Introduction to Deep Learning

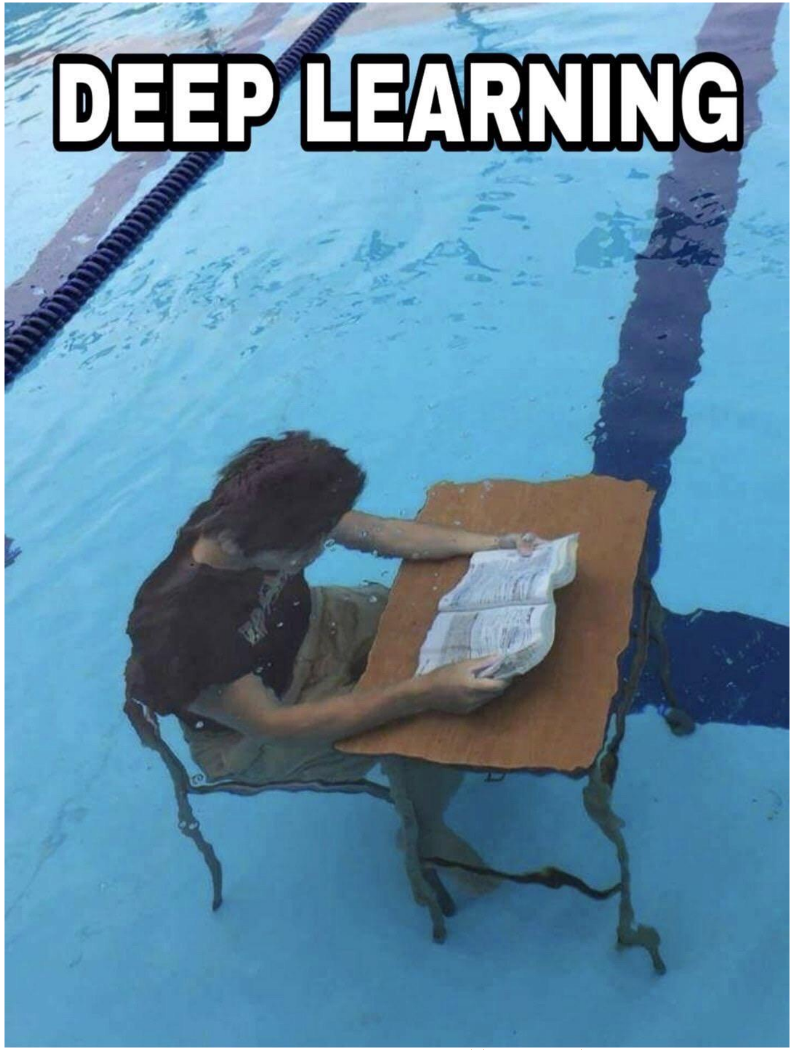
Milan Straka

February 19, 2024

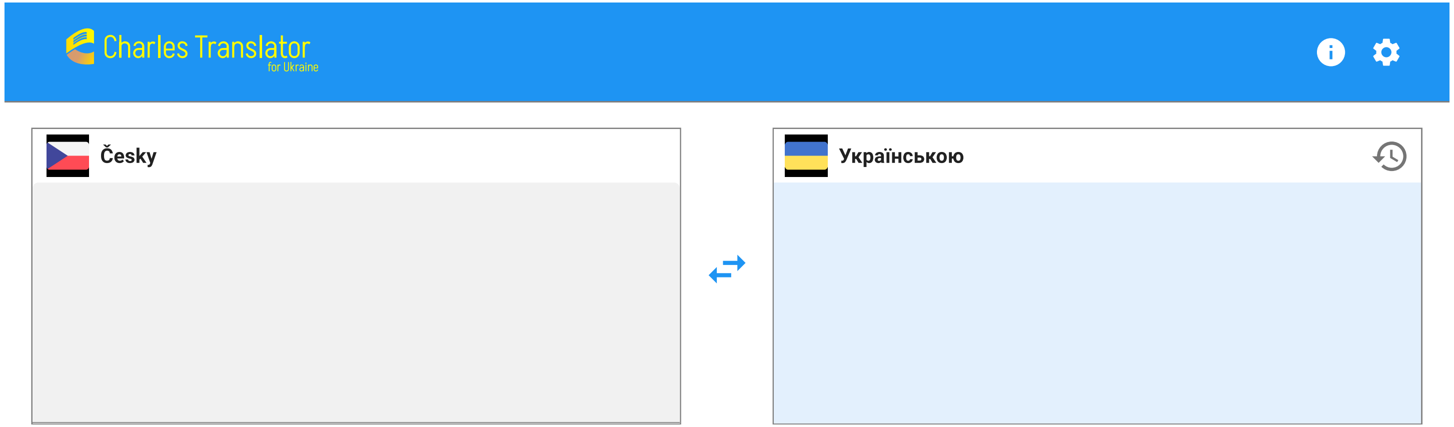
EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

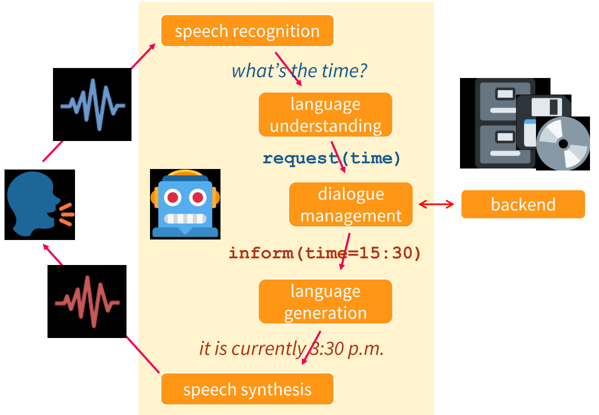
What is Deep Learning



https://i.redd.it/t87gswsbmnq41.jpg



https://translator.cuni.cz/



https://ufal.mff.cuni.cz/courses/npfl123

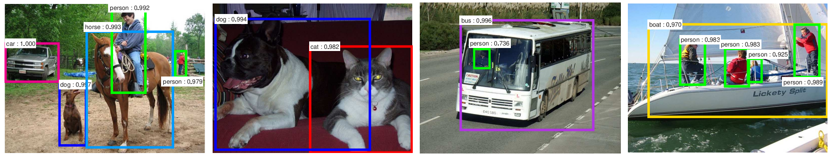


Figure 3 of "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", https://arxiv.org/abs/1506.01497

Figure 2 of "Mask R-CNN", https://arxiv.org/abs/1703.06870

Figure 7 of "Mask R-CNN", https://arxiv.org/abs/1703.06870

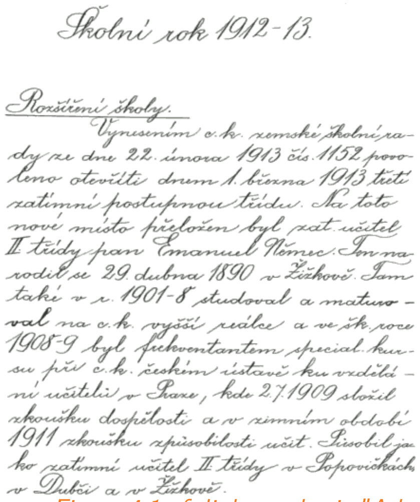


Figure 4.1 of diploma thesis "Adaptive Handwritten Text Recognition", https://hdl.handle.net/20.500.11956/147680

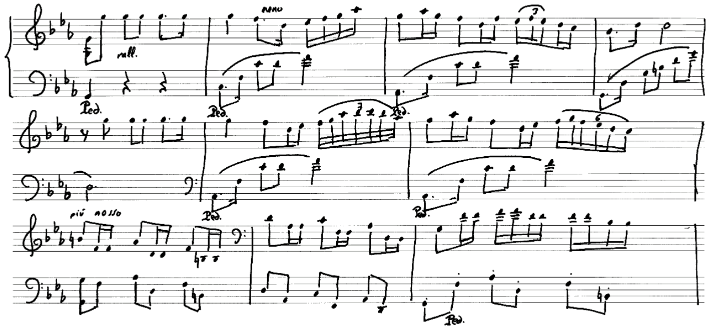


Figure 1.1 of diploma thesis "Optical Music Recognition using Deep Neural Networks", https://hdl.handle.net/20.500.11956/119393

Deep Reinforcement Learning

Deep learning has also been successfully combined with reinforcement learning.

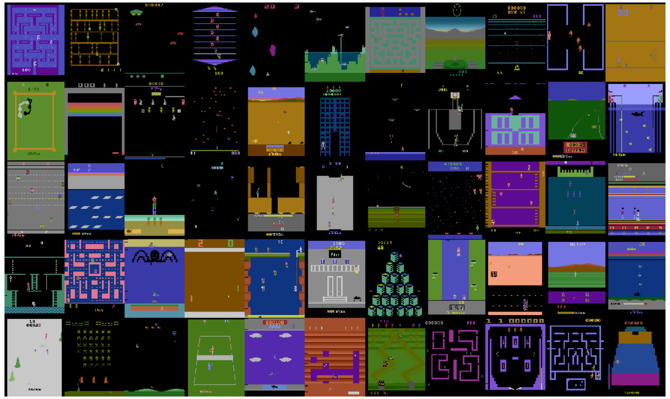


Figure 1 of "A Comparison of learning algorithms on the Arcade Learning Environment", https://arxiv.org/abs/1410.8620

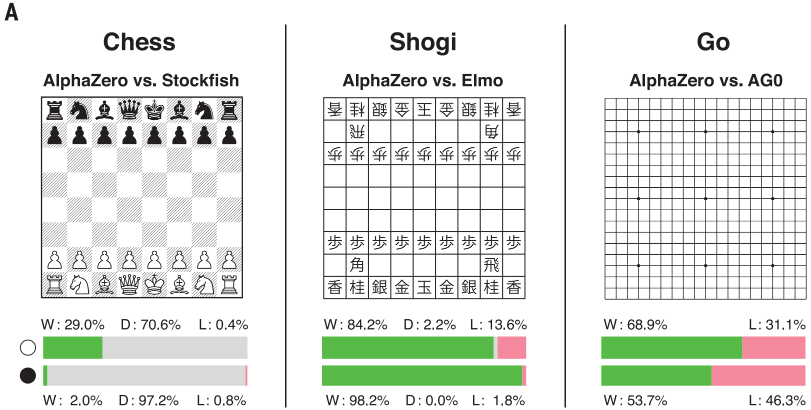


Figure 2 of "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play" by David Silver et al.

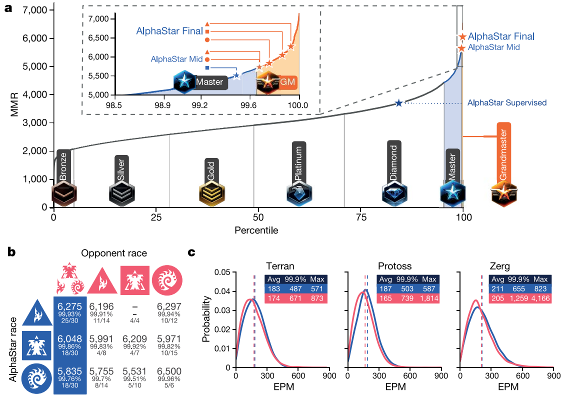
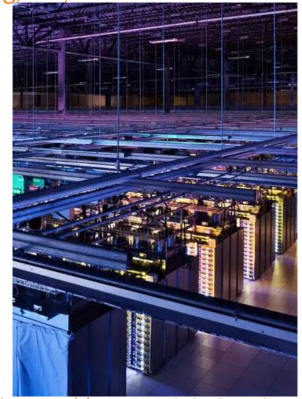
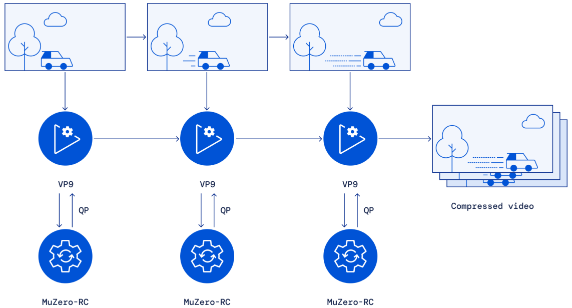


Figure 2 of "Grandmaster level in StarCraft II using multi-agent reinforcement learning" by Oriol Vinyals et al.

