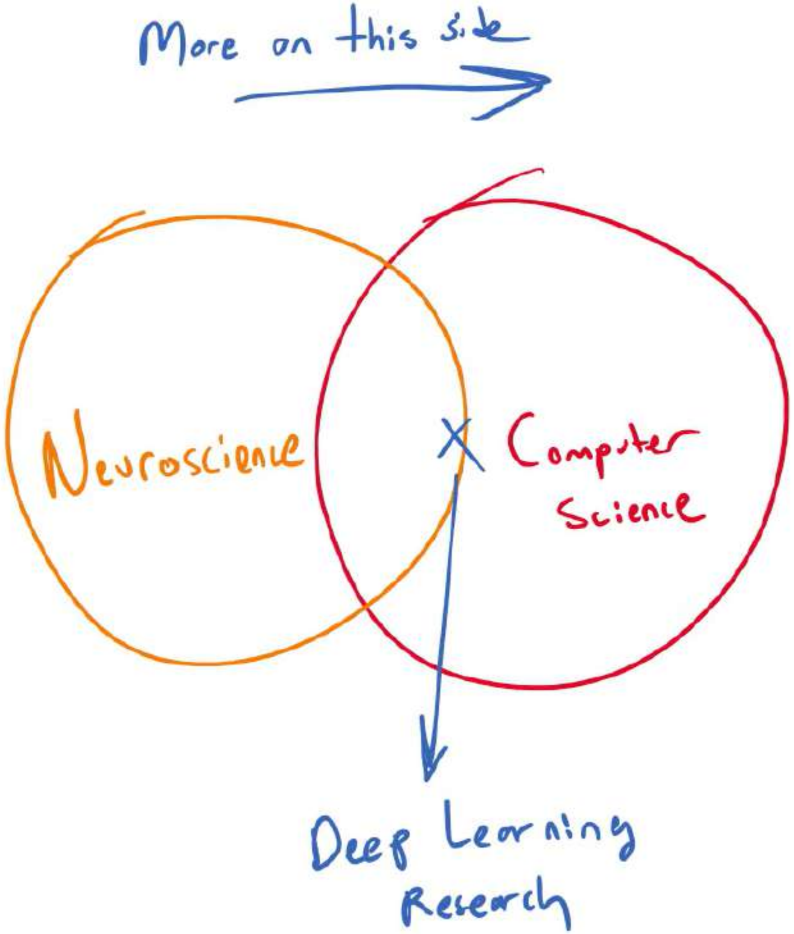
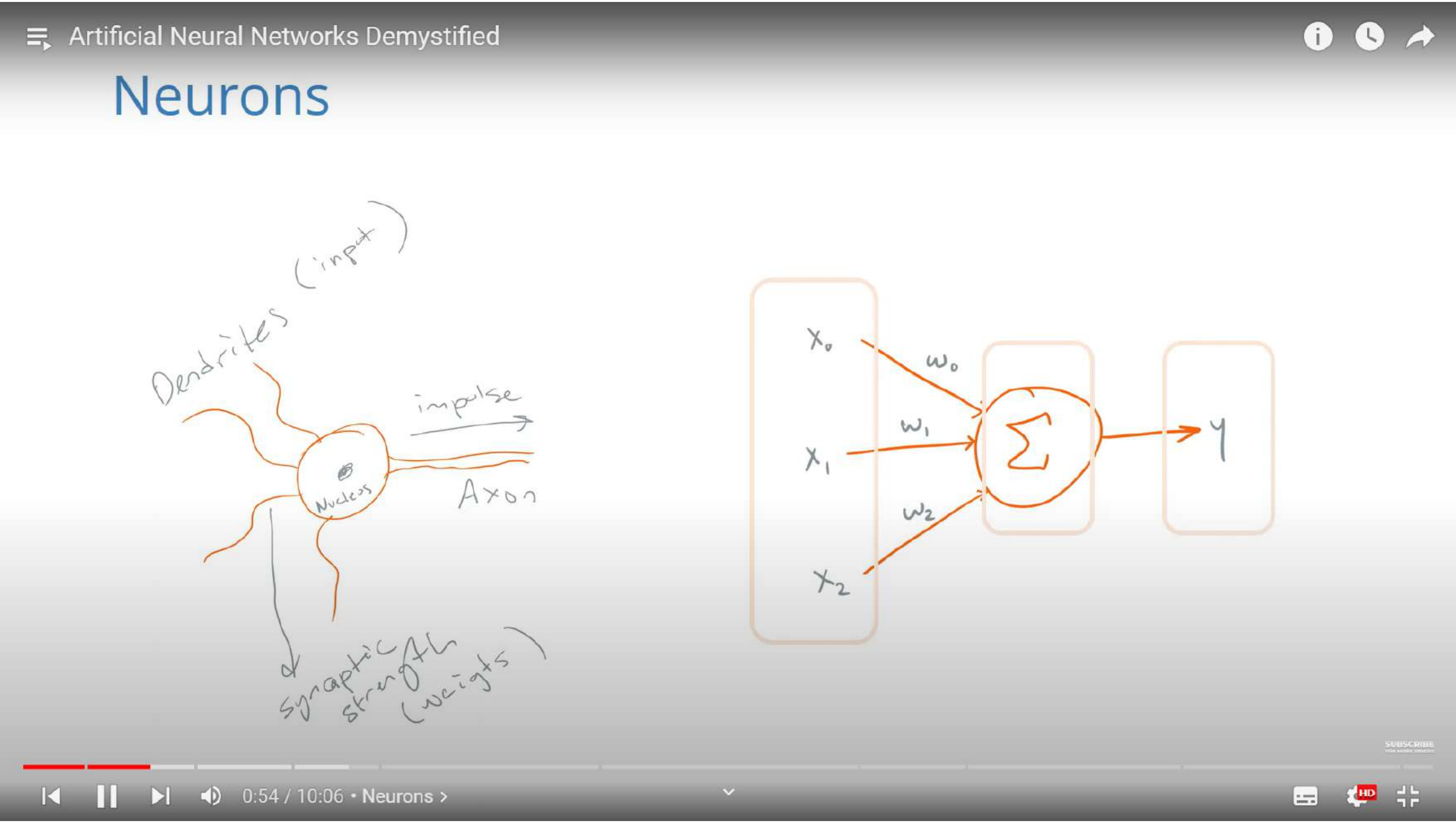
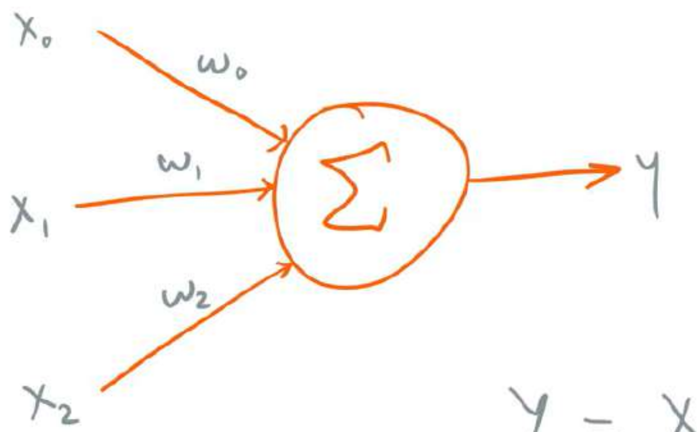

More on this side
Neuroscience
Computer Science
Deep Learning Research



Artificial Neural Networks Demystified
Neurons
Dendrites (input)
impulse
Nucleus
Axon
synaptic strength (weigts)
x0
x1
x2
w0
w1
w2
Σ
y
SUBSCRIBE
0:54 / 10:06 • Neurons >

Artificial Neural Networks Demystified

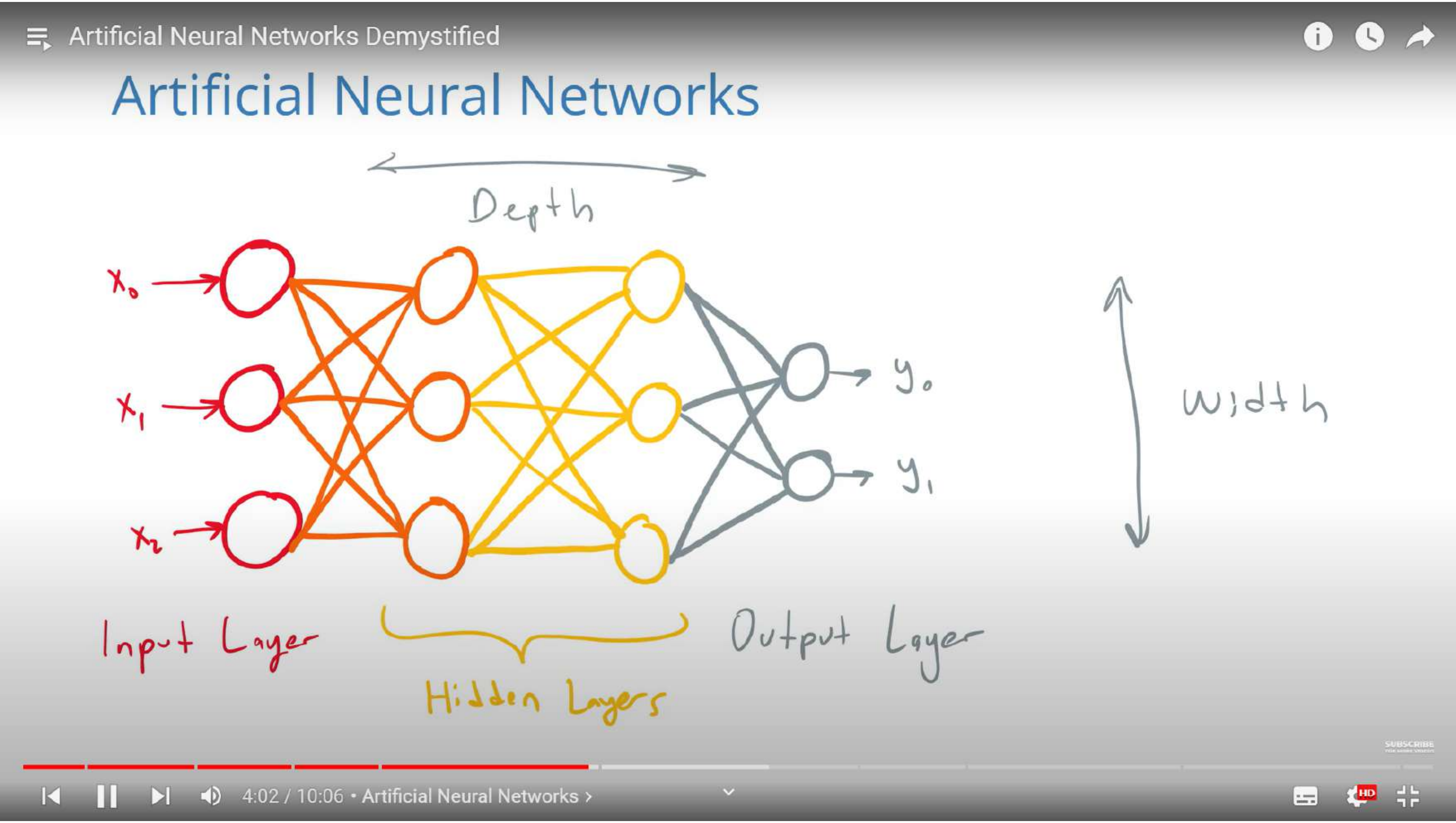
Artificial Neuron



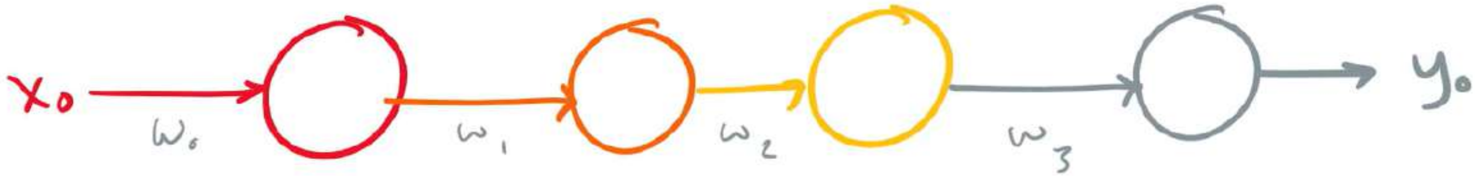
$$y = x_0 w_0 + x_1 w_1 + x_2 w_2 = \sum_i x_i w_i = w^T x$$

$$\begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = x_0 w_0 + x_1 w_1 + x_2 w_2$$

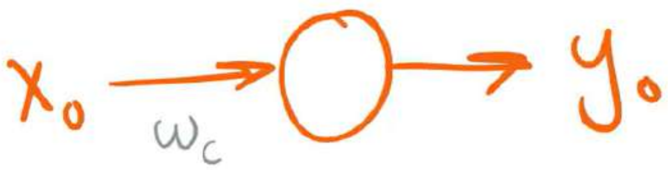
1:52 / 10:06 • Artificial Neuron ›


Artificial Neural Networks Demystified
Artificial Neural Networks
Depth
x_0
x_1
x_2
y_0
y_1
width
Input Layer
Hidden Layers
Output Layer
4:02 / 10:06 • Artificial Neural Networks ›

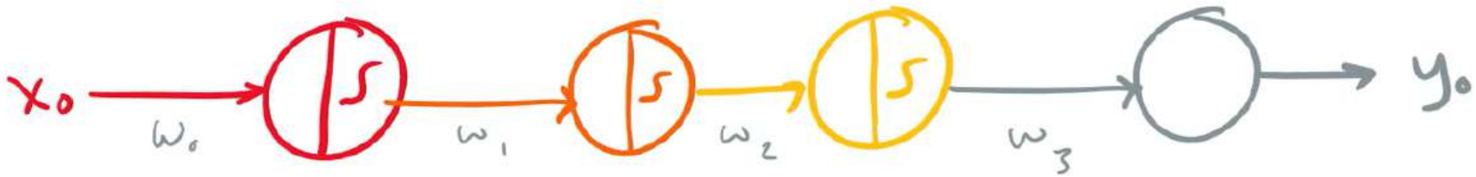
Activation Function



$$y_0 = x_0 w_0 w_1 w_2 w_3 = x_0 w_c$$



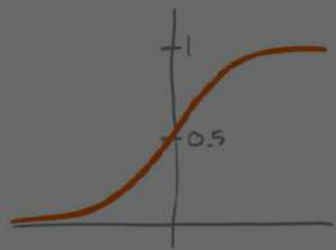
Activation Function



$$w^T x \longrightarrow \sigma(w^T x)$$

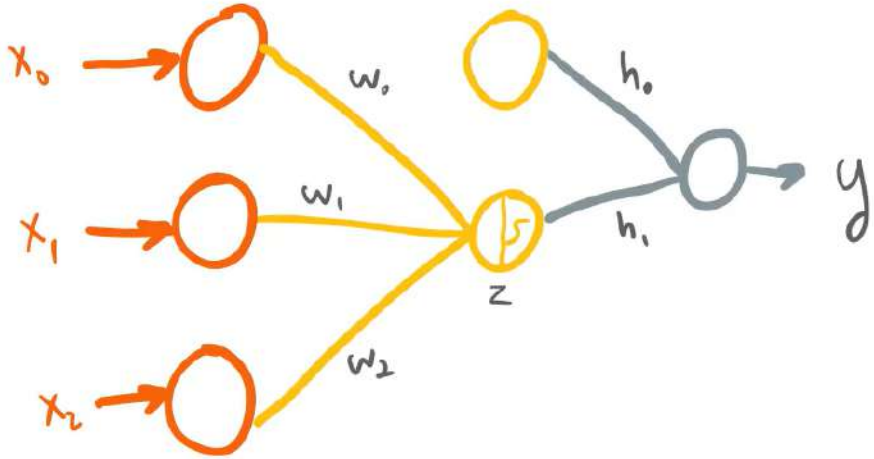
Activation Function

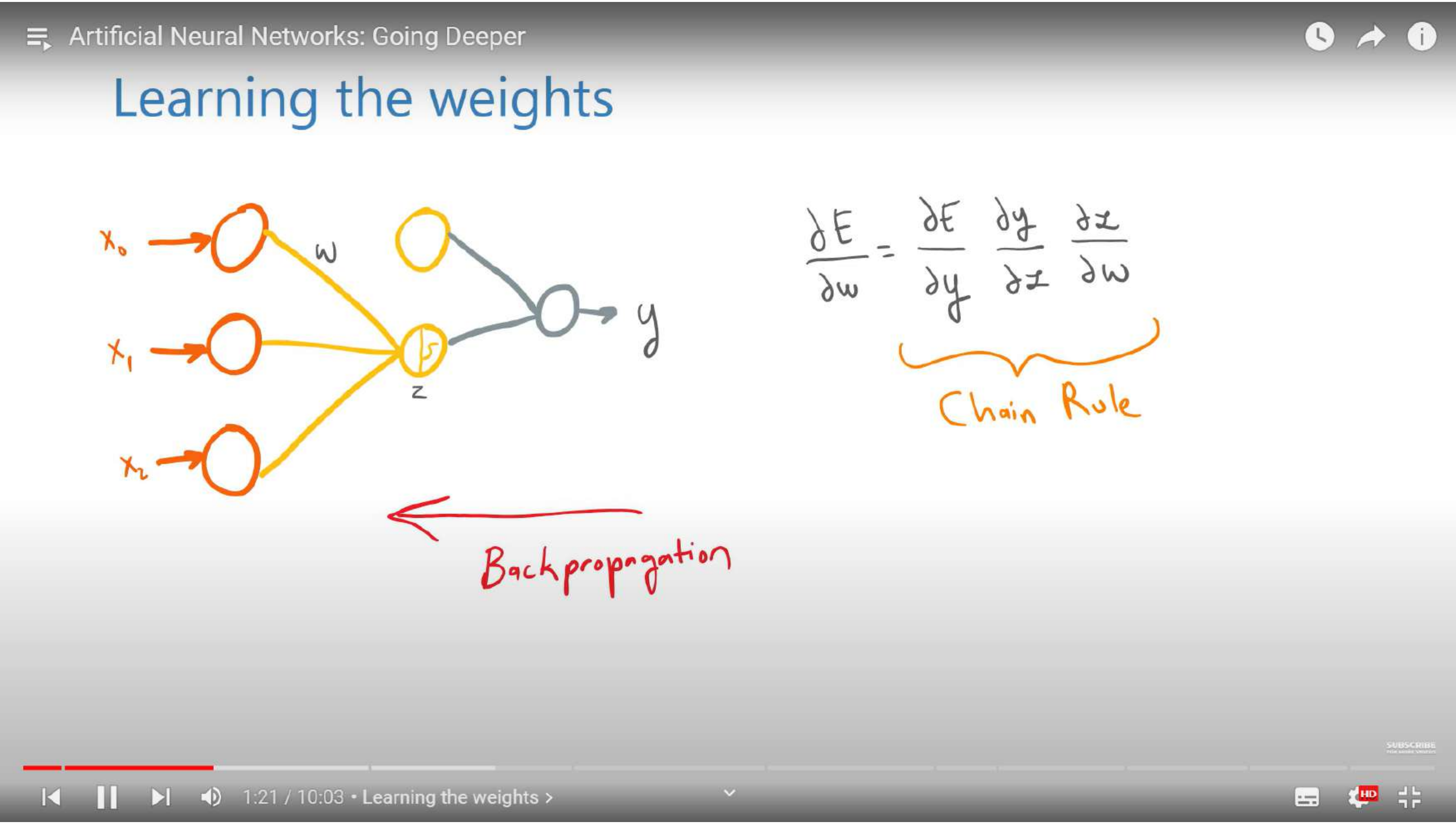
$$w^T x \longrightarrow \sigma(w^T x) \qquad \sigma(x) = \frac{1}{1+e^{-x}}$$



