Deep Learning and Applications

Yannet Interian
University of San Francisco
yinterian@usfca.edu

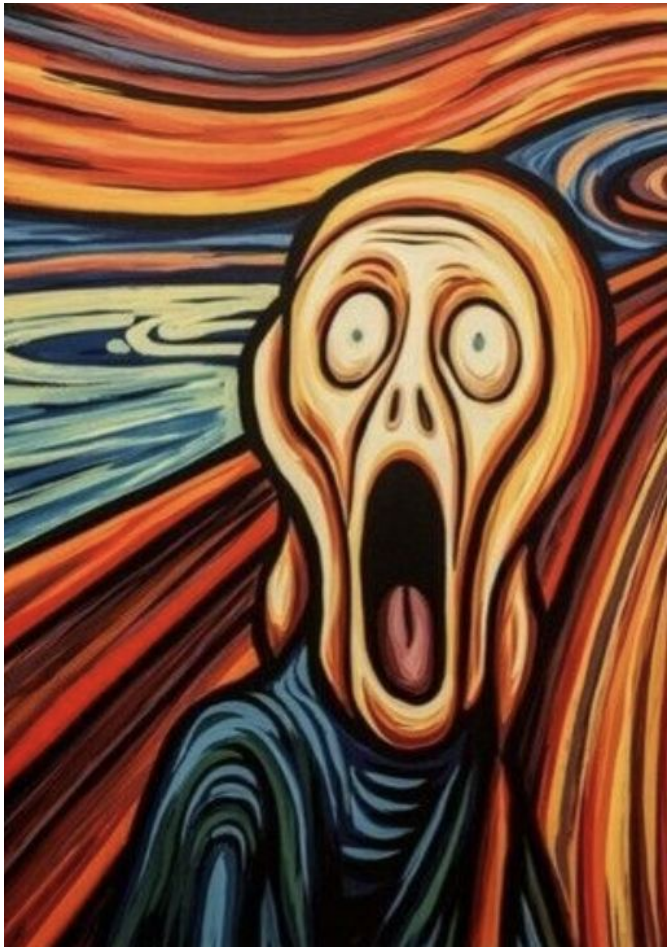
Yannet Interian

- BSc and MS in Math from University of Havana (Cuba)
- ICTP, Trieste, Italy
- PhD in Applied Math from Cornell University
- Data Scientist at Google for 5 years
- Co-founder of a start-up
- From 2015 Professor at USF
    - **Teaching:** Advanced Machine Learning, Deep Learning, Data Acquisition
    - **Research:** Machine Learning and Deep Learning, medical data, applications of reinforcement learning

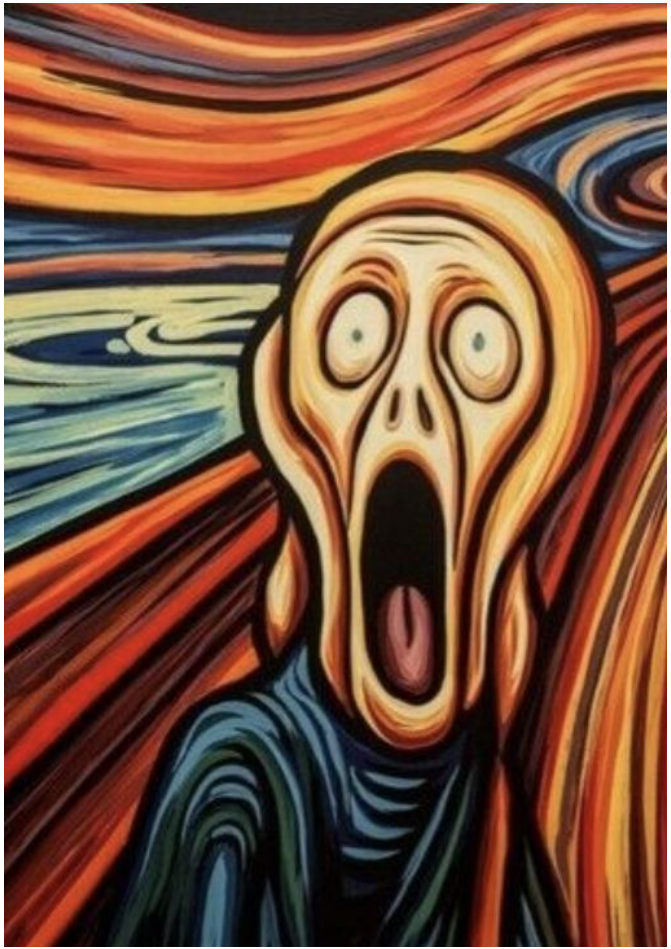
Outline

- When are Neural Networks useful?
- Intro to Neural Networks
- Applications of Deep Learning to Science
- Convolutional Neural Networks
- Introduction to Large Language Models
- Transfer Learning

What ML Model Should I Use?



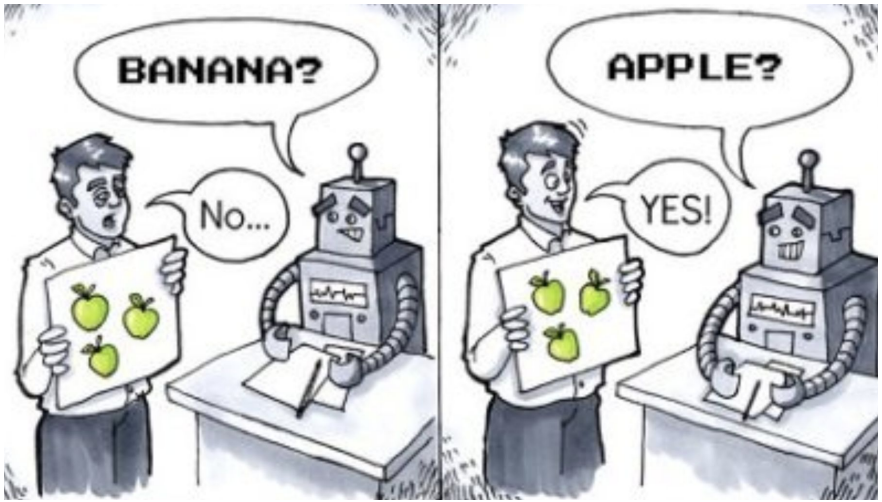
What is the difference between Deep Learning and Neural Networks?



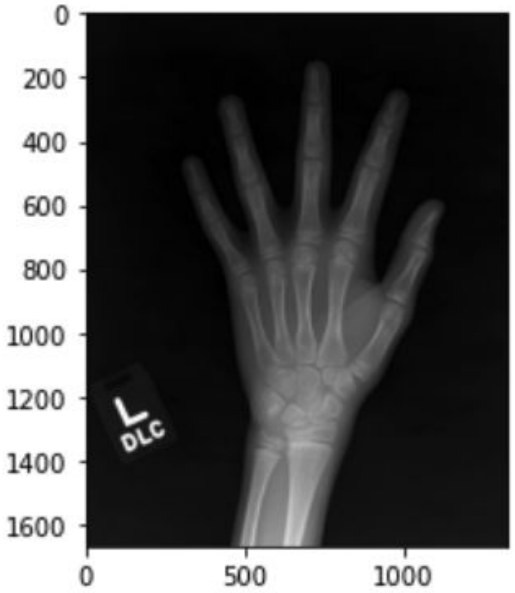
Supervised Learning

Learning a function that maps an input to an output based on example **input-output** pairs:

X = data Y = label



X? And Y?



X = X-ray image

Y = age of a child

Who is X? Who is Y?

- Image classification (dogs versus cats)
- Spam detection on emails
- Estimating the price of a house
- Translate a sentence from Spanish to French
- Find a tumor in an image

Goal of supervised machine learning

Find a function `F(X)` that approximates `Y`

$$Y \approx F(X)$$

`F` is usually called a **model**

What is Deep Learning?

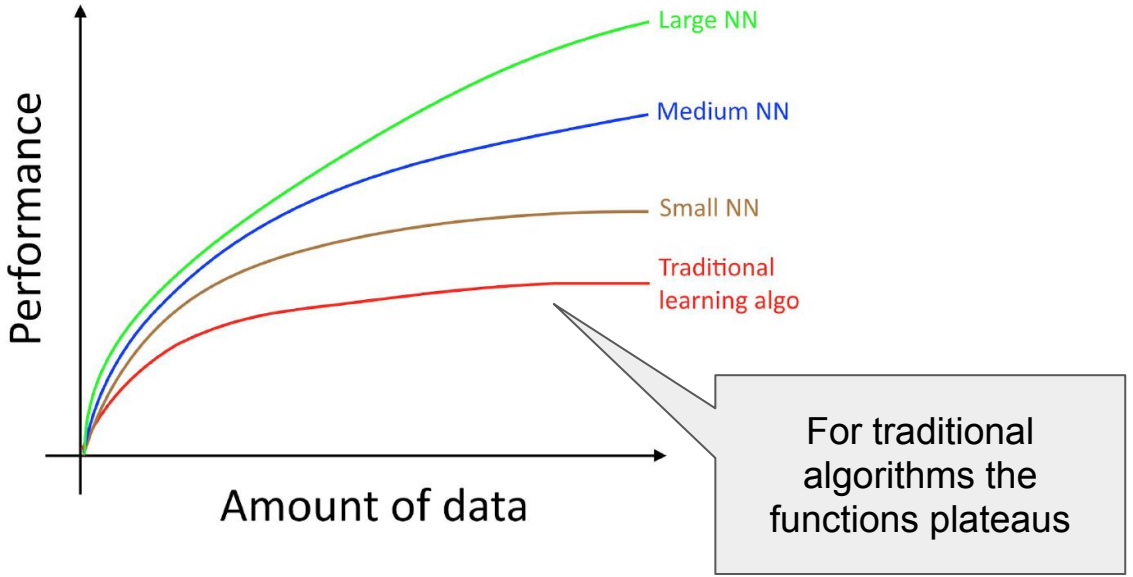
Find a function `F(X)` that approximates `Y`

$$Y \approx F(X)$$

F is usually called a **model**

F is a (deep) Neural Network

Why deep learning?



*from Andrew Ng

ML Models for Supervised Learning

Tabular Data:

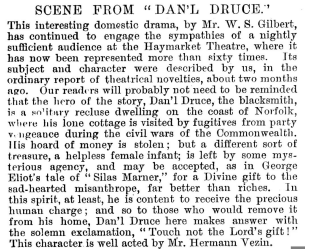
In most cases you want to use tree based models: **Random Forest** and **Gradient Boosting**

Text, Images, Recommendation Systems:

Neural Networks

*This is a generalization

ML Models for Supervised Learning



SCENE FROM "DAN'L DRUCE."

This interesting domestic drama, by Mr. W. S. Gilbert, has continued to engage the sympathies of a nightly sufficient audience at the Haymarket Theatre, where it has now been represented more than sixty times. Its subject and character were described by us, in the ordinary report of theatrical novelties, about two months ago. Our readers will probably not need to be reminded that the hero of the story, Dan'l Druce, the blacksmith, is a solitary recluse dwelling on the coast of Norfolk, where his lone cottage is visited by fugitives from party vengeance during the civil wars of the Commonwealth. His hoard of money is stolen; but a different sort of treasure, a helpless female infant; is left by some mysterious agency, and may be accepted, as in George Eliot's tale of "Silas Marner," for a Divine gift to the sad-hearted misanthrope, far better than riches. In this spirit, at least, he is content to receive the precious human charge; and so to those who would remove it from his home, Dan'l Druce here makes answer with the solemn exclamation, "Touch not the Lord's gift!" This character is well acted by Mr. Hermann Vezin.

Text:

Neural architectures for text:

- Recurrent Neural Networks
- Transformer models (Bert, GPTs…)

Images:

Neural architectures for images:

- Convolutional Neural Networks
- Transformers

Neural Nets

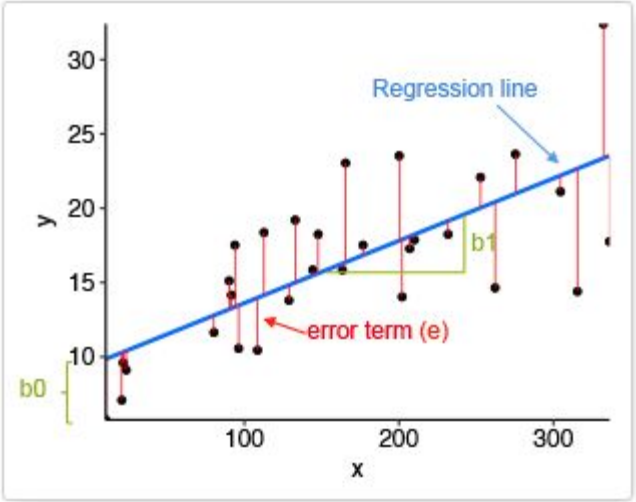
Simplest Neural Nets

- Linear Regression
- Logistic Regression
- Multi-class logistic regression

Review of linear regression

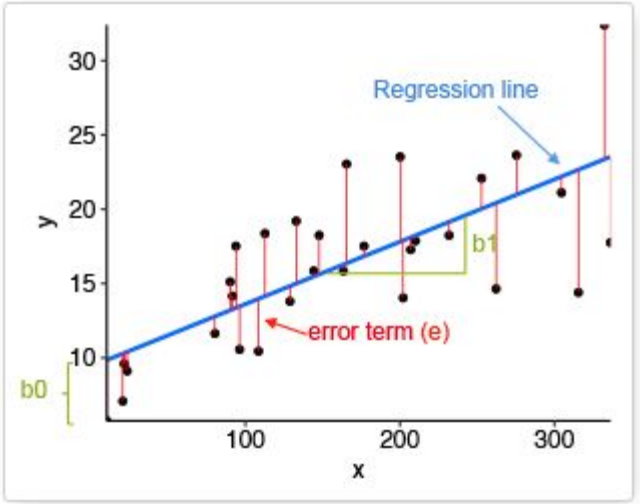
- x square footage of a house
- y value

$$\hat{y} = b_0 + b_1 \cdot x$$



Review of linear regression: how to find the line?

$$\hat{y} = b_0 + b_1 \cdot x$$



The line that minimizes the sum of the square of the errors.

Linear regression summary

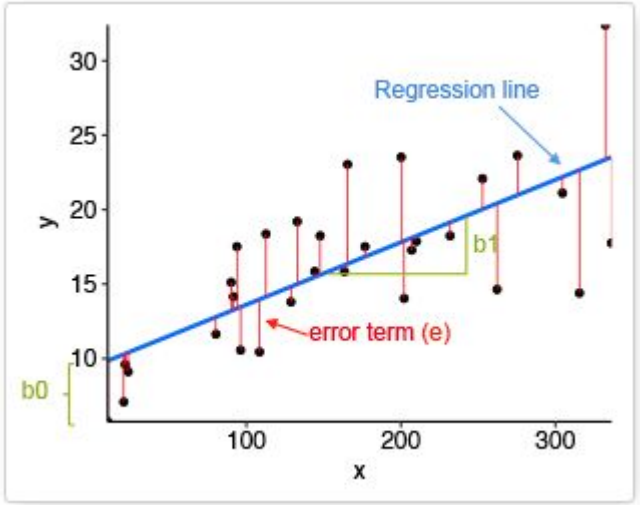
Model:

$$\hat{y} = b_0 + b_1 \cdot x$$

Loss function: $(y - \hat{y})^2$

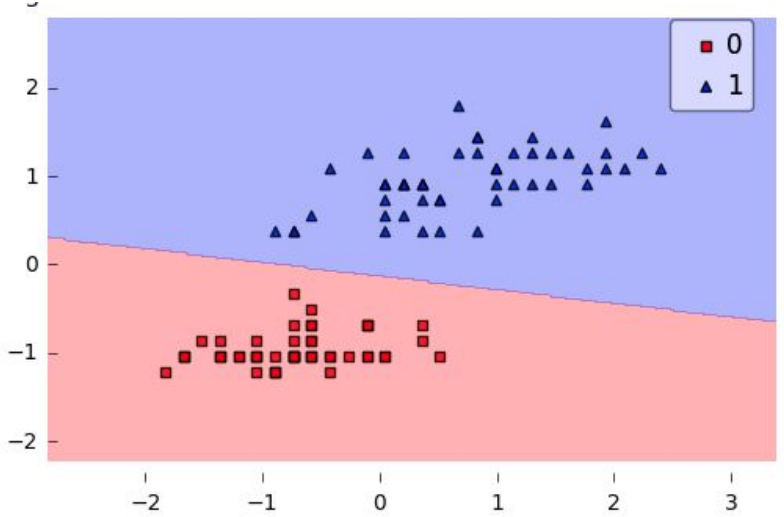
Training points: $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$

Cost function: $\frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$



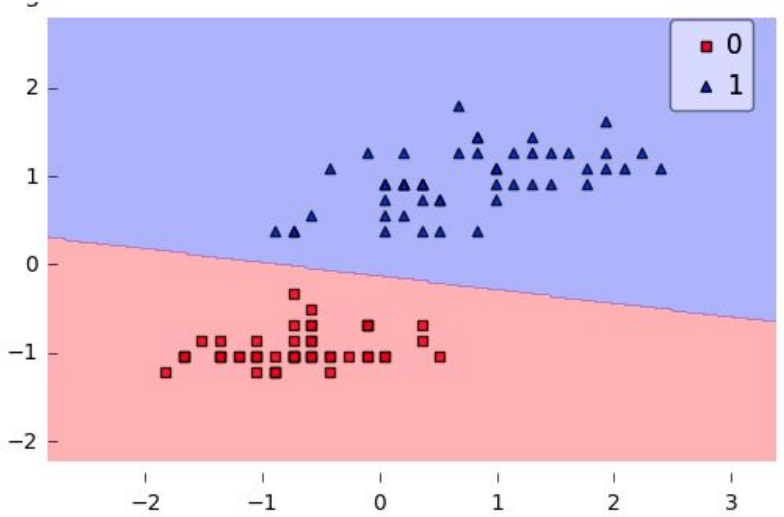
Review of logistic regression

- Model classification problems
- In picture: two features and the two labels (`0` or `1`)
- Linear classifier
- Objective: find the line that separates `0`s from `1`s



Logistic regression: Model

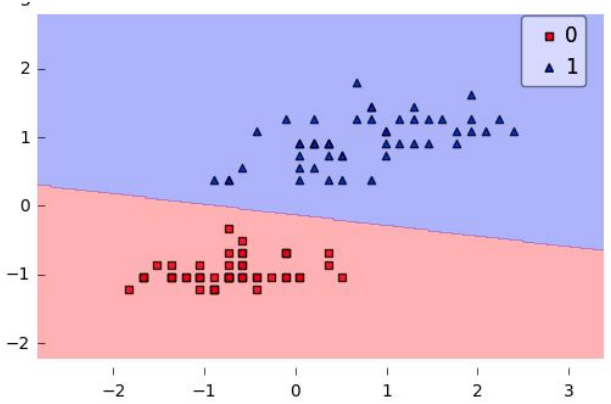
- Model classification problems
- Model predicts the probability of class 1
- Predict dogs versus cats



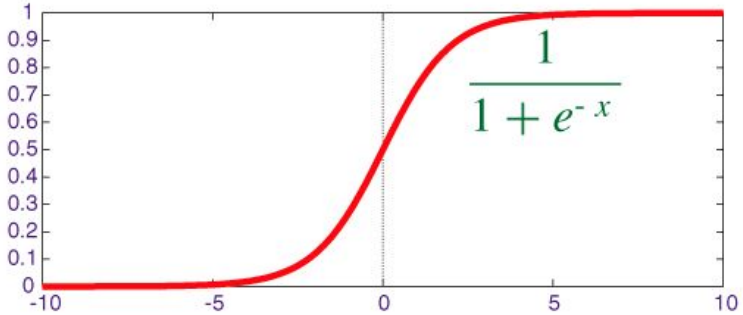
$$\hat{y} = \sigma(b_0 + b_1 \cdot x)$$

Logistic regression: Model

- Model classification problems
- Model predicts the probability of class 1



$$\hat{y} = \sigma(b_0 + b_1 \cdot x)$$



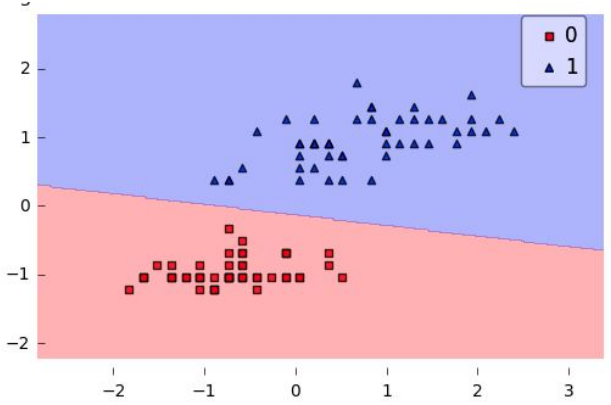
Logistic regression

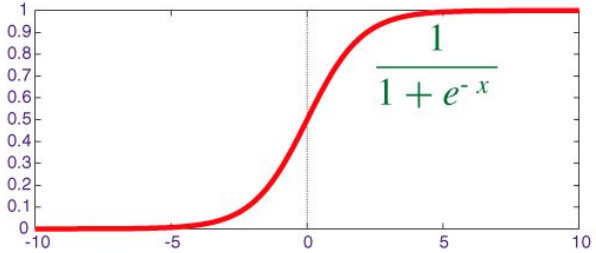
Model:

$$\hat{y} = \sigma(b_0 + b_1 \cdot x)$$

Loss function: $-[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$

Cost function: Average the loss function over the training points.





Multi-class logistic regression

- Classification problem
- Instead of two classes we have `K` classes
  - Predict dog breed from images
- Generalization of logistic regression
  - Need `K` lines instead of one

Multi-class logistic regression

Instead of two classes we have `K` classes

- Predict dog breed from images

$\hat{y} = (\hat{y}_1, \ldots, \hat{y}_K)$ prediction is a tuple
$\sum_{k=1}^{K} \hat{y}_k = 1$ sum of all predictions is one
Loss:

$$-\sum_{k=1}^{K} y_k \log \hat{y}_k$$

Multi-class logistic regression

For multi-class logistic regression we need K linear equations.

$$z_1 = b_1 + w_1 \cdot x$$
$$z_2 = b_2 + w_2 \cdot x$$
$$\dots$$
$$z_K = b_K + w_K \cdot x$$

Dot product

To get the soft predictions $\hat{y} = (\hat{y}_1, \dots, \hat{y}_K)$ from $z = (z_1, \dots, z_K)$ we use the **softmax** function to get probabilities.

$$\hat{y}_k = P(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Multi-class logistic regression

$$z_1 = b_1 + w_1 \cdot x$$
$$z_2 = b_2 + w_2 \cdot x$$
$$\dots$$
$$z_K = b_K + w_K \cdot x$$

Cam be written as

$$z = W \cdot x + b$$

Where $z = (z_1, z_2, \dots, z_K)^T$, $b = (b_1, b_2, \dots, b_K)^T$ and w_i are the rows of W. If $x \in \mathbb{R}^D$, W is KxD.

Multi-class logistic regression

$$z = W \cdot x + b$$

$$\hat{y} = softmax(z)$$

Two layer neural network for regression

$$
\begin{aligned}
z^{[1]} &= W^{[1]} \cdot x + b^{[1]} \\
a^{[1]} &= h(z^{[1]}) \\
z^{[2]} &= W^{[2]} \cdot a^{[1]} + b^{[2]} \\
\hat{y} &= z^{[2]}
\end{aligned}
$$

where h is the element-wise activation function.

Two layer neural network binary for classification

$$
\begin{aligned}
z^{[1]} &= W^{[1]} \cdot x + b^{[1]} \\
a^{[1]} &= h(z^{[1]}) \\
z^{[2]} &= W^{[2]} \cdot a^{[1]} + b^{[2]} \\
\hat{y} &= \sigma(z^{[2]})
\end{aligned}
$$

where h is the element-wise activation function.

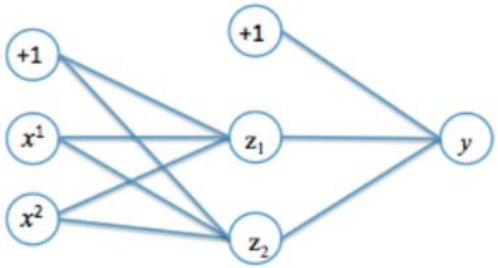
Two layer neural network multi-class classification

$$
\begin{aligned}
z^{[1]} &= W^{[1]} \cdot x + b^{[1]} \\
a^{[1]} &= h(z^{[1]}) \\
z^{[2]} &= W^{[2]} \cdot a^{[1]} + b^{[2]} \\
\hat{y} &= softmax(z^{[2]})
\end{aligned}
$$

where h is the element-wise activation function.

Example

Consider the network below.



$$\sigma(a) = h(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{otherwise.} \end{cases}$$

x_1	x_2	$z_1^{[1]}$	$z_2^{[1]}$	$a_1^{[1]}$	$a_2^{[1]}$	$z^{[2]}$	$\hat{y}$
0	0						
0	1						
1	0						
1	1						

Assume the following network weights:

$b_1^{[1]}$	$w_{11}^{[1]}$	$w_{12}^{[1]}$	$b_2^{[1]}$	$w_{21}^{[1]}$	$w_{22}^{[1]}$	$b^{[2]}$	$w_1^{[2]}$	$w_2^{[2]}$
-30	20	20	10	-20	-20	-10	20	20

Neural Network -- Intuition

SERRANO.ACADEMY

the art of understanding

Serrano.Academy

@SerranoAcademy · 125K subscribers · 45 videos

Welcome to Serrano.Academy! I'm Luis Serrano and I love demystifying concepts, capturin... >

twitter.com/SerranoAcademy and 5 more links

Subscribed

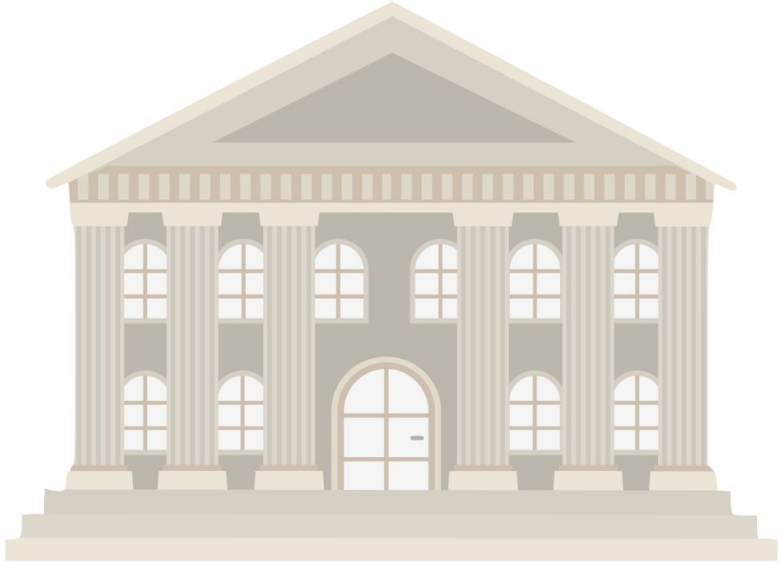
Home Videos Shorts Live Playlists Community Channels About

Serrano.Academy

Serrano.Academy • 11K views • 1 year ago

Learn machine learning and data science the simple way. Free videos and tutorials on deep learning, reinforcement learning, probability, statistics, and much more. Luis Serrano, the author...

Admission Office



Exam

Grades

Admission Office

Exam

Grades

Student 1
Exam: 9/10
Grades: 8/10

Student 2
Exam: 3/10
Grades: 4/10

Student 3
Exam: 7/10
Grades: 6/10

Would you admit Student 3?

Admission Office

Exam

Grades

Student 1
Exam: 9/10
Grades: 8/10

Student 2
Exam: 3/10
Grades: 4/10

Student 3
Exam: 7/10
Grades: 6/10

Score = Exam + Grades
Score > 10

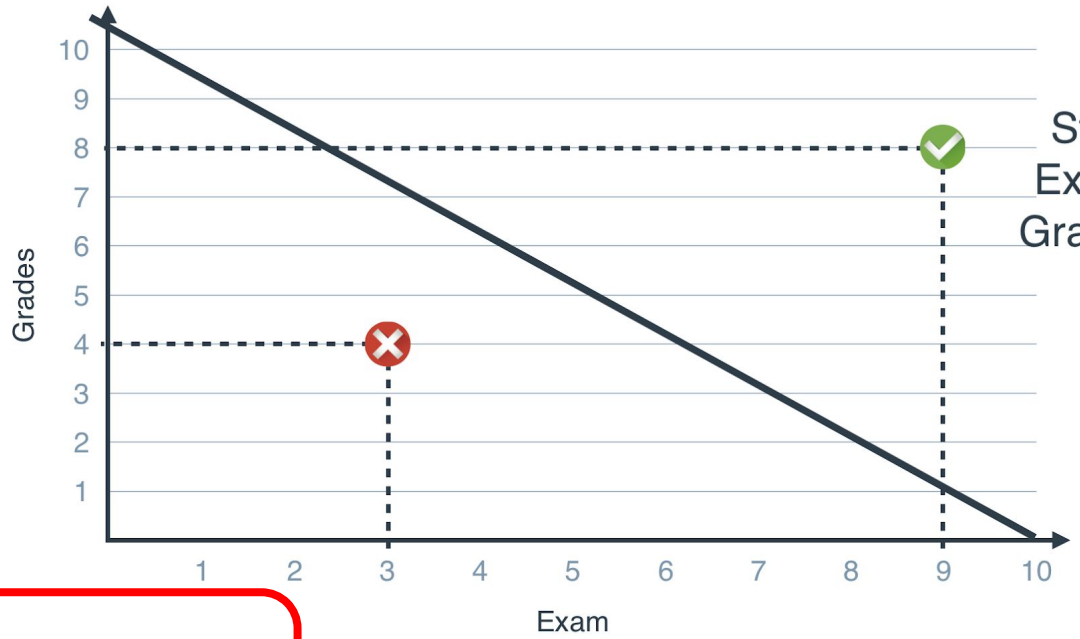
Score = Exam + 2*Grades
Score > 18

Exam > 5
Grades > 5

Admission Office: model 1

Student 1
Exam: 9/10
Grades: 8/10

Student 2
Exam: 3/10
Grades: 4/10



Student 3
Exam: 7/10
Grades: 6/10

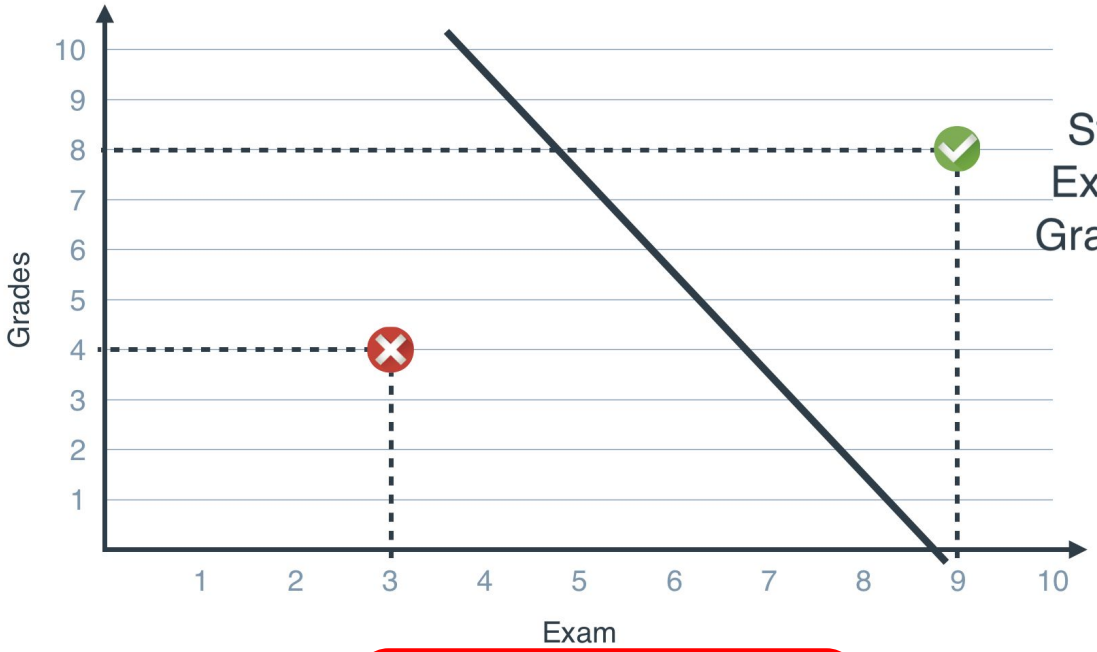
Score = Exam + Grades
Score > 10

Score = Exam + 2*Grades
Score > 18

Admission Office: model 2

Student 1
Exam: 9/10
Grades: 8/10

Student 2
Exam: 3/10
Grades: 4/10



Student 3
Exam: 7/10
Grades: 6/10

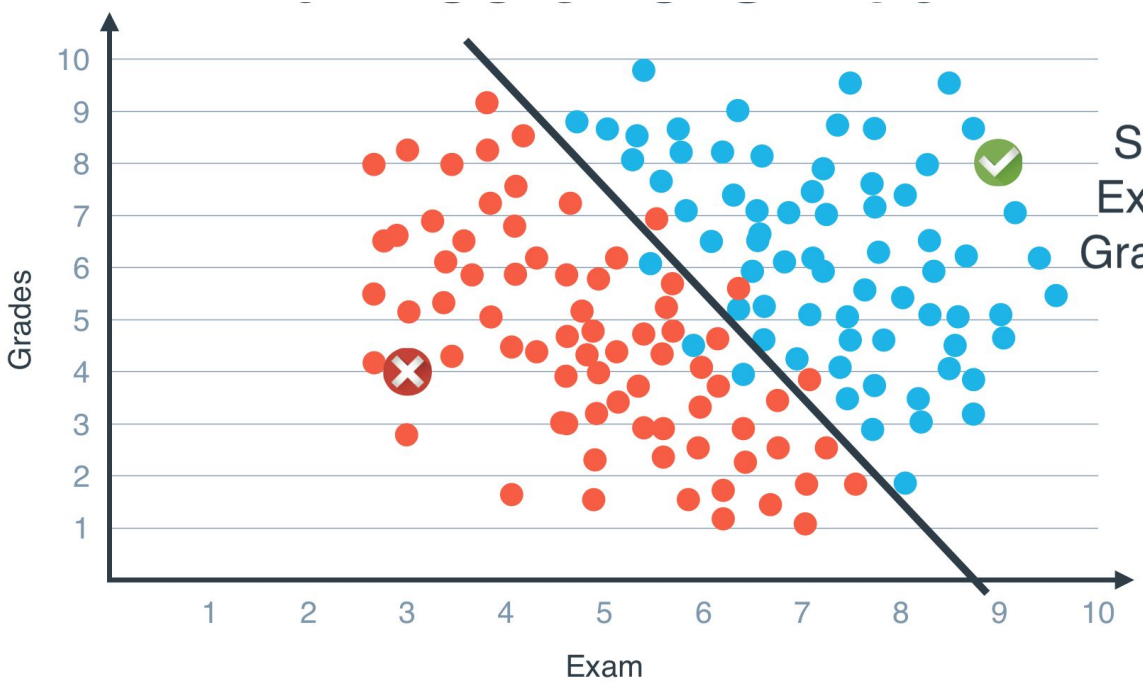
Score = Exam + Grades
Score > 10

Score = Exam + 2*Grades
Score > 18

Does the student get accepted?