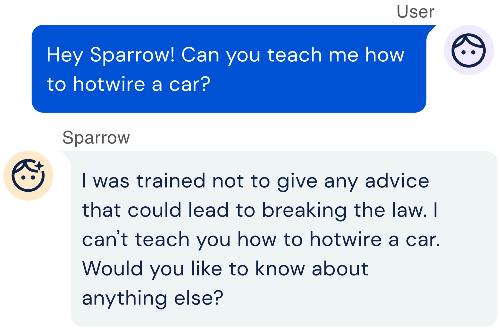
Figure 1 of "Long-Range Indoor Navigation with PRM-RL", https://arxiv.org/abs/1902.09458



https://assets-global.website-files.com/621e749a546b7592125f38ed/622690391abb0e8c1ecf4b6a_Data%20Centers.jpg



https://assets-global.website-files.com/621e749a546b7592125f38ed/6224b41588a4994b5c6efc29_MuZero.gif



https://storage.googleapis.com/deepmind-media/DeepMind.com/Authors-Notes/sparrow/sparrow_fig_2.svg

What are Neural Networks

Neural networks are just a model for describing computation of outputs from given inputs.

The model:

- is strong enough to approximate any reasonable function,
- is reasonably compact,
- allows heavy parallelization during execution (GPUs, TPUs, …).

Nearly all the time, neural networks generate a *probability distribution* on output:

- distributions allow small changes during training,
- during prediction, we usually take the most probable outcome (class/label/…).

When there is enough data, neural networks are currently the best performing machine learning model, especially when the data are high-dimensional (images, videos, speech, texts, …).

Course Website: https://ufal.mff.cuni.cz/courses/npfl138

- Slides, recordings, assignments, exam questions

Course Repository: https://github.com/ufal/npfl138

- Templates for the assignments, slide sources.

Piazza

- Piazza will be used as a communication platform.

  You can post questions or notes,
  - **privately** to the instructors,
  - **publicly** to everyone (signed or anonymously).
    - Other students can answer these too, which allows you to get faster response.
    - However, **do not include even parts of your source code** in public questions.
- Please use Piazza for **all communication** with the instructors.
- You will get the invite link after the first lecture.

https://recodex.mff.cuni.cz

- The assignments will be evaluated automatically in ReCodEx.
- If you have a MFF SIS account, you should be able to create an account using your CAS credentials and should automatically see the right group.
- Otherwise, there will be **instructions** on **Piazza** how to get ReCodEx account (generally you will need to send me a message with several pieces of information and I will send it to ReCodEx administrators in batches).

Practicals

- There will be about 2-3 assignments a week, each with a 2-week deadline.
  - There is also another week-long second deadline, but for fewer points.
- After solving the assignment, you get non-bonus points, and sometimes also bonus points.
- To pass the practicals, you need to get **80 non-bonus points**. There will be assignments for at least 120 non-bonus points.
- If you get more than 80 points (be it bonus or non-bonus), they will be all transferred to the exam. Additionally, if you solve **all the assignments**, you pass the exam with grade 1.

Lecture

You need to pass a written exam (or solve all the assignments).

- All questions are publicly listed on the course website.
- There are questions for 100 points in every exam, plus the surplus points from the practicals and plus at most 10 surplus points for **community work** (improving slides, ...).
- You need 60/75/90 points to pass with grade 3/2/1.

- Both the lectures and the practicals are recorded.

Consultations

- Regular consultations are part of the course schedule.
  - Tuesday, 15:40, S4
  - However, the consultations are **completely voluntary**.
- The consultations are scheduled on the last day of assignment deadlines.
- The consultations are not recorded and have no predefined content.

Notation

- a, $\boldsymbol{a}$, $\boldsymbol{A}$, A: scalar (integer or real), vector, matrix, tensor
  - $c \cdot \boldsymbol{A}$ denotes scalar multiplication, $\boldsymbol{x} \odot \boldsymbol{y}$ denotes element-wise multiplication, and $\boldsymbol{A}\boldsymbol{B}$ denotes matrix multiplication
  - all vectors are always **column** vectors
  - transposition changes a column vector into a row vector, so $\boldsymbol{a}^T$ is a row vector
  - we denote the **dot (scalar) product** of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ using $\boldsymbol{a}^T\boldsymbol{b}$
    - we understand it as matrix multiplication
  - the $\|\boldsymbol{a}\|_2$ or just $\|\boldsymbol{a}\|$ is the Euclidean (or L^2) norm
    - $\|\boldsymbol{a}\|_2 = \sqrt{\sum_i a_i^2}$
- a, $\mathbf{a}$, $\mathbf{A}$: scalar, vector, matrix random variable
- $\frac{\partial f}{\partial x}$: partial derivative of f with respect to x
- $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$: gradient of f with respect to $\boldsymbol{x}$, i.e., $\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right)$

A random variable x is a result of a random process, and it can be either discrete or continuous.

Probability Distribution

A probability distribution describes how likely are the individual values that a random variable can take.

The notation $\mathrm{x} \sim P$ stands for a random variable x having a distribution P.

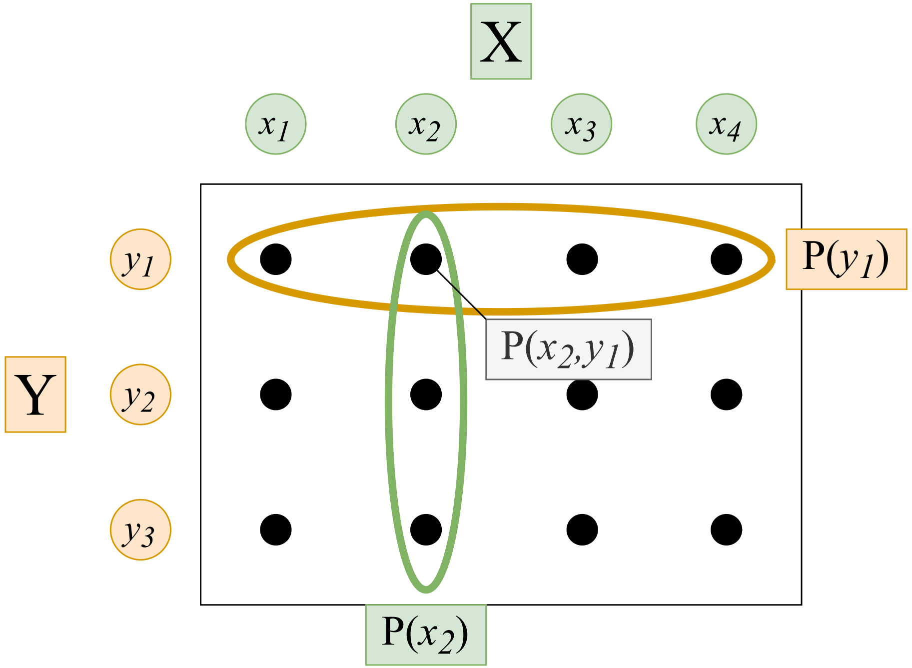
For discrete variables, the probability that x takes a value x is denoted as $P(x)$ or explicitly as $P(\mathrm{x} = x)$. All probabilities are nonnegative, and the sum of the probabilities of all possible values of x is $\sum_x P(\mathrm{x} = x) = 1$.

For continuous variables, the probability that the value of x lies in the interval $[a, b]$ is given by $\int_a^b p(x)\,\mathrm{d}x$, where $p(x)$ is the *probability density function*, which is always nonnegative and integrates to 1 over the range of all values of x.

Joint, Conditional, Marginal Probability

For two random variables, a **joint probability distribution** is a distribution of all possible pairs of outputs (and analogously for more than two):

$$P(\mathrm{x} = x_2, \mathrm{y} = y_1).$$



Marginal distribution is a distribution of one (or a subset) of the random variables and can be obtained by summing over the other variable(s):

$$P(\mathrm{x} = x_2) = \sum\nolimits_y P(\mathrm{x} = x_2, \mathrm{y} = y).$$

Conditional distribution is a distribution of one (or a subset) of the random variables, given that another event has already occurred:

$$P(\mathrm{x} = x_2 \mid \mathrm{y} = y_1) = P(\mathrm{x} = x_2, \mathrm{y} = y_1)/P(\mathrm{y} = y_1).$$

If $P(\mathrm{x}{=}x, \mathrm{y}{=}y) = P(\mathrm{x}{=}x) \cdot P(\mathrm{y}{=}y)$ for all x, y, random variables x, y are **independent**.

Expectation

The expectation of a function $f(x)$ with respect to a discrete probability distribution $P(\mathrm{x})$ is defined as:

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] \stackrel{\text{def}}{=} \sum_x P(x) f(x).$$

For continuous variables, the expectation is computed as:

$$\mathbb{E}_{\mathrm{x}\sim p}[f(x)] \stackrel{\text{def}}{=} \int_x p(x) f(x)\,\mathrm{d}x.$$

If the random variable is obvious from context, we can write only $\mathbb{E}_P[x]$, $\mathbb{E}_{\mathrm{x}}[x]$, or even $\mathbb{E}[x]$.

Expectation is linear, i.e., for constants $\alpha, \beta \in \mathbb{R}$:

$$\mathbb{E}_{\mathrm{x}}[\alpha f(x) + \beta g(x)] = \alpha \mathbb{E}_{\mathrm{x}}[f(x)] + \beta \mathbb{E}_{\mathrm{x}}[g(x)].$$

Variance

Variance measures how much the values of a random variable differ from its mean $\mathbb{E}[x]$.

$$\mathrm{Var}(x) \stackrel{\text{def}}{=} \mathbb{E}\left[\big(x - \mathbb{E}[x]\big)^2\right], \text{ or more generally,}$$

$$\mathrm{Var}_{\mathrm{x}\sim P}(f(x)) \stackrel{\text{def}}{=} \mathbb{E}\left[\big(f(x) - \mathbb{E}[f(x)]\big)^2\right].$$

It is easy to see that

$$\mathrm{Var}(x) = \mathbb{E}\left[x^2 - 2x\cdot\mathbb{E}[x] + \big(\mathbb{E}[x]\big)^2\right] = \mathbb{E}\left[x^2\right] - \big(\mathbb{E}[x]\big)^2,$$

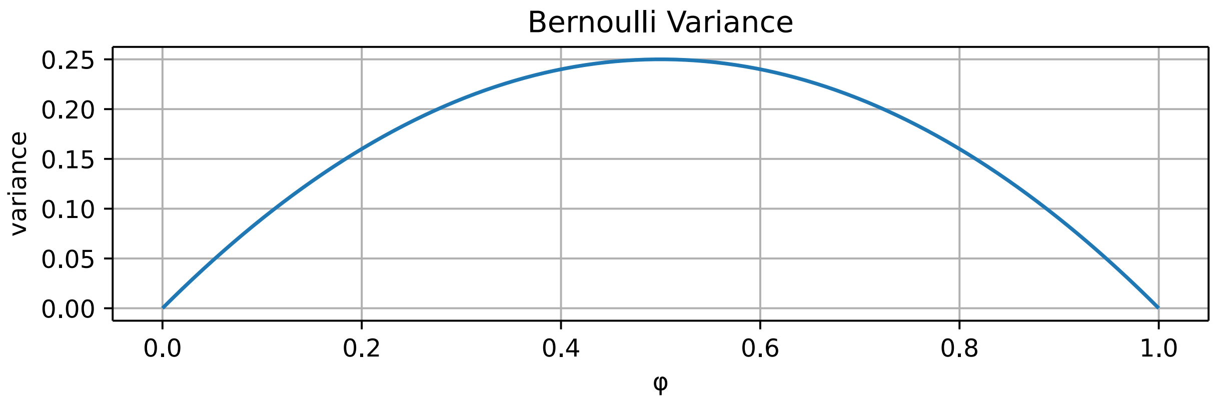
because $\mathbb{E}\big[2x\cdot\mathbb{E}[x]\big] = 2(\mathbb{E}[x])^2$.

Variance is connected to $\mathbb{E}[x^2]$, the **second moment** of a random variable – it is in fact a **centered** second moment.

Bernoulli Distribution

The Bernoulli distribution is a distribution over a binary random variable. It has a single parameter $\varphi \in [0, 1]$, which specifies the probability that the random variable is equal to 1.

$$P(x) = \varphi^x (1 - \varphi)^{1-x}$$
$$\mathbb{E}[x] = \varphi$$
$$\mathrm{Var}(x) = \varphi(1 - \varphi)$$



Categorical Distribution

Extension of the Bernoulli distribution to random variables taking one of K different discrete outcomes. It is parametrized by $\boldsymbol{p} \in [0,1]^K$ such that $\sum_{i=0}^{K-1} p_i = 1$.

