
Machine Listening for Music and Sound Analysis

Lecture 2 – Machine Learning/Deep Learning

Dr.-Ing. Jakob Abeßer

Fraunhofer IDMT

Jakob.abesser@idmt.fraunhofer.de

<https://machinelisting.github.io>

Learning Objectives

- Introduction
- Learning paradigms
- Machine learning (ML) project pipeline
- Deep learning

Introduction

■ Goals

- “...give computers the ability to learn without being explicitly programmed” [Samuels, 1959]
- Learning structures in given (un)labeled data to make predictions on new / unseen data

■ Paradigm change

- Now: joint representation learning (features) & data modeling (classification)
- Before: manually designed / general-purpose features

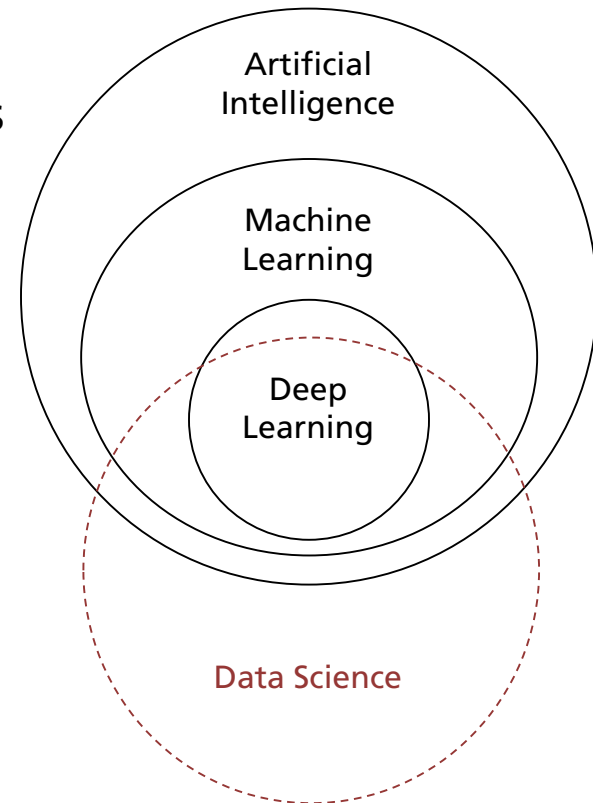
■ Related disciplines

- Statistics, data science, optimization

Introduction

Terminology

- Artificial Intelligence (AI)
 - „an agent’s ability to achieve goals in a wide range of environments“ [Legg & Hutter, 2007]
- Machine Learning (ML)
 - Pattern recognition, data modeling, learning, prediction
- Deep Learning (DL)
 - (Brain-inspired) artificial neural networks (ANN)
- Data Science
 - Knowledge extraction from data



Introduction

Application Scenarios

- Computational finance (credit scoring, algorithmic trading)
- Computer vision (face & object recognition, motion detection)
- Computational biology (tumor detection, drug discovery, DNA sequencing)
- Energy (price & load forecasting)
- Predictive Maintenance (automotive, aerospace, manufacturing)
- Natural language processing (sentiment classification, text search, translation)
- Machine Listening (music transcription, instrument recognition, sound event detection, acoustic scene classification)

Learning Paradigms

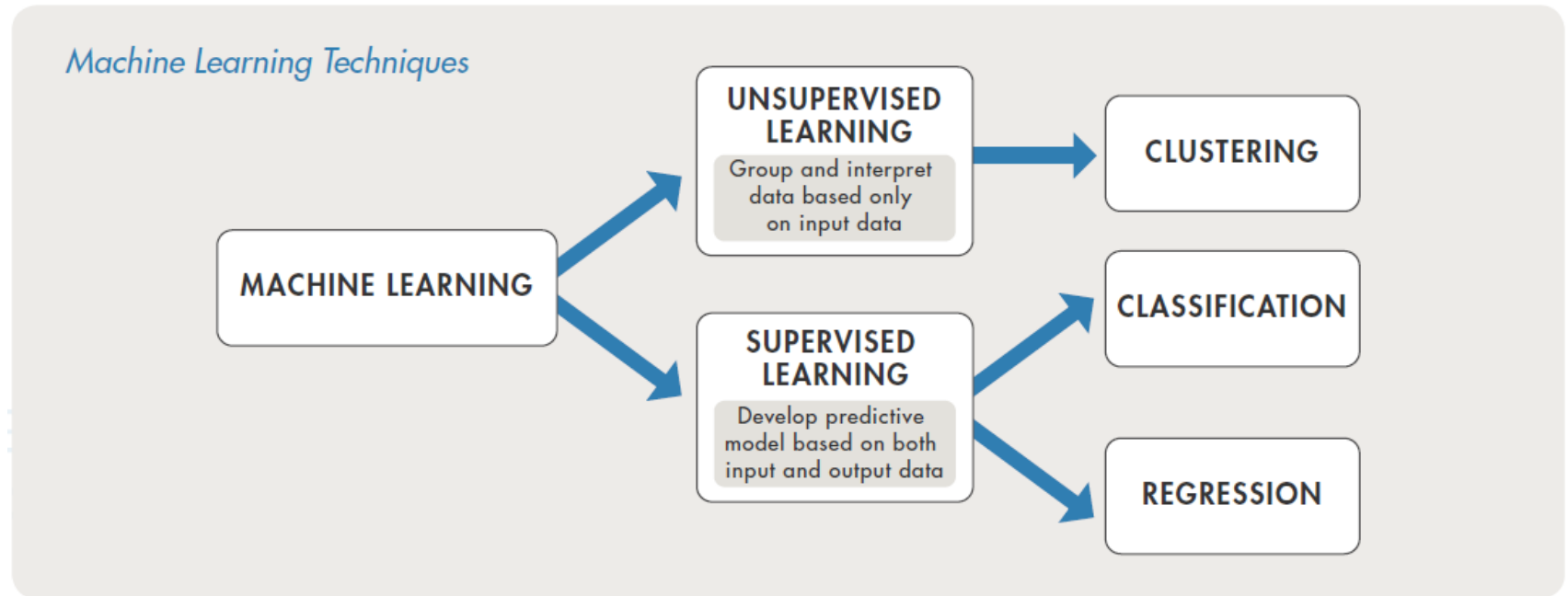


Fig. 1

Learning Paradigms

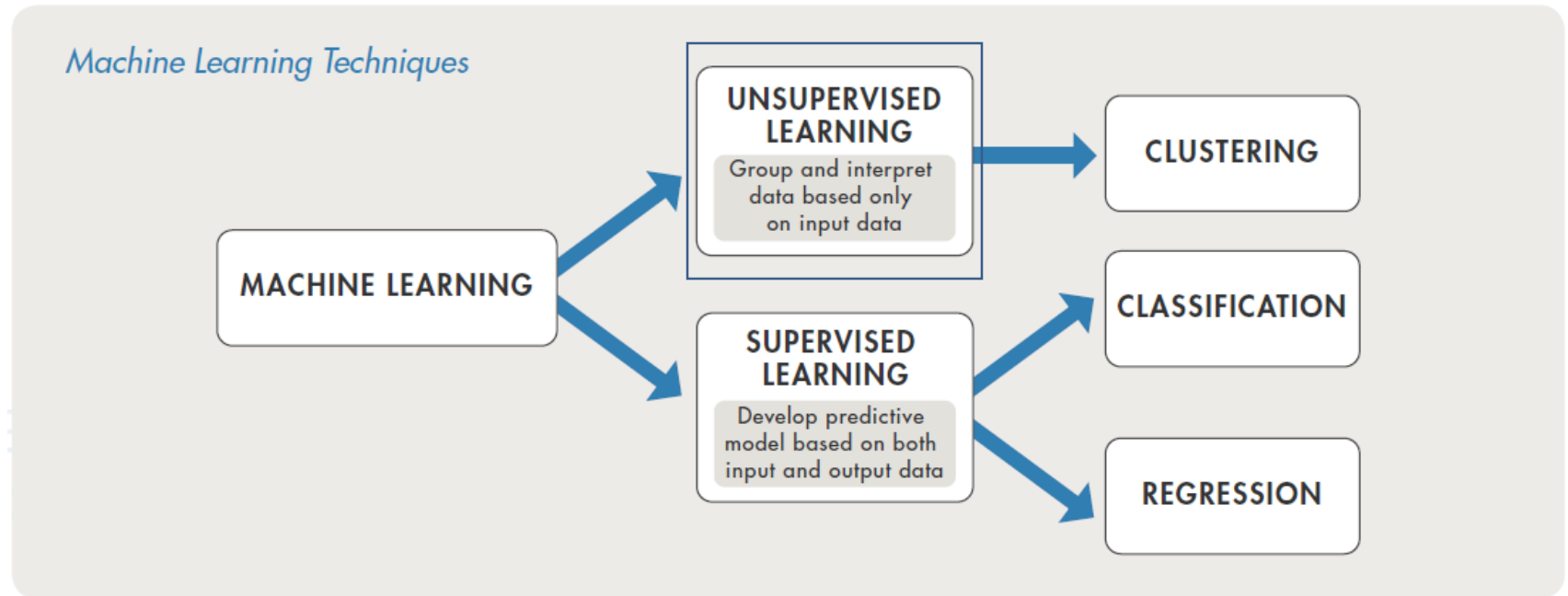


Fig. 1

Learning Paradigms

Unsupervised Learning

- Goal
 - Model hidden structure in data
 - Density estimation
- Example
 - K-means clustering
 - Data partitioning into K clusters
 - Minimize within-cluster variance

