A Practical Introduction to Data Science

Part 7

Deep Learning


Gergely Zsombor Haász


haasz.zsombi@gmail.com

Course Agenda

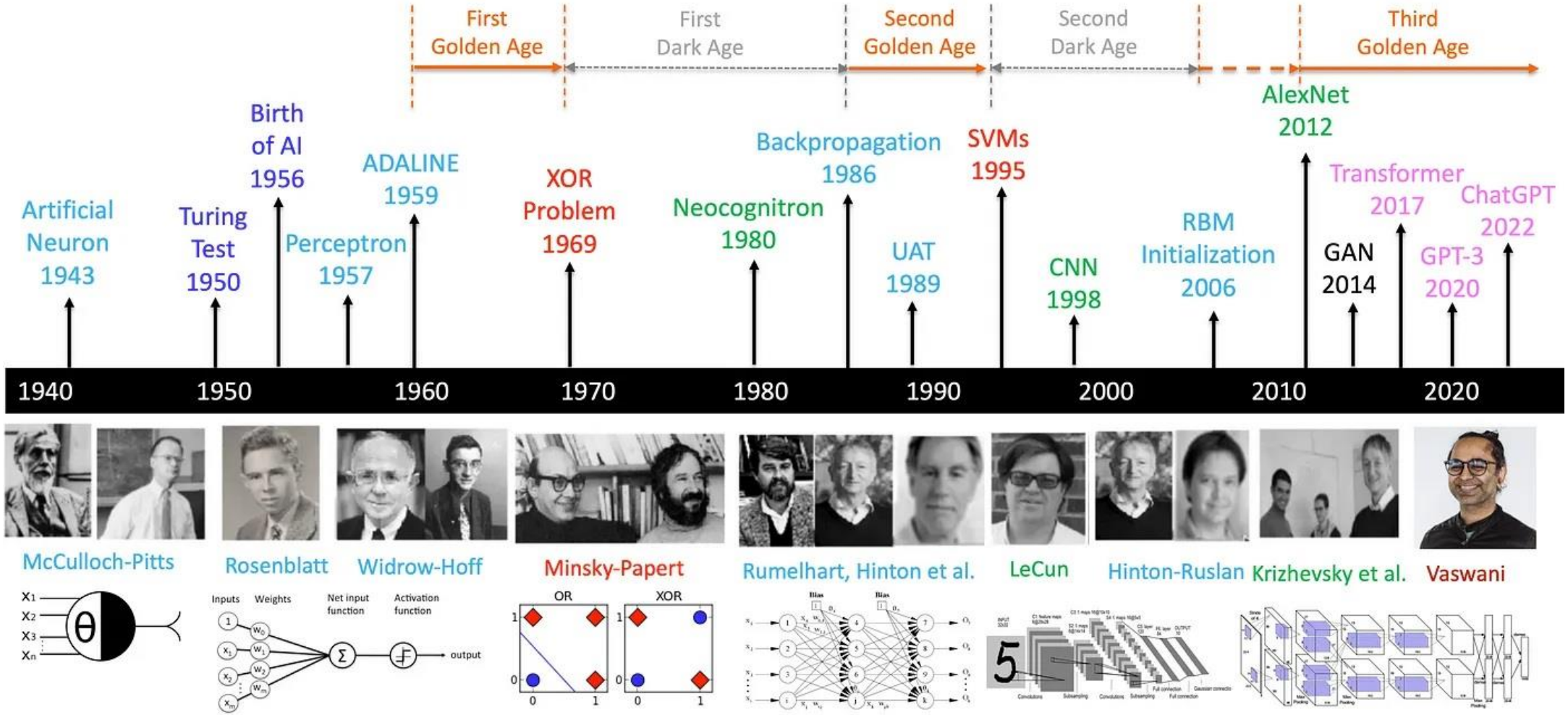
I.	Introduction to Data Science
II.	Business and Data Understanding
III.	Introduction to Supervised Learning
IV.	Advanced Supervised Learning
V.	Unsupervised Learning
VI.	Time Series Analysis
VII.	Deep Learning
VIII.	Machine Learning Operations

Deep Learning

- Introduction
- Architectures
- Training
- Optimization

Introduction

A Brief History of AI with Deep Learning

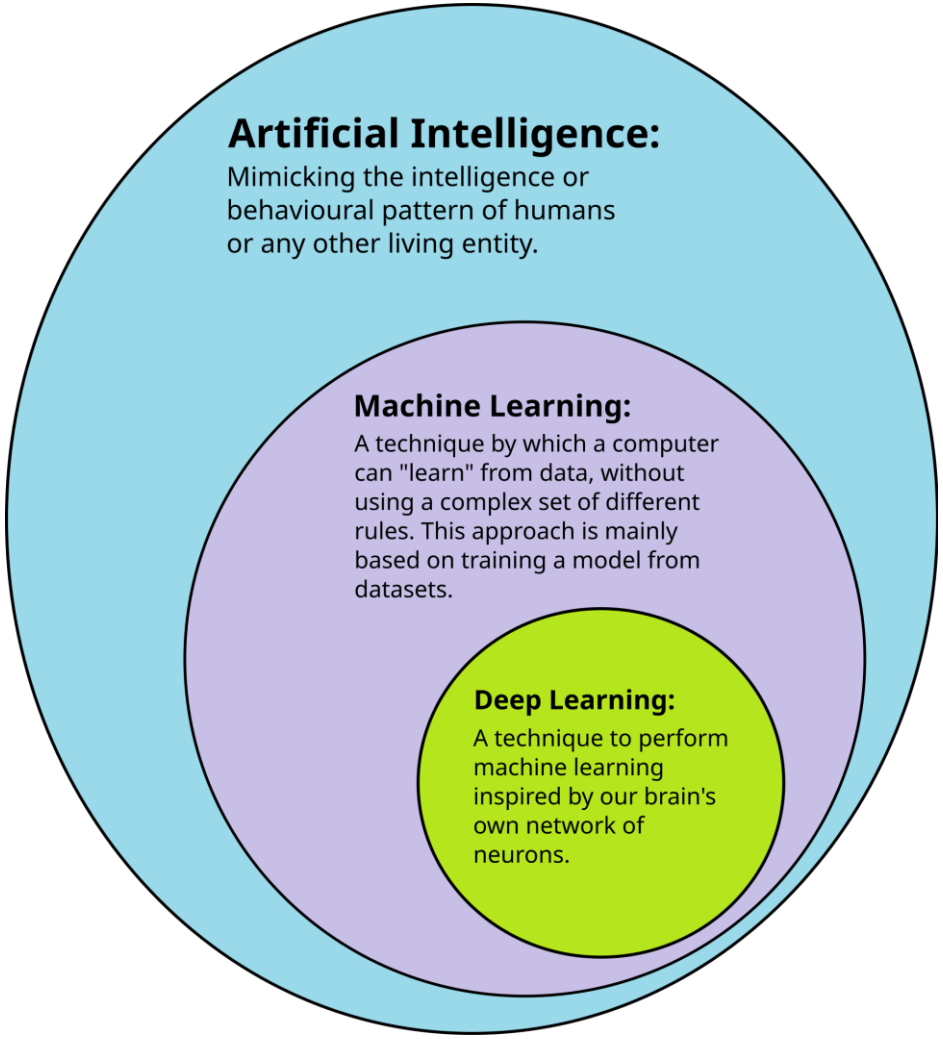


Data, Compute, Algorithms

Source: A Brief History of AI with Deep Learning | by LM Po | Medium

Deep Learning versus Machine Learning

- Built-in feature extraction
- Better architectures for text and vision:
  - Word embeddings for text
  - Convolutions for images
  - Sequence modelling for time series and text
- Performs well on large amounts of data and complex problems
- Computationally expensive, but great parallelism (GPUs)



Source: Deep learning - Wikipedia

What is Deep Learning?

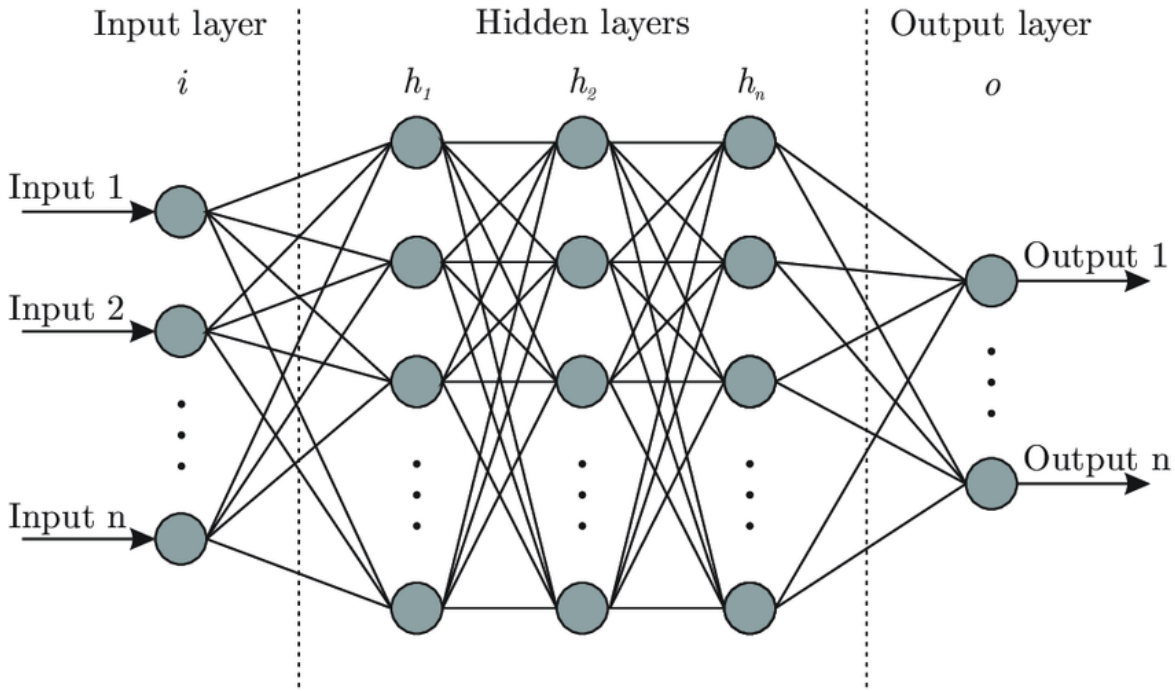
A subset of machine learning inspired by the human brain, utilizing deep neural networks to learn from data. A neural network is basically a sequence of nonlinear mathematical functions.

Model architecture outlines the sequence and connectivity of layers and neurons, along with the functions they perform.

Elements of a Neural Network:

- Layers
- Neurons (Nodes)
- Weights and biases
- Activation functions

Model Training: Gradient Descent and Backpropagation



Source: Designing Your Neural Networks. A Step by Step Walkthrough | by Lavanya Shukla | Towards Data Science

What is Deep Learning?

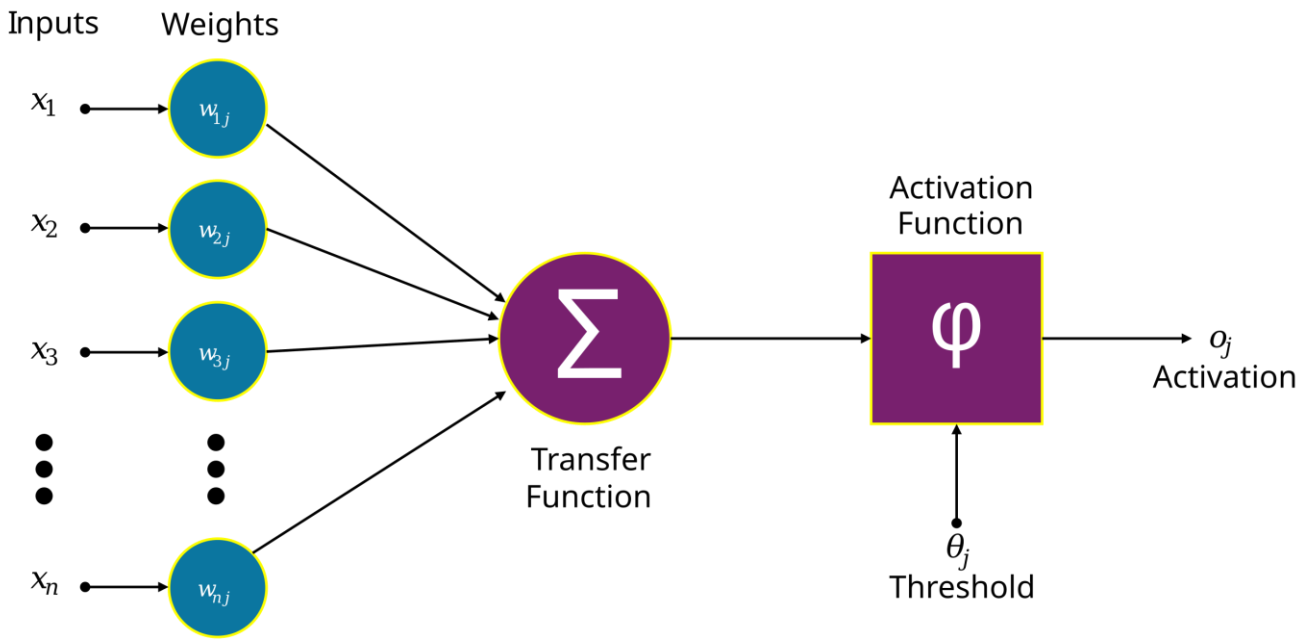
A subset of machine learning inspired by the human brain, utilizing deep neural networks to learn from data. A neural network is basically a sequence of nonlinear mathematical functions.

Model architecture outlines the sequence and connectivity of layers and neurons, along with the functions they perform.

