02456 – Week 1
Neural networks

Jes Frellsen

Technical University of Denmark

1 September 2025

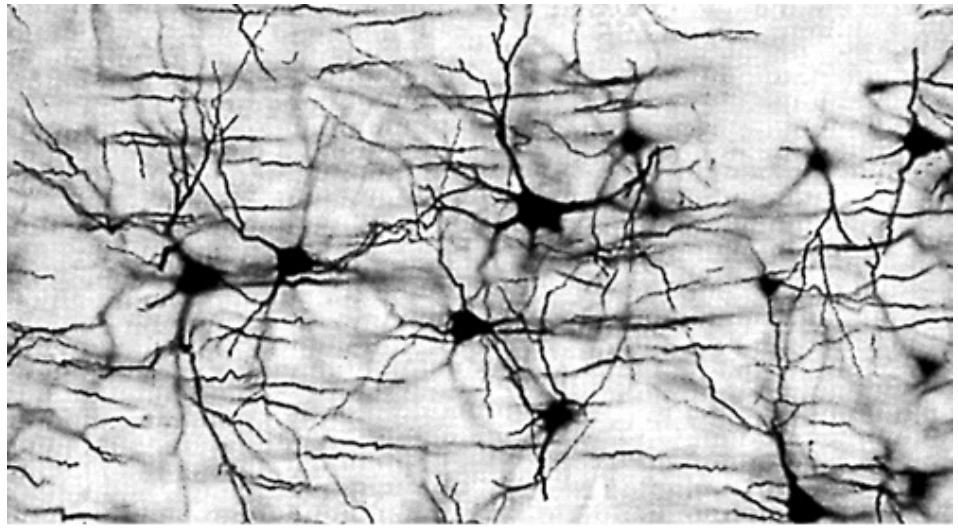
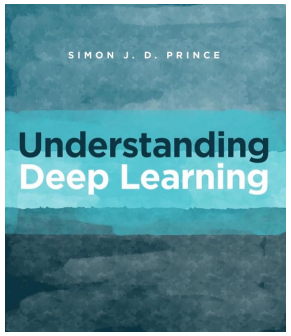


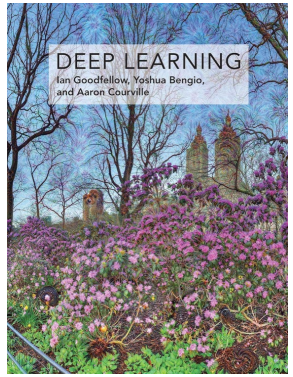
Image from: https://alanturing.net

Slides codeveloped by Ole Winther, Tapani Raiko and Jes Frellsen

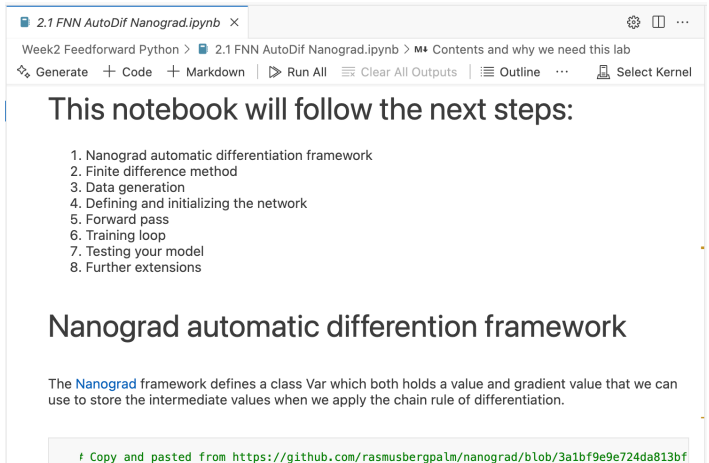
Course overview

Weeks 1-8
Teaching: Lectures + Exercises





Jes Frellsen



Weeks 9-15
Project based

Project list 2024

1) **Come with your own project.** Requirements: data and problem statement are in place so that time will be spent on modelling. Ideally, team up with other students but one student teams may be accepted in special circumstances. Supervised by Ole Winther <olwi@dtu.dk>, Jes Frellsen <jefr@dtu.dk> and others.
2) **Implementation of diffusion models.** The project is to re-implement the Denoising Diffusion Probabilistic Model (DDPM) in PyTorch and reproduce their results at least on MNIST and ideally on CIFAR-10. This paper is the one that kicked off the diffusion movement, it is a great way to learn what diffusion is all about and have hands-on experience. Link to paper: https://arxiv.org/abs/2006.11239. Supervised by Jes Frellsen <jefr@dtu.dk>, Paul Jeha <pauje@dtu.dk>, and Christian Raasteen <s204148@student.dtu.dk>.
3) **Lowering the downtime of wind turbines** is a priority for the wind industry. Around 20% of the faults leading to shutting the turbine down are related to the pitch system, which regulates the angle of attack of the wind with respect to each blade. For high-power offshore installations such systems are hydraulic.
   Most of the faults are consequent to degradation of the oil in the hydraulic circuit, which firstly manifests as an increased friction in the hydraulic actuator which rotates the blade. In order to correctly monitor the friction evolution, the mechanical load that the wind exerts on the blades needs to be derived. A measurement for it is generally not available. More information are available in the publication 10.1016/j.ifacol.2024.07.284 (search on google)
   The projects aim at developing a deep learning model which is capable of determing the correct load, starting from a set of common measurements that are available in wind turbines, as time series. The data for training are obtained in simulation and include the correct output.
   Two (or eventually more) architectures can be chosen, namely transformers and a combination of RBFNN/TCNN. A benchmark solution involving FFNN and a combo of CNN/RNN is available for comparing the performance.
   For more information visit https://aldalldtu.wordpress.com. Supervised by Alessio Dallabona <aldall@dtu.dk>.
4) **Velocity estimation from Doppler radar**. Golf simulators provide insights to the player by measuring or estimating key mechanical parameters during the golf swing. Some systems use Doppler radar to detect and track the velocity of the ball, but the signal is confounded by noise arising from other moving components during the swing.
   This project aims to estimate the initial radial velocity component (i.e. in the direction of the radar system) of the ball velocity after impact from Doppler spectrograms. It builds on the foundation of an existing implementation using a convolutional neural network and physics-informed regularisation. Thus data is available, pre-processed, annotated, split into training/testing/validation subsets, and ready to be used with Pytorch.
   The project aims are: 1) improving or maintaining the model performance with free reign over architecture and regularisation, under the constraint that 2) the number of

Teaching assistants

Weeks 1-8
Exercises


Andreas Theilgaard


Mustafa El-Madani


Christian Kjær


Nikolaj Hertz


Dan Edelstein


Robert Spralja


Matteo D'Souza

Weeks 9-15
Project supervisors (PhD students)


Emilie Wedenborg


Marisa Wodrich


Ekaterina Protsenko


Mikkel Rasmussen


Jonas Vestergaard


Viswanathan Sankar


Rasmus Tirsgaard

Locations

Why only 1 hour?

You should work with the material and get hands on experience with deep learning!

- In teaching weeks (Mondays)
  - Lectures 13:00-14:00 in B303A-A042 (and online)
  - Exercises 14:00-17:00 in all rooms
- In project weeks
  - Supervision 13:00-17:00 in all rooms
- Communication: DTU Learn and Slack (there is a link on Learn to join slack)

All rooms:

- B303A-A042
- B303A-IT046
- B303A-IT047
- B303A-IT048
- B303A-HVEST
- B303A-HOEST
- Zoom

Main learning objectives

- **Understand** key **deep learning** concepts, terminology, and limitations
- **Apply** and **analyse** models in exercises and project work
- **Plan** and **carry out** an **applied** or **methods-oriented DL** project in teams
- **Use a computational framework** for GPU programming (PyTorch)
- **Assess**, **summarize**, and **interpret** project **results**
- **Communicate results** through a structured **report**

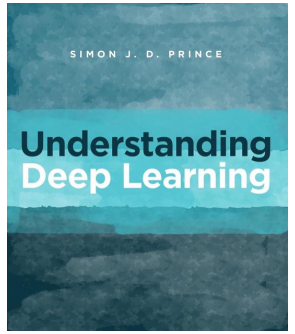
Course alignment and evaluation

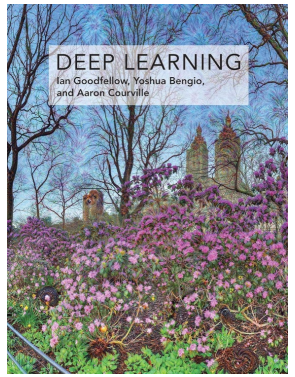
- The weekly exercises consist of
  - **Notebooks** that will prepare you for the **project** and **written exam**
  - **Problems from the book** that will prepare you for the **written exam**

- Your grade is based on the evolution of:
  - The **final written exam (~25%)**
    - Multiple choice
    - 2 hours on premise
    - No electronic aids
    - Written works of reference are permitted
  - The **projects (~75%)**
    - Groups of 3-4 people
    - Select from a long list of project proposal
    - Report: max 4 page + reference and code

Content (week 1-8)

Week	Topics	Reading
1	Shallow and deep neural networks	2, 3, 4
2	Loss functions, back propagation and initialisation	5, 6, 7
3	Measuring performance, regularisation and residual networks	8, 9, 11
4	Convolutional networks	10
5	Recurrent neural network	10 (Goodfellow)
6	Transformers	12
7	Unsupervised learning (GANs and VAEs)	14, 15, 17
8	Mini project	