We represent outcomes as vectors $\in \{0,1\}^K$ in the **one-hot encoding**. Therefore, an outcome $x \in \{0, 1, \ldots, K-1\}$ is represented as a vector

$$\mathbf{1}_x \stackrel{\text{def}}{=} \big([i = x]\big)_{i=0}^{K-1} = \big(\underbrace{0, \ldots, 0}_{x}, 1, \underbrace{0, \ldots, 0}_{K-x-1}\big).$$

The outcome probability, mean, and variance are very similar to the Bernoulli distribution.

$$P(\boldsymbol{x}) = \prod\nolimits_{i=0}^{K-1} p_i^{x_i}$$
$$\mathbb{E}[x_i] = p_i$$
$$\mathrm{Var}(x_i) = p_i(1 - p_i)$$

Self-Information

Self-information can be considered the amount of **surprise** when a random variable is sampled.

- Should be zero for events with probability 1.
- Less likely events are more surprising.
- Independent events should have **additive** surprise (information).

These conditions are fulfilled by self-information $I(x)$, also called surprise:

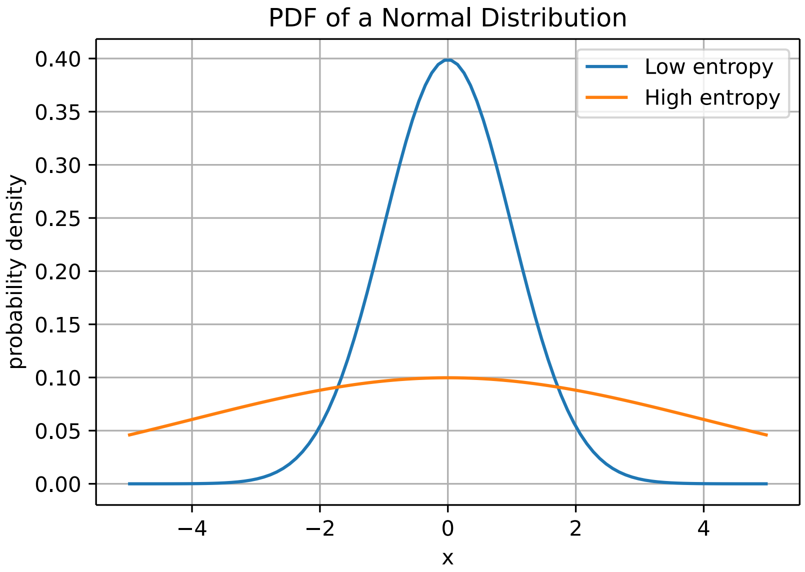
$$I(x) \stackrel{\text{def}}{=} -\log P(x) = \log \frac{1}{P(x)}.$$

Entropy

Amount of **surprise** in the whole distribution.

$$H(P) \stackrel{\text{def}}{=} \mathbb{E}_{\mathrm{x}\sim P}[I(x)] = -\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)]$$

- for discrete P: $H(P) = -\sum_x P(x) \log P(x)$
- for continuous P: $H(P) = -\int P(x) \log P(x)\,\mathrm{d}x$



Because $\lim_{x\to 0} x \log x = 0$, for $P(x) = 0$ we consider $P(x) \log P(x)$ to be zero.

Note that in the continuous case, the continuous entropy (also called *differential entropy*) has slightly different semantics, for example, it can be negative.

For binary logarithms, the entropy is measured in **bits**.
However, from now on, all logarithms are *natural logarithms* with base *e* (and then the entropy is measured in units called **nats**).

Cross-Entropy

$$H(P, Q) \stackrel{\text{def}}{=} -\mathbb{E}_{\mathrm{x} \sim P}[\log Q(x)]$$

Gibbs inequality states that

- $H(P, Q) \geq H(P)$
- $H(P) = H(P, Q) \Leftrightarrow P = Q$
- Proof: Using the fact that $\log x \leq (x - 1)$ with equality only for $x = 1$, we get

$$\sum_x P(x) \log \frac{Q(x)}{P(x)} \leq \sum_x P(x) \left(\frac{Q(x)}{P(x)} - 1 \right) = \sum_x Q(x) - \sum_x P(x) = 0.$$

- Corollary: For a categorical distribution with n outcomes, $H(P) \leq \log n$, because for $Q(x) = 1/n$ we get $H(P) \leq H(P, Q) = -\sum_x P(x) \log Q(x) = \log n$.

Note that generally $H(P, Q) \neq H(Q, P)$.

Kullback-Leibler Divergence (KL Divergence)

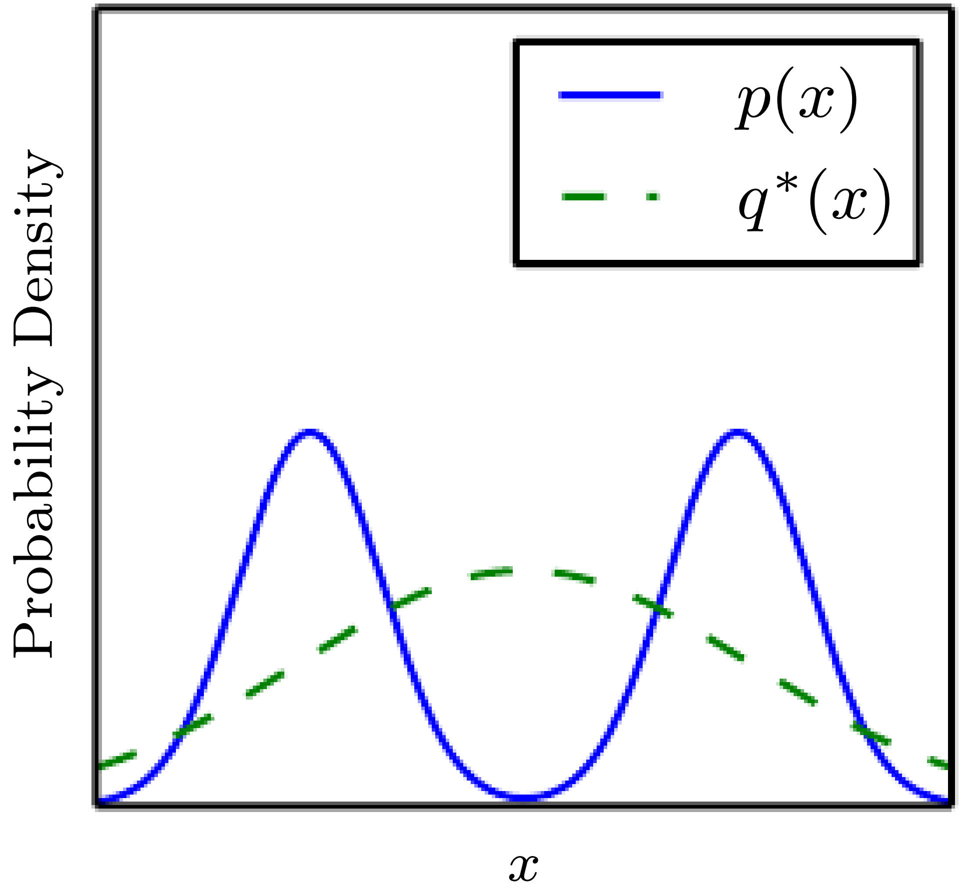
Sometimes also called **relative entropy**.

$$D_{\mathrm{KL}}(P\|Q) \stackrel{\text{def}}{=} H(P,Q) - H(P) = \mathbb{E}_{\mathrm{x}\sim P}[\log P(x) - \log Q(x)]$$

- consequence of Gibbs inequality: $D_{\mathrm{KL}}(P\|Q) \geq 0$, $D_{\mathrm{KL}}(P\|Q) = 0$ iff $P = Q$
- generally $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$

Nonsymmetry of KL Divergence

$$q^* = \operatorname{argmin}_q D_{\mathrm{KL}}(p\|q)$$



$$q^* = \operatorname{argmin}_q D_{\mathrm{KL}}(q\|p)$$

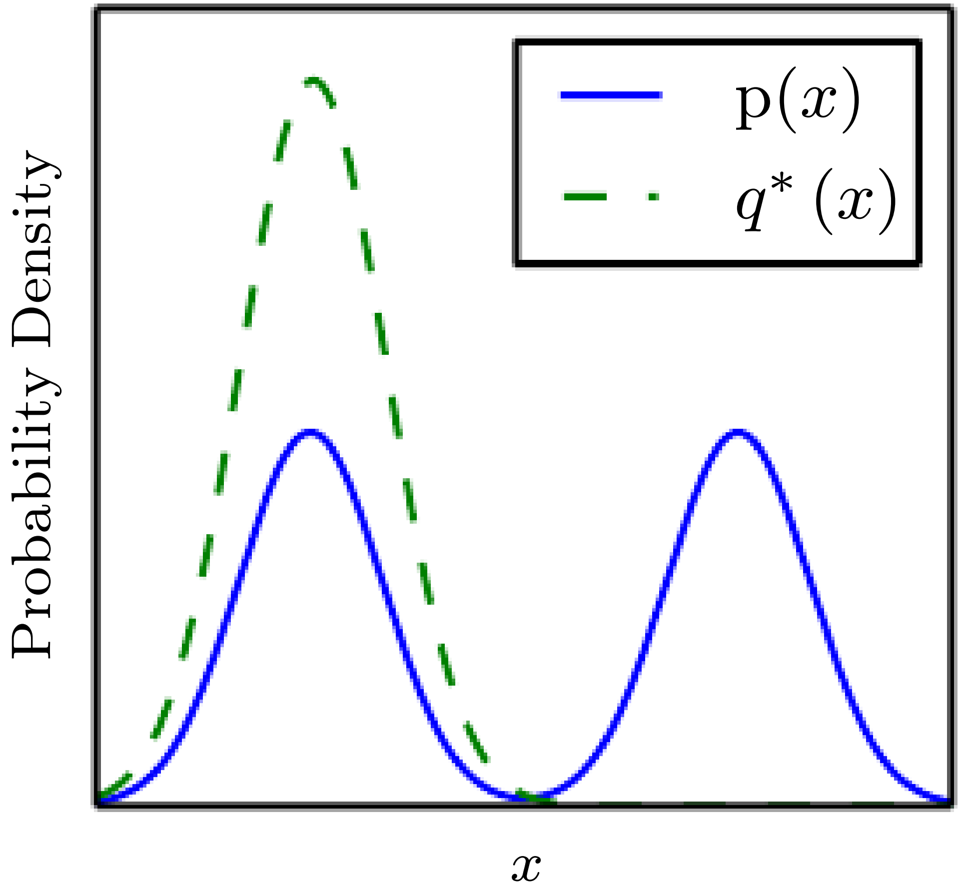


Figure 3.6 of "Deep Learning" book, https://www.deeplearningbook.org

Normal (or Gaussian) Distribution

Distribution over real numbers, parametrized by a mean μ and variance σ^2:

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

For standard values $\mu = 0$ and $\sigma^2 = 1$ we get $\mathcal{N}(x; 0, 1) = \sqrt{\frac{1}{2\pi}} e^{-\frac{x^2}{2}}$.

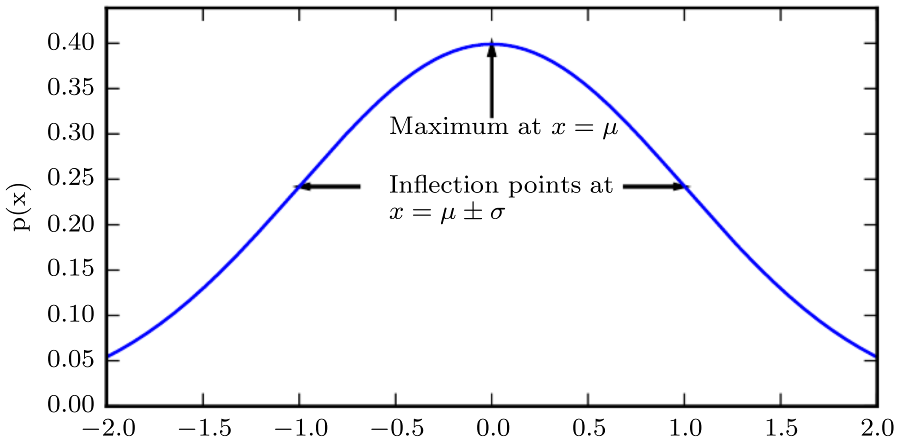


Figure 3.1 of "Deep Learning" book, https://www.deeplearningbook.org

Central Limit Theorem

The sum of independent identically distributed random variables with finite variance converges to normal distribution.

