Neural-network based models in chemistry

Summer School 2025: Machine Learning and Artificial Intelligence in Synthetic Chemistry

University of Helsinki and the Finnish Society for Synthetic Chemistry

Lucía Morán González

Outline

1. Idea of Neural Network (NN)
2. Components of a NN
3. Training a NN model
4. Play with them – playground tensorflow
5. NN APIs
6. Jupyter notebook

Neural Network (NN) in chemistry

ACS central science

ACS AuthorChoice

Research Article

http://pubs.acs.org/journal/acscii

Neural Networks for the Prediction of Organic Chemistry Reactions

Jennifer N. Wei,[†] David Duvenaud,[‡] and Alán Aspuru-Guzik*,[†]

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

Cite This: J. Chem. Inf. Model. 2020, 60, 47−55

pubs.acs.org/jcim

Predicting Retrosynthetic Reactions Using Self-Corrected Transformer Neural Networks

Shuangjia Zheng,[†,‡,⊥] Jiahua Rao,[‡,⊥] Zhongyue Zhang,[‡] Jun Xu,*,[†,§] and Yuedong Yang*,[‡,∥]

ARTICLE

https://doi.org/10.1038/s41467-019-12875-2 OPEN

Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions

K.T. Schütt[1], M. Gastegger[1], A. Tkatchenko[2*], K.-R. Müller[1,3,4*] & R.J. Maurer[5*]

SCIENCE ADVANCES | RESEARCH ARTICLE

MATERIALS SCIENCE

Inverse design of porous materials using artificial neural networks

Baekjun Kim*, Sangwon Lee*, Jihan Kim[†]

ARTICLE

doi:10.1038/nature25978

Planning chemical syntheses with deep neural networks and symbolic AI

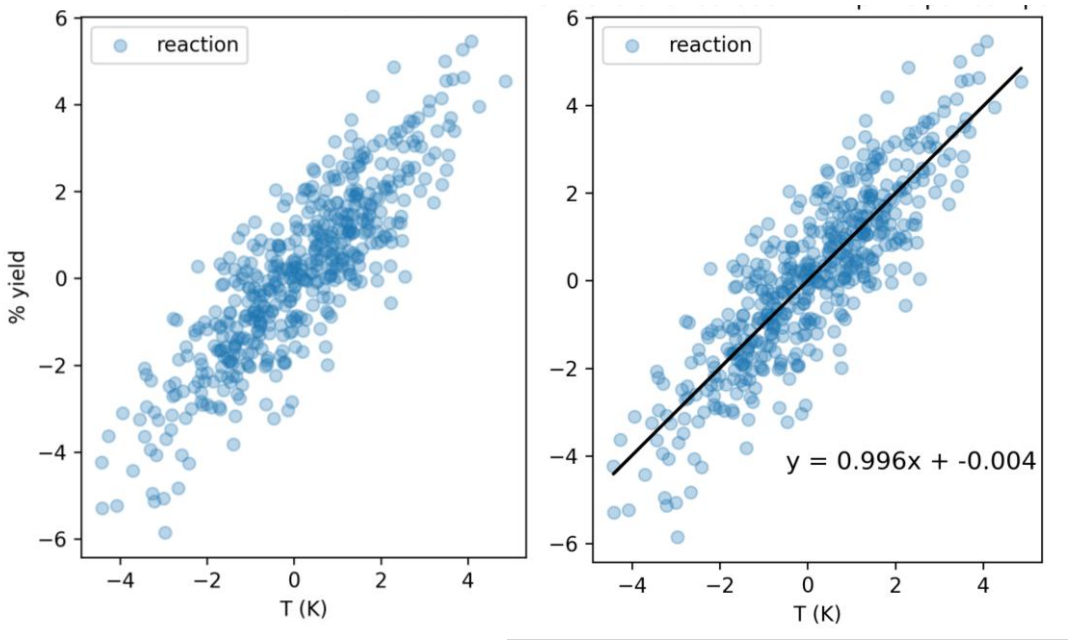
Marwin H. S. Segler[1,2], Mike Preuss[3] & Mark P. Waller[4]

Neural Network (NN) in chemistry

- NN can model **complex** mathematical functions
- **Flexible**: adaptability to complex, high-dimensional data

Neural Network (NN) in chemistry

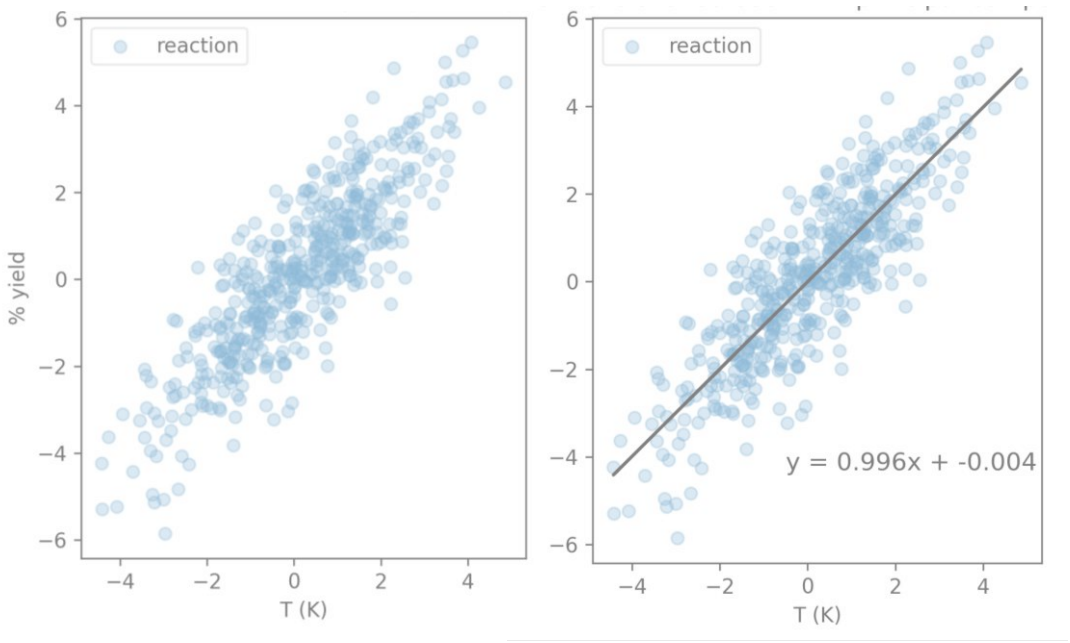
- NN can model **complex** mathematical functions
- **Flexible**: adaptability to complex, high-dimensional data



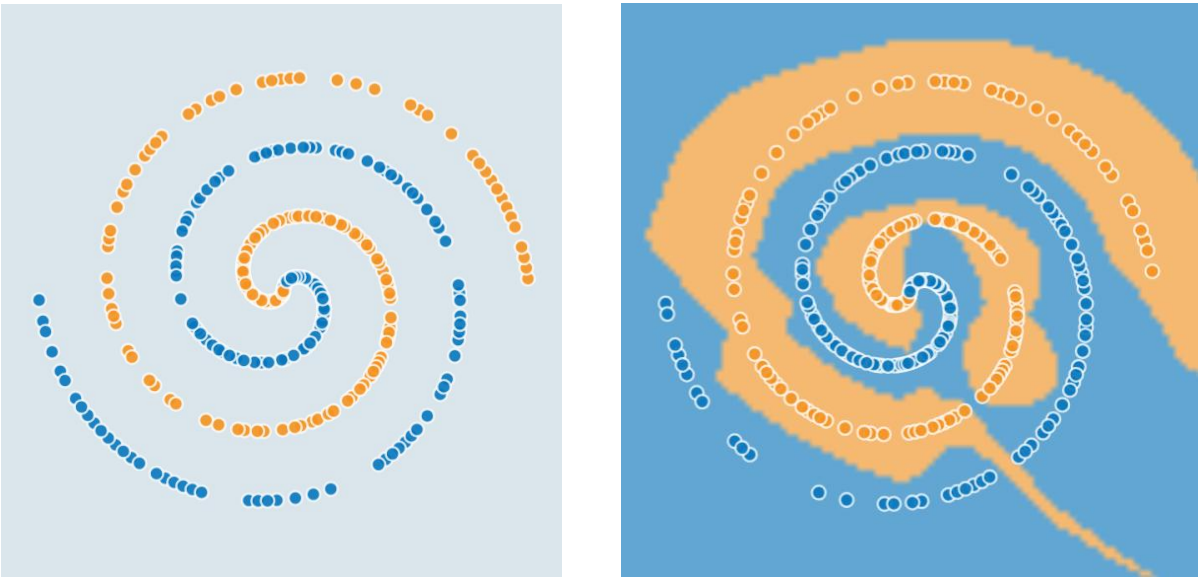
'Ideal' simple maths

Neural Network (NN) in chemistry

- NN can model **complex** mathematical functions
- **Flexible**: adaptability to complex, high-dimensional data