Our expectations of you

- This is a MSc-level
- We expect that you have the right prerequisites
  - Basic calculus (chain rule of differentiation)
  - Basic linear algebra
  - Basic multivariate probability theory and statistics
  - Basic programming in Python.
  - 02450 Introduction to ML and Data Mining
- We expect that you
  - Read the chapters in the book
  - Do on the exercises (at least look at them before the sessions)

Major changes from last years

This year

- 1 hour lecture
- Mainly the Prince book
- No exercise hand-in
- Written exam
- Project hand-in at end of semester
- No deep reinforcement learning

Last years

- Video lectures (outdated)
- Several different books
- Mandatory peer grading
- Poster exam
- Project hand-in at end of exam period
- Deep reinforcement learning

- Update the course
- Reduces course load

About the lectures

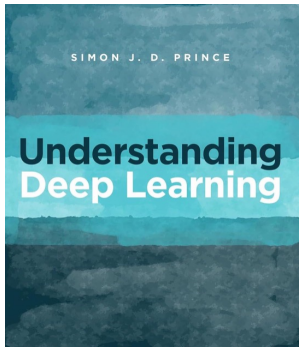
- **Purpose:** given an overview of the material
- You need to study the book for **details**

- **Notation:** may deviate from the book

$$\mathbf{y} = \mathbf{f}[\mathbf{x}, \phi] \rightarrow \mathbf{y} = f_{\mathbf{w}}(\mathbf{x})$$

- **Useful** to get used to different notation
- This **abstraction** is expected at MSc level

Menu of the day



- Week 1: Neural nets
- Week 2: Learning
- Week 3: Tricks of The Trade
- Week 4: CNNs
- Week 5: RNNs
- Week 6: Transformers
- Week 7: Unsupervised
- Week 8: Mini-project

Lecture

- Part 1: Deep learning revolution
- Part 2: Neural nets
  - Chapters 2, 3, 4

Exercises

- **Problems:** 3.5, 3.11, 4.1, 4.11
- **Notebook**:
  - `1.1 FNN Pen and Paper.ipynb`
  - Contains concepts from week 2, but it rather self-contained.

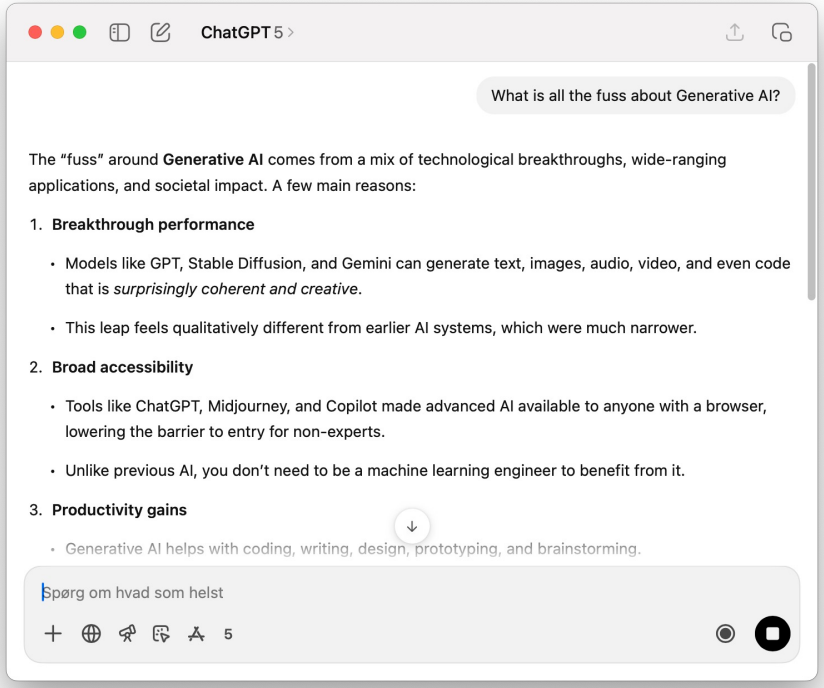
Part 1:
Deep learning revolution

AI hype: Generative AI

The Economist

Science & technology | Generative AI

Large, creative AI models will transform lives and labour markets



← THE BEST INVENTIONS OF 2022

TIME

Artificial Imagination

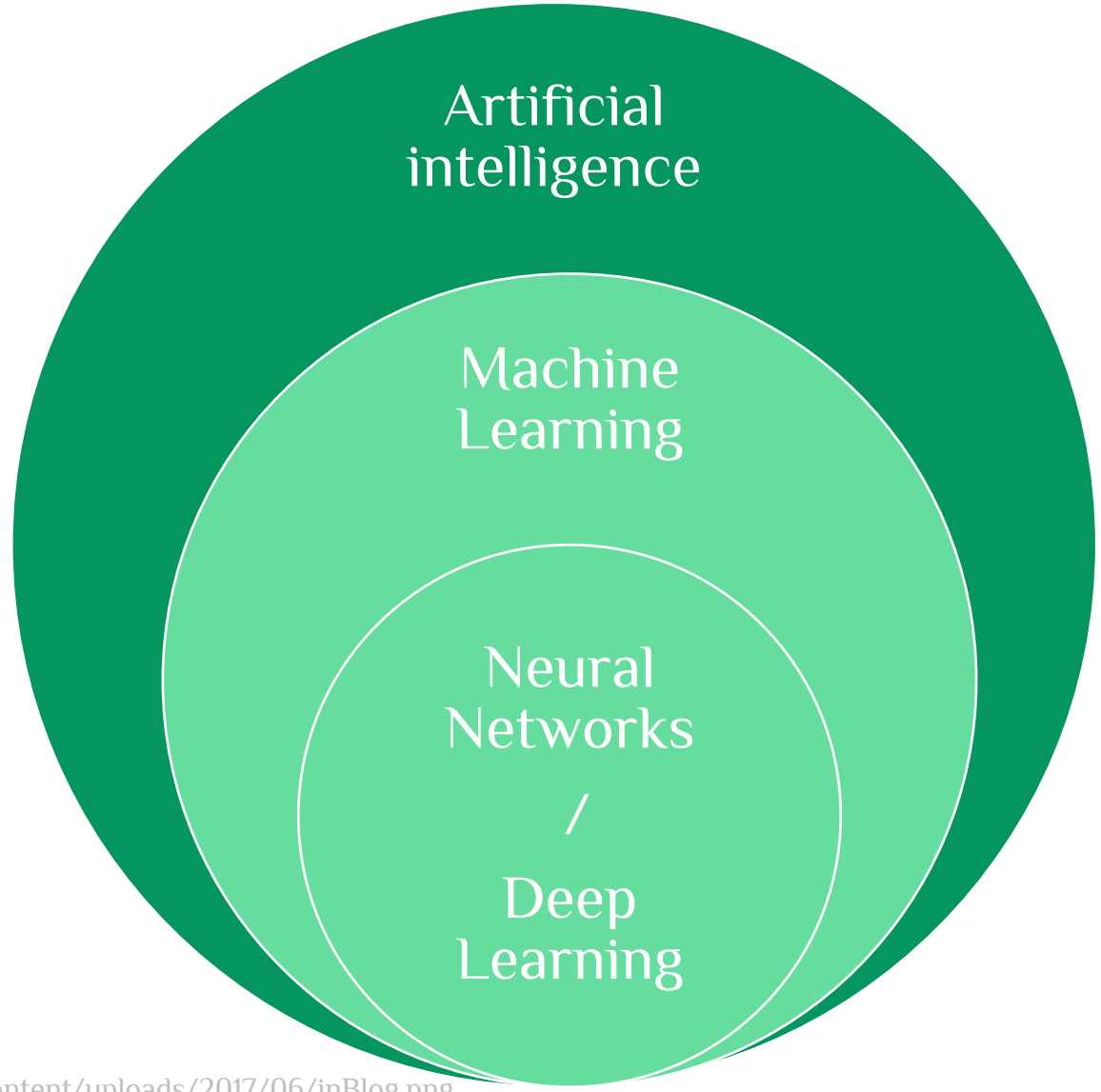
OpenAI DALL-E 2

Major Major Application Areas of AI

- Computer vision
- Natural Language Processing (NLP)
- Speech & audio Processing
- Robotics & autonomous systems
- Recommendation & personalization
- Healthcare & life sciences
- Climate, environment & sustainability

What is this?

