Deep Learning Theory and Applications

Yale

AMTH/CBB 663
CPSC 452/552

- Whats your favorite show to binge-watch during the pandemic?

Outline

1. Course logistics
2. What is Deep Learning?
3. Deep learning examples
   - CNNs
   - GNNs
   - RNNs
   - Transformers
   - Autoencoders
   - Ultra deep learning (ResNet)
   - Generative models (e.g. GANs)
   - Deep Reinforcement Learning

Course Logistics

- Textbooks (available online)
  - Neural Networks and Deep Learning by Michael Nielsen
  - Deep Learning by Goodfellow, Bengio, and Courville
  - Graph Representation Learning by William Hamilton

- Required background
  - Basic probability
  - Linear algebra & calculus
  - Programming experience
    - Python and PyTorch will be used in this course
- Canvas
  - Lectures recorded and posted after class
  - Announcements & HW
- Piazza
  - Please sign in via link
  - piazza.com/yale/spring2022/cpsc45201cpsc55201amth55201cbb66301/home

Instructor TA and ULA

- Instructor: Smita Krishnaswamy
- TA: Sumner Magruder
- ULAs: Kincaid Macdonald, Ross Johnson, Chae Young Lee

Sumner

Kincaid

Ross

Chae Young

Office hours

- Office hours by ZOOM: 991 8356 1572
  - Instructor Office Hours: After class Tuesdays 1-2
    - OR by appointment (email me).
  - Sumner: Monday 10-12pm
  - Kincaid: Friday 3-5pm
  - Ross: Tuesdays 7:3—9:30pm
  - Chae Young: Wednesdays 7-9pm
- These may be subject to change

Homeworks and projects

- 4 HW assignments
  - All/most will include some programming (Python & Pytorch)
  - Approximately 3-4 weeks to do them
- Final project (details forthcoming)
  - In groups of 3-4
- Intro to Pytorch:
  - https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

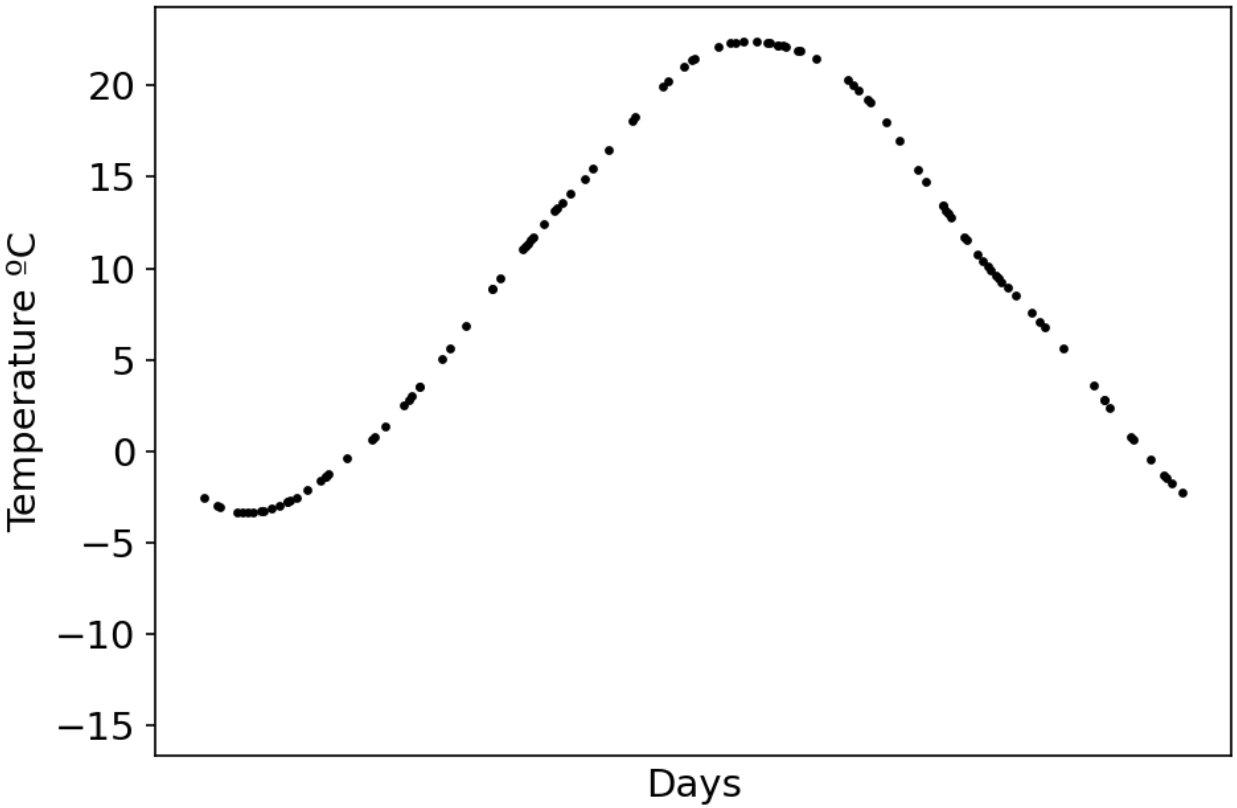
have milestone
proposal
progress report
CS paper no presentation

- Pytorch bootcamp: Sat 10-12pm 1.31
- final
- no midterm

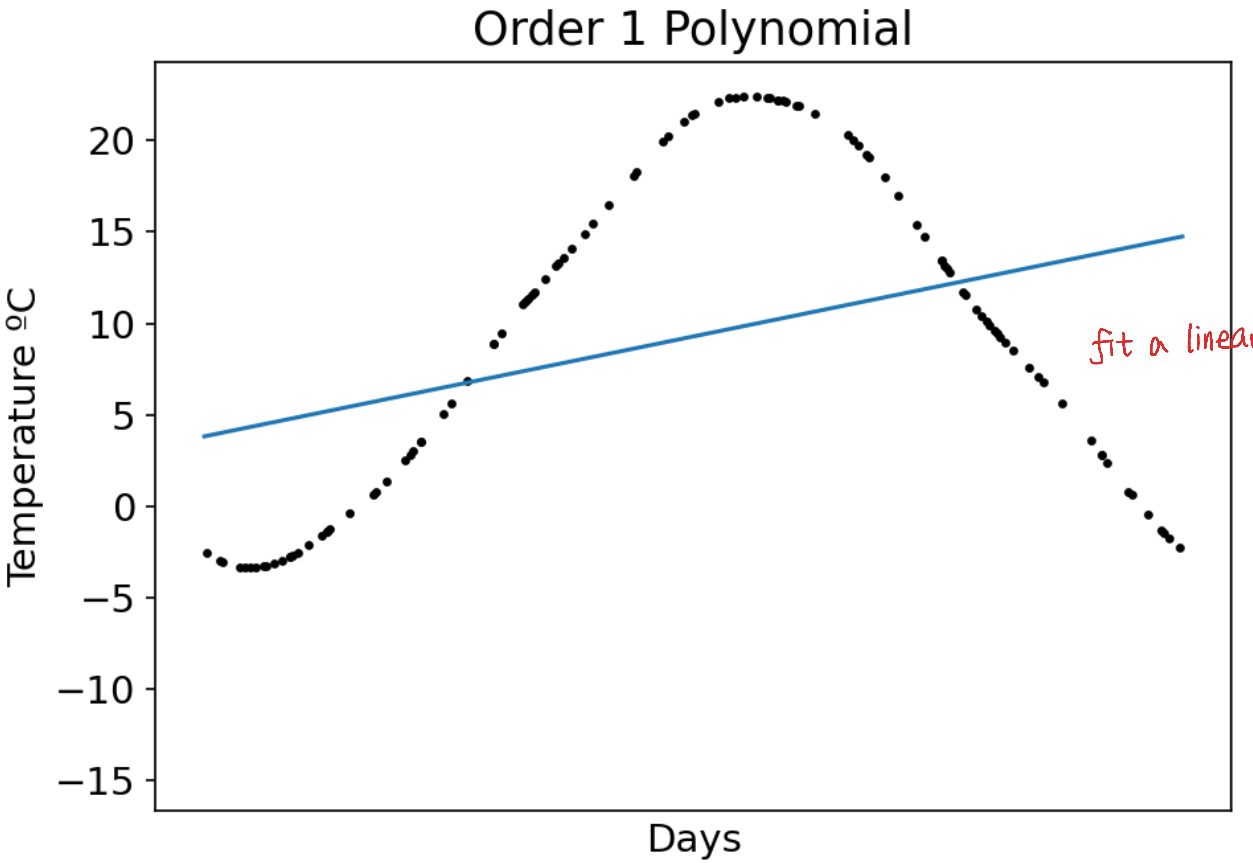
Goals of the Course

- Intuition behind "depth" in deep learning
  - Abstract features
  - Representation learning
- An understanding of prominent deep architectures
  for different tasks in different domains
- The ability to design and train novel architectures for problems of interest
- An understanding of how deep networks are trained and optimized
  trained by give answers of a partlcular set cases
  optimized by backpropagate the error
- Awareness of emergent understanding of why neural networks actually generalize, and in what circumstances they "memorize"
  by no means completely figure out
- Familiarity with problems in neural networks and future directions

How can we train a model to predict something on unseen data?



How can we train a model to predict a value on unseen data?