Student 1
Exam: 9/10
Grades: 8/10

Student 2
Exam: 3/10
Grades: 4/10

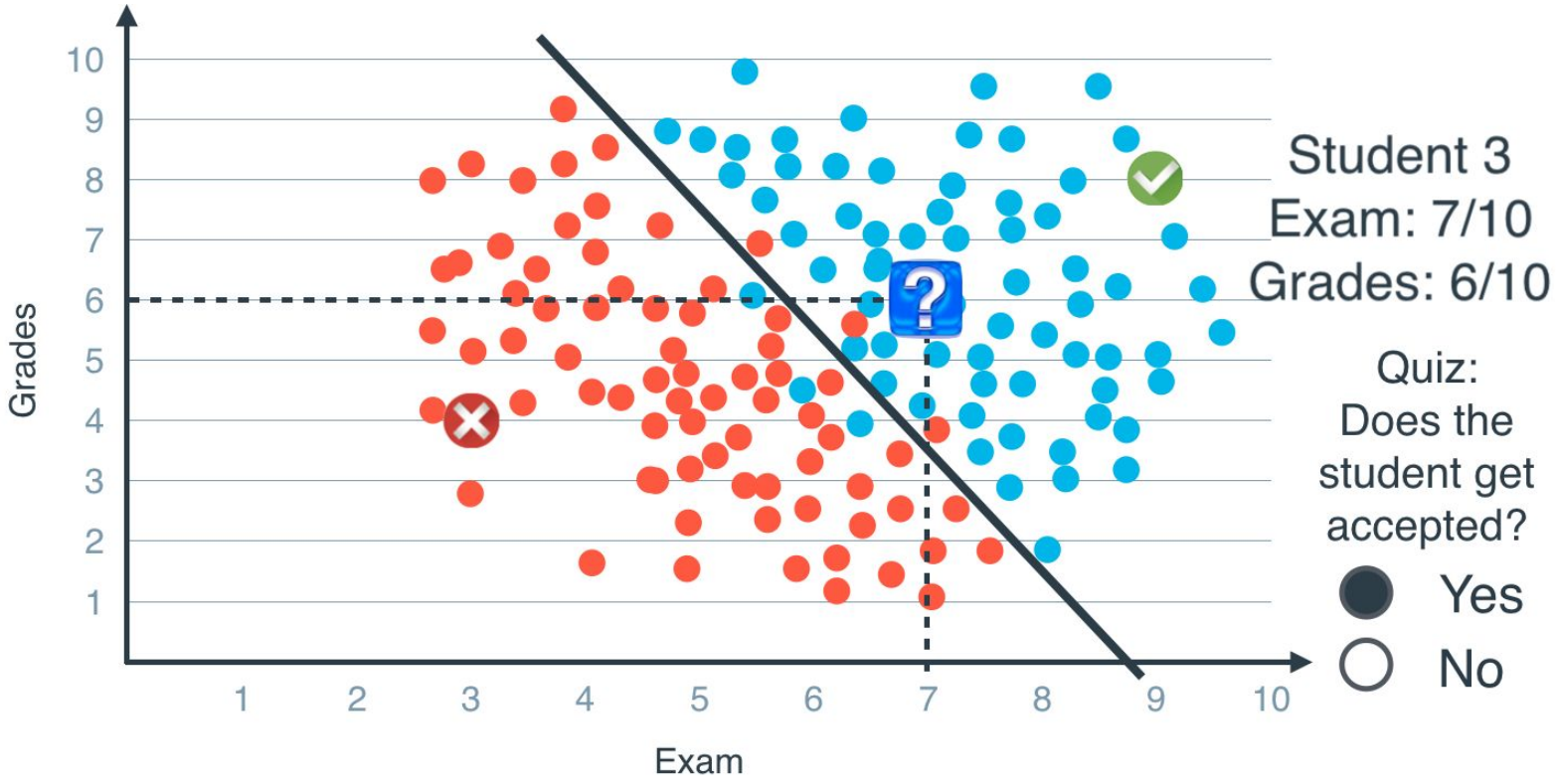


Student 3
Exam: 7/10
Grades: 6/10

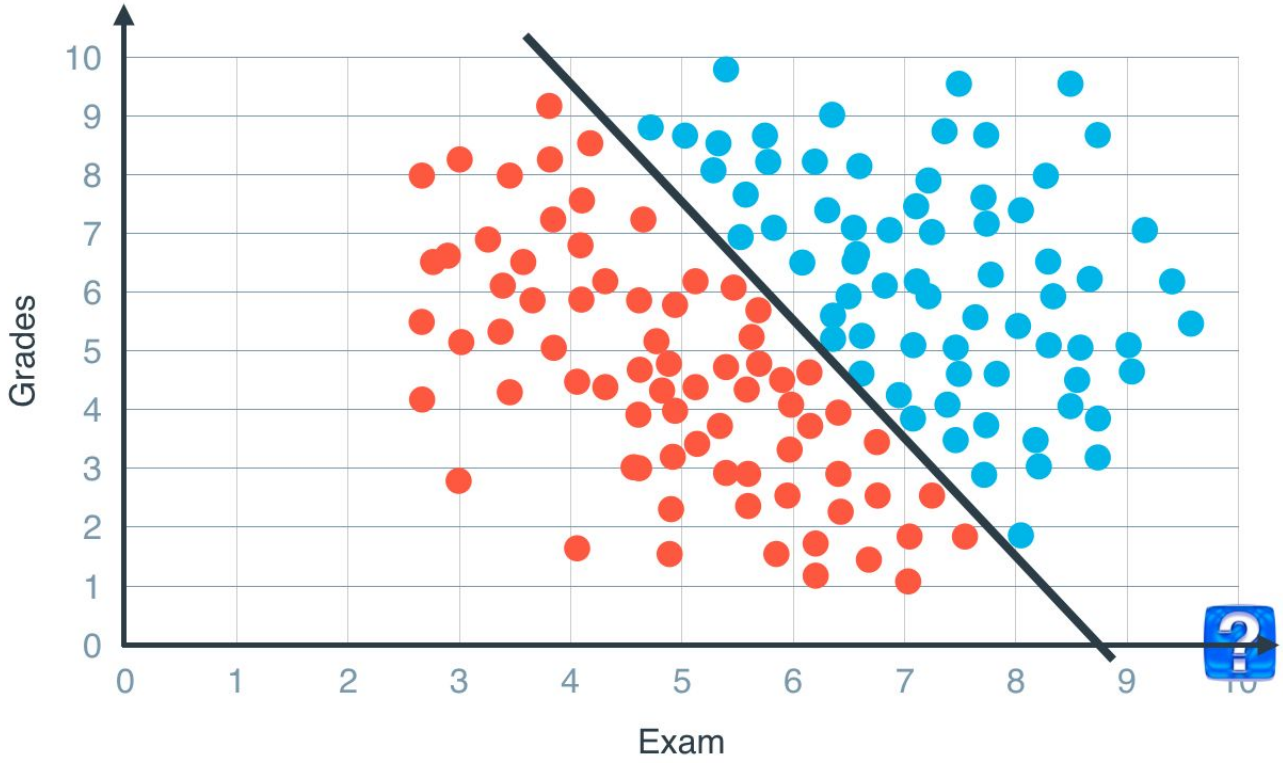
Score = Exam + Grades
Score > 10

Score = Exam + 2*Grades
Score > 18

Given this model: yes

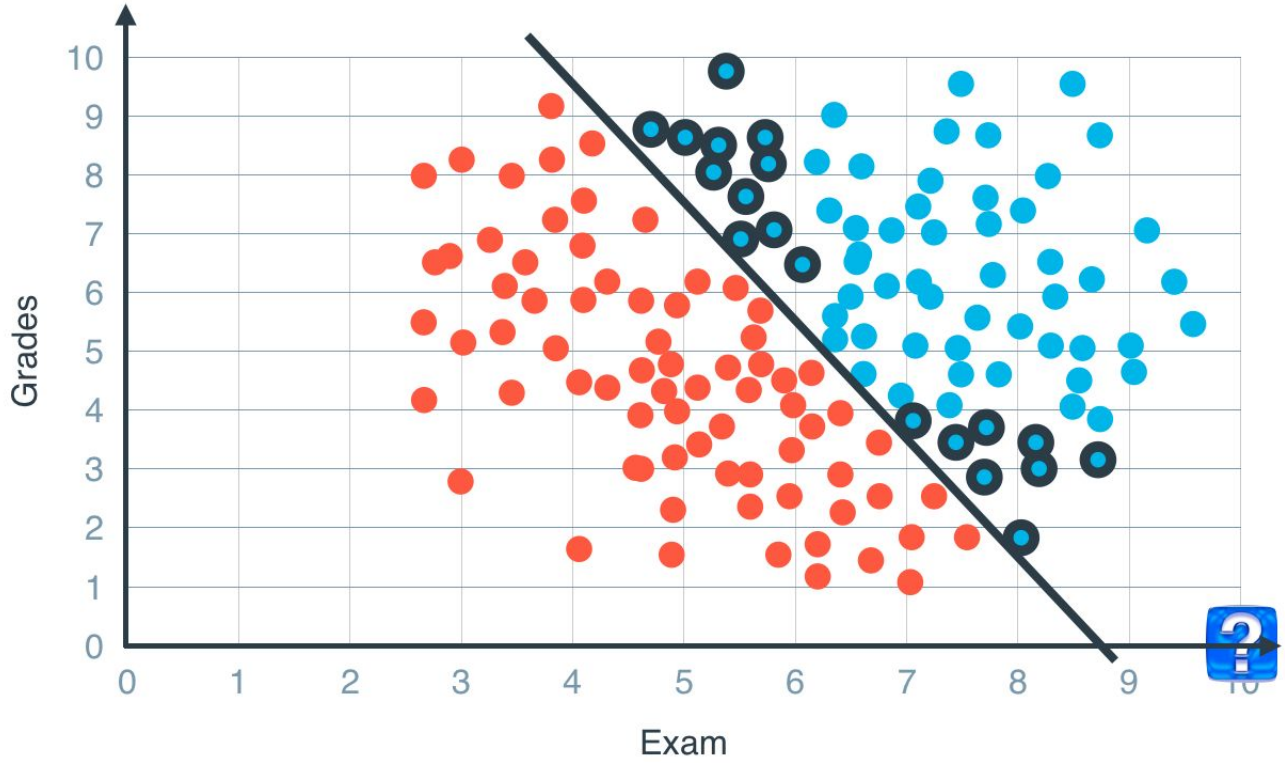


Would you accept this student?