AI is an umbrella term



Based on: https://www.prowesscorp.com/wp-content/uploads/2017/06/inBlog.png

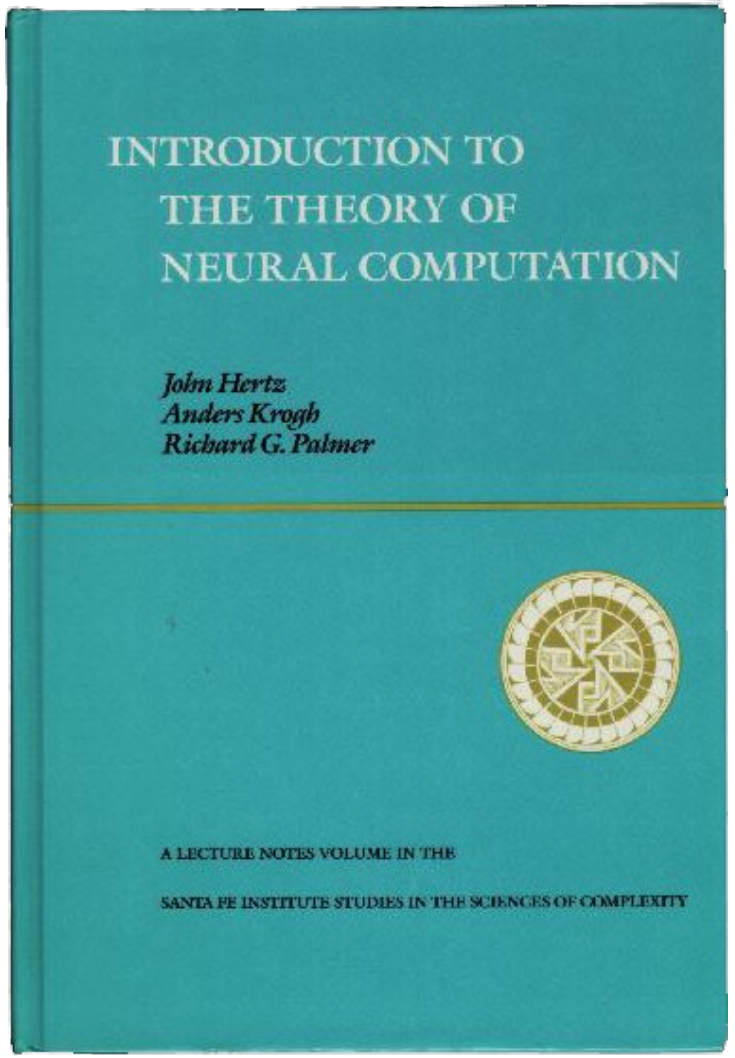
Dartmouth Summer Research Project on AI (1956)

"We propose that a **2 month, 10 man study** of **artificial intelligence** be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. **An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.**"

Image from: https://achievement.org/achiever/marvin-minsky-ph-d/#gallery
Citation from: McCarthy, J., M. Minsky, N. Rochester, and C. Shannon (1955). "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence".

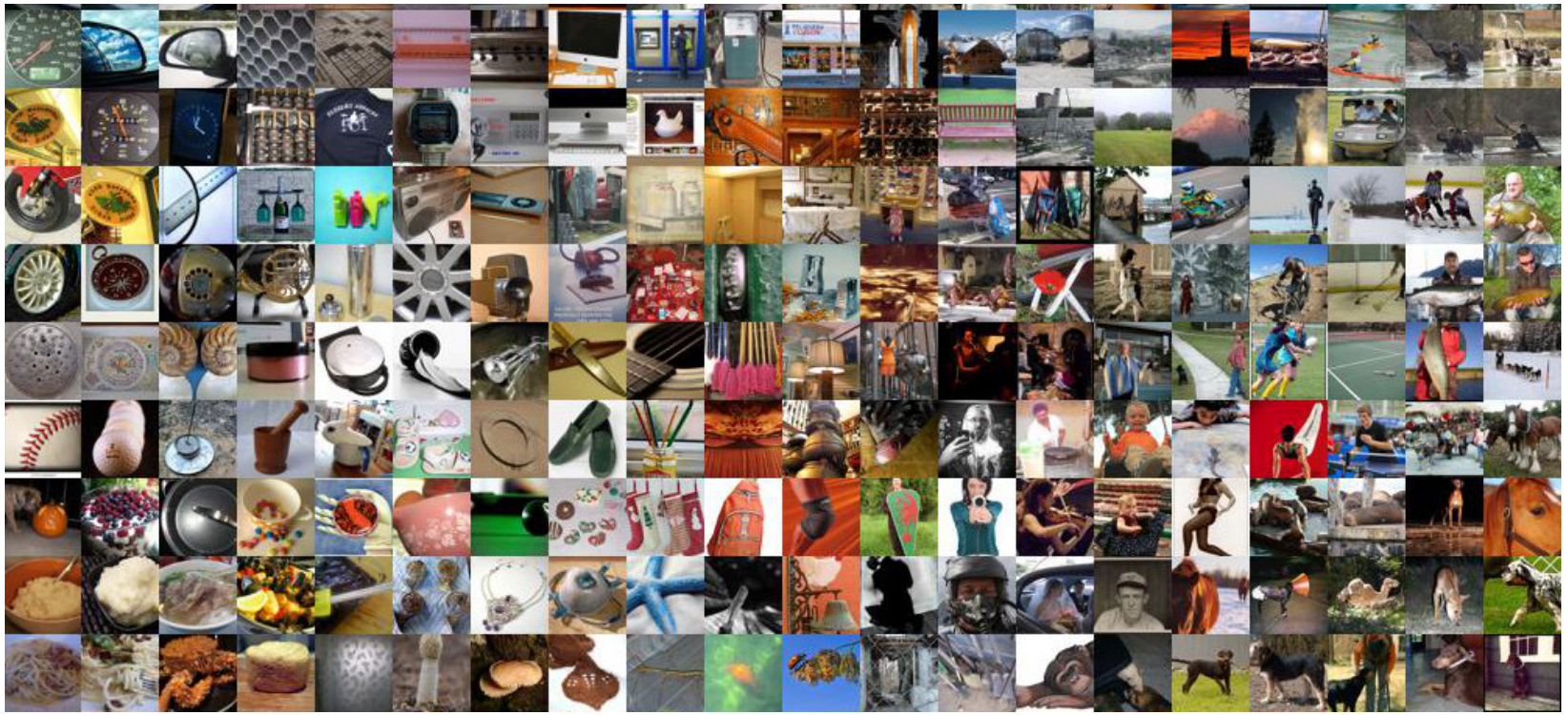
Neural networks

- The core component in **Deep Learning (DL)** is Neural Networks (NNs)
  - Initial works started in the **1940s**
  - Much of the theory of NNs was developed in the 1970s–1990s
- In the early 2010s DL showed extremely successful results for application in
  - Image recognition
  - Machine translation
  - Speech recognition



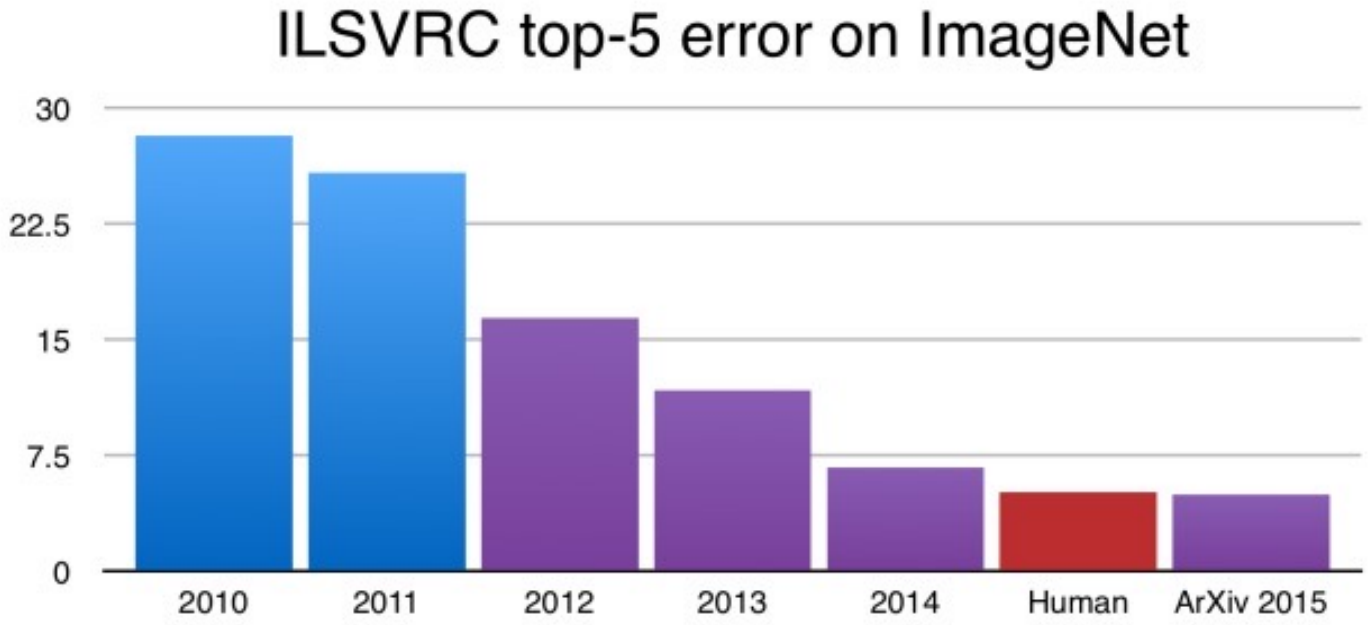
(1991)

ImageNet classification challenge



- Annual competition in computer vision
- ~1.2M training images
- 1000 classes

ImageNet classification challenge



- Krizhevsky et al. (2012) won with huge margin with **deep learning**
  - (16.4% error compared to 26.2%)
- Soon everyone started using **deep learning** and GPUs.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional, NIPS, 2012.

