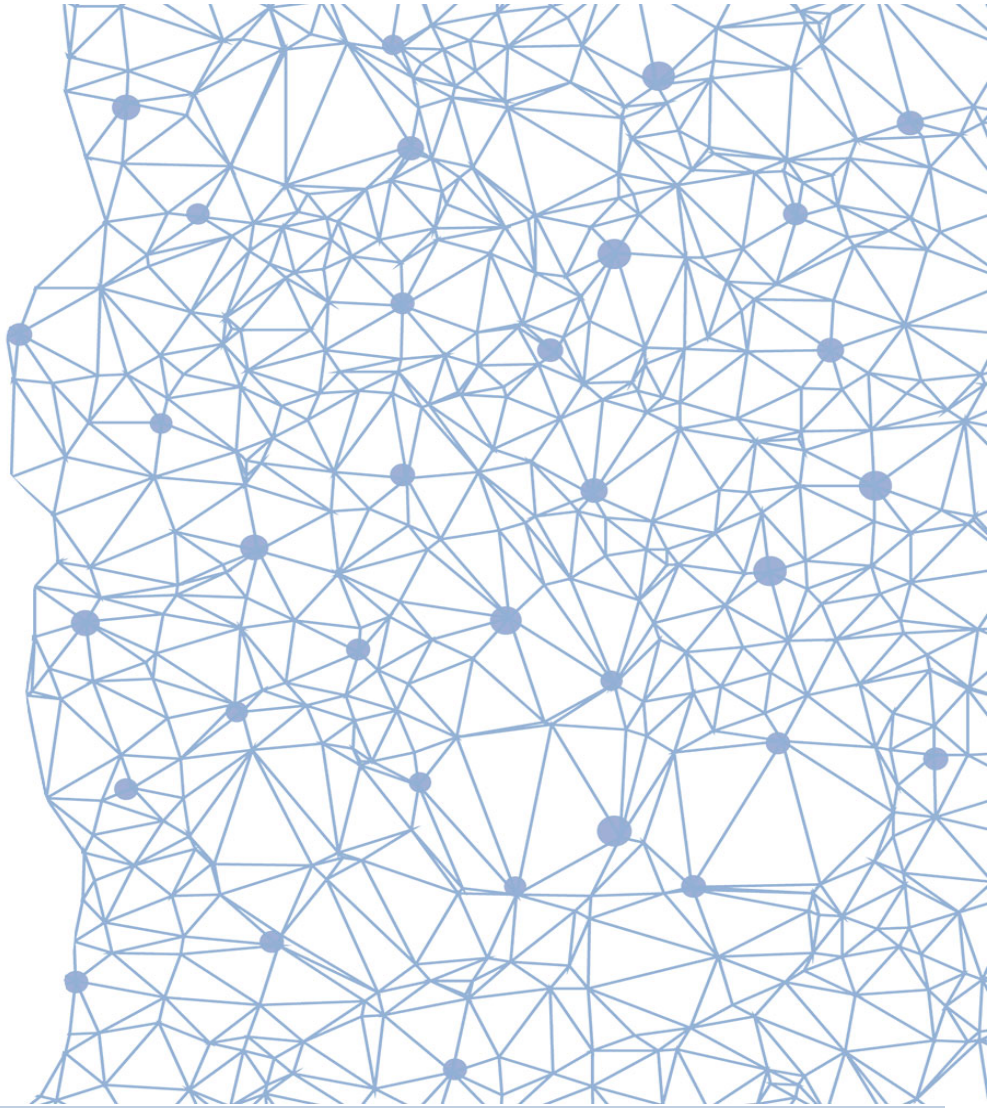
BITVILLE

Deep Learning

Antti Keurulainen Lic.Sc (Tech.)

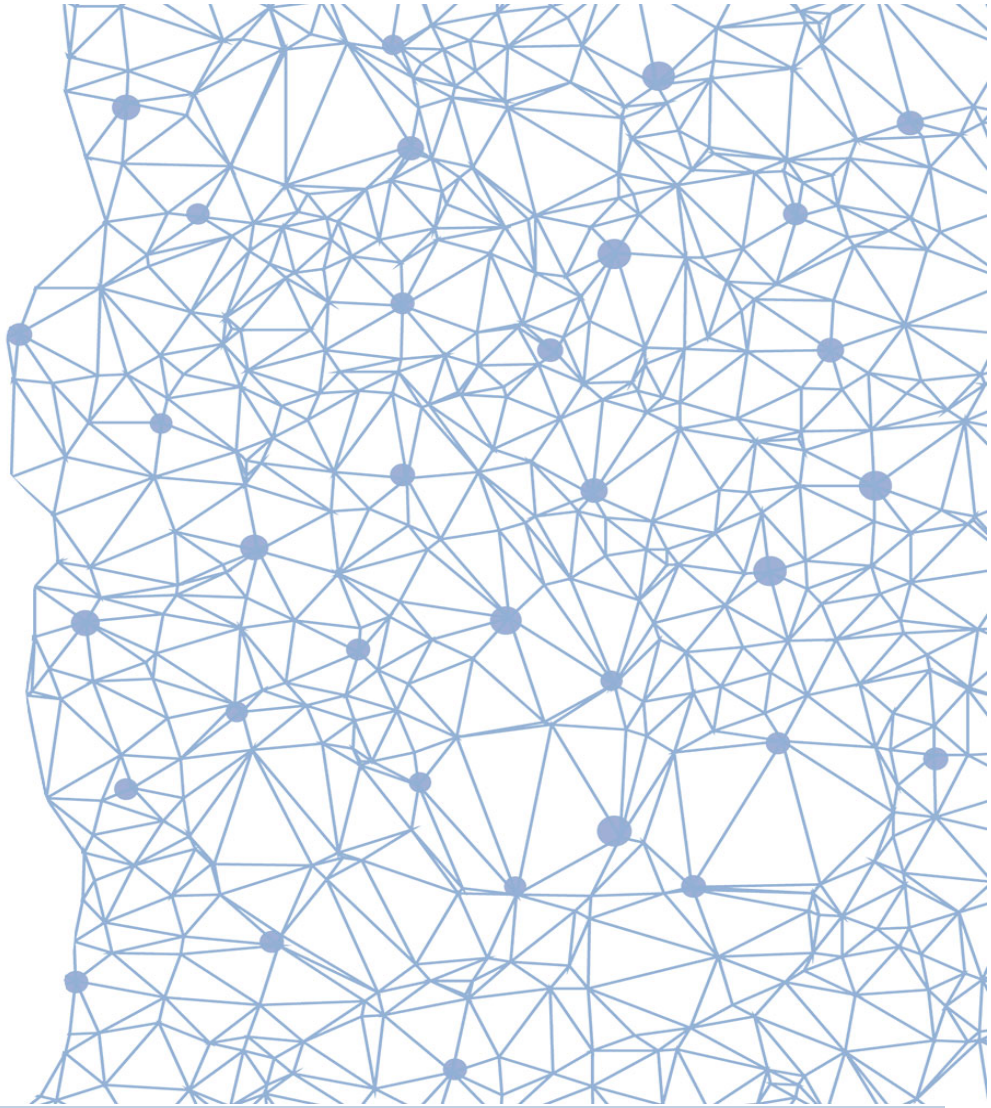
Member in Aalto University Probabilistic Machine Learning research group
Research director at Bitville AI

