

# INTRODUCTION TO MACHINE LEARNING

Deep Learning for Climate Scientists  
ML Workshop Fall 2026 @ DKRZ

<Name>

Deutsches Klimarechenzentrum (DKRZ)

# Course Expectations



- What is your name?
- What do you expect to learn in this course?
- Name one thing that you associate with ML. Prior experiences with it?
- How could your projects/research benefit from applying ML?

# Workshop Outline – Day 1

10:30

## Day 1

### Sitzung

10:30–12:30 Uhr **Introduction to Machine Learning | Lecture**

**Sprecher**

Alexander Fischer

12:30–14:00 Uhr **Lunch Break**

14:00–15:45 Uhr **Architectures and Applications | Lecture**

**Sprecher**

Paul Keil

15:45–16:00 Uhr **Coffee break**

16:00–17:30 Uhr **Introduction to PyTorch: Core Concepts | Hands-on**

**Sprecher**

Etienne Plésiat

17:30

# Workshop Outline – Day 2

09:00

## Day 2 Sitzung

09:00–10:45 Uhr **Introduction to PyTorch: Application to climate data | Hands-on**

**Sprecher**  
Etienne Plésiat

10:45–11:00 Uhr **Coffee Break**

11:00–12:00 Uhr **Introduction to PyTorch: Application to climate data | Hands-on**

**Sprecher**  
Etienne Plésiat

12:00–13:30 Uhr **Lunch break**

17:30

13:30–14:30 Uhr **Introduction to Pytorch Lightning | Hands-on**

**Sprecher**  
Caroline Arnold

14:30–15:15 Uhr

**Advanced Deep Learning use-case: Reconstructing missing climate data | Hands-on**

**Sprecher**  
Johannes Meuer

15:15–15:30 Uhr **Coffee break**

15:30–16:30 Uhr

**Advanced Deep Learning use-case: Reconstructing missing climate data | Hands-on**

**Sprecher**  
Johannes Meuer

16:30–17:30 Uhr

**Advanced Deep Learning Use Case: AI Weather Prediction | Lecture**

**Sprecher**  
Caroline Arnold, Paul Keil

# Workshop Outline – Day 3

09:00

## Day 3 Sitzung

09:00–09:45 Uhr **Beyond CNNs: Transformers & Attention | Lecture**

### Sprecher

Alexander Fischer

### Ort

DKRZ Main Building, Room 034, DKRZ Main Building, Bundesstraße 45a, 20146 Hamburg

09:45–10:30 Uhr **Probabilistic Deep Learning | Lecture**

### Sprecher

Paul Keil

### Ort

DKRZ Main Building, Room 034, DKRZ Main Building, Bundesstraße 45a, 20146 Hamburg

10:30–10:45 Uhr **Coffee break**

10:45–12:00 Uhr **Probabilistic Deep Learning | Hands-on**

### Sprecher

Paul Keil

### Ort

DKRZ Main Building, Room 034 / 023 / 120, DKRZ Main Building, Bundesstraße 45a, 20146 Hamburg

10:45–12:00 Uhr **Attention & Transformer | Hands-on**

### Sprecher

Alexander Fischer, Johannes Meuer, Maximilian Witte

### Ort

DKRZ Main Building, Room 034 / 023 / 120, DKRZ Main Building, Bundesstraße 45a, 20146 Hamburg

10:45–12:00 Uhr **Q&A | Recap**

### Sprecher

Caroline Arnold

### Ort

DKRZ Main Building, Room 034 / 023 / 120, DKRZ Main Building, Bundesstraße 45a, 20146 Hamburg

12:00–12:30 Uhr **Closing remarks | Discussion**

### Sprecher

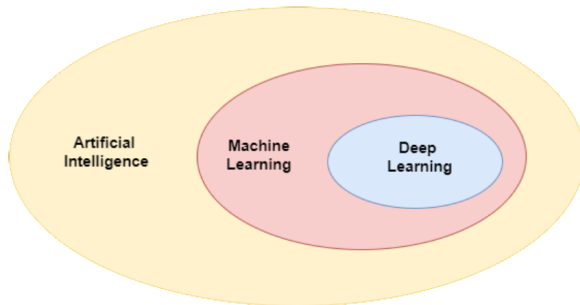
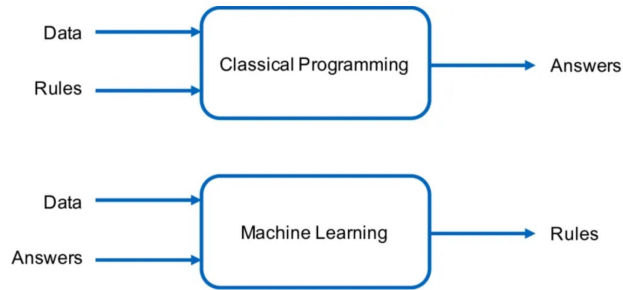
Paul Keil

### Ort

DKRZ Main Building, Room 034, DKRZ Main Building, Bundesstraße 45a, 20146 Hamburg

12:30

# What exactly is Machine Learning?



- **Classical Programming:** Humans define *deterministic rules (code)* to process input data → well-defined, predictable outputs
- **Machine Learning (ML):** Rules/patterns are purely *learned* from input data (features) and output labels
- **Deep Learning (DL):** Uses multi-layered neural networks to *automatically* learn complex, high-level features

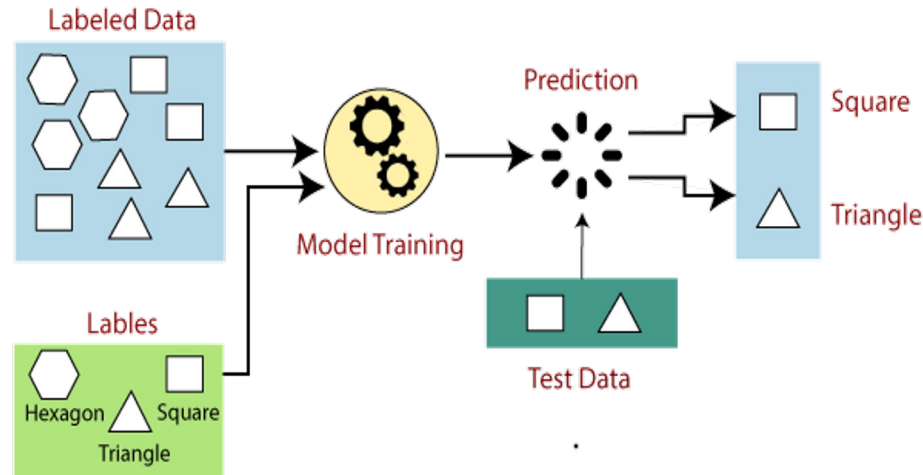
# Machine Learning – A Short Timeline...