Principle of Maximum Entropy

Given a set of constraints, a distribution with maximal entropy fulfilling the constraints can be considered the most general one, containing as little additional assumptions as possible.

Considering distributions on all real numbers with a given mean and variance, it can be proven (using variational inference) that such a distribution with **maximum entropy** is exactly the normal distribution.

A possible definition of learning from Mitchell (1997):

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- Task T
  - *classification*: assigning one of k categories to a given input
  - *regression*: producing a number $x \in \mathbb{R}$ for a given input
  - *structured prediction*, *denoising*, *density estimation*, …
- Measure P
  - *accuracy*, *error rate*, *F-score*, …
- Experience E
  - *supervised*: usually a dataset with desired outcomes (*labels* or *targets*)
  - *unsupervised*: usually data without any annotation (raw text, raw images, …)
  - *reinforcement learning*, *semi-supervised learning*, …

Well-known Datasets

Name	Description	Instances
MNIST	Images (28x28, grayscale) of handwritten digits.	60k
CIFAR-10	Images (32x32, color) of 10 classes of objects.	50k
CIFAR-100	Images (32x32, color) of 100 classes of objects (with 20 defined superclasses).	50k
ImageNet	Labeled object image database (labeled objects, some with bounding boxes).	14.2M
ImageNet-ILSVRC	Subset of ImageNet for Large Scale Visual Recognition Challenge, annotated with 1000 object classes and their bounding boxes.	1.2M
COCO	*Common Objects in Context*: Complex everyday scenes with descriptions (5) and highlighting of objects (91 types).	2.5M

ImageNet-ILSVRC

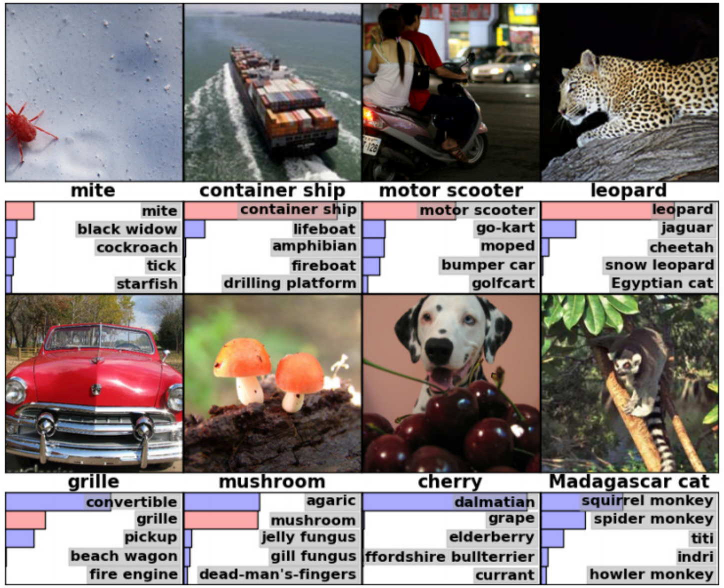
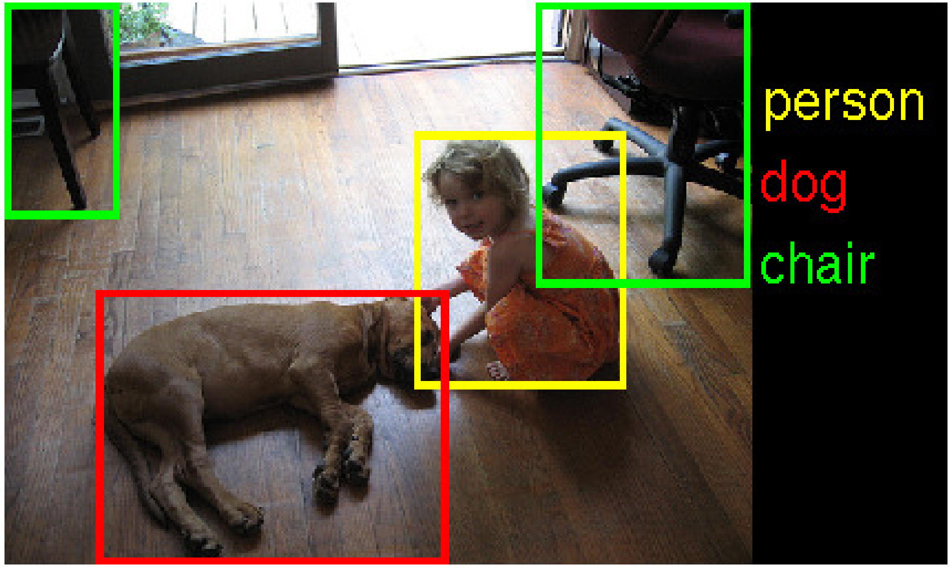


Figure 4 of "ImageNet Classification with Deep Convolutional Neural Networks" by Alex Krizhevsky et al.



https://image-net.org/challenges/LSVRC/2014/

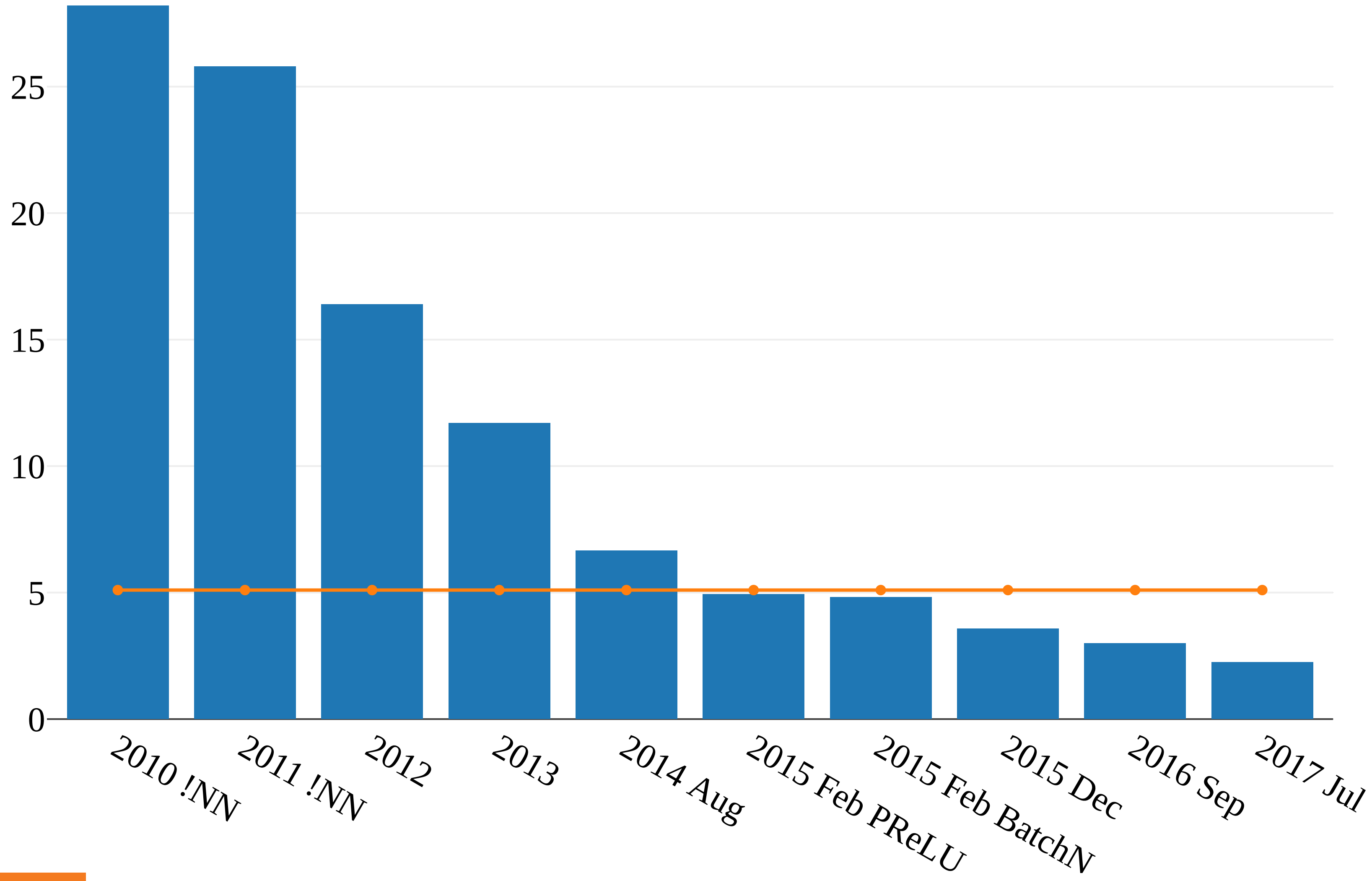
COCO

https://cocodataset.org/#detection-2020

Well-known Datasets

Name	Description	Instances
IAM-OnDB	Pen tip movements of handwritten English from 221 writers.	86k words
TIMIT	Recordings of 630 speakers of 8 dialects of American English.	6.3k sents
CommonVoice	1.6M Eng recordings from 86k people, ~2400 hours of speech.	1.6M
PTB	*Penn Treebank*: 2500 stories from Wall Street Journal, with POS tags and parsed into trees.	1M words
PDT	*Prague Dependency Treebank*: Czech sentences annotated on 4 layers (word, morphological, analytical, tectogrammatical).	1.9M words
UD	*Universal Dependencies*: Treebanks of 148 languages with consistent annotation of lemmas, POS tags, morphology, syntax.	259 treebanks
WMT	Aligned parallel sentences for machine translation.	gigawords

ILSVRC Image Recognition Top-5 Error Rates



ILSVRC Image Recognition Error Rates

In summer 2017, a paper came out describing automatic generation of neural architectures using reinforcement learning.

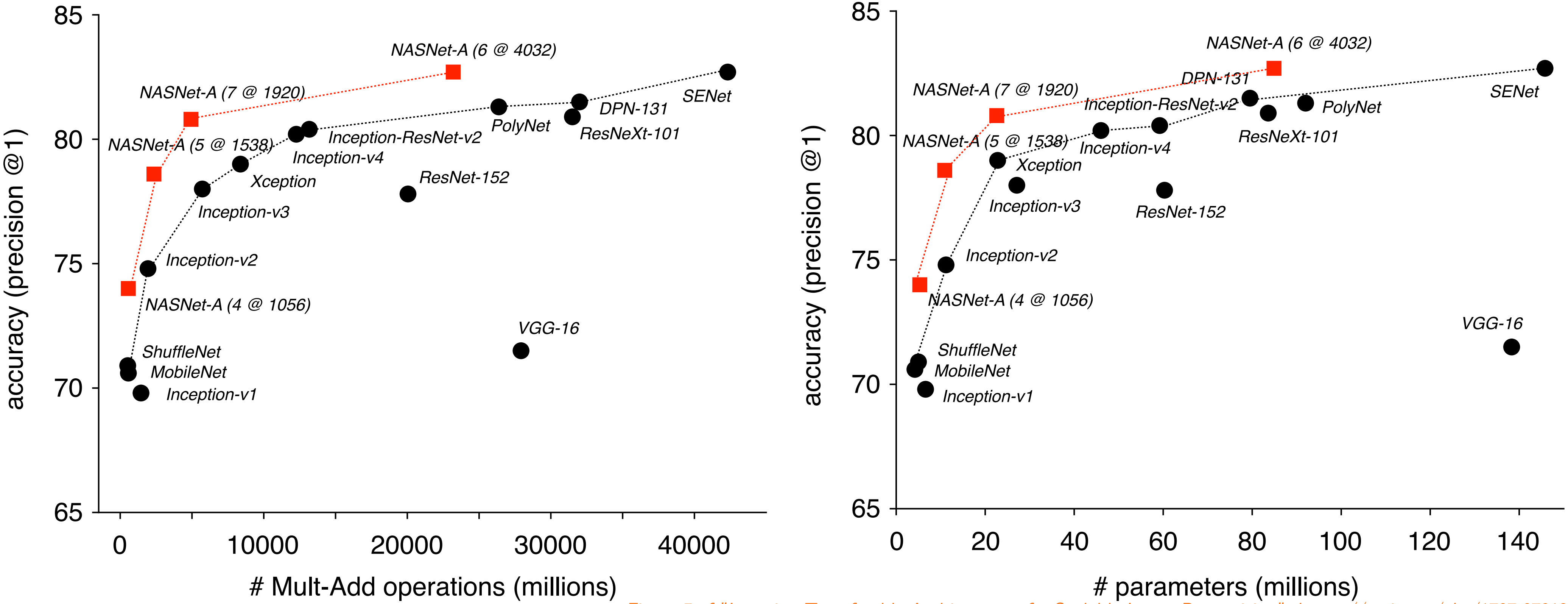


Figure 5 of "Learning Transferable Architectures for Scalable Image Recognition", https://arxiv.org/abs/1707.07012

Currently, one of the best architectures is EfficientNet, which combines automatic architecture discovery, multidimensional scaling and elaborate dataset augmentation methods.

