Deep Learning for Data Scientists

Antonio Rueda-Toicen

Data Science Retreat

August 2023

About me

- Senior Data Scientist at Vinted
- Background in academia (computer science and bioengineering)
  - Organizer of the Berlin Computer Vision Group
  - Arepa-lover (try them, they're *awesome*)

Fig 1: Arepas

Meetup

Start a new group | Explore | Messages

Berlin Computer Vision Group

Berlin, Germany

384 members · Public group

Organized by **Antonio Rueda Toicen**

Share:

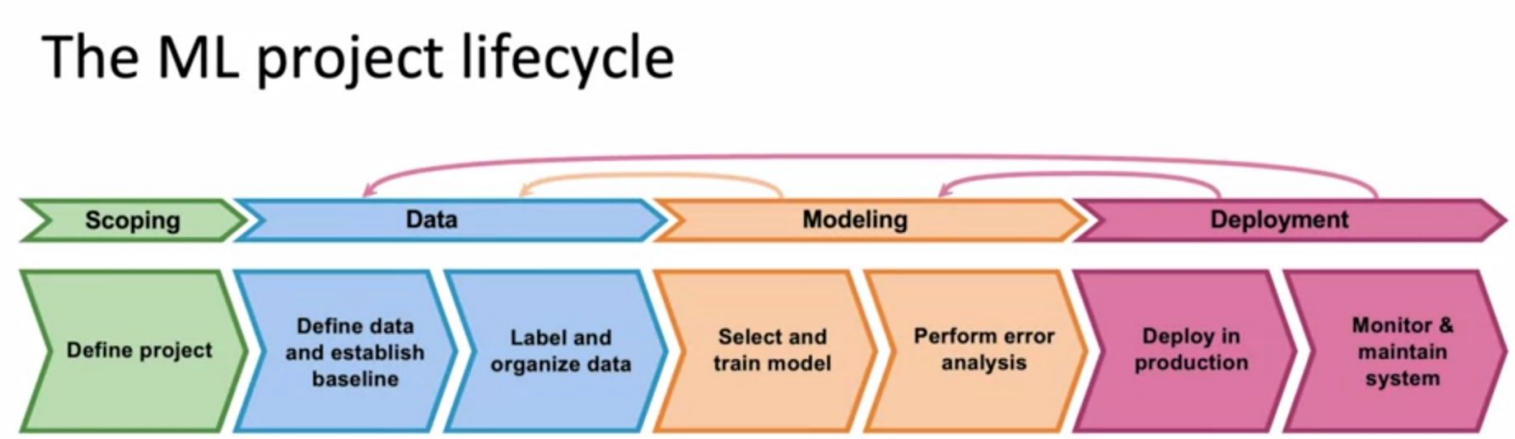
About | Events | Members | Photos | Discussions | More

Manage group | Create event

https://www.meetup.com/Berlin-Computer-Vision-Group/

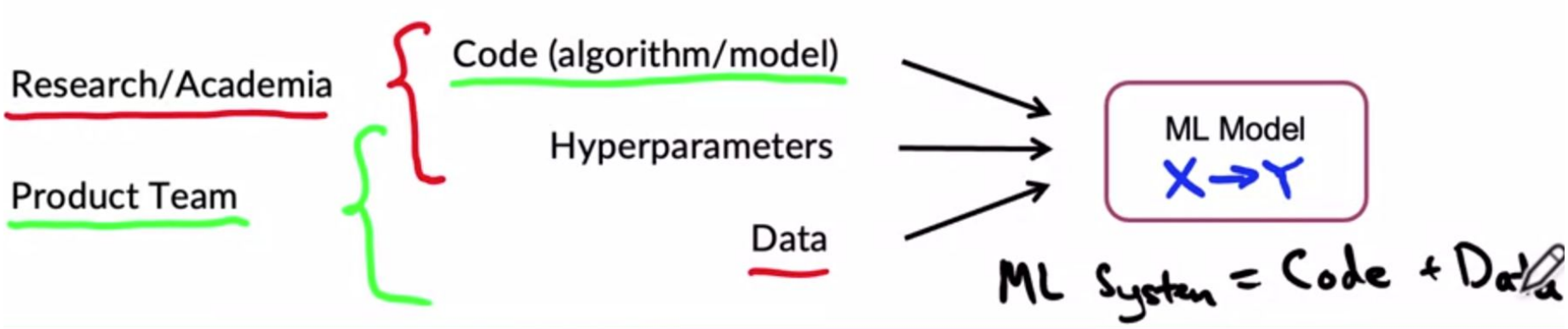
Agenda

- Intro to deep learning
  - DL in the machine learning project-cycle
  - What is DL and why should we use it?
    - Neural networks
    - Perceptrons
    - The feedforward pass
    - Backpropagation
    - Activation functions
    - Stochastic gradient descent
    - Convolutional neural networks
    - Data augmentation
    - Residual networks (ResNet)
    - Binary vs multiclass classifiers
    - Transfer learning and embeddings
  - Our goal: build a cats vs dogs image classifier in PyTorch



From Andrew Ng's Intro to ML in Production

The data driven-approach to ML



From Andrew Ng's Intro to ML in Production

Knowing the jargon

Model aka network aka architecture (although a fine-grained distinction exists here with respect to training)

Weight aka parameter aka connection strength

Hyperparameter

Capacity

Loss aka cost aka error function

Error rate aka 1 - accuracy

Sensitivity aka recall

Activation function

Transfer learning

Epoch vs Batch

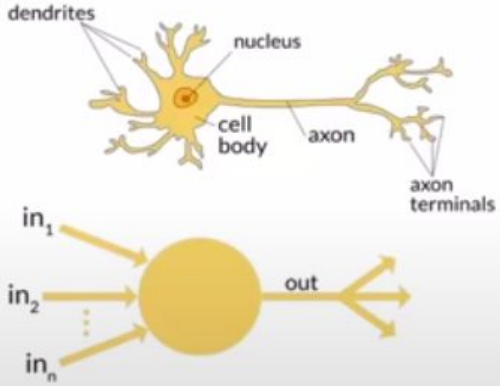
Convolution, receptive field

Linear aka “Dense”aka ‘fully connected’ layer

What is an artificial neural network?

In 1943 Warren McCulloch, a neurophysiologist, and Walter Pitts, a logician, teamed up to develop a mathematical model of an artificial neuron. They declared that:

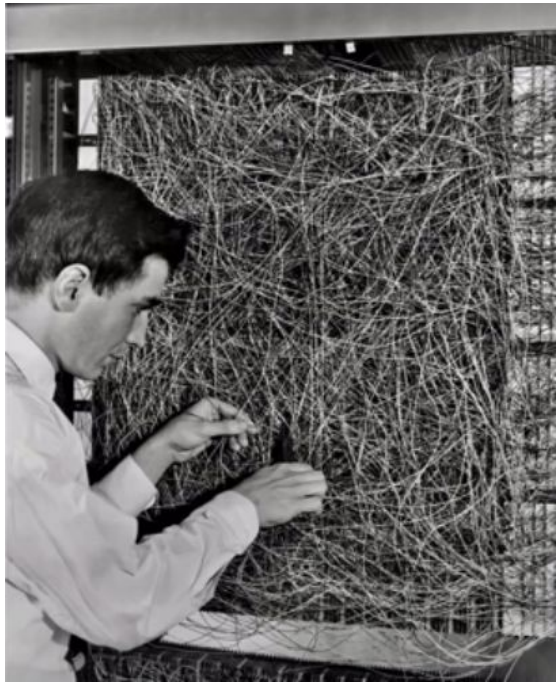
> *Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms.* (Pitts and McCulloch; A Logical Calculus of the Ideas Immanent in Nervous Activity)



Artificial “neurons” are also called “processing units”

Recommended read: “The Man Who Tried to Redeem the World with Logic”

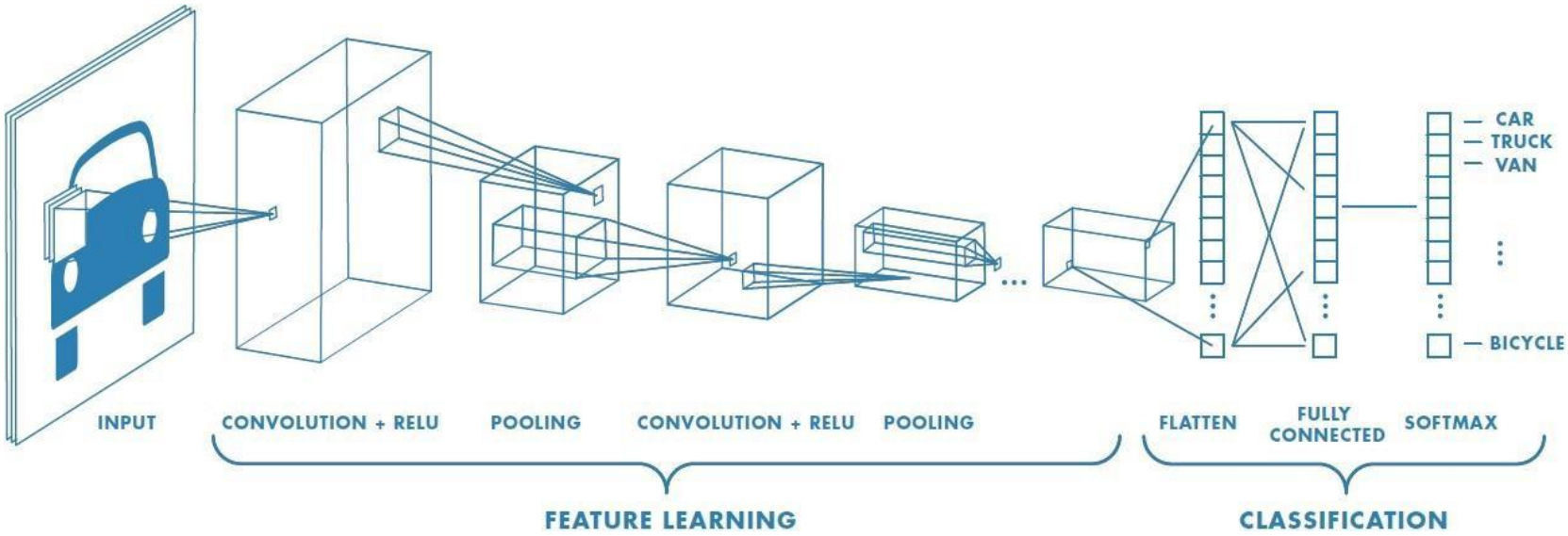
What is an artificial neural network?



"we are about to witness the birth of such a machine – a machine capable of perceiving, recognizing and identifying its surroundings without any human training or control".
Frank Rosenblatt

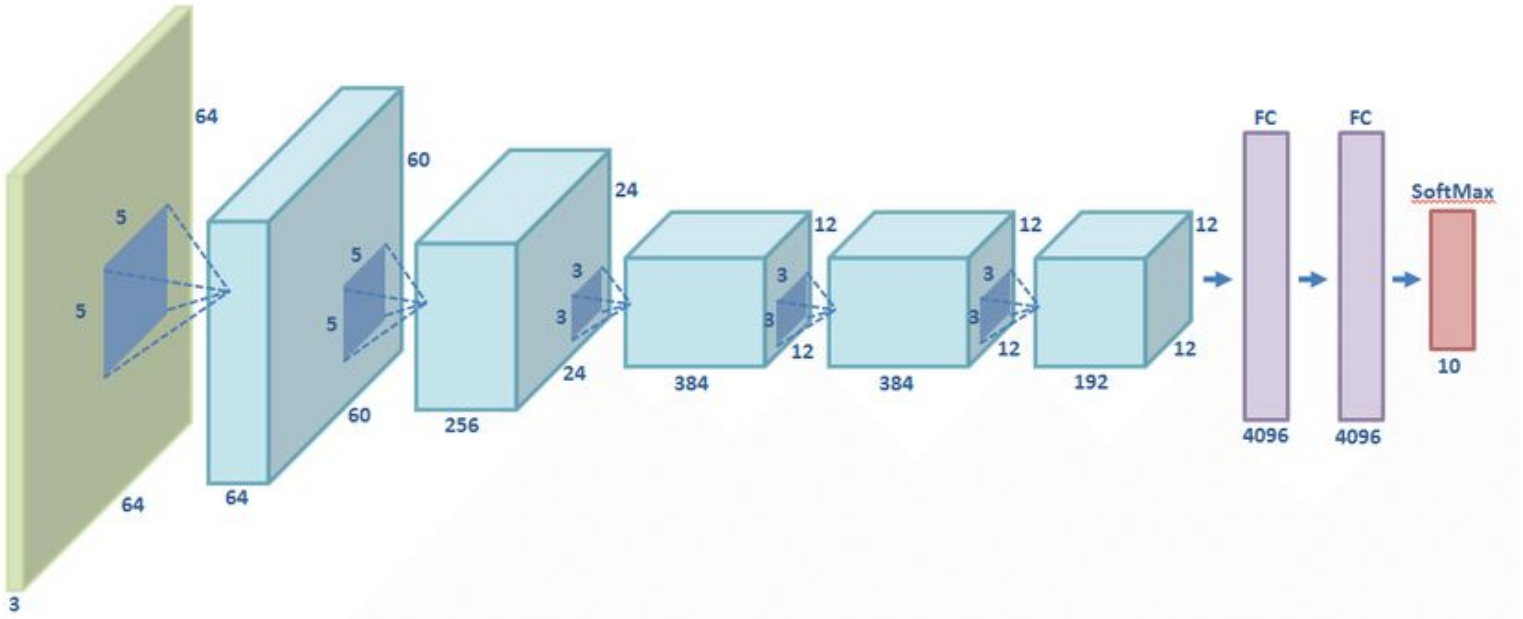
Image: Frank Rosenblatt with the Mark I Perceptron at Cornell University (1961)

What is deep learning?



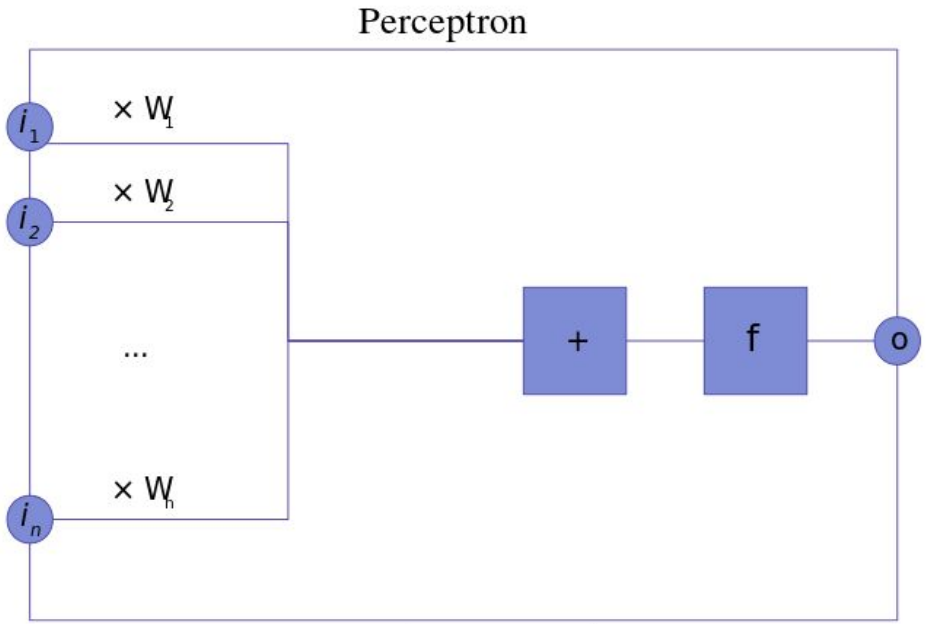
Stir the pile

Different network architectures represent different ways to ‘stir the pile’



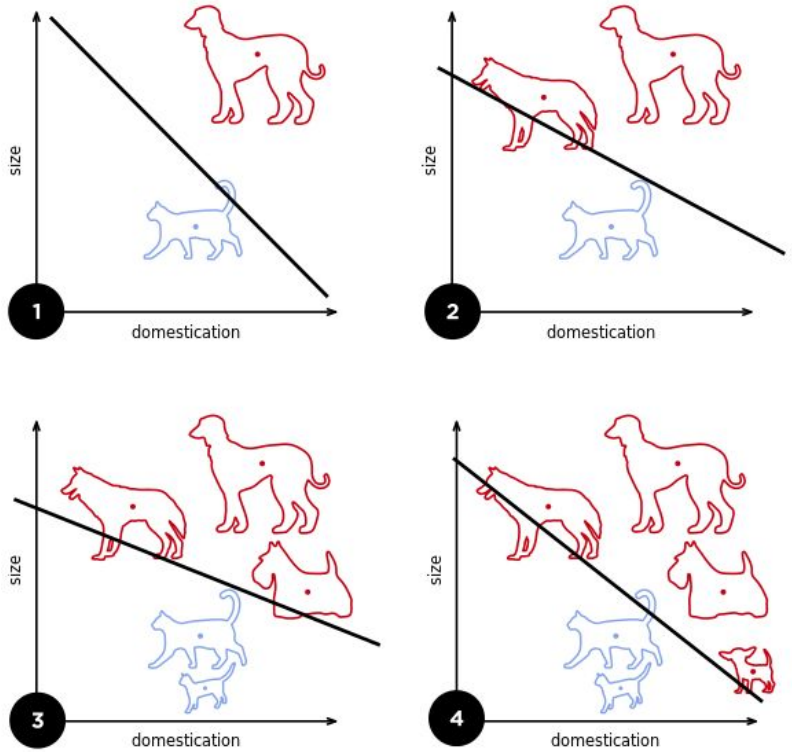
Architecture of the Alexnet network

What is a perceptron?



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

What is a perceptron?



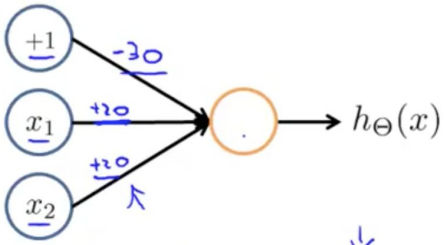
A fine-grained discussion about the differences in logistic regression and the perceptron algorithm

What does a neural network ‘learn’?

It learns “*weights*” aka “*parameters*” aka “*connection strengths*” that minimize a measure of error (aka “loss” aka “cost”) function given a set of training inputs x (aka “a dataset”)

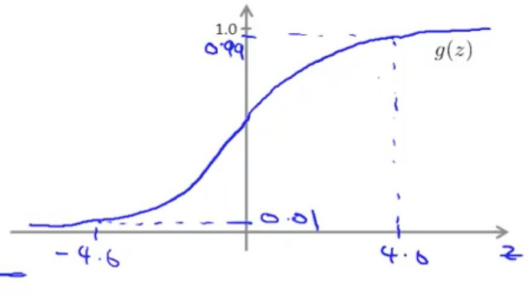
Simple example: AND

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1$ AND x_2



$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$

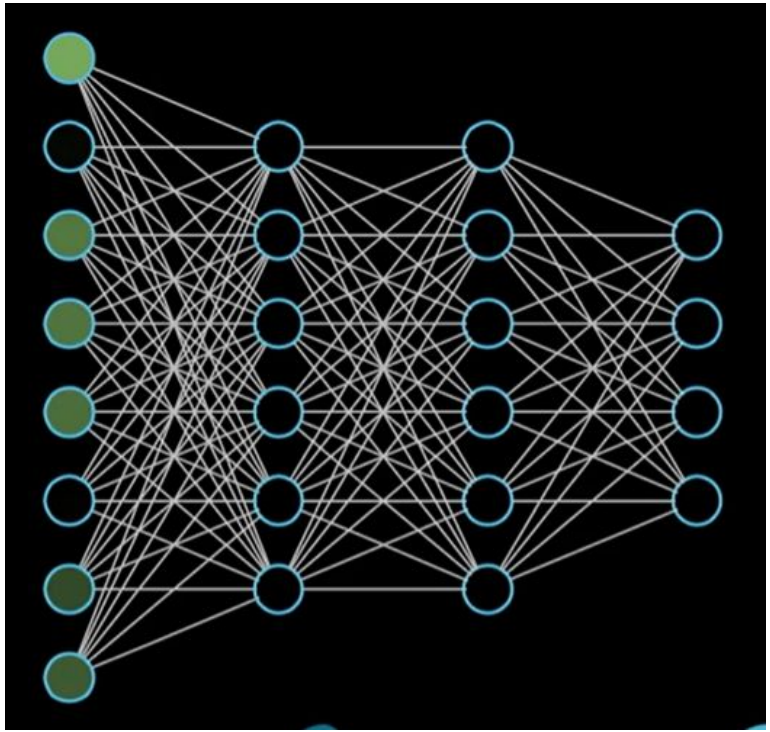
$\Theta_{10}^{(1)}$ $\Theta_{11}^{(1)}$ $\Theta_{12}^{(1)}$



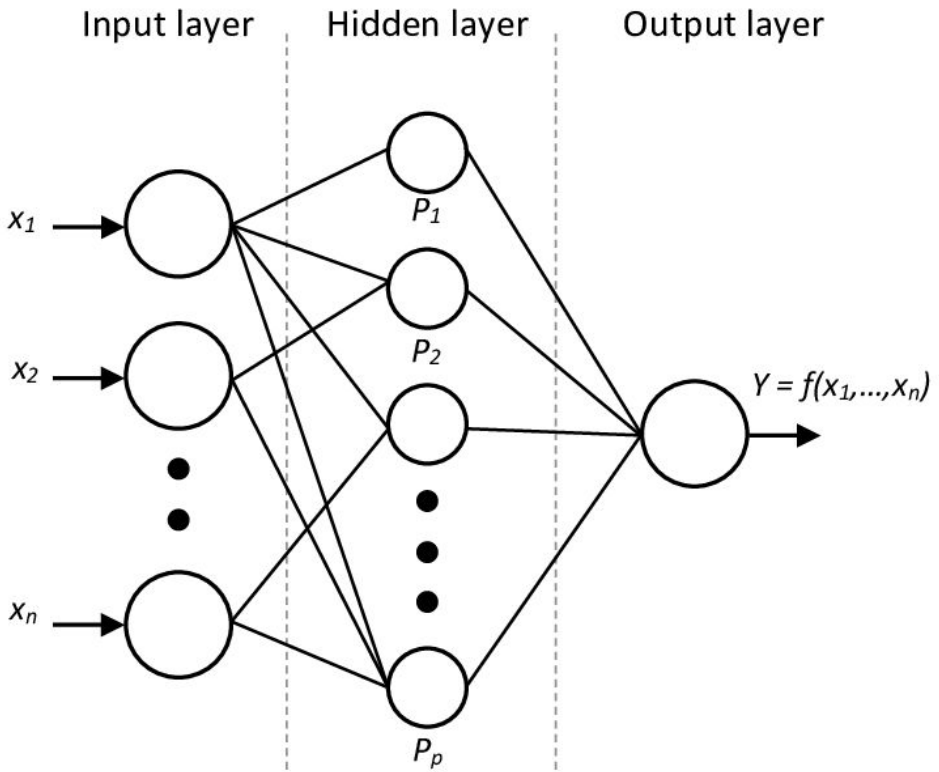
x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