Elements of a Neural Network:

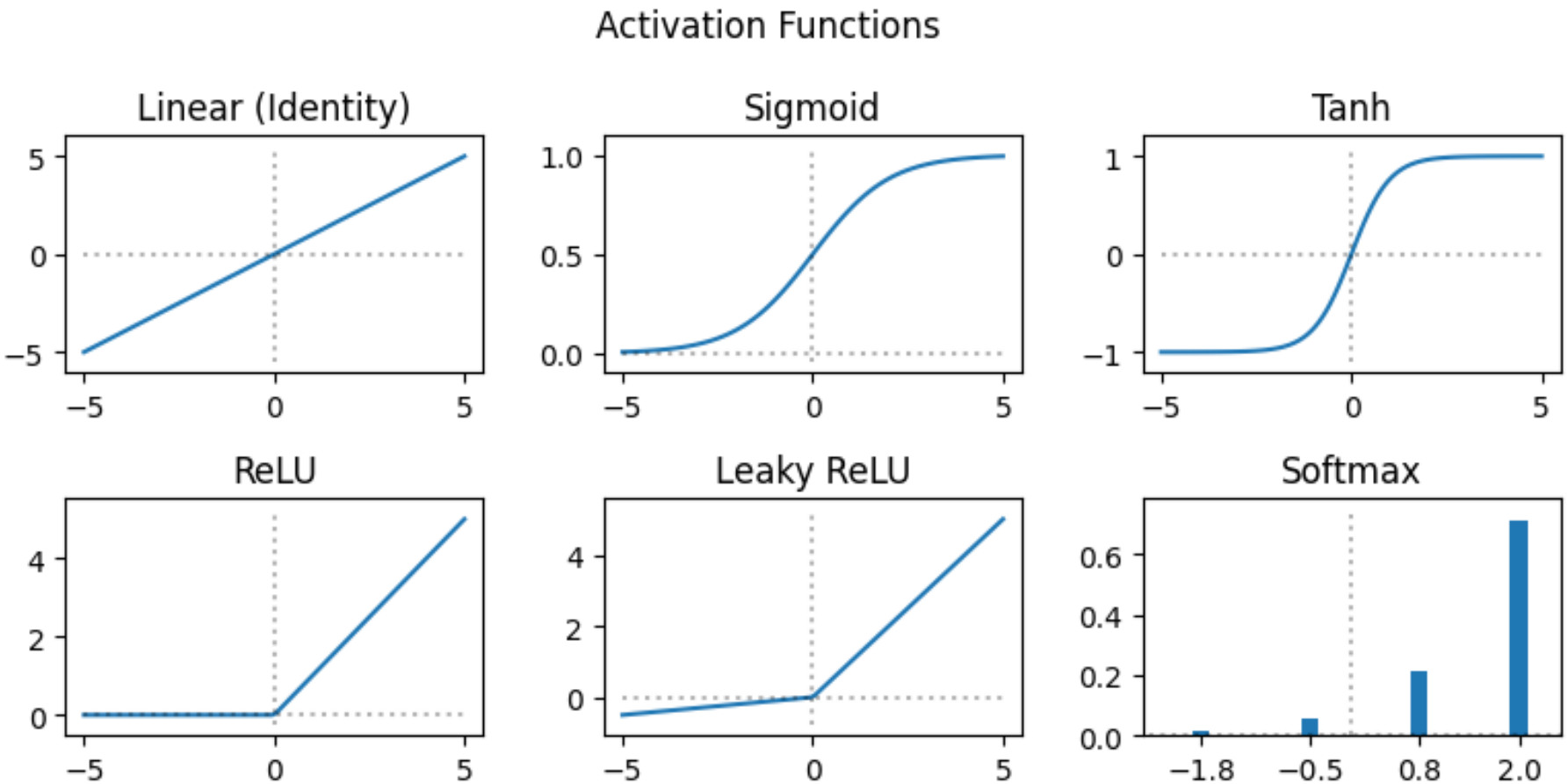
- Layers
- Neurons (Nodes)
- Weights and biases
- Activation functions

Model Training: Gradient Descent and Backpropagation

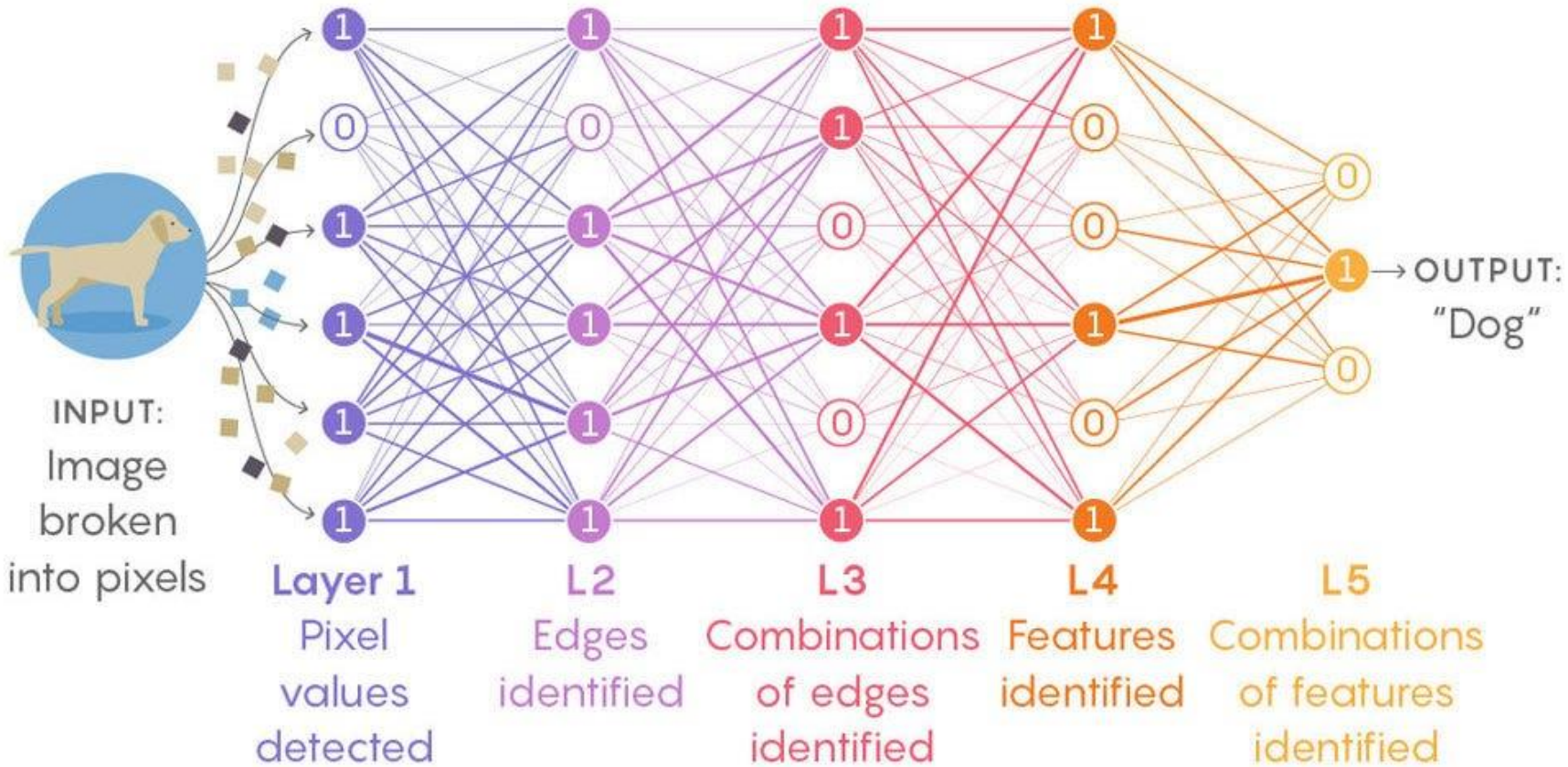


Source: Artificial neuron – Wikipedia

Activation functions



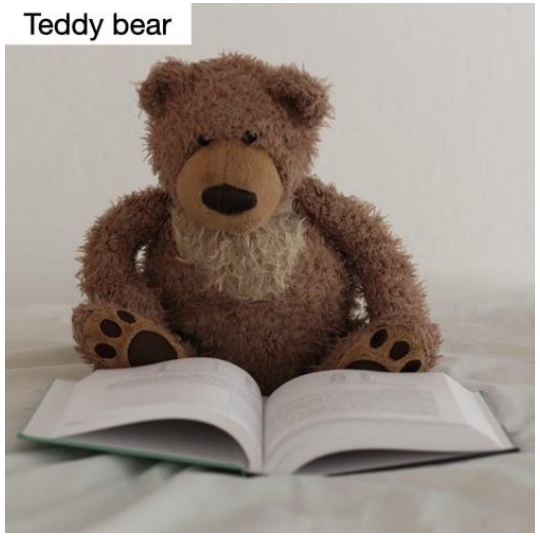
Feature Extraction

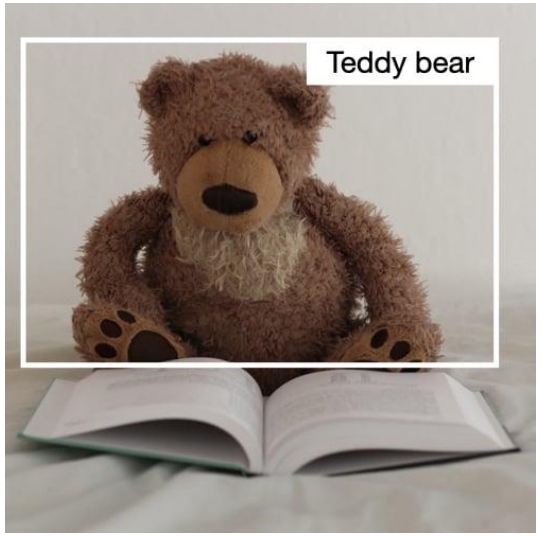


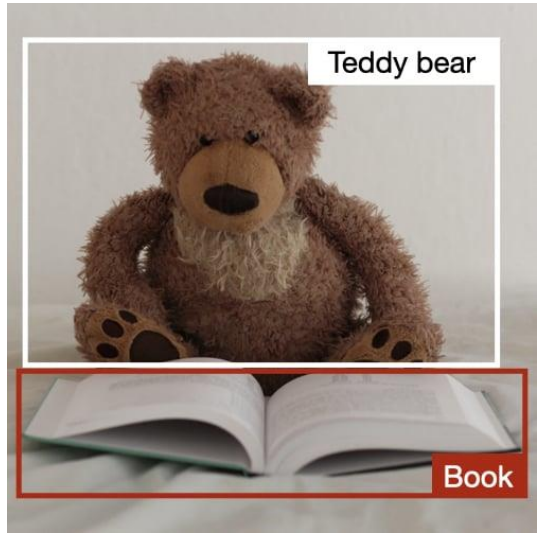
Source: Designing Your Neural Networks. A Step by Step Walkthrough | by Lavanya Shukla | Towards Data Science

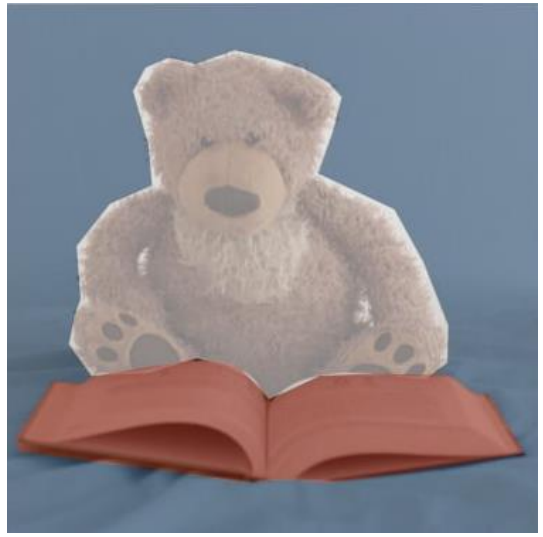
Architectures

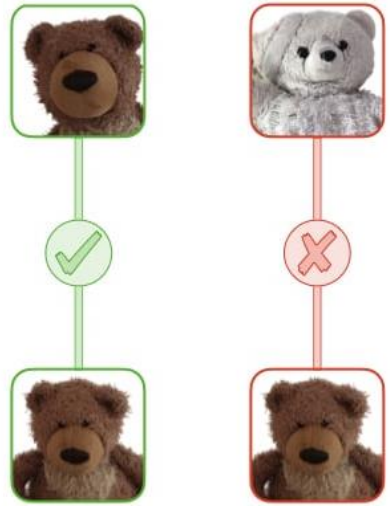
Deep Learning for Vision

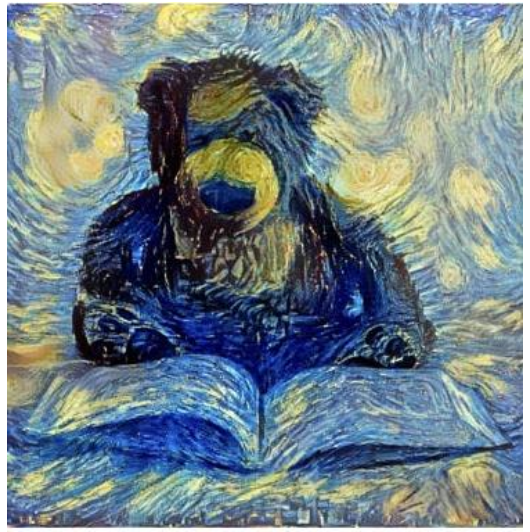












Source: CS 230 - Convolutional Neural Networks Cheatsheet

Deep Learning for Vision

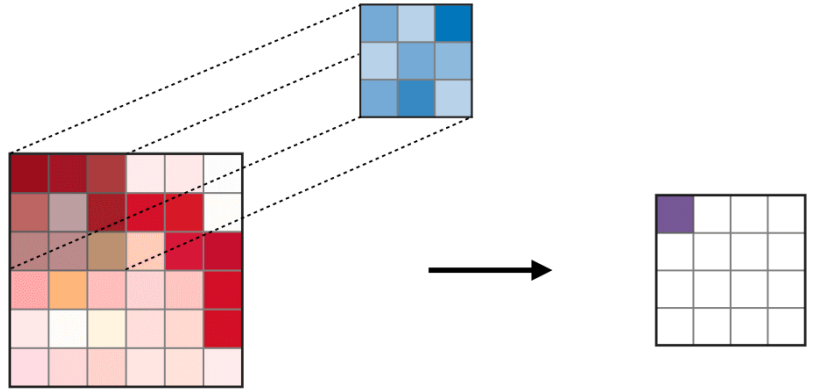
Image	Generative	Video / 3D	Architectures	Pretrained Models