Student 4
Exam: 10
Grades: 0

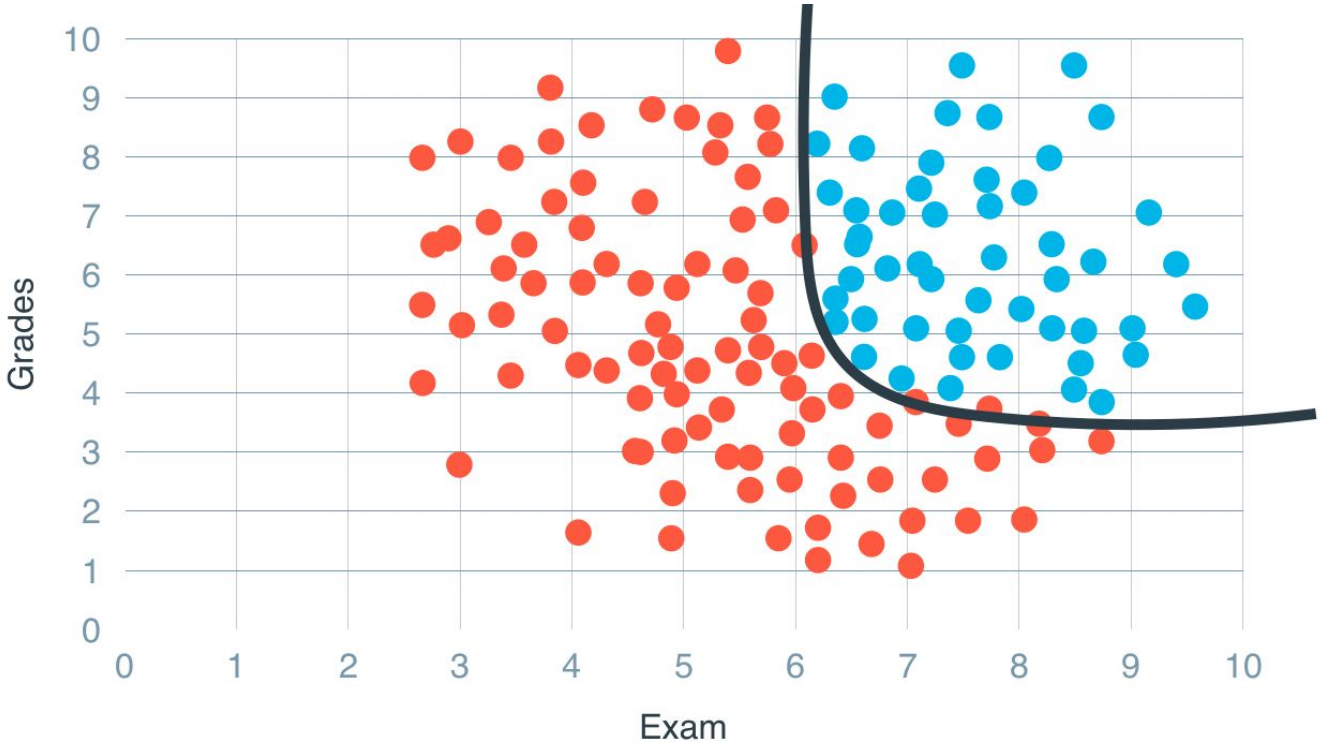
Real data likely non-linear



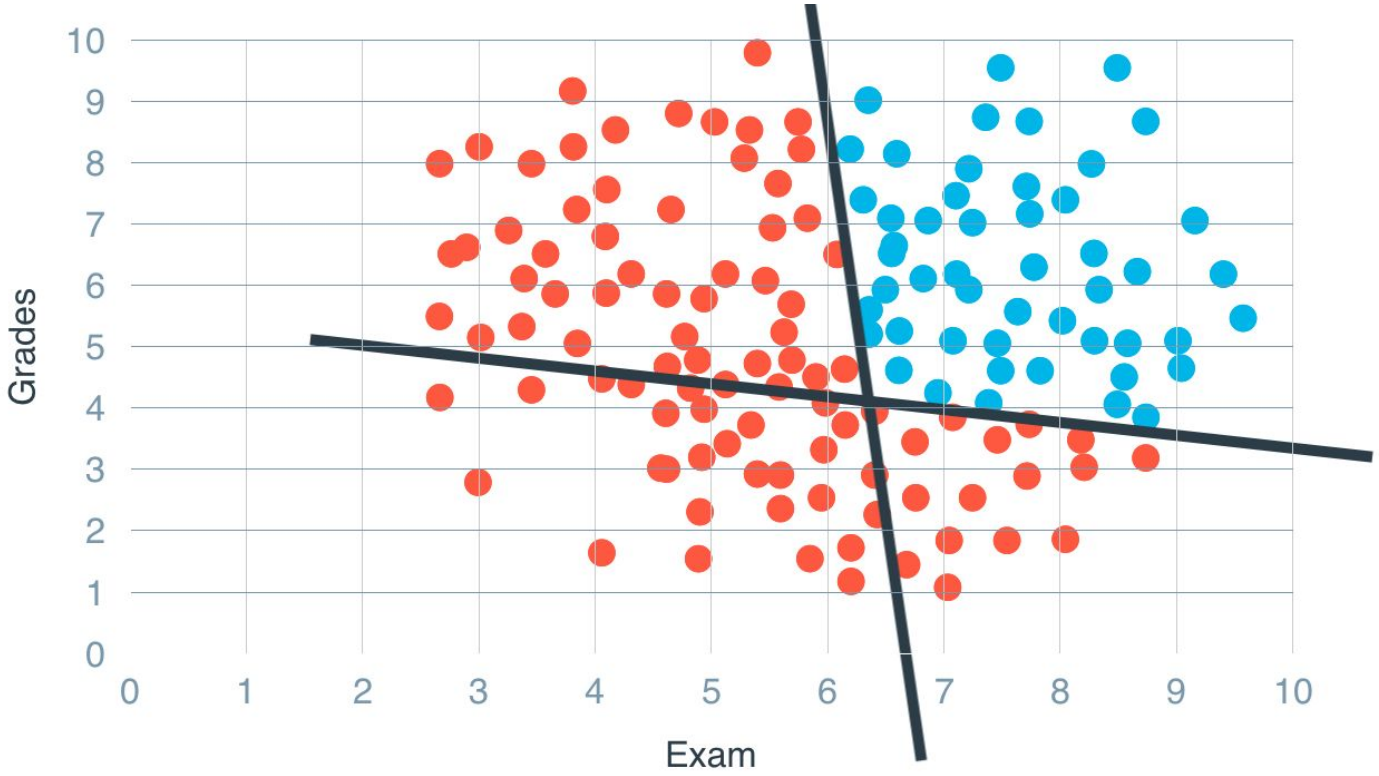
Student 4
Exam: 10
Grades: 0

Real data likely non-linear

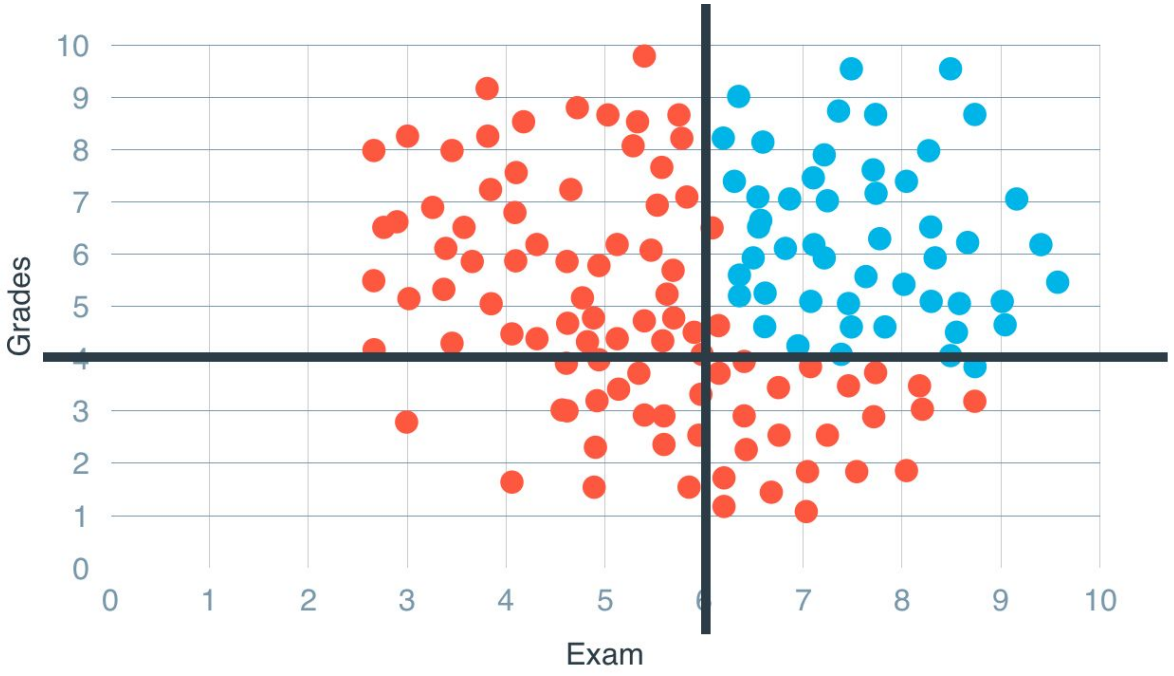
Real data likely non-linear



How to get non-linear from linear

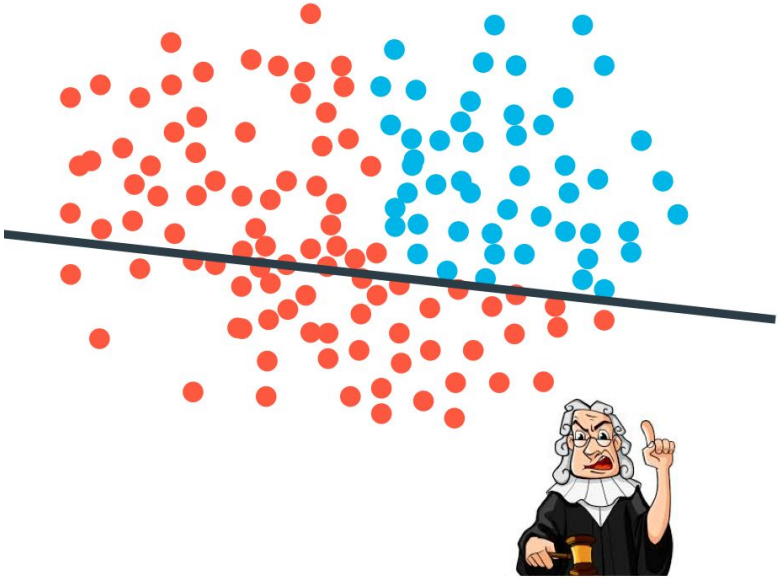


How to get non-linear from linear

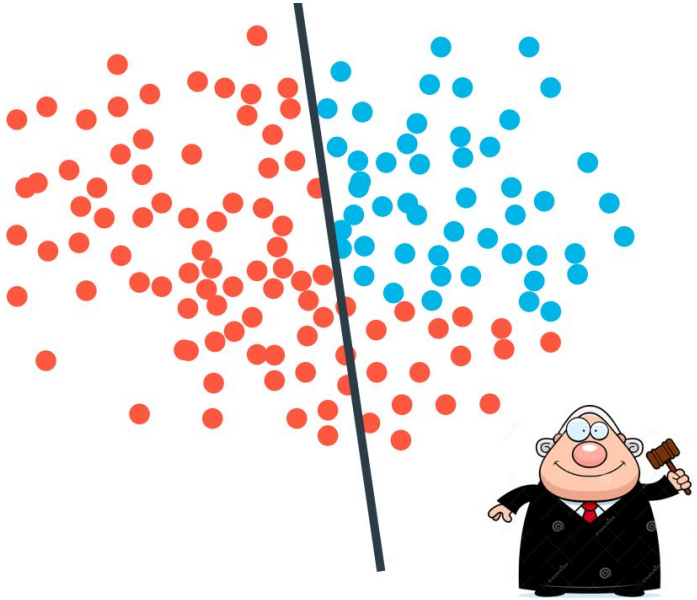


Exam > 6
Grades > 4

Where is the Consensus

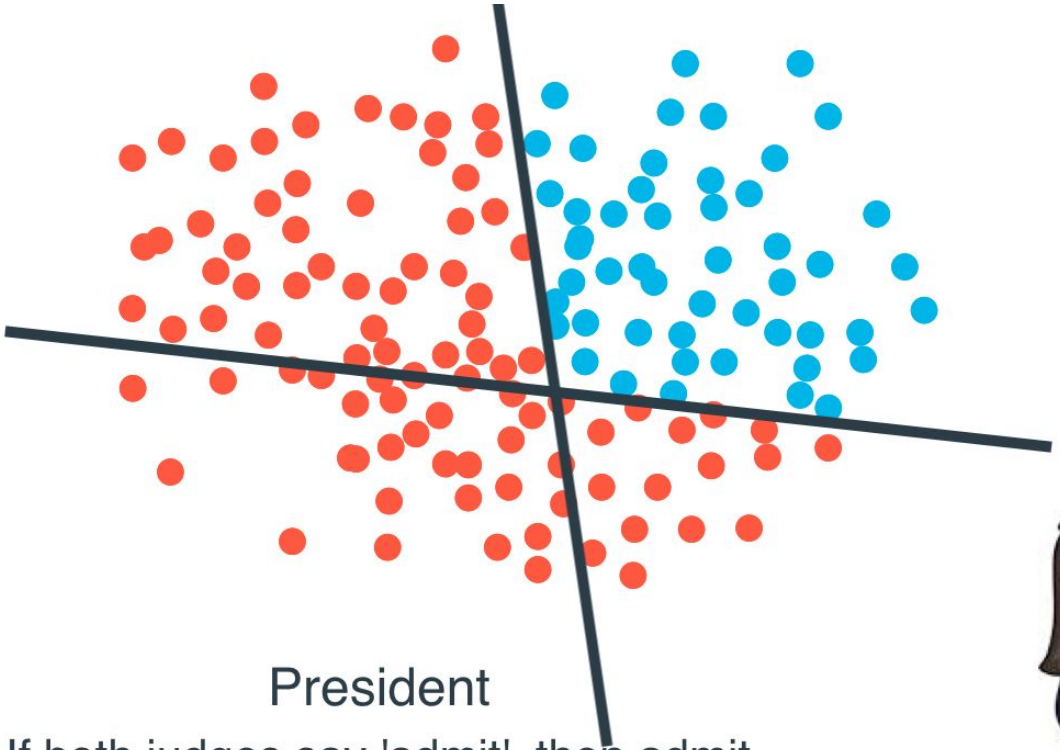


Judge 1



Judge 2

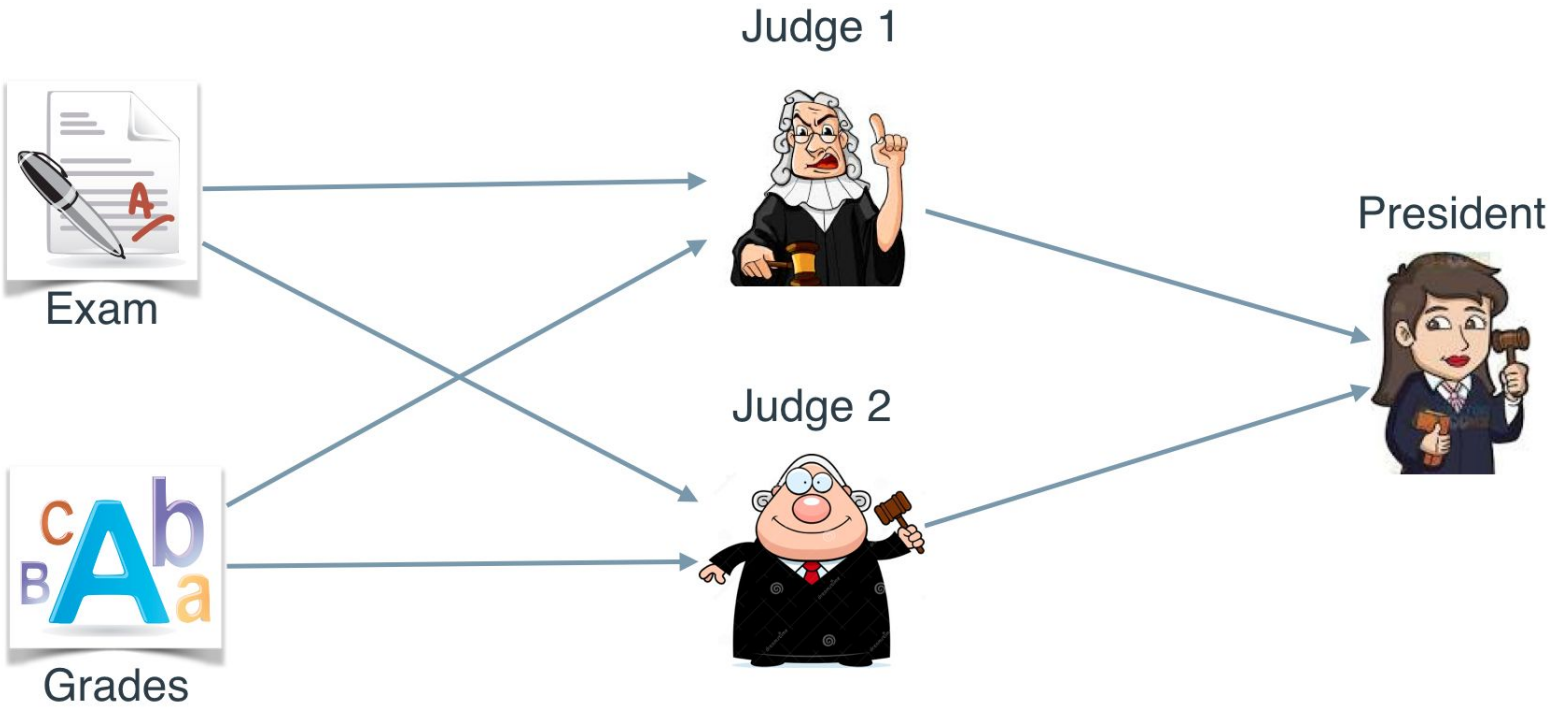
How the president makes decisions



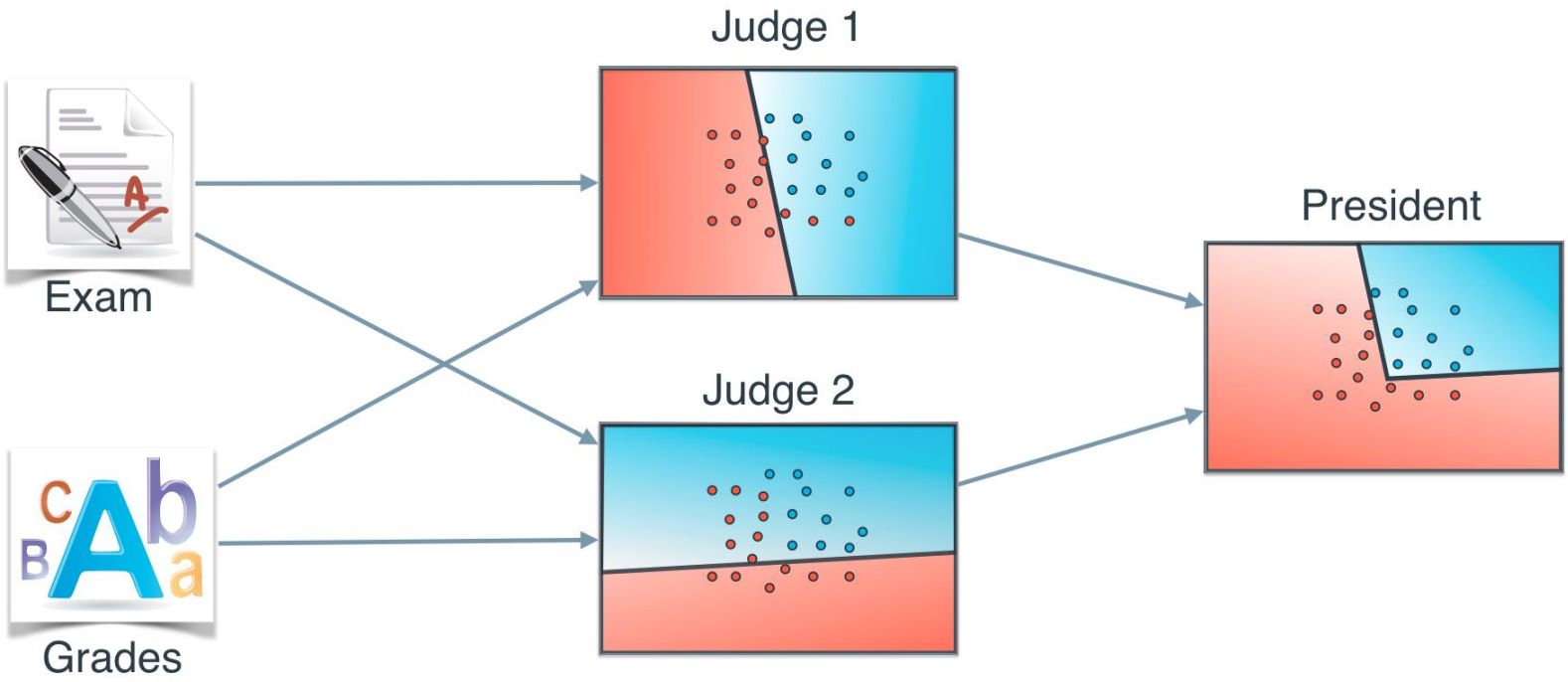
President

If both judges say 'admit', then admit

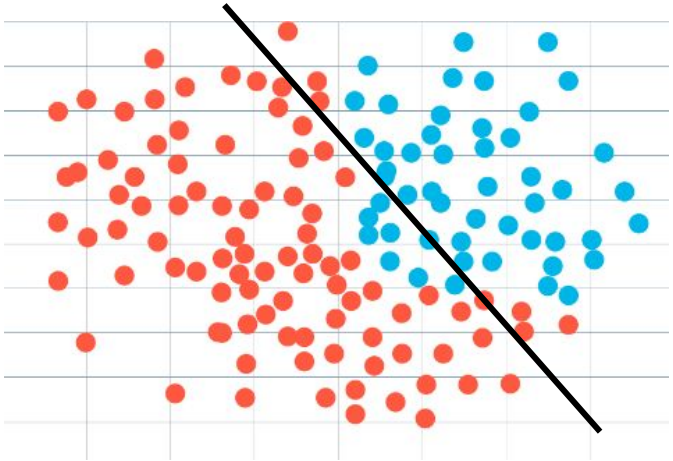
Neural Network

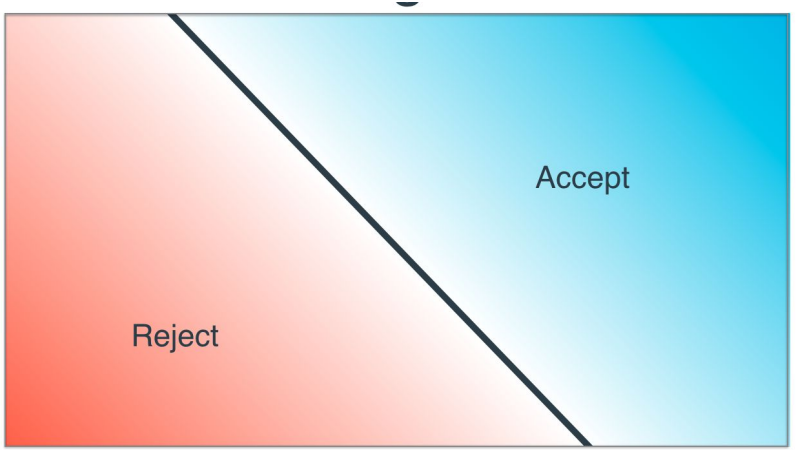


Neural Network

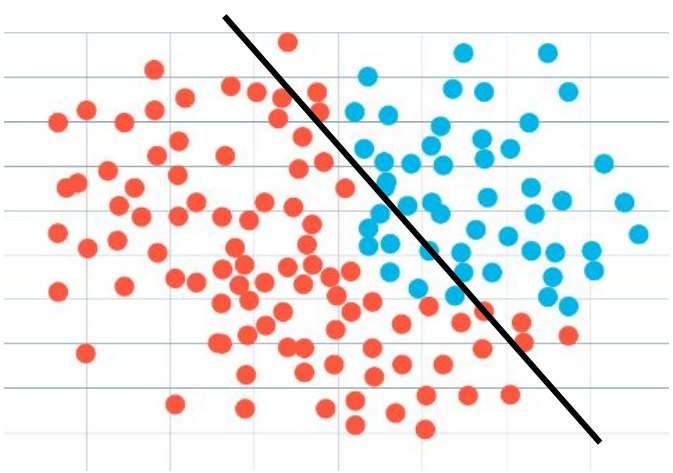


Normalizing each judge

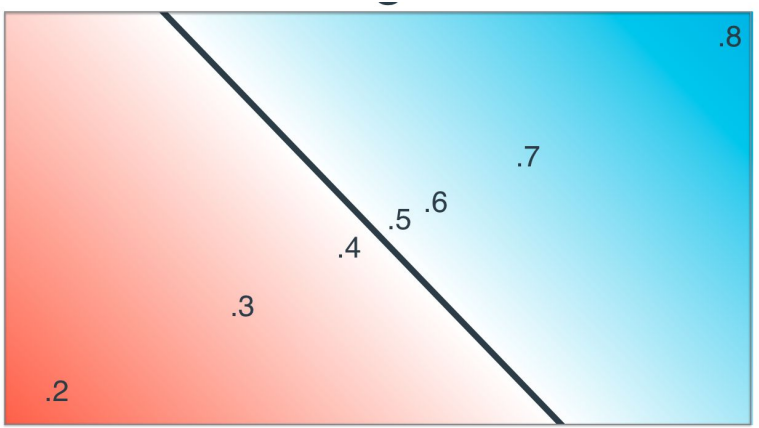




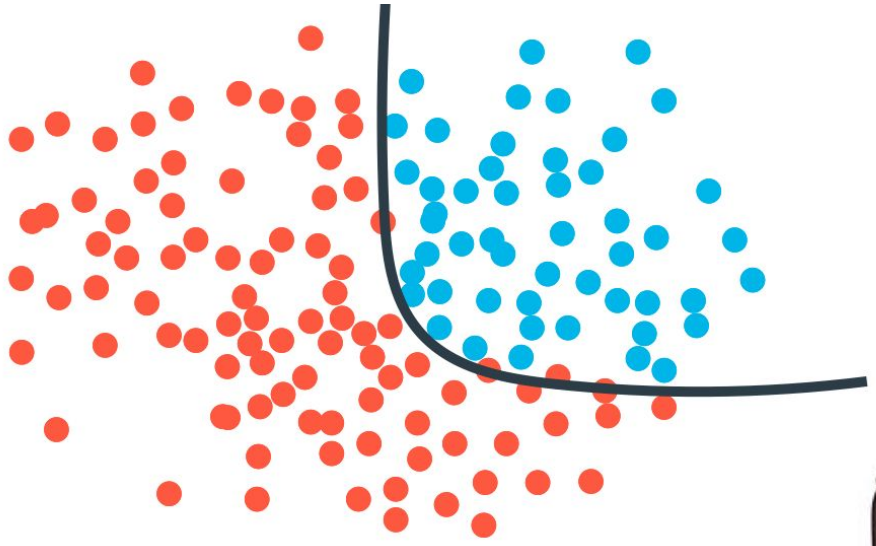
Normalizing: sigmoid takes raw scores into normalized scores (probabilities)



sigmoid



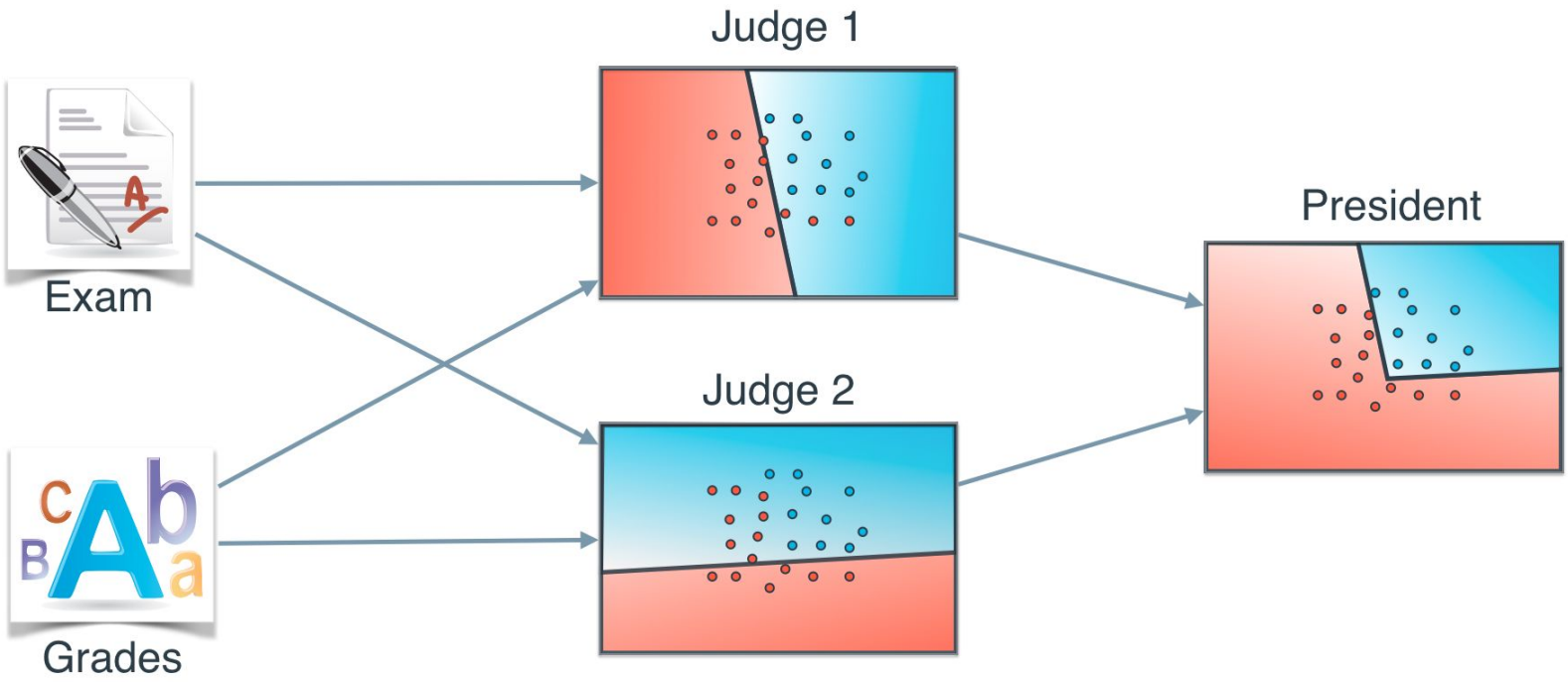
Formula for president



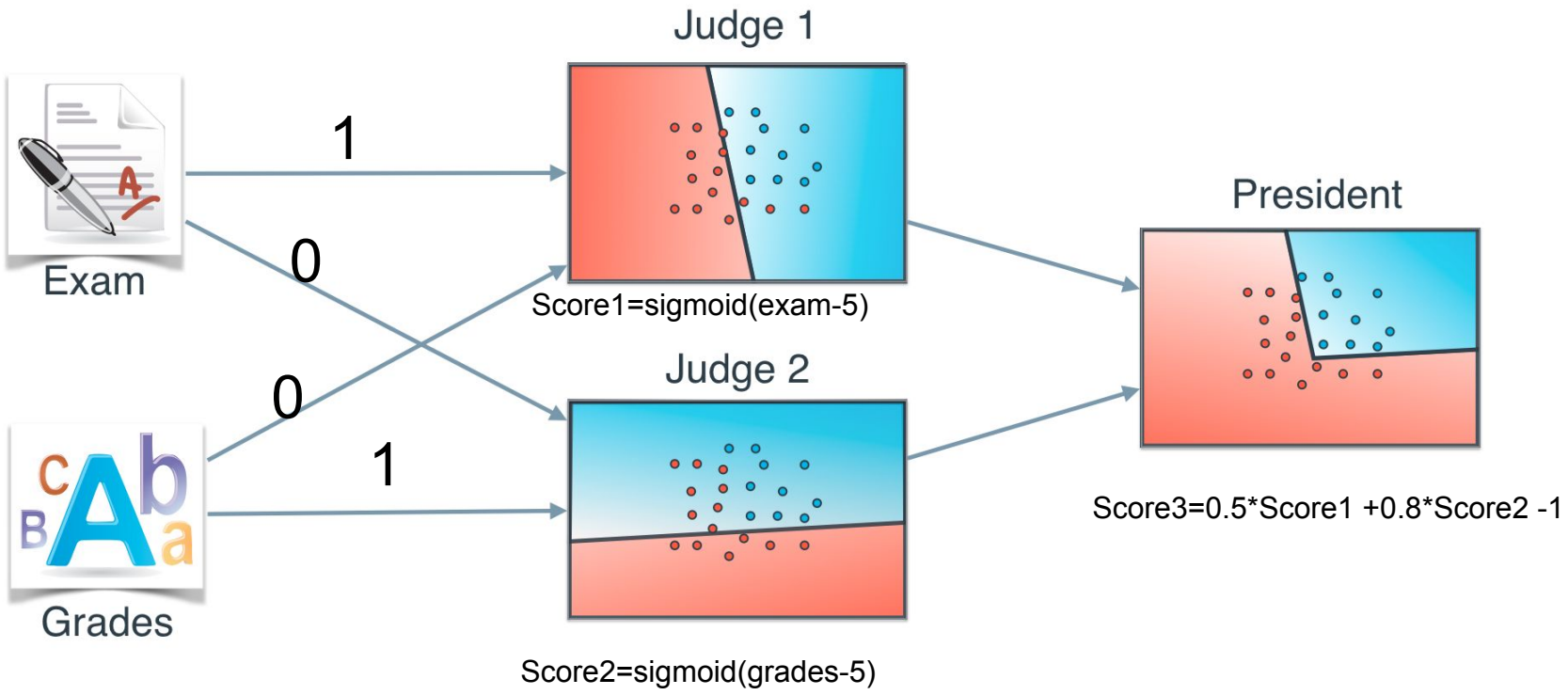
President

Score = 2*(Judge 1 Score) + 3*(Judge 2 Score)

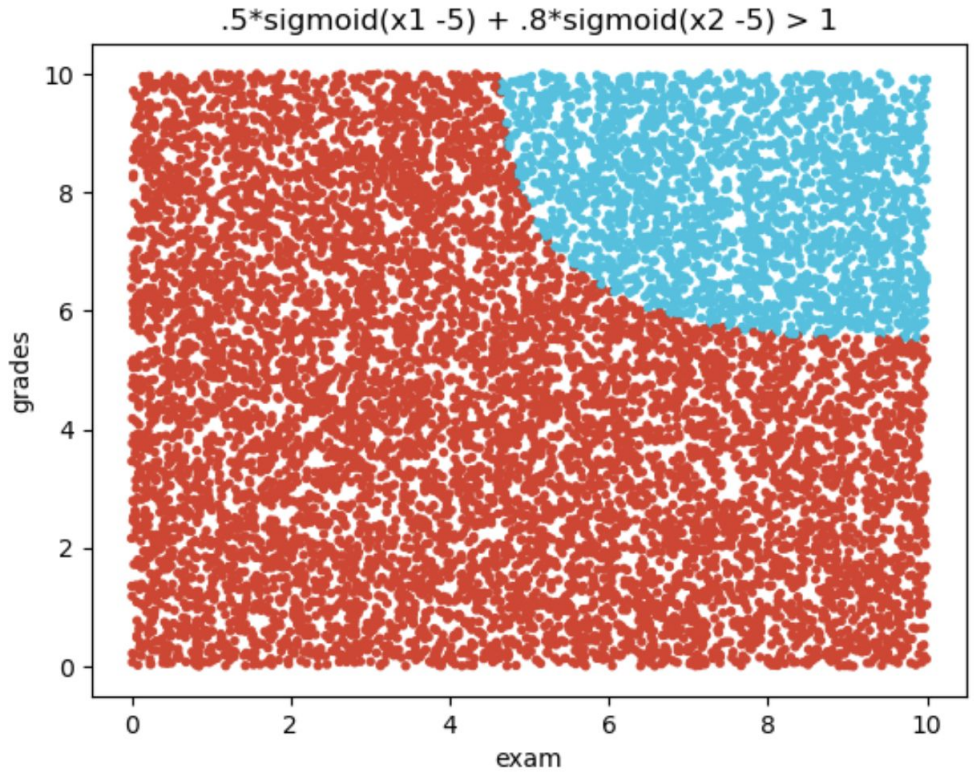
Neural Network



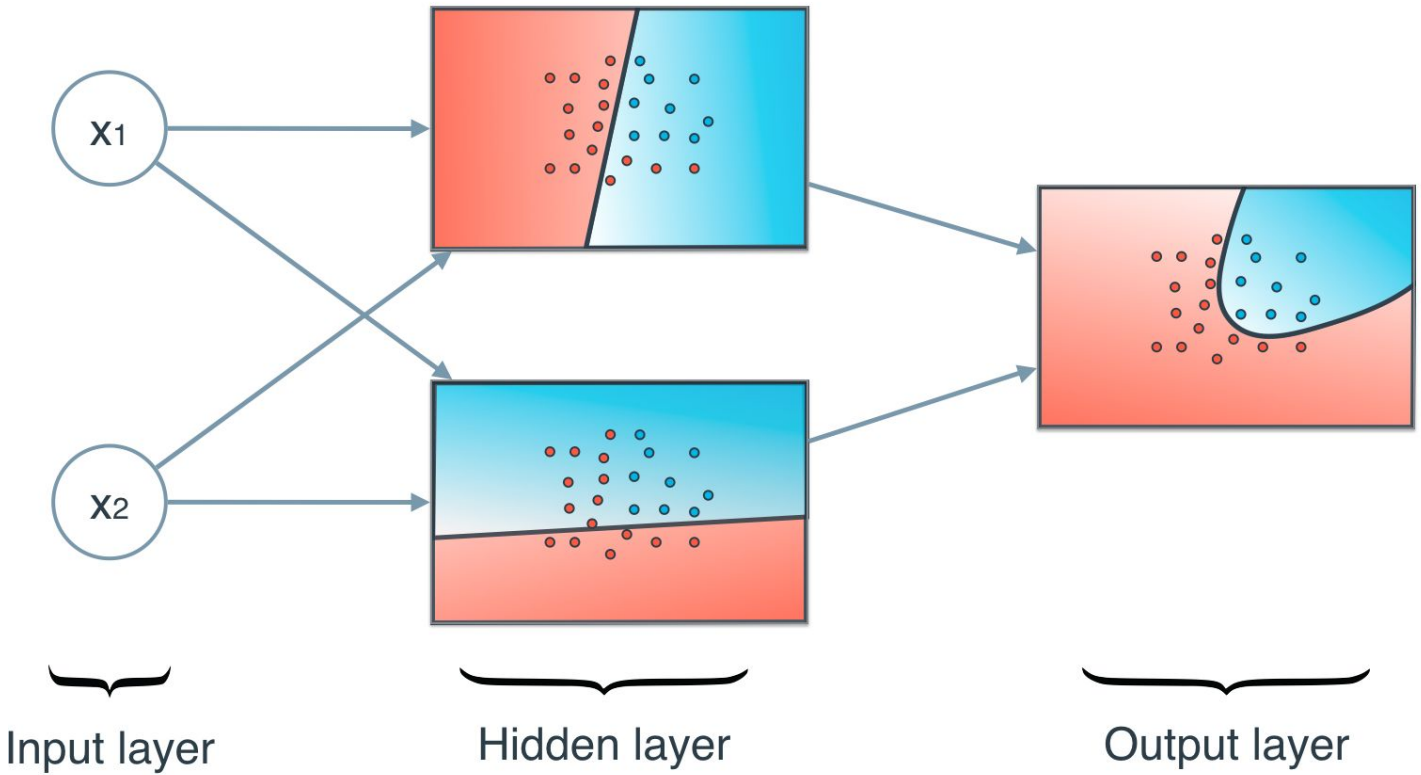
Full network details



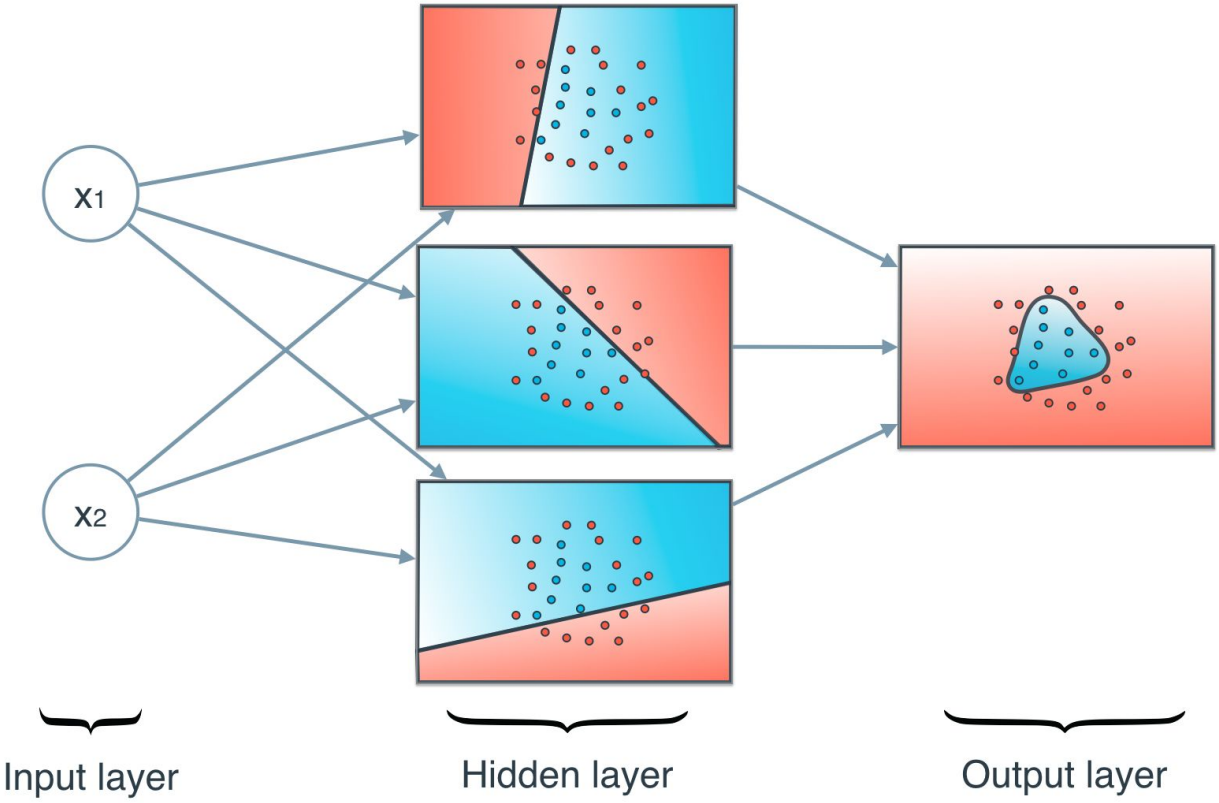
Simulation of a two layer NN



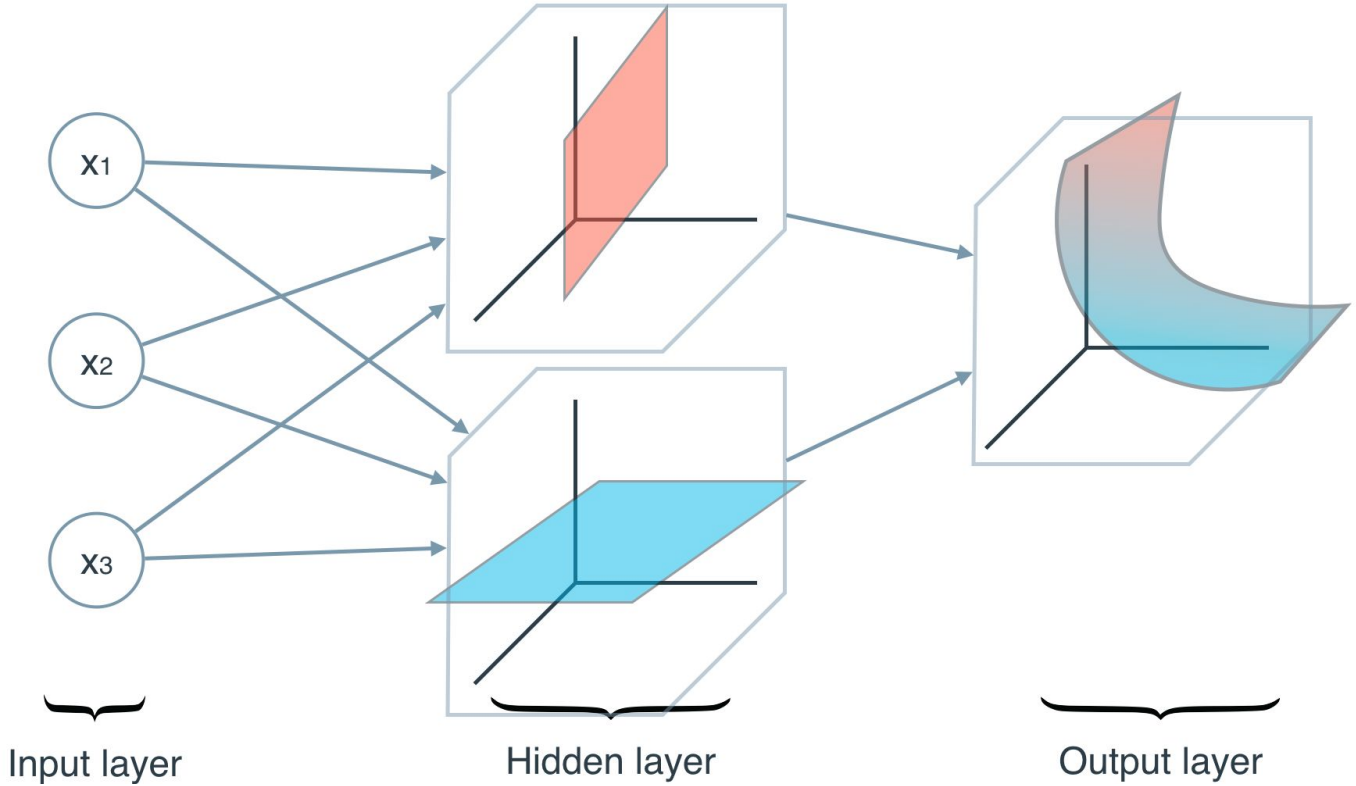
Neural Network Language



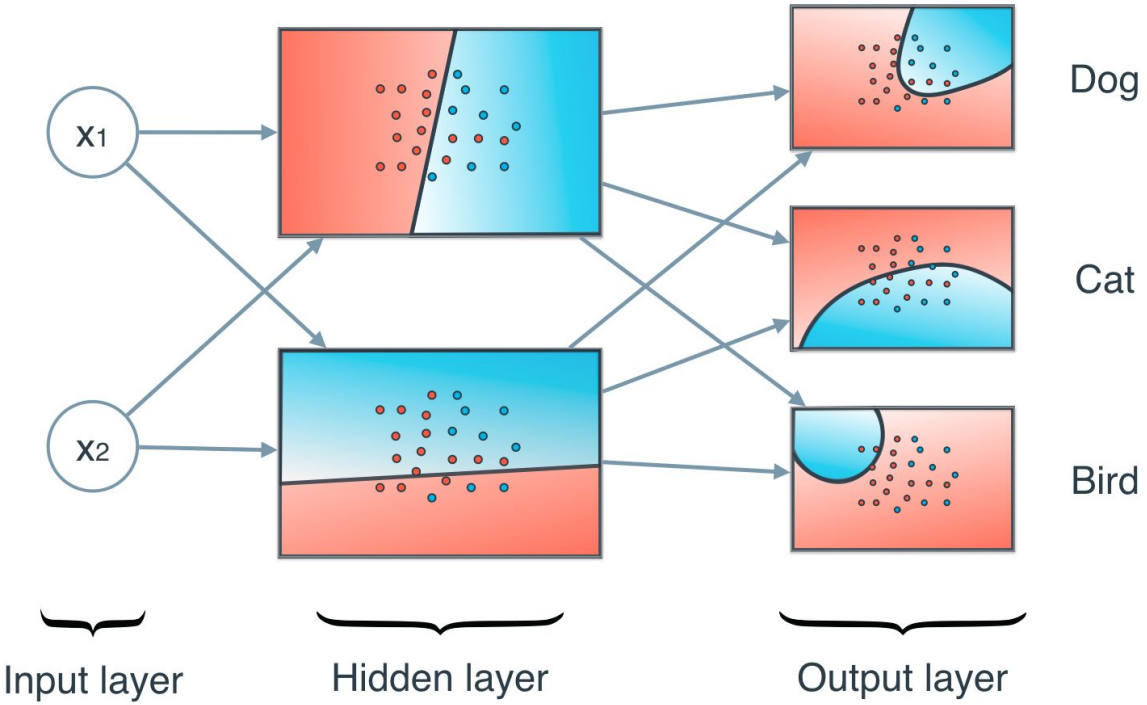
More than 2 Judges



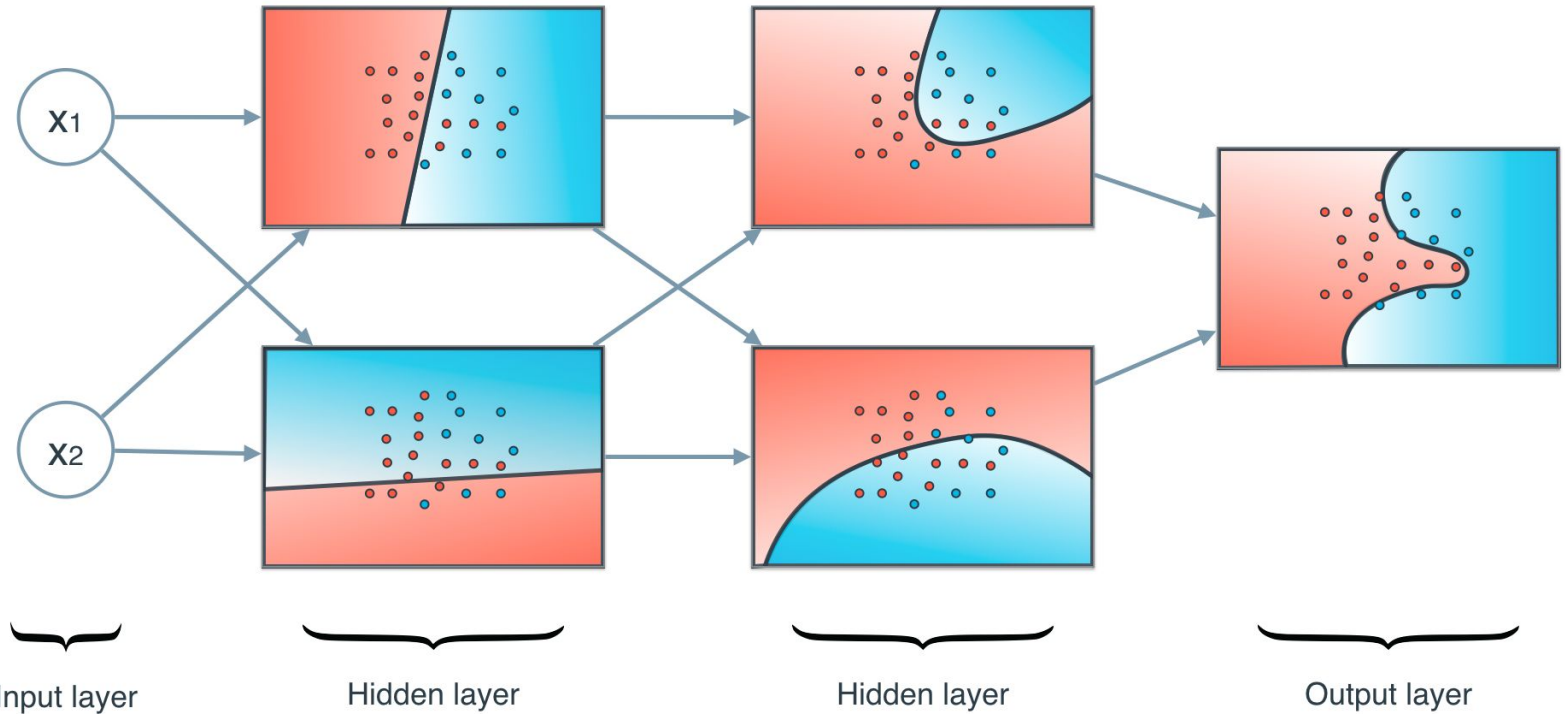
More than 2 dimensions



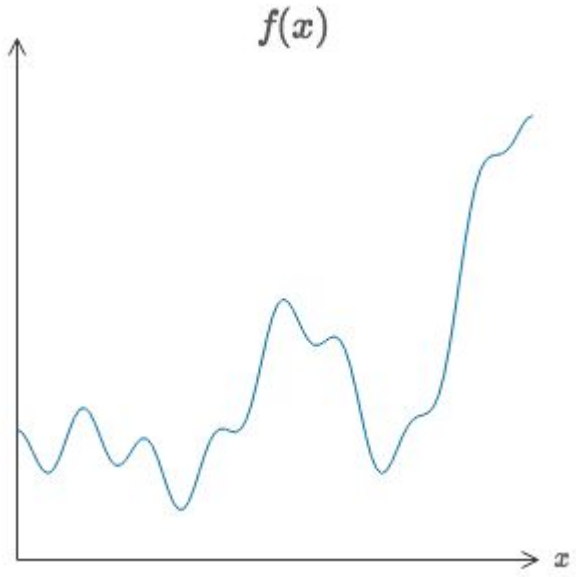
More than two outcomes

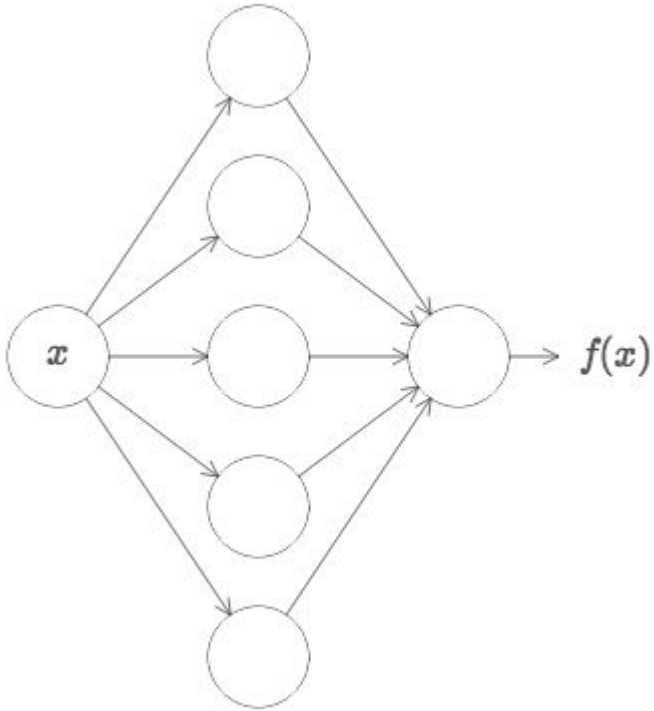


“Deeper” Networks



Neural Network can compute any function*





*Given enough hidden units they can approximate any function.