underfitting

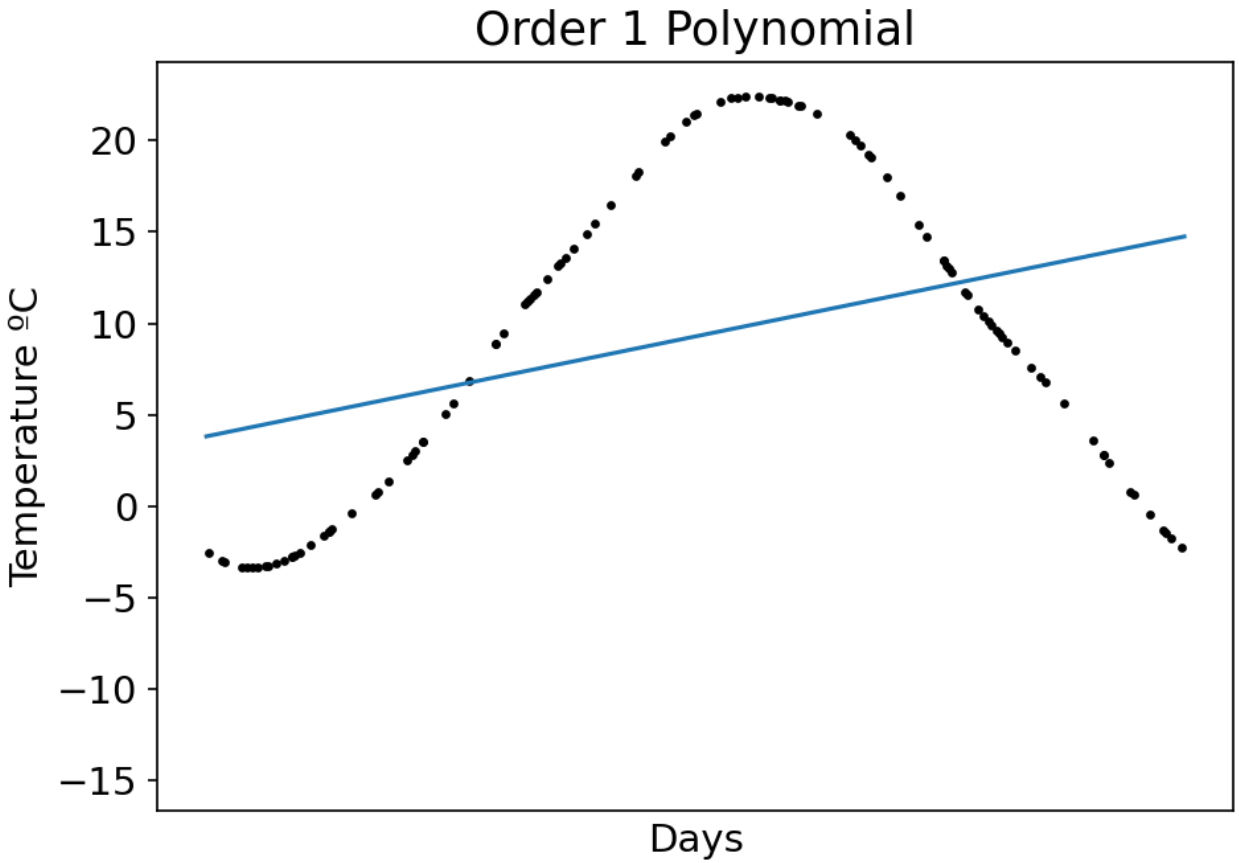
fit a linear model

$$y = f(x, \theta)$$

$$y = \theta_0 + \theta_1 x$$

$$y = 3.81 + 0.03x$$

Predictions



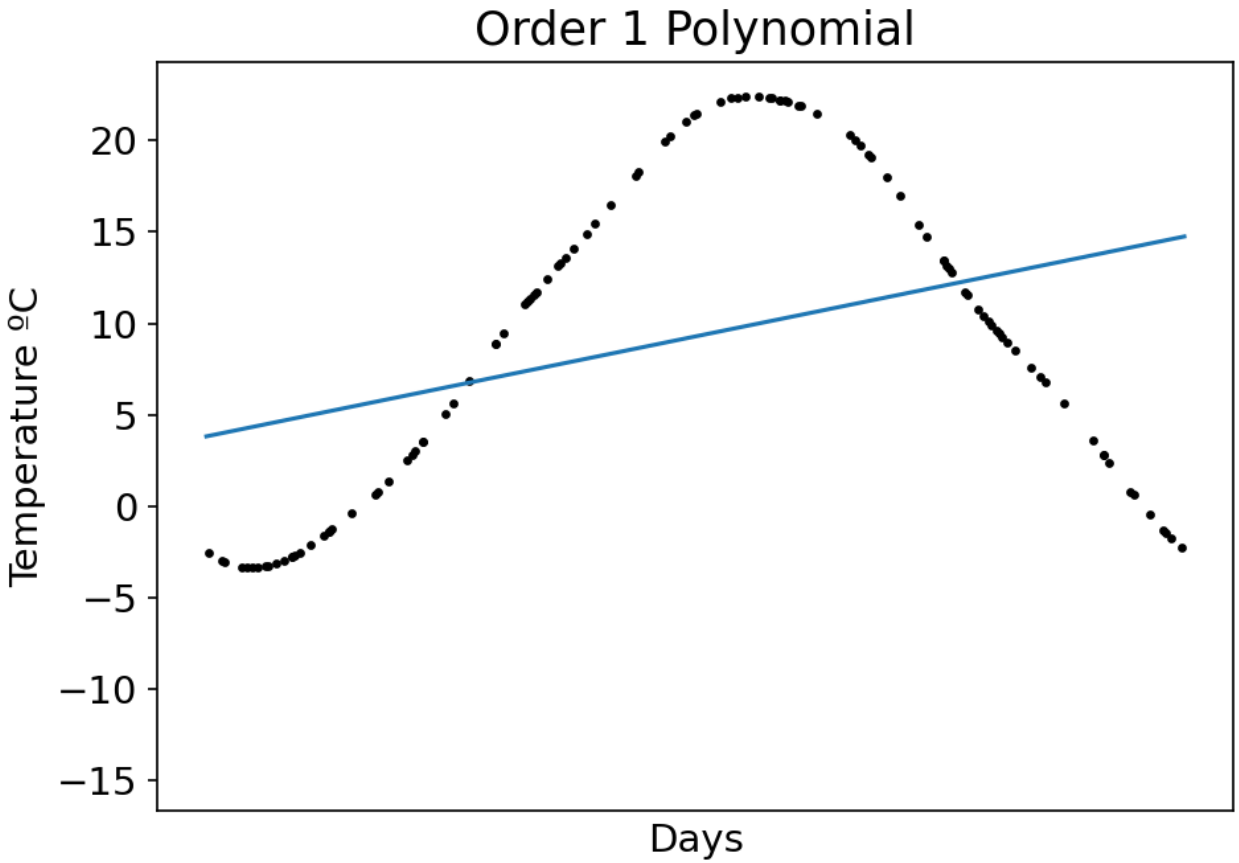
$$y = f(x, \theta)$$

$$y = \theta_0 + \theta_1 x$$

$$y = 3.81 + 0.03x$$

Bias Weight

Optimizing Parameters



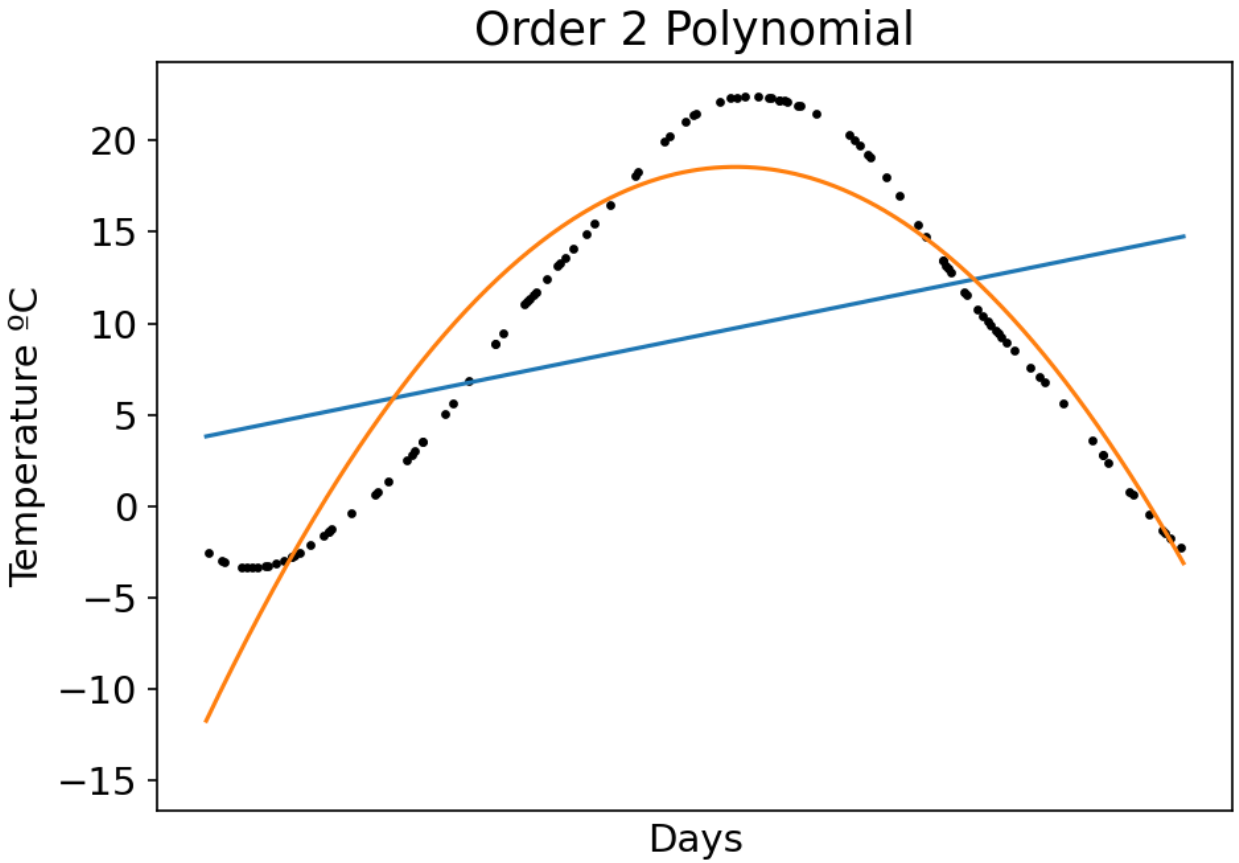
$$y = f(x, \theta)$$

$$y = \theta_0 + \theta_1 x$$

$$y = 3.81 + 0.03x$$

$$y = \theta_0 + \sum_i \theta_i x^i$$

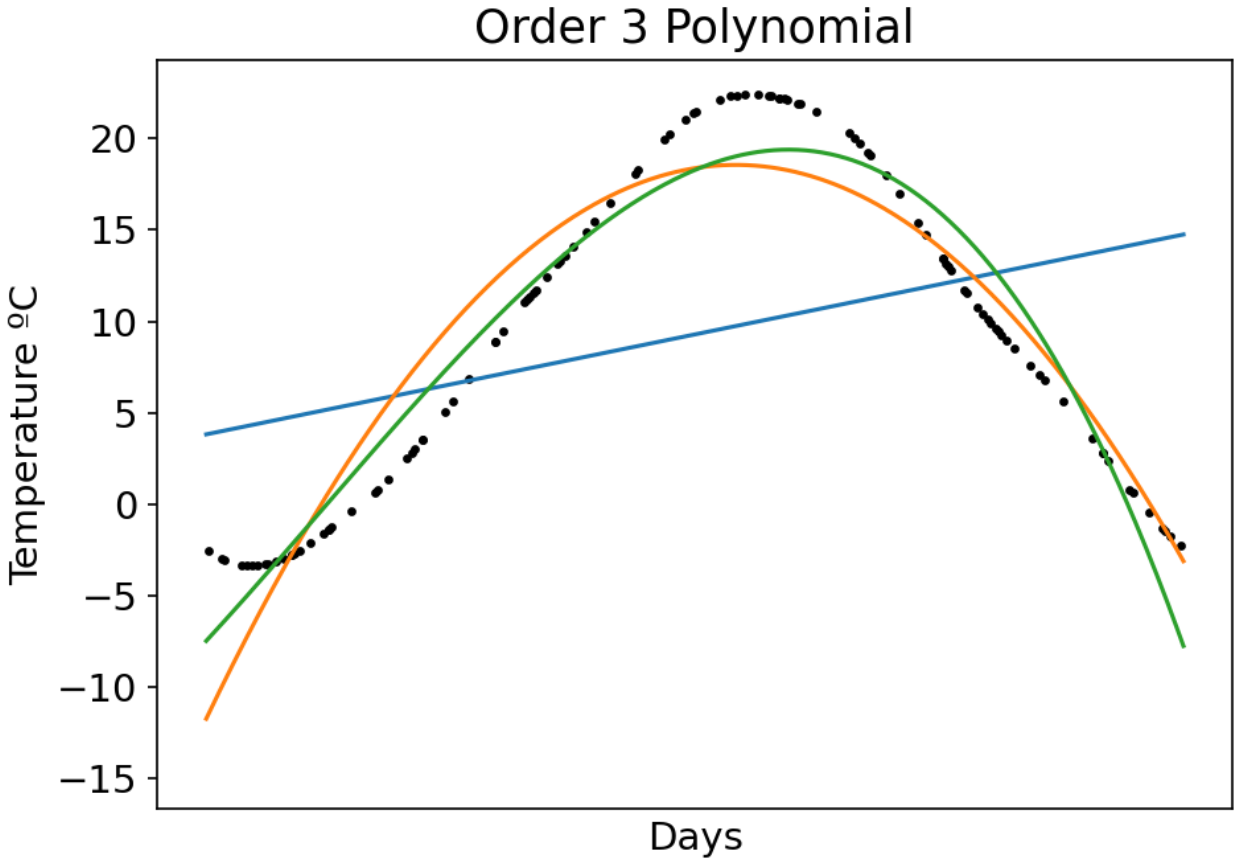
More parameters



$$y = f(x, \theta)$$

$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

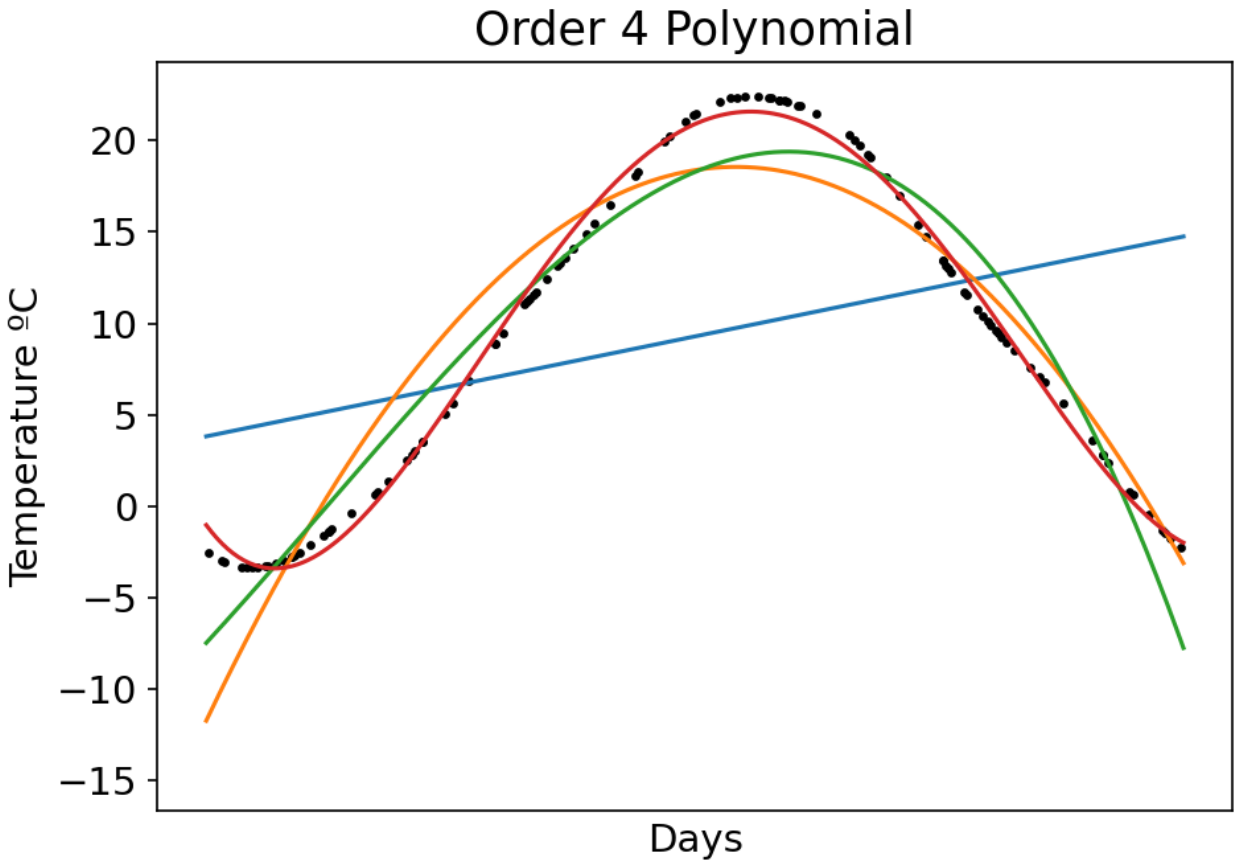
… even more parameters



$$y = f(x, \theta)$$

$$y = \theta_0 + \cdots + \theta_3 x^3$$

…even more parameters

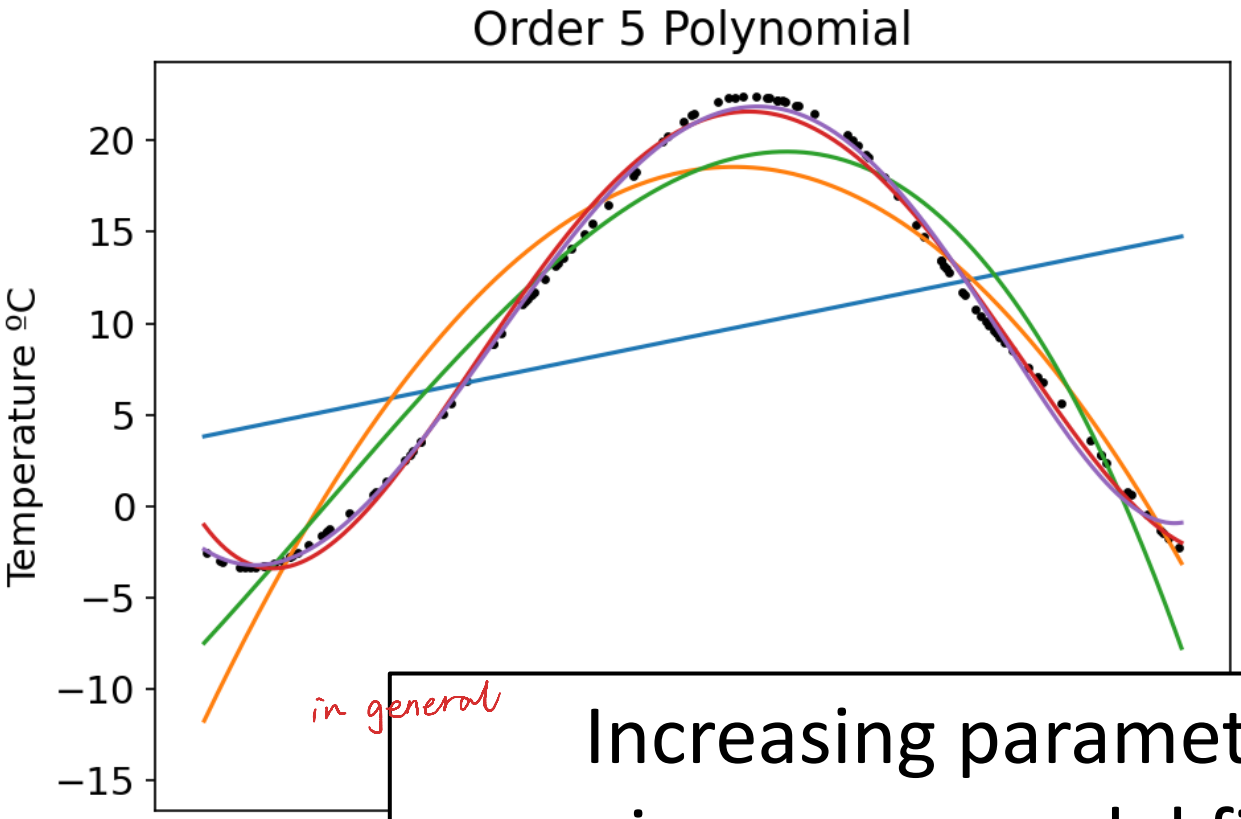


$$y = f(x, \theta)$$

$$y = \theta_0 + \cdots + \theta_4 x^4$$

…even more parameters

neural networks with enough params can fit any continuous / discrete functions that can be well approximated by any continuous function



$$y = f(x, \theta)$$

$$y = \theta_0 + \cdots + \theta_5 x^5$$

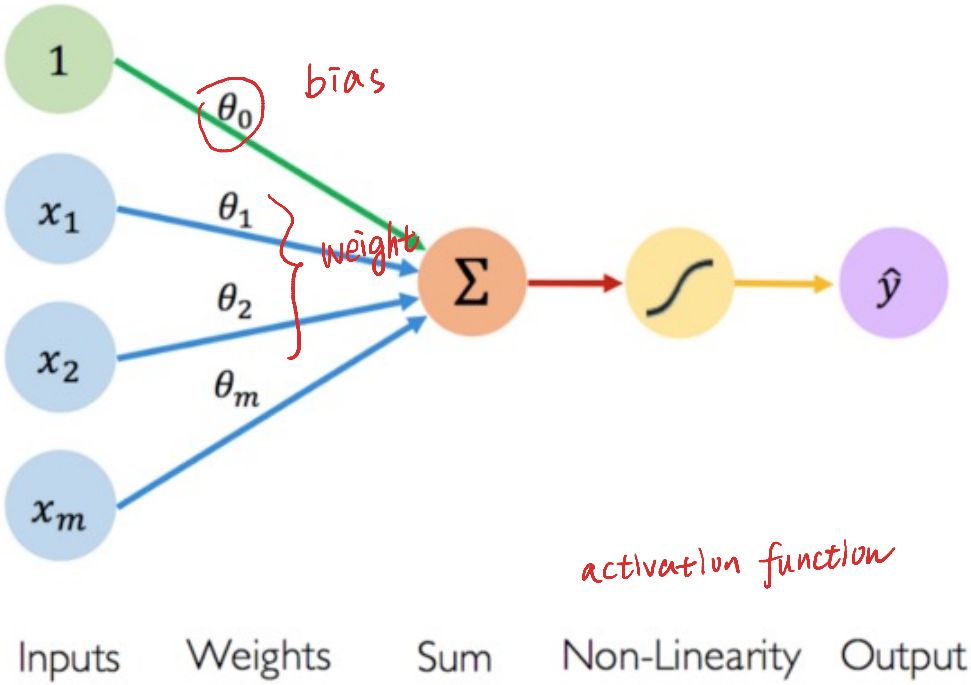
core of neural network
basically NN can fit anything
↗ universal function of estimators

in general

Increasing parameters improves model fit!

richness of expressibility ↗
richer class of functions that can be expressed with those params