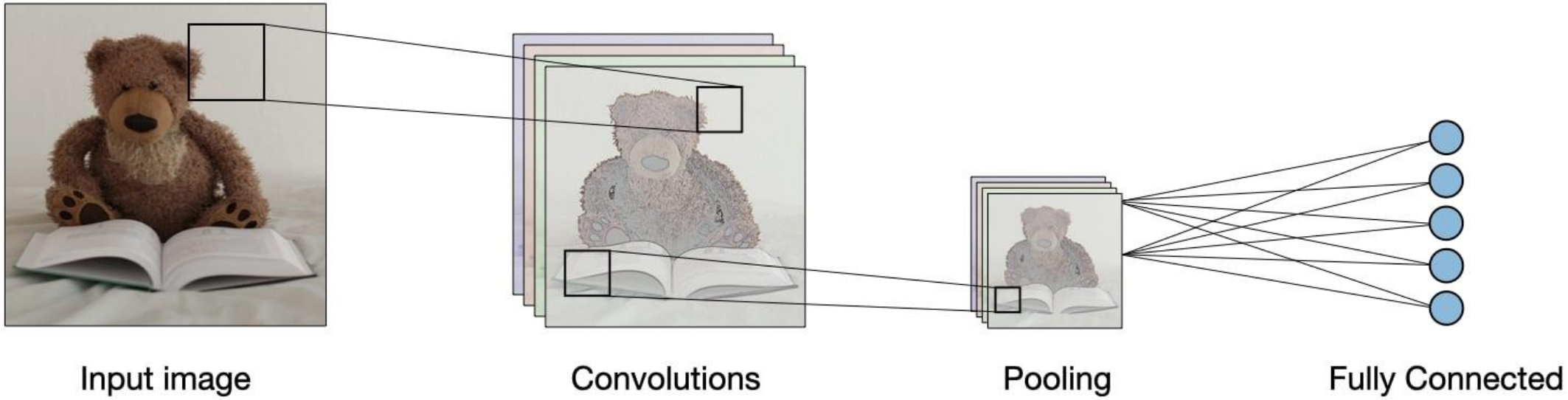
• Image recognition
• Object detection
• Semantic /Instance Segmentation
• Optical Character Recognition
• Facial Recognition
	• Image Enhancement
	• Style Transfer
	• Image Captioning
	• Video Manipulation
	• Image/Video Generation
		• Object Tracking
		• Pose estimation
		• Depth Estimation
		• 3D Reconstruction
		• SLAM
			• Traditional CV
			• Deep Learning:
			  • CNN
			  • Autoencoder
			  • GAN
			  • Diffusion Model
			  • Vision Transformer
			  • Multimodality
				• AlexNet
				• VGG
				• Resnet
				• EfficientNet
				• U-Net
				• SAM
				• R-CNN
				• Yolo
				• SSD
				• ViT
				• DETR

Deep Learning for Vision

Convolutional Neural Networks

- Convolutional layers + Pooling layers
- Less parameters compared to FNN
- Space invariant
- AlexNet won the ImageNet competition in 2012



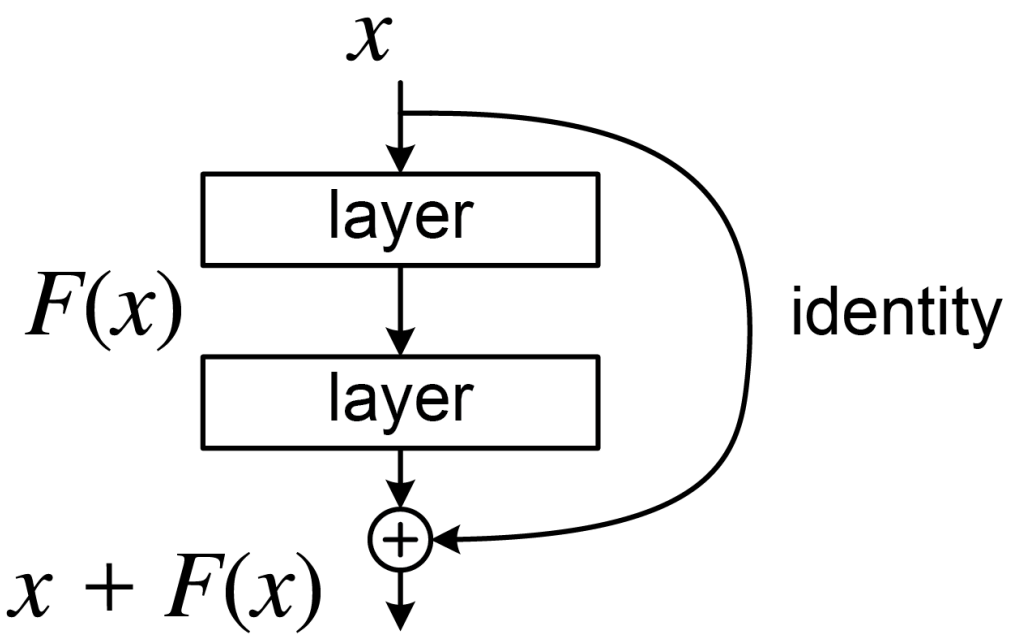


Source: CS 230 - Convolutional Neural Networks Cheatsheet

Deep Learning for Vision

Residual Connections

- Deal with vanishing gradient problem
- Encourages the reuse of features learned in previous layers
- More flexible architecture design
- More stable training
- First used in ResNet (2015, SOTA image recognition at the time)
- Part of transformers



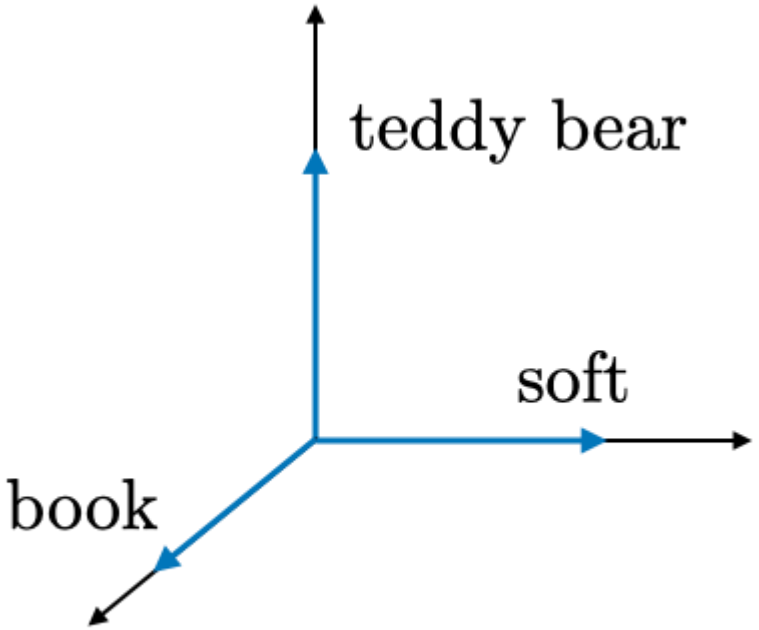
Source: Residual neural network – Wikipedia

Deep Learning for Sequential Data

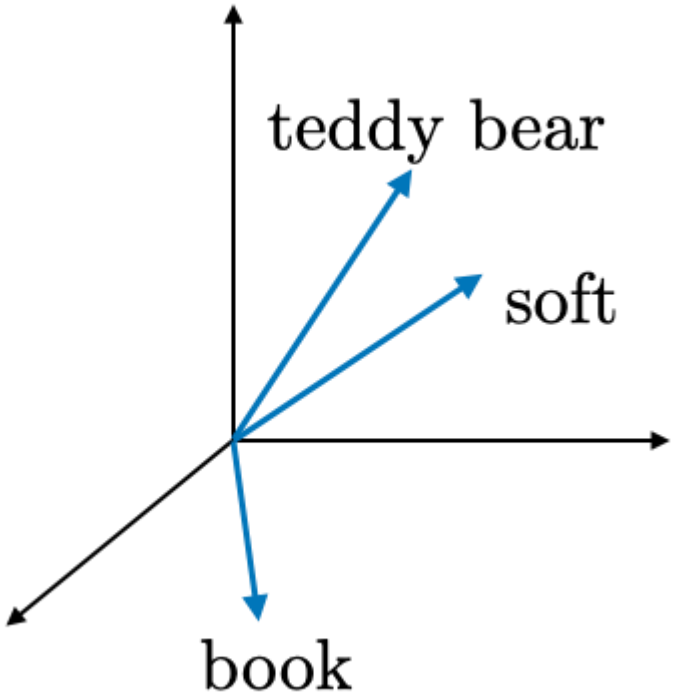
Time Series	Text	Audio	Architectures
• Forecasting • Anomaly Detection • Classification • Imputation	• Text classification • Named Entity Recognition • Sentiment Analysis • Text Summarization • Machine Translation • Text Generation	• Text-to-speech • Speech-to-text • Music Generation	• RNN • LSTM • GRU • Encoder-decoder • Transformers (e.g. BERT, GPT) • LLMs

Deep Learning for Text

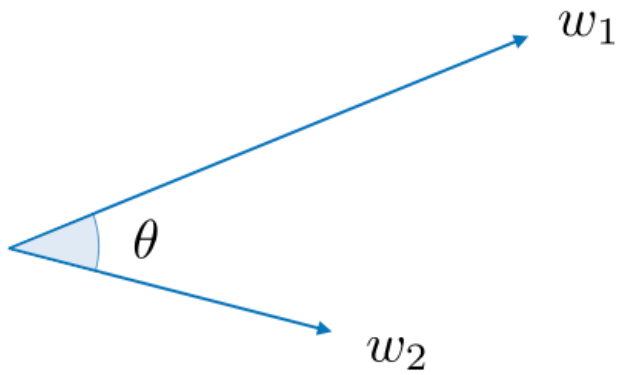
One-hot representation



Vector Embedding



Cosine Similarity



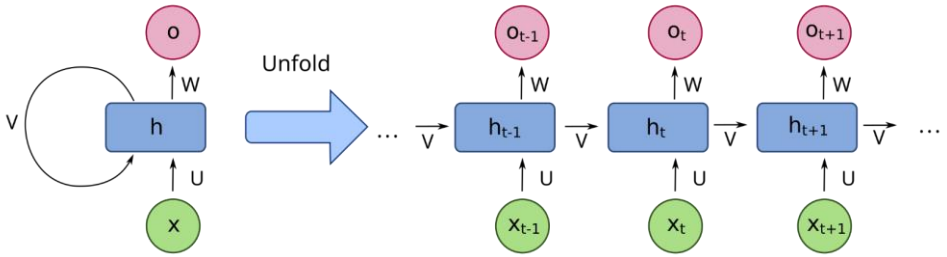
$$similarity = \frac{\omega_1 \omega_2}{\|w_1\|\|w_2\|} = \cos(\theta)$$

Embedding: words (tokens) are represented as multidimensional vectors. Directions and distances describe the relationships between words. Embeddings are optimized to best model these relationships.

Source: CS 230 - Recurrent Neural Networks Cheatsheet

Deep Learning for Sequential Data

Recurrent Neural Networks



Source: Residual neural network – Wikipedia

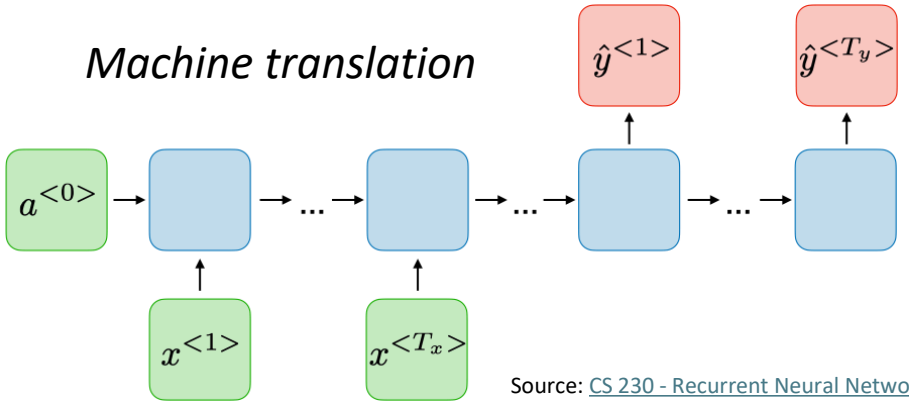
Advantages

- Previous are used as inputs while also having hidden states
- Input sequence can have any length, model
- Model size is not increasing with longer input sequence
- Weights are shared across time

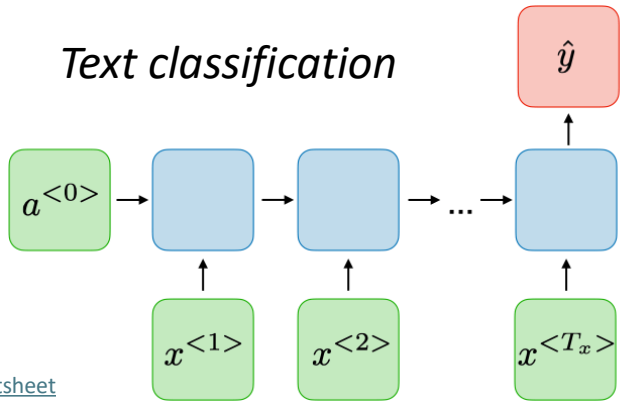
Disadvantages

- Slow computation
- "Forgets" information from a long time ago (vanishing gradients)
- Cannot consider any future input for the current state

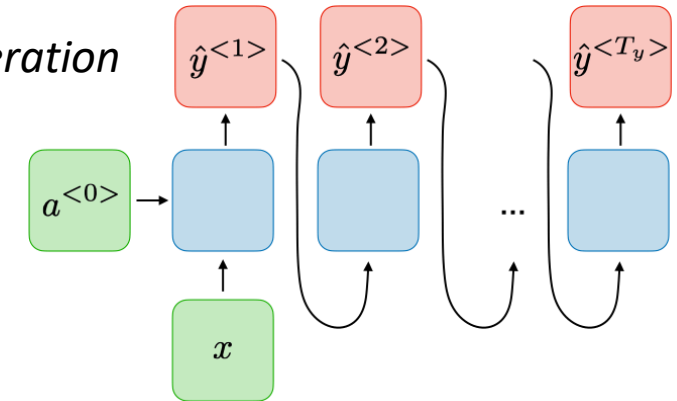
Machine translation



Text classification



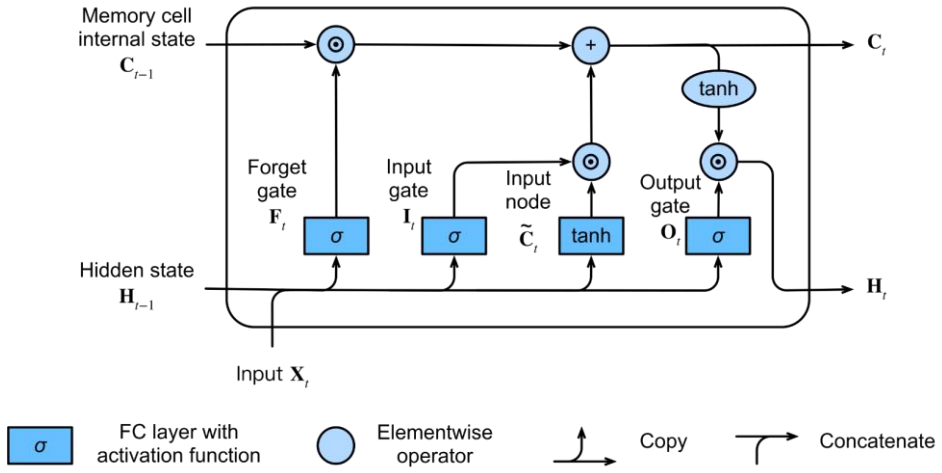
Generation



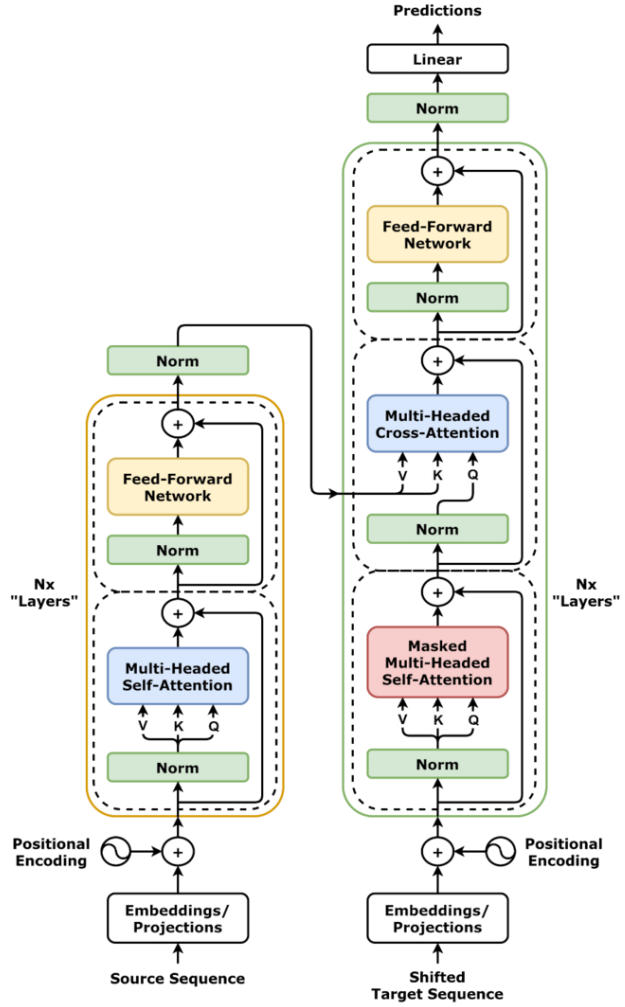
Source: CS 230 - Recurrent Neural Networks Cheatsheet

