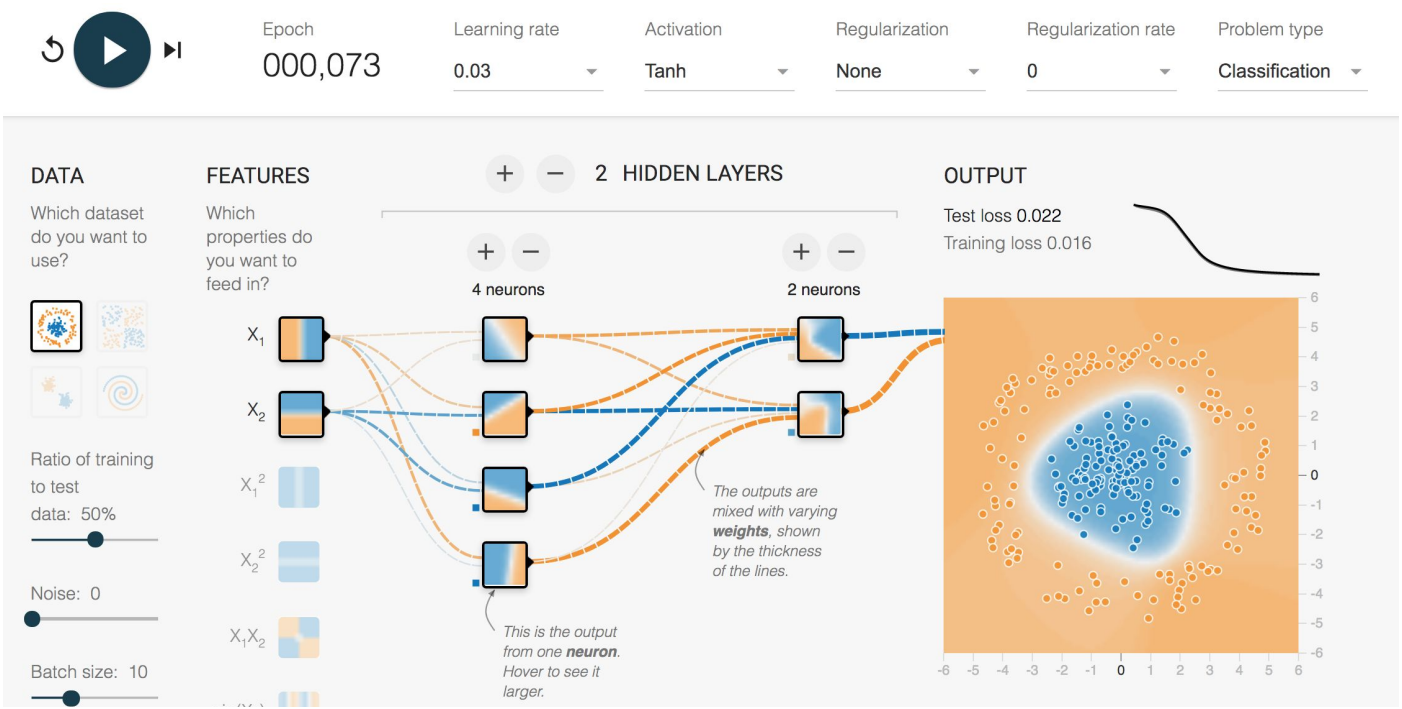


Good training



Exercise: Underfit a dataset in TF Playground

https://playground.tensorflow.org



Why do we need a test set?

Since we use the validation set to choose architecture and hyperparameters, it is possible to overfit to the validation set.

The test data set is used to give us an unbiased estimate of out-of-sample performance.

This underscores the importance of using the test set *only once*.

Regularization

One way to combat overfitting is to add a regularization term to the cost function.

This constrains the capacity of the model so that it has a lower chance of overfitting.

$$J(W, b) + \lambda \boxed{R(W, b)}$$

$$\| W \|^2$$

Final tips and tricks

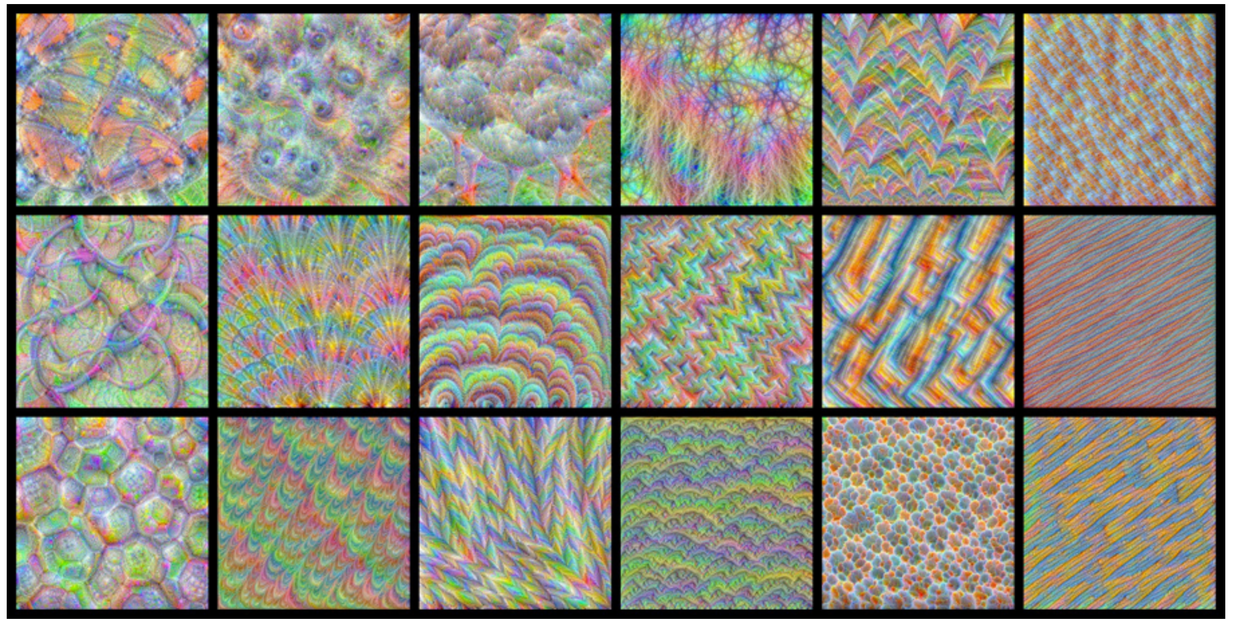
1. Try other machine learning methods first
2. If you have trouble with training, first try to overfit a small subset of the dataset
3. If loss does not decrease smoothly, try to increase the number of hidden neurons in each layer
4. Play around with the learning rate; try to increase it as well as decrease it over time
5. Check online for others who have tried to solve your problem before attempting your own solution

ICME Summer Workshops 2020

Introduction to Deep Learning

Session 3: 2:00—3:15 PM
Instructor: Sherrie Wang

icme-workshops.github.io/deep-learning



https://towardsdatascience.com/how-to-visualize-convolutional-features-in-40-lines-of-code-70b7d87b0030

Workshop Schedule

Session 1 (9:00—10:30 AM)
- Introduction
- Current state-of-the-art in deep learning
- Math review
- Fully connected neural networks

Session 2 (10:45—12:00 PM)
- Loss functions
- Gradient descent
- Backpropagation
- Overfitting and underfitting

Lunch (12:00—2:00 PM)

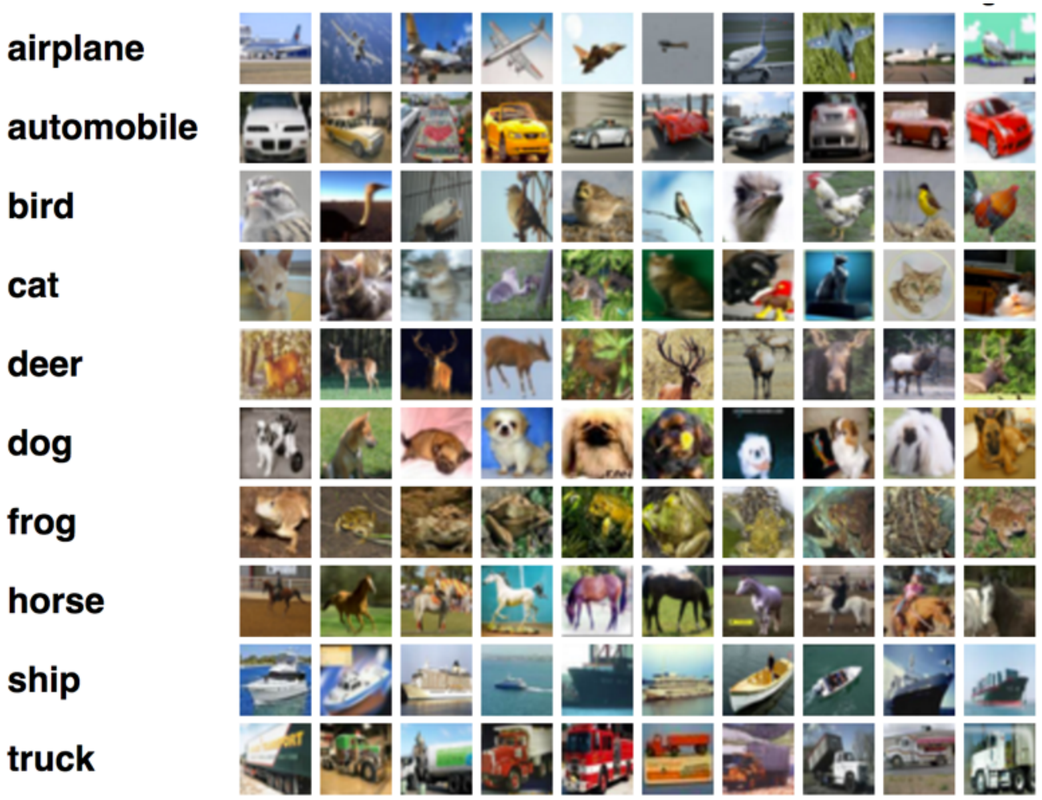
Session 3 (2:00—3:15 PM)
- Convolutional neural networks
- Recurrent neural networks
- Other architectures
- Deep learning libraries
- Hands-on coding session in Tensorflow

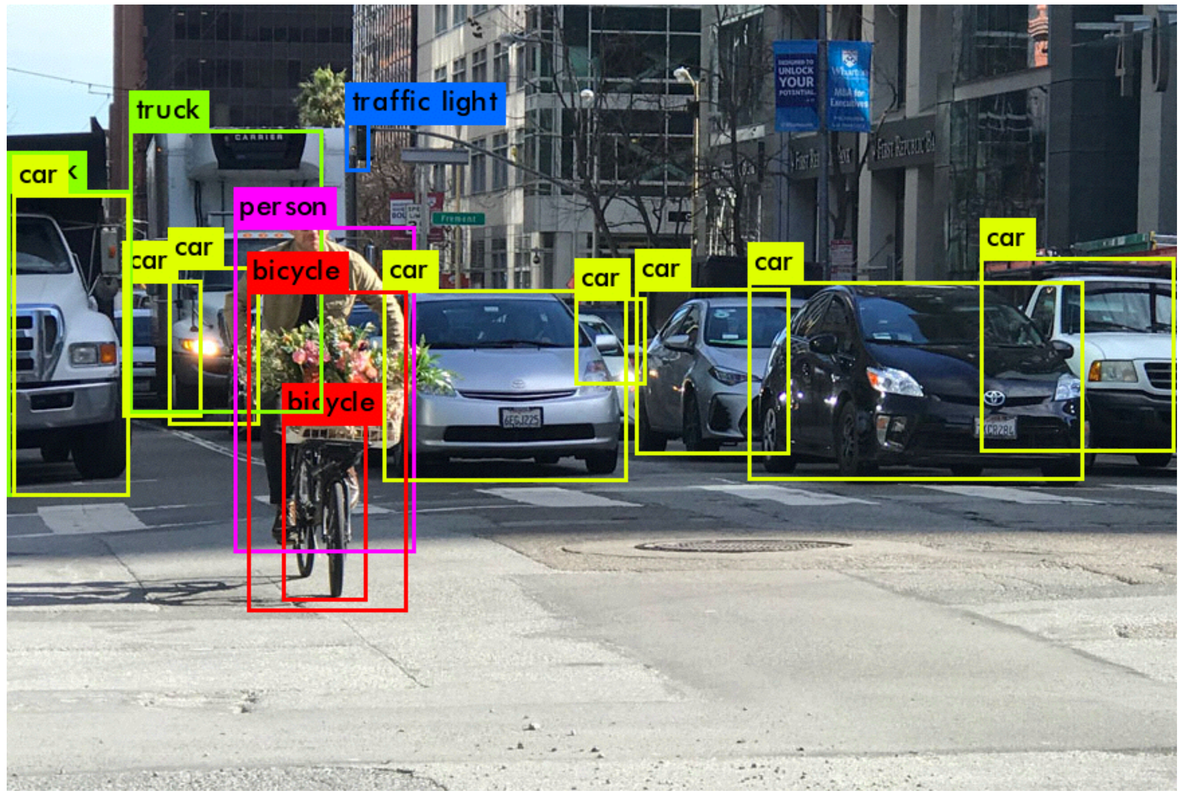
Session 4 (3:30—4:45 PM)
- Hands-on coding session in Keras
- Hands-on coding session on transfer learning
- Failures of deep learning

Convolutional Neural Networks

Convolutional neural networks

- Neural networks have been very successful at image classification and object detection





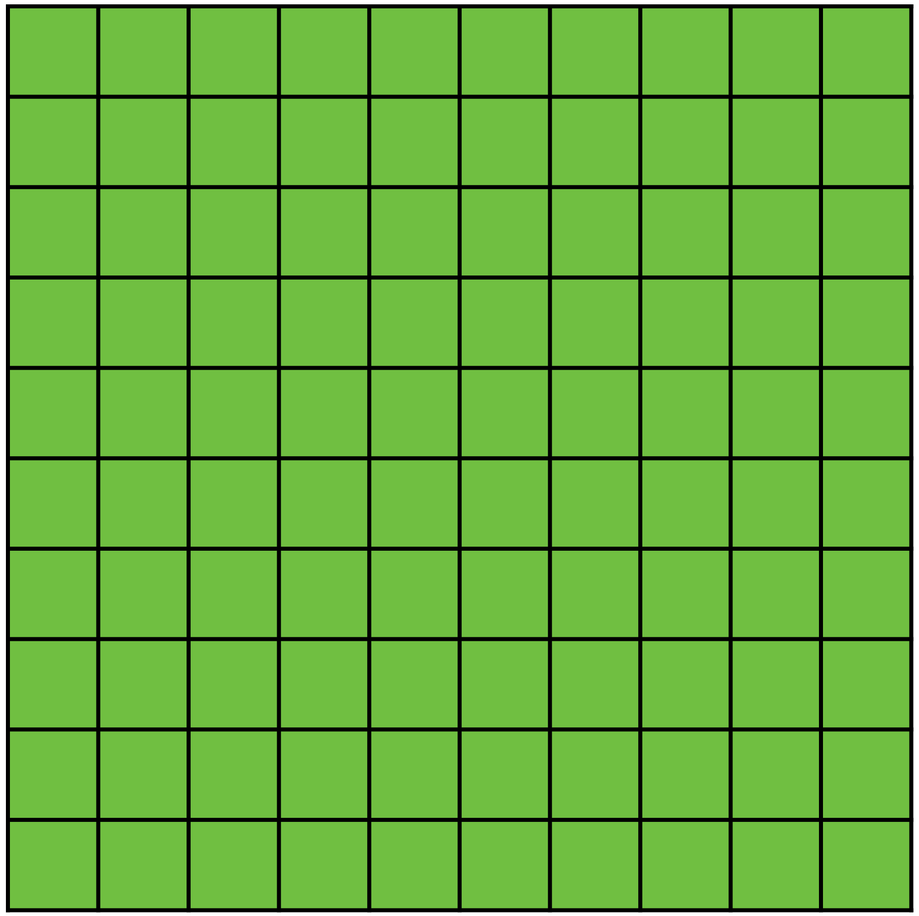
Convolutional layers

- Convolutional layers are a special type of layer for image inputs
- Convolutions replace matrix multiplication with the convolution operation