Deep reinforcement learning

(2015)

(2016)

Generative AI

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Transformers (~2017)

Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu
Ajay Jain UC Berkeley ajayj@berkeley.edu
Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

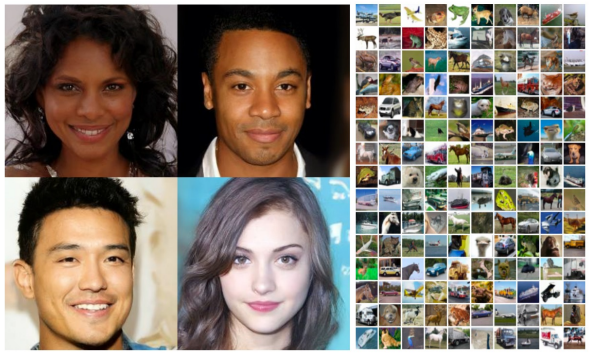
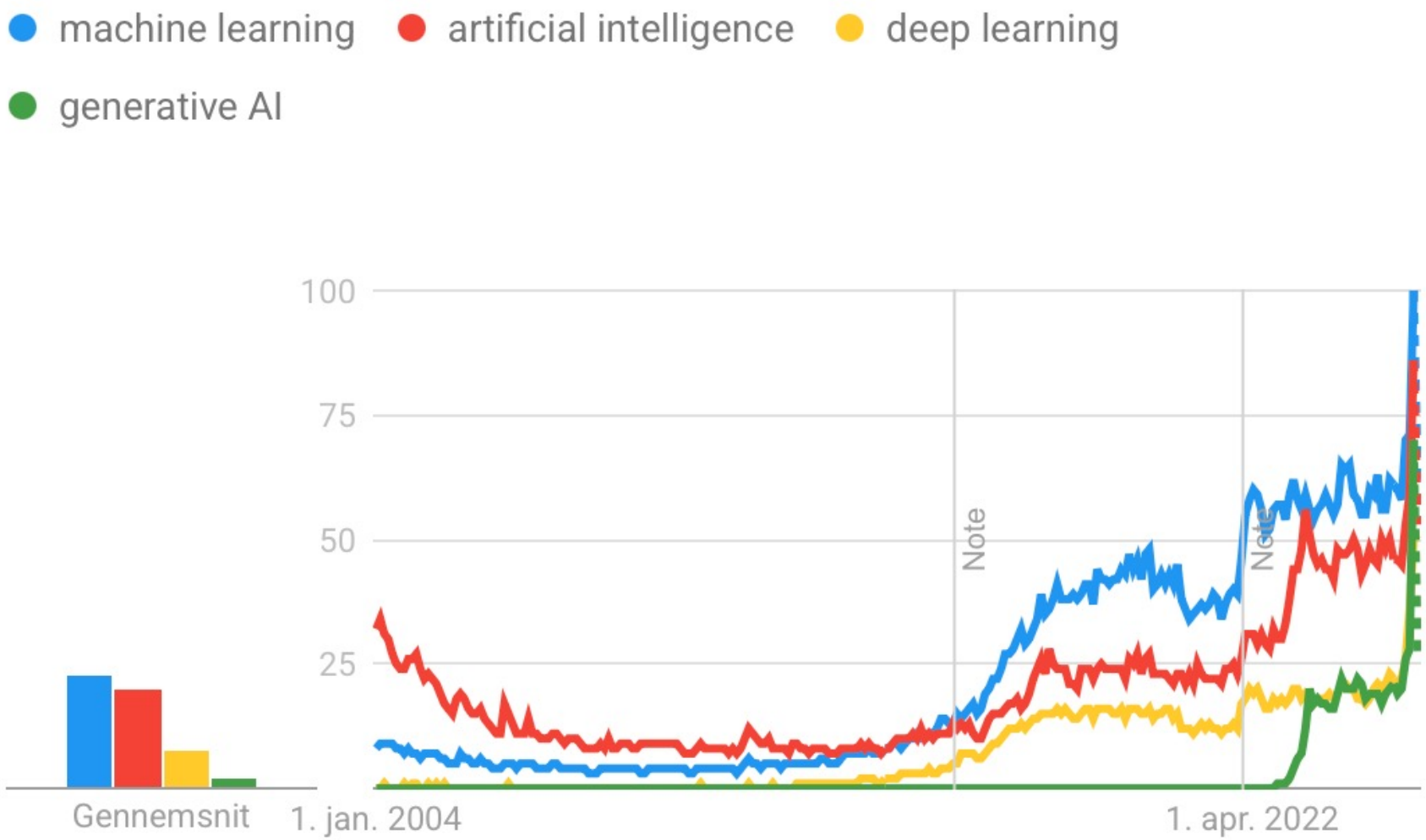


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

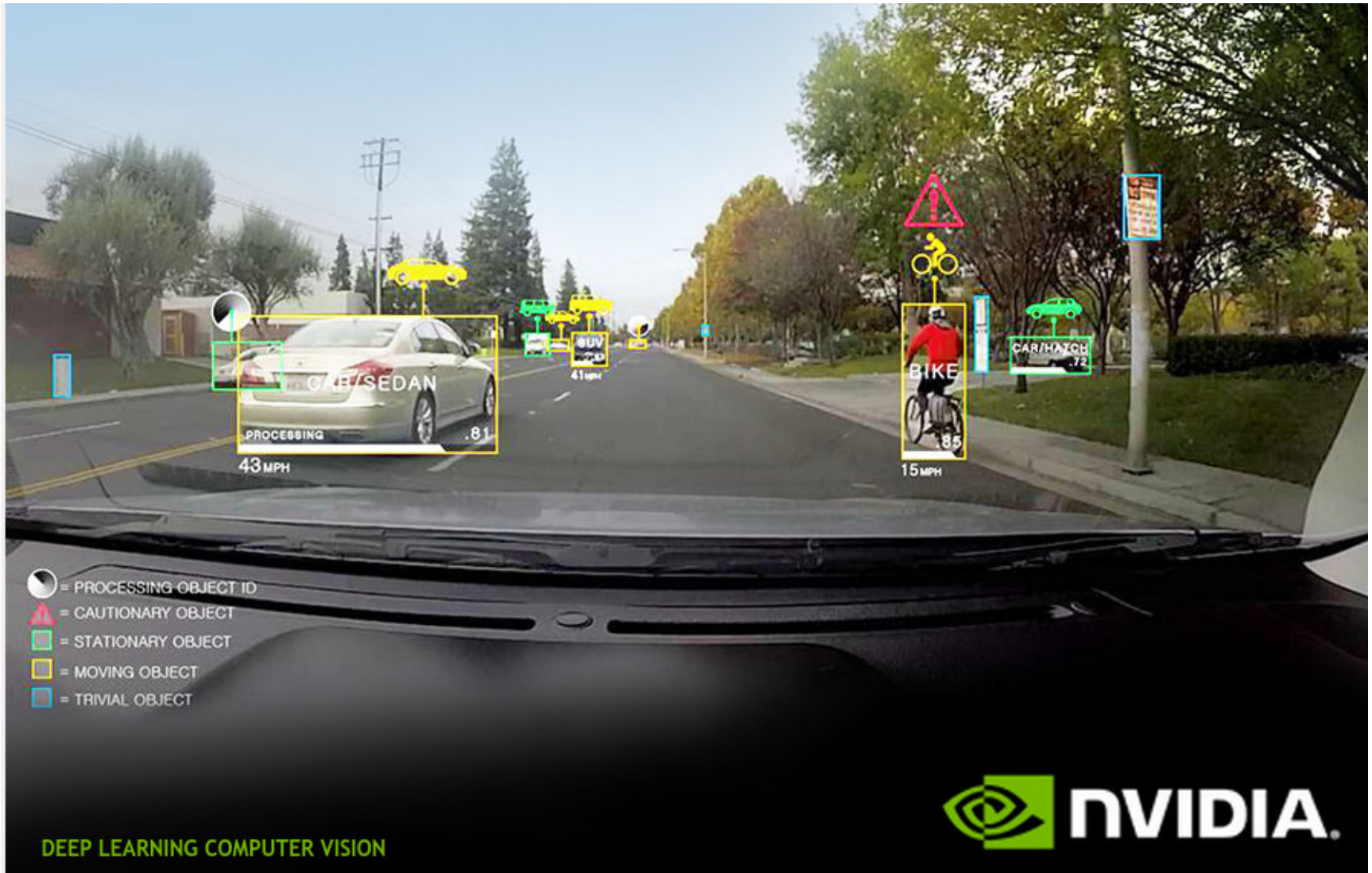
34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Diffusion models (~2020)

Google trends



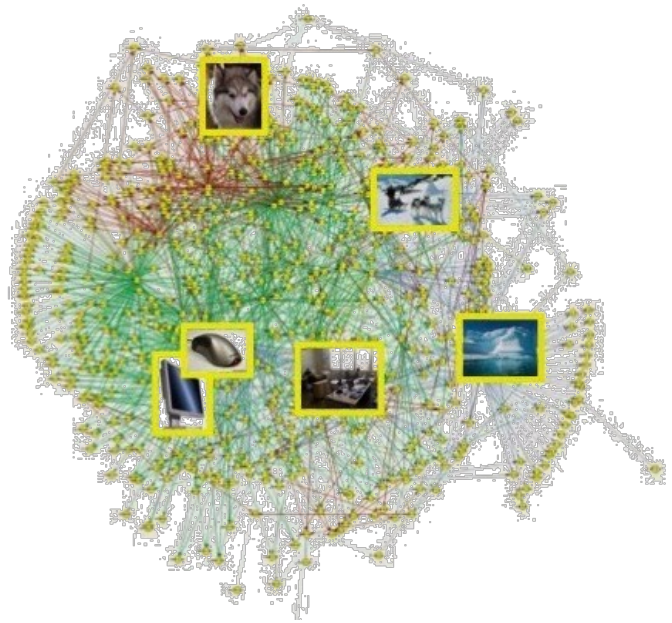
Deep learning has become a standard tool in the toolbox



What has been initially driving this Deep Learning revolution?