Deep Learning for Sequential Data

Long Short-Term Memory (LSTM)



Transformers

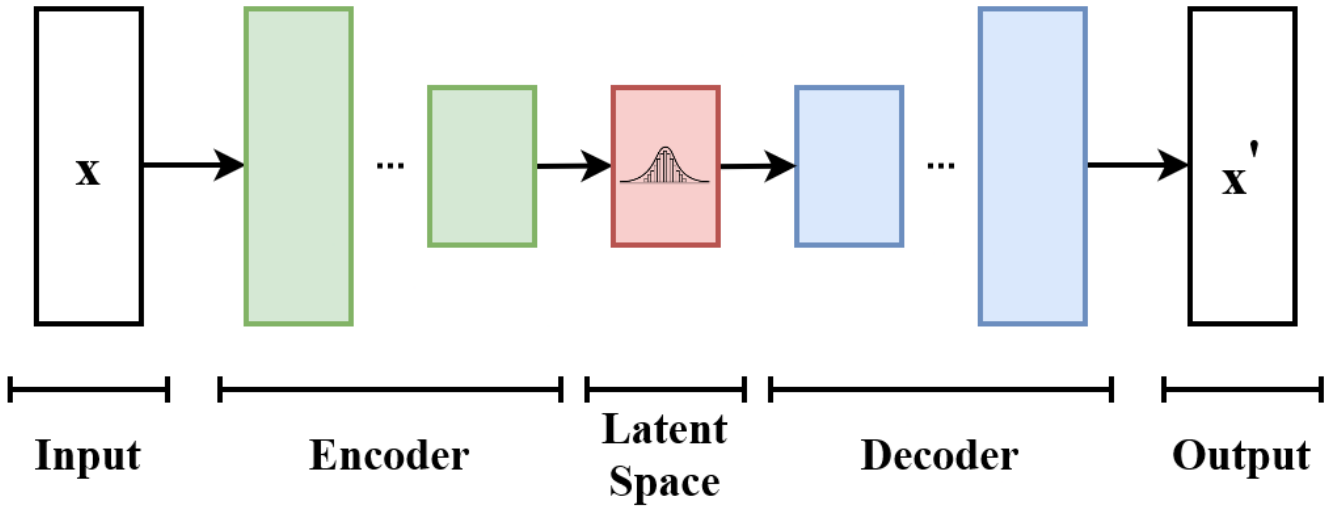


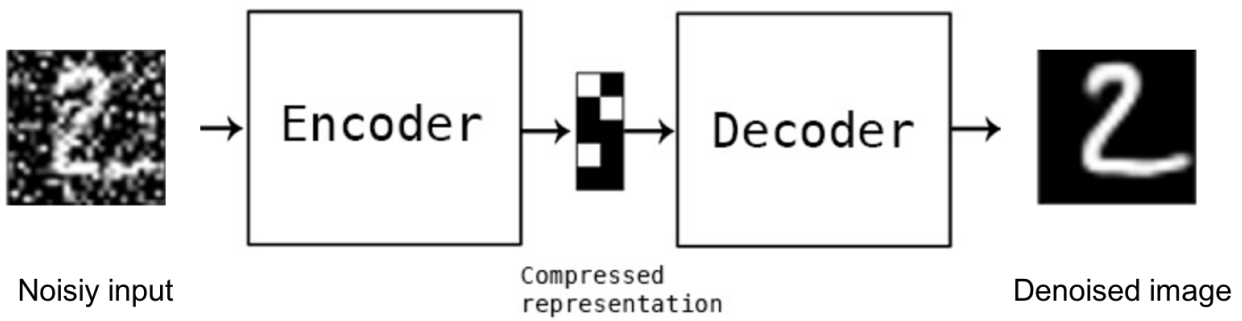
Source: Residual neural network – Wikipedia

Unsupervised Deep Learning

Autoencoder

- Dimension reduction
- Denoising
- Facial recognition
- Anomaly detection
- Data generation

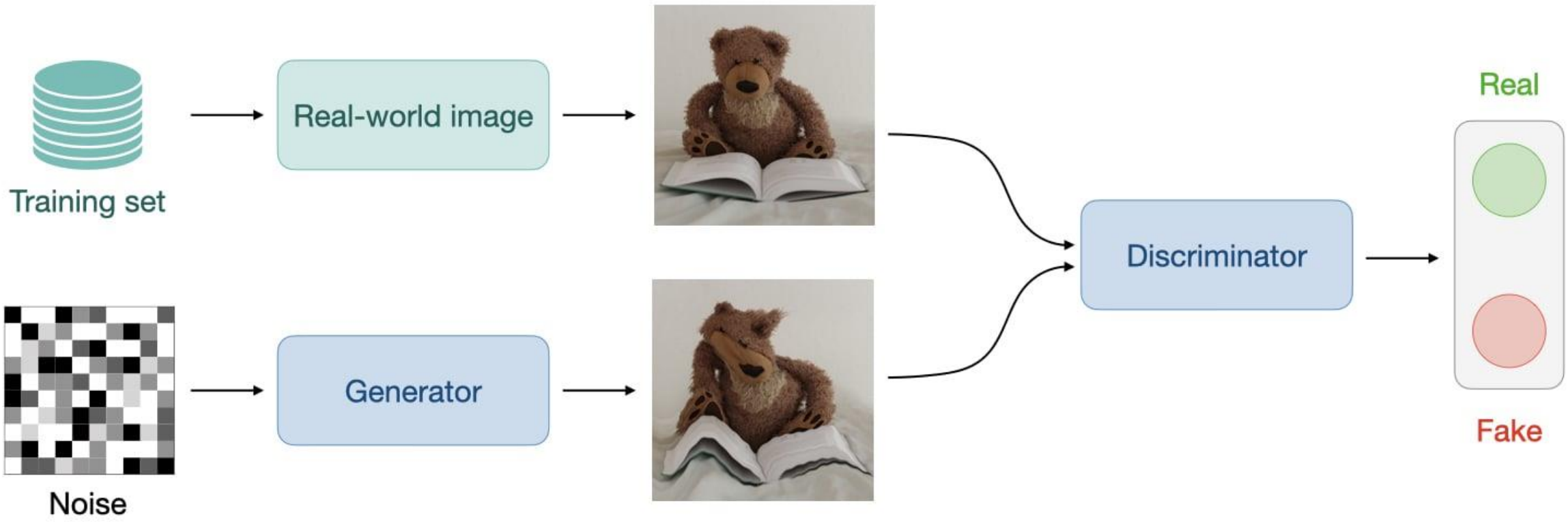




Source: Autoencoder – Wikipedia

Unsupervised Deep Learning

Generative Adversarial Network (GAN, 2014)



Source: CS 230 - Convolutional Neural Networks Cheatsheet

What are LLMs?

- **Deep learning models** designed and optimized specifically for conversations.
- **Transformer:** groundbreaking model architecture based on the **attention** mechanism – the word representation is influenced by the context
- **Pre-training** on large data, **post-training** on high quality data with methods like RLHF
- **System prompt:** instructions to determine how the AI model responds to the **user prompt**
- The trained weights determine the model's behaviour by storing the model's understanding of the world (memory). But this is not human intelligence – more like "autocorrect on steroids"

- Learn more:
  - 3Blue1Brown – Neural networks
  - Andrej Karpathy – Deep Dive into LLMs like ChatGPT
  - Stanford CS229 I Machine Learning I Building Large Language Models (LLMs)

Training

Training a Neural Network

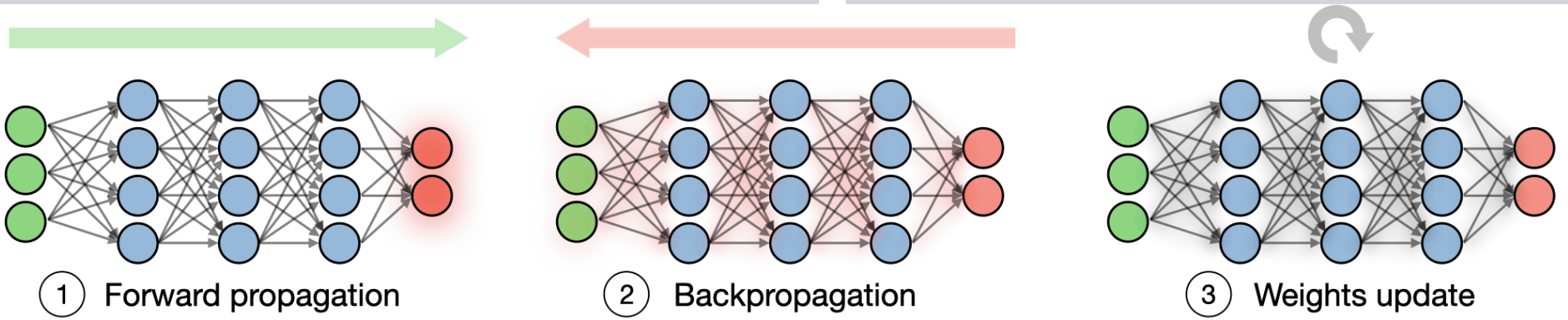
Model Initialization

1. Optimizer
   - Update method
   - Learning rate
2. Loss function
3. Evaluation metrics
4. Number of epochs and batch size
5. Validation split
6. Callbacks
7. Weights Initialization

Training Loop – for each epoch (for each batch):

1. Forward pass
2. Calculate the loss
3. Calculate the gradients (Backpropagation)
4. Update the weights (Gradient Descent)
5. Evaluate on the validation set

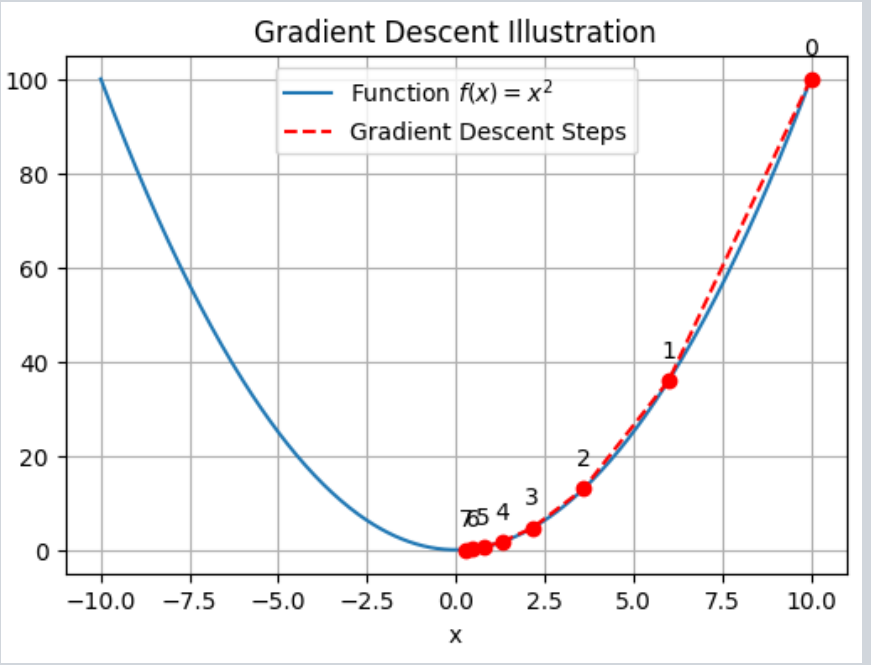
Repeat for a preset number of epochs or until convergence



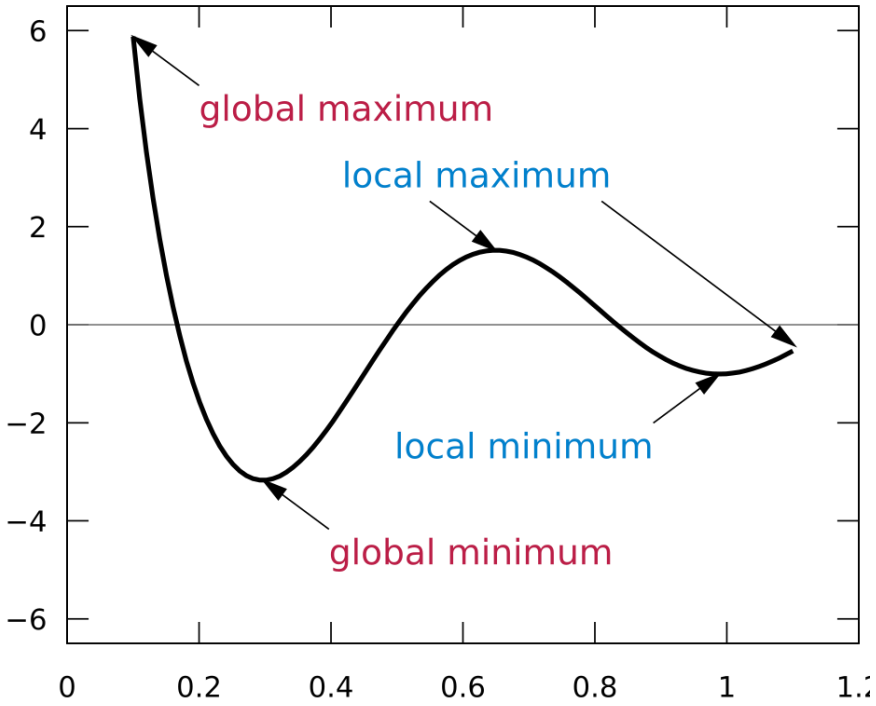
Source: CS 230 - Deep Learning Tips and Tricks Cheatsheet