The heart of neural networks are weights and biases

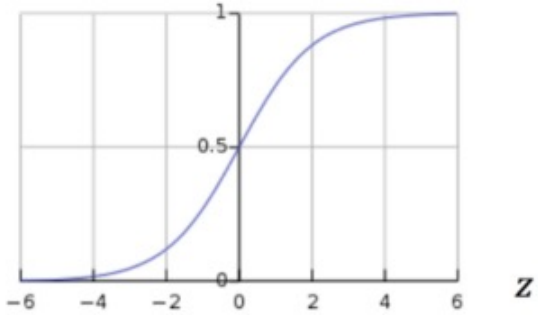


Activation Functions

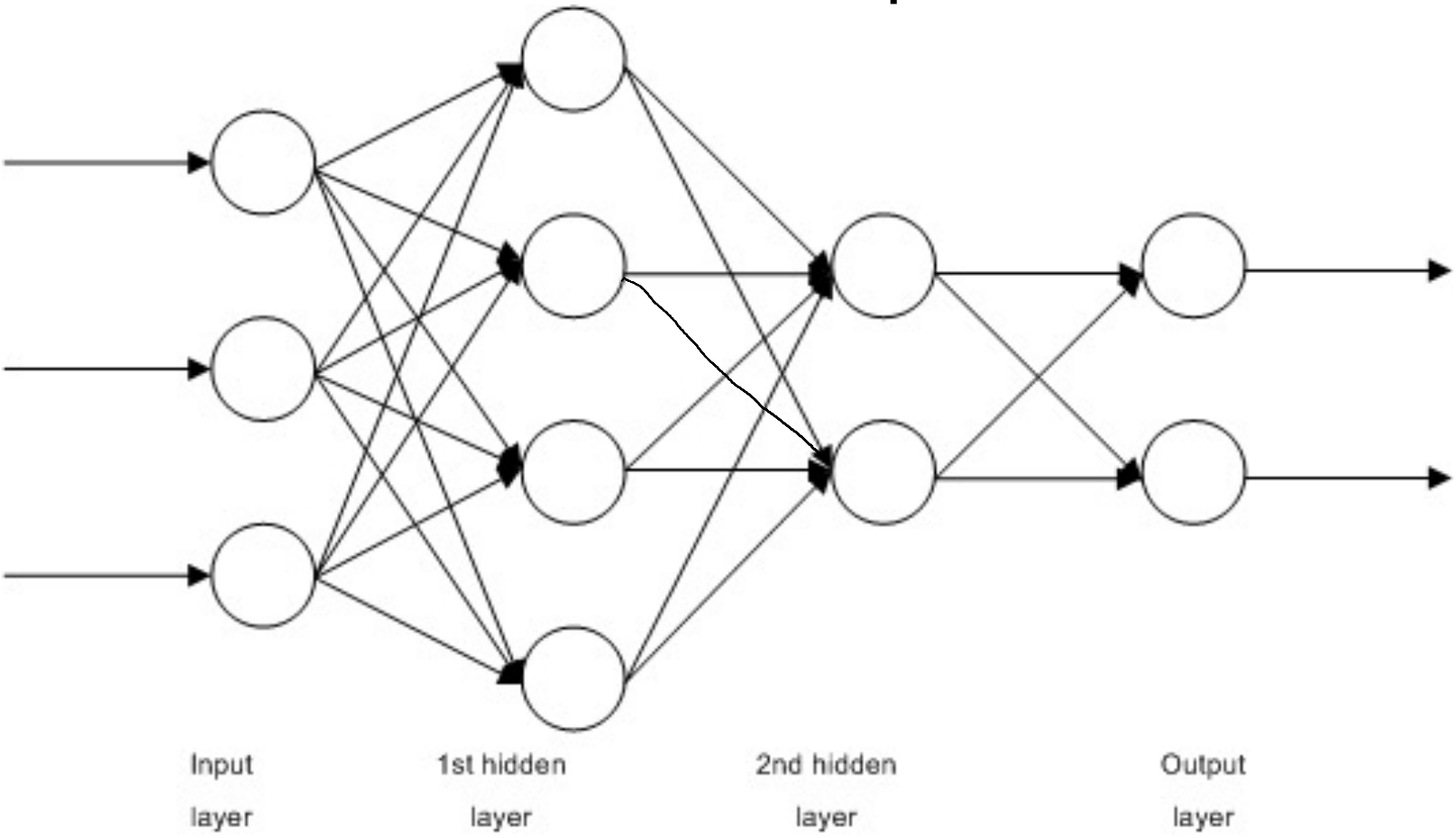
$$\hat{y} = g(\theta_0 + \boldsymbol{X}^T\boldsymbol{\theta})$$

- Example: sigmoid function

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Nodes of a network are arranged in layers to form deep networks



WikiMedia

Network depth creates a composition of functions

a higher level function that takes input as its lower level function

Deep learning has the tendency to break the data down into different representations at different layers.

Input is presented at the visible layer and the hidden layers extract increasingly abstract features which are then used, for example, to classify

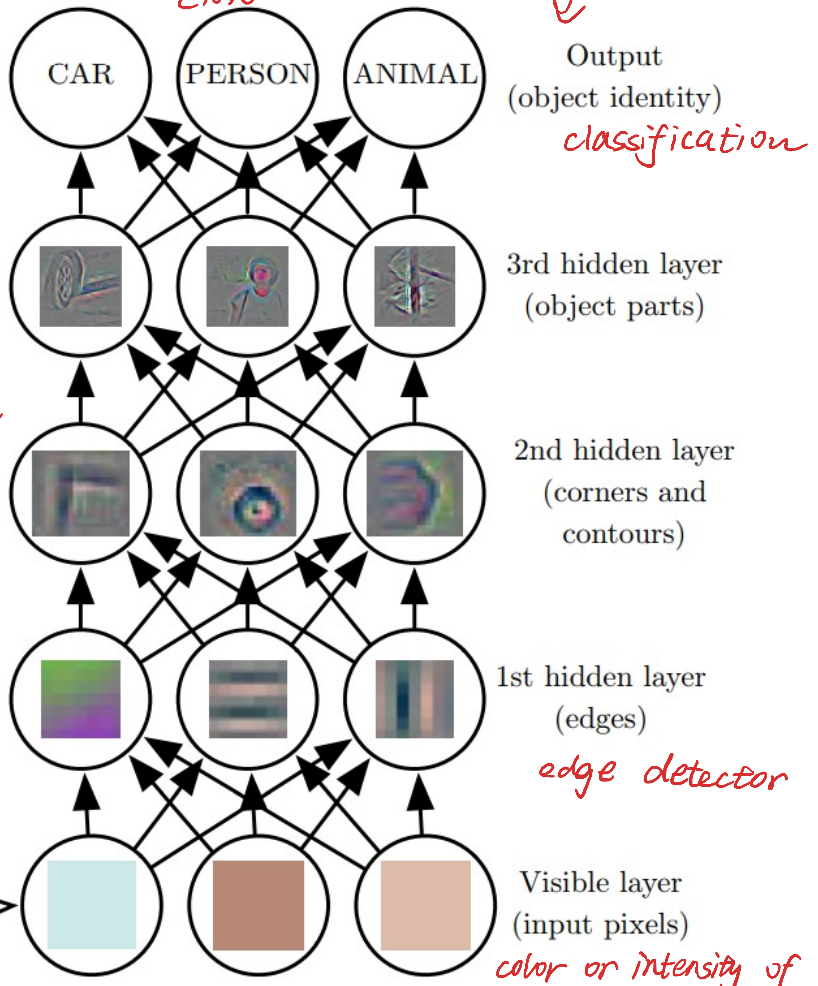
output of CNN

image is good for conceptualize this

granularity

low

high



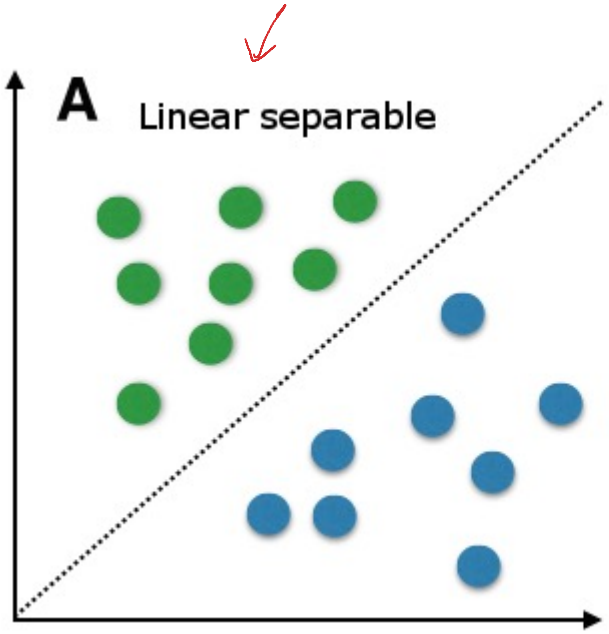
Goodfellow et al., 2016

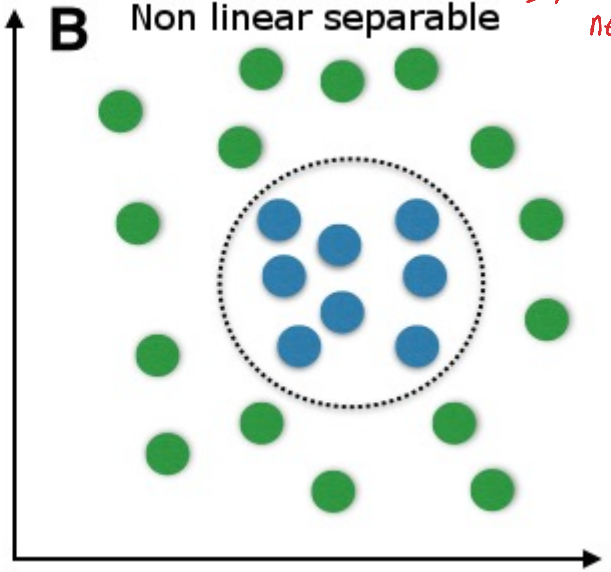
Why go deep?

Depth = complexity

deeper ⇒ more complex decision boundary

Single layer perceptrons can only handle linear separability





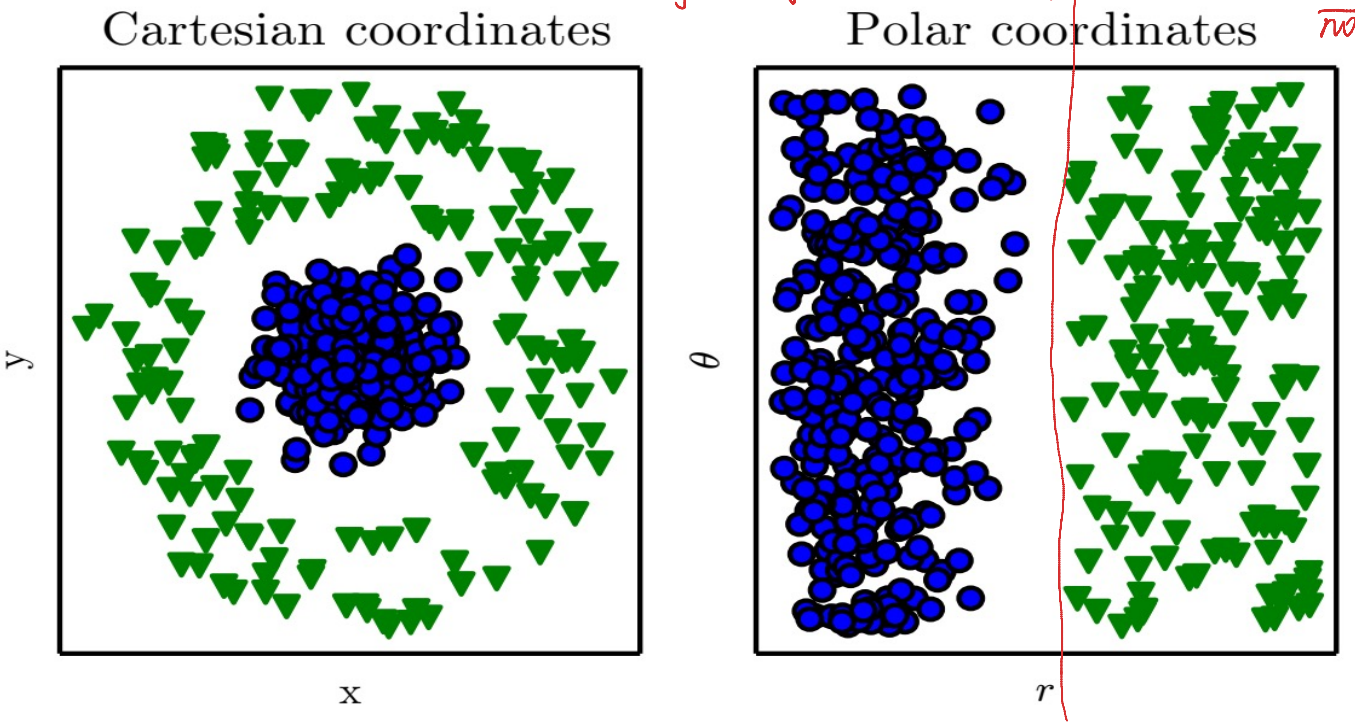
to separate need more layers

Representations matter

every layer of NN is learning a representation, this learning is slowly evolving in a specific way
get rid of info irrelevant for decision making. people track info flow from input layer to output layer
middle layer is more concerned with faithfully transform

- A major reason is that how the data is represented is important. On the left, if you were tasked with using a straight line to separate the two types of data, you would fail. On the right though, by simply transforming the data into a different coordinate system, it becomes very easy to do.

input, so they have lots of mutual info with input, at some point, there are a phase more interested in having a high mutual info with output, it turns out they are both important (maintain info from input and info useful for predicting output)
final layer / decision layer
non-linear transformation / dim reduction



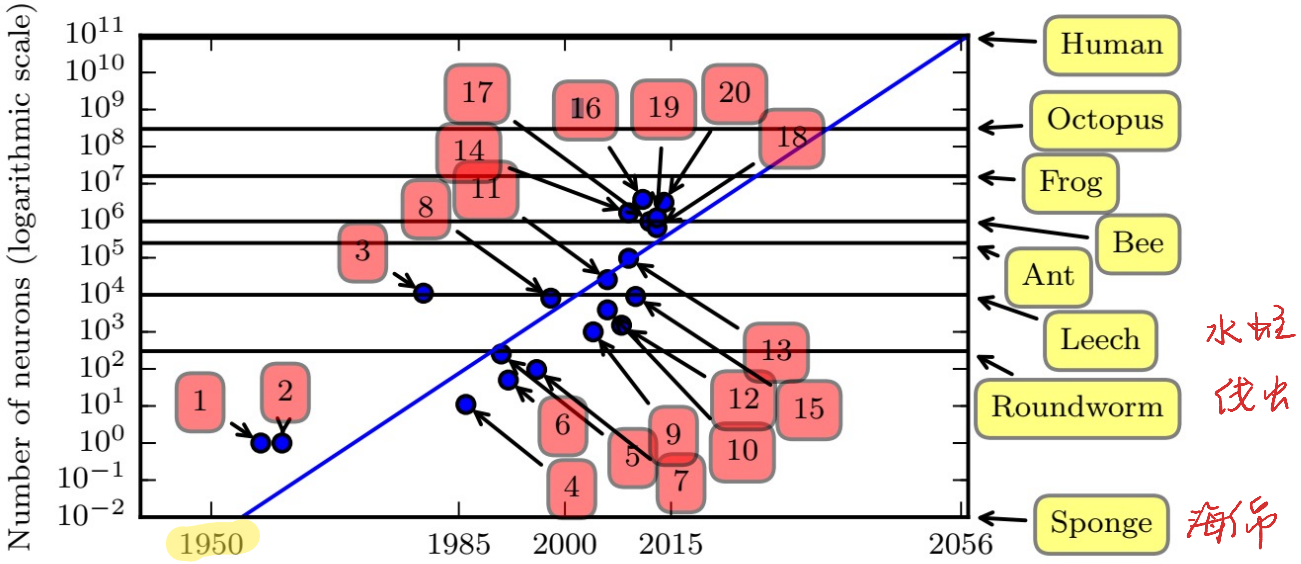
NN can also learn data manifold

Goodfellow et al., 2016

Increasing # of neurons

trend of NN

Neural network size is increasing over time.
Note that we're still below the level of a frog.



水蛭

伐虫

海绵

1. Perceptron (Rosenblatt, 1958)
4. Early backpropagation network
6. MLP for speech recognition (Bengio et al, 1991)
11. GPU-accelerated convolutional network (Challeapilla et al., 2006)

hardware development accelerate NN development

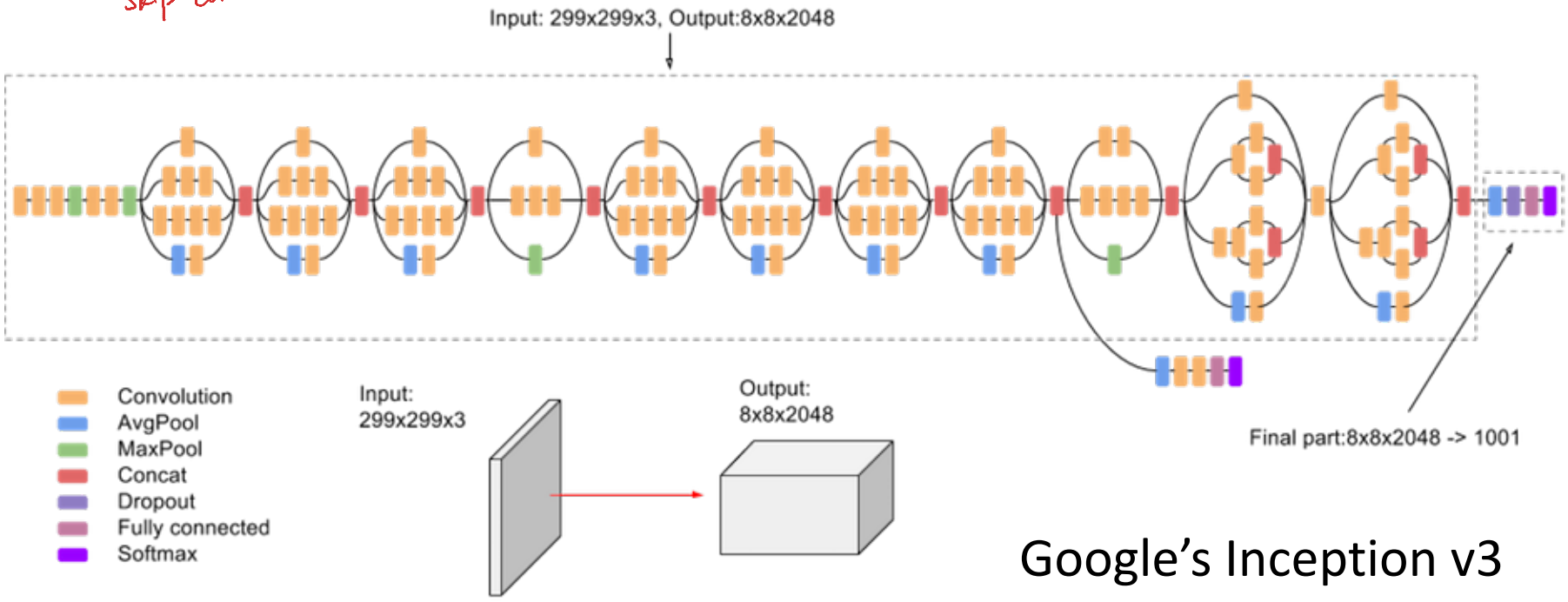
20. GoogLeNet (Szegedy et al., 2014a)