antti.keurulainen@bitville.com
@AnttiKeurulaine



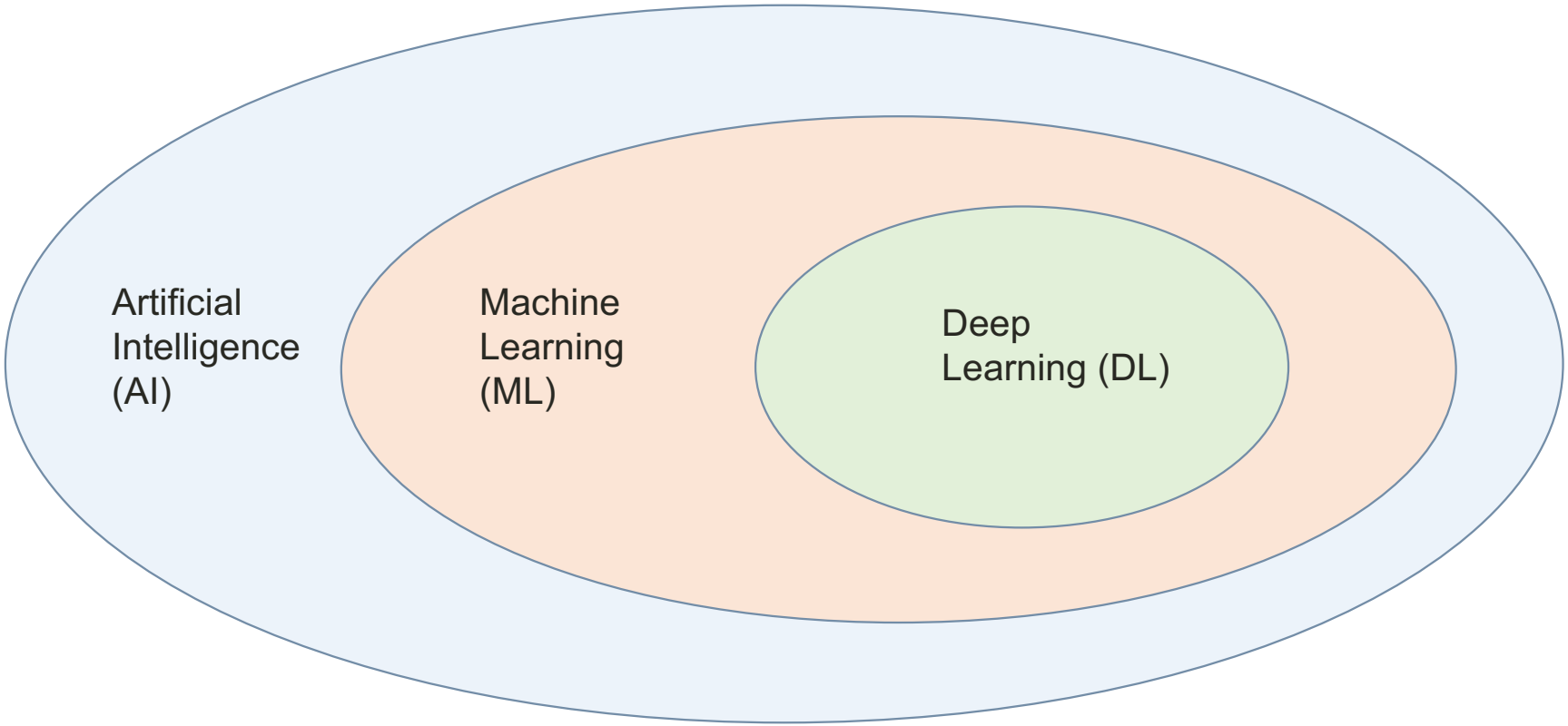
Agenda

- Introduction to Deep Learning
- Success stories and milestones
- Biological neuron and artificial neuron
- Deep neural network
- Convnets
- Recurrent nets
- Demos



Introduction to Deep Learning

Deep learning



Machine intelligence has a long history

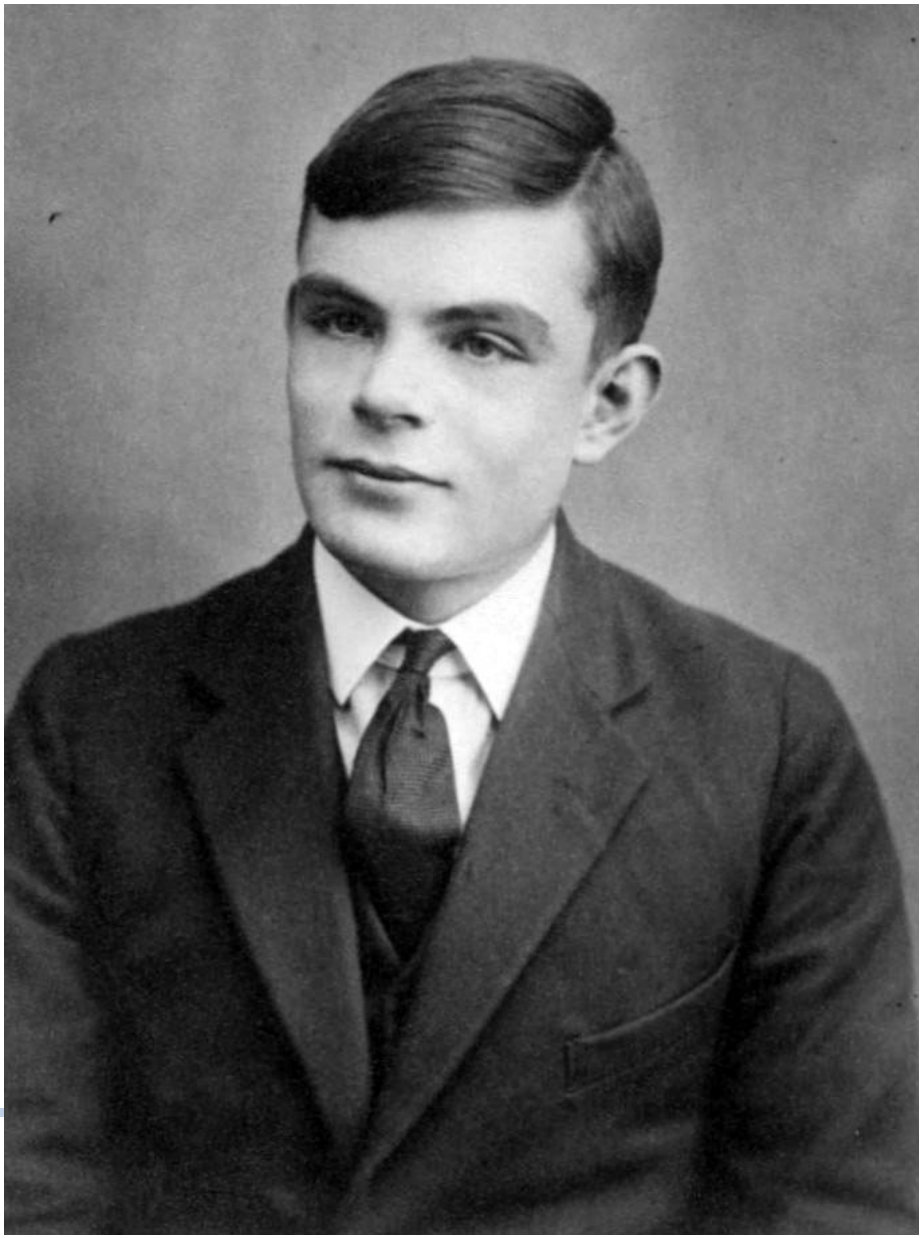
A. M. Turing (1950) Computing Machinery and Intelligence. ***Mind 49:*** **433-460.**

COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

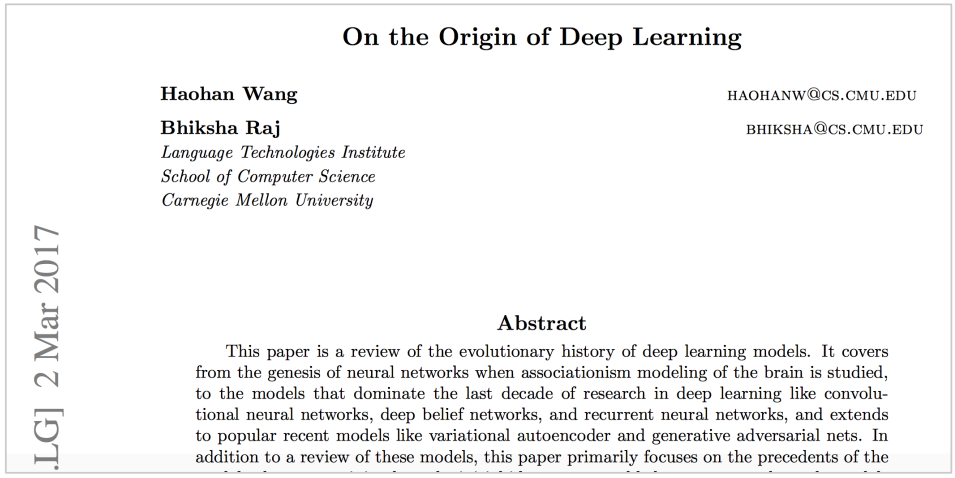
1. The Imitation Game

I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think." The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous, If the meaning of the words "machine" and "think" are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, "Can machines think?" is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a



Wang & Raj (Carnegie Mellon University 2017)

