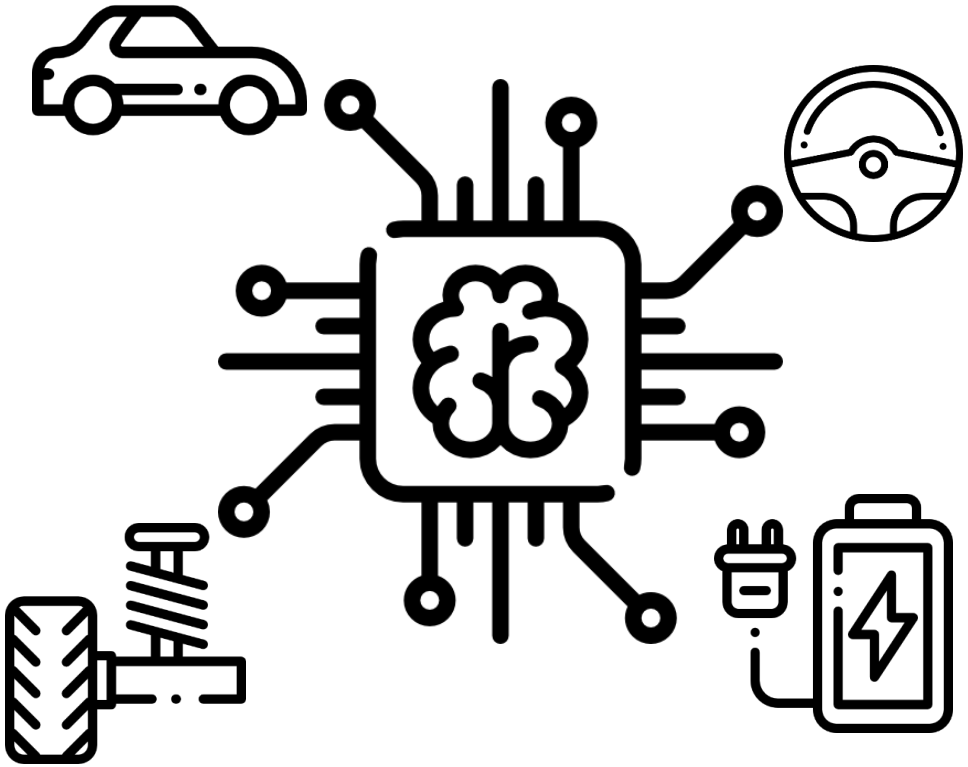
Institute of Automotive Technology
TUM School of Engineering and Design
Technical University of Munich

Artificial Intelligence in Automotive Technology

Maximilian Geißlinger / Fabian Netzler

Prof. Dr.-Ing. Markus Lienkamp



Lecture Overview

Lecture 16:15-17:45 | Practice 17:45-18:30	
1 Introduction: Artificial Intelligence	20.10.2022 – Maximilian Geißlinger
2 Perception	27.10.2022 – Sebastian Huber
3 Supervised Learning: Regression	03.11.2022 – Fabian Netzler
4 Supervised Learning: Classification	10.11.2022 – Andreas Schimpe
5 Unsupervised Learning: Clustering	17.11.2022 – Andreas Schimpe
6 Introduction: Artificial Neural Networks	24.11.2022 – Lennart Adenaw
7 Deep Neural Networks	08.12.2022 – Domagoj Majstorovic
8 Convolutional Neural Networks	15.12.2022 – Domagoj Majstorovic
9 Knowledge Graphs	12.01.2023 – Fabian Netzler
10 Recurrent Neural Networks	19.01.2023 – Matthias Rowold
11 Reinforcement Learning	26.01.2023 – Levent Ögretmen
12 AI-Development	02.02.2023 – Maximilian Geißlinger
13 Guest Lecture	09.02.2023 – driveblocks

Announcement: Guest Lecture driveblocks

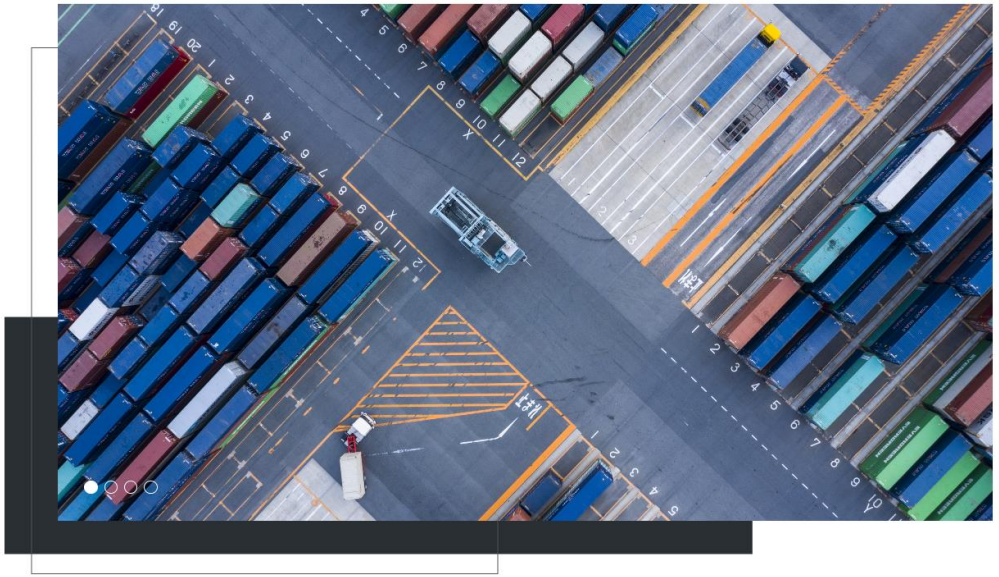
CHIEF TECHNICAL OFFICER

Alexander Wischnewski

Alexander is responsible for the product strategy of the driveblocks autonomy platform and its overall design and architecture. Prior to driveblocks, he was the team leader and system architect of the winning Indy Autonomous Challenge team of the Technical University of Munich. In addition, he designed vehicle motion control systems during his time as a research assistant at the Chair of Automatic Control and has a strong background in software engineering and robotic systems implementation.

Autonomous driving start-up with founders from TUM

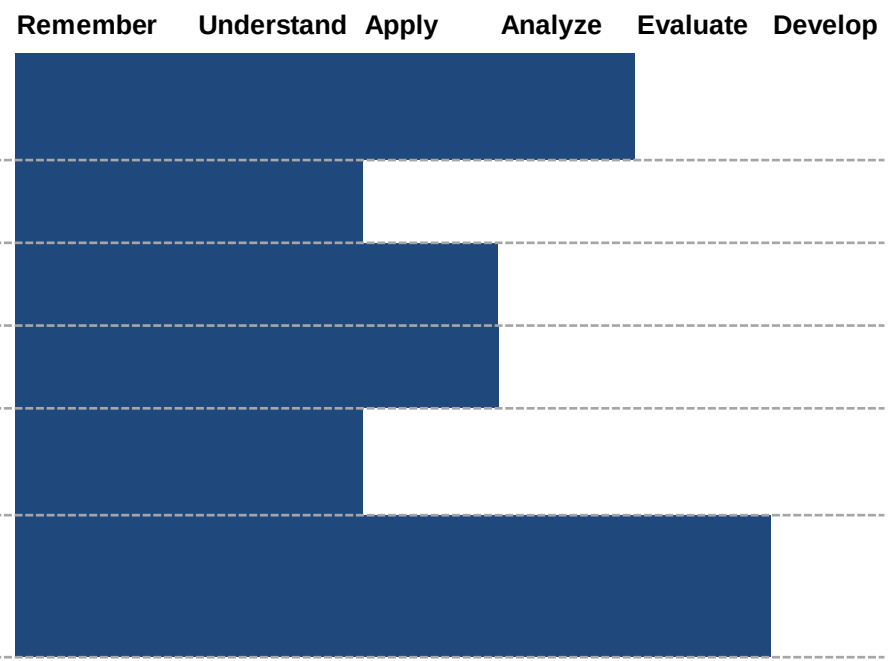
https://driveblocks.ai/



Objectives for Lecture 12: AI Development

Depth of understanding

After the lecture you are able to…



… analyze ML problems and choose according model types and training methods

… understand the impact of data on ML problems

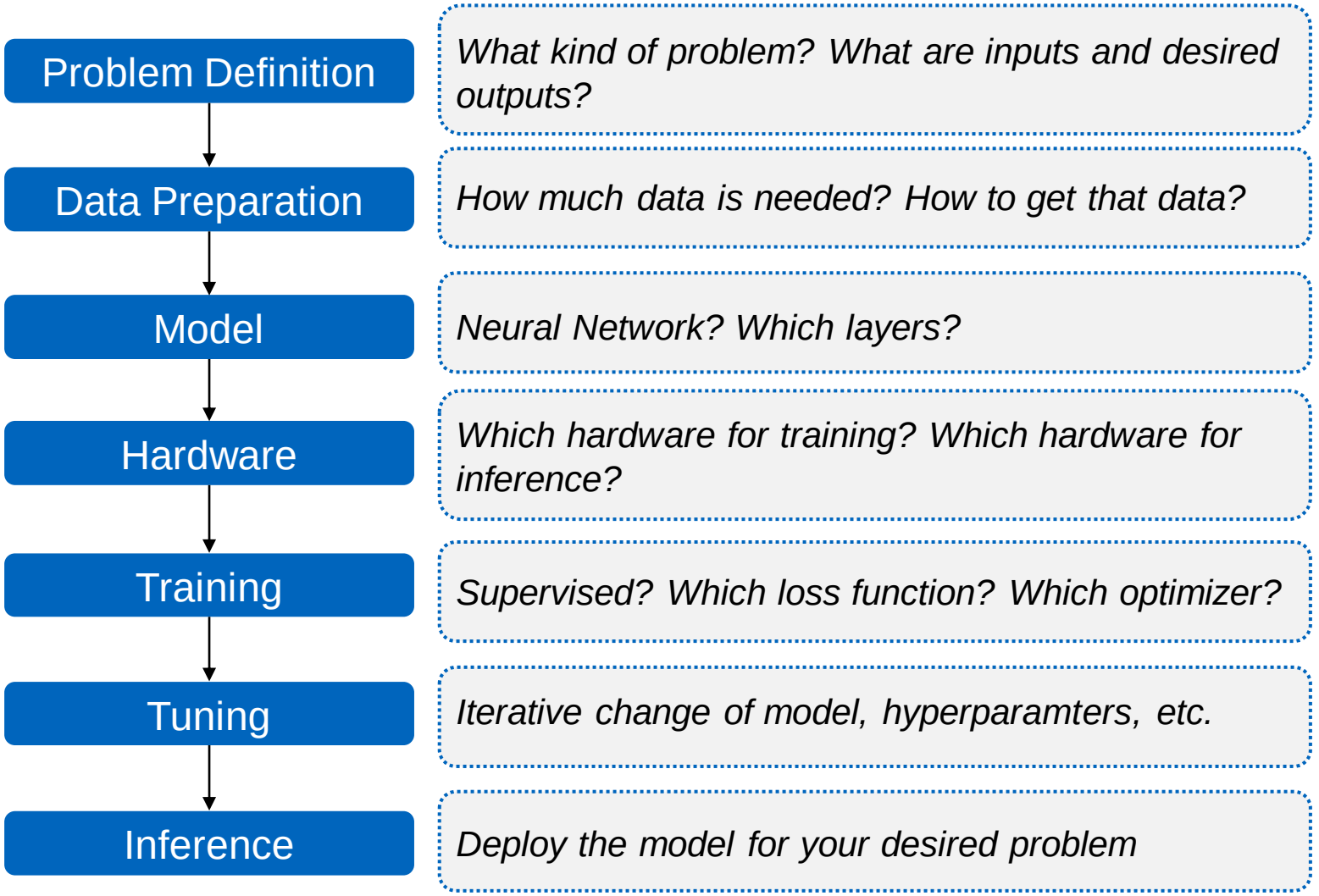
… apply data augmentation to a given set of data

... apply methods to avoid overfitting of DL models

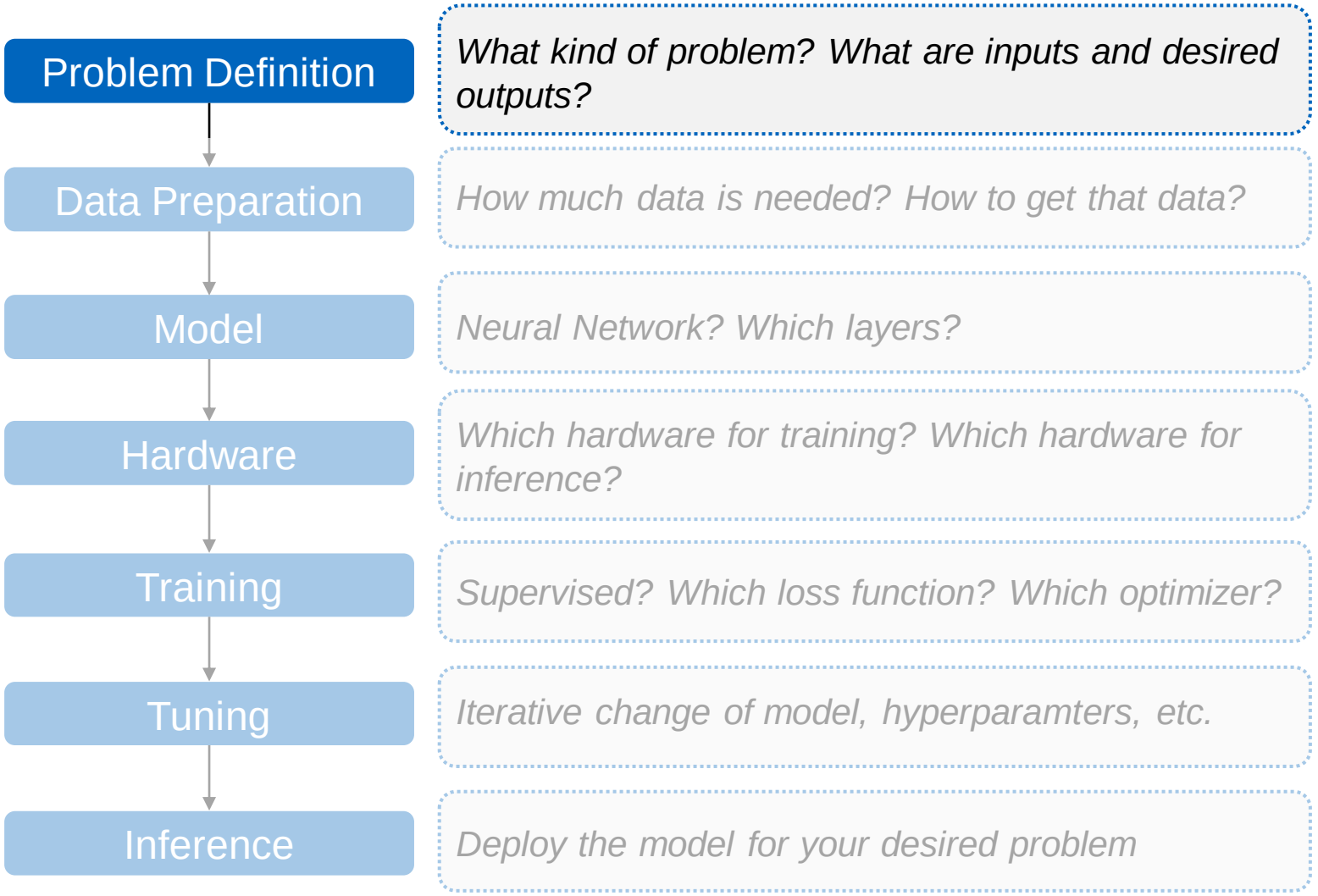
… understand different computing hardware types for training and inference

… analyze results from the training of a DL algorithm and evaluate the hyperparameter regarding the performance of the algorithm

ML Development Workflow



ML Development Workflow



ML Development: Problem Definition

Problem Definition

What kind of problem? What are inputs and desired outputs?

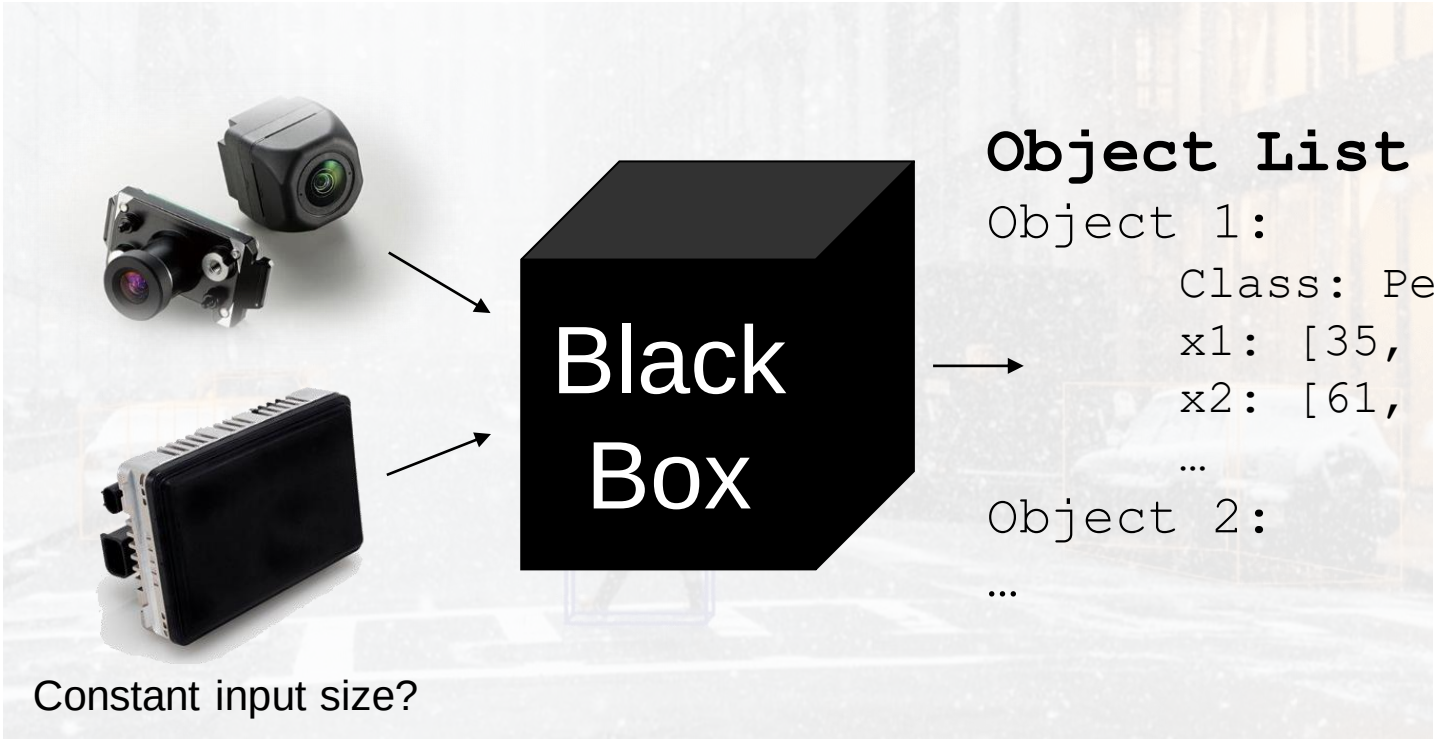
Example: Robust object detection for autonomous driving