- **1990s:** Support Vector Machines
- **2010s:** Deep Learning Resurgence: CNNs
- **2014:** Generative Adversarial Networks
- **2015:** AlphaGo: Reinforcement Learning
- **2018:** Transformer (BERT)
- **2020:** AlphaFold, Diffusion Models
- **2022:** LLMs (GPT-3)



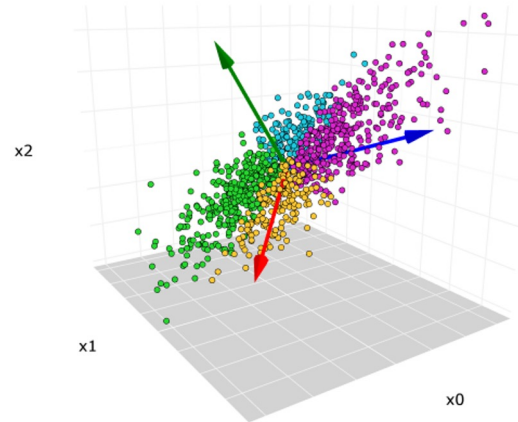
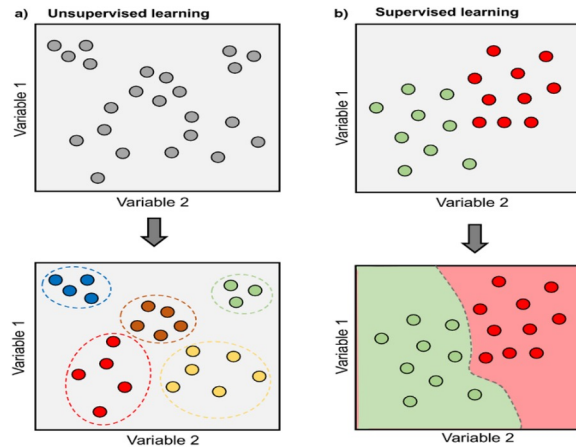
# Types of Machine Learning – Supervised ML

- The algorithm is trained on a *labeled* dataset
- Input data is paired with corresponding target labels
- Examples: Classification, Regression



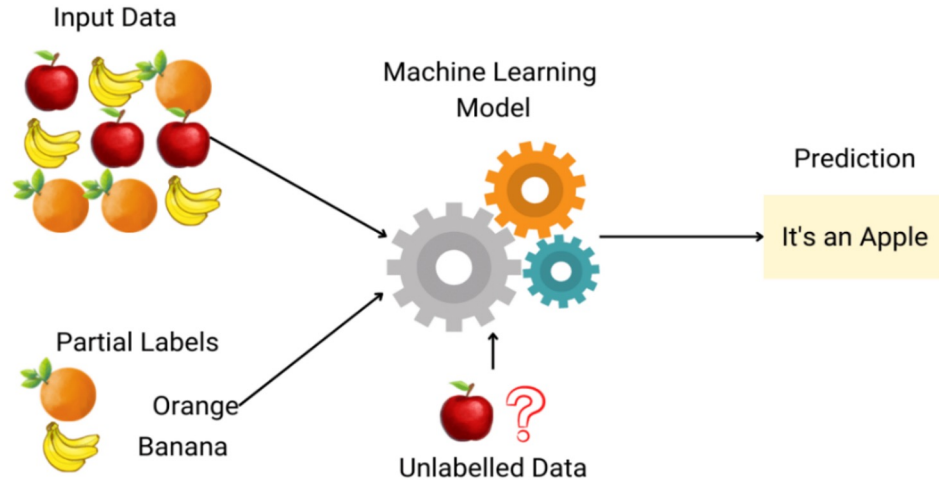
# Types of Machine Learning – Unsupervised ML

- The algorithm is trained on an *unlabeled* dataset
- Discover hidden patterns, relationships, or coherent structures within the data
- Examples: Clustering, Dimensionality Reduction, PCA



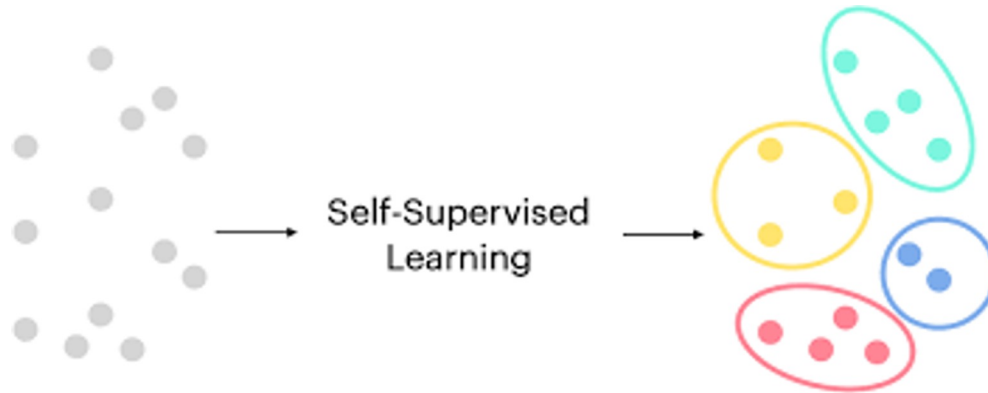
# Types of Machine Learning – Semi-Supervised ML

- The algorithm is trained on a labeled *and* an unlabeled dataset
- Leveraging on labelled and unlabeled data to improve performance