$h_\Theta(x) \approx x_1$ AND x_2

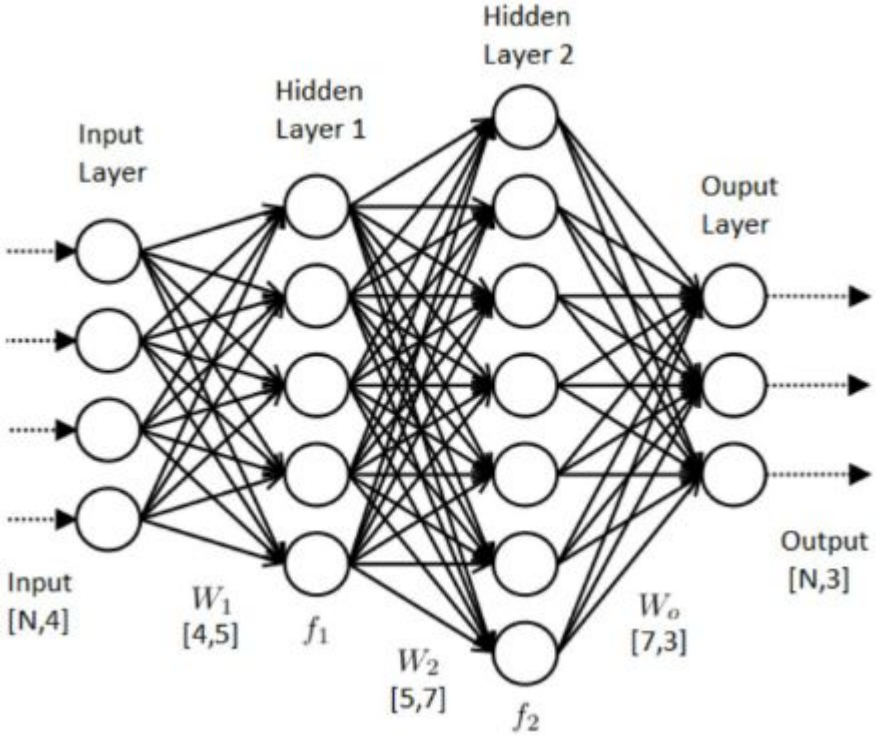
Multilayer perceptrons



Hidden layers

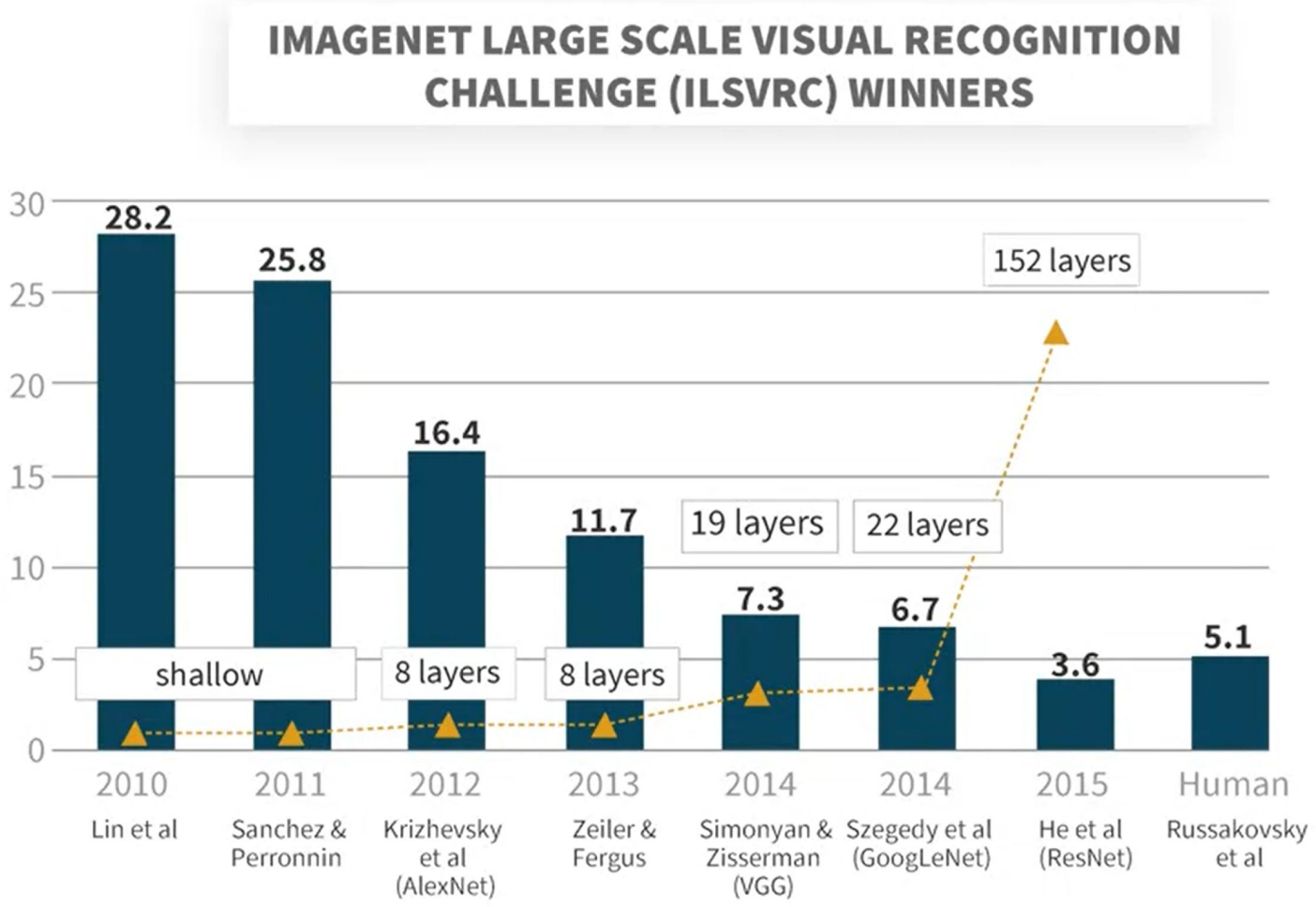


Hidden layers

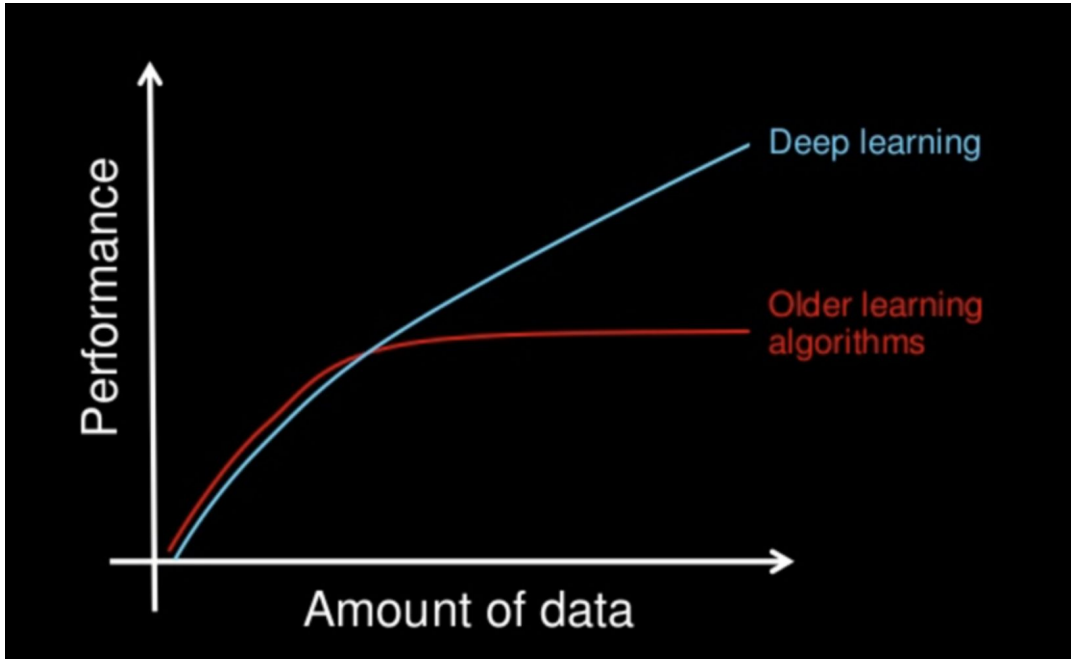


Observation: the term ‘deep learning’ is **pure hype**

- Most of the techniques for “deep learning” have been around for at least 50 years
  - 50 years if we count the invention of computation graphs for backpropagation
  - 80 years if we count the first neural network with manually set weights
  - 60 years if we count the invention of the perceptron
- ‘Deep learning = new AI’ became a thing when people started using neural networks to win the ImageNet challenge about 10 years ago
  - This rebranding and popularization came mainly through four academic labs: Hinton’s, LeCun’s, Bengio’s and Schmidhuber’s
  - The real breakthrough came not from new techniques but from more computing power (GPUs)
- Deep Learning does not do anything that linear and/or logistic regression cannot do
  - Think of deep learning as ‘regression on steroids’ and ‘fancy curve fitting’
  - ‘Deep Learning’ does not understand causality, it only picks on correlations


IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al

Why do we use deep learning?



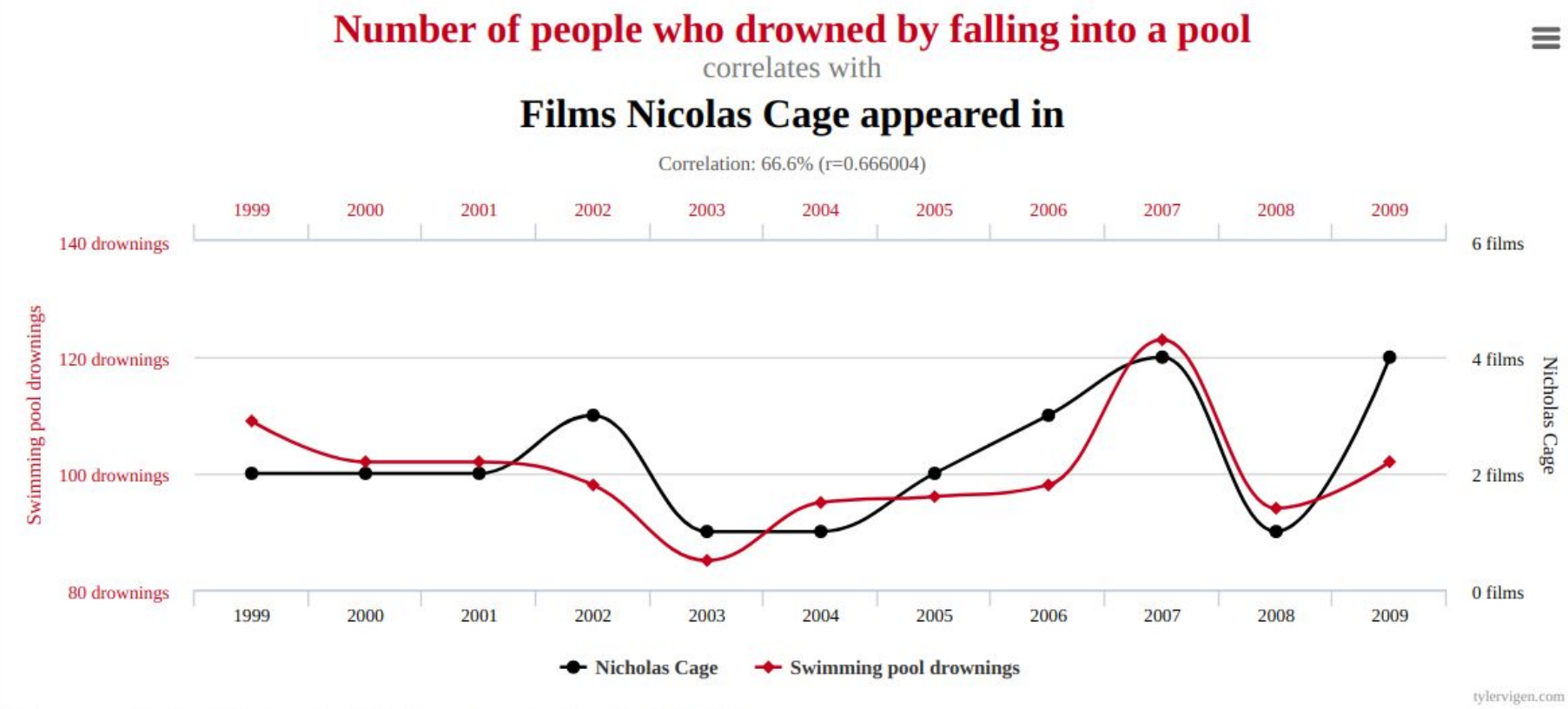
- Disclaimer: the No Free Lunch Theorems explain why this graph might be hype, although things like the current GPT experiments show promise

Applications of deep learning

- Computer vision: image classification, object detection, segmentation, image description, image generation, satellite and drone image analysis
- Natural Language Processing: translation, text and DNA sequence prediction, text generation
- Audio understanding, audio generation, lip reading
- Playing games, finding optimal strategies (when paired with reinforcement learning)
- Any system that requires the finding of **significant correlations** or patterns/”intuitions” in the data, i.e. the ‘thinking fast’ systems referred by Daniel Kahneman in his famous book.

Why do we use deep learning?

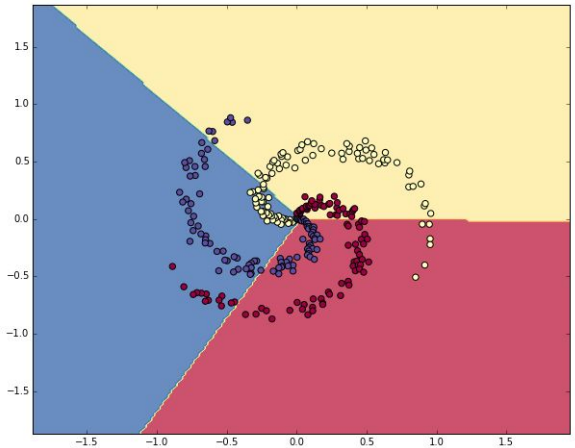
- Deep neural networks are ***universal function approximators***
  - *This means that given the right architecture and training, they can **compute everything that's computable**, just like a Universal Turing Machine*
  - *In practice this means that deep networks are **very good** at finding very sophisticated decision boundaries (aka "curve fitting")*
  - *Finding the function means the finding **architecture and or training** of the neural network*
  - *It's **fair** to call deep neural networks "**linear regression on steroids**"*
  - *Deep neural networks pick hidden **correlations** in the data, but most architectures don't pick on **causality** (current research topic)*

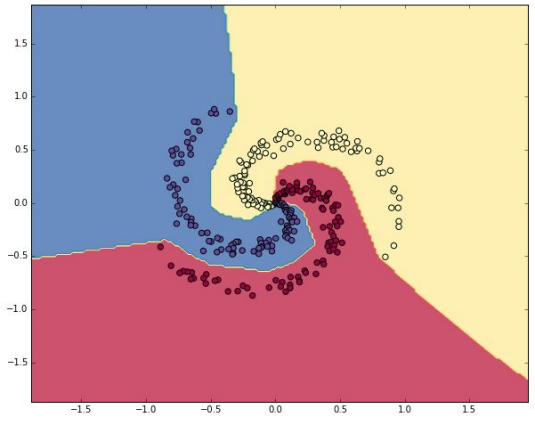


Data sources: Centers for Disease Control & Prevention and Internet Movie Database

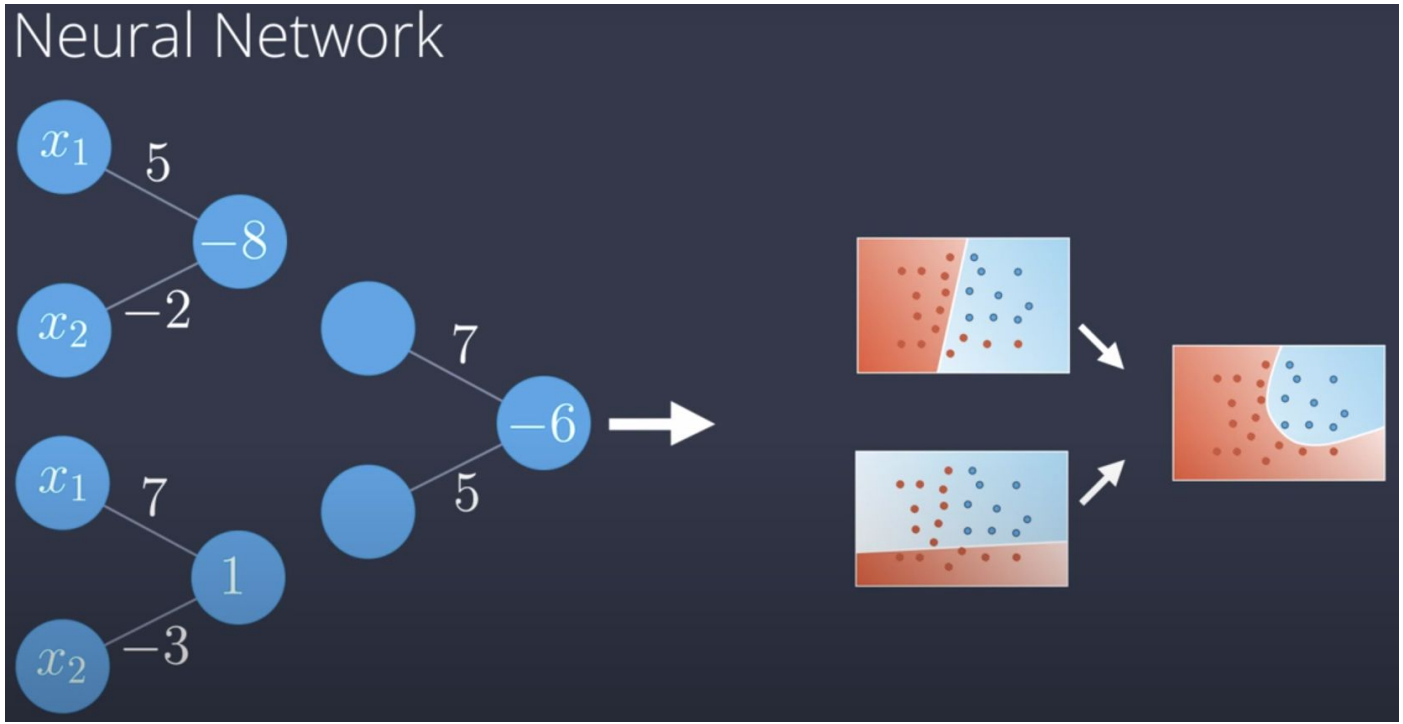
‘Deep learning’ is the art of making squiggly functions

- Squiggly function = ‘sophisticated decision boundary’
- More weights and non linear activations ~ more squiggly boundaries



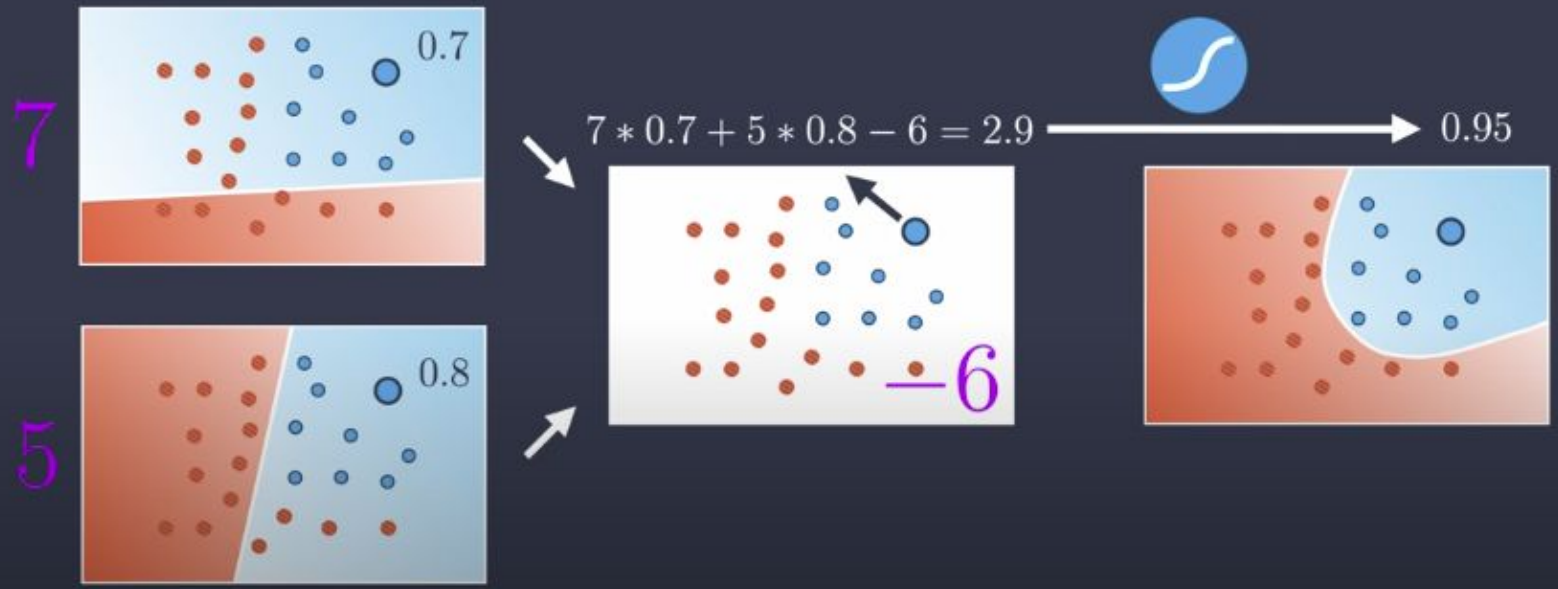


Learning non-linearities

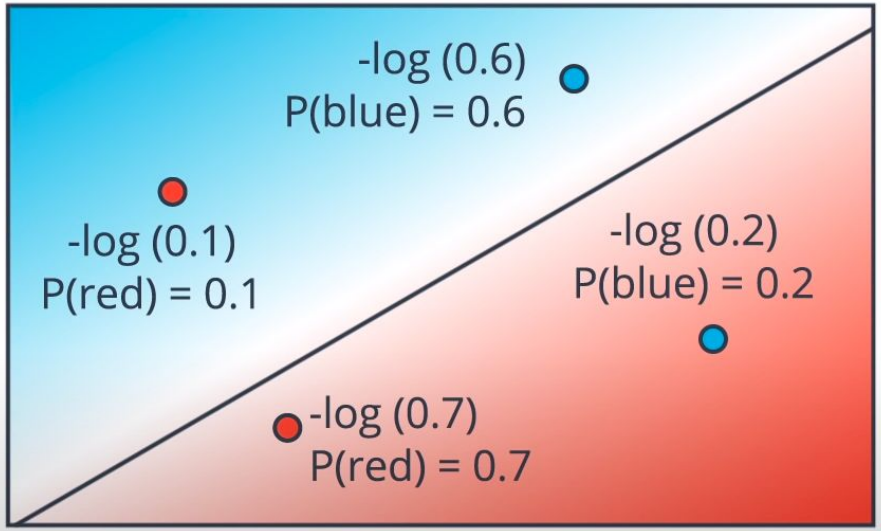


How neural networks learn non-linearities

Neural Network



Error Function



$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = 4.8$

0.51 1.61 2.3 0.36

If $y = 1$

$P(\text{blue}) = \hat{y}$

$\text{Error} = -\ln(\hat{y})$

If $y = 0$

$P(\text{red}) = 1 - P(\text{blue}) = 1 - \hat{y}$

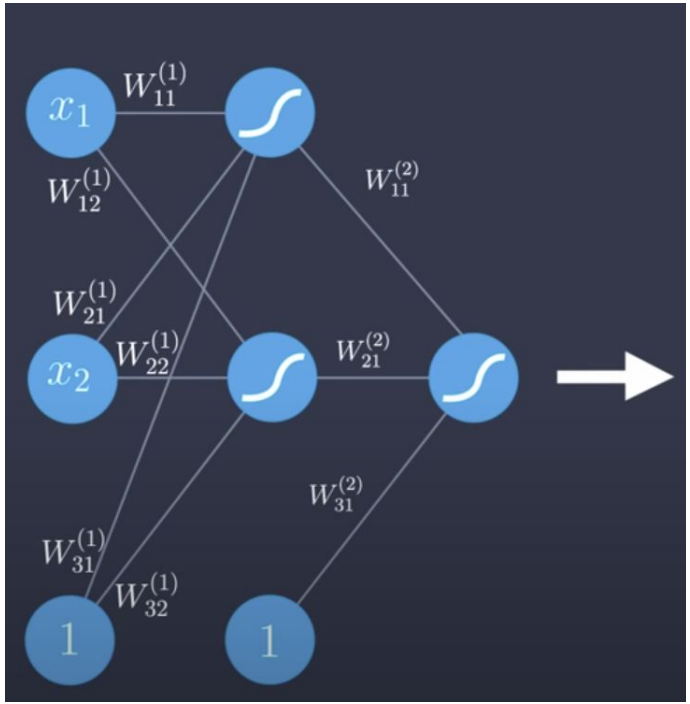
$\text{Error} = -\ln(1 - \hat{y})$

$$\text{Error} = -(1-y)(\ln(1-\hat{y})) - y\ln(\hat{y})$$

$$\text{Error Function} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\hat{y}_i)) + y_i\ln(\hat{y}_i)$$

Q: Which of these points have higher cross entropy?

The feedforward pass

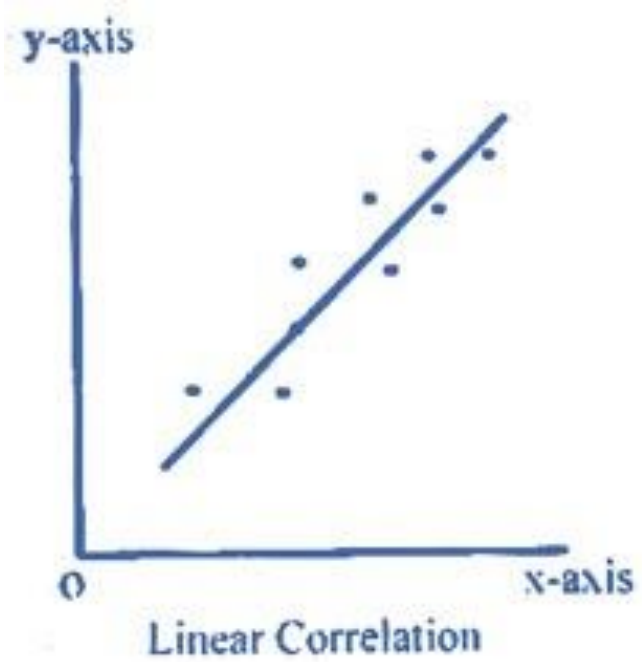


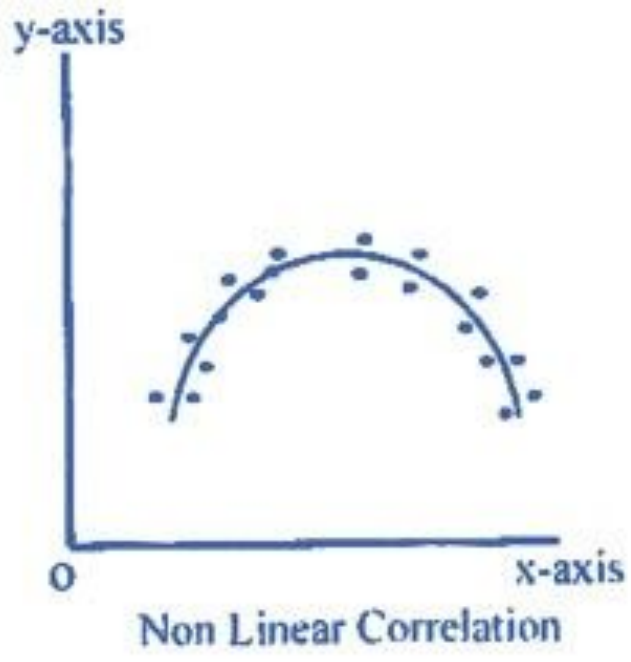
$$\hat{y} = \sigma \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix} \sigma \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

$$\hat{y} = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

[Understanding the feedforward pass]

Learning non-linearities

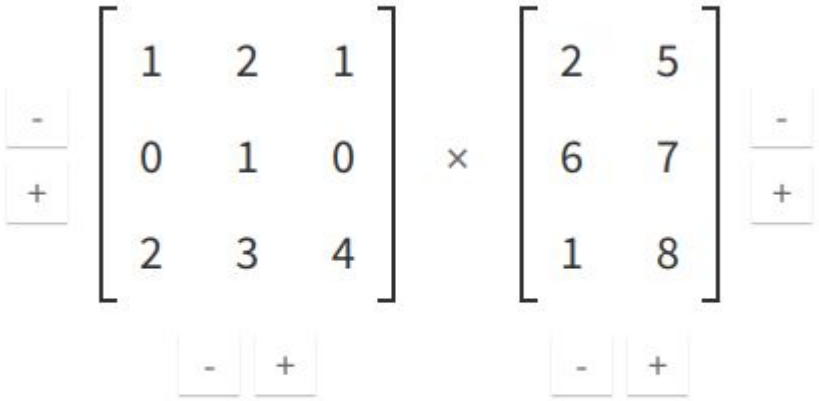




How neural networks learn non-linearities

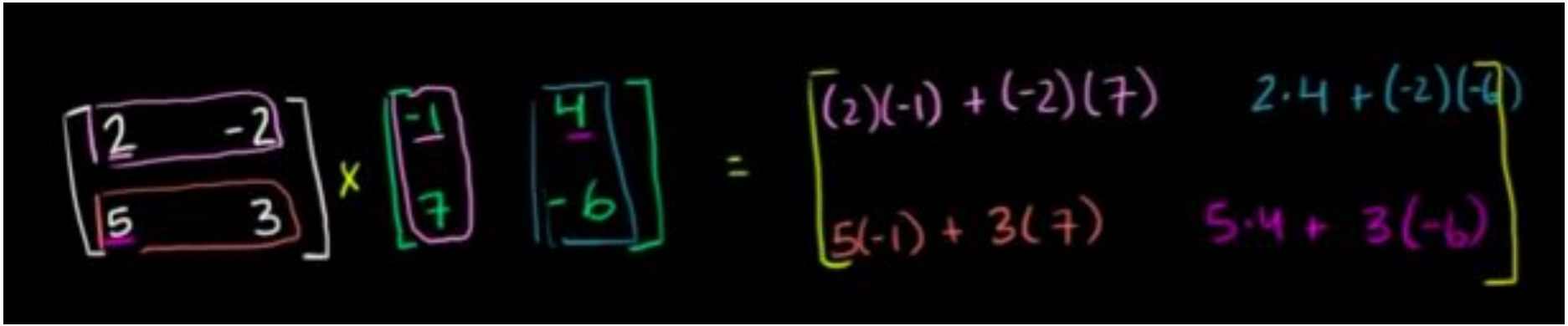
- Read: scalars, vectors, matrices, and tensors
  - Understanding rank terminology

Matrix multiplication



http://matrixmultiplication.xyz/

Matrix multiplication



Khan Academy Tutorial

Pay attention to the size of the input

Anthony Wang @aytwang · 15h

Replying to @the_antlr_guy

Wow, this is extraordinary. I can't count the number of hours I've spent in the debugger combing through every variable that could have gone wrong using print(x.shape) statements 😂

1 16

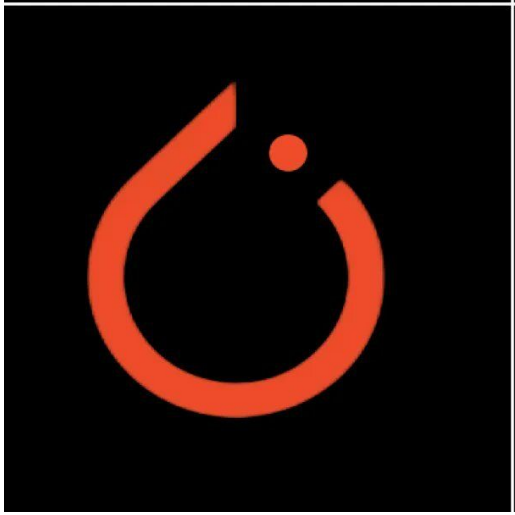
Terence Parr @the_antlr_guy · 15h

haha! Same here! Seems like print(x.shape) is the most common deep learning programming construct there is!