On the Origin of Deep Learning



On the Origin of Deep Learning

Haohan Wang HAOHANW@CS.CMU.EDU

Bhiksha Raj BHIKSHA@CS.CMU.EDU

Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Abstract

This paper is a review of the evolutionary history of deep learning models. It covers from the genesis of neural networks when associationism modeling of the brain is studied, to the models that dominate the last decade of research in deep learning like convolutional neural networks, deep belief networks, and recurrent neural networks, and extends to popular recent models like variational autoencoder and generative adversarial nets. In addition to a review of these models, this paper primarily focuses on the precedents of the

Table 1: Major milestones that will be covered in this paper

Year	Contributer	Contribution
300 BC	Aristotle	introduced Associationism, started the history of human's attempt to understand brain.
1873	Alexander Bain	introduced Neural Groupings as the earliest models of neural network, inspired Hebbian Learning Rule.
1943	McCulloch & Pitts	introduced MCP Model, which is considered as the ancestor of Artificial Neural Model.
1949	Donald Hebb	considered as the father of neural networks, introduced Hebbian Learning Rule, which lays the foundation of modern neural network.
1958	Frank Rosenblatt	introduced the first perceptron, which highly resembles modern perceptron.
1974	Paul Werbos	introduced Backpropagation
1980	Teuvo Kohonen	introduced Self Organizing Map
	Kunihiko Fukushima	introduced Neocogitron, which inspired Convolutional Neural Network
1982	John Hopfield	introduced Hopfield Network
1985	Hilton & Sejnowski	introduced Boltzmann Machine
1986	Paul Smolensky	introduced Harmonium, which is later known as Restricted Boltzmann Machine
	Michael I. Jordan	defined and introduced Recurrent Neural Network
1990	Yann LeCun	introduced LeNet, showed the possibility of deep neural networks in practice
1997	Schuster & Paliwal	introduced Bidirectional Recurrent Neural Network
	Hochreiter & Schmidhuber	introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks
2006	Geoffrey Hinton	introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.
2009	Salakhutdinov & Hinton	introduced Deep Boltzmann Machines
2012	Geoffrey Hinton	introduced Dropout, an efficient way of training neural networks

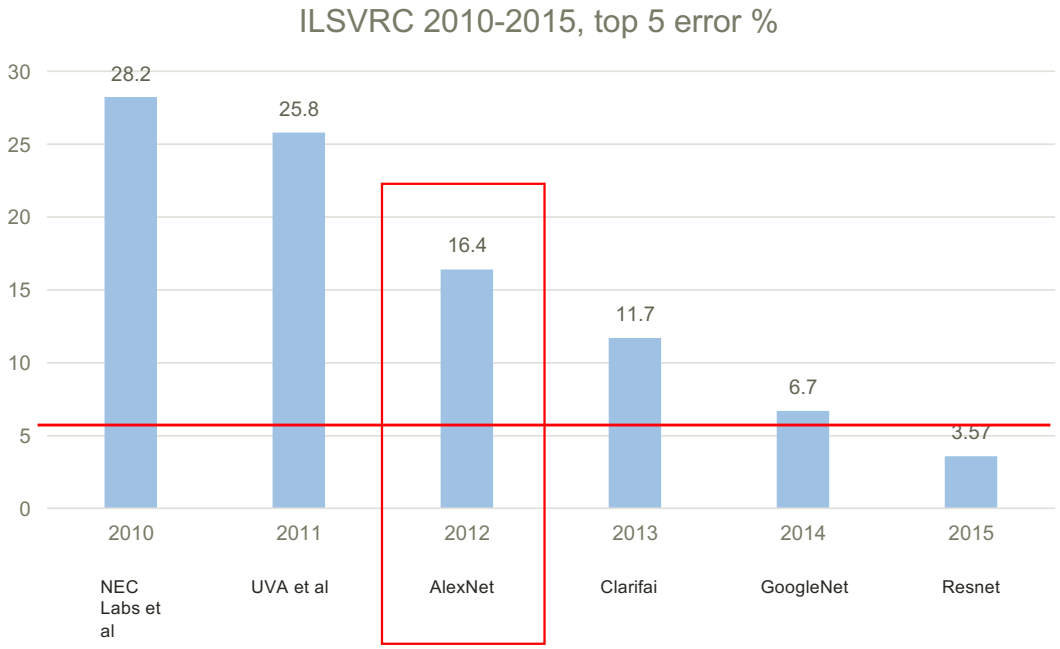
Why is now the time for machine intelligence?

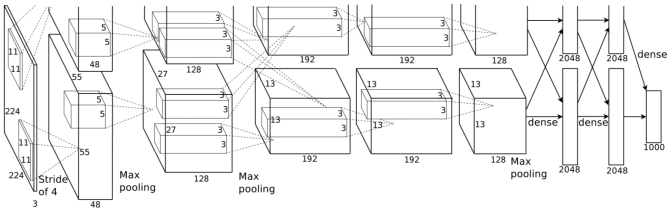
- Lot of data available (big data)
- Lot of computing power (cloud computing, affordable GPUs, …)
- New powerful libraries (Python packages, Tensorflow, Pytorch, Keras, …)
- New algorithms and advances in machine learning

Success stories and milestones

Deep Learning examples – computer vision

Error rate in ImageNet Large Scale Visual Recognition Challenge 2010->





AlexNet architecture

A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Human accuracy

Examples of successful Deep Learning implementations

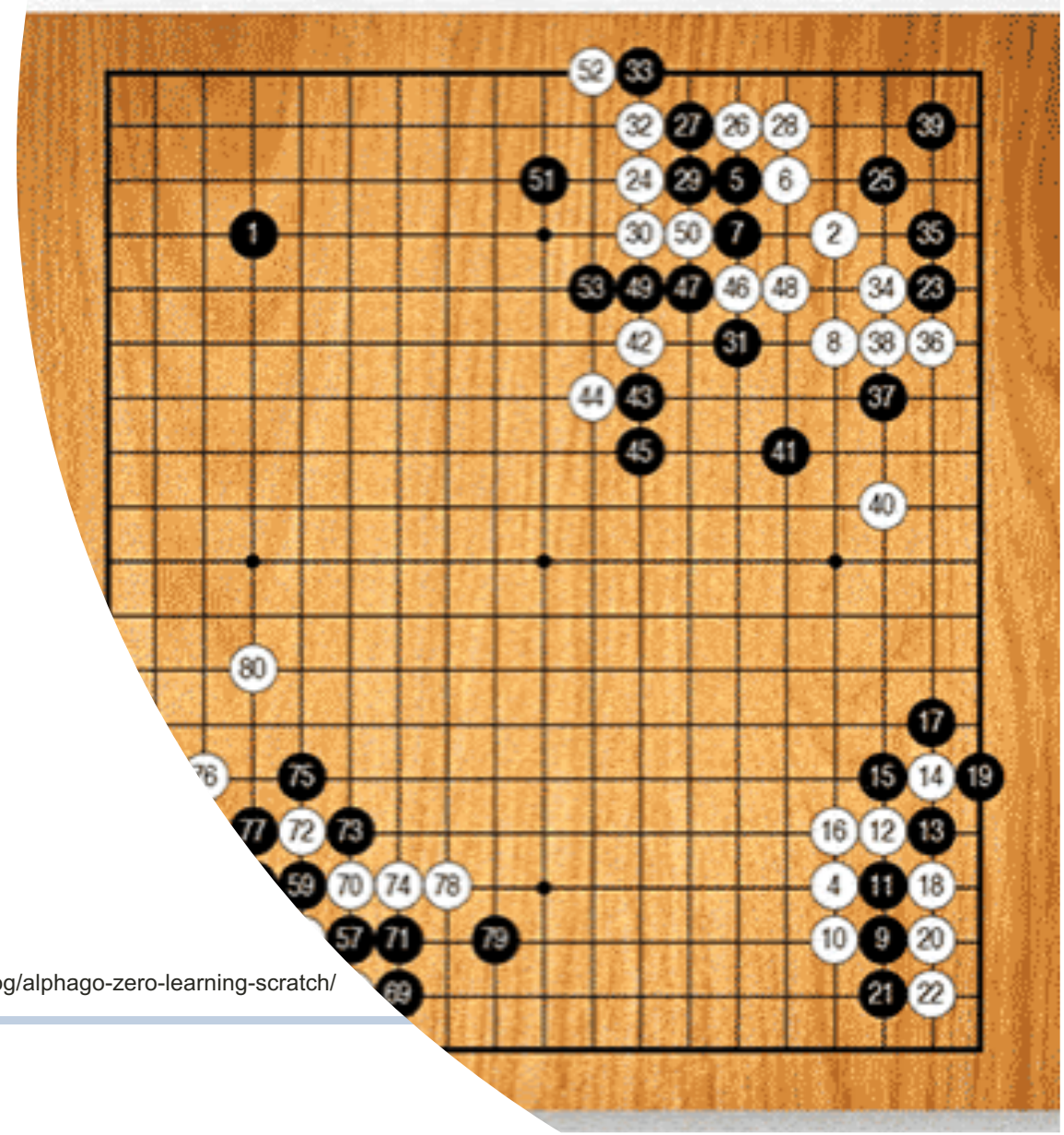
- Computer vision
- Natural Language Processing (NLP)
- Robotics
- Self driving cars
- Personal assistants
- Generative tasks (images, audio,…)
- Anomaly detection
- Fraud detection
- Game AI
- Denoising
- etc