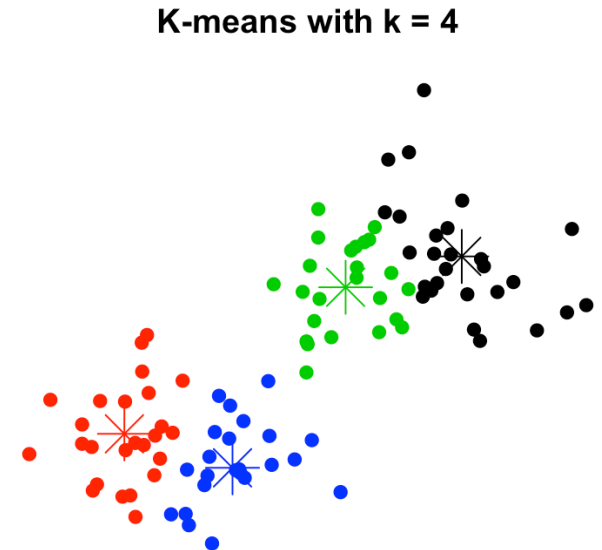


Fig. 2

Learning Paradigms

Supervised Learning

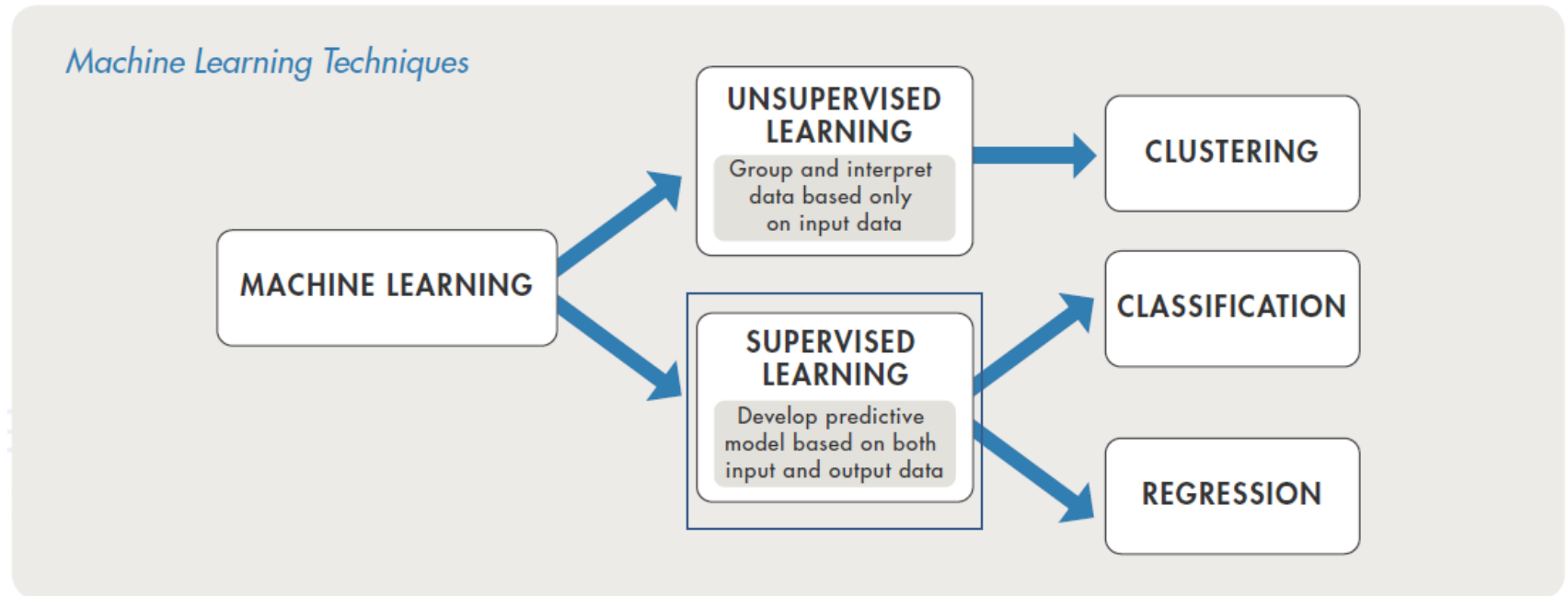
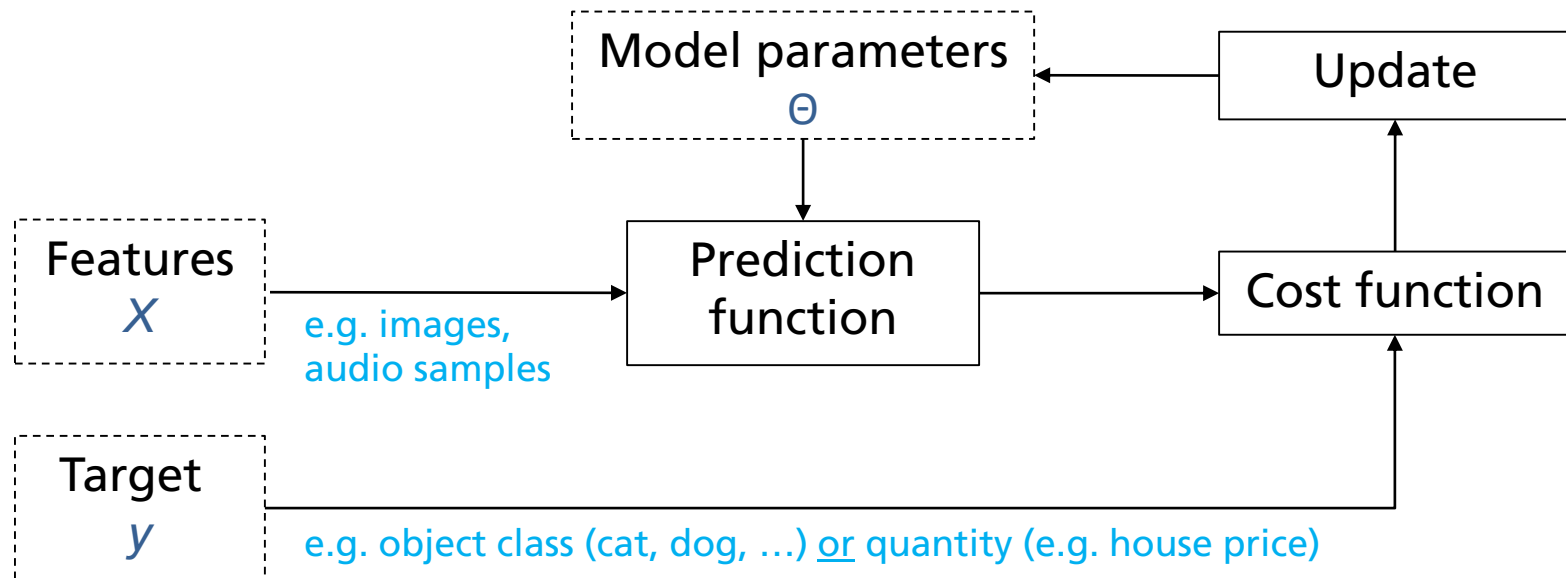


