MachineLearnAthon - Microlecture Neural Networks

MachineLearnAthon
A project Co-funded by the Erasmus+ programme of the European Union

Lecture outline

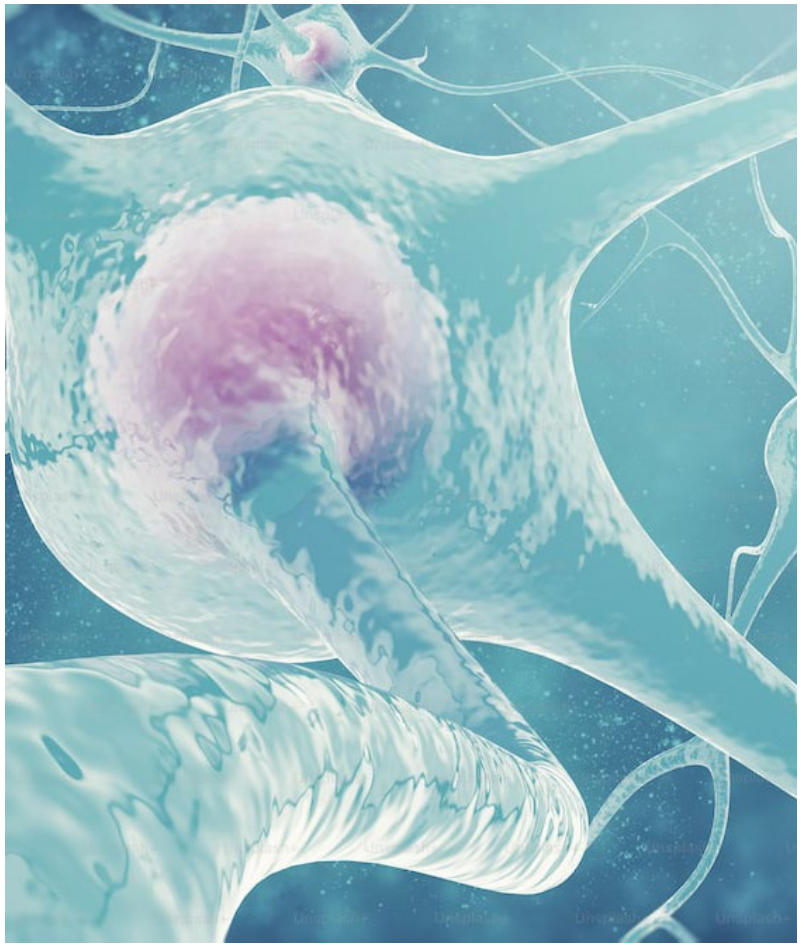
- Neural network basics
- Neural network training
- Architecture and hyperparameters
- Applications of neural networks for regression

Lecture outline

- **Neural network basics**
- Neural network training
- Architecture and hyperparameters
- Applications of neural networks for regression

Artificial neural networks (ANNs)

- ANNs are:
  - inspired by the neural networks in human brains



[1] Figure taken from: https://unsplash.com/photos/neurons-of-the-nervous-system-3d-illustration-of-nerve-cells-aNna7e9jDCE (Accessed on 22.06.2022)

[2] Hornik, Kurt; Tinchcombe, Maxwell; White, Halbert (1989). Multilayer Feedforward Networks are Universal Approximators (PDF). Neural Networks. Vol. 2. Pergamon Press. pp. 359–366.

Biological and artificial neurons

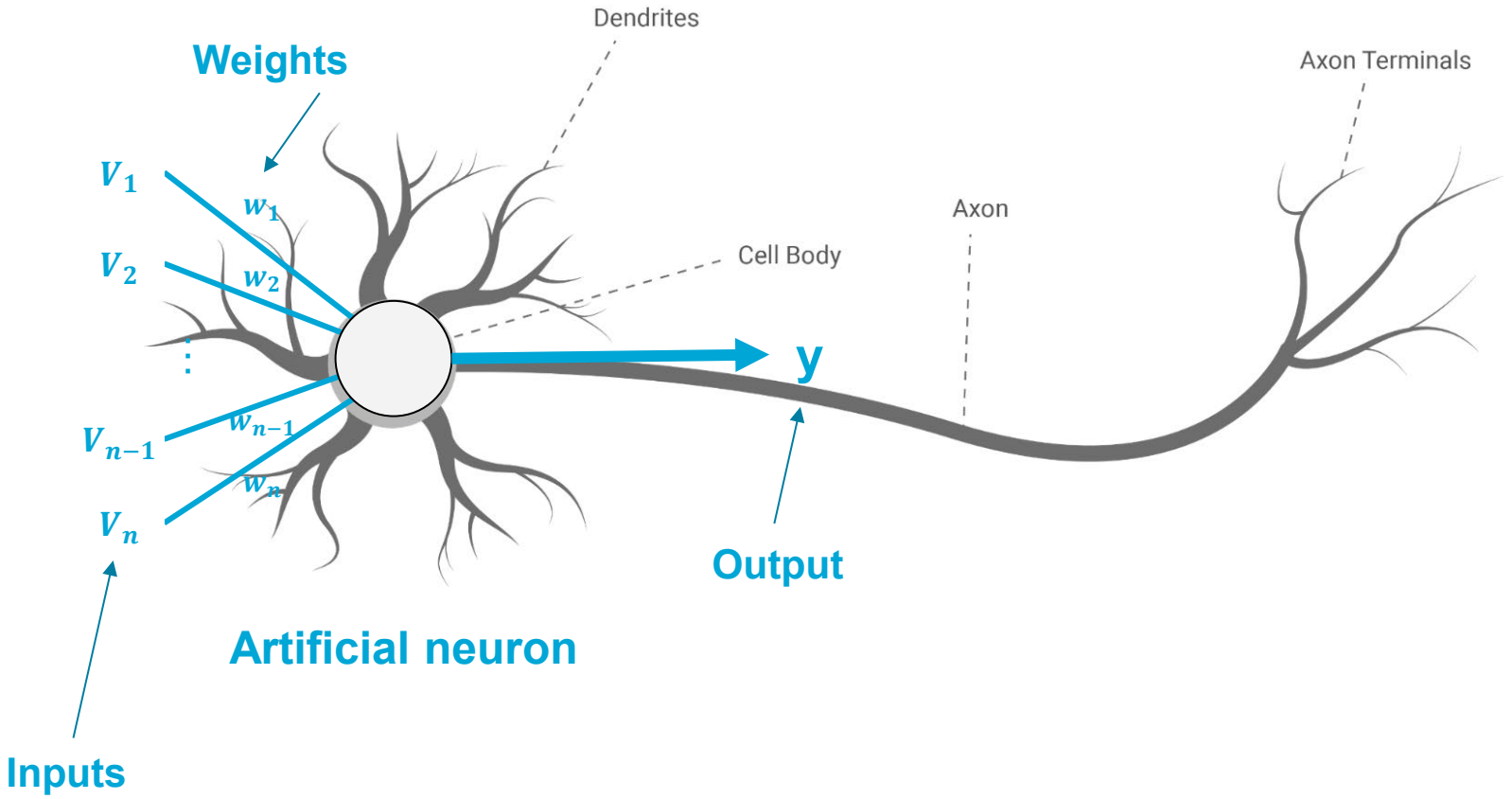
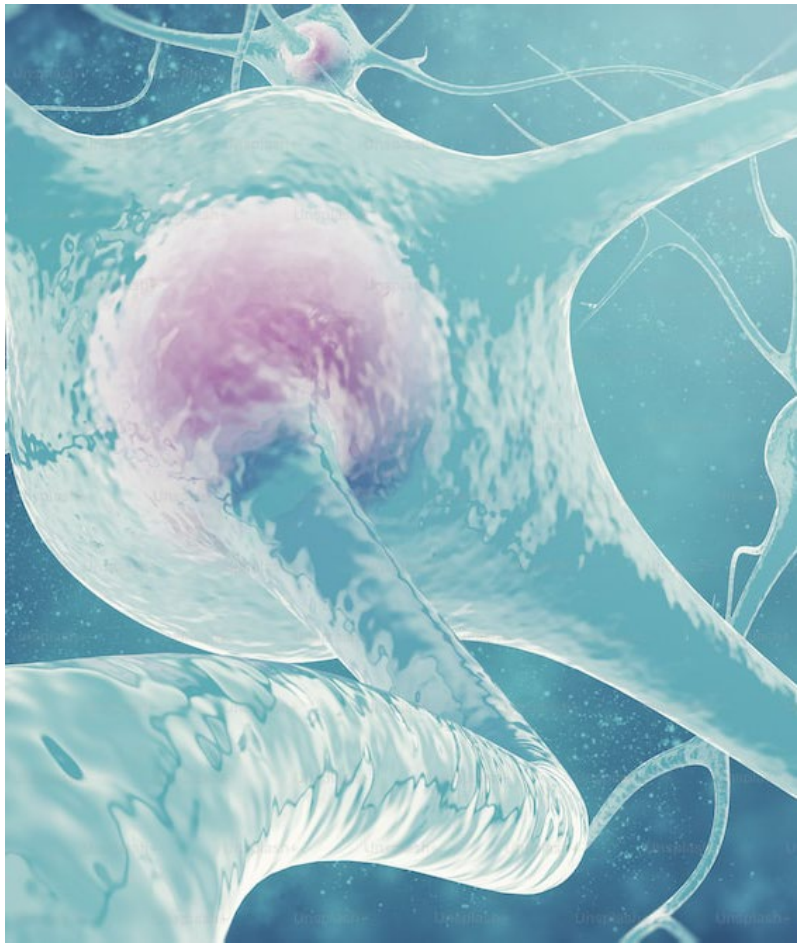


Figure taken from: : https://machineslearn.co.uk/wp-content/uploads/2018/01/biological-neuron-1024x438.png

Artificial neural networks (ANNs)

- ANNs are:
  - inspired by the neural networks in human brains
  - flexible machine learning models that can (theoretically) approximate any continuous function to a given accuracy, i.e., they are universal approximators [2]
  - able to handle large input and output dimensions and big training data
  - very commonly used in various supervised and unsupervised learning applications
- In this lecture, we will focus on ANNs for regression problems



[1] Figure taken from: https://unsplash.com/photos/neurons-of-the-nervous-system-3d-illustration-of-nerve-cells-aNna7e9jDCE (Accessed on 22.06.2022)

[2] Hornik, Kurt; Tinchcombe, Maxwell; White, Halbert (1989). Multilayer Feedforward Networks are Universal Approximators (PDF). Neural Networks. Vol. 2. Pergamon Press. pp. 359–366.

Biological and artificial neurons

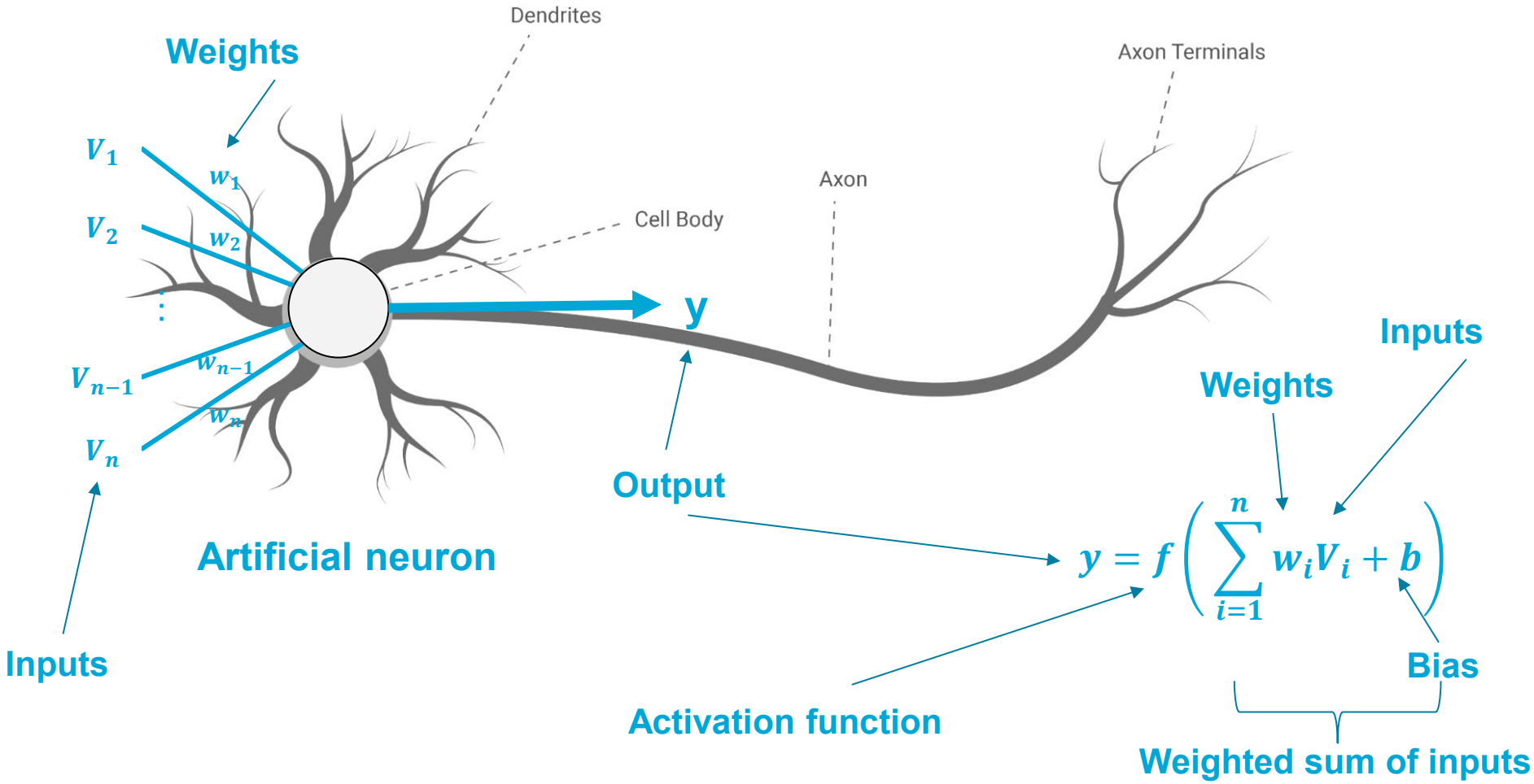
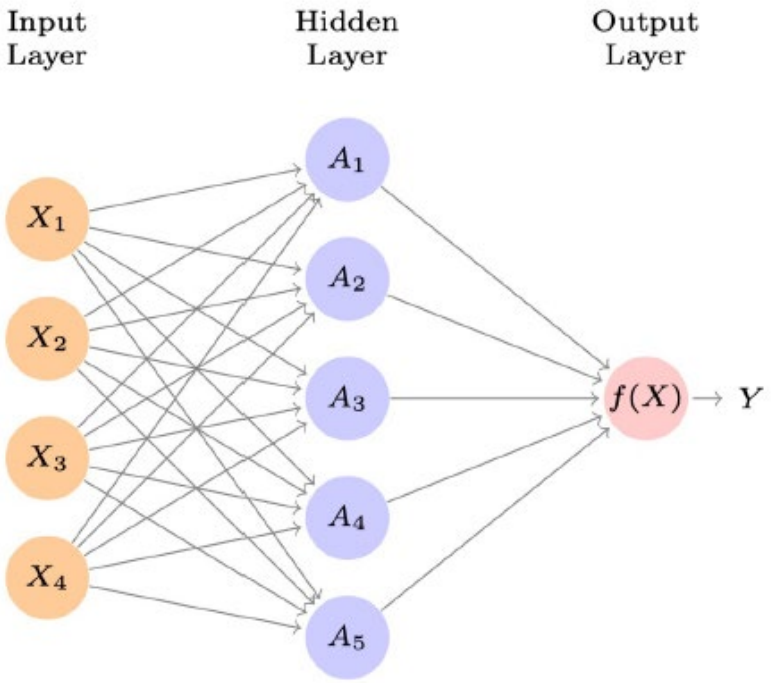