Deep Learning examples – self driving cars

- Sensor and camera data fed into a deep neural network
- The deep neural networks are able to recognize objects such as other cars, pedestrians, etc

Deep Learning examples – AlphaGo Zero

- No human input, learns by self-play
- Combination of reinforcement learning and deep learning (deep reinforcement learning)



https://deepmind.com/blog/alphago-zero-learning-scratch/

Deep Learning examples – image generation

T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018

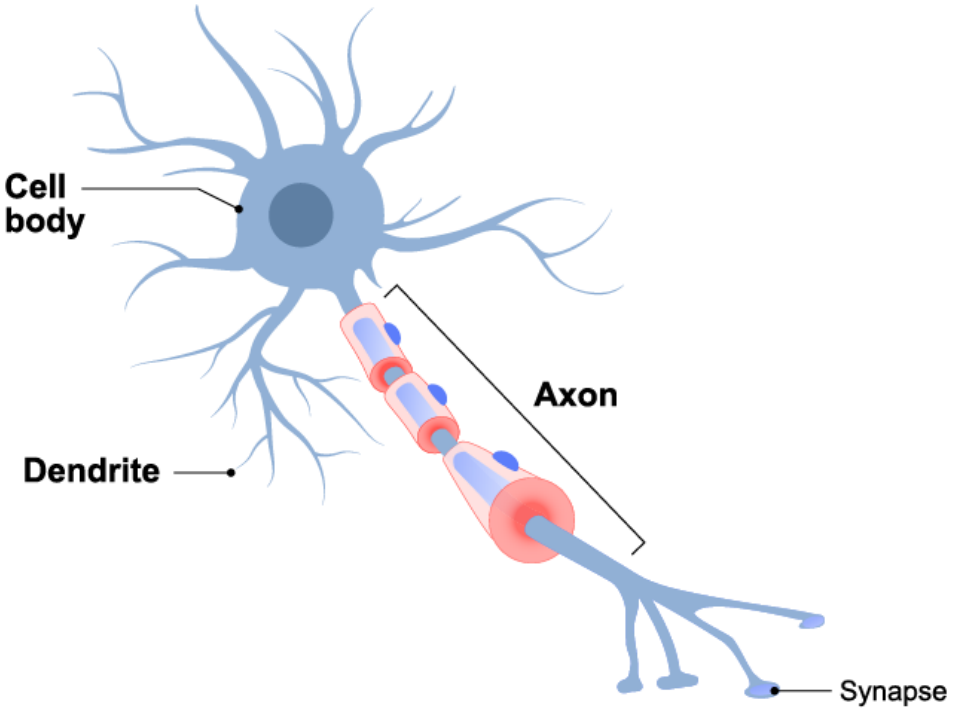
- Can generate high quality images by using generative adversarial networks

https://research.nvidia.com/publication/2017-10_Progressive-Growing-of

Biological neuron and artificial neuron

Biological Neuron

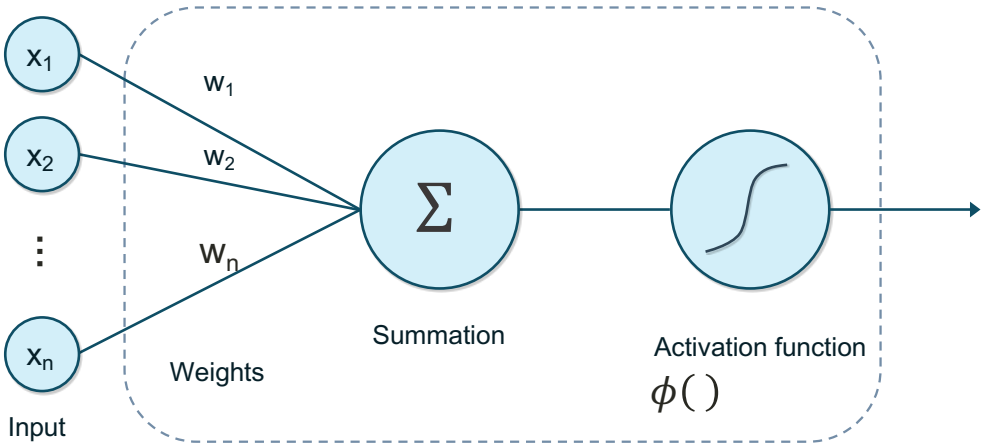
- Human brain consist of around 85 billion neurons
- Each neuron has up to 50 000 connections depending on the neuron type



Artificial Neuron

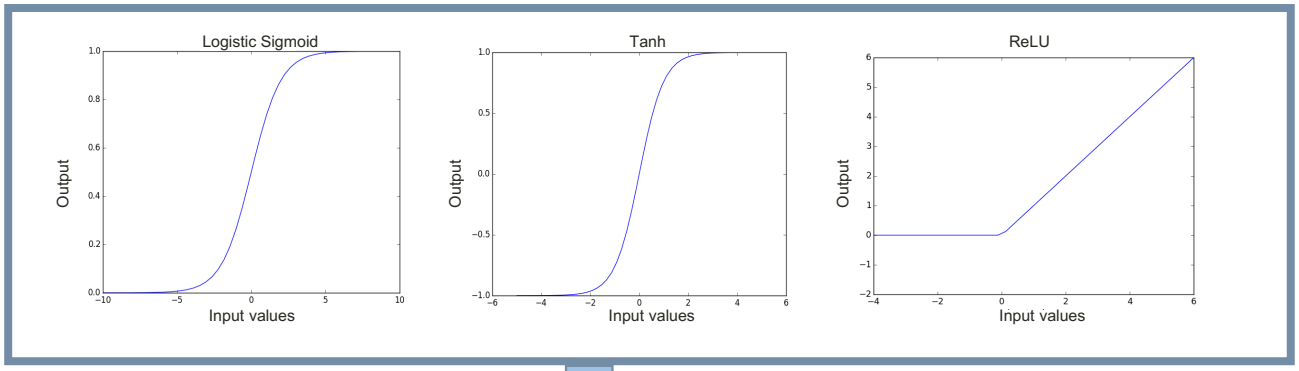
feature vector

$\boldsymbol{x} = x_1, \ldots, x_n$



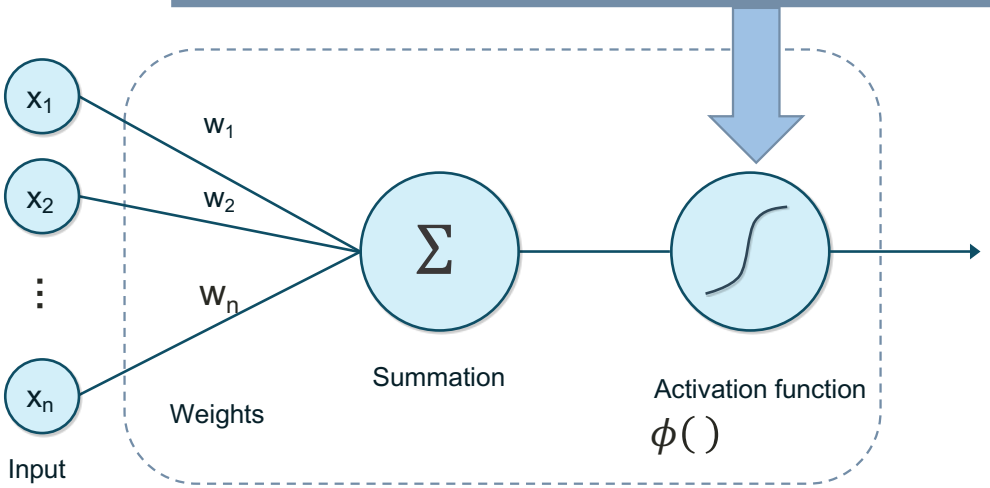
$$h(\boldsymbol{x}; \boldsymbol{w}) = \phi\left(\sum_{i=1}^{n} w_i x_i\right)$$