ML Development: Problem Definition

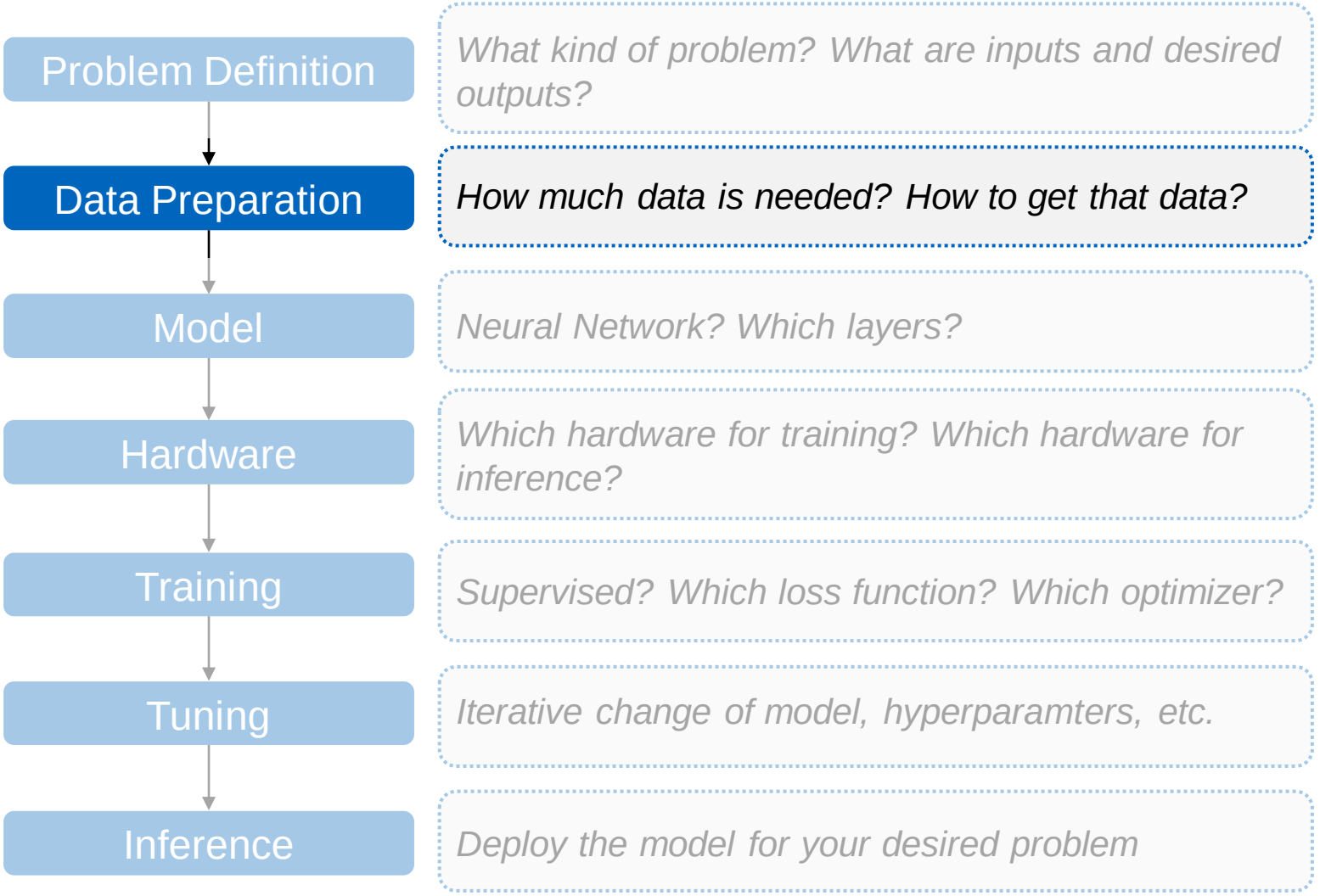
Problem Definition

What kind of problem? What are inputs and desired outputs?

Example: Robust object detection for autonomous driving



ML Development Workflow



AI Development Workflow

Data Preparation

How much data is needed? How to get that data?

- Public data sets

  - How well does this fit your problem definition? (Inputs, Outputs, …)
  - Can data be processed to fit your problem? (e.g. 3D Bounding Boxes → 2D Bounding Boxes)

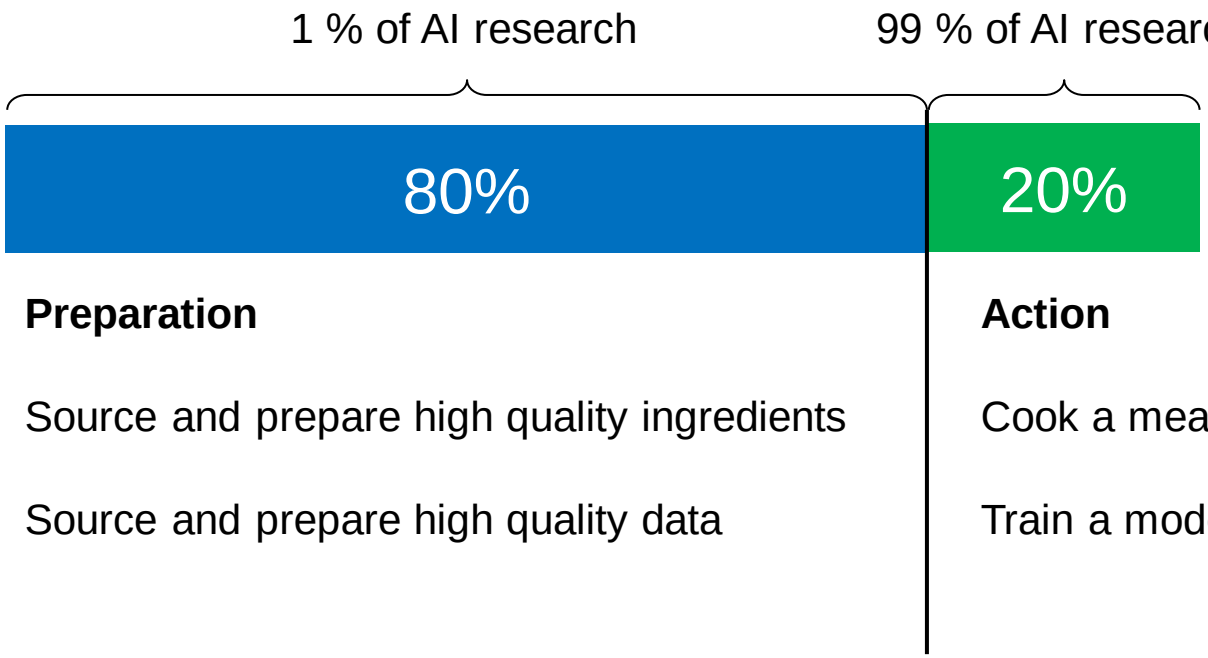
- Create your own data set
  - Prototype needed?
  - Data infrastructure (How to save data? How to handle large files?)
  - Time consuming and expensive

Improving the Code vs. Improving the Data

	Steel defect detection	Solar panel	Surface inspection
Baseline	76.2%	75.68%	85.05%
Model-centric	+0% (76.2%)	+0.04% (75.72%)	+0.00% (85.05%)
Data-centric	+16.9% (93.1%)	+3.06% (78.74%)	+0.4% (85.45%)

Source: https://www.youtube.com/watch?v=06-AZXmwHjo&t=1428s

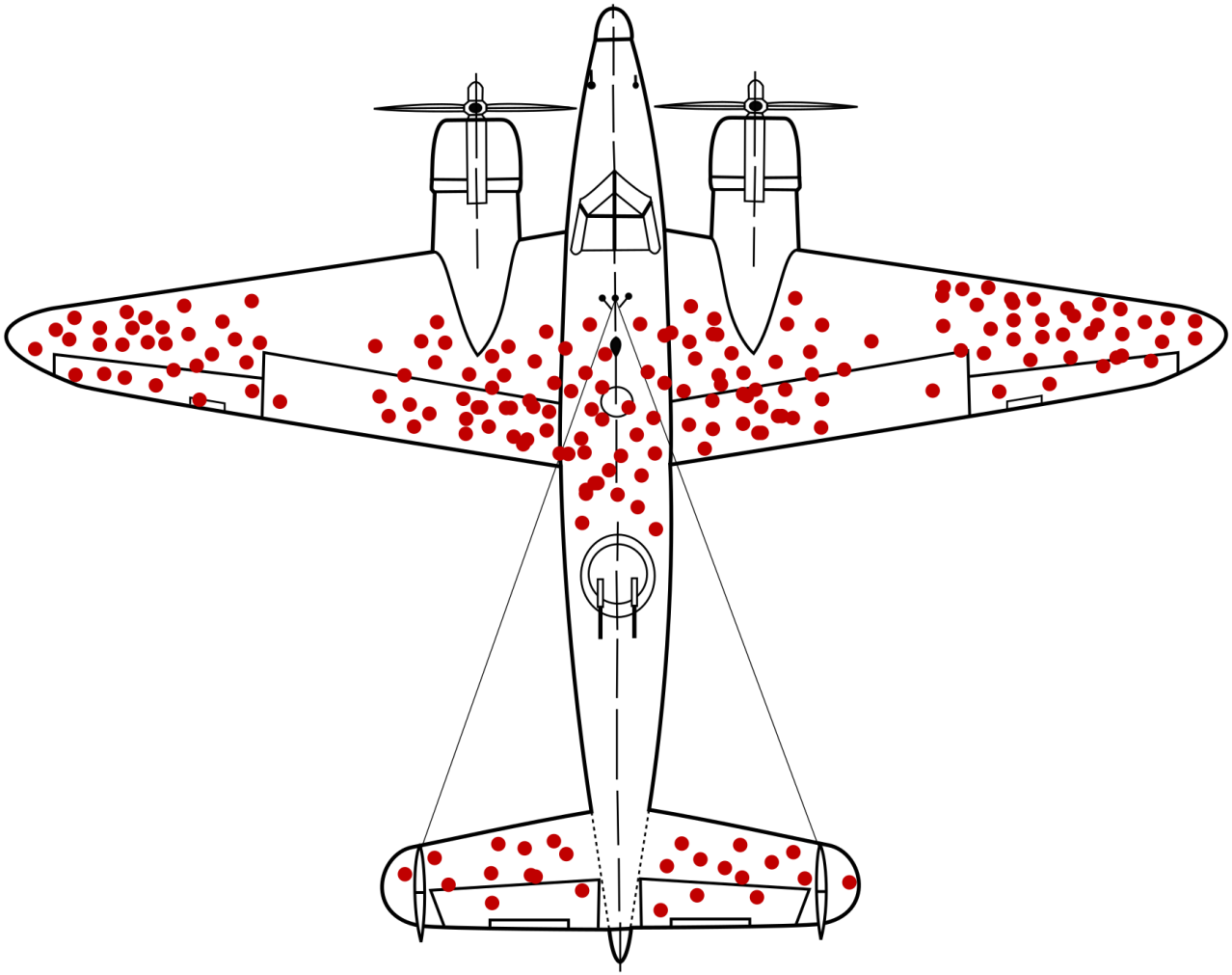
Data is Food for AI



	Preparation	**Action**
e.g. Cooking	Source and prepare high quality ingredients	Cook a meal
e.g. ML	Source and prepare high quality data	Train a model

Source: https://www.youtube.com/watch?v=06-AZXmwHjo&t=1428s

Data Recording: Survivorship Bias



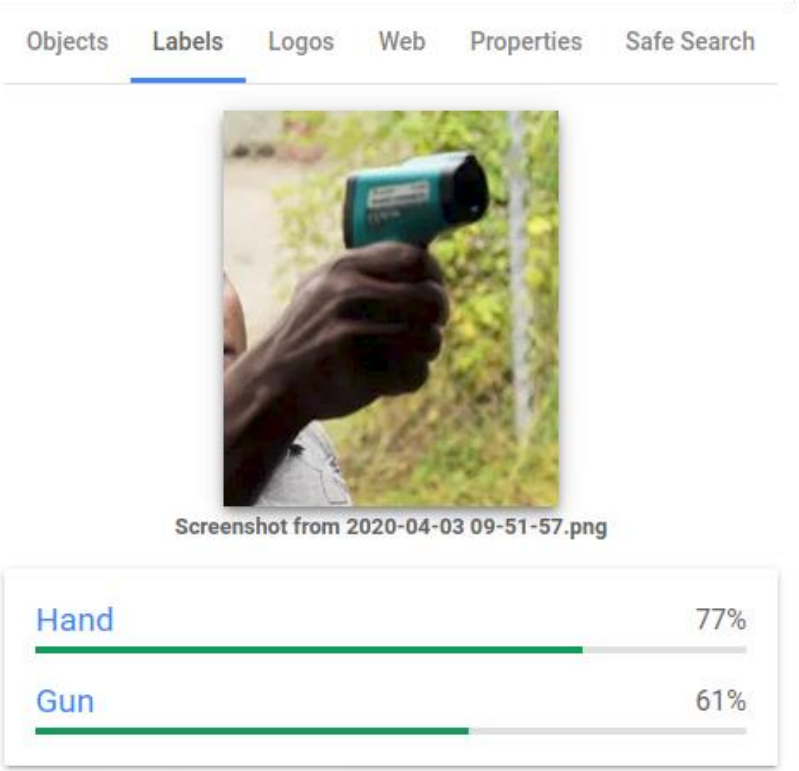
Data Recording: Sample Bias

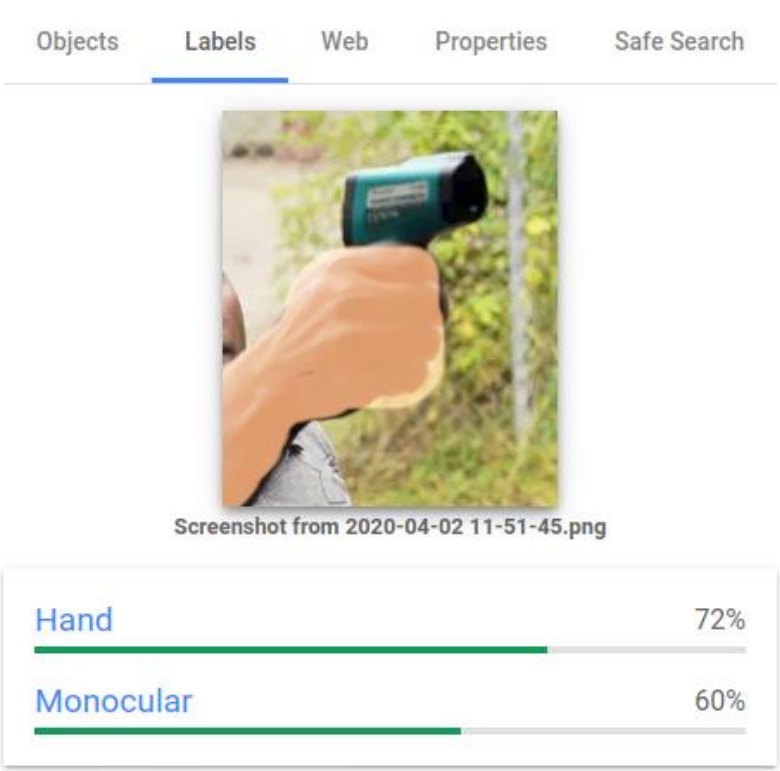
- Example: Detection of Tanks

- Bias Detection: Open Field of research

Source: The Neural Net Tank Urban Legend: https://www.gwern.net/Tanks#could-it-happen

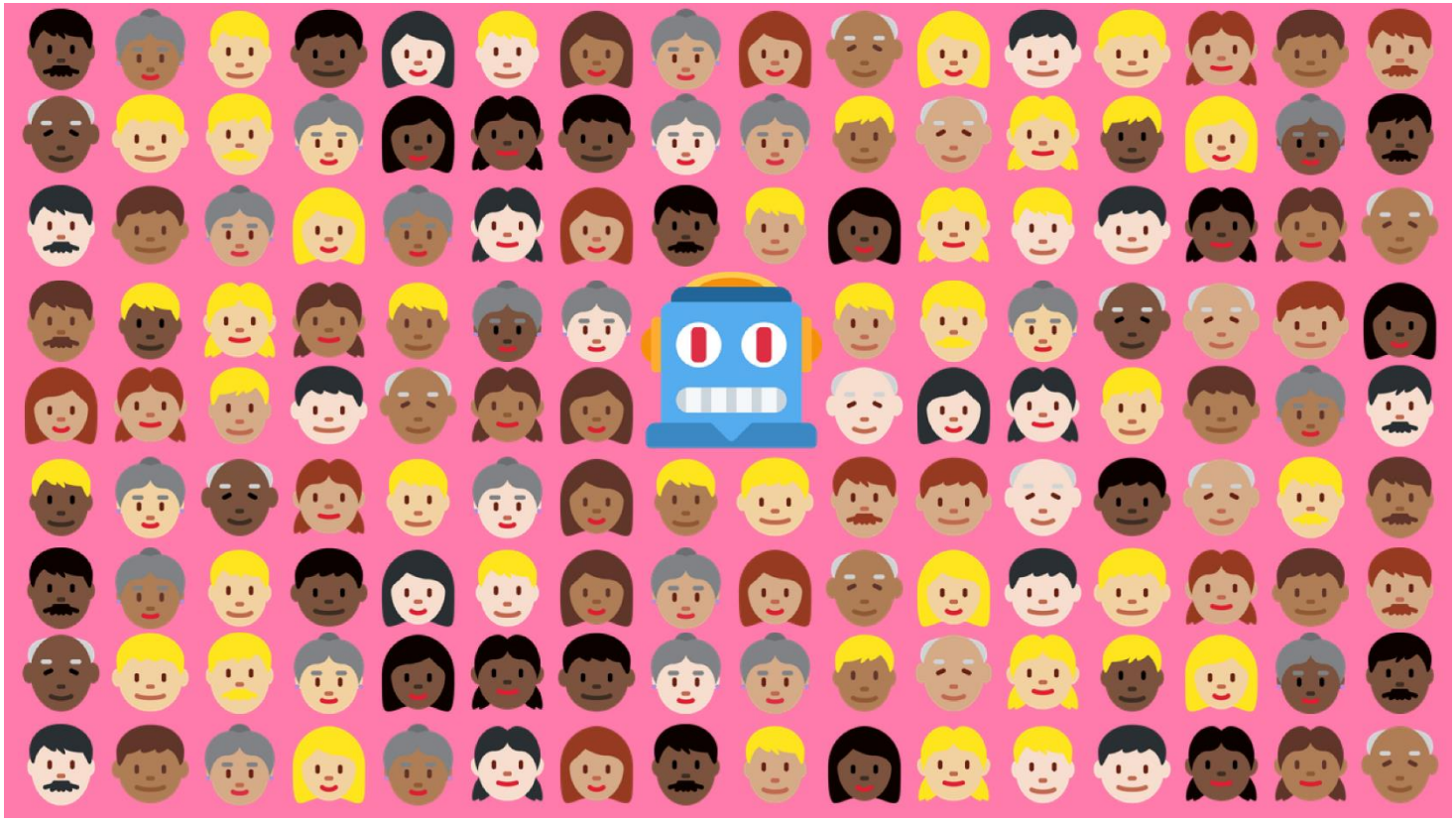
Data Recording: Bias and Ethical Issues





Source: Google Labels

Data Recording: Bias and Ethical Issues



Class Imbalance

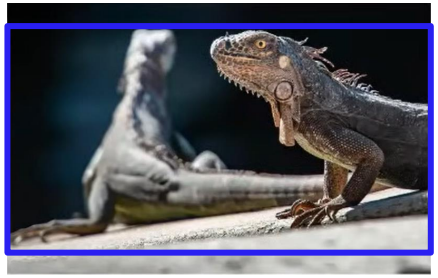
vehicle.bus.rigid	14,501
vehicle.car	493,322
vehicle.construction	14,671
vehicle.emergency.ambulance	49
vehicle.emergency.police	638
vehicle.motorcycle	12,617