2 6 17

Upgrade from 1.x to 2.0:

Nothing works
Documentation missing.
What happened to tflite
CUDA hell
Dafuq is gradient_tape
Where's .contrib



Upgrade from 1.x to 2.0:

Everything works
Live Q&A before release.
Clear deprecations
.compile() ❤️
Actually open source
The best is yet to come

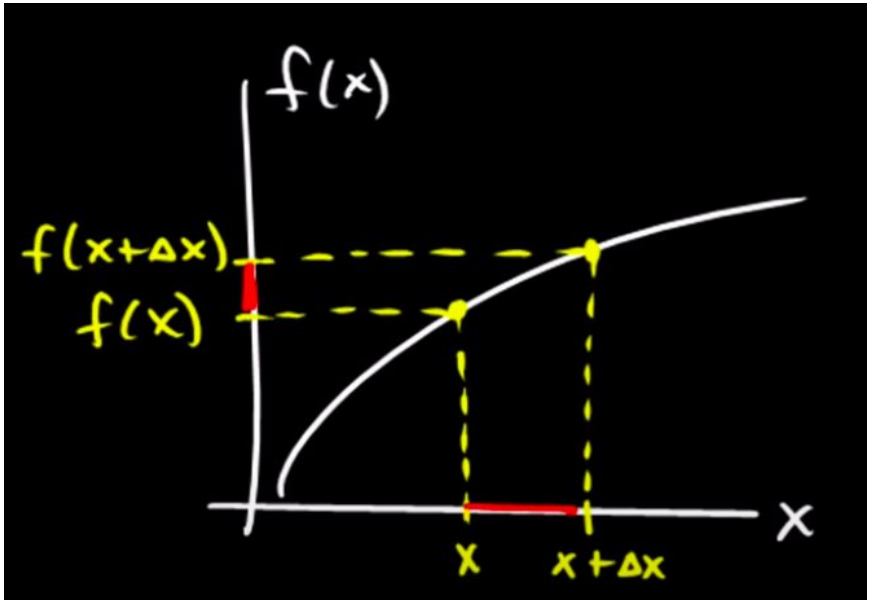
Practice: tensors in Pytorch

- Notebook
- Solutions

Questions:

- Why do we use bias terms?
- How is a torch tensor different from a numpy ndarray?
- Is a torch tensor always run on a GPU?
- What is an in-place operation? How are in-place methods named in Pytorch?
- Is the rectifier linear unit a linear or a non-linear function?

Numerical calculation of derivatives



Numerical derivatives

Numerical and analytic derivatives in Python

Backpropagation

Finding the derivative of the error [edit]

Calculating the partial derivative of the error with respect to a weight w_{ij} is done using the chain rule twice:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net}_j}\frac{\partial \mathrm{net}_j}{\partial w_{ij}} \qquad \textbf{(Eq. 1)}$$

In the last factor of the right-hand side of the above, only one term in the sum net_j depends on w_{ij}, so that

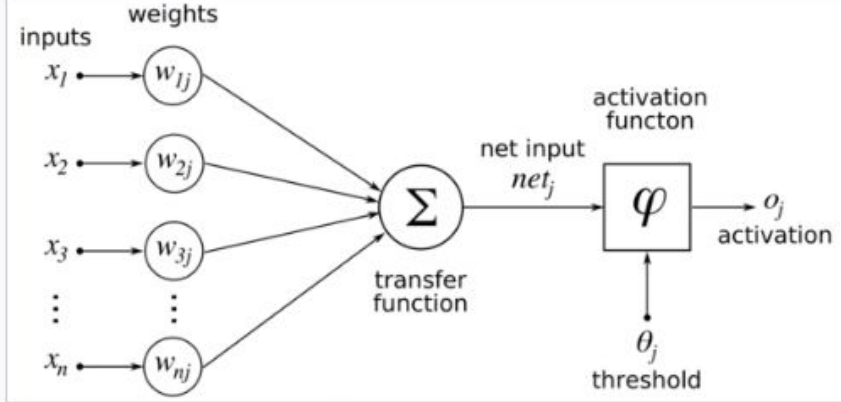


Diagram of an artificial neural network to illustrate the notation used here.

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

The chain rule

Computing the derivative of a composite function:

$$f(x) = u(v(x)) \qquad \frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

$$f(x) = u(v(w(x))) \qquad \frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dw} \cdot \frac{dw}{dx}$$

Gradient computation graphs

Backpropagation: a simple example

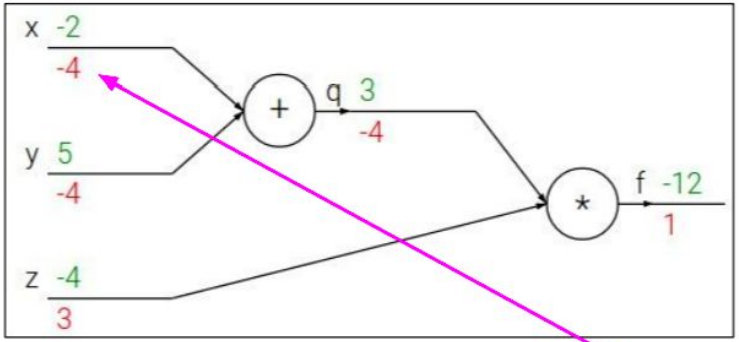
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

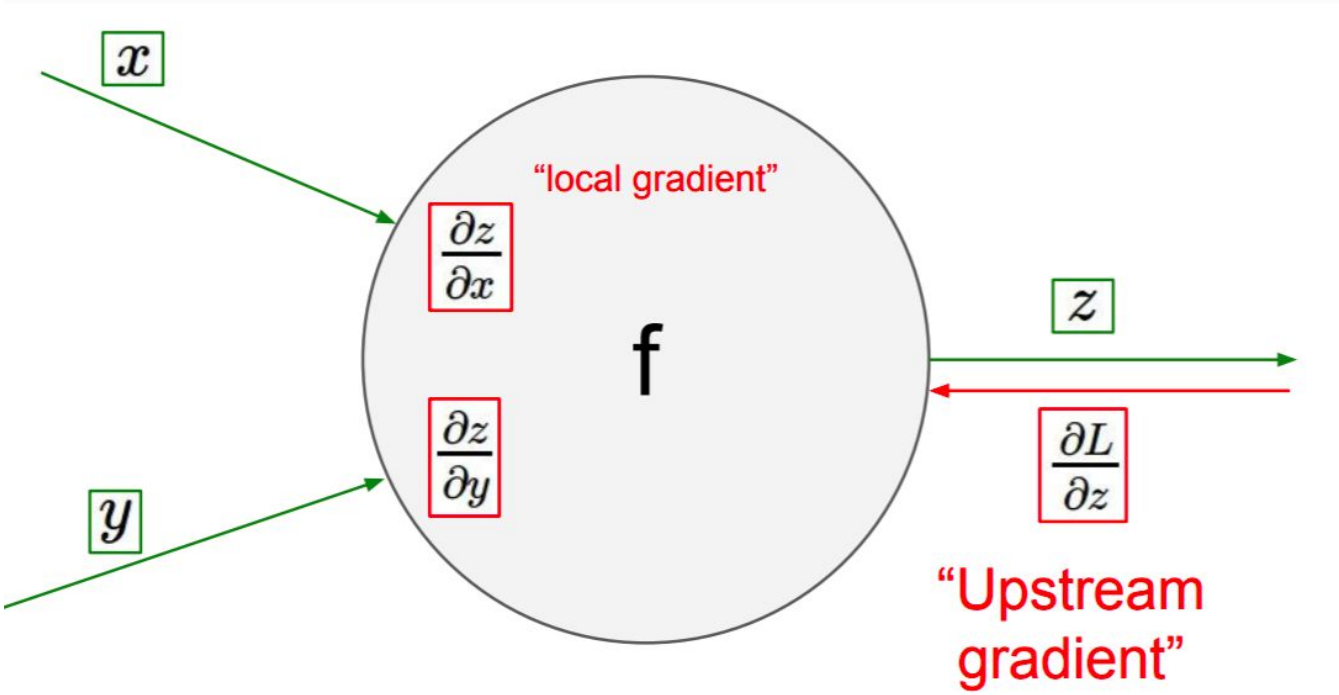
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Upstream gradient

Local gradient

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Gradient computation graphs

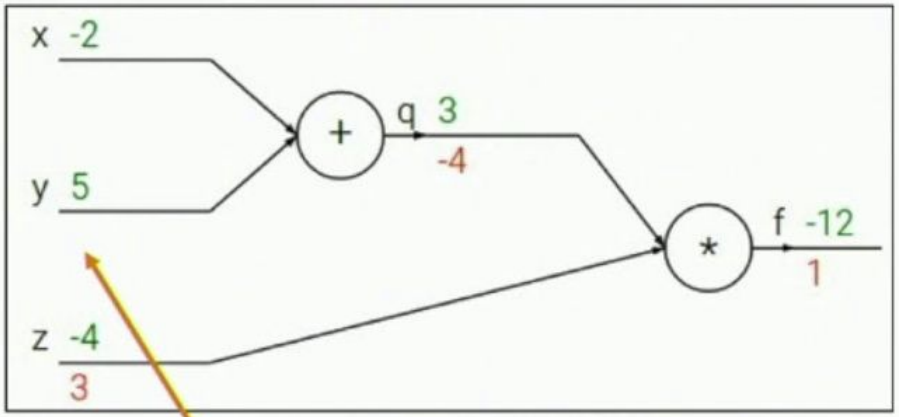


http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



1. Forward pass: Compute outputs

$$q = x + y \qquad f = qz$$

2. Backward pass: Compute derivatives

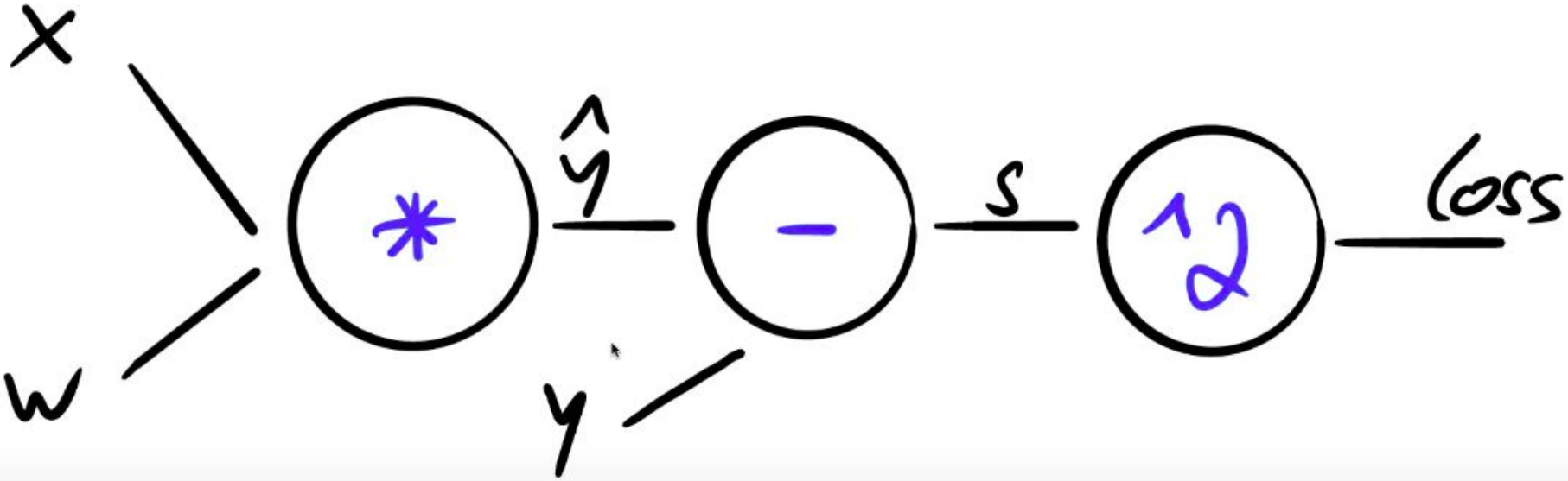
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Chain Rule

$$\frac{\partial f}{\partial y} = \frac{\partial q}{\partial y}\frac{\partial f}{\partial q}$$

Justin Johnson's lecture

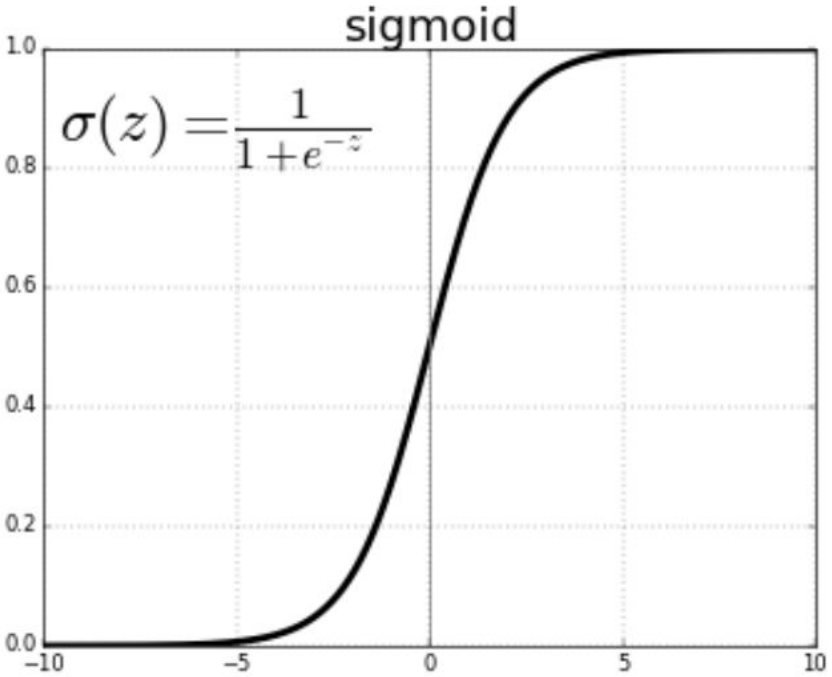
$\hat{y} = w \cdot x$ $\quad loss = (\hat{y} - y)^2 = (wx - y)^2$



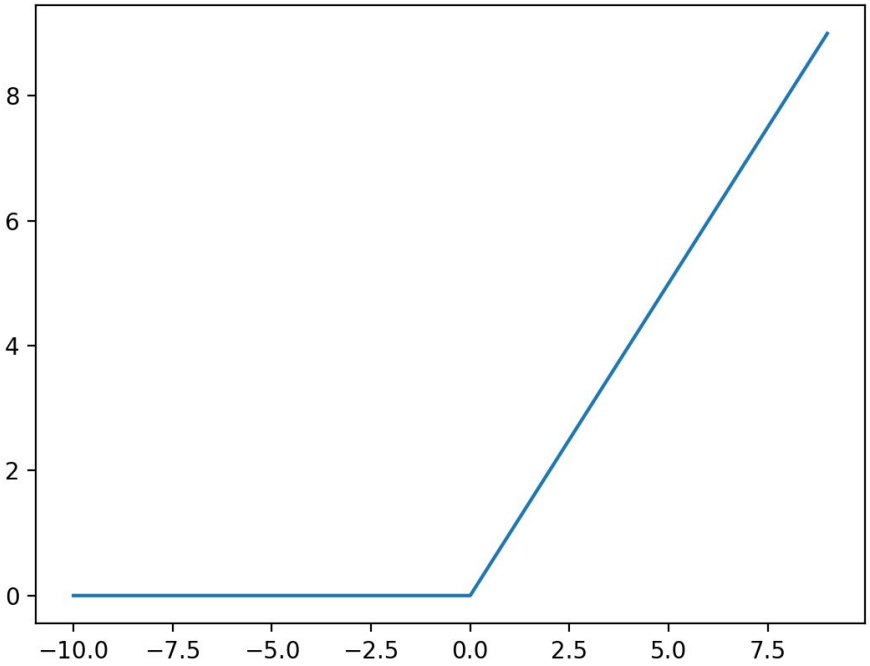
Backpropagation in PyTorch

- Watch this (clearest, most practical explanation on backprop ever)
  - Notebook
  - Solution
- Review questions
  - What is backpropagation doing?
  - What needs to happen before we call **backward()** in PyTorch?
  - What is the gradient computation graph?
  - Why is **grad_fn** changing with every operation done to something involving the weights?

The sigmoid activation function



The rectifier linear unit (ReLU) activation function



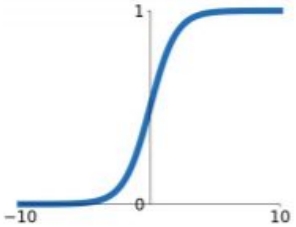
$$f(x) = x^+ = \max(0, x)$$

Activation functions

ReLU is a good default choice for most problems

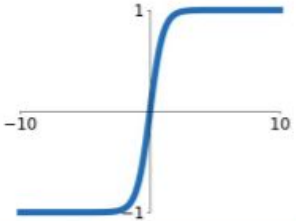
Sigmoid

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



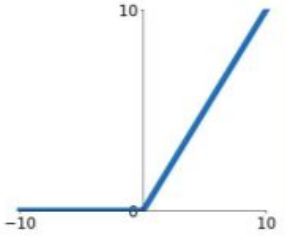
tanh

$$\tanh(x)$$



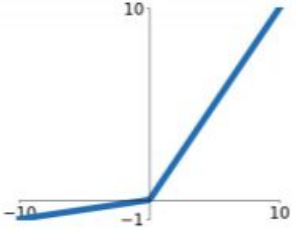
ReLU

$$\max(0, x)$$



Leaky ReLU

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

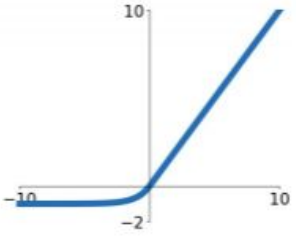


Maxout

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

ELU

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

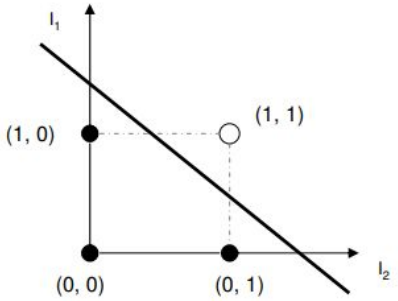


http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

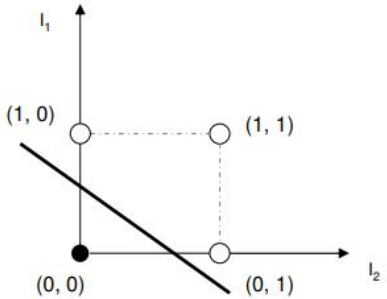
Doing logic with a neural network

- Experiments, Solution notebook

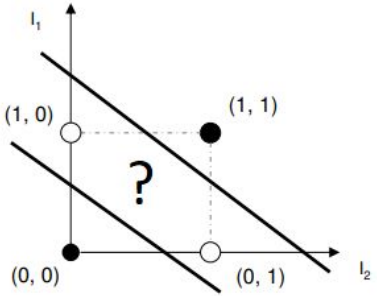
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



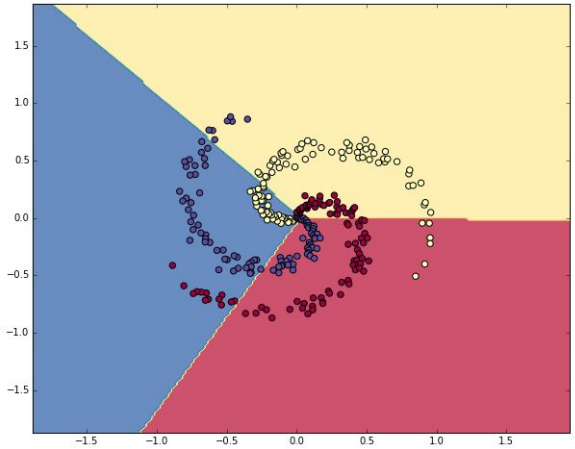
OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1

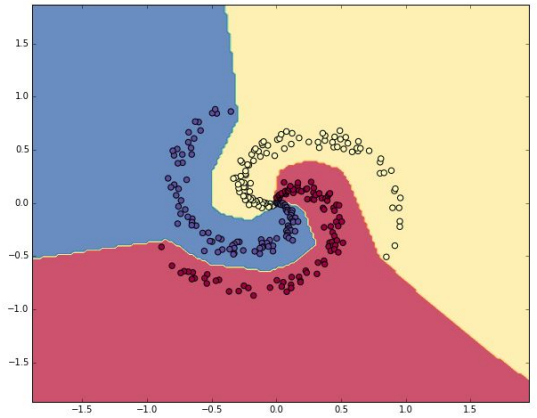


XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



Learning non-linearities





https://cs231n.github.io/neural-networks-case-study

Experiment notebook

Q: What happens with large learning rates? What happens with large coefficients for l2 regularization?

What is a GPU?

Most deep learning libraries require the use of NVIDIA graphical processing units with CUDA capabilities

Why use a GPU for deep learning?

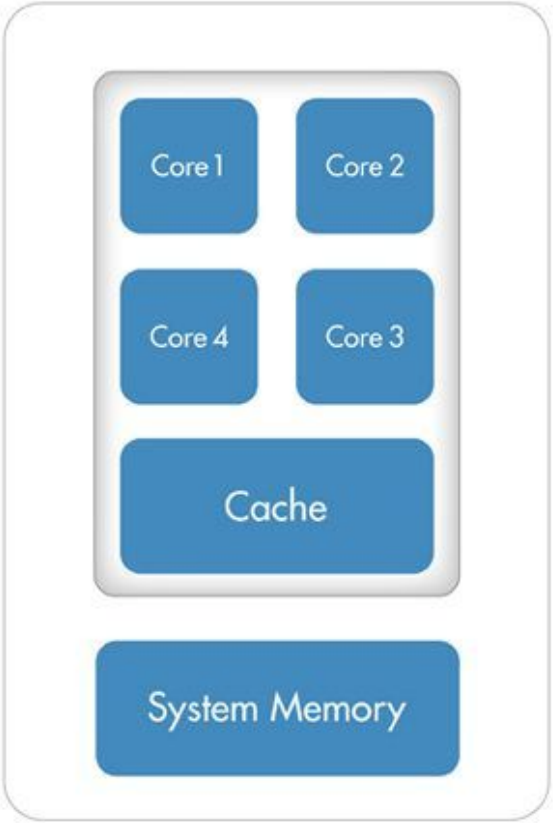
How is a CPU different from a GPU?

- More cores!