Artificial neuron



feature vector

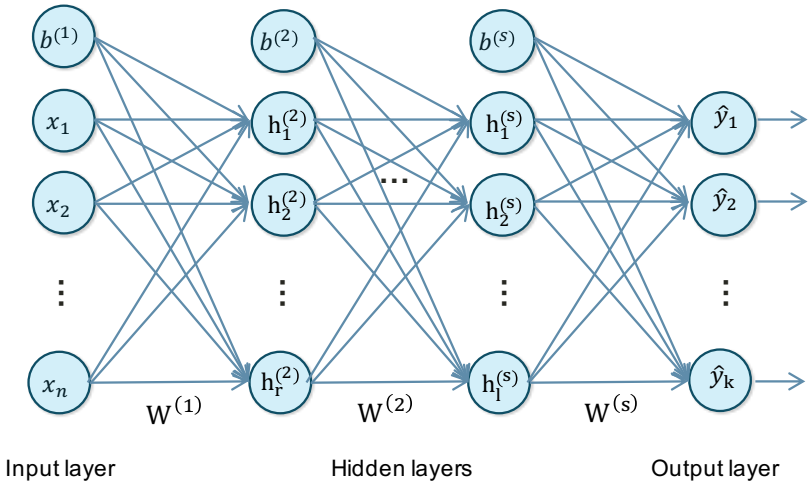
$\boldsymbol{x} = x_1, \ldots, x_n$

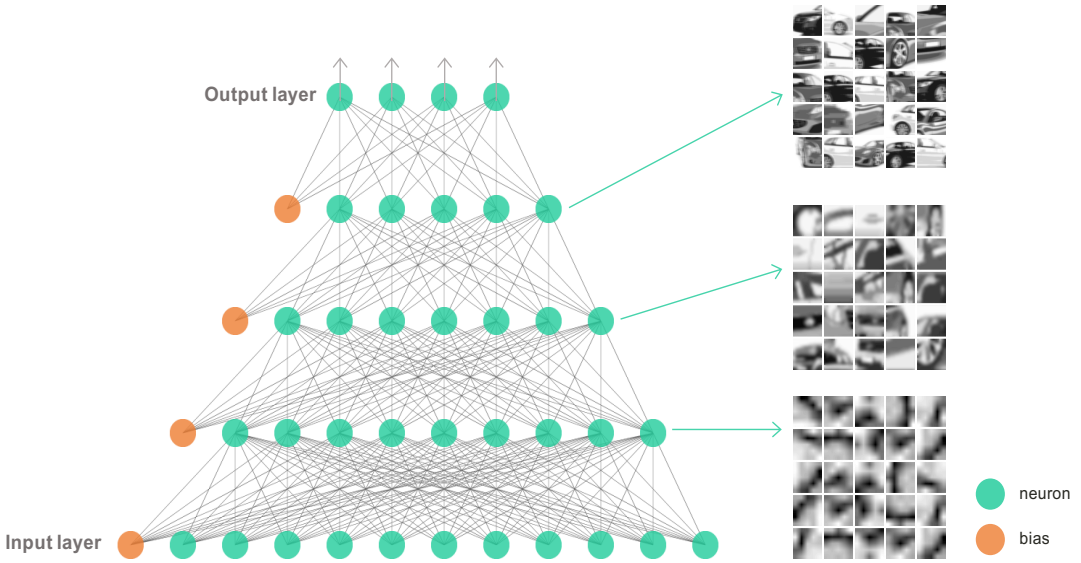


$$h(\boldsymbol{x}; \boldsymbol{w}) = \phi\left(\sum_{i=1}^{n} w_i x_i\right)$$

Multilayer Perceptron

Multilayer perceptron





In theory, a feed-forward network with a single hidden layer containing a sufficiently (but finite) number of neurons can approximate any function up to any desired accuracy.

Deep Neural Network optimization

Simple iteration example recipe for supervised learning:

1. Feed in sample(s) and calculate the network output (forward pass)
2. Measure how well the network estimates the desired value(s) (cost)
3. Calculate the partial derivatives of the cost with respect to the parameters (backpropagation)
4. Update the parameters (gradient descent)

Most of this procedure can be automated with the powerful DL libraries (TensorFlow, Pytorch etc)

Forward pass

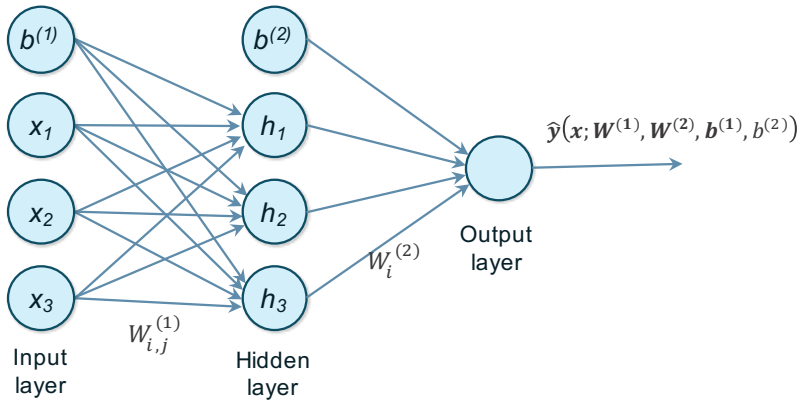
The forward pass is performed by calculating the responses of the network elements layer by layer, starting from the input layer, and progressing towards the output of the network.

$$h_1 = g\left(b_1^{(1)} + W_{1,1}^{(1)}\, x_1 + W_{1,2}^{(1)}\, x_2 + W_{1,3}^{(1)}\, x_3\right)$$

$$h_2 = g\left(b_2^{(1)} + W_{2,1}^{(1)}\, x_1 + W_{2,2}^{(1)}\, x_2 + W_{2,3}^{(1)}\, x_3\right)$$

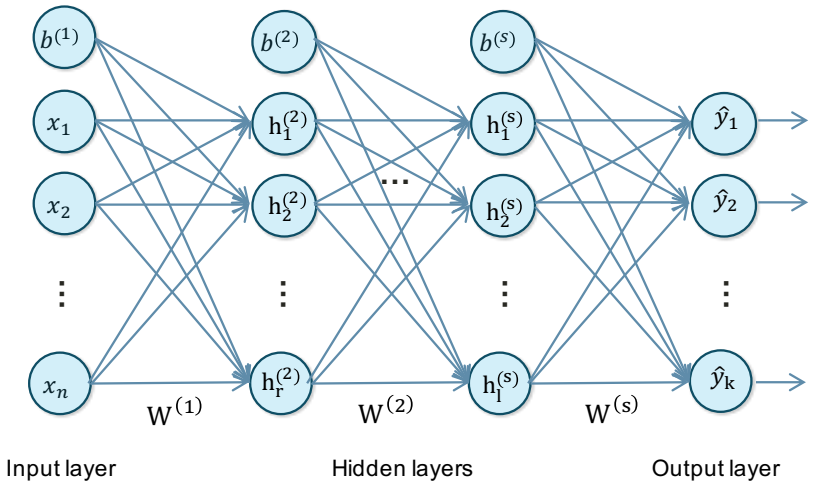
$$h_3 = g\left(b_3^{(1)} + W_{3,1}^{(1)}\, x_1 + W_{3,2}^{(1)}\, x_2 + W_{3,3}^{(1)}\, x_3\right)$$

$$\hat{y}\left(\boldsymbol{x};\, \boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(1)}, b^{(2)}\right) = g\left(b_1^{(2)} + W_1^{(2)}\, h_1 + W_2^{(2)}\, h_2 + W_3^{(2)}\, h_3\right)$$



$\boldsymbol{W}$ is the weight matrix, $\boldsymbol{b}$ is the bias vector, $\boldsymbol{h}$ is a vector consisting the neuron outputs, $\hat{y}$ is the output of the neural network, g is the activation function