Figure taken from: : https://machineslearn.co.uk/wp-content/uploads/2018/01/biological-neuron-1024x438.png

Universal approximation theorem

- Assume:
  - A single hidden layer of arbitrary size (number of neurons)
  - A non-polynomial continuous activation function, such as the sigmoid function, in the hidden layer
  - Linear activation in the output layer
- It can be demonstrated that the neural network has the capability to approximate any function $f(x): R^n \mapsto R$ with arbitrary precision



Nomenclature in a feed-forward ANNs

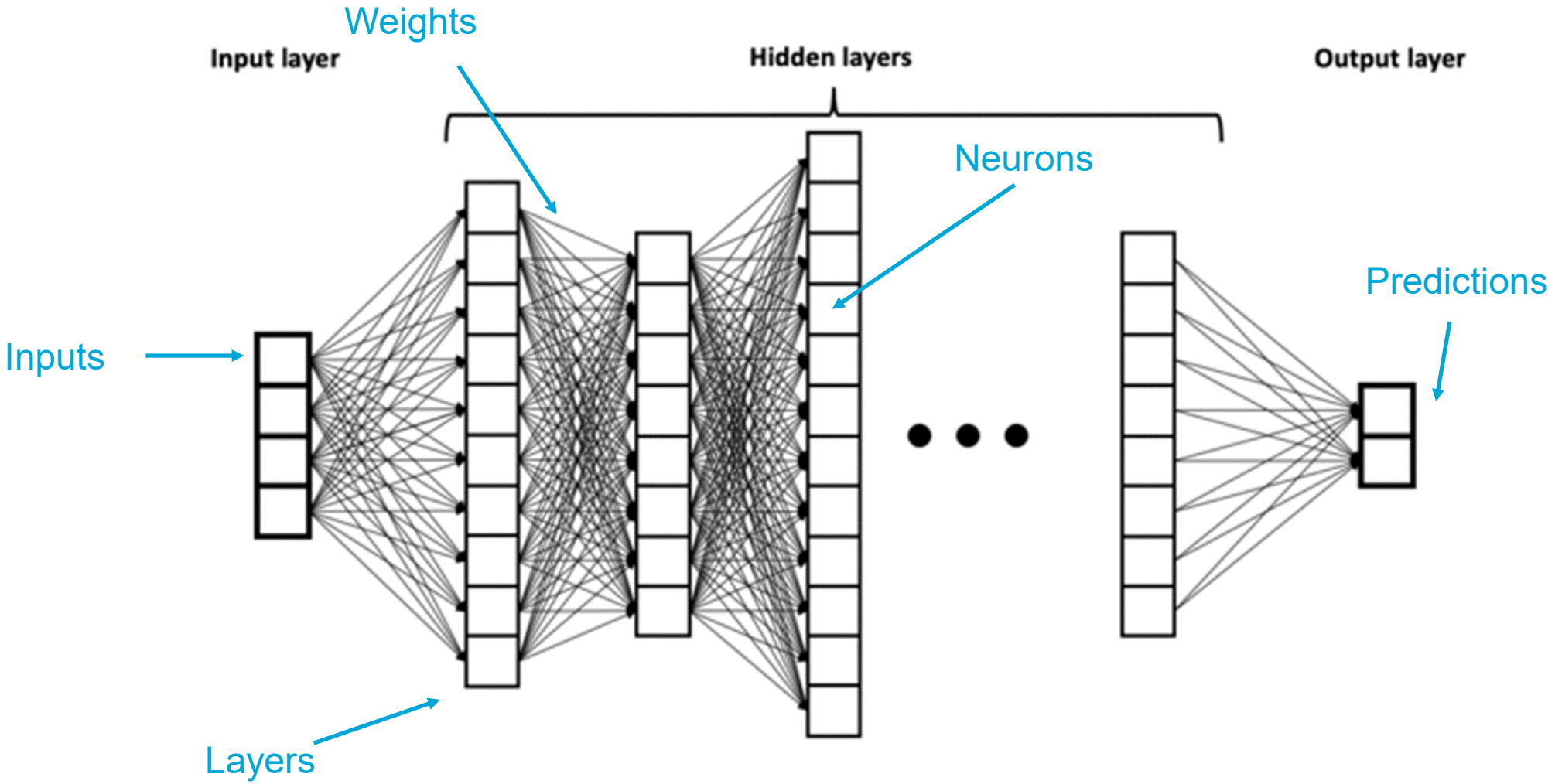


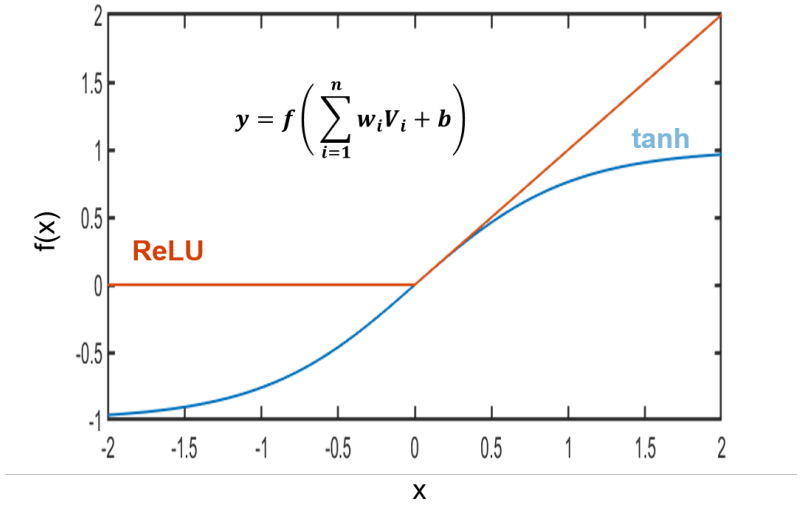
Image: https://upload.wikimedia.org/wikipedia/commons/thumb/2/2f/Example_of_a_deep_neural_network.png/640px-Example_of_a_deep_neural_network.png

Activation function

- In both artificial and biological neural networks, a neuron does not just output the bare input it receives
- Action potential firing in the human brain inspired activation functions in ANNs [1]
- There exist several possible activation functions, e.g.,
  - **Hyperbolic tangent function**

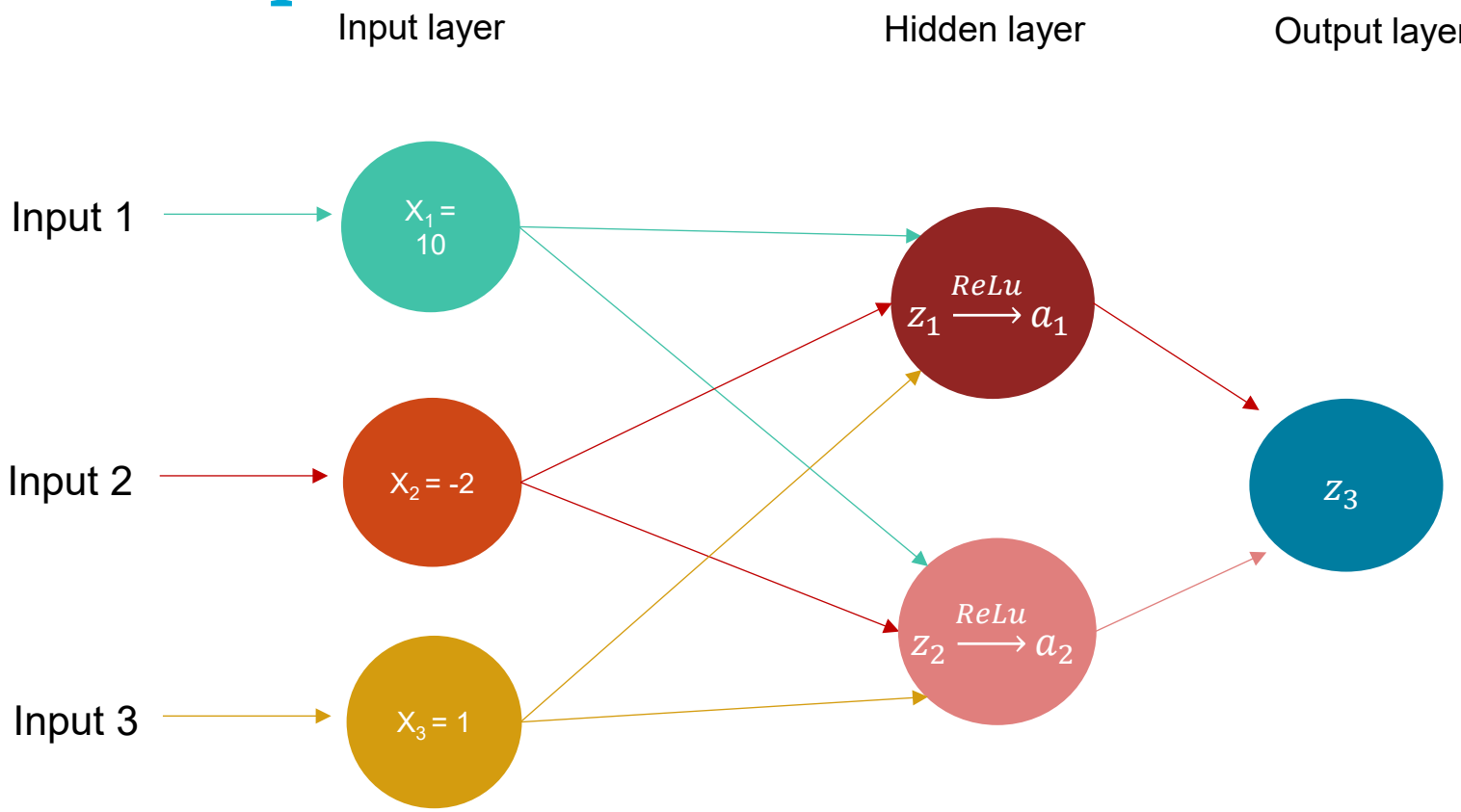
    f(x) = tanh(x)
  - Saturates quickly (For x ≥ 5 almost constant, gradient vanishes)
  - **Rectified Linear Unit (ReLU)**

    f(x) = max(0,x)
  - Used in deep ANNs

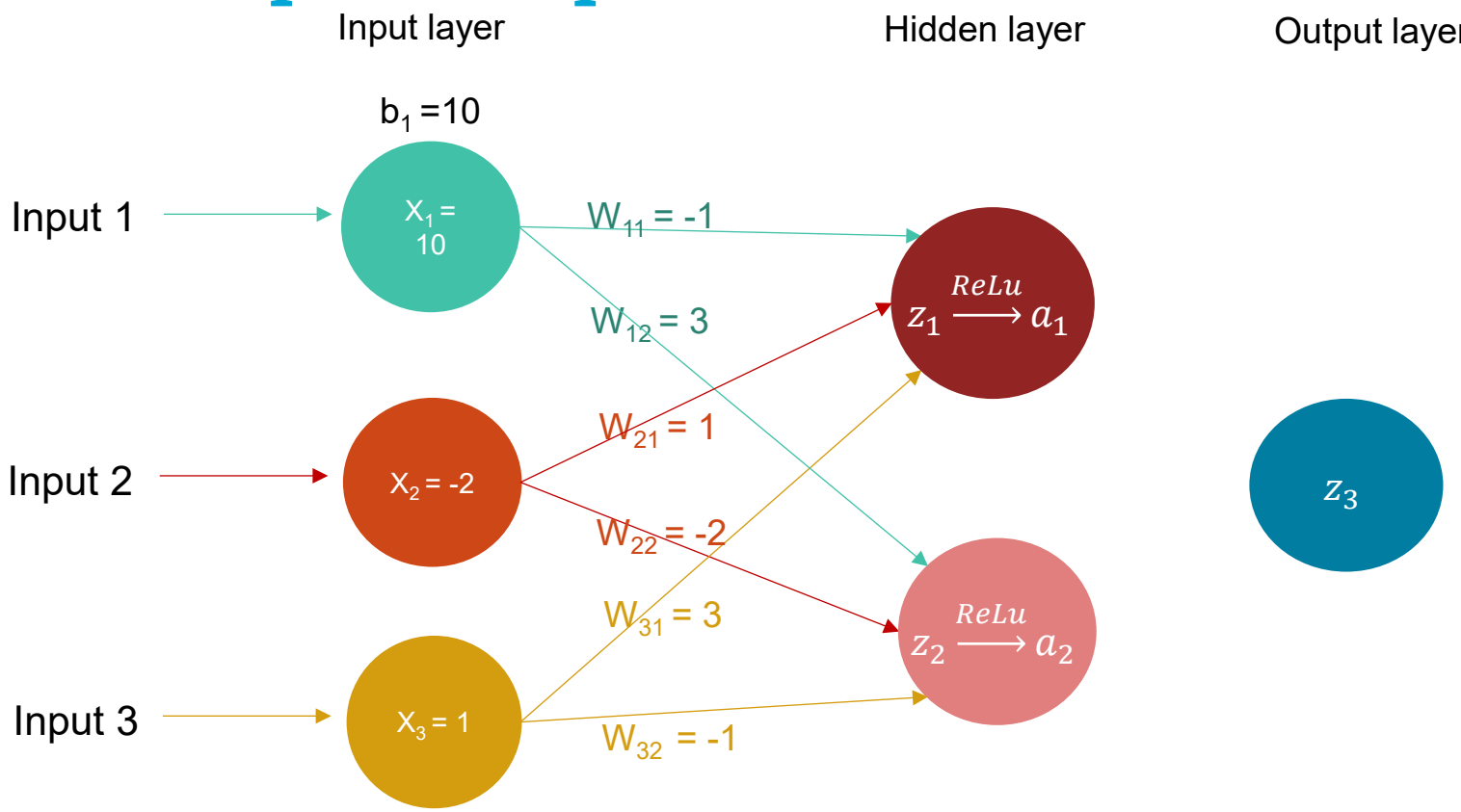


[1] Montesinos López, O.A., Montesinos López, A., Crossa, J. (2022). Fundamentals of Artificial Neural Networks and Deep Learning. In: Multivariate Statistical Machine Learning Methods for Genomic Prediction. Springer, Cham. https://doi.org/10.1007/978-3-030-89010-0_10