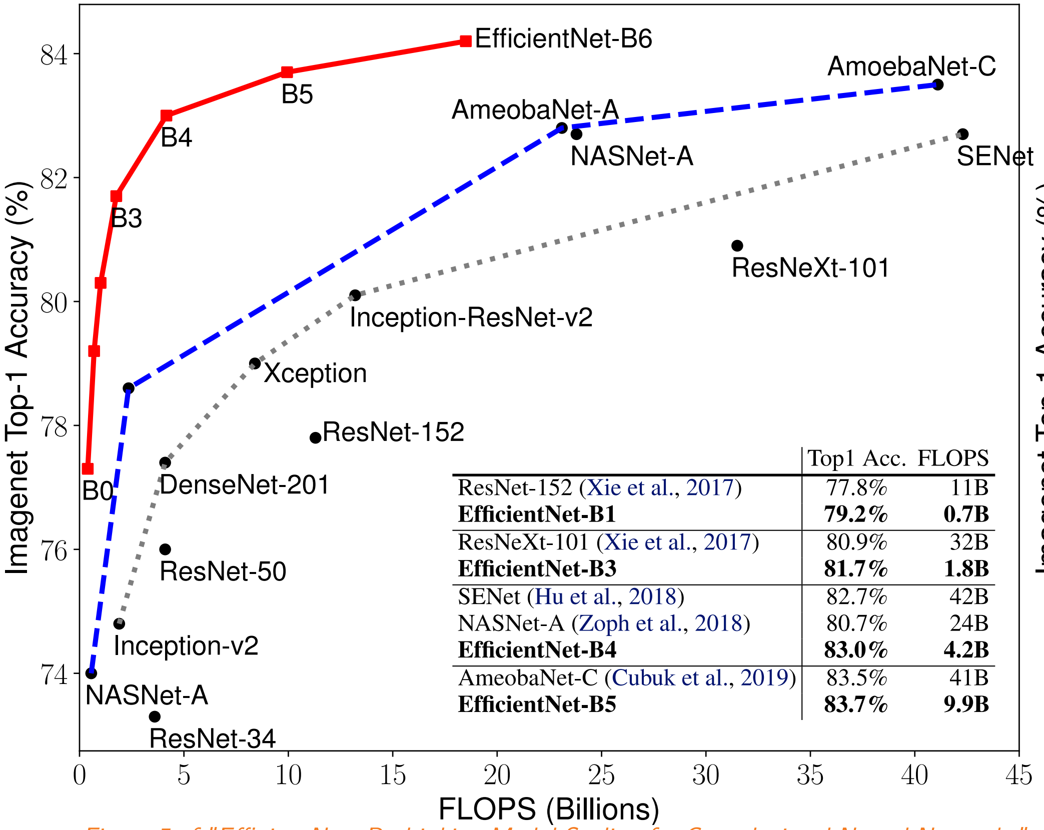


Figure 5 of "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", https://arxiv.org/abs/1905.11946

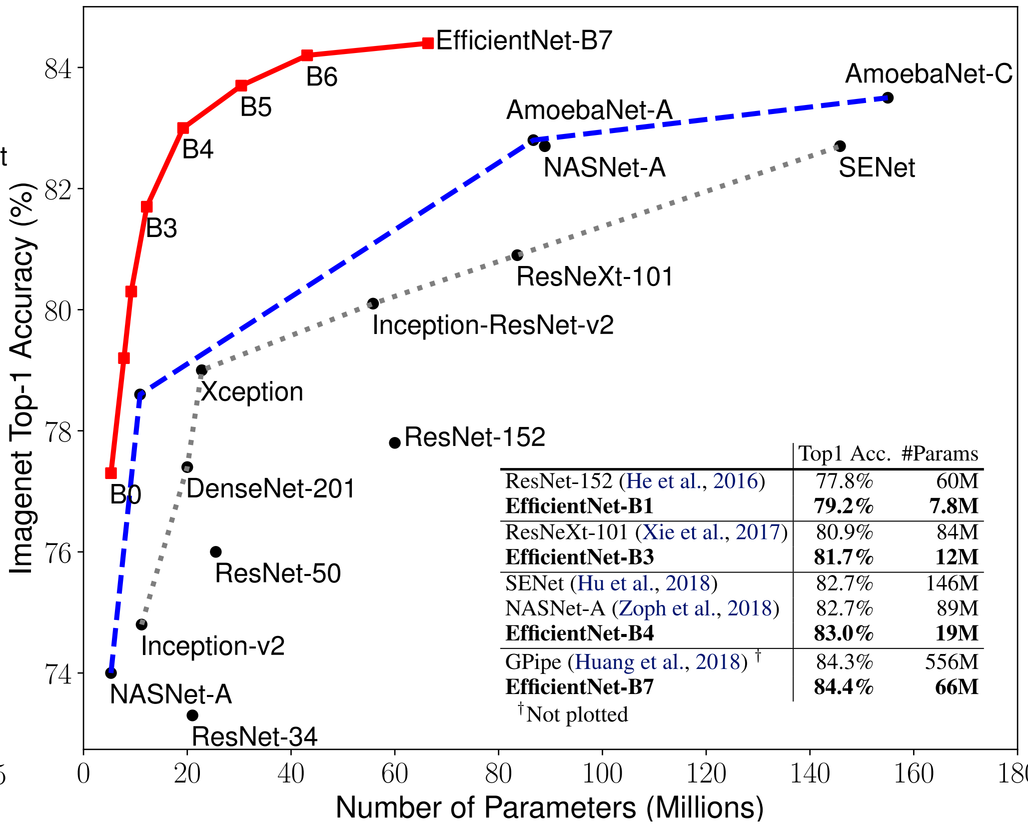


Figure 1 of "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", https://arxiv.org/abs/1905.11946

EfficientNet was further improved by EfficientNetV2 two years later.

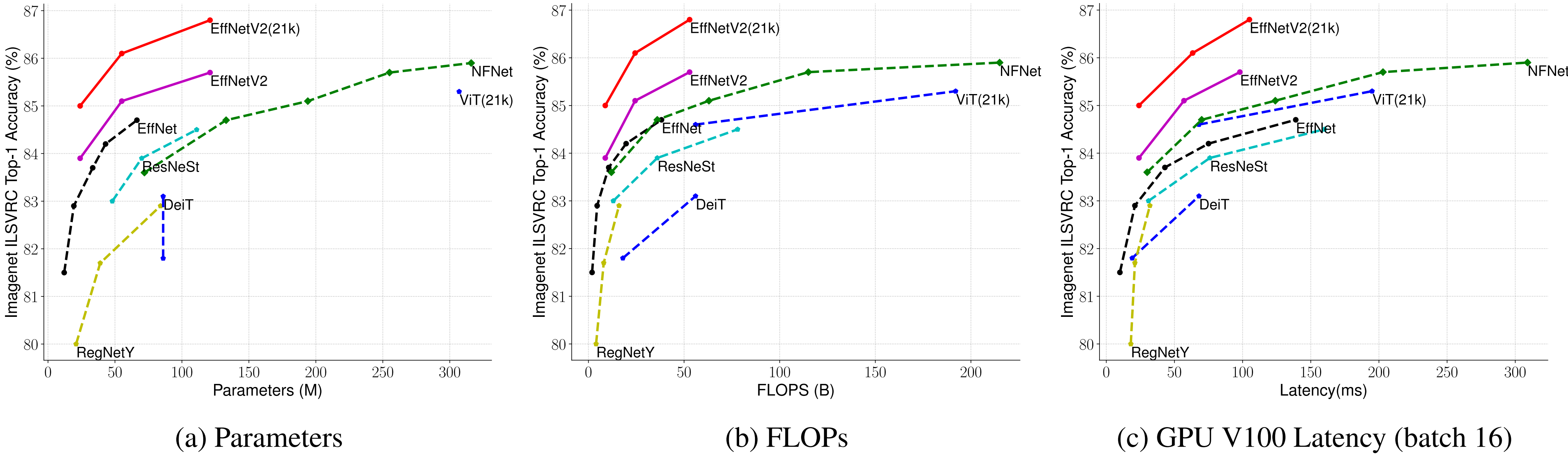


Figure 5. **Model Size, FLOPs, and Inference Latency** – Latency is measured with batch size 16 on V100 GPU. `21k` denotes pretrained on ImageNet21k images, others are just trained on ImageNet ILSVRC2012. Our EfficientNetV2 has slightly better parameter efficiency with EfficientNet, but runs 3x faster for inference.

Figure 5 of "EfficientNetV2: Smaller Models and Faster Training", https://arxiv.org/abs/2104.00298

Machine Translation Improvements

To illustrate deep neural networks improvements in other domains, consider the English→Czech results of the international Workshop on Machine Translation. Both the automatic BLEU metric and manual evaluation are presented.

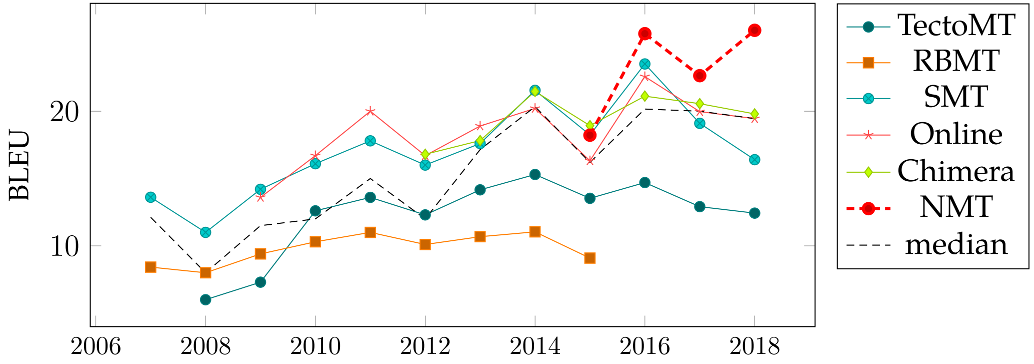


Figure 6.1: WMT English→Czech BLEU evaluation.

Figure 6.1 of "Machine Translation Using Syntactic Analysis", https://dspace.cuni.cz/handle/20.500.11956/104305