Goodfellow et al., 2016

Layers can be arranged in arbitrary configurations to increase model power

CNN has more restricted connectivity than FC NN, help for training

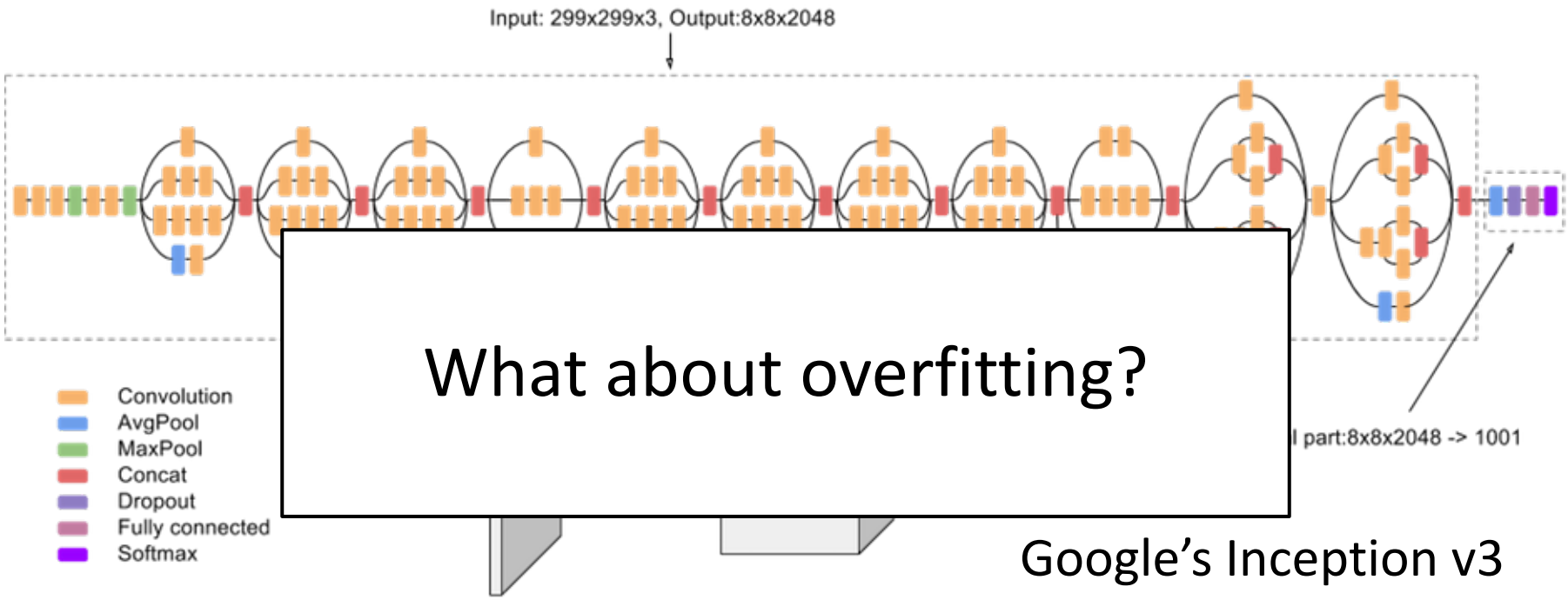
skip connection



Google's Inception v3 Image Recognition network has 24M parameters!

Google

Layers can be arranged in arbitrary configurations to increase model power



Google's Inception v3 Image Recognition network has 24M parameters!

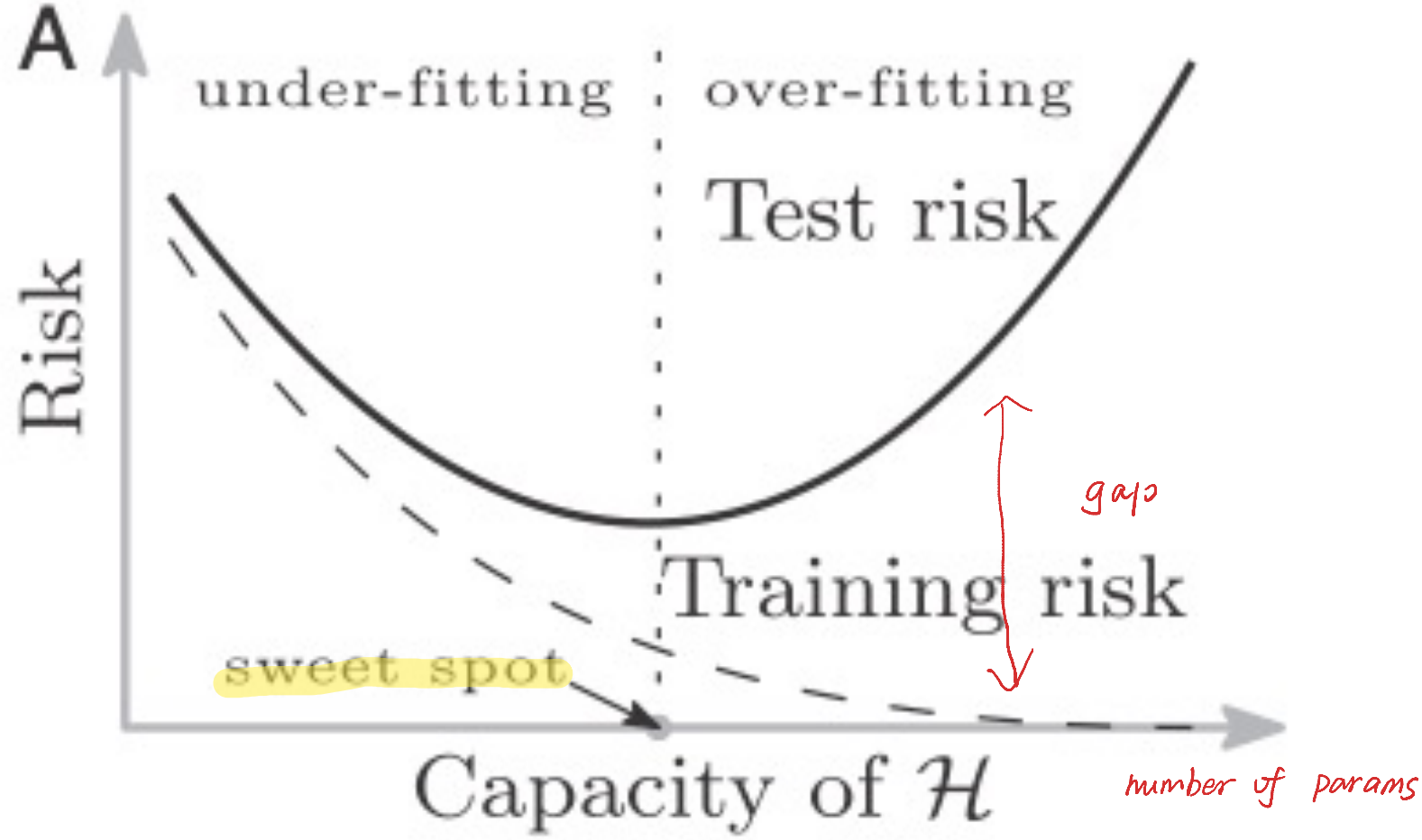
Google

Classic risk curve

statistical inference

over-fitting : a gap between training risk and test risk

sweet-spot: where both training and test error are low

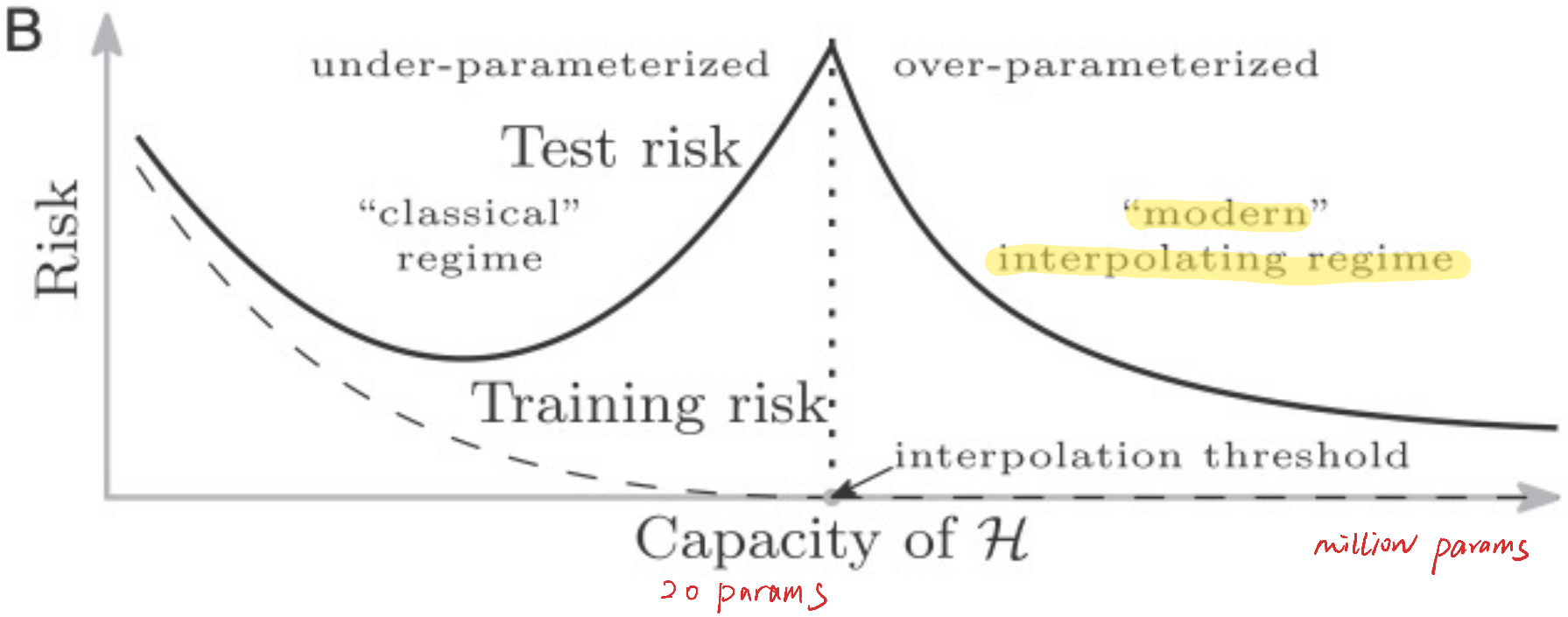


Belkin et al. PNAS (2019)

Double Descent

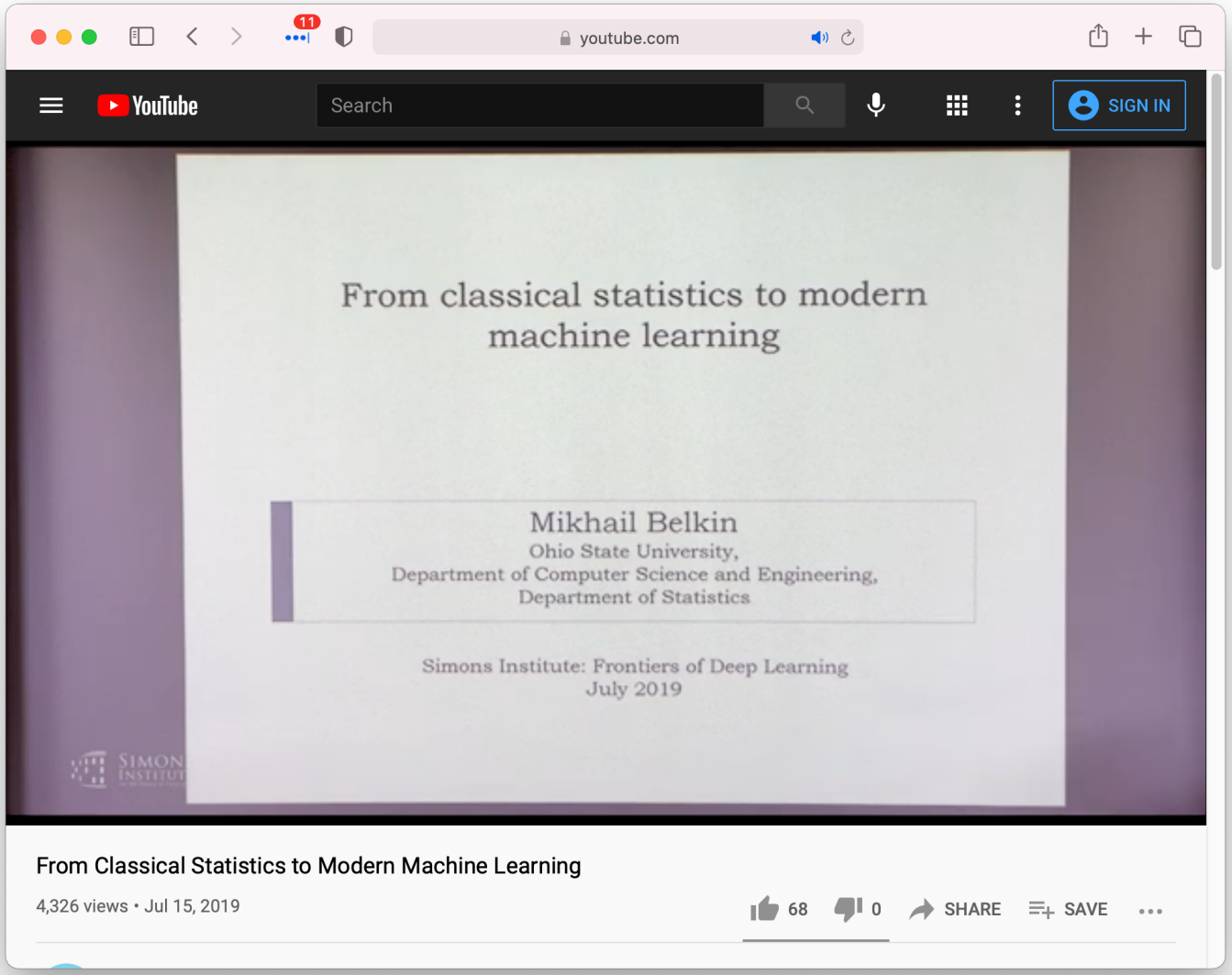
reason for no overfitting with deep NN in DL

smoother functions are more generalizable



Belkin et al. PNAS (2019)

Double Descent



https://www.youtube.com/watch?v=OBCciGnOJVs&feature=youtu.be

Differentiable Computing

微积分计算

entire paradigm of DL

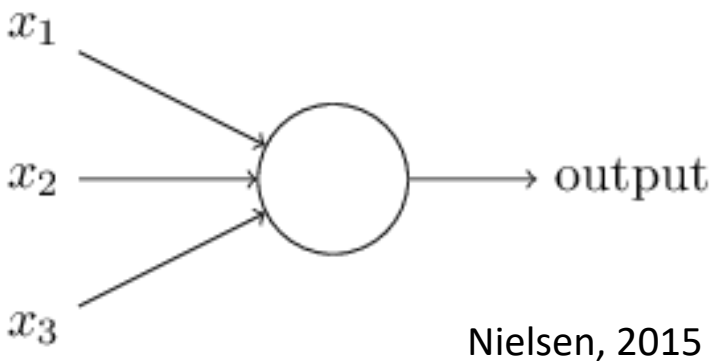
NN should be differentiable

The important point is that people are now building a new kind of software by assembling networks of parameterized functional blocks and by training them from examples using some form of gradient-based optimization….It's really very much like a regular program, except it's parameterized, automatically differentiated, and trainable/optimizable.

- *Yann LeCun, Director of FAIR*

The perceptron

- Developed in 1950's and 1960's by Frank Rosenblatt
- Binary inputs
- Single binary output
- Example:

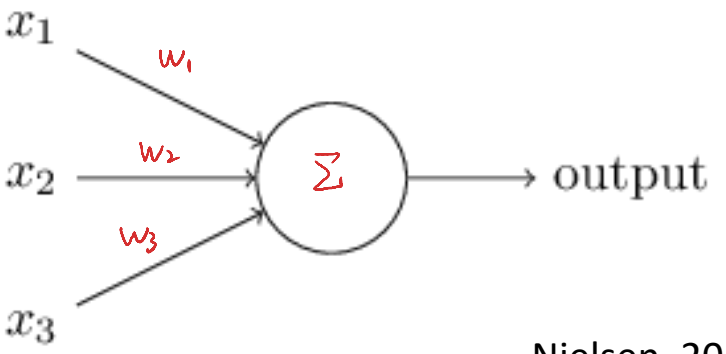


Nielsen, 2015

In general could have more or fewer inputs.