Simple example



Simple example - step 1 & 2



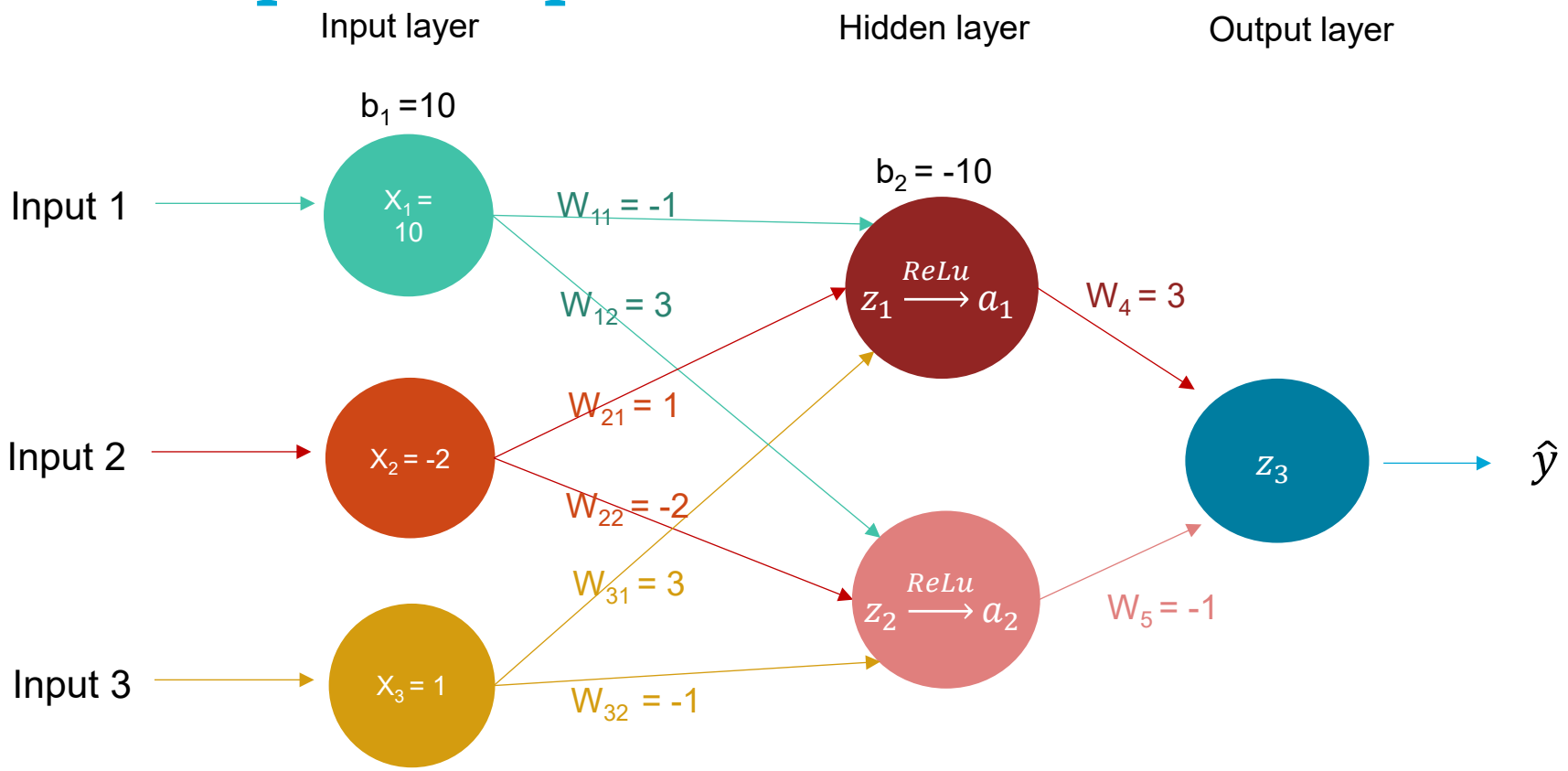
Step 1:

$$z_1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 + w_{31} \cdot X_3 + b_1 = 1$$
$$a_1 = ReLU(z_1) = 1$$

Step 2:

$$z_2 = w_{12} \cdot X_1 + w_{22} \cdot X_2 + w_{32} \cdot X_3 + b_1 = 43$$
$$a_2 = ReLU(z_2) = 43$$

Simple example - step 3 & 4



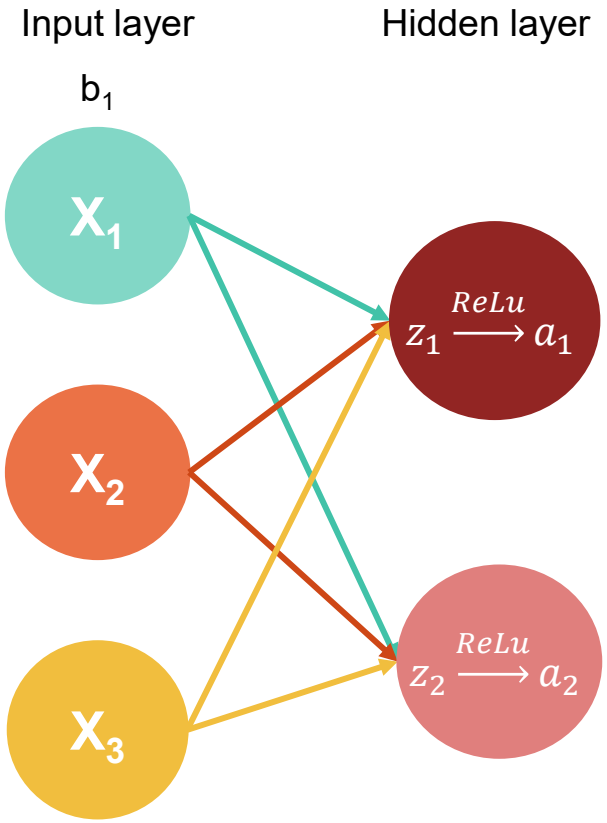
Step 3:

$$z_3 = w_4 \cdot a_1 + w_5 \cdot a_2 + b_2 = -50$$
$$\hat{y} = z_3 = 0$$

Step 4:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - \hat{y})^2$$

Matrix representation of neural networks

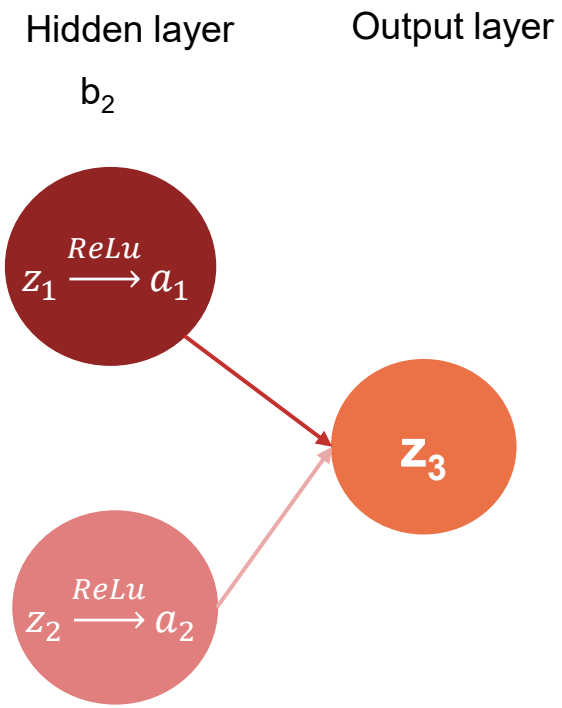


- Taking the input – hidden layer section
  - Computation of the values to pass to the activation function (z) follows:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_{i11} & w_{i12} & w_{i13} \\ w_{i21} & w_{i22} & w_{i23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_1$$

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_{i11} \cdot x_1 + w_{i12} \cdot x_2 + w_{i13} \cdot x_3 + b_1 \\ w_{i21} \cdot x_1 + w_{i22} \cdot x_2 + w_{i23} \cdot x_3 + b_1 \end{pmatrix}$$

Matrix representation of neural networks

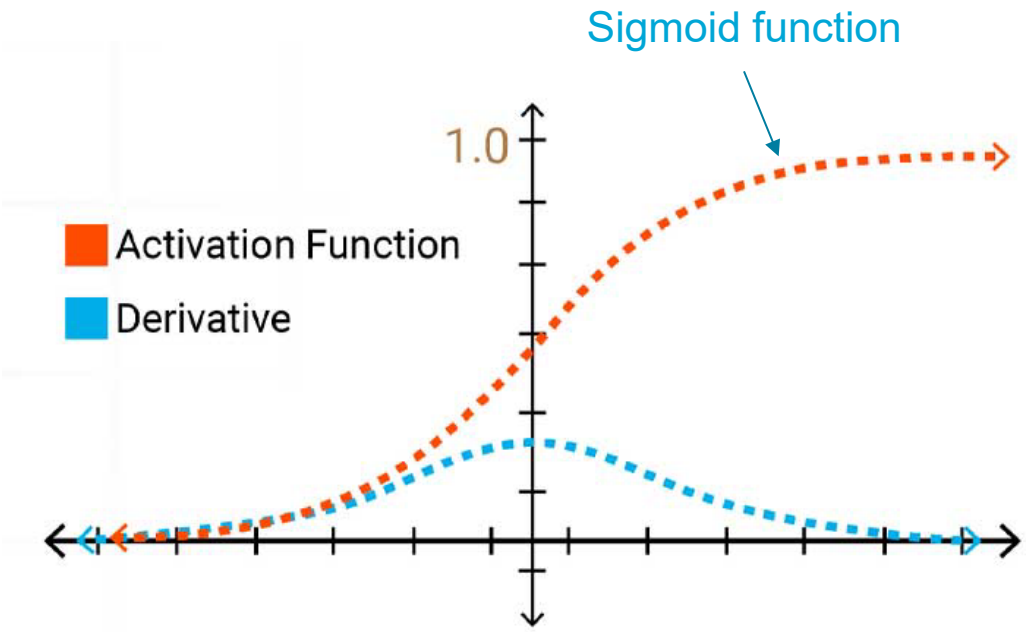


- Taking the hidden layer – output section
  - Computation of the values to pass to obtain the output (z_3) from the activation function outputs (a) follows:

$$z_3 = (w_{h11} \quad w_{h12}) \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} + b_2 = (w_{w11} \cdot a_1 + w_{h12} \cdot a_2 + b_2)$$

Choosing activation functions for hidden layers

- Activation function choice is influenced by the problem type [1].
- Sigmoid/Logistic and Tanh functions have bounded outputs (between 0 and 1 for Sigmoid/Logistic and between -1 and 1 for Tanh) which can lead to gradient saturation, contributing to the vanishing gradient issue [2].
- ReLU (Rectified Linear Unit) has become popular due to its non-linearity and avoidance of gradient saturation. However, it has its own limitation, the "dying ReLU" problem.

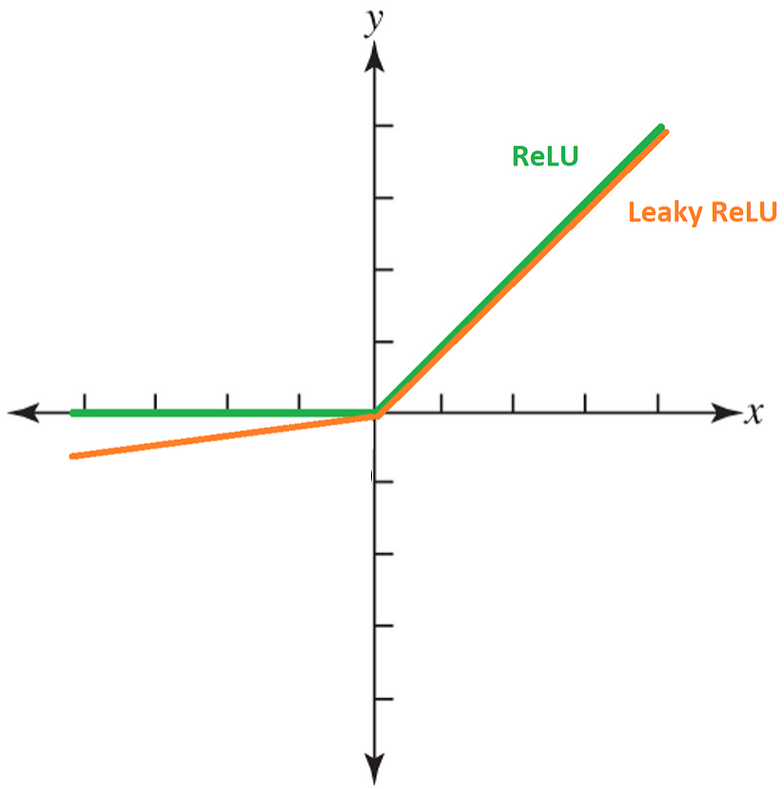


[1] Baheti, P. (2021) Activation functions in Neural networks [12 Types & Use cases]. Available in: https://www.v7labs.com/blog/neural-networks-activation-functions (Accessed on 10.01.2024)
[2] https://www.engati.com/glossary/vanishing-gradient-problem

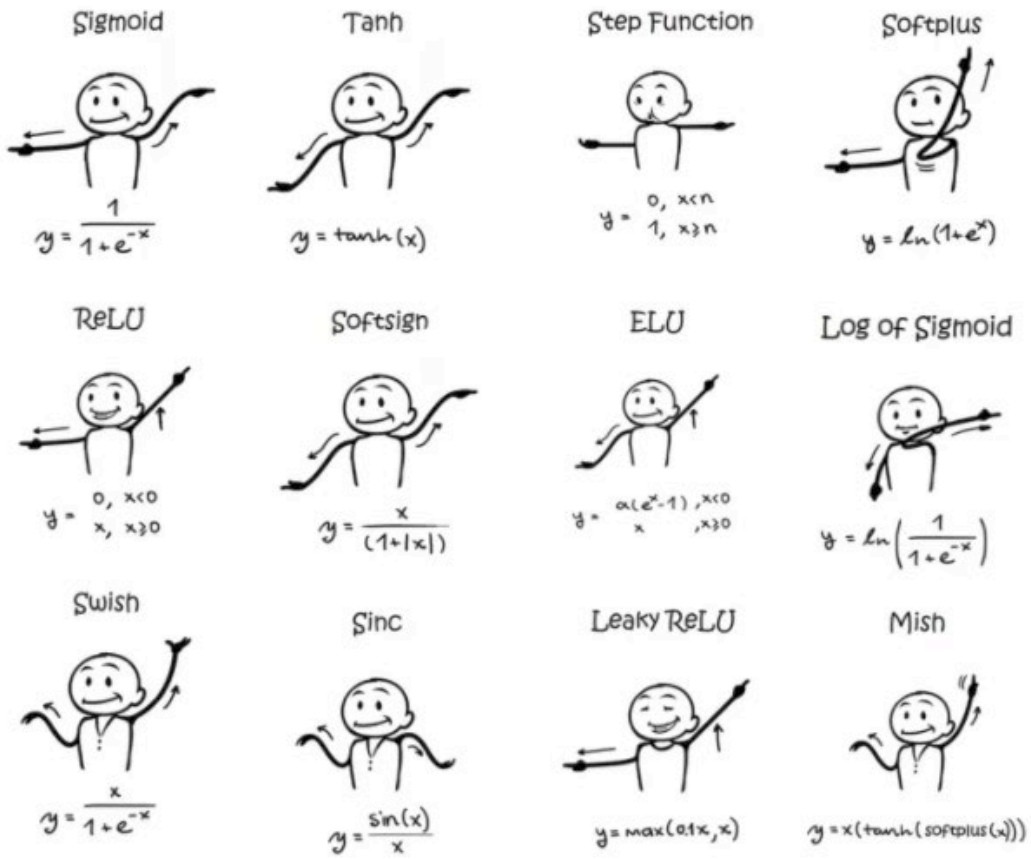
Co-funded by the Erasmus+ programme of the European Union

Dying ReLU

- "Dying ReLU" is a common issue in deep neural networks with the Rectified Linear Unit (ReLU) activation.
- It occurs when ReLU neurons consistently output zero, preventing them from learning.
- This happens because of zero gradients during backpropagation when inputs to ReLU are persistently negative.
- Variants like Leaky ReLU and PReLU were introduced to mitigate this problem by allowing learning for negative inputs.