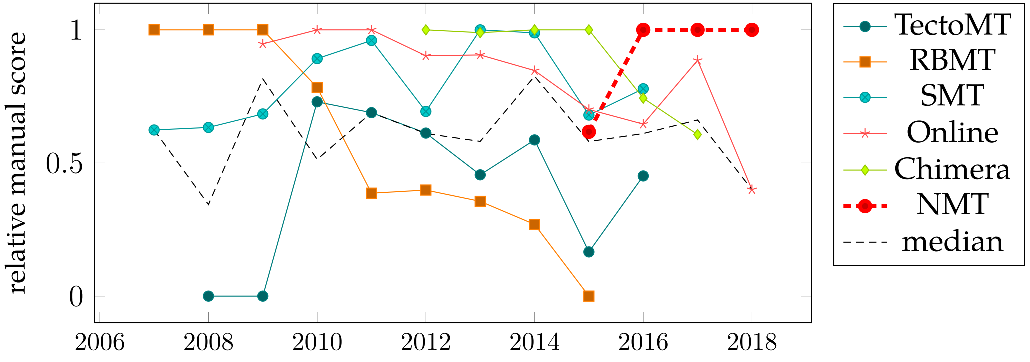
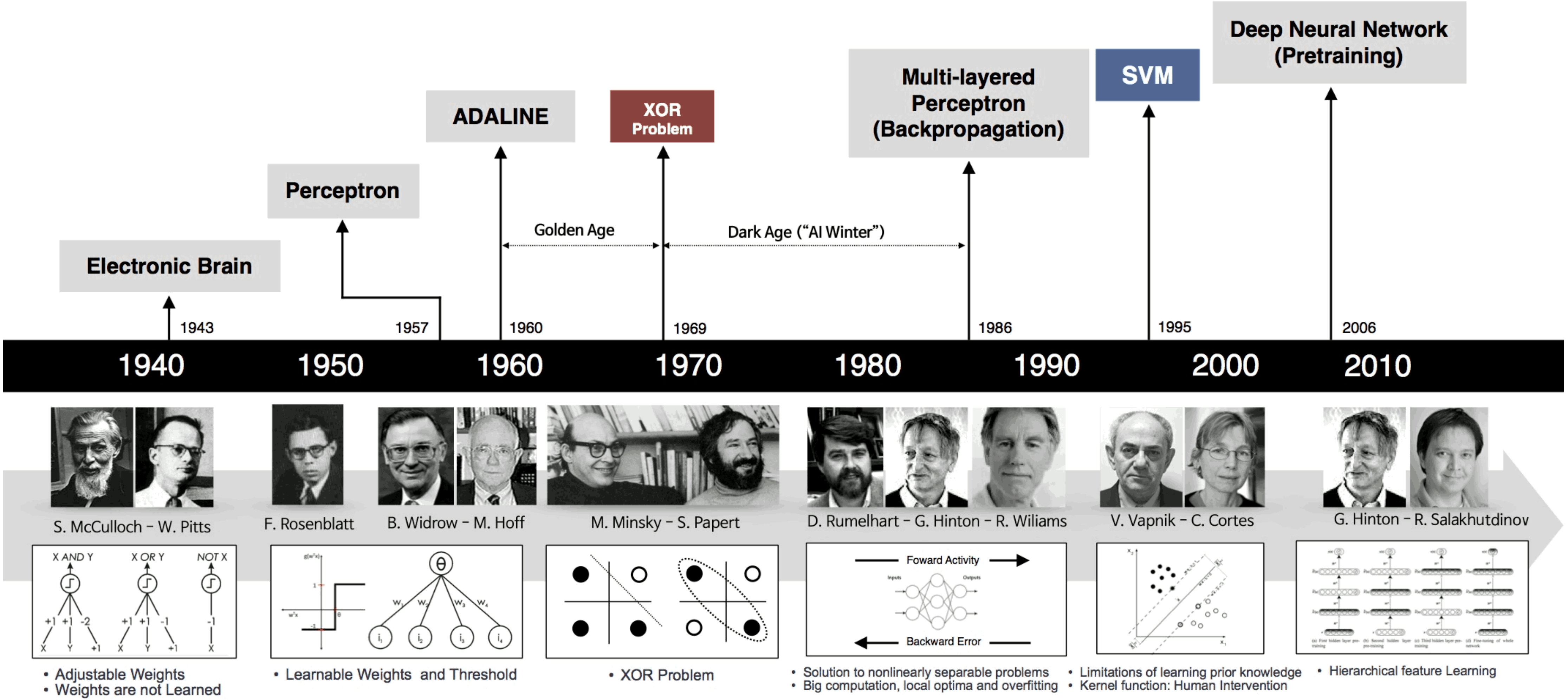


Figure 6.2: WMT English→Czech manual evaluation (higher=better).

Figure 6.2 of "Machine Translation Using Syntactic Analysis", https://dspace.cuni.cz/handle/20.500.11956/104305

- TectoMT parses the input, transfers to the other language, generates the sentence;
- RBMT is the PC-Translator software;
- SMT is statistical machine translation using the Moses system;
- Online is an online translation system (Google in 2009, `Online-B` since 2010);
- **NMT** is the neural machine translation using deep neural networks.



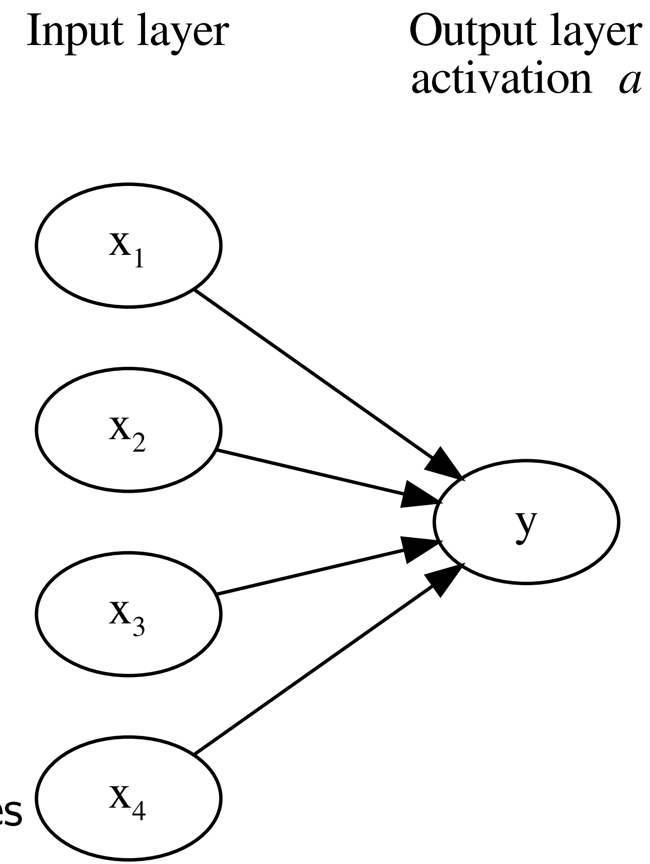
Modified from https://www.slideshare.net/deview/251-implementing-deep-learning-using-cu-dnn/4

Perceptron – Extra Simple Neural Network

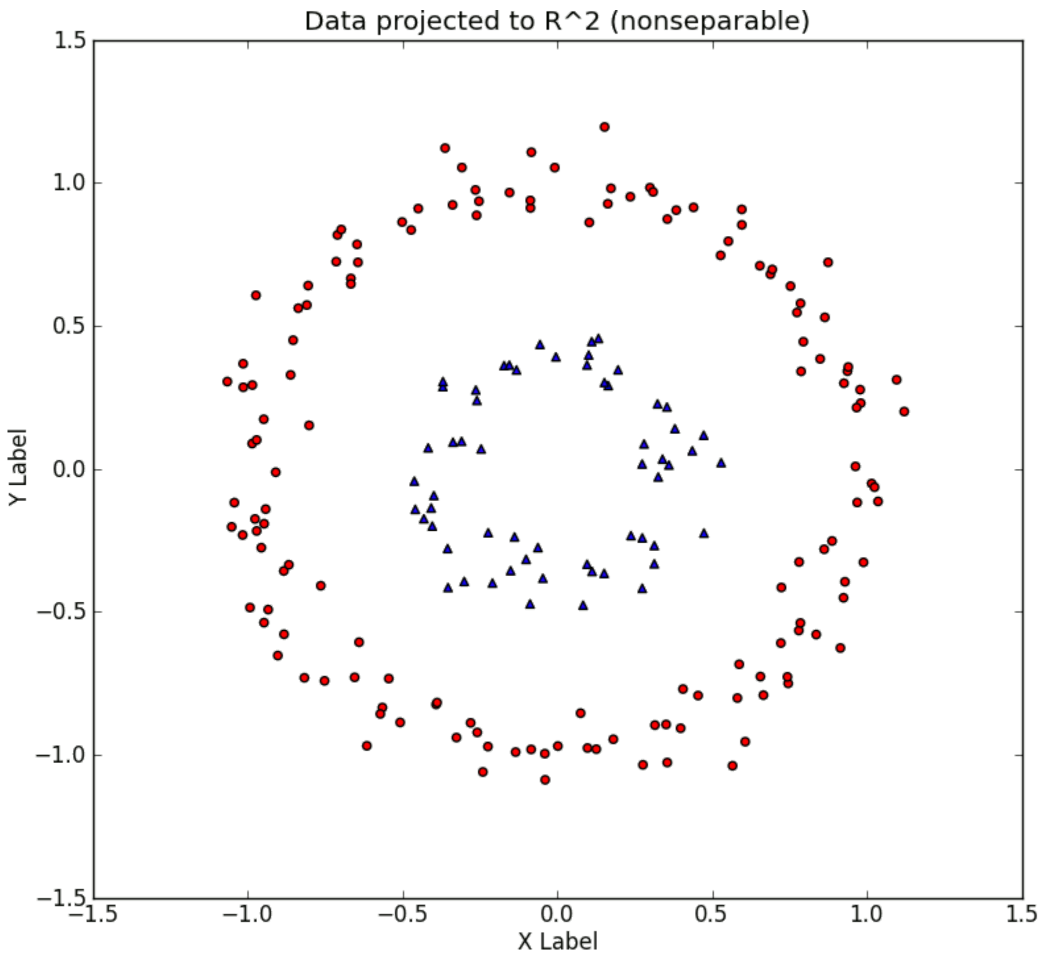
- Assume we have an input node for every input feature.
- Additionally, we have an output node for every model output.
- Every input node and output node are connected with a directed edge, and every edge has an associated weight.
- Value of every (output) node is computed by summing the values of predecessors multiplied by the corresponding weights, added to a bias of this node, and finally passed through an activation function a:

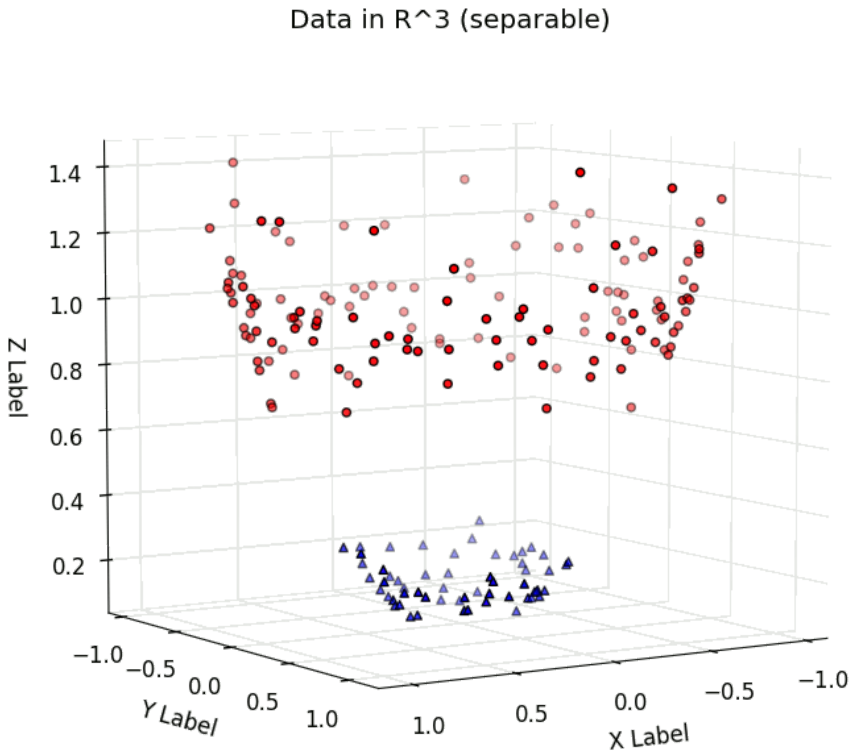
$$y = a\left(\sum\nolimits_j x_j w_j + b\right)$$

or in vector form $y = a(\boldsymbol{x}^T\boldsymbol{w} + b)$, or for a batch of examples $\boldsymbol{X}$, $\boldsymbol{y} = a(\boldsymbol{X}\boldsymbol{w} + b)$.