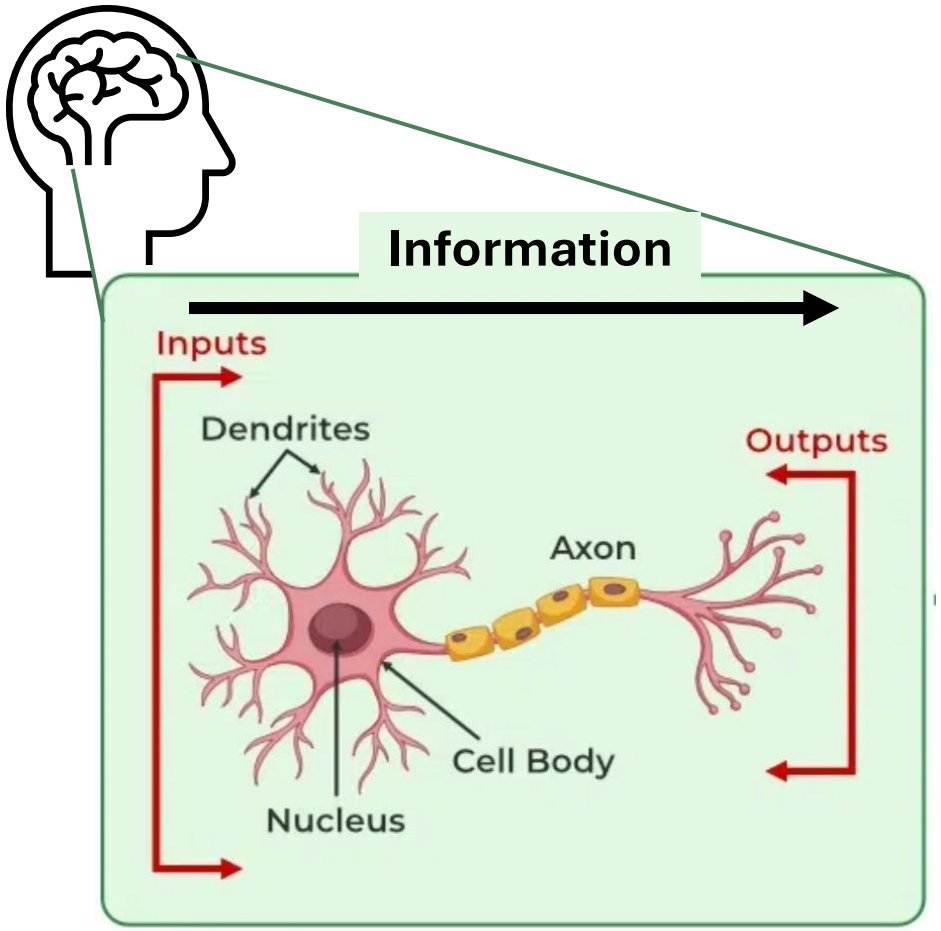
'Ideal' simple maths

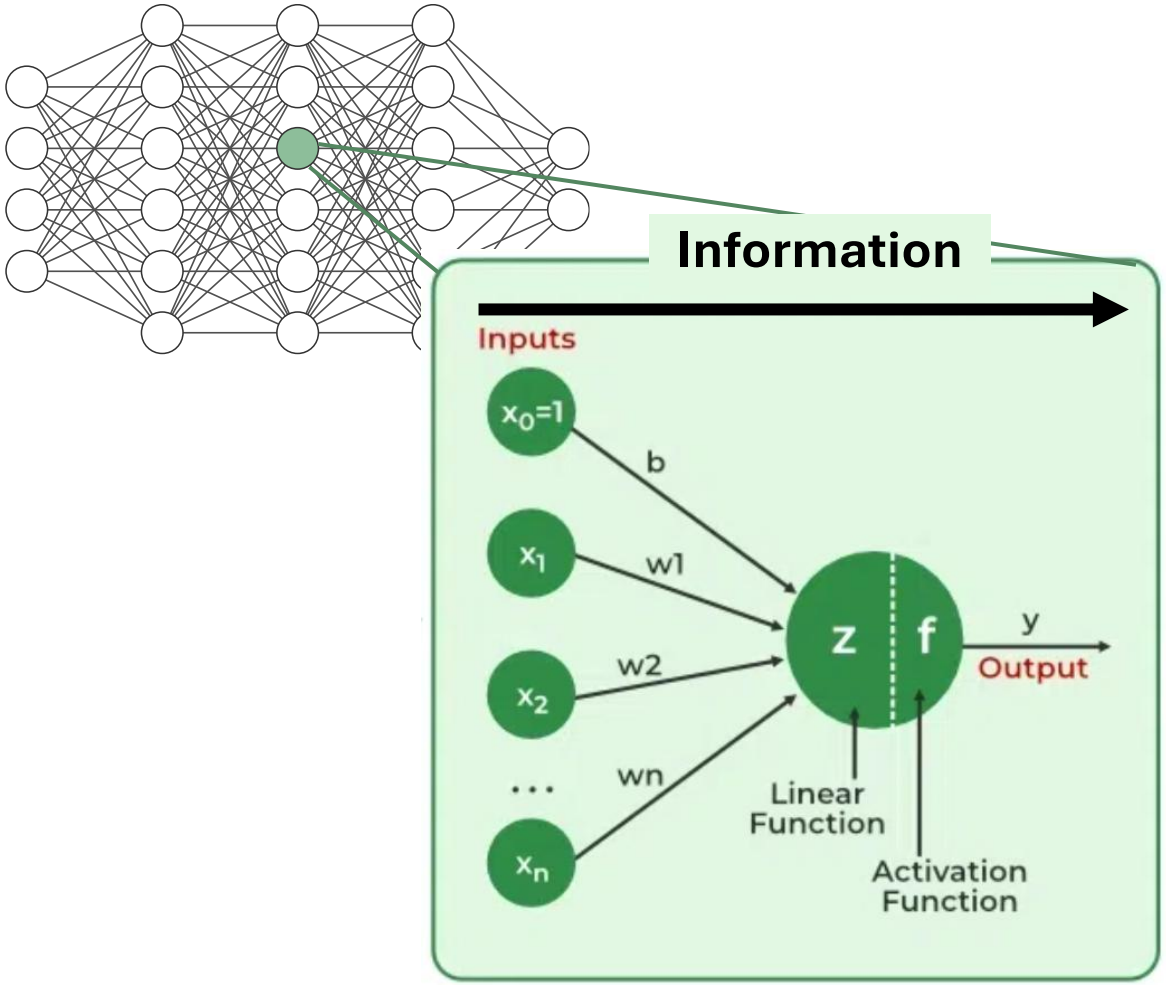


Reality is more complex

Core concept of NN

Information flows through connected neurons



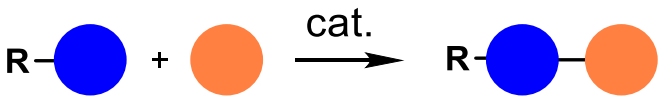


Rosenblatt, F. *Psychological Review*, 1957, 65, 386-408

Adapted from https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/

Supervised ML: $\boldsymbol{X}$ and $\boldsymbol{y} \rightarrow \hat{\boldsymbol{y}}$

Supervised ML: $\boldsymbol{X}$ and $\boldsymbol{y} \rightarrow \hat{\boldsymbol{y}}$



$\boldsymbol{X}$ = sterics of R, Hammet σ, T, [cat],...
features

$\boldsymbol{y}$ = %yield
target property

Dataset

Reactions	X = sterics of R, Hammet σ, T, ...	y = %yield
Reaction 1	Volume(-Ph_3), σ(-Ph_3), 100ºC, 1mM	80%
Reaction 2	Volume(-Me), σ(-Me), 100ºC, 3mM	20%
Reaction 3	Volume(-CF_3), σ(-CF_3), 25ºC, 5mM	67%
...		...
Reaction n	Volume(-NO_2), σ(-NO_2), 25ºC, 3mM	45%

Goal

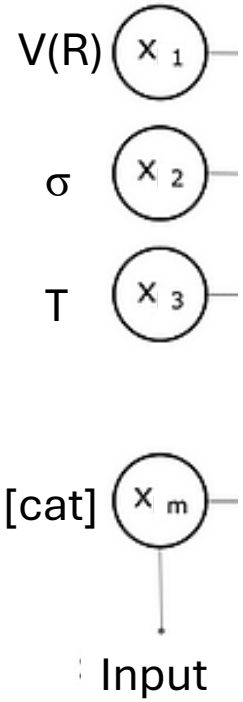
New reactions	X = sterics of R, Hammet σ, T, ...	y = pred(%yield)
Reaction --	Volume(-R), σ(-R), 120ºC, 2mM	% ---