Large variation of activation functions in deep learning



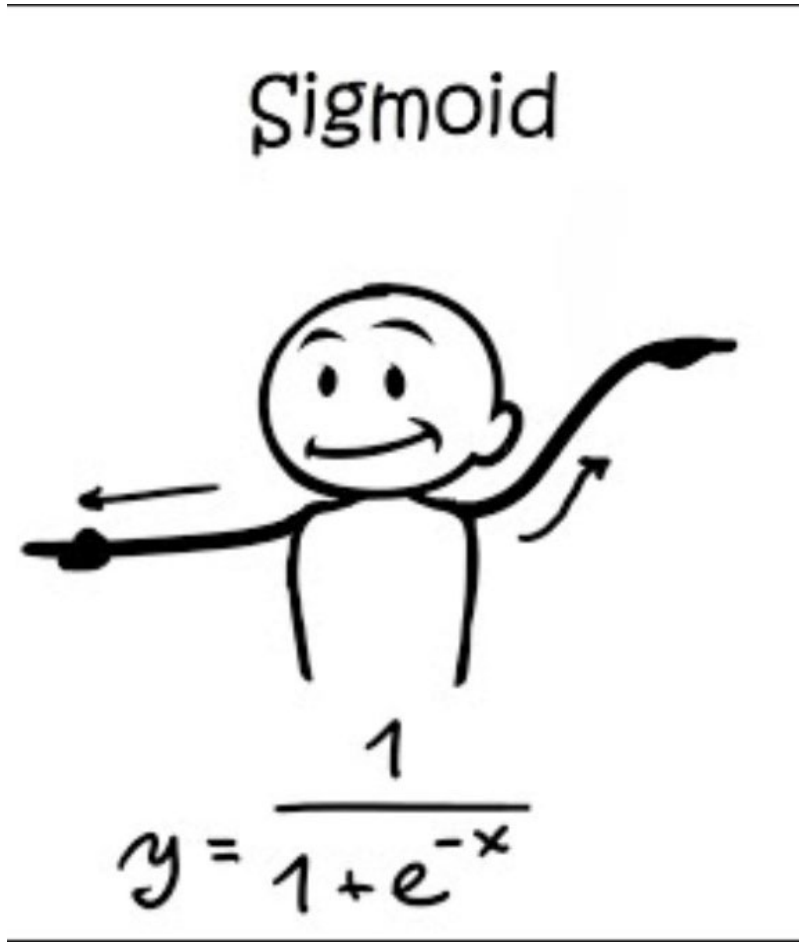


Image taken from: theaidream.com Available in: https://www.theaidream.com/post/an-overview-of-activation-functions-in-deep-learning

Choosing activation functions for **output** layers

- For Regression tasks: Linear activation function is used.
  - Produces continuous output for numerical predictions.
- For Binary Classification: Sigmoid/Logistic activation function is employed.
  - Squashes output between 0 and 1, ideal for binary decisions (e.g., yes/no).
- For Multiclass Classification: Softmax activation function is utilized.
  - Assigns exclusive class probabilities, ensuring the sum equals 1.
- For Multilabel Classification: Sigmoid activation function is chosen.
  - Allows activation of multiple classes independently for cases with multiple labels.
- ReLU (Rectified Linear Unit) should not be used in output layers because of its unbounded nature, which can lead to instability during training and make it challenging to interpret the model's predictions.

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **165-026-438**

Or scan the QR code

Join at: **vevox.app** ID: **165-026-438**

Question slide

Which activation function is typically used for Regression tasks?

A) Sigmoid/Logistic

0%

B) Tanh

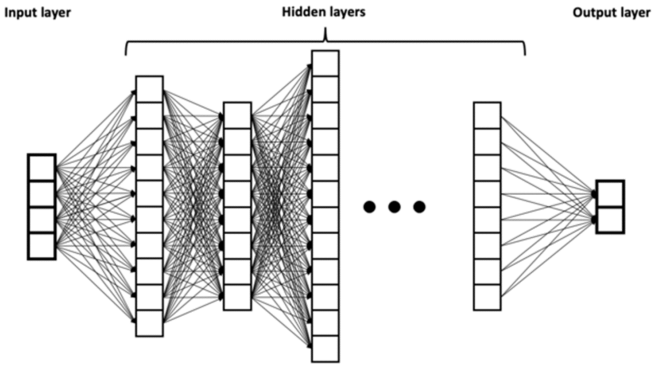
0%

C) Linear

0%

D) Softmax

0%



Which activation function is typically used for Regression tasks?

Join at: **vevox.app** ID: **165-026-438**

Question slide

What is the primary advantage of using the Sigmoid/Logistic activation function in Binary Classification?

A) It produces continuous output — 0%

B) It squashes the output between 0 and 1 — 0%

C) It allows for multiple class activations — 0%

D) It prevents the vanishing gradient problem — 0%

What is the primary advantage of using the Sigmoid/Logistic activation function in Binary Classification?

A) It produces continuous output

0%

B) It squashes the output between 0 and 1

97.3%

C) It allows for multiple class activations

2.7%

D) It prevents the vanishing gradient problem

0%

Join at: **vevox.app** ID: **165-026-438**

Question slide

Why is the Softmax activation function preferred for Multiclass Classification?

A) It produces continuous output — 0%

B) It allows for multiple class activations — 0%

C) It ensures that the sum of class probabilities equals 1 — 0%

D) It is suitable for binary decisions — 0%

Why is the Softmax activation function preferred for Multiclass Classification?

A) It produces continuous output — 0%

B) It allows for multiple class activations — 42.86 %

C) It ensures that the sum of class probabilities equals 1 — 57.14 % ✓

D) It is suitable for binary decisions — 0%

When is the Sigmoid activation function typically used?

A) In Regression tasks

0%

B) In Binary Classification

0%

C) In Multiclass Classification

0%

D) In Multilabel Classification

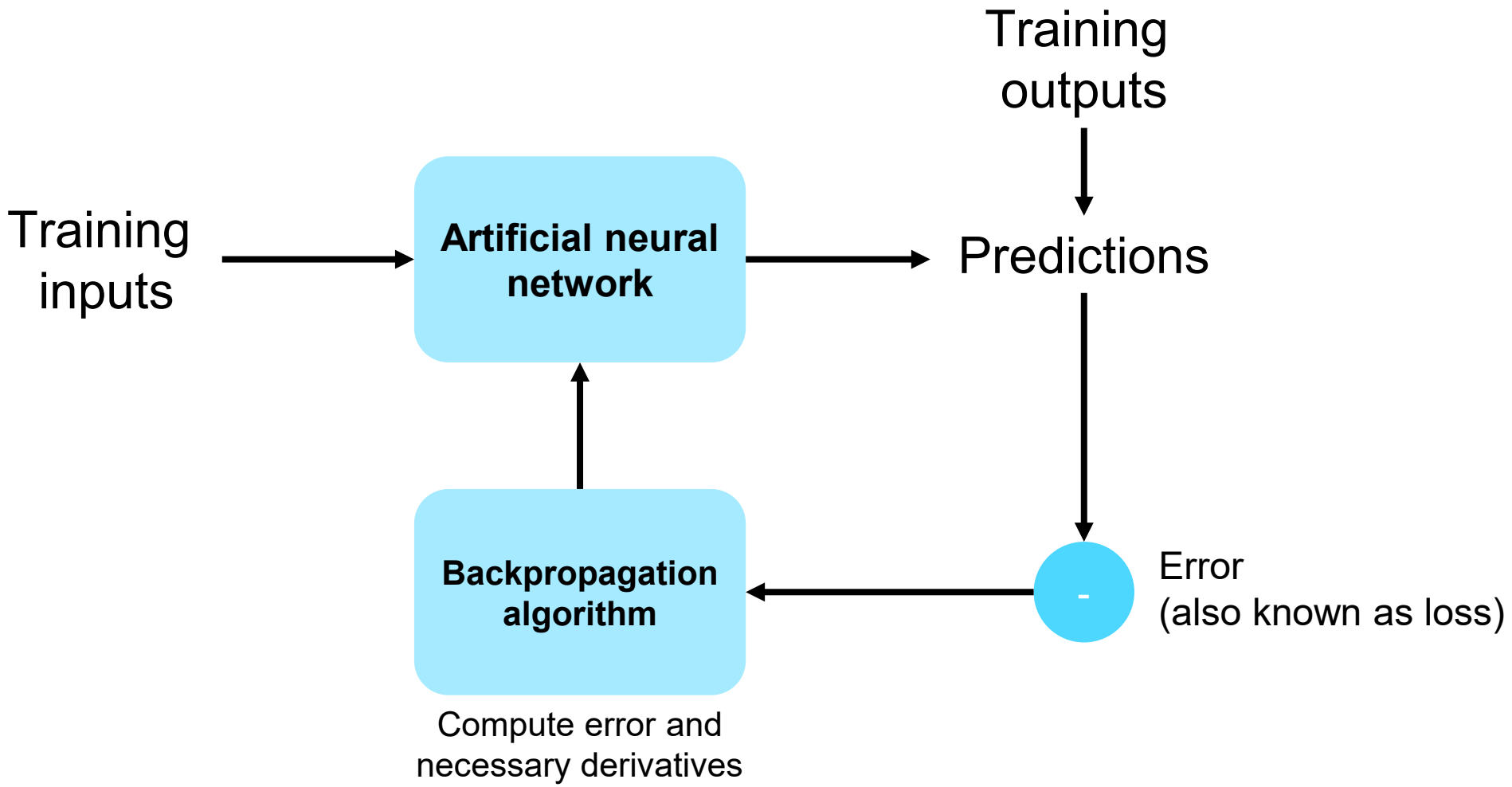
0%

When is the Sigmoid activation function typically used?

Lecture outline

- Neural network basics
- **Neural network training**
- Architecture and hyperparameters
- Applications of neural networks for regression

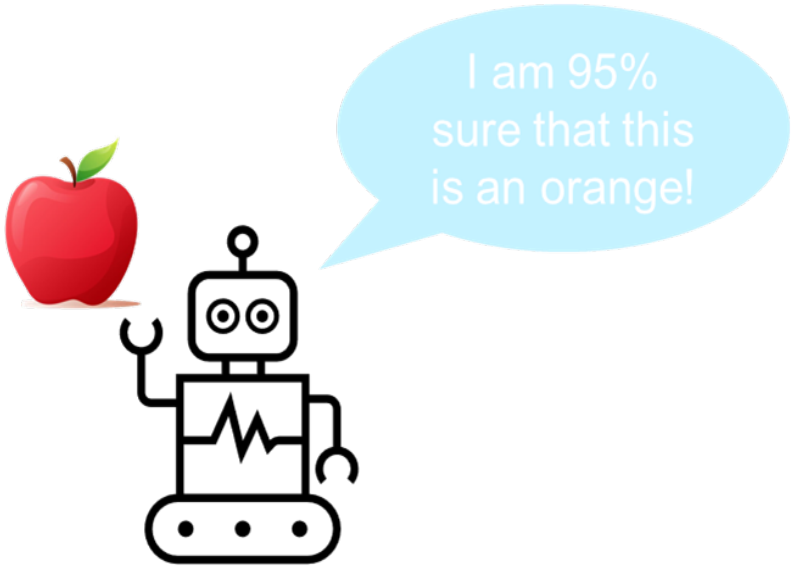
Training


Training outputs
Training inputs
Artificial neural network
Predictions
Backpropagation algorithm
-
Error
(also known as loss)
Compute error and necessary derivatives

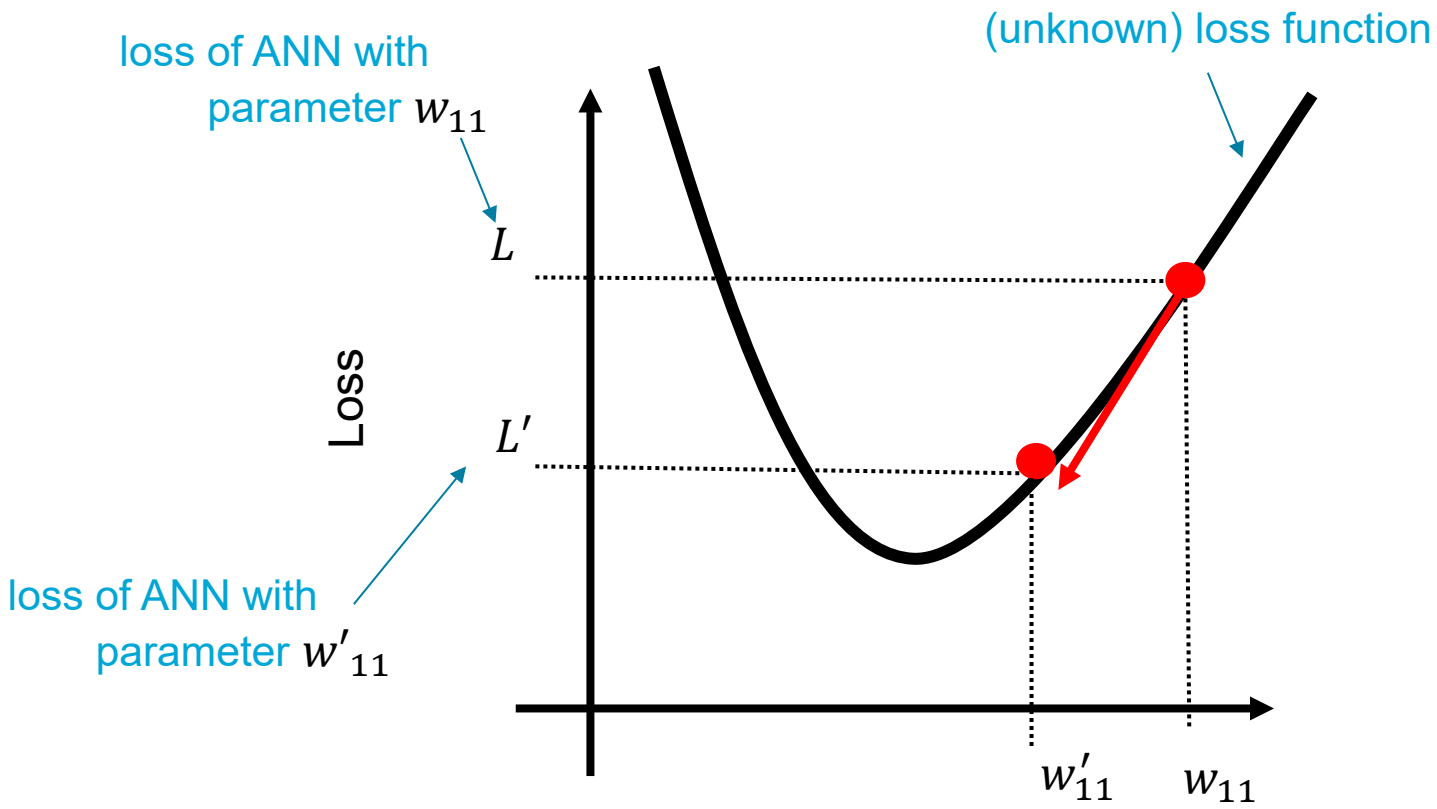
How does a loss function look like?