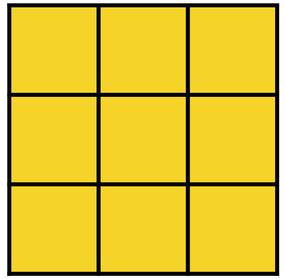
$$g(\mathbf{W}\mathbf{x} + \mathbf{b}) \longrightarrow g(\mathbf{f} * \mathbf{x} + \mathbf{b})$$

Convolutions

Image

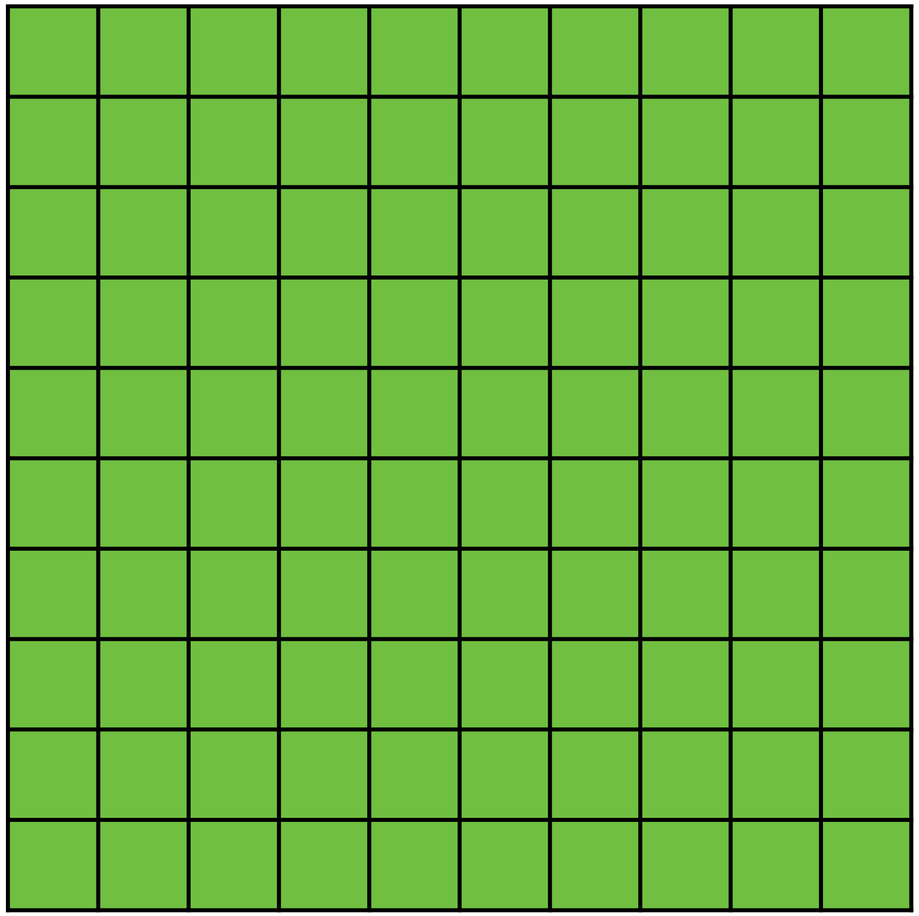


Filter



Convolutions

Image



Filter

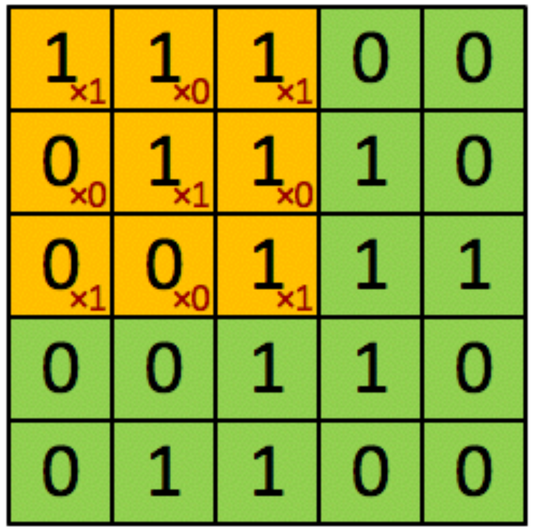
1	4	0
0	2	1
3	0	1

Learnable weights

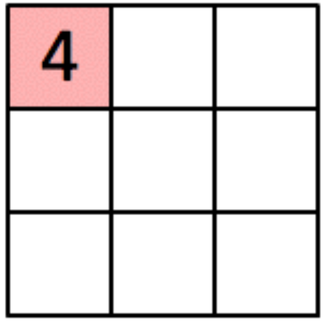
Convolutions

An example on an image with 1 channel

Notice that the dimensions of the output are smaller than the dimensions of the image by 2 pixels



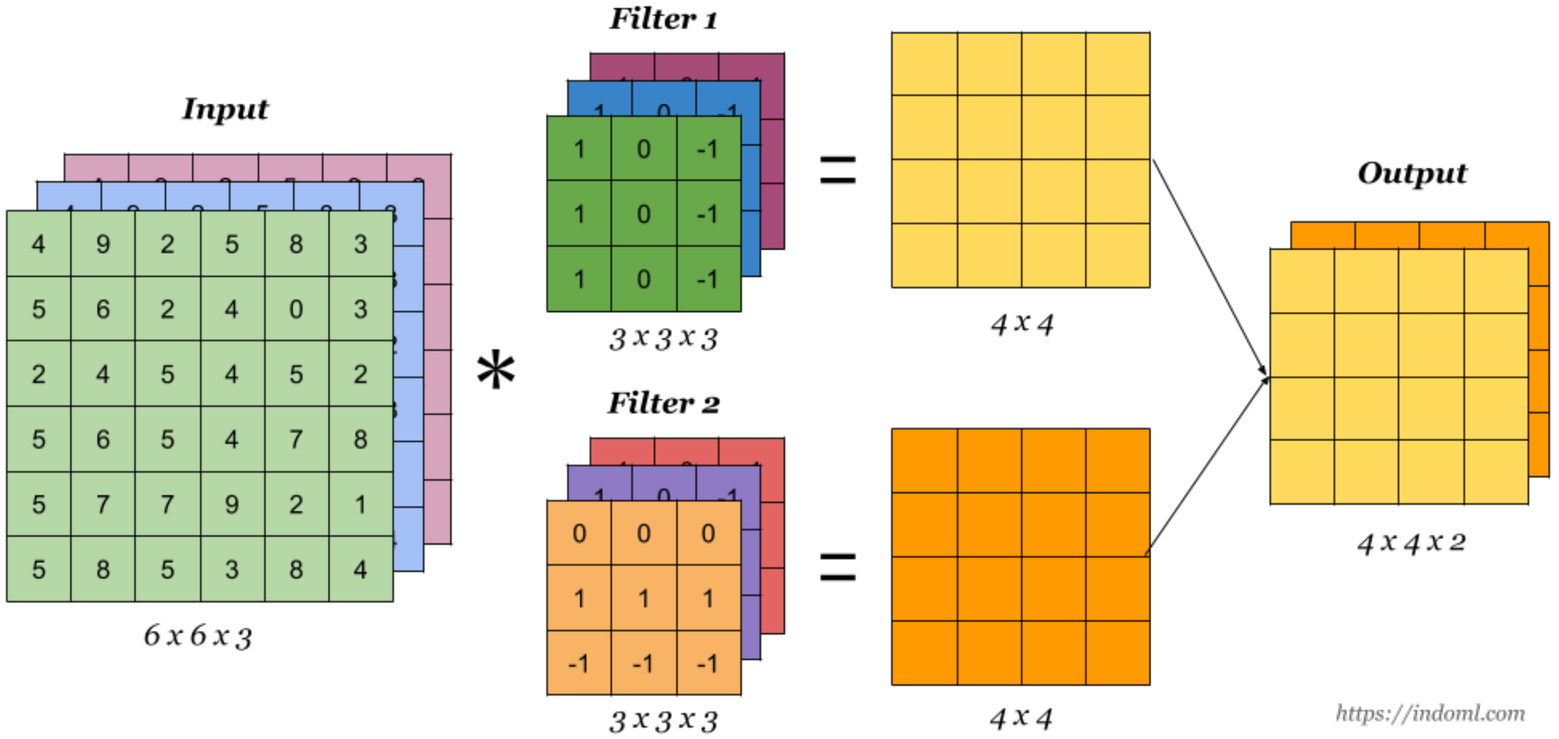
Image



Convolved Feature

Convolutions: 3 channels

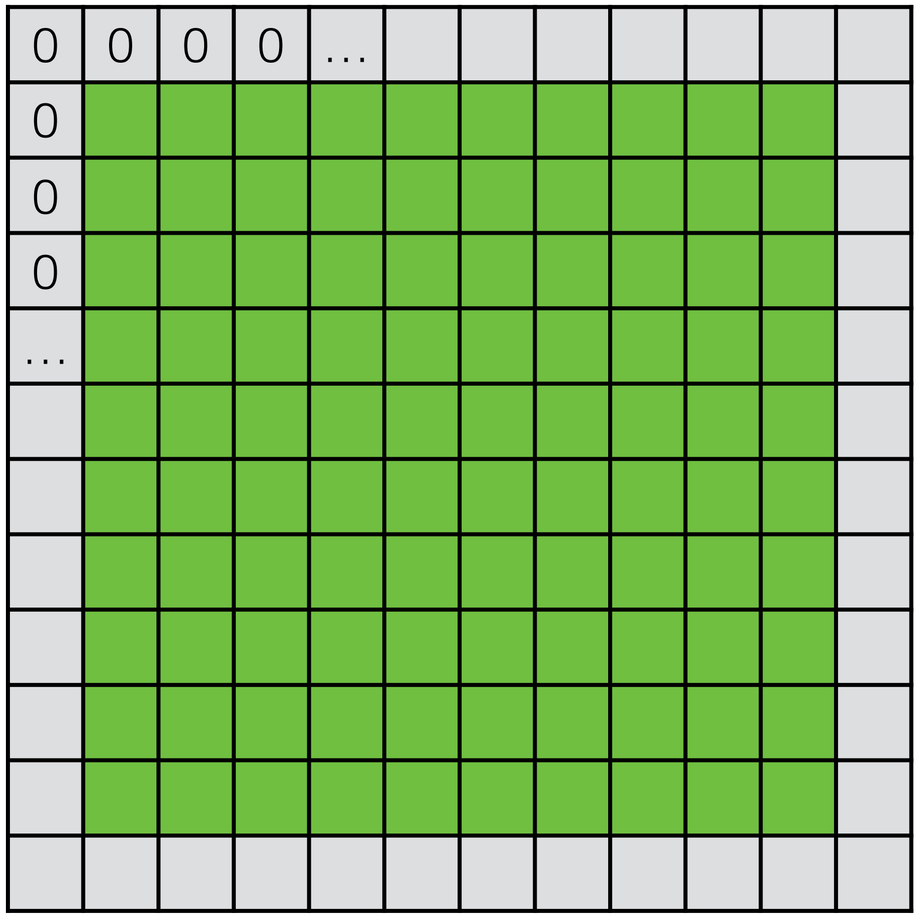
Most images are RGB



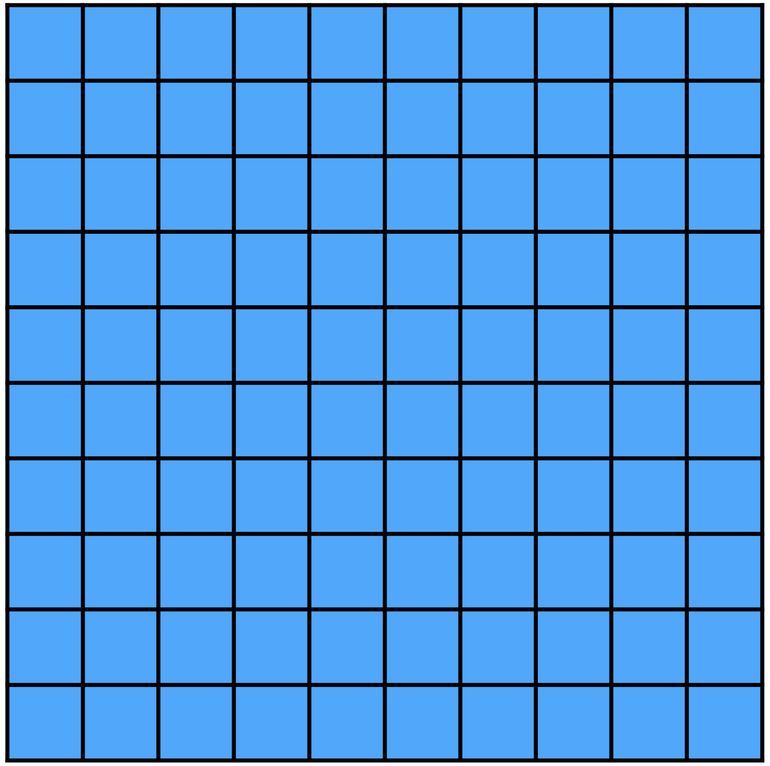
Padding

Zero-padding allows us to control the spatial size of the output — can be a hyperparameter

Image

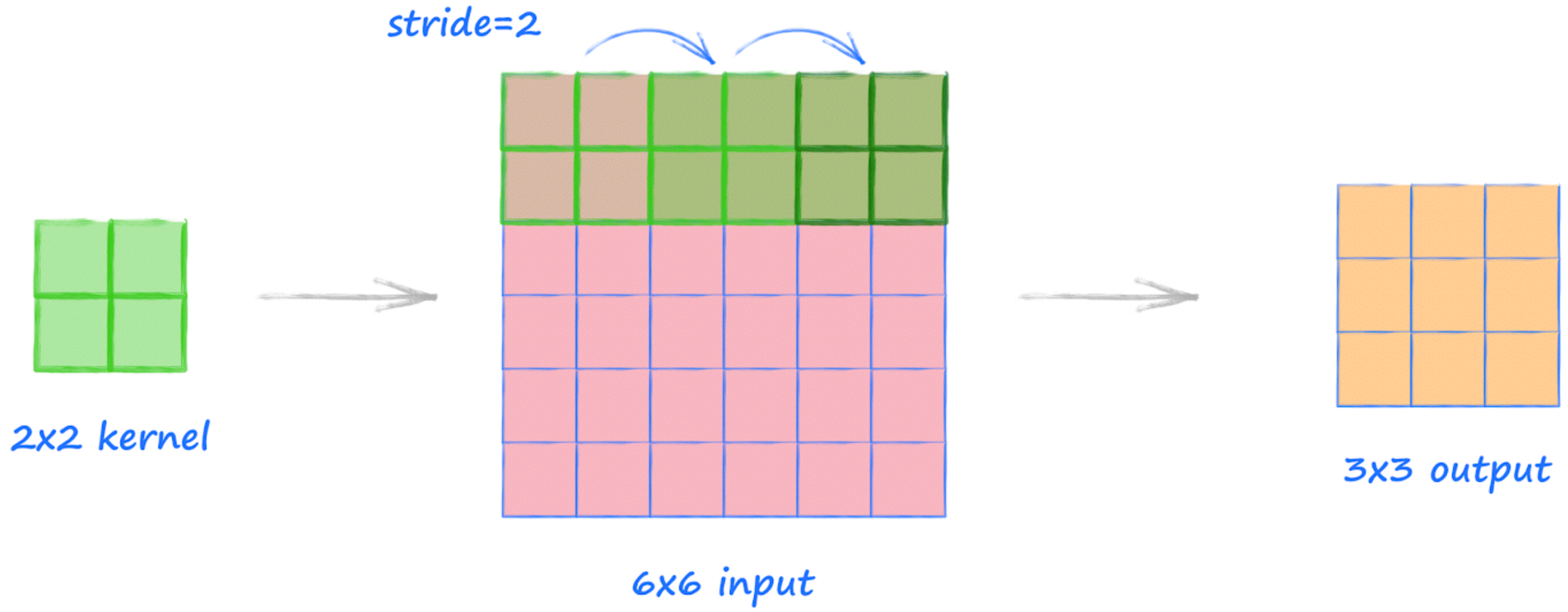


Output



Stride

Stride refers to how many pixels we shift the filter when convolving



http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html

Filter size

Filters can be of any spatial dimension smaller than or equal to the image size, but must match the number of channels in the image

The filter size is also known as the **receptive field** of the neuron

3x3

0.91	0.32	0.07
0.73	0.26	0.81
0.53	0.68	0.14

5x5

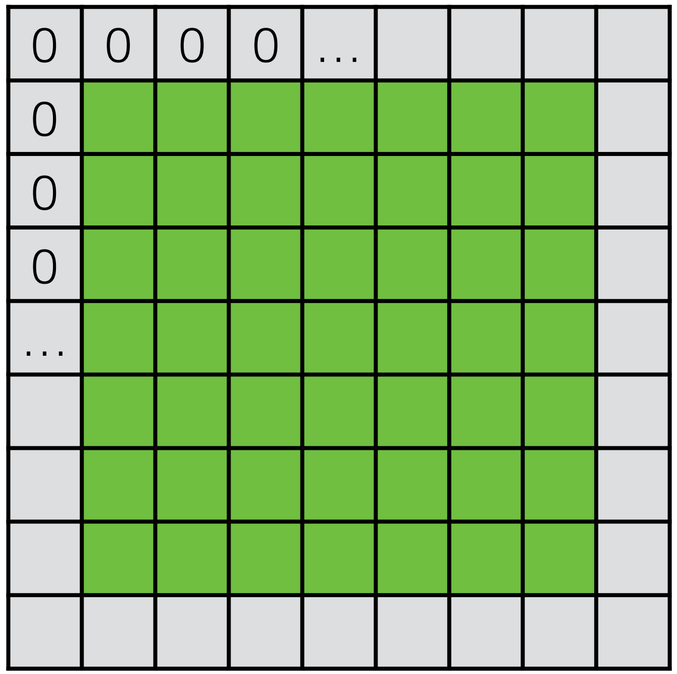
0.27	0.64	0.44	0.84	0.29
0.28	0.06	0.89	0.99	0.33
0.64	0.67	0.08	0.38	0.03
0.04	0.31	0.16	0.57	0.08
0.87	0.85	0.97	0.71	0.96

Bigger filters have wider receptive fields — but this can also be accomplished with a deeper CNN

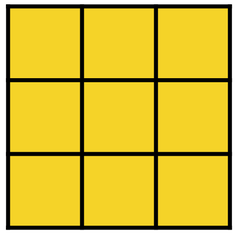
Poll: Output dimensions

Given an input image of dimension 7x7x1, one filter of size 3x3x1, 1-pixel thick zero-padding, and a stride of 3, what is the output size?