Mention that figures come from Michael Nielsen's book

The perceptron

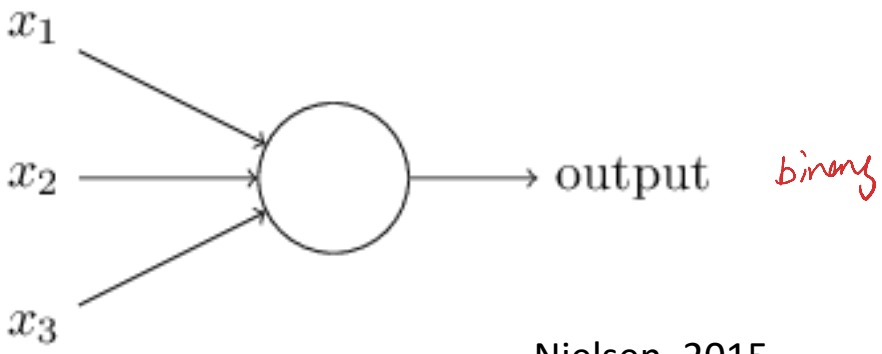


Nielsen, 2015

- Computing the output:
  - Assign weights to each input
  - Determine if weighted sum of inputs is greater than some threshold

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

weight How much toggle these inputs have effect on output?
拨动开关



binary

Nielsen, 2015

- Example: Decide whether to attend a cheese festival
- Three factors: binary features
  1. Is the weather good? x_1
  2. Does your friend want to accompany you? x_2
  3. Is the festival near public transit? (you don't own a car) x_3

$$x_j = \begin{cases} 0, \text{if no} \\ 1, \text{if yes} \end{cases}$$

- Example: Decide whether to attend a cheese festival
- Three factors:
  1. Is the weather good? x_1
  2. Does your friend want to accompany you? x_2
  3. Is the festival near public transit? (you don't own a car) x_3
- **Case 1**: Love cheese but hate bad weather
  - $w_1 = 6$
  - $w_2 = 2$
  - $w_3 = 2$
  - Threshold= 5
  - $\sum_j w_j x_j \geq 6 >$ threshold whenever weather is **good** $(x_1 = 1)$
  - $\sum_j w_j x_j \leq 2 + 2 = 4 <$ threshold whenever weather is **bad** $(x_1 = 0)$

The large value for w_1 indicates that the weather matters a lot more than whether bf or gf attend or nearness of public transit

$4 < \underset{5}{\text{threshold}} < 6$

- Example: Decide whether to attend a cheese festival
- Three factors:
  1. Is the weather good? x_1
  2. Does your friend want to accompany you? x_2
  3. Is the festival near public transit? (you don't own a car) x_3
- **Case 2**: Love cheese but don't hate bad weather as much
  - $w_1 = 6$
  - $w_2 = 2$
  - $w_3 = 2$
  - Threshold= 3
  - $\sum_j w_j x_j >$ threshold whenever weather is **good** ($x_1 = 1$) *or* friend will go ($x_2 = 1$) *and* when the festival is near public transit ($x_3 = 1$)

Dropping the threshold means you're more willing to go to the festival

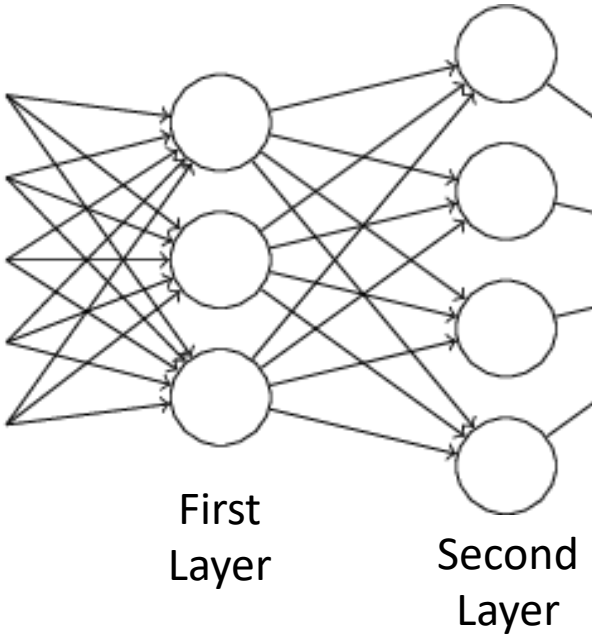
Summary so far

- Neural networks are collections of individual neurons or nodes that multiple input values by weights and apply a non-linearity
- Layers of nodes allow for increased complexity of calculations (functions of functions)
- Neural networks can theoretically approximate any function (given appropriate size)
- Creative design of network architecture admits many practical applications across domains

The multilayer perceptron (MLP)

- A single perceptron is pretty simple
- A complex network of perceptrons can make subtle decisions

  - In this network, the first column of perceptrons - what we'll call the first *layer* of perceptrons - is making three very simple decisions, by weighing the input evidence.
  - What about the perceptrons in the second layer? Each of those perceptrons is making a decision by weighing up the results from the first layer of decision-making. In this way a perceptron in the second layer can make a decision at a more complex and abstract level than perceptrons in the first layer.
  - And even more complex decisions can be made by the perceptron in the third layer.
  - In this way, a many-layer network of perceptrons can engage in sophisticated decision making.



Nielsen, 2015

Bias

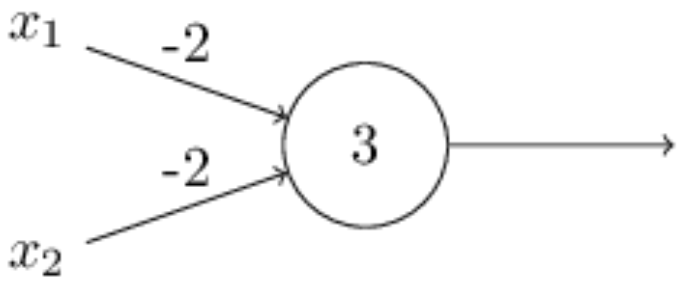
- $w \cdot x = \sum_j w_j x_j$
  - w and x are the weight and input vectors, respectively
- Replace the threshold with perceptron *bias*
  - Bias $b = -$threshold

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

- Bias is a measure of ease in *firing* the perceptron

Logic circuits with perceptrons

- $w_1, w_2 = -2, b = 3$



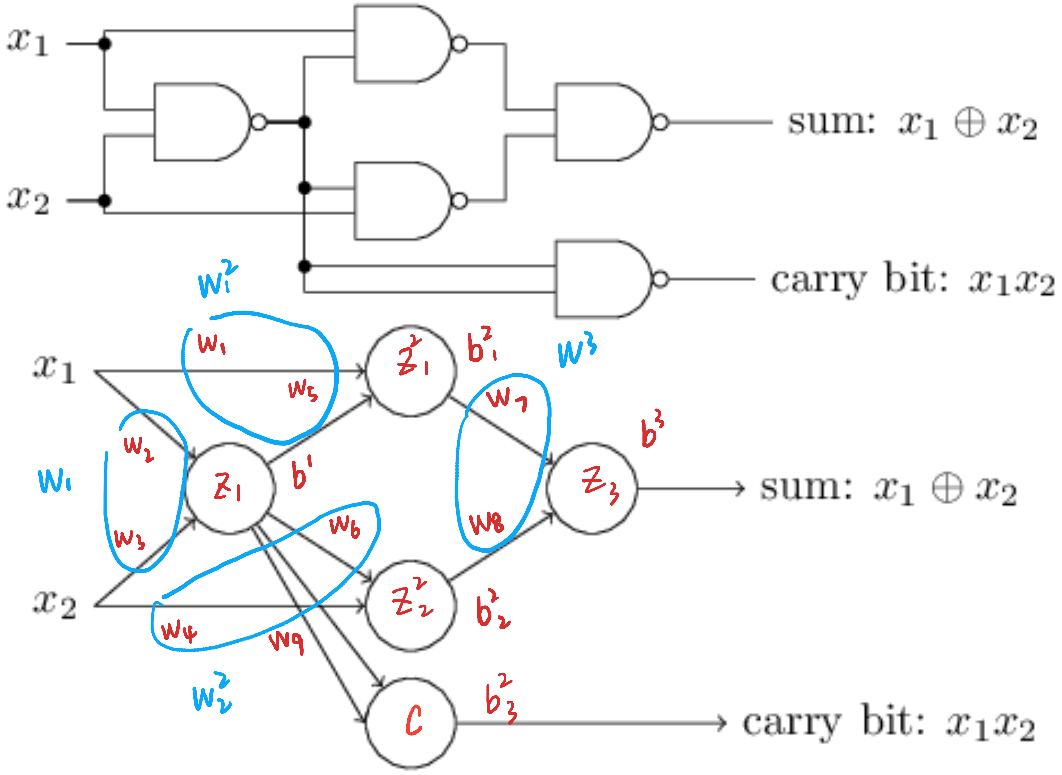
Nielsen, 2015

- What is the output of this perceptron for each possible input?
- What logic circuit is this?
- Input 00 produces 1
- Input 01 or 10 produce 1
- Input 11 produces 0
- This is a NAND gate!

} ⇒ 1

Logic circuits with perceptrons

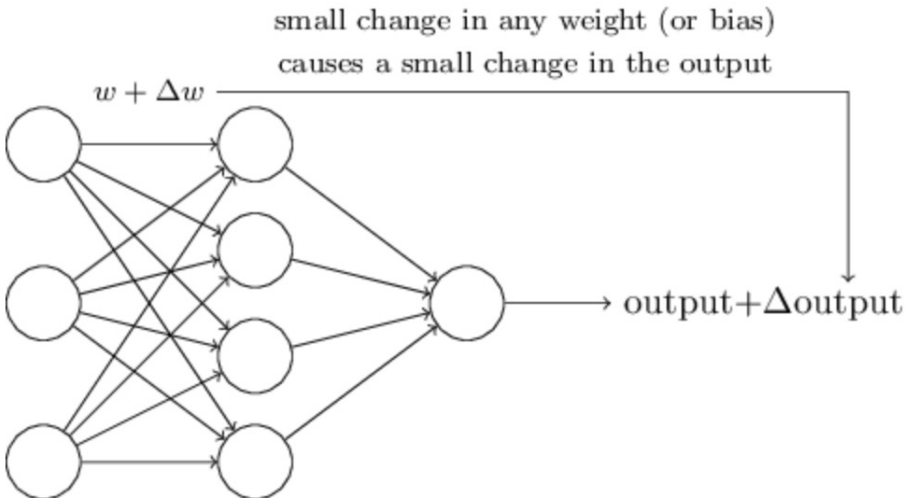
- NAND gates are universal for computation
  - Any computation can be built from NAND gates
  - Therefore, perceptrons are universal for computation
- Bitwise addition:



Nielsen, 2015

Problem with Logical Functions

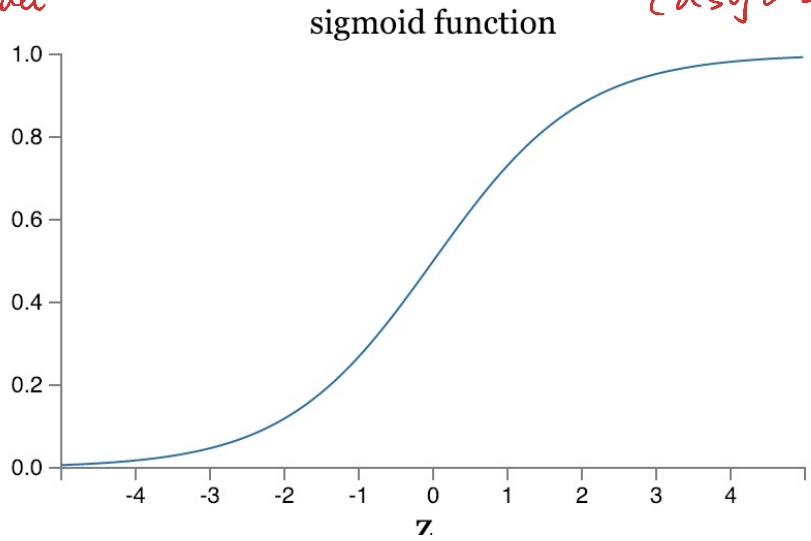
- Hard to "tune" in a traditional sense ⇒ not good at learning
- Small change in weight can lead to large changes in conditions (or on change at all)
  - Think the volume knob in your car behaving this way
- For this to happen we need neurons to perform a continuous *differentiable* function



Sigmoidal Neuron

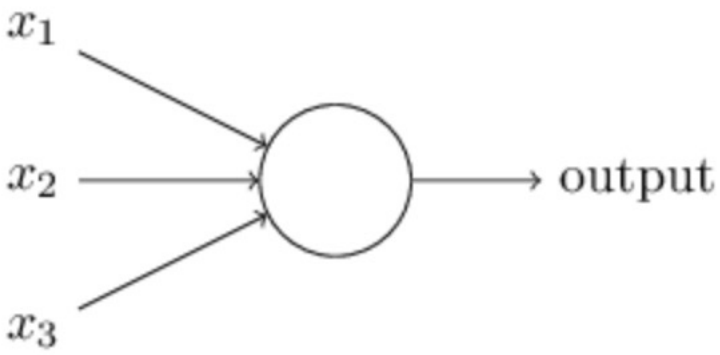
why current NN don't use sigmoid ?
gradients will get to 0 when $z \to 0$ or $z \to 1$
gradient vanishing problem
use ReLU instead

(a soft step function)



continuous value $\in (0,1)$

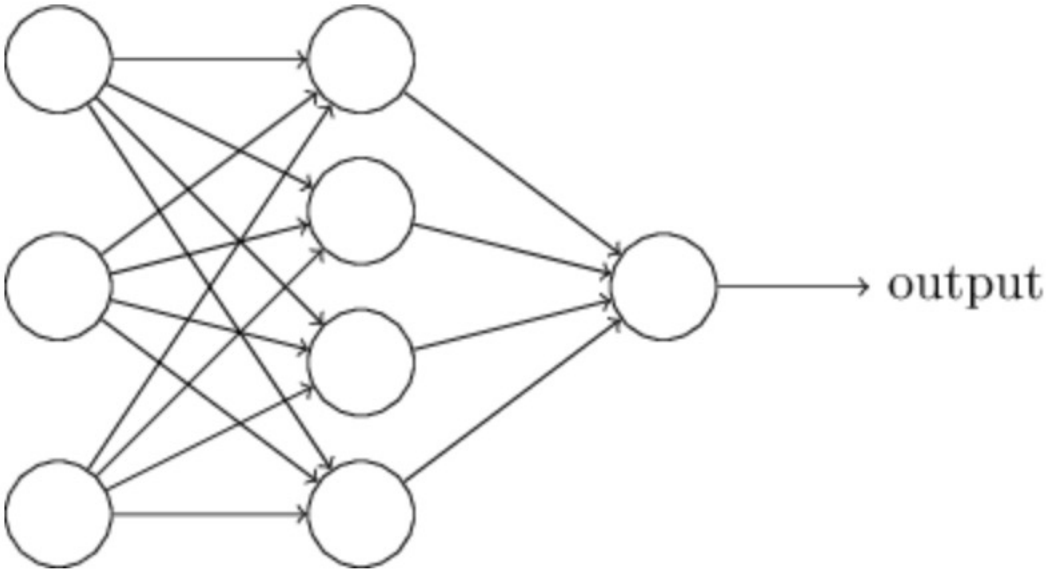
don't need to be binary can be continous