- Cross-Entropy Loss (Log Loss)
  - Type: Classification
  - Formula: $\mathcal{L}_{\mathrm{i}} = -\sum_{c=1}^{M} y_{i,c} \log(model_C(x_i))$ where M is number of classes, y is binary indicator (0 or 1) if class label C is the correct classification for observation i
- Mean Squared Error (MSE) or L2 loss
  - Type: Regression
  - Formula: $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - model(x_i)\right)^2$

    where N is the number of samples, y are labels and x are the features



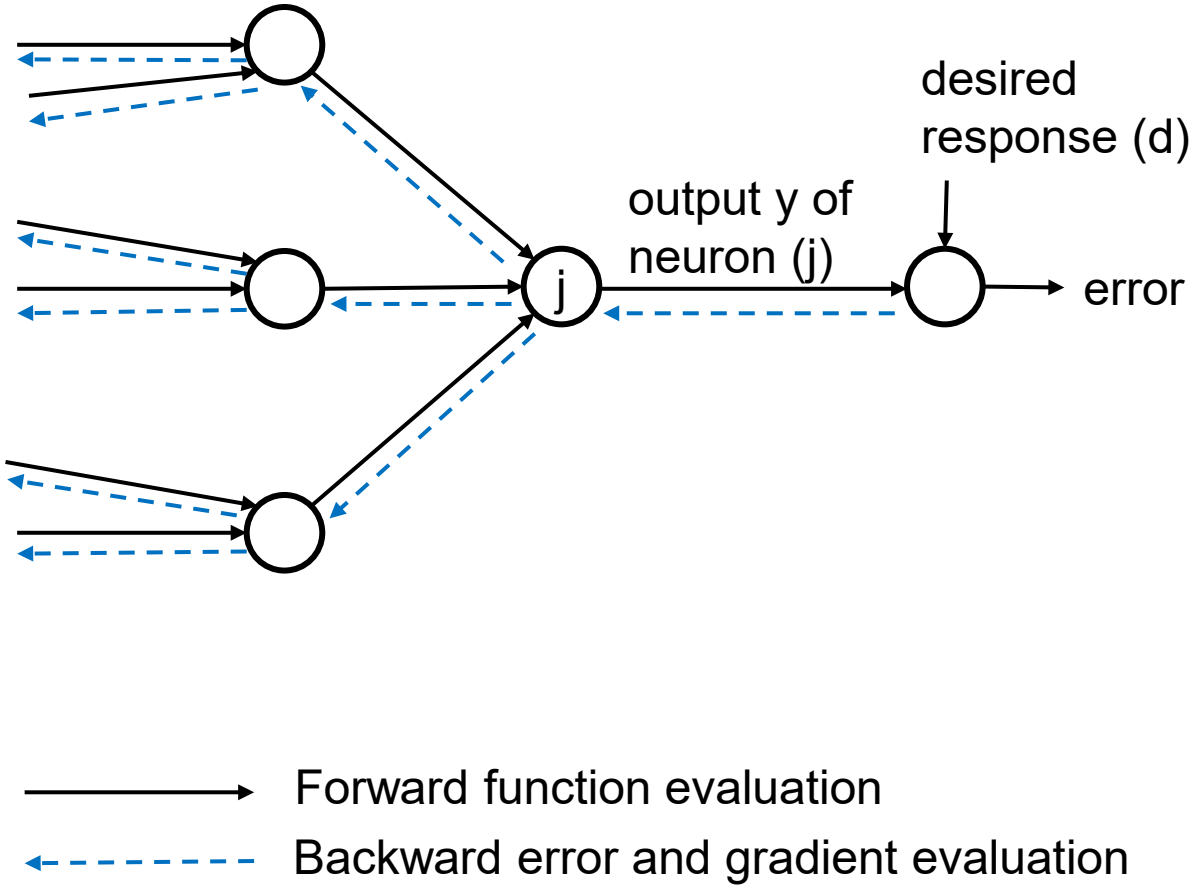
Training ANNs with gradient descent algorithm



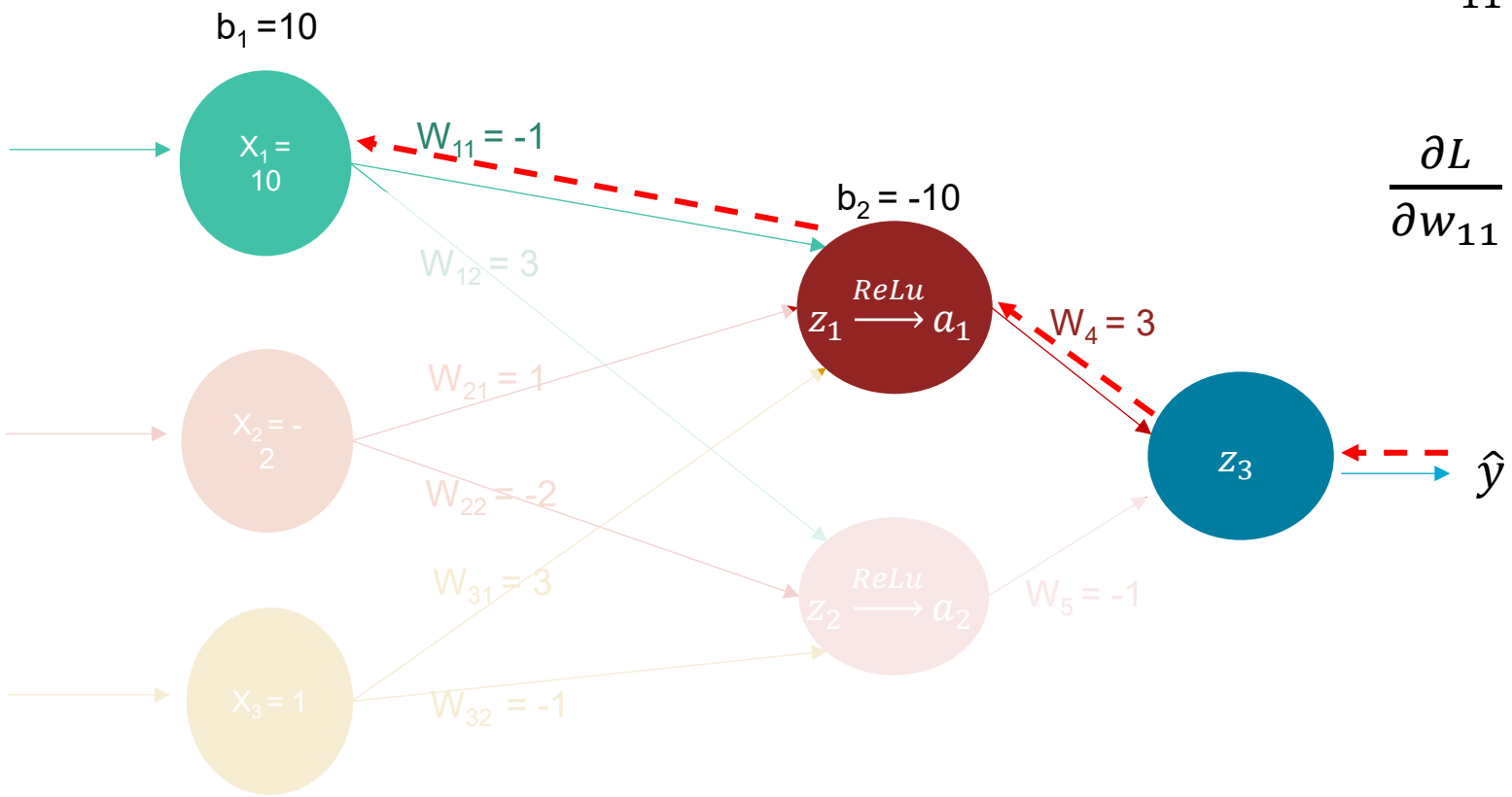
$$w'_{11} = w_{11} - \alpha \frac{\partial L}{\partial w_{11}}$$ (α :learning rate)

The two passes in neural network training

- The forward pass…
  - passes data through the network layer by layer
  - applies weights in each neuron, adds a bias, and uses an activation function
- The backward pass…
  - adjusts network *weights* to minimize prediction error
  - computes the gradient of complex functions through *chain rule*
  - uses *gradient descent* to find good parameter



Example of backpropagation

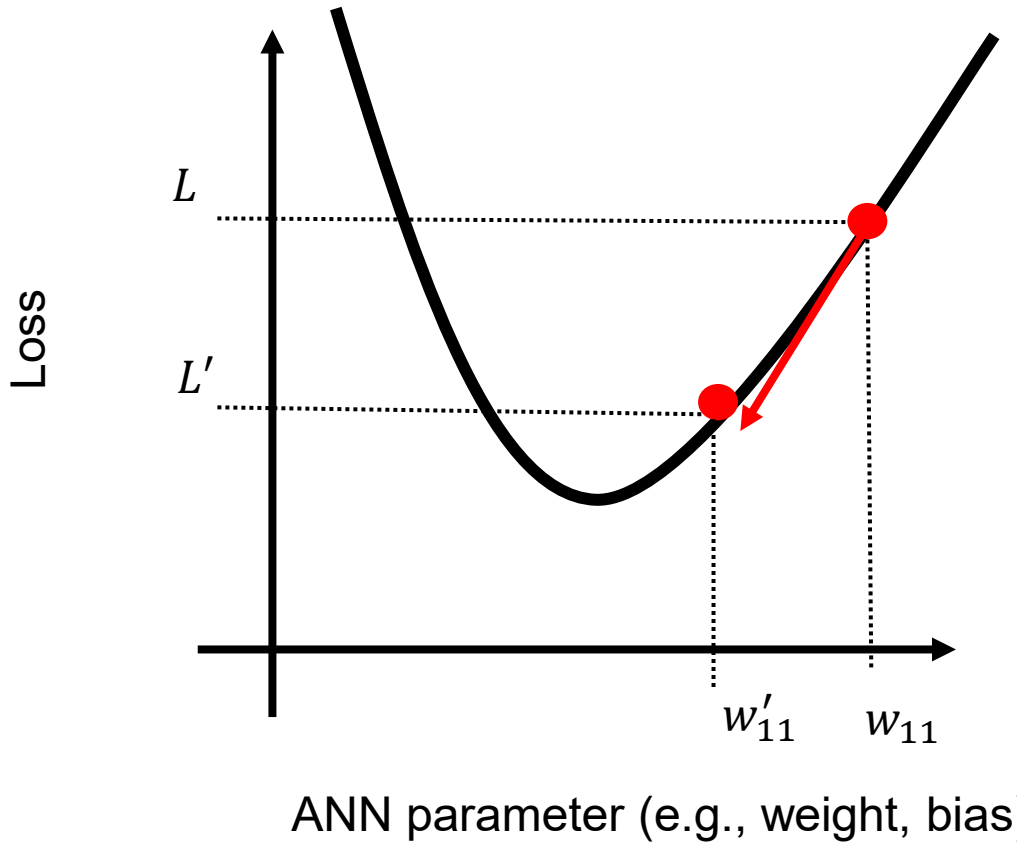


Example: Update w_{11}

$$w'_{11} = w_{11} - \alpha \frac{\partial L}{\partial w_{11}} \quad (\alpha : \textit{learning rate})$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3} \frac{\partial z_3}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_{11}}$$

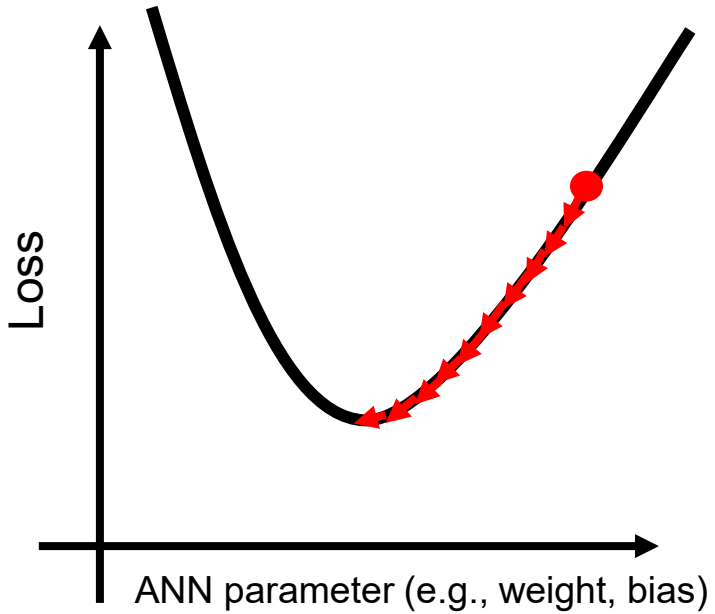
What is the influence of the learning rate α?



$$w'_{11} = w_{11} - \alpha \frac{\partial L}{\partial w_{11}}$$ *(α :learning rate)*

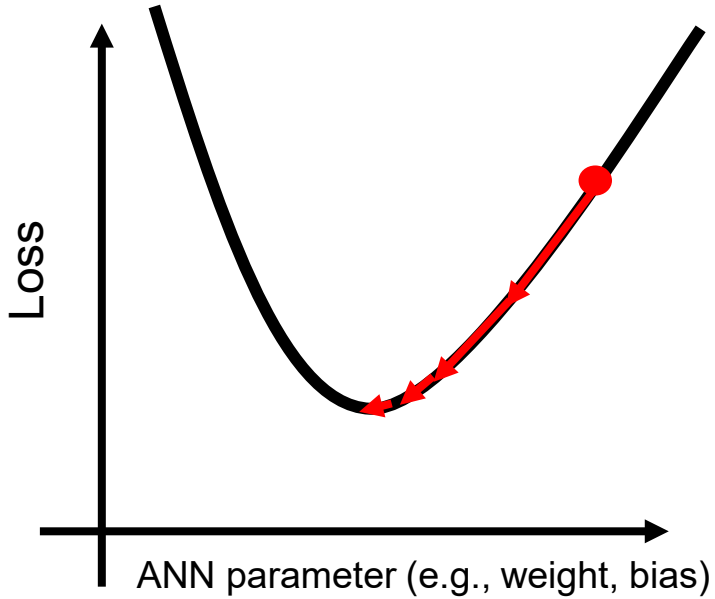
The influence of the learning rate

Too low



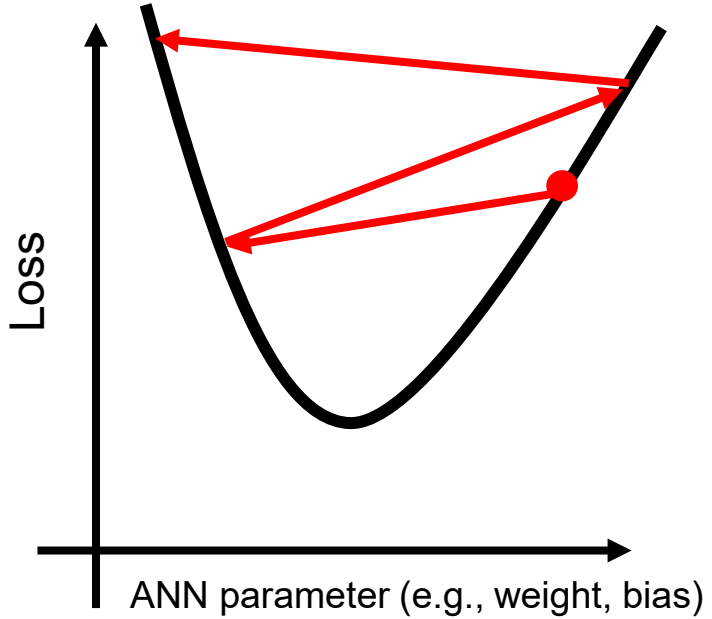
A small learning rate requires many updates before reaching the minimum point