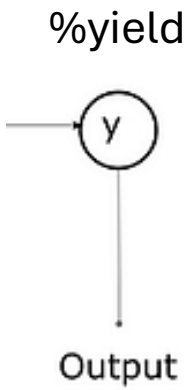
NN architecture

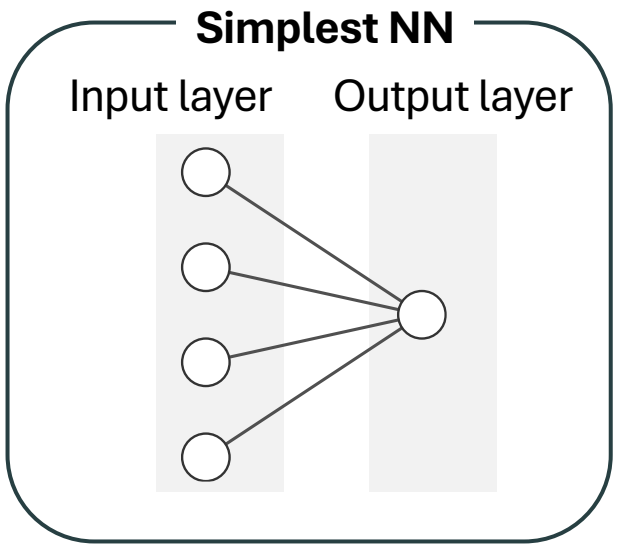
NN training

$\boldsymbol{X}$ = sterics of R, Hammet σ, T, [cat],... *features*

$\boldsymbol{y}$ = %yield *target property*



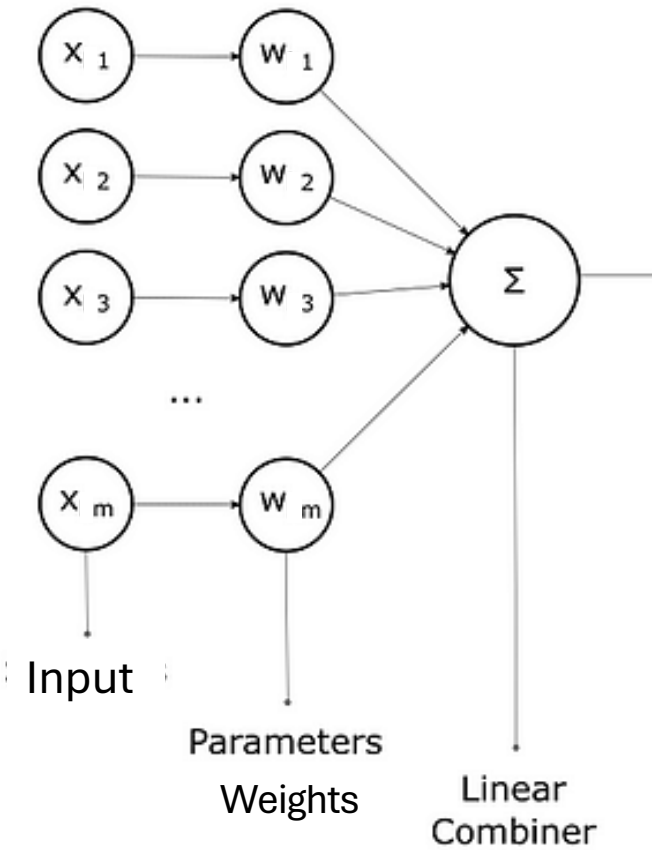




NN architecture

NN training

$$u = W \cdot X = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$



- **Weights**: Initially they are random values that need to be adjust

NN architecture

NN training

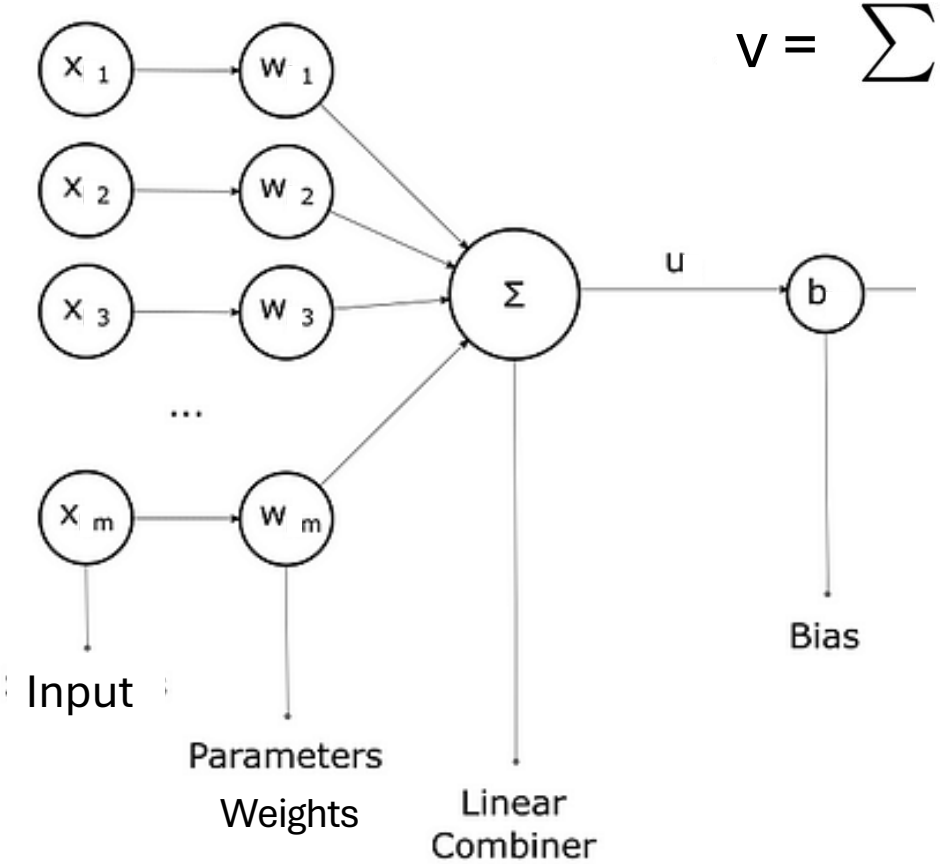
u = $W \cdot X = w_1x_1 + w_2x_2 + ... + w_mx_m = \sum_{i=1}^{m} w_ix_i$

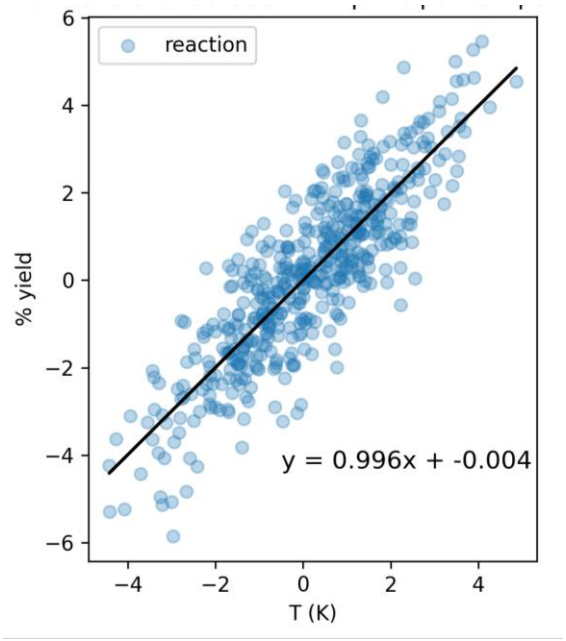
y = *ax*

v = $\sum_{i=1}^{m} w_ix_i + bias$

y = *ax* + *b*

Linear Functions





NN architecture

NN training

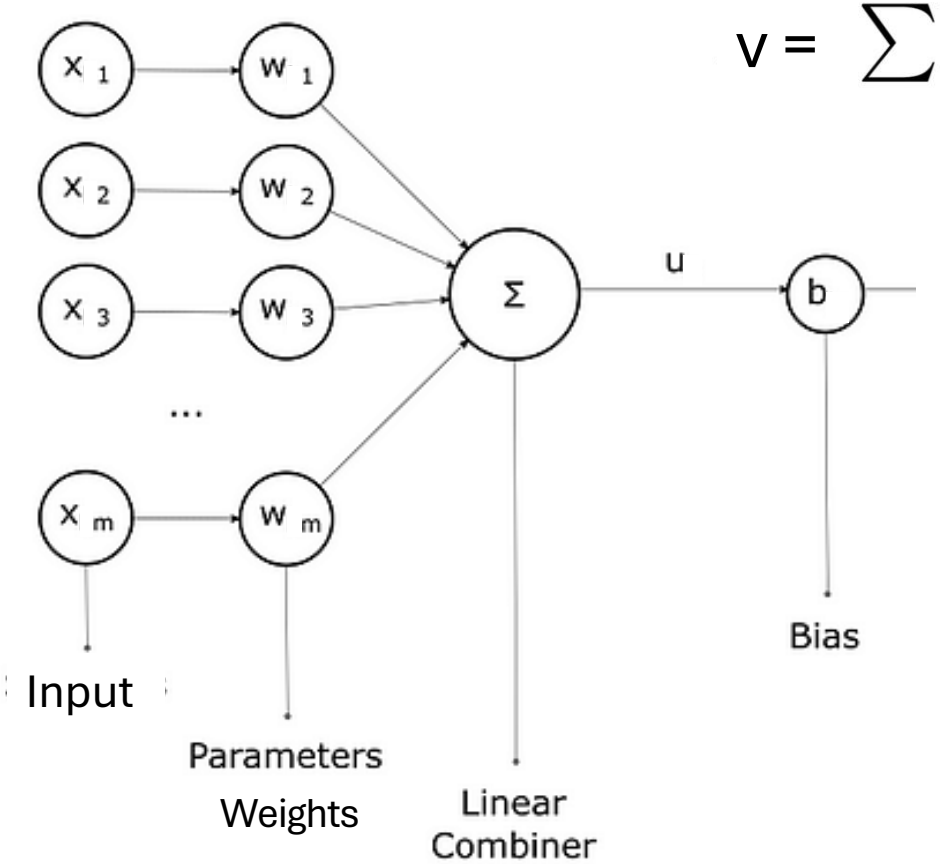
$$u = W \cdot X = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$