$$y_0 = \sigma(\sigma(\sigma(x_0 w_0) w_1) w_2) w_3$$

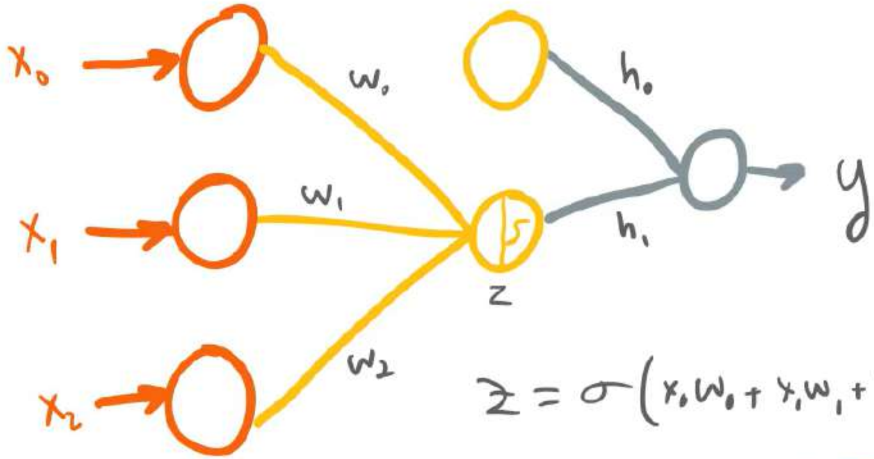
Learning the weights





Artificial Neural Networks: Going Deeper
Learning the weights
x_0
x_1
x_2
w
z
y
$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$
Chain Rule
Backpropagation
1:21 / 10:03 • Learning the weights >

Learning the weights



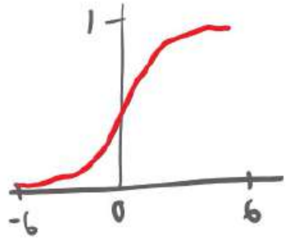
$$z = \sigma(x_0 w_0 + x_1 w_1 + x_2 w_2) \qquad y = h_0 + h_1 z$$

Forward Propagation

Backpropagation

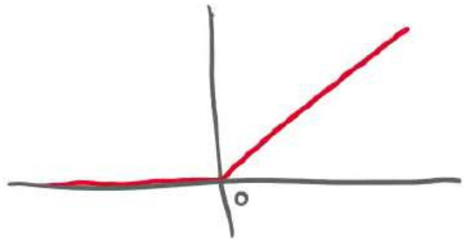
Activation functions

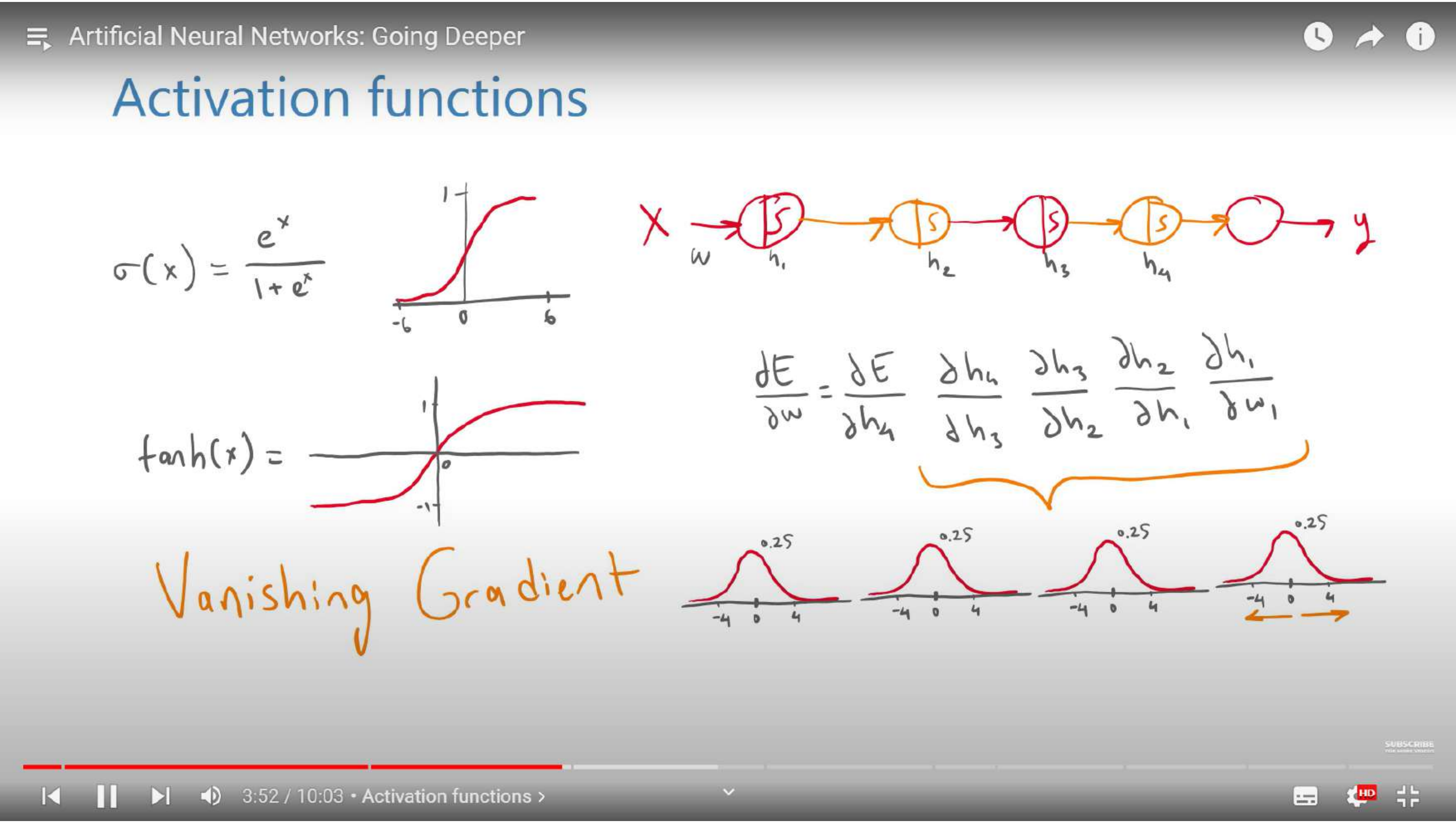
$$\sigma(x) = \frac{e^x}{1+e^x}$$



ReLU: Rectificed Linear Unit

$y = \max(0, x)$

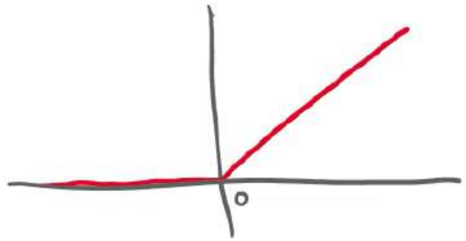


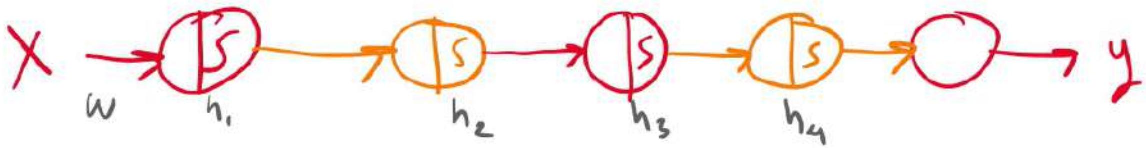


Artificial Neural Networks: Going Deeper
Activation functions
σ(x) = e^x / (1 + e^x)
1
-6
0
6
tanh(x) =
1
0
-1
Vanishing Gradient
x
w
h1
h2
h3
h4
y
∂E/∂w = ∂E/∂h4 · ∂h4/∂h3 · ∂h3/∂h2 · ∂h2/∂h1 · ∂h1/∂w1
0.25
-4
0
4
3:52 / 10:03 • Activation functions ›

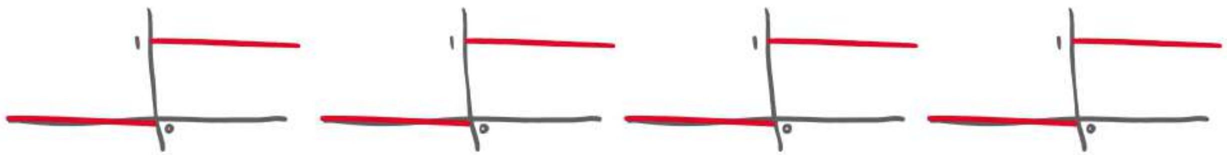
ReLU: Rectificed Linear Unit

$y = \max(0, x)$



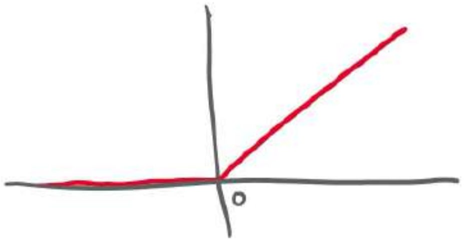


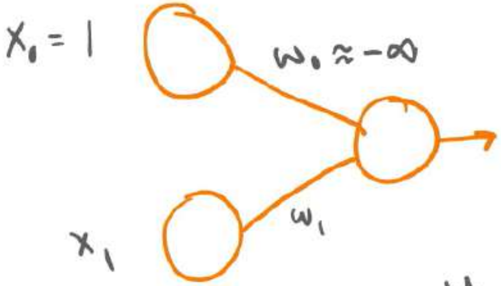
$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$



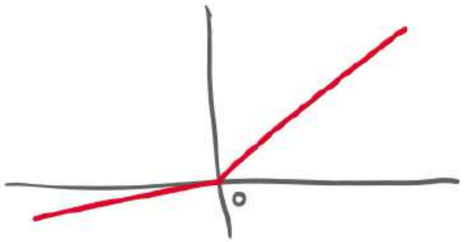
ReLU variants

$y = \max(0, x)$



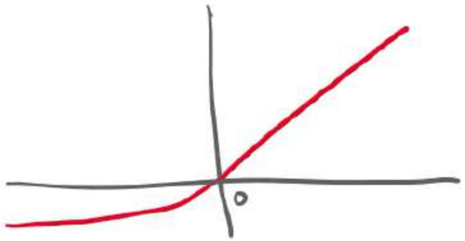


$y \leftarrow \max(0, y) = 0$



Leaky ReLU

Parametric ReLU



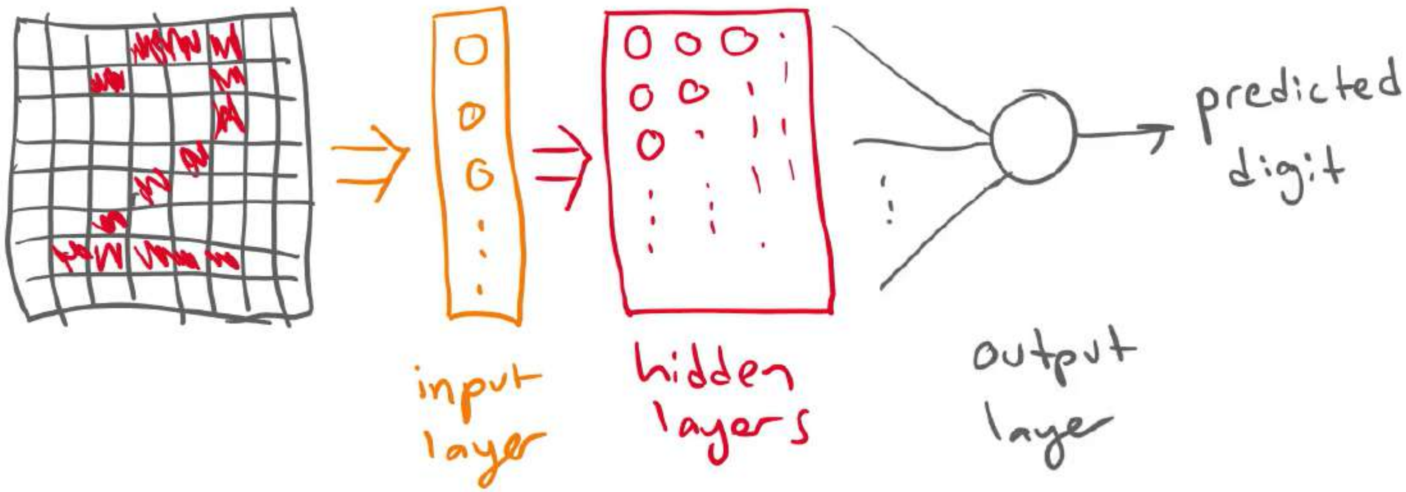
ELU

(Exponential Linear Unit)

Loss functions

$$MSE = \frac{1}{n}\sum_{i}^{n}(y-\hat{y})^2$$

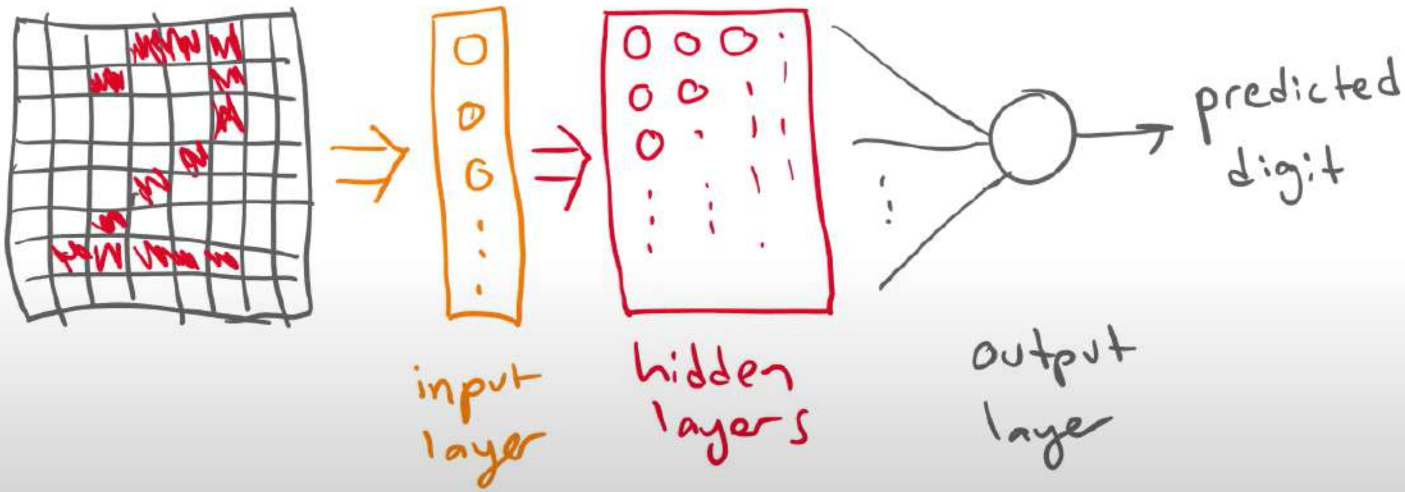
Classification



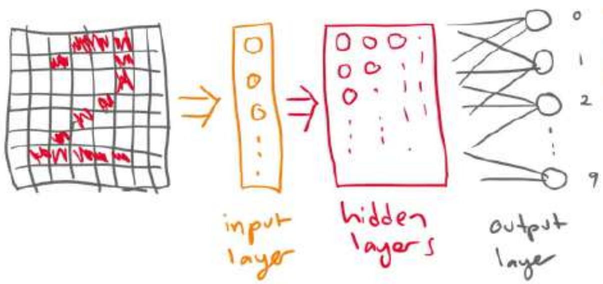
Loss functions

$$MSE = \frac{1}{n} \sum_{i}^{n} (y - \hat{y})^2$$

Classification

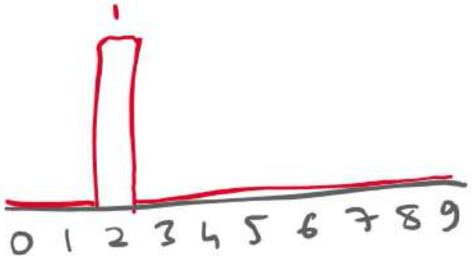


Softmax function



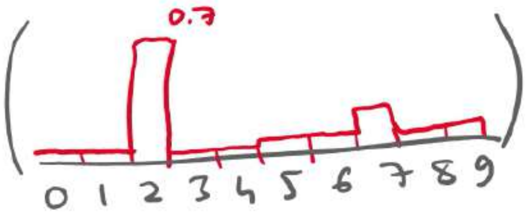
Logits

$$\text{Softmax}(y_i) = \frac{e^{y_i}}{\sum_i e^{y_i}}$$



Ground truth probabilities (one-hot encoding)