Cost function



$$\hat{\boldsymbol{y}} = \boldsymbol{h}\left(\boldsymbol{x};\, \boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots, \boldsymbol{W}^{(s)}, \boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, \ldots, \boldsymbol{b}^{(s)}\right)$$

$$\boldsymbol{W}^{(1)} = \begin{pmatrix} W_{1,1}^{(1)} & \cdots & W_{1,j}^{(1)} \\ \vdots & \ddots & \vdots \\ W_{i,1}^{(1)} & \cdots & W_{i,j}^{(1)} \end{pmatrix}$$

Mean squared error cost (regression tasks):

$$J(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{m}\sum_{i=1}^{m}(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2$$

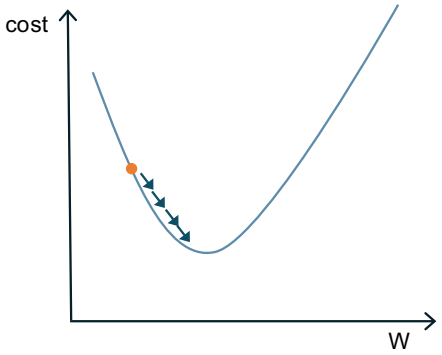
Cross entropy cost or negative log-likelihood cost (classification tasks):

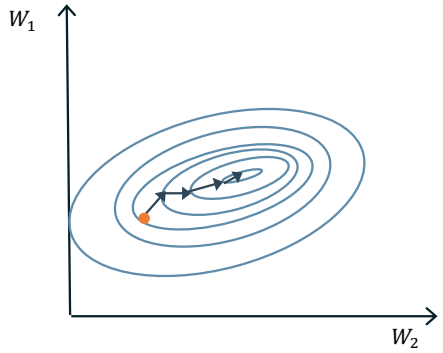
$$J(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}\boldsymbol{y}_{i,j}\log\hat{\boldsymbol{y}}_{i,j}$$

Gradient descent

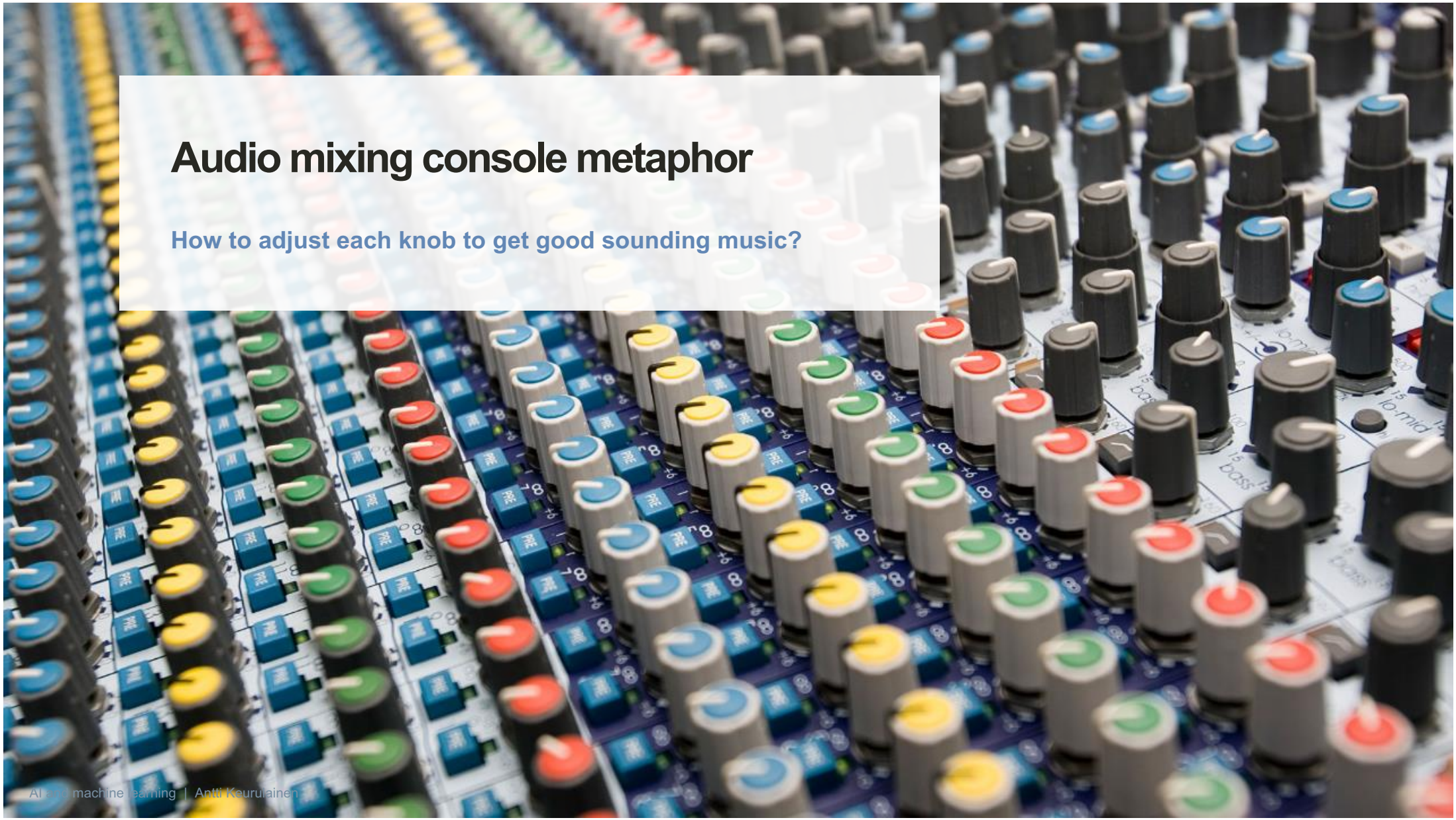
The gradient descent algorithm offers a method to adjust the parameters towards the assumed correct values iteratively by a small step after certain computations.

$$W_j := W_j - \varepsilon \frac{\partial J(\boldsymbol{x}; \boldsymbol{W})}{\partial W_j}$$



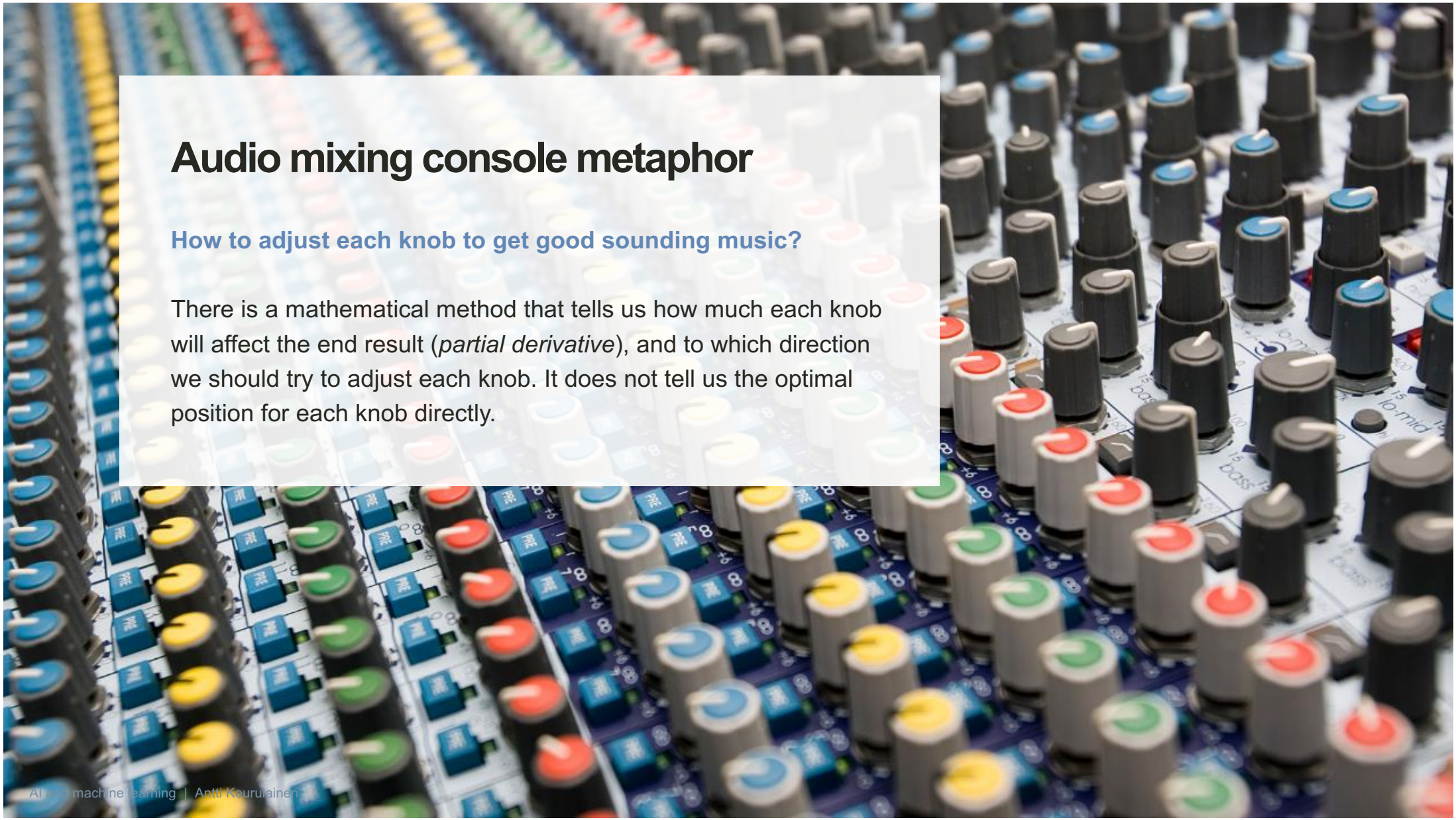


ε is called the “learning rate”