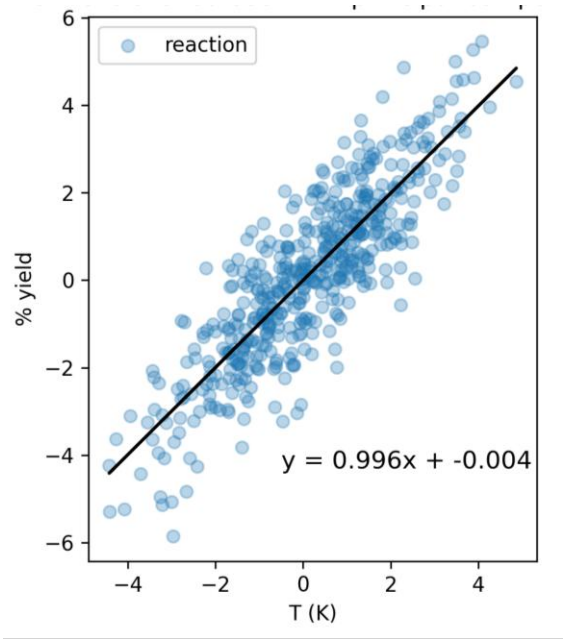
$y = ax$

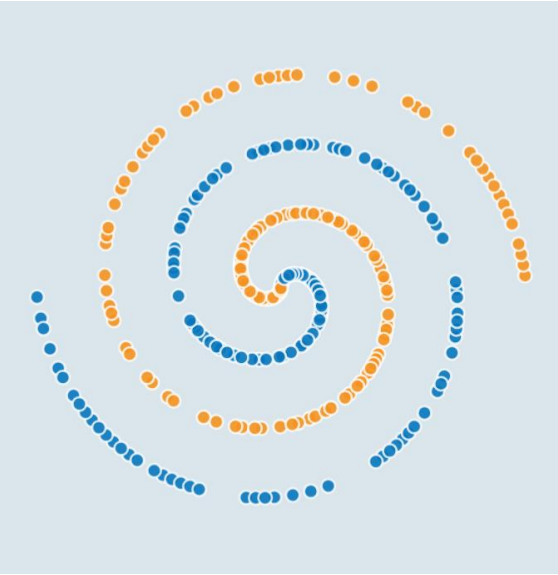
$$v = \sum_{i=1}^{m} w_i x_i + bias$$

$y = ax + b$

Linear Functions







NN architecture

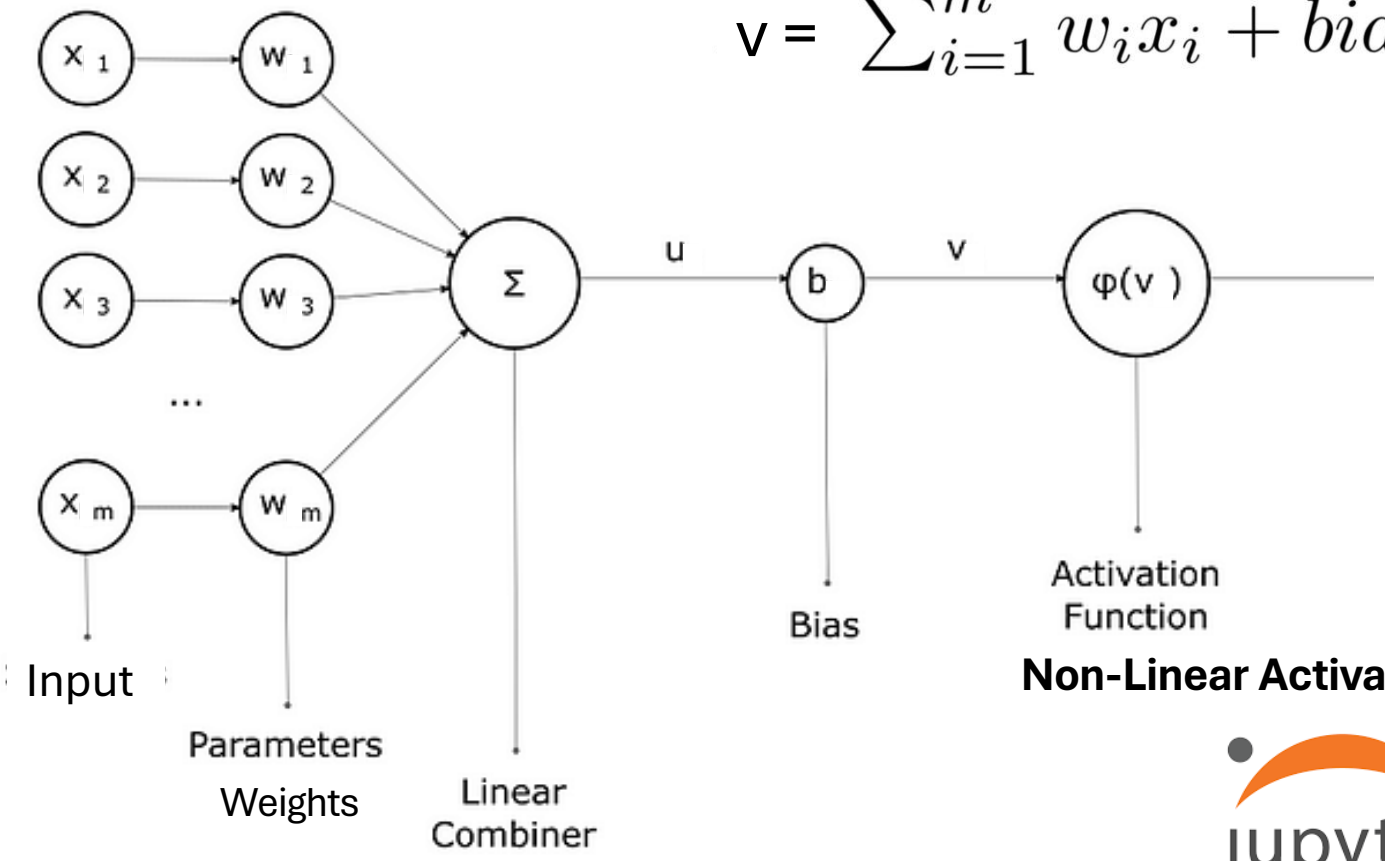
NN training

$$u = W \cdot X = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$

$y = ax$

$$v = \sum_{i=1}^{m} w_i x_i + bias$$

$y = ax + b$



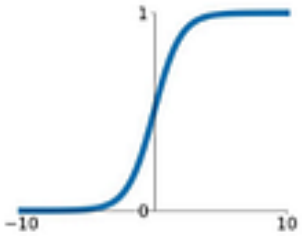
Non-Linear Activation Functions

$f(y) = f(ax + b)$

Activation Functions

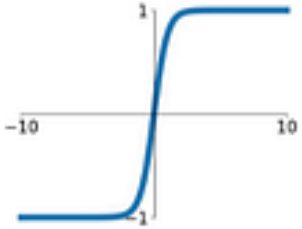
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



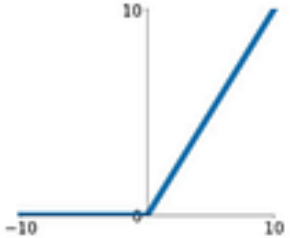
tanh

$\tanh(x)$



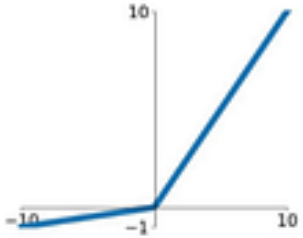
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