Fig. 1

Learning Paradigms

Supervised Learning



Learning Paradigms

Supervised Learning - Classification

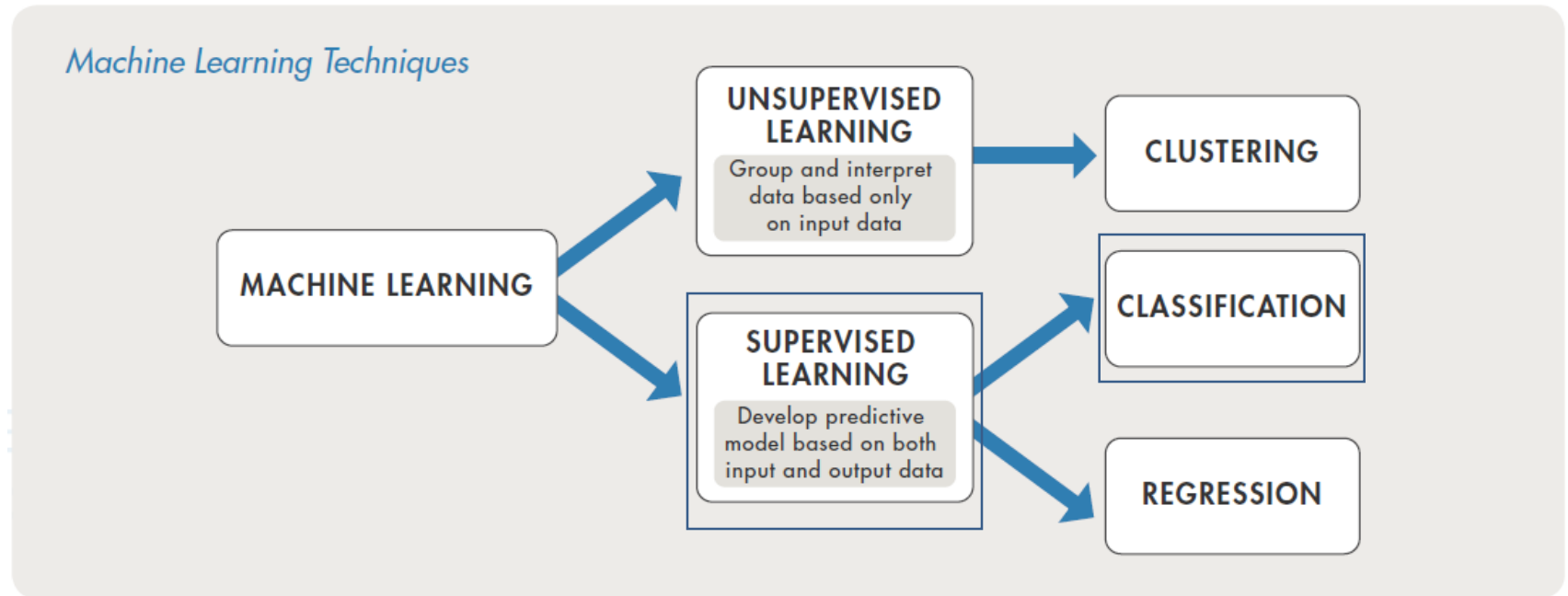


Fig. 1

Learning Paradigms

Supervised Learning - Classification

- Predict one / multiple categorical label from features
 - Examples → music genre, instrument(s), key
- Feature space modeling (Example: 2 classes)

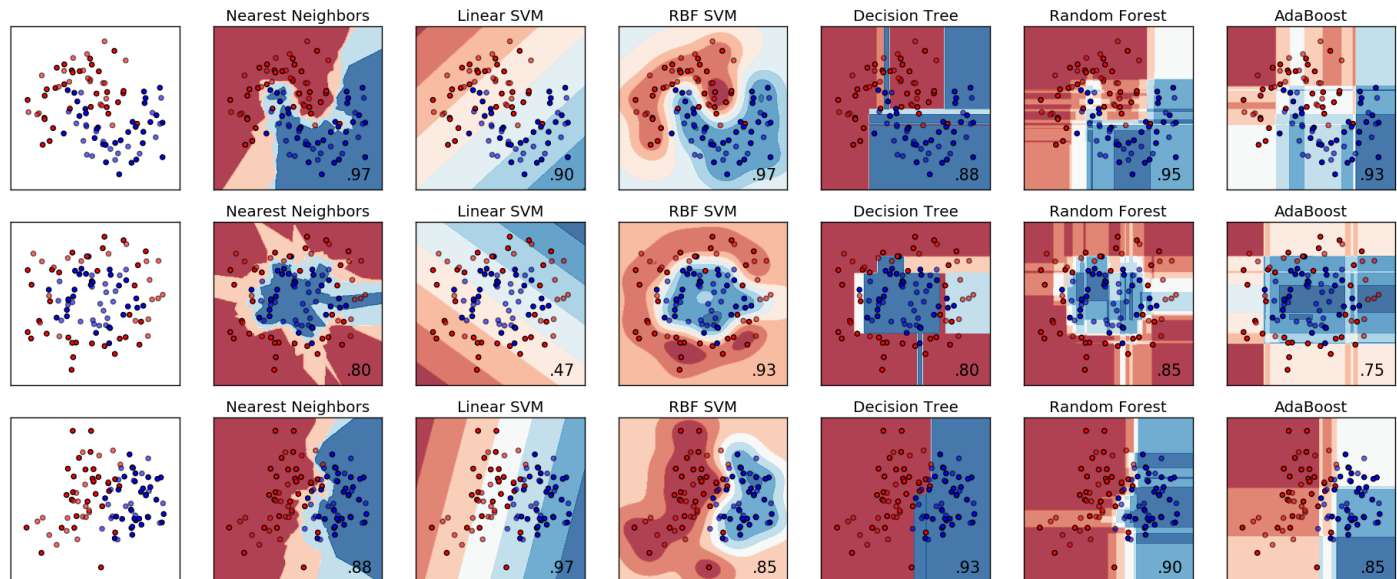


Fig. 3

Learning Paradigms

Supervised Learning - Classification

- Example: K-Nearest Neighbors
 - Training → store all examples
 - Development → find best K
 - Test → assign test item to dominant class label of the K closest training data items
- Distance measures
 - Euclidean distance, manhattan distance, cosine distance, ...