Audio mixing console metaphor

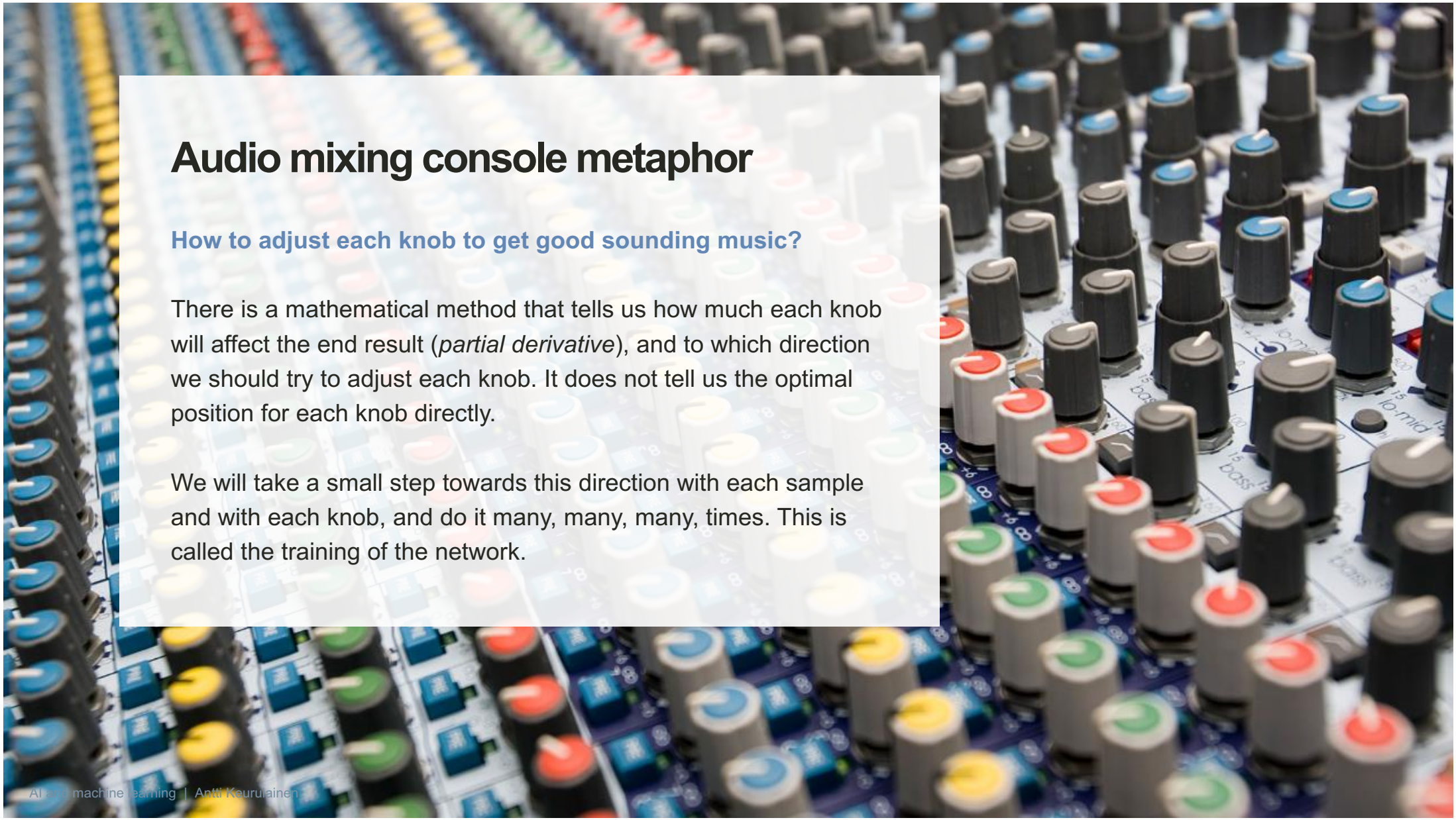
How to adjust each knob to get good sounding music?



Audio mixing console metaphor

How to adjust each knob to get good sounding music?

There is a mathematical method that tells us how much each knob will affect the end result (*partial derivative*), and to which direction we should try to adjust each knob. It does not tell us the optimal position for each knob directly.



Audio mixing console metaphor

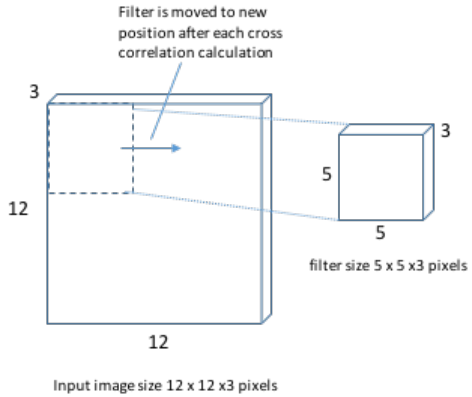
How to adjust each knob to get good sounding music?

There is a mathematical method that tells us how much each knob will affect the end result (*partial derivative*), and to which direction we should try to adjust each knob. It does not tell us the optimal position for each knob directly.

We will take a small step towards this direction with each sample and with each knob, and do it many, many, many, times. This is called the training of the network.

Convolutions Neural Networks (“Convnets”)

Convolutional Networks



$$\mathrm{h}(x;\boldsymbol{W}) = \phi\left(\sum_{k=1}^{3}\sum_{i=1}^{5}\sum_{j=1}^{5} x_{kij}W_{kij}\right)$$

- After convolution, some other operations are performed such as applying the activation function (nonlinearity) and pooling
- During training, the values that are used in the filters are updated and gradually learned
- The parameter sharing concept brings invariance

Convolutional Networks

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

Filter (Kernel)

0.2	0.7
-0.5	0.7

Convolutional Networks

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

$$y_1 = 0.2 * 1 + 0.7 * 3 - 0.5 * 6 + 0.7 * 0 = -0.7$$

Filter (Kernel)

0.2	0.7
-0.5	0.7

Convolutional Networks

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

$$y_1 = 0.2 * 1 + 0.7 * 3 - 0.5 * 6 + 0.7 * 0 = -0.7$$

Feature map

-0.7

Filter (Kernel)

0.2	0.7
-0.5	0.7

Convolutional Networks

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

$$y_1 = 0.2 * 1 + 0.7 * 3 - 0.5 * 6 + 0.7 * 0 = -0.7$$

Feature map

-0.7

Filter (Kernel)

0.2	0.7
-0.5	0.7

Convolutional Networks

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

$$y_1 = 0.2 * 1 + 0.7 * 3 - 0.5 * 6 + 0.7 * 0 = -0.7$$

$$y_2 = 0.2 * 3 + 0.7 * 5 - 0.5 * 0 + 0.7 * 2 = 5.5$$

Feature map

-0.7	5.5

Filter (Kernel)