Just right



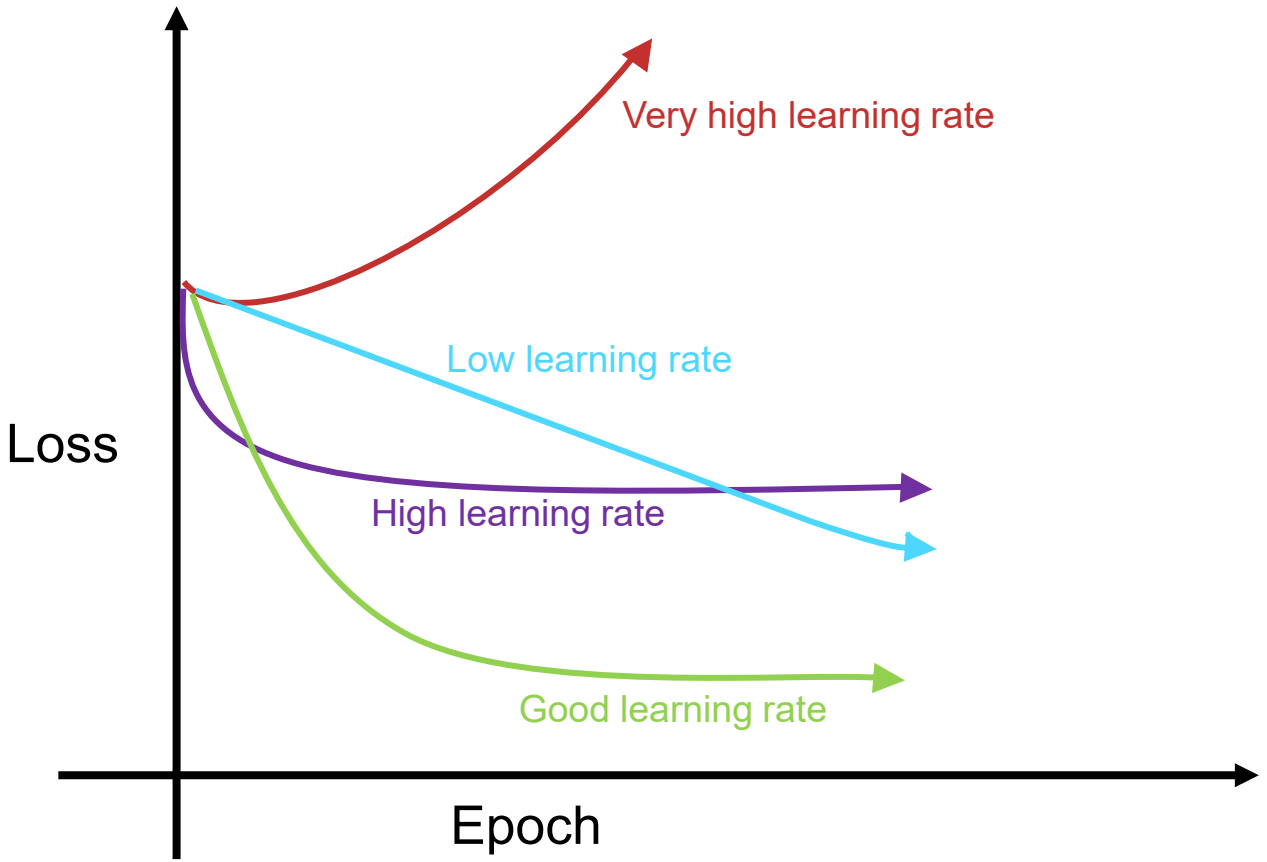
The optimal learning rate swiftly researches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviours (e.g. overshooting)

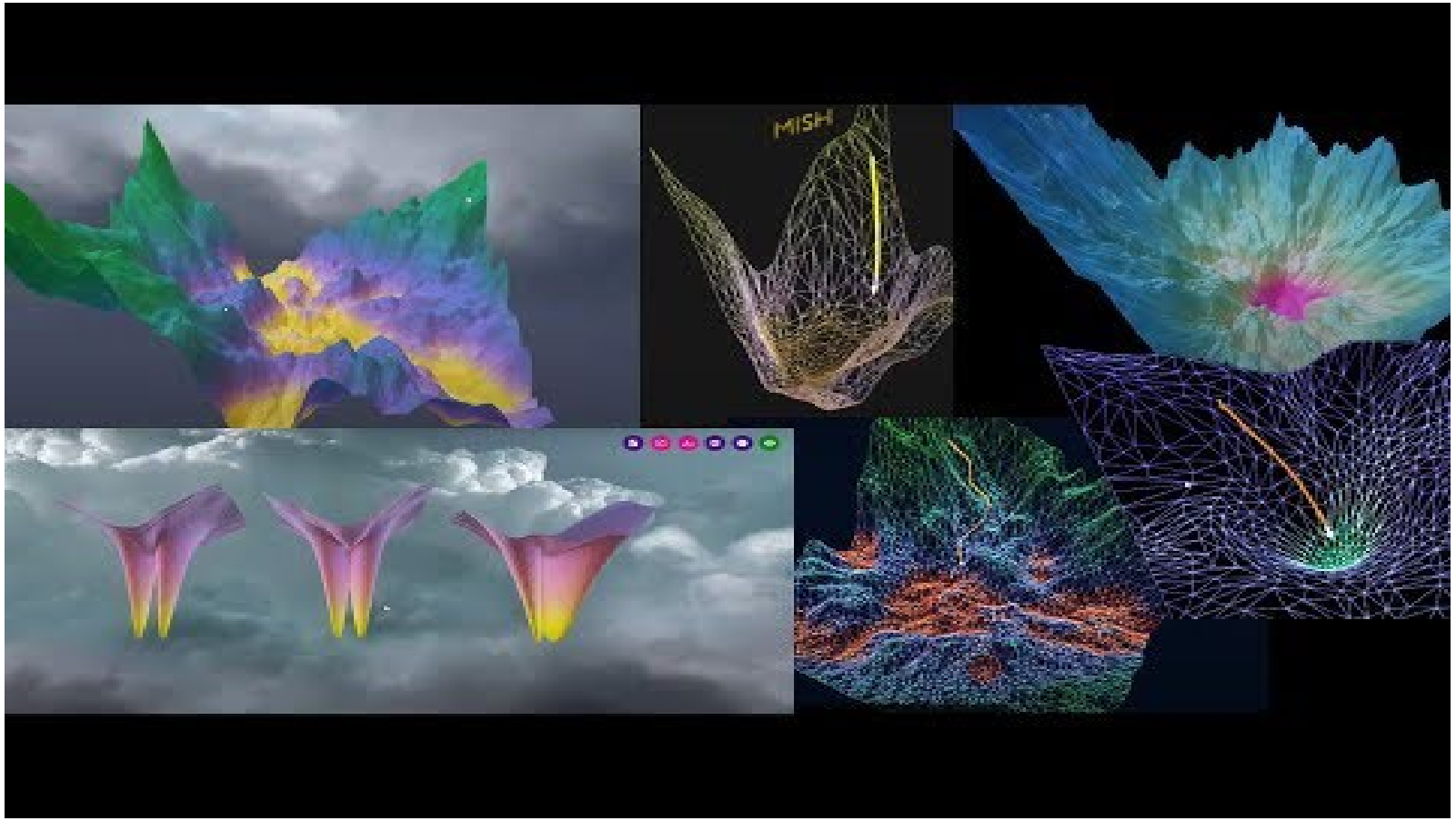
The influence of the learning rate in the convergence plot



Training deep networks using (mini) batches

- Batch Gradient Descent
  - In traditional gradient descent, we update model parameters using the entire dataset in each iteration.
  - Computationally expensive for large datasets. Can lead to slow convergence and high memory usage.
- Mini-Batch Gradient Descent
  - Divides the dataset into smaller, manageable subsets called "mini-batches.“ Each mini-batch contains a subset of the data, typically ranging from a few to a few hundred examples.
- Advantages of Mini-Batches
  - Reduced memory requirements
  - Faster convergence: Updates are more frequent, allowing for faster convergence.
  - Improved generalization: Stochasticity introduced by mini-batches can help escape local minima.

Loss landscape



See animation online: https://losslandscape.com/explorer

Example 1/3

https://github.com/process-intelligence-research/AI-in-Bio-Chemical-Engineering-Lecture-Coding

```
…
# Define a simple neural network model
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(1, 10)  # Input size: 1, Output size: 10
        self.fc2 = nn.Linear(10, 1)  # Input size: 10, Output size: 1

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Example 2/3

https://github.com/process-intelligence-research/AI-in-Bio-Chemical-Engineering-Lecture-Coding

```
model = Net()

# Define loss and optimizer

criterion = nn.MSELoss()  # Mean Squared Error loss

optimizer = optim.Adam(model.parameters(), lr=0.01)  # Adam optimizer
with learning rate 0.01
```

Example 3/3

https://github.com/process-intelligence-research/AI-in-Bio-Chemical-Engineering-Lecture-Coding