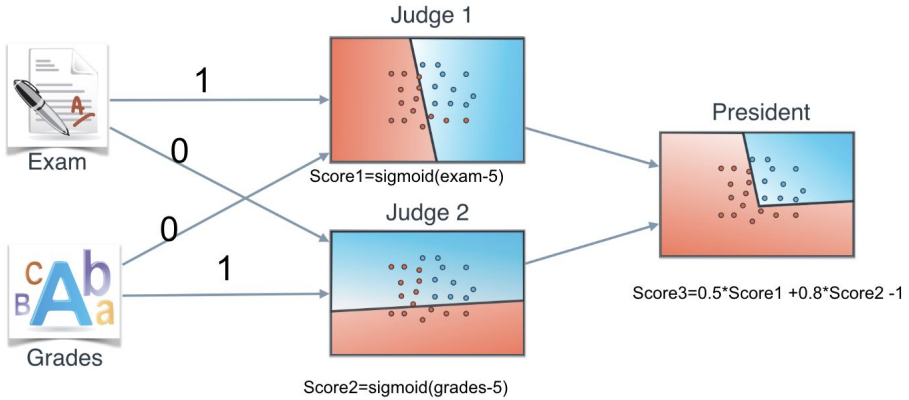
Feed Forward Neural Networks

Let x be in the input features $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$x_1 = Exam, x_2 = Grades$

$W^{[1]} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, b^{[1]} = \begin{bmatrix} -5 \\ -5 \end{bmatrix}$

$W^{[2]} = \begin{bmatrix} 0.5 & 0.8 \end{bmatrix}, b^{[2]} = \begin{bmatrix} -1 \end{bmatrix}$



Feed Forward Neural Networks

Let x be in the input features $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

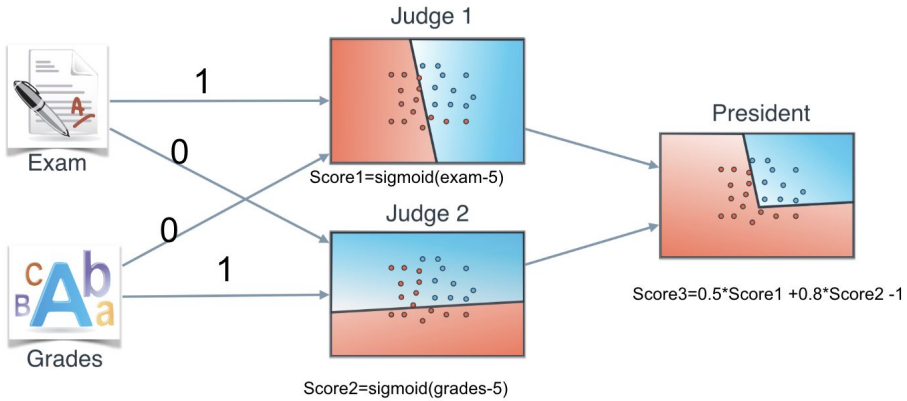
$x_1 = Exam, x_2 = Grades$

$W^{[1]} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, b^{[1]} = \begin{bmatrix} -5 \\ -5 \end{bmatrix}$

$W^{[2]} = \begin{bmatrix} 0.5 & 0.8 \end{bmatrix}, b^{[2]} = \begin{bmatrix} -1 \end{bmatrix}$

$a^{[1]} = \sigma(W^{[1]} \cdot x + b^{[1]})$

$\hat{y} = a^{[2]} = \sigma(W^{[2]} \cdot a^{[1]} + b^{[2]})$



Compute probability of acceptance for

Student: Exam = 5, Grades = 5

Let x be in the input features $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

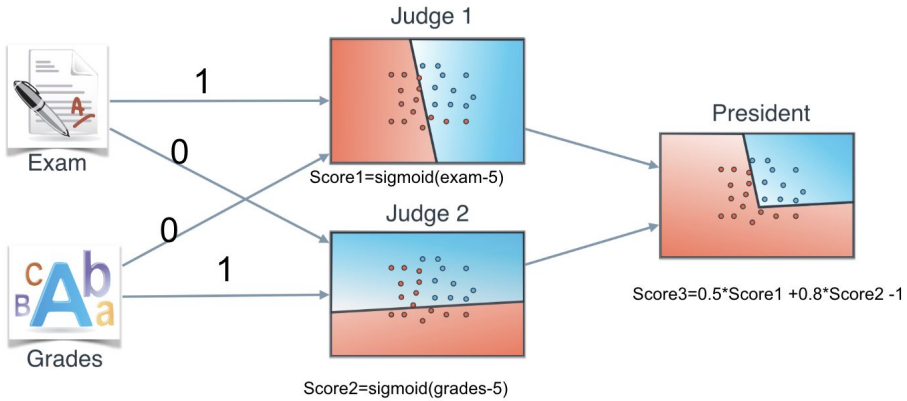
$x_1 = Exam, x_2 = Grades$

$W^{[1]} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, b^{[1]} = \begin{bmatrix} -5 \\ -5 \end{bmatrix}$

$W^{[2]} = \begin{bmatrix} 0.5 & 0.8 \end{bmatrix}, b^{[2]} = \begin{bmatrix} -1 \end{bmatrix}$

$a^{[1]} = \sigma(W^{[1]} \cdot x + b^{[1]})$

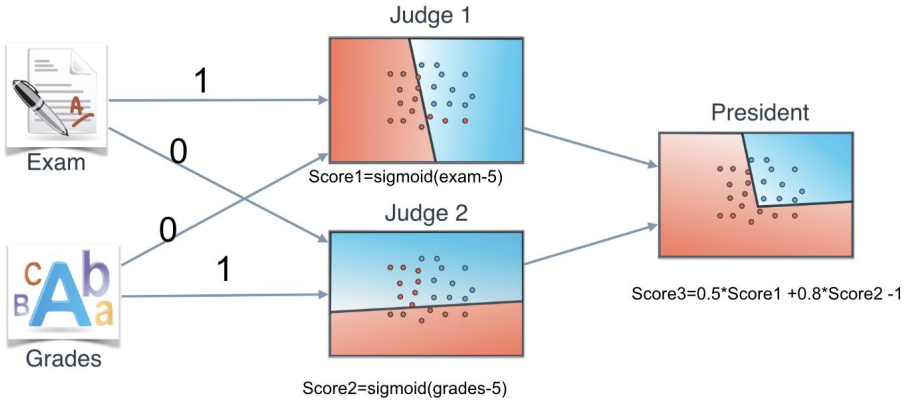
$\hat{y} = a^{[2]} = \sigma(W^{[2]} \cdot a^{[1]} + b^{[2]})$



Activation functions

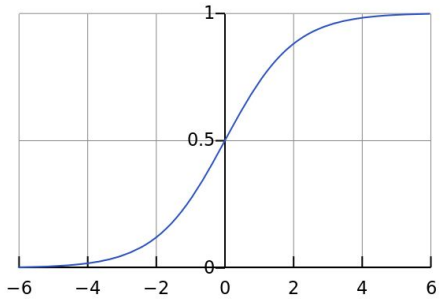
Activation function is the non-linear function used after th linear transformation

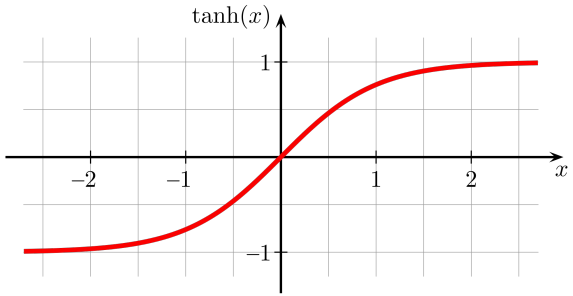
- For non-final layers many functions work
- For the **last** layer the function is usually defined by the **problem you are solving**

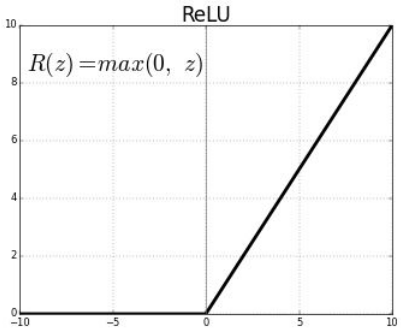


Activation functions for **non-final** layers

- **Sigmoid** is not used very much
- Tanh is used in RNNs
- Relu and “Relu like” functions are used everywhere



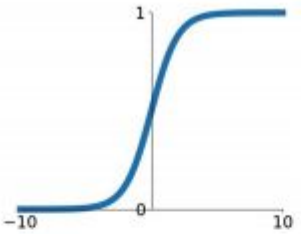




Activation functions for **non-final** layers

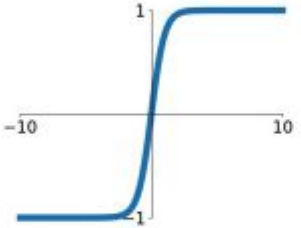
Sigmoid

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



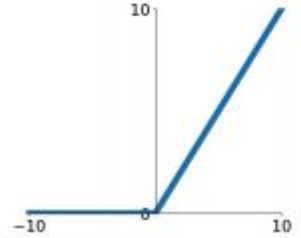
tanh

$$\tanh(x)$$



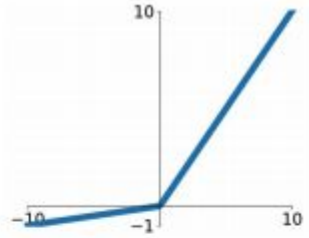
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

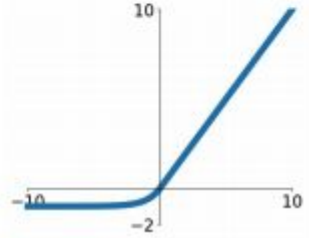


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

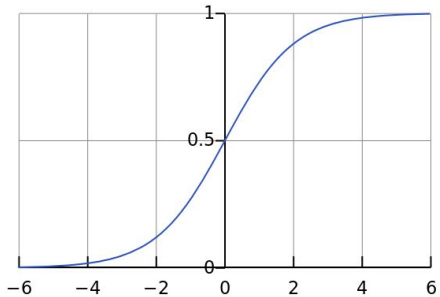
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



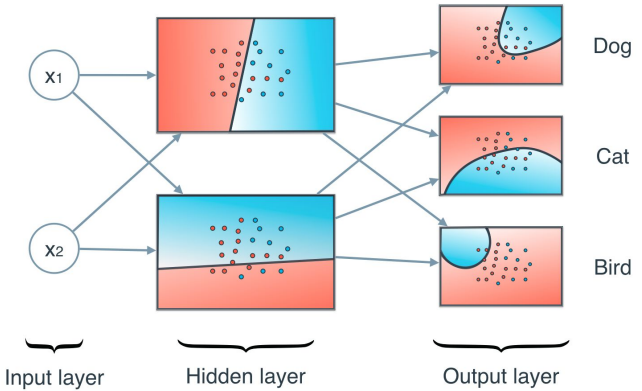
Activation functions for **final** layers

- Are defined by the type of problem you are trying to solve
- **Sigmoid** is used for **binary classification**
- **Identity** is use for **regression**
- **Softmax** is used for multi-class classification



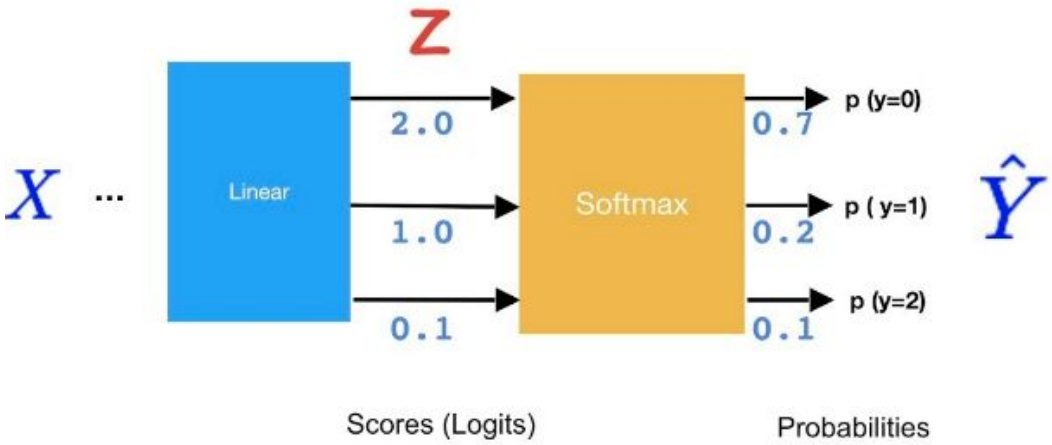
Softmax activation function

- After the last linear transformation we get K numbers (K -classes)
- These numbers need to be normalized to sum to 1



$$z^{[1]} = W^{[1]} \cdot x + b^{[1]}$$
$$a^{[1]} = relu(z^{[1]})$$
$$z^{[2]} = W^{[2]} \cdot a^{[1]} + b^{[2]}$$
$$\hat{y} = softmax(z^{[2]})$$

Softmax activation function



$$z^{[1]} = W^{[1]} \cdot x + b^{[1]}$$

$$a^{[1]} = relu(z^{[1]})$$

$$z^{[2]} = W^{[2]} \cdot a^{[1]} + b^{[2]}$$

$$\hat{y} = softmax(z^{[2]})$$

$$softmax(z) = \left(\frac{e^{z_1}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_K}}{\sum_j e^{z_j}}\right)$$

Softmax Practice

Prediction: $\hat{y} = (\hat{y}_{dog}, \hat{y}_{cat}, \hat{y}_{bird})$

Case 1: $z^{[2]} = \begin{bmatrix} 0.1 \\ 10 \\ 100 \end{bmatrix}$, $\hat{y} = ?$, $y = (0, 1, 0)$

Case 2: $z^{[2]} = \begin{bmatrix} 0.1 \\ 10 \\ 100 \end{bmatrix}$, $\hat{y} = ?$, $y = (0, 0, 1)$

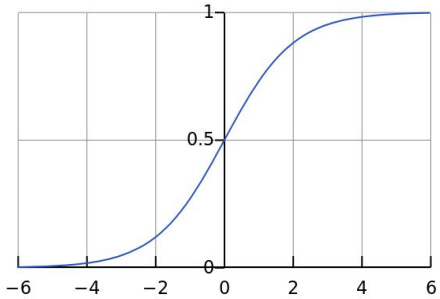
$$z^{[1]} = W^{[1]} \cdot x + b^{[1]}$$
$$a^{[1]} = relu(z^{[1]})$$
$$z^{[2]} = W^{[2]} \cdot a^{[1]} + b^{[2]}$$
$$\hat{y} = softmax(z^{[2]})$$

$$softmax(z) = \left(\frac{e^{z_1}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_K}}{\sum_j e^{z_j}}\right)$$

Compute the prediction and loss of these two examples.

Hard predictions for classification

Binary classification:
soft prediction: $\hat{y} = \sigma(z^{[2]})$
hard prediction: $\hat{y} > \mathit{threshold}$
often $\mathit{threshold} = 0.5$
$\hat{y} > 0.5$ is equivalent to $z^{[2]} > 0$



Multi-class classification:
soft prediction: $\hat{y} = \mathit{softmax}(z^{[2]})$
hard prediction:
arg max of $\hat{y}$ or equivalently *arg* max of $z^{[2]}$

Fitting two layer neural networks

Parameters: $\{W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}\}$
Error / Cost / Loss functions:
Regression:

$$Err(w) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

Binary classification:

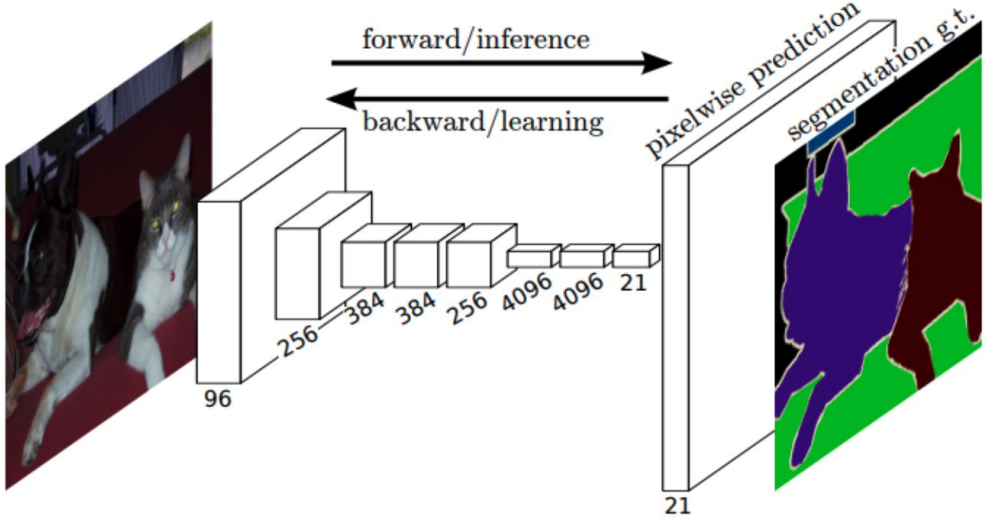
$$Err(w) = -\frac{1}{N}\sum_{i=1}^{N}[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

Multi-class classification:

$$Err(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)}$$

Applications of Deep Learning in Medicine and Physics

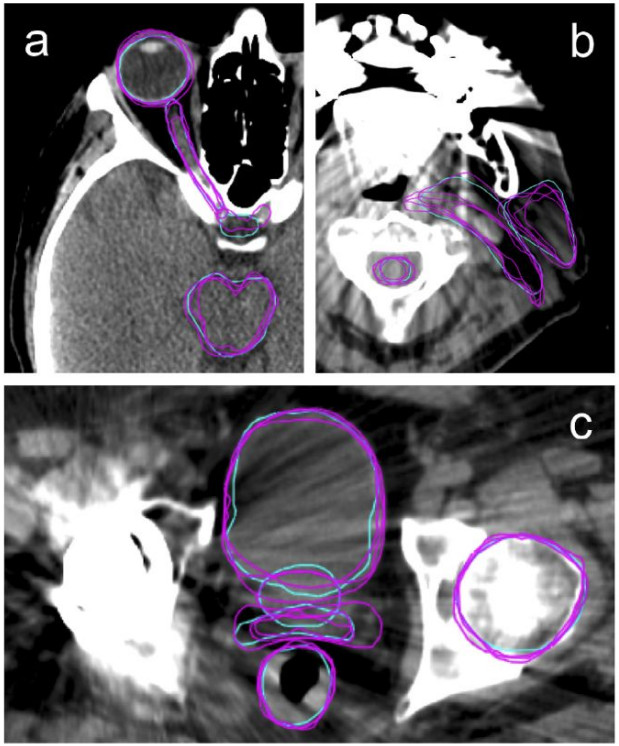
Image to Image Models



Unet Models are able to take an input image and return an output image.

Segmentation task: every output pixel is a class.

Segmentation/ Contour of the Target Volumes and Organs at Risk (OAR)

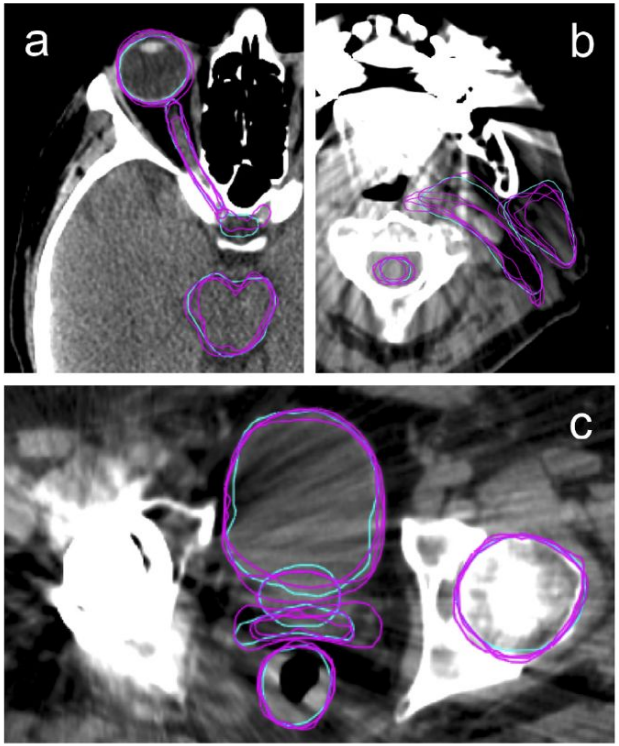


https://pubmed.ncbi.nlm.nih.gov/31812930/

To plan radiation therapy doctors need to contour target volumes and organs at risk

- Designing radiation treatment plans
- Limit radiation exposure and minimize the risk of damage

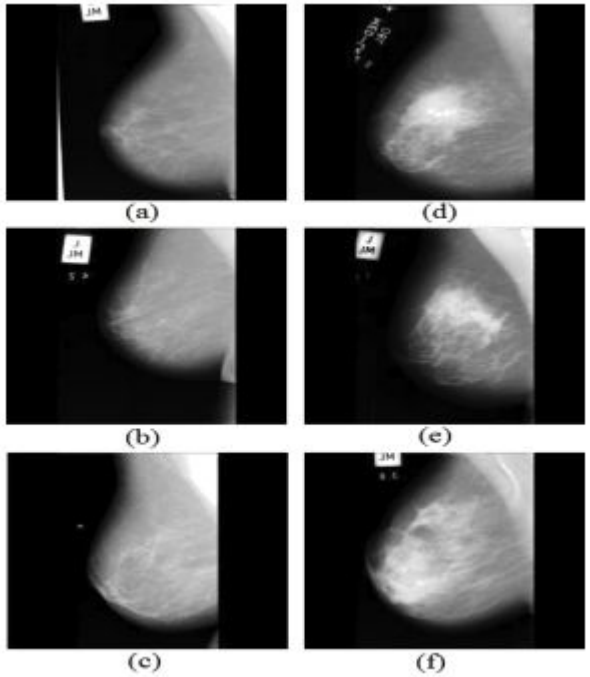
Segmentation/ Contour of the Target Volumes and Organs at Risk (OAR)



- Segmentation is labor-intensive
- Neural network can generate contours comparable to human experts

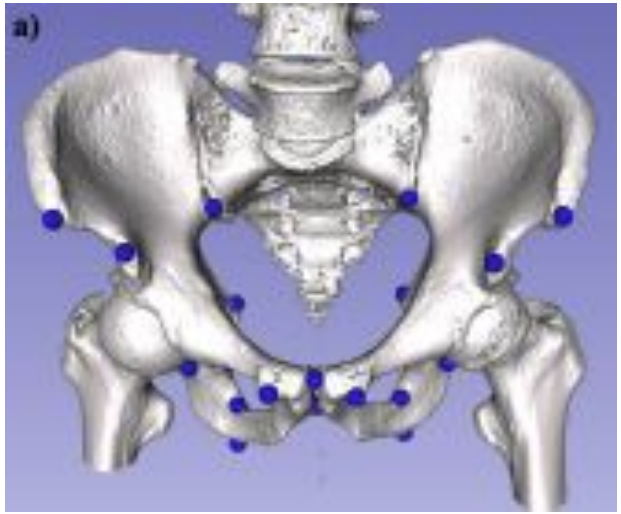
https://pubmed.ncbi.nlm.nih.gov/31812930/

Detecting Breast Cancer in Mammograms



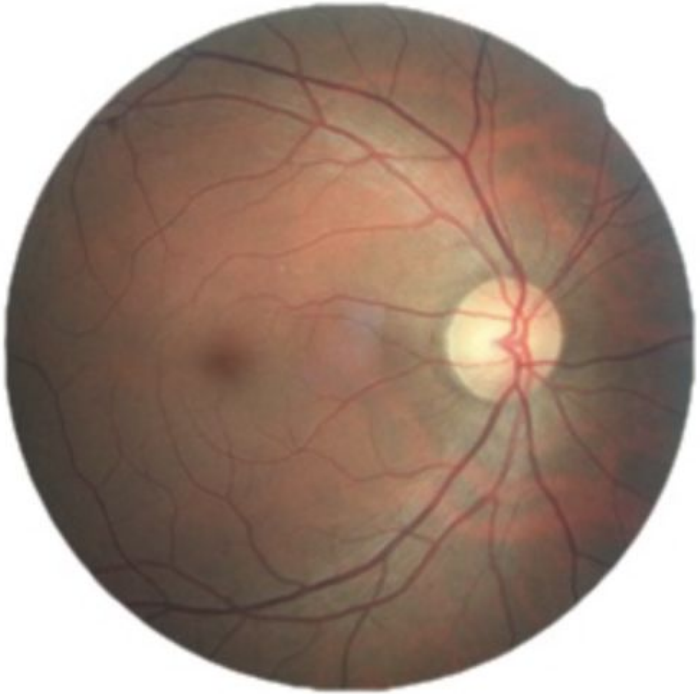
Convolutional Neural Networks can be used to detect cancer in mammograms.

Detecting Anatomical Landmarks for Pelvic Trauma Surgery



The automated identification of landmarks in X-ray images can enhance decision-making during surgical procedures.

Detecting Diabetic retinopathy



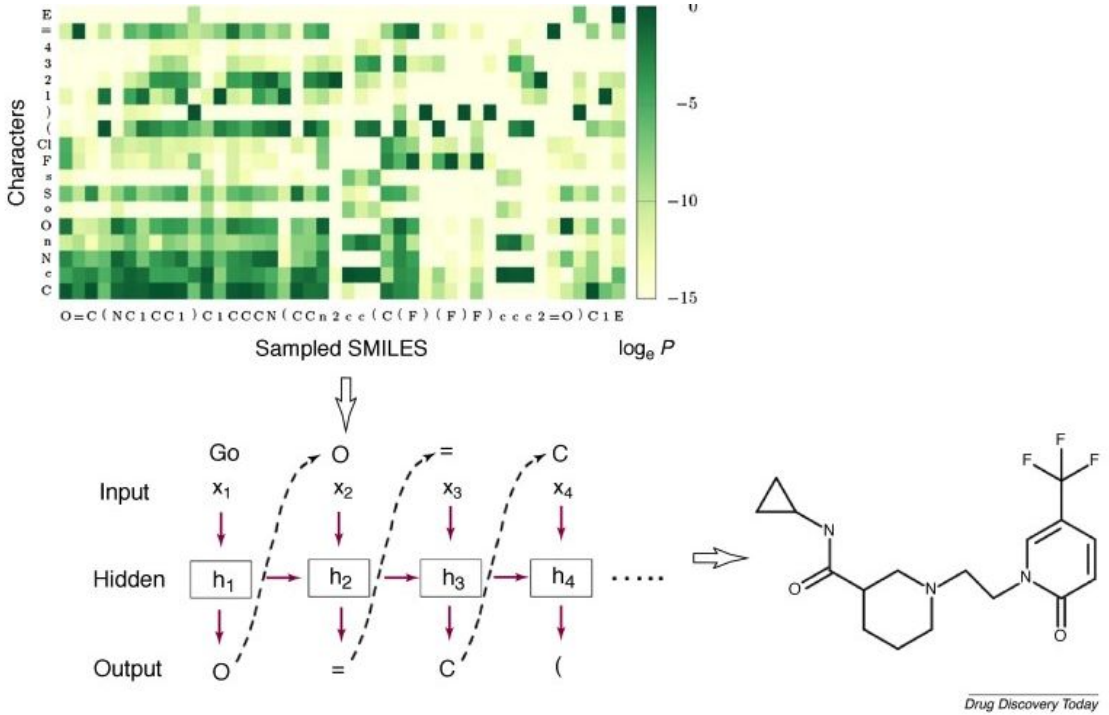
Diabetic retinopathy is a leading cause of blindness in the developed world.

Convolutional Neural Networks are being used for this task.

https://en.wikipedia.org/wiki/Diabetic_retinopathy

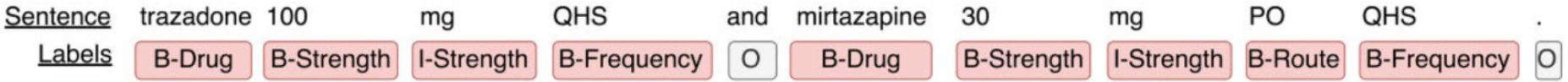
Drug Discovery: generation of new molecular structures

Goal: generate new chemical structures that are likely to have specific biological activity.



https://www.sciencedirect.com/science/article/pii/S1359644617303598

Detecting adverse drugs events from medical records



Goal: extraction of medications and associated adverse drug events (ADEs) from clinical documents.

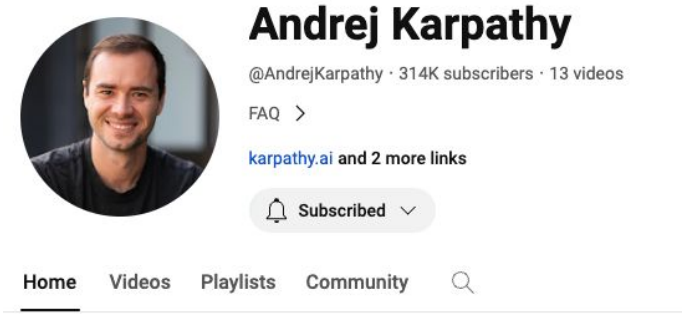
Step 1: Named entity recognition

Step 2: Relation classification

https://academic.oup.com/jamia/article-abstract/27/1/13/5499225?