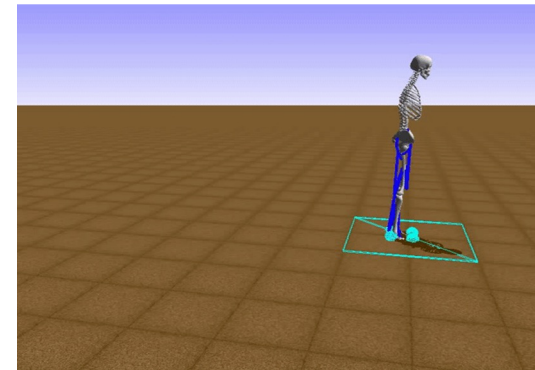
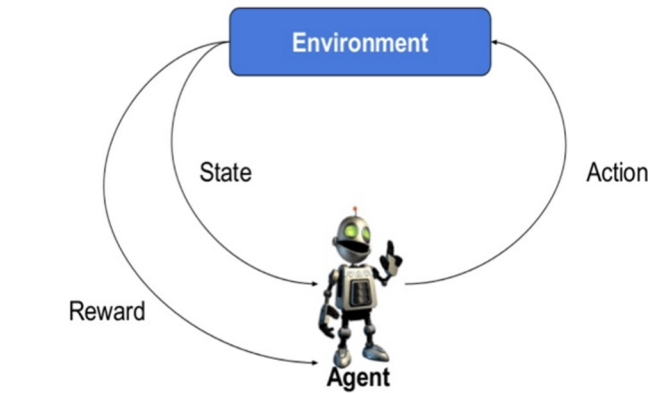
# Types of Machine Learning – Self-Supervised ML

- The algorithm generates *its own labels* for training from input dataset  
→ does not require external / manual labels
- Examples: Auto-encoders, Contrastive learning, Masked Language Modeling



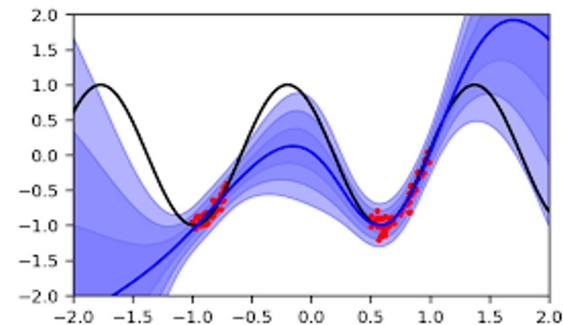
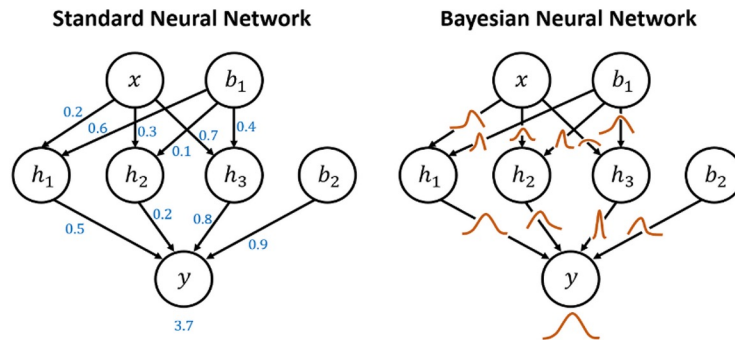
# Types of Machine Learning – Reinforcement Learning

- An agent interacting with an environment → model learns based on feedback
- Learn a policy that maximizes the cumulative reward
- Examples: Humanoid robots, Games, autonomous systems



# Types of Machine Learning – Probabilistic ML

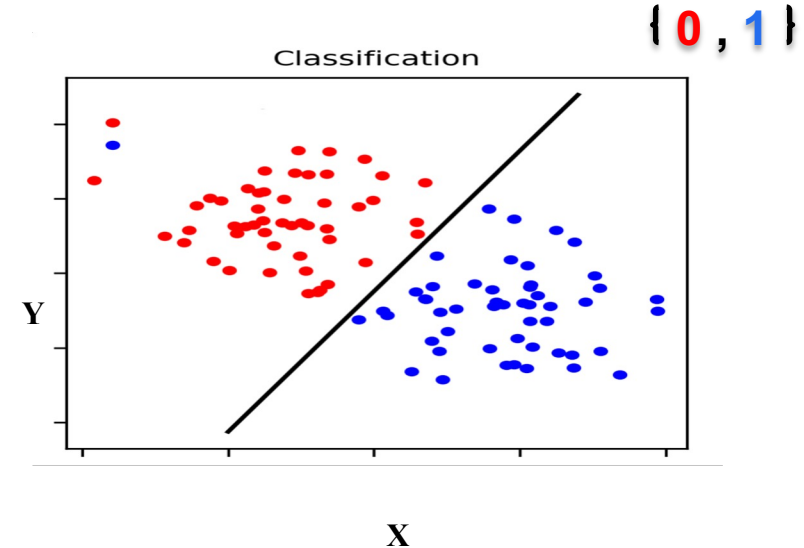
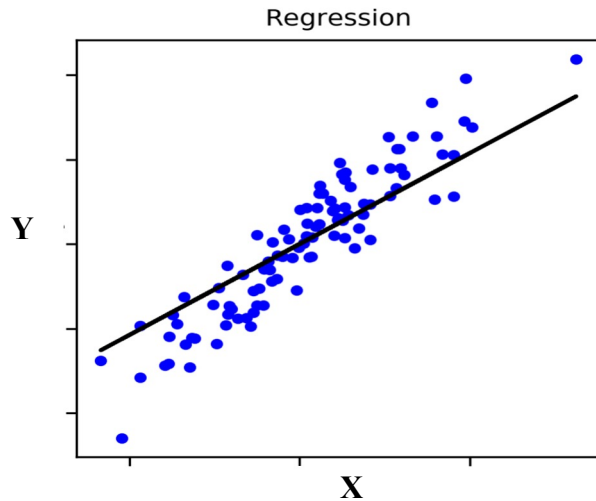
- Data and Model include *uncertainties (aleatoric, epistemic)*
- Capture uncertainties for more reliable predictions
- Probability distributions are maintained over weights
- Example: Mean-Variance Estimation, Bayesian Neural Networks (BNN)



# Questions?

# Regression and Classification in Machine Learning

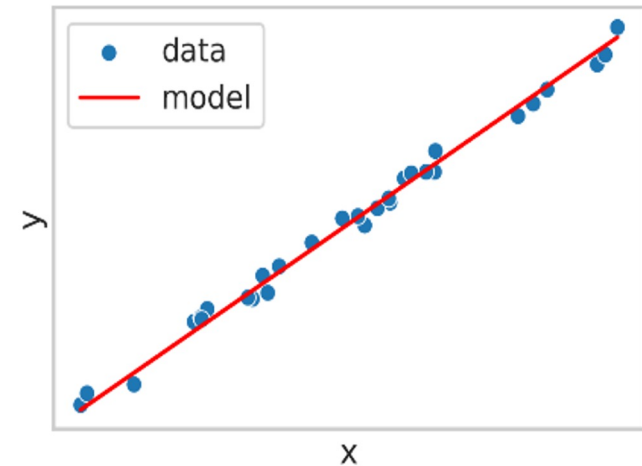
- **Regression:** Predicting a *continuous* outcome
- In- & outputs are continuous values
- **Classification:** Assigning *discrete* instances to classes
- The outputs are seen as class *probabilities*



# Typical Machine Learning Procedure (Linear Regression)

- Data set:  $D = \{x, y\}$
- 1D Model:
  - Defined as  $\hat{y} = wx + b$
  - Trainable parameters:  $w, b$
- Loss function:
  - $L(\hat{y}, y; w, b) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$
- Training objective:
  - Minimize loss function  
→ yields optimal parameters  $w^*, b^*$

Is this data  
labeled or  
not?