```
# Train the model
loss_history = []  # To store the training loss at each epoch
for epoch in range(100):
    optimizer.zero_grad()  # Reset gradients to zero
    outputs = model(X)  # Perform a forward pass to get predictions
    loss = criterion(outputs, y)  # Calculate the mean squared error
loss
    loss.backward()  # Perform backpropagation to compute gradients
    optimizer.step()  # Update model parameters using the optimizer
    loss_history.append(loss.item())  # Append the current loss value
to the history list
```

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **165-026-438**

Or scan the QR code

Join at: **vevox.app** ID: **165-026-438**

Question slide

You observe the following learning curve. What is likely the reasons for the increase of the loss over epochs?

A) The learning rate is too low

0%

B) The learning rate is too high

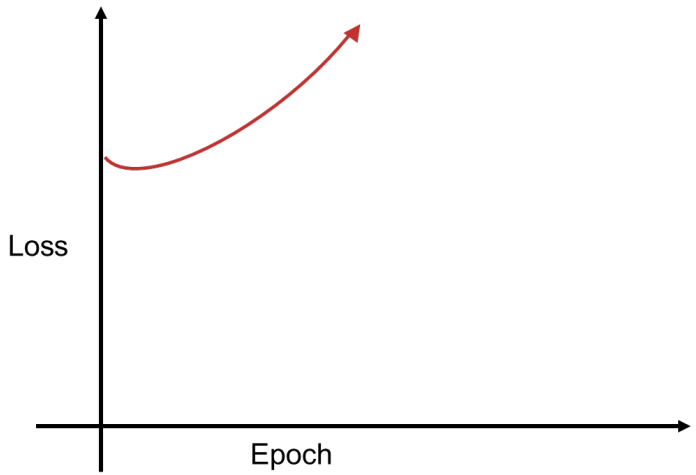
0%

C) The wrong loss function is used

0%

D) The number of epoch is too low

0%



You observe the following learning curve. What is likely the reasons for the increase of the loss over epochs?

Join at: **vevox.app** ID: **165-026-438**

Question slide

During the forward pass in ANN training, what is calculated?

A) Gradients

0%

B) Activation Functions

0%

C) Output Values

0%

D) Weights and Biases

0%

During the forward pass in ANN training, what is calculated?

A) Gradients 15.15%

B) Activation Functions 12.12%

C) Output Values ✓ 57.58%

D) Weights and Biases 15.15%

In neural network training, why is gradient descent often preferred over second-order optimization methods like Newton's method?

A) Gradient descent is faster and requires fewer iterations. 0%

B) Second-order derivatives do not exist for deep networks because of the ReLU activation function 0%

C) Gradient descent is less computationally intensive. 0%

D) Second-order methods can be computationally expensive due to Hessian calculations and storage, making them less practical for large-scale neural networks. 0%

In neural network training, why is gradient descent often preferred over second-order optimization methods like Newton's method?

A) Gradient descent is faster and requires fewer iterations. 9.09%

B) Second-order derivatives do not exist for deep networks because of the ReLU activation function 30.3%

C) Gradient descent is less computationally intensive. 18.18%

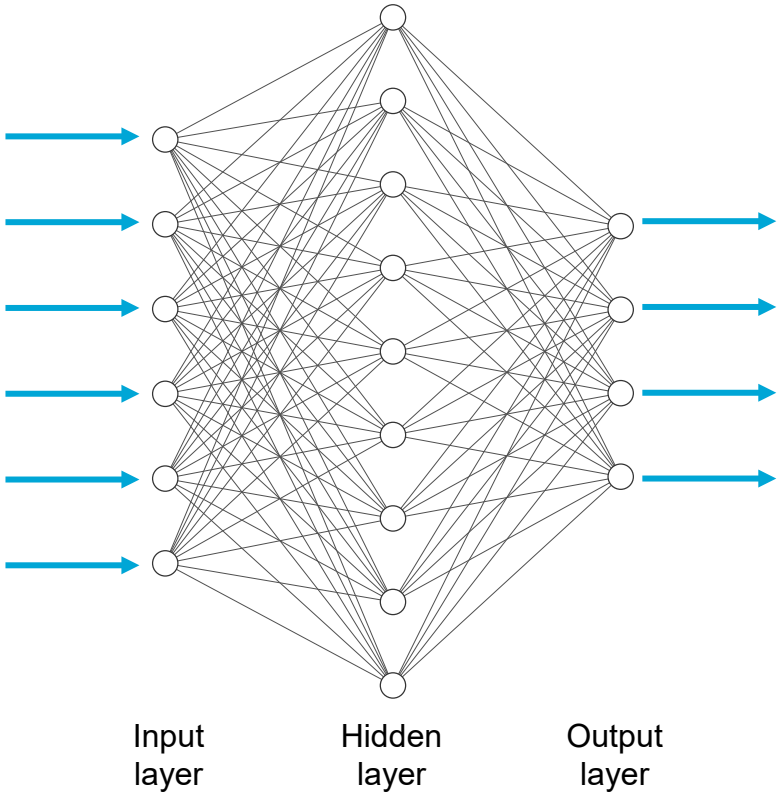
D) Second-order methods can be computationally expensive due to Hessian calculations and storage, making them less practical for large-scale neural networks. 42.42%

Lecture outline

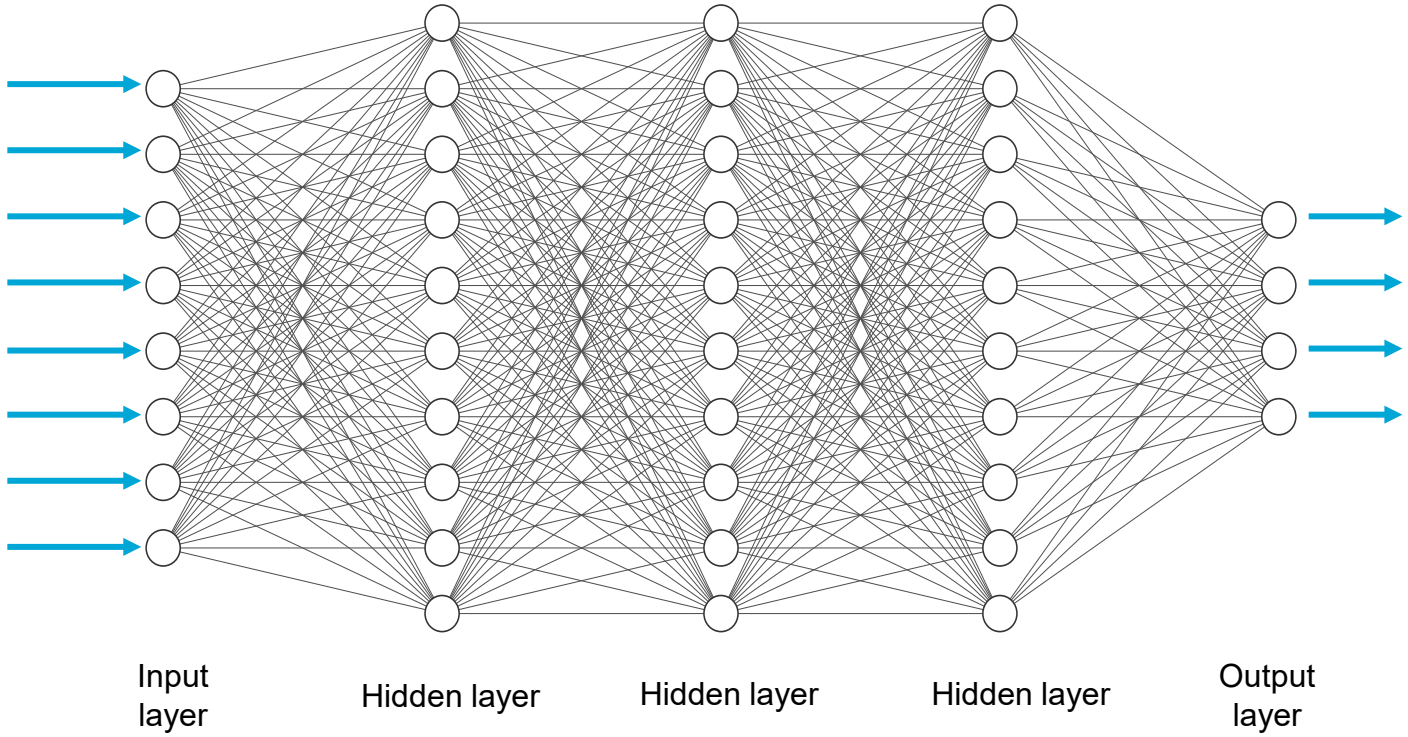
- Neural network basics
- Neural network training
- **Architecture and hyperparameters**
- Applications of neural networks for regression

How to identify a suitable ANN architecture?

Shallow artificial neural network



Deep artificial neural network

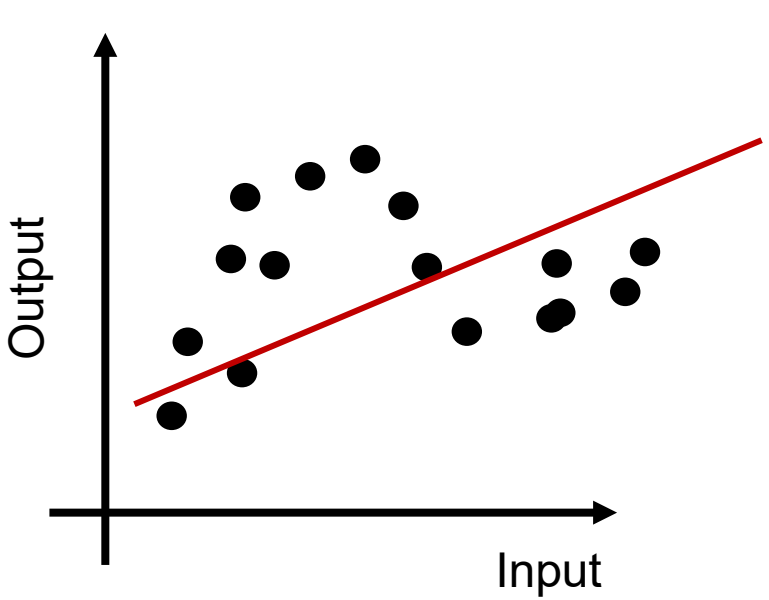


Figures generated in https://alexlenail.me/NN-SVG/index.html

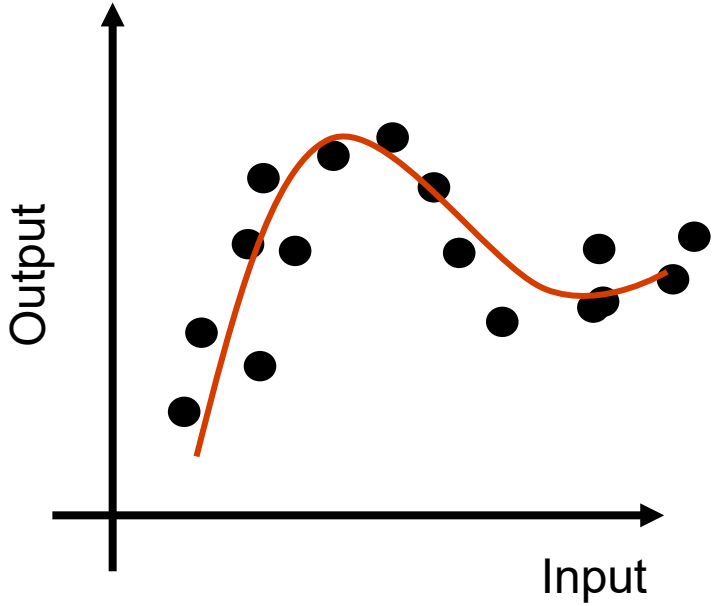
Wide or deep?

- Nomenclature
  - Width W: Number of neurons in hidden layer
  - Depth D: Number of hidden layers
  - “deep learning”: depth > 2
- Complexity
  - Fully connected network: # parameters $\sim O(W^2 \cdot D)$
- Generalization
  - Deep can learn intermediate features and may generalize better
- Depth and width are hyperparameters → optimize them!

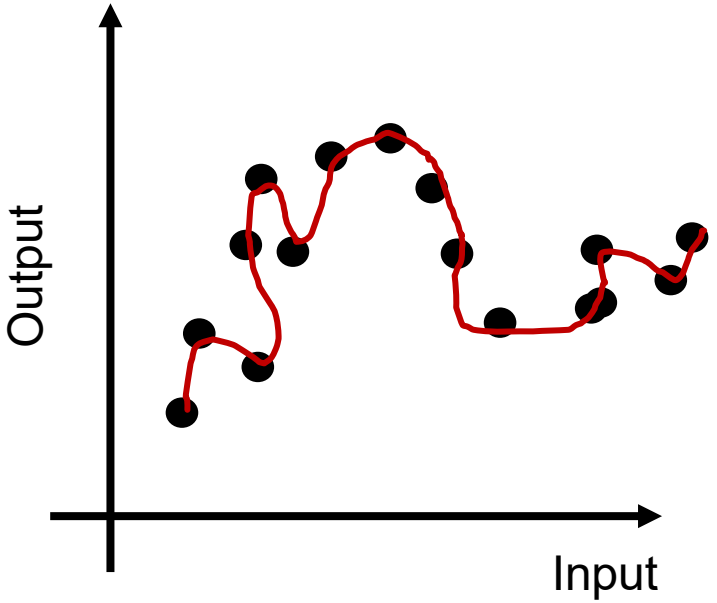
Underfitting and overfitting



Underfitted



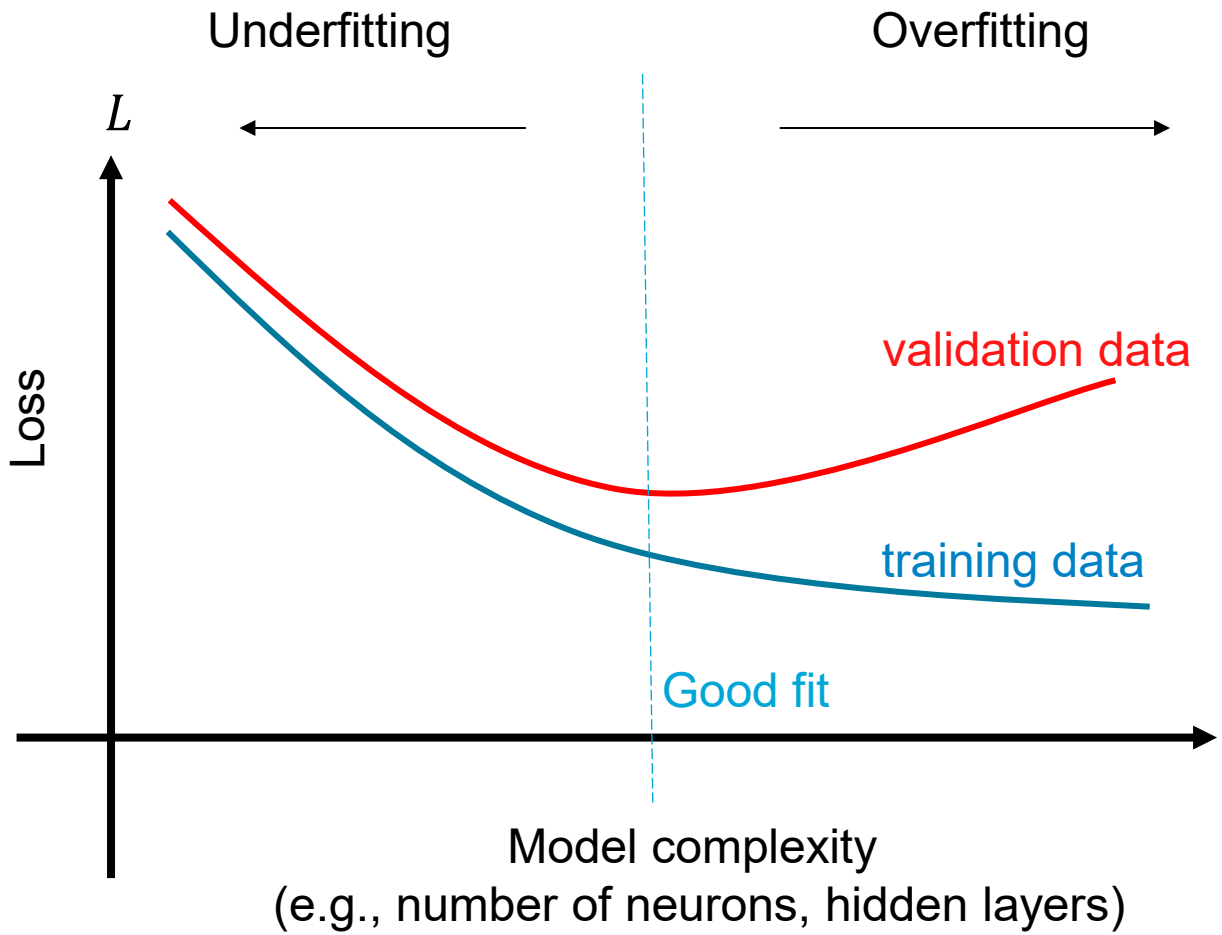
Good fit/ Robust



Overfitted

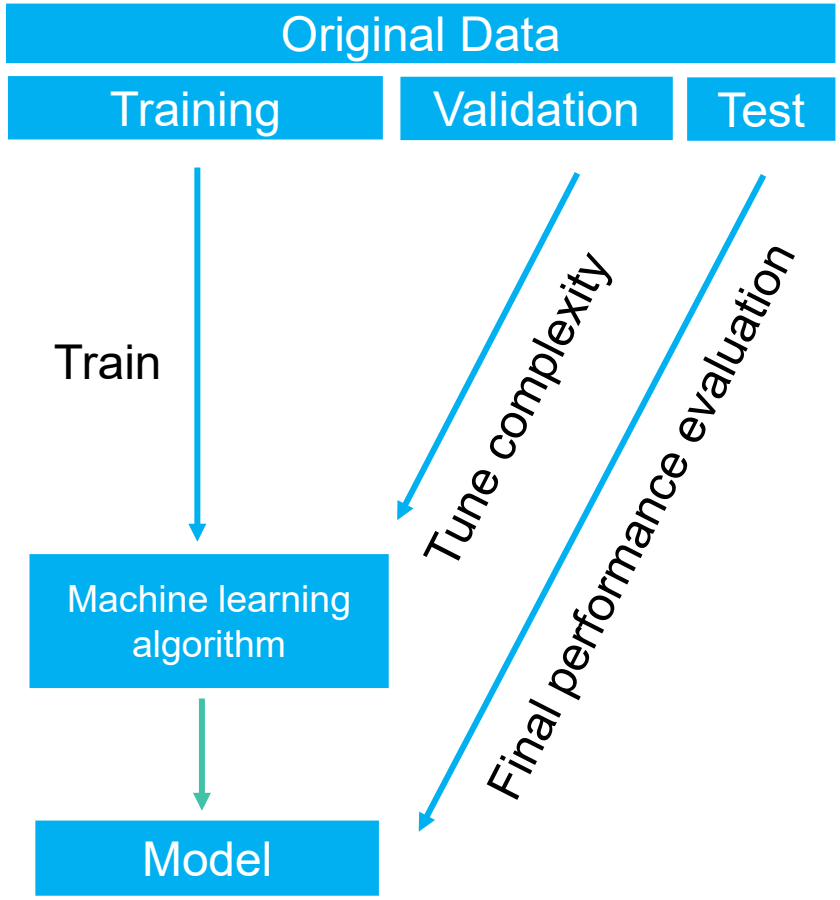
Figure: https://miro.medium.com/max/1125/1*_7OPgojau8hkiPUiHoGK_w.png

How to identify overfitting?



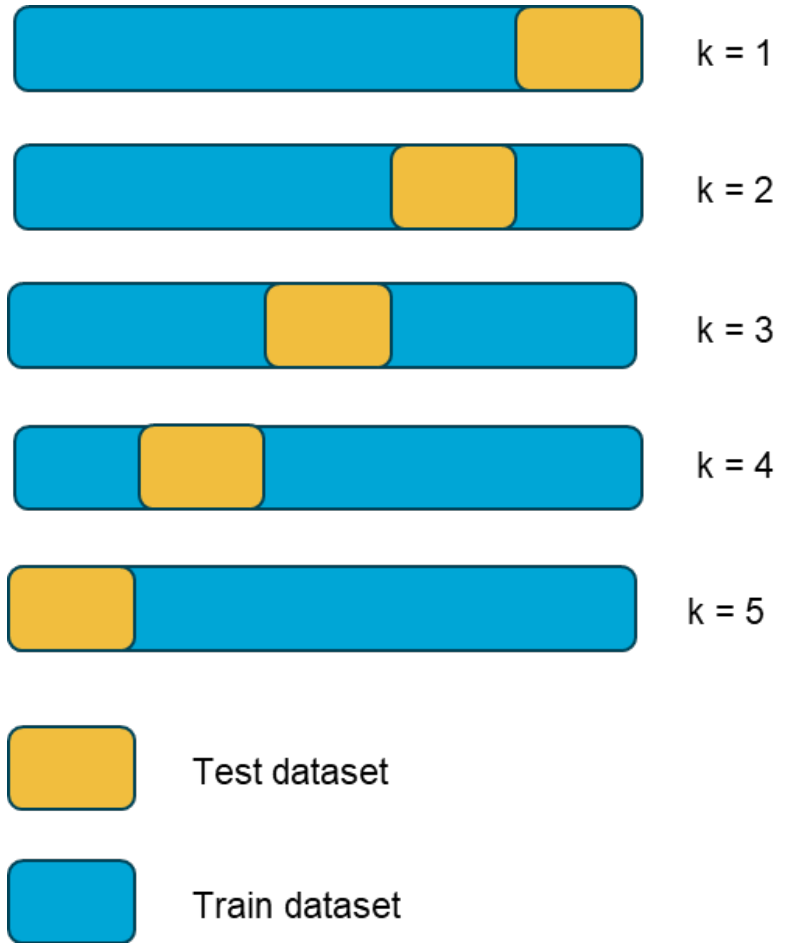
Splitting the dataset into training, validation, and test

- The original data set is randomly divided into training, test and validation set according to a predefined ratio,
  - e.g., 70%:15%:15%
- The training is done using a training set
- The model complexity (i.e., #layers and #neurons) are chosen using an independent validation set
- The performance of data-driven models is evaluated on an independent test set



k-fold cross-validation

- Cross-validation is a valuable alternative when dealing with limited training data.
- Instead of the traditional training/test/validation split, cross-validation randomly divides the dataset into k subsets.
- The model is tested k times, each time using a different subset for validation.
- Parameter scores from each fold are averaged to assess overall performance.
- Cross-validation ensures thorough testing across the entire dataset, enhancing model evaluation.
- While cross-validation provides robust evaluations, it can be computationally expensive due to multiple model training and evaluation iterations.



Feeling like learning more…

- Test and explore the different effects the settings of a neural network can have in the output
  - Visit *playground.tensorflow.org*
  - Play with the settings and reflect on your findings

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **165-026-438**

Or scan the QR code

If your neural network has overfit the training set in modelling a product quality based on historical data, what does it mean?

A) It makes accurate predictions for examples in the training set and generalizes well on new, previously unseen examples.

0%