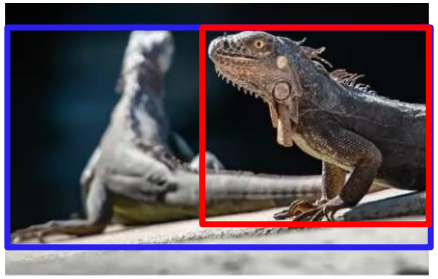
Source: nuScenes

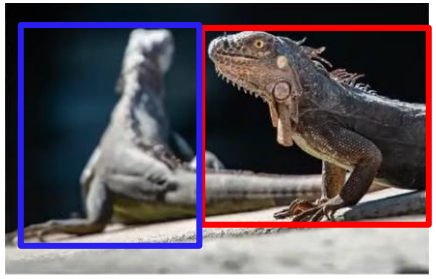
→ Repeat samples with under-represented data
→ Consider imbalance in loss function

Data Annotation

Task: Use Bounding boxes to indicate the position of iguanas







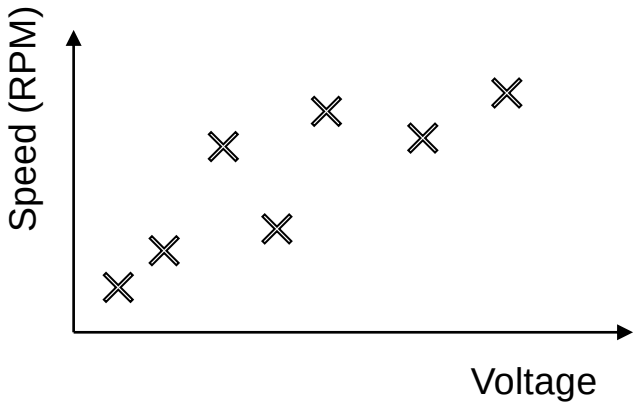
Data Annotation: Consistency

- Ask different people to label the same image with the same annotator instruction
- Measure for consistency:

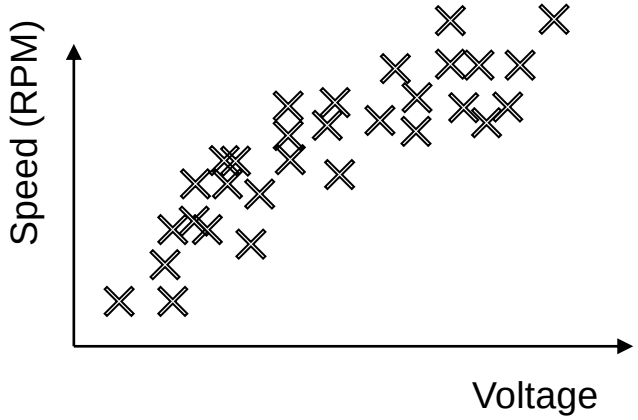
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

- Inconsistent labels can be considered as noise

What about Noise?

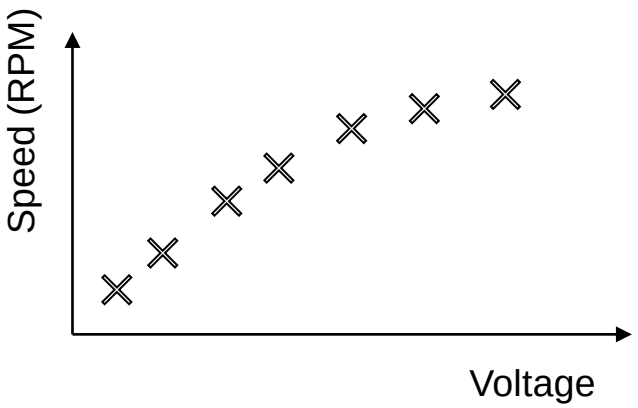


- Small data
- Noisy labels



- Big data
- Noisy labels

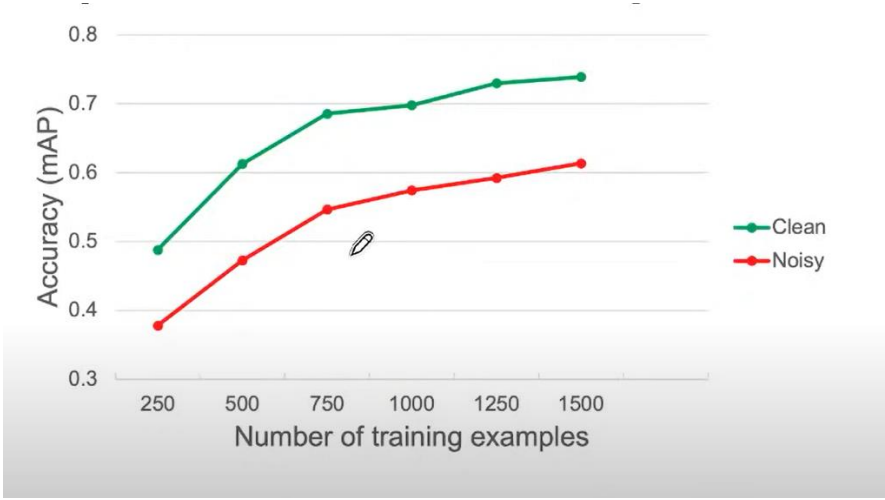
Clean vs. Noisy Data



- Small data
- Clean (consistent) labels

Example
You have 500 samples and 12% are noisy (incorrectly/inconsistently labeled)

The following are about equally effective:
- Clean up the noise
- Collect another 500 new samples (double the training set)



Data – More Data With Data Augmentation

1. **Flip** images

2. **Rotate** images

3. **Scale** images outward or inward

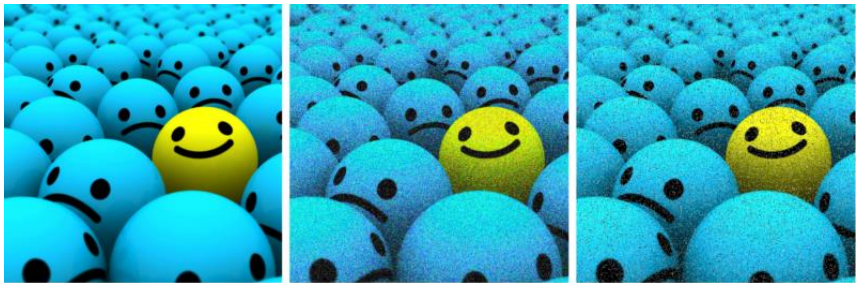
Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – More Data With Data Augmentation

4. **Crop** images

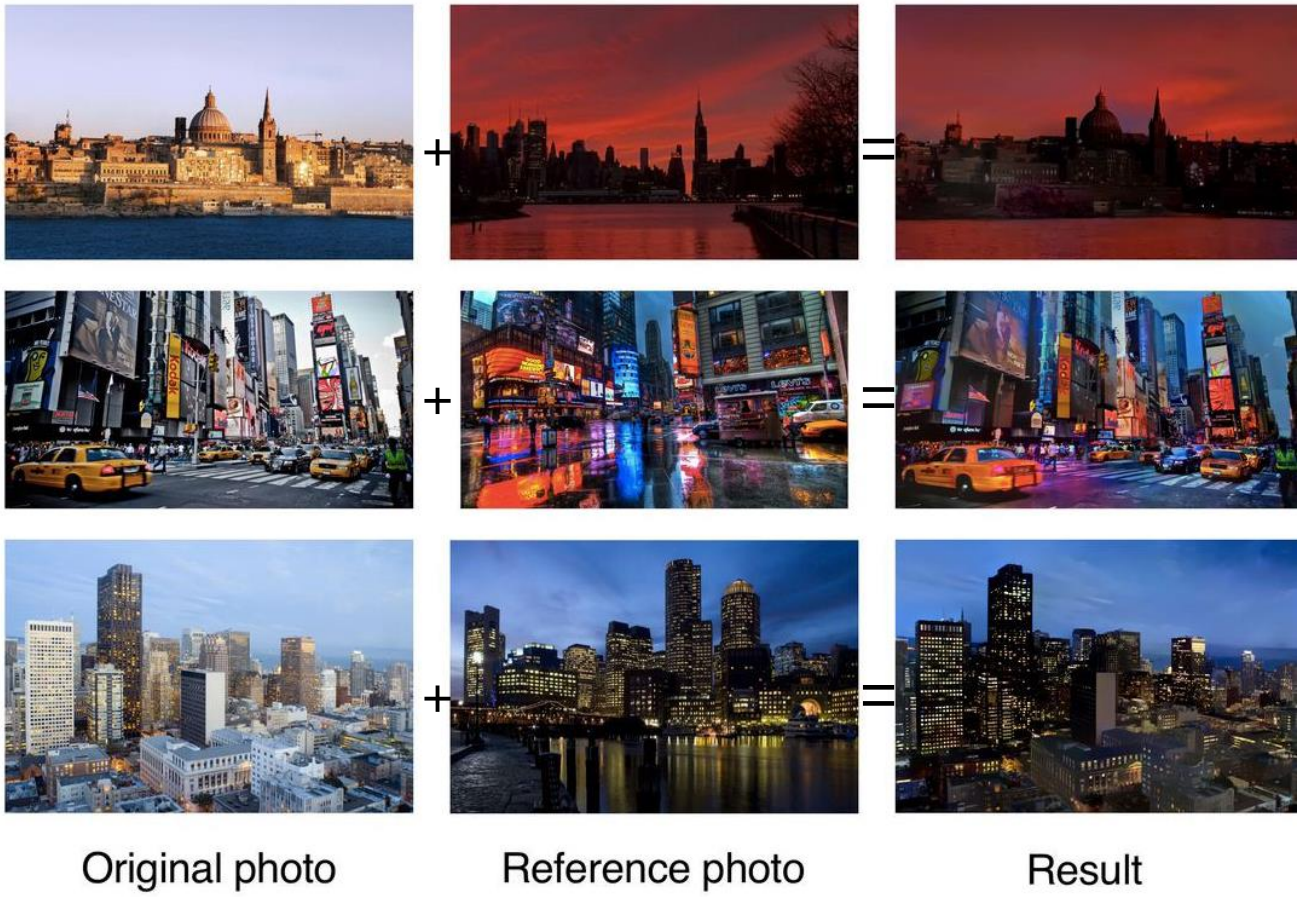
5. **Translation** of objects in x,y-position

6. Add **Gaussian Noise**



Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

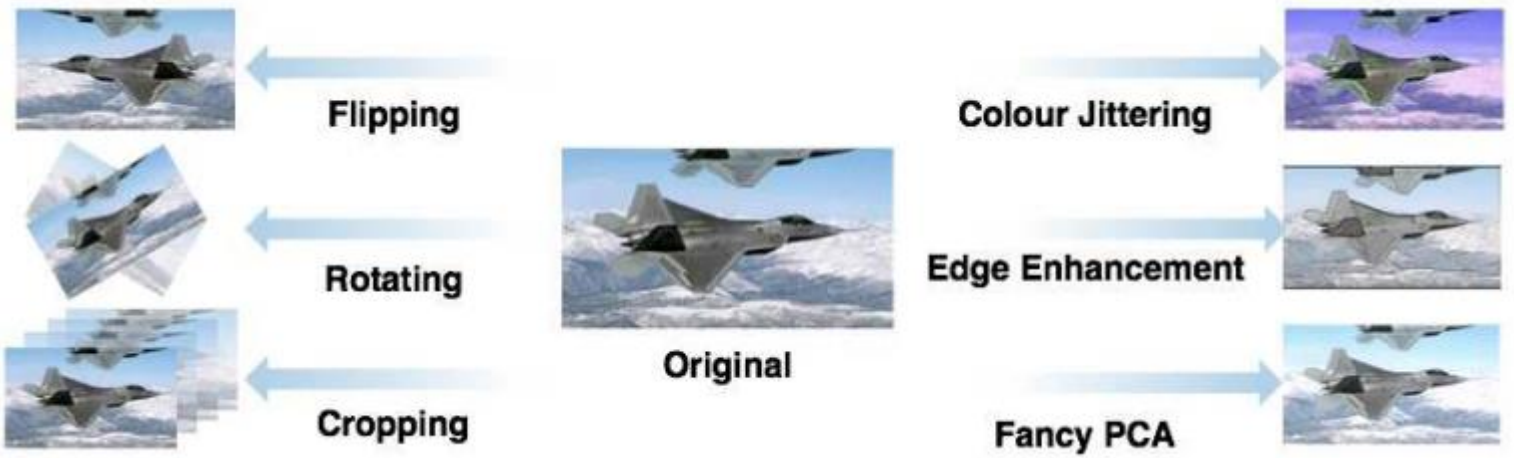
Data – More Data With Data Augmentation



7. **Deep photo style transfer:** Transform an image from one domain to an image of another domain using deep learing

Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – Improvements With Data Augmentation



	Top-1 Accuracy	Top-5 Accuracy
Baseline	48.13 ± 0.42%	64.50 ± 0.65%
Flipping	49.73 ± 1.13%	67.36 ± 1.38%
Rotating	50.80 ± 0.63%	69.41 ± 0.48%
Cropping	**61.95 ± 1.01%**	**79.10 ± 0.80%**
Color Jittering	**49.57 ± 0.53%**	67.18 ± 0.42%
Edge Enhancement	49.29 ± 1.16%	66.49 ± 0.84%
Fancy PCA	49.41 ± 0.84%	**67.54 ± 1.01%**

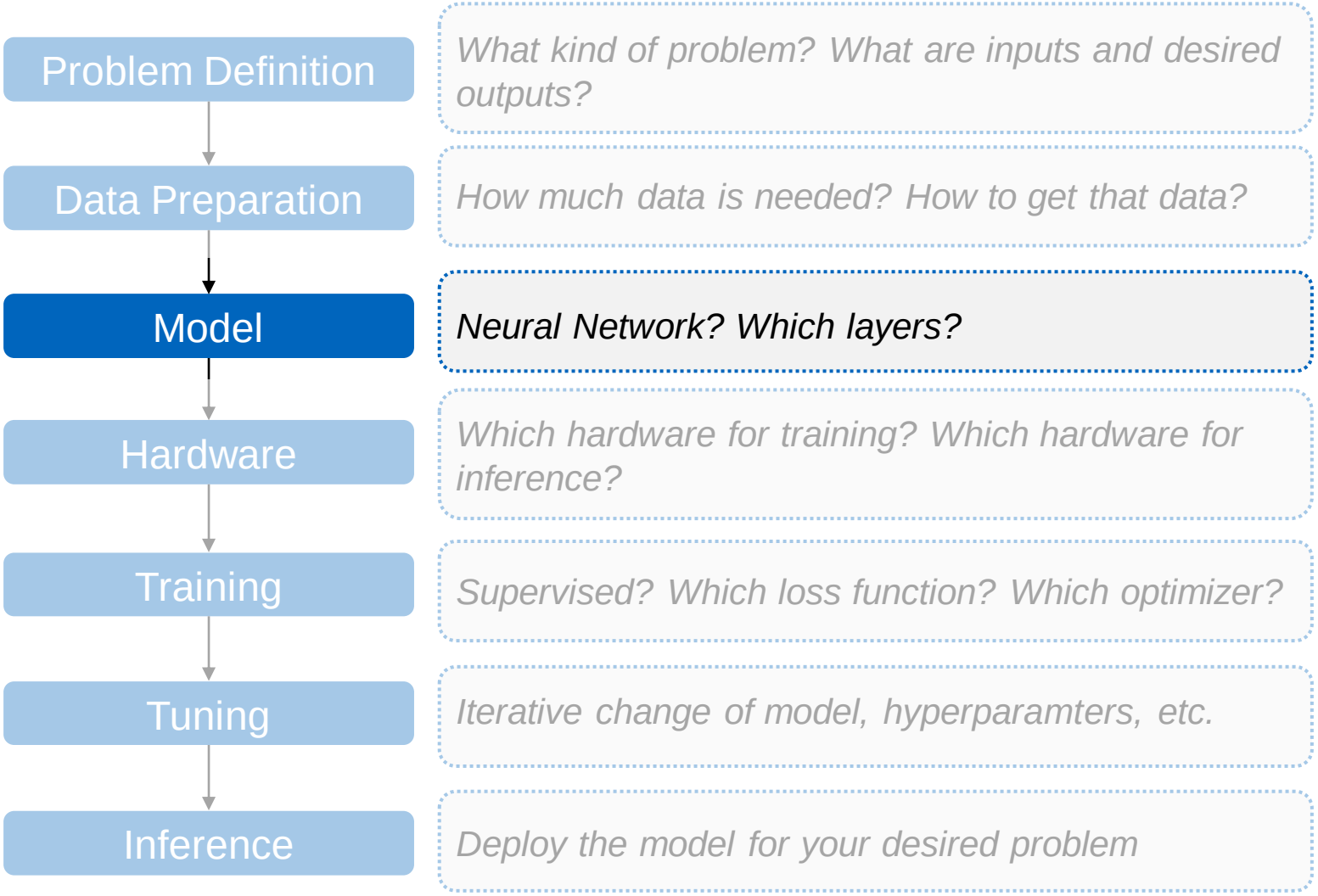
Source: https://arxiv.org/pdf/1708.06020.pdf

Different Types of Bias in Machine Learning

- **Sample Bias**: Samples do not reflect the environment well enough
- **Exclusion Bias**: (Systematic) exclusion of information
- **Measurement Bias**: Data measurement for training differs from inference
- **Recall Bias**: Label similar types of data inconsistantly
- **Observer Bias**: Effect of seeing what you expect to see or want to see in data