Faster computers

Bigger datasets



PyTorch TensorFlow™ Keras

Better software libraries

Representation learning

Traditional machine learning way:

- Data → Feature engineering → Machine learning
  - Feature selection
  - Feature extraction (e.g., PCA)
  - Feature construction (e.g., SIFT)

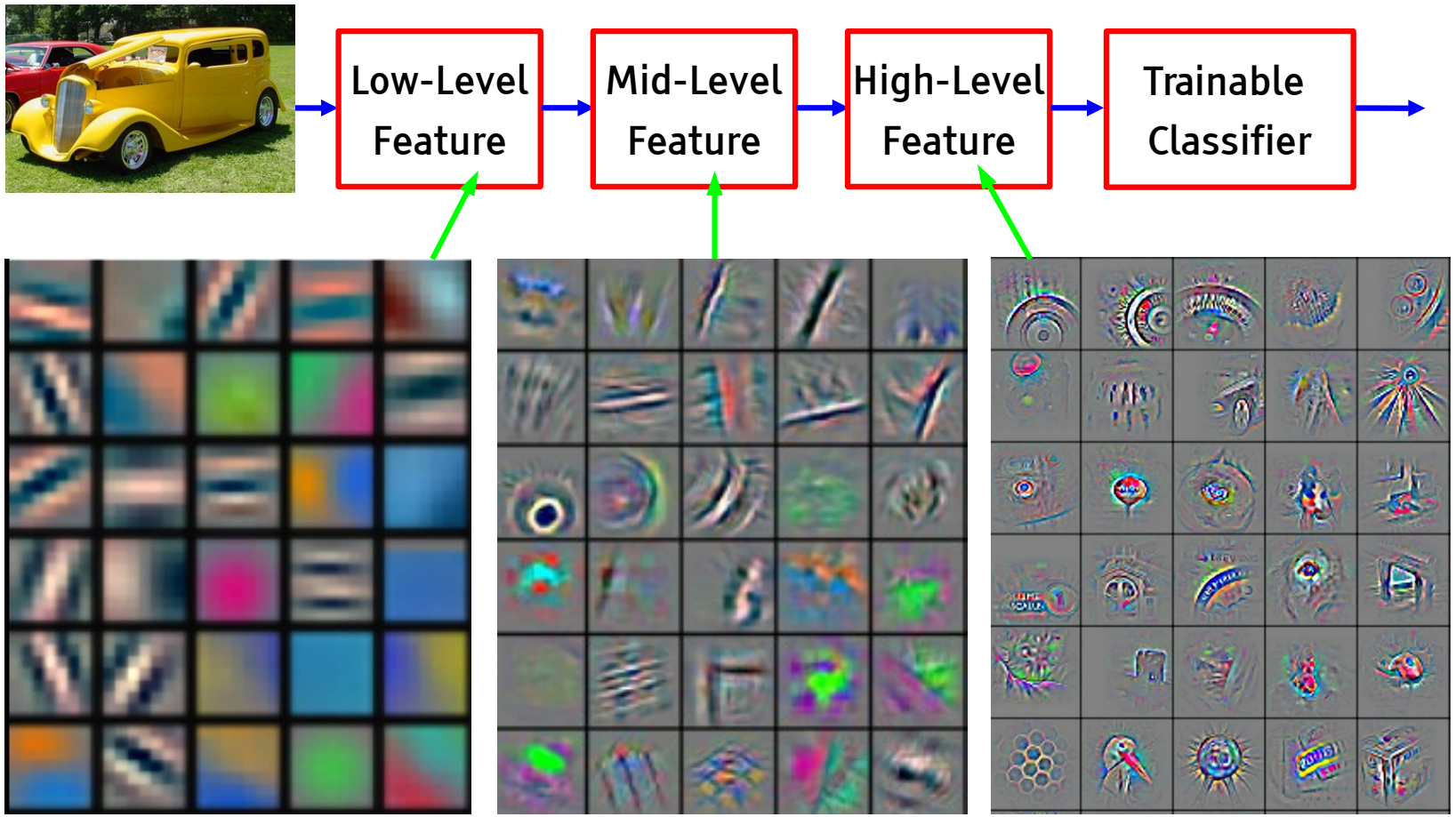
Deep learning way:

- Data → End-to-end learning

Deep Learning = Learning Hierarchical Representations

Y LeCun

- It's deep if it has more than one stage of non-linear feature transformation



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

Part 2: Neural networks

Supervised learning

Given dataset $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$, we want to learn a **model/function**

$$y = f_\phi(\mathbf{x})$$

where ϕ are parameters

- E.g., in **linear regression**

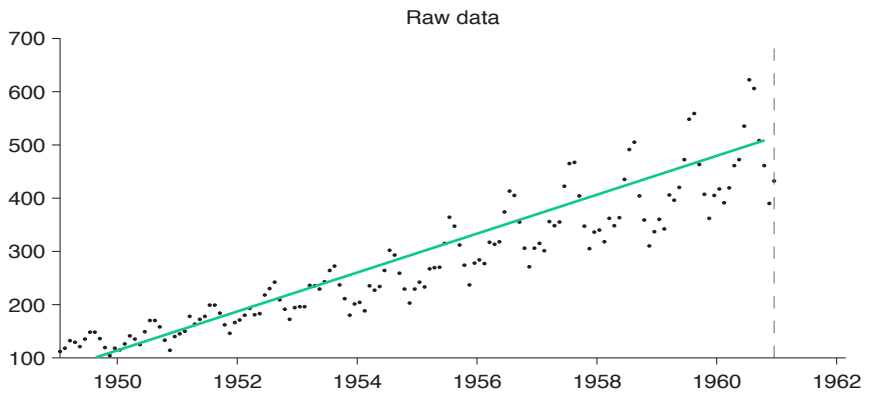
$$f_\phi(x) = ax + b \qquad \text{and} \qquad \phi = (a, b)$$

- **Loss/error** function: mismatch between data and model, e.g., mean squared error

$$L(\phi) = \frac{1}{n}\sum_{i=1}^{n} \left(f_\phi(\mathbf{x}) - y_i\right)^2$$

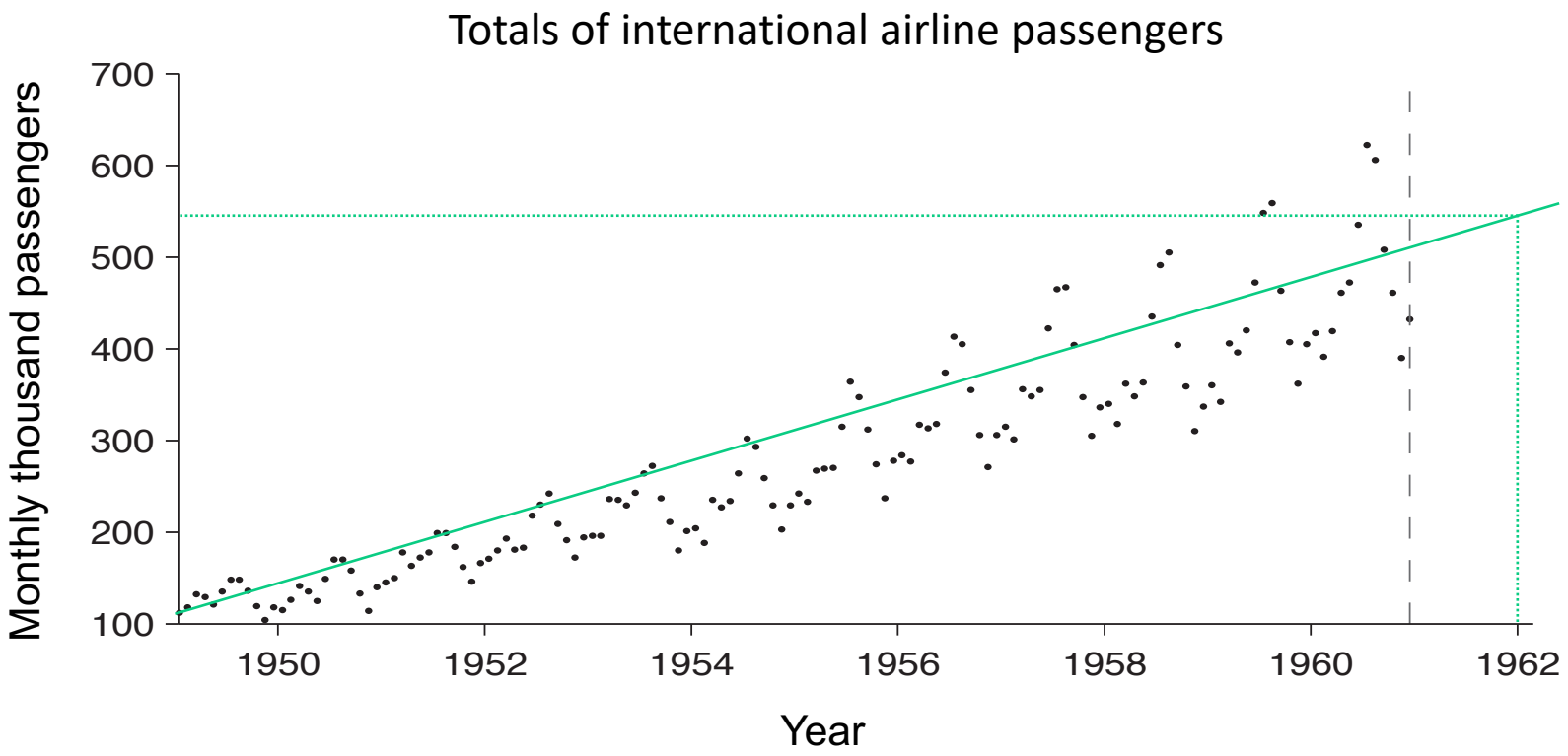
- We **learn/fit** the model, but minimising the loss

$$\hat{\phi} = \text{argmin}_\phi \, L(\phi)$$



Plot from Duvenaud et al. "Structure Discovery in Nonparametric Regression through Compositional Kernel Search", ICML 2023