Image



Filter

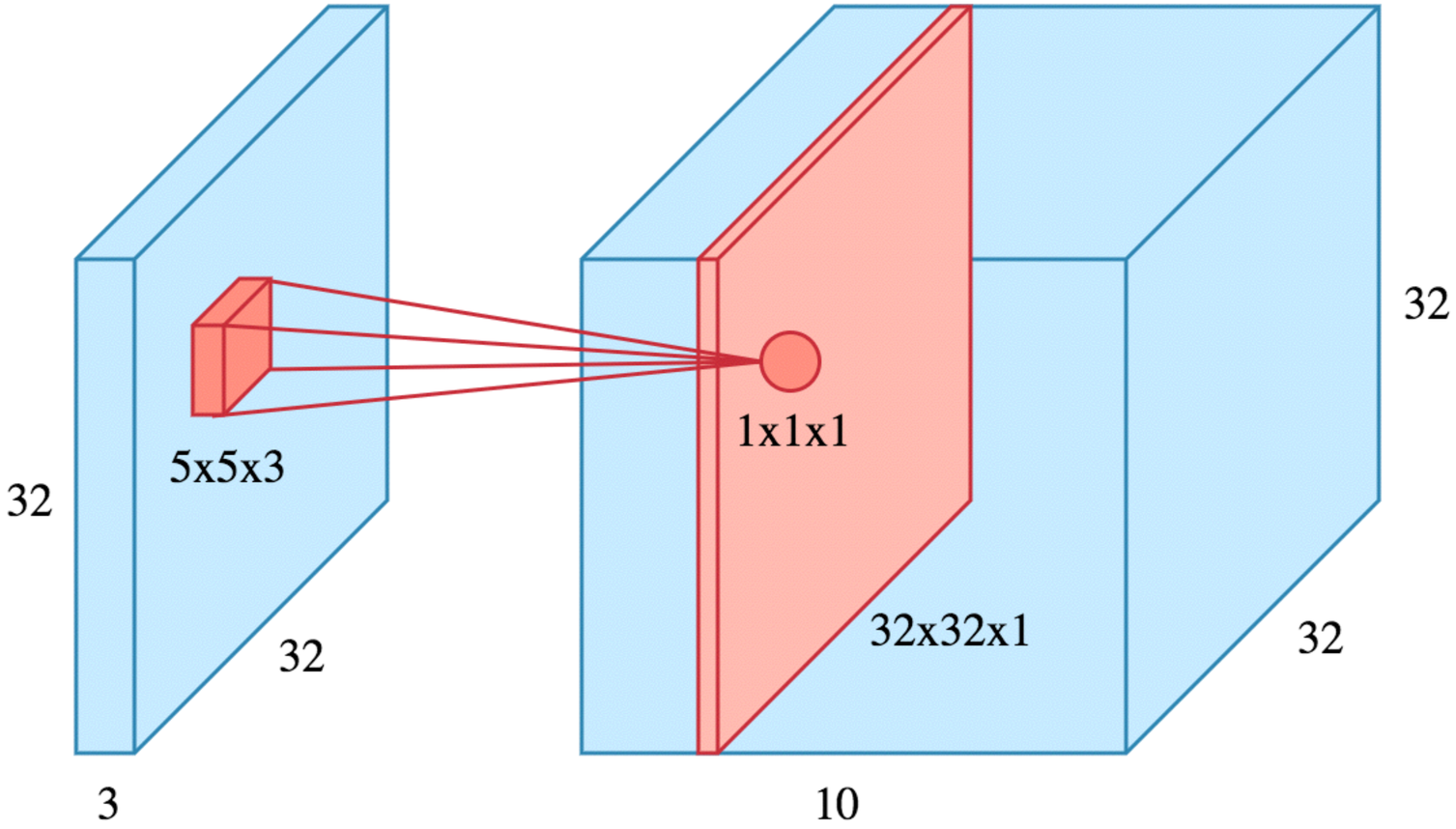


A. 3x3x1
B. 4x4x1
C. 5x5x1
D. 7x7x1

Go to PollEv.com/dlworkshop2020

Depth

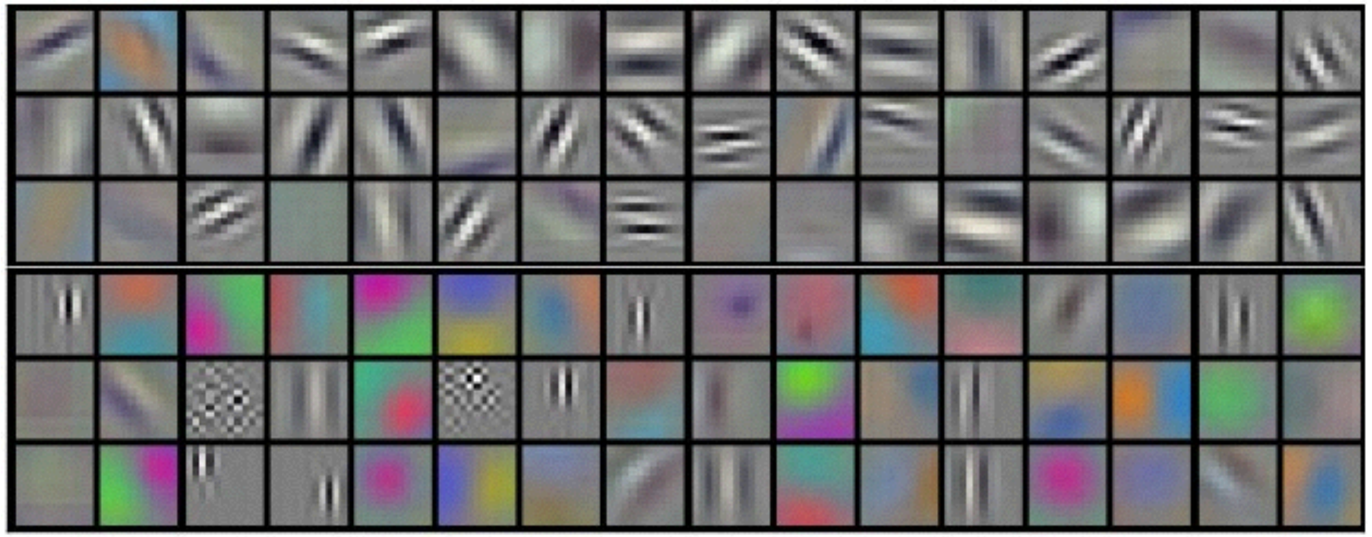
The number of filters used — more filters mean more learnable features



https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

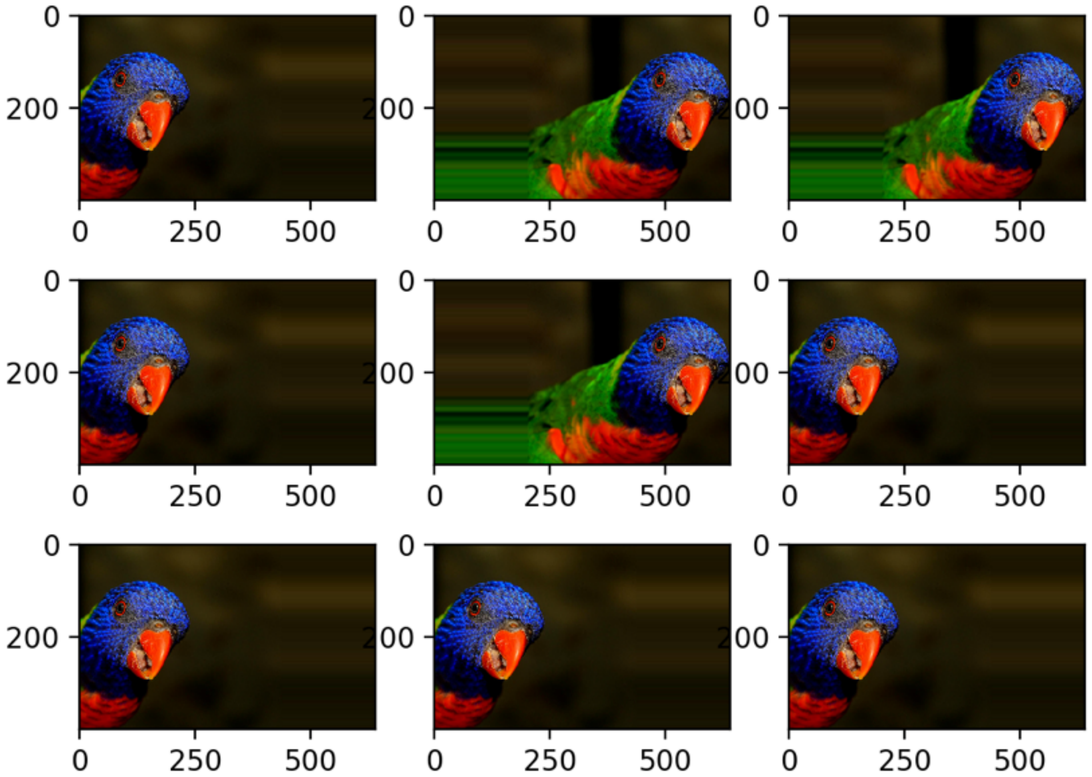
Example learned filters

- 96 filters learned by the AlexNet architecture trained on ImageNet
- Each filter is of dimension 11x11



Why use convolutions?

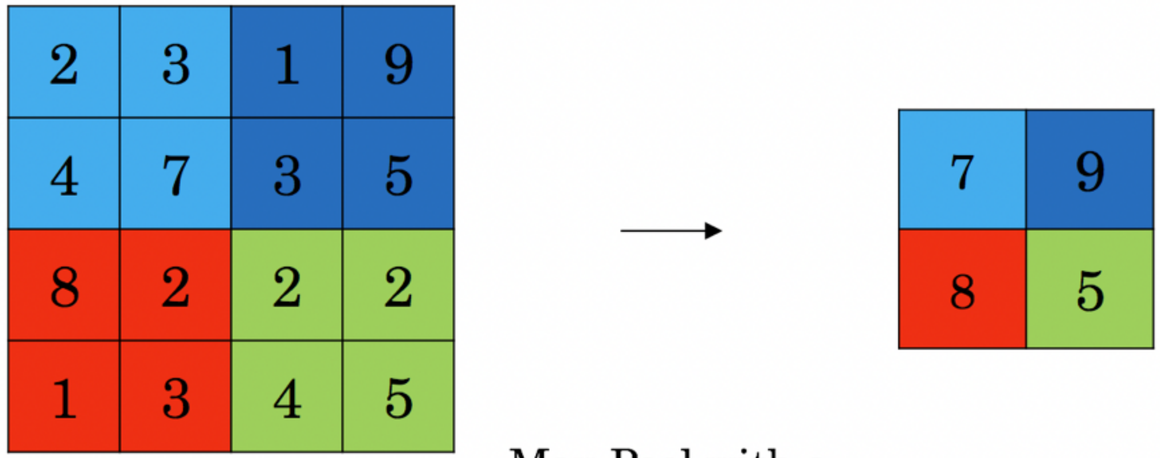
- Features in images should be invariant to translations
- Weight sharing — reduces the number of parameters to learn



No matter where the parrot is in the image, the image still contains a parrot

Pooling layers

It is common to insert pooling layers between successive Conv layers.
Pooling reduces the spatial size of the representation.
The most common pooling layer:

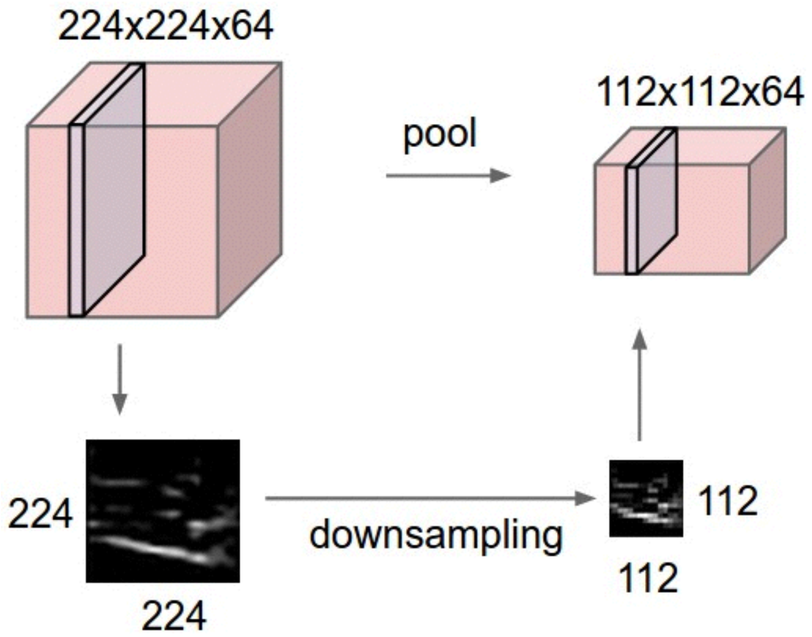


Max-Pool with a 2 by 2 filter and stride 2.

Andrew Ng

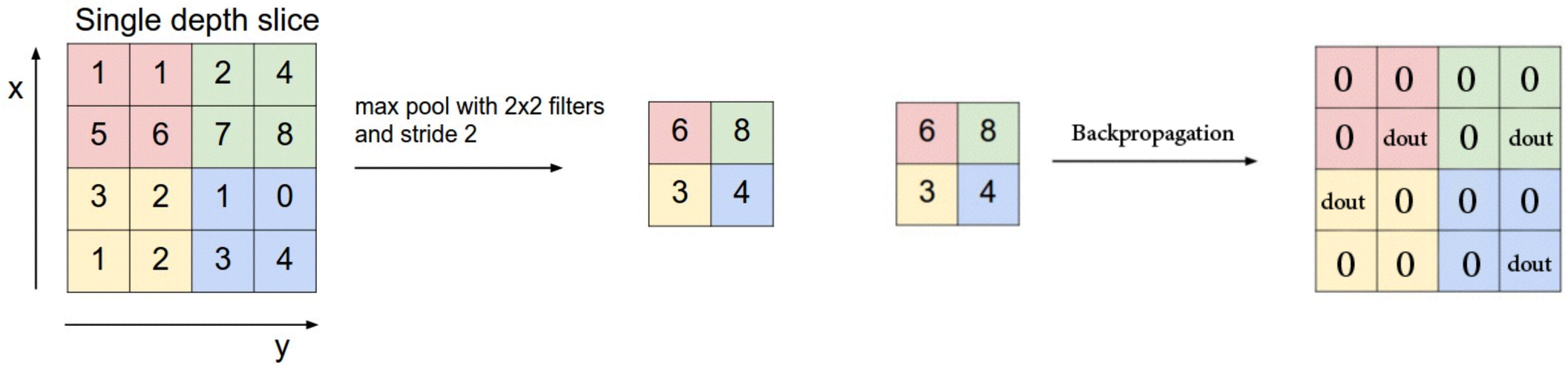
Pooling layers

Pooling layers contain no trainable parameters.
It is uncommon to use padding in pooling layers.