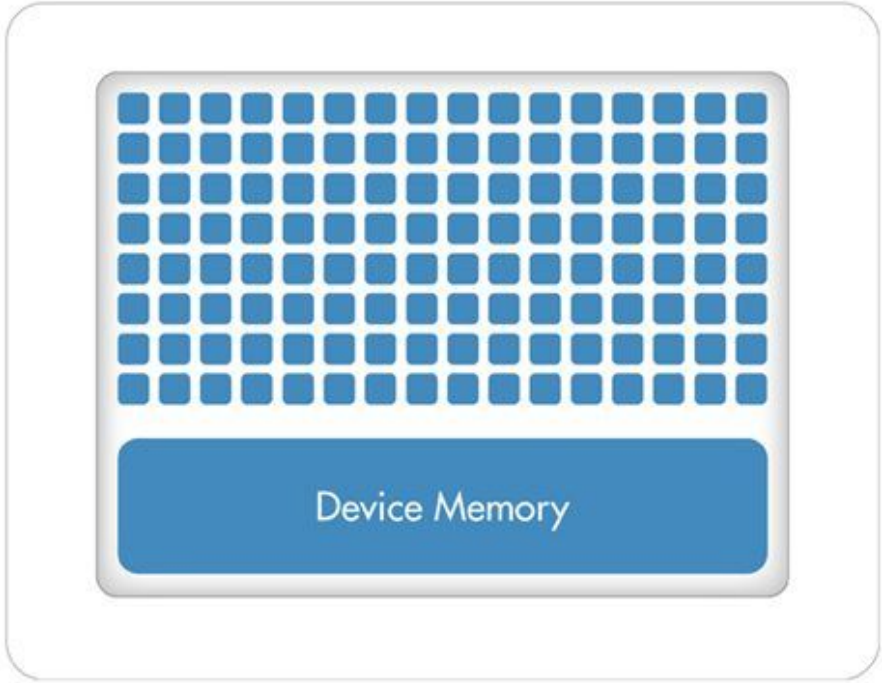
Why are CPUs less effective for deep learning?

- Less cores :(

CPU (Multiple Cores)



GPU (Hundreds of Cores)



Choose the right loss function for the task

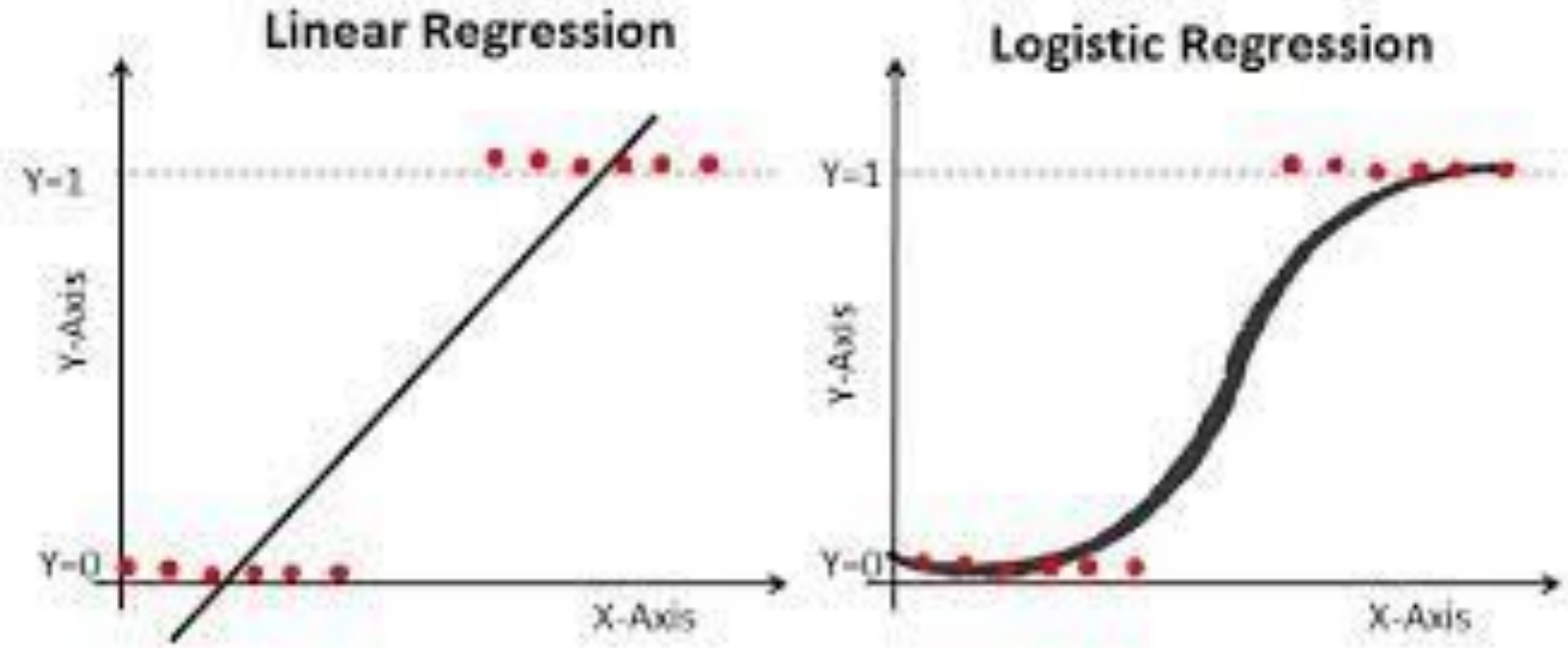
$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Mean Squared Error is a loss function for regression

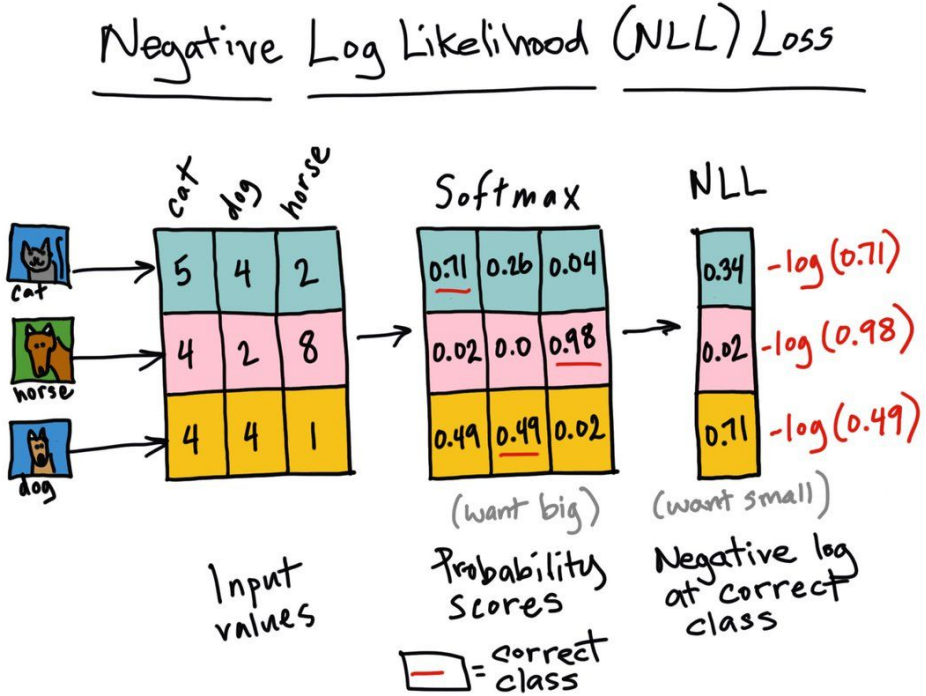
Choose the right loss function for the task

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Cross entropy is a function for classification, used interchangeably with negative log-likelihood, here we see it in its binary case


Linear Regression
Y=1
Y-Axis
Y=0
X-Axis
Logistic Regression
Y=1
Y-Axis
Y=0
X-Axis

Can we solve this task with logistic regression?



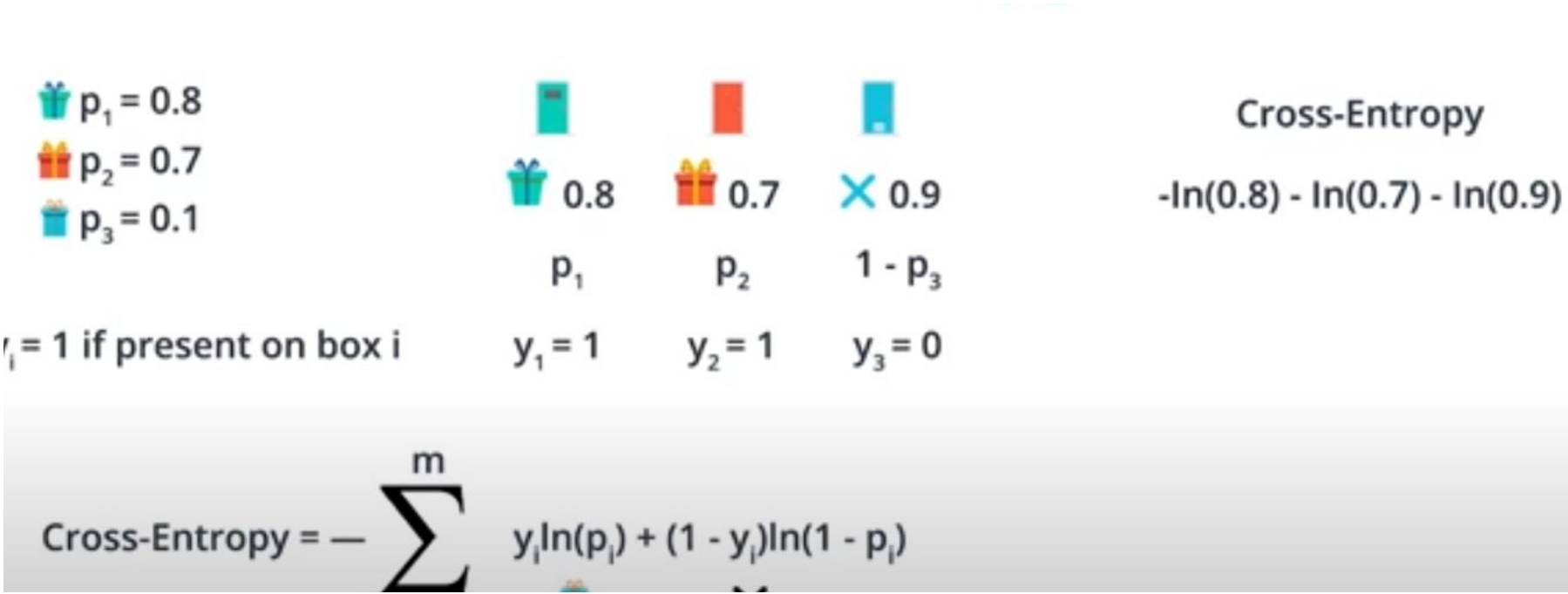
Q: Which loss function should we use for binary logistic regression?

A. Mean Squared Error
B. Cross Entropy
C. Negative log-likelihood (huh? Maybe read this too later.)
D. Precision & Recall
E. None of the above

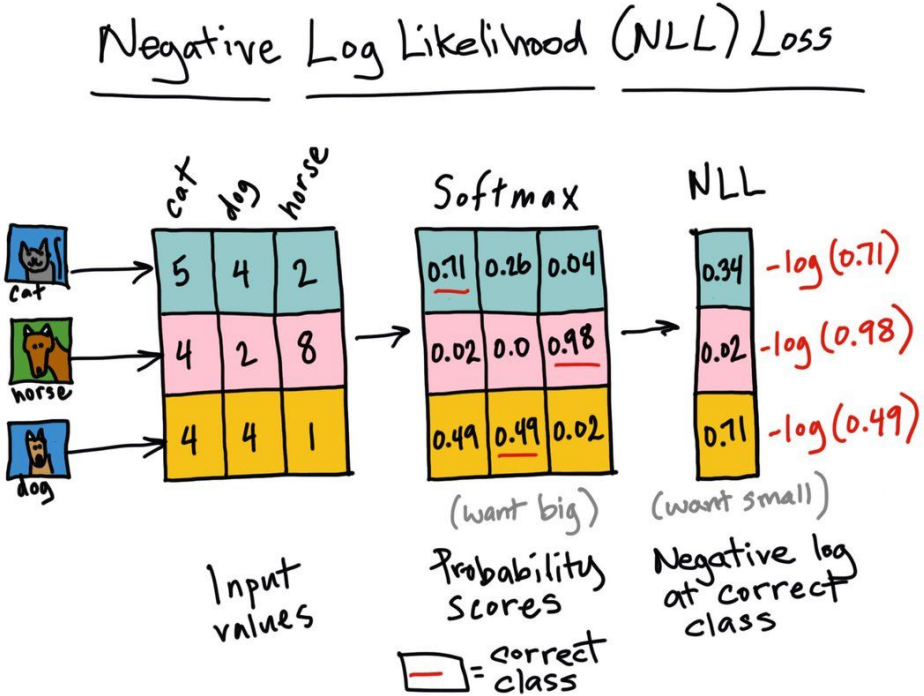
Cross Entropy

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Cross entropy



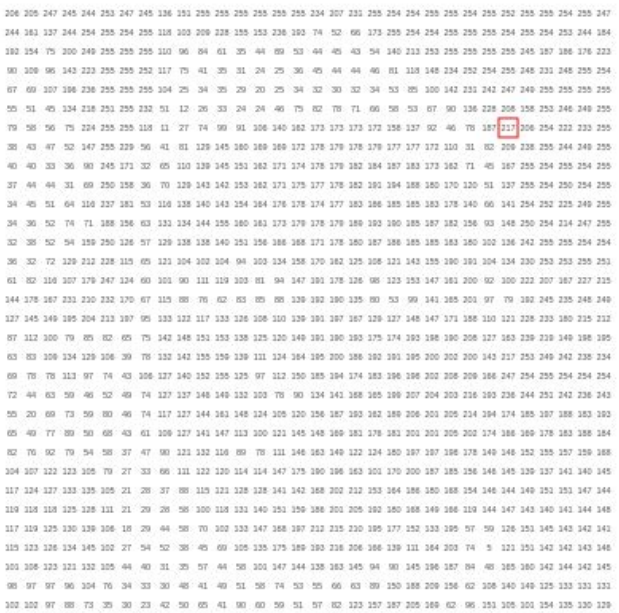
[Explore the Colab notebook]



[Exercise](#)
[Solution](#)

Refer to [this tutorial](#) and [this discussion](#)

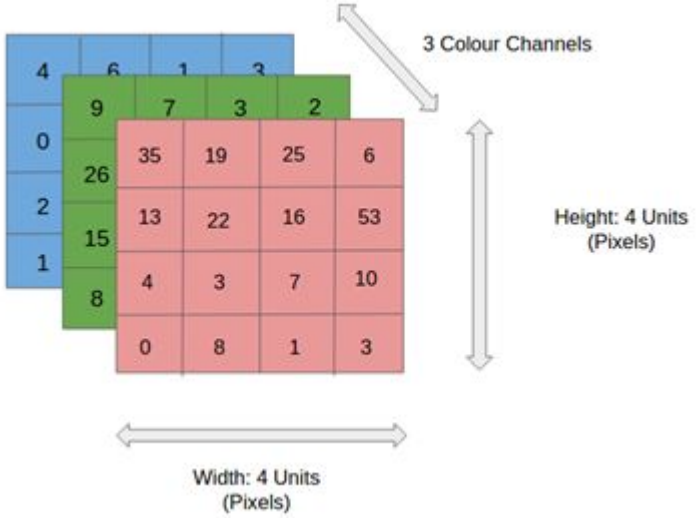
How do computers represent images?



https://setosa.io/ev/image-kernels/

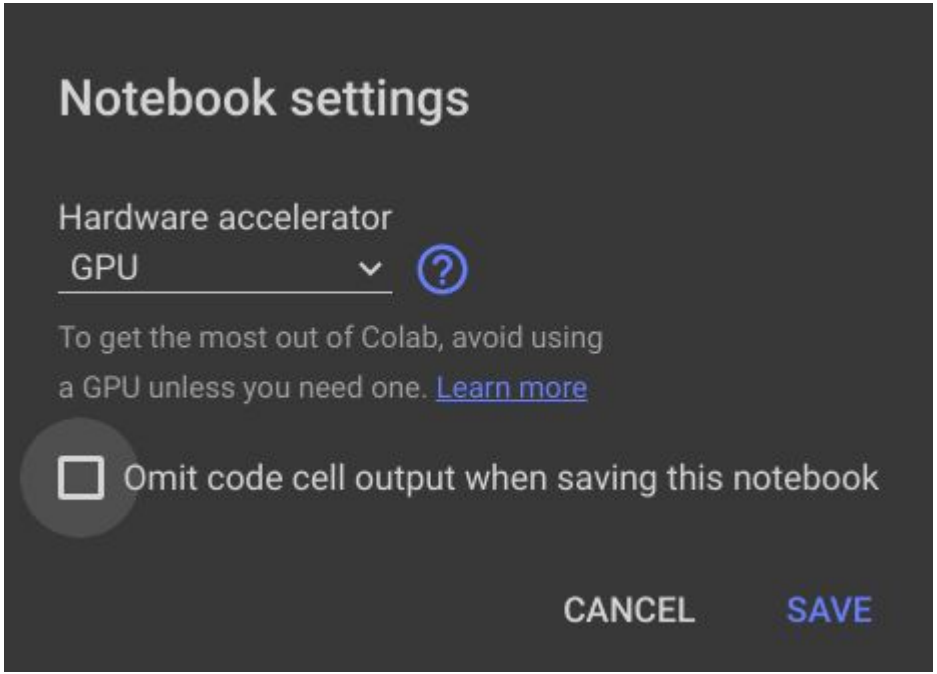
Images as tensors

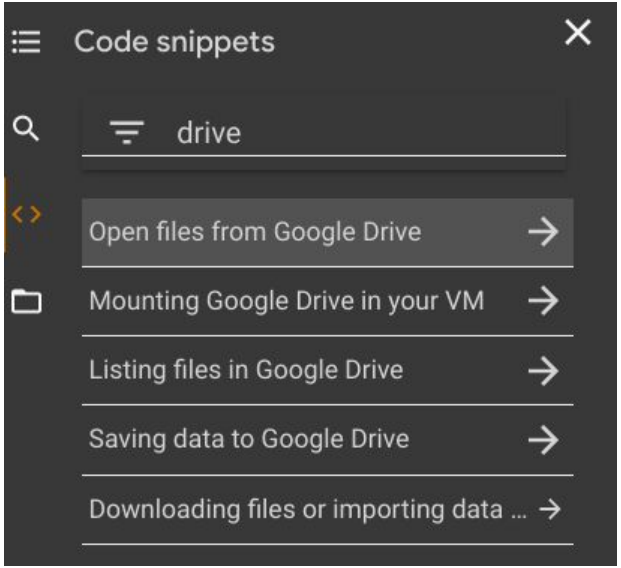
What a human sees



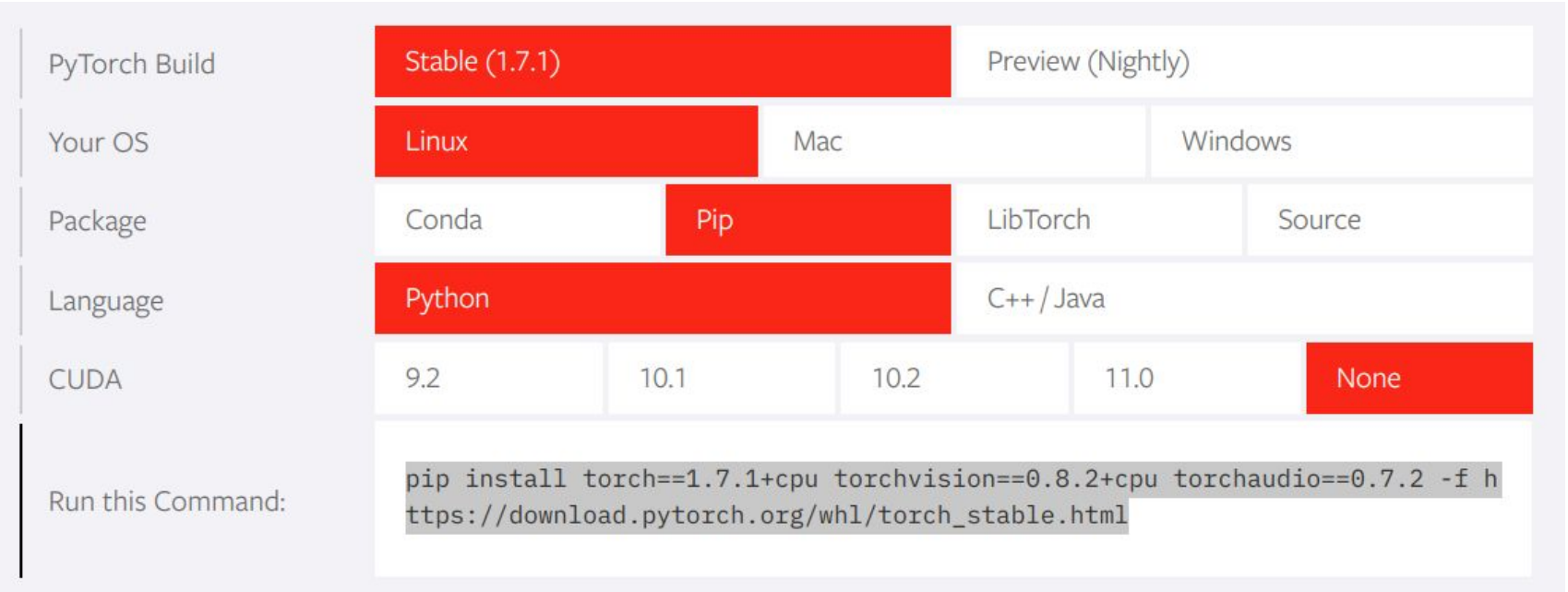
What the computer ‘sees’

Setting up Google Colab



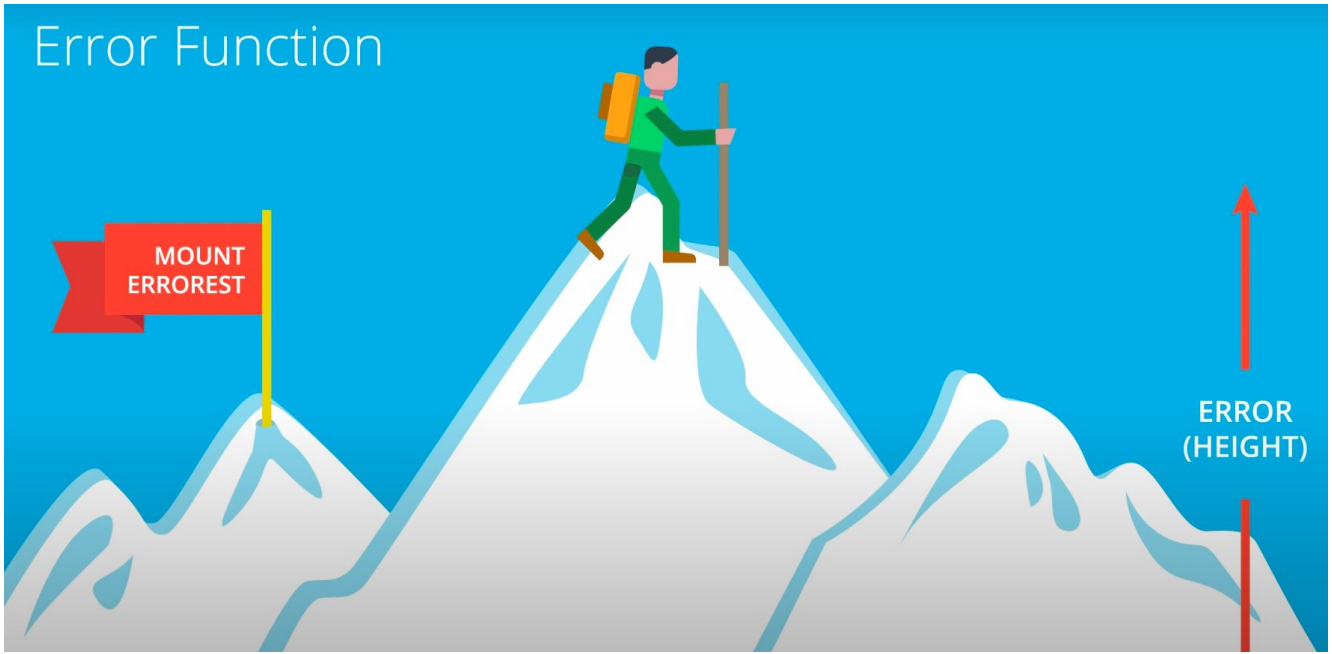


Installing PyTorch



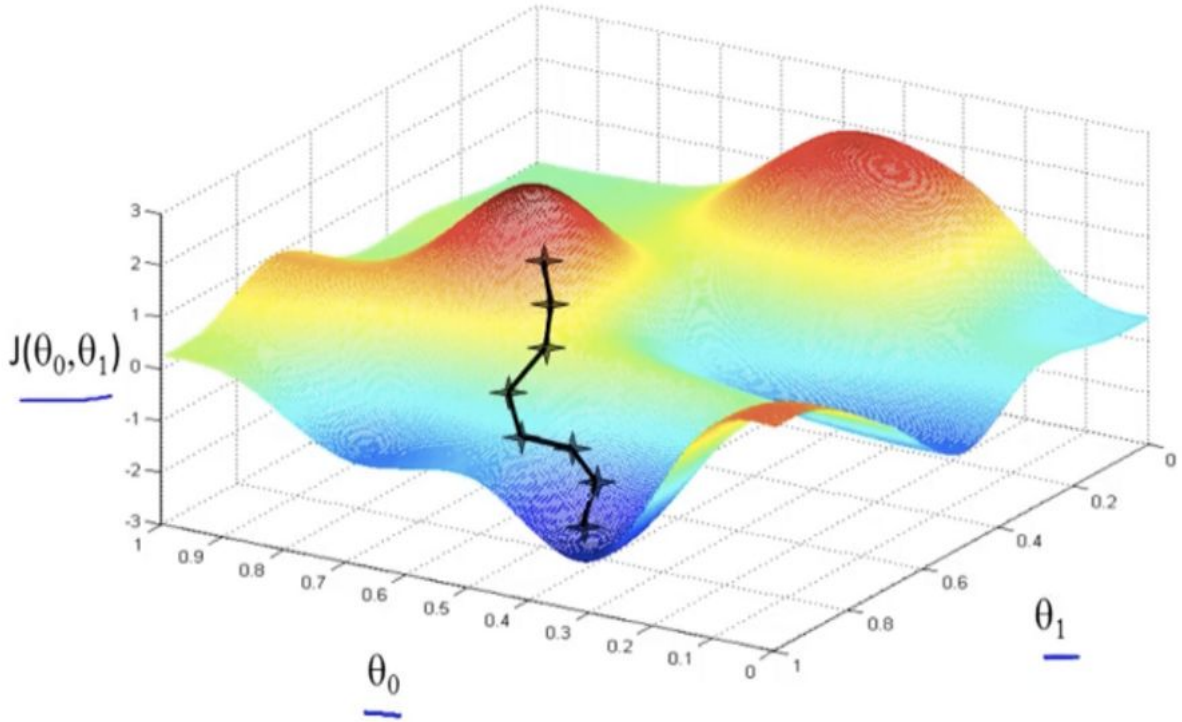
https://pytorch.org/

Gradient descent

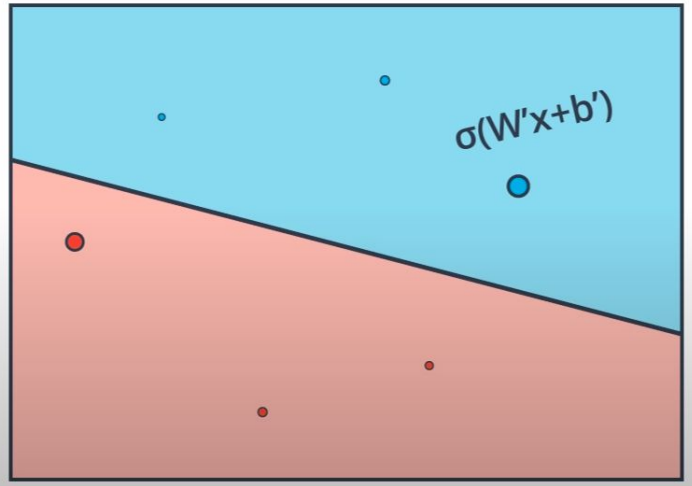


https://github.com/fastai/fastbook/blob/master/04_mnist_basics.ipynb

Stochastic gradient descent



Gradient Descent Algorithm



1. Start with random weights:

   $w_1, \ldots, w_n, b$

2. For every point $(x_1, \ldots, x_n)$:

   2.1. For i =1 . . . n

      2.1.1. Update $w_i' \longleftarrow w_i - \alpha(\hat{y}-y)x_i$

      2.1.2. Update $b' \longleftarrow b - \alpha(\hat{y}-y)$

3. Repeat until error is small

Implementing gradient descent

Scalar to Scalar

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If x changes by a small amount, how much will y change?