$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

– output

Multi-layers of Neurons

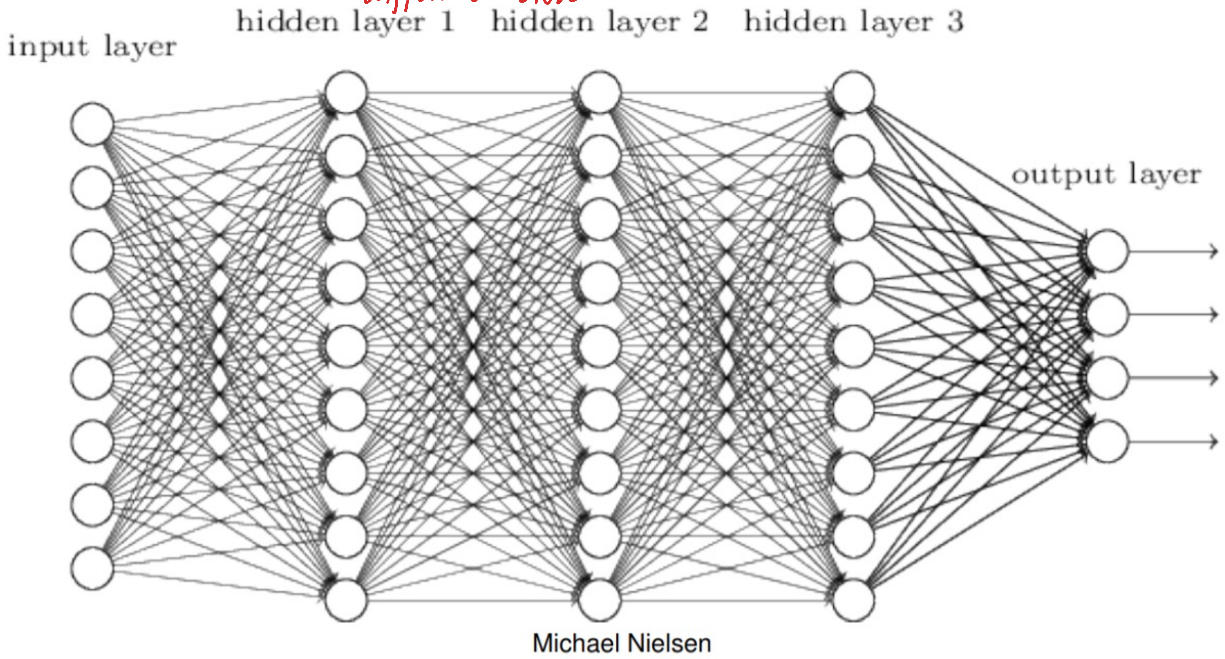


Differentiable Computing

We already know how to make NAND gates, so this is hardly big

- We can create *learning algorithms* that automatically tune the weights and biases with *Gradient Descent*
  - Tuning occurs in response to external stimuli and w/o direct intervention
  - Creates a circuit designed for the problem at hand

chain rule can speed up as a result of backpropagation differentiation



Michael Nielsen

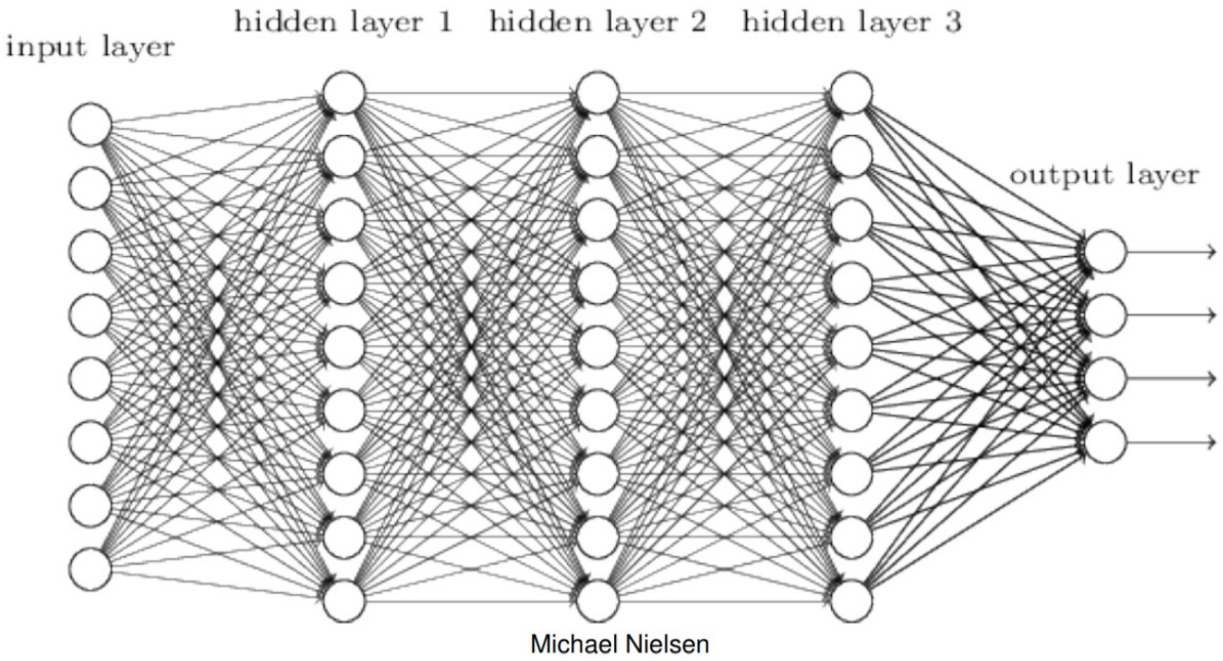
Design choices for an ANN

- Activation function (e.g. threshold)
- Cost functions
- Number and dimension of layers
- Connections between layers
- Regularizations
  - Layers
  - Batches
- More…

- You can imagine that we can solve a rich set of problems by various design choices.
- Emphasize that we can have nonlinear functions and also continuous inputs and outputs instead of binary.
- We will cover all of these in detail throughout the course and is a large focus of the beginning

Fully connected network

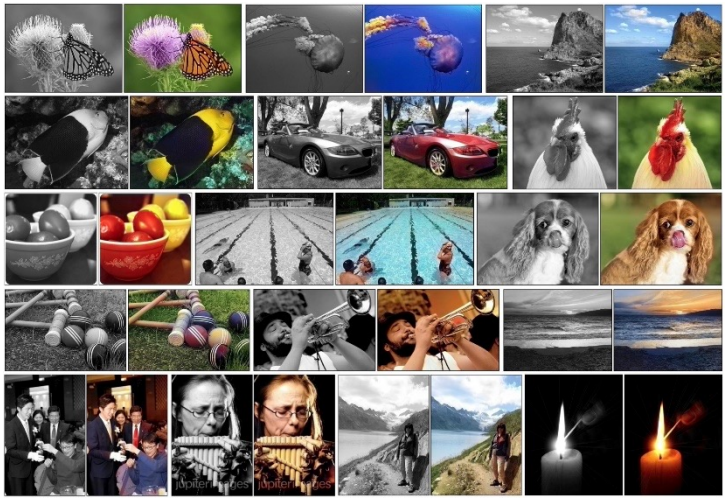
- Every feature interacts with every other feature
- Weight matrix at every level allowed to be dense

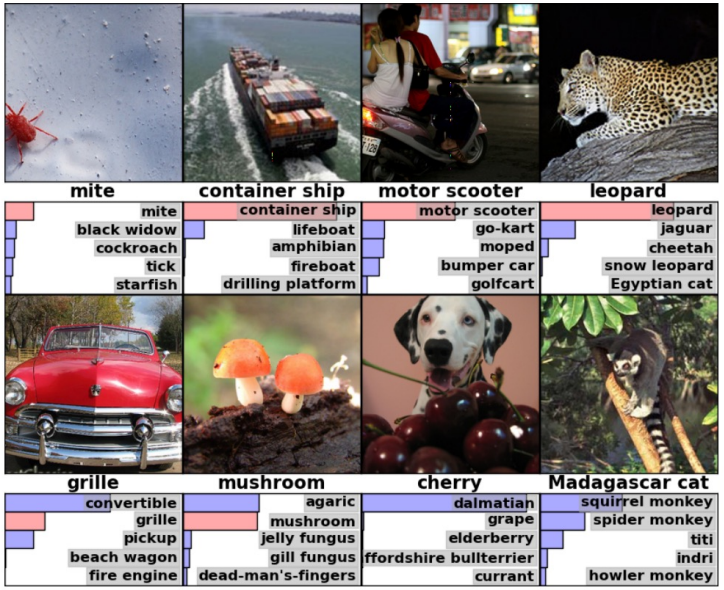


Types of Neural Networks

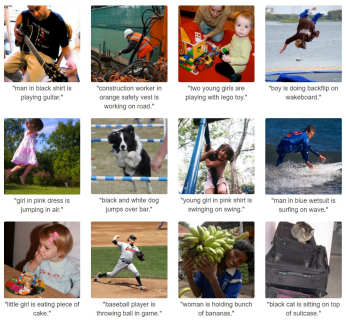
Convolutional Neural Networks

- Very successful in images



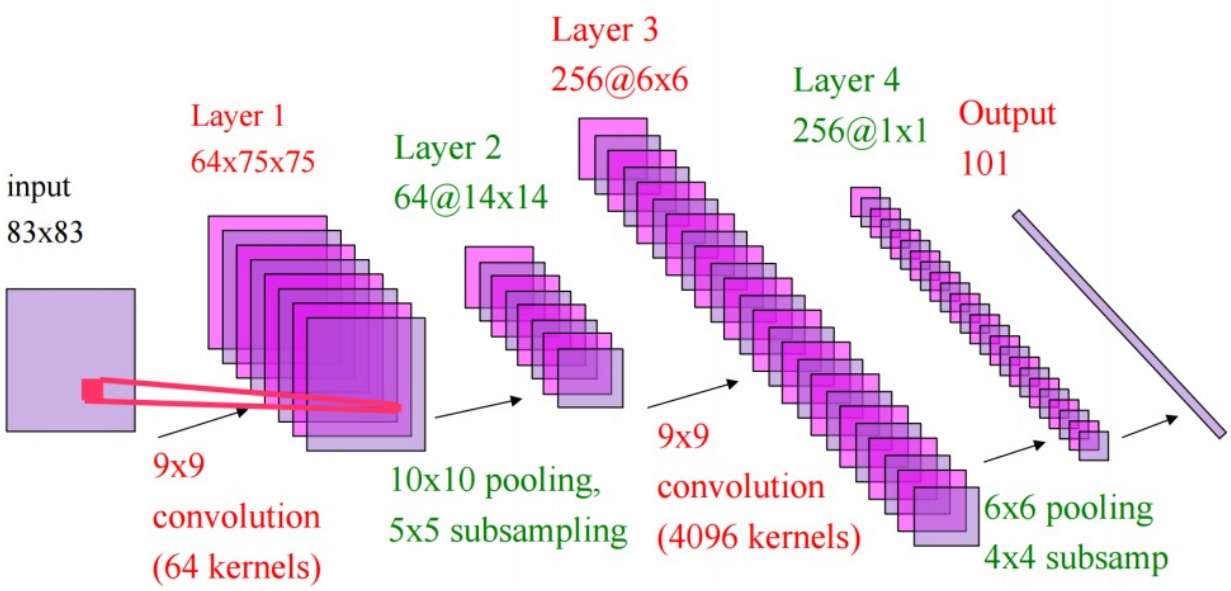


Emphasize that CNNs were used for object detection in bottom left



Convolutional Neural Networks

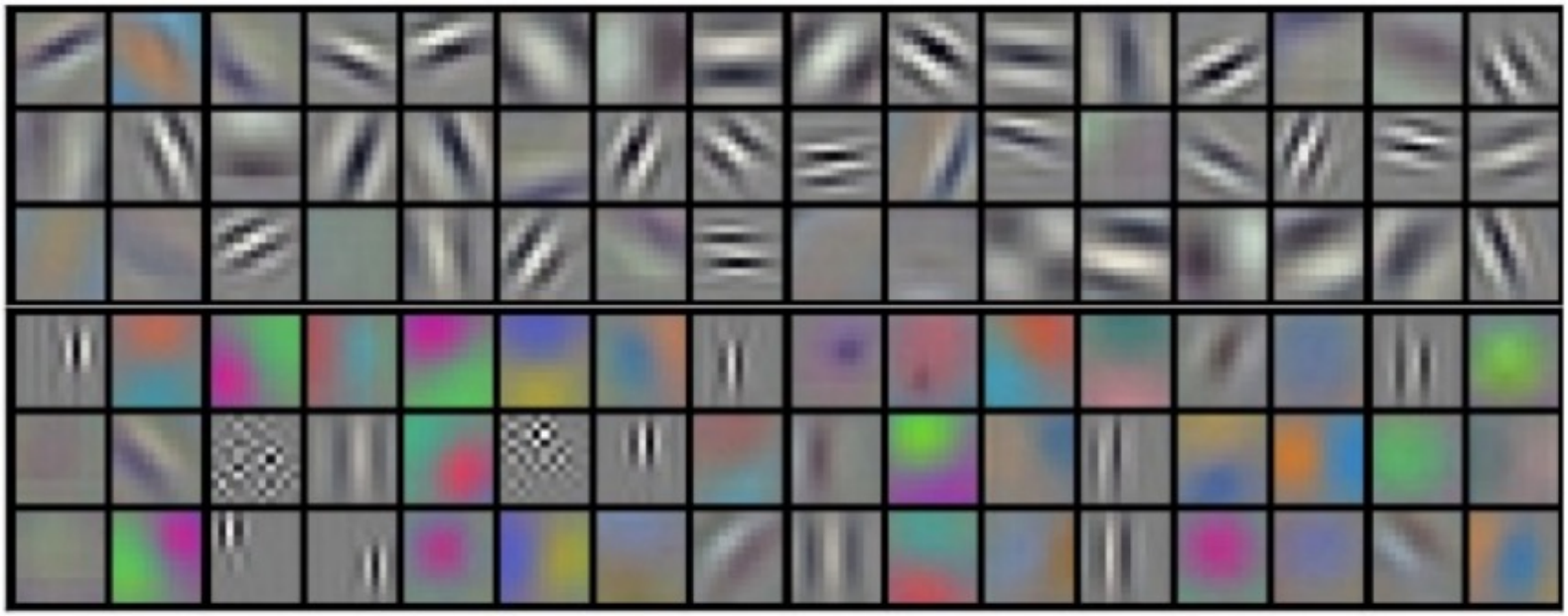
- Only pixels that are close to each other in the image interact with each other (convolution layer)
- Weight matrices are highly structured
- "Pooling" helps to simplify output of convolution layer



Yann LeCun

Convolutional Neural Networks

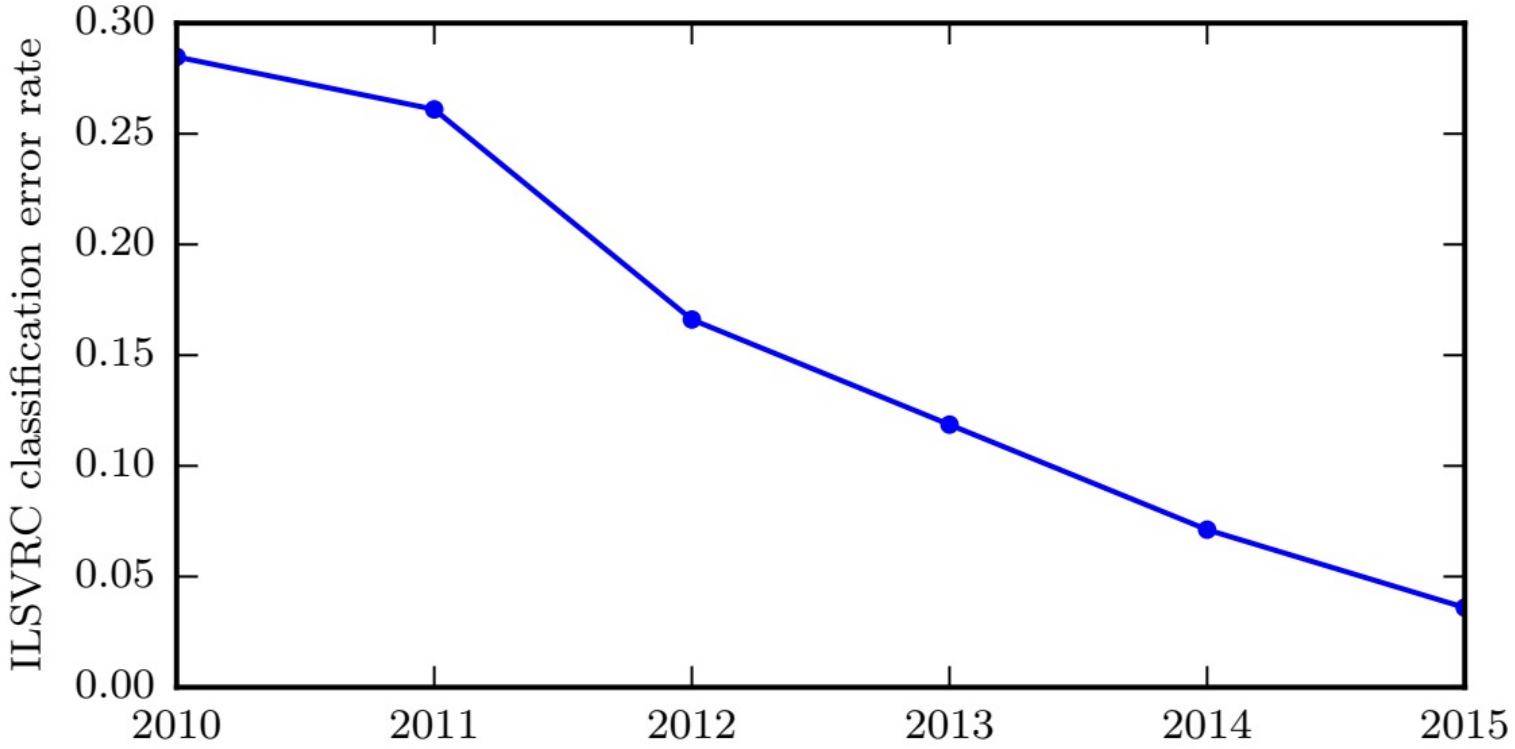
- Weights from the first layer tend to look like directional filters after training
  - Detects edges, color change, etc.



CS 231n, Karpathy

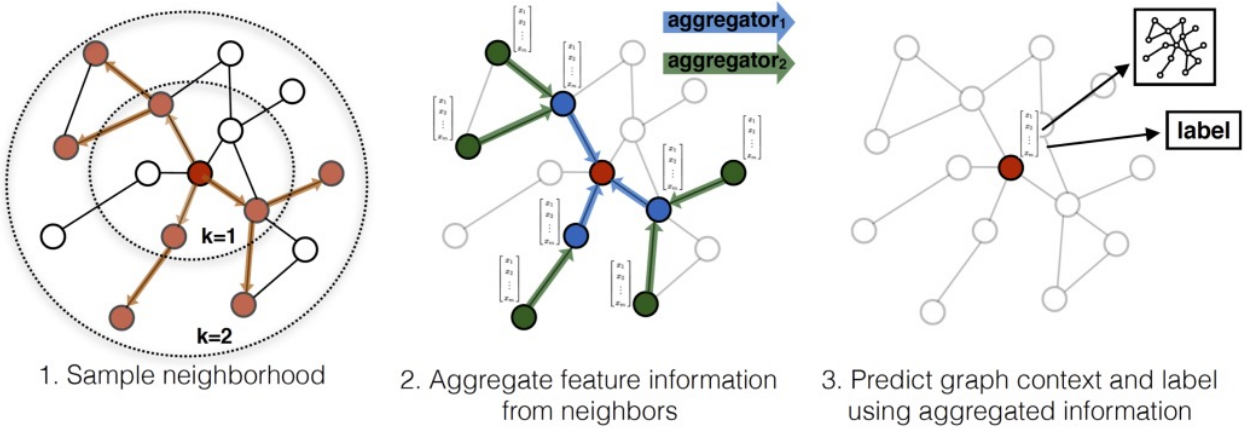
Convolutional Neural Networks (CNNs)

deep neural networks have led to great success in image classification every year, largely due variations on CNNs



Goodfellow et al., 2016

Graph Neural Networks



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

[Hamilton et. Al]

Recurrent Neural Networks (RNNs)

sequential input language, EEG, DNA Seq

- Useful when time is important

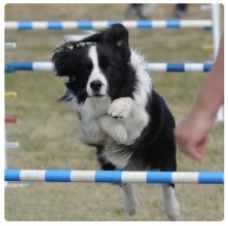
"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."