Gradient Descent

- The **gradient vector** is composed of the partial derivatives of the function with respect to each parameter
- It represents the direction and rate of the steepest increase of the function.
- $\nabla f(x_1, x_2, x_3) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}\right)$
- **Gradient Descent** is an optimization algorithm used to minimize the loss function in machine learning
  1. Start with random weights (weight initialization)
  2. Compute the gradient of the cost function with respect to each parameter
  3. Adjust the parameters in the opposite direction of the gradient
     - $\theta_t = \theta_{t-1} - \alpha * \nabla Loss(\theta_{t-1})$
     - where α is the *learning rate*
  4. Repeat the process until the cost function converges to a minimum



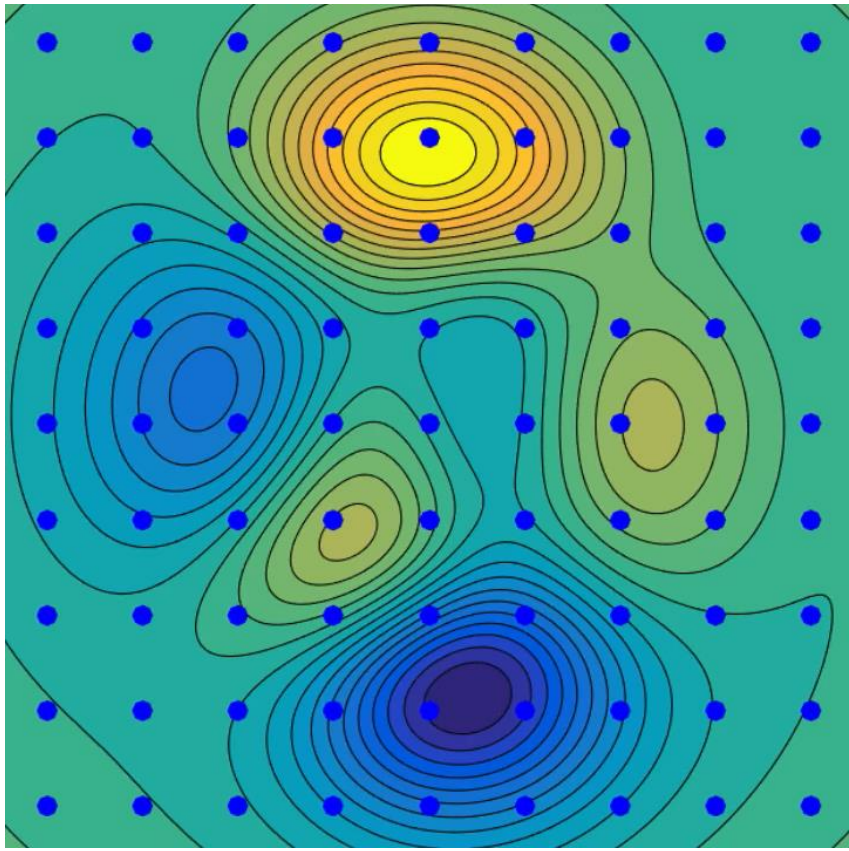
Gradient Descent



Source: Backpropagation – Wikipedia

Limitations:

- Sensitive to weight initialization
- Sensitive to the learning rate
- Might get stuck in local minima
- Might not converge
- Computational cost can be high



Source: Gradient descent – Wikipedia

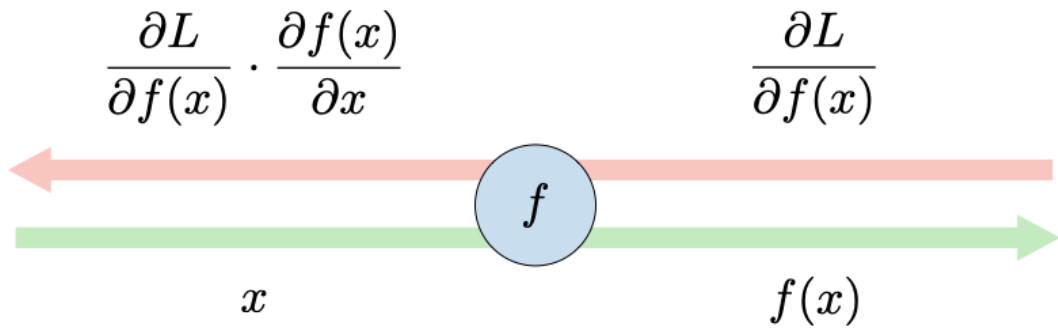
Backpropagation

With the **chain rule**, we can determine the derivative of the loss function with respect to each parameter:

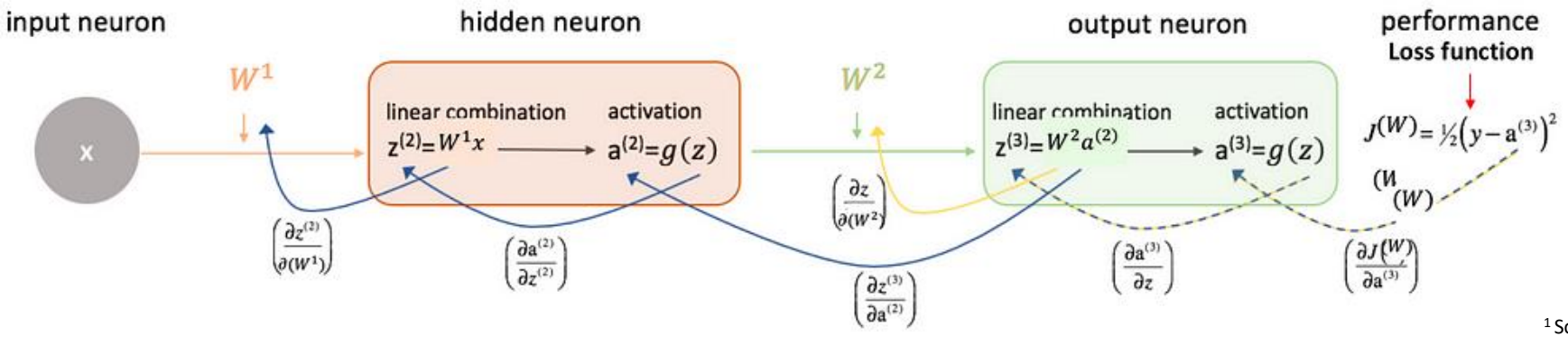
$$h(x) = f\big(g(x)\big)$$
$$h'(x) = f'\big(g(x)\big) * g'(x)$$

"If a car travels twice as fast as a bicycle and the bicycle is four times as fast as a walking man, then the car travels 2 × 4 = 8 times as fast as the man."[1]

This process is called **backpropagation**. Next, we can update the weights: $\theta_t = \theta_{t-1} - \alpha * \nabla Loss(\theta_{t-1})$



Source: CS 230 - Deep Learning Tips and Tricks Cheatsheet

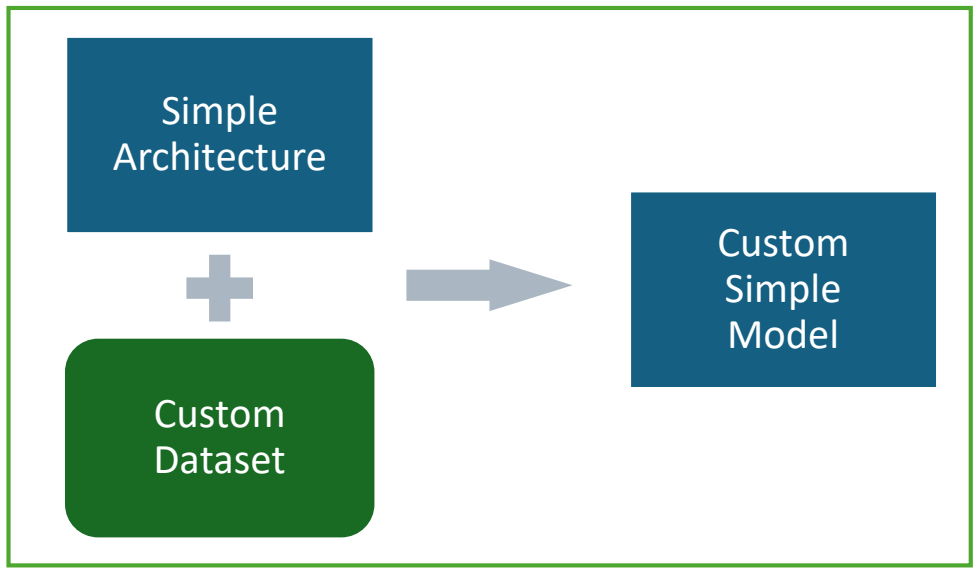


Source: How to Initialize Weights in Neural Networks?

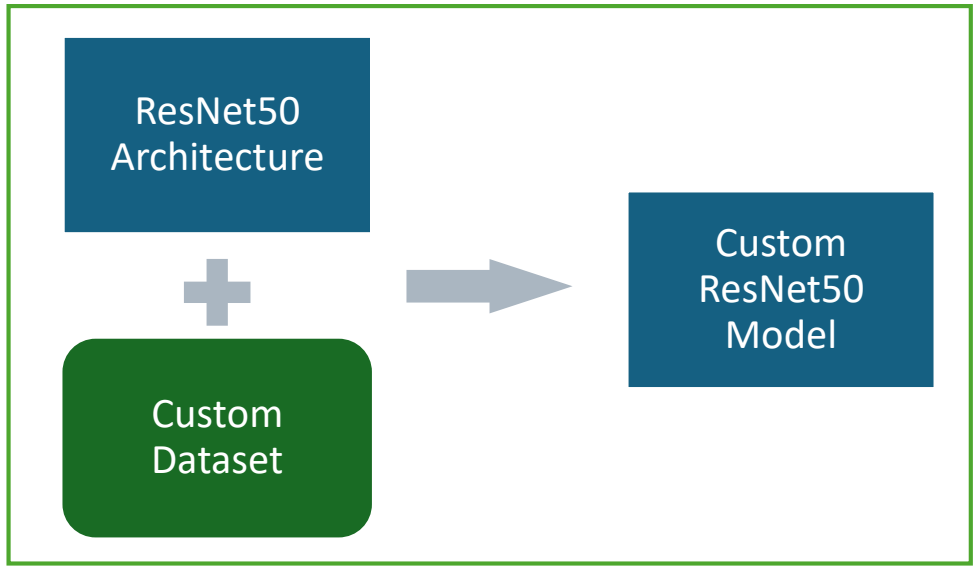
[1] Source: George F. Simmons, Calculus with Analytic Geometry (1985), p. 93

Source: Designing Your Neural Networks. A Step by Step Walkthrough | by Lavanya Shukla | Towards Data Science

Transfer Learning

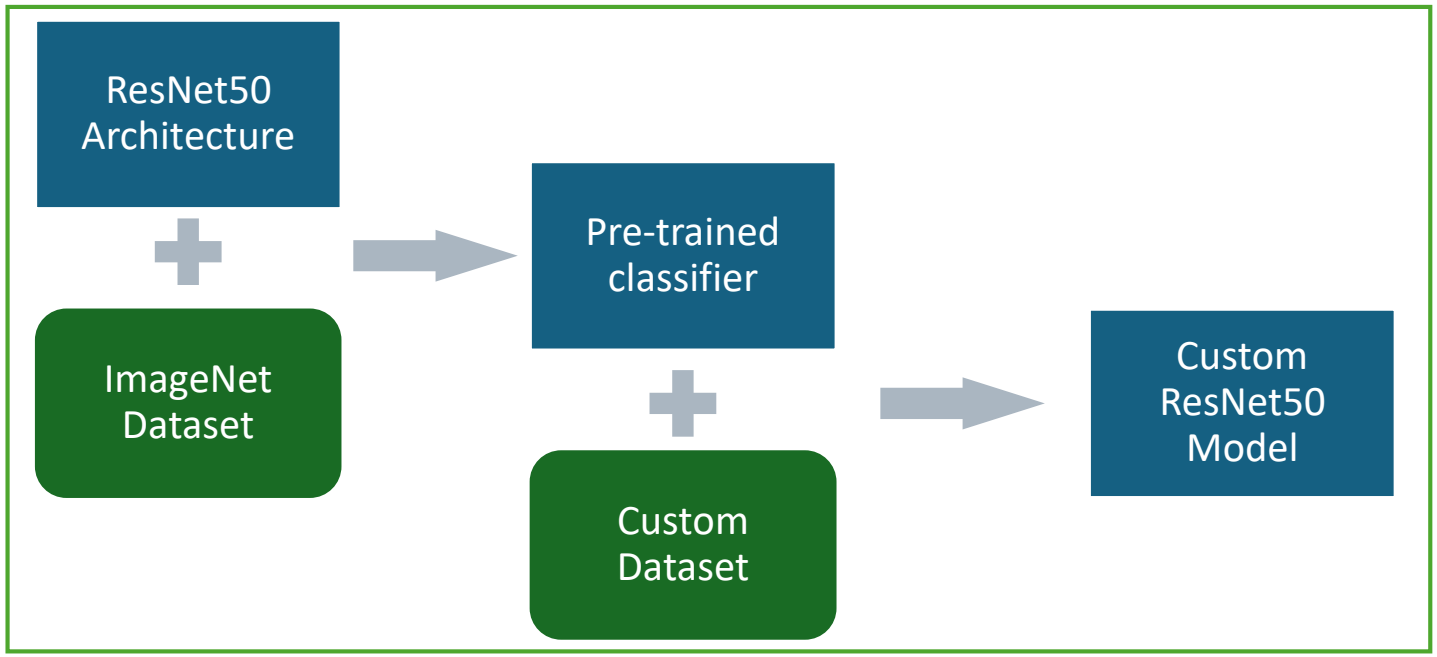


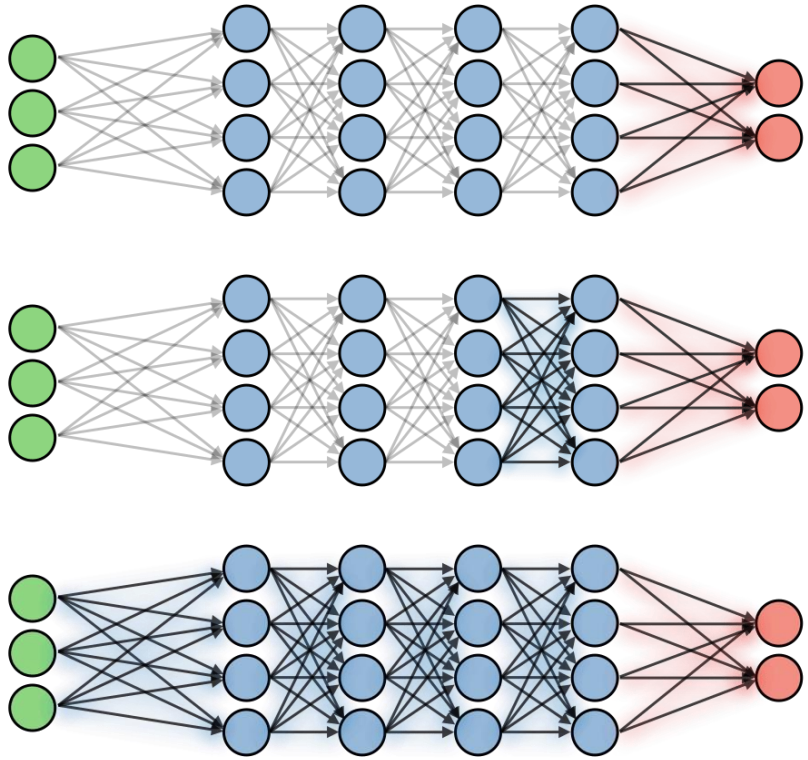
- × Insufficient model complexity
- × Not enough data