Recall backpropagation

Backprop through max-pooling layers routes the gradient through the index of the max activation



https://cs231n.github.io/convolutional-networks/

Stacking layers

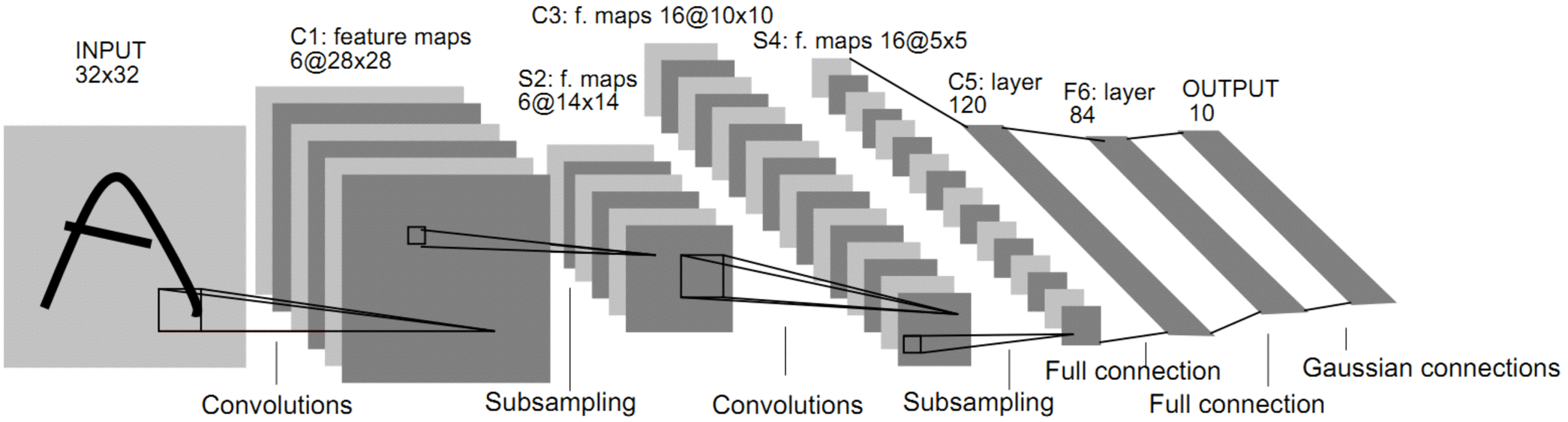
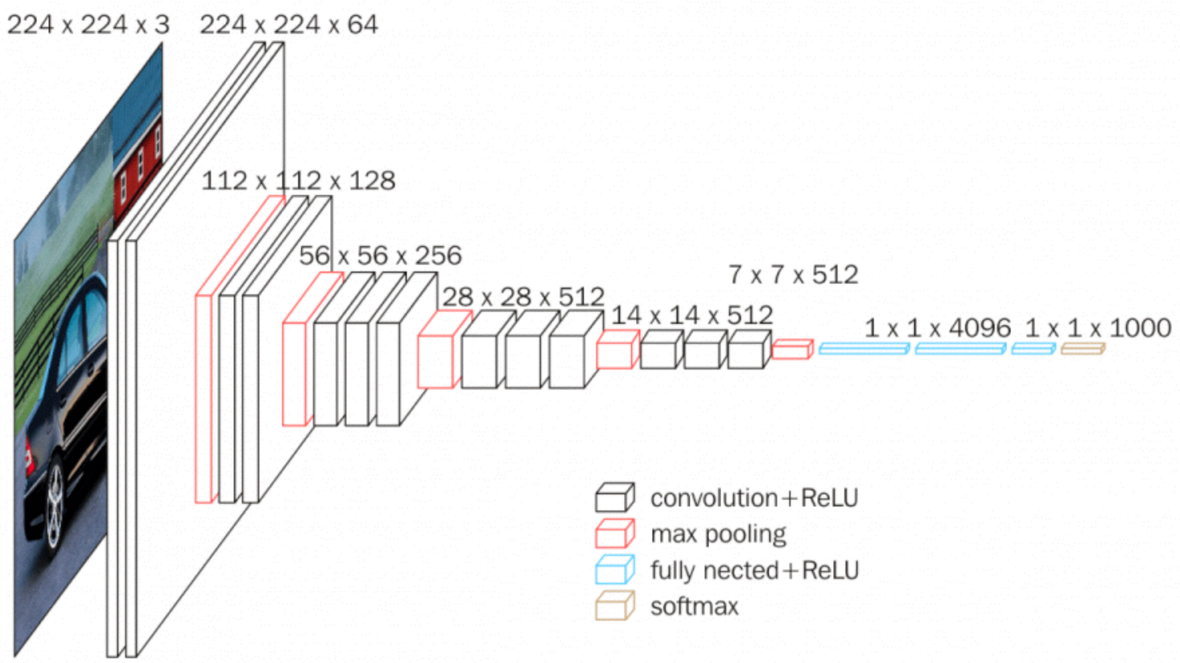


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Stacking layers

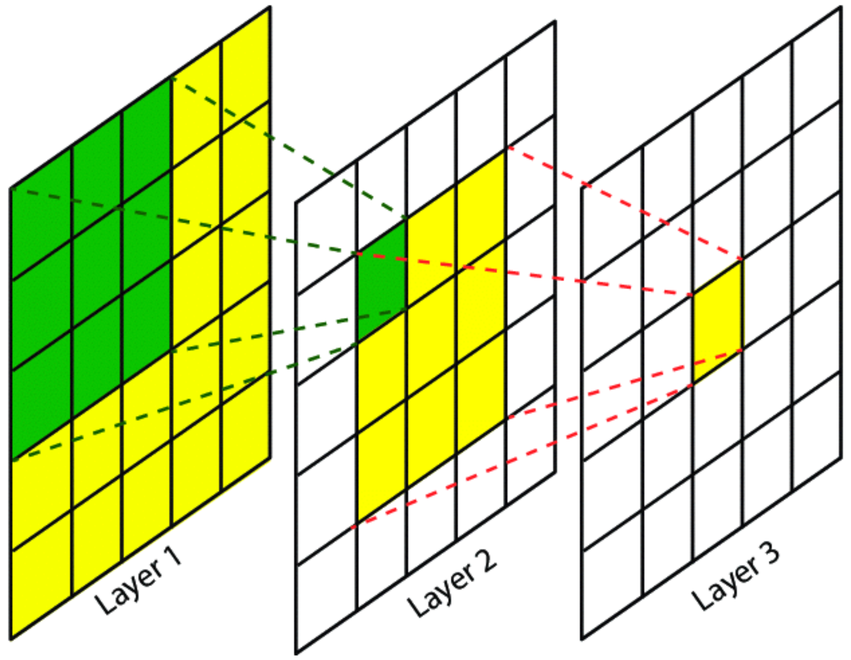
VGG-16



https://neurohive.io/en/popular-networks/vgg16/

Growing the receptive field

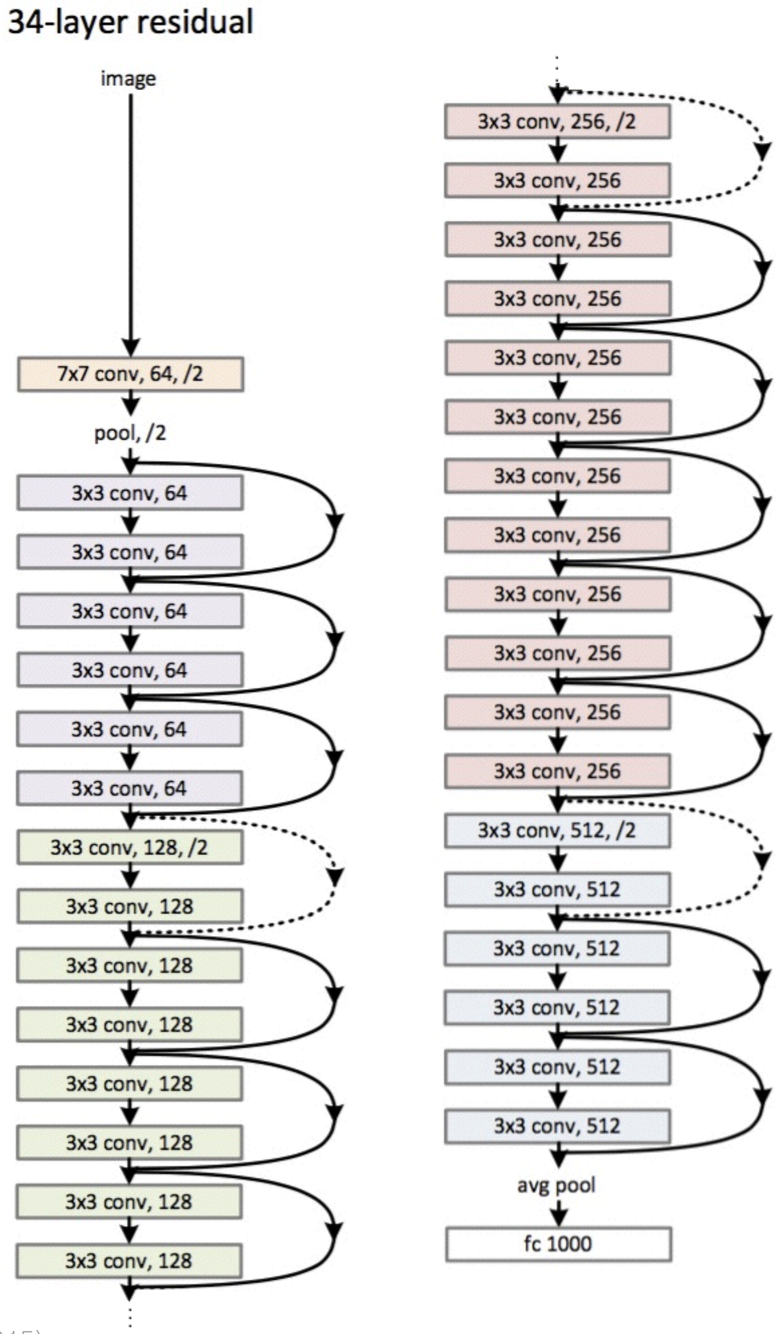
Deeper layers “see” more and more of the input all at once



https://dev.to/jai00271/cnn-general-terms-and-their-meaning-42nc

ResNet

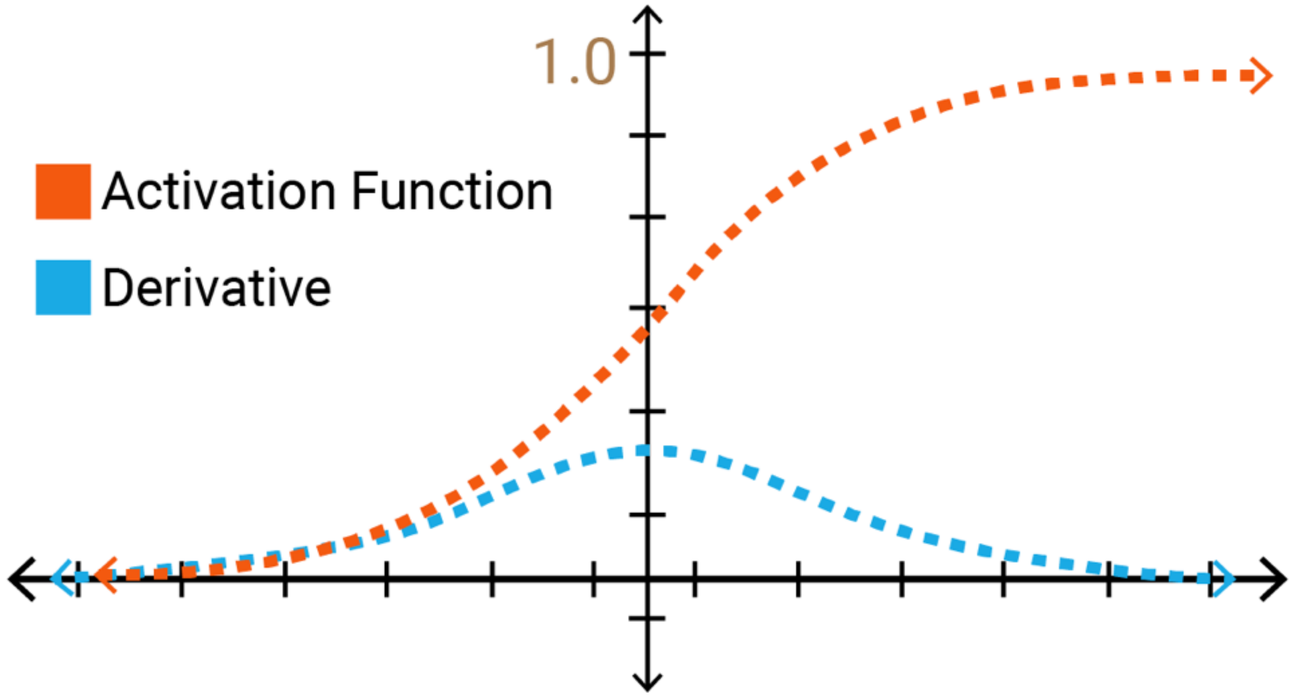
Residual neural network (ResNet) is one of the most commonly used architectures.
It contains **skip connections** that jump over some layers.
Skip connections improve training by avoiding vanishing gradients.



He *et al.* “Deep Residual Learning for Image Recognition” (2015)

Aside: Vanishing gradients

Gradients can become very small after stacking many layers of a neural network. This makes training difficult.



Recap: CNNs

CNNs take advantage of the structure of images.

They use:

- Convolutional layers
- Pooling layers

Parameters are shared for more efficient learning and translational invariance.

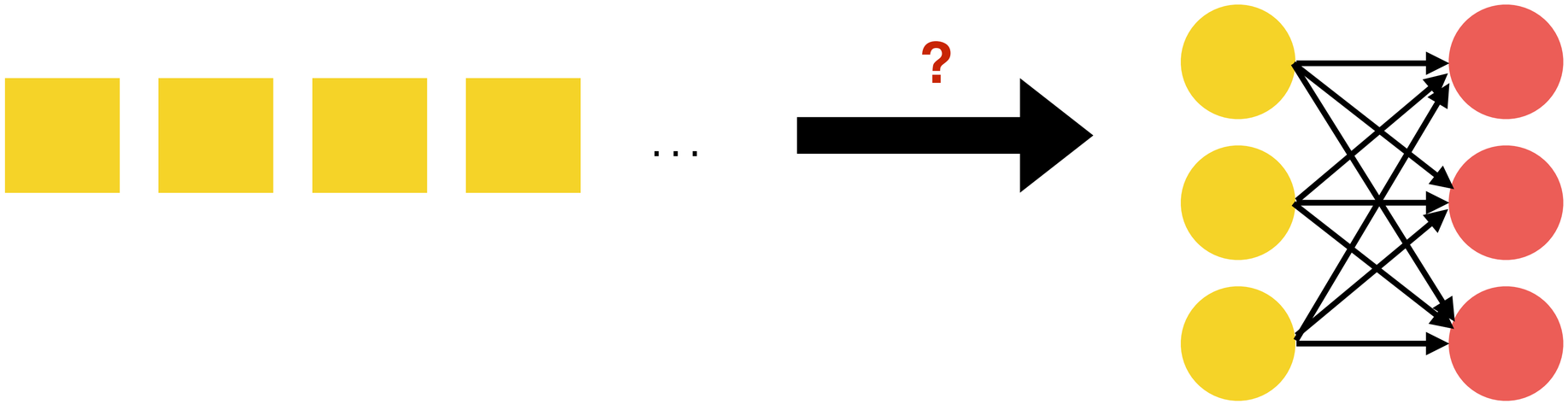
Recurrent Neural Networks

Recurrent neural networks

How to use neural networks for time series input?