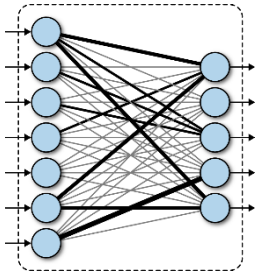
Source: https://www.telusinternational.com/articles/7-types-of-data-bias-in-machine-learning

ML Development Workflow



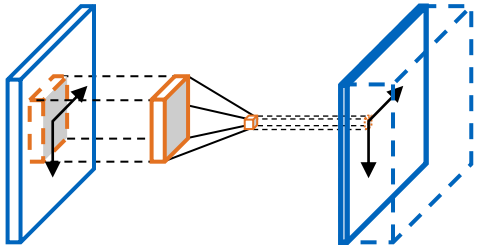
Model Architectures

Fully Connected Layer



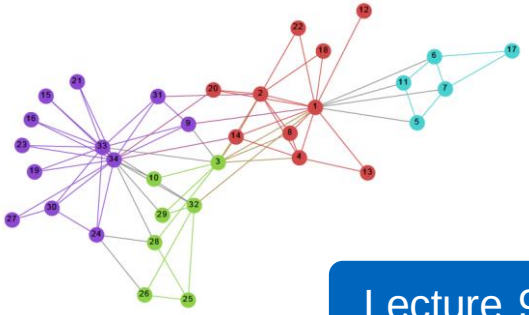
Lecture 7

Convolutional Neural Network



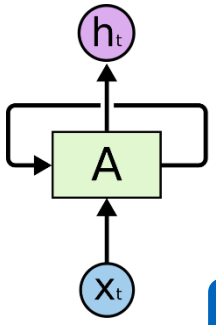
Lecture 8

Graph Neural Network



Lecture 9

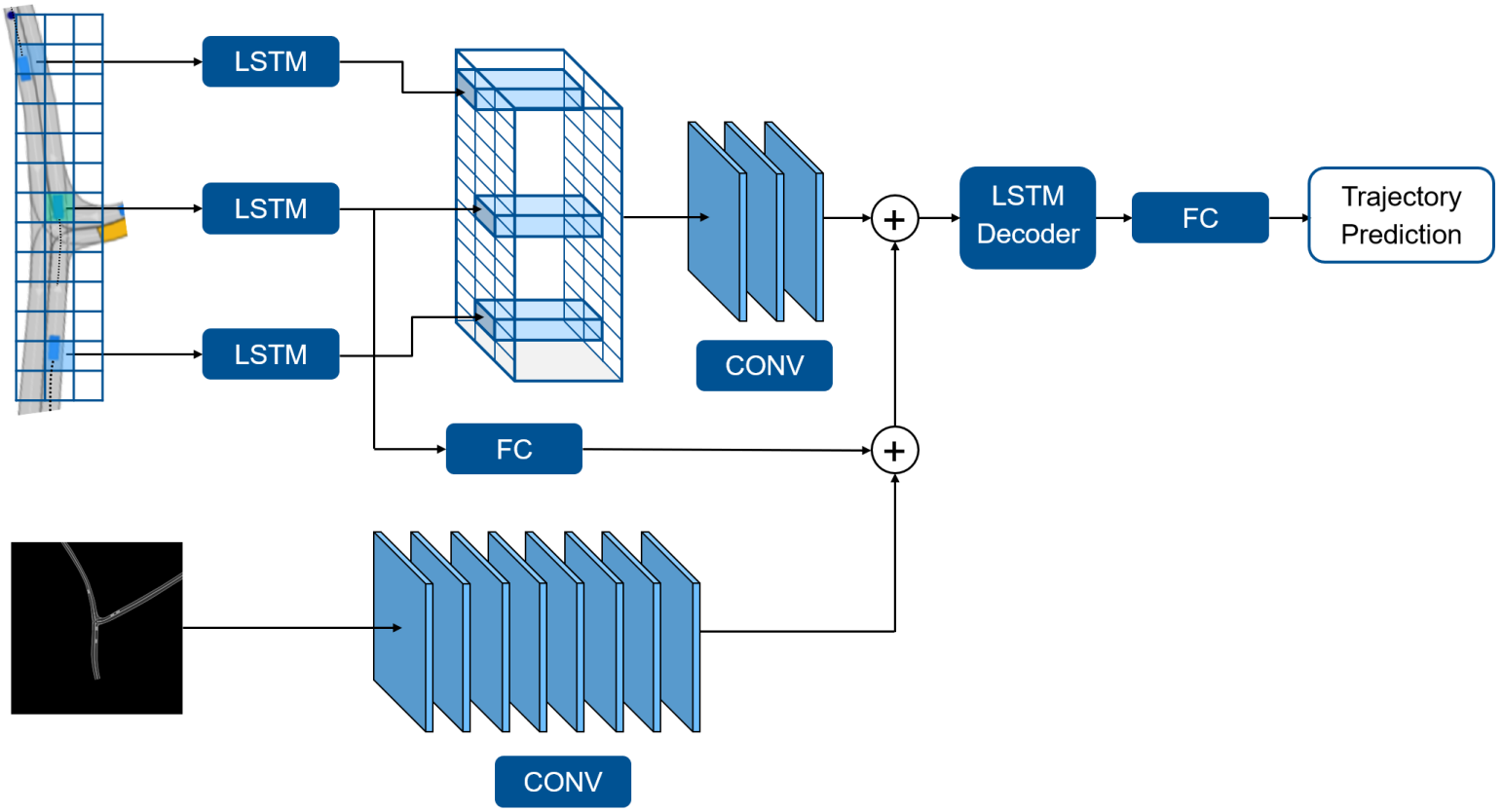
Recurrent Neural Network



Lecture 10

Model Architecture: Example

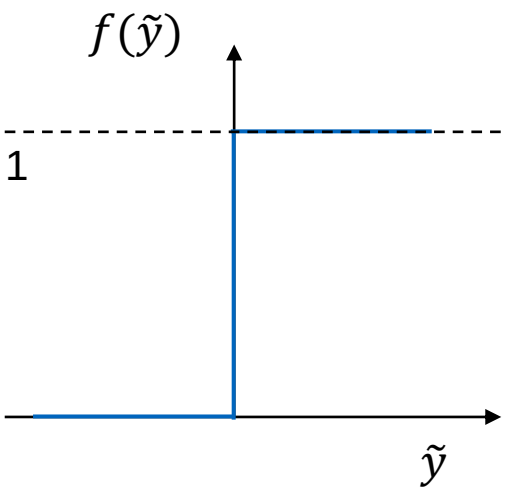
Vehicle Trajectory Prediction „Wale-Net“



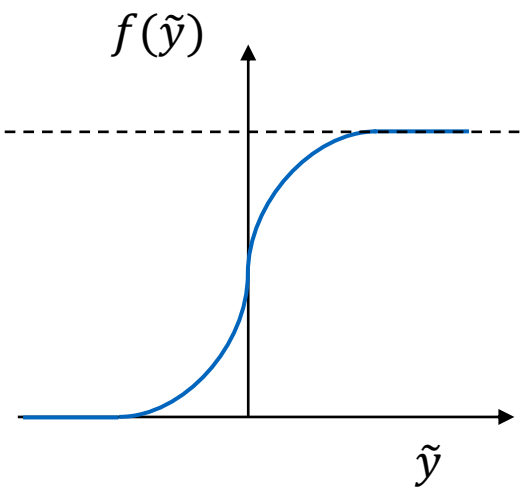
Source: https://github.com/TUMFTM/Wale-Net

Activation Functions

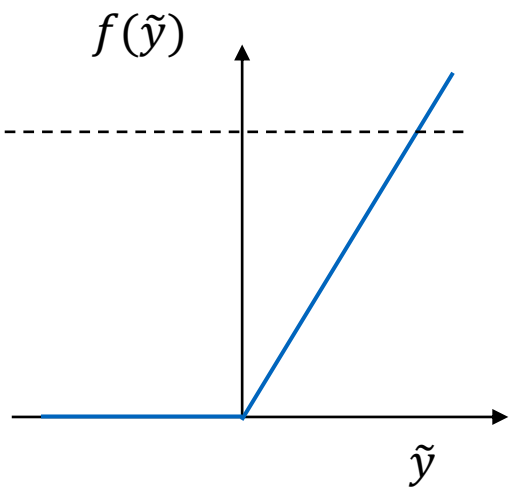
Step Function



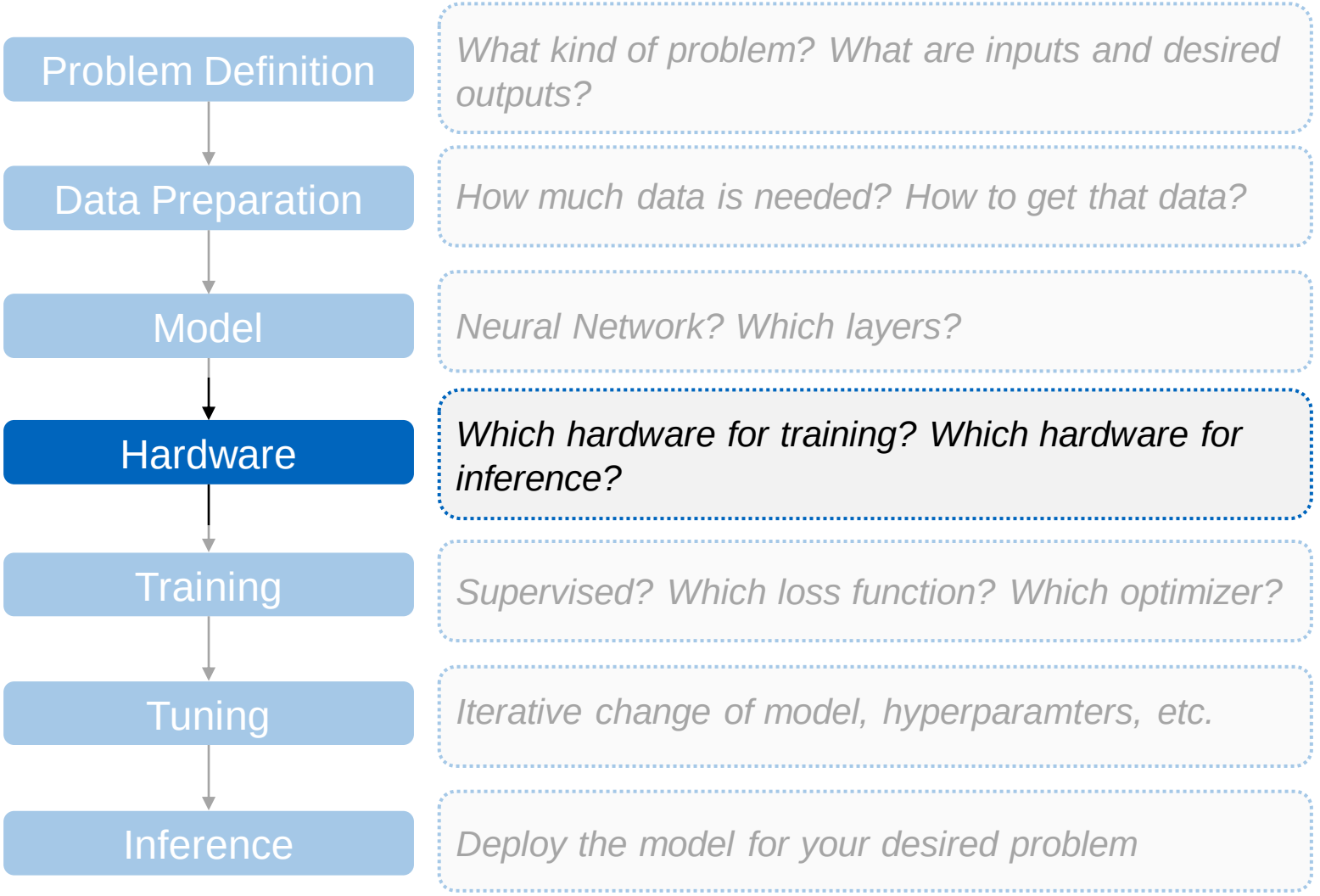
Sigmoid Function



ReLU Function



ML Development Workflow



Computing Hardware - Overview

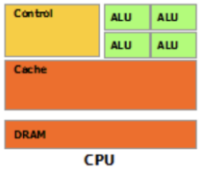
CPU

GPU

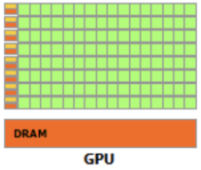
TPU

FPGA

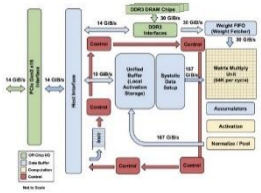
ASIC

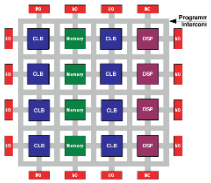


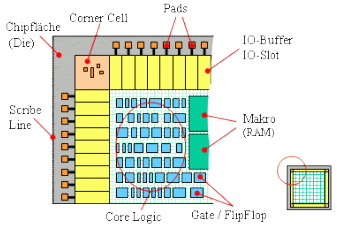
CPU



GPU







CPU, GPU & TPU

Central **P**rocessing **U**nit (CPU)

- Solve every computational problem in a general fashion
- Cache and memory design designed to be optimal for any general programming problem

Graphics **P**rocessing **U**nit (GPU)

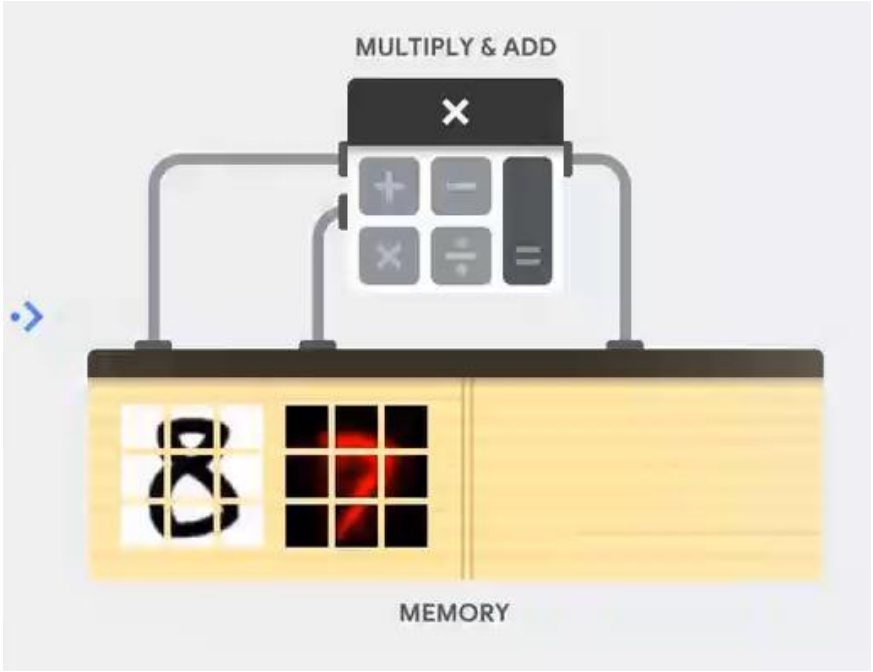
- Designed to accelerate the renderings of graphics
- Focus on parallelization

Tensor **P**rocessing **U**nit (TPU) (Google)

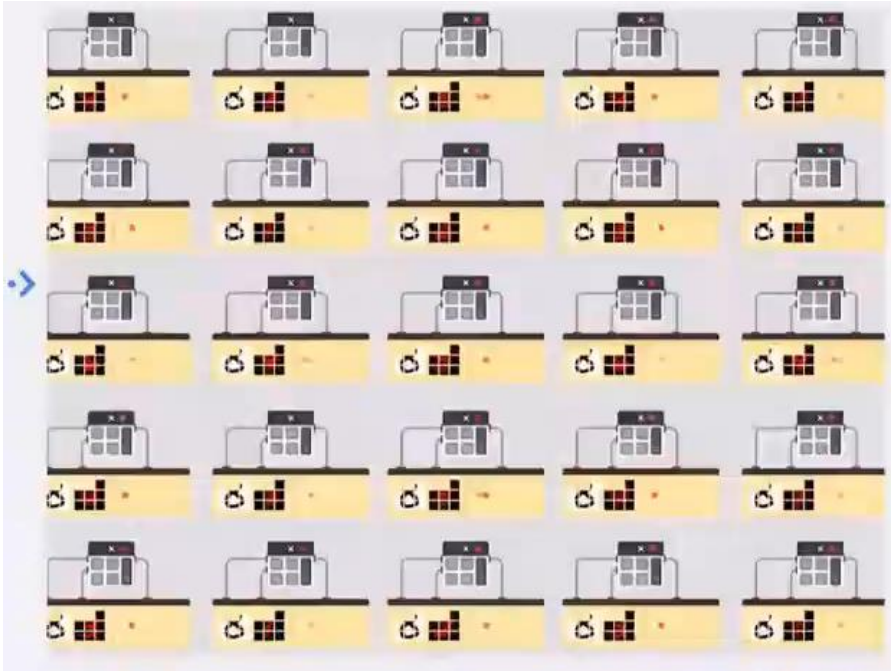
- Designed to accelerate deep learning tasks
- TPUs from Google especially for Tensorflow