"black and white dog jumps over bar."

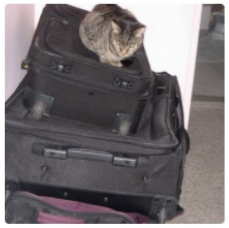
"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

"little girl is eating piece of cake."

"baseball player is throwing ball in game."

"woman is holding bunch of bananas."



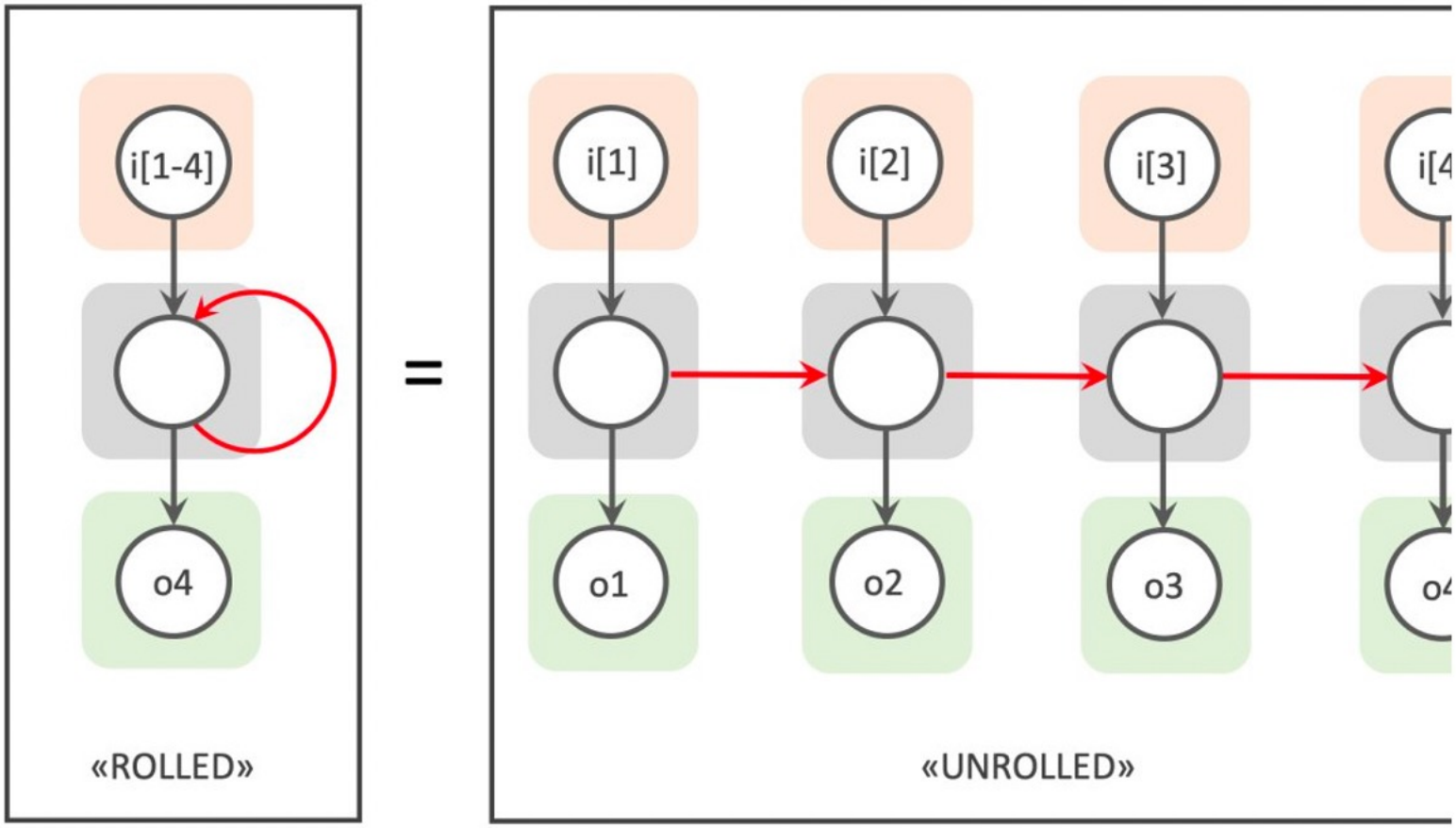
"black cat is sitting on top of suitcase."

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

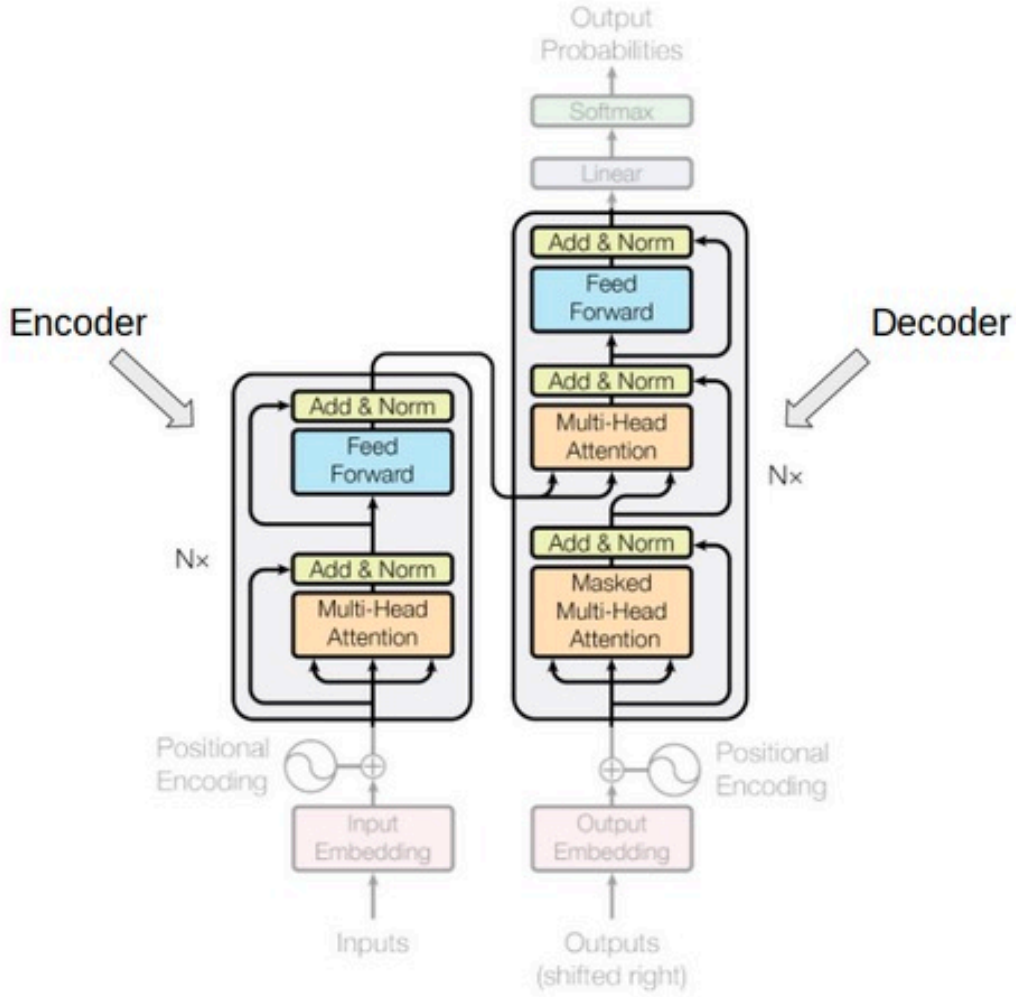
Recurrent Neural Networks (RNNs)

- In feedforward nets (everything we've considered so far), activations of later layers are completely determined by the input
- RNNs allow the hidden layers to be affected by activations at earlier times (i.e. feedback)
  - E.g. a neuron's activation may include as input its activation at an earlier time
  - Cycles are now included in the network
- This time-varying behavior make RNNs useful for analyzing data that change over time (e.g. speech)
- Training can be difficult for long-term dependencies

Recurrent Neural Network



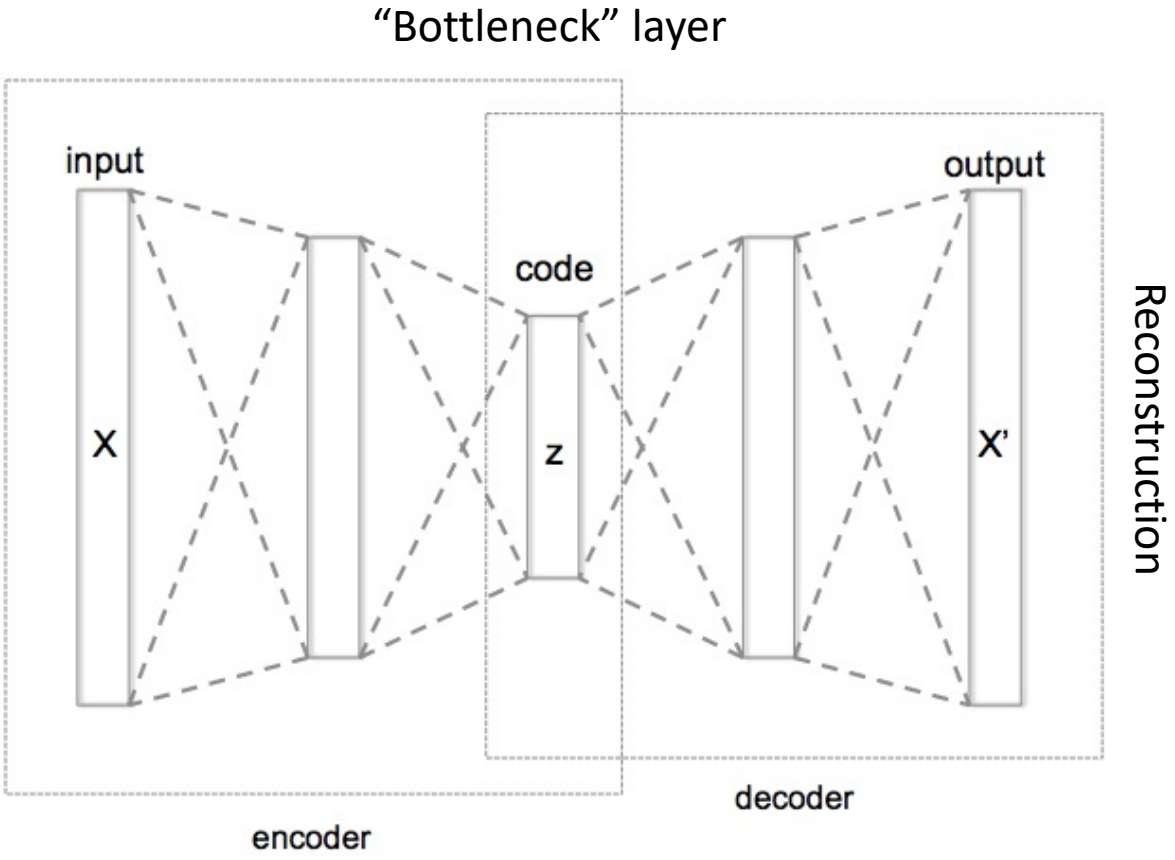
Transformers



Autoencoders

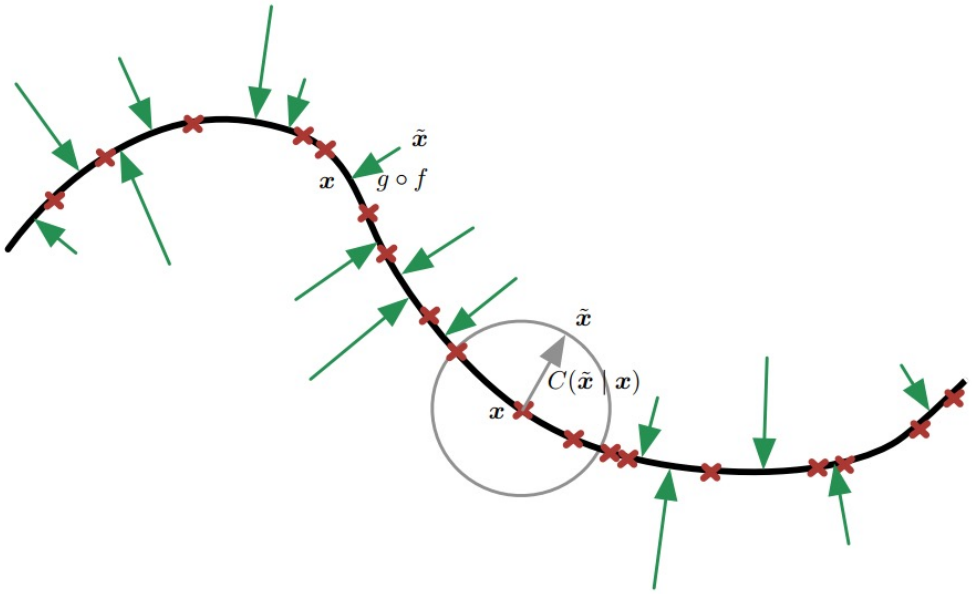
Somewhat similar idea to PCA

- Attempts to compress the data and then reconstruct the input



Autoencoder Applications

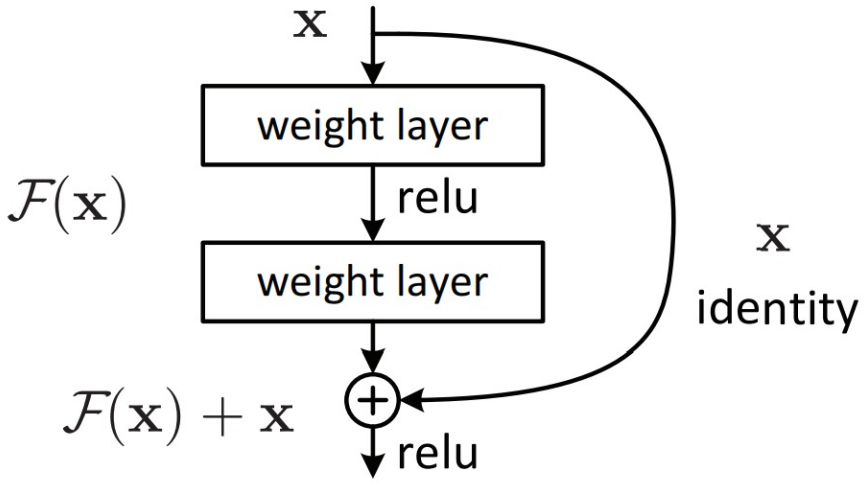
- Pretraining
- Dimensionality reductio
  - Information retrieval
  - Denoising
  - Data compression
- Generative modeling
- Batch correction