sum(



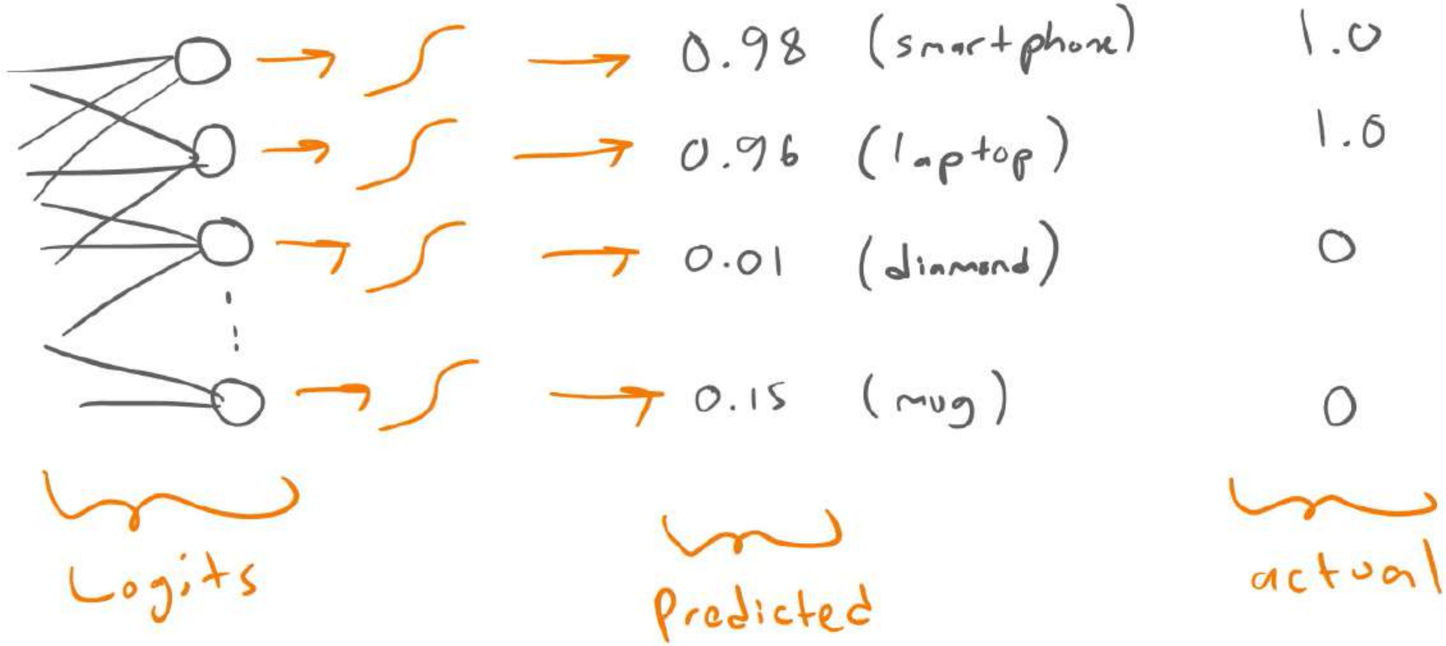
) = 1

Predicted probabilities

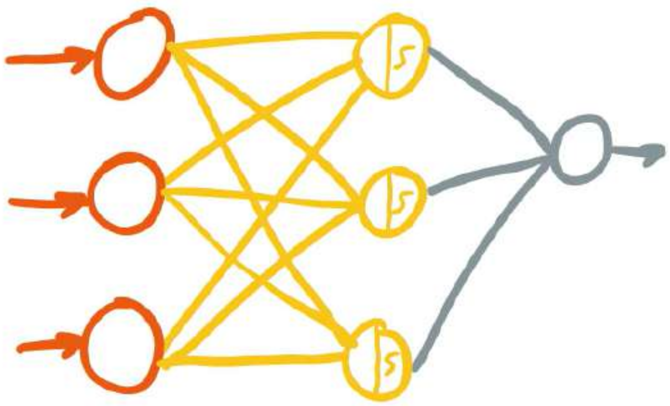
Cross Entropy

$$H(p,q) = -\sum_i p_i \log q_i$$

Multi-label classification



One more thing: do we really need deep models?



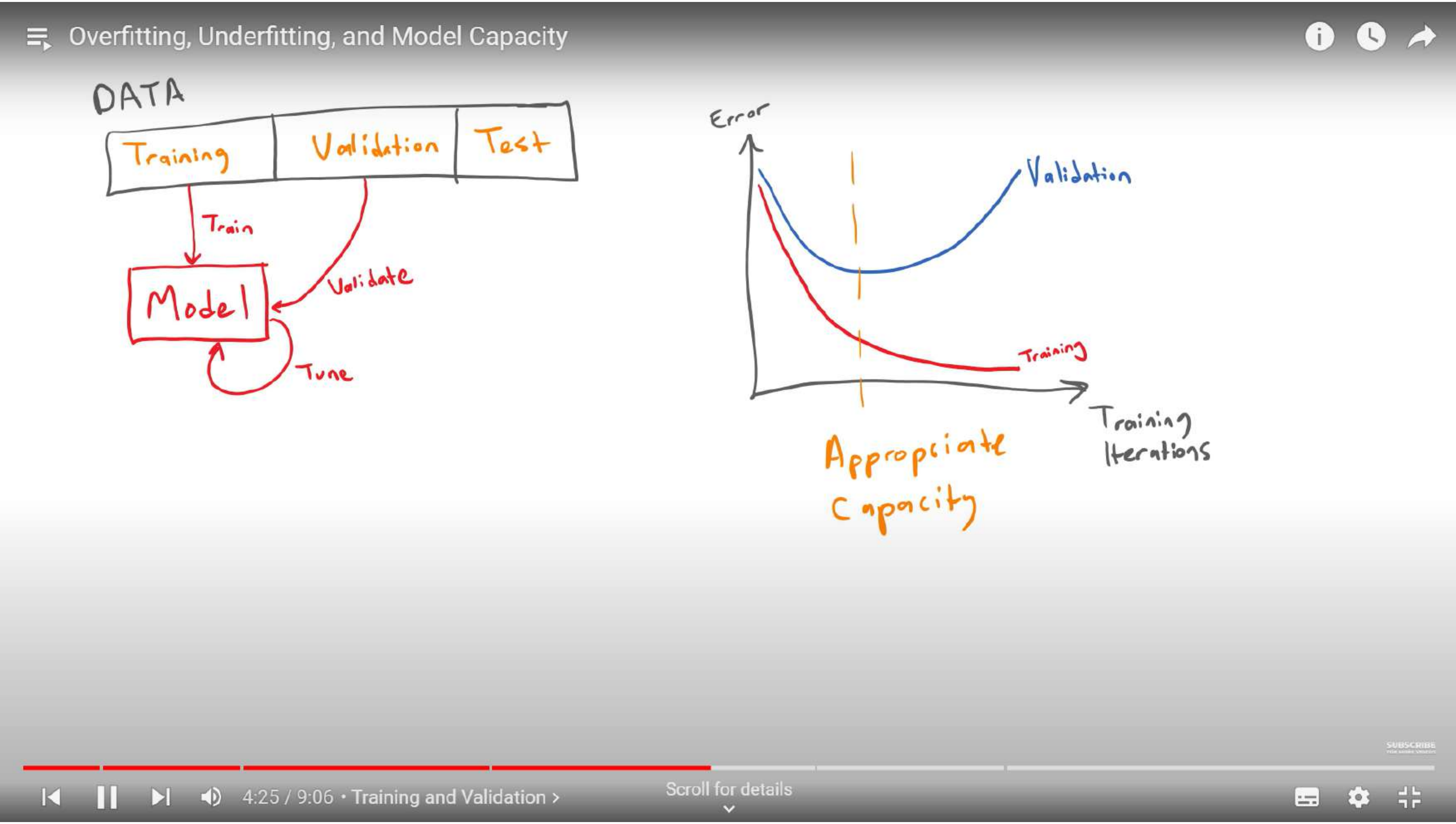
might be too wide

Universal Approximator

can represent $\neq$ can learn

Training loss	Validation loss	
High	High	Underfitting: increase capacity
Low	High	Overfitting: decrease capacity
Low	Low	Good fit: run test
High	Low	Unlikely: debug

What's next?


Overfitting, Underfitting, and Model Capacity
DATA
Training
Validation
Test
Train
Model
Validate
Tune
Error
Validation
Training
Training
Iterations
Appropriate
Capacity
4:25 / 9:06 • Training and Validation ›
Scroll for details

Overfitting, Underfitting, and Model Capacity

Hyperparameters

- # of training iterations
- # of layers (depth)
- # of hidden units (width)
- types of layers
- regularization methods
- regularization strength
- optimization algorithm
- learning rate
- batch size
- ...

4:41 / 9:06 · Training and Validation ›

Scroll for details


Overfitting, Underfitting, and Model Capacity
DATA
Training
Validation
Test
5:06 / 9:06 • Partitioning Data >
Scroll for details


Regularization and Data Augmentation
Fewer parameters
demand less computational power
Best performing models tend to be
large but trained in a way that
restricts the utilization of their entire potential
Regularization
- encourages models to have a preference
towards simpler models
- reduces the risk of overfitting
SUBSCRIBE
1:05 / 8:08
Scroll for details
HD

$$\text{Loss} \leftarrow \underbrace{\text{Loss}}_{\sum_i (y_i - \hat{y}_i)^2} + \underbrace{(\text{weight penalty})}_{\alpha \sum_i |w_i|}$$

$$\mathcal{L}oss \leftarrow \underbrace{\mathcal{L}oss}_{\sum_i (y_i - \hat{y}_i)^2} + \underbrace{(\text{weight penalty})}_{\alpha \sum_i w_i^2}$$

y_i: actual

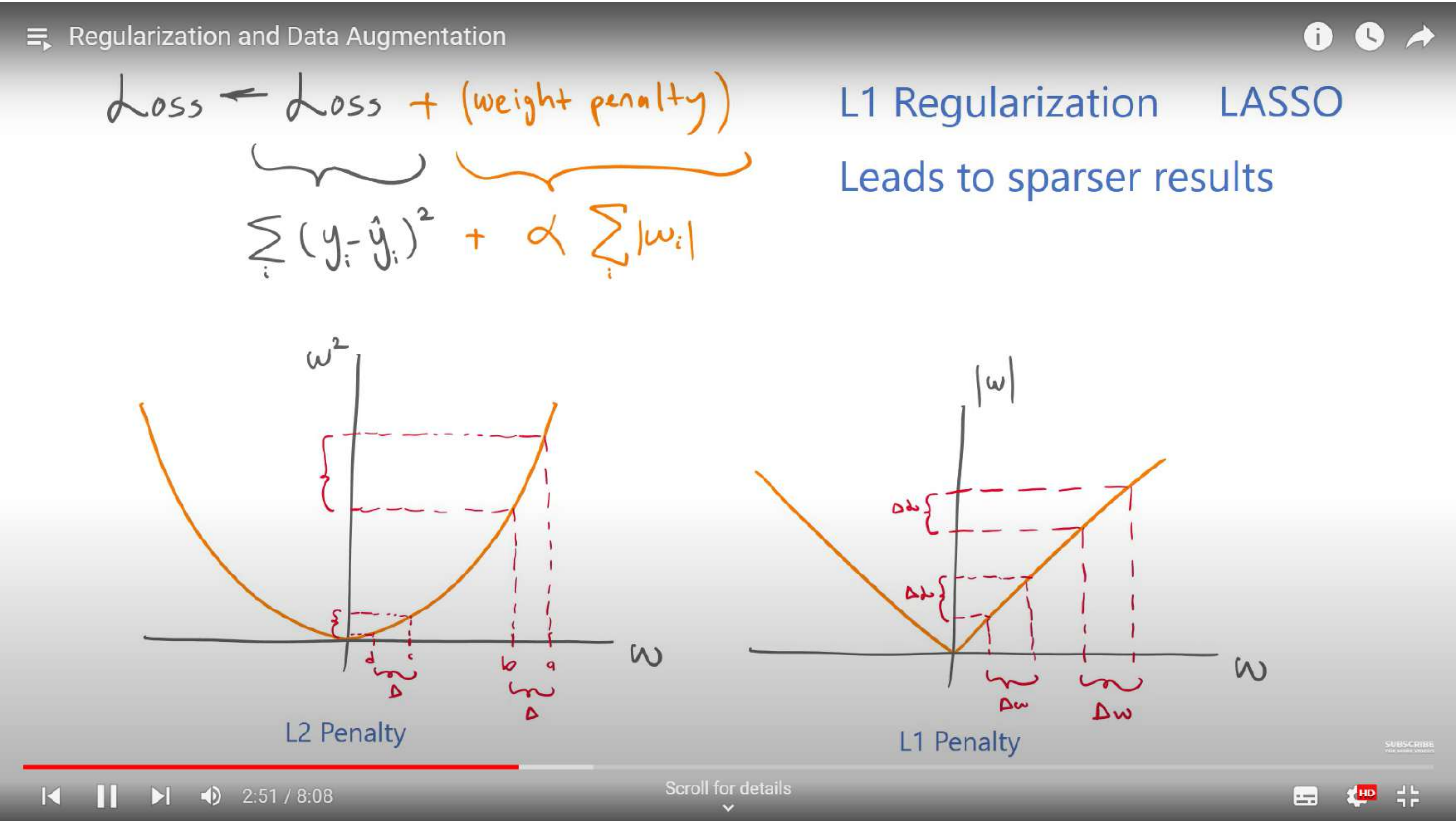
$\hat{y}_i$: predicted

α: regularization strength

L2 Regularization

$$\text{Minimize} \left(\sum_i (y_i - \hat{y}_i)^2 + \alpha \sum_i w_i^2 \right)$$

Ridge Regression


Regularization and Data Augmentation
Loss ← Loss + (weight penalty)
$\sum_i (y_i - \hat{y}_i)^2 + \alpha \sum_i |w_i|$
L1 Regularization
LASSO
Leads to sparser results
w^2
w
d
c
b
a
Δ
Δ
L2 Penalty
|w|
Δw
Δw
Δw
Δw
w
L1 Penalty
2:51 / 8:08
Scroll for details

Data Augmentation


Original Image


Scaled Image


Translated Image


Rotated Image


Mirrored Image


Brightness / Contrast Shifted Image

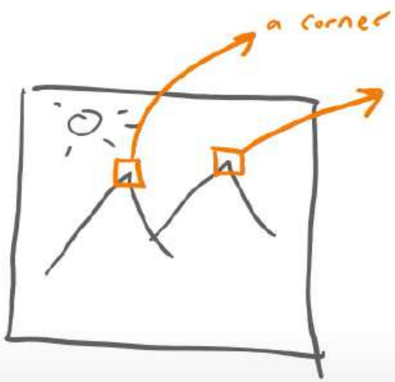

Noise Added Image

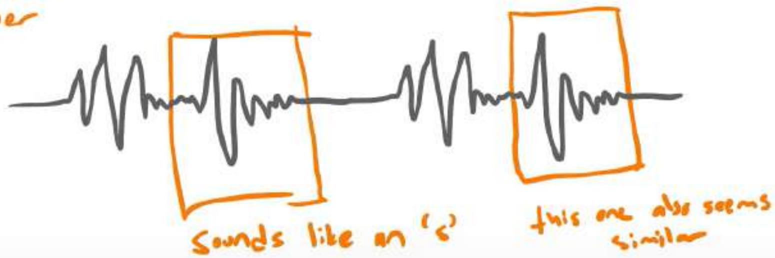
Regularization methods introduce additional information such as

- *smaller weights are better*
- *parameter sharing is useful*
- *distorting the input in some ways shouldn't change the results*