Convolutional Neural Networks

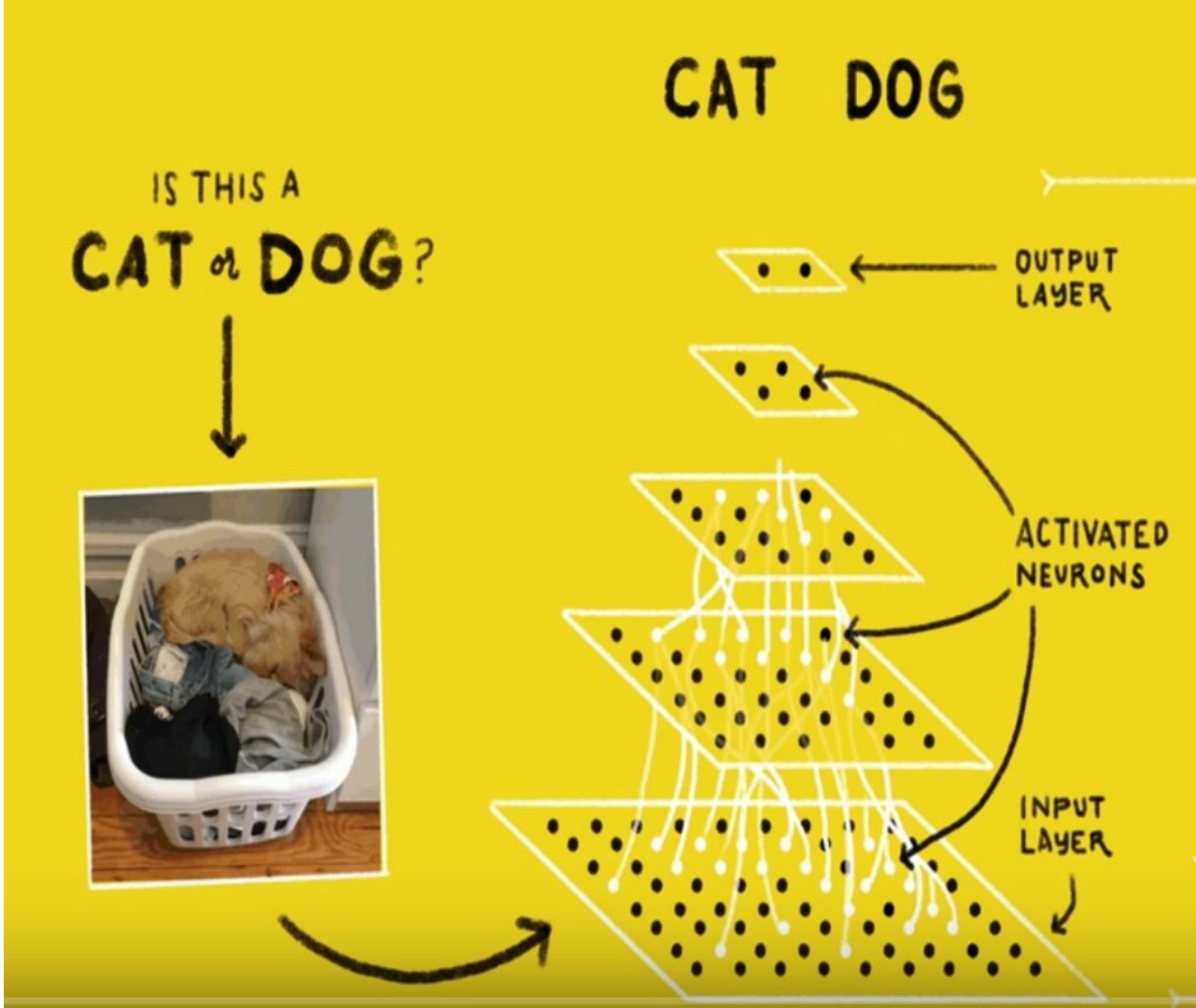
- Intro to Convolutional Neural Networks
- Convolutional operation
- Convolutional layers
- Max pool, Dense Layers
- Case Study LeNet-5



CS231n Winter 2016 on youtube

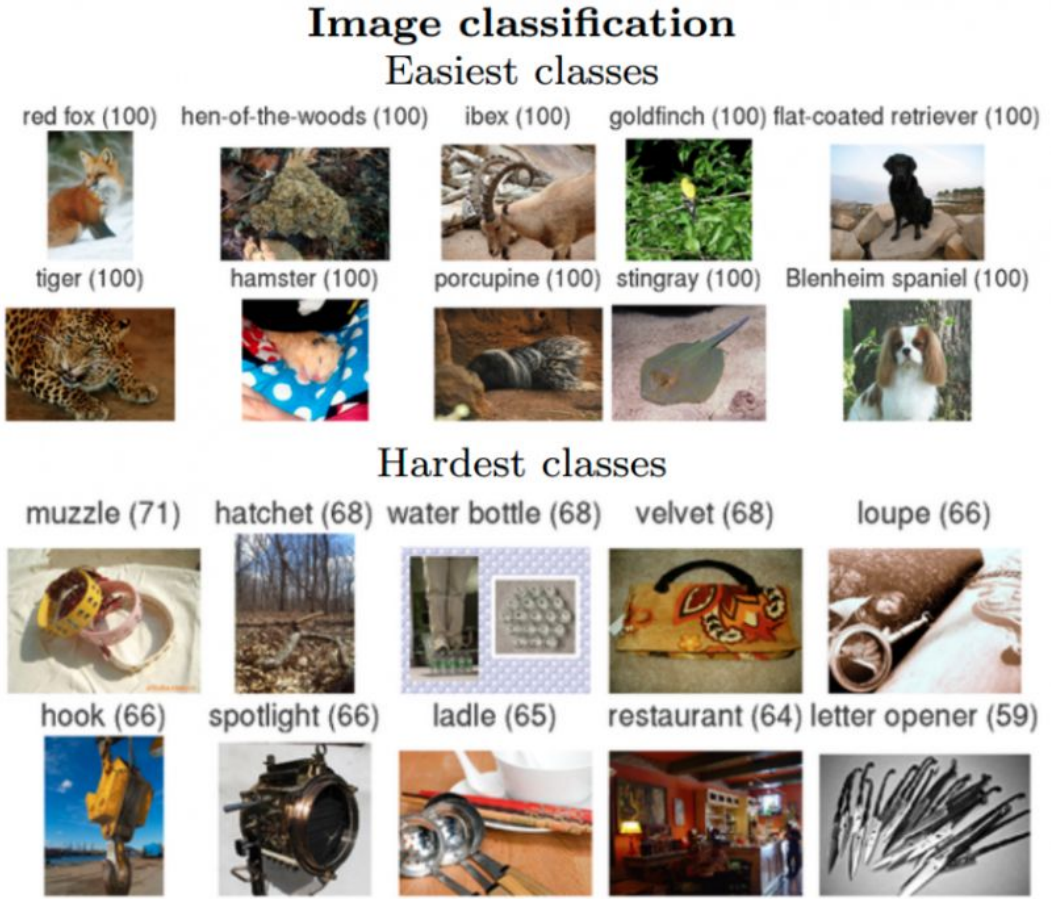
“Summary of Deep Learning”

https://www.youtube.com/watch?v=UAq961jQjYg



ImageNet Challenge

- Competition that ran since 2010 to evaluate image recognition algorithms.
- The training data: 1.2 million images
- Tasks:
  - Classification into 1000 categories
  - Object detection

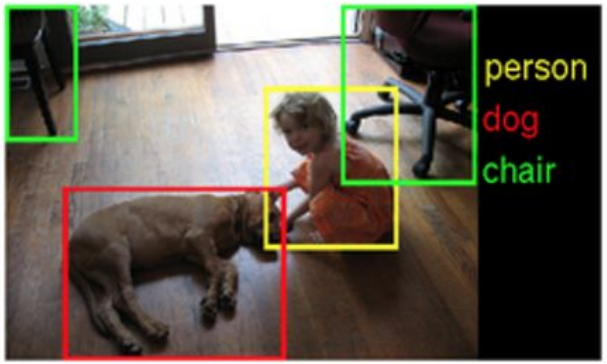


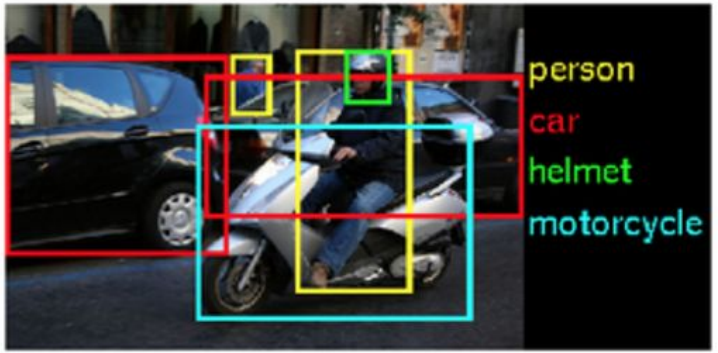
ImageNet Challenge

The training data: 1.2 million images

Tasks:

- Classification into 1000 categories
- Object detection:
  - find a particular object and draw a box around it

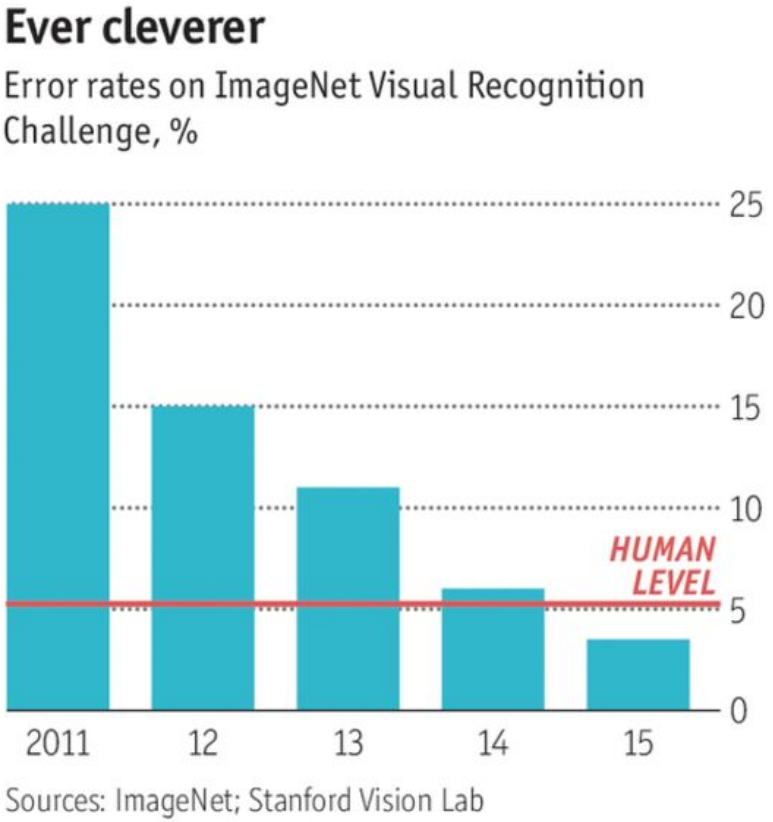




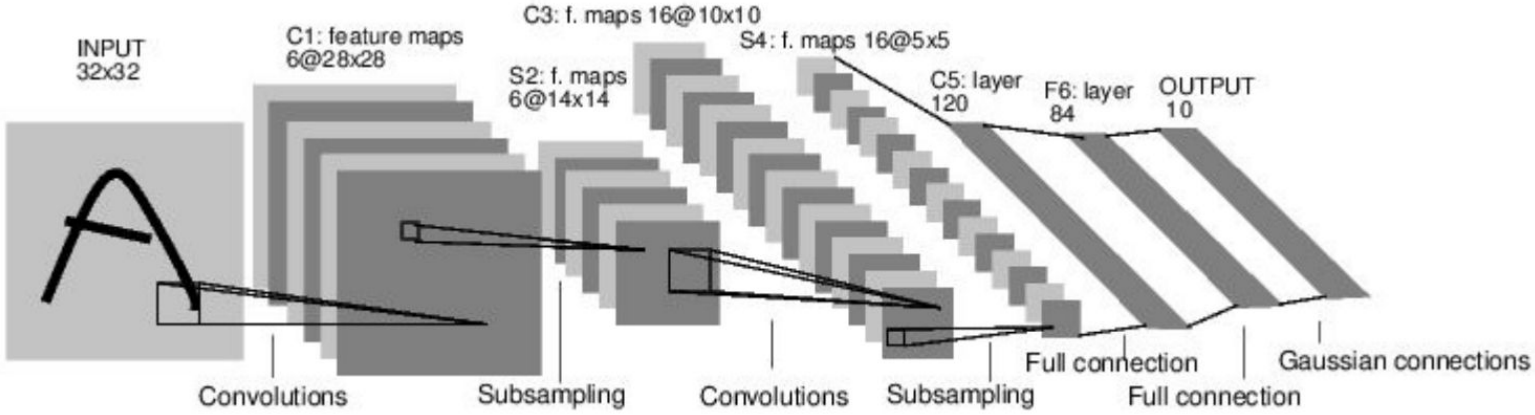
Start of the deep learning “revolution”

- ImageNet Challenge 2012
- Convolutional neural network (CNN) called AlexNet from University of Toronto won the competition
  - First time a neural network won the competition
  - 16.4% error, while the second place had 26.2% error
  - The Economist, "Suddenly people started to pay attention, not just within the AI community but across the technology industry as a whole."

Better than humans



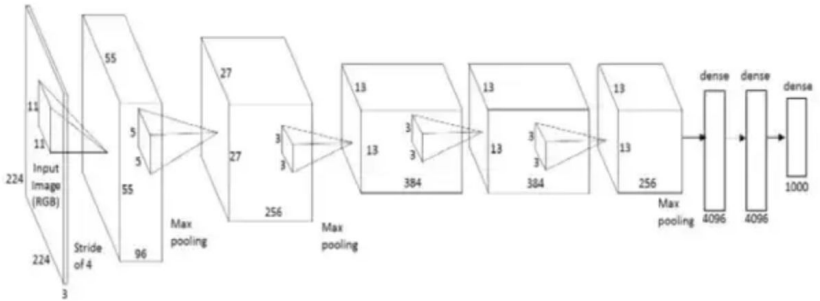
Convolutional Neural Networks (CNN): History



[LeNet-5, LeCun 1980]

- CNNs were design to mimic how researches in image classification think about images
- CNNs have much less parameters than fully connected neural networks

Alexnet wins Imagenet in 2012



Alex Net is very similar to the LetNet5

- similar architecture
- different activation function (relu)
- better initialization
- trained on GPU
- smaller filters than LeNet5

ImageNet Classification with Deep. Convolutional Neural Networks. Alex Krizhevsky. Ilya Sutskever. Geoffrey Hinton.

CNN are everywhere

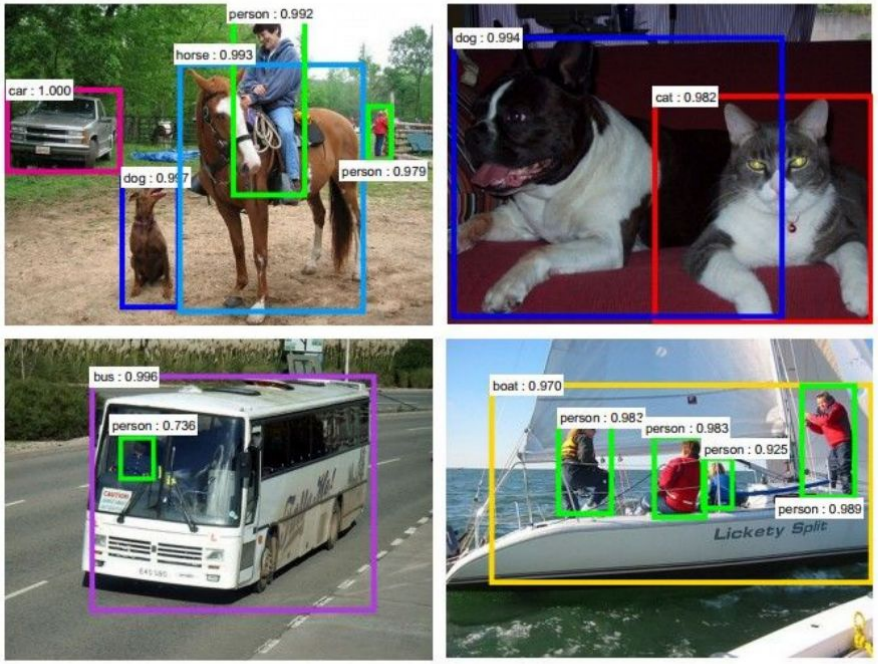
Classification

Retrieval

[Krizhevsky 2012]

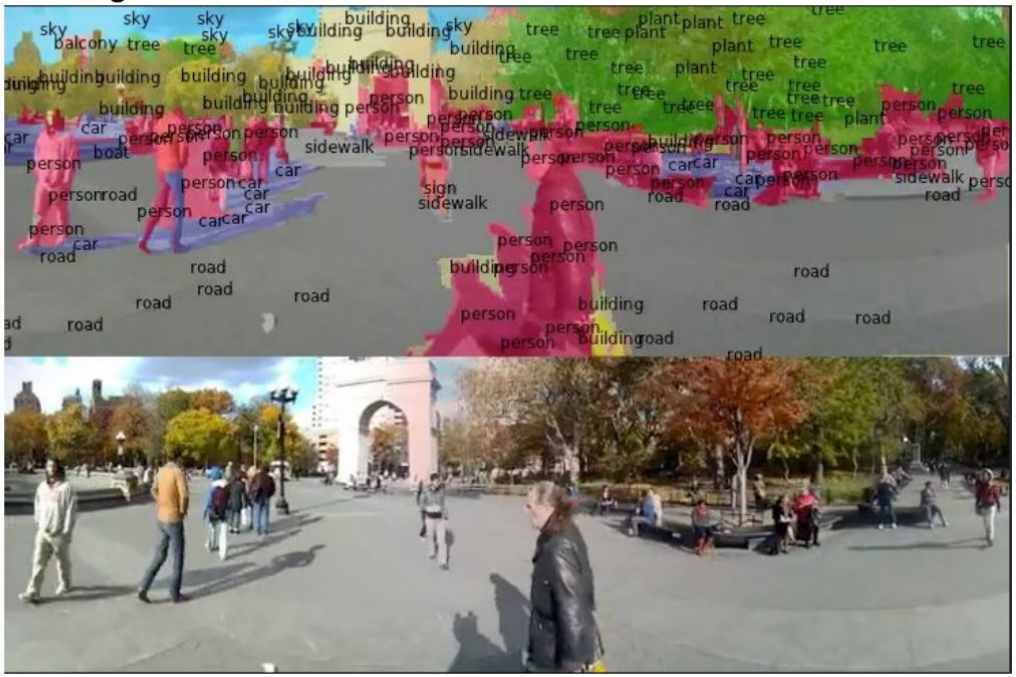
CNN are everywhere

Detection



[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



[Farabet et al., 2012]

CNN are everywhere

Self driving cars

CNN are everywhere

[Toshev, Szegedy 2014]

[Mnih 2013]

Estimate poses, play computer games

CNN are everywhere

Whale recognition, Kaggle Challenge

Mnih and Hinton, 2010

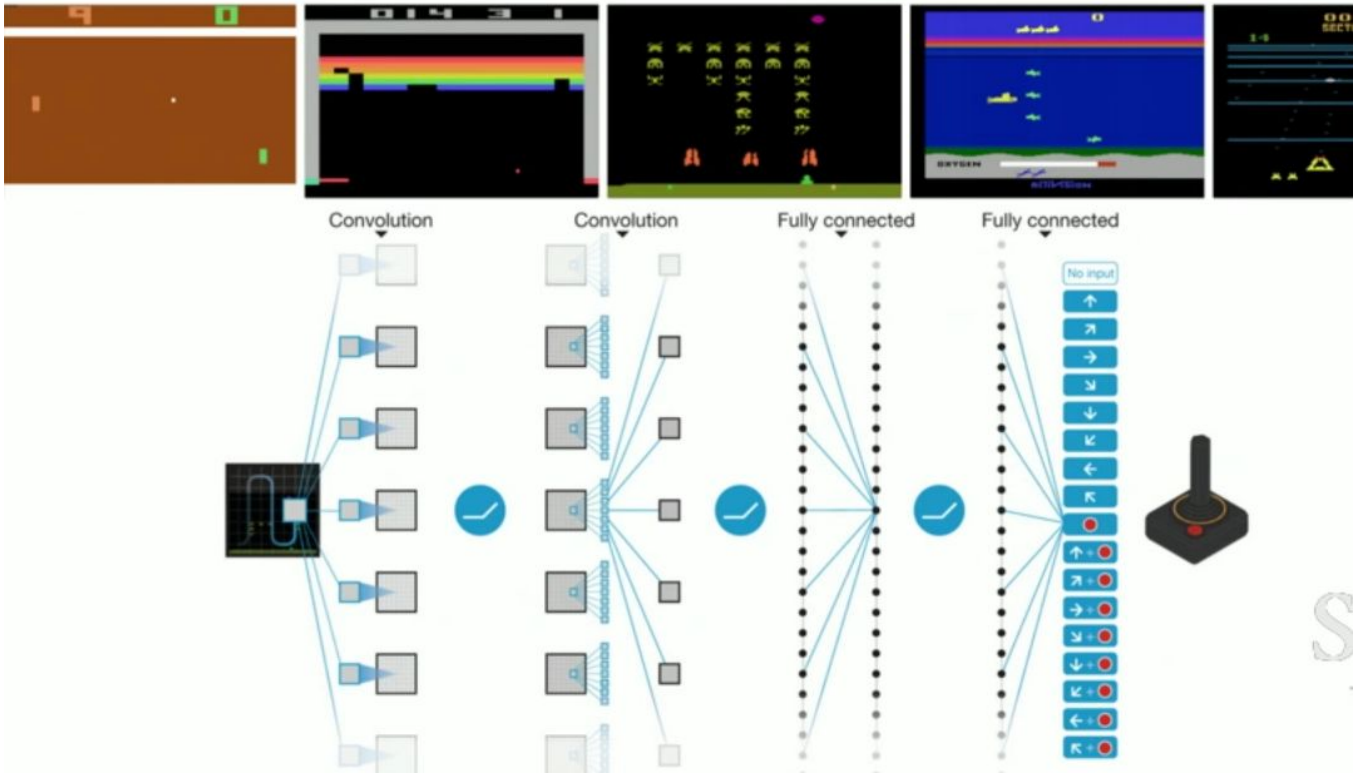
Image Captioning

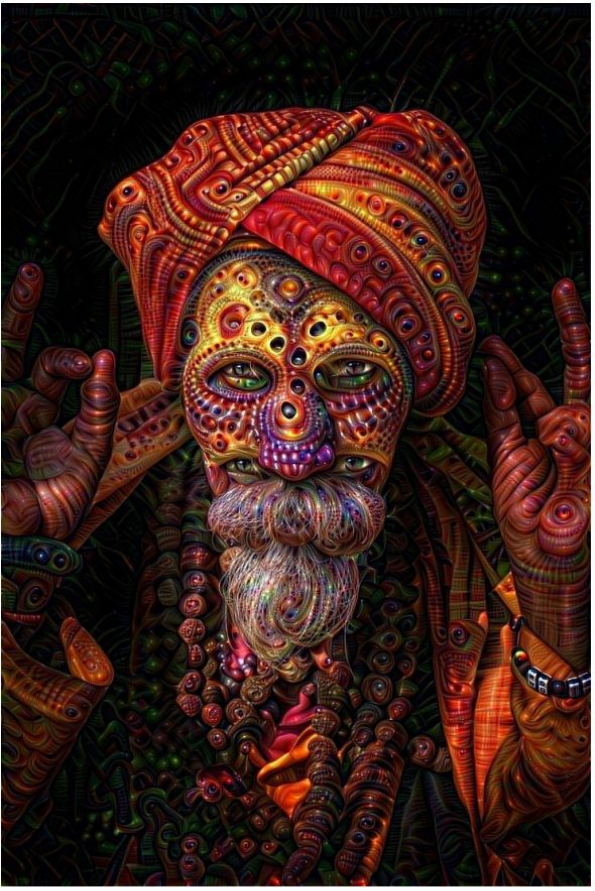
[Vinyals et al., 2015]

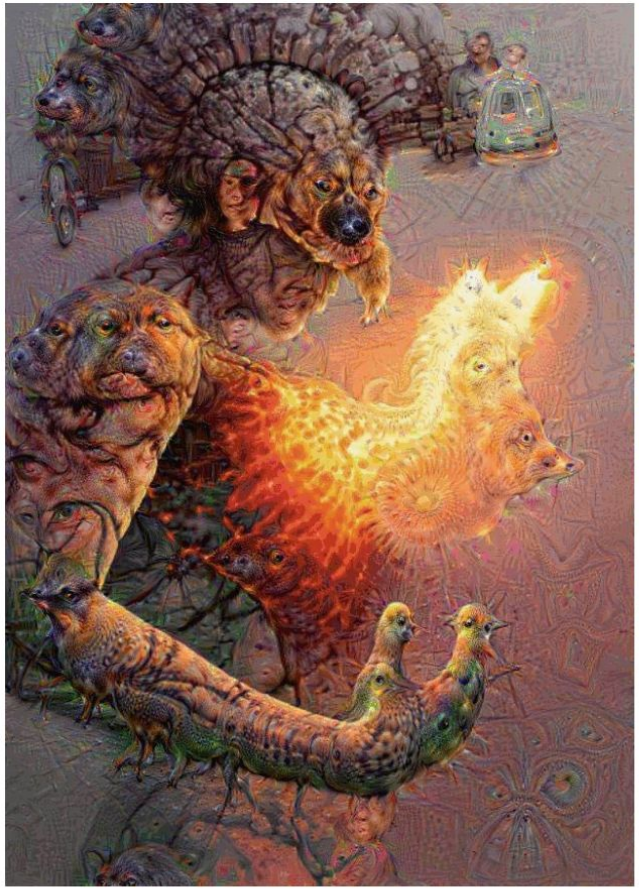
Given an image the CNN will generate text describing the image

Deep Reinforcement Learning

Human-level control through deep reinforcement learning,
Mnih et al, Nature 2015



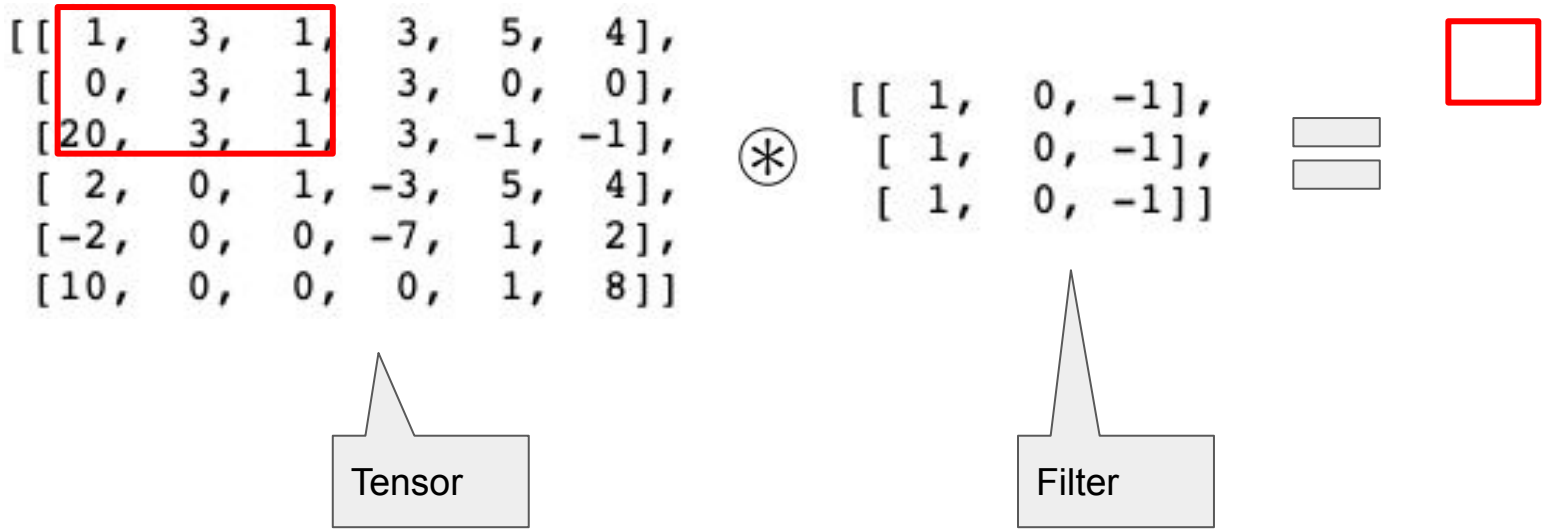




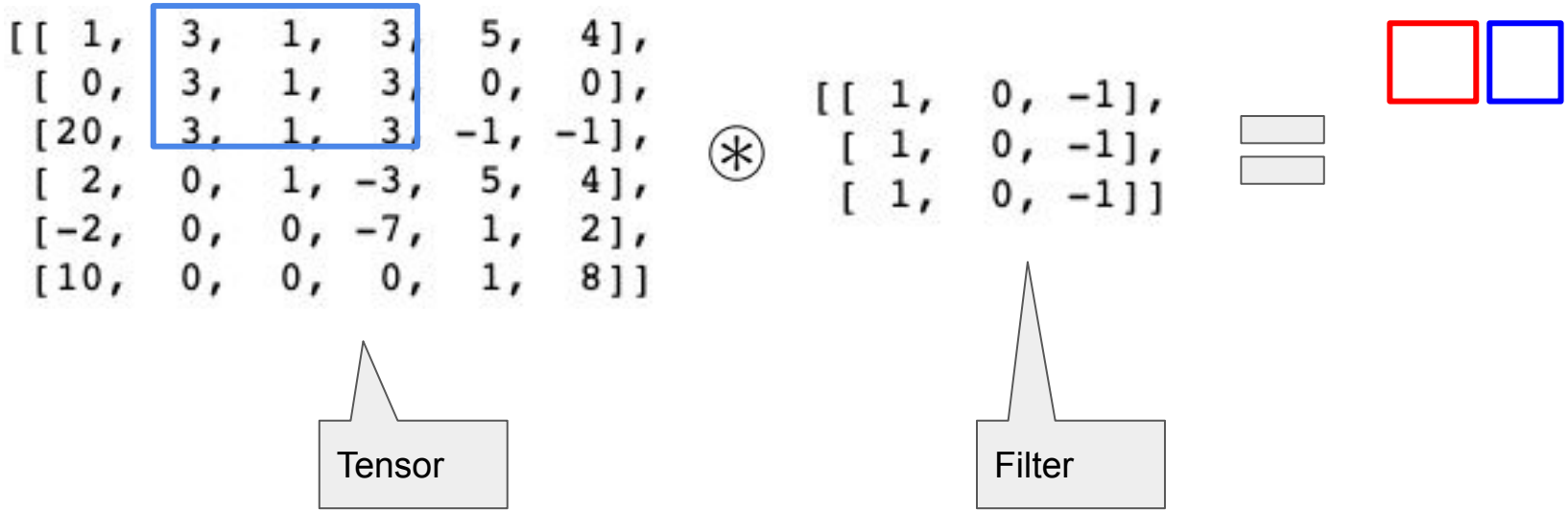
reddit.com/r/deepdream

Style Transfer

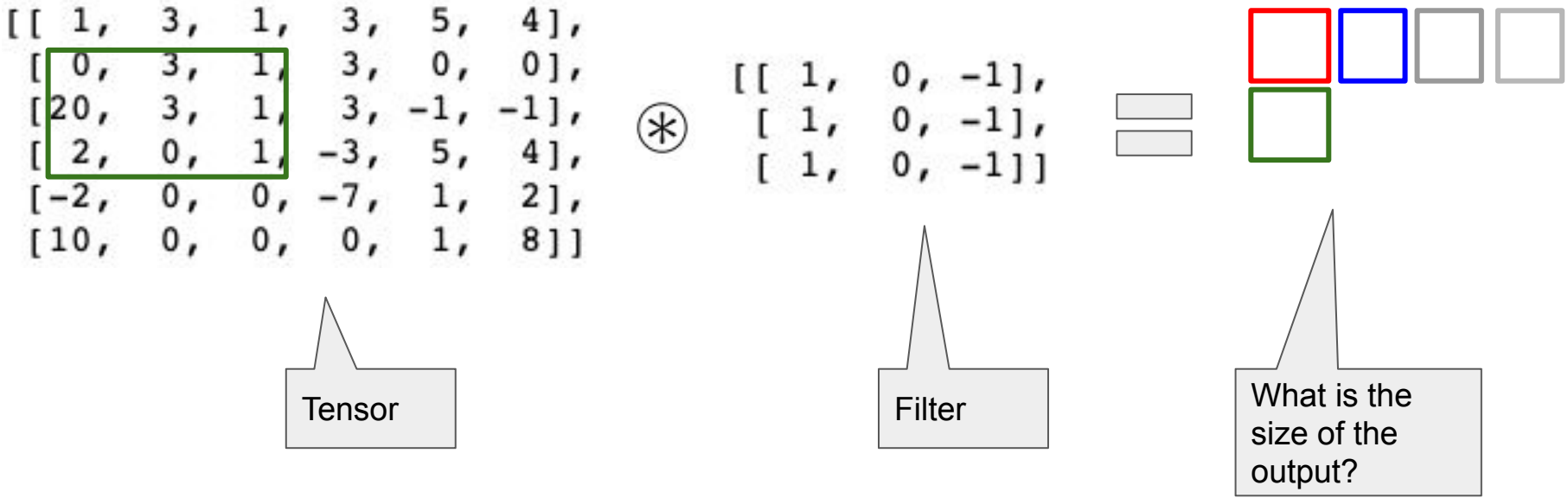
What is a 2D convolution?



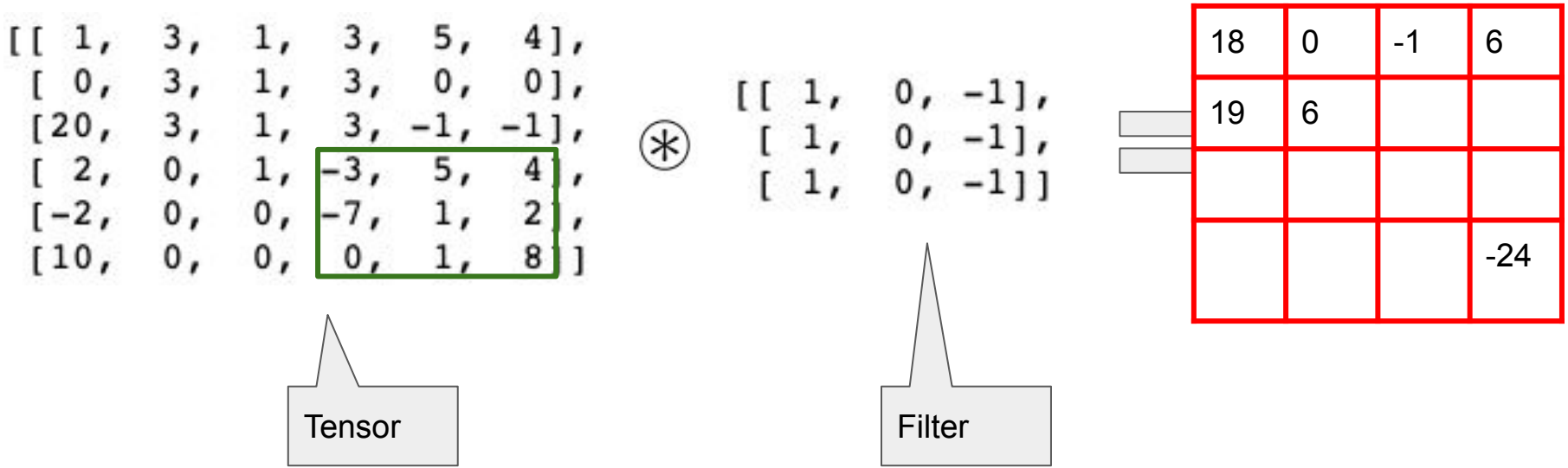
What is a 2D convolution?