Supervised learning: predictions

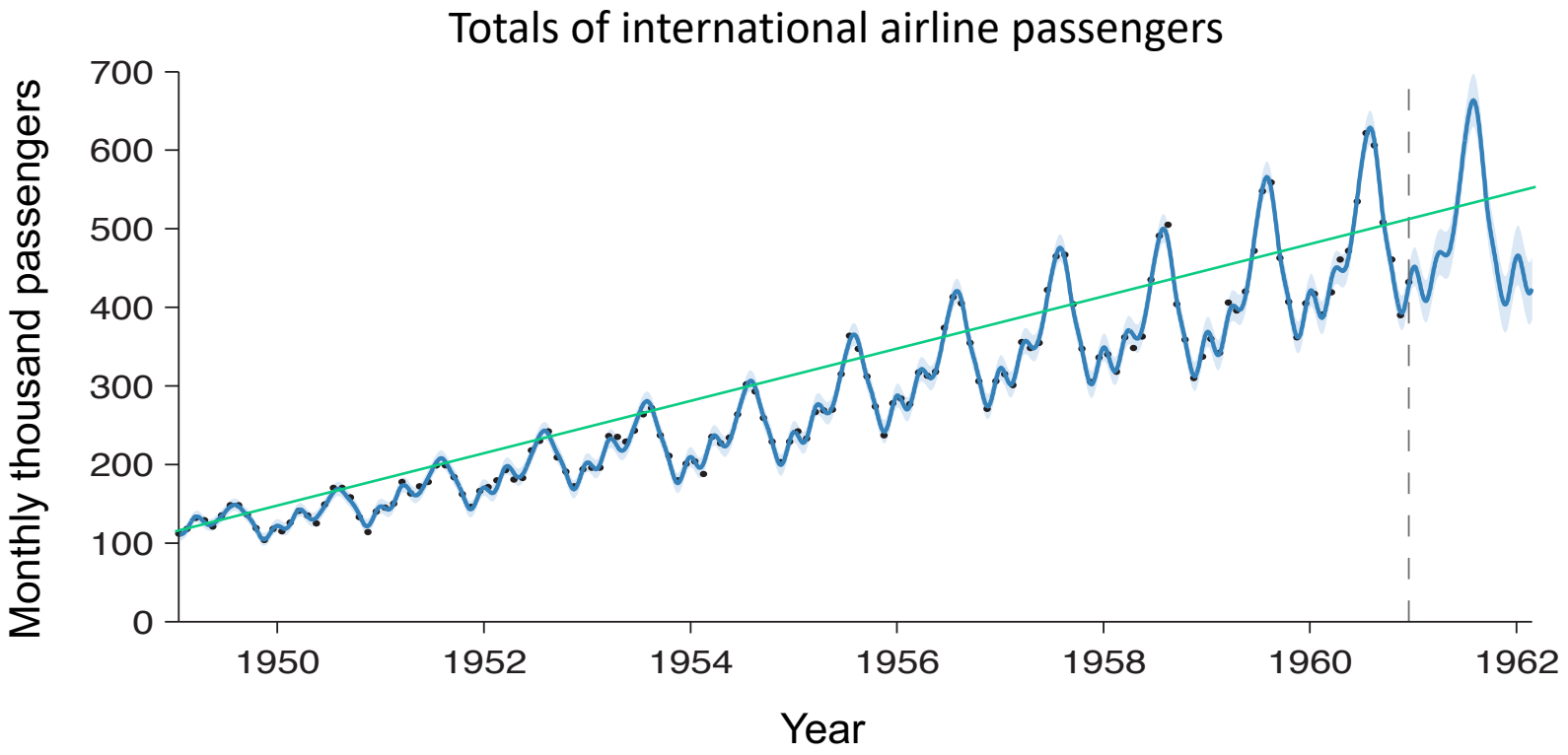


I can make **predictions** (**inference**) using the **model**, e.g., at $\hat{x} = 1962$

$$f_\phi(\hat{x}) \approx 550$$

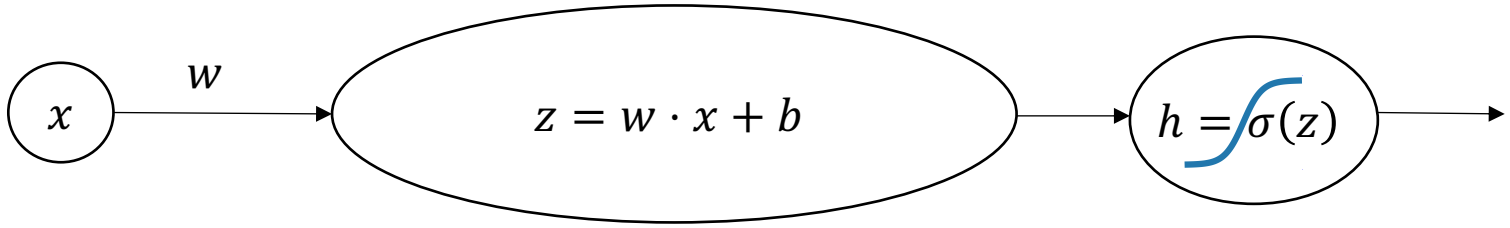
Plot from Duvenaud et al. "Structure Discovery in Nonparametric Regression through Compositional Kernel Search", ICML 2023

Limitation of linear models



- Does not work well, if data is nonlinear
- How can we construct functions that approximate arbitrary complex mappings?
- Neural networks!

Building block: Artificial neuron (1D input)



Input: x

Weight: w

Bias: b

Pre-activation: $z = w \cdot x + b$

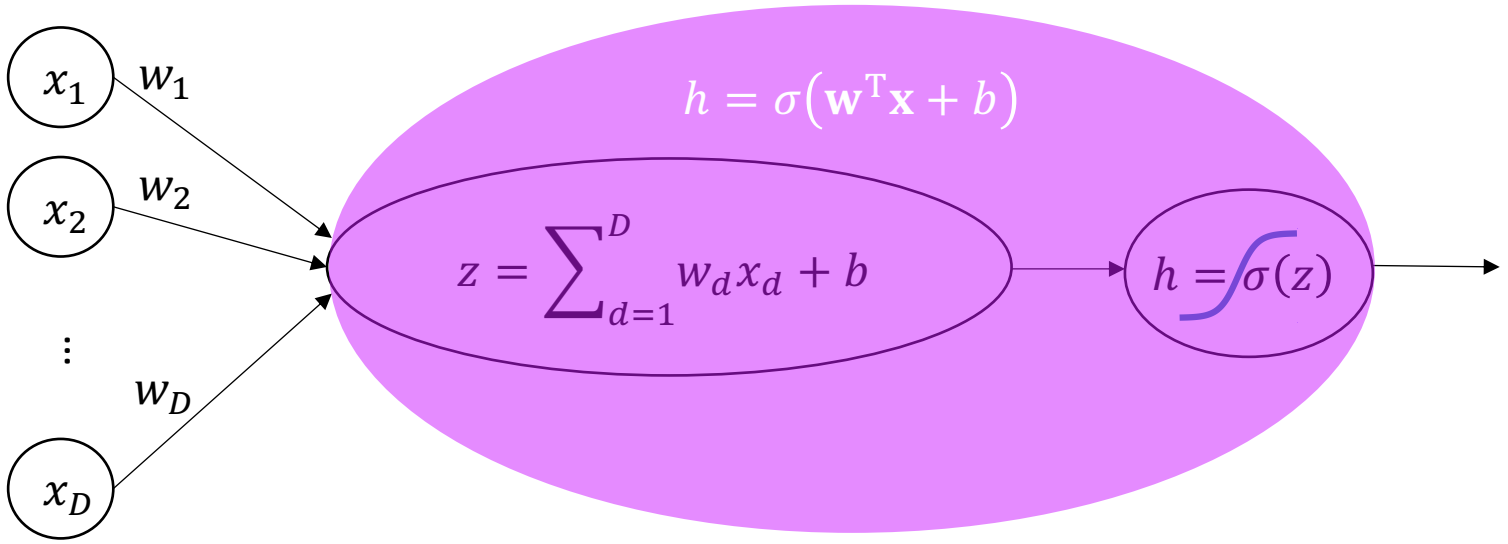
Activation: $h = g(z)$

Where σ is **non-linear** activation function

The function is:

$$f(x) = \sigma(w \cdot x + b)$$

Building block: Artificial neuron



Input: $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$

Weights: $\mathbf{w} = (w_1, \ldots, w_D)^{\mathrm{T}}$

Bias: b

Pre-activation: $z = \sum_{d=1}^{D} w_d x_d + b = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$