Example: CNN with MNIST

CPU



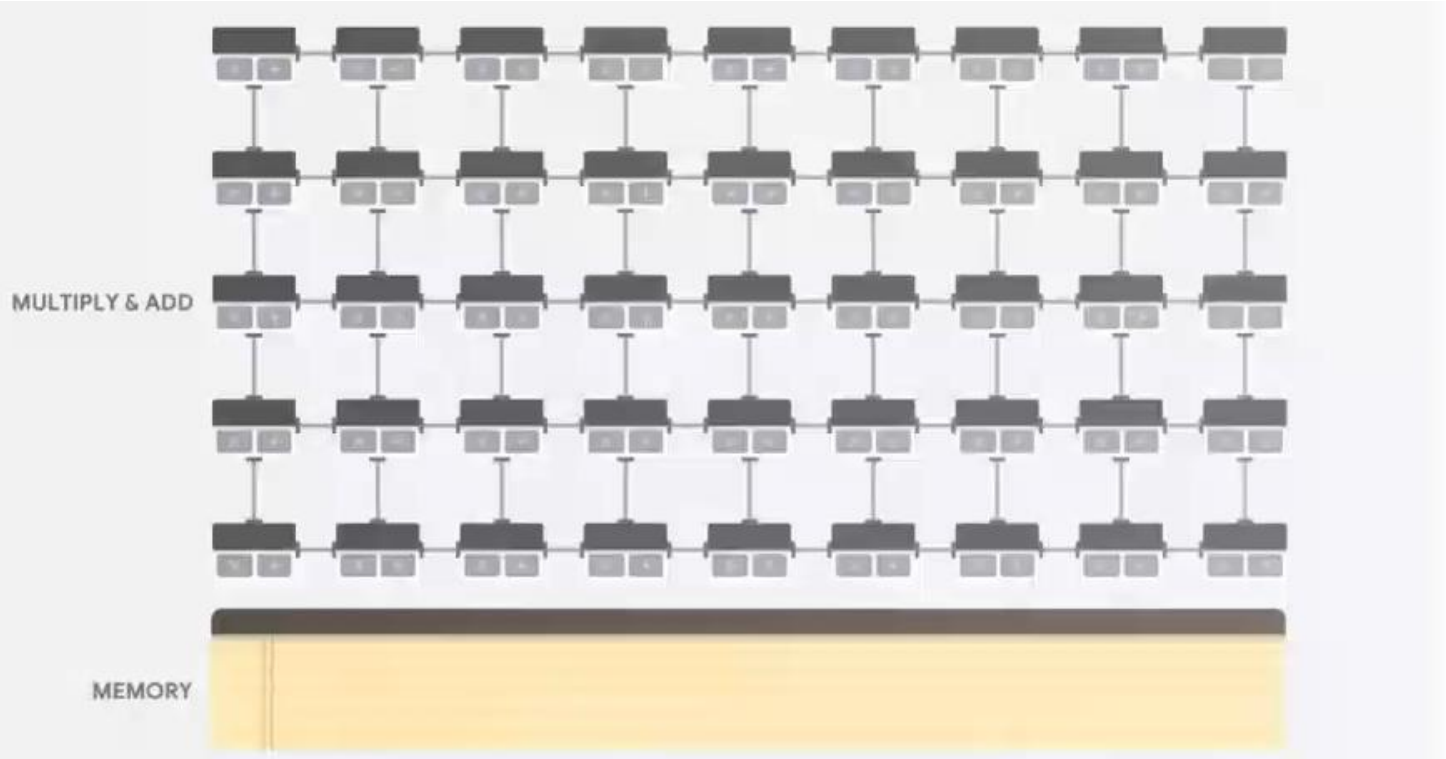
GPU



Source: https://www.youtube.com/watch?v=6ZDoFomU10A

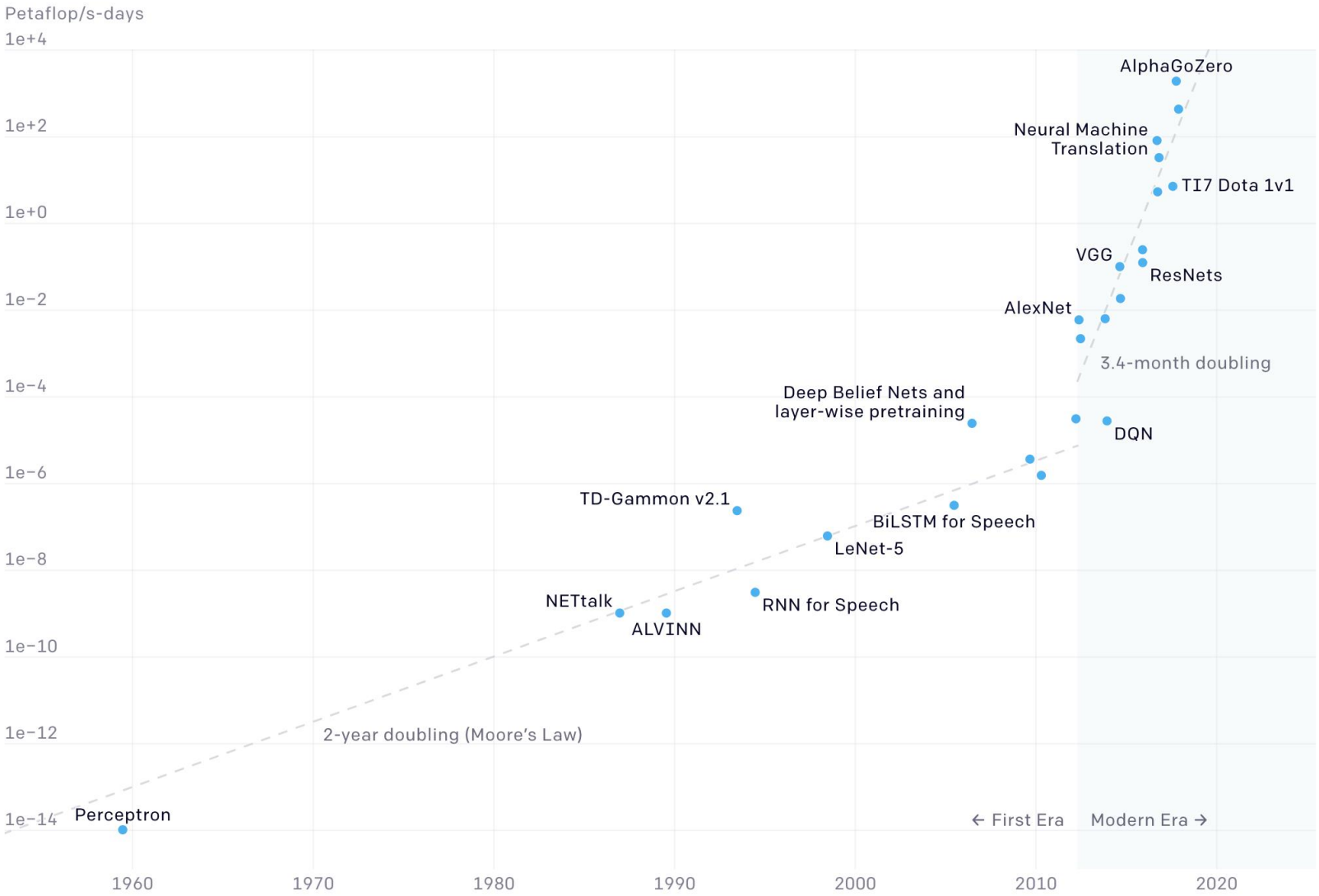
Example: CNN with MNIST

TPU

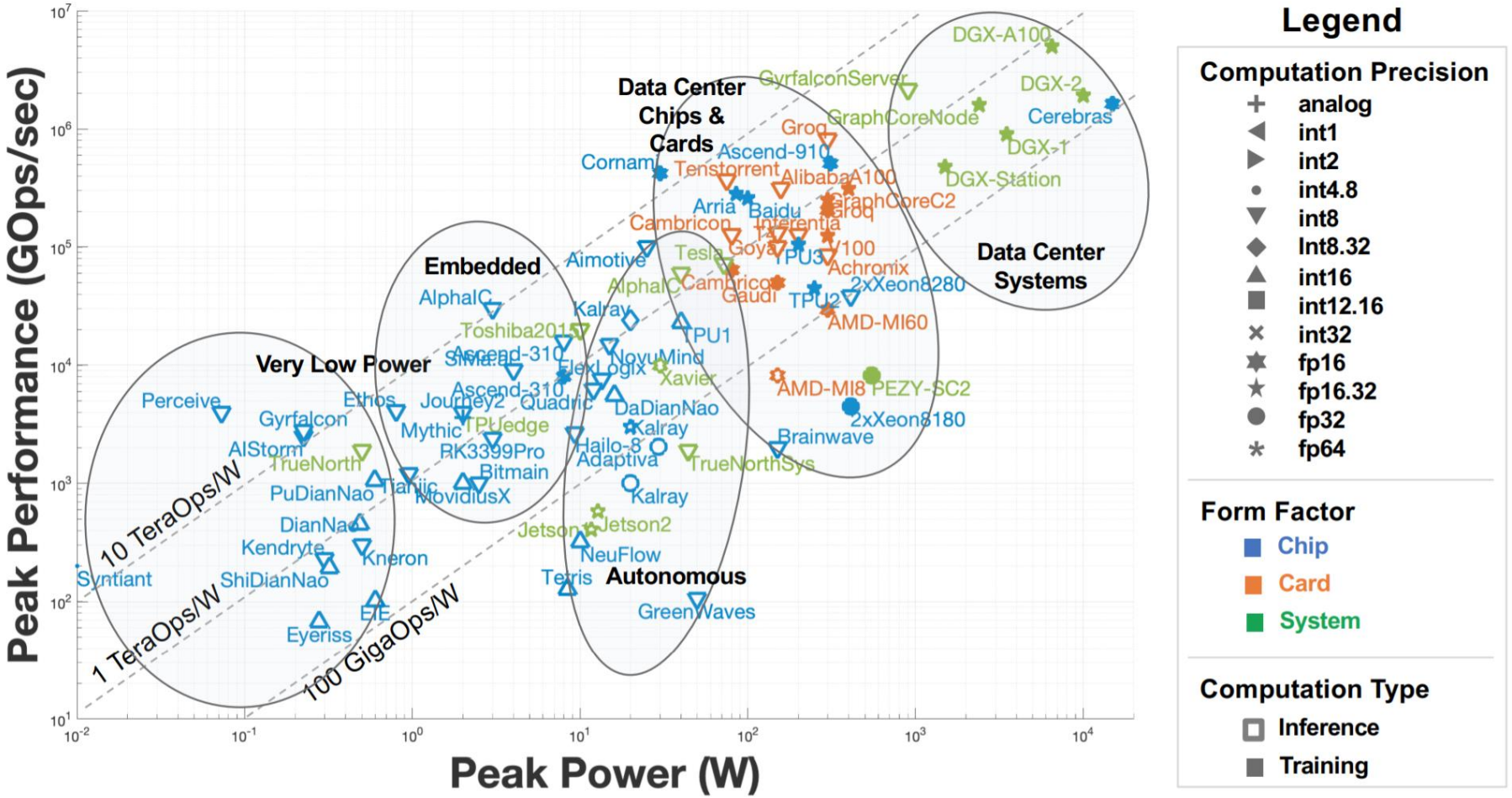


Source: https://www.youtube.com/watch?v=6ZDoFomU10A

Need for Computation Power

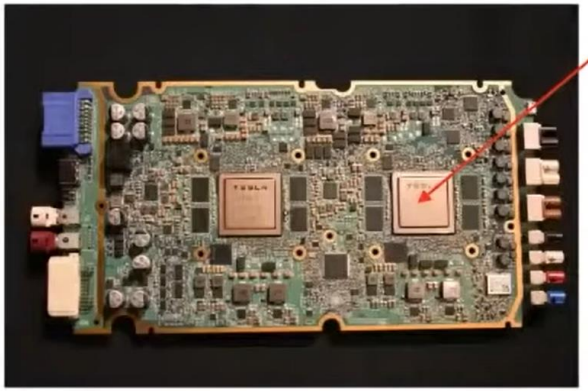


AI Accelerators



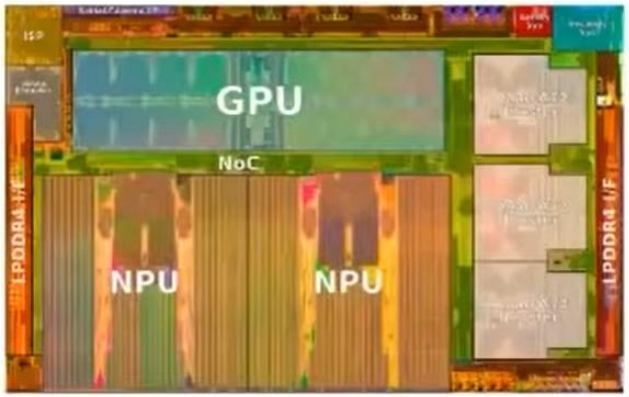
Example: Tesla FSD Computer

FSD computer

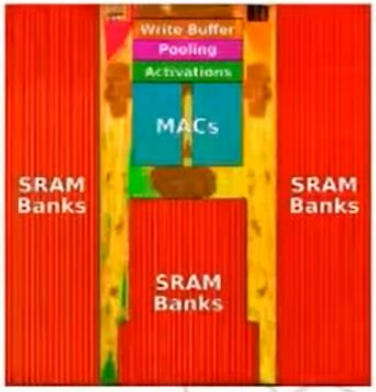


1 FSD Chip:
12 CPUs @ 2.2 GHz
GPU (600 GFLOPS)
2X NPU (36.86 TOPS / NPU)
36W

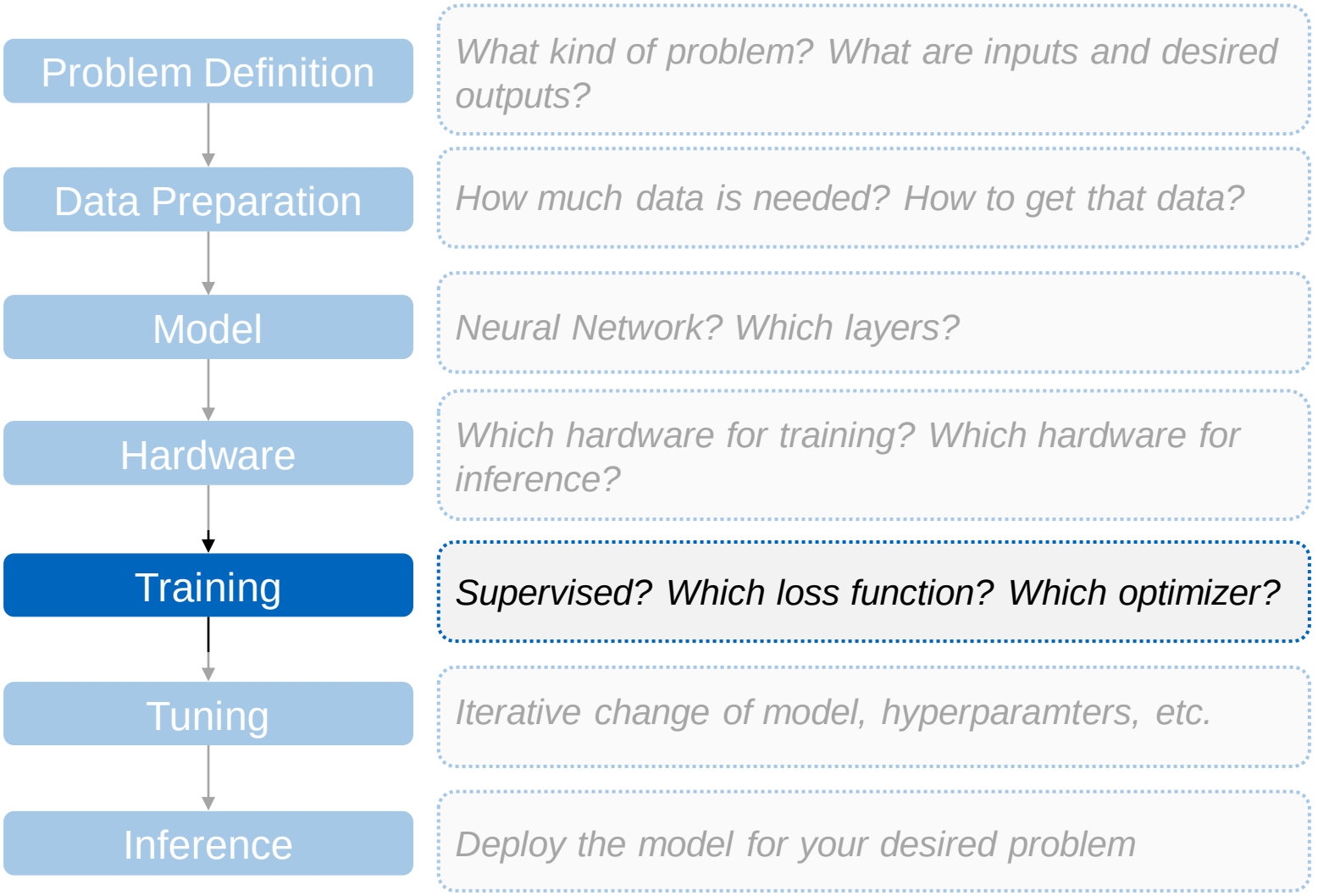
FSD chip



NPU



ML Development Workflow



Common Loss Functions

Mean Absolute Error (L1 Loss) $$MAE = \frac{\sum_{i=1}^{n} | y_i - \hat{y}_i |}{n}$$

Mean Square Error (L2 Loss) $$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

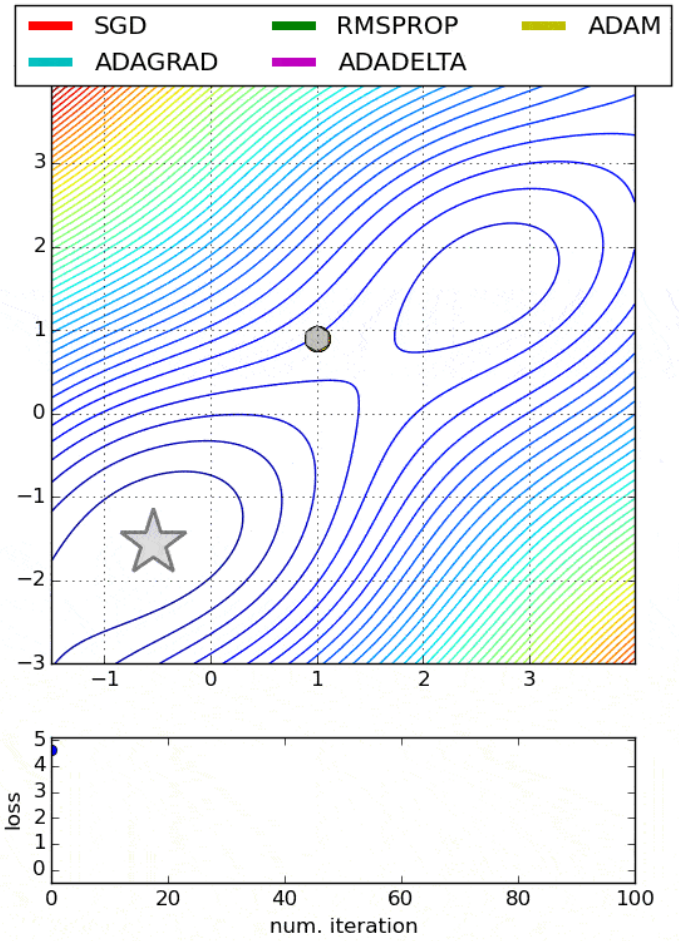
Binary Cross Entropy Loss $$-(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Loss Function

Loss Function	Use cases
Binary Cross-Entropy	Works best in equal data distribution among classes scenarios Bernoulli distribution based loss function
Weighted Cross-Entropy	Widely used with skewed dataset Weighs positive examples by β coefficient
Balanced Cross-Entropy	Similar to weighted-cross entropy, used widely with skewed dataset weighs both positive as well as negative examples by β and $1 - \beta$ respectively
Focal Loss	works best with highly-imbalanced dataset down-weight the contribution of easy examples, enabling model to learn hard examples
Distance map derived loss penalty term	Variant of Cross-Entropy Used for hard-to-segment boundaries
Dice Loss	Inspired from Dice Coefficient, a metric to evaluate segmentation results. As Dice Coefficient is non-convex in nature, it has been modified to make it more tractable.
Sensitivity-Specificity Loss	Inspired from Sensitivity and Specificity metrics Used for cases where there is more focus on True Positives.
Tversky Loss	Variant of Dice Coefficient Add weight to False positives and False negatives.
Focal Tversky Loss	Variant of Tversky loss with focus on hard examples
Log-Cosh Dice Loss(ours)	Variant of Dice Loss and inspired regression log-cosh approach for smoothing Variations can be used for skewed dataset
Hausdorff Distance loss	Inspired by Hausdorff Distance metric used for evaluation of segmentation Loss tackle the non-convex nature of Distance metric by adding some variations
Shape aware loss	Variation of cross-entropy loss by adding a shape based coefficient used in cases of hard-to-segment boundaries.
Combo Loss	Combination of Dice Loss and Binary Cross-Entropy used for lightly class imbalanced by leveraging benefits of BCE and Dice Loss
Exponential Logarithmic Loss	Combined function of Dice Loss and Binary Cross-Entropy Focuses on less accurately predicted cases
Correlation Maximized Structural Similarity Loss	Focuses on Segmentation Structure. Used in cases of structural importance such as medical images.