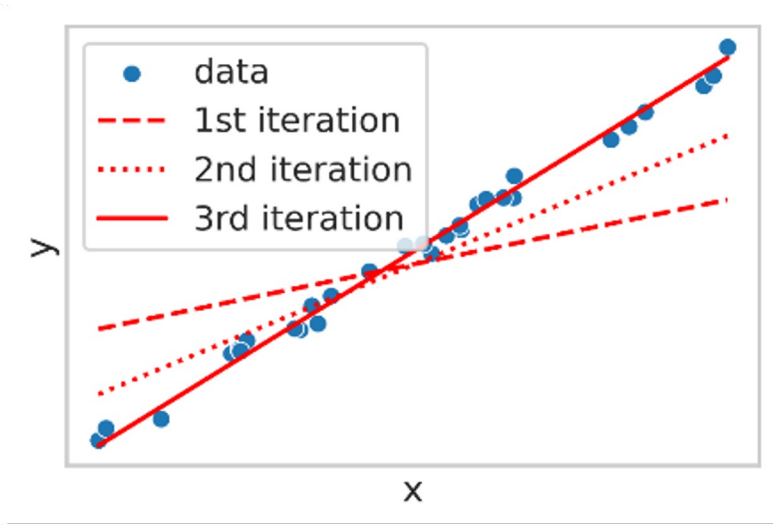


# Optimization with Gradient Descent

- Start with a random guess for the trainable parameters
- Compute the output, then the loss function value
- Parameter update in the direction of the *negative gradient*

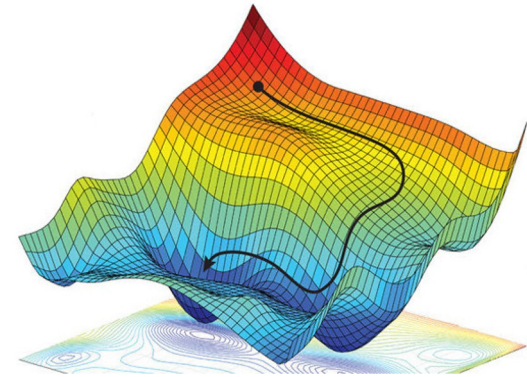
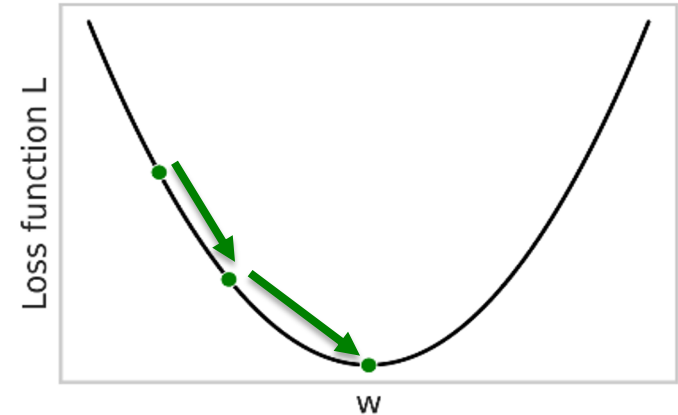
$$w_{i+1} = w_i - \alpha \nabla_{w_i} L(w_i)$$

- Learning Rate  $\alpha \in [0.0001, 0.1]$



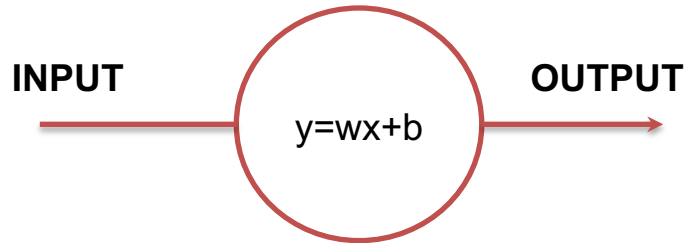
# Optimization with Gradient Descent

- Loss functions measure the *disparity* between predicted and actual values (scalar value)
- Learning objective: minimize the loss *w.r.t.* the model's parameters
- Step-wise optimization helps to find the *most suitable path*
  - Stochastic Gradient Descent (SGD)
  - Adaptive Momentum (Adam)
  - Adaptive Gradient (Adagrad)

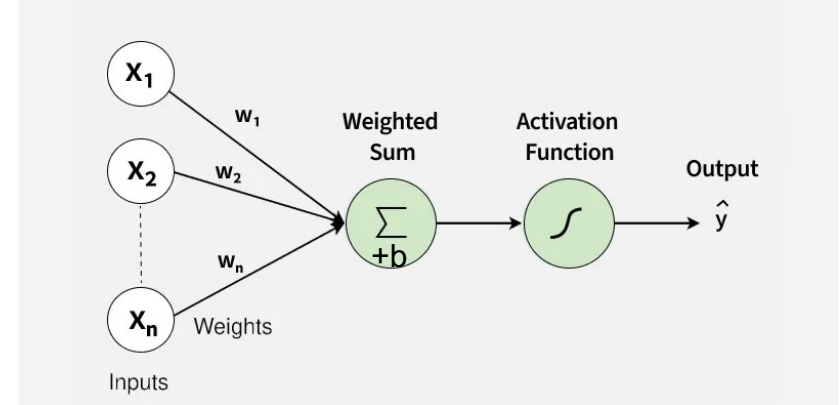


# From Linear Regression to Neural Networks

- Linear regression: building block for complex models
- Simplest form:  
**perceptron = neuron**