- × Computationally expensive
- × Not enough data

Transfer Learning





- ✓ Sufficient model complexity
- ✓ Pre-trained on large amounts of data
- ✓ Fine-tuned on problem-specific data

Source: CS 230 - Deep Learning Tips and Tricks Cheatsheet

Overfitting

Overfitting in Deep Learning

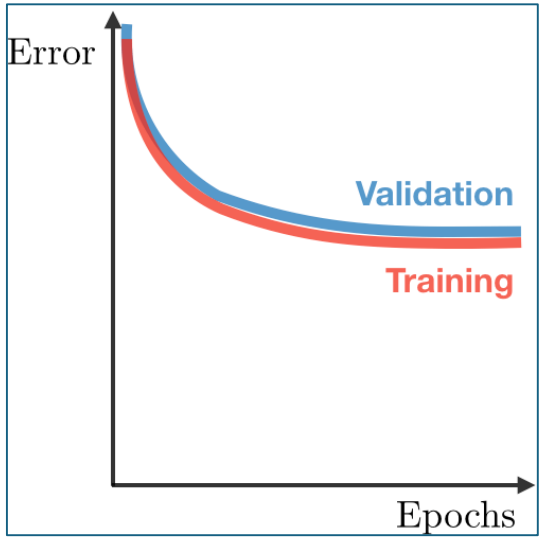
Universal approximator (even works with shuffled labels)

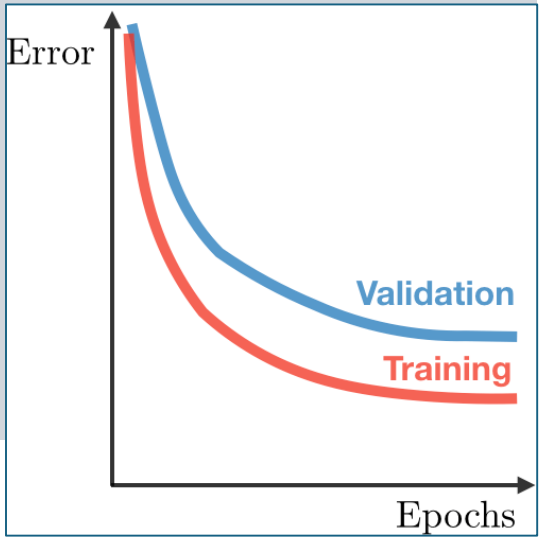
Causes of Underfitting:

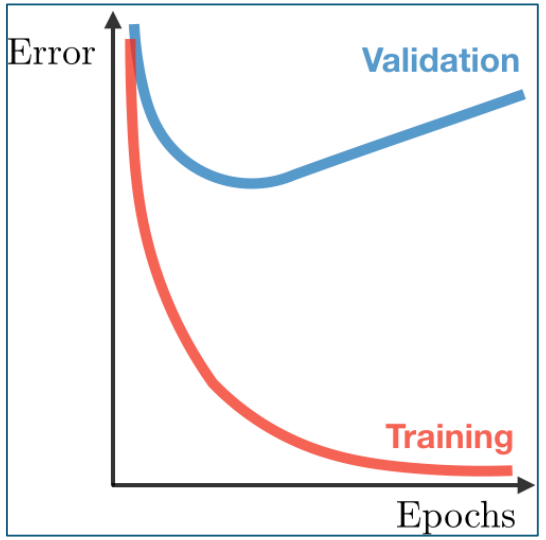
- Wrong architecture type
- Small model
- Training stops too early (epochs, batch size, learning rate)
- Stuck in local minima (bad optimization)

Causes of Overfitting:

- Large model
- Small data
- Irrelevant features
- Long training







Source: CS 229 - Machine Learning Tips and Tricks Cheatsheet

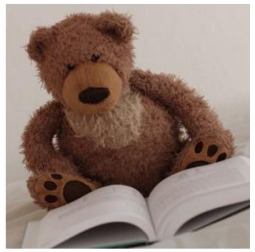
Regularization and Optimization

1. Get more data
2. Augment your data
3. Use a pretrained model
4. Change model size as needed
5. Weight decay (L1/L2 regularization)
6. Early Stopping
7. Add Dropout
8. Try different batch sizes and learning rates
9. Batch Normalization
10. Try different optimizers

Steps:

1. Build a simple model
2. Overfit (see if you can)
3. Reduce complexity

Data Augmentation

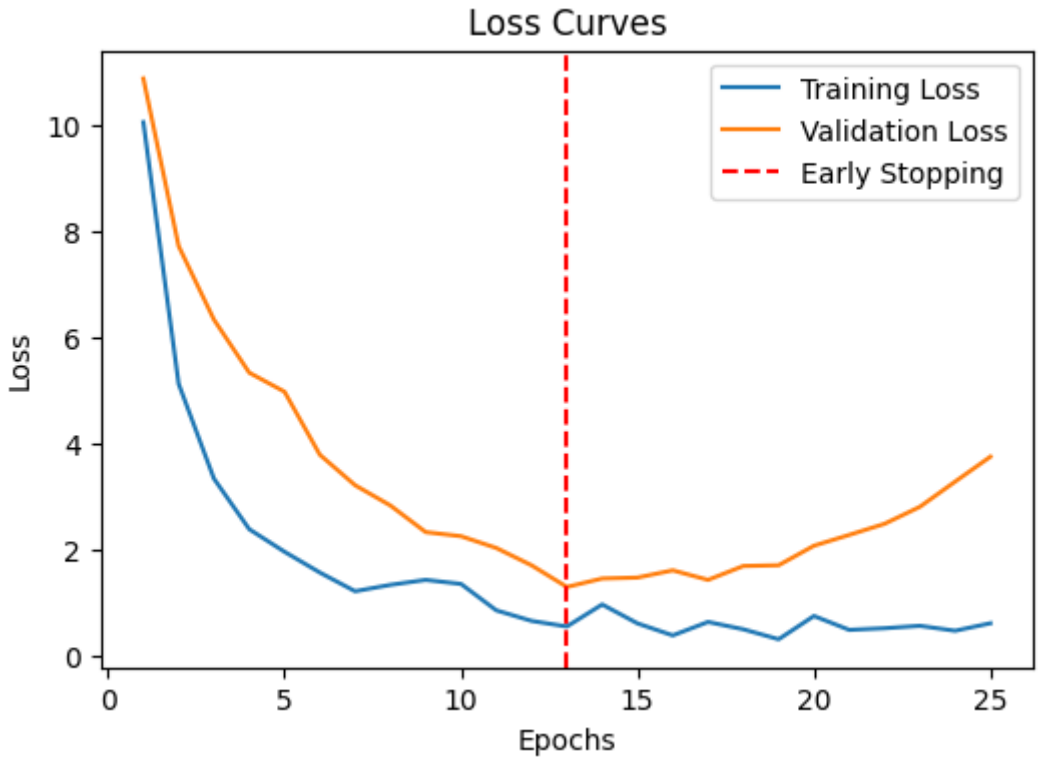
Original	Flip	Rotation	Random crop
			
• Image without any modification	• Flipped with respect to an axis for which the meaning of the image is preserved	• Rotation with a slight angle • Simulates incorrect horizon calibration	• Random focus on one part of the image • Several random crops can be done in a row

Color shift	Noise addition	Information loss	Contrast change
			
• Nuances of RGB is slightly changed • Captures noise that can occur with light exposure	• Addition of noise • More tolerance to quality variation of inputs	• Parts of image ignored • Mimics potential loss of parts of image	• Luminosity changes • Controls difference in exposition due to time of day

Source: CS 230 - Deep Learning Tips and Tricks Cheatsheet

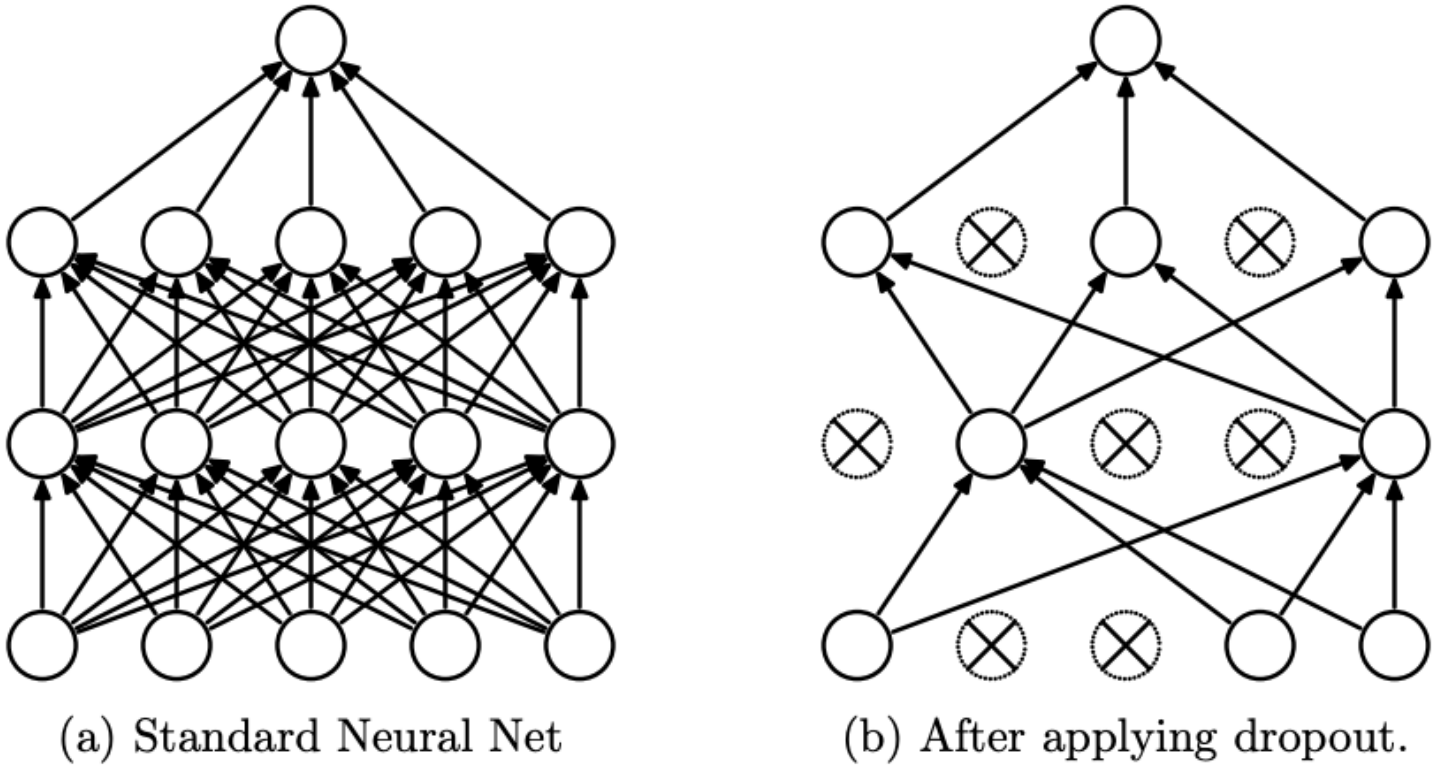
Early Stopping

- Monitor performance (loss curves)
- Save checkpoints
- Set patience parameter
- Stop training and fall back to checkpoint
- Result: better performance and less computation



Dropout

- Randomly turn off a percentage of neurons at each layer, at each training step
- The network can't rely on any particular neuron for making predictions
- Common value is 0.5



(a) Standard Neural Net

(b) After applying dropout.