Source: S. Jadon 2020, „A survey of loss functions for semantic segmentation“

Optimizer



Aspects of different optimizers:

- Computation time
- Local minima / global minima
- Convergence time
- Learning rate dependency
- Number of hyperparameters (e.g. momentum)

Own field of research!
→ Start with default optimizers (ADAM, SGD)

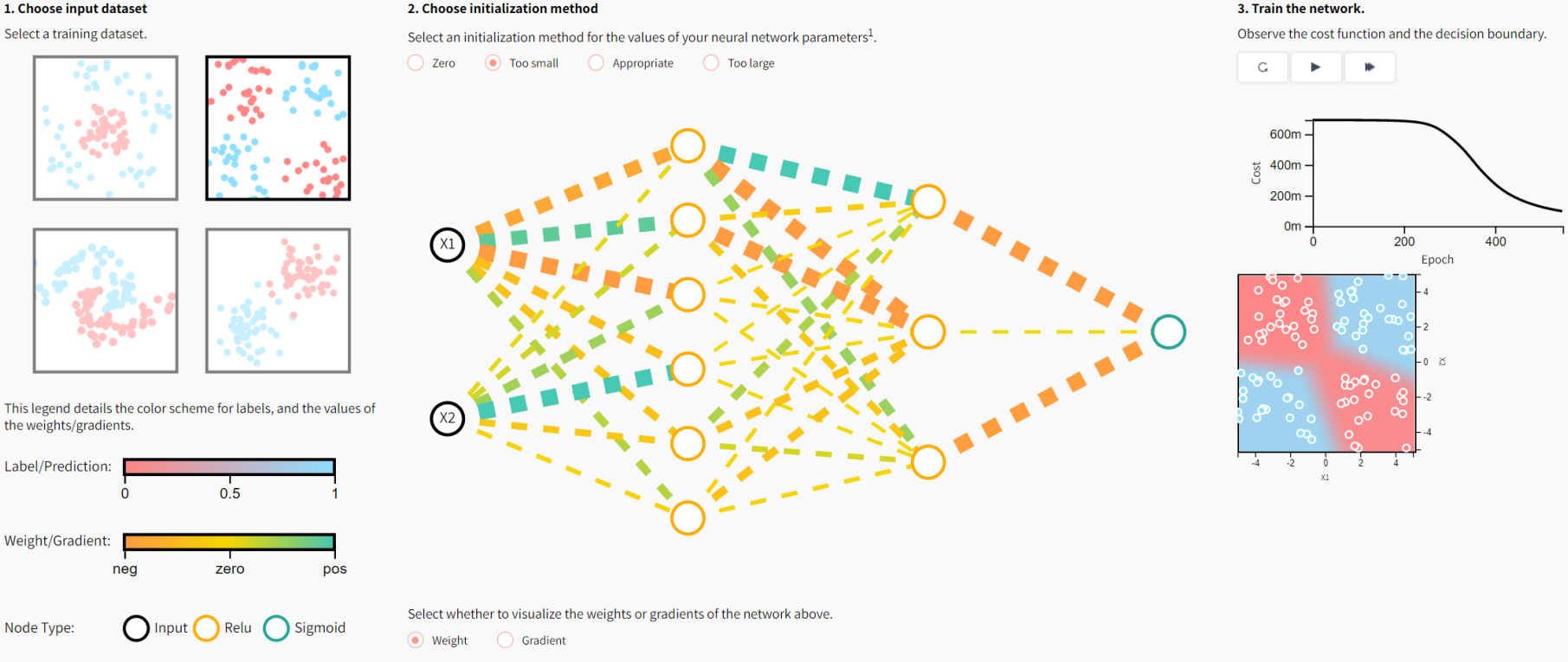
Learn more:
https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Weight initialization

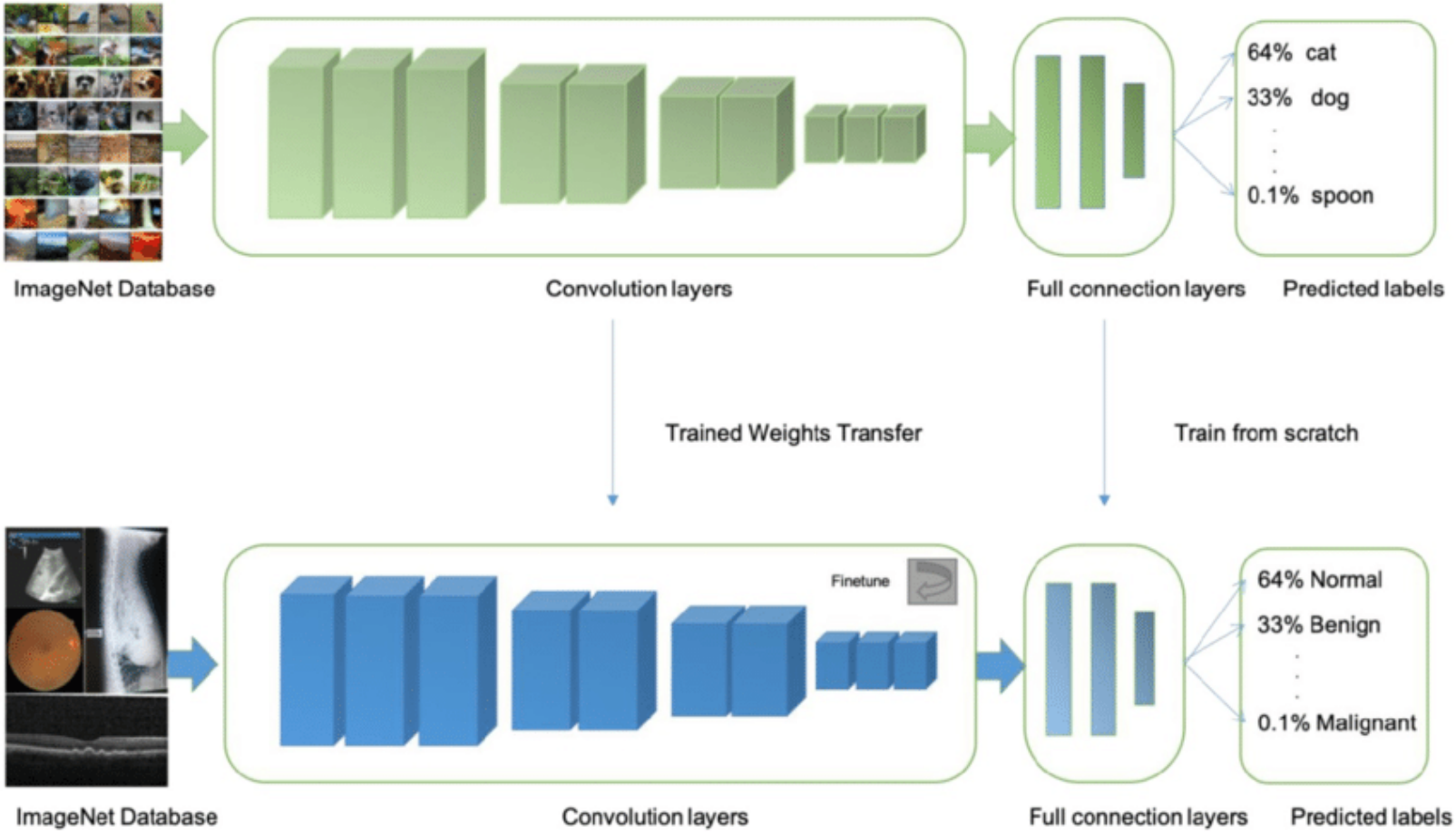
- Necessary to have a starting point for the optimization
- Two important dimensions:
  - Value Range
    - Too small values lead to slow learning
    - Too large vales may lead to divergence
    - Keep in mind vanishing and exploding gradient (Lecture 10)
  - Distribution
    - Constant initialization performs very poorly → Need for randomness
    - Xavier initialization with
      - Mean of activations is zero
      - Variance of activations stays the same across every layer
- Best practice: start with default initialization from your ML library

Weight initialization: Playgorund

https://www.deeplearning.ai/ai-notes/initialization/

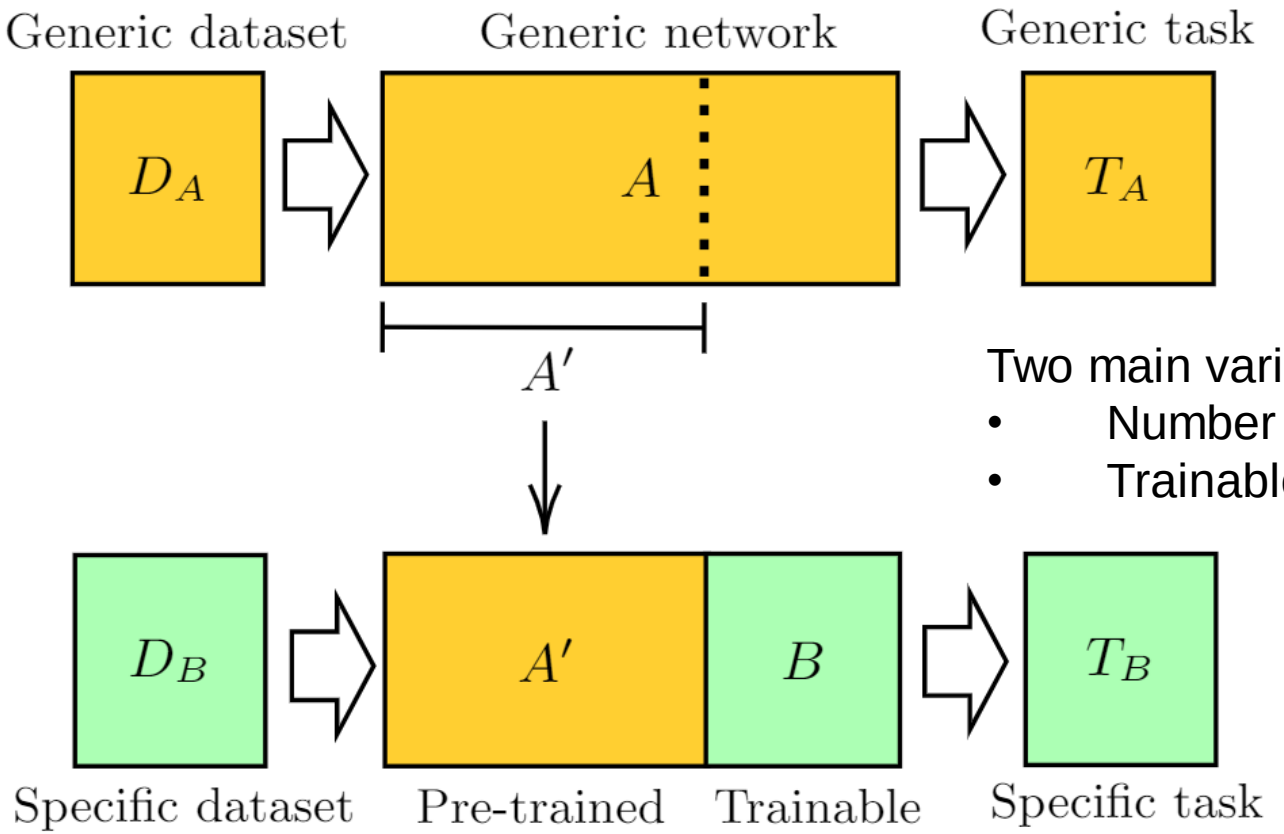


Transfer Learning



Source: https://www.researchgate.net/publication/336874848_Current_status_and_future_trends_of_clinical_diagnoses_via_image-based_deep_learning

Transfer Learning

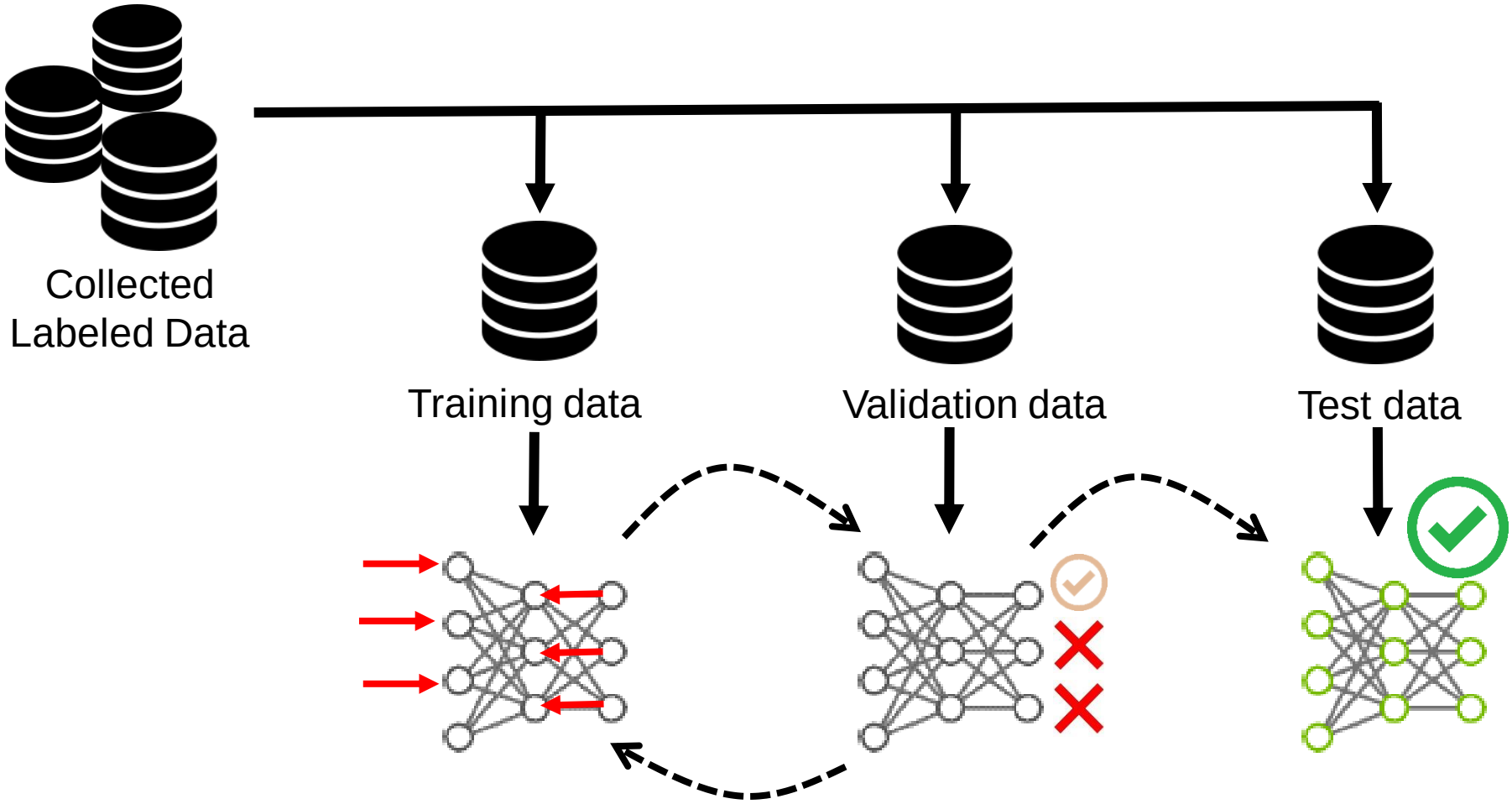


Two main variables:

- Number of layers (A´)
- Trainable/fixed layers

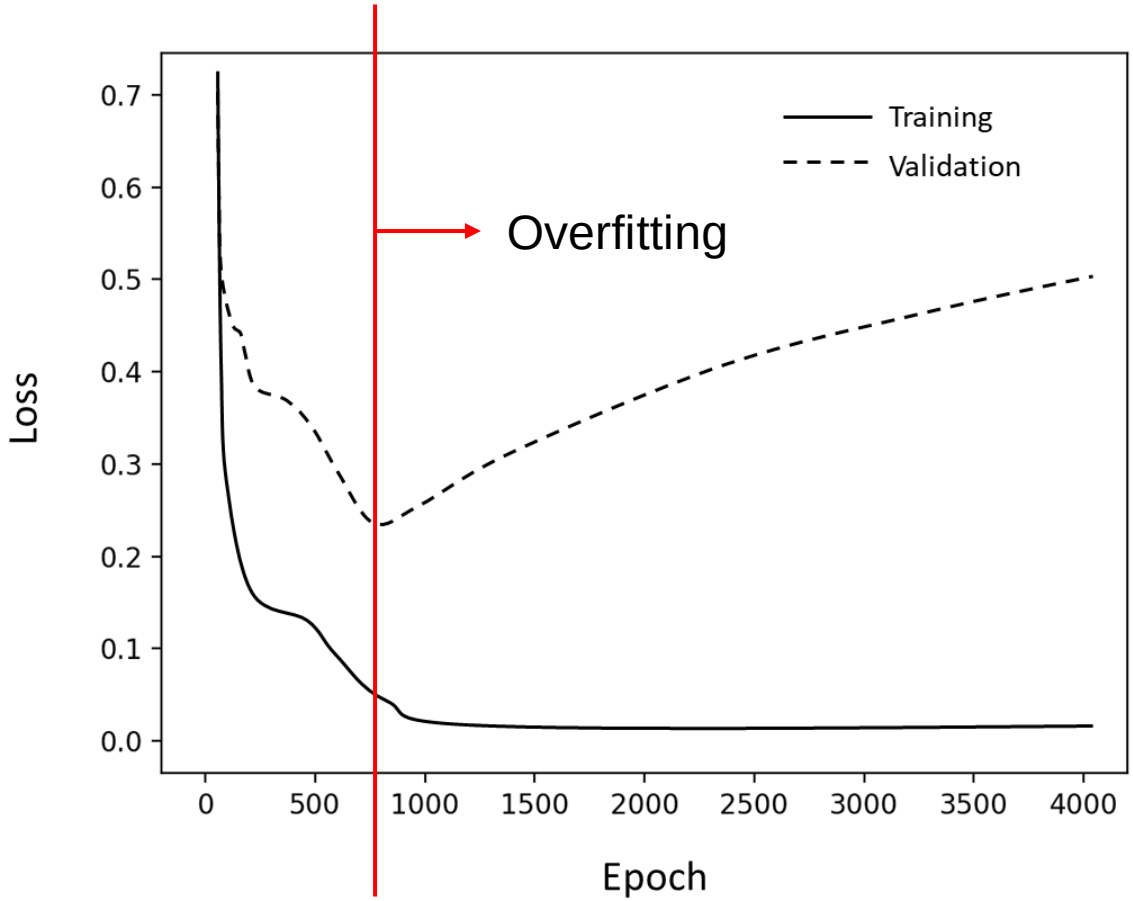
Image Source: https://pennylane.ai/qml/demos/tutorial_quantum_transfer_learning.html