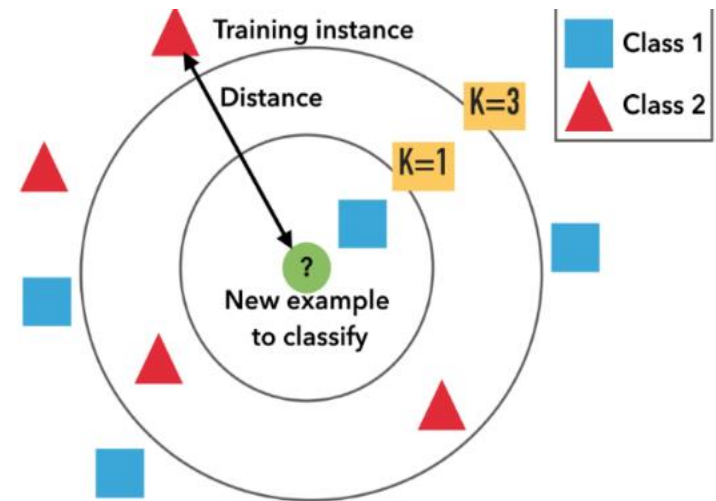


Fig. 4

Machine Learning

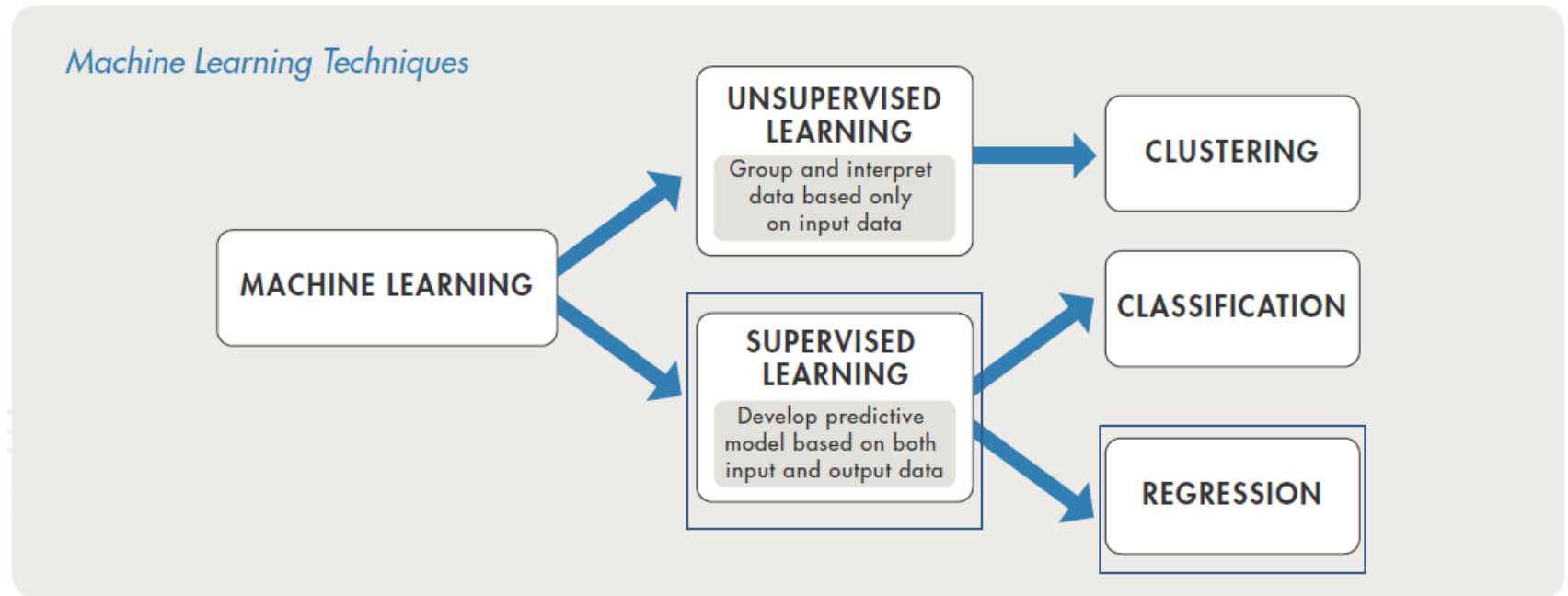
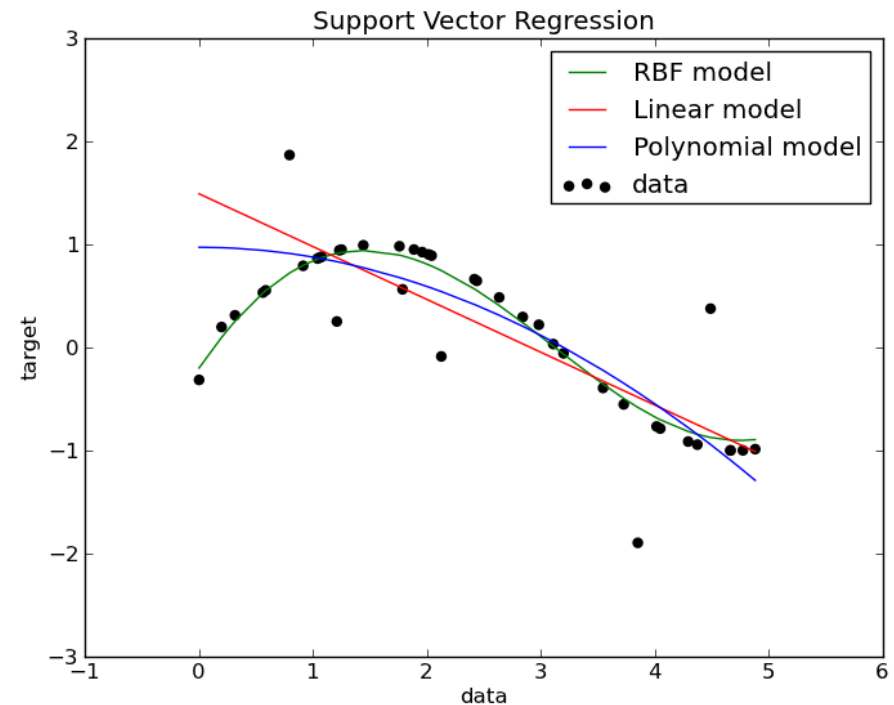


Fig. 1

Learning Paradigms

Supervised Learning - Regression

- Predicting a continuous quantity from features
- Examples
 - House price prediction



Learning Paradigms

Self-supervised Learning

■ Problem:

- Supervised learning requires a lot of (annotated) data → expensive!

■ Solution:

- Train model on related (proxy) task (derive labels from the data!)

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the occluded from the visible
- ▶ Pretend there is a part of the input you don't know and predict that.

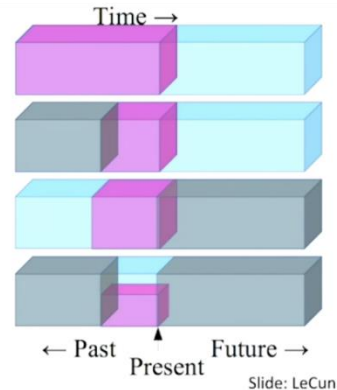
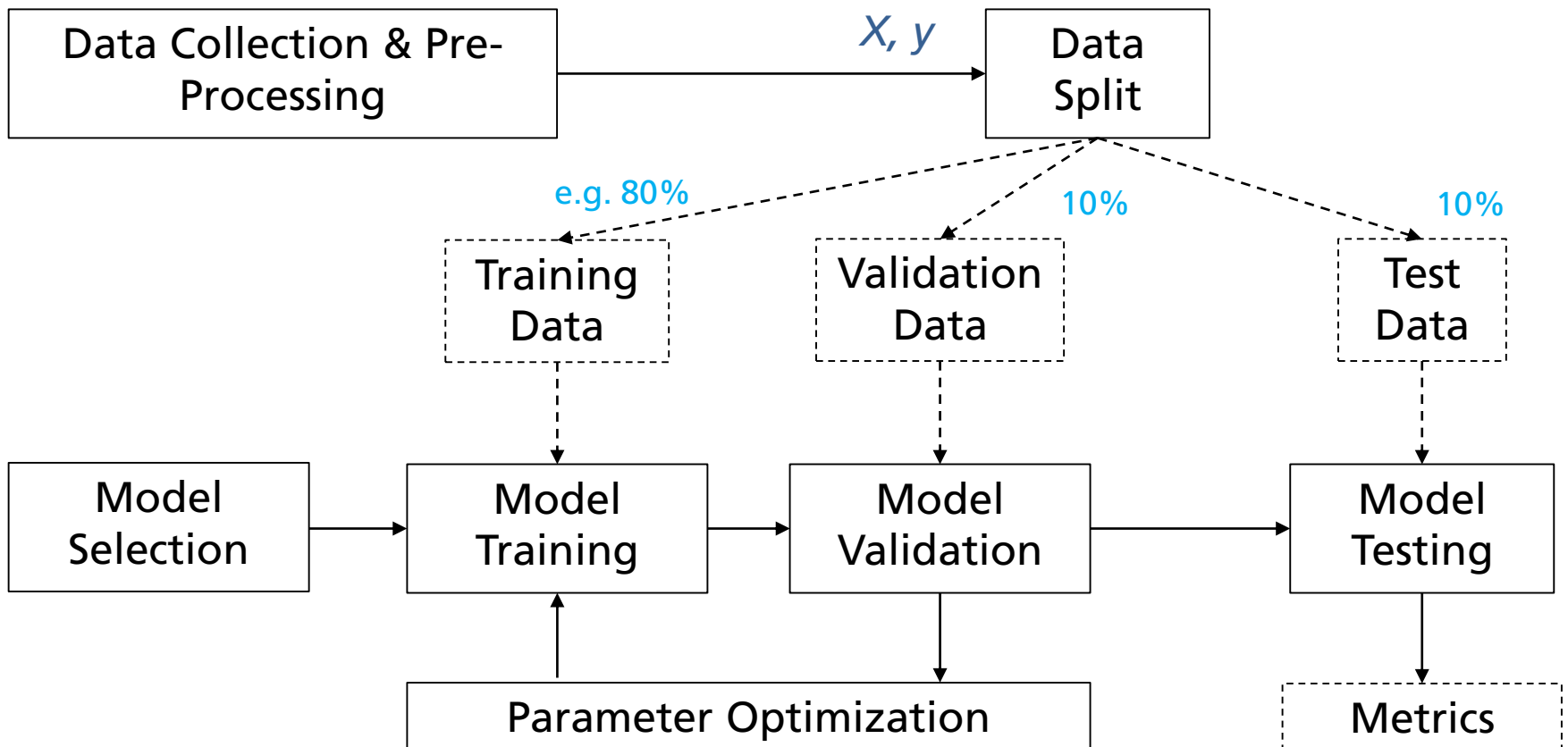


Fig. 5

- Transfer learning → Train model with fewer data on target task (e.g., sentiment classification in text)

ML Project Pipeline



ML Project Pipeline

Data Collection & Pre-Processing

- Data collection




- Check for available data resources for given (or related) task
- Collect / record / annotate new data
- Ensure data variability
 - Example (acoustic condition monitoring) → include different motor engine types & conditions, recording locations, microphones, ...