Source: Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Srivastava et al., 2014)

Optimization

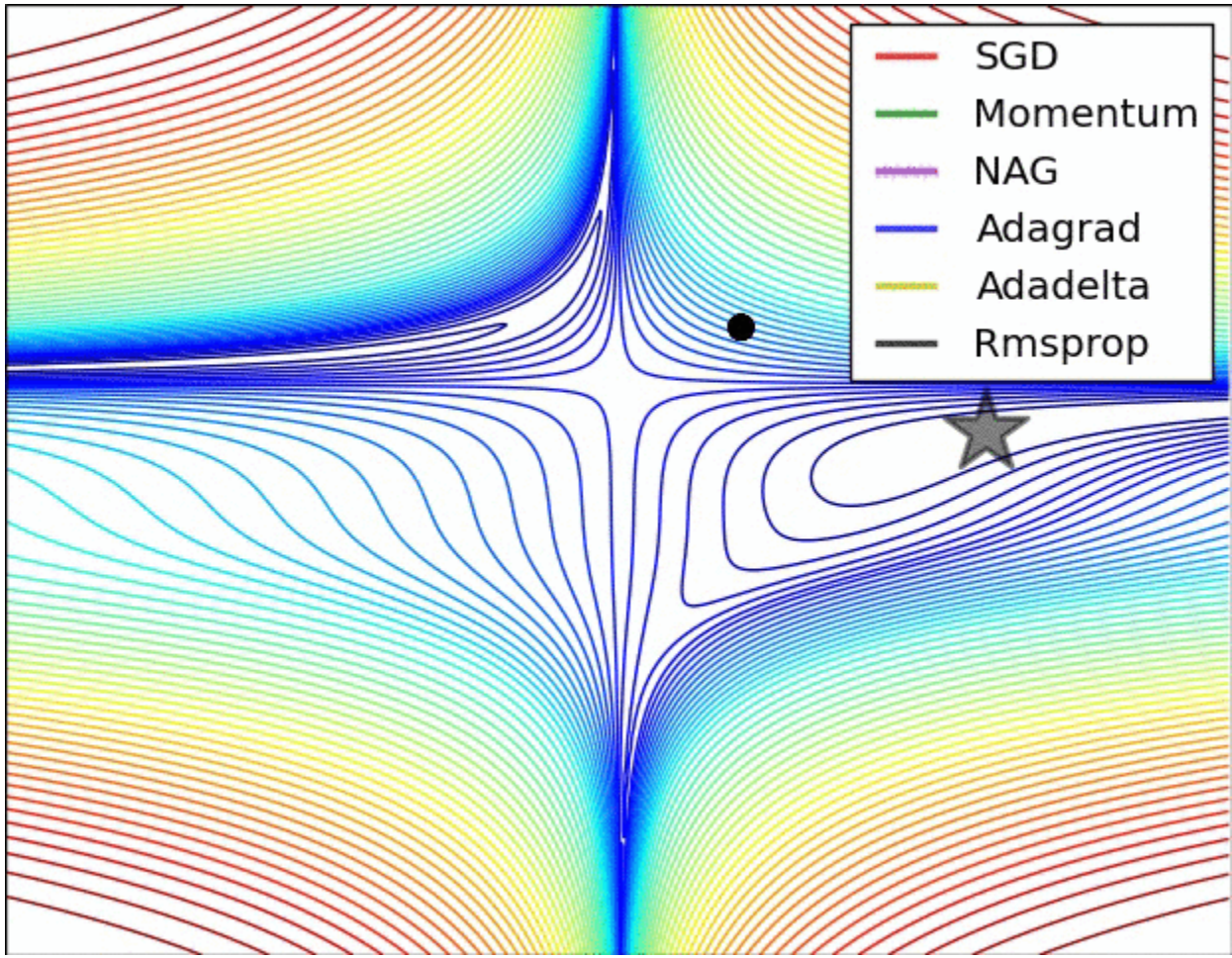
Deep Learning Optimization

Main aspects:

- Convergence speed
- Local minima
- Vanishing and Exploding Gradients

Choices:

- Activation Functions
- Weight Initialization
- Batch Size
- Learning Rate
- Optimizer



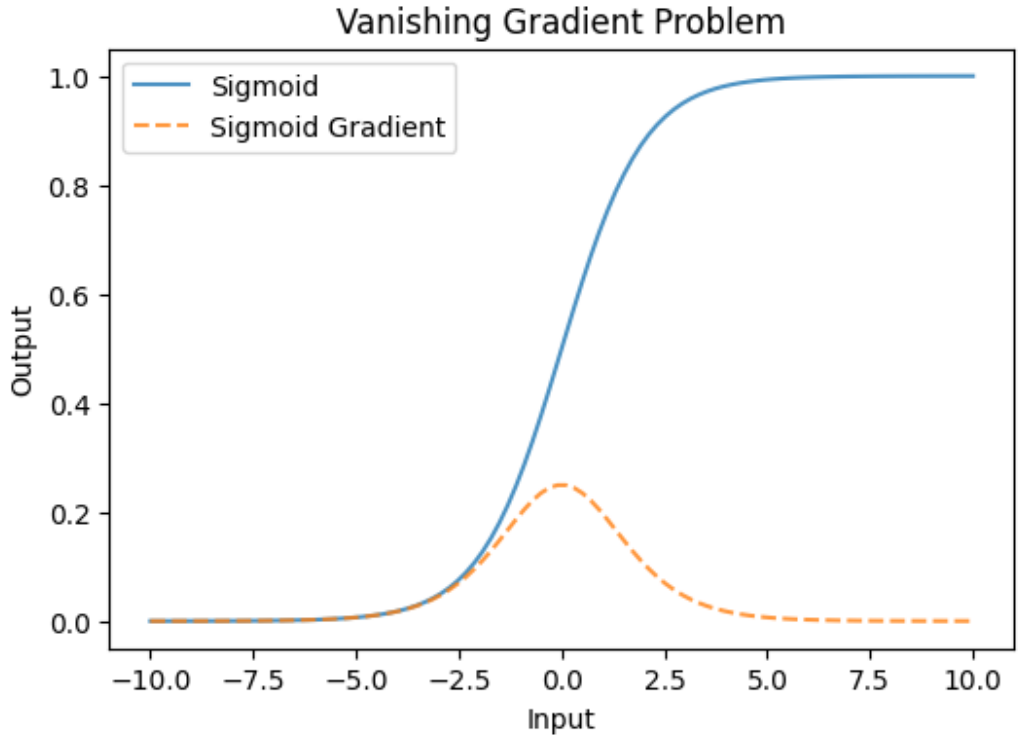
Source: Designing Your Neural Networks. A Step by Step Walkthrough | by Lavanya Shukla | TDS Archive | Medium

Vanishing and Exploding Gradients

- Sigmoid activation: smooth, monotonic, differentiable everywhere
- However, a deep network and the chain rule can cause a vanishing gradient
- $f(x_1, x_2, w_1, w_2) = \sigma(w_1 x_1) = \sigma\big(prod(x_1, w_1)\big)$
- $\frac{\partial f}{\partial w_1} = \frac{\partial \sigma}{\partial prod} * \frac{\partial prod}{\partial w_1}$

- $x_1 = 10, w_1 = 1$
- $\mathrm{prod}(\mathrm{x}_1, \mathrm{w}_1) = 10$
- $\frac{\partial \sigma}{\partial prod}(10) \approx 0$
- $\frac{\partial (w_1 x_1)}{\partial w_1} = x_1 = 10$
- $\frac{\partial f}{\partial w_1} = 0 * 10 = 0$

- $x_1 = 10, w_1 = 0.1$
- $\mathrm{prod}(\mathrm{x}_1, \mathrm{w}_1) = 1$
- $\frac{\partial \sigma}{\partial prod}(1) \approx 0.2$
- $\frac{\partial (w_1 x_1)}{\partial w_1} = x_1 = 10$
- $\frac{\partial f}{\partial w_1} = 0.2 * 10 = 2$



Vanishing and Exploding Gradients

Factors:

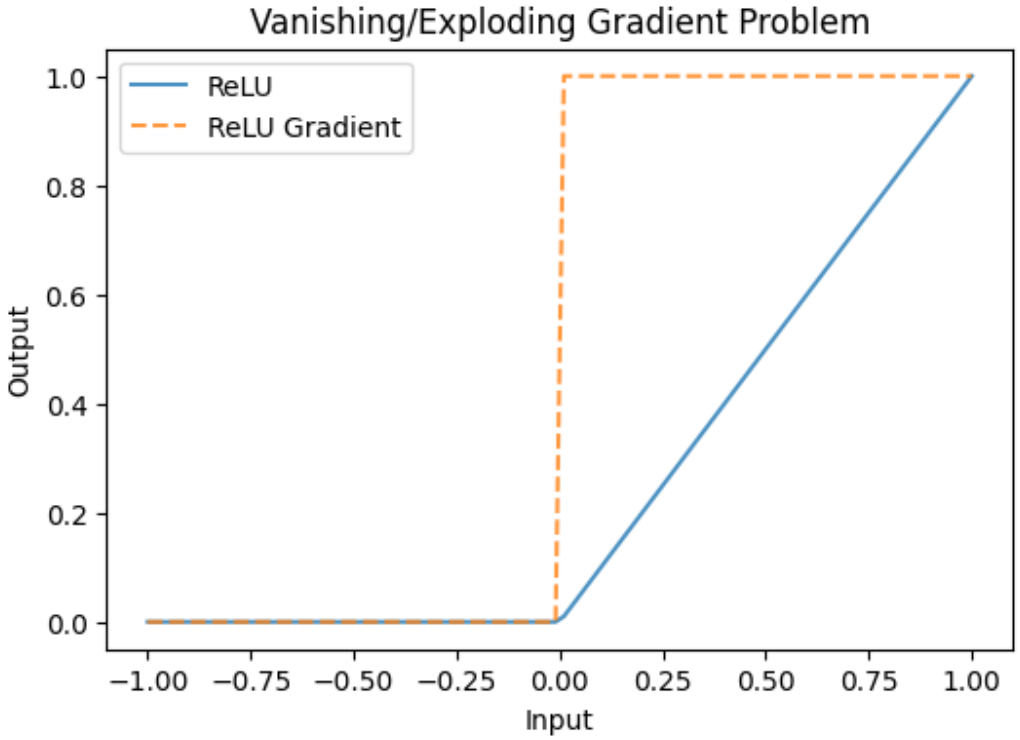
1. Function shape
2. Weight value
3. Data value
4. Chain length

Problems:

- Unstable and slow training
- Training diverges
- Weight updates stop

Possible Solutions:

- Proper weight initialization and activation function
- Feature standardization
- Changing the architecture (depth, LSTM)
- Batch Normalization
- Gradient clipping / adding noise
- Weight Regularization



Stochastic Gradient Descent

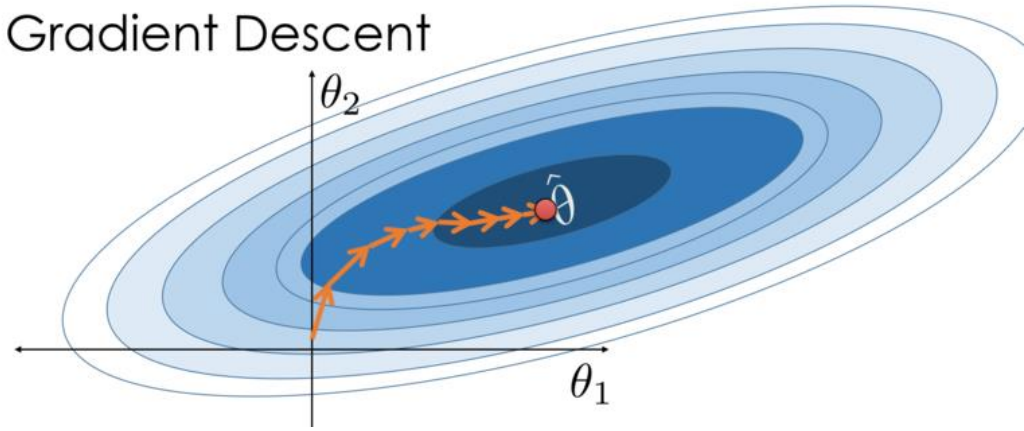
(Full) Gradient Descent: batch_size = n

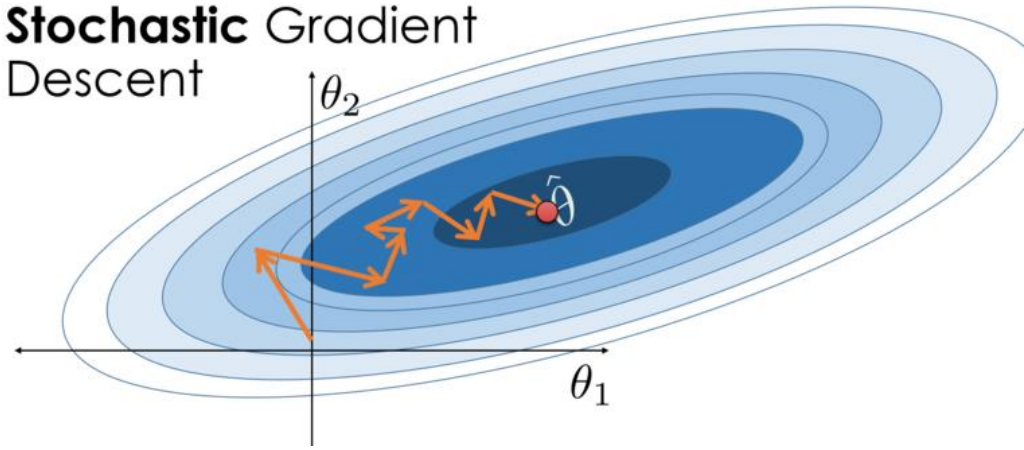
Mini-batch Gradient Descent: 1 < batch_size < n

Stochastic Gradient Descent (SGD): batch_size = 1

Effect of the batch size

- Larger is more efficient (parallelism and less updates) but consumes more memory
- Smaller batches introduce more noise and act as a form of regularization





Source: Stochastic gradient descent - Cornell University Computational Optimization Open Textbook - Optimization Wiki

Batch Normalization

$$x_i \leftarrow \gamma \frac{x_i - \mu_{batch}}{\sqrt{\sigma^2_{batch} + \epsilon}} + \beta$$

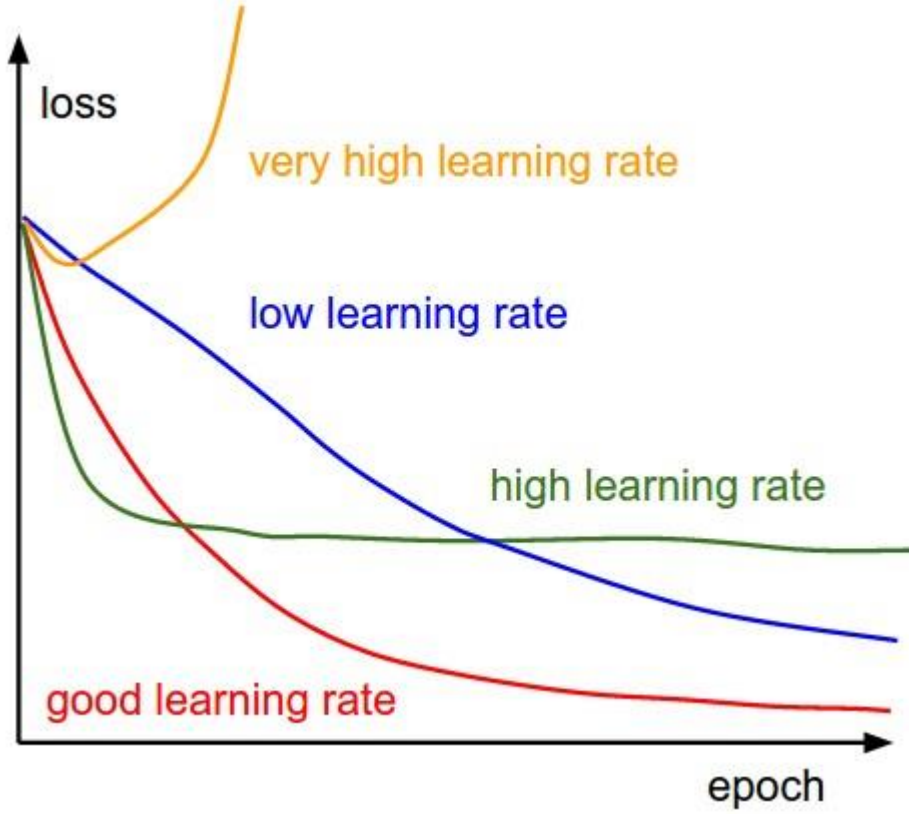
- Introduced in 2015
- Extra layer before an activation function
- Allows higher learning rates
- Reduces dependence on initialization
- Faster convergence
- Acts as a form of regularization
- Reduces vanishing/exploding gradients problem
- Extra computation