Prevent overfitting

- *limit model capacity*
- *get more data*





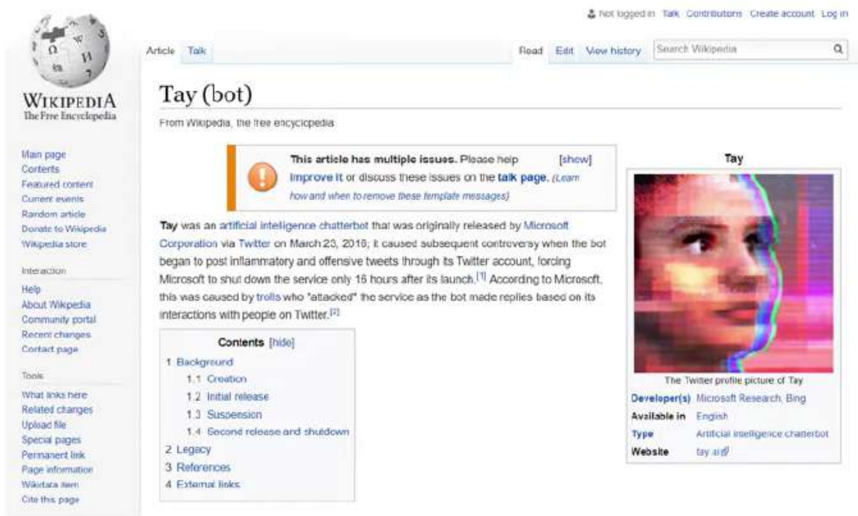
Data augmentation

Dataset Bias

data can be biased no matter how big it is

Models used for

medical, financial, and legal purposes
affect people's lives



Biased models can further reinforce biases

Subjective Studies

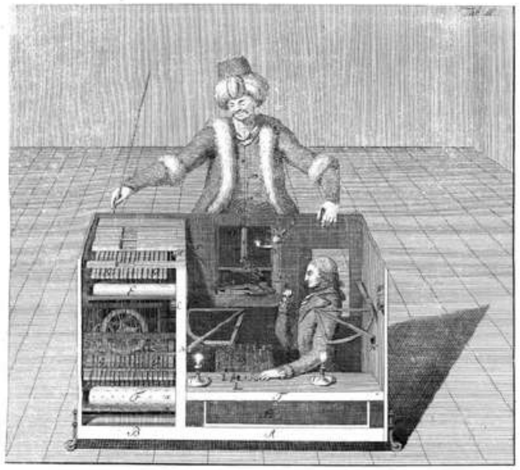
Surveys

Games

Crowdsourcing

Data collection guidelines

- *Ask unbiased questions*
- *Design an easy to use interface*
- *Make it fun for the users*
- *Ensure that the entire process is ethical*



Data Imputation

Caveat: *data might not be missing at random*

Gender	Likes
M	Cats
M	Dogs
F	Missing
M	Cats
F	Dogs
F	Dogs
F	Cats

Classes
Cats
Dogs
Missing

Feature scaling

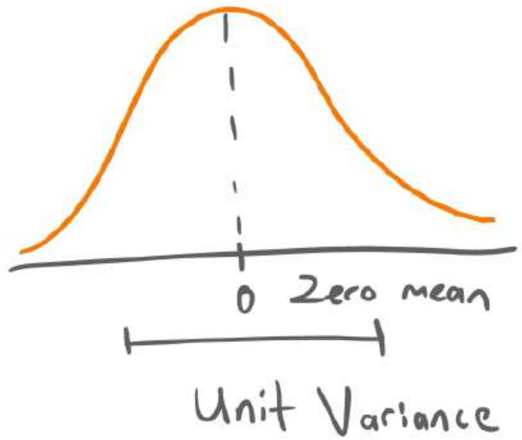
Variable	Range of values
Age	0 – 100+
Annual income	0 – 1,000,000+
Years of experience	0 – 40+

Range of values
0 – 1
0 – 1
0 – 1

$$\hat{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$\widehat{age} = \frac{age - 18}{99 - 18}$$

Standardization



$$\hat{x} = \frac{x - \mu}{\sigma}$$

Data Imbalance

Classes	Number of samples
Cats	5000
Dogs	5000
Tigers	150
Caracals	50
Axolotls	25

Undersampling

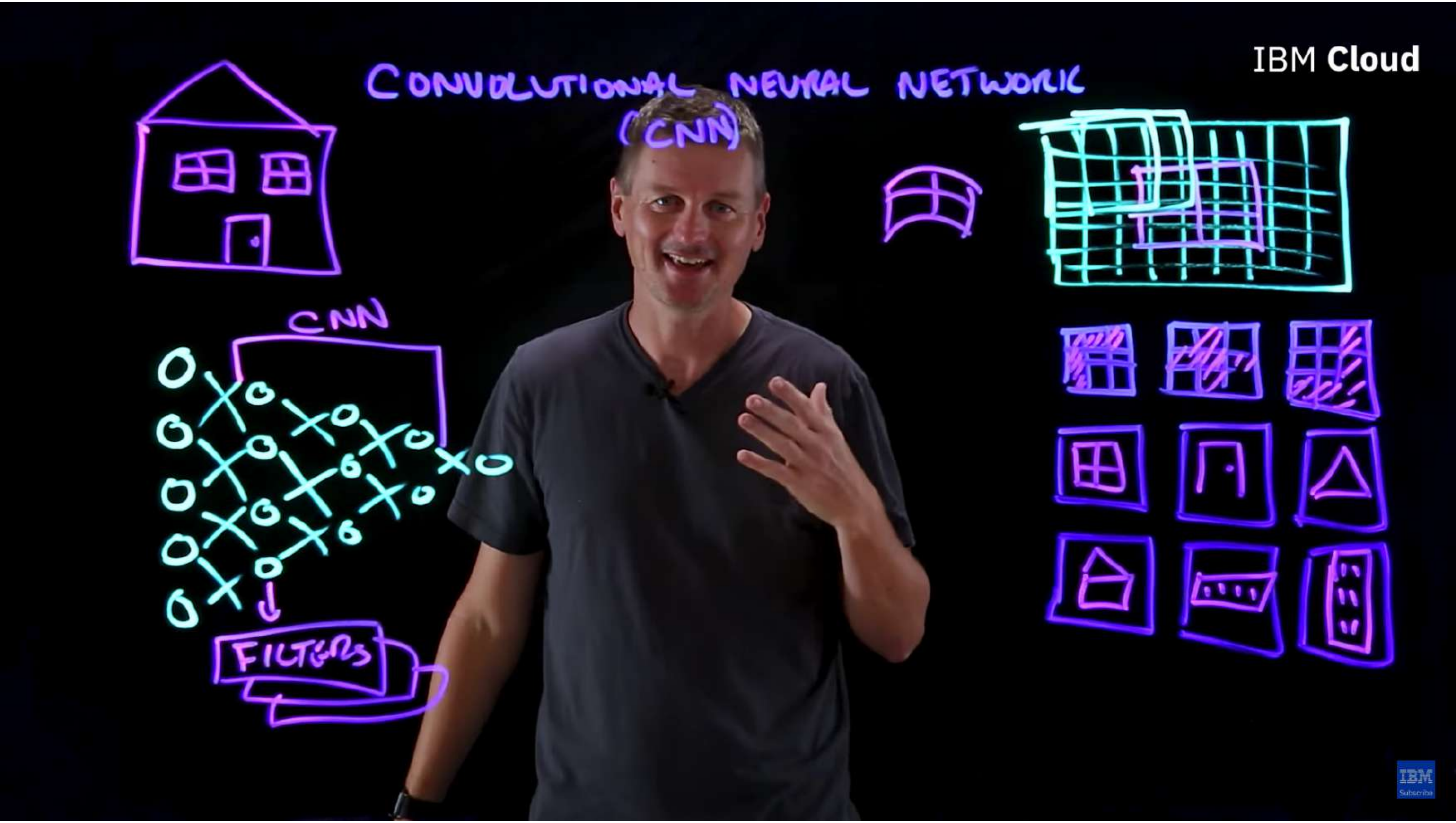
Classes	# samples
Cats	25
Dogs	25
Tigers	25
Caracals	25
Axolotls	25

Oversampling

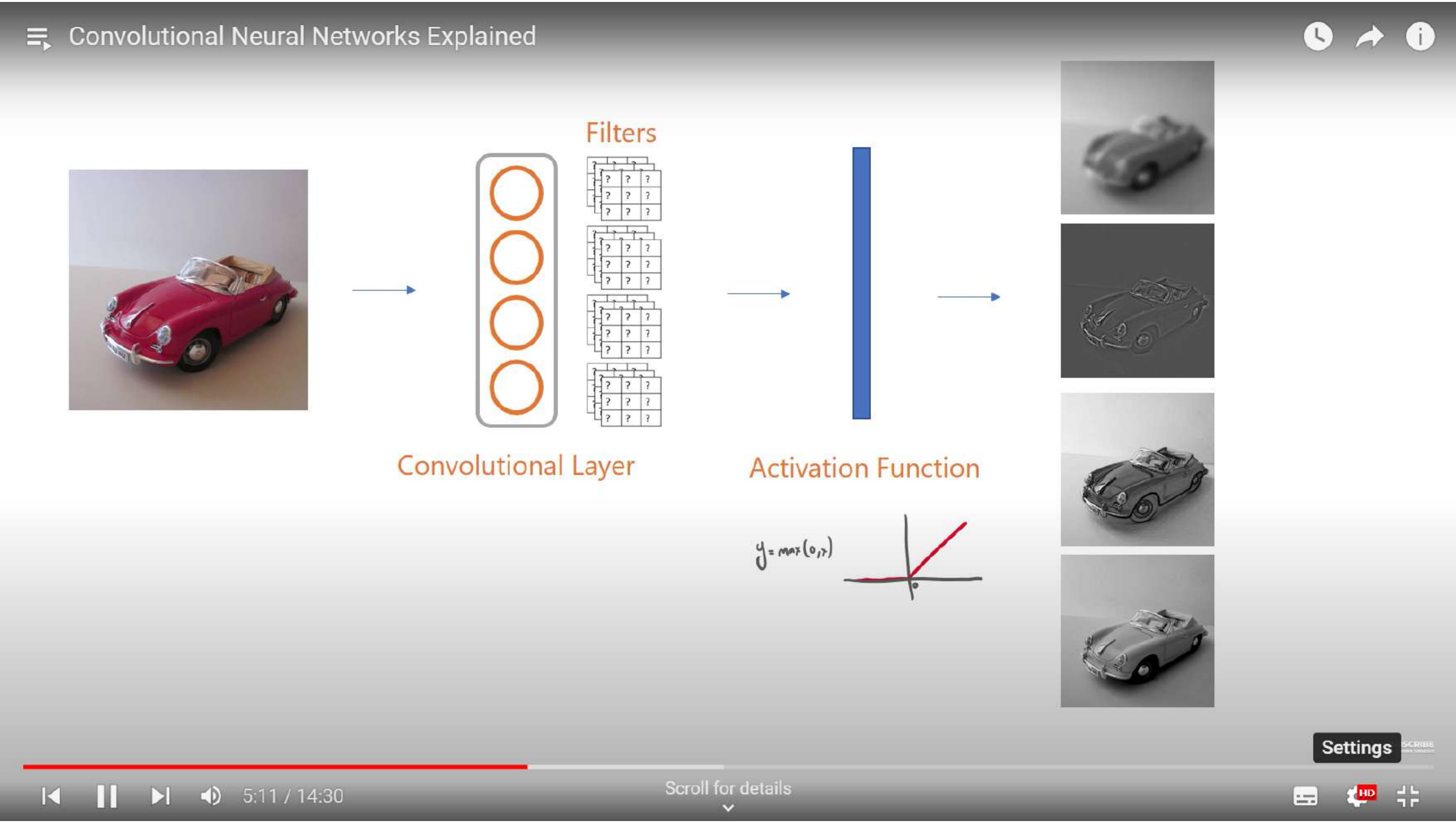
Classes	# samples
Cats	5000
Dogs	5000
Tigers	5000
Caracals	5000
Axolotls	5000

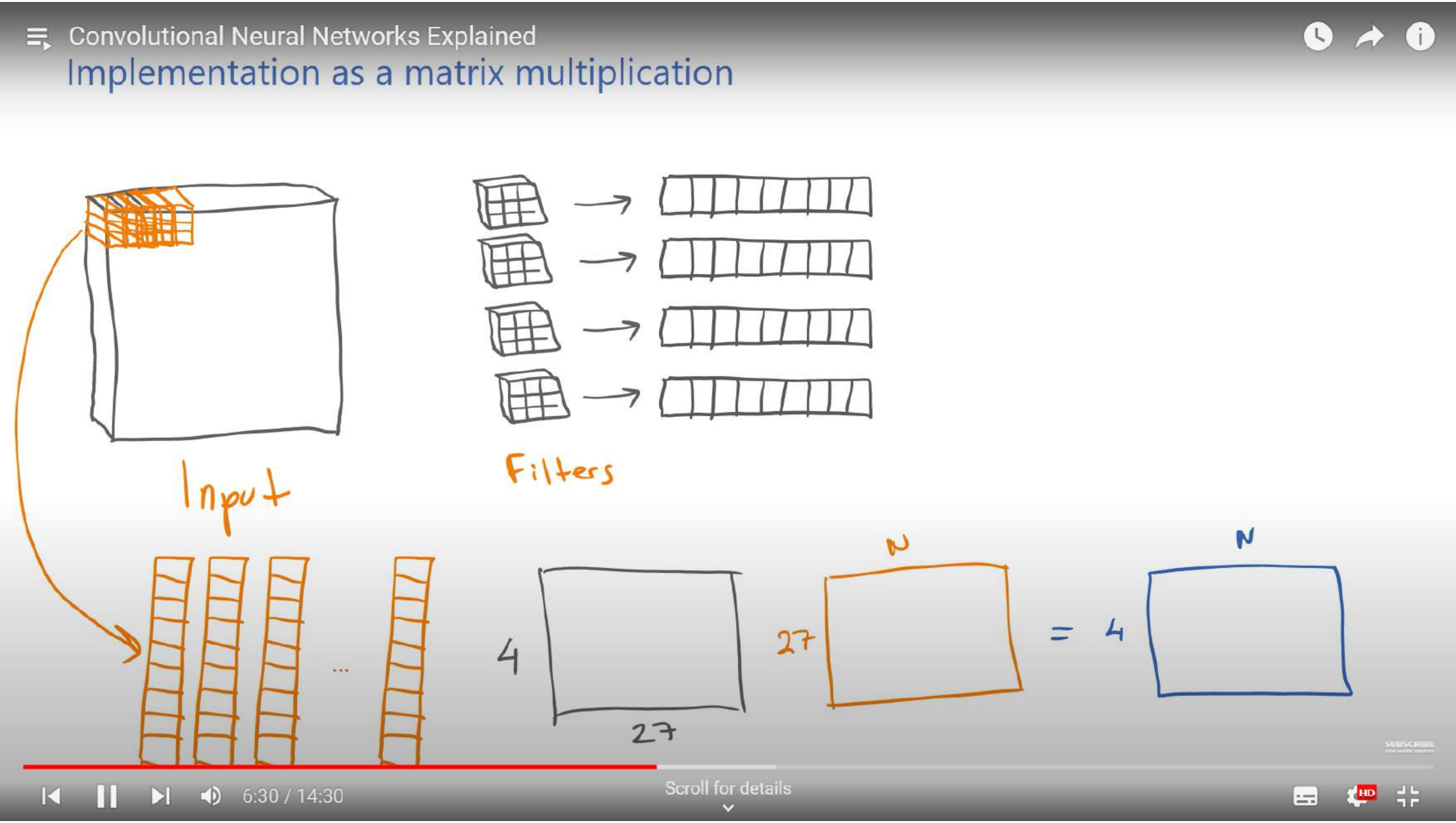
Class-weighted loss function

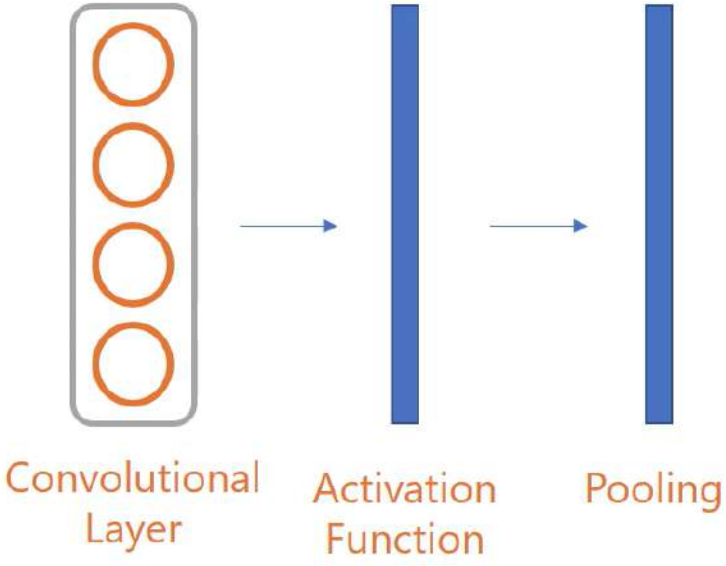
$$Loss = \sum \underbrace{w}_{\text{Class weights}} \, q \log p$$