Data – Provide the Data of Your Dataset



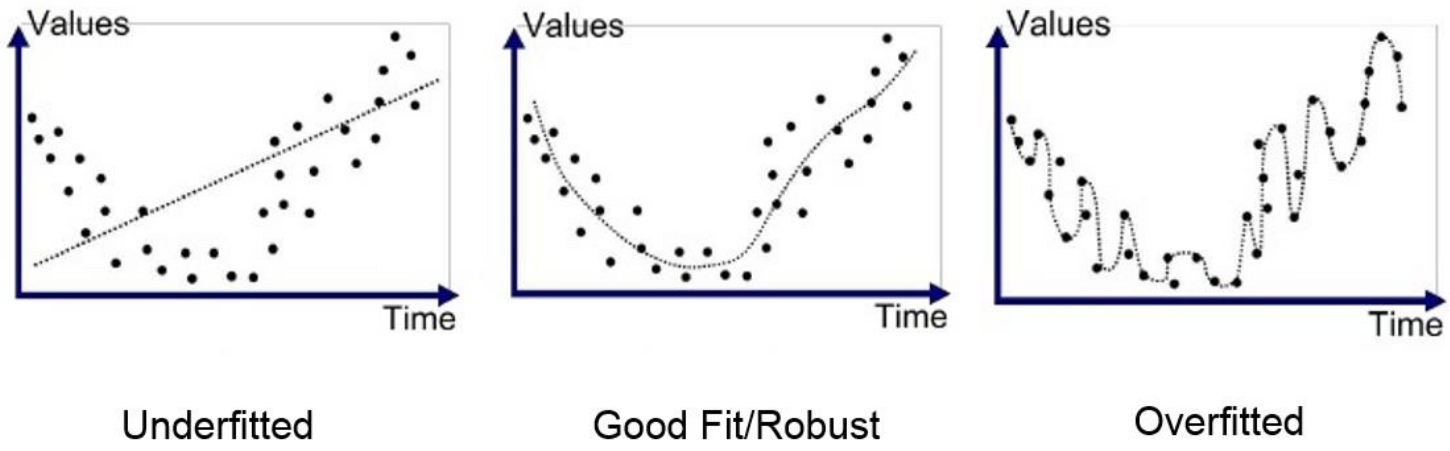
Split your dataset into training (60%), validation (20%) and test (20%) sets

Training and Validation Set



How to prevent overfitting?

1. Increase data set / data set augmentation
2. Reduce the model size
3. Early stopping
4. Regularization
5. Dropouts

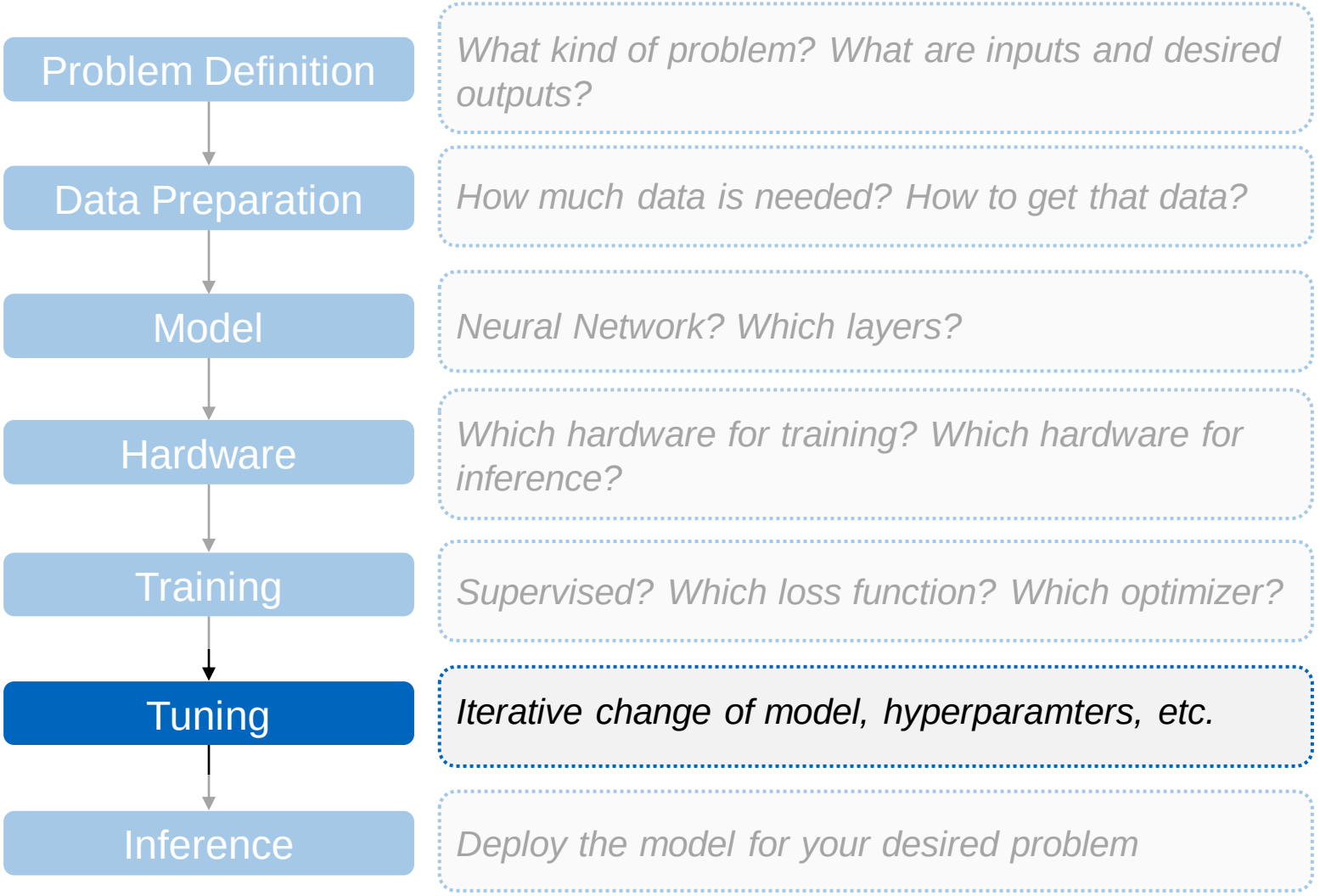


Neural networks and randomness

- Randomness plays an important role e.g.
  - Sequence of the data
  - Initialization
  - Optimizer (e.g. SGD)
  - Random regularization e.g. dropouts
  - Calculation architecture may be non-deterministic

- Setting a random seed helps to maintain determinism but not all calulcations are necessarily deterministic
- ➔ Train your neural network several times with the same settings to account for the impact of randomness
- ➔ The comparability of neural network results is often difficult

ML Development Workflow



What are Hyperparameters?

Training

- Learning Rate, Decay Rate
- Batch-Size
- Number of epochs
- Dropout Rate
- Regularization
- …

Model

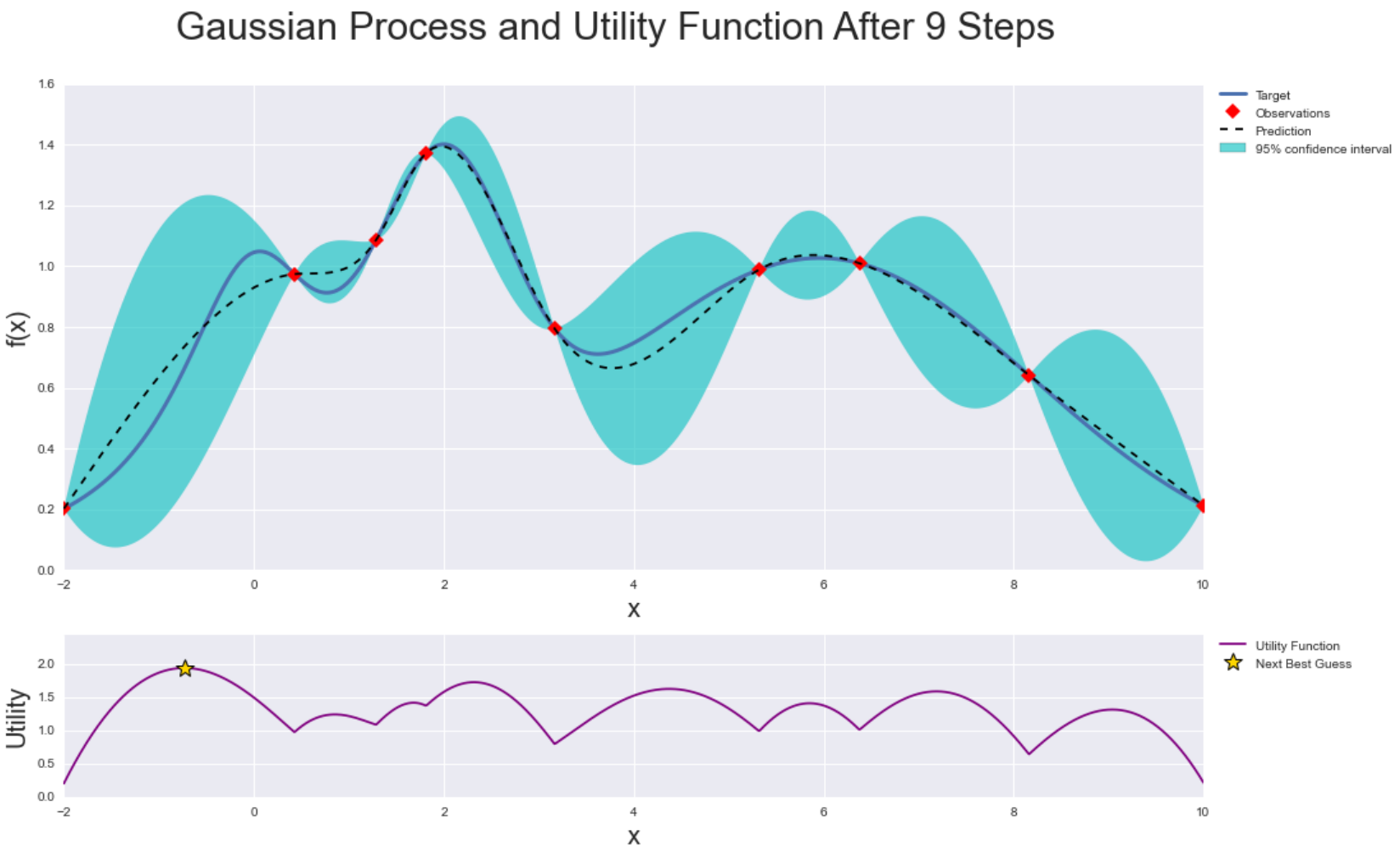
- Number of layers
- Number of parameters in a layer
- …

```
{
    "gpu": "0",
    "worker": 60,
    "enc_dec_layer": "gru",
    "encoder_size": 32,
    "decoder_size": 64,
    "in_length": 30,
    "out_length": 40,
    "grid_size": [13,3],
    "soc_conv_depth": 32,
    "conv_3x1_depth": 8,
    "dyn_embedding_size": 16,
    "input_embedding_size": 16,
    "num_img_filters": 32,
    "train_flag": true,
    "continue_training_from": "trained_models/gru/gru_rmse.tar",
    "save_path": "trained_models/",
    "model_name": "max",
    "dataset": "data/max",
    "img_path": "data/sc_imgs_64_max",
    "save_best": true,
    "lr_rmse": 1.5e-4,
    "decay_rmse": 0.0,
    "lr_nll": 2.5e-5,
    "decay_nll": 1e-3,
    "scene_images": true,
    "scene_image_method": "self-rendered",
    "tb_logs": "tb_logs",
    "pretrainEpochs": 0,
    "trainEpochs": 3,
    "dec_img_size": 32,
    "batch_size": 128,
    "watch_radius": 64
}
```

Hyperparameter Optimization for Neural Networks

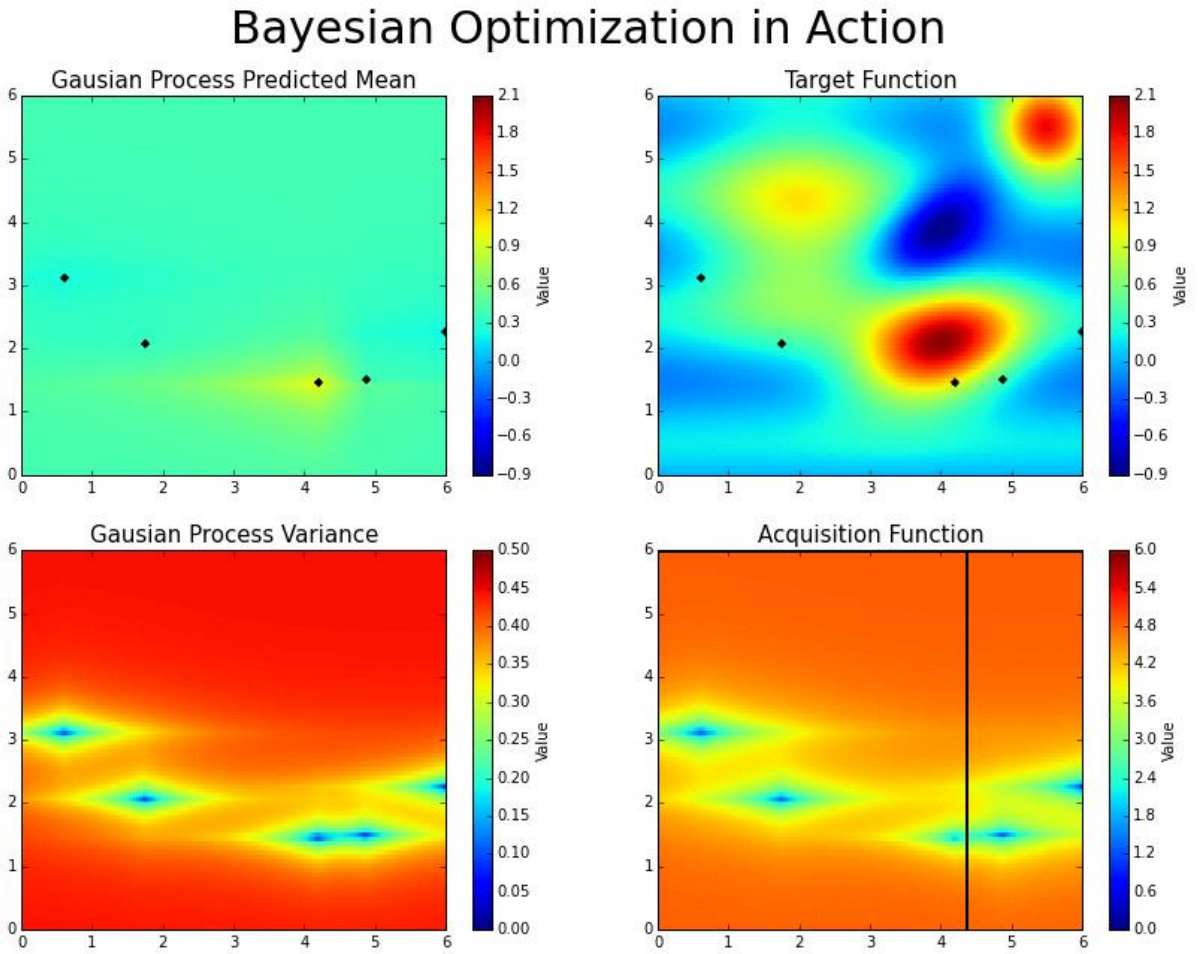
- Lots of hyperparameters → High-dimensional optimization problem
- Hyperparameters are not independent of each other
  - For some of them rough relation explainable: e.g. learning rate and number of epochs
  - For some of them relation not explainable: e.g. dropout rate and batch size
- Training may take hours or days → limited number of trainings for optimization
- Different from problem to problem
- Gradient-free optimization

Bayesian Optmization for Hyperparamter Tuning



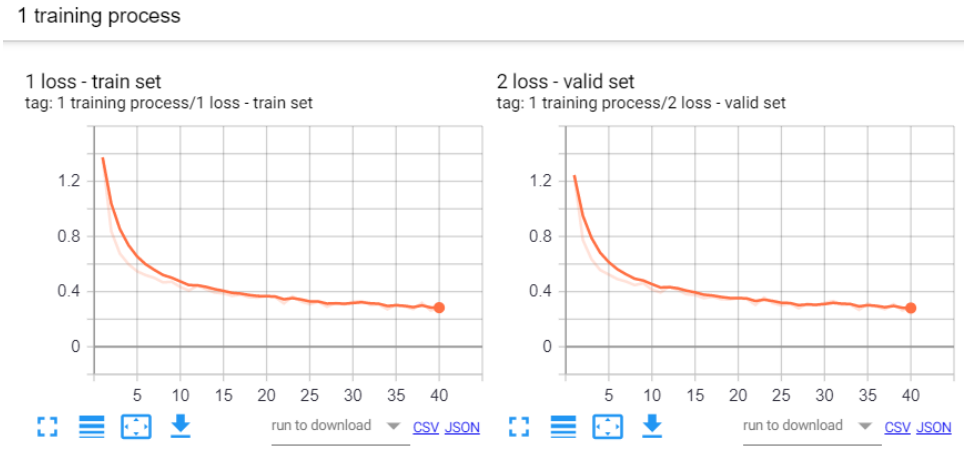
Source: https://github.com/fmfn/BayesianOptimization