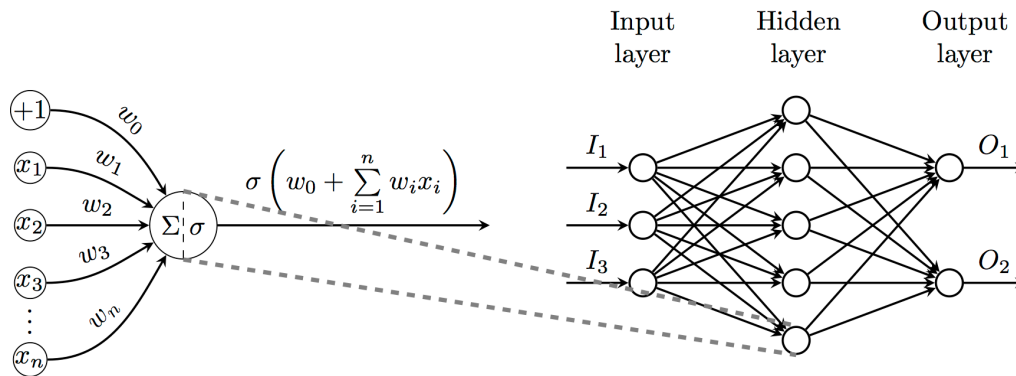


- Neural network: stack neurons and add nonlinear activation function  
 $O = \sum_{i=1}^n w_i x_i + b; \hat{y} = \sigma(O)$



# From Linear Regression to Neural Networks

- **Universal approximation theorem:**  
NN can approximate any “well-behaved” non-linear function
- Combine and stack several neurons per layer  
→ different processing variants of the same input
- Add hidden layer(s) for more *representative capacity*



Matrix-Vector notation

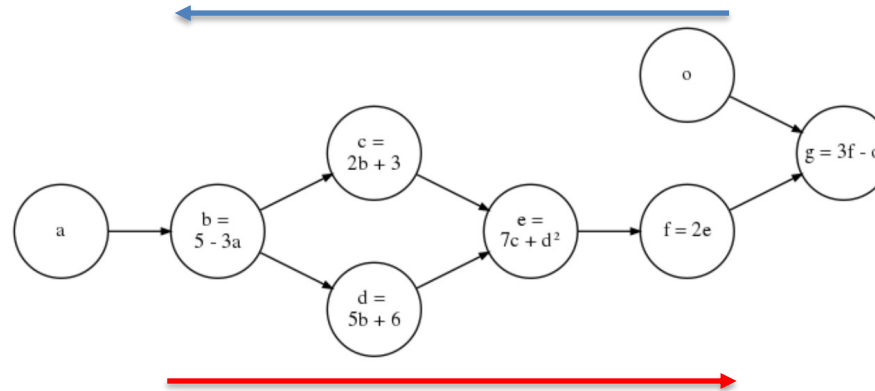
$$\mathbf{O}_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}, \mathbf{a}_1 = \sigma(\mathbf{O}_1)$$

$$\mathbf{O}_{l+1} = \mathbf{W}_{l+1} \mathbf{a}_l + \mathbf{b}_{l+1}$$

$$\hat{\mathbf{y}} = \sigma(\mathbf{a}_l) = f_1 \circ f_2 \circ \dots \circ f_L$$

# Forward & Backward Pass – How do models infer & learn?

- Outputs generated by *passing the inputs through the parametrized model*  
→ forward pass  $\hat{y} = f(x; \theta)$ , where  $f$  can be any complex, nested function with a set of learnable parameters  $\theta = \{w, b\}$
- Training: Compute the gradients of weights to iteratively update them  
→ we need *partial derivatives* of loss w.r.t. all learnable params  $\frac{\partial L}{\partial w_i}; \frac{\partial L}{\partial b_i}$   
→ use *chain rule* to recursively *propagate back* the loss through the model



# Loss Function Types

- Regression:  
Mean Squared Error (MSE)  
Mean Absolute Error (MAE)

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

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

- Classification: Cross-Entropy Loss (Log-likelihood)

$$\text{Cross Entropy Loss}_b = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- Domain-informed loss function

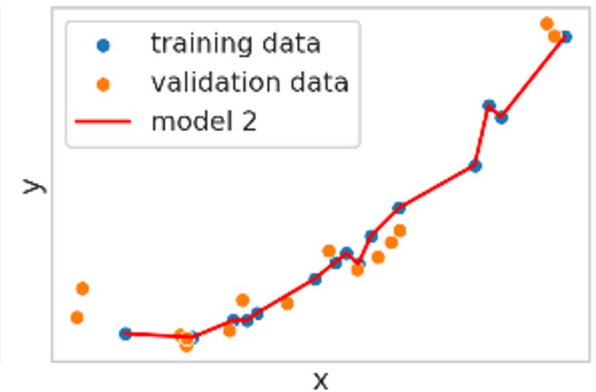
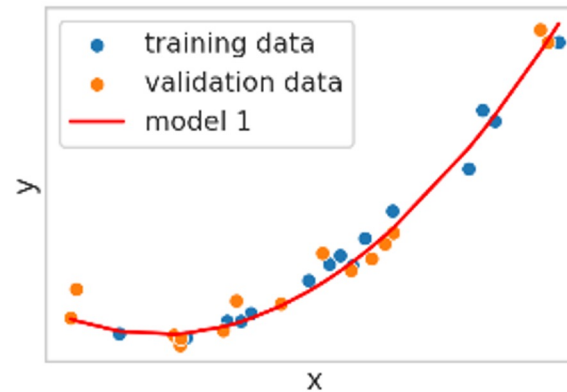
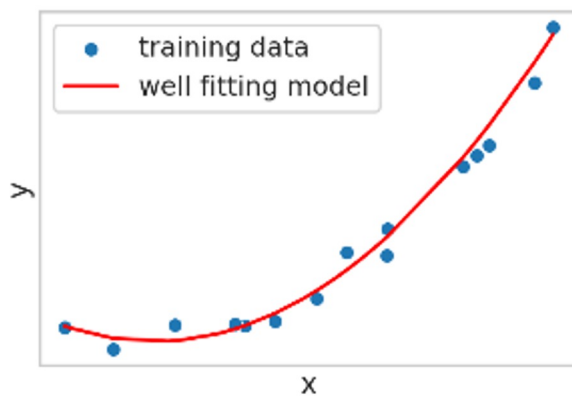
$$\text{Cross Entropy Loss}_m = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m y_{ij} \log(\hat{y}_{ij})$$