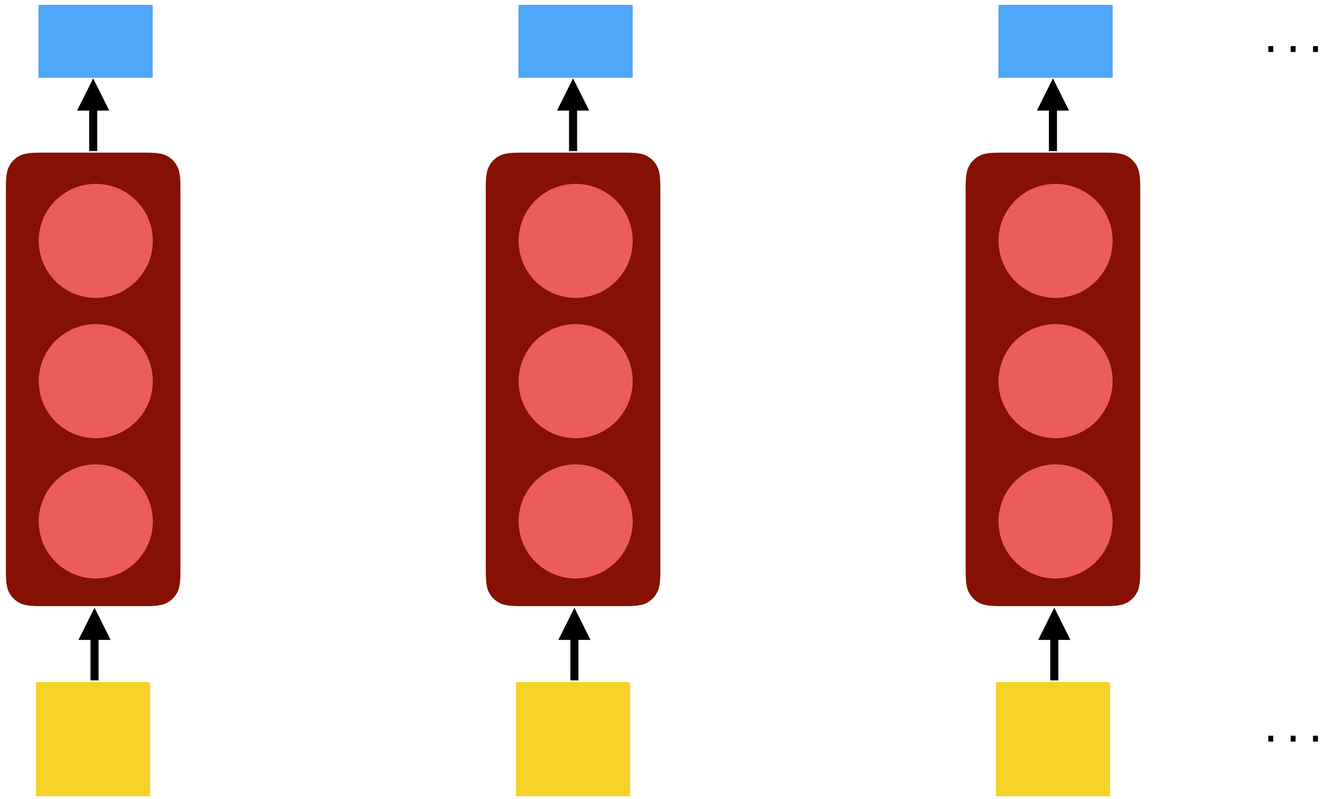
The length of input is not fixed or of indefinite length

Examples: audio, text, financial time series



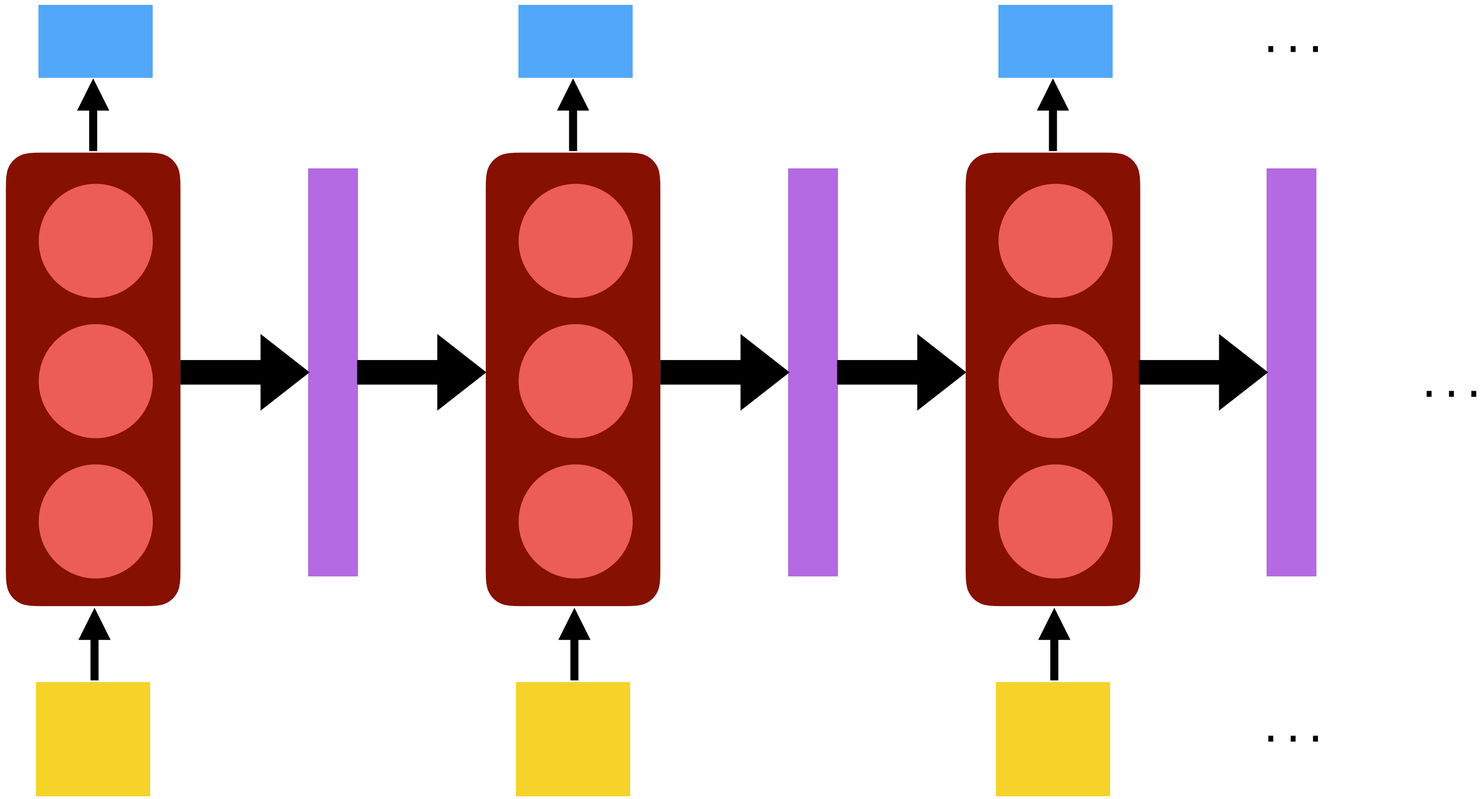
Recurrent neural networks

Share weights for each time step



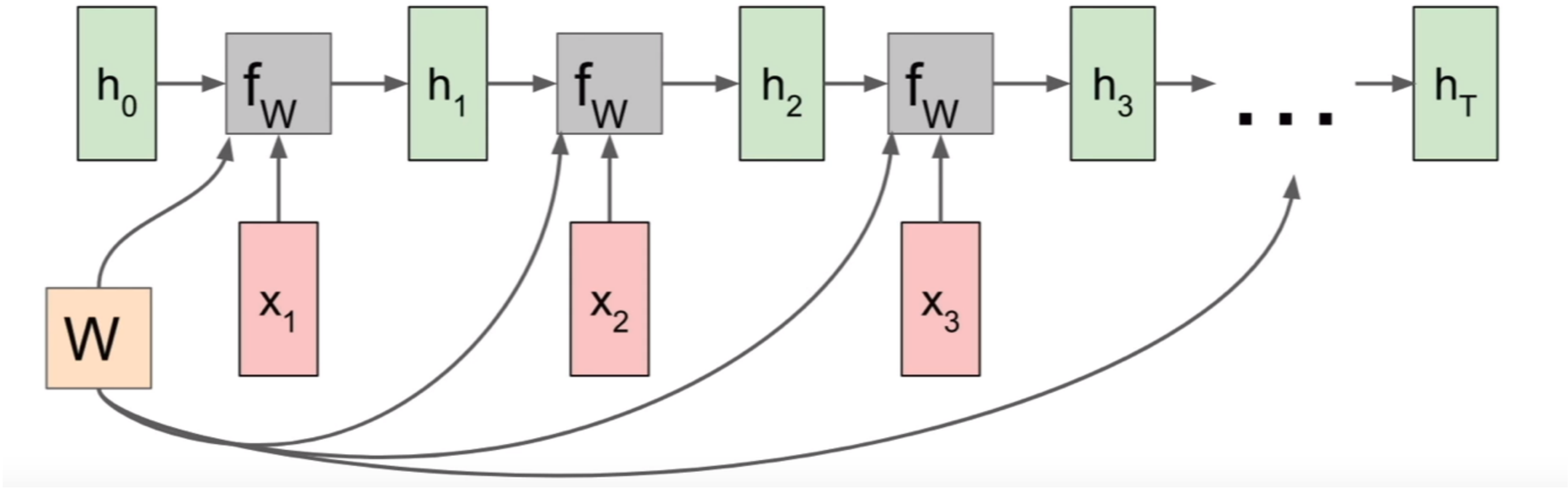
Recurrent neural networks

Pass information across time steps using **hidden states**



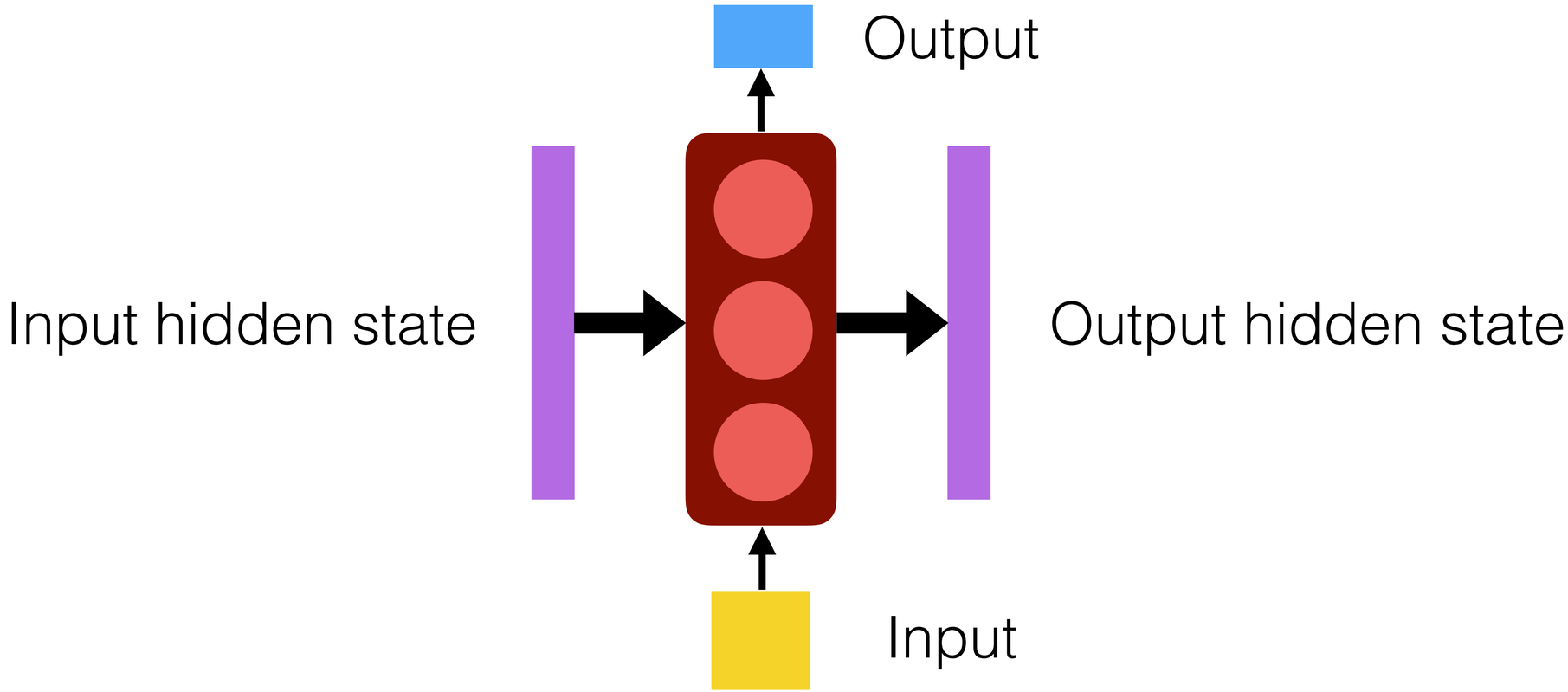
Recurrent neural networks

Computational graph view of things:

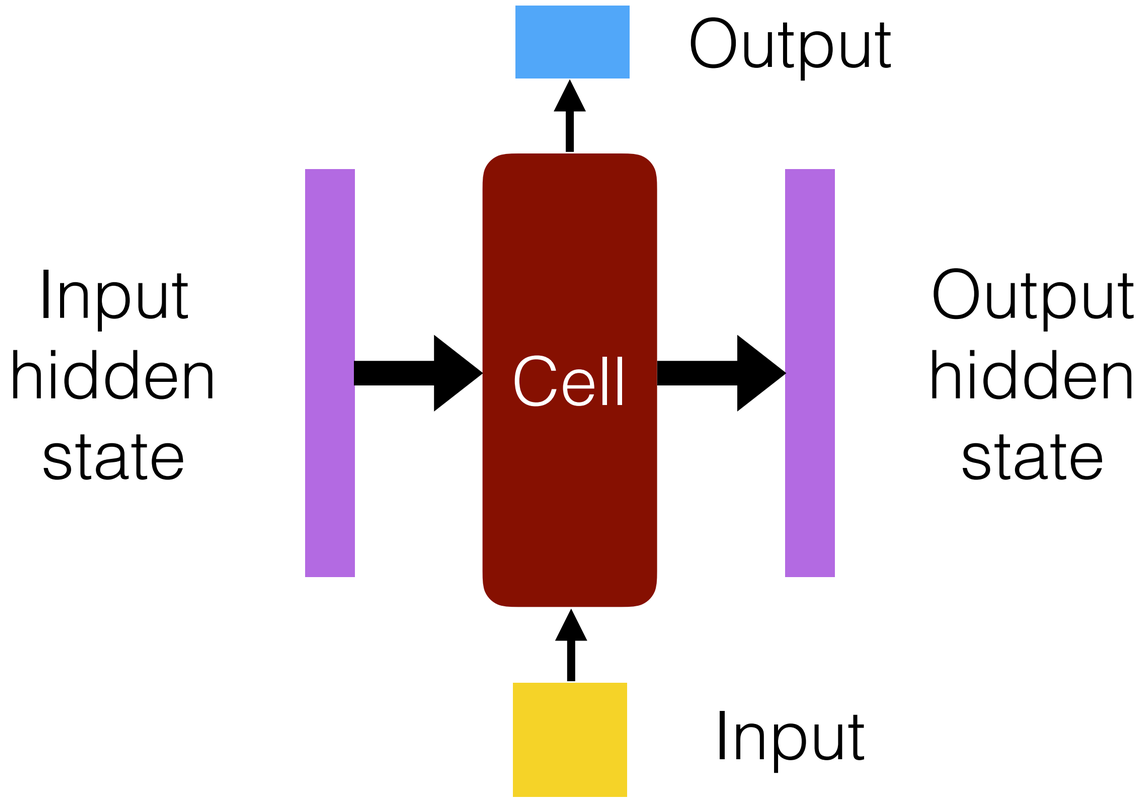


https://calvinfeng.gitbook.io/machine-learning-notebook/supervised-learning/recurrent-neural-network/recurrent_neural_networks