Activation: $h = \sigma(z)$

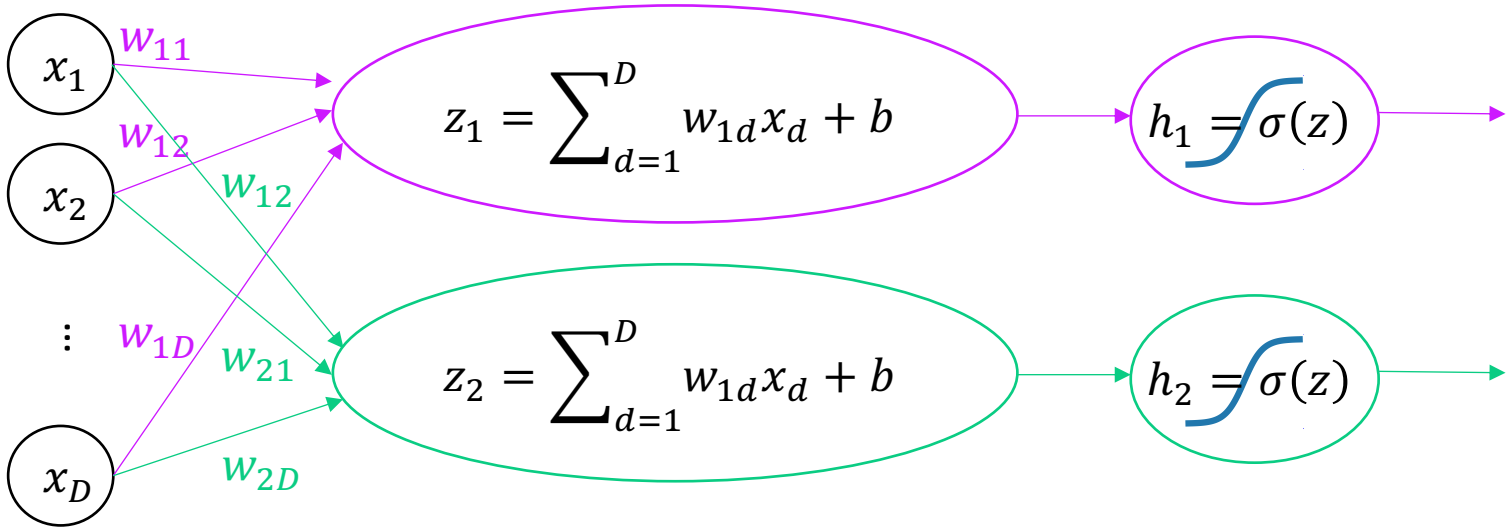
Where σ is **non-linear** activation function

The function is:

$$f(x_1, \ldots, x_d) = \sigma\left(\sum_{d=1}^{D} w_d x_d + b\right)$$

$$f(\mathbf{x}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right)$$

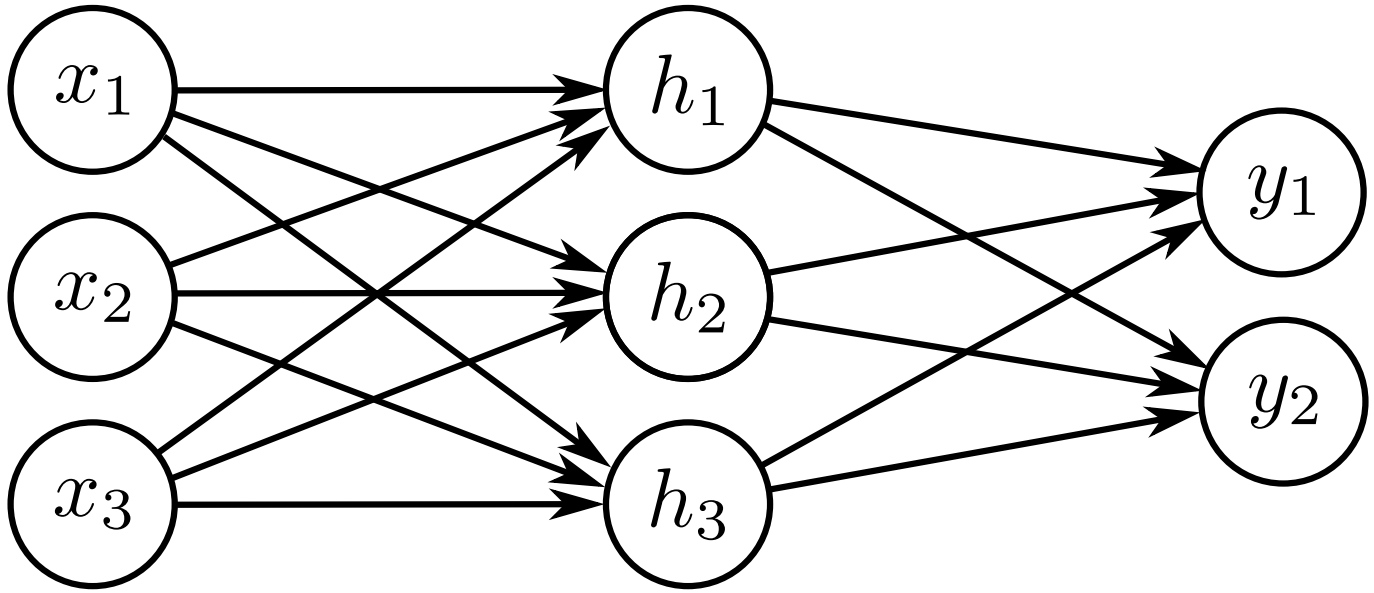
Network of neurons



$\mathbf{x}$ $\mathbf{z} = W\mathbf{x} + \mathrm{b}$ $\mathbf{h} = \sigma(\mathrm{z})$

Where $W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1D} \\ w_{21} & w_{22} & \cdots & w_{2D} \end{bmatrix}$ and σ is applied elementwise

A shallow neural network (1 hidden layer)



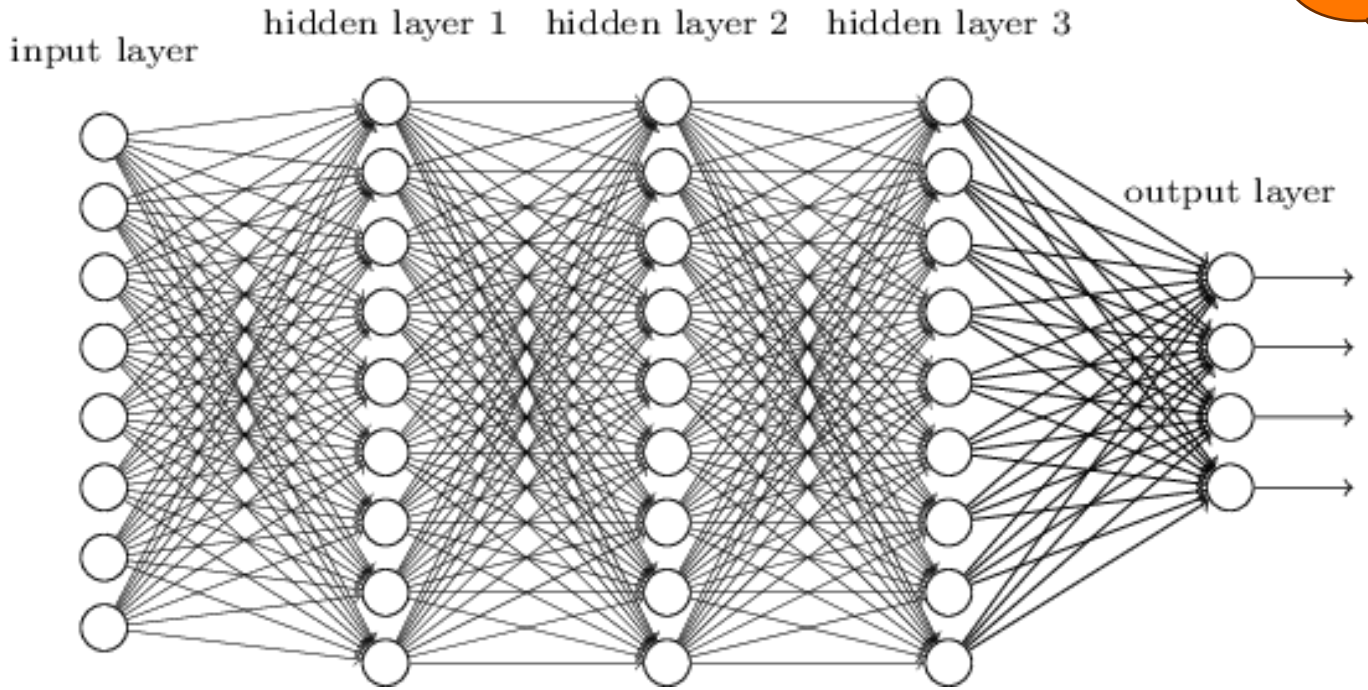
$$\mathbf{x} \qquad \mathbf{h} = \sigma^{(1)}\left(W^{(1)}\mathbf{x} + \mathrm{b}^{(1)}\right) \qquad \mathbf{y} = \sigma^{(2)}\left(W^{(2)}\mathbf{h} + \mathrm{b}^{(2)}\right)$$

$$f_\phi(\mathbf{x}) = \sigma^{(2)}\left(W^{(2)}\sigma^{(1)}\left(W^{(1)}\mathbf{x} + \mathrm{b}^{(1)}\right) + \mathrm{b}^{(2)}\right)$$

Figure from: Prince SJD. Understanding deep learning. MIT press, 2023.