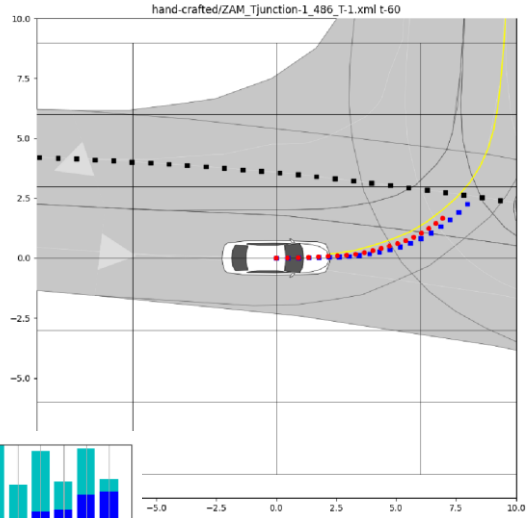
Bayesian Optimization in Action

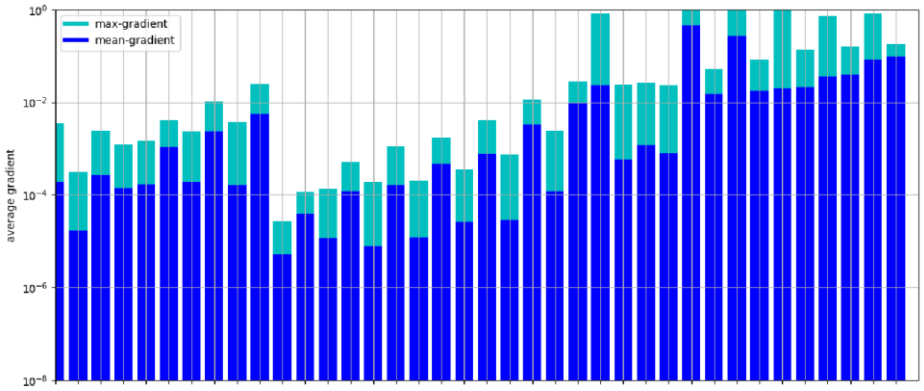


Source: https://github.com/fmfn/BayesianOptimization

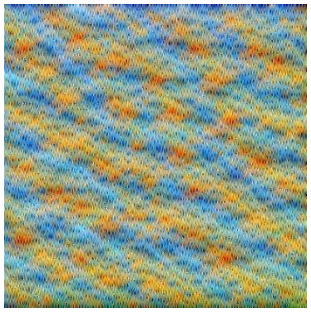
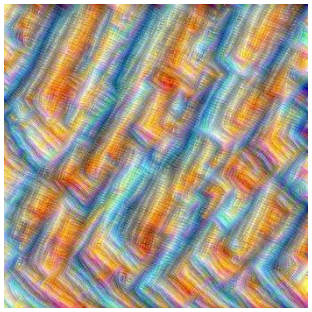
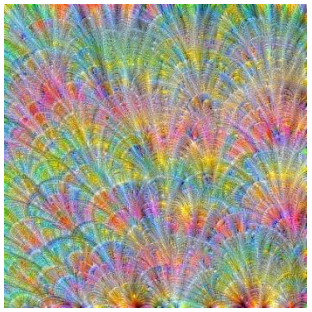
Training Monitoring: Tensorboard



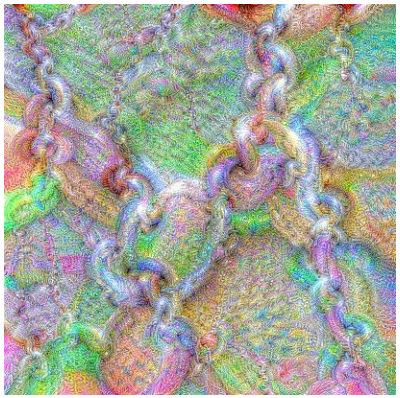




Looking into the BlackBox: Feature Visualization

Early layer → Medium layer →

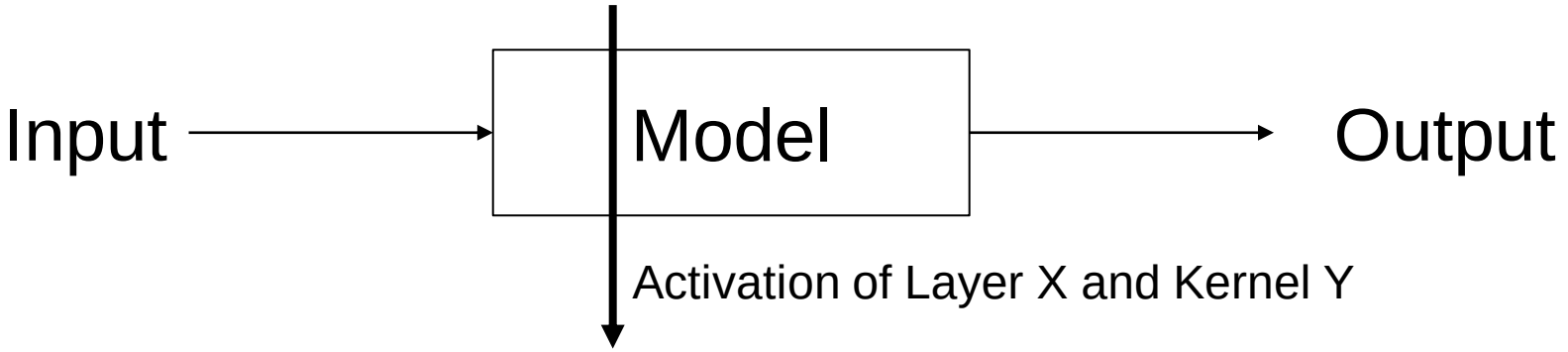


Last layer

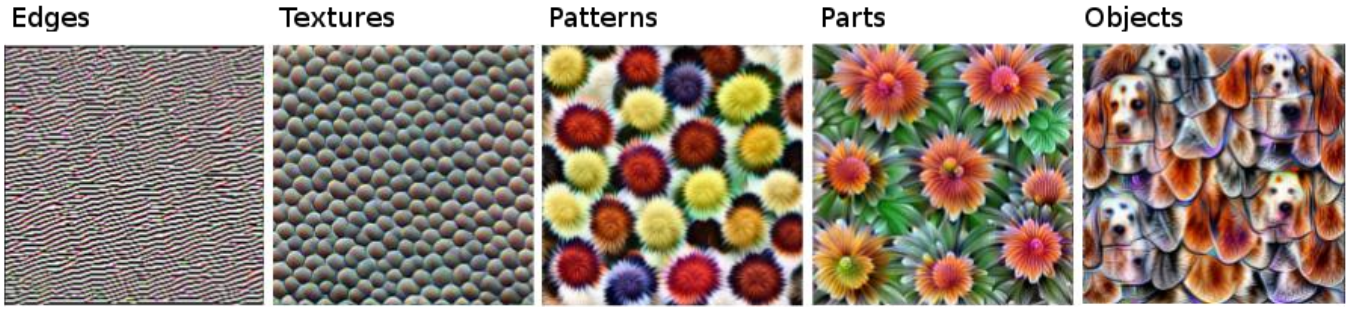
Generate/optimize input images, so that a high activation in a certain layer is reached

Source: Erhan, Dumitru & Bengio, Y. & Courville, Aaron & Vincent, Pascal. (2009). Visualizing Higher-Layer Features of a Deep Network. Technical Report, Univeristé de Montréal.

Looking into the BlackBox: Feature Visualization



1. Start with a random generated input and a trained model
2. Use backpropagation to maximize activations in a certain layer
3. Stop when optimum is reached
4. The „optimized“ input indicates the purpose of a (convolutional) layer

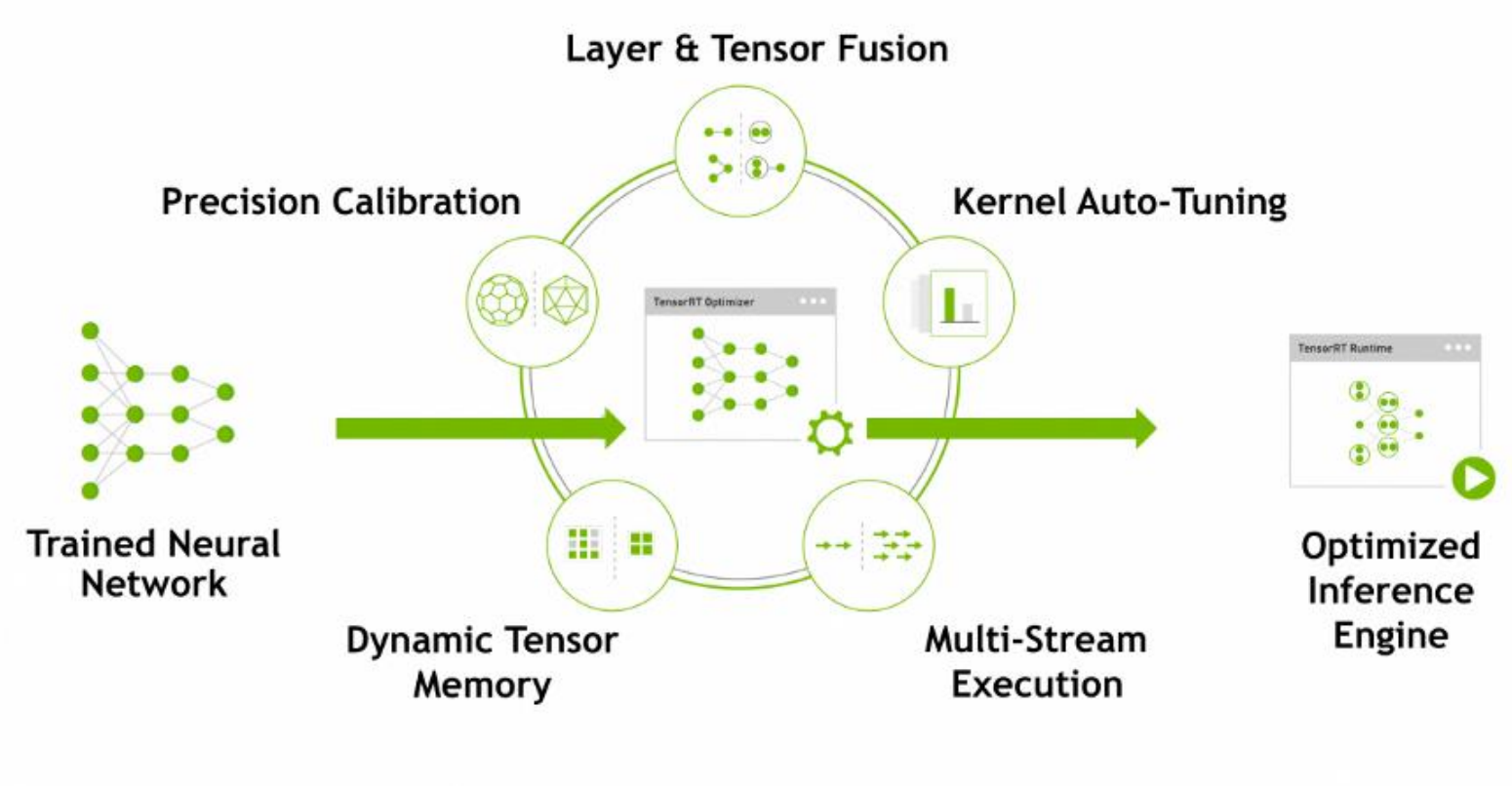


Looking into the BlackBox: Activation Maps

Which inputs are important for the generated output of the network?

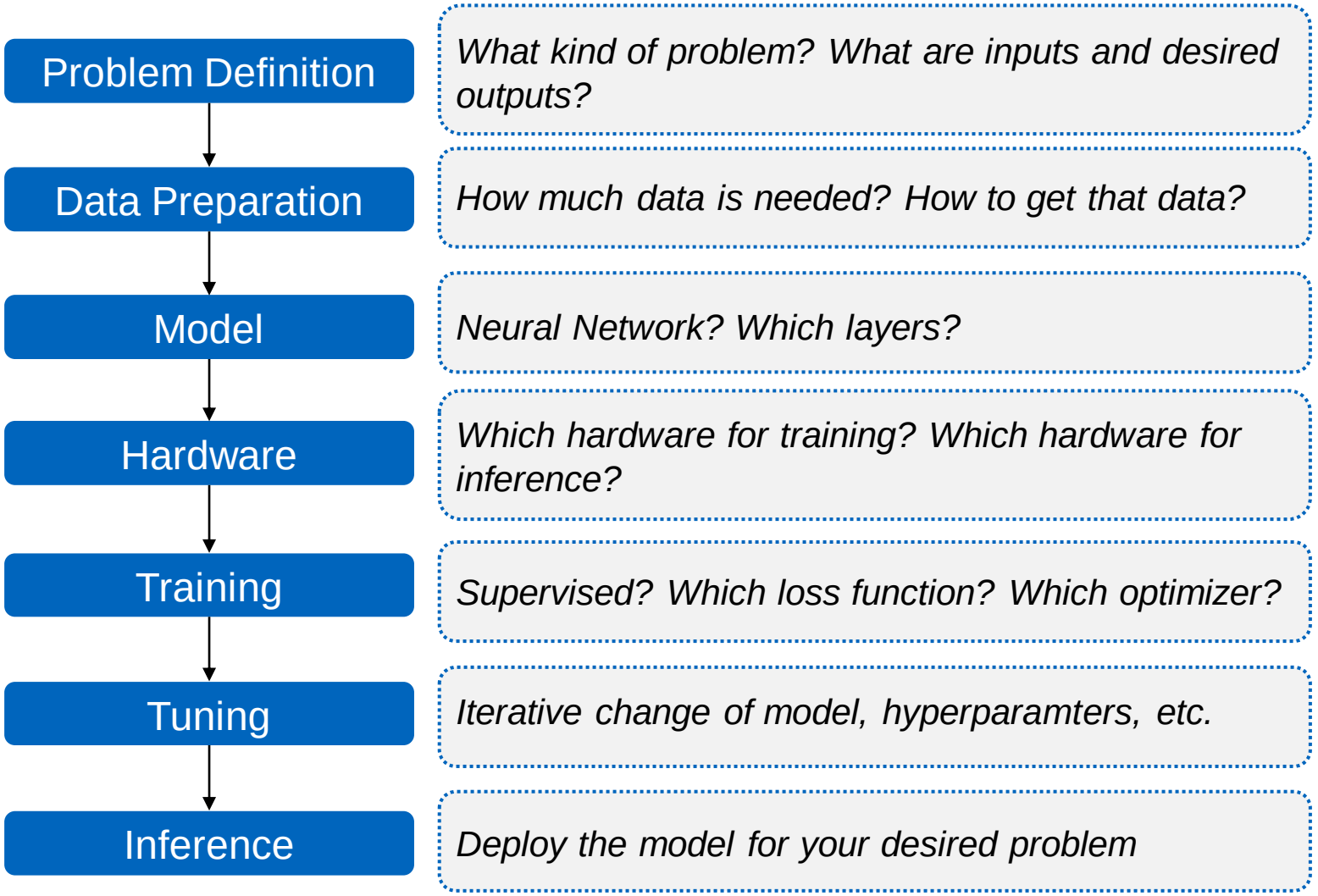
Source: https://glassboxmedicine.com/2019/06/11/cnn-heat-maps-class-activation-mapping-cam/

Post-training Optimization



Source: NVIDIA, TensorRT

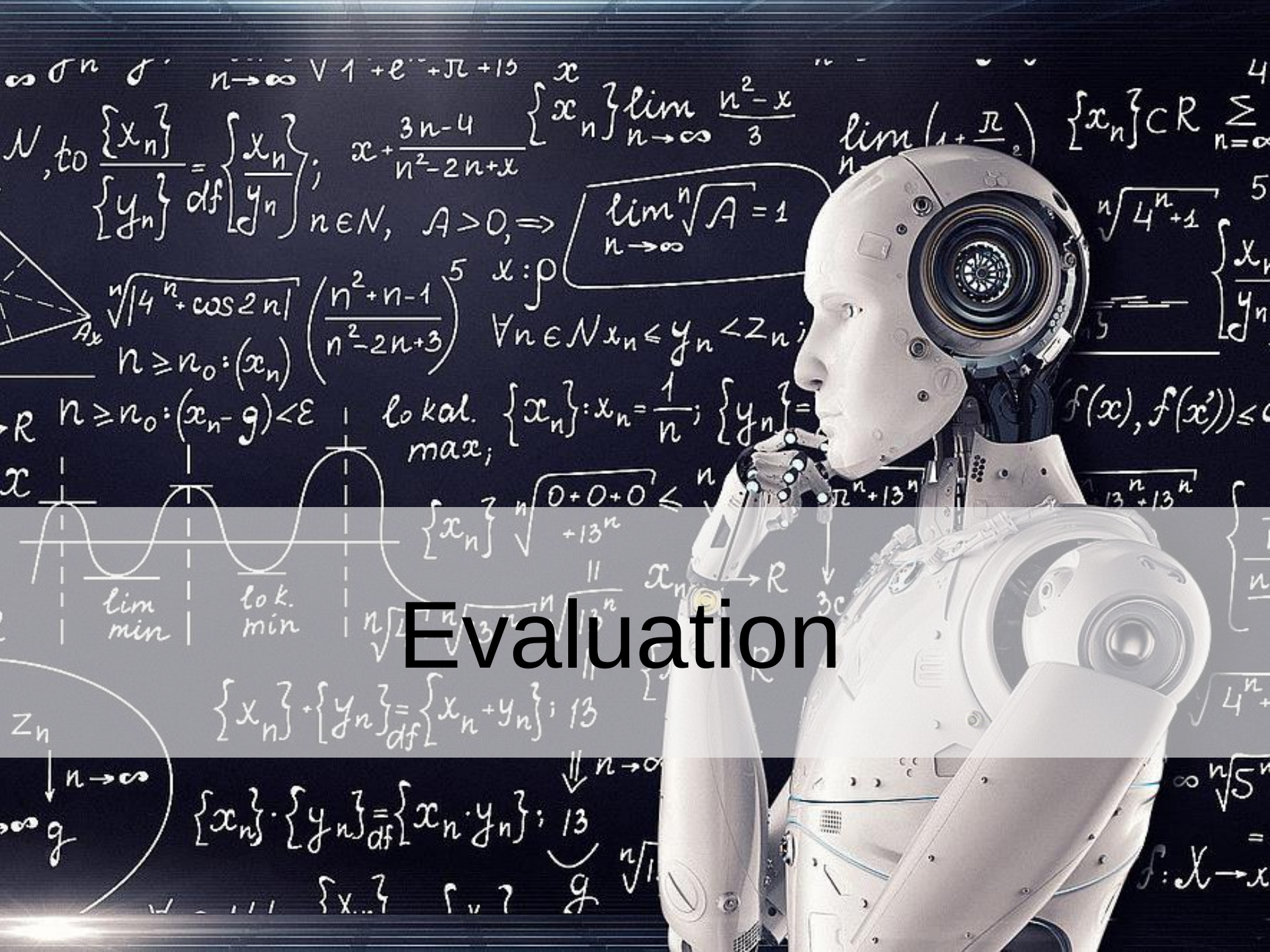
ML Development Workflow



Summary

What did we learn today:

- The development for deep learning algorithm is an **iterative process**
- **Data is a crucial part** in deep learning:
  - The better and specific your data, the better your results will be
  - The more data you have, the better your results will be.
  - Consider different kinds of bias
- Consider **different hardware types** for **training** as well as for **inference**
- Hyperparamter tuning is tricky. Try to understand the correlations by **monitoring the training** and **making use of visualization methods**
- Make use of **published datasets, deep learning frameworks and even pretrained models**


Evaluation