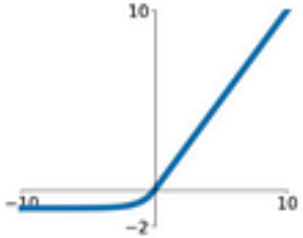


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



NN architecture

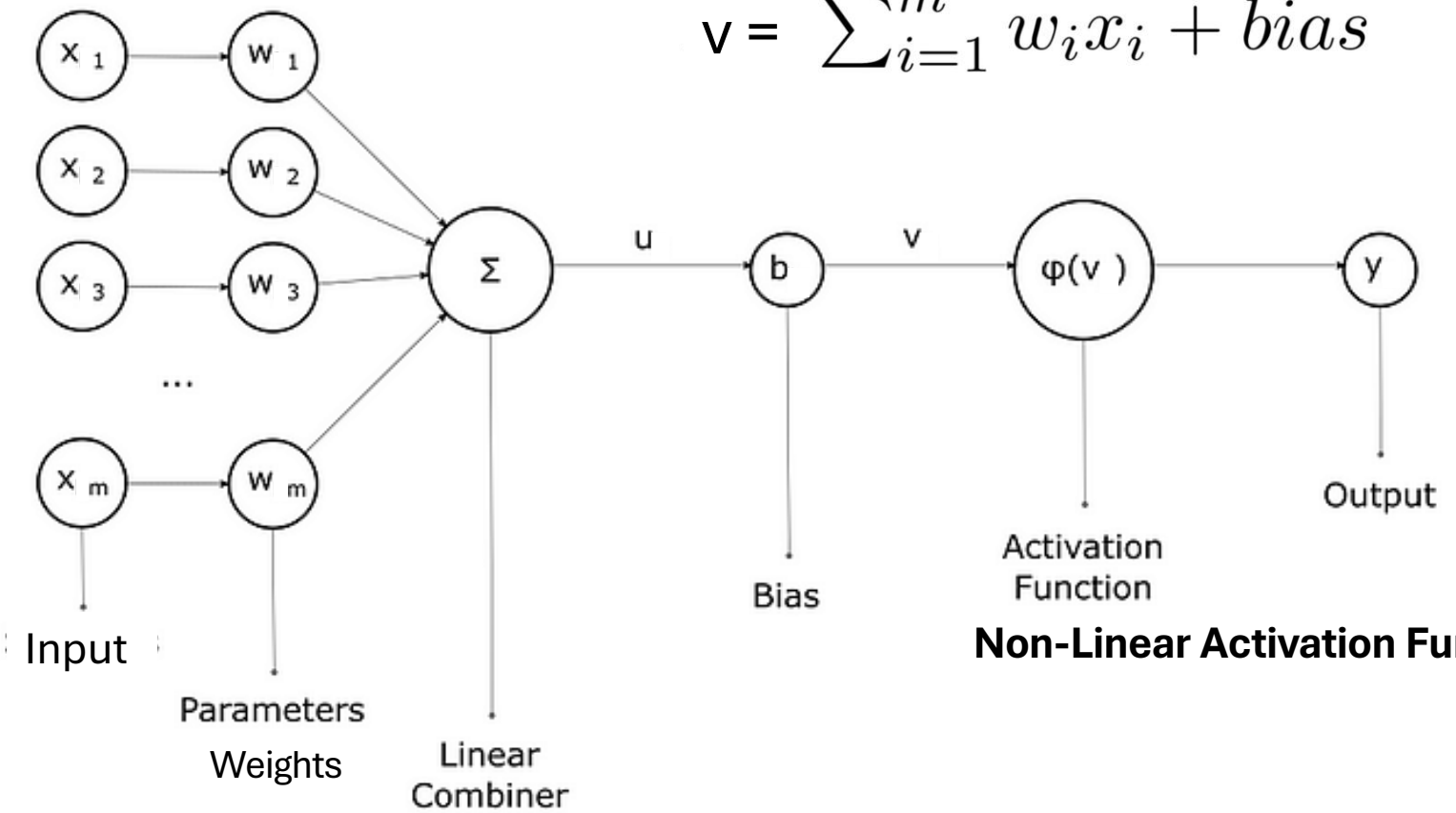
NN training

$$u = W \cdot X = w_1x_1 + w_2x_2 + \ldots + w_mx_m = \sum_{i=1}^{m} w_ix_i$$

$y = ax$

$$v = \sum_{i=1}^{m} w_ix_i + bias$$

$y = ax + b$



Non-Linear Activation Functions

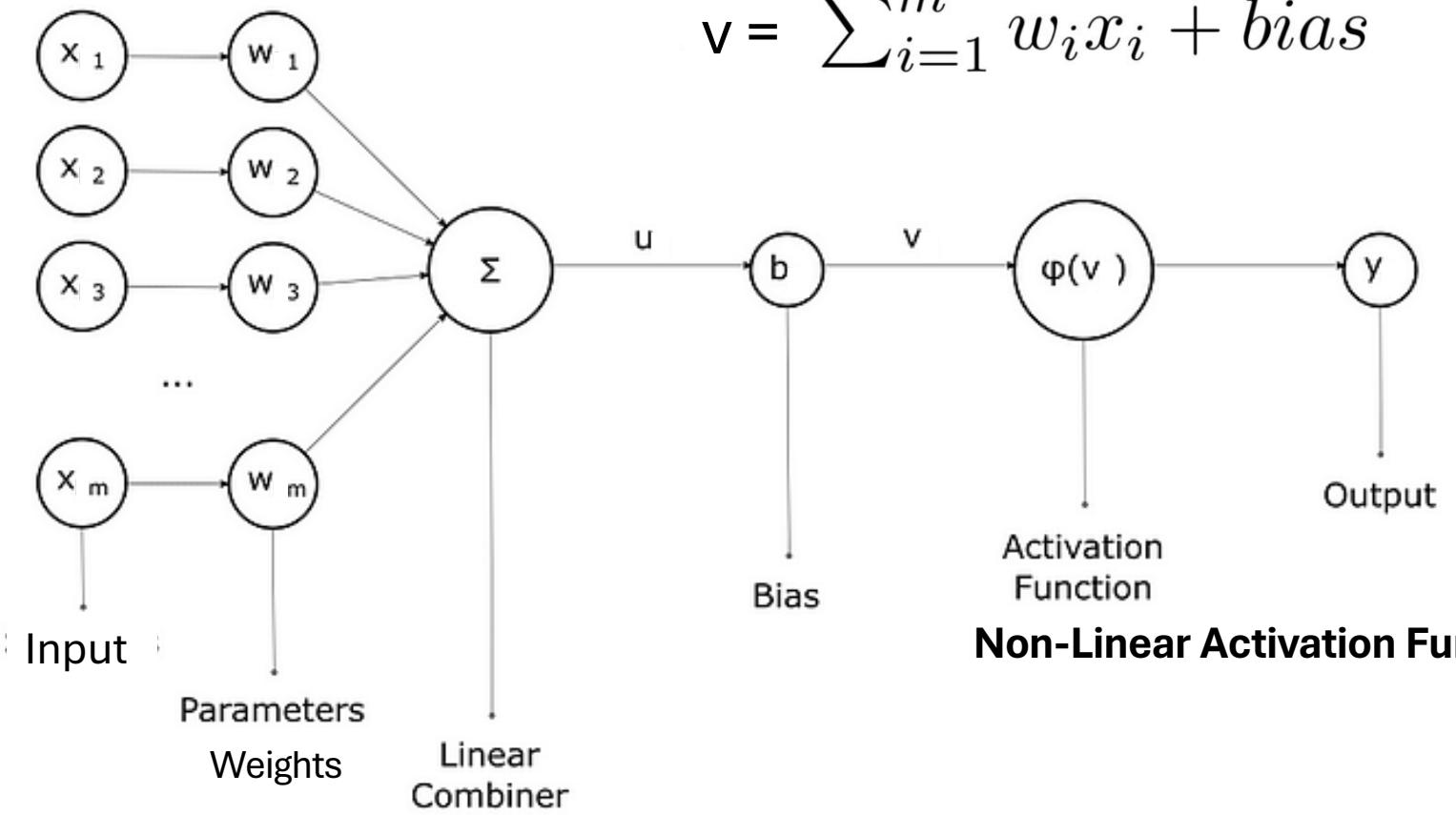
$f(y) = f(ax + b)$

NN architecture

NN training

$$u = W \cdot X = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$

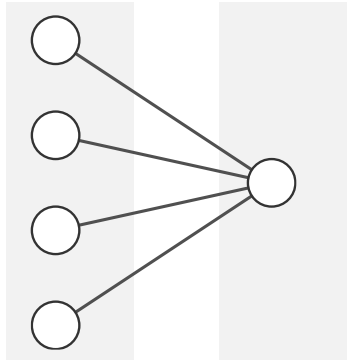
$$v = \sum_{i=1}^{m} w_i x_i + bias$$



Non-Linear Activation Functions

Simplest NN

Input layer Output layer

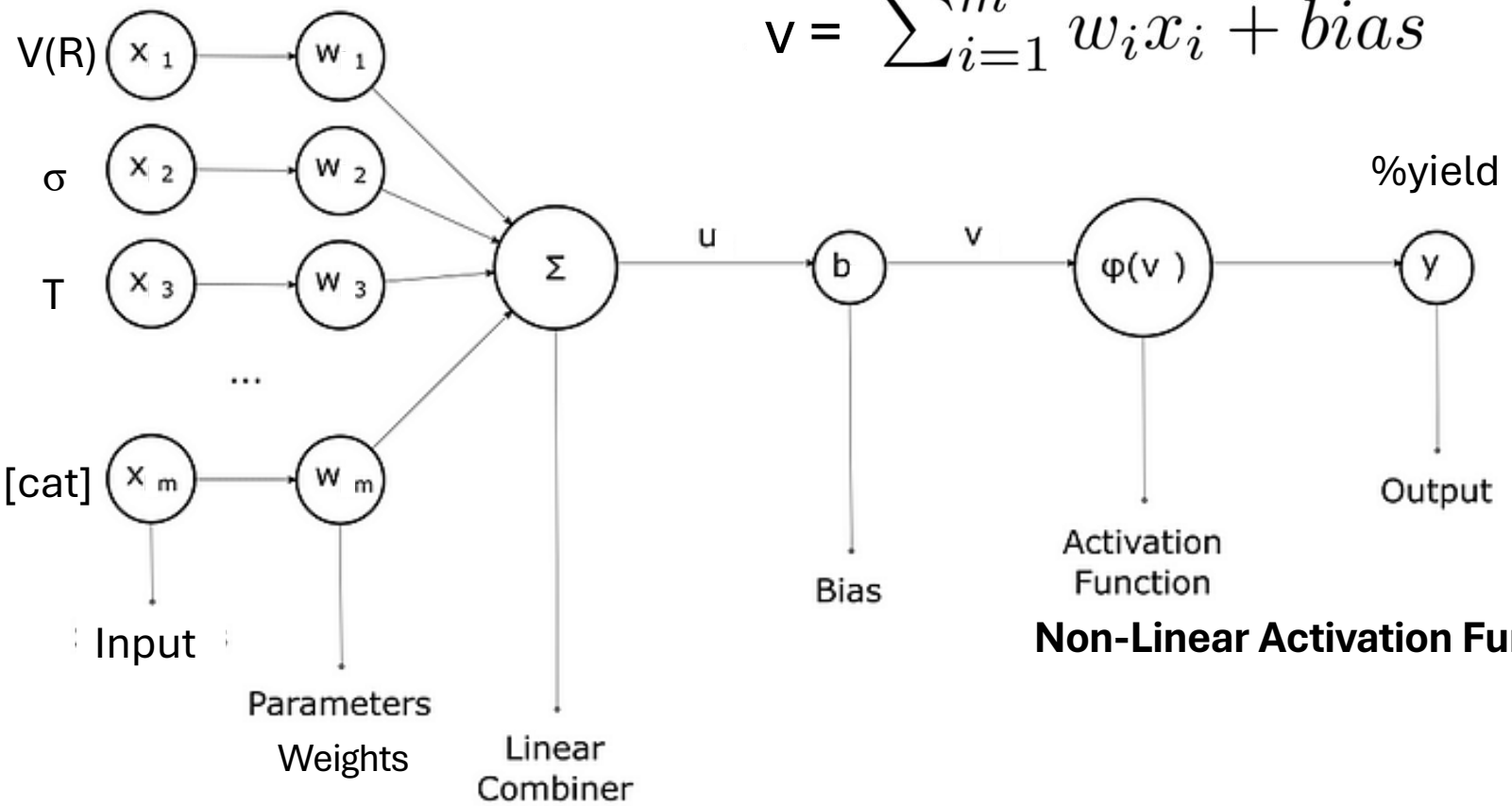


NN architecture

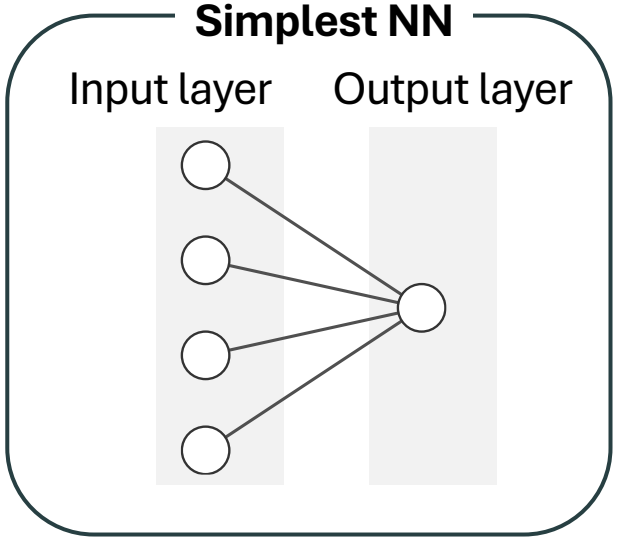
NN training

$$u = W \cdot X = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$

$$v = \sum_{i=1}^{m} w_i x_i + bias$$

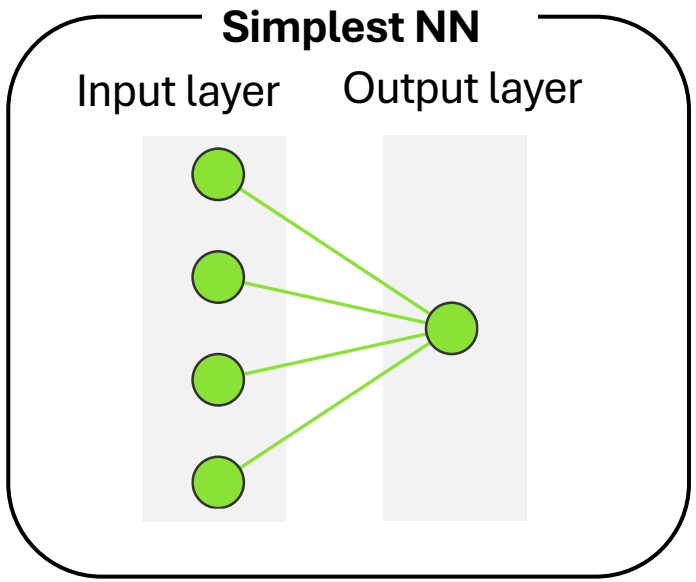


Non-Linear Activation Functions



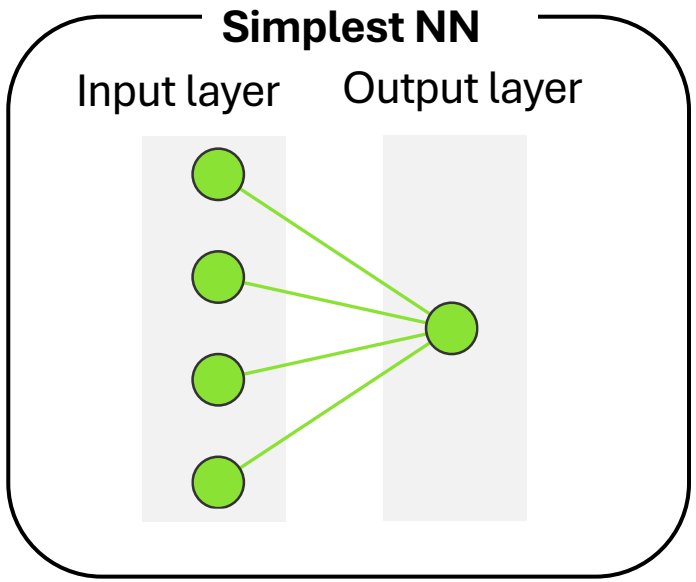
NN architecture

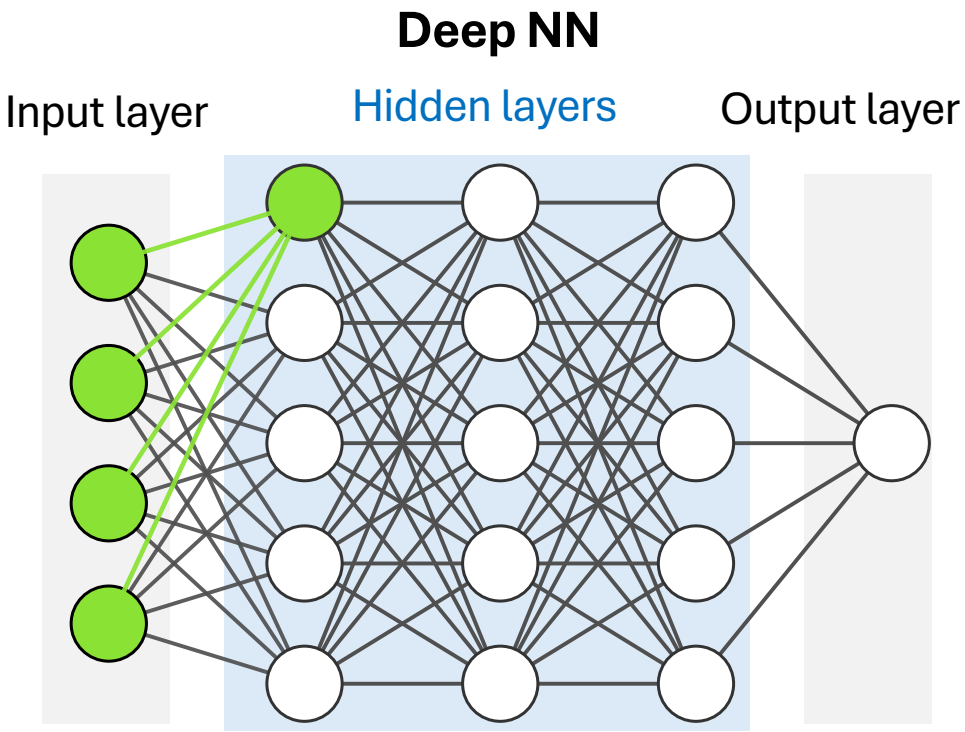
NN training



NN architecture

NN training





- Deep NN: neural network with one or more hidden layers
- The number of parameters increase exponentially
- Decisions to make for NN architecture: number of hidden layers, number of nodes per layer, the activation function → **Hyperparameters**, which are not trainable, there are MORE!

NN architecture

NN training

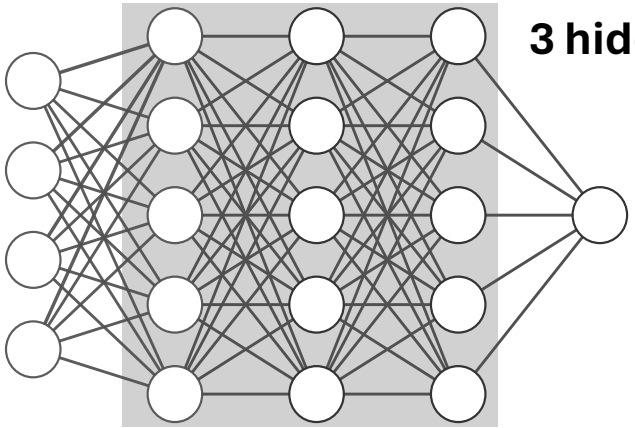
Hyperparameter	Chemistry n>1000	Comments
Number of hidden layers	2 - 3	
Number of hidden nodes	32, 64, 128, 256 based on heuristics	Between the input number of features and the output feature. **NB!** Over- and under- fitting
Activation function	ReLu	
Random state	42	Initialize the weights (W) and biases (b)

Training of the model: train the trainable parameters, weights (W) and biases (b)

NN architecture

NN training

Hyperparameter	Chemistry n>1000	Comments