Vector to Scalar

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is **Gradient**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will y change?

Vector to Vector

$$x \in \mathbb{R}^N, y \in \mathbb{R}^M$$

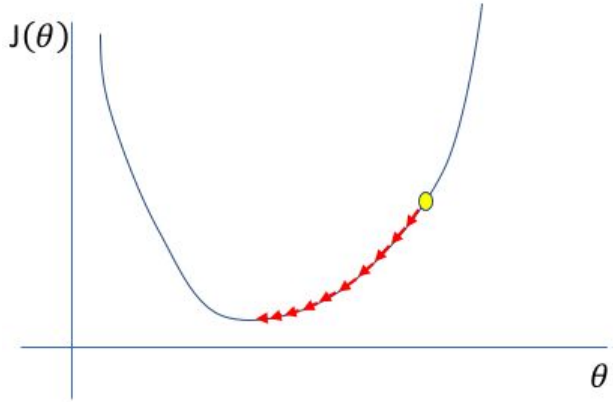
Derivative is **Jacobian**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \quad \left(\frac{\partial y}{\partial x}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will each element of y change?

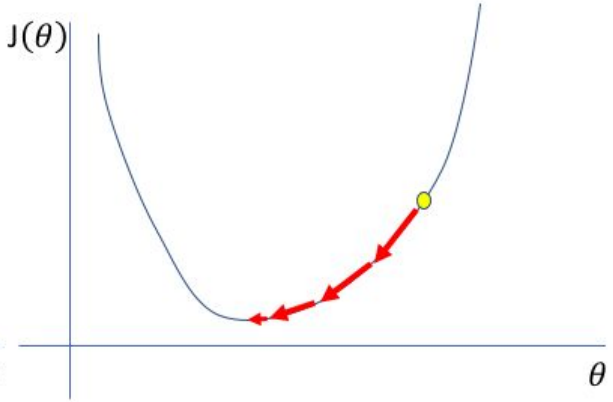
The importance of the learning rate

Too low



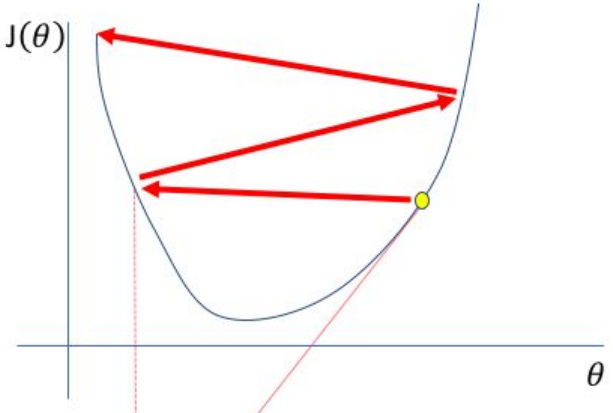
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high

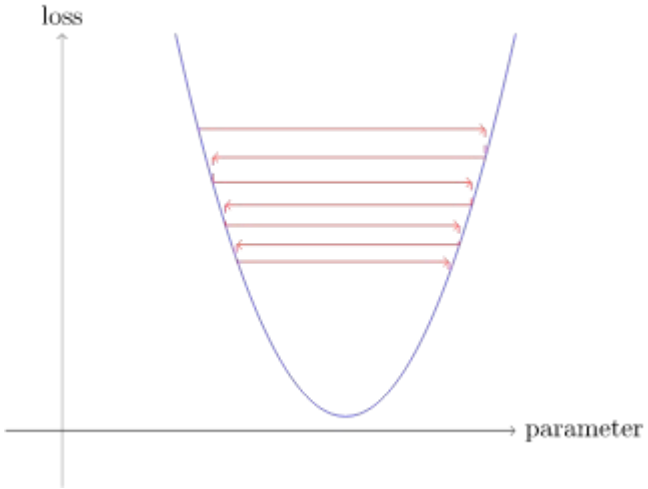


Too large of a learning rate causes drastic updates which lead to divergent behaviors

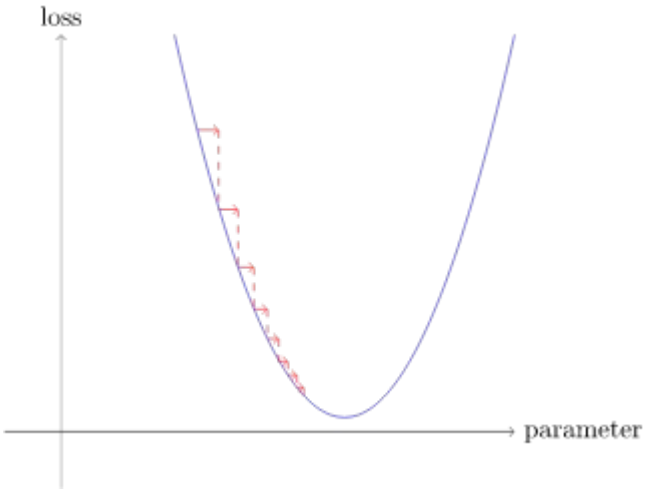
The importance of the learning rate

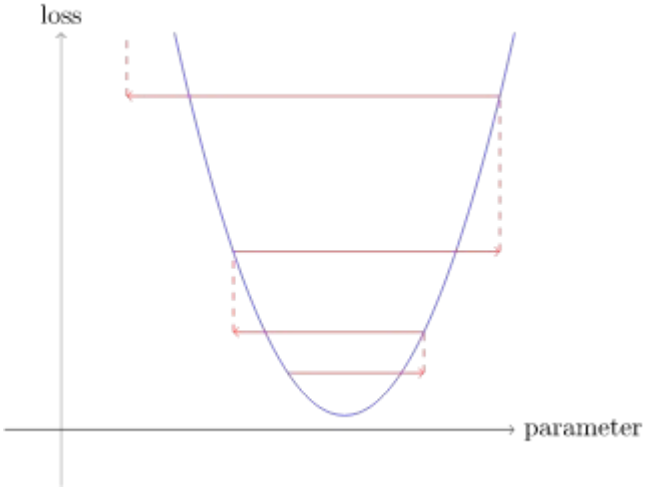
$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

Choosing a learning rate



Choosing a learning rate





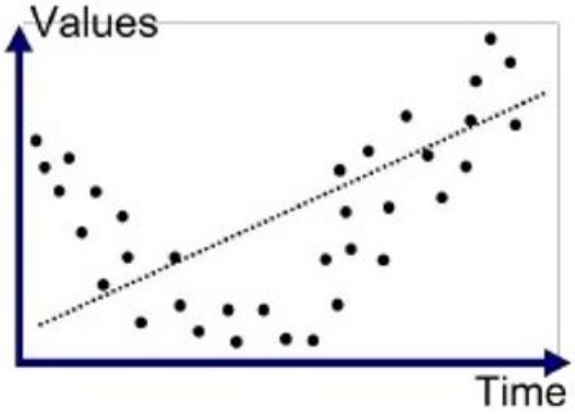
Weight decay aka lx-regularization

L1 ERROR FUNCTION $= -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i)+y_i ln(\hat{y}_i)+\lambda(|w_1|+...+|w_n|)$

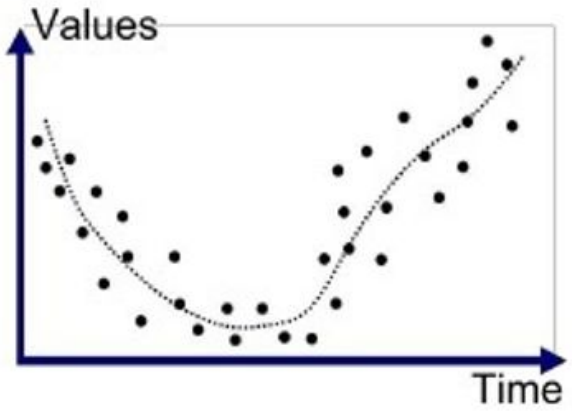
L2 ERROR FUNCTION $= -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i)+y_i ln(\hat{y}_i)+\lambda({w_1}^2+...+{w_n}^2)$

L1 and L2 regularization

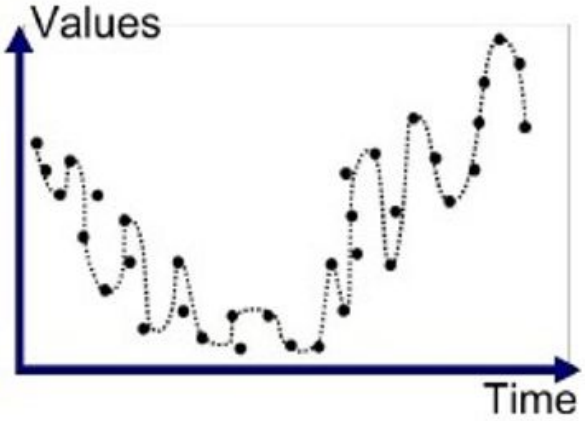
The bias vs variance tradeoff



Underfitted

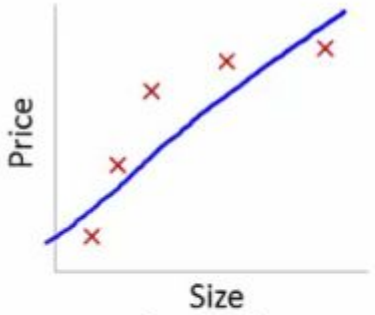


Good Fit/Robust



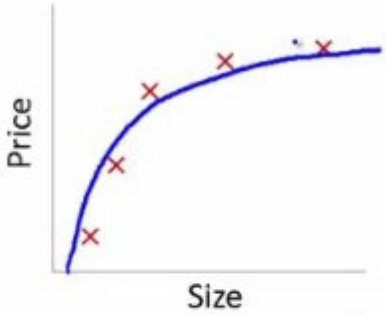
Overfitted

The bias vs variance tradeoff



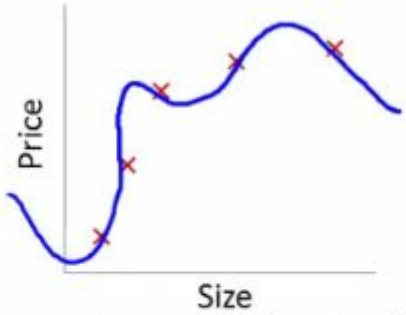
$\theta_0 + \theta_1 x$

High bias
(underfit)



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

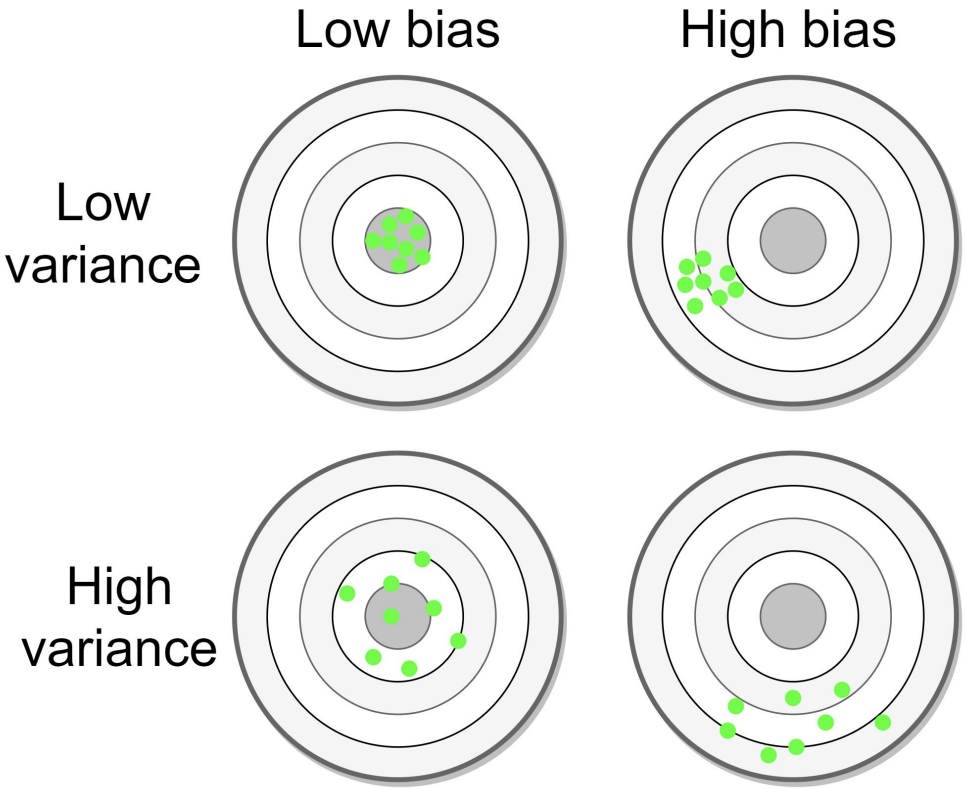
"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

The bias vs variance tradeoff



Practice: neural networks in PyTorch

- Notebook
- Solutions

Questions

- How do we subclass nn.Module?
    - What is a subclass?
- How many layers do we need to learn non-linear mappings?
- What is the purpose of a dataloader?
- What is nn.Linear?
- What is the purpose of nn.Sequential?
- Why would we use OrderedDict to define an architecture?

The Softmax function

The softmax function takes as input a vector *z* of *K* real numbers, and normalizes it into a probability distribution consisting of *K* probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval $(0, 1)$, and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities.

The standard (unit) softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

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

The softmax activation function in Python

Practice: training in PyTorch

- Notebook
- Solutions

Questions:

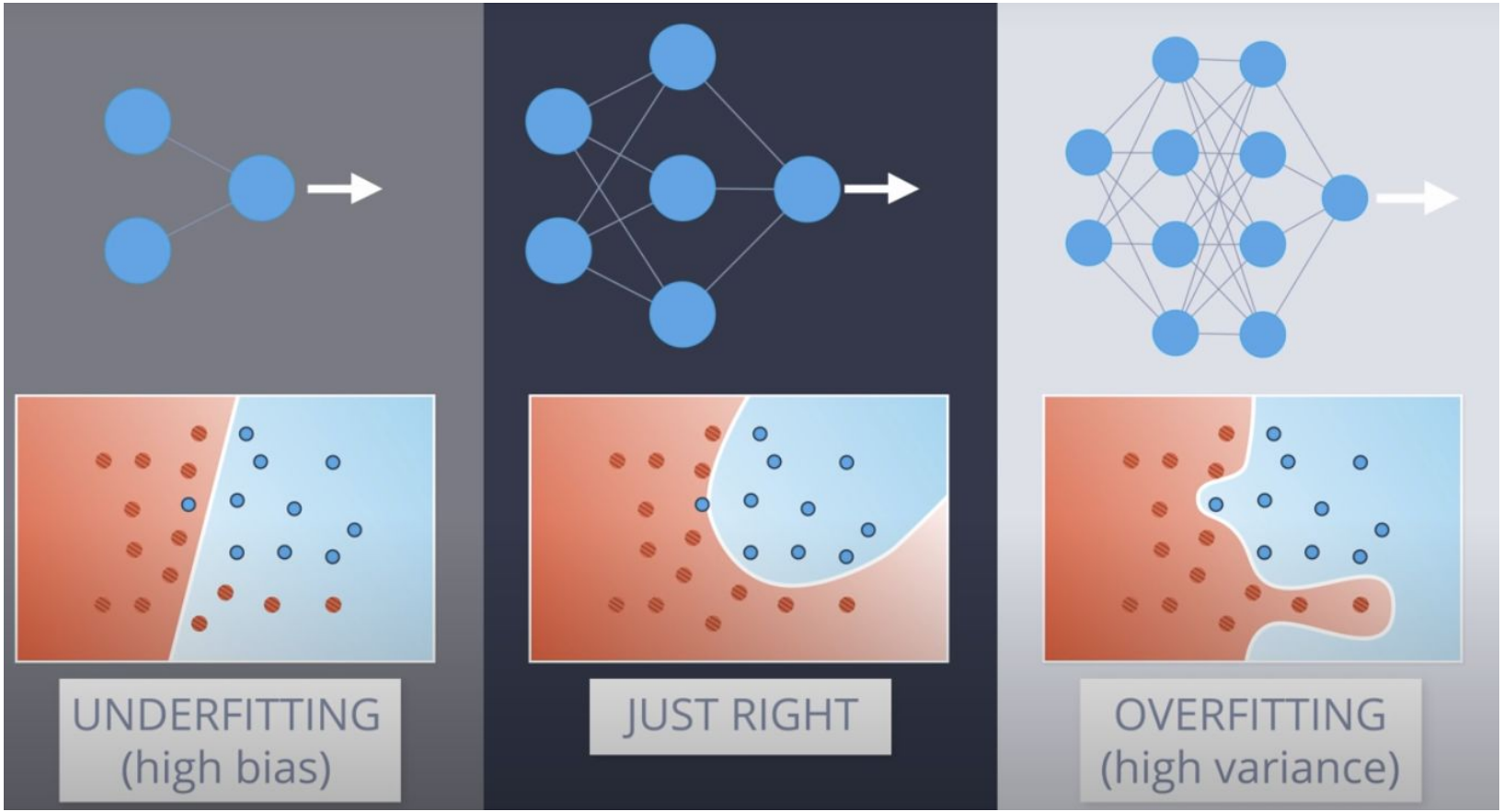
- What is a logit (in the context of neural networks)?
- How do we turn logits into probabilities?
- What criterion should we use if we change nn.LogSoftmax to Softmax? (Read this discussion)
- Where do we define the size of the batch?

Performance metrics vs loss functions

A model performance metric measures the quality of predictions in the validation or test sets. Accuracy, precision, recall, specificity, fbeta, f1, and ROC are performance metrics, not loss functions

The default classification error metric in fastai is error_rate = 1 - accuracy

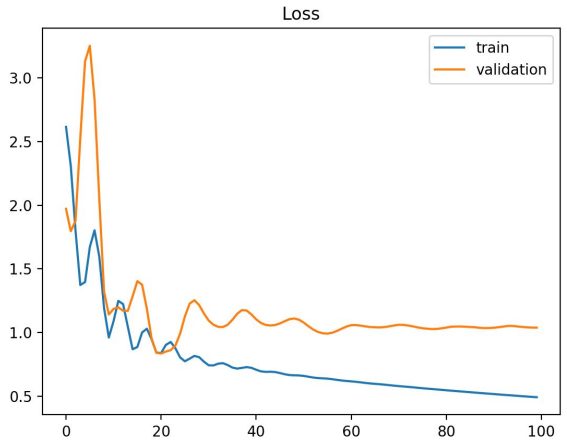
The loss function is used to adjust the weights of the model (through its gradient). The error metric is a report of performance and it can be opposite to the loss function

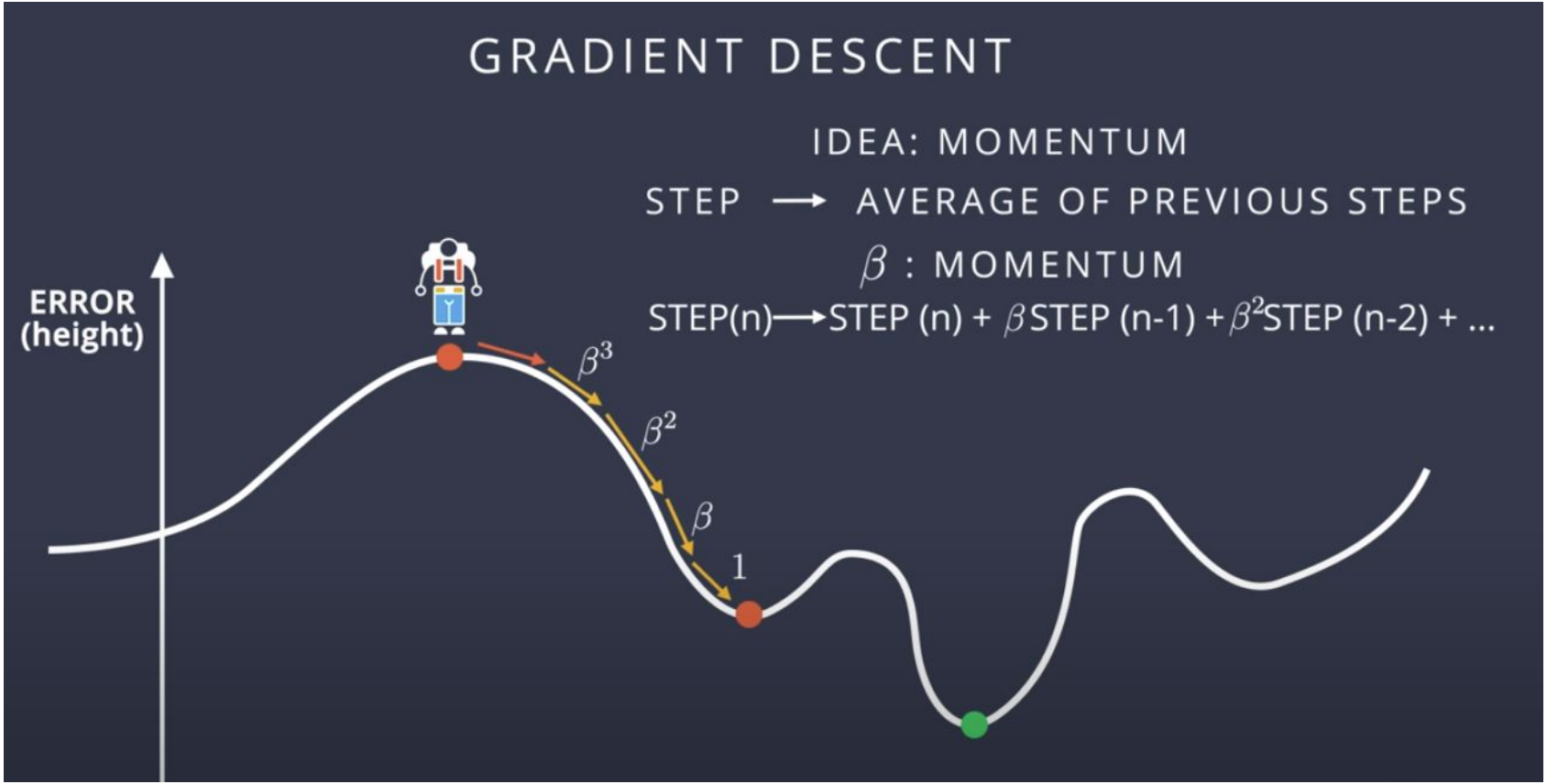


Understanding the bias vs variance tradeoff in neural network architectures

How long should we train the network?