A deep neural network

Hvorfor hedder lagene mellem intput og ouput "skjulte lag"?



We can write the **output of layer ℓ** as

$$f^{(\ell)}(\mathbf{h}) = g\left(W^{(\ell)}\mathbf{h} + \mathbf{b}^{(\ell)}\right).$$

The **joint function** is then

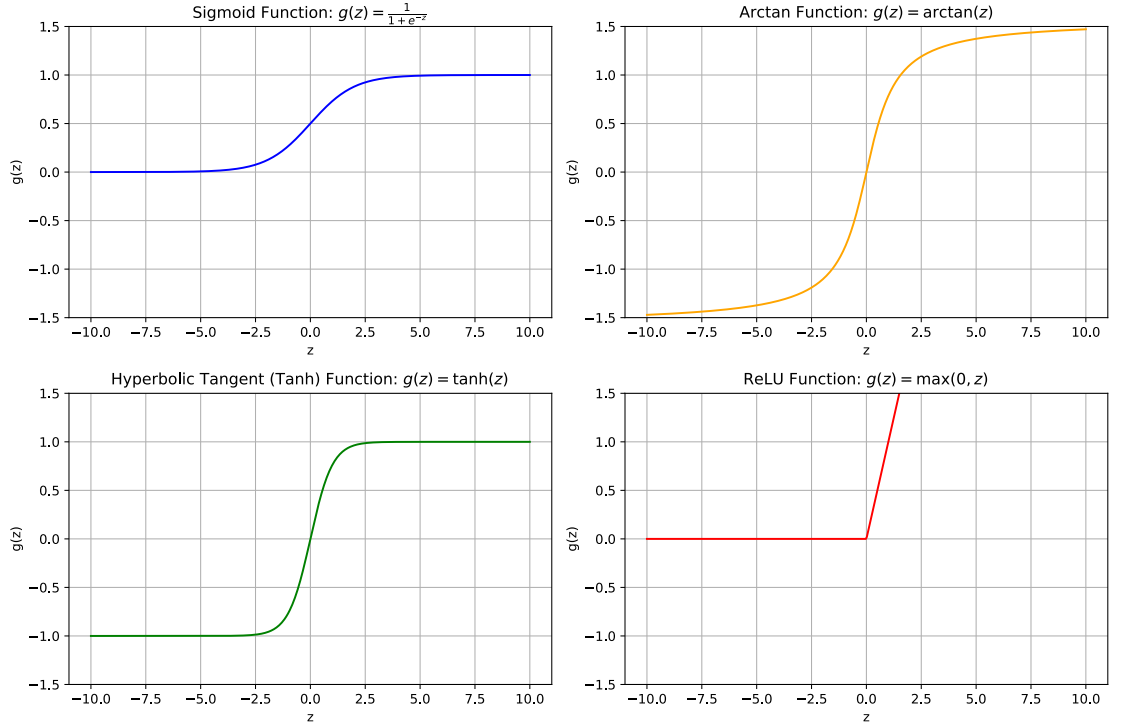
$$f(\mathbf{x}) = \mathbf{y} = f^{(4)}\left(f^{(3)}\left(f^{(2)}\left(f^{(1)}(\vec{x})\right)\right)\right)$$

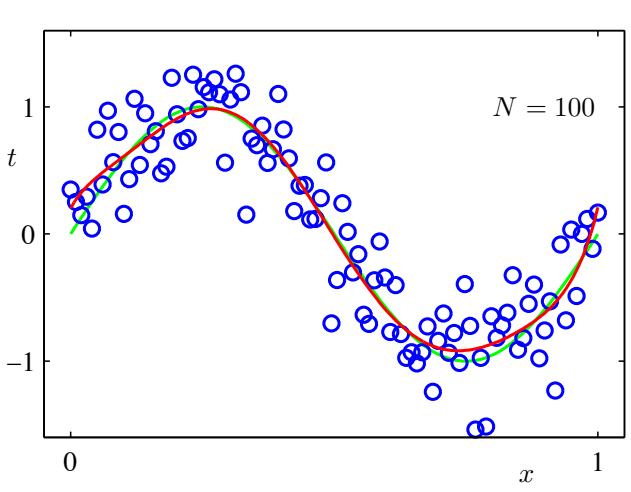
Figur fra: neuralnetworksanddeeplearning.com/chap6.html

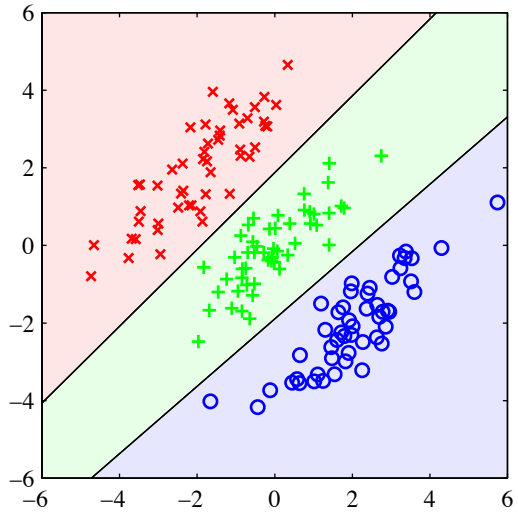
Activation functions

- For hidden layers:
  - Sigmoid: $\sigma(z) = \frac{1}{1+e^z}$
  - Arc-tangent: $\sigma(z) = \text{atan}(z)$
  - Hyperbolic tangent: $\sigma(z) = \tanh(z)$
  - Rectified linear: $\sigma(z) = \max(0, z)$

  The network is a piecewise linear function
- For output layers:
  - Regression: identity: $\sigma(z) = z$
  - Classification: softmax $\sigma(z_i) = \frac{e^{z_i}}{\sum_i e^{z_i}}$







Bottom figures from: Bishop, Christopher M. Pattern Recognition and Machine Learning. Springer, 2006.

Universal approximation theorem

Neural networks are universal approximators (Bishop, 2006):

"A two-layer network with linear outputs can uniformly **approximate any continuous** function on a compact input domain (compact subset of $\mathbb{R}^N$) to arbitrary accuracy provided the network has sufficiently large number of hidden unites"

Quote from: Bishop, Christopher M. Pattern Recognition and Machine Learning. Springer, 2006.

Training (supervised)

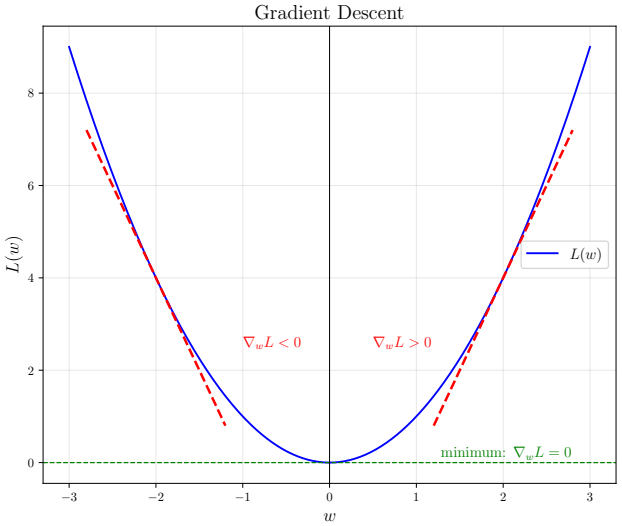
- Givet training data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, we want to minimise some error function, e.g., MSE

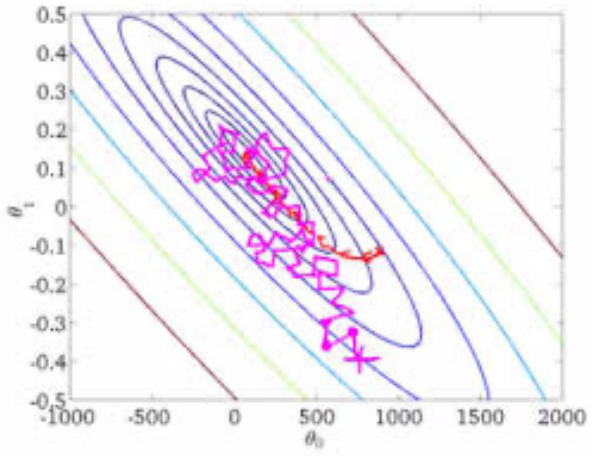
$$L(\phi) = \frac{1}{n}\sum_{i=1}^{n}\left(f_\phi(\mathbf{x}) - y_i\right)^2$$

- **No closed form solution** contrary to linear regression
- Minimize error by (stochastic) gradient decent
  - Initialise $\mathbf{w}^{(0)}$
  - Iterate

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}), \qquad \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \begin{pmatrix} \frac{\partial\mathcal{L}}{\partial w_1} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial w_m} \end{pmatrix}$$

backpropagation algorithm PyTorch





Today's exercises!

- A pen-and-paper notebook (`1.1 FNN Pen and Paper.ipynb`)
  - Neural networks (the feed-forward model)
  - Loss functions
  - Stochastic gradient descent
  - Error backpropagation

Contains concepts from week 2, but it rather self-contained.

- Four **problems** on shallow and deep neural networks

Thank you!