- Data pre-processing

- Cleanup remove errors, silence, empty files, ...
- Balance dataset (proportion among class examples)
- Normalize (depends on the model)

ML Project Pipeline

Data Split

- Training Set 
 - Model learns from this data
- Validation Set (a.k.a development / dev set) 
 - Used to fine-tune the model (hyper)parameters
 - Model occasionally sees but does not learn from this data
- Test set 
 - Only used once after the model training & tuning is completed
 - Should reflect the targeted real-world use case for the model
- Common split ratios
 - 80% - 10% - 10% or even 98% - 1% - 1% (for bigger datasets)



ML Project Pipeline

Model Selection

- Many models and approaches exist
 - Types (SVM, GMM, logistic regression, DNNs)
 - Hyperparameters (SVM kernel functions, DNN layer types)
- Often constrained by the use-case / task
 - Model complexity (memory, training time, training data amount)
- Different models require different feature pre-processing
- Use simple models for simple tasks

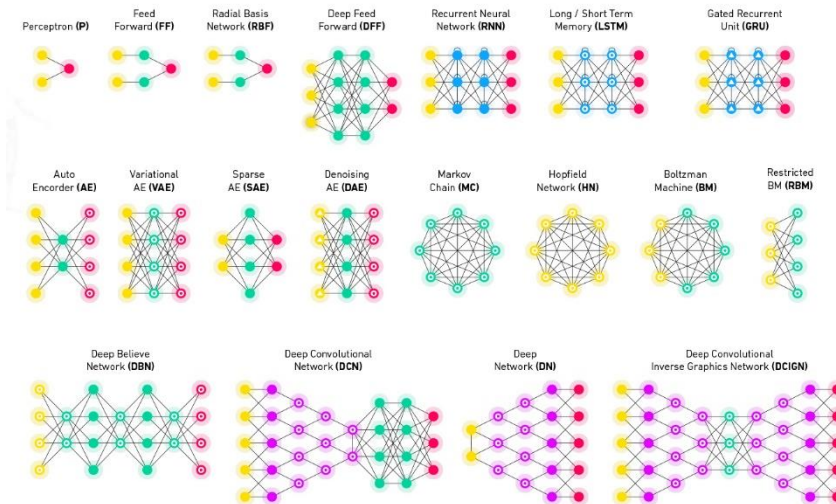


Fig. 6

ML Project Pipeline

Model Training

- Iterative process
 - Use (batches) of training data to iteratively improve model predictions (optimization)
 - Learn from examples
 - Update model parameters according to loss function
- Often start with random parameter initialization

ML Project Pipeline

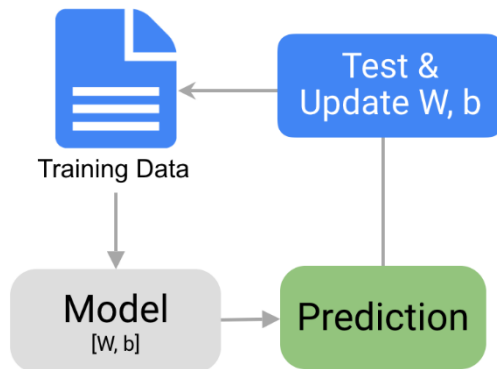
Model Training

- Example: linear regression

$$y = W \cdot x + b$$

y	→ target (output)
x	→ features (input)
W	→ weights
b	→ bias

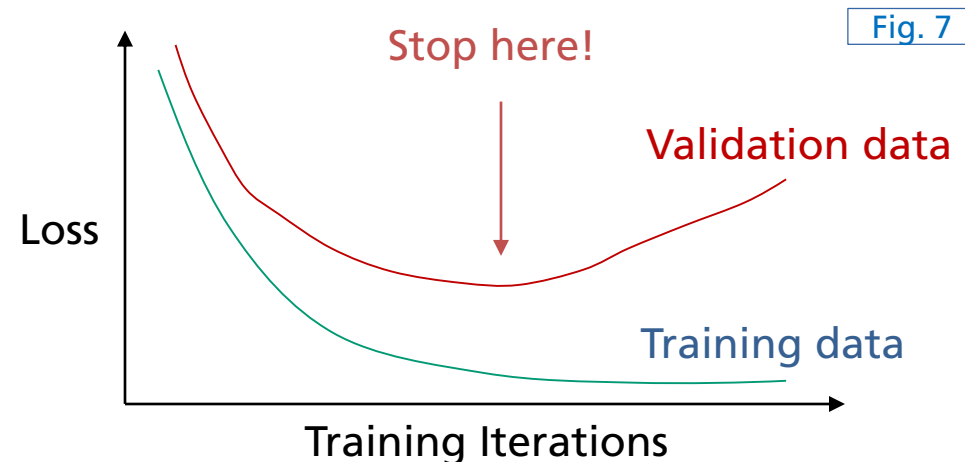
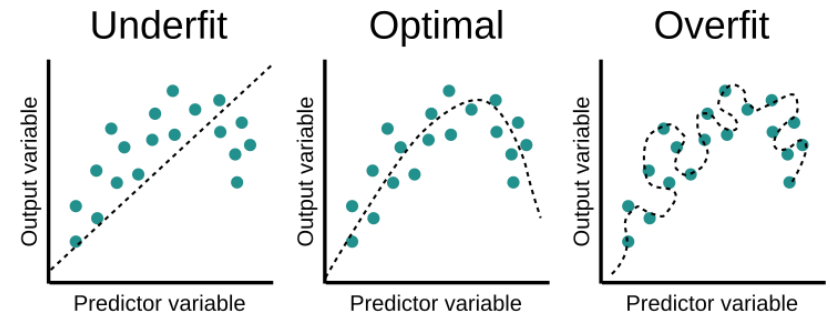
- Training loop



ML Project Pipeline

Model Validation

- Unbiased evaluation of the model after each training iteration
- Helps to
 - optimize model (hyper)parameters
 - detect overfitting on training data
 - stop the training



ML Project Pipeline

Model Testing

- Final model evaluation
- Test set should reflect final application scenario (same data distribution)
- Evaluation metrics (binary classification)
 - Accuracy – % of correct classifications
 - Recall - % of actual positive examples were classified as positive
 - Precision - % of positive predictions are actually positive
 - F-Score – harmonic mean of precision and recall

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Fig. 8

Deep Learning

Introduction

- Artificial neural networks → mimic brain processing
 - Connected neurons (weighted input summation & non-linear processing)
- Deep neural networks → multiple layers

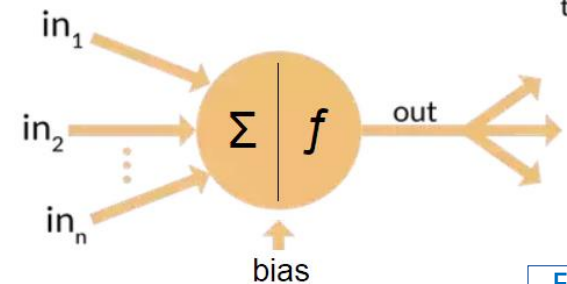
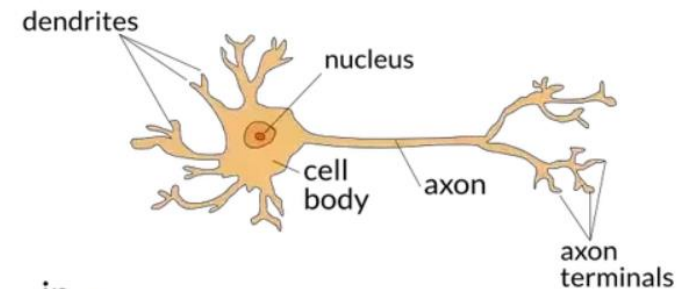
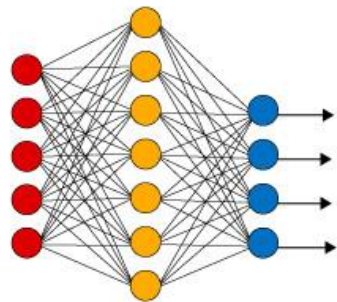
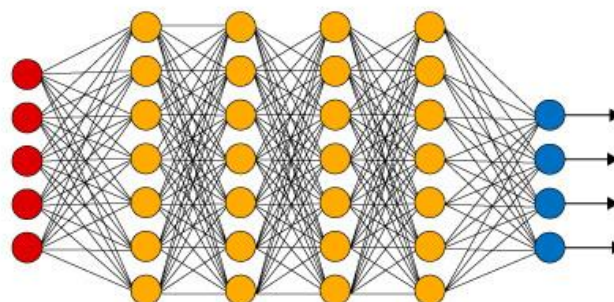


Fig. 9

Simple Neural Network



Deep Learning Neural Network



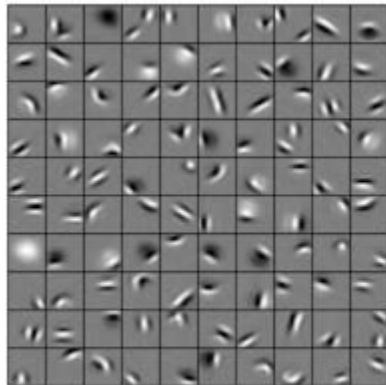
● Input Layer ● Hidden Layer ● Output Layer

Fig. 10

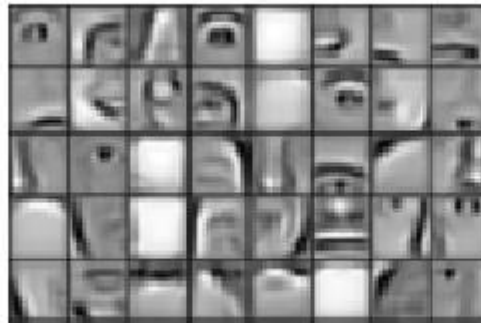
Deep Learning

Introduction

- Hierarchical feature learning
 - Example (face recognition)



Edges, curves



Shapes, object parts



Objects (faces)