0.2	0.7
-0.5	0.7

Convolutional Networks

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

$$y_1 = 0.2 * 1 + 0.7 * 3 - 0.5 * 6 + 0.7 * 0 = -0.7$$

$$y_2 = 0.2 * 3 + 0.7 * 5 - 0.5 * 0 + 0.7 * 2 = 5.5$$

$$y_3 = \cdots$$

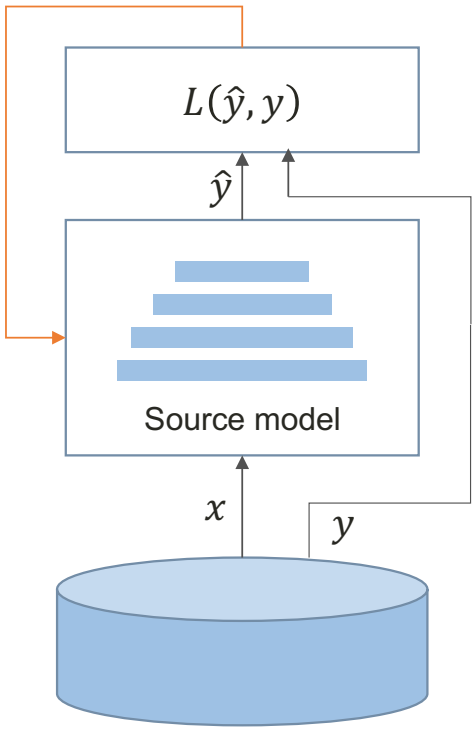
Feature map

-0.7	5.5	…

Filter (Kernel)

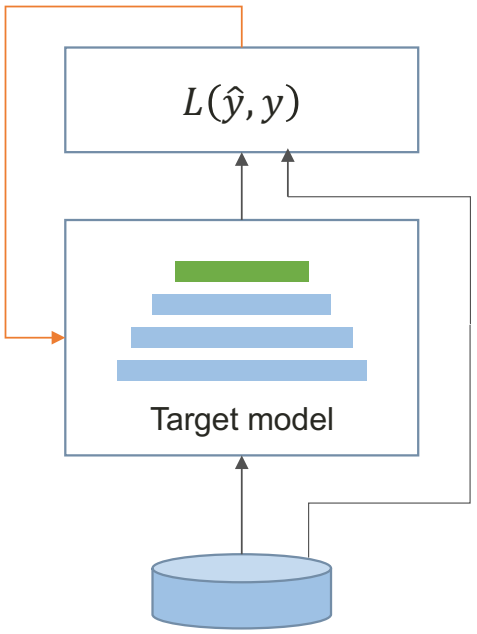
0.2	0.7
-0.5	0.7

Convolutional networks – Transfer learning



Database with a lot of images and labels e.g. Imagenet

Transfer learning



Target data – your smaller dataset

Forward

Backward

Convolutional networks – data augmentation

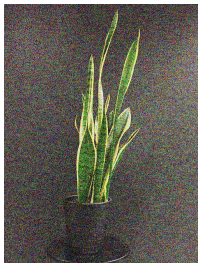
- Main idea is to generate synthetically new data samples from existing samples
- Works best in classification tasks
- Transformations such as rotation and scaling have proofed to be efficient and help the classifier to be more invariant.

Original

Hue shift

mirror

Black&white



noise

crop

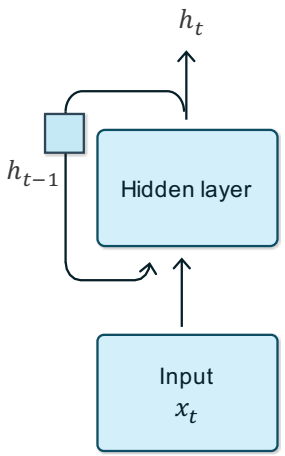
Examples of data augmentation

Recurrent Neural Networks

Recurrent networks

Neural network for sequential data (time series)

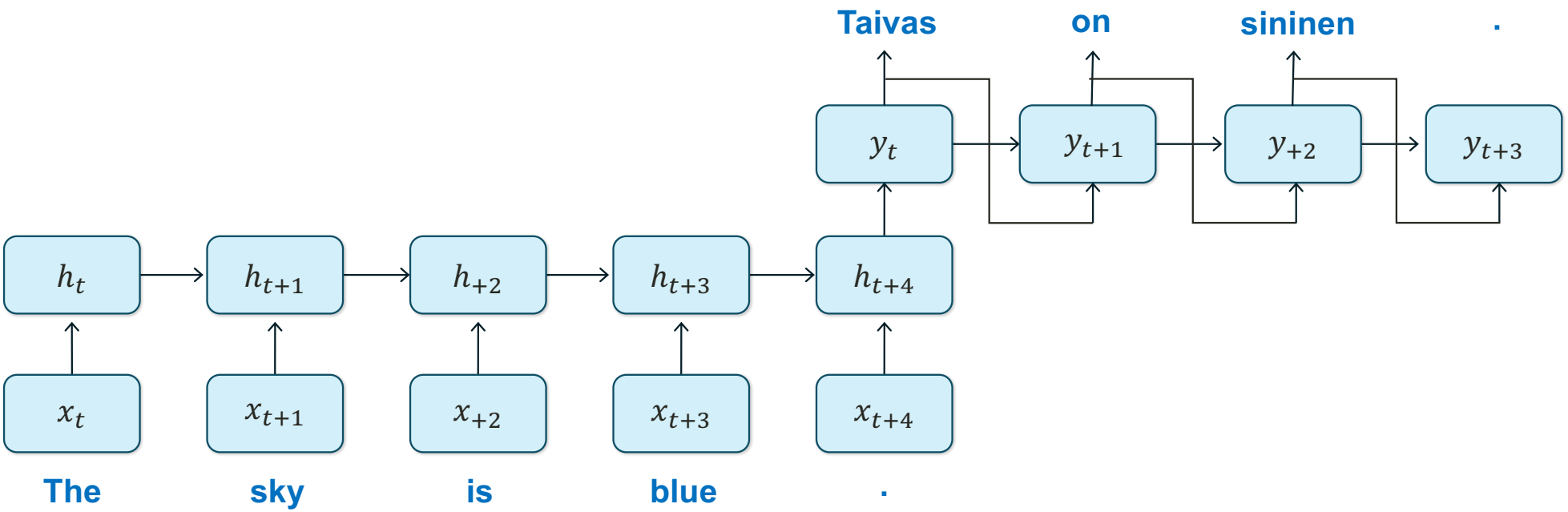
Basic principle: $\boldsymbol{h}^{(t)} = f\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{W}\right)$



"With enough neurons and time, RNNs can compute anything that can be computed by your computer"
(Geoff Hinton)

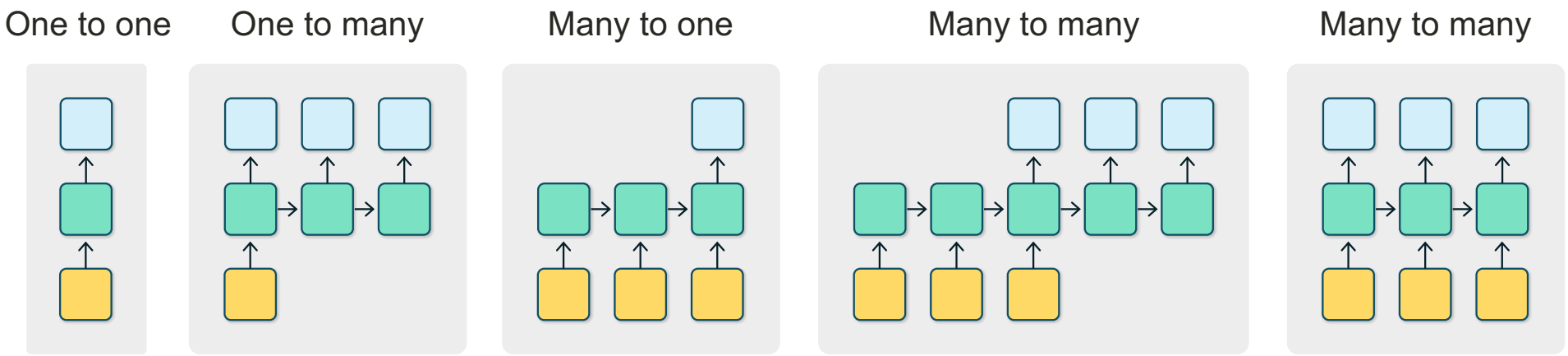
"RNNs could potentially learn to implement lots of small programs that each capture a nugget of knowledge and run in parallel, interacting to produce very complicated effects"
(Geoff Hinton)

Recurrent networks – sequence to sequence model



Recurrent networks – different recurrent configurations

Recurrent networks offer a lot of flexibility



Live demo

AI-based speech synthesizer

Live demo

AI-based essay grading