- Until it overfits! (Training loss goes down while validation loss goes up)
- This general principle applies to **all** hyperparameters related to model complexity





The intuition of momentum

Exploring momentum's beta (notebook)

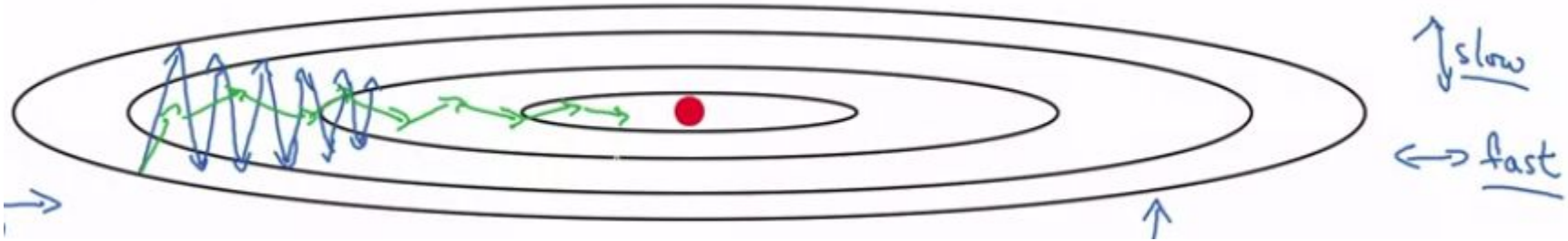
Understanding momentum

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor

weight increment, previous iteration

The ADAM optimizer



Exponentially moving averages and used to smoothen the trajectory of the gradient descent. Averages are computed across training batches. This optimization algorithm usually trains faster than SGD

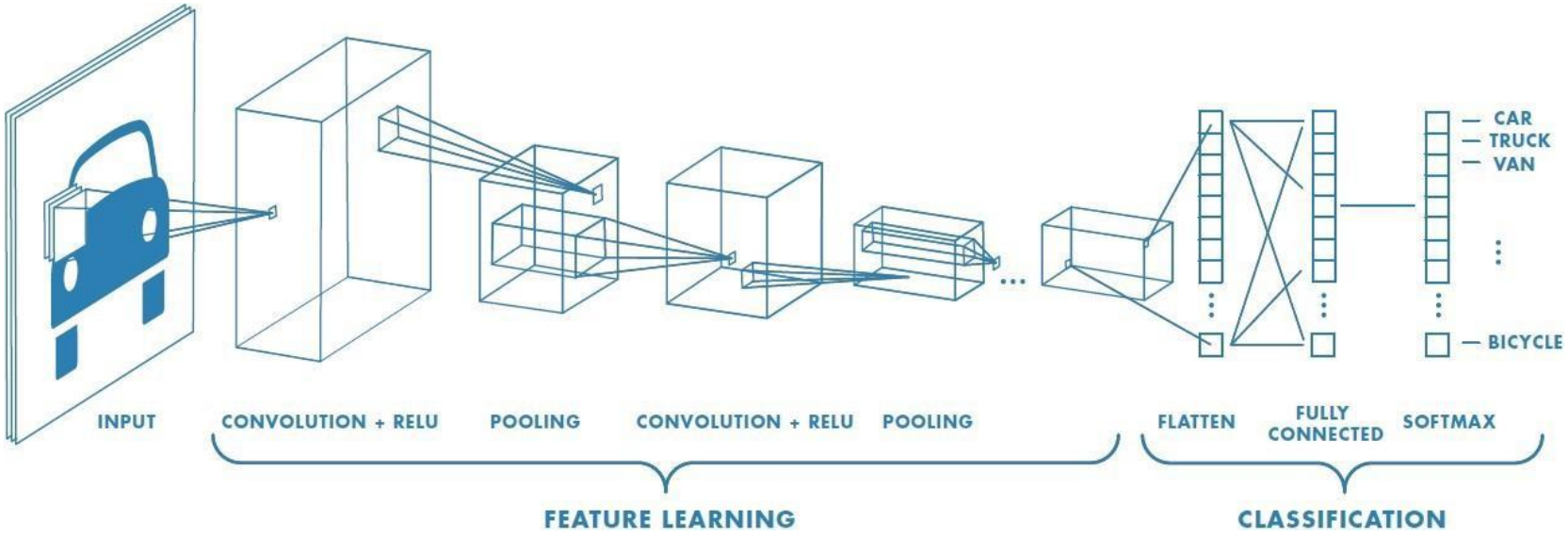
Practice: classifying images with fully connected networks

- Notebook
- Solutions

Questions:

- How is the image fed into the network?
- Why shouldn't we use the network's accuracy as a Pytorch **criterion**?
- What is the optimizer?
- How does ADAM differ from Stochastic Gradient Descent? How are they similar?
- What is the **lr** parameter that we pass to the optimizer?
- Why do we pass model.parameters() to the optimizer?

Convolutional Neural Networks



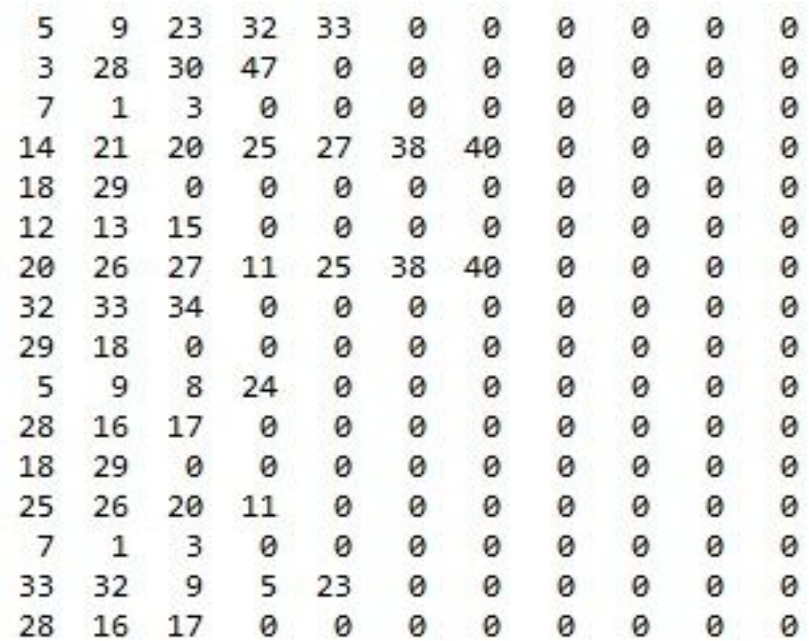
[CNNs vs multilayer perceptrons]

Image Classification

what we see

- Outdoor
- Building
- Snow

Image Classification



5	9	23	32	33	0	0	0	0	0	0
3	28	30	47	0	0	0	0	0	0	0
7	1	3	0	0	0	0	0	0	0	0
14	21	20	25	27	38	40	0	0	0	0
18	29	0	0	0	0	0	0	0	0	0
12	13	15	0	0	0	0	0	0	0	0
20	26	27	11	25	38	40	0	0	0	0
32	33	34	0	0	0	0	0	0	0	0
29	18	0	0	0	0	0	0	0	0	0
5	9	8	24	0	0	0	0	0	0	0
28	16	17	0	0	0	0	0	0	0	0
18	29	0	0	0	0	0	0	0	0	0
25	26	20	11	0	0	0	0	0	0	0
7	1	3	0	0	0	0	0	0	0	0
33	32	9	5	23	0	0	0	0	0	0
28	16	17	0	0	0	0	0	0	0	0

what the computer “sees”

- Outdoor
- Building
- Snow? *Do we care about snow?*
  - Enough of these images need to be shown to the algorithm

Defining image taxonomies

- Bedroom
- Terrace
- Desk
- Vegetation

- Do we have enough images with these things?

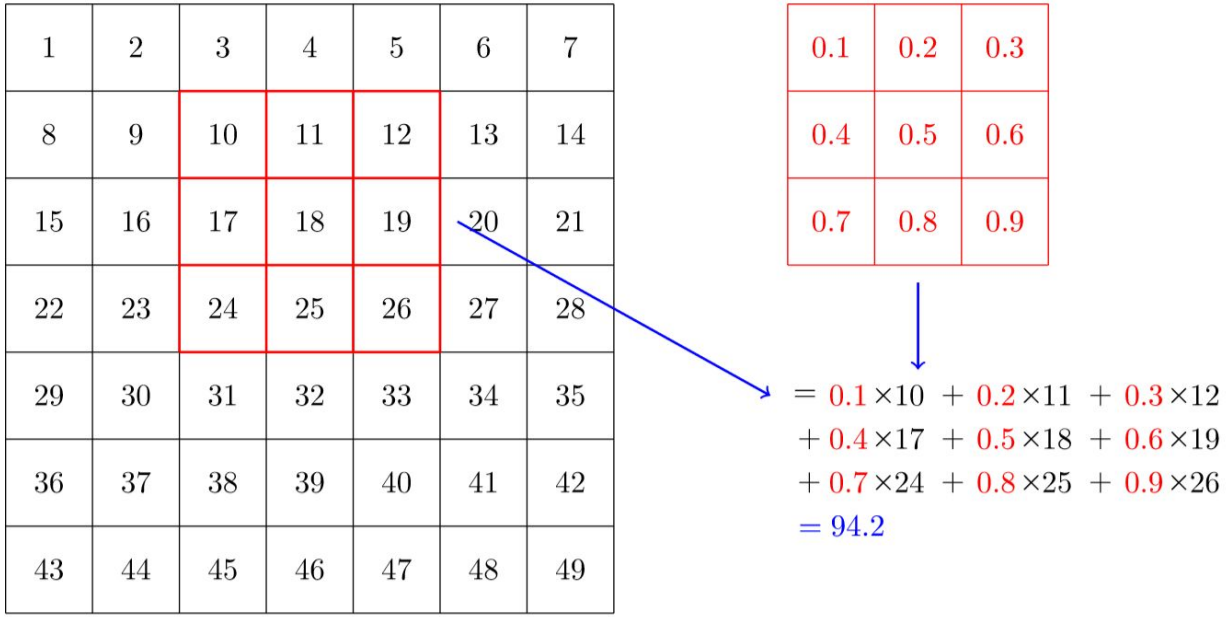
Labels for hard cases

- Should we have added ‘neon lights’ to our taxonomy?
- How many of these things we have?
- Should we invest effort on this?
- **Stacks of binary models** are appropriate for this problem

Hot Dog not a Hot Dog

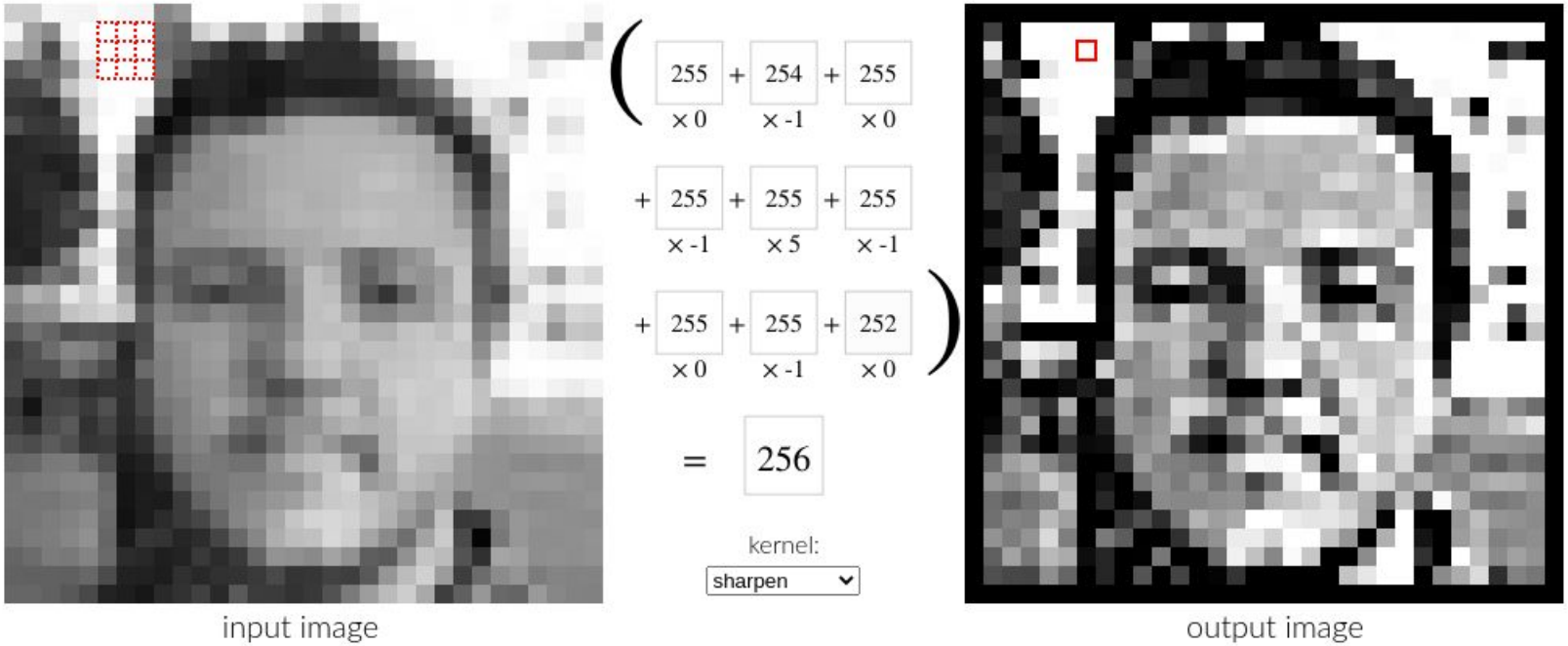
https://www.youtube.com/watch?v=vIci3C4JkL0

What is a convolution filter?



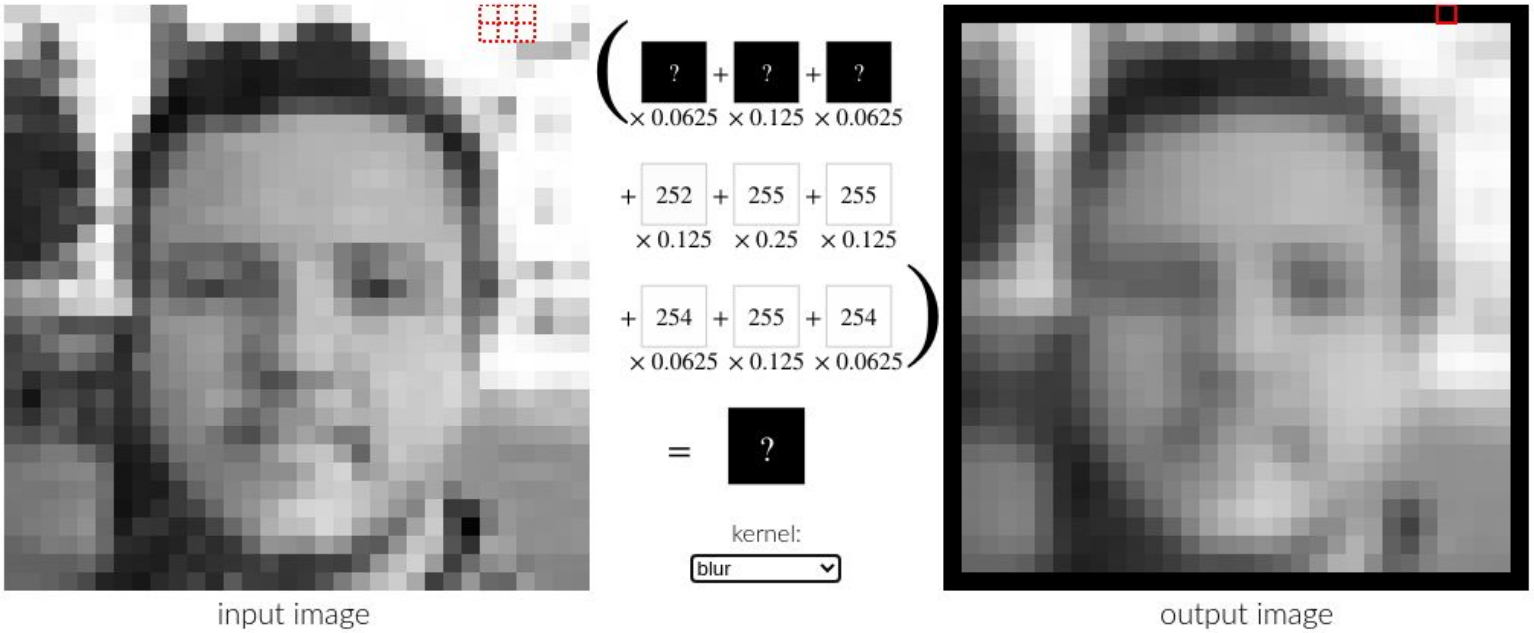
https://github.com/fastai/fastbook/blob/master/13_convolutions.ipynb

What is a convolution filter?



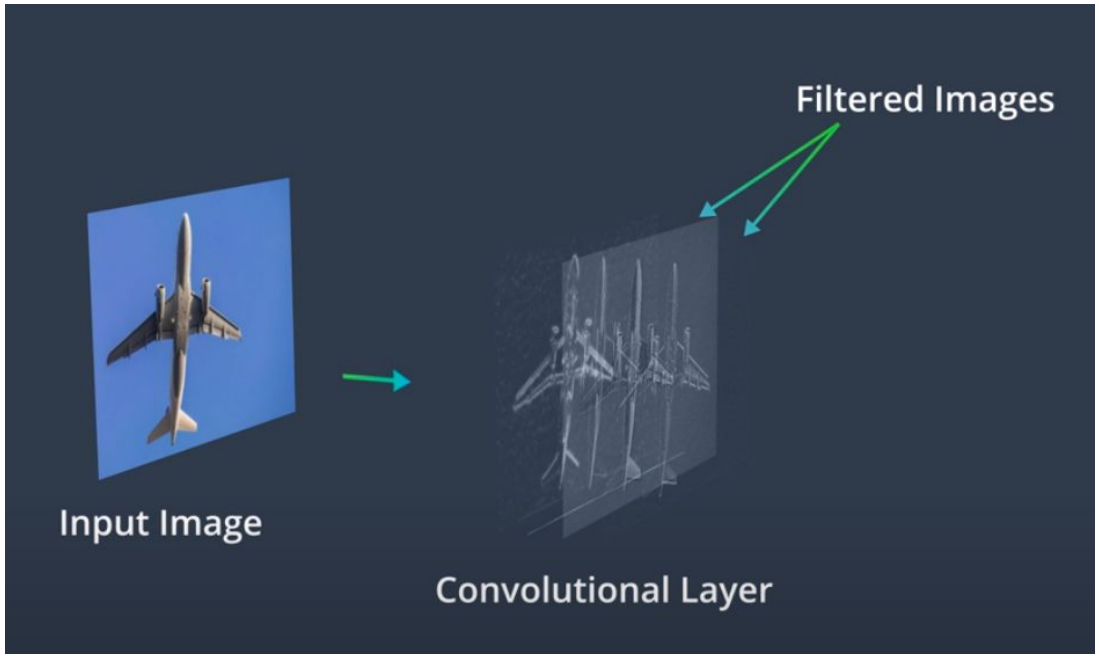
https://setosa.io/ev/image-kernels/

What is a convolution filter?



https://setosa.io/ev/image-kernels/

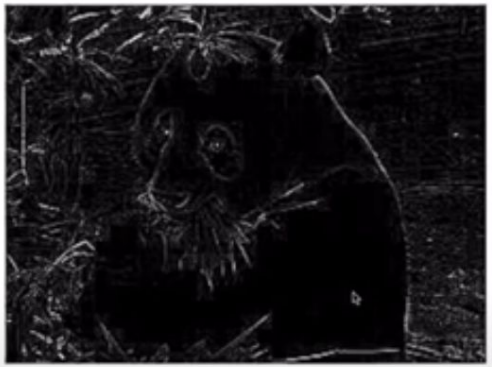
What is a convolution filter?



Understanding the output of convolutional layers

What is a convolution filter?

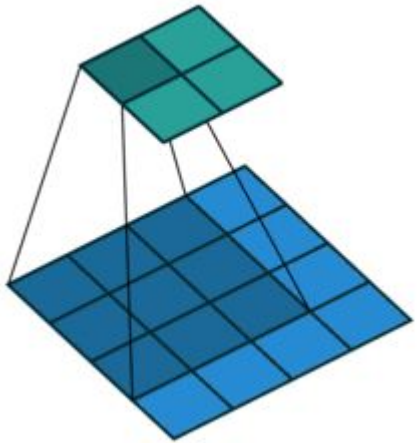
0	-1	0
-1	4	**-2**
0	-1	0



$$0 + -1 + 0 + -1 + 4 + \mathbf{-2} + 0 + -1 + 0 = \mathbf{-1}$$

Visualizing convolution kernels

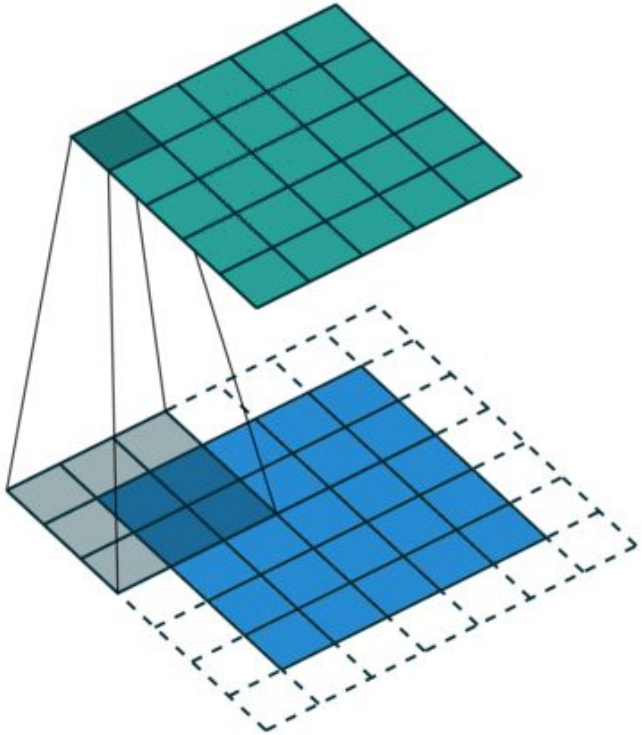
What is a convolution filter?



Convolution of 3x3 and stride = 1 without padding

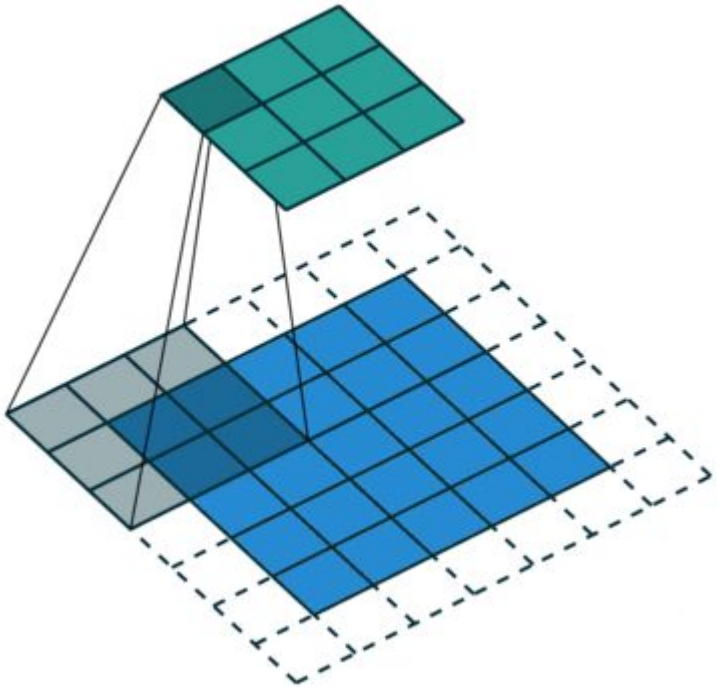
Effect: the output loses one pixel on each dimension

What is a convolution filter?



Convolution of 3x3 and stride = 1 padded with zeros
Effect: the output preserves original image size

What is a convolution filter?