Fig. 11

First layers

Final layers

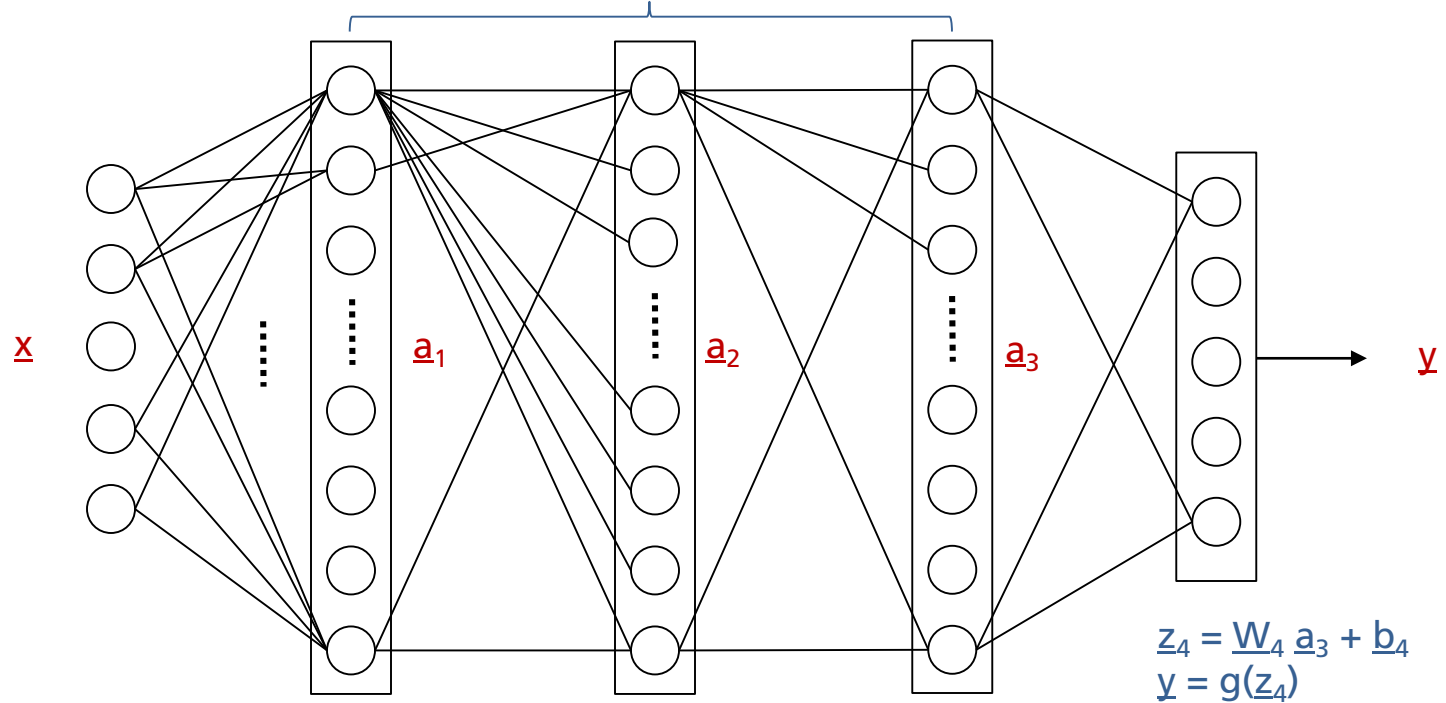
Deep Learning

Deep Neural Networks

Input layer

Hidden layers

Output layer



- 1) Weights w & bias b
- 2) Non-linear activation function $g(z)$

$\underline{W}_1, \underline{b}_1$

$$\underline{z}_1 = \underline{W}_1 \underline{x} + \underline{b}_1$$
$$\underline{a}_1 = g(\underline{z}_1)$$

$\underline{W}_2, \underline{b}_2$

$$\underline{z}_2 = \underline{W}_2 \underline{a}_1 + \underline{b}_2$$
$$\underline{a}_2 = g(\underline{z}_2)$$

$\underline{W}_3, \underline{b}_3$

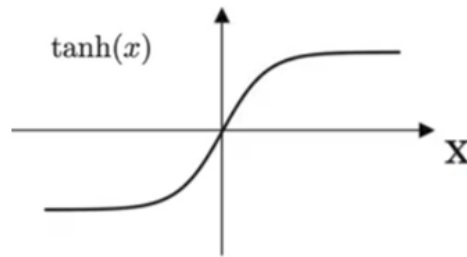
$$\underline{z}_3 = \underline{W}_3 \underline{a}_2 + \underline{b}_3$$
$$\underline{a}_3 = g(\underline{z}_3)$$

Deep Learning

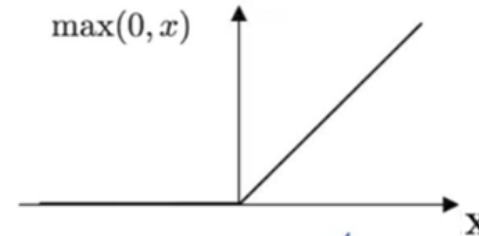
Activation Functions

- Activation functions add non-linearity
- Make networks more powerful in (complex) pattern recognition
- Examples:

Hyper Tangent Function



ReLU Function



Sigmoid Function

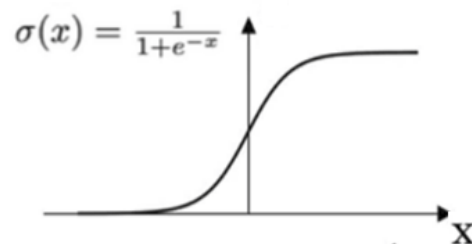
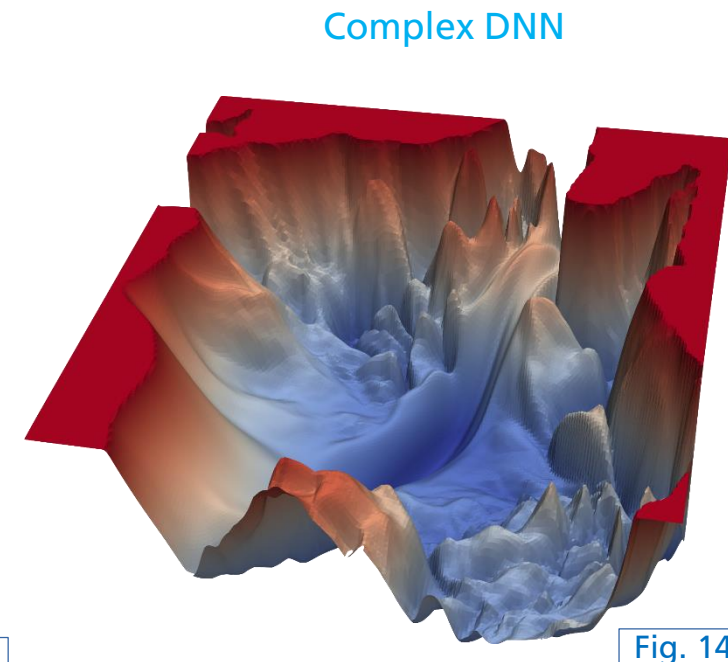
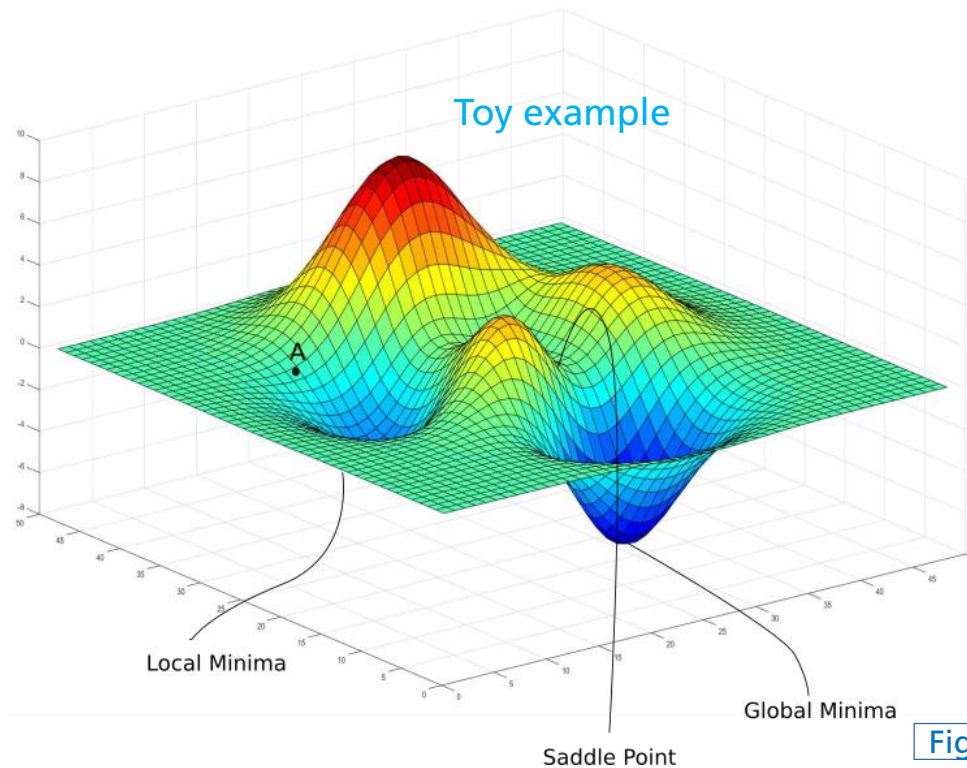