- Coherence with physical principles, e.g. based on governing equations
- Example: PINN (Physics Informed Neural Network)

# Questions?

# Evaluating Machine Learning Algorithms

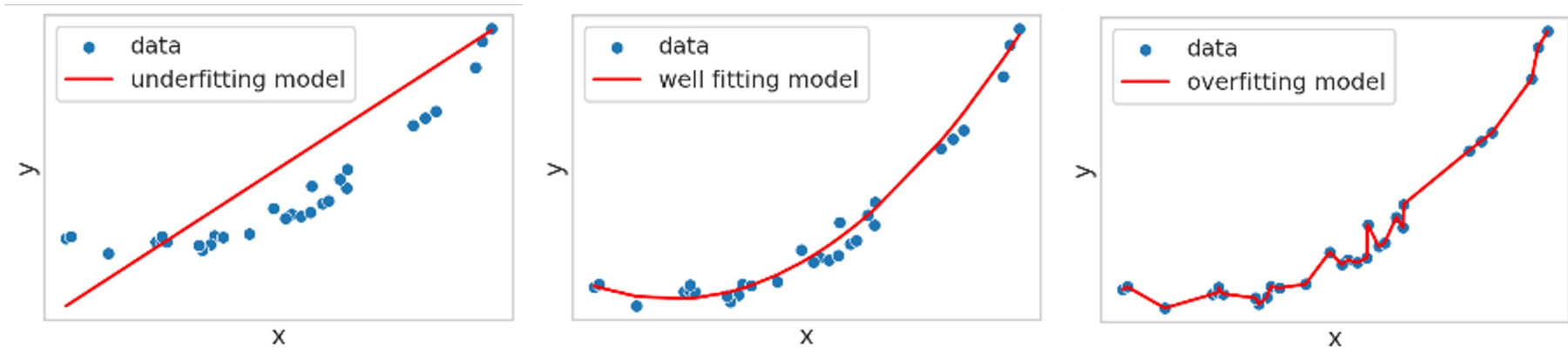
What do you think:  
Which model performs  
better overall?



# Evaluating Machine Learning Algorithms

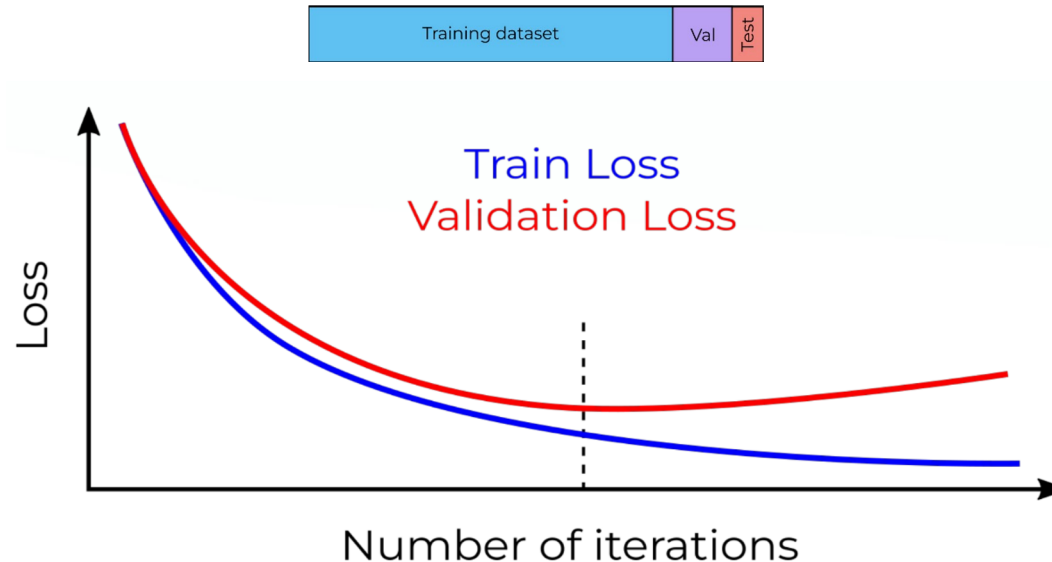
## Identifying Overfitting

- Neural network eventually learns to reproduce training data exactly  $\rightarrow$  inefficient + does *not* generalize well to unseen data
- *Bias-variance tradeoff*



# Solution I – Training-Validation-Test Split

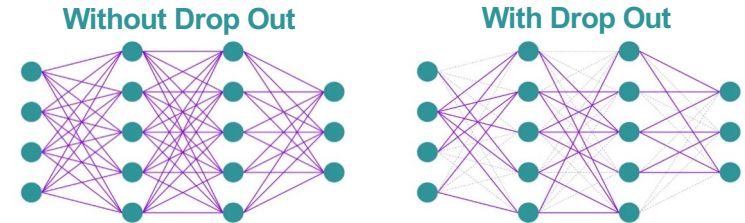
- Separate validation data (typically 10-20%)
- After each training step, calculate the loss function using *only* the validation set



# Solution II – Regularization Techniques

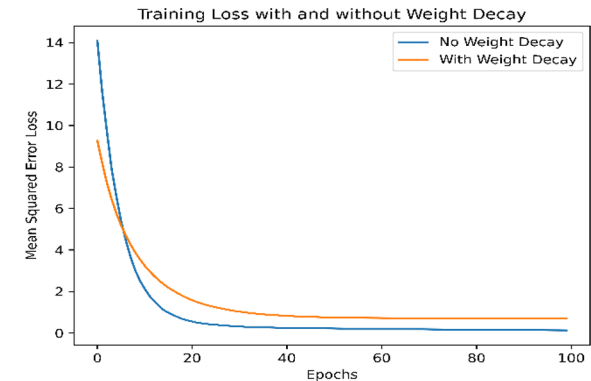
## ■ Dropout

- Neurons are randomly dropped during training
- It prevents over-dependencies on particular neurons