Learning Rate

- Very high learning rate causes unstable training or even divergence
- High learning rate might overshoot the minima
- Low learning rate causes slow convergence

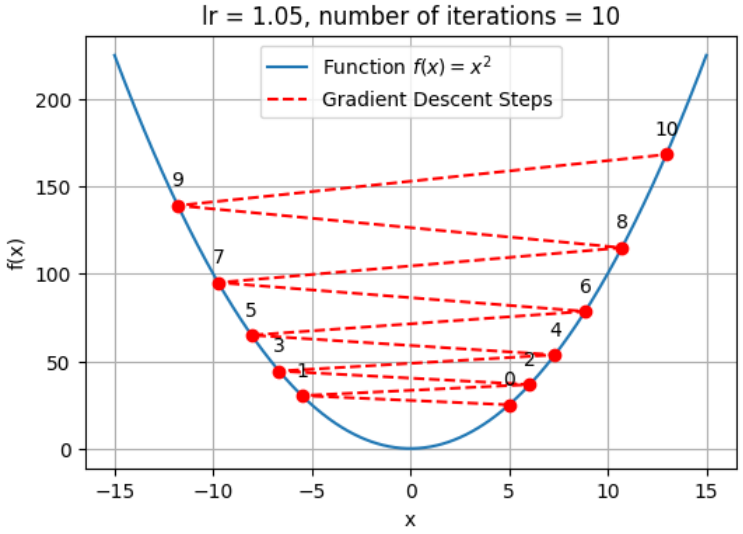
Tips:

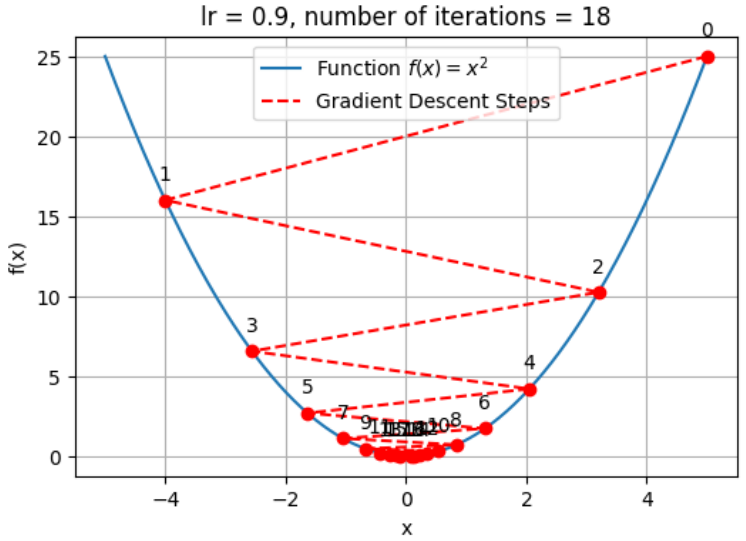
- Experiment with multiple learning rates
- It is often useful to start high and lower the learning rate as the training progresses
- Use a learning rate scheduler (step-based or performance based)
- Optimizers with adaptive learning rates

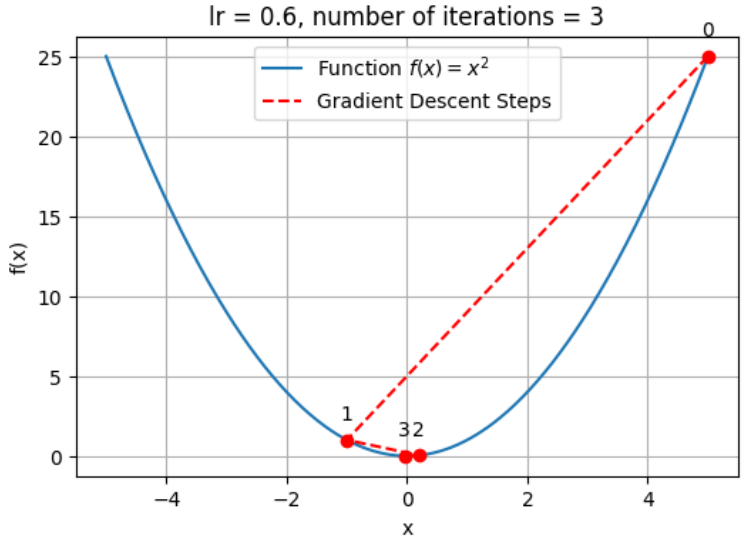


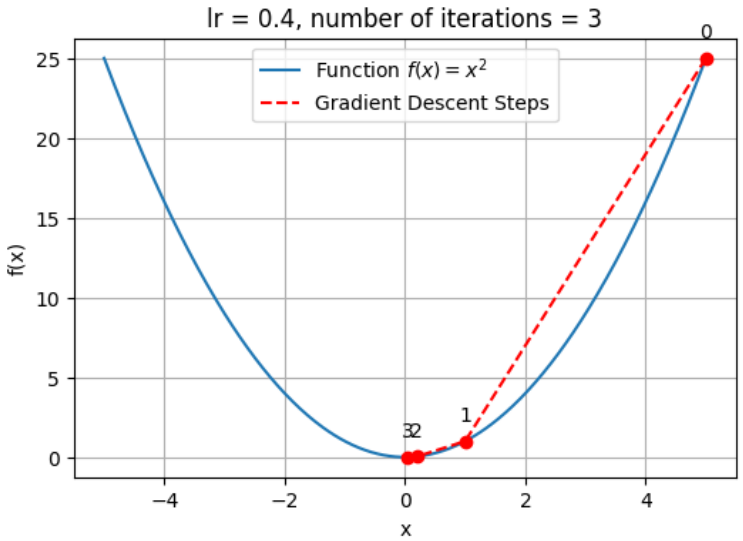
Source: Designing Your Neural Networks. A Step by Step Walkthrough | by Lavanya Shukla | TDS Archive | Medium

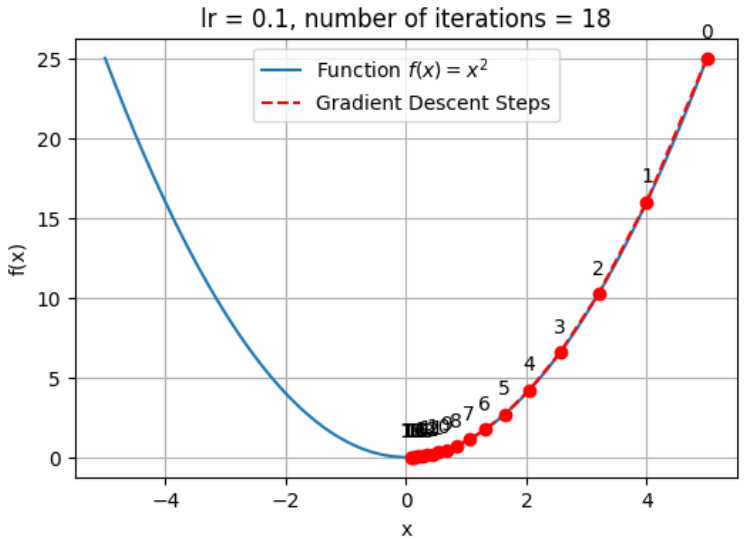
Learning Rate

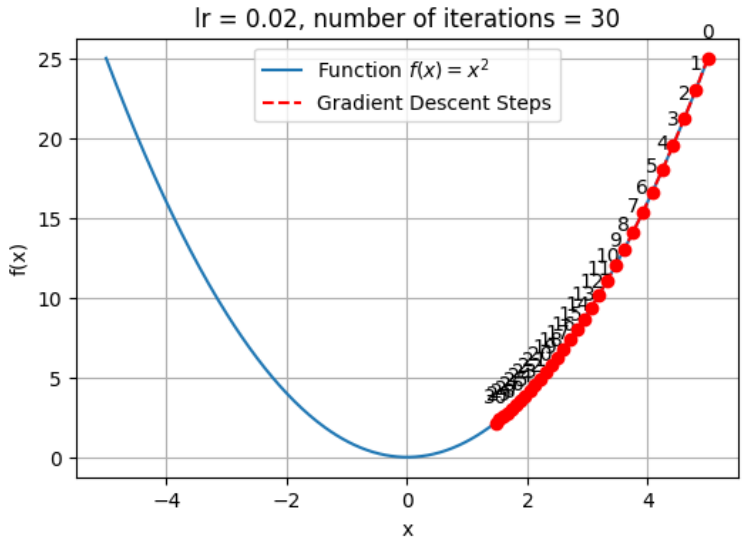












Momentum

Gradient Descent:

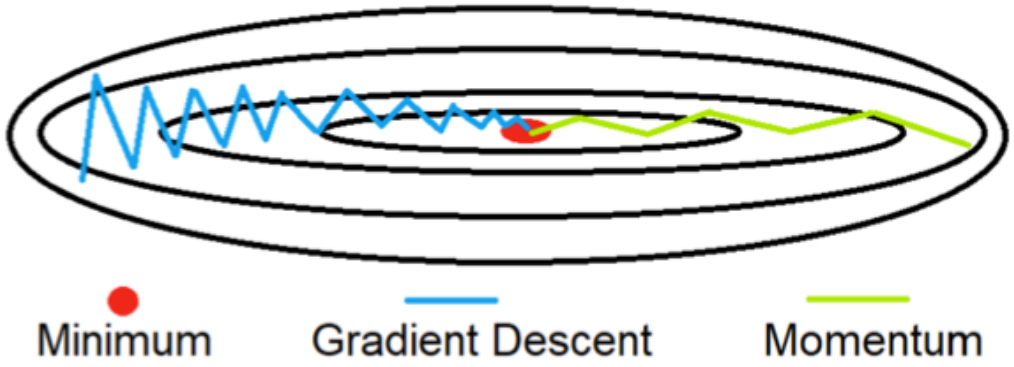
- $G_t = \nabla Loss(\theta_{t-1})$
- $\theta_t = \theta_{t-1} - \alpha * G_t$

Momentum:

- Accelerates gradient descent in the relevant direction and dampens oscillations
- Speeds up training and helps escape local minima
  - $m_t = \mu * m_{t-1} - \alpha * G_t$
  - $\theta_t = \theta_{t-1} + m_t$

Moving average of the gradient (first moment):

- $m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * G_t$
- $\theta_t = \theta_{t-1} - \alpha * m_t$
- β_1: exponential decay rate for the 1st moment estimates (~0.99)



Source: Momentum - Cornell University Computational Optimization Open Textbook - Optimization Wiki

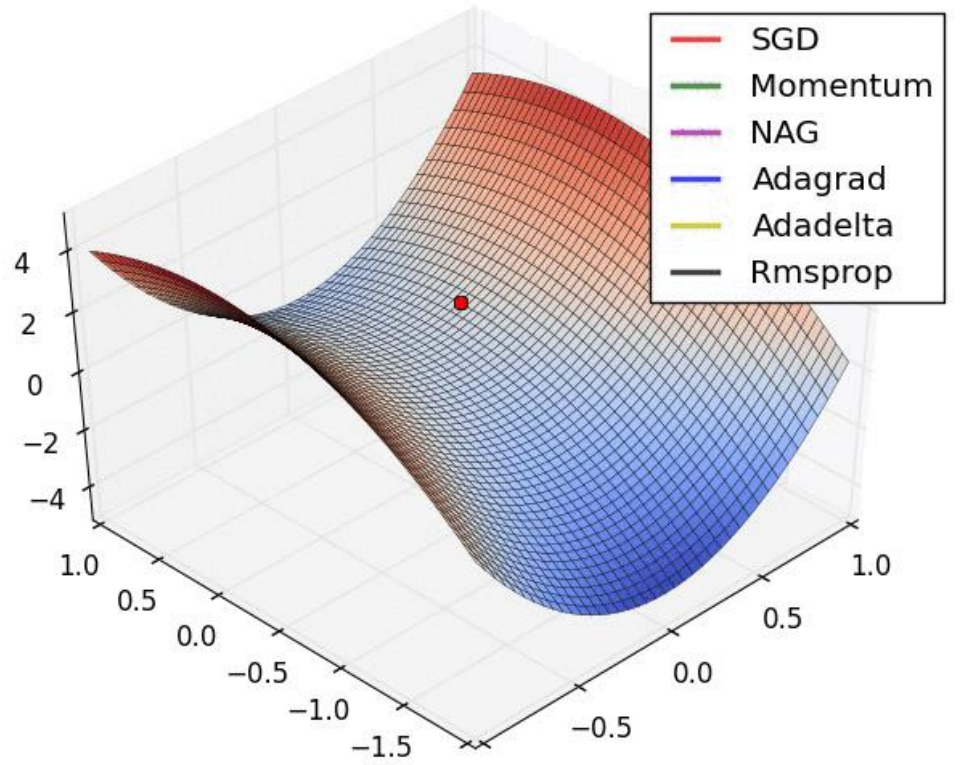
Root Mean Squared Propagation (RMSProp)

Gradient Descent:

- $G_t = \nabla Loss(\theta_{t-1})$
- $\theta_t = \theta_{t-1} - \alpha * G_t$

RMSProp:

- Restricts the oscillations in the steeper directions
- The algorithm can take larger steps in other directions instead
- The learning rate can be increased which reduces convergence time
- Moving average of the second moment:
  - $v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * G_t^2$
  - $\theta_t = \theta_{t-1} - \alpha * \frac{G_t}{\sqrt{v_t} + \epsilon}$
  - β_2*:* exponential decay rate for the 2nd moment estimates



Source: RMSProp - Cornell University Computational Optimization Open Textbook - Optimization Wiki

ADAM (Adaptive Moment Estimation)

- Combines Momentum and RMSProp
- Unbiased estimates of moments:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- Weight Update Formula:

$$\theta_t = \theta_{t-1} - \alpha * \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$$

Deep Learning

- Introduction
- Architectures
- Training
- Optimization

Thank you for your attention!

Your feedback would be much appreciated:

Any Questions?

Gergely Zsombor Haász

haasz.zsombi@gmail.com