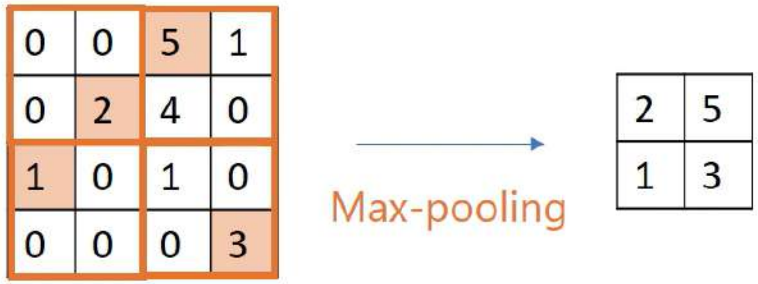


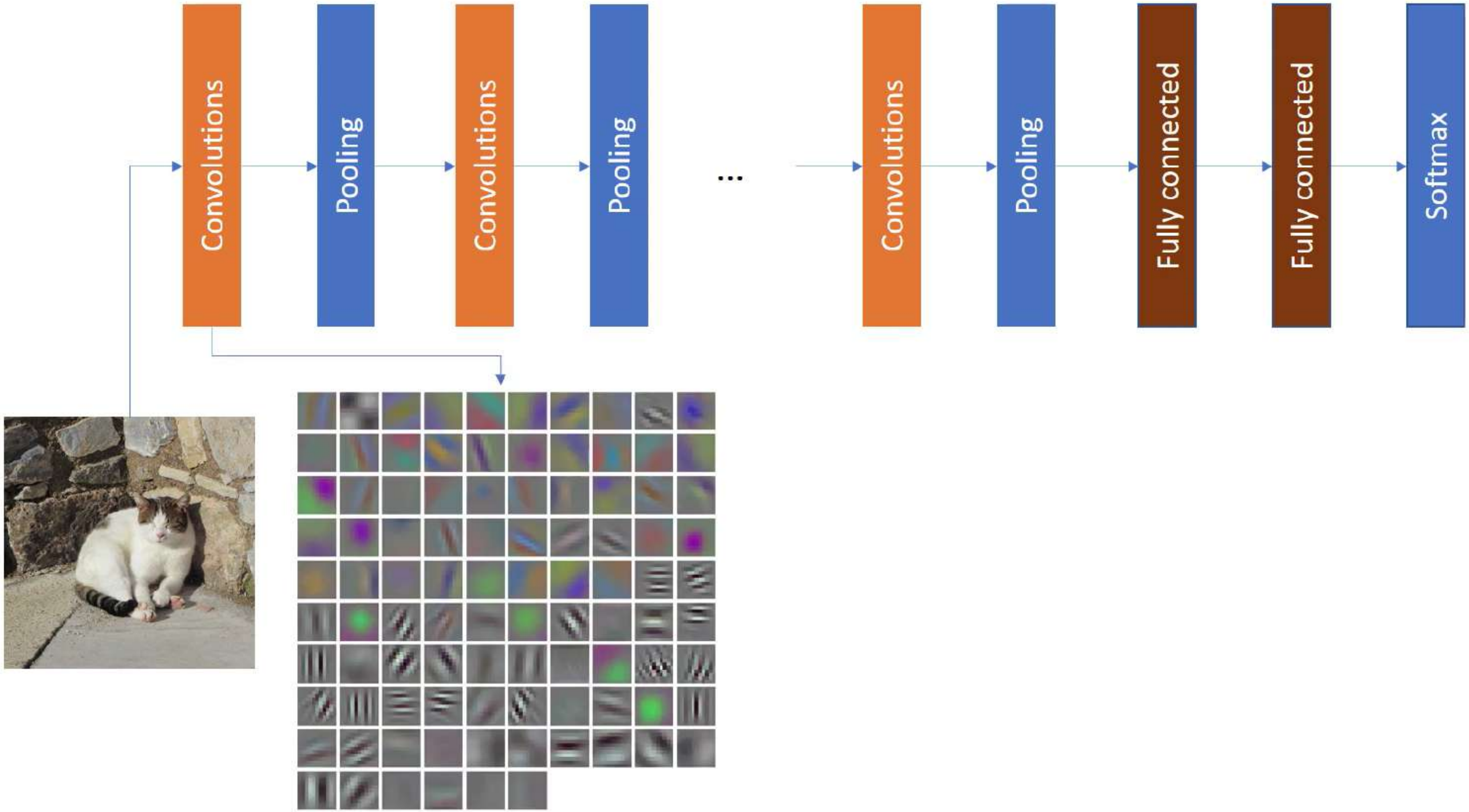
CONVOLUTIONAL NEURAL NETWORK
(CNN)
IBM Cloud
CNN
FILTERS


Convolutional Neural Networks Explained
Filters
Convolutional Layer
Activation Function
y = max(0,x)
Settings
5:11 / 14:30
Scroll for details


Convolutional Neural Networks Explained
Implementation as a matrix multiplication
Input
Filters
...
4
27
N
27
=
4
N
6:30 / 14:30
Scroll for details
HD

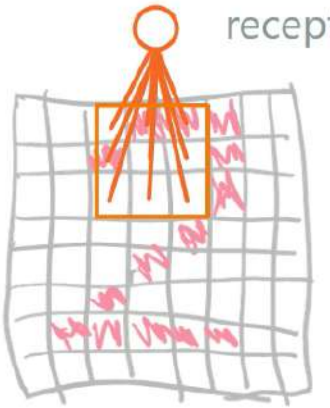

Convolutional Layer
Activation Function
Pooling

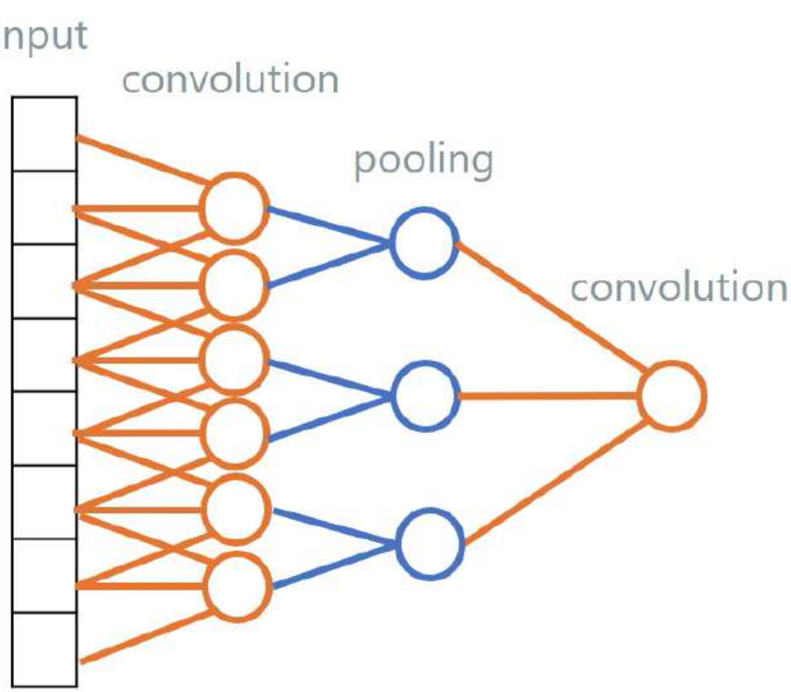

0 0 5 1
0 2 4 0
1 0 1 0
0 0 0 3
Max-pooling
2 5
1 3

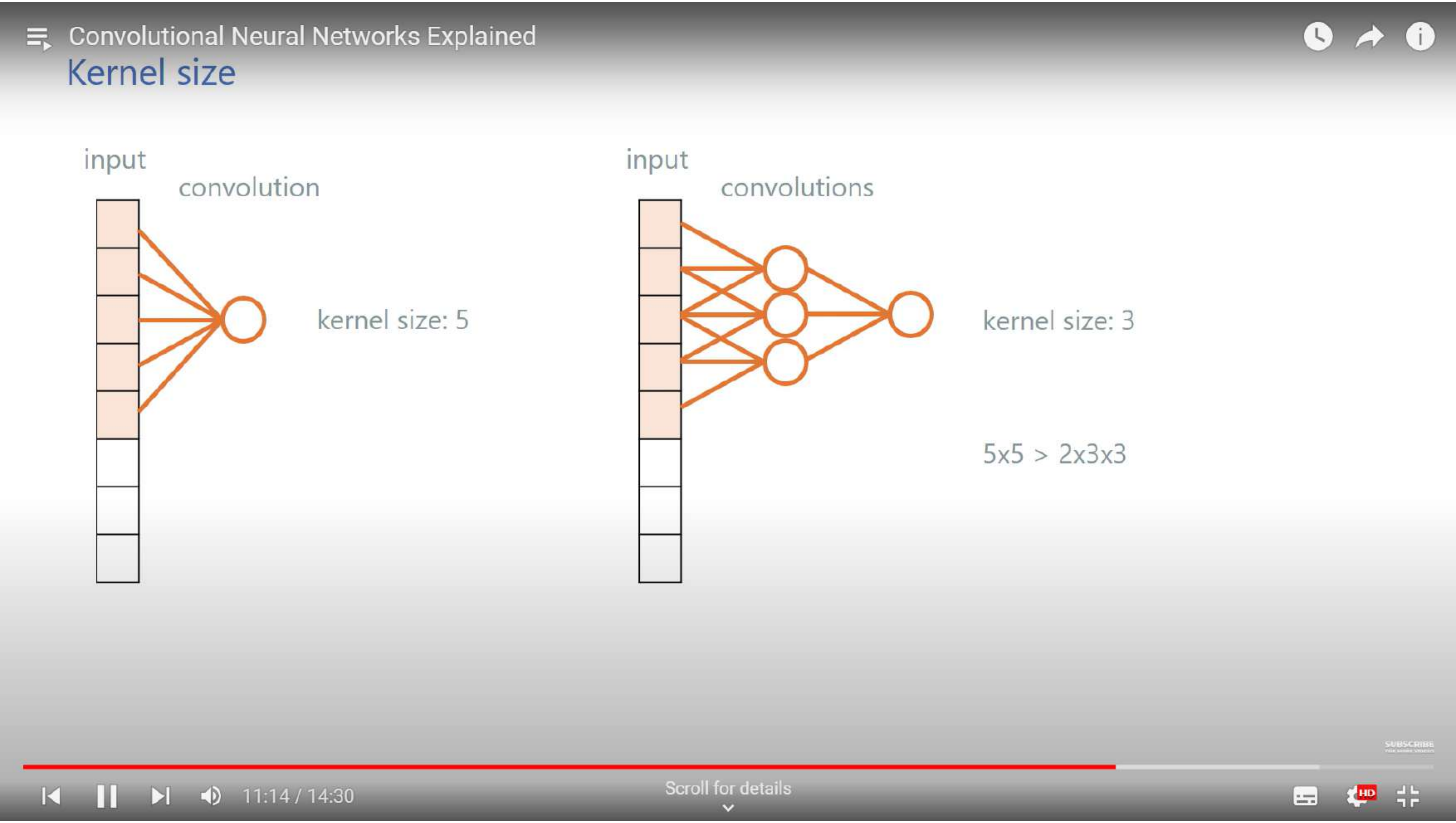


Convolutions
Pooling
Convolutions
Pooling
...
Convolutions
Pooling
Fully connected
Fully connected
Softmax

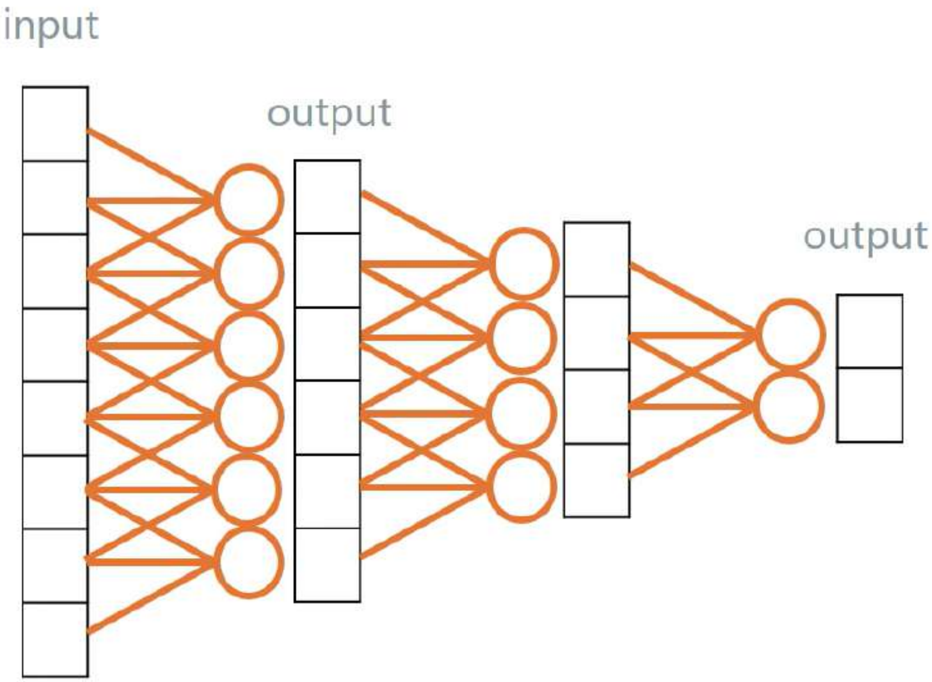
Receptive field



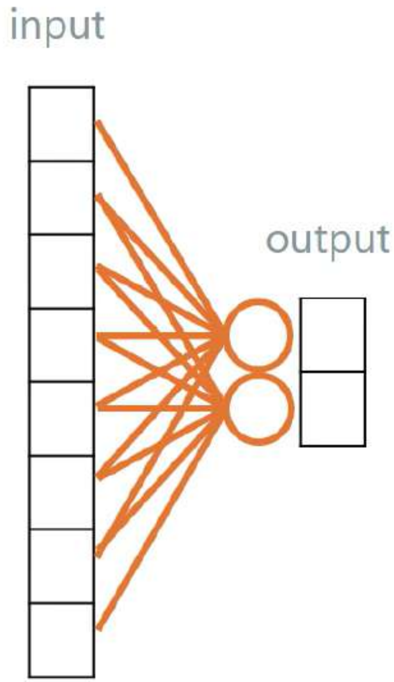



Convolutional Neural Networks Explained
Kernel size
input
convolution
kernel size: 5
input
convolutions
kernel size: 3
5x5 > 2x3x3
11:14 / 14:30
Scroll for details

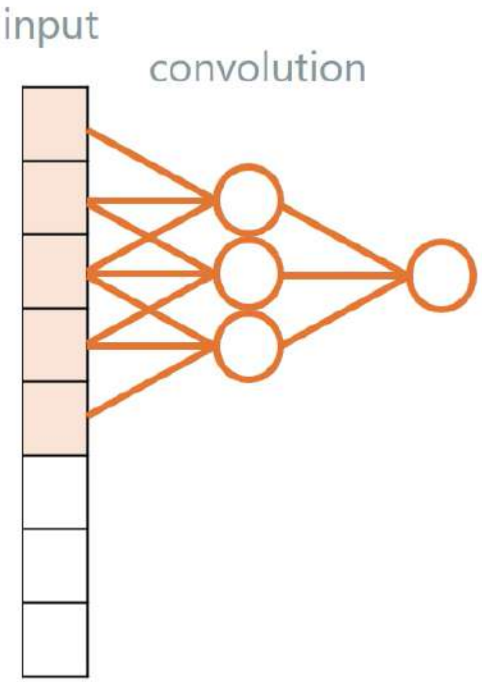
Padding



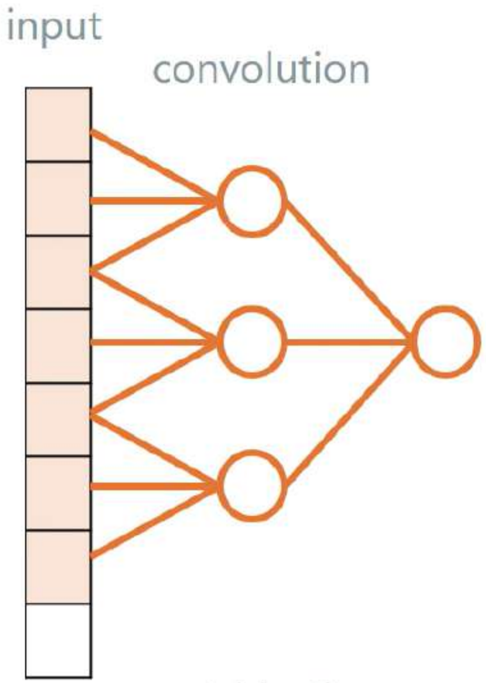
Padding='VALID'



Stride



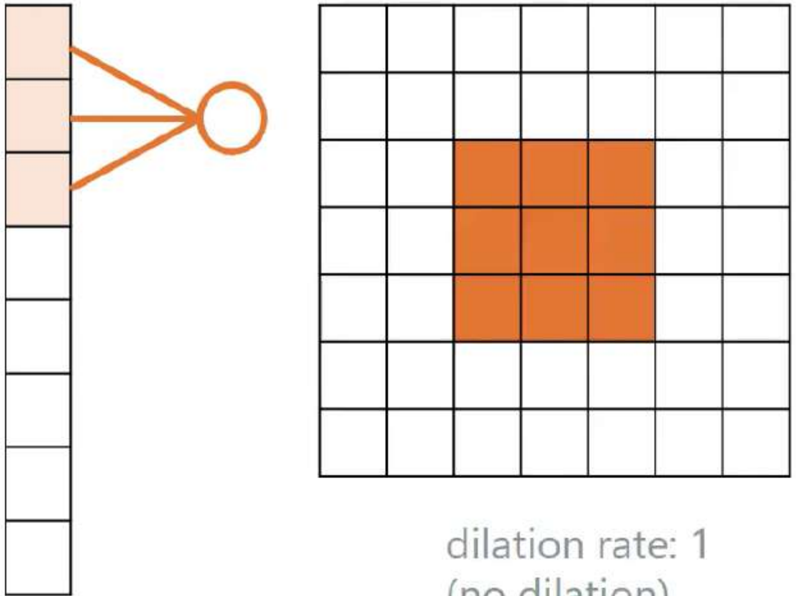
stride: 1



stride: 2

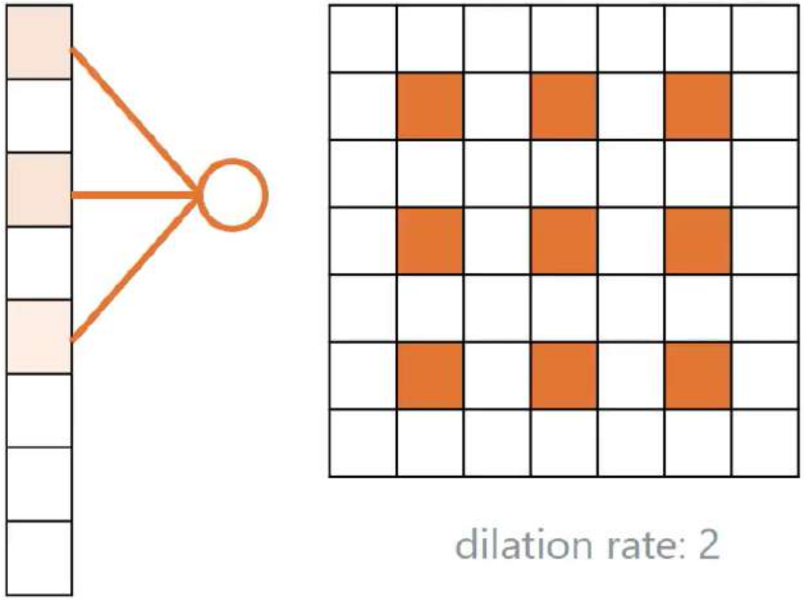
Dilation rate

kernel



dilation rate: 1
(no dilation)

kernel



dilation rate: 2

dilated convolution
atrous convolution
à trous convolution

Building blocks of CNNs

- Convolution
- Pooling

Hyperparameters

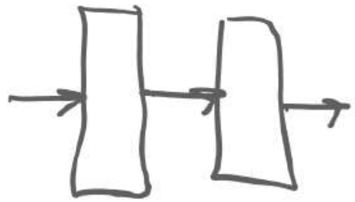
- Kernel size, stride, dilation rate
- Number of filters, number of layers
- Pooling size & type
- Padding type
- The way we arrange the layers

What's next?