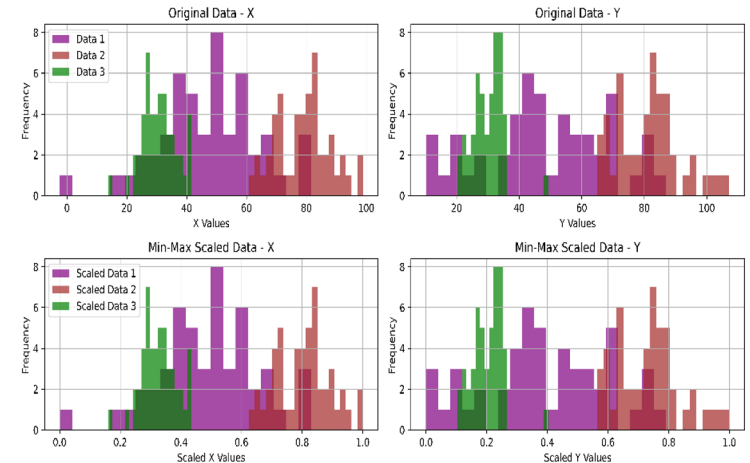
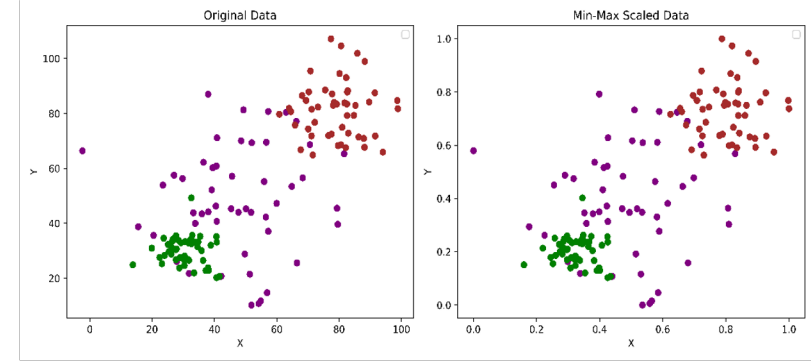
## ■ Weight Decay

- Penalize large weight values
- Promote sparsity in weight matrices



# Improve Model Training – Normalization

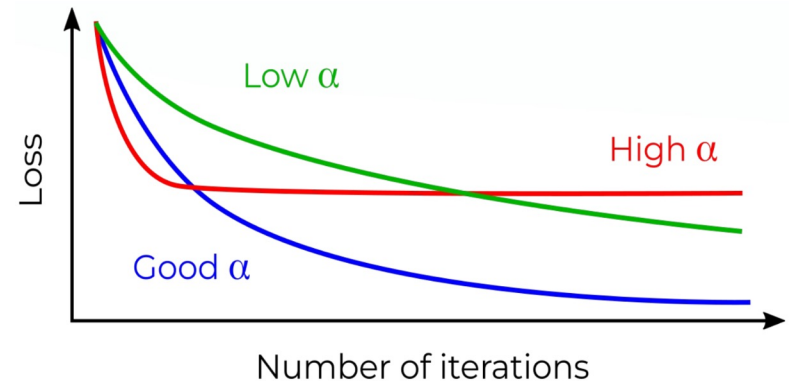
- General Idea
  - Ensure features of similar scale
  - Normalize skewed data distributions
  - Helps in convergence of model
  - Smooths out loss landscape
- Common Normalization Methods
  - Mix-Max Scaler
  - Z-score Normalization
  - Log Transform
  - Robust Scaler



# Improve Model Training – Hyper-parameter Optimization

- Common Hyper-parameters

- Learning rate  $\alpha$
- Batch size  $B$
- Epoch size
- Activation function(s)
- # of neurons & layers
- Dropout rate

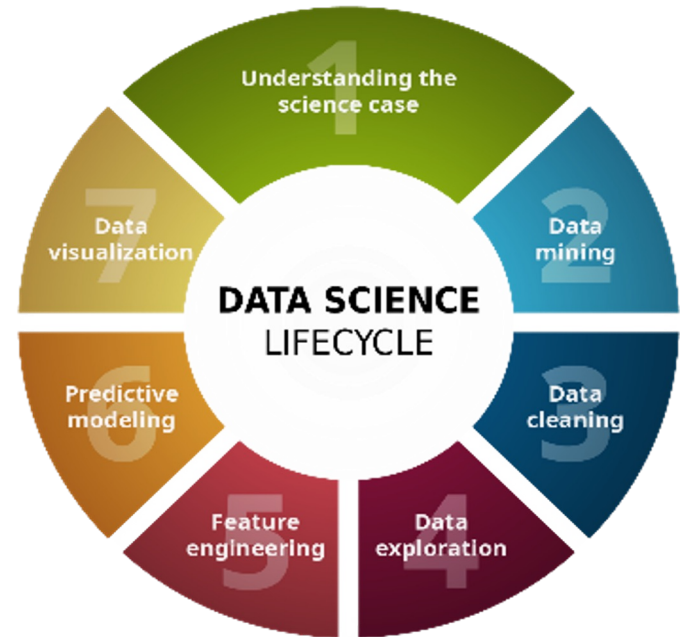


- General aspects to consider

- Batching  $\rightarrow$  stochastic, full vs. mini batch: how “big” is the mean loss?
- Choice of model  $\rightarrow$  what should the model learn, what are good candidates?
- Choice of loss function & optimizer  $\rightarrow$  how well can the model learn?

# Typical machine learning project cycle

- Common thinking: I will spend a lot of time in model development
- Reality: 90% of time is spent in data science parts
- Always set your code up for an iterative process
- Always follow best practices



# What is Needed for Successful Machine Learning Projects?

- Data that holds the necessary information and of good quality
  - “Garbage in, garbage out”
  - think in advance: How much data do you have? Can you obtain more?
- Model: find the right model for your task (we will cover some in the course)
  - what *data structures / properties* should be “covered”?
  - what can the *model represent*?
- Computational resources: ML training & inference relies on GPUs
- Storage & data through-put (model size, VRAM)

# Python Libraries for Data Science & ML/DL

- Foundational data science
  - **NumPy, Pandas, SciPy** → exploratory data evaluation, sorting & filtering, feature engineering, optimization & simulation
  - **Visualization** → Matplotlib, Seaborn

- Machine Learning

- [Scikit-learn](#), **XGBoost** → classical machine learning algorithms



 PyTorch

- Deep Learning

- [PyTorch](#) (**Tensorflow**) → highly flexibel, fast, abstract API
  - [JAX](#) → modern, functional, very fast DL overlay on NumPy
  - [Hugging Face Transformers](#) → open-source NLP models