Fig. 12

Deep Learning Training

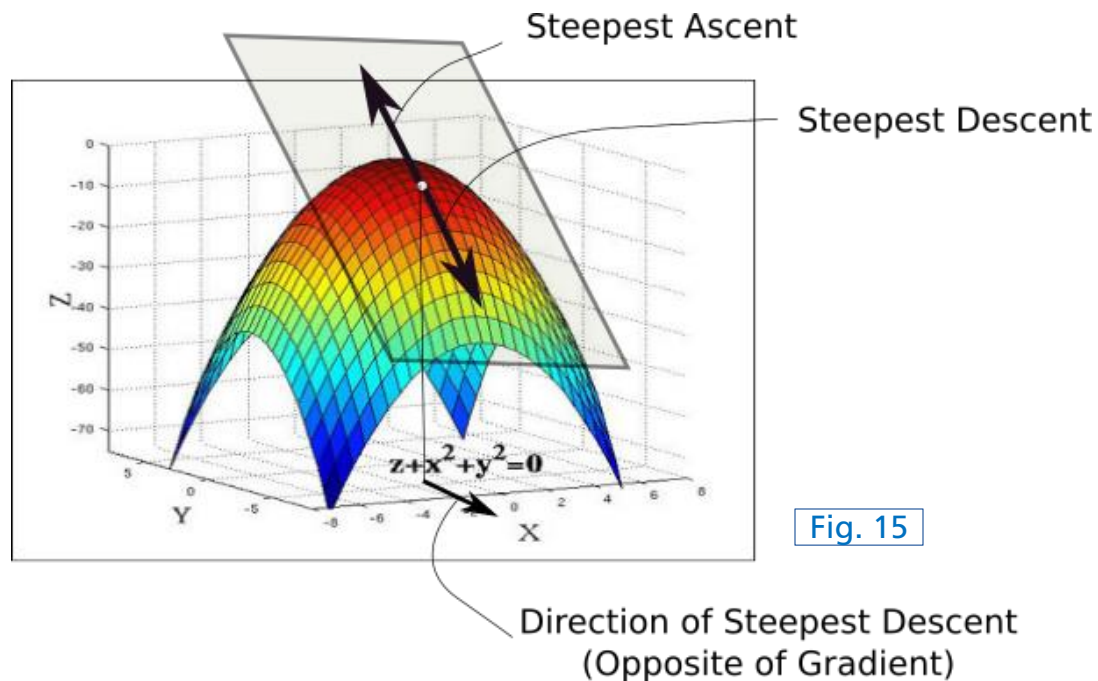
- Loss contour

- Goal → find global minima



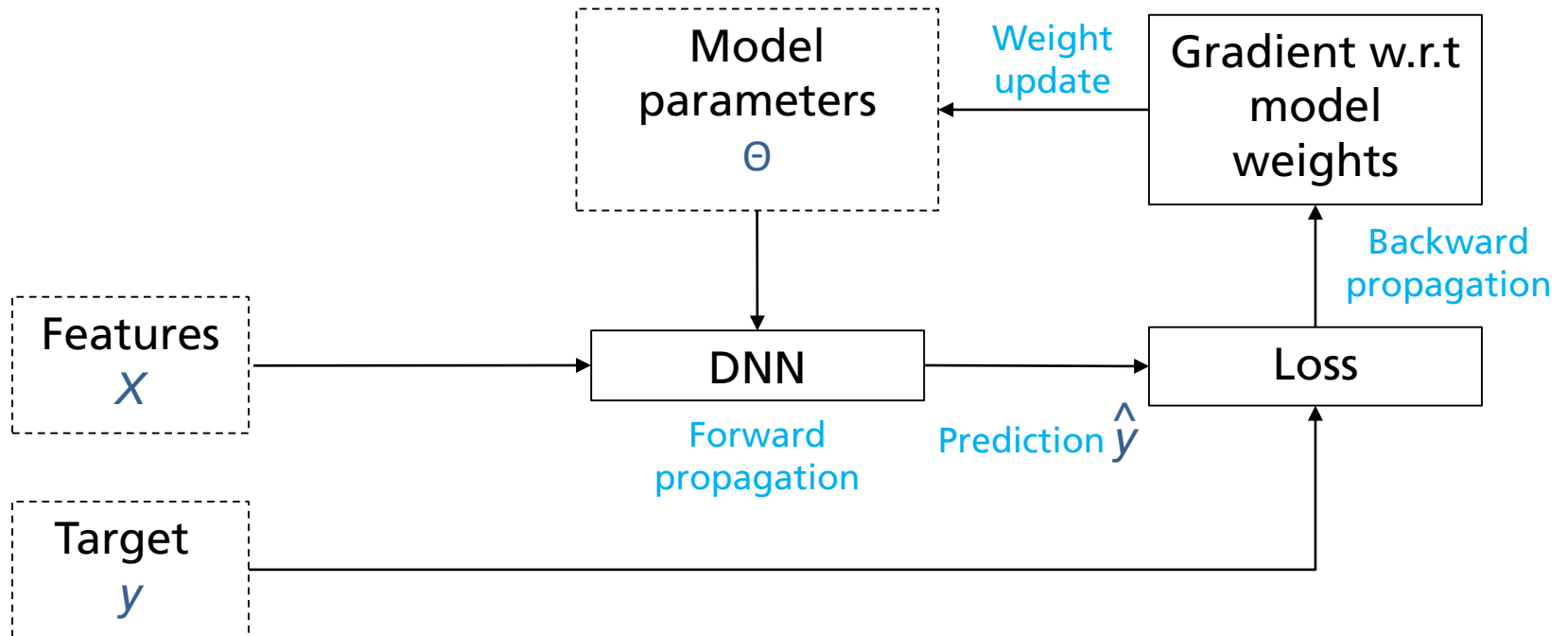
Deep Learning Training

- Gradient descend
 - Move into opposite direction of gradient
 - Learning rate effects step size



Deep Learning Training

■ Back propagation



Deep Learning Playground

- A neural network playground!
 - <https://playground.tensorflow.org>

Deep Learning

Convolutional Neural Networks (CNN)

- Convolutional layers
 - Local connectivity → receptive field
 - Shared weights
 - Translation Equivariance → translation of input = translation of activations
- Pooling → local aggregation

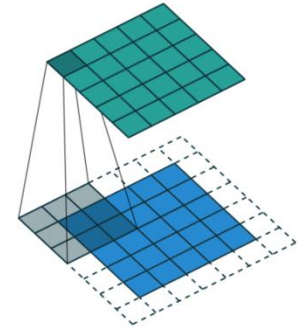


Fig. 16

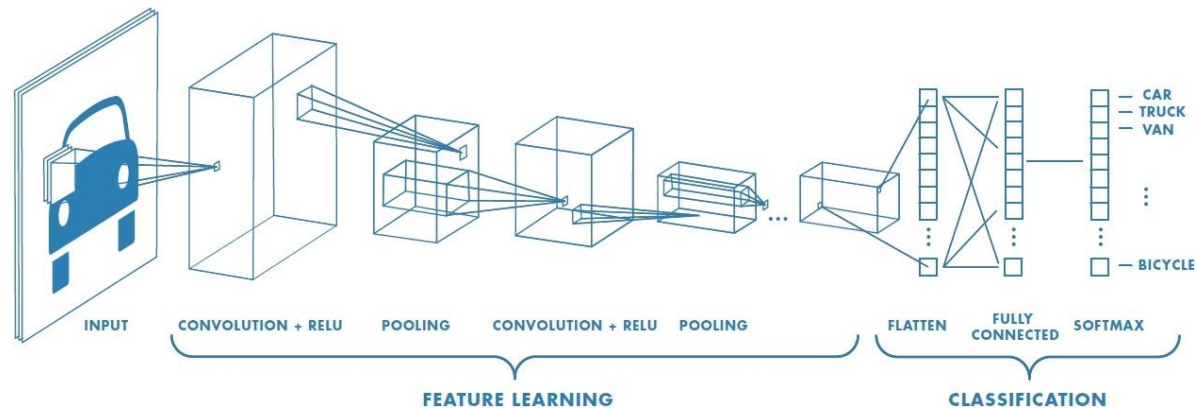


Fig. 17

Deep Learning

Recurrent Neural Networks (RNN)