Number of hidden layers	2 - 3	
Number of hidden nodes	32, 64, 128, 256 based on heuristics	Between the input number of features and the output feature. **NB!** Over- and under- fitting
Activation function	ReLu	
Random state	42	Initialize the weights (W) and biases (b)



3 hidden layers

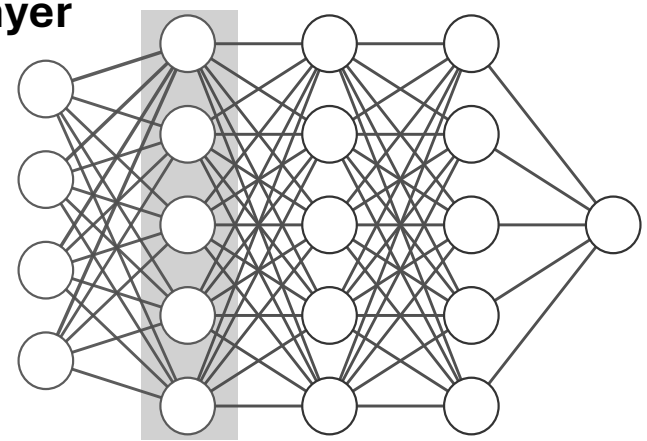
Training of the model: train the trainable parameters, weights (W) and biases (b)

NN architecture

NN training

Hyperparameter	Chemistry n>1000	Comments
Number of hidden layers	2 - 3	
Number of hidden nodes	32, 64, 128, 256 based on heuristics	Between the input number of features and the output feature. **NB!** Over- and under- fitting
Activation function	ReLu	
Random state	42	Initialize the weights (W) and biases (b)

5 nodes per hidden layer



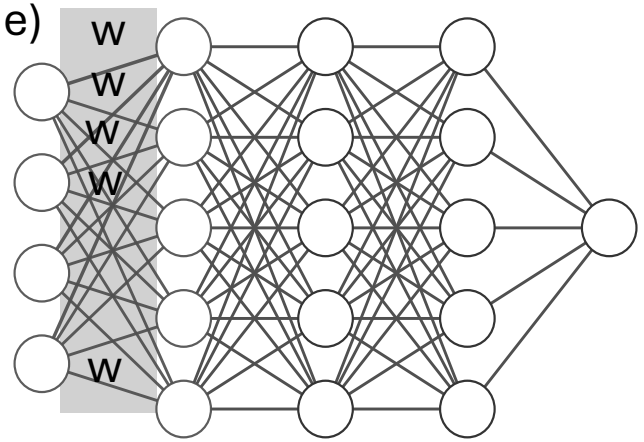
Training of the model: train the trainable parameters, weights (W) and biases (b)

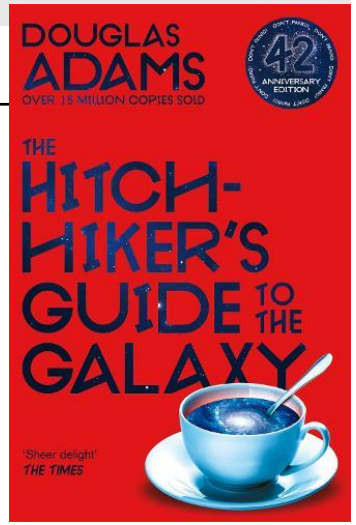
NN architecture

NN training

Hyperparameter	Chemistry n>1000	Comments
Number of hidden layers	2 - 3	
Number of hidden nodes	32, 64, 128, 256 based on heuristics	Between the input number of features and the output feature. **NB!** Over- and under- fitting
Activation function	ReLu	
Random state	42	Initialize the weights (W) and biases (b)

Initial random **w** values (1 w/line)





Training of the model: train the trainable parameters, weights (W) and biases (b)

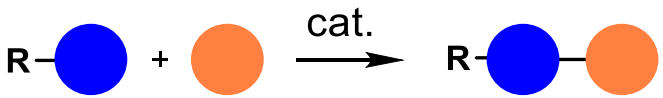
NN architecture

NN training

Hyperparameter	Chemistry n>1000	Comments
Number of hidden layers	2 - 3	
Number of hidden nodes	32, 64, 128, 256 based on heuristics	Between the input number of features and the output feature. **NB!** Over- and under- fitting
Activation function	ReLu	
Random state	42	Initialize the Weights (W) and biases (b)
Number of epochs		
Loss function		
Learning rate		
Solver		
Batch size (optional)		
Validation (optional)		