Hidden states



Types of cells

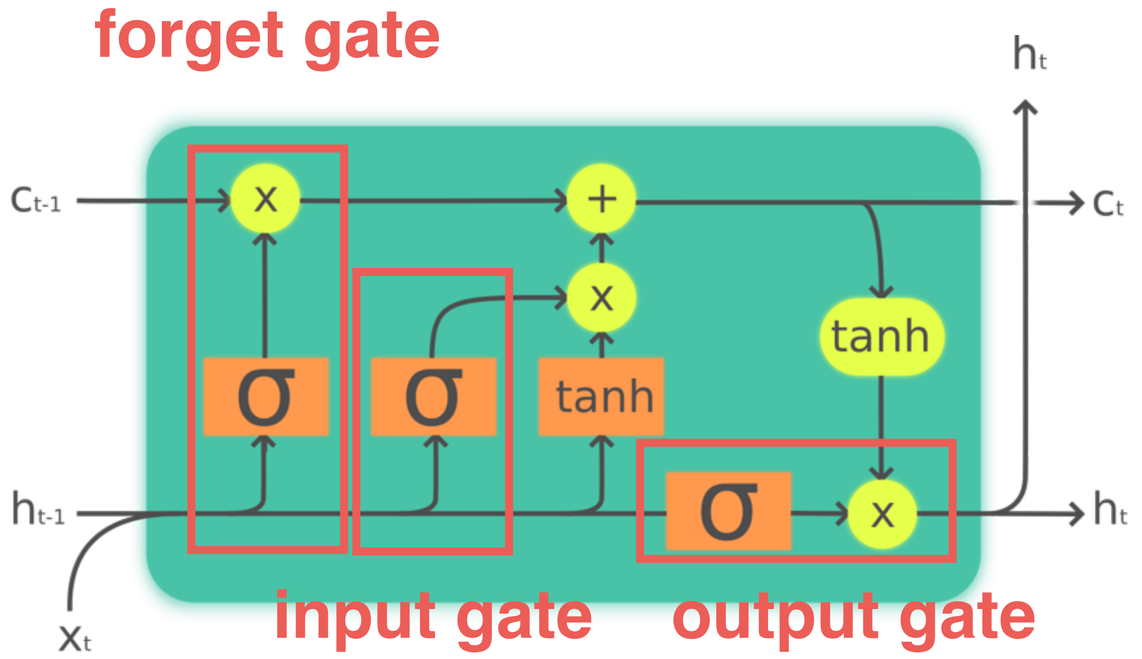


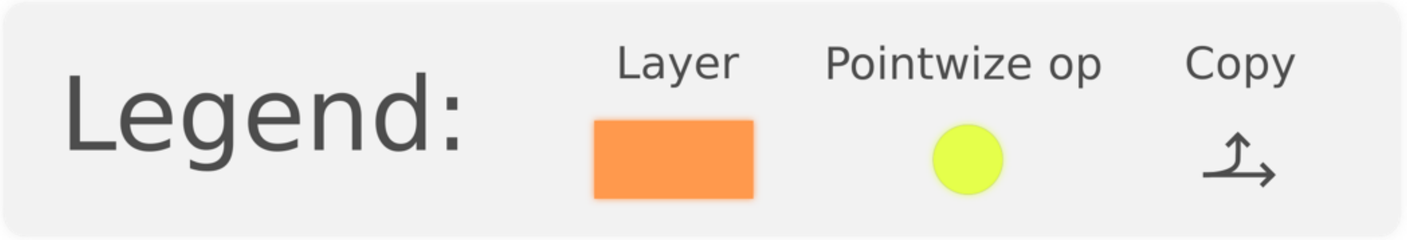
Many types of cells:

1. Vanilla RNN
2. **LSTM** - Long short term memory
3. **GRU** - Gated recurrent unit

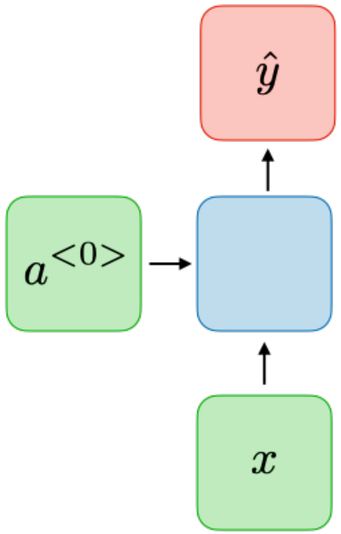
LSTM

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

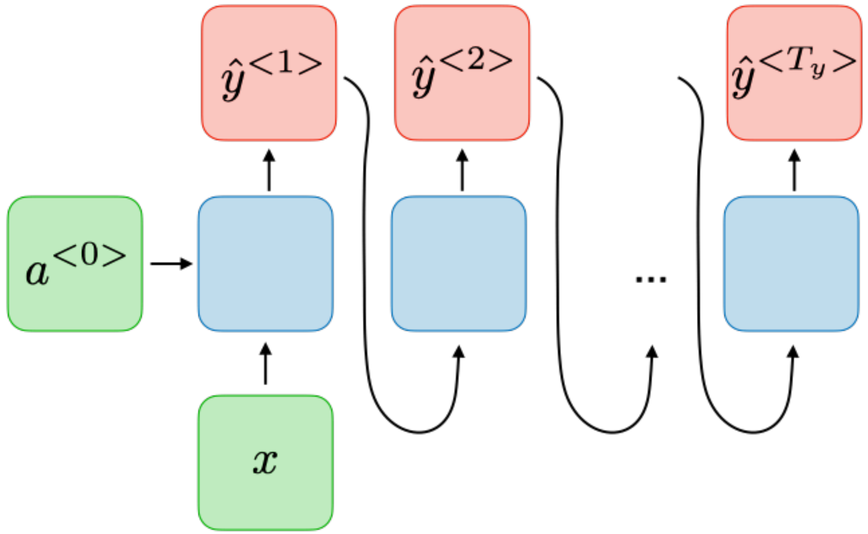




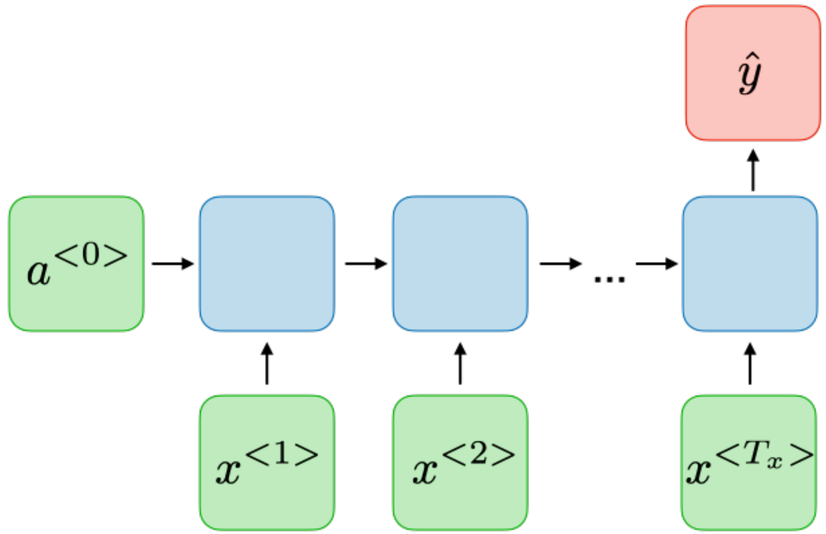
How many outputs?



One-to-one



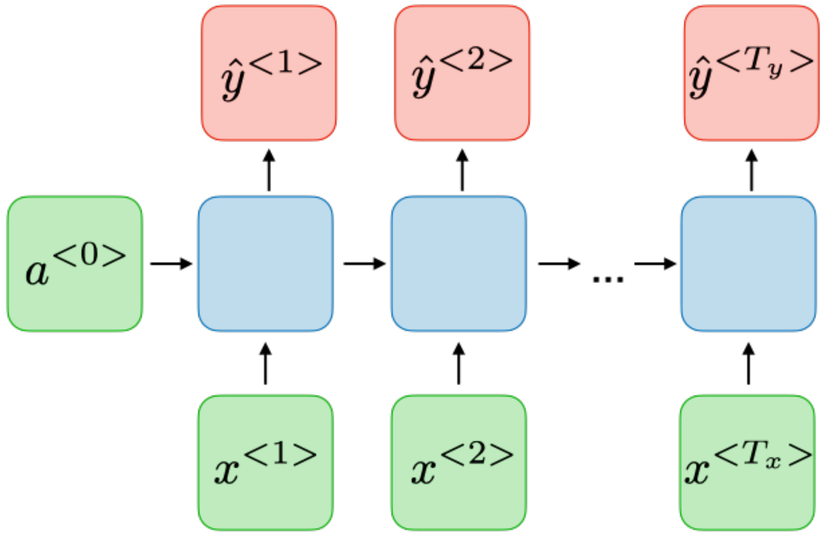
One-to-many



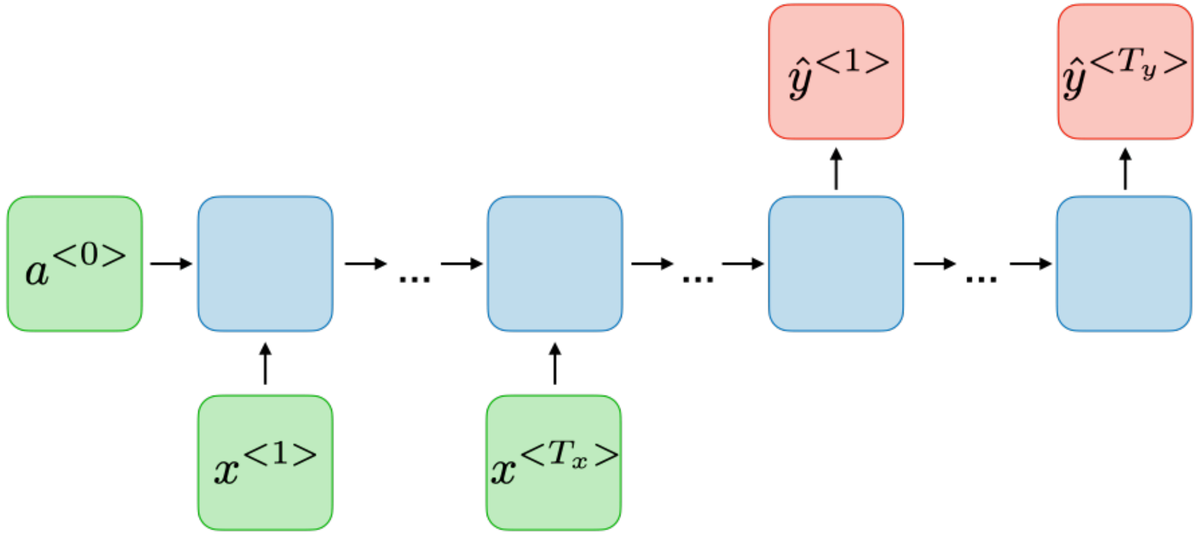
Many-to-one

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

How many outputs?



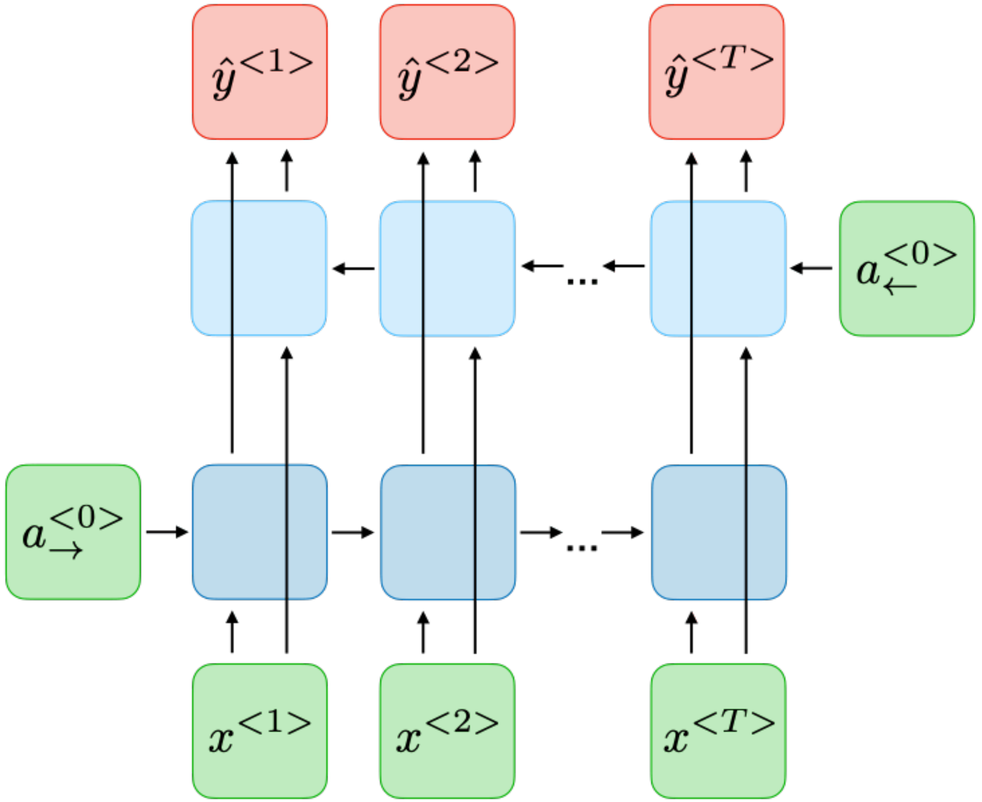
Many-to-many



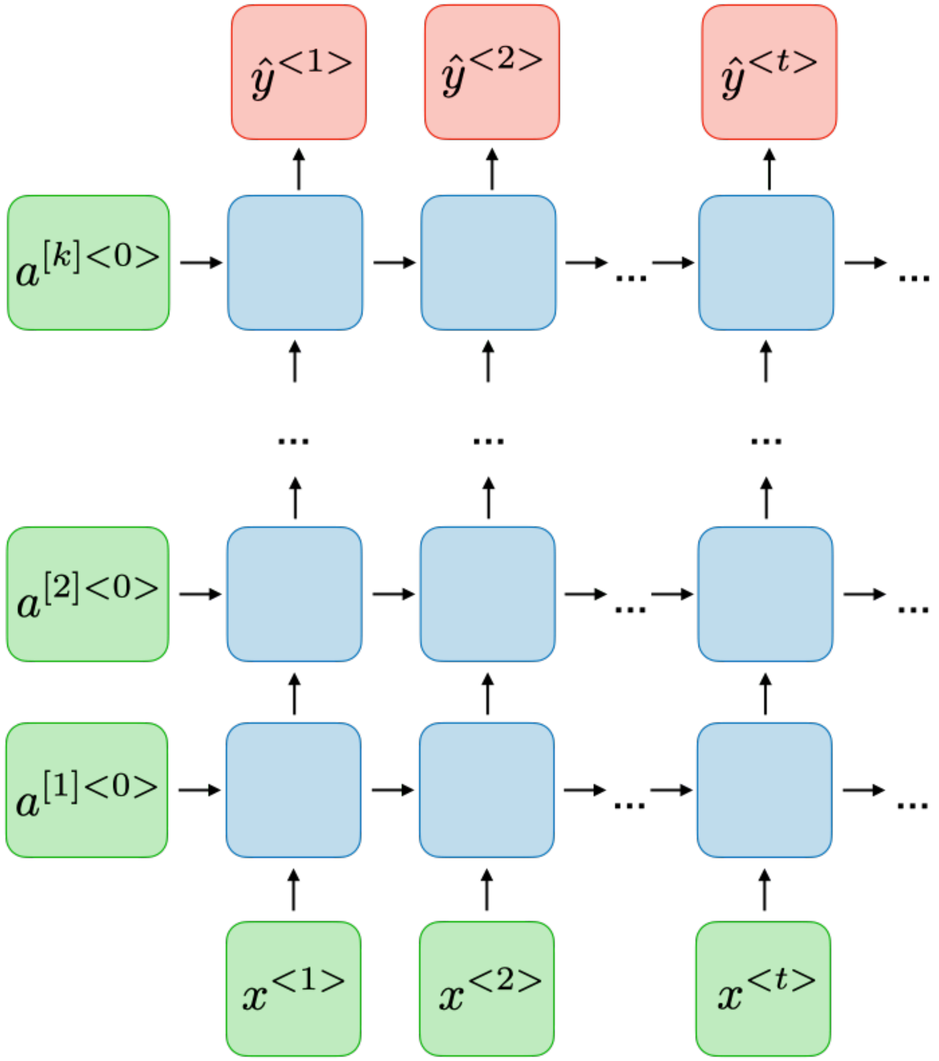
Many-to-many

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

RNN variants



Bidirectional



Deep

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recap: RNNs

RNNs allow for processing of variable-length inputs

Weights are shared across time steps; model size does not increase with size of input