Convolution of 3x3 and stride = 2 padded with zeros

Effect: the output is downsampled to about half its size

Activity: Conv2D in Pytorch (Colab notebook)

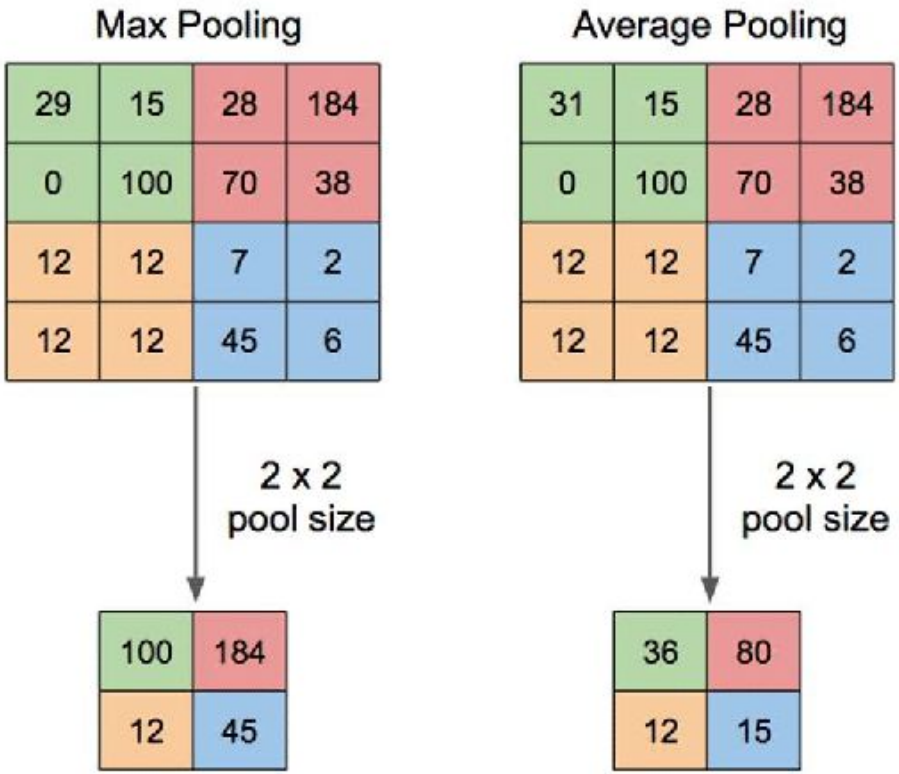
Experiment: finding edges with convolutions

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

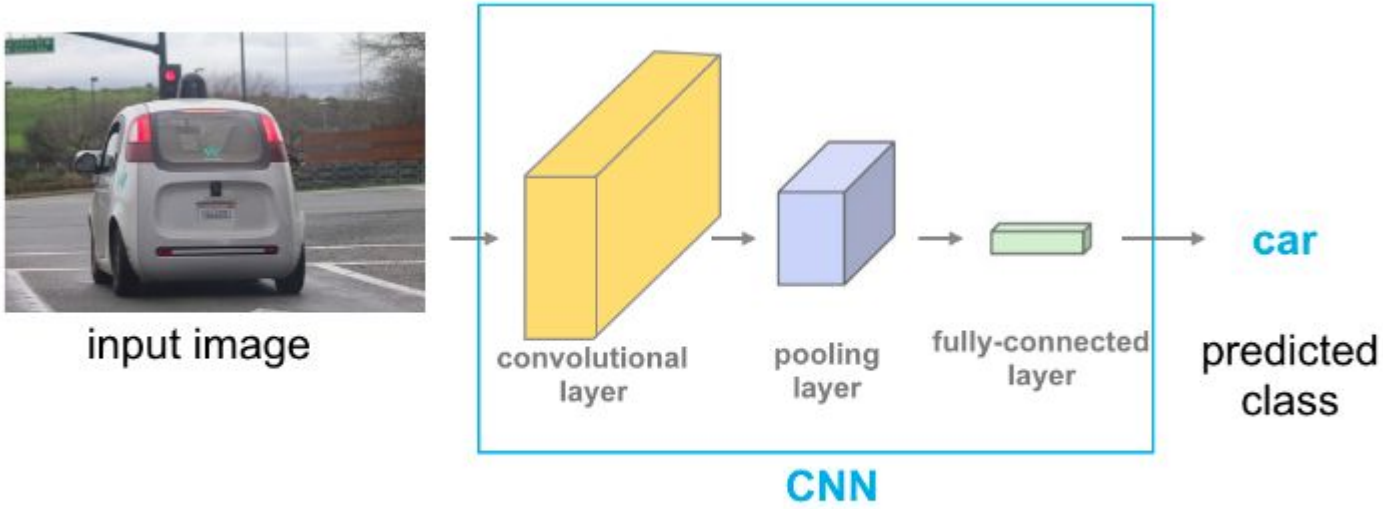
$$S_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Finding edges with convolution kernels

Pooling

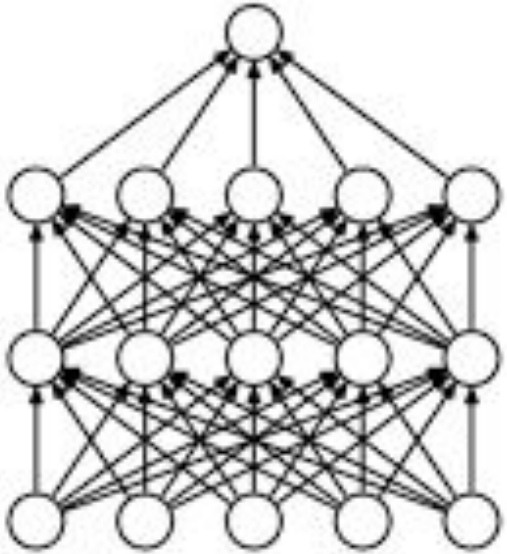

Max Pooling
29 15 28 184
0 100 70 38
12 12 7 2
12 12 45 6
2 x 2
pool size
100 184
12 45
Average Pooling
31 15 28 184
0 100 70 38
12 12 7 2
12 12 45 6
2 x 2
pool size
36 80
12 15

CNN architectures

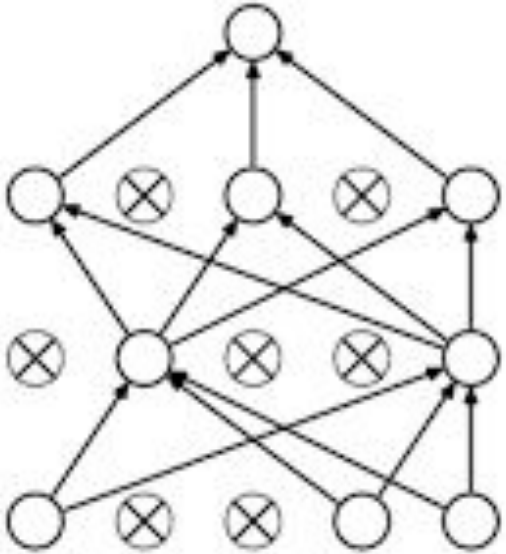


Layers in a CNN.

Dropout regularization



(a) Standard Neural Net

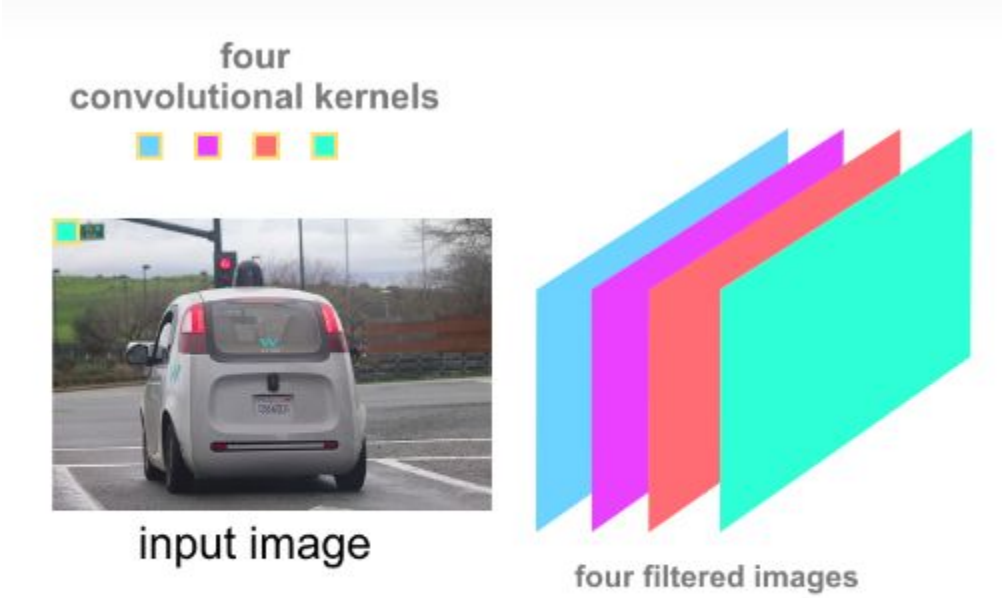


(b) After applying dropout.

Practice: Inference and validation

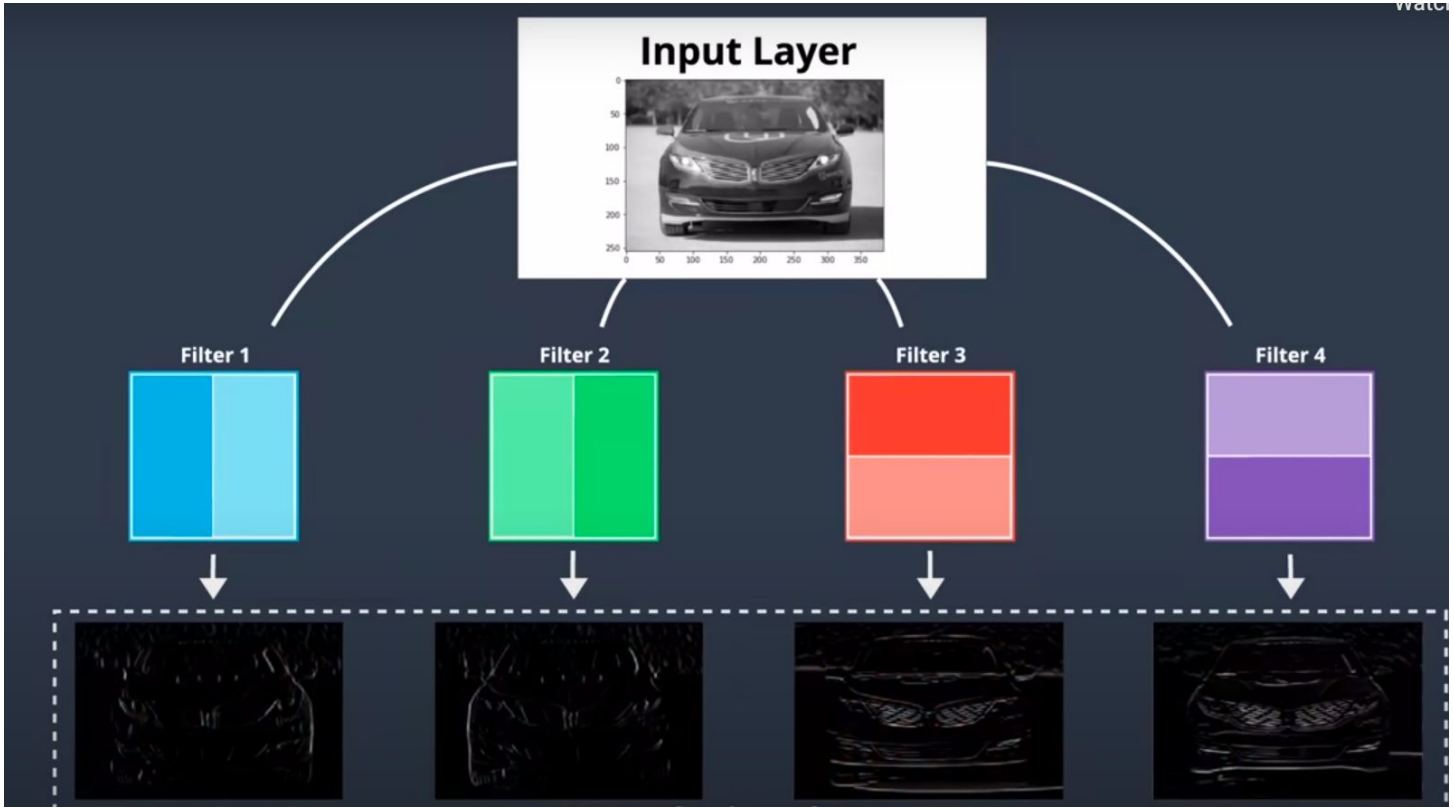
- Notebook
- Solutions
- Saving models notebook
- What are the differences between a validation set and a test set? Could you change the notebook use sklearn's train_test_split to create new validation and test sets?
- Does increasing the amount of dropout increase the bias or the variance of the network?
- If the training loss is lower than the validation loss is the network overfitted or underfitted?
- Why do we want to use **torch.nograd()** when evaluating the network?
- Experiment: try adding a 'save model' functionality to P5, implement patience

CNN architectures

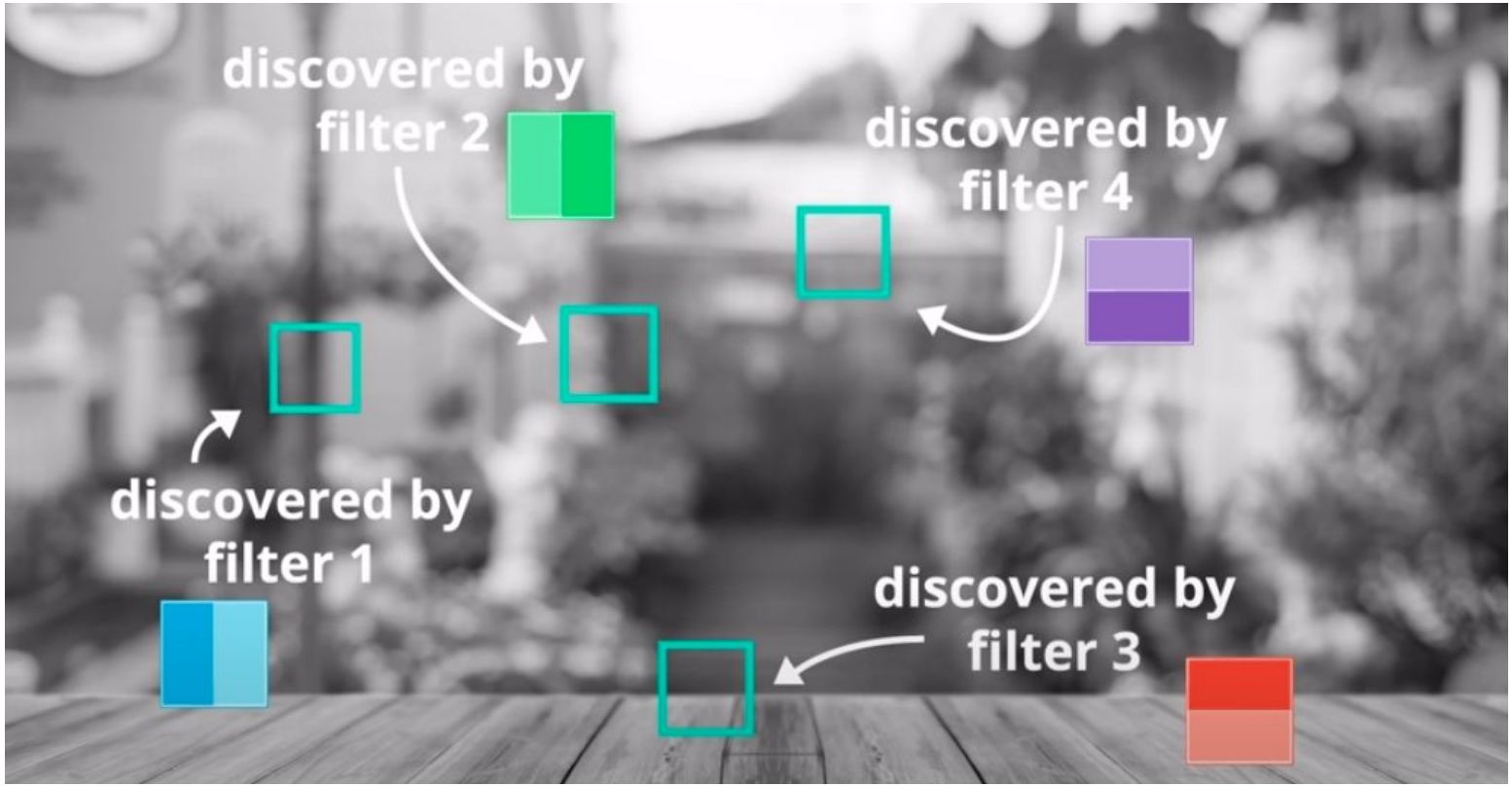


4 kernels = 4 filtered images.

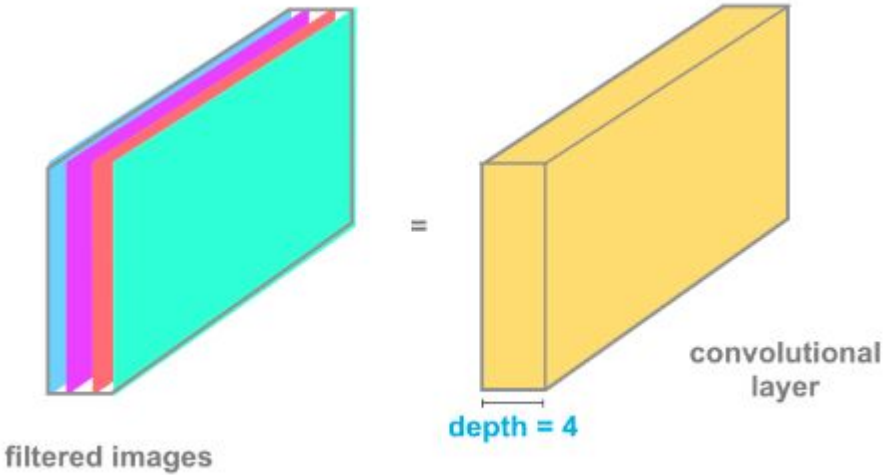
CNN architectures



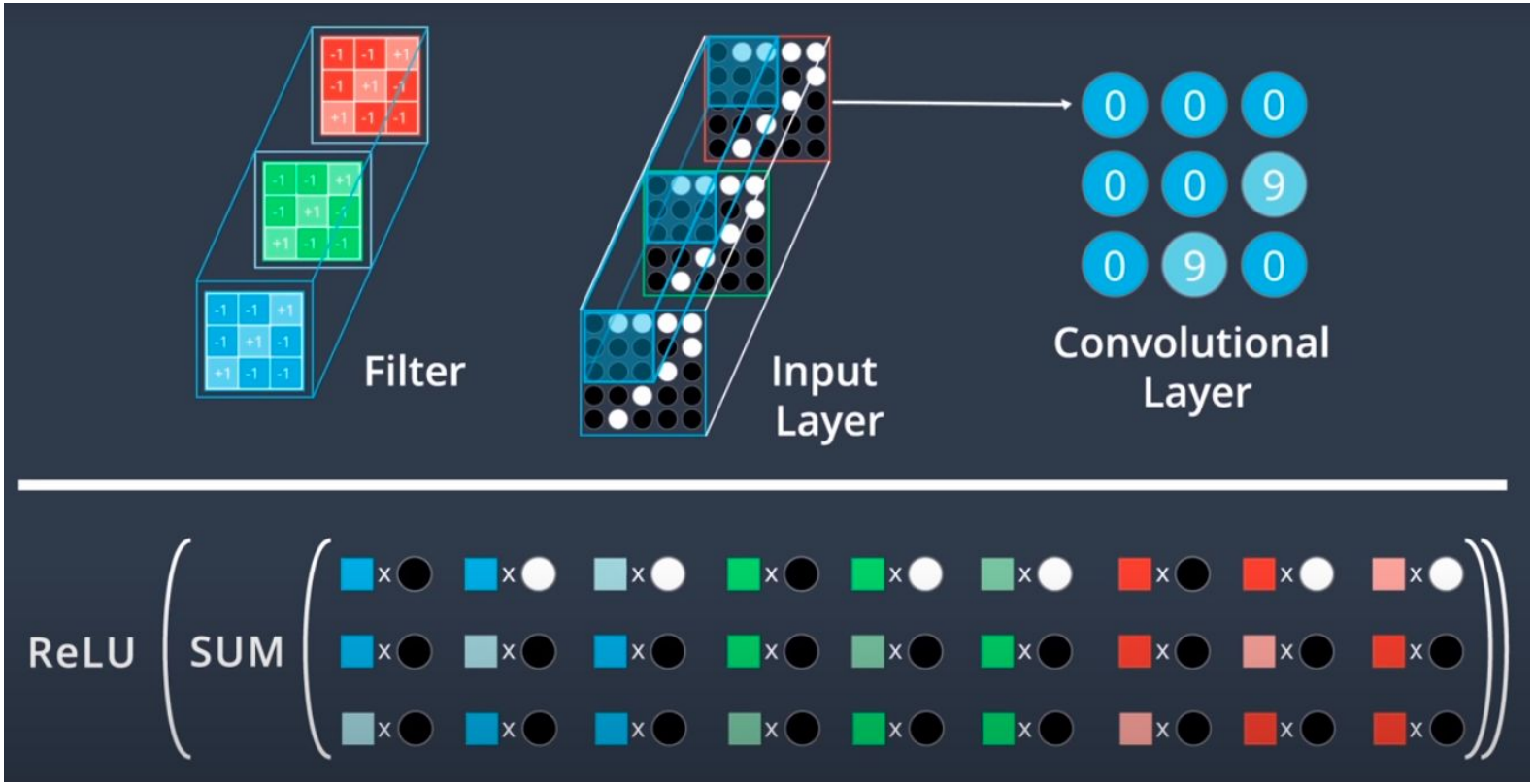
CNN architectures

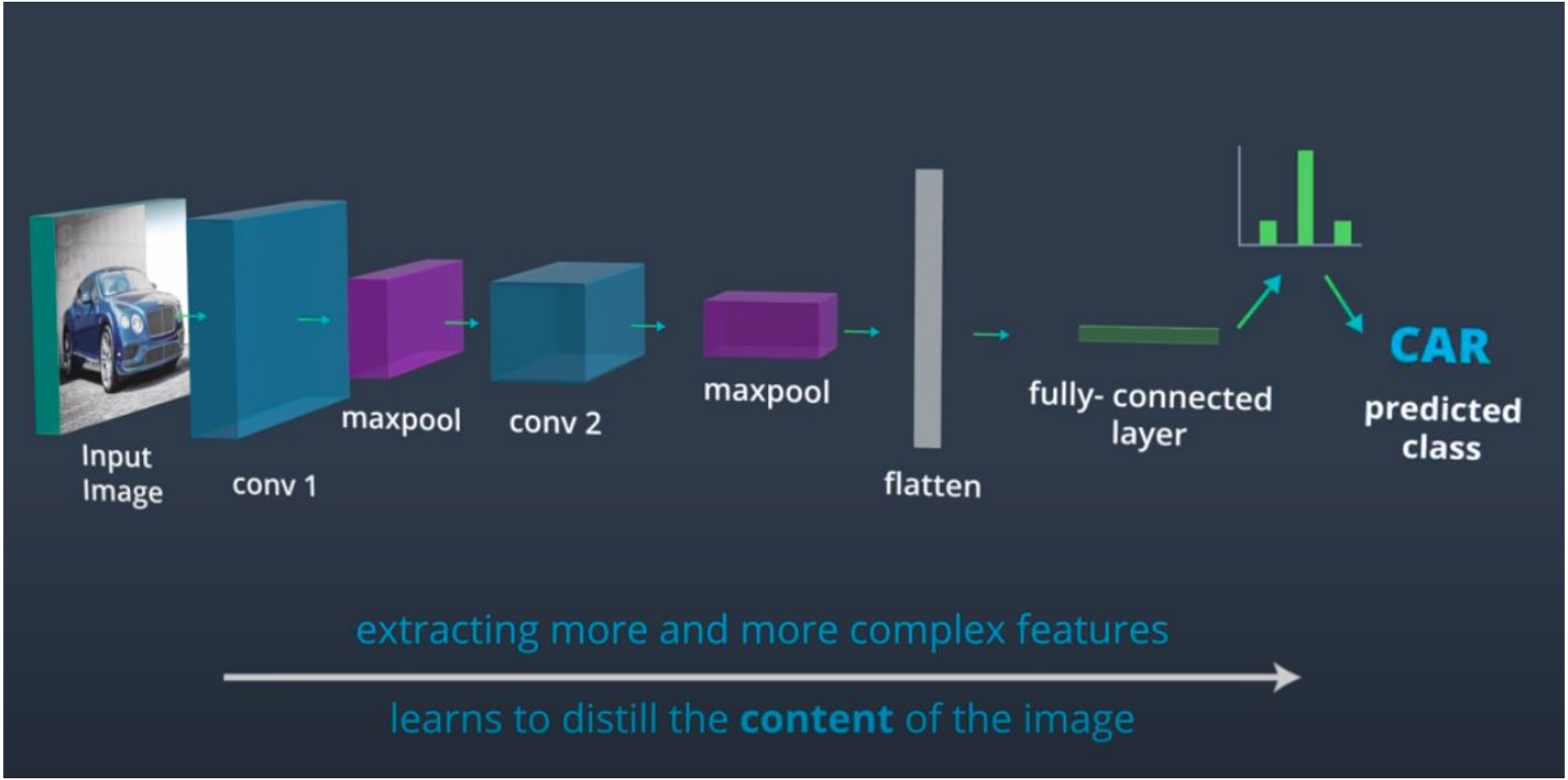


CNN architectures



CNN architectures




Input Image
conv 1
maxpool
conv 2
maxpool
flatten
fully- connected layer
CAR
predicted class
extracting more and more complex features
learns to distill the **content** of the image

Normalizing images

```
transforms = torch.nn.Sequential(
    transforms.CenterCrop(10),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
)
scripted_transforms = torch.jit.script(transforms)
```

https://pytorch.org/docs/stable/torchvision/transforms.html#transforms-on-torch-tensor

Images need to be resized for the GPU

zeros

border

reflection

https://docs.fast.ai/vision.augment

Data augmentation

https://docs.fast.ai/vision.augment

Augmentations composition in PyTorch

```
train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                       transforms.RandomResizedCrop(100),
                                       transforms.RandomHorizontalFlip(),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.5, 0.5, 0.5],
                                                            [0.5, 0.5, 0.5])])
```

https://sparrow.dev/torchvision-transforms/

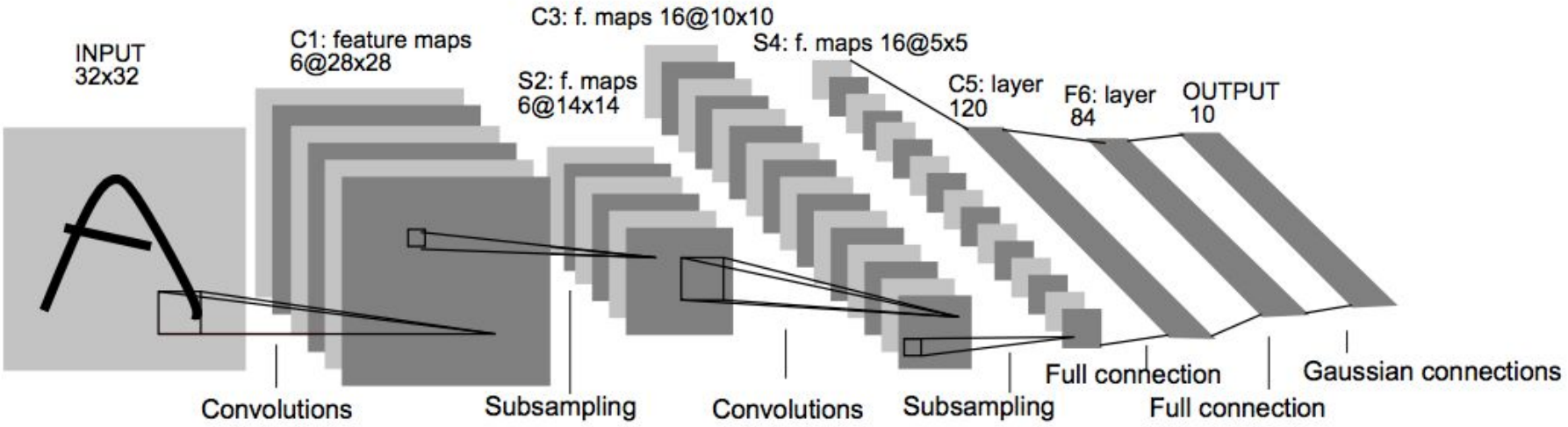
Practice: loading image data

- Notebook
- Solutions

Questions:

- Why are we using different means and standard deviations to normalize each image channel?
- How are we changing the dimensions of the input images before feeding them to the network?
- What is the effect of calling Resize before CenterCrop?