- How to design a CNN
- Popular CNN architectures

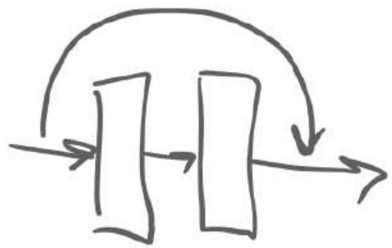
No human intervention in deep learning?

- humans are still in the loop!



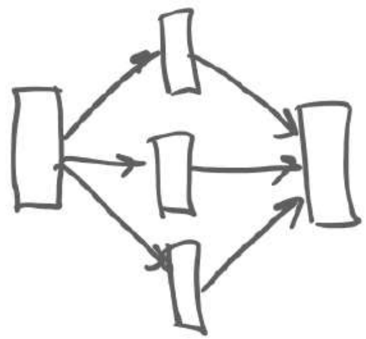
Grid search on all hyperparameters?

- too many hyperparameters
- infinitely many ways to design a network



Designing a deep neural network involves

- human expertise
- trial and error



How to design a ConvNet?

TL;DR

- you don't design
- pick something that works and use it

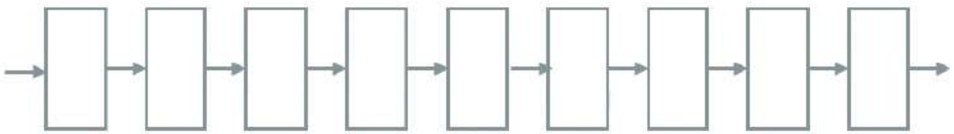
Working on novel problems?

- borrow ideas from successful models
- design your own model

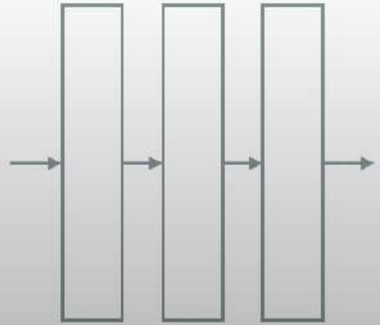
How to Design a Convolutional Neural Network

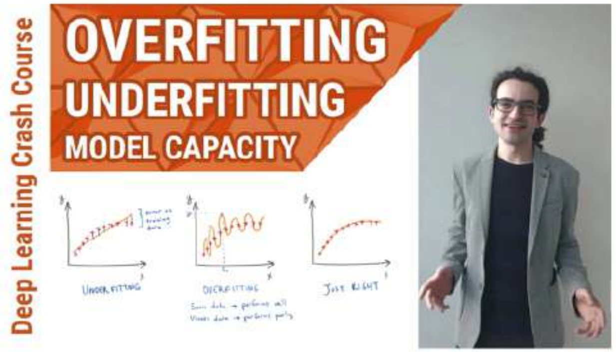
How do choose the number of layers and units?

- start small
- gradually increase model size
- smallest model: linear regression
- deeper:



- wider:





3:56 / 11:46 • How do choose the number of layers and units? >

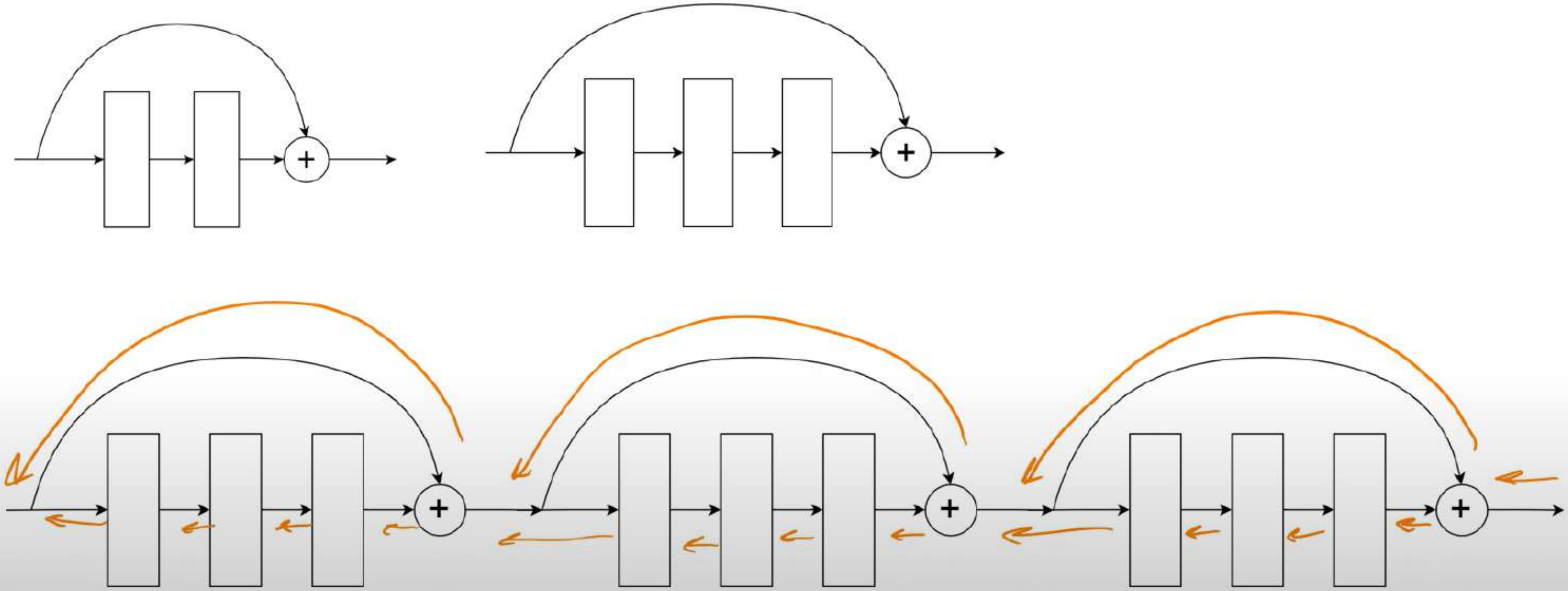
Scroll for details

Skip connections

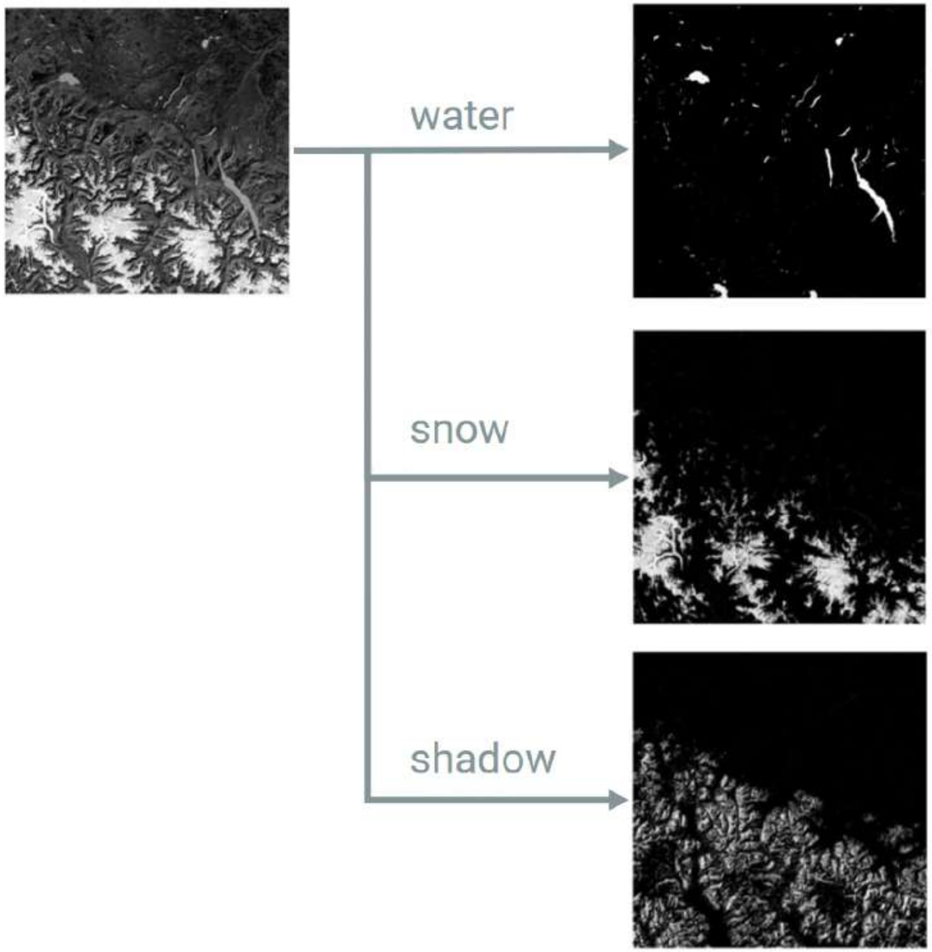
- Fully Convolutional Networks

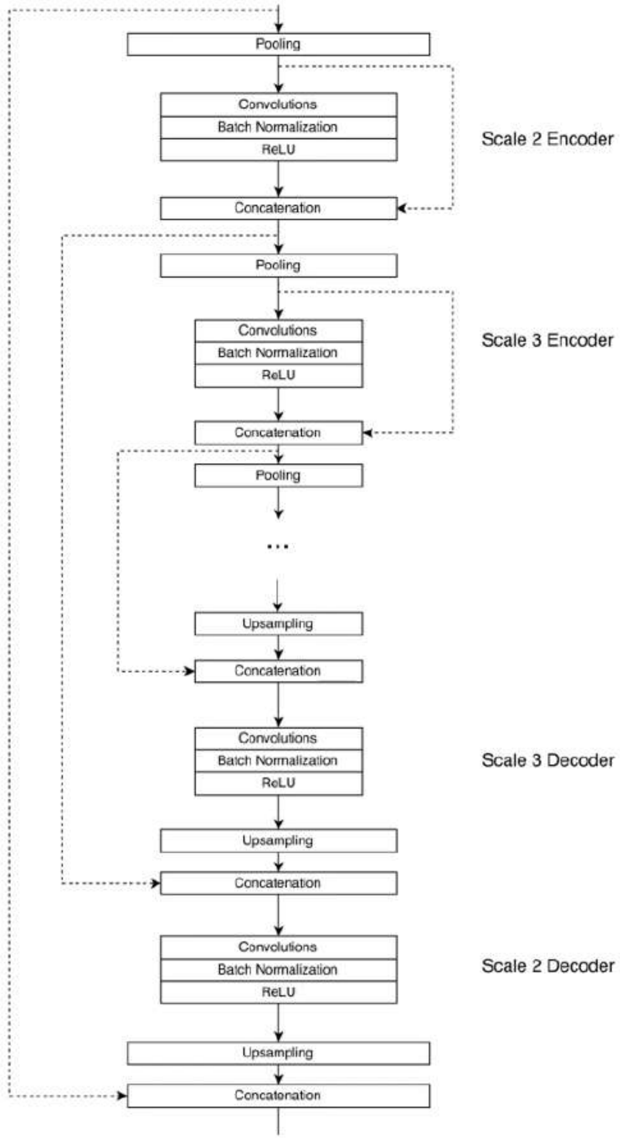
Skip connections

- building blocks of the ResNet architecture



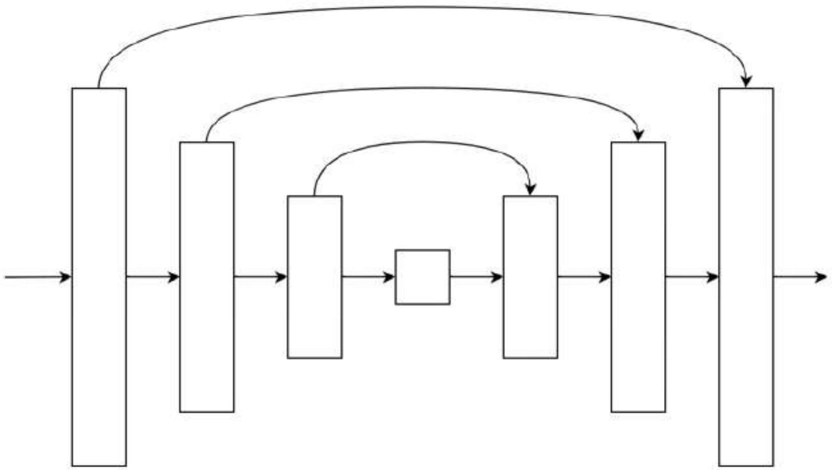
Skip connections





Skip connections

- Fully Convolutional Networks



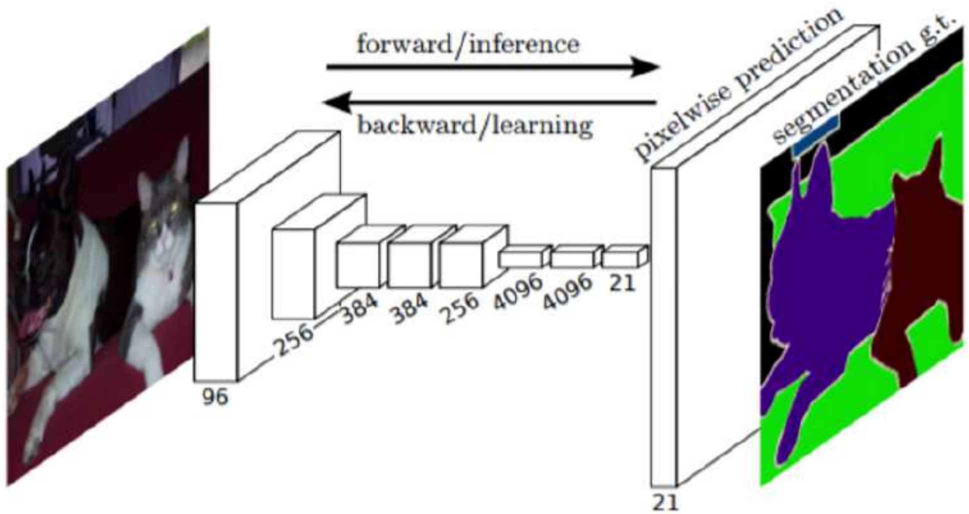


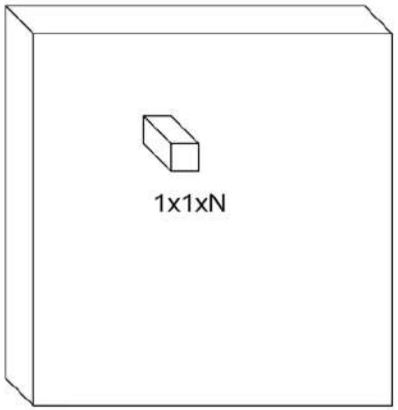
Figure credit: Long and Shelhamer (2015)

How to choose kernel size?