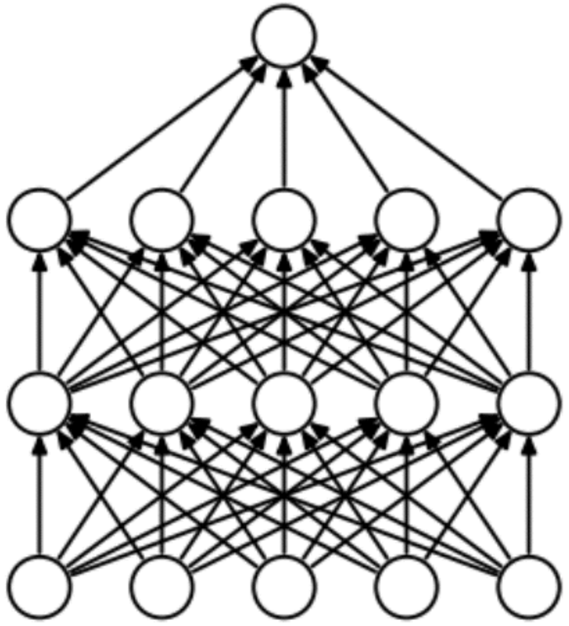
Computation is often slow, and the network “forgets” information from a long time ago

Other Architectures

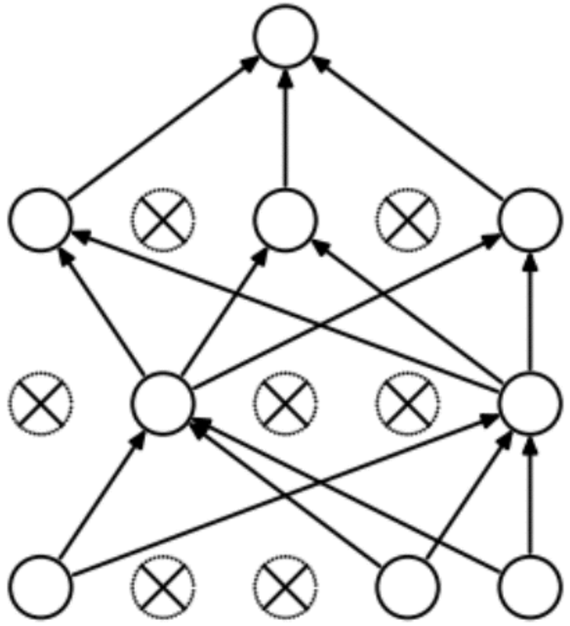
Dropout layers

Another way to regularize neural networks.

Randomly drop some fraction of neurons at a layer during training.



(a) Standard Neural Net

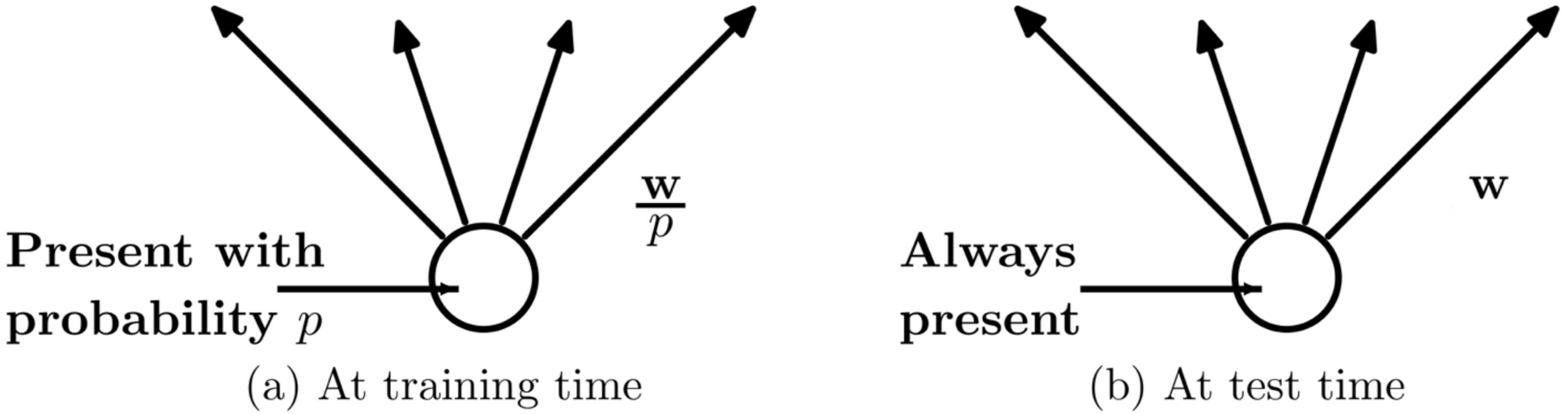


(b) After applying dropout.

Dropout layers

The output of non-dropped neurons is scaled by the inverse probability of keeping the neuron alive.

At test time, you do not use dropout.



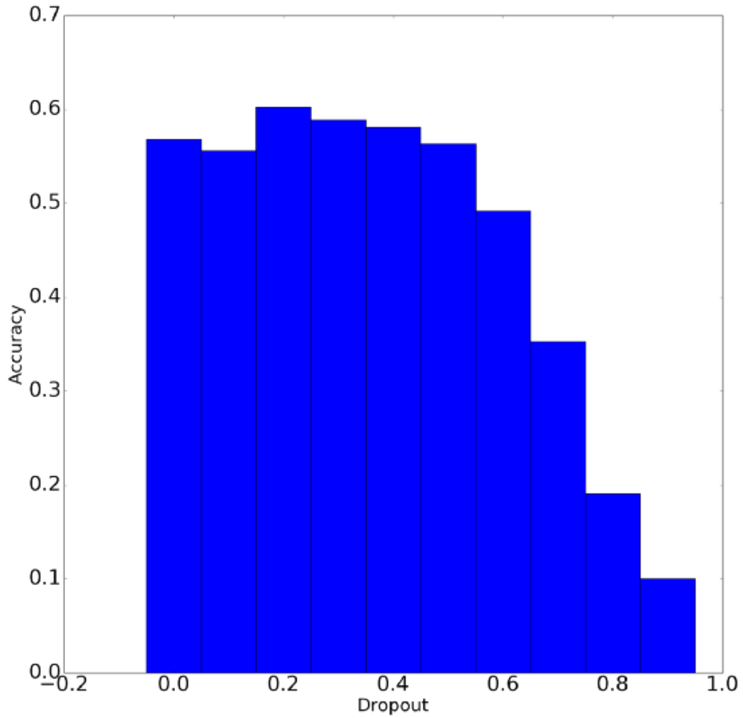
(a) At training time

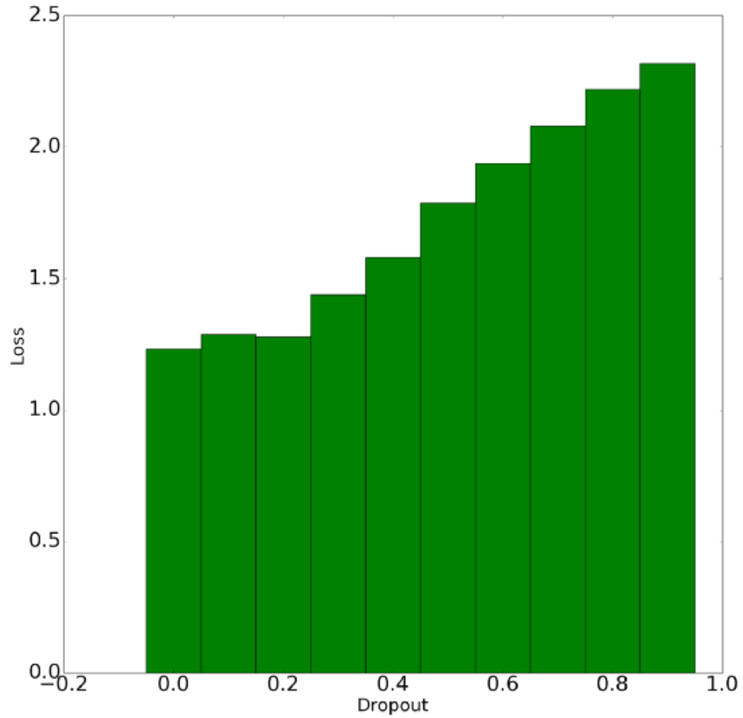
(b) At test time

Dropout layers

The point is to reduce dependence across neurons

Empirical results:



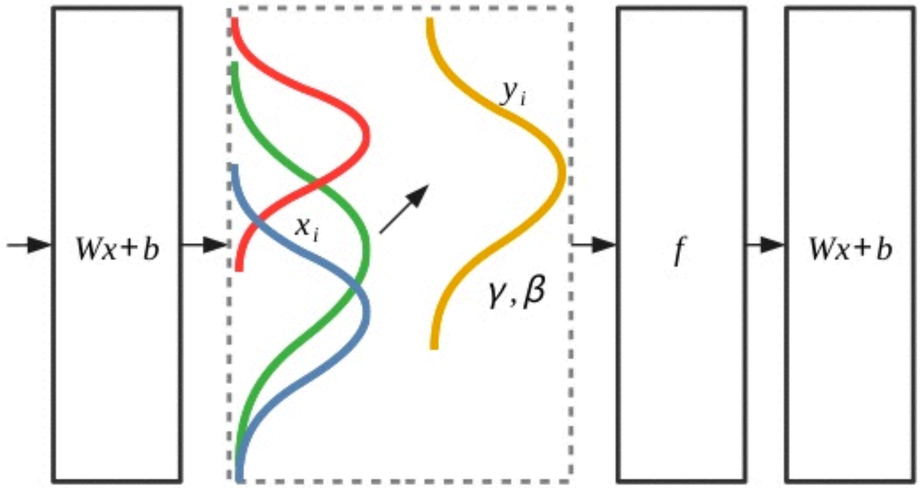


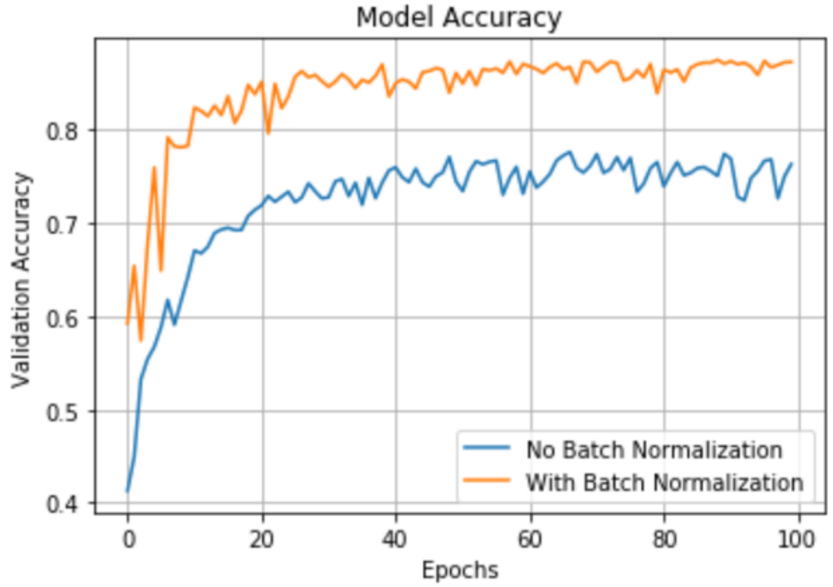
https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

Batch normalization

Goal is to control the mean and variance of each layer's output.

1. Normalize output of each neuron to be mean zero and unit variance across the batch at every training step.
2. Rescale the output of the neuron to be some specific mean and variance, both of which are trainable parameters.



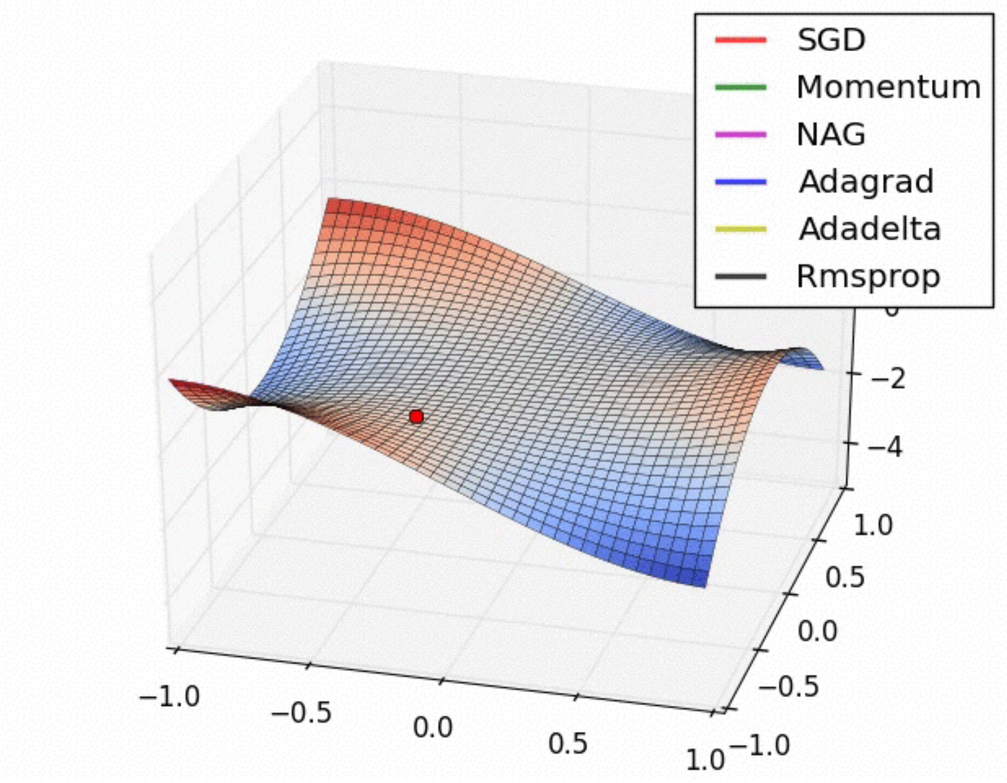


Other optimizers

Adaptive learning rate: adjust learning rate based on mean and variance of gradients

Momentum: instead of current gradient, take weighted average of past gradients to update the weights

Techniques: RMSProp, Adagrad, Adam, etc.