B) It does not make accurate predictions for examples in the training set but generalize well on new, previously unseen examples.

0%

C) It makes accurate predictions for examples in the training set but does not generalize well on new, previously unseen examples.

0%

D) It does not make accurate predictions for examples in the training set and does not generalize well on new, previously unseen examples.

0%

If your neural network has overfit the training set in modelling a product quality based on historical data, what does it mean?

A) It makes accurate predictions for examples in the training set and generalizes well on new, previously unseen examples.

0%

B) It does not make accurate predictions for examples in the training set but generalize well on new, previously unseen examples.

3.03%

C) It makes accurate predictions for examples in the training set but does not generalize well on new, previously unseen examples.

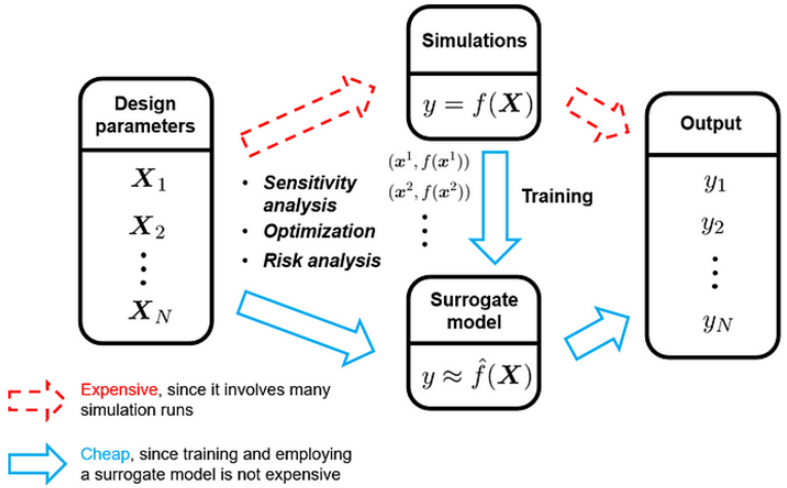
87.88%

D) It does not make accurate predictions for examples in the training set and does not generalize well on new, previously unseen examples.

9.09%

Surrogate models – a data-driven approach

- Surrogate models serve the purpose of simplifying the representation of a system's behavior [1].
- They enable efficient analysis of a system's key factors with minimal computational resources.
- Various techniques can be employed to derive surrogate models, including:
  - Linear or nonlinear regression
  - Projection onto subdomains using methods like POD (Proper Orthogonal Decomposition), SVD (Singular Value Decomposition), and DMD (Dynamic Mode Decomposition).
  - Machine learning algorithms

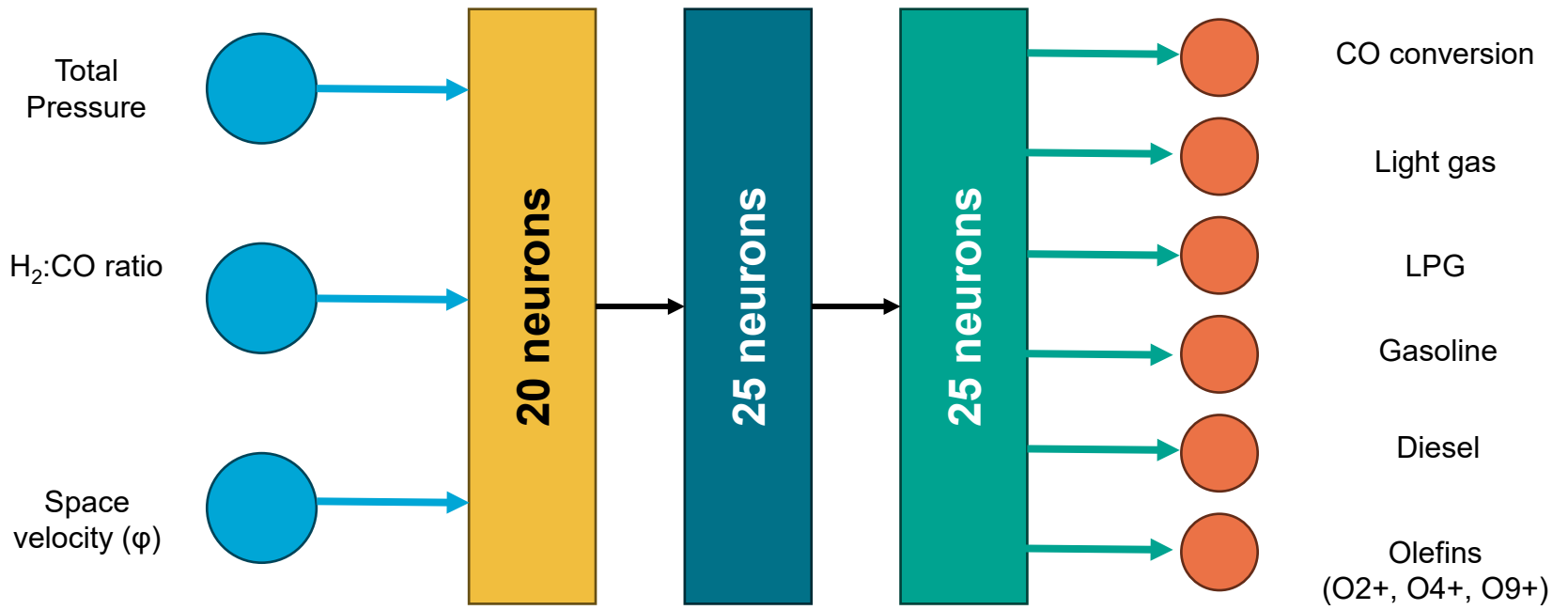


[1] Gargalo, C., et al., 2021. Towards the Development of Digital Twins for the Bio-manufacturing Industry. In: Herwig, C., Pörtner, R, and J., Möller. 2021. https://doi.org/10.1007/10_2020_142
[2] Guo, S. An introduction to Surrogate Modeling, Part I: Fundamentals. Available in: https://towardsdatascience.com/an-introduction-to-surrogate-modeling-part-i-fundamentals-84697ce4d241

Surrogate models – example in ChemE

Check implementation in repository

- **Optimization of FTS using NNs** [3]
  - Fernandes et al. used a NN in substitution to the complex reaction mechanism to estimate the product distribution and optimize the production of gasoline and diesel.



Goal:
Replace complex kinetic model for FTS

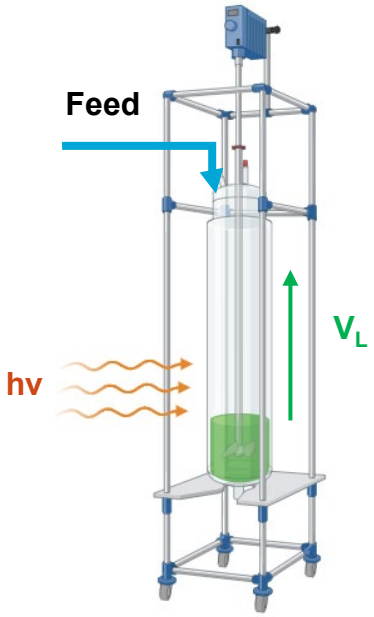
Training:
150k epochs
75 % data splitting
(15: 5 experimental values)

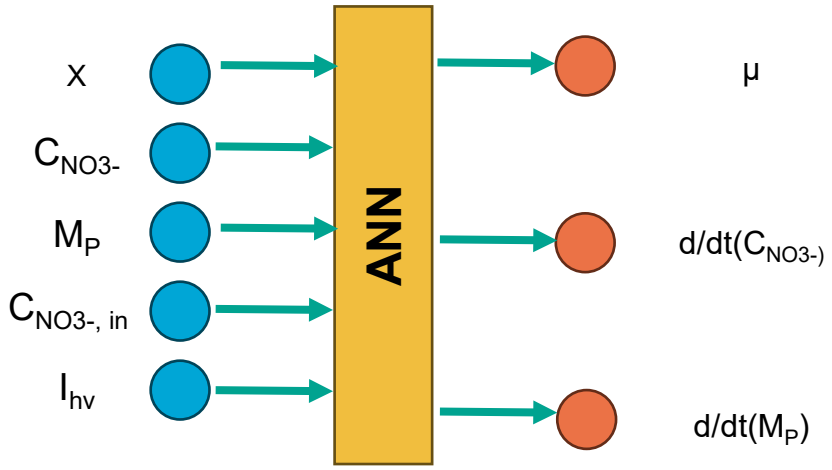
Assessment:
Absolute and relative error (OPE, MPE)

[1] Palmer, K. and M. Realff. 2002. Metamodeling approach to optimization: Model generation. Trans IChemE. 80(7): 760-772.
[2] Mujtaba, I., Aziz, N. and M. Hussain. 2006. Neural network based modelling and control in Batch reactor. Chemical Engineering Research and Design. 84(A8): 635-644
[3] Fernandes, F. 2006. Optimization of Fischer-Tropsch Synthesis using Neural Networks. Chemical Engineering and Technology. 29(4): 449-453. Figure based on publication.

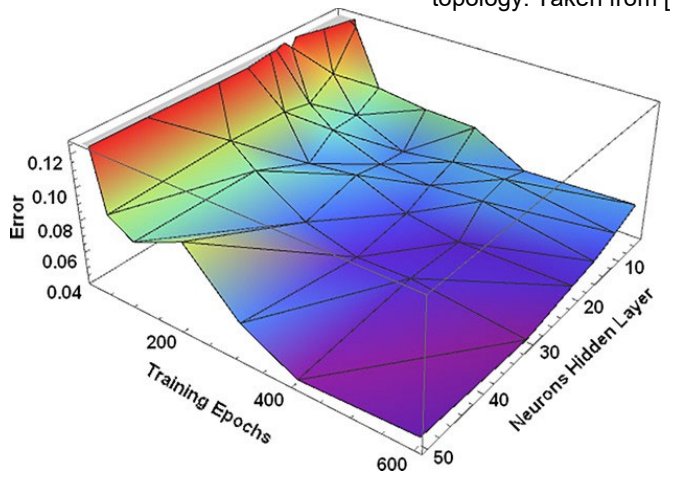
Surrogate models – example in BioChemE

- **Prediction of microalgal lutein photo-production under dynamic conditions** [1]
  - Authors study how two robust ANN topologies (one vs. two hidden layers) can compute the rate of change in a fed-batch process (i.e., growth, accumulation, and production rates).





Elbow rule for 2*h topology. Taken from [1]



Hyperparameter optimization performed using the 'elbow rule'

[1] del Rio-Chanona, E., Fiorelli, F. Zhang, D., Nuur, A., Jing, K and N. Shah. 2017. An efficient model construction strategy to simulate microalgal lutein photo-production dynamic process. Biotechnology and Bioengineering 114(11): 2518-2527. DOI: 10.1002/bit.26373

Surrogate models – example in BioChemE

- **Prediction of microalgal lutein photo-production under dynamic conditions** [1]

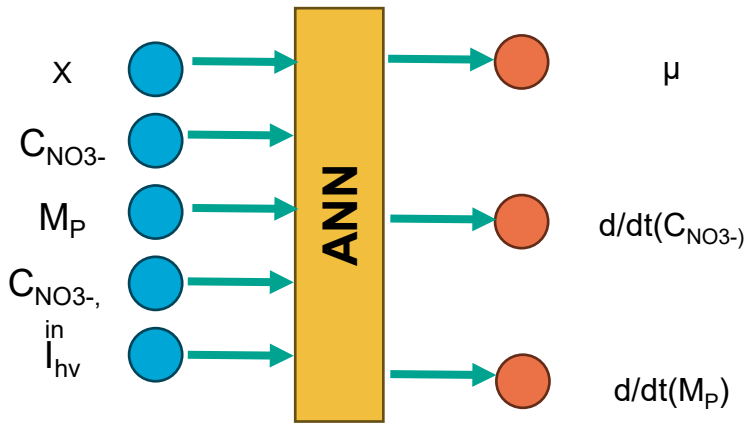


Figure 1. ANN inputs and outputs for the framework evaluated in [1]

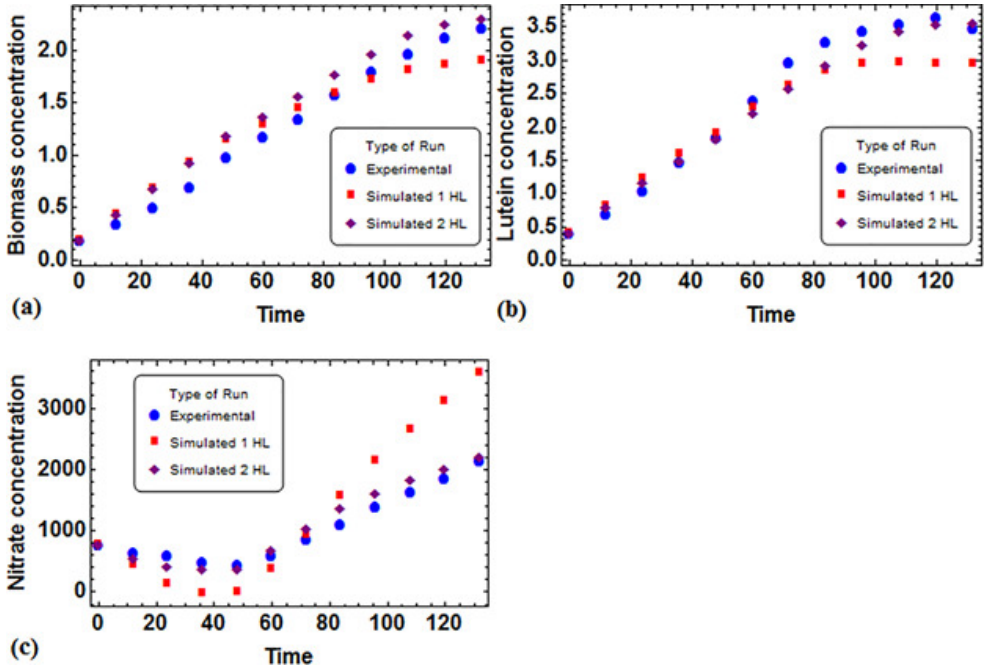


Figure 2. Comparison of prediction results for topologies evaluated after hyperparameter optimization, and experimental data. Taken from [1]

[1] del Rio-Chanona, E., Fiorelli, F. Zhang, D., Nuur, A., Jing, K and N. Shah. 2017. An efficient model construction strategy to simulate microalgal lutein photo-production dynamic process. Biotechnology and Bioengineering 114(11): 2518-2527. DOI: 10.1002/bit.26373

Learning goals of this lecture

After successfully completing this lecture, you are able to….

- Explain the fundamentals on artificial neural networks along with some of their limitations
- Explain the fundamentals on training with backpropagation
- Analyse a machine learning result in the context of overfitting
- Develop and implement neural networks in the context of basic (Bio)Chemical Engineering applications (i.e., regression and classification)

Assignment 2: Deep learning for regression

- **Case study 1:**
  - Illustrative 1-D case example
  - Analyse influence of model complexity
  - Assess the extrapolation capabilities of the model
- **Case study 2:**
  - Surrogate model of process simulations
  - Implement a deep NN for predicting process variables
- **Case study 3 (optional)**:
  - Molecular property prediction example using QSPR
  - Analyse the influence of molecular descriptors for predicting boiling temperature

Thank you very much for your attention!