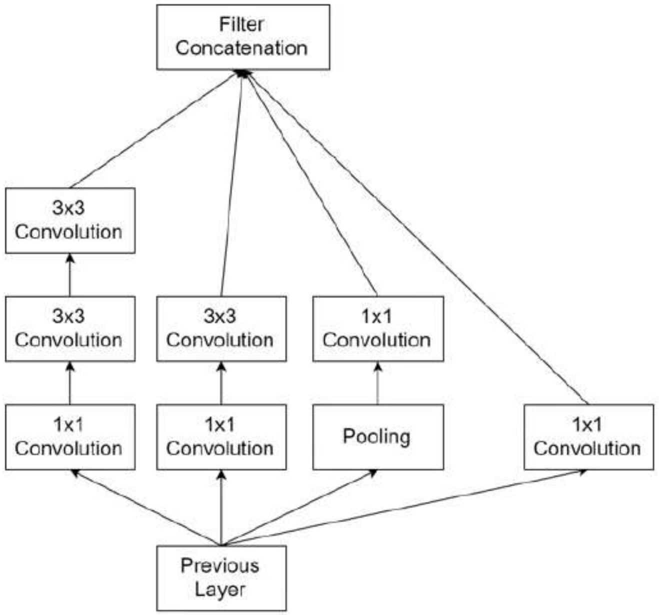
- 3x3 and 1x1 kernels usually work the best

Pointwise (1x1) filters

- channel-wise dense layers
- learn cross-channel features

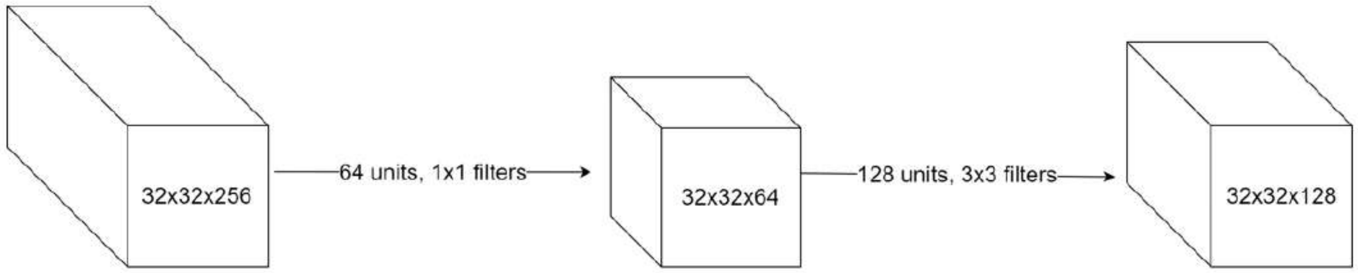


Inception module

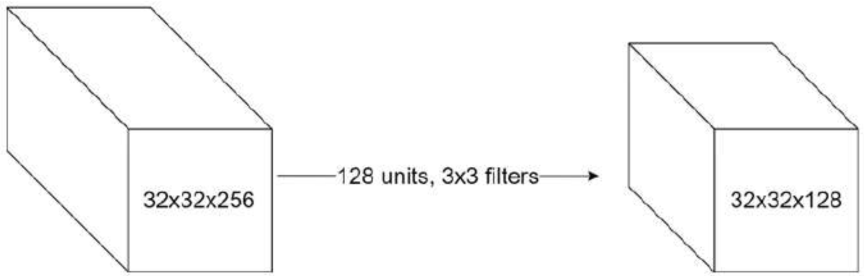


Pointwise (1x1) filters

- Dimensionality reduction



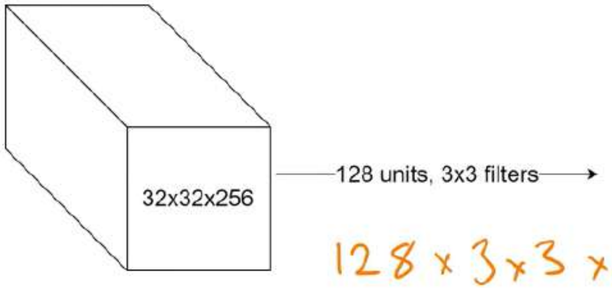
$$32 \times 32 \times 64 \times 1 \times 1 \times 256 \; + \; 32 \times 32 \times 128 \times 3 \times 3 \times 64 \approx 92\text{ M}$$



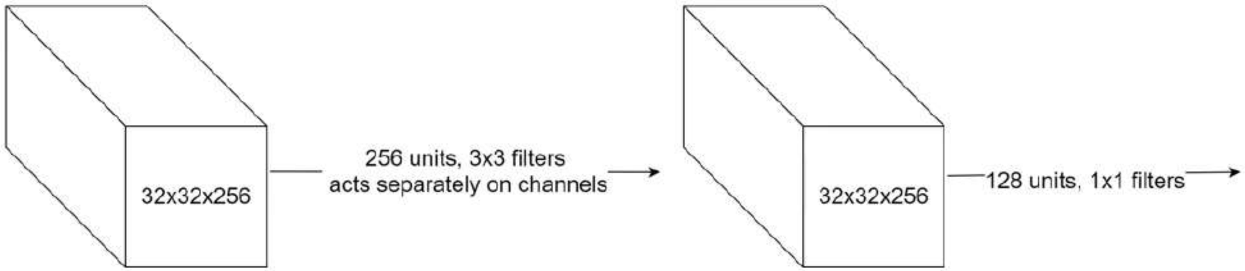
$$32 \times 32 \times 128 \times 3 \times 3 \times 256 \approx 300\text{ M}$$

Pointwise (1x1) filters

- Depthwise separable convolution



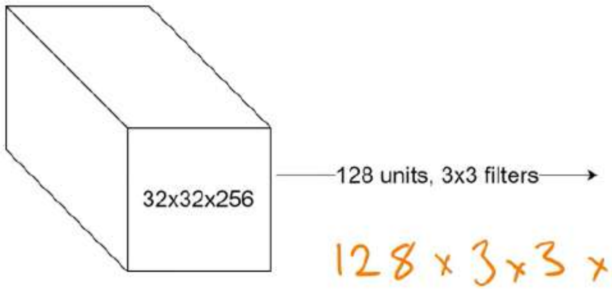
$128 \times 3 \times 3 \times 256 = 294,912$



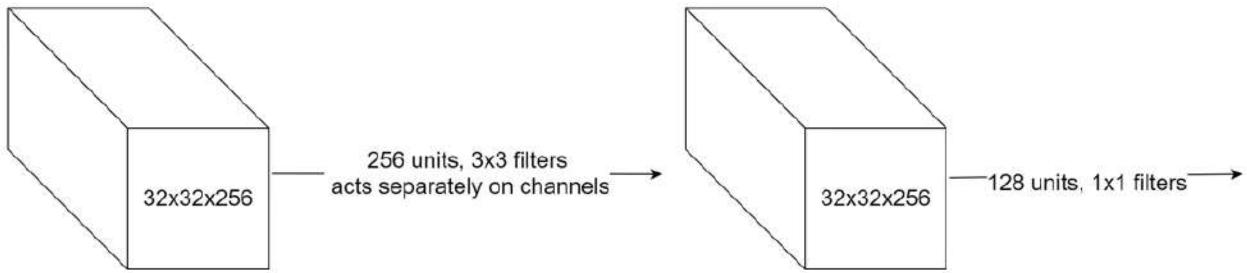
$256 \times 3 \times 3 \times 1$

Pointwise (1x1) filters

- Depthwise separable convolution

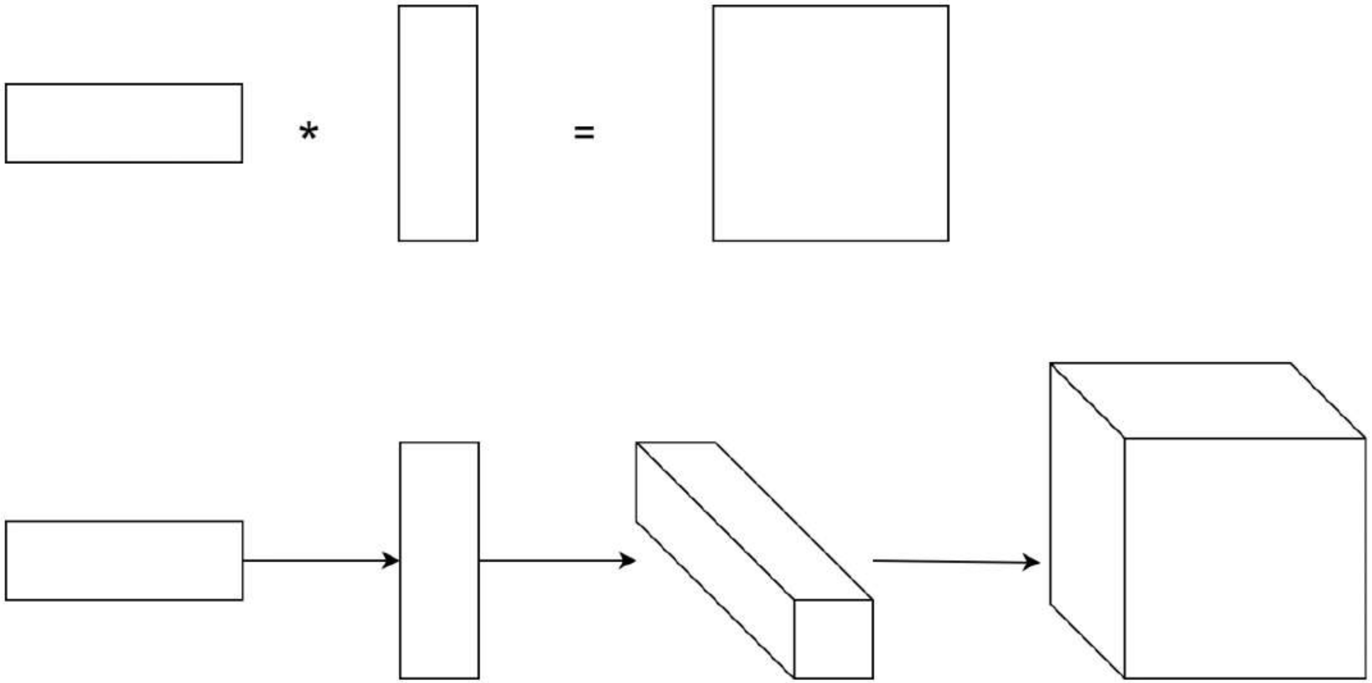


$128 \times 3 \times 3 \times 256 = 294,912$



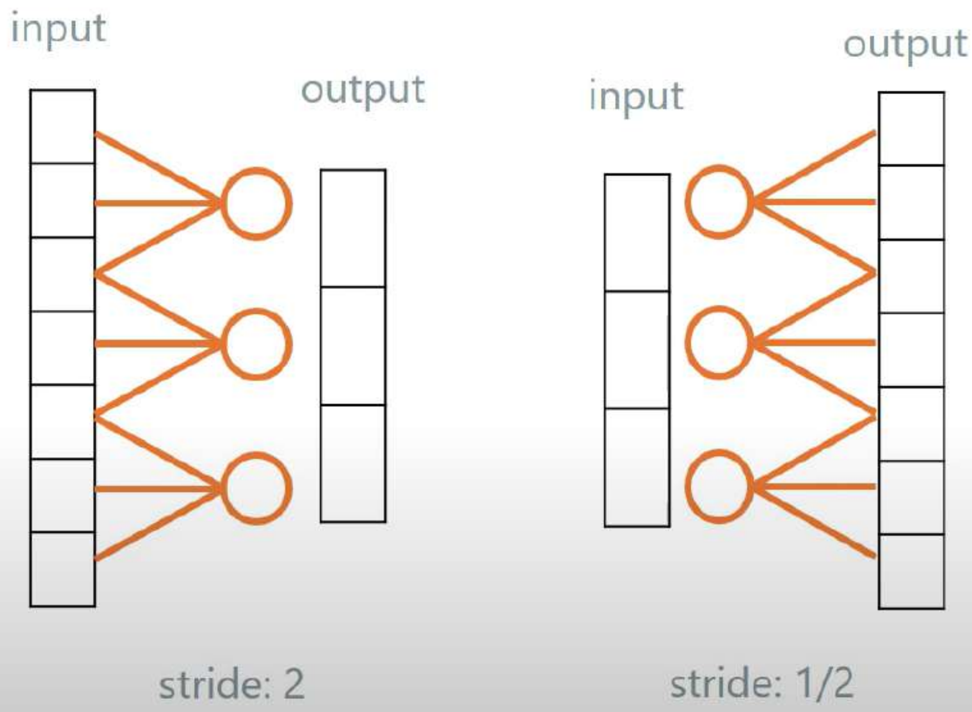
$256 \times 3 \times 3 \times 1 \quad + \quad 128 \times 1 \times 1 \times 256 = 35,072$

Separable convolution



How to choose stride?

- Stride 1: preserve spatial resolution
- Stride 2: downsample
- Fractional stride (1/2): upsample
  - transposed convolution
  - deconvolution



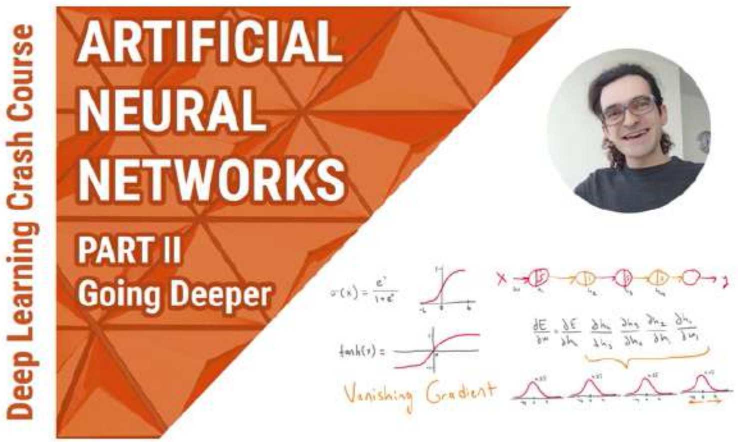
How to choose pooling parameters?

- A very common setting
  - max pooling
  - pooling size: 2x2
  - same padding

- Global average pooling / pooling to fixed size
  - handles variable-sized inputs

How to choose activation functions?

- Short answer: choose ReLU
- For more information:



What type of regularization to use?

- Short answer: use L2 weight decay and dropout
- For more information:

How to choose the batch size?

- Image recognition tasks
  - batch size: 32
- Noisy gradient?
  - use larger batches
- Stuck in local minima? Out of memory?
  - use smaller batches

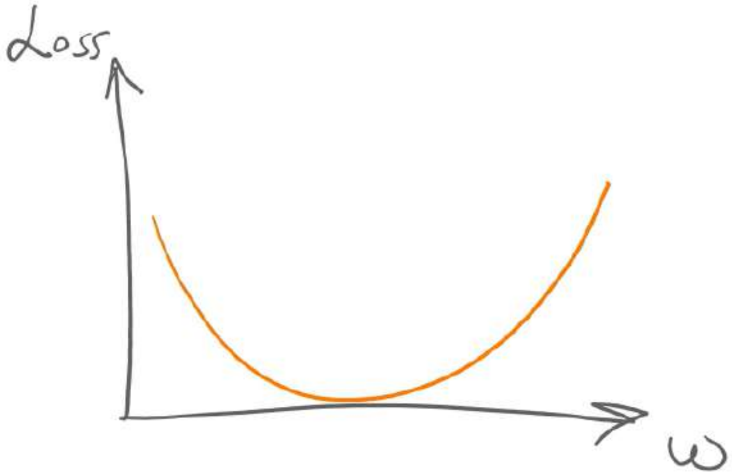
Optimization

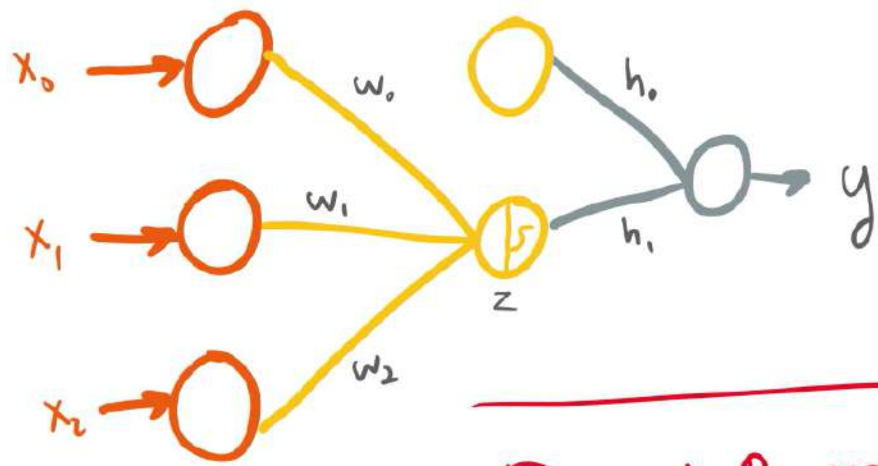
What we want to achieve:
Maximize accuracy = (#correct samples) / (#all samples)
on a test set

What we actually do:
Minimize cross entropy = $\sum p \log(q)$

True class probabilities   Predicted class probabilities

Vanilla Stochastic Gradient Descent





Forward Propagation

Backpropagation

$$w \leftarrow w - \alpha \frac{\partial E}{\partial w}$$

new weights $\leftarrow$ (old weights) − (learning rate)(gradient)

Momentum

(new weights) <- (old weights) – (learning rate) (gradient)

Momentum

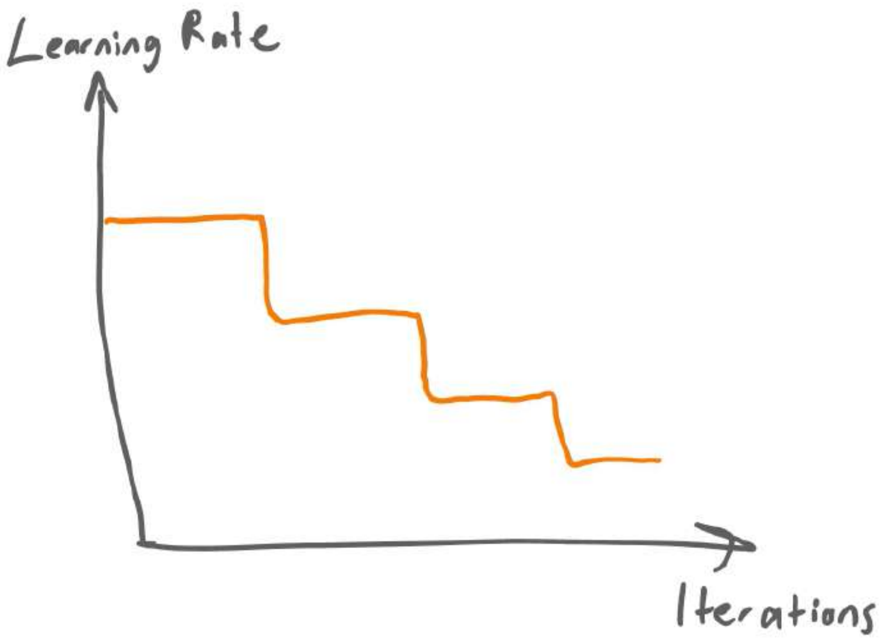
(new weights) <- (old weights) – (learning rate) (gradient)
+ past gradients