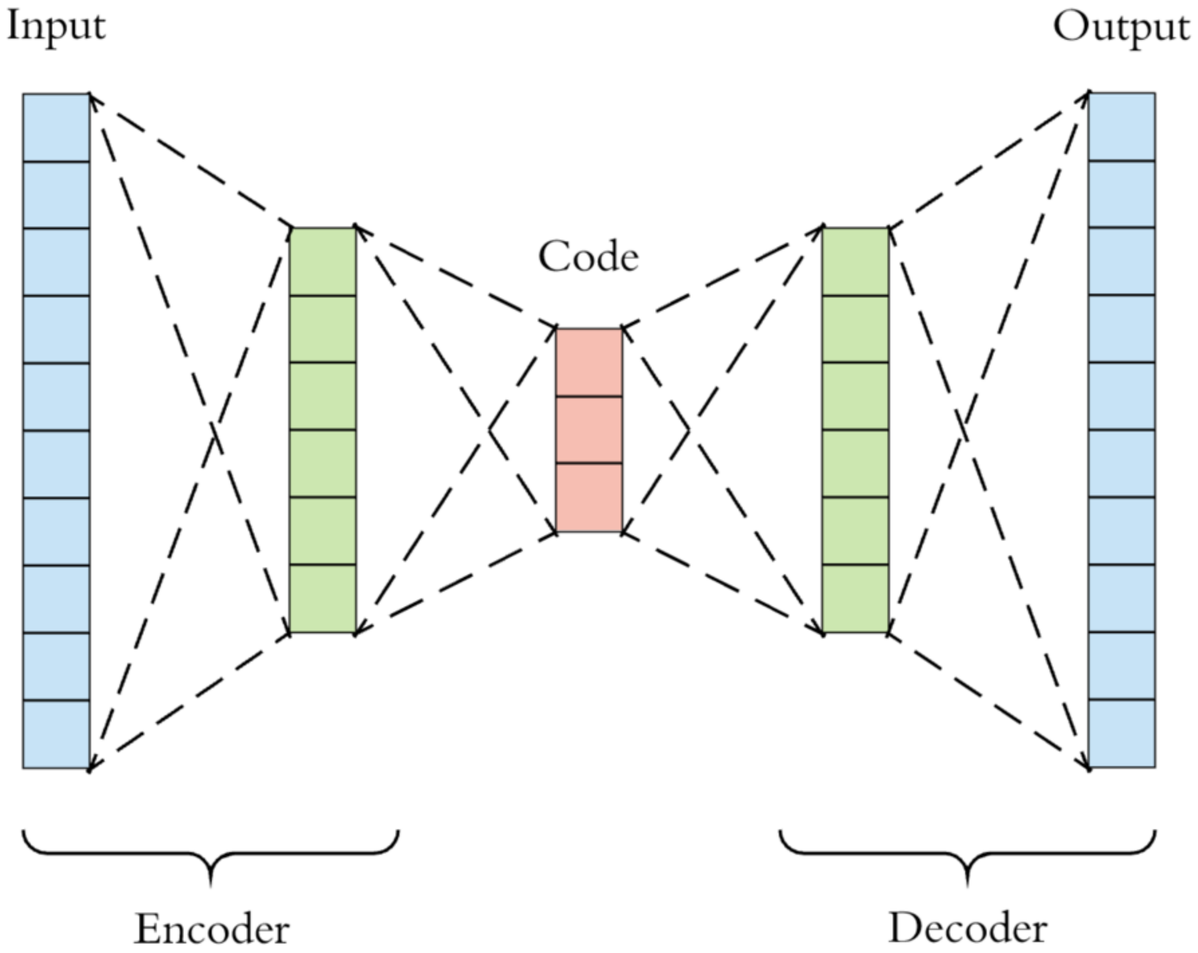


Autoencoders

A way to do unsupervised deep learning

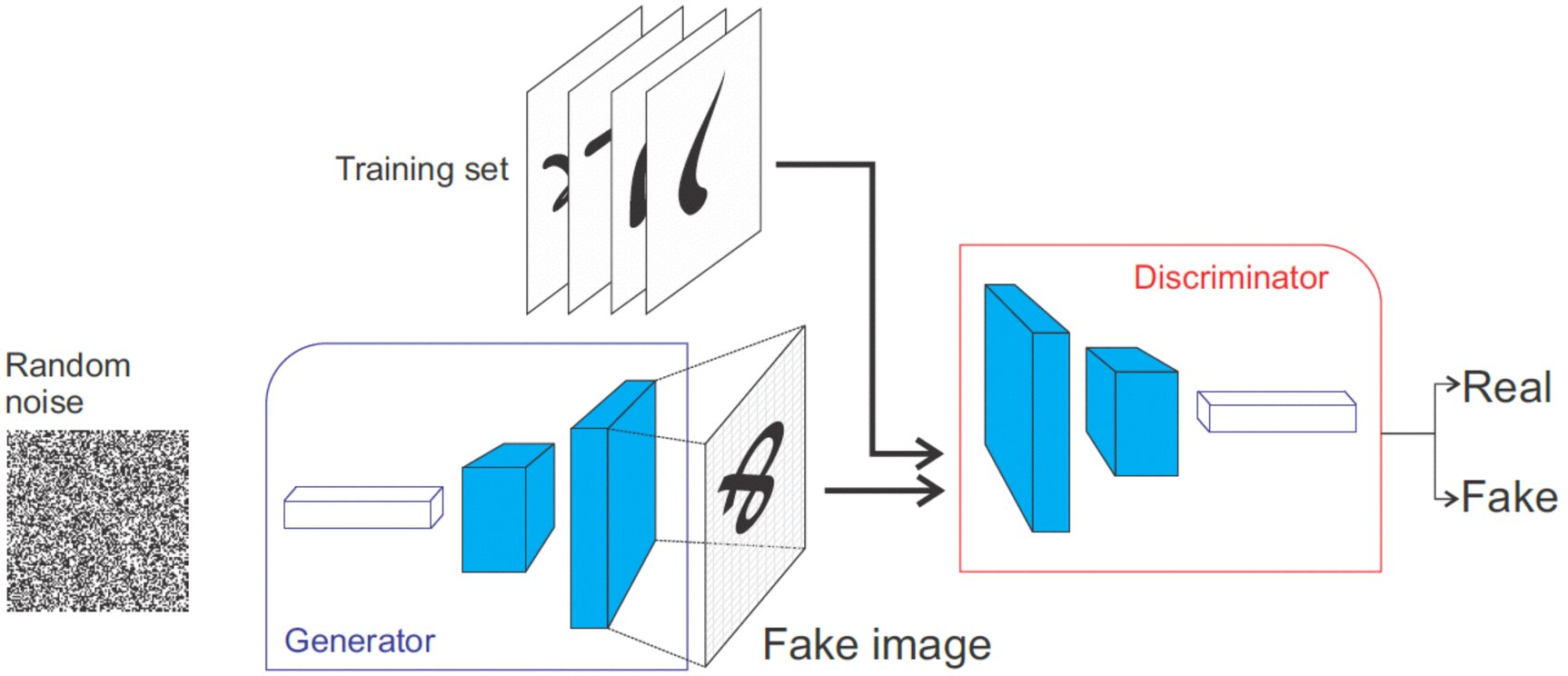
Input and output are the same: the network is tasked with recreating the input but going through a lower-dimensional bottleneck

Bottleneck layer is interpreted as a latent representation for the input



https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

Generative adversarial networks



https://www.oreilly.com/library/view/java-deep-learning/9781788997454/60579068-af4b-4bbf-83f1-e988fbe3b226.xhtml

Generative adversarial networks

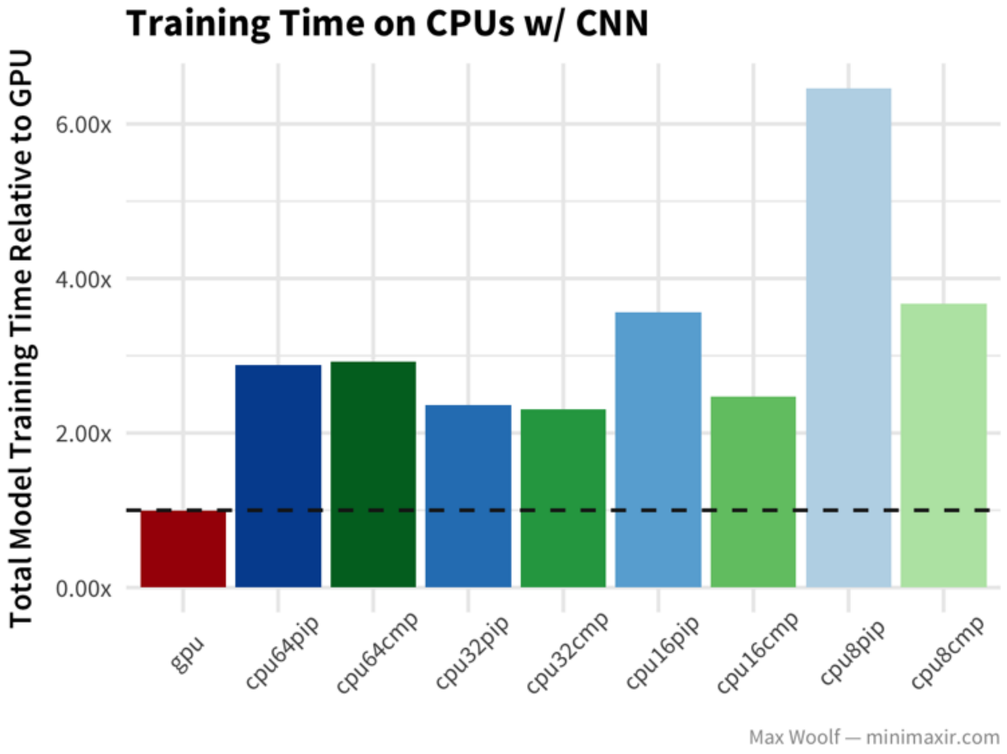
Deep Learning Libraries

What do deep learning packages provide?

1. Ability to run on GPUs
2. Build computation graphs and provide automatic differentiation
3. Useful medium- to high-level components for deep learning already implemented (convolutions, batch norm, VGG, ResNet, etc.)

CPUs vs GPUs

GPUs are optimized to perform matrix multiplications, and speed up neural network training many-fold



Deep learning libraries

1. TensorFlow — mainly used in industry
2. Keras — easy-to-use API for TensorFlow
3. PyTorch — mainly used in research
4. Others include Caffe, Theano, MXNET…

Coding Session

ICME Summer Workshops 2020

Introduction to Deep Learning

Session 4: 3:30—4:45 PM
Instructor: Sherrie Wang

icme-workshops.github.io/deep-learning

Eykholt *et al.* “Robust Physical-World Attacks on Deep Learning Visual Classification” (2018)

Workshop Schedule

Session 1 (9:00—10:30 AM)
- Introduction
- Current state-of-the-art in deep learning
- Math review
- Fully connected neural networks

Session 2 (10:45—12:00 PM)
- Loss functions
- Gradient descent
- Backpropagation
- Overfitting and underfitting

Lunch (12:00—2:00 PM)

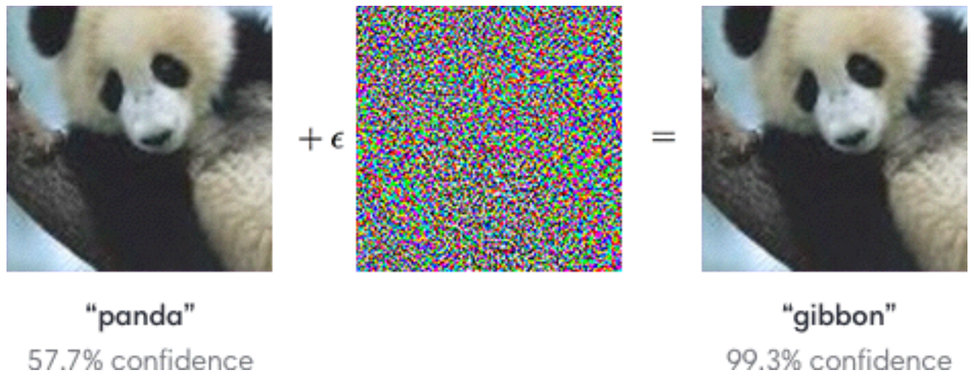
Session 3 (2:00—3:15 PM)
- Convolutional neural networks
- Recurrent neural networks
- Other architectures
- Deep learning libraries
- Hands-on coding session in Tensorflow

Session 4 (3:30—4:45 PM)
- Hands-on coding session in Keras
- Hands-on coding session on transfer learning
- Failures of deep learning

Failures and Limits

Adversarial attacks

- Noise that is imperceptible to humans has been shown to dramatically change neural network output



Real-world adversarial attacks

- Many self-driving cars rely on convolutional neural networks…

Table 1: Sample of physical adversarial examples against LISA-CNN and GTSRB-CNN.

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

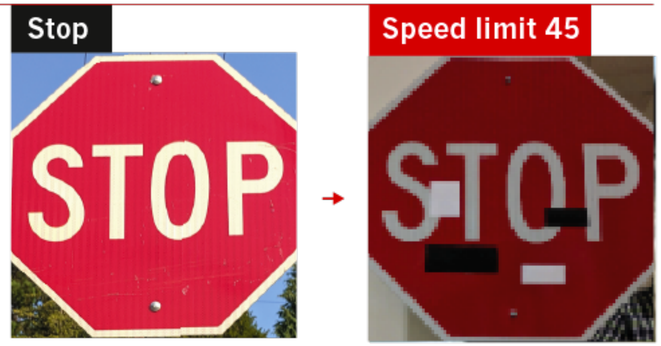
Eykholt *et al.* “Robust Physical-World Attacks on Deep Learning Visual Classification” (2018)

Adversarial attacks

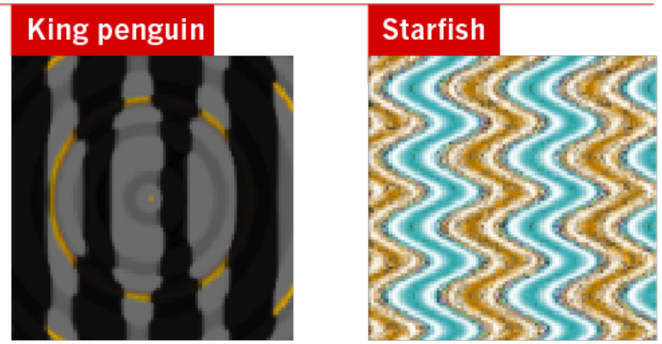
FOOLING THE AI

Deep neural networks (DNNs) are brilliant at image recognition — but they can be easily hacked.

These stickers made an artificial-intelligence system read this stop sign as ‘speed limit 45’.

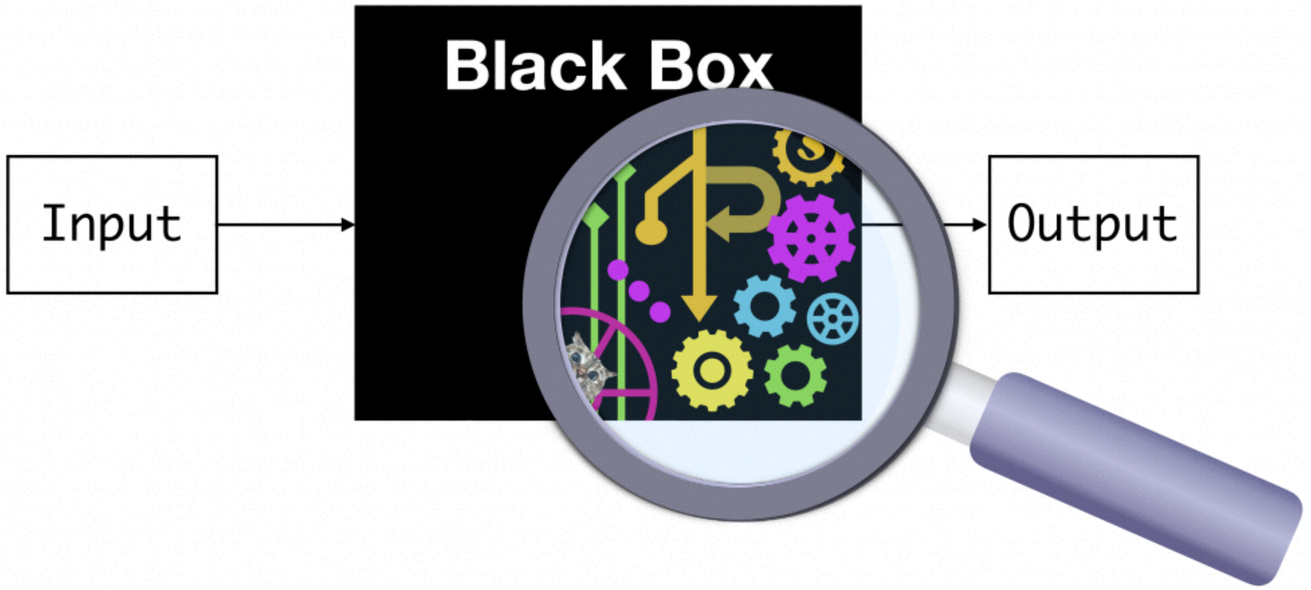


Scientists have evolved images that look like abstract patterns — but which DNNs see as familiar objects.



“Brittle, greedy, opaque, and shallow”

“People can rationalize what’s going on in their thought processes. Deep learning can’t: These systems have no idea how they’re thinking or how they’re categorizing themselves.” —Rodney Brooks



https://www.flagshippioneering.com/stories/brittle-greedy-opaque-and-shallow-the-promise-and-limits-of-todays-artificial-intelligence

“Brittle, greedy, opaque, and shallow”

“True intelligence is being able to approach a new problem you haven’t had a lot of direct experience with. A human being can play a game that they’ve never played before and in a matter of minutes figure out something about what’s going on. Machines still can’t do that.”

—Gary Marcus



??

Deep learning can be notoriously brittle

https://www.flagshippioneering.com/stories/brittle-greedy-opaque-and-shallow-the-promise-and-limits-of-todays-artificial-intelligence

Thank you!