Understanding architecture diagrams: LeNet



Understanding architecture diagrams: AlexNet

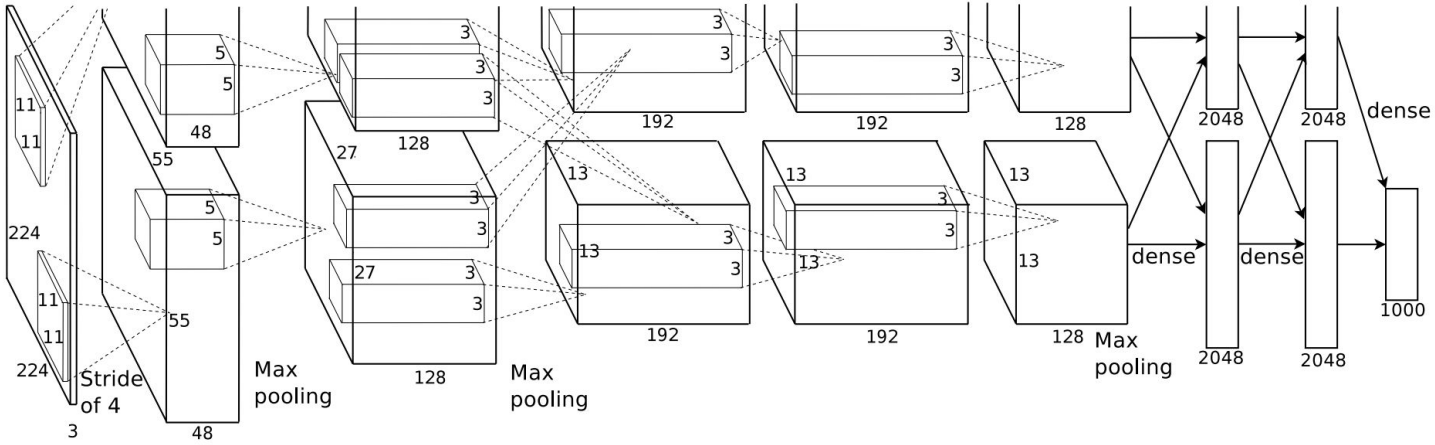


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Transfer learning

Using a pretrained model for a task different to what it was originally trained on is called *transfer learning*

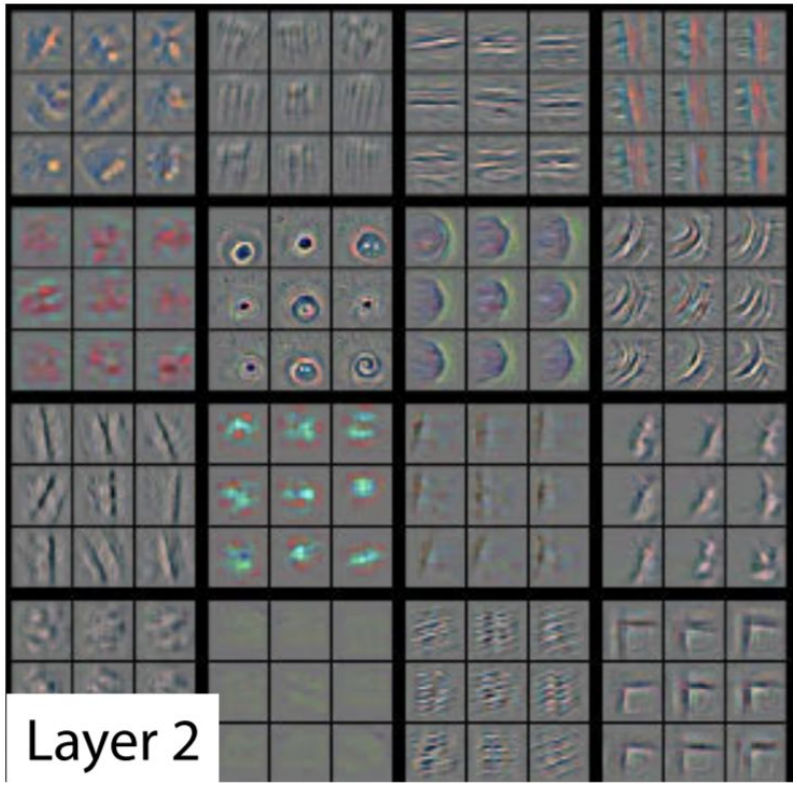
ImageNet is the single most important dataset in computer vision, it contains 1.3 million labeled images of a thousand categories and has been used to benchmark classification models during the last decade. Most popular models have been pretrained and published on ImageNet

What a pretrained network already knows

https://arxiv.org/pdf/1311.2901.pdf

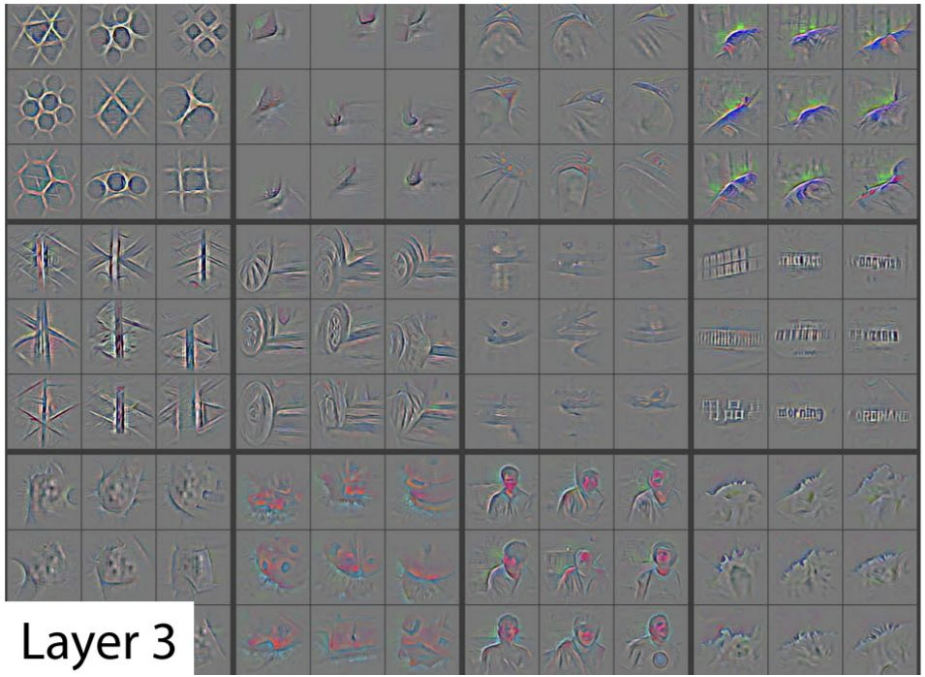
What a pretrained network already knows

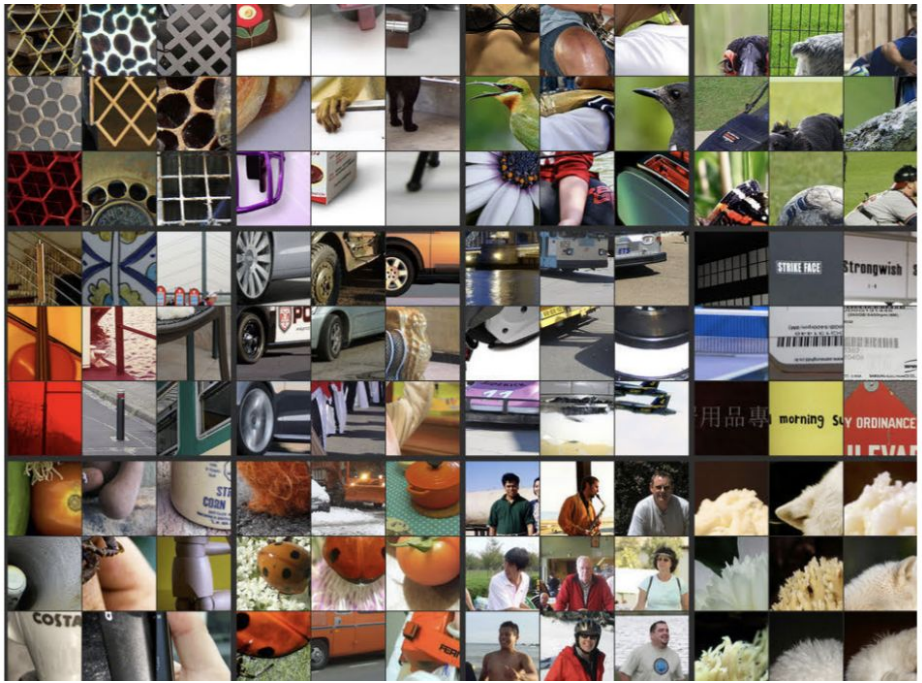
Layer 1



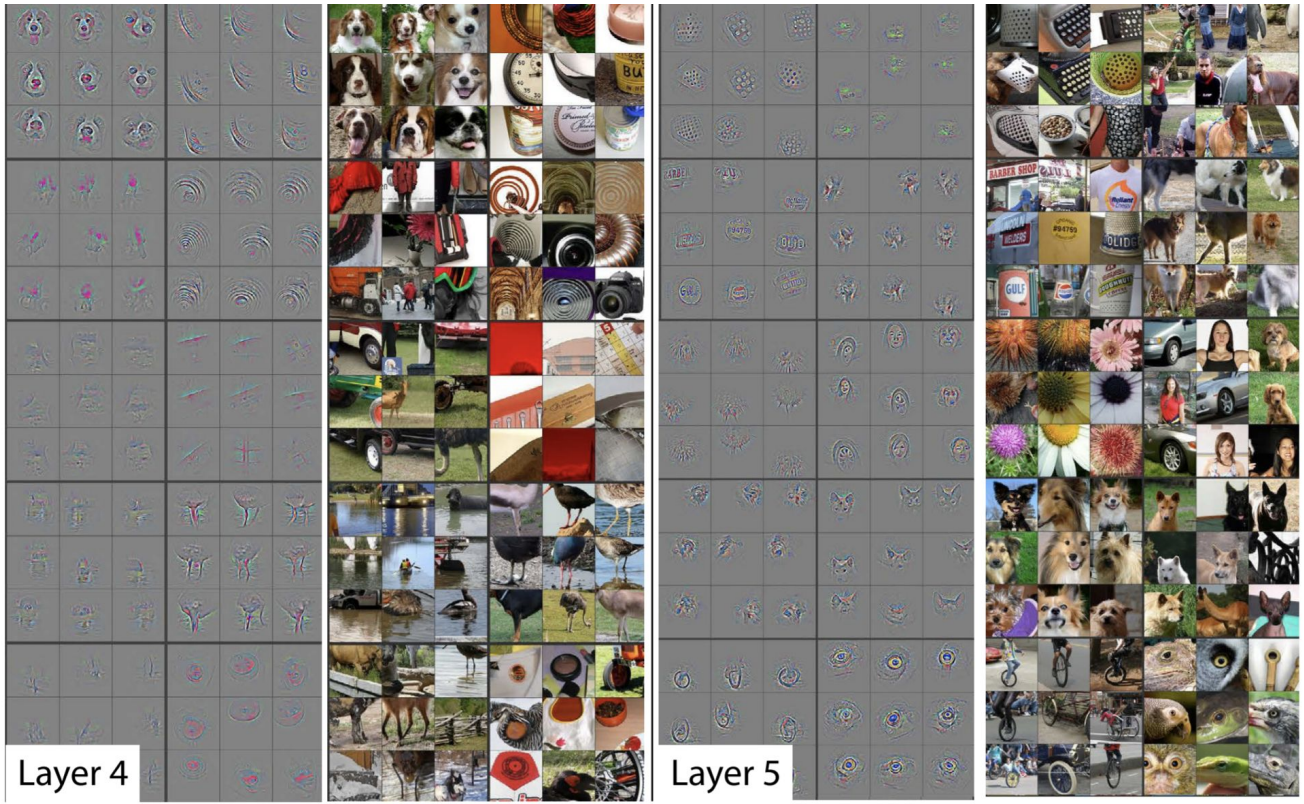
Layer 2

What a pretrained network already knows

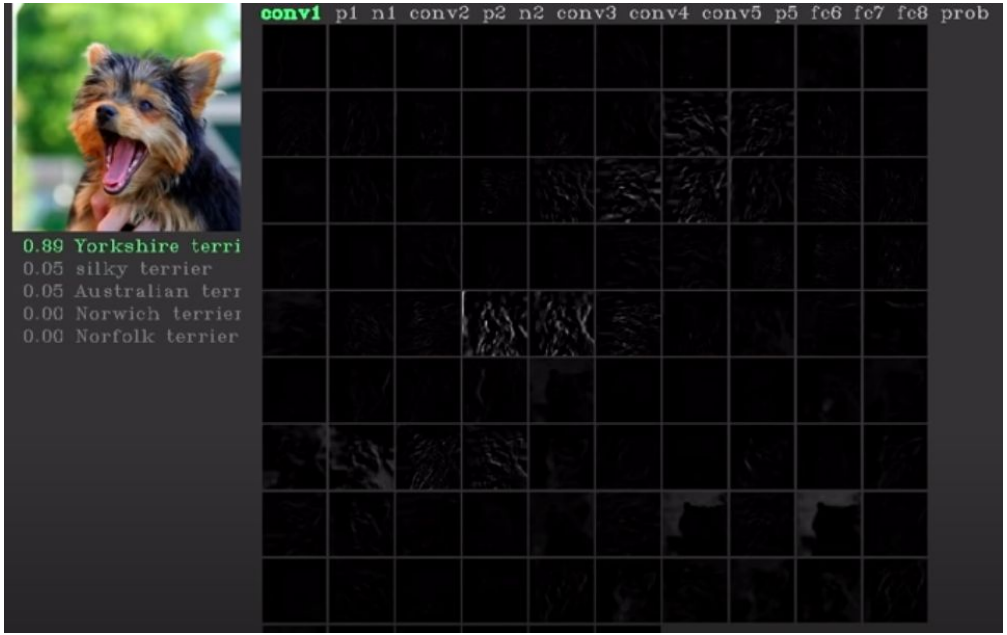




What a pretrained network already knows

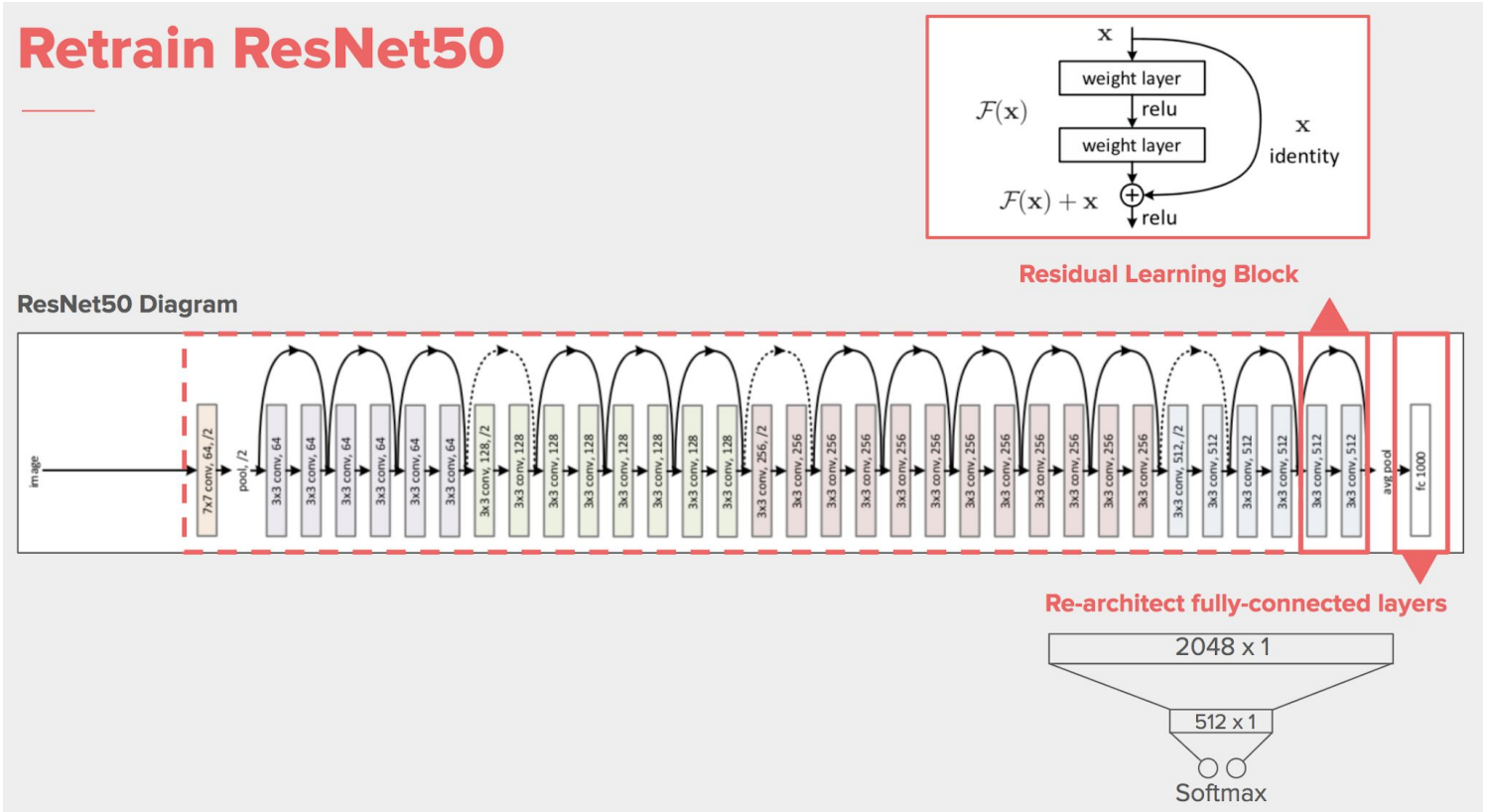


What a pretrained network already knows



[Visualizing the layers of an Alexnet trained on Imagenet]

Resnet


Retrain ResNet50
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
Residual Learning Block
ResNet50 Diagram
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
avg pool
fc 1000
Re-architect fully-connected layers
2048 x 1
512 x 1
Softmax

Densenet

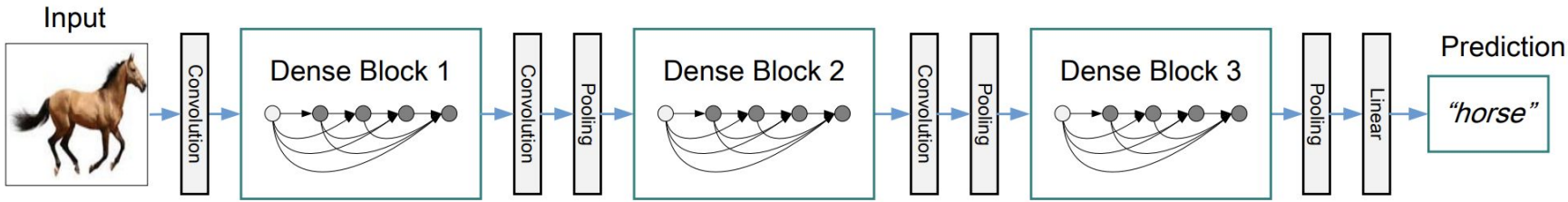
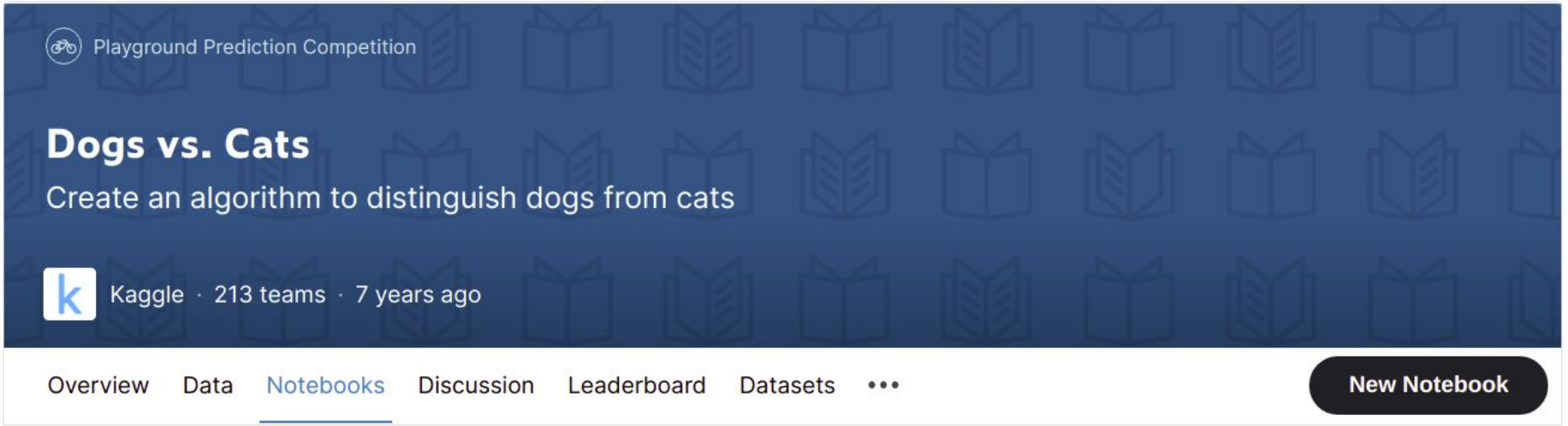


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Building a dogs vs cats image classifier



https://www.kaggle.com/c/dogs-vs-cats/

Practice: transfer learning

- Notebook
- Solution with Resnet
- Solution with Densenet

- Why do we need to explore the model architecture before defining our own final layers?

Review questions

- What is a loss function? Is the f1 measure a loss function or a performance metric?
- Can we use accuracy as a loss function? Can we use precision or recall as loss functions?
- Is the accuracy guaranteed to increase as the value of the loss function goes down?
- Which parts of the model need to be differentiable for an architecture to be trainable?
- What are we trying to minimize when using gradient descent? What is the learning rate?
- What is a GPU? Why are GPUs useful for deep learning?

Review questions

- What is an epoch? What is a batch? What is the maximum size of a batch? How does the batch size influence training?
- What is the difference between stochastic gradient descent (SGD) and ‘vanilla’ AKA ‘regular’ gradient descent?
- What is momentum? What is the difference between ADAM and SGD?
- What is a Jacobian? What is a Hessian? Do we use Hessians in deep learning?
- Suppose that we have a ground truth for a ‘cat’ label that is encoded as [0, 1]. We can assume that the ground truth for a ‘dog’ label is [1, 0] (**cat** is the training point, **y = cat**).
    - The network first outputs [0.75, 0.25] as prediction (after softmax)
    - In the next epoch it outputs [0.99, 0.01] as prediction (after softmax)
    - Has the cross entropy loss increased or decreased?

Review questions

- What does it mean for a dataset to be ‘linearly separable’?
- How many layers does a neural network need in order to learn non-linear correlations?
- What’s the difference between a performance metric and a loss function?
- Why should we shuffle the data in every training epoch?
- Explain the bias vs variance tradeoff.
  - Effect of increasing the number of training epochs
  - Increasing the number of layers
  - Increasing the amount of dropped out connections
- How would you define ‘regularization’?
- What is the purpose of dropout? How does it work? Should dropout be active during inference time?

Review questions

- What is the vanishing gradient problem?
- How does a Resnet avoid the vanishing gradient problem?
- What does it mean to ‘freeze’ a layer?
- How does batch normalization work?
- Why should we call model.eval() before evaluating data from the validation set?
- What does it mean to “transfer a tensor” to the GPU?
- What is CUDA?
- How does the amount of RAM available in the GPU affect the types of models and data that we can use to train?
- What should we try when torch reports ‘CUDA out of memory’ errors?

Review questions

- Why do we need to call `optimizer.zero_grad()` in the training loop?
- What's the difference between a parameter and a hyperparameter? What's another common name for 'parameter' in a neural network?
- How can we know what's the "optimal" number of units in a hidden layer?
- How can we know the "optimal" number of layers in a network?
- Is data augmentation a regularization technique? Why should we apply data augmentations probabilistically?
- Why should we shuffle the data inside a training batch?
- If a model is increasing its loss through several epochs should we increase or decrease the learning rate?
- How do we evaluate training and validation loss to do early stopping?

Review questions

- What is weight decay? What is learning rate decay?
- Suppose that after going through certain number of epochs the training loss is higher than the validation loss. Is this an example of underfitting or overfitting?
- What is the vanishing gradient problem? Why does it appear?
- Are vanishing gradients more likely to appear when representing Jacobians and weight vectors with 8 bits, 16 bits, or 32 bits of precision?
- Why are ReLU activation functions less likely to display vanishing gradients than sigmoid activation functions?
- What is a skip connection?
- What is the difference between a Resnet and a Densenet?

Review questions

- Why do we resize images before training?
- If we retrain or do inference on a network pre-trained on Imagenet, why do we need to normalize its image channels using the mean and std of the intensities of the original dataset?
- What is a receptive field?
- What is a convolution?
- Why do we apply padding to images before doing convolutions?
- When calling conv2d(...,..., 64) how many output channels are we producing? How are they different from each other?
- Why is the vanishing gradient a ‘numerical’ problem? What does that mean?
- Are we more likely to get vanishing gradients when using fp16 or fp32 data types for our weights?

Resources

- Intro to Deep Learning with PyTorch at Udacity
- Convolutional Neural Networks at Coursera
- Fast.ai - Deep Learning for Coders
- PyImageSearch Blog
- Stanford's CS231: Convolutional Neural Networks for Visual Recognition
- 3blue1brown's series on neural networks
- Goodfellow's Deep Learning Book (great for theory)
- Visualizing and understanding neural networks (Stanford lecture)
- Review content on cross entropy
  - On ML Mastery
  - Brief video by Udacity
  - Aurelien Geron's explanation