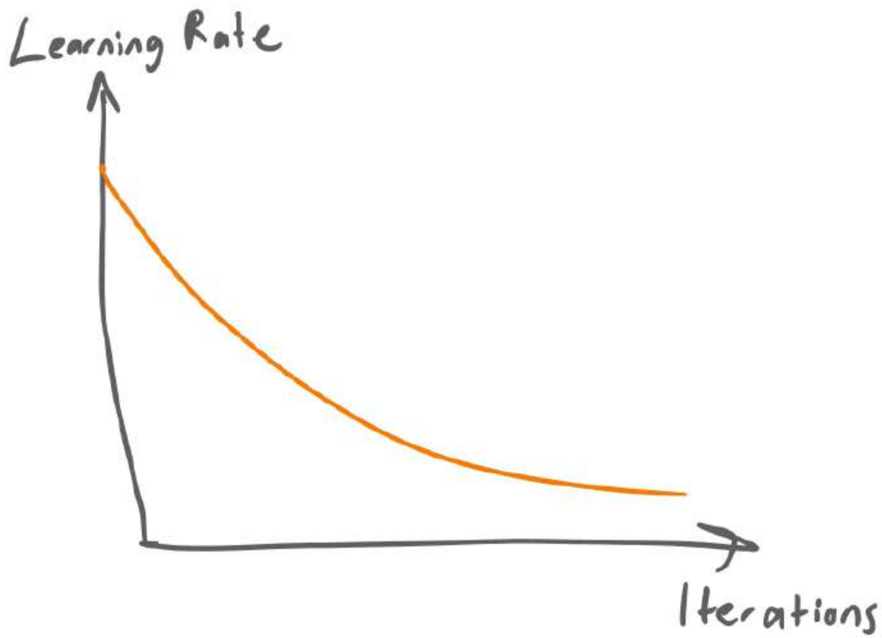
(accumulator) <- (old accumulator) (momentum) + (gradient)

(new weights) <- (old weights) – (learning rate) (accumulator)

Learning rate schedules

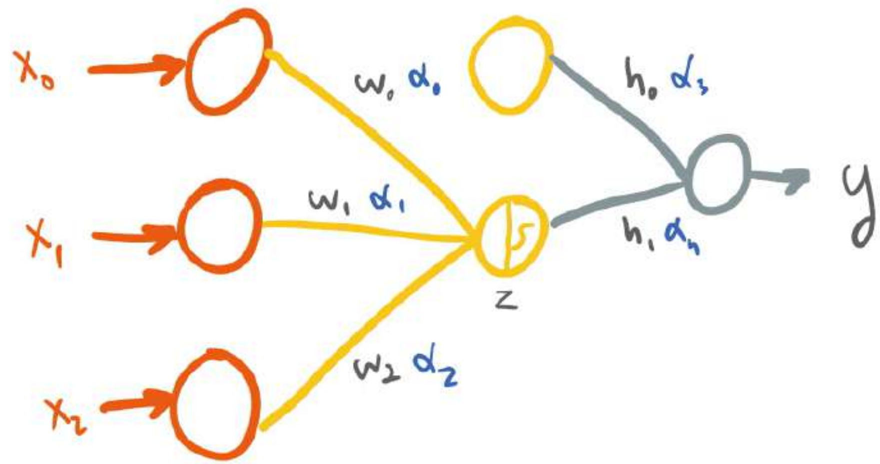
(new weights) <- (old weights) – (learning rate) (gradient)



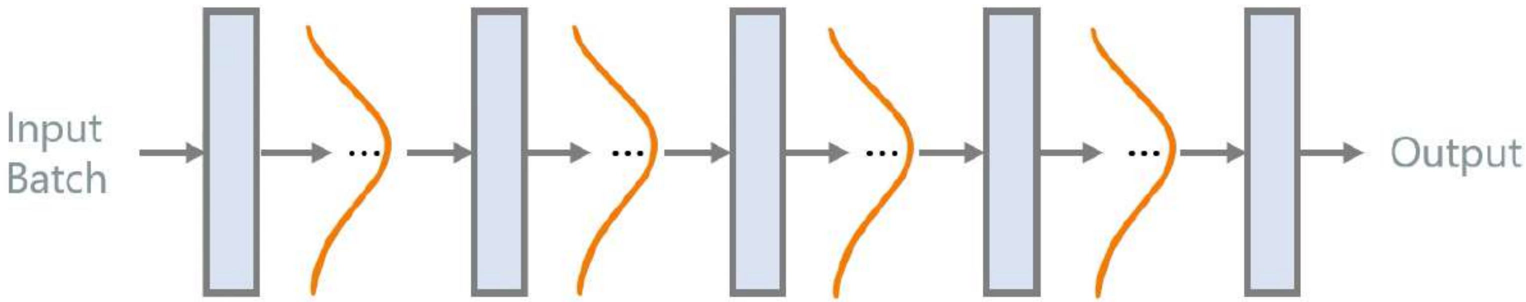


Adaptive Methods

- AdaGrad
  - Large gradient: decrease α faster
  - Small gradient: decrease α slower
- RMSProp
  - Moving average of gradients
- Adam
  - Adaptive Moment Estimation
  - RMSProp + Momentum



Batch Normalization



Subtract batch mean
Divide by standard deviation

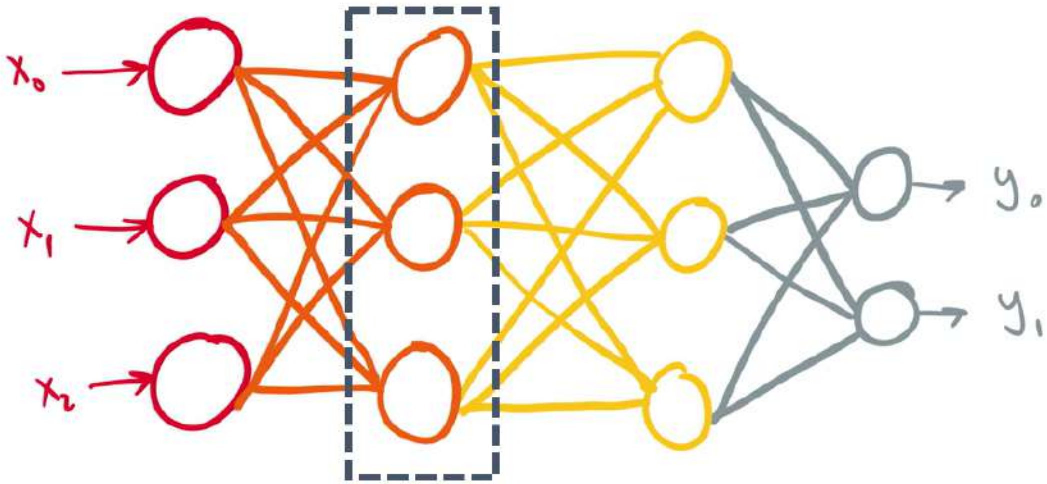
$$h \leftarrow \frac{h - \mu}{\sigma}$$

Scale and shift

$$h \leftarrow \gamma h + \beta$$

Initialization

- Biases
  - Initialize to zero (or another small constant)
- Weights
  - Initialize to zero? No!
  - Break the symmetry
  - Random values

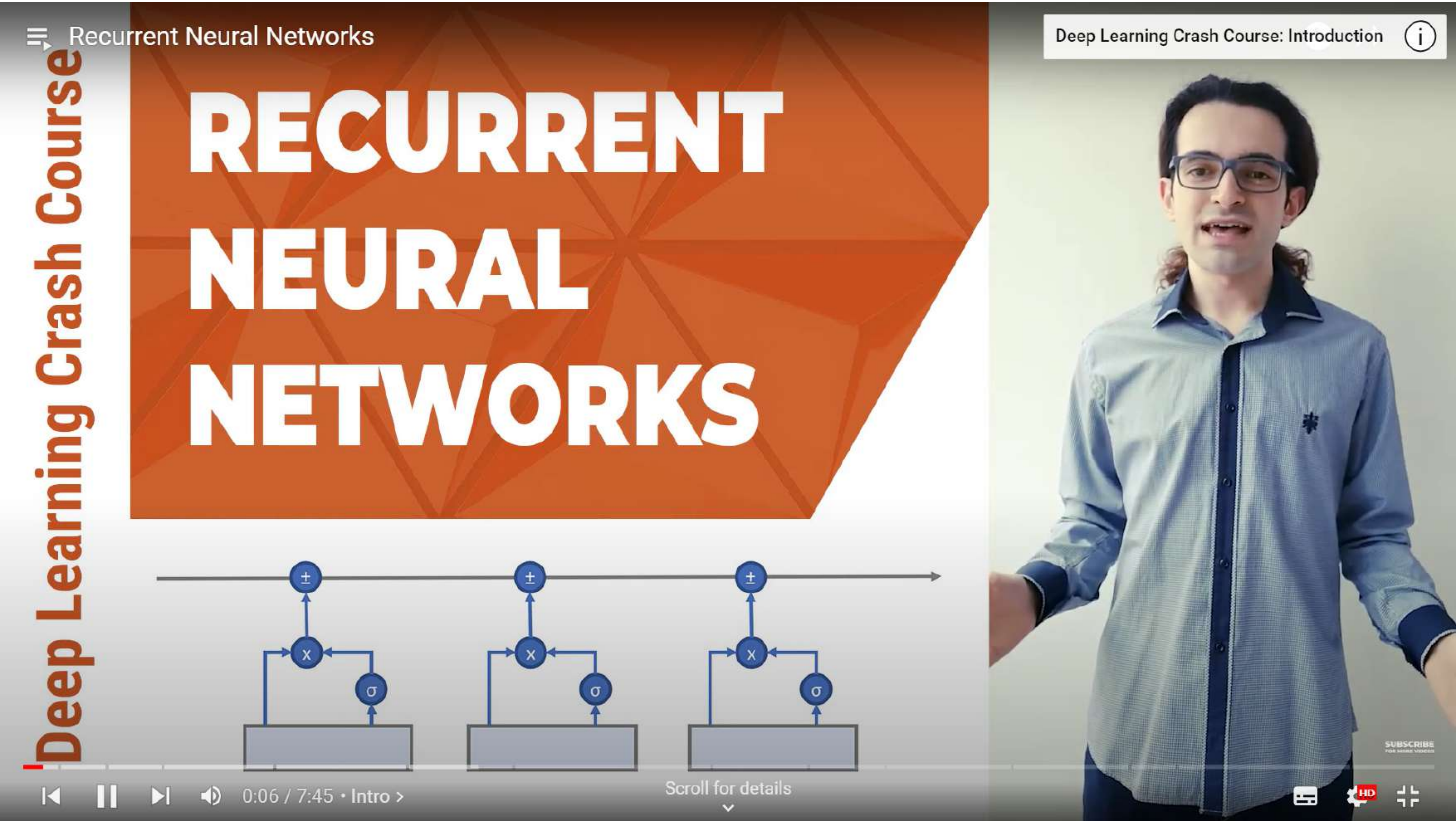


All would get
the same updates

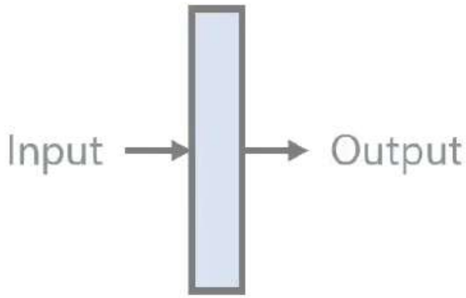
Initialization

- Glorot (Xavier) initializer

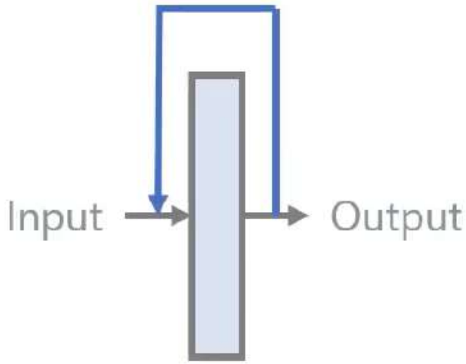
$$W \sim \text{Uniform}(-r, r) \qquad r = \sqrt{\frac{6}{n_{in} + n_{out}}}$$


Recurrent Neural Networks
Deep Learning Crash Course: Introduction
Deep Learning Crash Course
RECURRENT
NEURAL
NETWORKS
±
±
±
x
x
x
σ
σ
σ
0:06 / 7:45 • Intro
Scroll for details

Feedforward vs Recurrent



Feedforward Neural Network



Recurrent Neural Network (RNN)

Sequential Data

- Audio
- Video
- Text
- Biomedical data

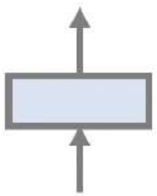
bla bla bla

Types of Inputs and Outputs

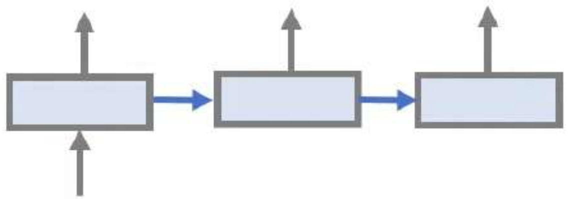
Task	Input	Output
Machine translation	Text	Text
Speech recognition	Audio	Text
Speech synthesis	Text	Audio
Sentiment analysis	Text	Categorical variable
Text generator	Random number	Text

Types of RNNs

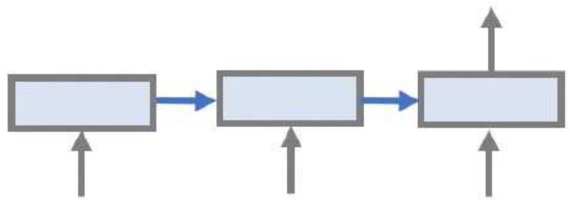
One to one



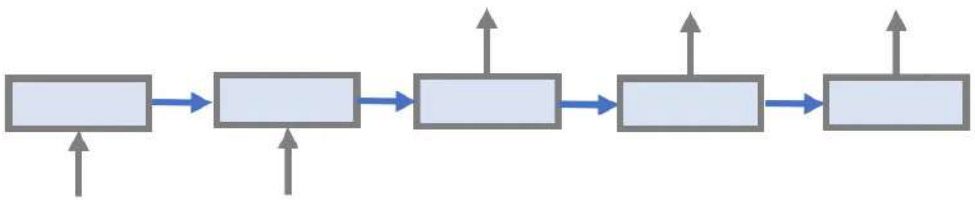
One to many

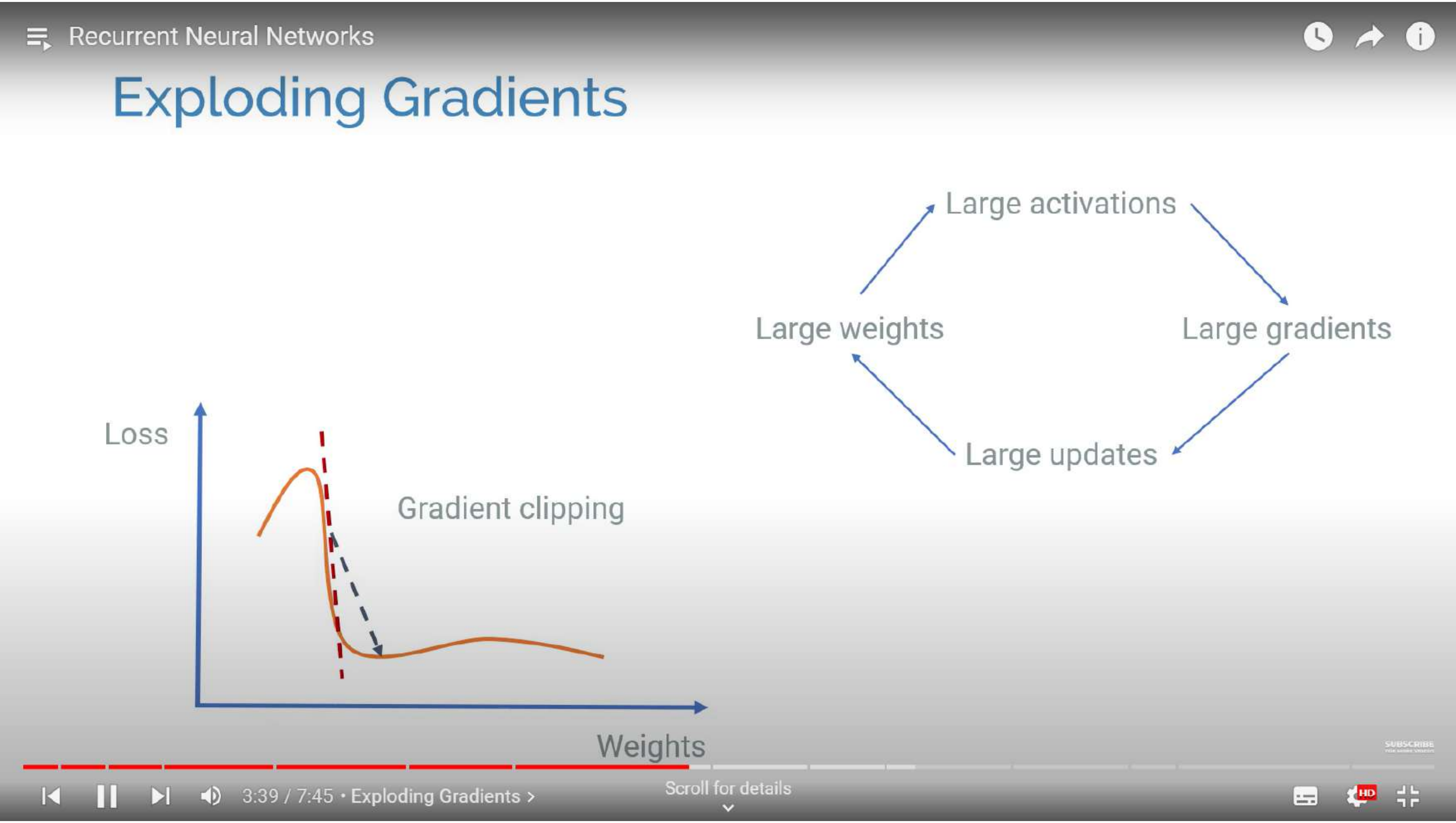


Many to one



Many to many





Recurrent Neural Networks
Exploding Gradients
Large activations
Large weights
Large gradients
Large updates
Loss
Gradient clipping
Weights
3:39 / 7:45 • Exploding Gradients >
Scroll for details

Gated Modules

- LSTM (Long Short Term Memory)
- GRU (Gated Recurrent Unit)