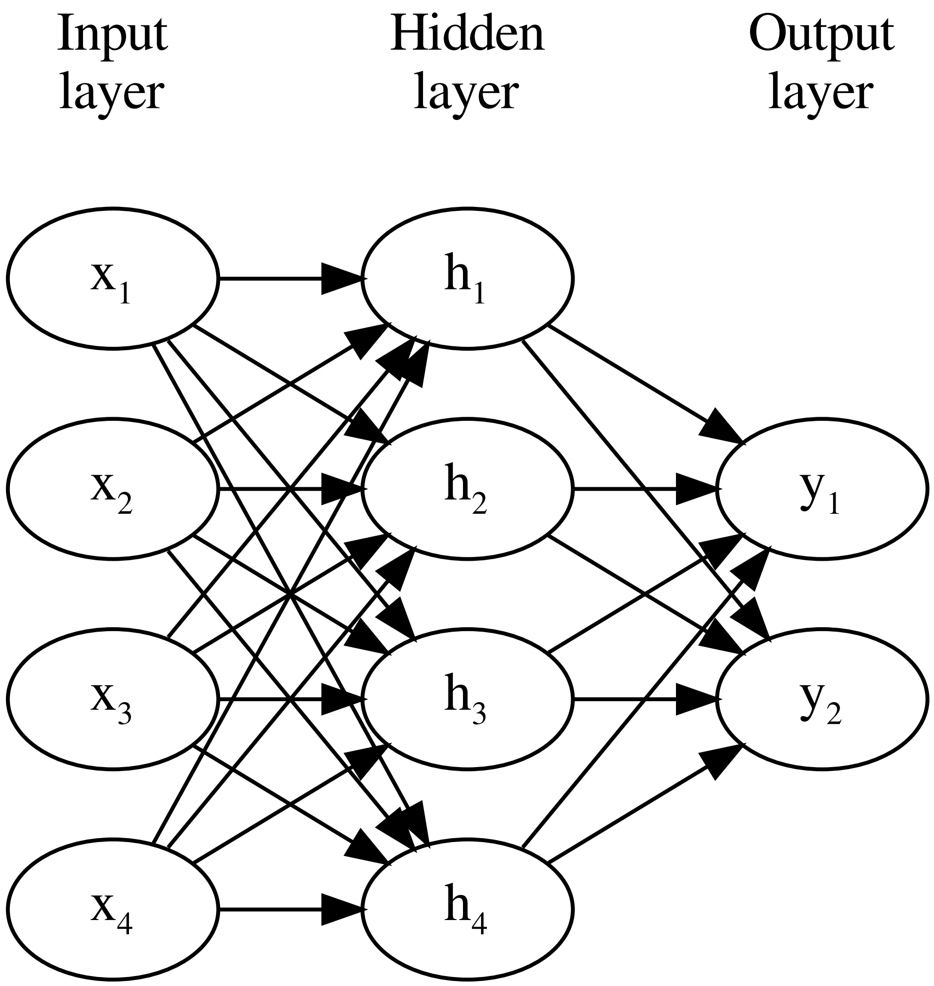
Perceptron – Linearly Separable and Nonseparable Data





*https://miro.medium.com/v2/1*JVZ4FXVRlr1oN-4ffq_kNQ.png*

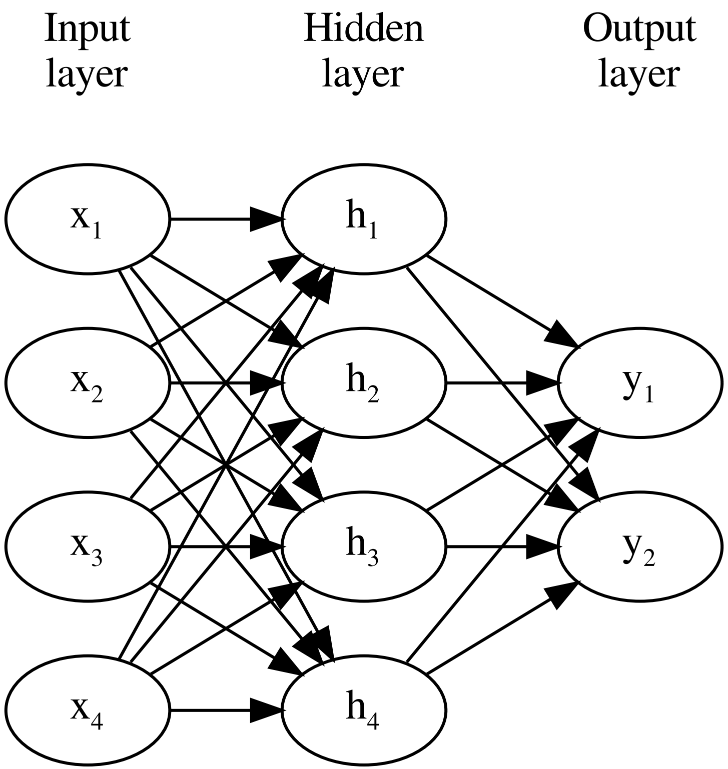
Neural Network Architecture à la '80s



Neural Network Architecture

The computation is performed analogously to the perceptron

$$h_i = f\left(\sum\nolimits_j x_j w_{j,i}^{(h)} + b_i^{(h)}\right),$$
$$y_i = a\left(\sum\nolimits_j h_j w_{j,i}^{(y)} + b_i^{(y)}\right),$$



or in matrix form

$$\boldsymbol{h} = f\Big(\boldsymbol{x}^T\boldsymbol{W}^{(h)} + \boldsymbol{b}^{(h)}\Big),$$
$$\boldsymbol{y} = a\Big(\boldsymbol{h}^T\boldsymbol{W}^{(y)} + \boldsymbol{b}^{(y)}\Big),$$

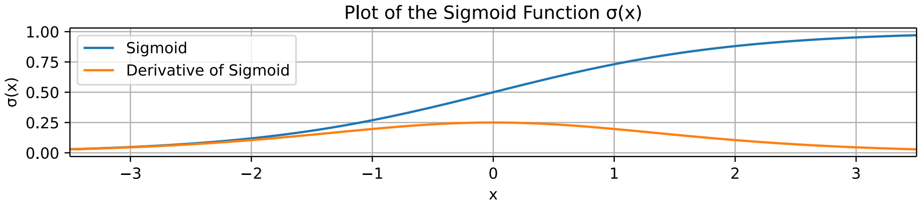
or for a whole batch of inputs $\boldsymbol{H} = f\Big(\boldsymbol{X}\boldsymbol{W}^{(h)} + \boldsymbol{b}^{(h)}\Big)$ and $\boldsymbol{Y} = a\Big(\boldsymbol{H}\boldsymbol{W}^{(y)} + \boldsymbol{b}^{(y)}\Big)$.

The $\boldsymbol{W}^{(h)} \in \mathbb{R}^{|input|\cdot|hidden|}$ is a matrix of weights and $\boldsymbol{b}^{(h)} \in \mathbb{R}^{|hidden|}$ a vector of biases of the first layer, and $\boldsymbol{W}^{(y)} \in \mathbb{R}^{|hidden|\cdot|output|}$, $\boldsymbol{b}^{(y)} \in \mathbb{R}^{|output|}$ are parameters of the second layer.

Output Layers

- none (linear regression if there are no hidden layers);
- σ (sigmoid; logistic regression if there are no hidden layers)

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



is used to model a Bernoulli distribution, i.e., the probability φ of one of the outcomes;
  - the input of the sigmoid is called a **logit**, and it has a value of $\log \frac{\varphi}{1-\varphi}$

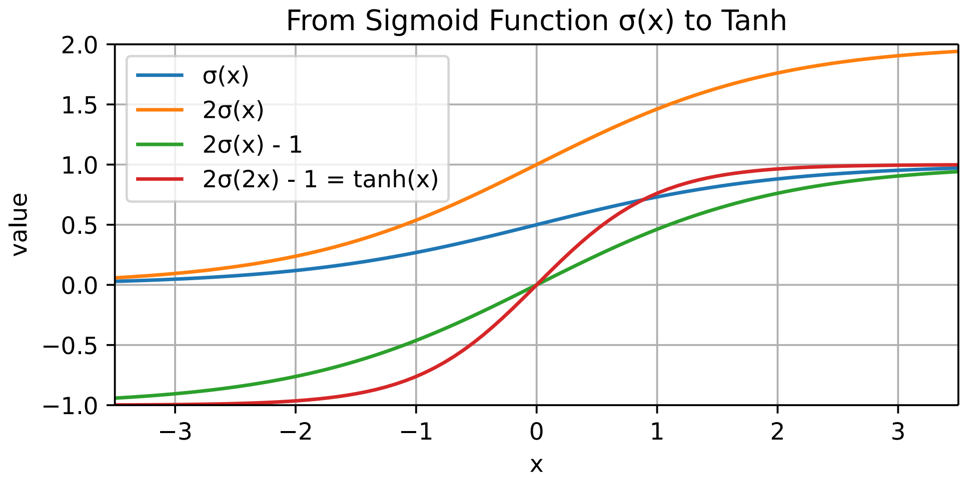
- $\operatorname{softmax}$ (maximum entropy model if there are no hidden layers)

$$\operatorname{softmax}(\boldsymbol{x}) \propto e^{\boldsymbol{x}}, \qquad \operatorname{softmax}(\boldsymbol{x})_i \stackrel{\text{def}}{=} \frac{e^{x_i}}{\sum_j e^{x_j}}$$

is used to model probability distribution $\boldsymbol{p}$; its input is called a **logit**, $\log(\boldsymbol{p}) + c$.

Hidden Layers

- none: does not help, composition of linear/affine mapping is a linear/affine mapping
- σ: does not work great – nonsymmetrical, repeated application converges to the fixed point $x = \sigma(x) \approx 0.659$, and $\frac{d\sigma}{dx}(0) = 1/4$
- $\tanh$
  - result of making σ symmetrical and making the derivative in zero 1
  - $\tanh(x) = 2\sigma(2x) - 1$



- ReLU: $\max(0, x)$

Universal Approximation Theorem '89

Let $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ be a nonconstant, bounded and nondecreasing continuous function.
(Later a proof was given also for $\varphi = \operatorname{ReLU}$ and even for any nonpolynomial function.)

For any $\varepsilon > 0$ and any continuous function $f : [0,1]^D \rightarrow \mathbb{R}$, there exists $H \in \mathbb{N}$, $\boldsymbol{v} \in \mathbb{R}^H$, $\boldsymbol{b} \in \mathbb{R}^H$ and $\boldsymbol{W} \in \mathbb{R}^{D \times H}$, such that if we denote

$$F(\boldsymbol{x}) = \boldsymbol{v}^T \varphi(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b}) = \sum_{i=1}^H v_i \varphi(\boldsymbol{x}^T \boldsymbol{W}_{*,i} + b_i),$$

where φ is applied element-wise, then for all $\boldsymbol{x} \in [0,1]^D$:

$$|F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon.$$

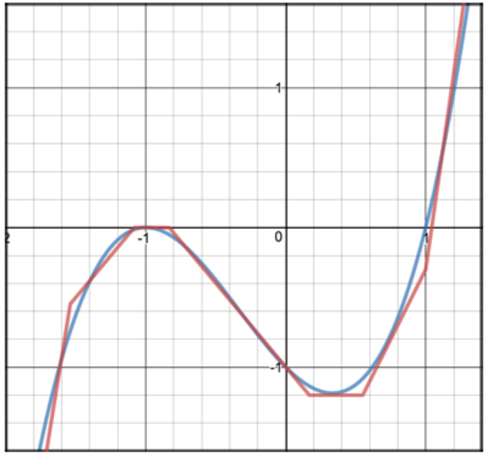
One Possible Interpretation

It is always possible to create features using just a single linear layer followed by a nonlinearity, such that the resulting dataset is always linearly separable.

Universal Approximation Theorem for ReLUs

Sketch of the proof:

- If a function is continuous on a closed interval, it can be approximated by a sequence of lines to arbitrary precision.



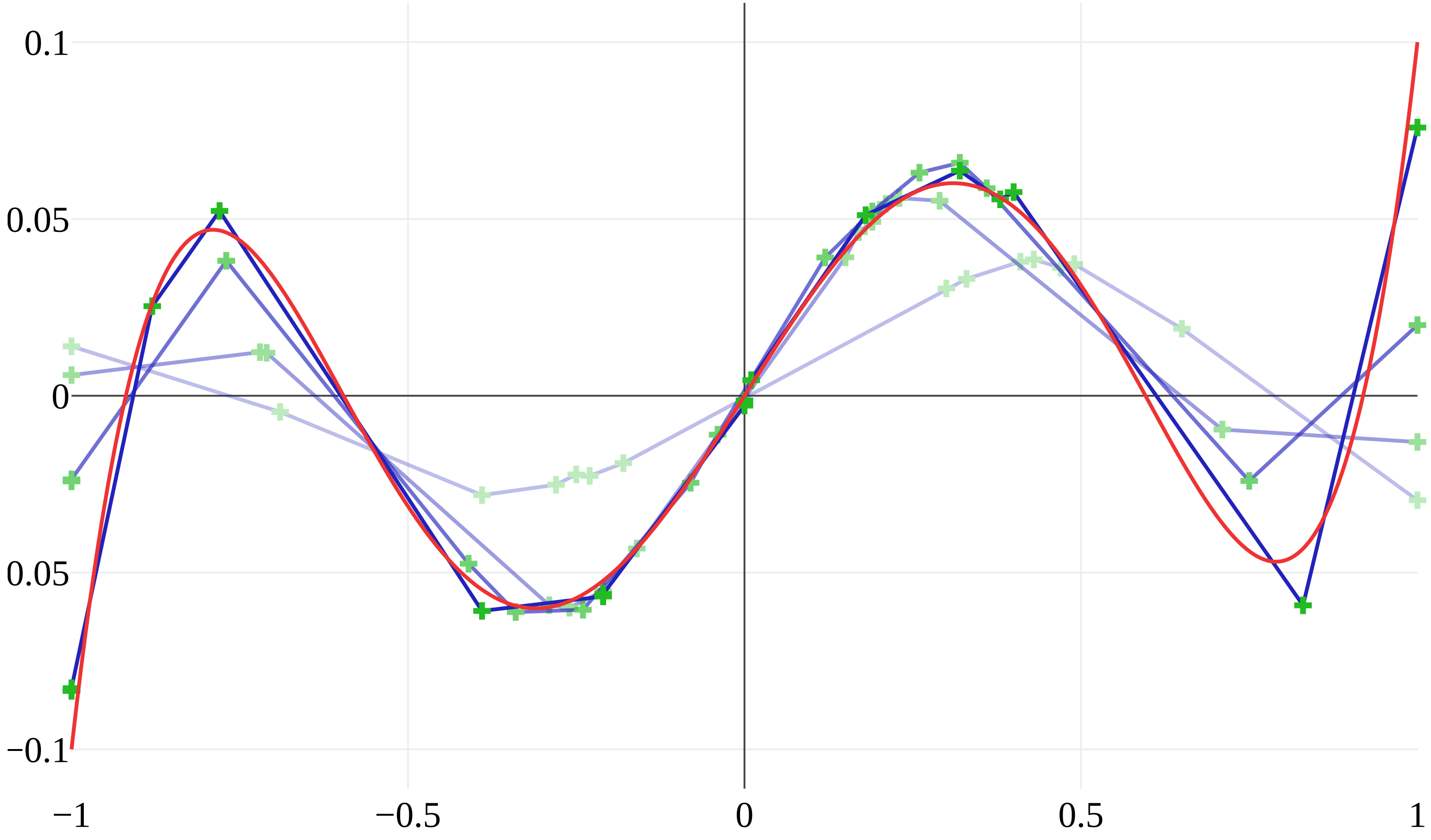
$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