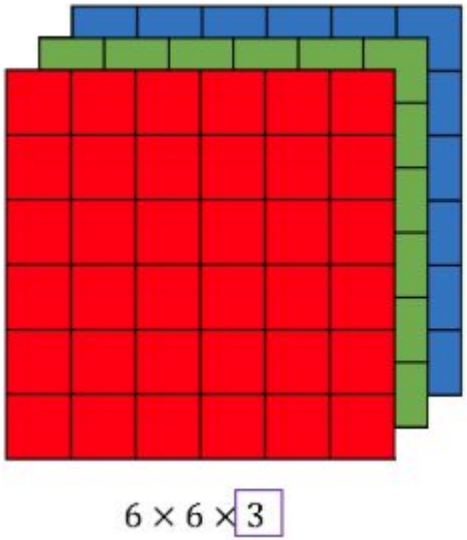
What is a 2D convolution?



What is a 2D convolution?

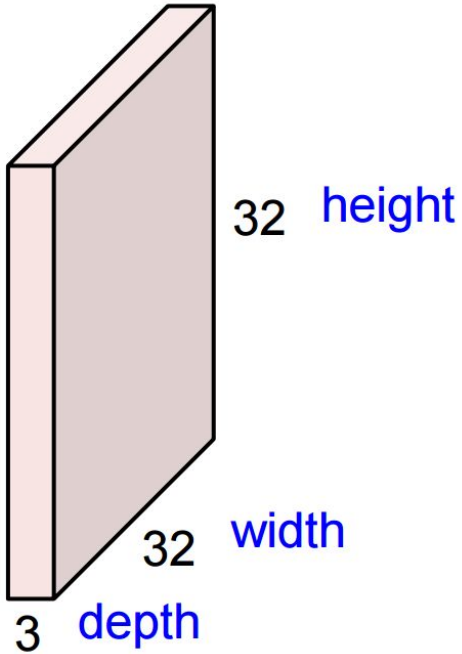


2D Convolution with RGB images



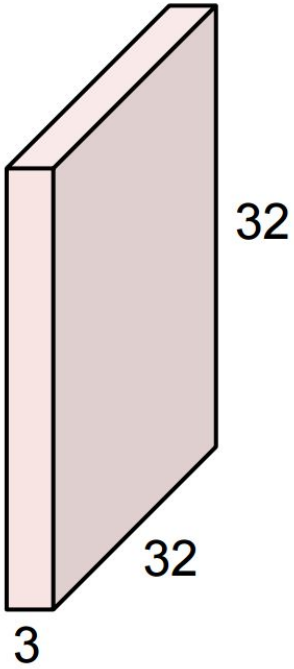
Convolution Layer

32x32x3 image

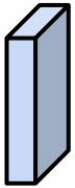


Convolution Layer

32x32x3 image



5x5x3 filter

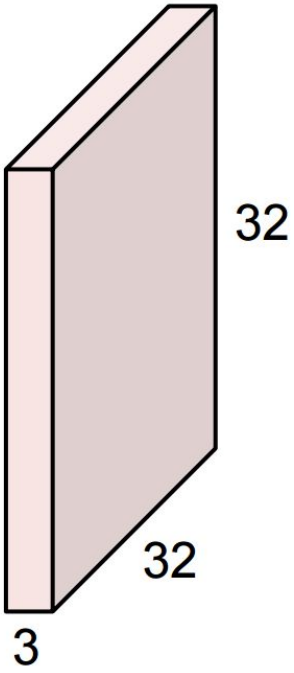


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

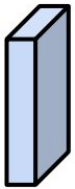
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

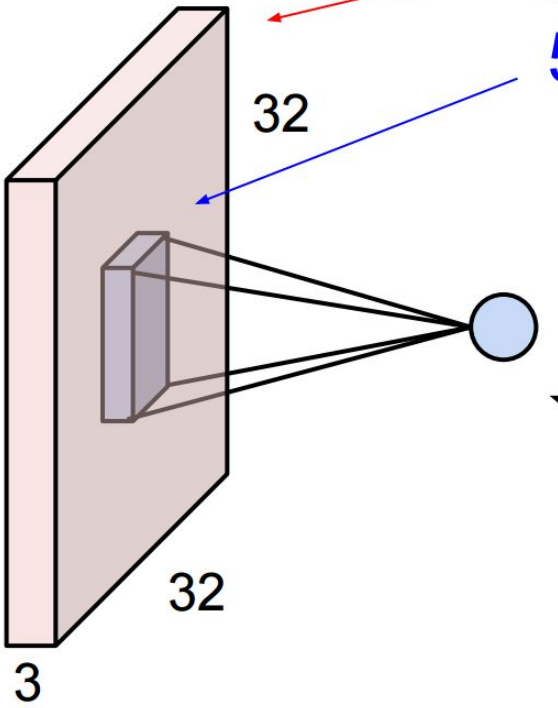


5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer



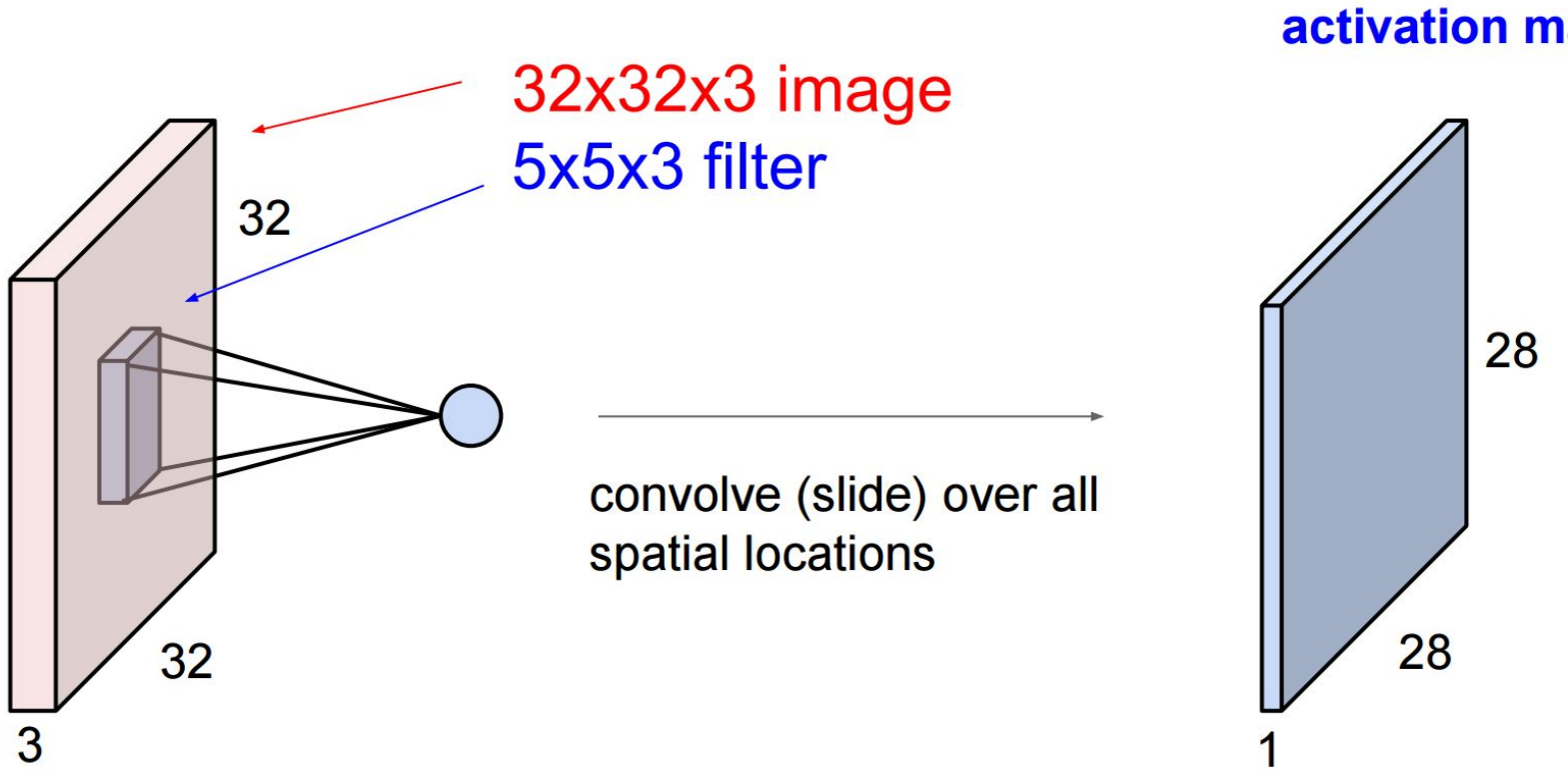
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

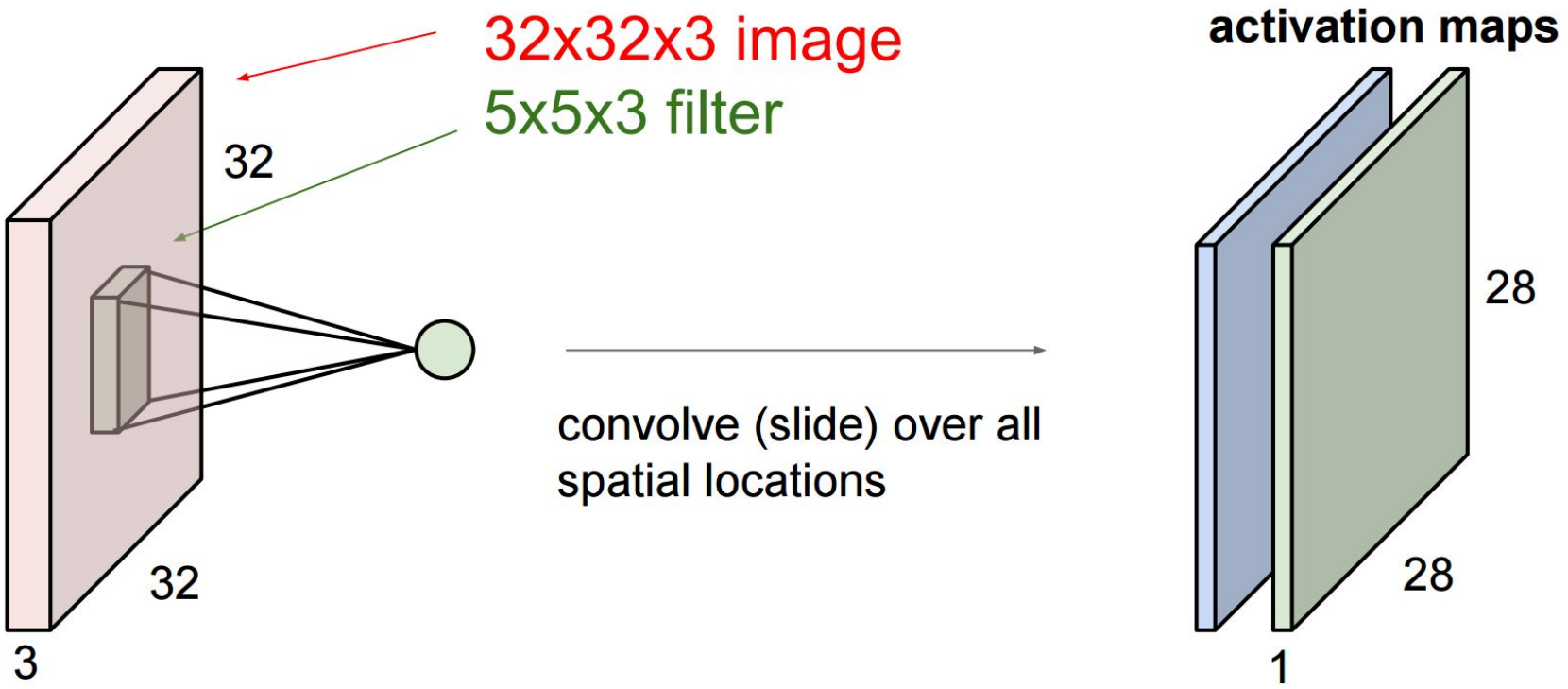
$$w^T x + b$$

Convolution Layer

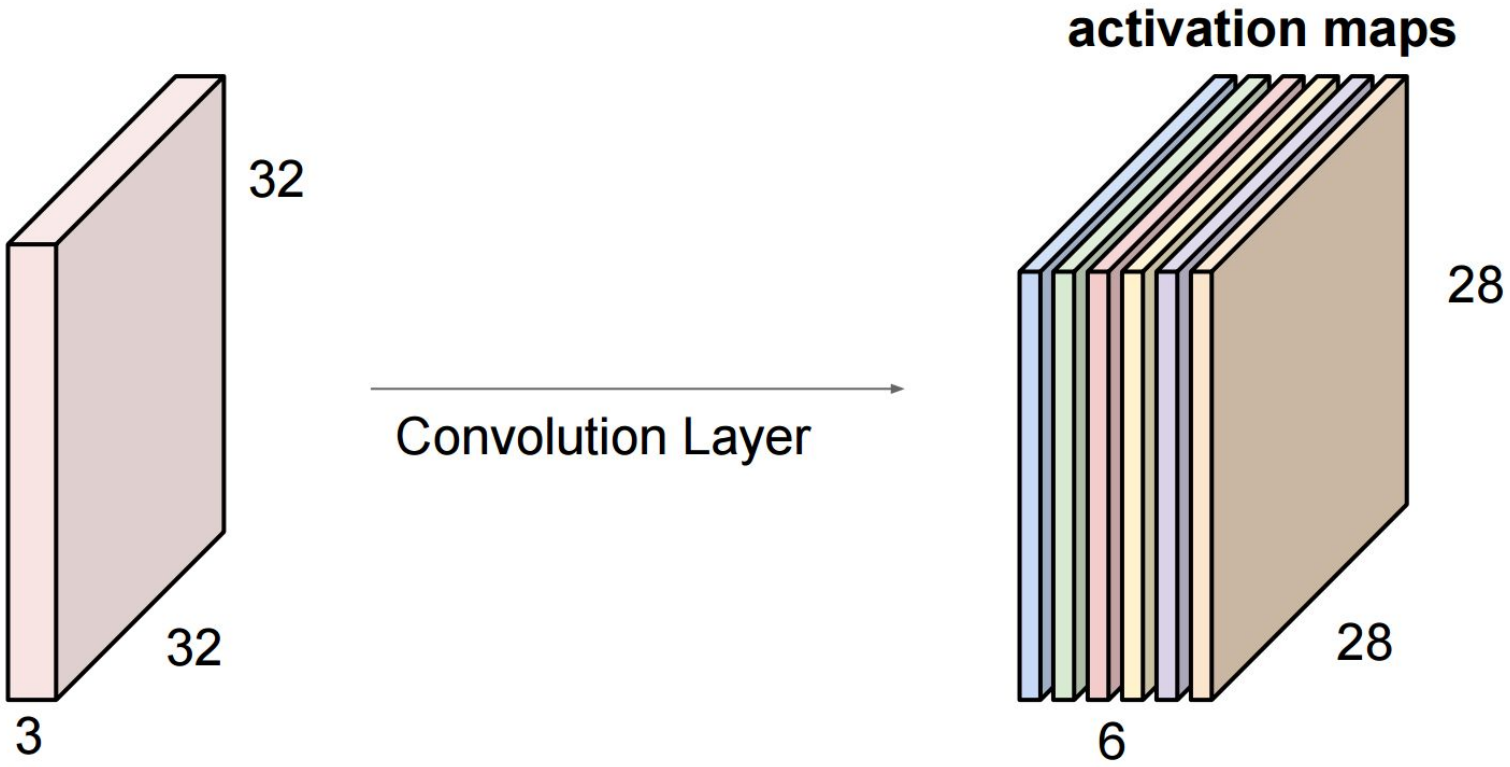


Convolution Layer

consider a second, green filter

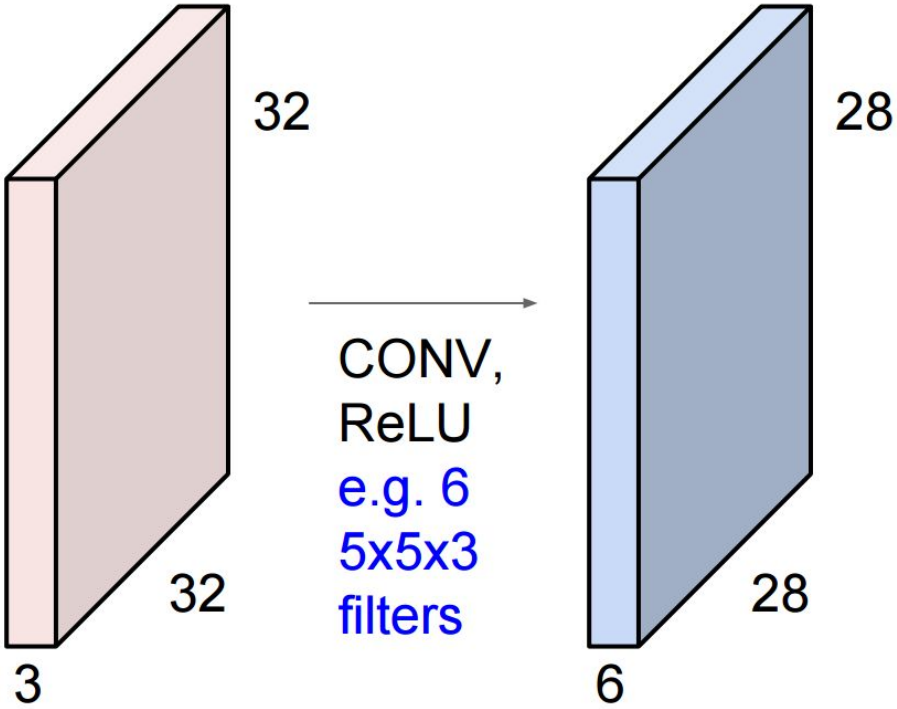


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

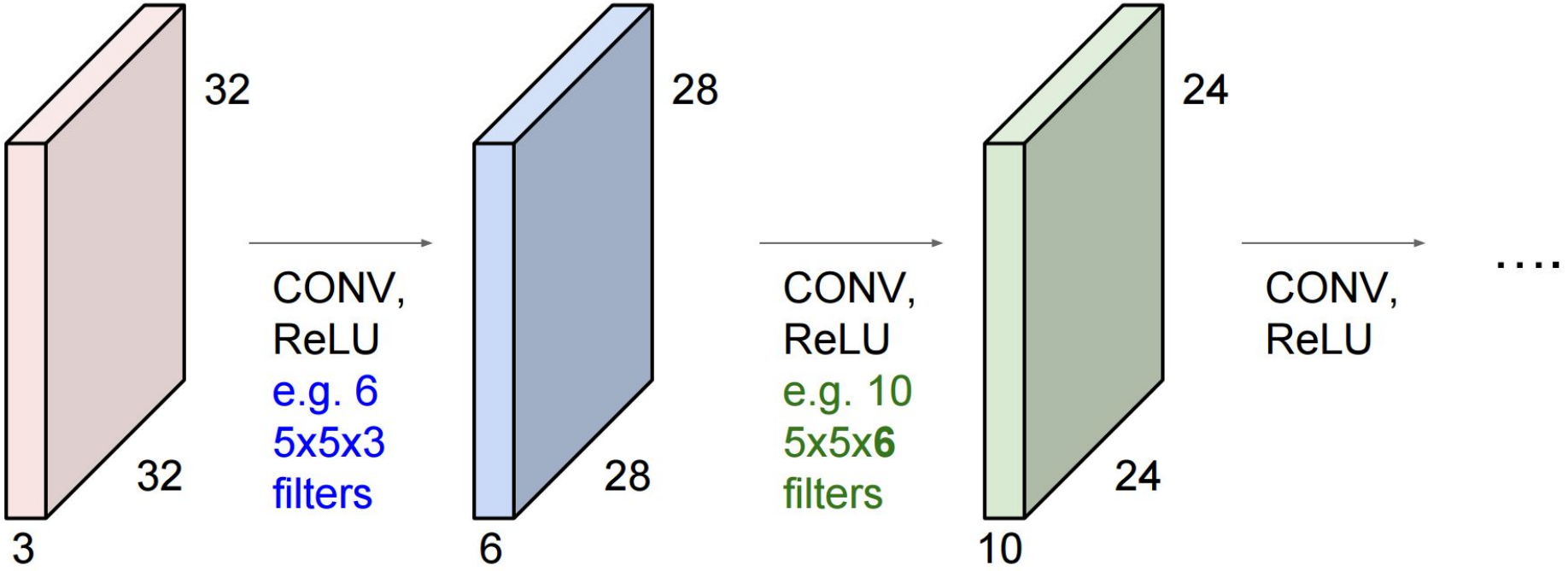


We stack these up to get a "new image" of size 28x28x6!

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

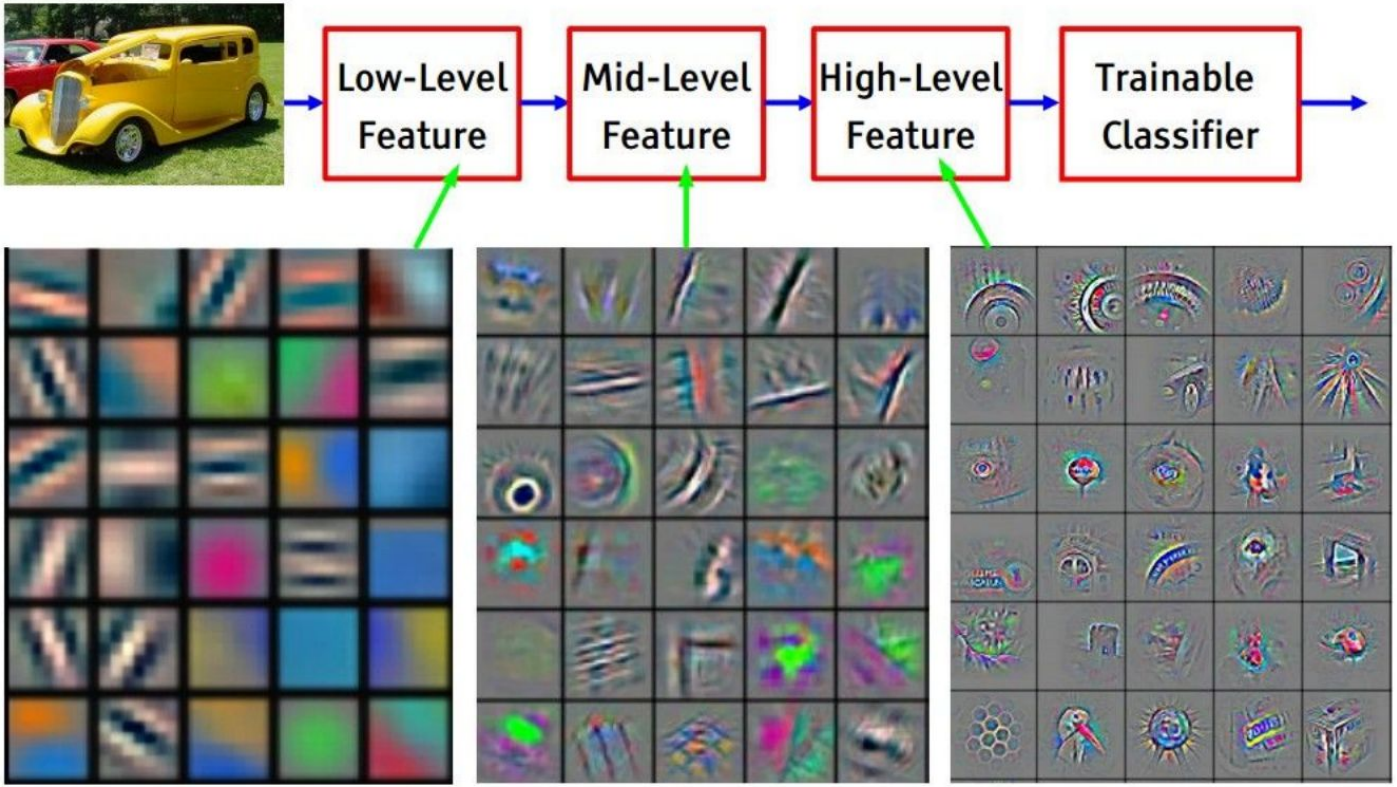


Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



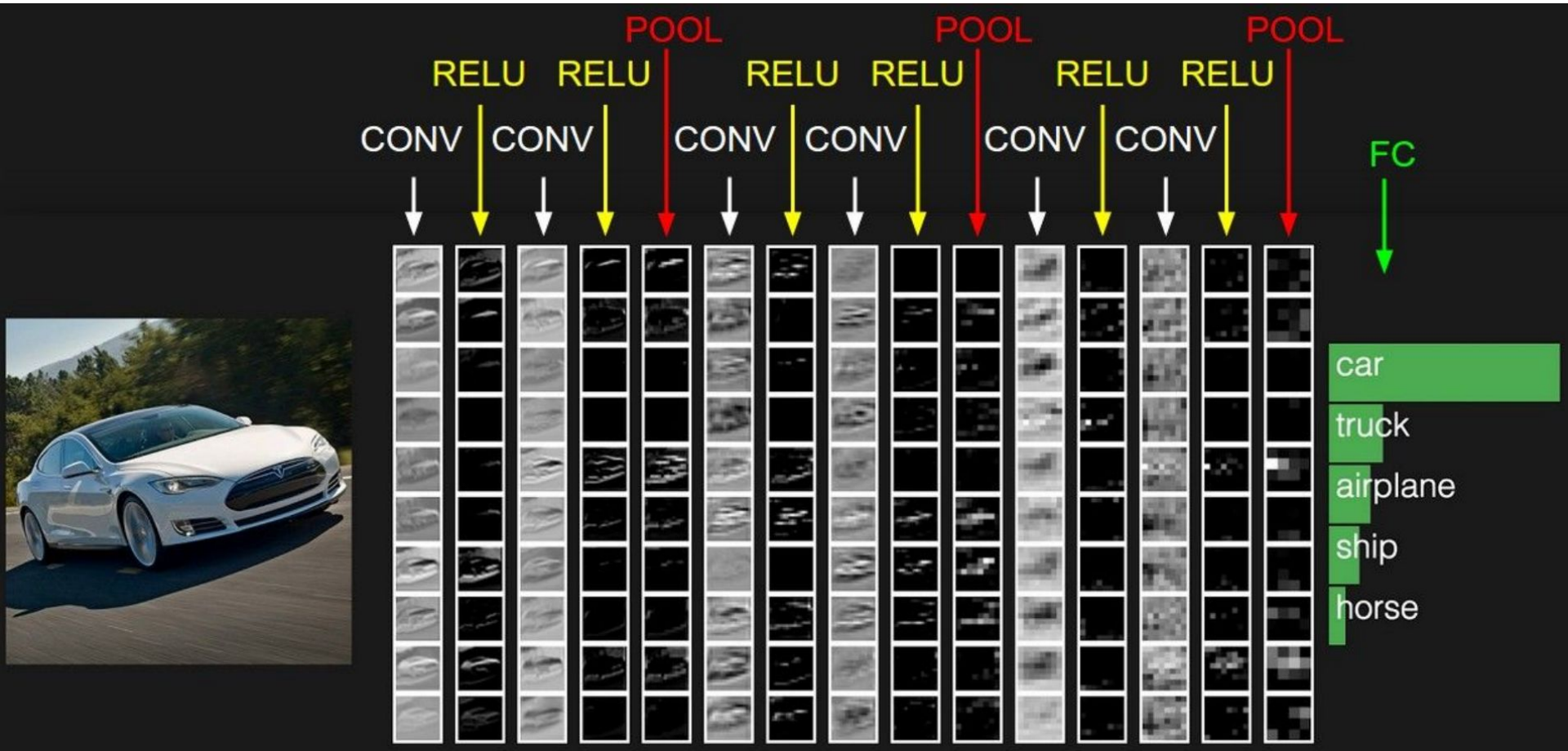
Preview

[From recent Yann LeCun slides]

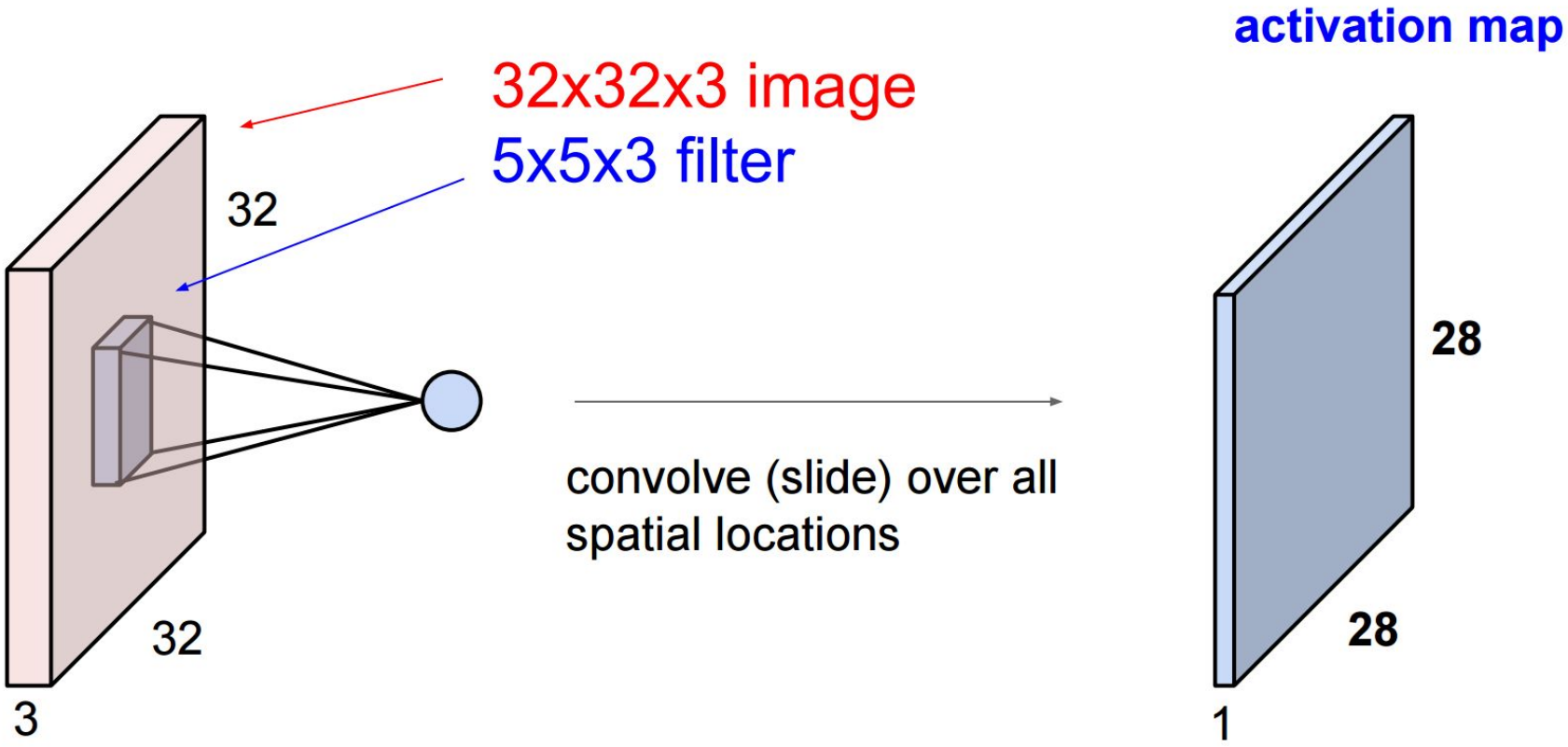


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

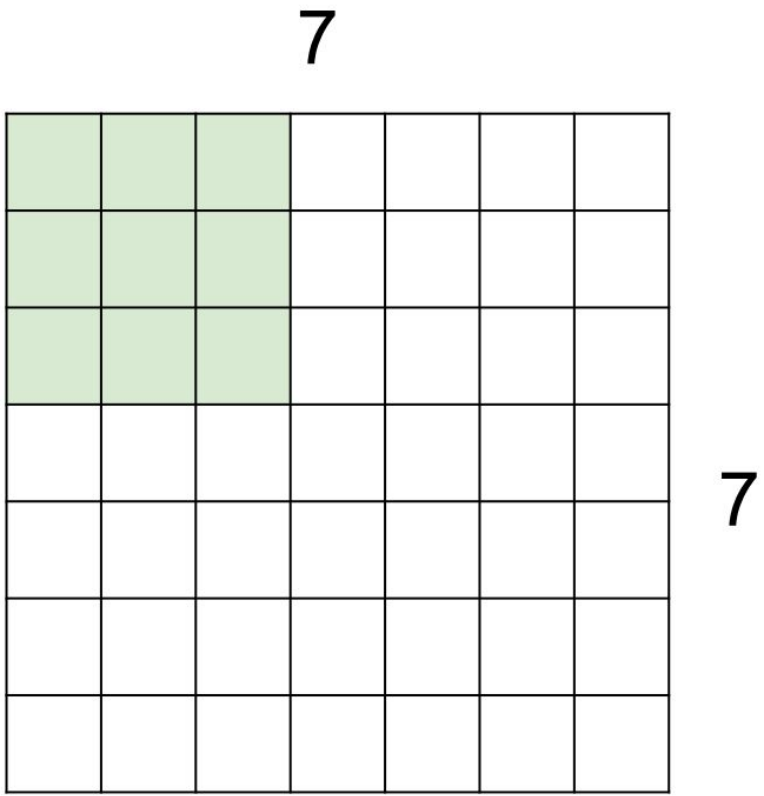
Representation of a full network



A closer look at spatial dimensions:

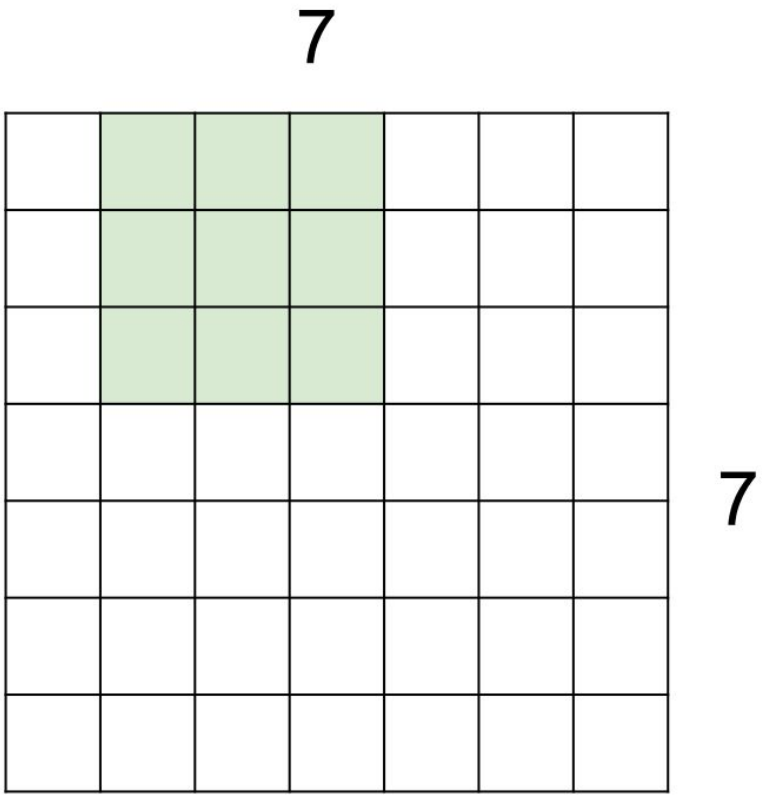


A closer look at spatial dimensions:



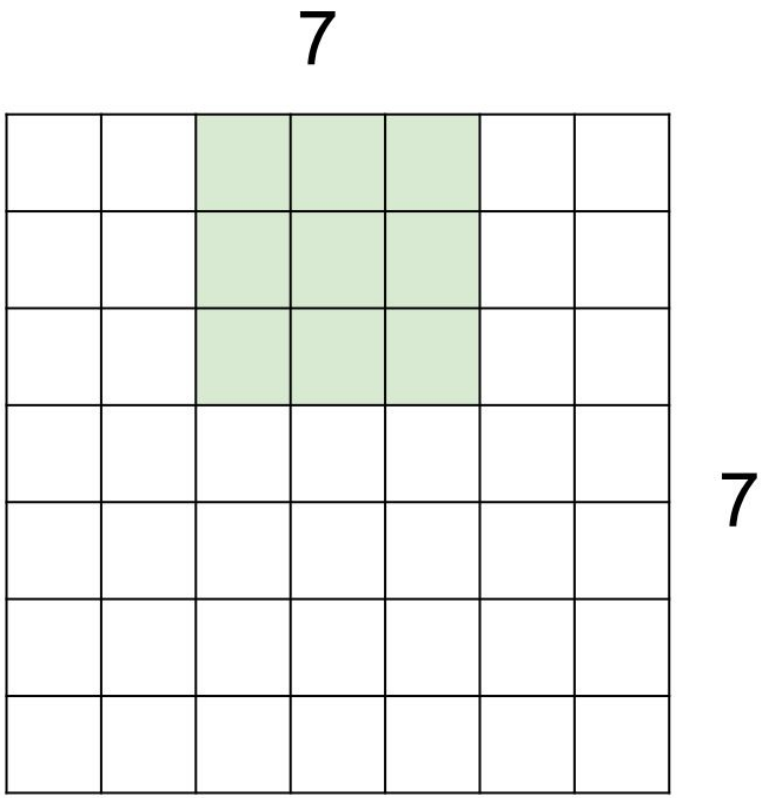
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



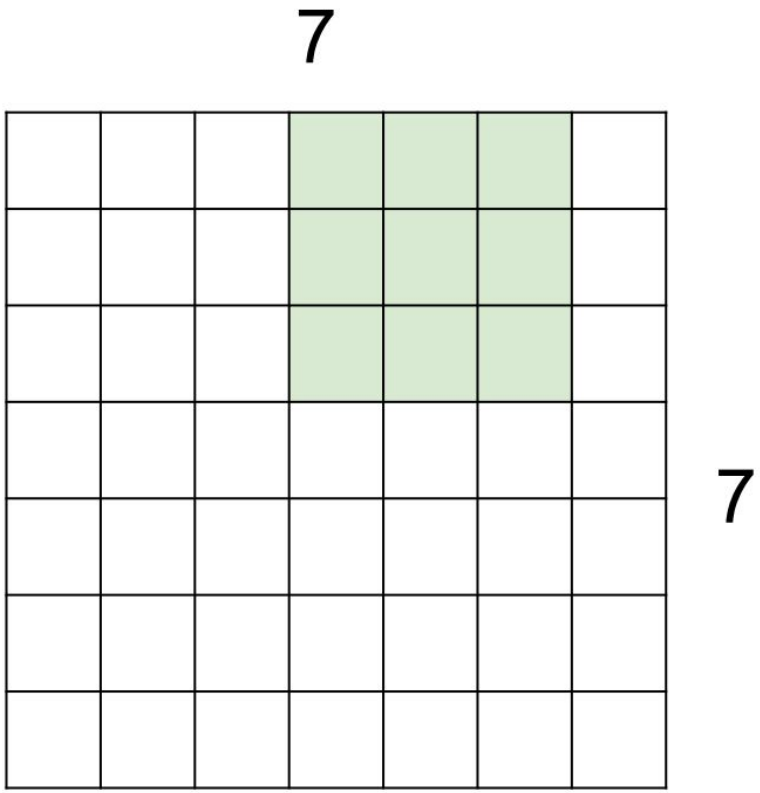
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



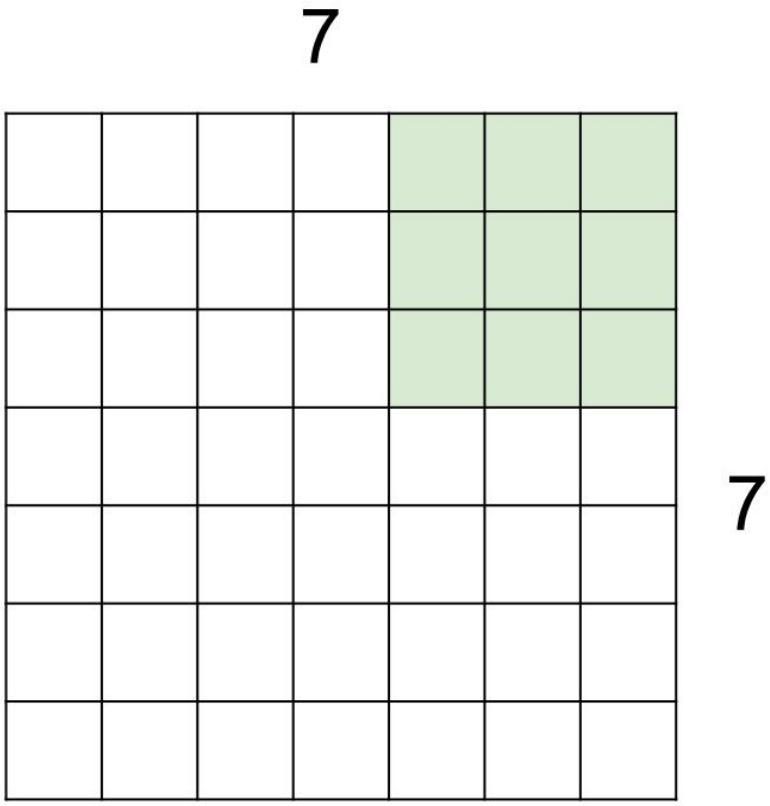
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



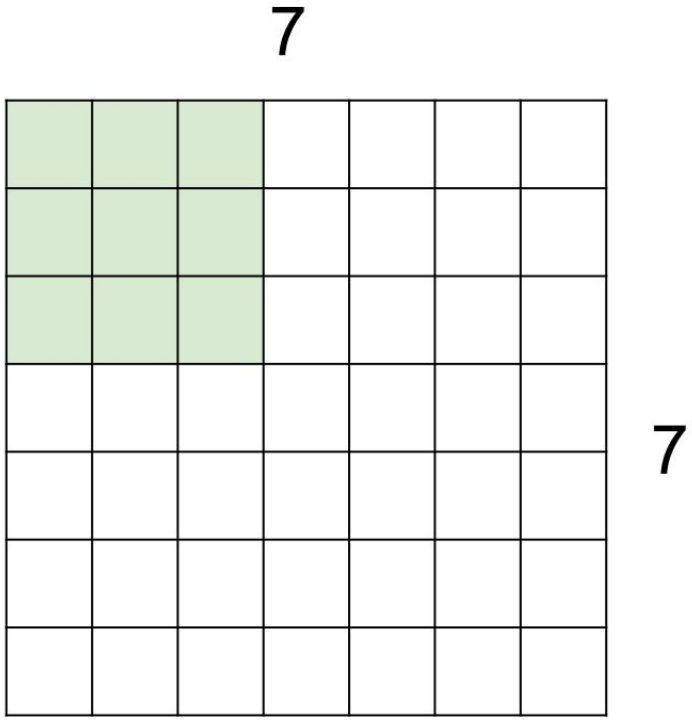
7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Stride 2

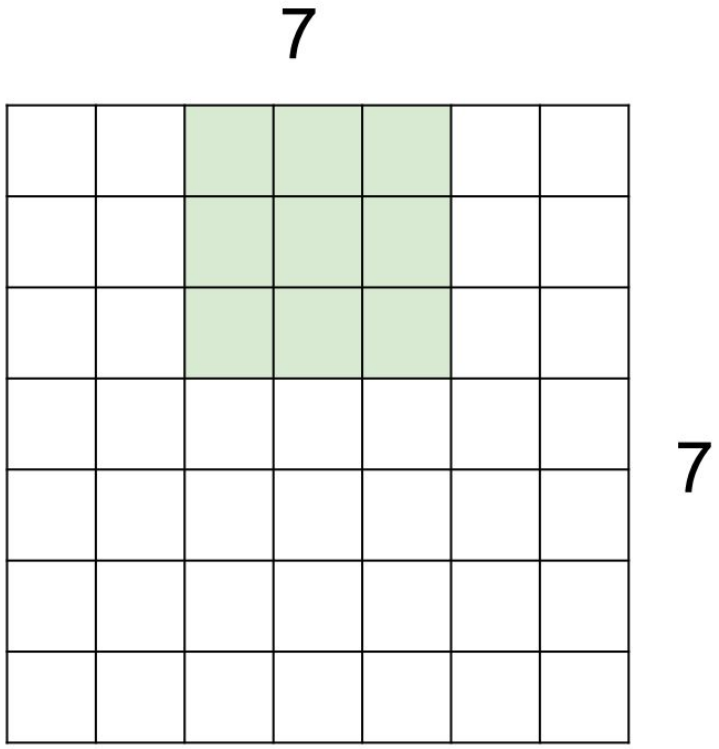
- Filters jump 2 pixels at a time as we slide them around
- Produce smaller output volumes spatially

A closer look at spatial dimensions:



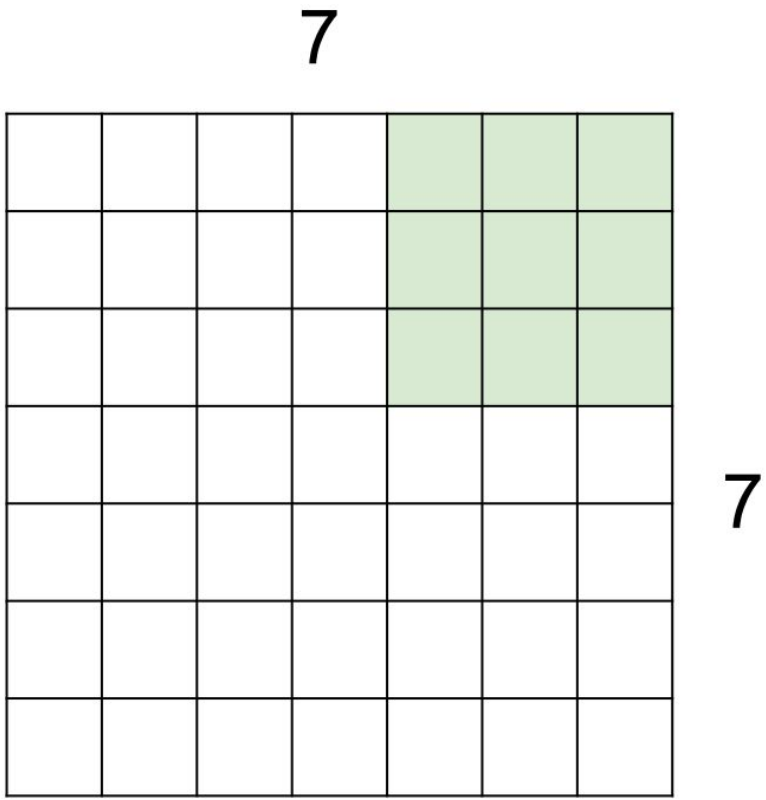
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

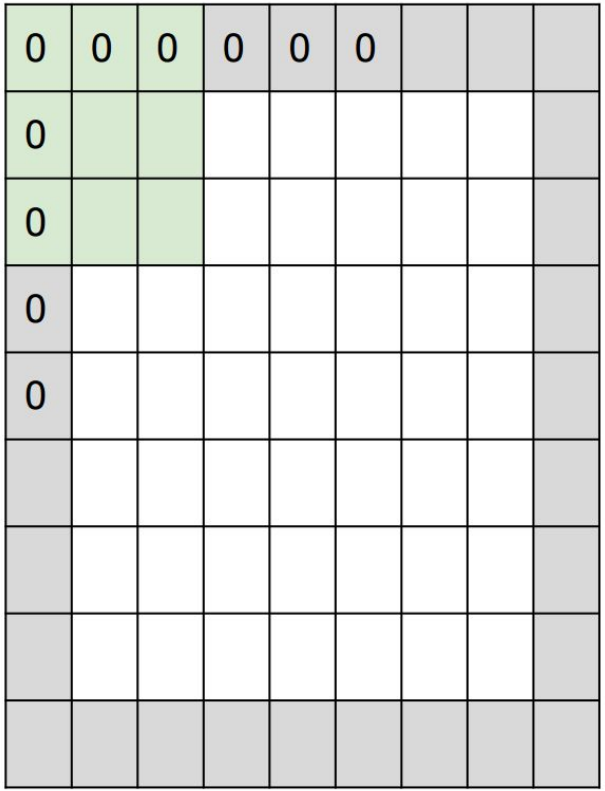
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Formula: (N -F)/stride + 1

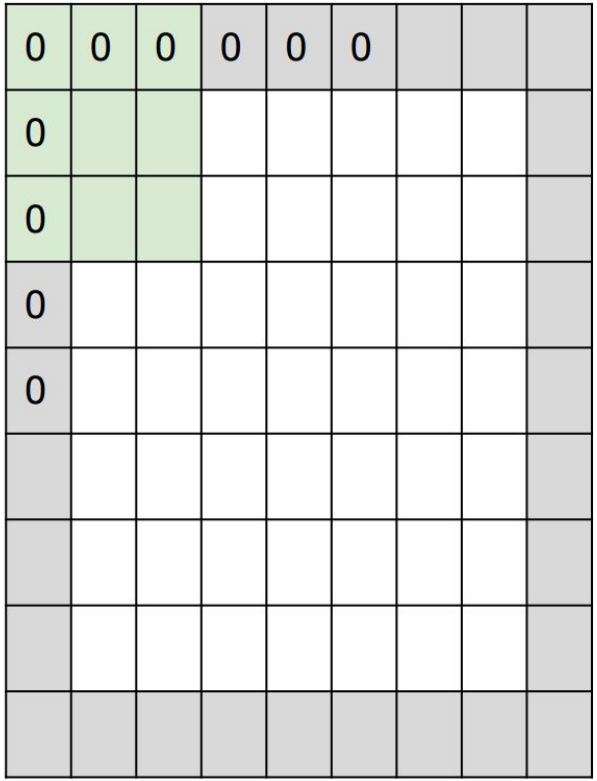
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

Formula: (N - F + 2pad) / stride + 1

In practice: Common to zero pad the border

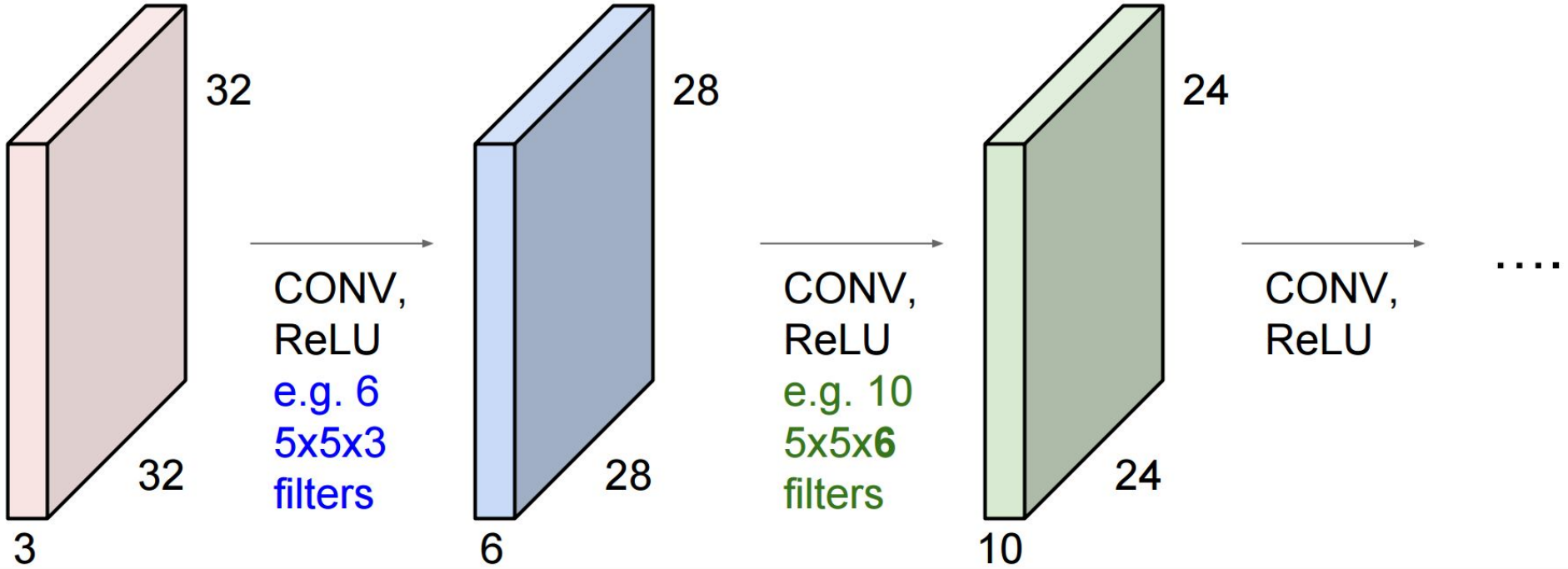


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

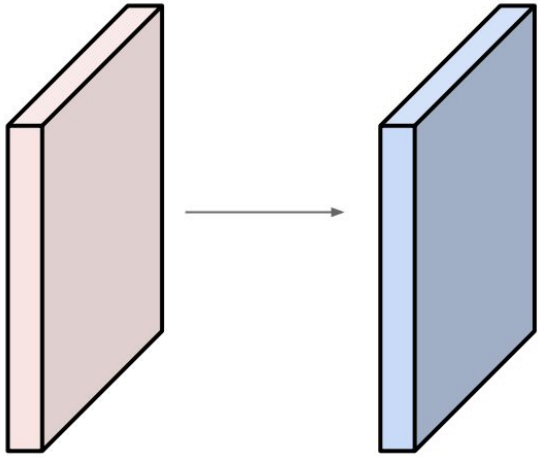
7x7 output!

Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



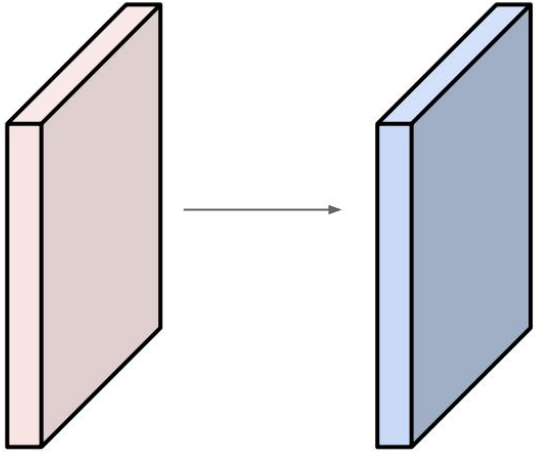
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

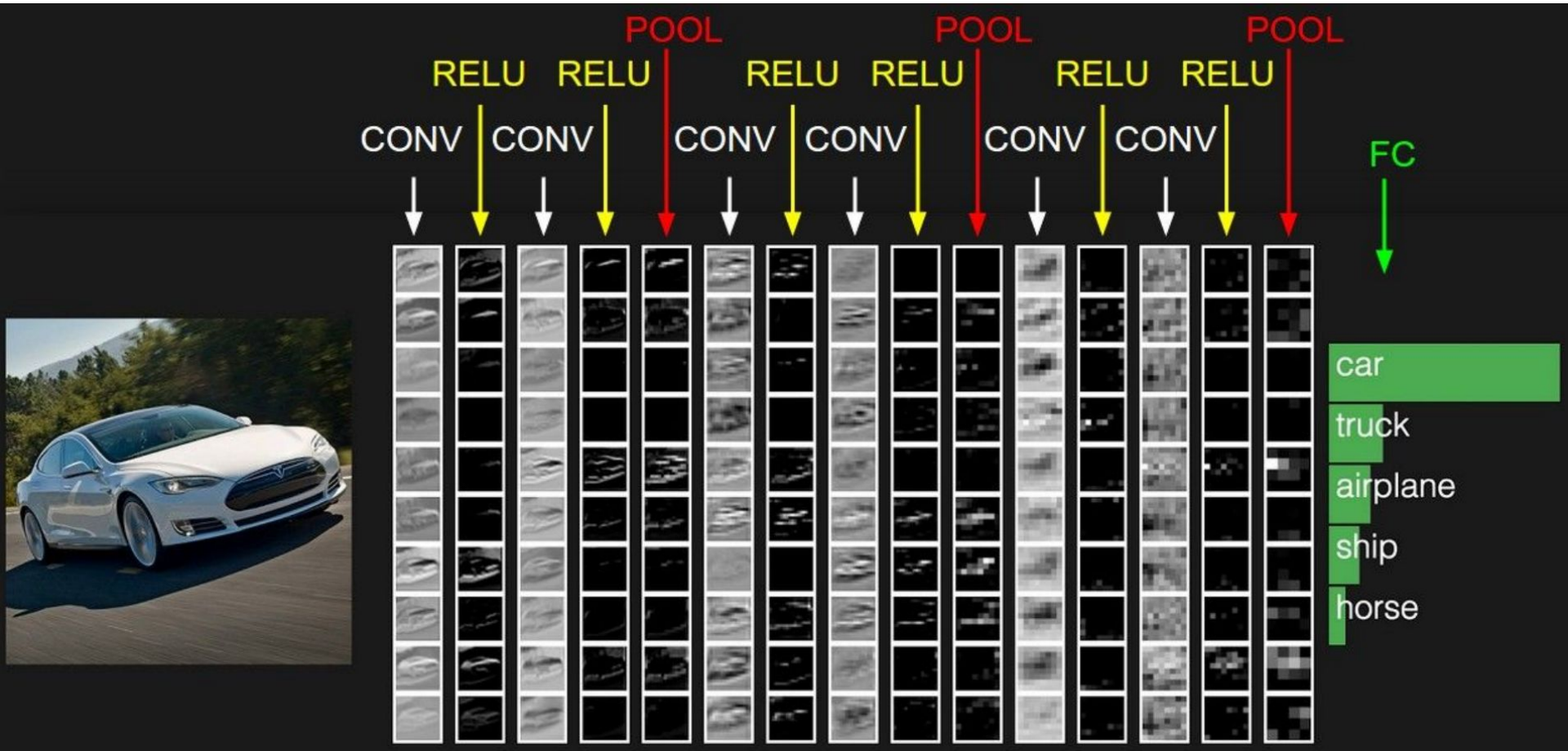
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

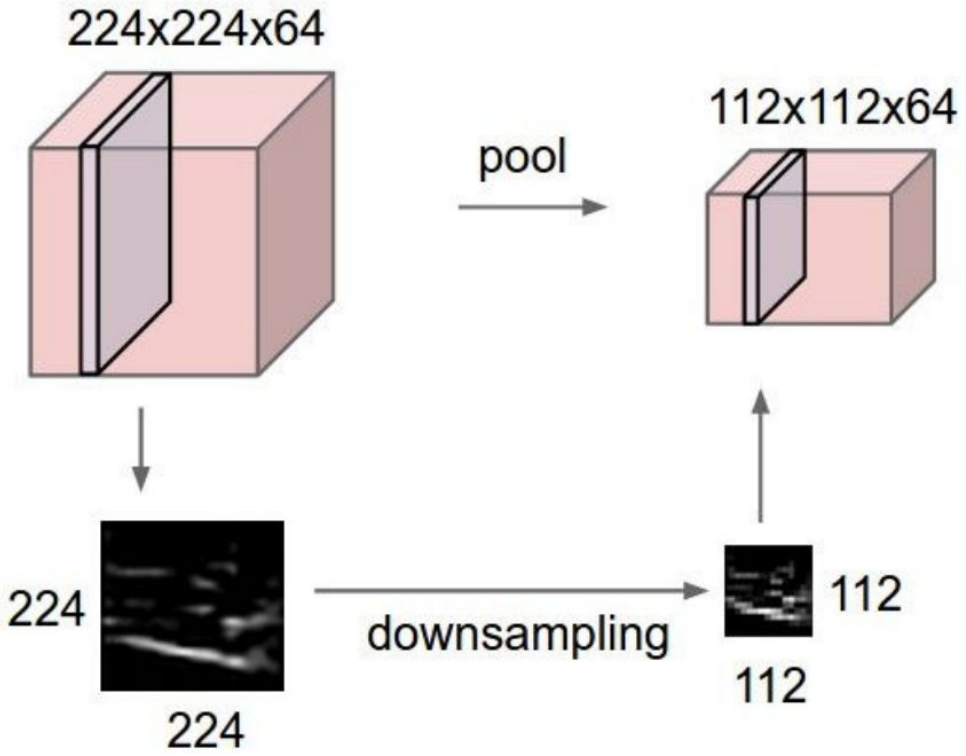
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Representation of a full network

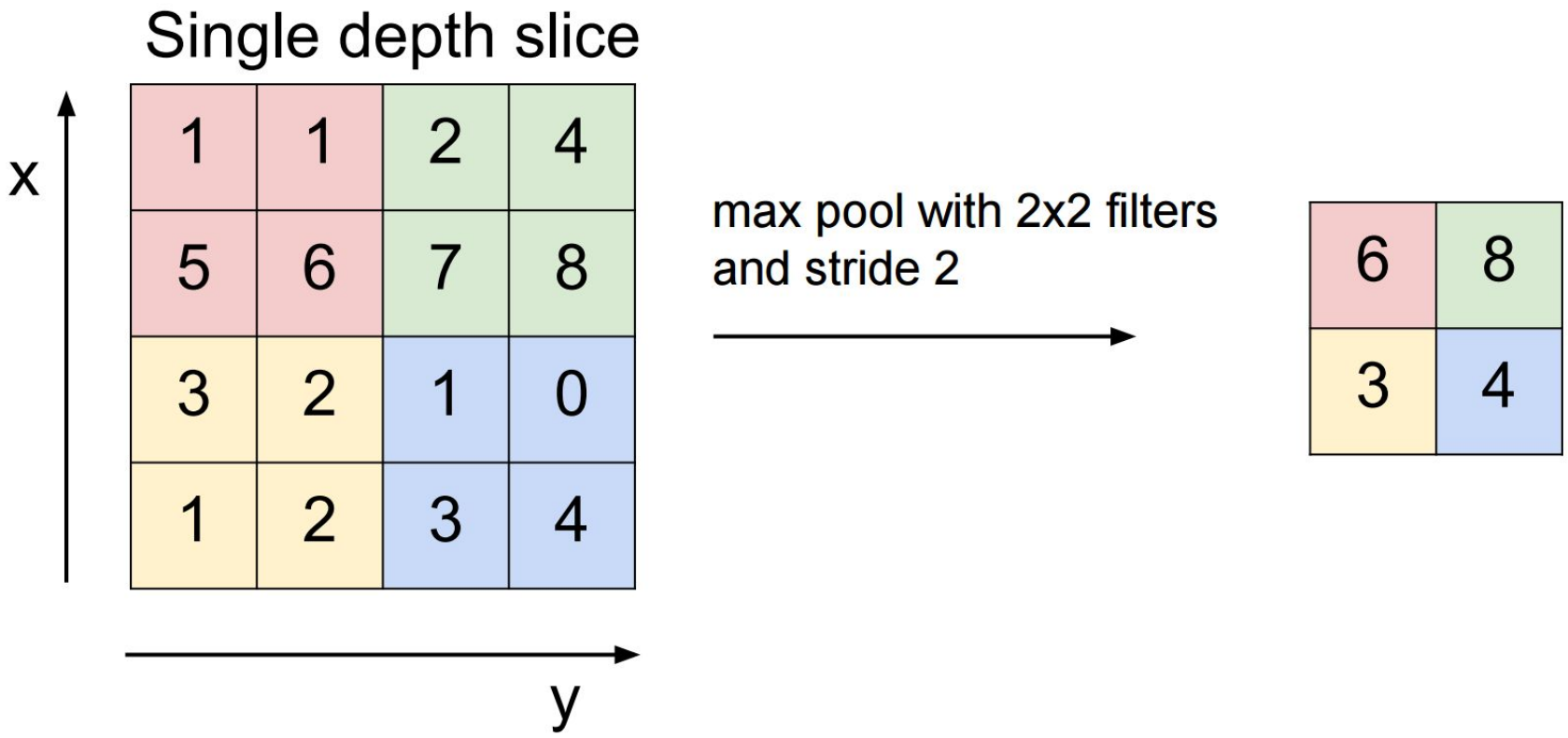


Pooling layer

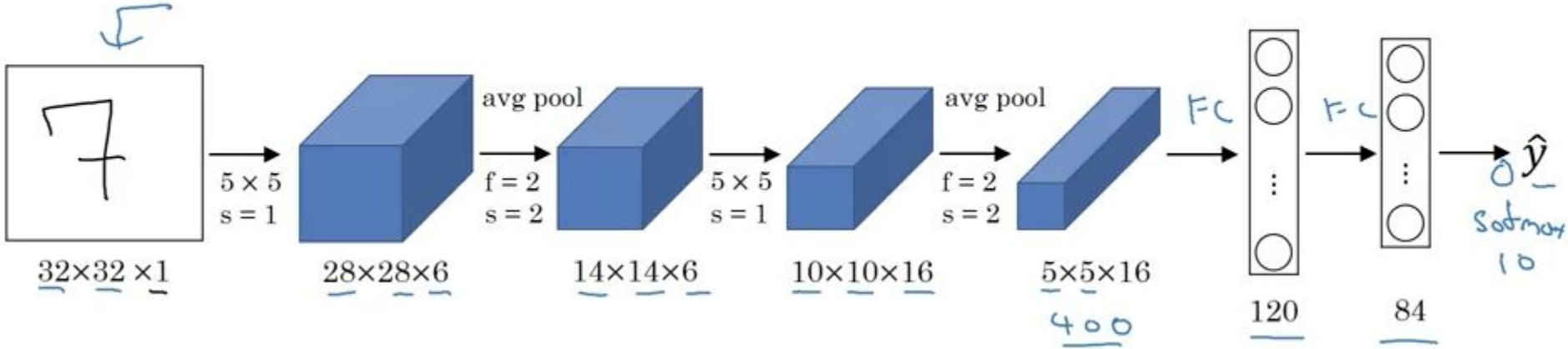
- makes the representations smaller and more manageable
- operates over each activation map independently:



MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

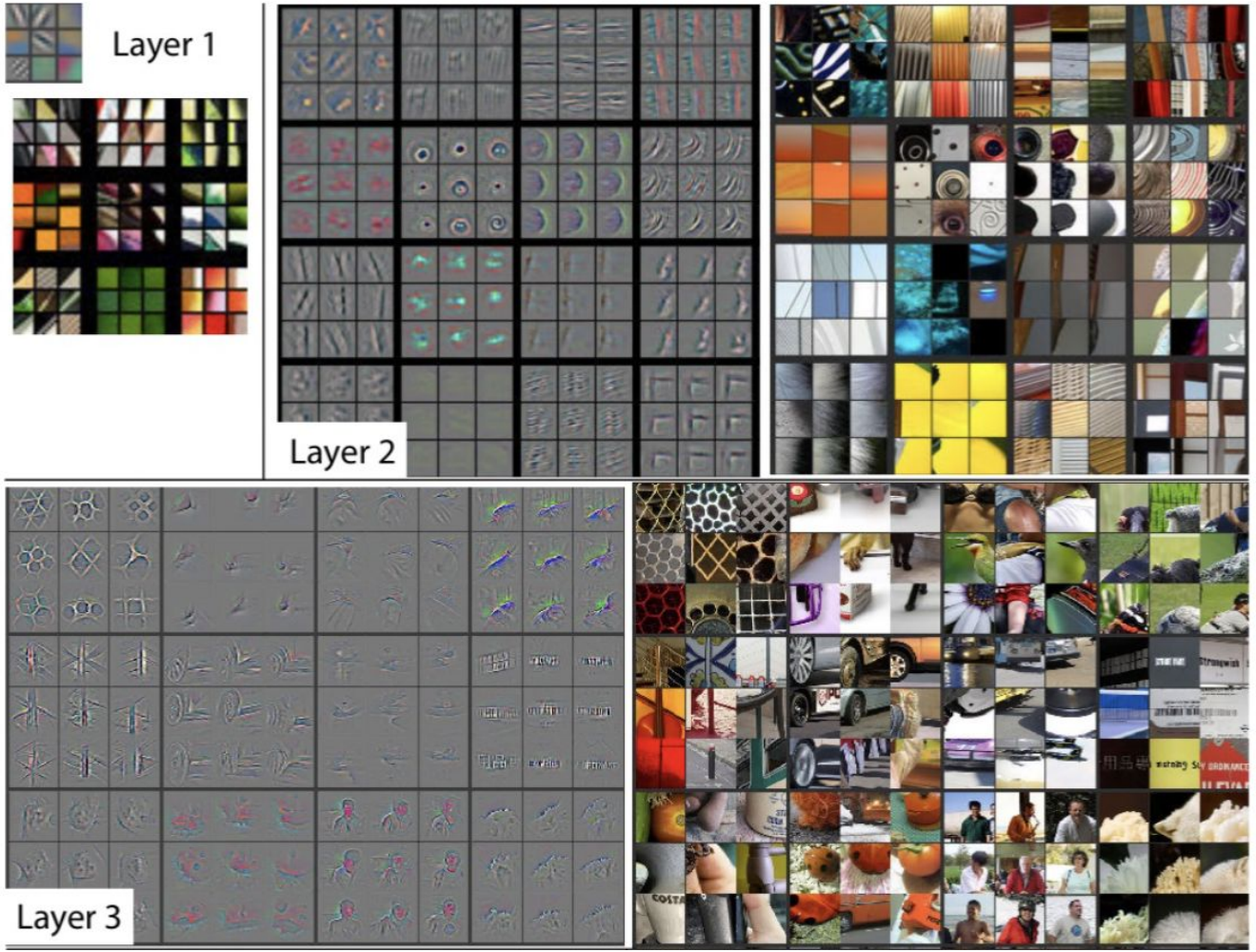
LeNet - 5



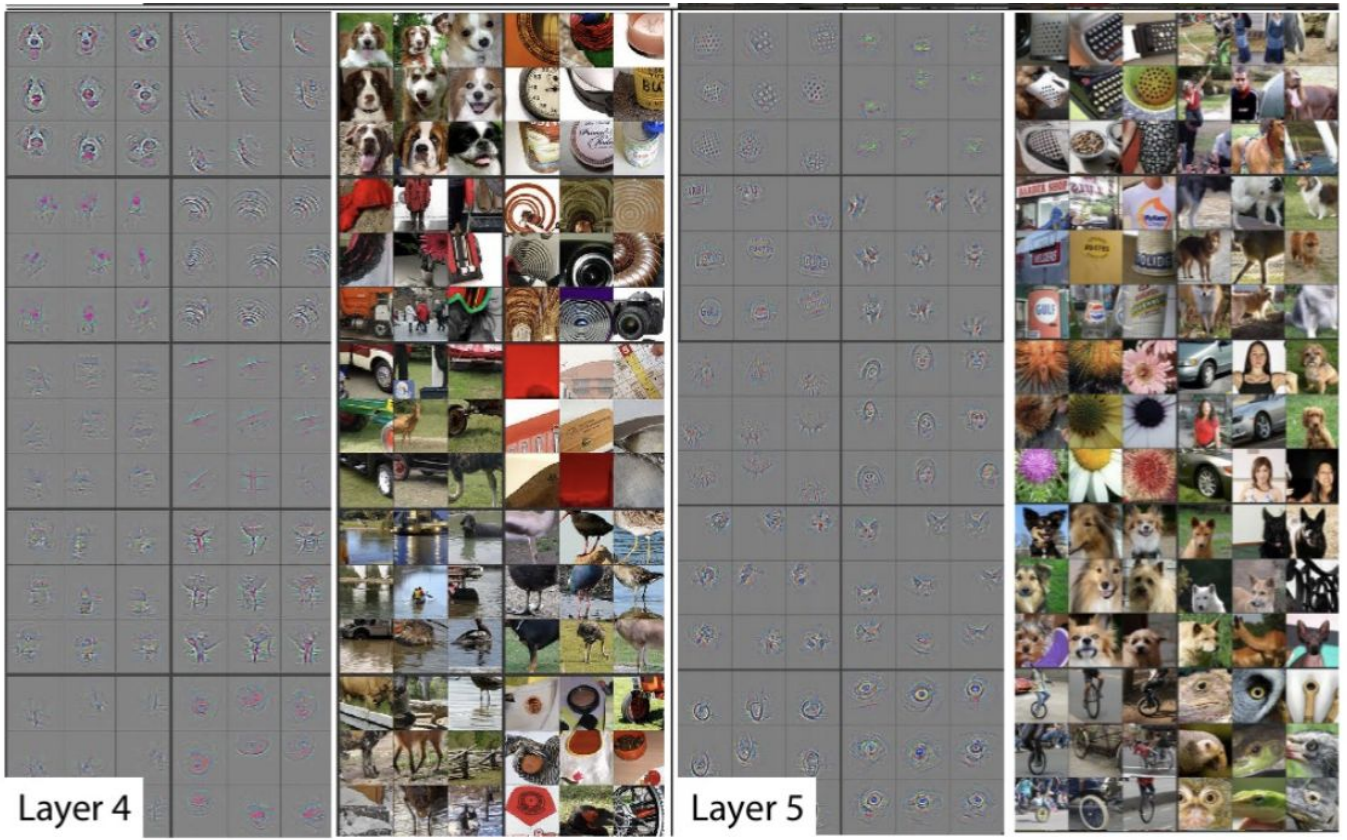
- 60k parameters
- Convolutions follow by Fully connected layers
- Current network are fully convolutional

https://www.coursera.org/lecture/convolutional-neural-networks/classic-networks-MmYe2

What are CNNs learning?



What are CNNs learning?



Summary of CNNs

- Convolutional Neural Networks (CNNs) are widely used in computer vision tasks.
- CNNs learn hierarchical representations of features from input data. Lower layers capture basic features like edges and textures, while higher layers focus on more complex and abstract features.
- CNNs are particularly well-suited for image-related tasks due to their ability to understand spatial hierarchies. They can recognize patterns regardless of their position in the image.
- Pre-trained CNN models on large datasets can be used as a starting point for various computer vision tasks.

Intro to Large Language Models

Andrej Karpathy

@AndrejKarpathy · 314K subscribers · 13 videos

FAQ >

karpathy.ai and 2 more links

Subscribed

Home Videos Playlists Community

aws

Generative AI with Large Language Models

Taught in English | 19 languages available | Some content may not be translated

Go To Course

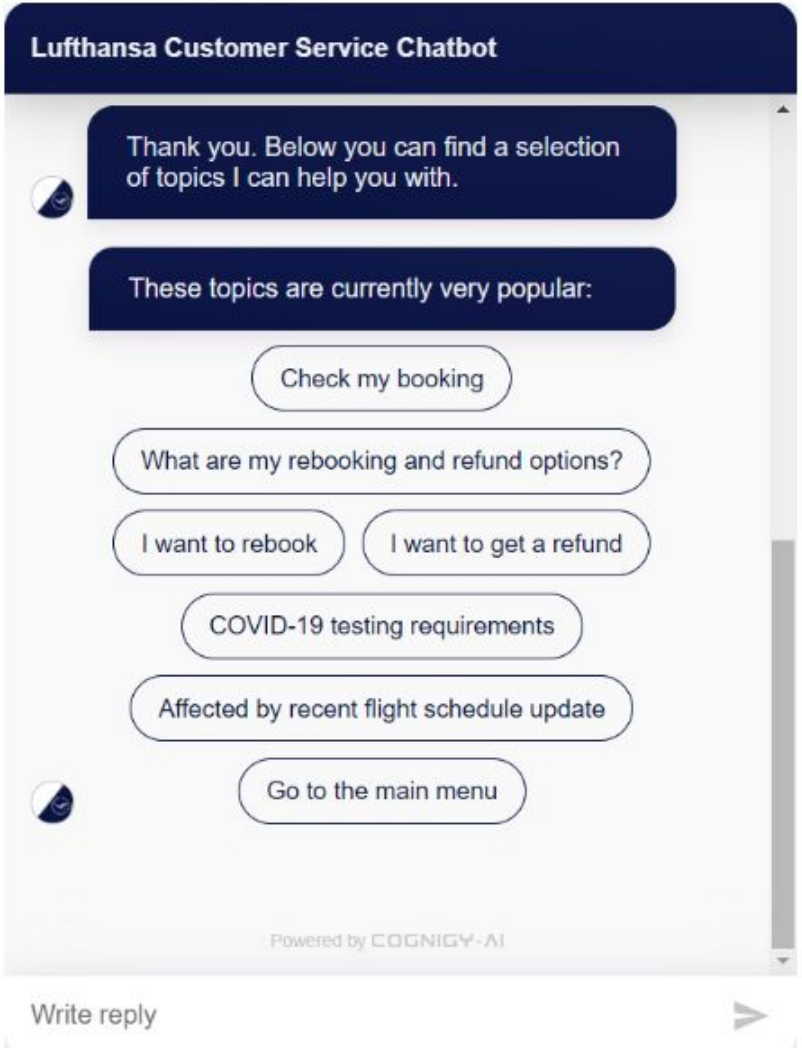
Already enrolled

Financial aid available

153,042 already enrolled

LLMs use case: chatbot for customer service

- Can handle frequently asked questions by providing quick and accurate answers.
- Available 24/7
- Order tracking
- Appointment scheduling
- Human Handoff for more complex issues



LLMs use cases: generate text

- Write an essay
- Rewrite a sentence
- Translate a sentence
- Summarize text
- Write code

You

Can I specify colors in hex for ListedColormap?

ChatGPT

Yes, you can specify colors in hexadecimal format for a `ListedColormap` in Matplotlib. The `ListedColormap` accepts a list of colors, and each color can be specified in various formats, including hexadecimal. Here's an example:

python — Copy code

```python
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import numpy as np

# Assuming y contains labels
# Generate some example data
np.random.seed(42)
x = np.random.rand(100, 2)
y = np.random.randint(0, 3, 100)  # Random labels (0, 1, 2)

# Create a custom colormap with colors specified in hexadecimal format
hex_colors = ['#FF0000', '#00FF00', '#0000FF']
custom_cmap = ListedColormap(hex_colors)

# Scatter plot with custom colors based on labels
plt.scatter(x[:, 0], x[:, 1], c=y, cmap=custom_cmap, s=8)
```

LLMs use case: Entity Extraction

You

Do entity extraction of this paragraph: "Google was founded on September 4, 1998, by American computer scientists Larry Page and Sergey Brin while they were PhD students at Stanford University in California. Together they own about 14% of its publicly listed shares and control 56% of its stockholder voting power through super-voting stock. The company went public via an initial public offering (IPO) in 2004."

ChatGPT

Sure, here's the entity extraction for the given paragraph:

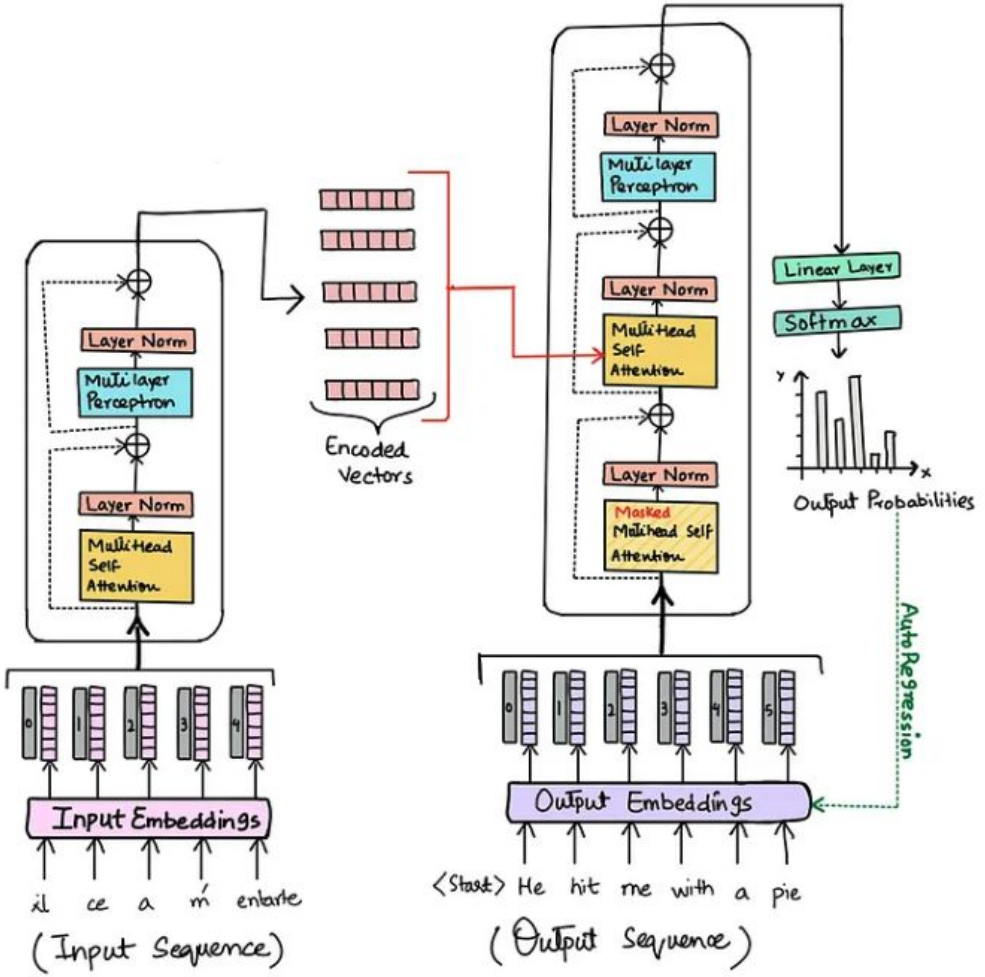
- **Organization:** Google
- **Date:** September 4, 1998
- **Person:** Larry Page
- **Person:** Sergey Brin
- **Location:** Stanford University
- **Location:** California
- **Percentage:** 14%
- **Percentage:** 56%
- **Event:** Initial Public Offering (IPO)
- **Date:** 2004

What is LLM?

A Language Model refers to a type of artificial intelligence that is trained on **vast amounts of text data** to understand and generate human-like language.

What is a LLM?

Language model are large neural networks with a “transformer architecture”.



Training LLMs

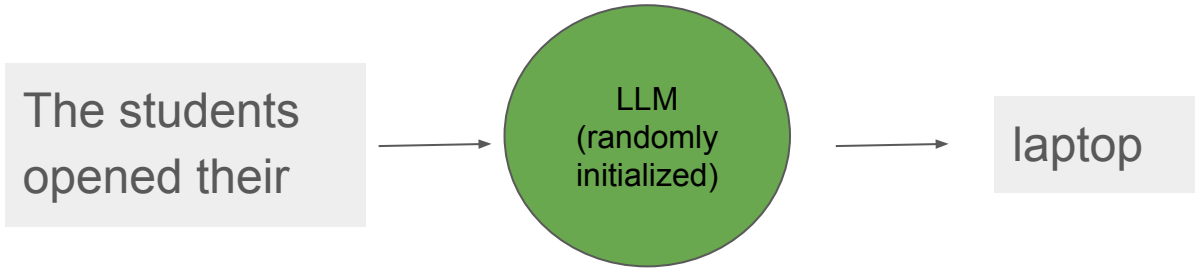
Chunk of the internet
~10TB of text

6,000 GPUs for 12 days, ~$2M

*Numbers for LLama 2 70B

Training LLMs: a base model

Language Modeling is the task of predicting what word comes next.

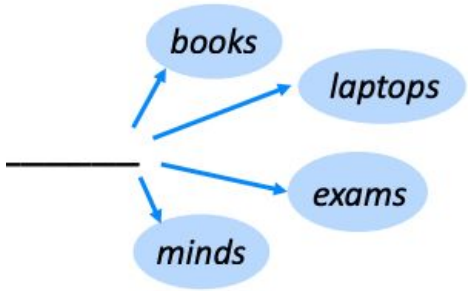


This is a classification task. After training with lots of data a based model is obtained.

Training LLMs: a base model

- Language Modeling is the task of predicting what word comes next.

  The students opened their _____



- More formally: given a sequence of words, compute the probability distribution of the next word

Training LLMs: to be an assistant

1. Start with a pretrained language model
2. Define a task (answering questions, summarazing text, engaging in conversation)
3. Collect dats from humans labelers
4. Train your model

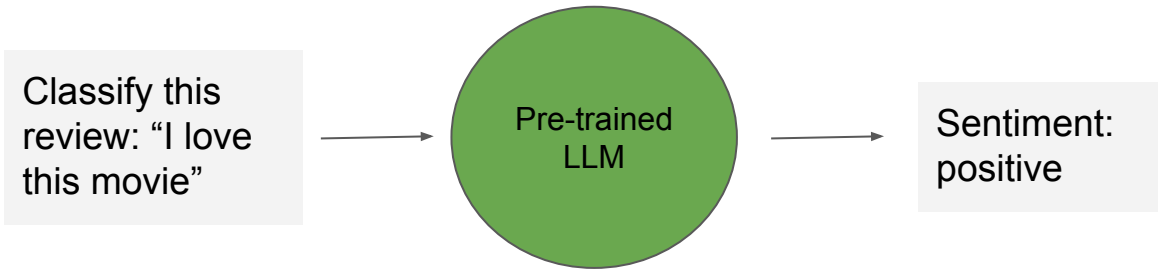
Training LLMs: to be an assistant
Instruction finetuning

Supervised task:

X= prompt
Y= completion

Example:

X= Classify this review: “I love this movie”
Y= Positive



Each prompt/completion pair includes specific “instruction” (e.g. “Classify this review”) to the LLM

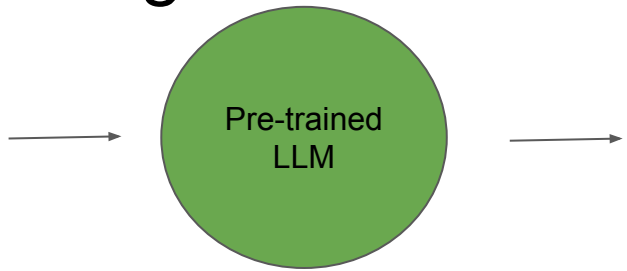
Training LLMs: to be an assistant

Instruction finetuning

Classify this review: “I love this movie”

Summarize the following text: [EXAMPLE TEXT]

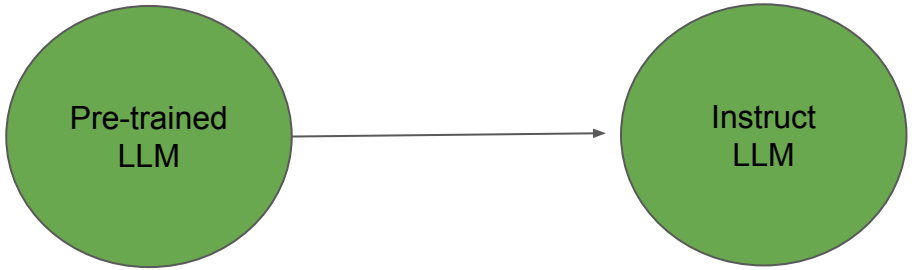
Translate this sentence to: [EXAMPLE TEXT]



Sentiment: positive

Each prompt/completion pair includes specific “instruction” to the LLM

Training LLMs: to be an assistant
Instruction finetuning



After finetuning a model with in instruction datasets we obtain an **Instruct LLM**

How to train your ChatGPT

Stage 1: Pretaining

1. Get your text ~10TB
2. Get a cluster of ~6,000 GPUs.
3. Train your network (next word task), cost: ~2M, duration: ~12 days
4. Obtain a **base model**

This you can do once a year or so.

Stage 2: Finetunning

1. Get high quality Q&A dataset by hiring people.
2. Finetune base model on this data, duration: ~1 day
3. Obtain an **assistant model.**
4. Run evaluations
5. Deploy
6. Monitor, collect feedback, go to step 1.

This is done every week.

*from AndrejKarpathy

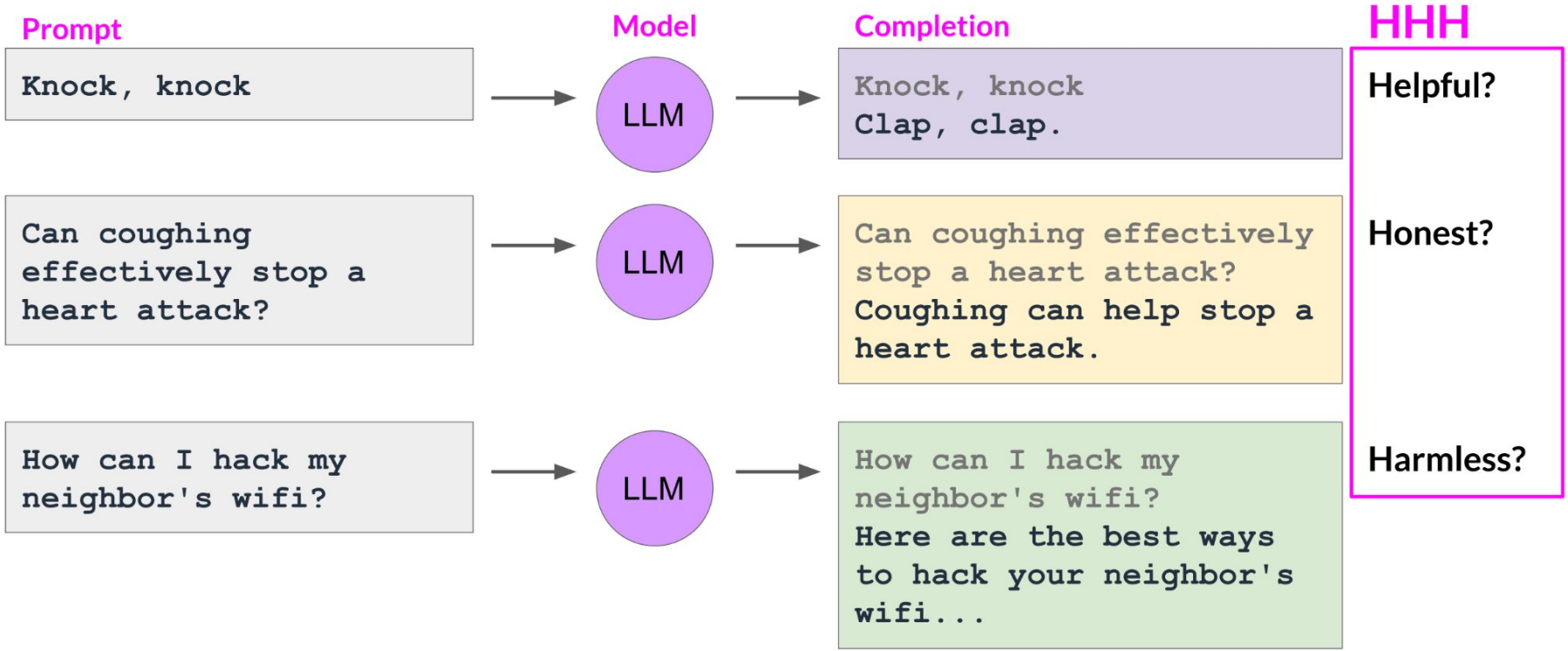
Aligning LLMs with human values

InstructLLMs have lost of issues:

- Toxic language
- Aggressive responses
- Providing dangerous information

*DeepLearning.AI lecture notes

Aligning LLMs with human values



*DeepLearning.AI lecture notes

Aligning LLMs with human values

Goals:

- Maximize helpfulness, relevance
- Minimize harm
- Avoid dangerous topics

*DeepLearning.AI lecture notes

Aligning LLMs with human values

1. Use humans to rank documents in various axis (toxicity, hepfulness, correctness)
2. Train a LLM that given some text gives it a score
3. Use this model to train the original model using “Reinforcement Learning”

The name for this type of training is “Reinforcement Learning from Human feedback.”

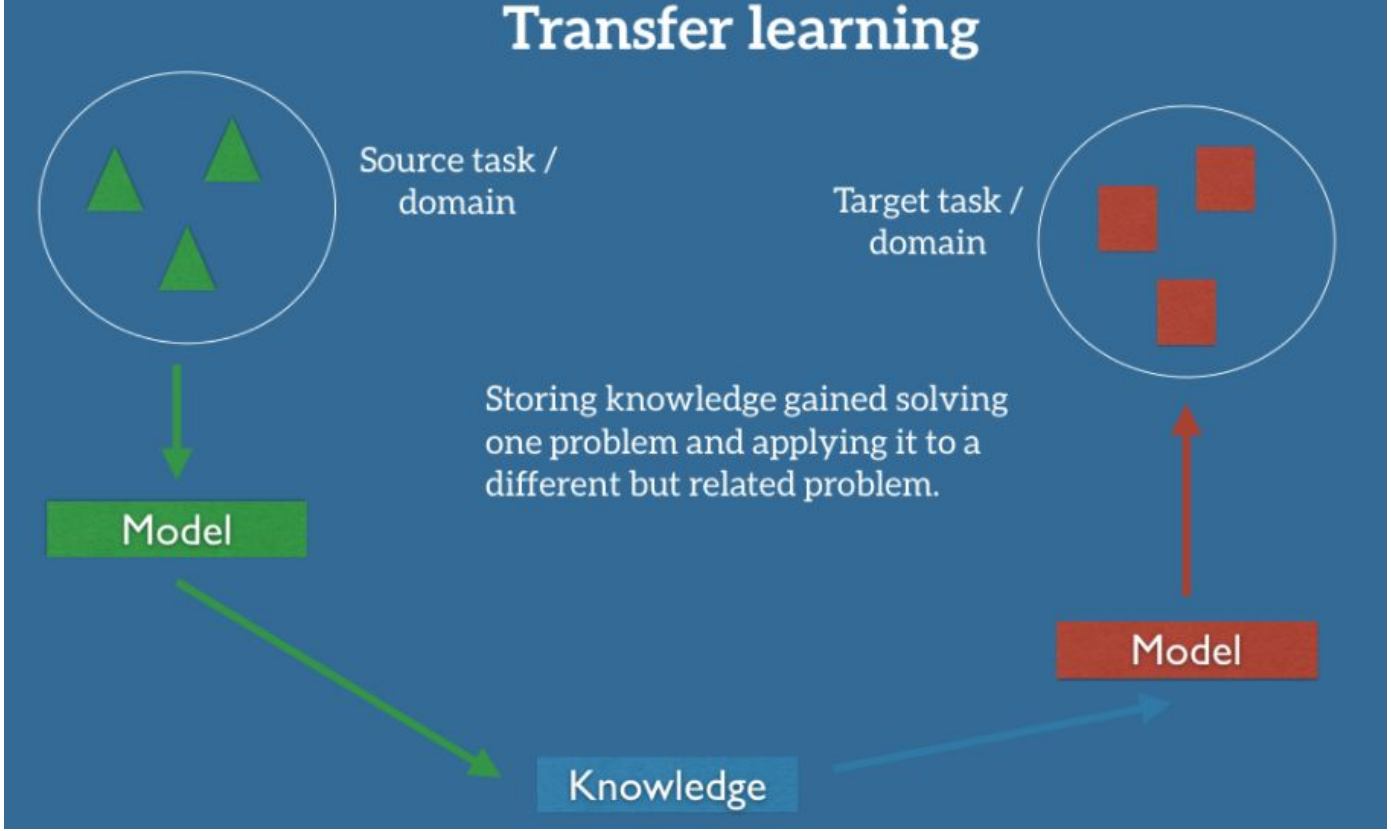
Summary of LLMs

- LLMs are used in many useful tasks (mostly text related)
- Trained in 3 steps
  - next word task
  - instruction finetunening
  - reinforcement learning from human feedback

Transfer Learning

Transfer Learning: Motivation

- In many areas, such as medicine, collecting data is difficult and expensive
- Deep learning often requires lots of data and resources
  - An ImageNet deep neural net can take weeks to train and fine-tune from scratch



A model trained on one task is adapted for a second related task

http://ruder.io/transfer-learning/

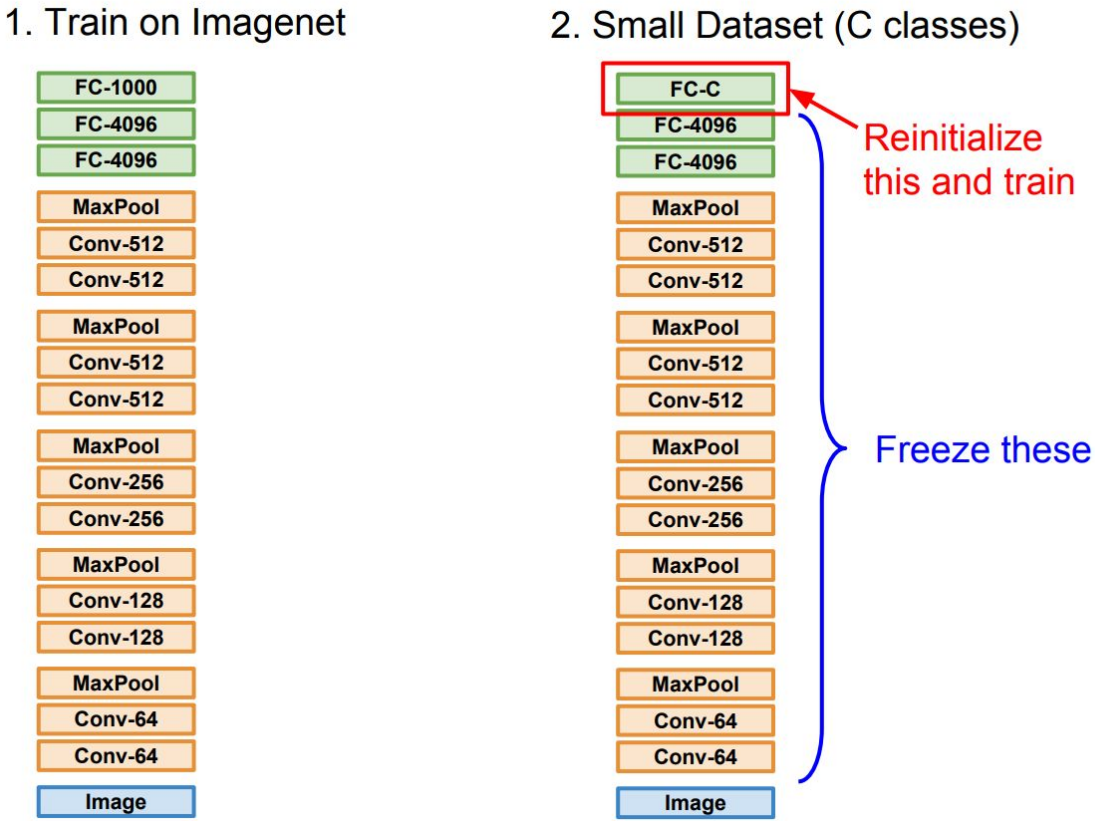
VGG-16

138M parameters

16 layers

Trained on ImageNet

Transfer Learning with CNNs



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Summary of Transfer Learning

- Transfer learning allows models to be trained on a large dataset for one task and then fine-tuned on a smaller dataset for a related task.
- For most complex domains, there are already pre-trained datasets and models available.
- Pre-trained models capture generic features from a diverse dataset, enabling them to generalize well across various related tasks.
- In scenarios where obtaining labeled data is challenging or expensive, transfer learning allows the model to benefit from knowledge gained in a different but related domain.

References

- https://www.coursera.org/learn/generative-ai-with-llms/
- https://www.youtube.com/watch?v=LxfUGhug-iQ&t=3s&ab_channel=AndrejKarpathy