Training of the model: train the trainable parameters, weights (W) and biases (b)

NN training



X = sterics of R, Hammet σ, T, [cat],...
features

y = %yield
target property

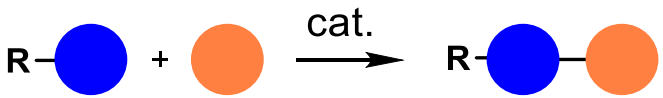
Training set: X, y_{real}	80%
Valid. set X, y_{real}	10%
Test set X, y_{real}	10%

Already known information – dataset

Reactions	X = sterics of R, Hammet σ, T, ...	y = %yield
Reaction 1	Volume(-Ph_3), σ(-Ph_3), 100ºC, 1mM	80%
Reaction 2	Volume(-Me), σ(-Me), 100ºC, 3mM	20%
Reaction 3	Volume(-CF_3), σ(-CF_3), 25ºC, 5mM	67%
...		...
Reaction n	Volume(-NO_2), σ(-NO_2), 25ºC, 3mM	45%

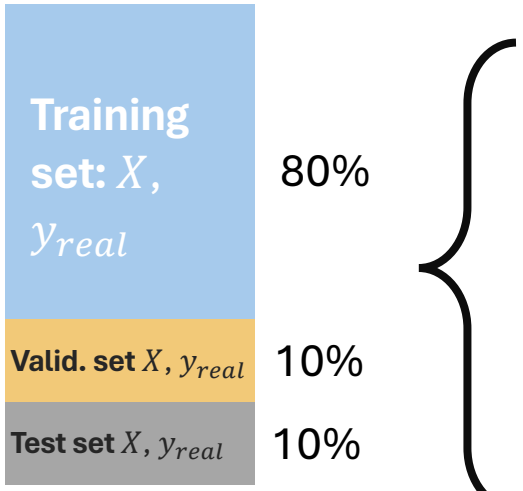
New reactions	X = sterics of R, Hammet σ, T, ...	y = pred(%yield)
Reaction --	Volume(-R), σ(-R), 120ºC, 2mM	% ---

NN training



X = sterics of R, Hammet σ, T, [cat],...
features

y = %yield
target property



Reactions	X = sterics of R, Hammet σ, T, ...	y = %yield
Reaction 1	Volume(-Ph_3), σ(-Ph_3), 100ºC, 1mM	80%
Reaction 2	Volume(-Me), σ(-Me), 100ºC, 3mM	20%
Reaction 3	Volume(-CF_3), σ(-CF_3), 25ºC, 5mM	67%
...		...
Reaction n	Volume(-NO_2), σ(-NO_2), 25ºC, 3mM	

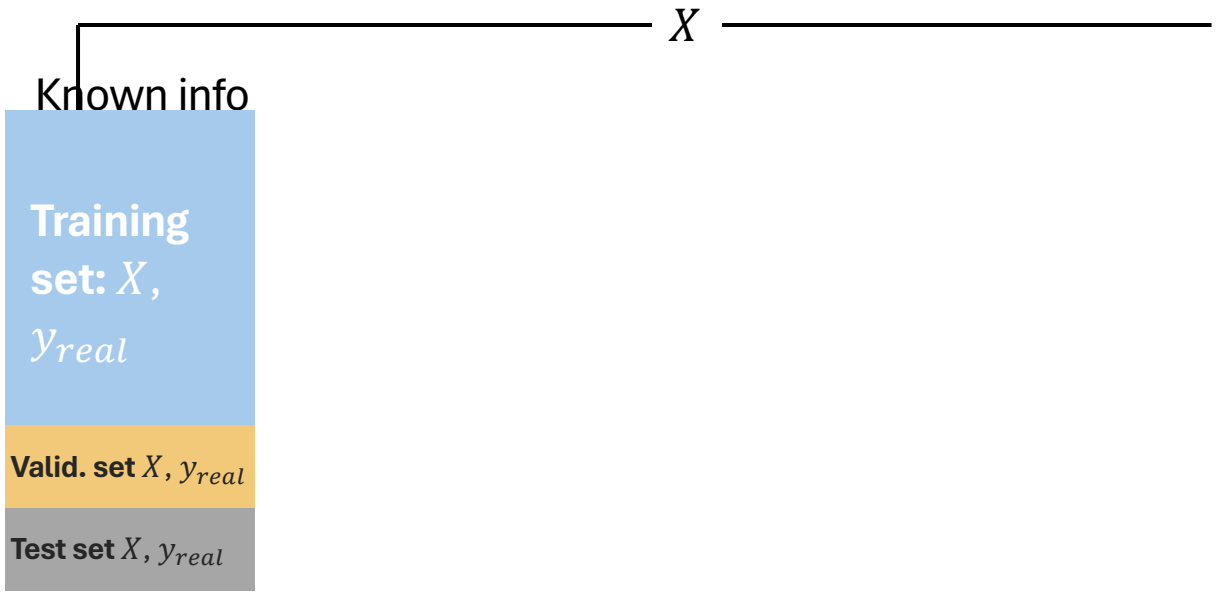
Training of the model

Already known information – dataset

New reactions	X = sterics of R, Hammet σ, T, ...	y = pred(%yield)
Reaction --	Volume(-R), σ(-R), --ºC, --mM	% ---

NN training

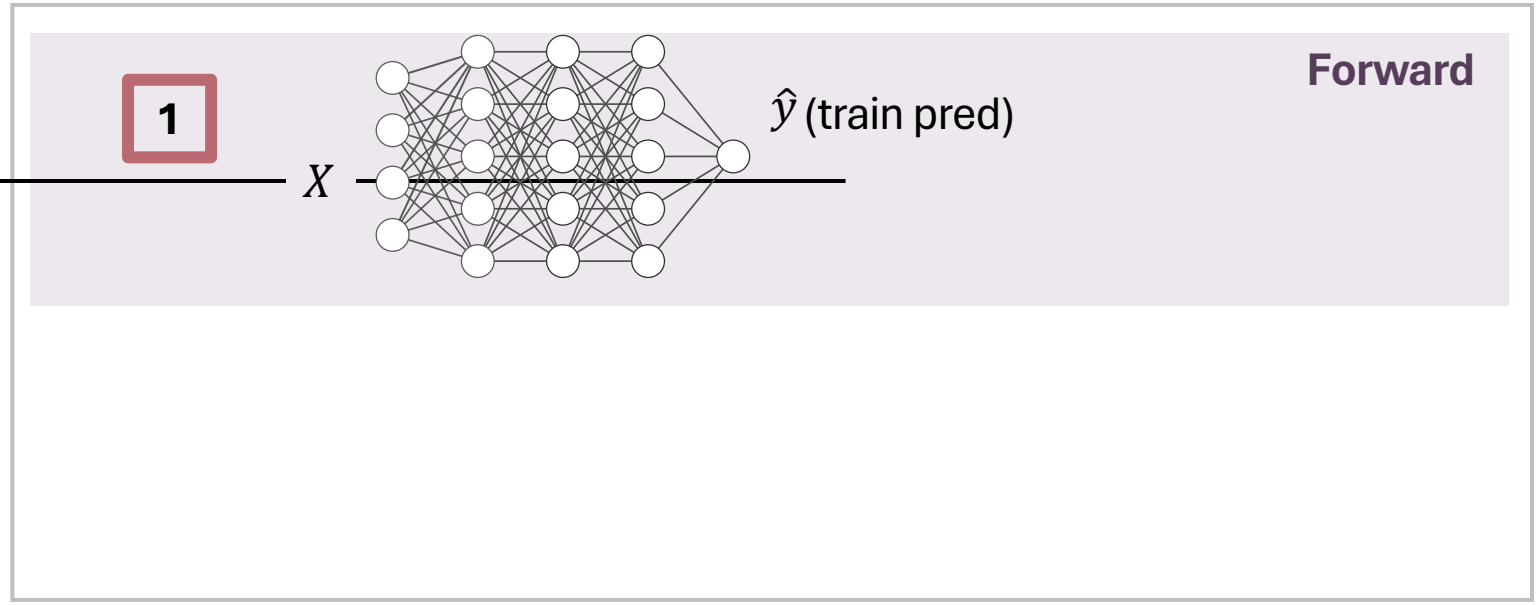
EPOCH 1
(each iteration)



NN training

EPOCH 1
(each iteration)