Goodfellow et al., 2016

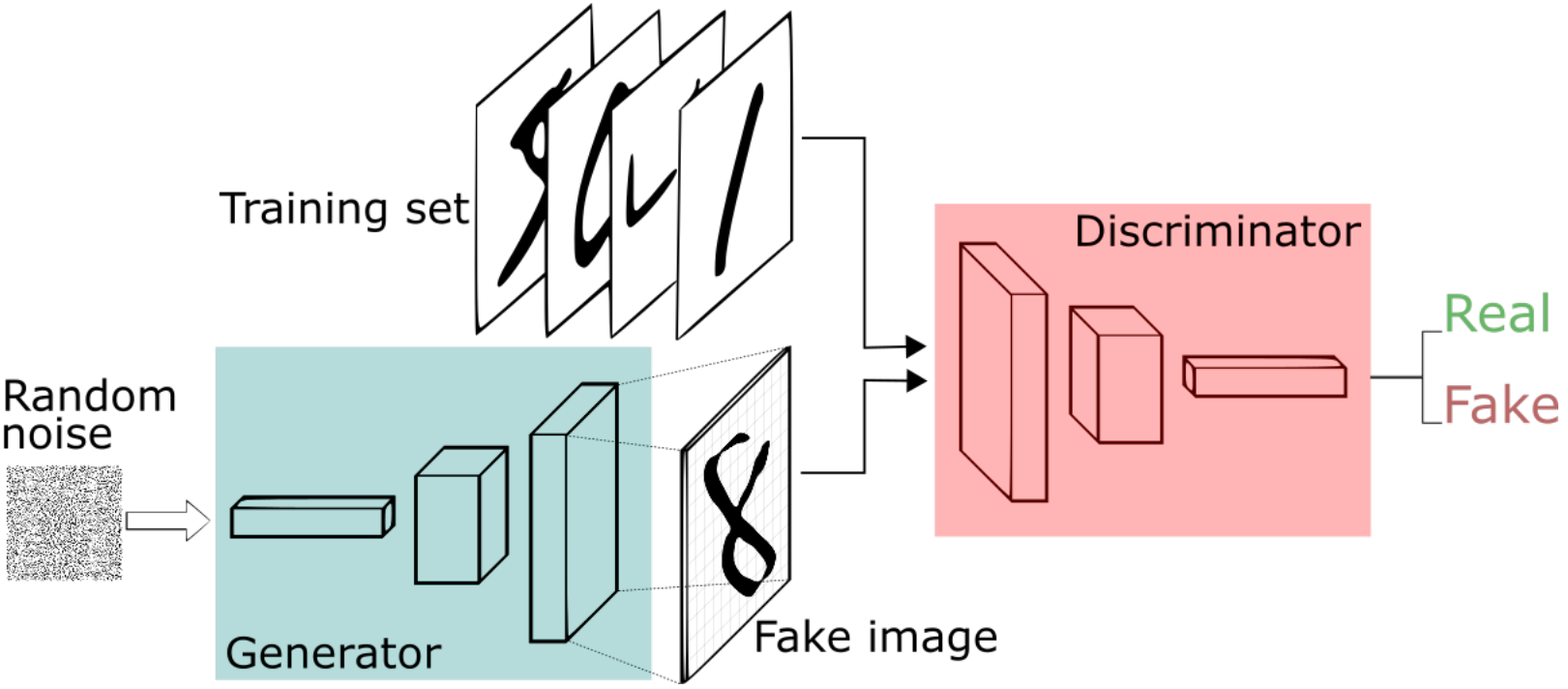
Ultra Deep Learning (e.g. ResNet)

- Very deep neural nets are difficult to train
  - Accuracy can degrade with deeper networks
- ResNet developed a framework to address this degradation
  - Successfully trained a 152 layer network
  - Won the ILSVRC 2015 image classification task



arxiv.org/abs/1512.03385

GANs

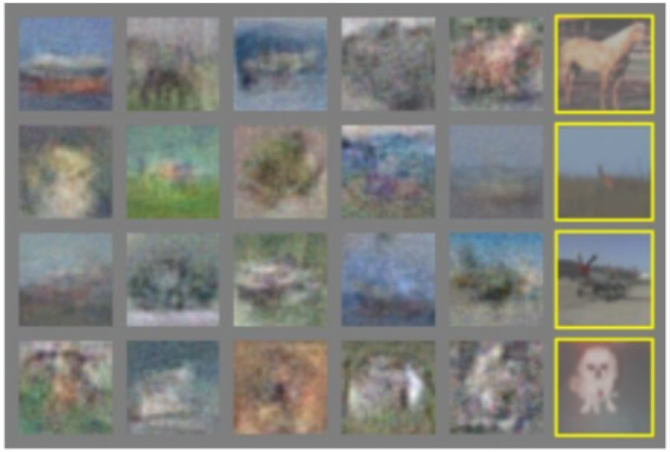

Training set
Discriminator
Real
Fake
Random noise
Generator
Fake image

Generative Models

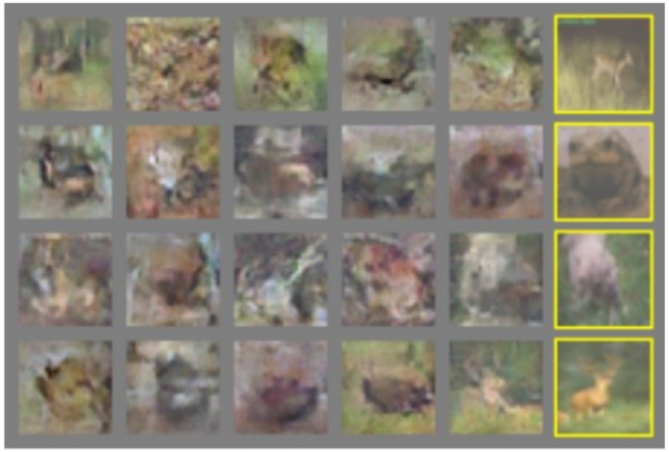
Visualization of samples from the model.
The rightmost column shows the nearest
training example of the column just to the left.

a)

b)



c)



d)

Deep Reinforcement

Alpha Go Zero

nature
International journal of science

Access provided by Yale University

Altmetric: 2188 Citations: 1 More detail »

Article

Mastering the game of Go without human knowledge

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel & Demis Hassabis

Nature **550**, 354–359 (19 October 2017)
doi:10.1038/nature24270
Download Citation

Computational science
Computer science Reward

Received: 07 April 2017
Accepted: 13 September 2017
Published online: 18 October 2017

Alpha Zero

arXiv.org > cs > arXiv:1712.01815

Search or
(Help | Advanc

Computer Science > Artificial Intelligence

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis

(Submitted on 5 Dec 2017)

The game of chess is the most widely-studied domain in the history of artificial intelligence. The strongest programs are based on a combination of sophisticated search techniques, domain-specific adaptations, and handcrafted evaluation functions that have been refined by human experts over several decades. In contrast, the AlphaGo Zero program recently achieved superhuman performance in the game of Go, by tabula rasa reinforcement learning from games of self-play. In this paper, we generalise this approach into a single AlphaZero algorithm that can achieve, tabula rasa, superhuman performance in many challenging domains. Starting from random play, and given no domain knowledge except the game rules, AlphaZero achieved within 24 hours a superhuman level of play in the games of chess and shogi (Japanese chess) as well as Go, and convincingly defeated a world-champion program in each case.

Subjects: **Artificial Intelligence (cs.AI)**; Learning (cs.LG)
Cite as: arXiv:1712.01815 **[cs.AI]**
(or arXiv:1712.01815v1 **[cs.AI]** for this version)

Submission history

From: David Silver [view email]
[v1] Tue, 5 Dec 2017 18:45:38 GMT (272kb,D)

Applications

Genomics: Enhancer Binding

(Alipanahi et al. 2015)

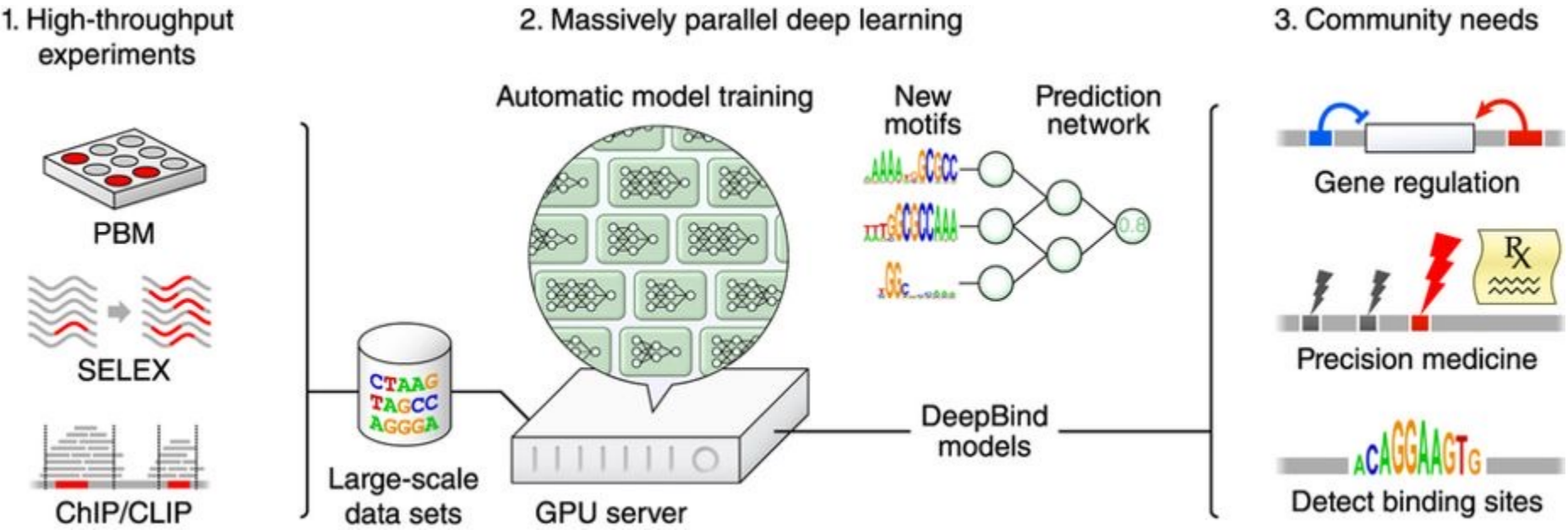
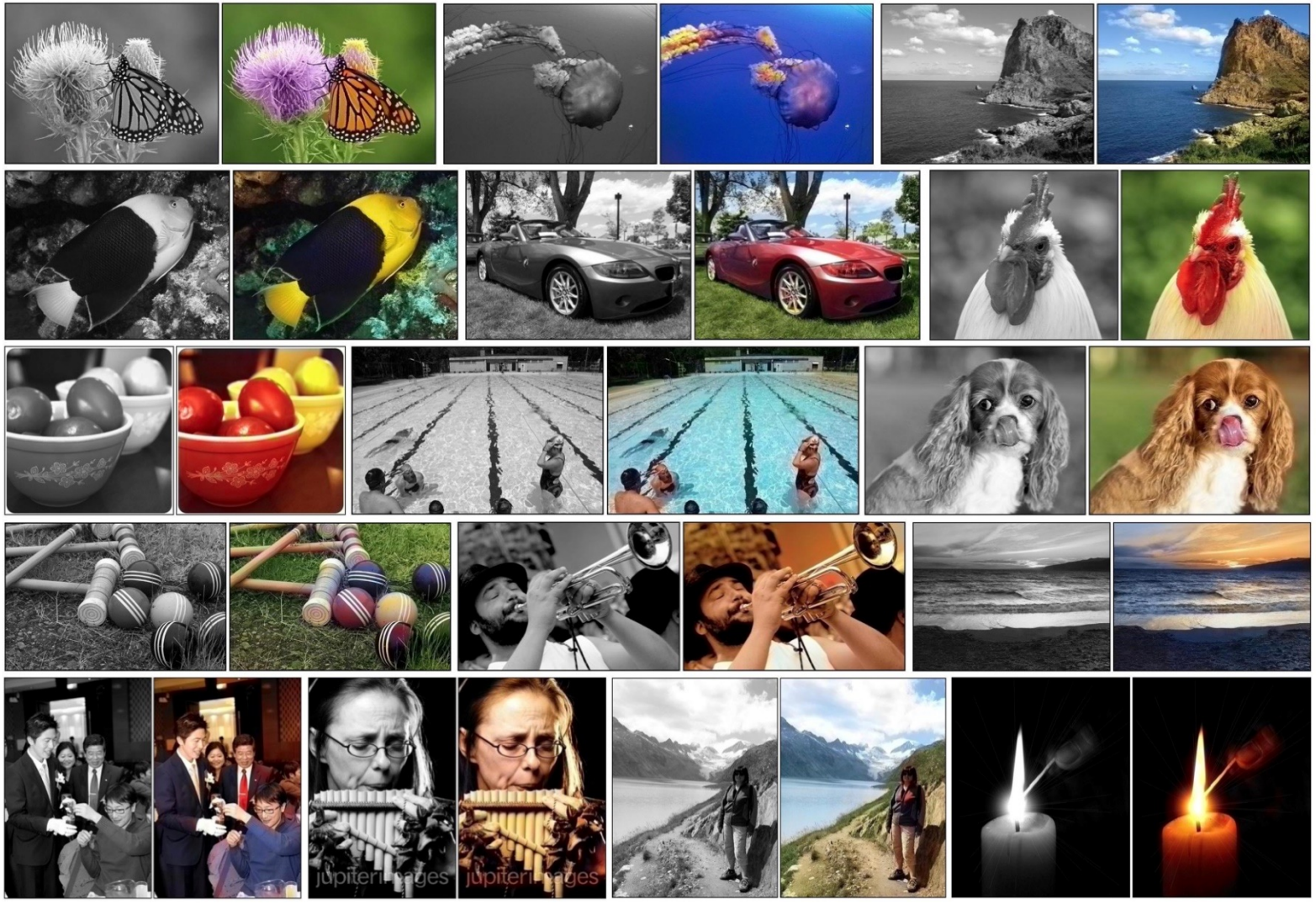


Image Colorization

(Zhang et al., 2016)



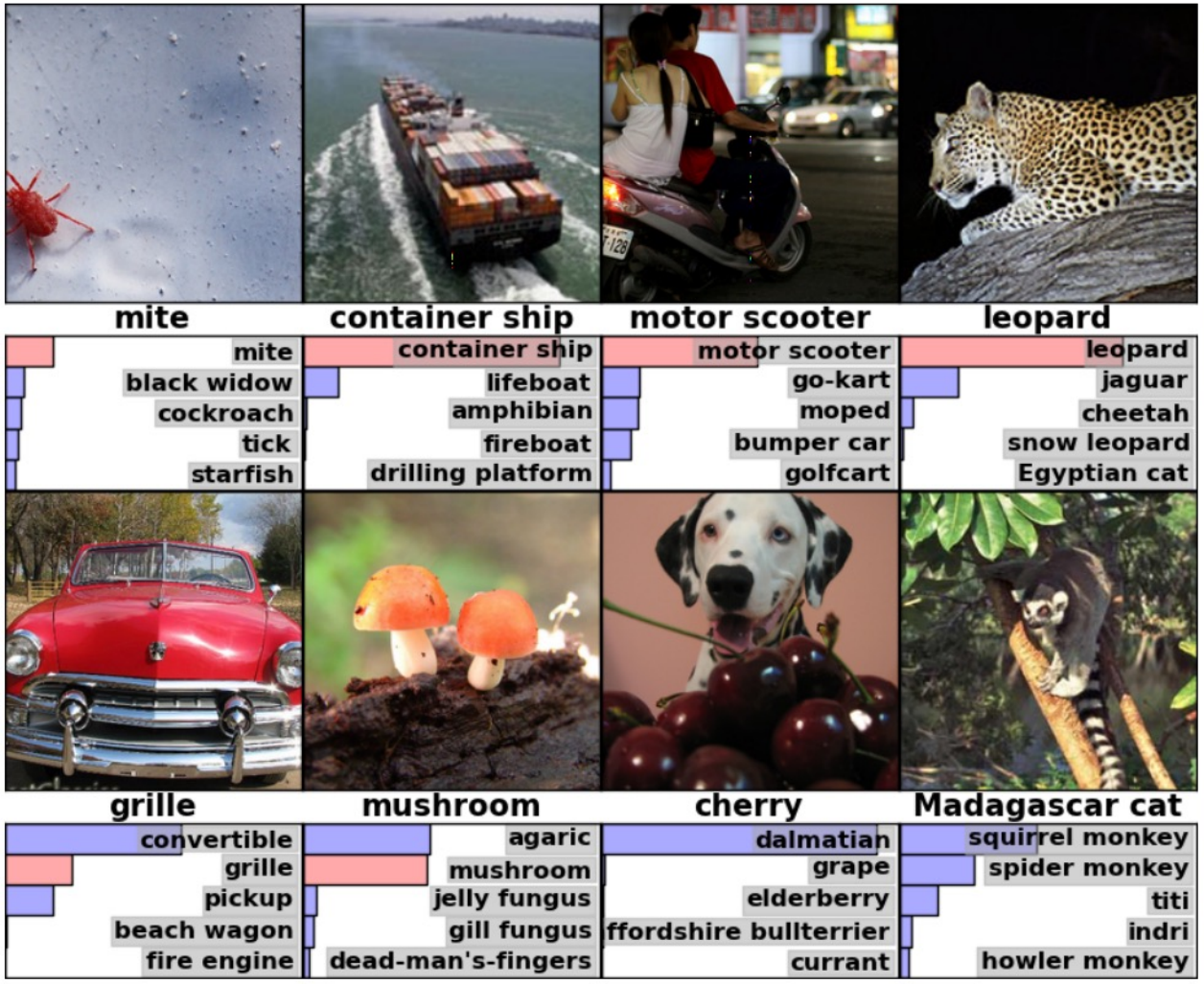
Style Transfer

(Gatys et al., 2015)

Object Detection

(Krizhevsky et al., 2012)

This was one of the breakthroughs, and even better results can be obtained now.
Agaric is a type of mushroom



Text Generation

This was trained on entire works of Shakespeare

(Andrej Karpathy blog, 2015)

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

Image Captioning

(Karpathy & Fei-Fei, 2015)

Extension to captioning videos

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

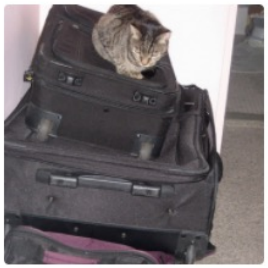
"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

"little girl is eating piece of cake."

"baseball player is throwing ball in game."

"woman is holding bunch of bananas."



"black cat is sitting on top of suitcase."

Further reading

- Nielsen, Chapter 1
- Goodfellow et al., Chapter 1

- So if you were confused by some of the terms used like supervised, unsupervised, etc,
- this is when we'll cover it a lot of those basics.
- For more background material, look into the Goodfellow book.