**Hugging Face**

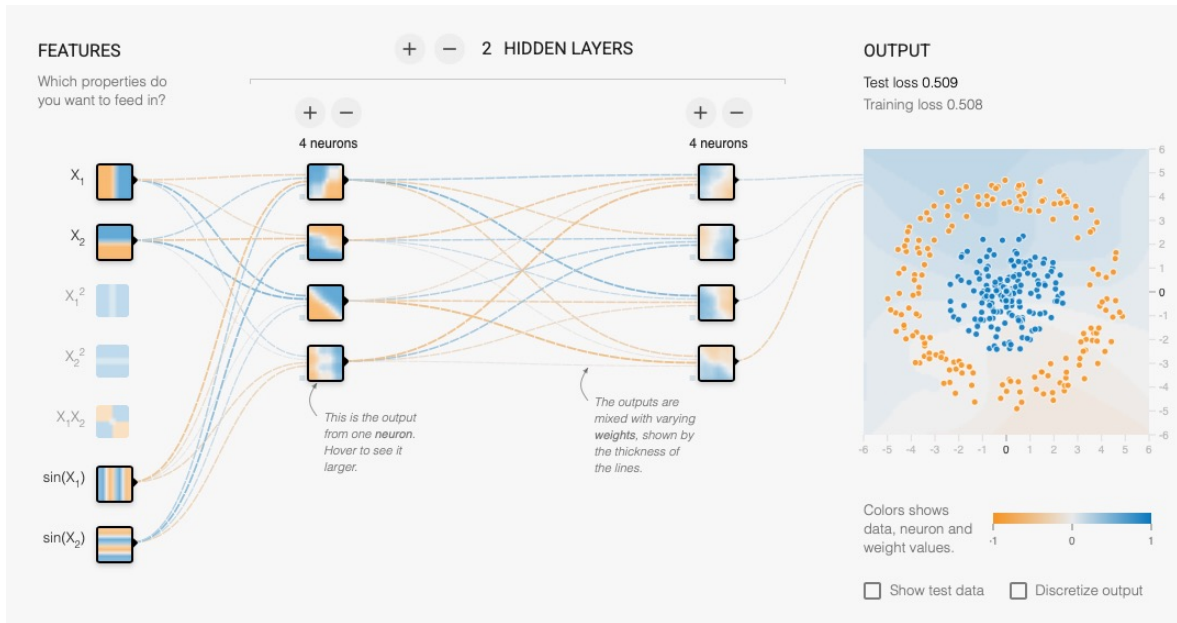


# Thank you for your attention!

## Questions? Thoughts?

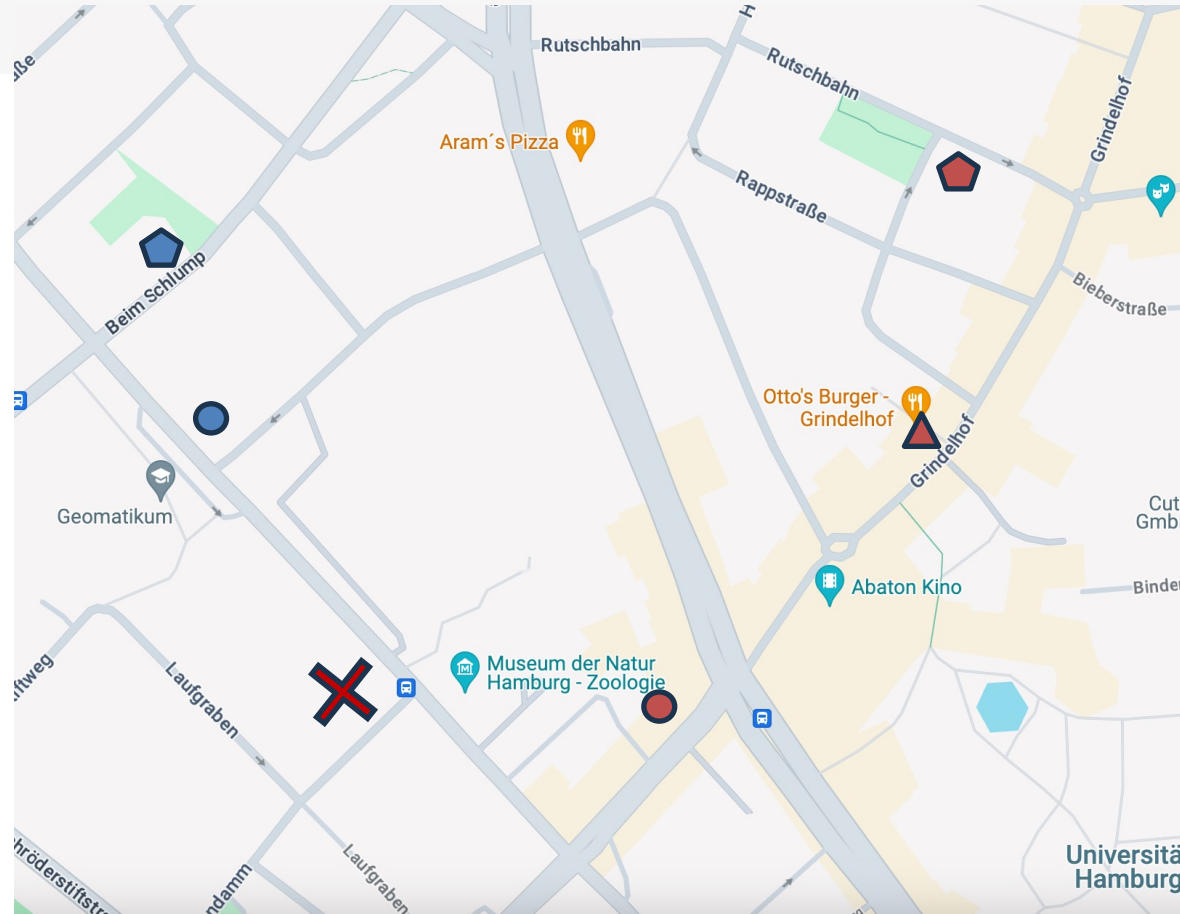
# Bonus Material – Intuitive ML Understanding

## Tensorflow ML Playground



# Lunch Options

- Uni Hamburg Mensa
- ⬠ Rucola e Parma, Italian
- Quan19, Vietnamese
- ▲ Ruffs Burger
- ⬠ Hindukusch, Afghan
- Many more...
  
- ✗ We are here (DKRZ)



**See you at 2 PM!**