- Recurrent layers

- Model sequential data → model dynamic temporal behaviour
- Internal memory state(s) → memorize previous data for future predictions

- Vanishing gradient problem

- Gating mechanisms (Gated Recurrent Units (GRU), Long Short-term Memory (LSTM))

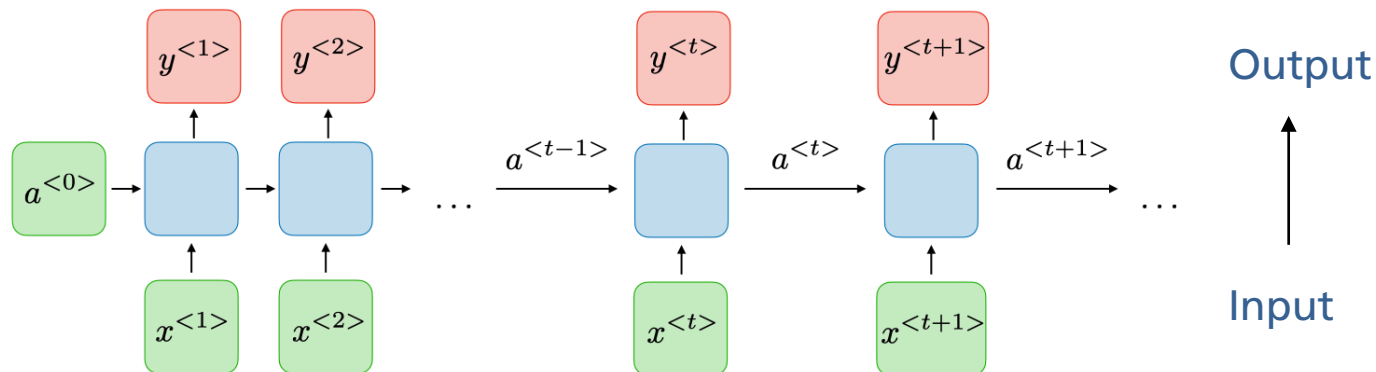


Fig. 18

Deep Learning

Recurrent Neural Networks (RNN)

■ Application Examples

- One-to-many: sequential music generation (given a starting note)
- Many-to-one: sentiment classification (positive vs. negative)
- Many-to-many: machine translation (e.g. Spanish to German)

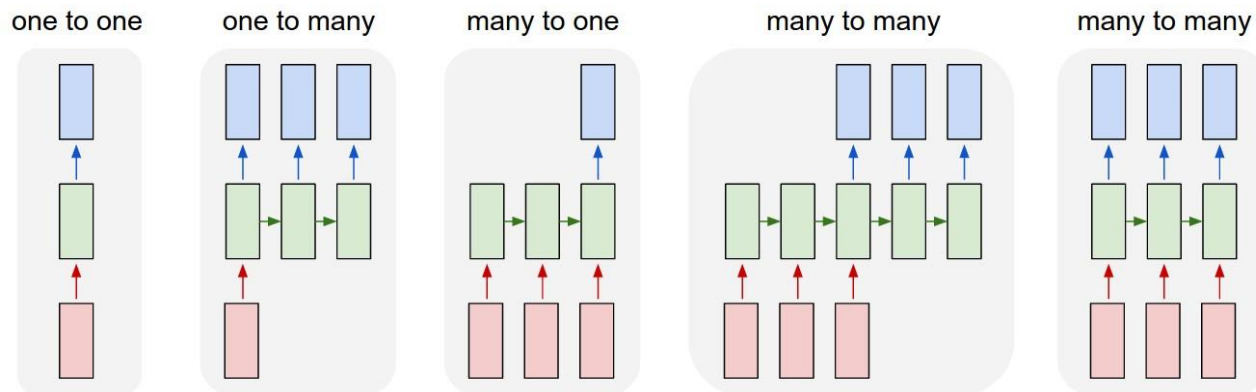
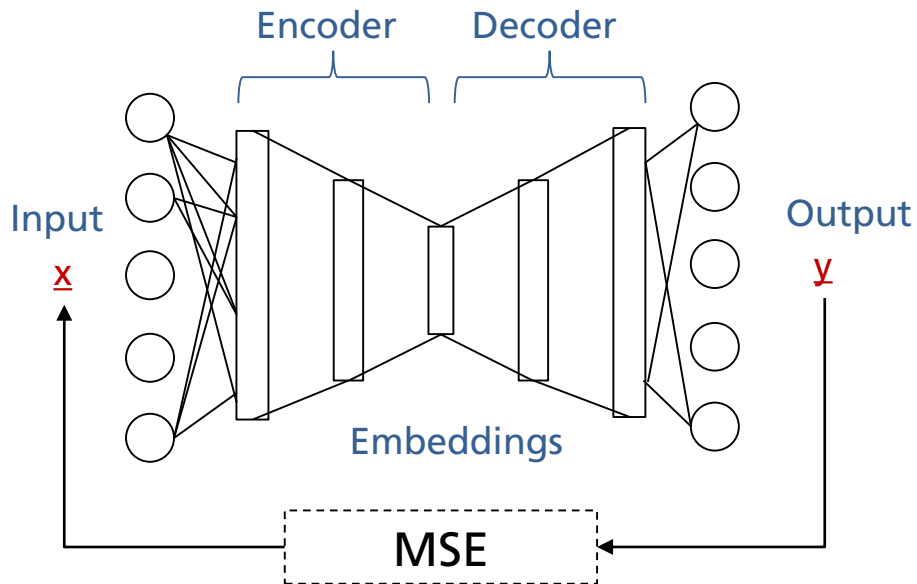


Fig. 19

Deep Learning

Autoencoders

- Symmetric architecture (decoder & encoder)
- Objective: minimize reconstruction error (e.g. mean squared error)
- Compression of input (embedding)
- Prioritize on important information → learn useful representations



Summary

- Introduction
 - Terminology, application scenarios
- Learning Paradigms
 - Unsupervised, supervised, self-supervised learning
- ML project pipeline
 - Data collection, pre-processing, split
 - Model selection, validation, training, testing
- Deep Learning
 - DNN, CNN, RNN, Autoencoders

References

Introducing Machine Learning. (2016). Retrieved from https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/i/88174_92991v00_machine_learning_section1_ebook.pdf

S. Legg, M. Hutter (2007). Universal Intelligence: A Definition of Machine Intelligence. *Minds & Machines*. 17 (4): 391-444.

A. L. Samuel (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*. 3(3), 210-229

Images

Fig. 1: [Machine Learning, 2016], p. 4, Fig. 2

Fig. 2: <https://i0.wp.com/www.sthda.com/sthda/RDoc/figure/clustering/partitioning-cluster-analysis-k-means-plot-4-groups-1.png>

Fig. 3: <https://i.stack.imgur.com/hsilO.png> (https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)

Fig. 4: https://miro.medium.com/max/975/1*OyYyr9qY-w8RkaRh2TKo0w.png

Fig. 5: <https://lilianweng.github.io/lil-log/assets/images/self-sup-lecun.png>

Fig. 6: <https://www.asimovinstitute.org/wp-content/uploads/2019/04/NeuralNetworkZoo20042019.png>

Fig. 7: <https://www.educative.io/api/edpresso/shot/6668977167138816/image/5033807687188480>

Fig. 8: https://miro.medium.com/max/534/1*Z54JgbS4DUwWSknhDCvNTQ.png

Fig. 9: https://miro.medium.com/max/915/1*SJPacPhP4KDEB1AdhOFy_Q.png

Fig. 10: https://www.skampakis.com/wp-content/uploads/2018/03/simple_neural_network_vs_deep_learning.jpg

Fig. 11: https://pic4.zhimg.com/80/v2-057b248288a8af2f01272a956f862873_1440w.png

Fig. 12: https://blog.e-kursy.it/deeplearning4j-workshop/video/html/presentation_specific/img/4_activation_functions.png

Images

Fig. 13: <https://blog.paperspace.com/content/images/2018/05/challenges-1.png>

Fig. 14: <https://www.cs.umd.edu/~tomg/img/landscapes/noshort.png>

Fig. 15: <https://blog.paperspace.com/content/images/2018/05/grad.png>

Fig. 16: <https://www.wandb.com/articles/intro-to-cnns-with-wandb>

Fig. 17: <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>

Fig. 18: <https://wiki.tum.de/download/attachments/22578349/RNN1.png>

Fig. 19: <https://stanford.edu/~shervine/teaching/cs-230/illustrations/architecture-rnn-ltr.png>

Thank you!

■ Any questions?

Dr.-Ing. Jakob Abeßer

Fraunhofer IDMT

Jakob.abesser@idmt.fraunhofer.de

<https://machinelisting.github.io>