$$Z(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

*https://miro.medium.com/max/844/1*lihbPNQgl7oKjpCsmzPDKw.png*

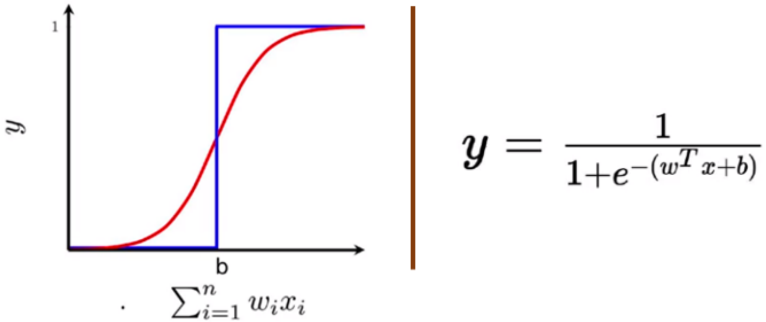
- However, we can create a sequence of k linear segments as a sum of k ReLU units – on every endpoint a new ReLU starts (i.e., the input ReLU value is zero at the endpoint), with a tangent which is the difference between the target tangent and the tangent of the approximation until this point.

Evolving ReLU Approximation



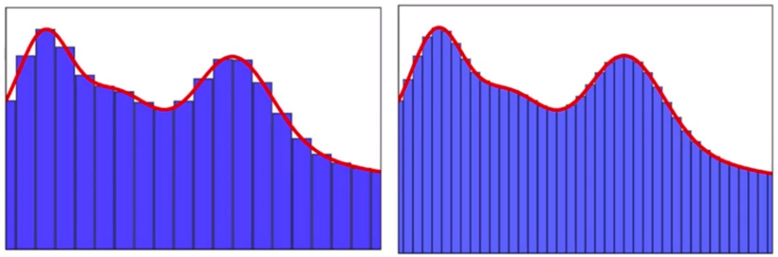
Sketch of the proof for a squashing function $\varphi(x)$ (i.e., nonconstant, bounded and nondecreasing continuous function like sigmoid):

- We can prove φ can be arbitrarily close to a hard threshold by compressing it horizontally.



*https://hackernoon.com/hn-images/1*N7dfPwbiXC-Kk4TCbfRerA.png*

- Then we approximate the original function using a series of straight line segments



*https://hackernoon.com/hn-images/1*hVuJgUTLUFWTMmJhl_fomg.png*

How Good is Current Deep Learning

- DL has seen amazing progress in the last ten years.
- Is it enough to get a bigger brain (datasets, models, computer power)?
- Problems compared to Human learning:
  - Sample efficiency
  - Human-provided labels
  - Robustness to data distribution change
  - Stupid errors

https://intl.startrek.com/sites/default/files/styles/content_full/public/images/2019-07/c8ffe9a587b126f152ed3d89a146b445.jpg

How Good is Current Deep Learning

Ilya Sutskever
@ilyasut

it may be that today's large neural networks are slightly conscious

Přeložit Tweet

12:27 dop. · 10. 2. 2022 · Twitter Web App

https://twitter.com/ilyasut/status/1491554478243258368

Yann LeCun
@ylecun

Odpověď uživateli @ilyasut

Nope.
Not even for true for small values of "slightly conscious" and large values of "large neural nets".
I think you would need a particular kind of macro-architecture that none of the current networks possess.

Přeložit Tweet

10:02 odp. · 12. 2. 2022 · Twitter for Android

https://twitter.com/ylecun/status/1492604977260412928

Murray Shanahan
@mpshanahan

Odpověď uživateli @ilyasut

... in the same sense that it may be that a large field of wheat is slightly pasta

Přeložit Tweet

11:08 dop. · 10. 2. 2022 · Twitter Web App

https://twitter.com/mpshanahan/status/1491715721289678848

Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

Paper "Sparks of Artificial General Intelligence: Early experiments with GPT-4", https://arxiv.org/abs/2303.12712

How Good is Current Deep Learning

r/singularity · 7 mo. ago
[deleted]

Is GPT-4, overall, more intelligent than a dog?

AI

We know they exist in different realms of reality, and their way of processing is somewhat different, also they each have at least one modality that the other one has not. Plus one is frozen in time while the other one is continuously updating itself. But just for the sake of it, if you had to answer with one word:

Is GPT-4 more intelligent than a dog?

Closed · 386 total votes

263	Yes
123	No

Voting closed 7 months ago

https://www.reddit.com/r/singularity/comments/14xyn6n/is_gpt4_overall_more_intelligent_than_a_dog/

Gah_Duma · 7mo ago

Bro it's smarter than at least half of all humans

https://www.reddit.com/r/singularity/comments/14xyn6n/is_gpt4_overall_more_intelligent_than_a_dog/

M00nch1ld3 · 7mo ago

Let's look at some aspects of intelligence to see:

Adaptability: The ability to learn and adapt to new situations. Dog wins. AI is pretrained. To adapt requires outside intervention.
Problem-solving: The ability to identify and solve problems. AI wins some, dog wins some.
Reasoning: The ability to use logic and reasoning to reach conclusions. AI wins some, Dog wins some
Creativity: The ability to generate new ideas and solutions. Dog wins
Communication: The ability to understand and use language. Dog wins. AI does not understand anything.
Social skills: The ability to interact effectively with others. AI wins in textual interactions. Dog wins all else.

So, no. GPT-4 is not more intelligent than a dog.

https://www.reddit.com/r/singularity/comments/14xyn6n/is_gpt4_overall_more_intelligent_than_a_dog/

Curse of Dimensionality

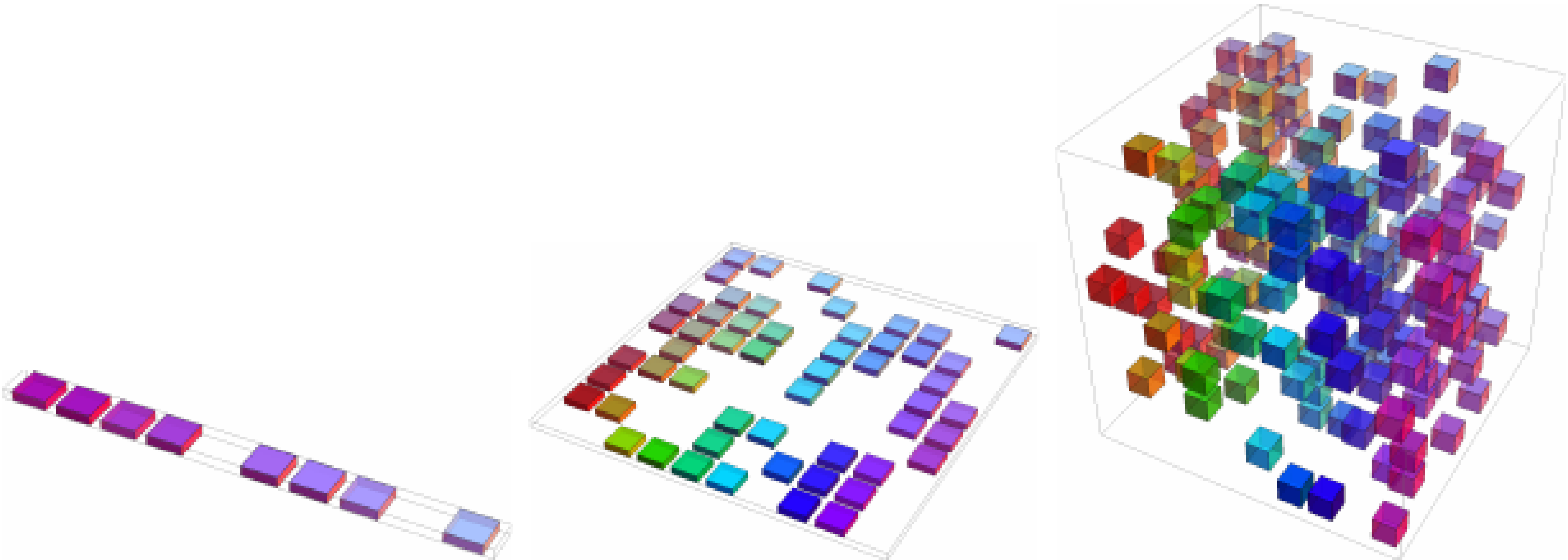


Figure 5.9 of "Deep Learning" book, https://www.deeplearningbook.org

Machine and Representation Learning

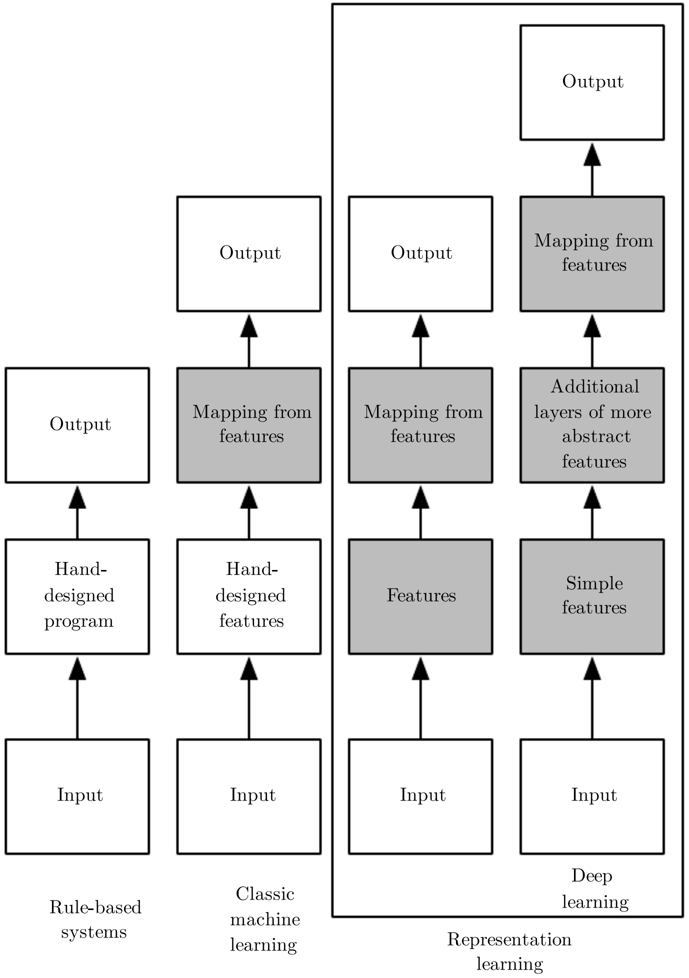


Figure 1.5 of "Deep Learning" book, https://www.deeplearningbook.org