Training of the model



Training set: X, y_{real}

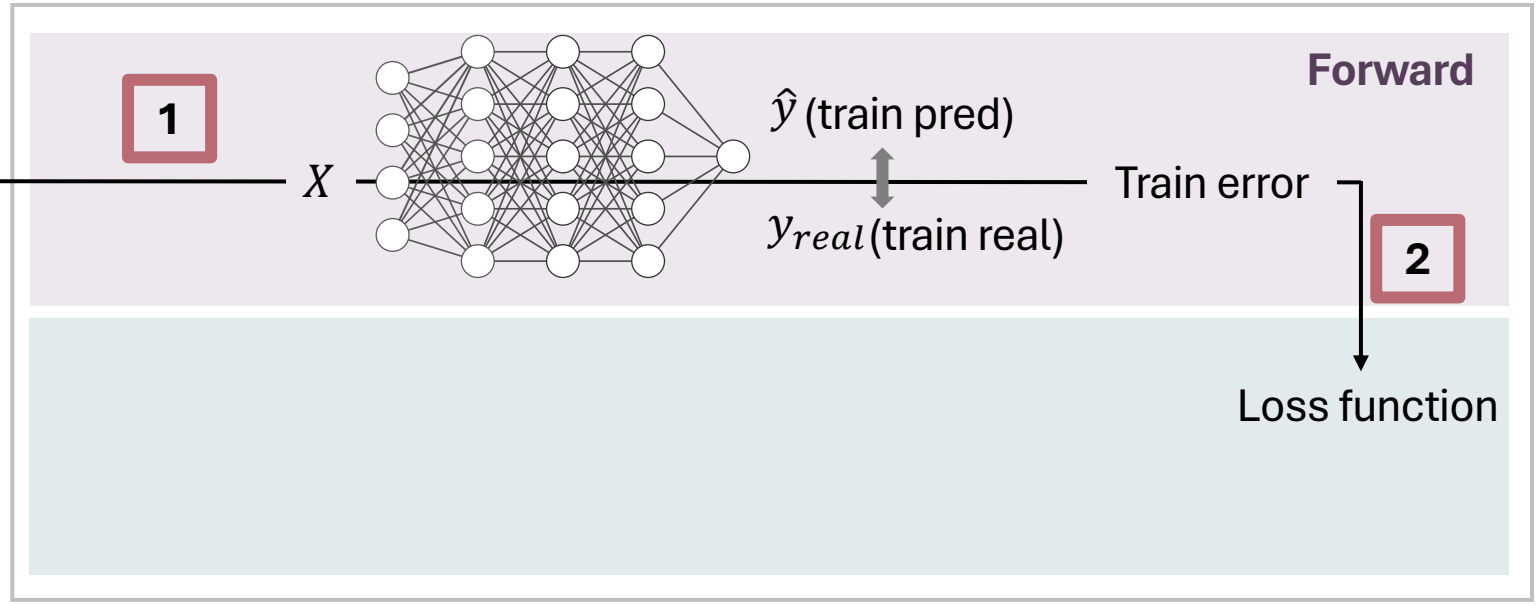
Valid. set X, y_{real}

Test set X, y_{real}

NN training

EPOCH 1
(each iteration)

Training of the model



Training set: X, y_{real}

Valid. set X, y_{real}

Test set X, y_{real}

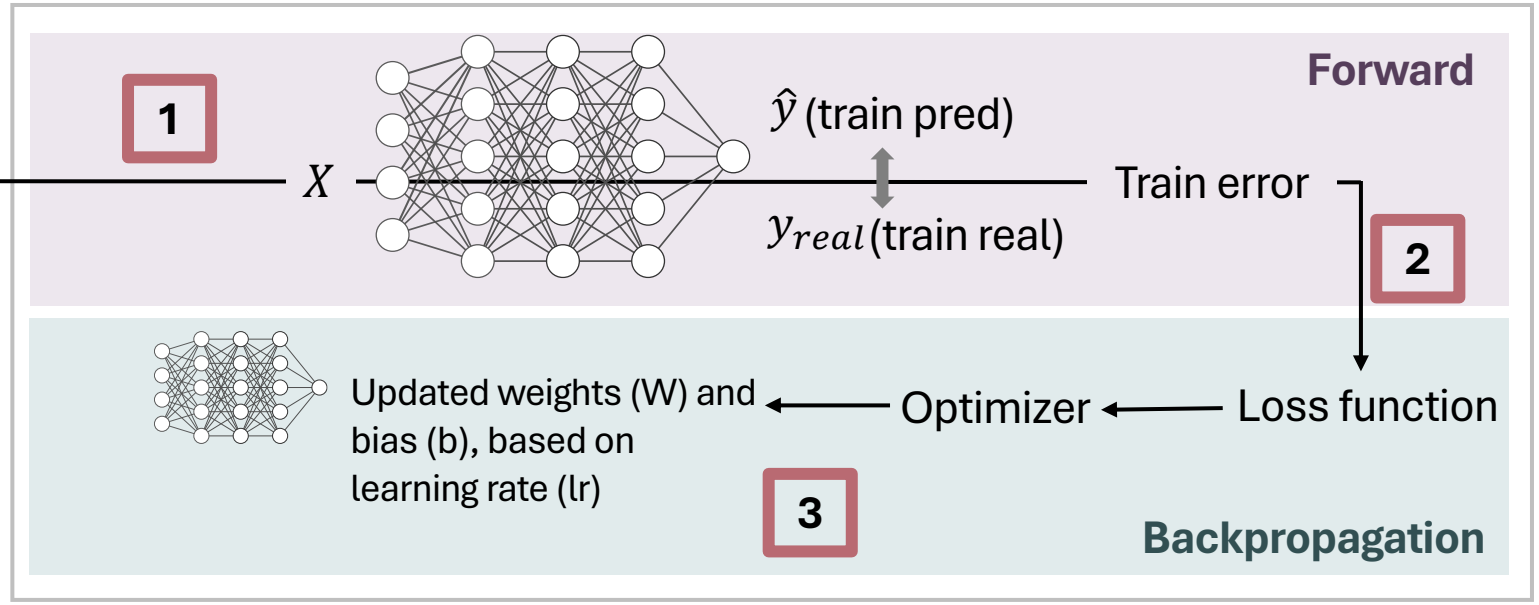
Loss function: quantifies the error of the model: $\hat{y} - y_{real}$
E.g. mean squared error (MSE)

GOAL: *reduce loss during the training*

NN training

EPOCH 1
(each iteration)

Training of the model



Training set: X, y_{real}

Valid. set X, y_{real}

Test set X, y_{real}

Loss function: quantifies the error of the model. $\hat{y} - y_{real}$
E.g. mean squared error (MSE)

GOAL: *reduce loss during the training*

Optimizer determines the direction in which to optimize the trainable parameters. Updates W and b based on the learning rate (η)

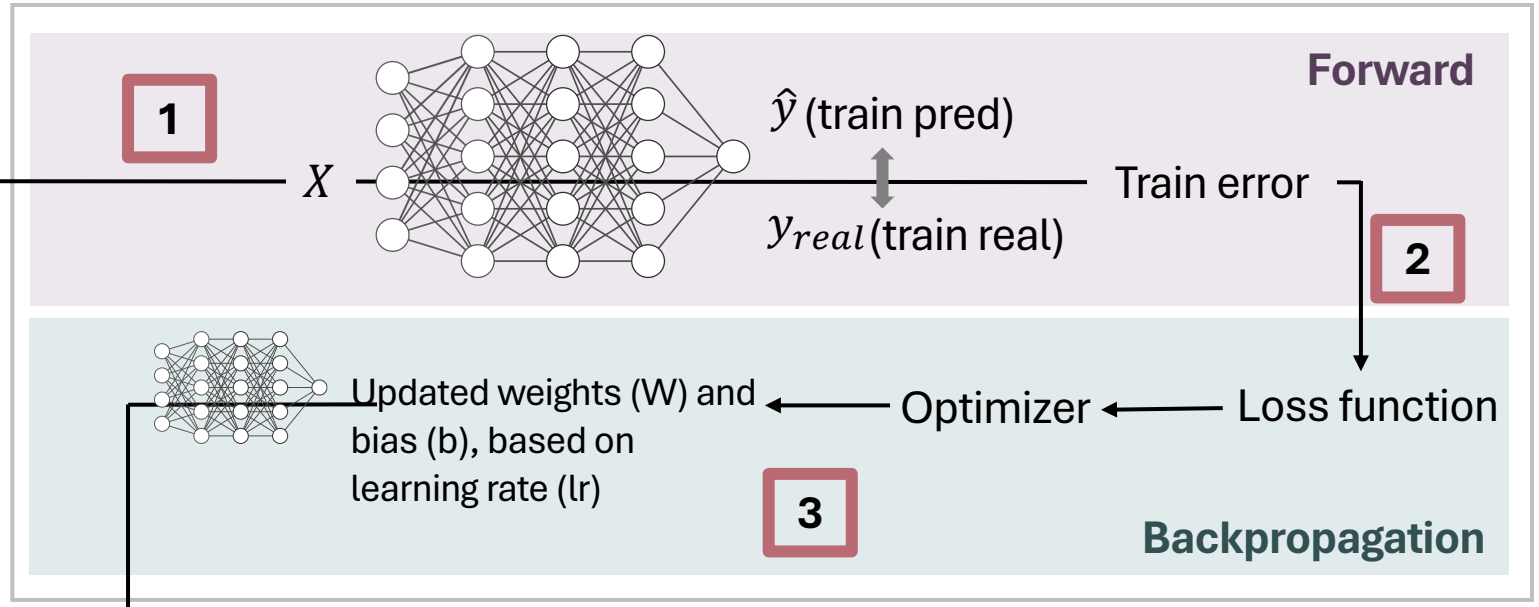
$$w_{new} = w - \eta * \frac{\partial out}{\partial w}$$

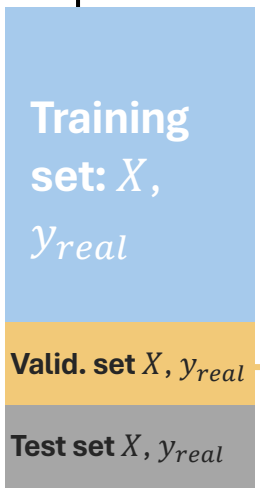
GOAL: *smoothly reduce the performance error*

NN training

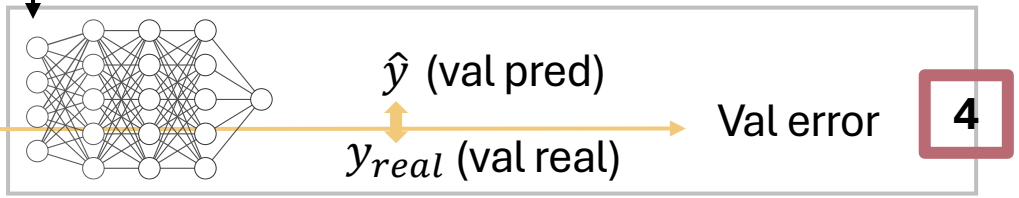
EPOCH 1
(each iteration)

Training of the model





Evaluate Model Performance – Blind check



Loss function: quantifies the error of the model. $\hat{y} - y_{real}$
E.g. mean squared error (MSE)

GOAL: *reduce loss during the training*

Optimizer determines the direction in which to optimize the trainable parameters. Updates W and b based on the learning rate (η)

$$w_{new} = w - \eta * \frac{\partial out}{\partial w}$$

GOAL: *smoothly reduce the performance error*

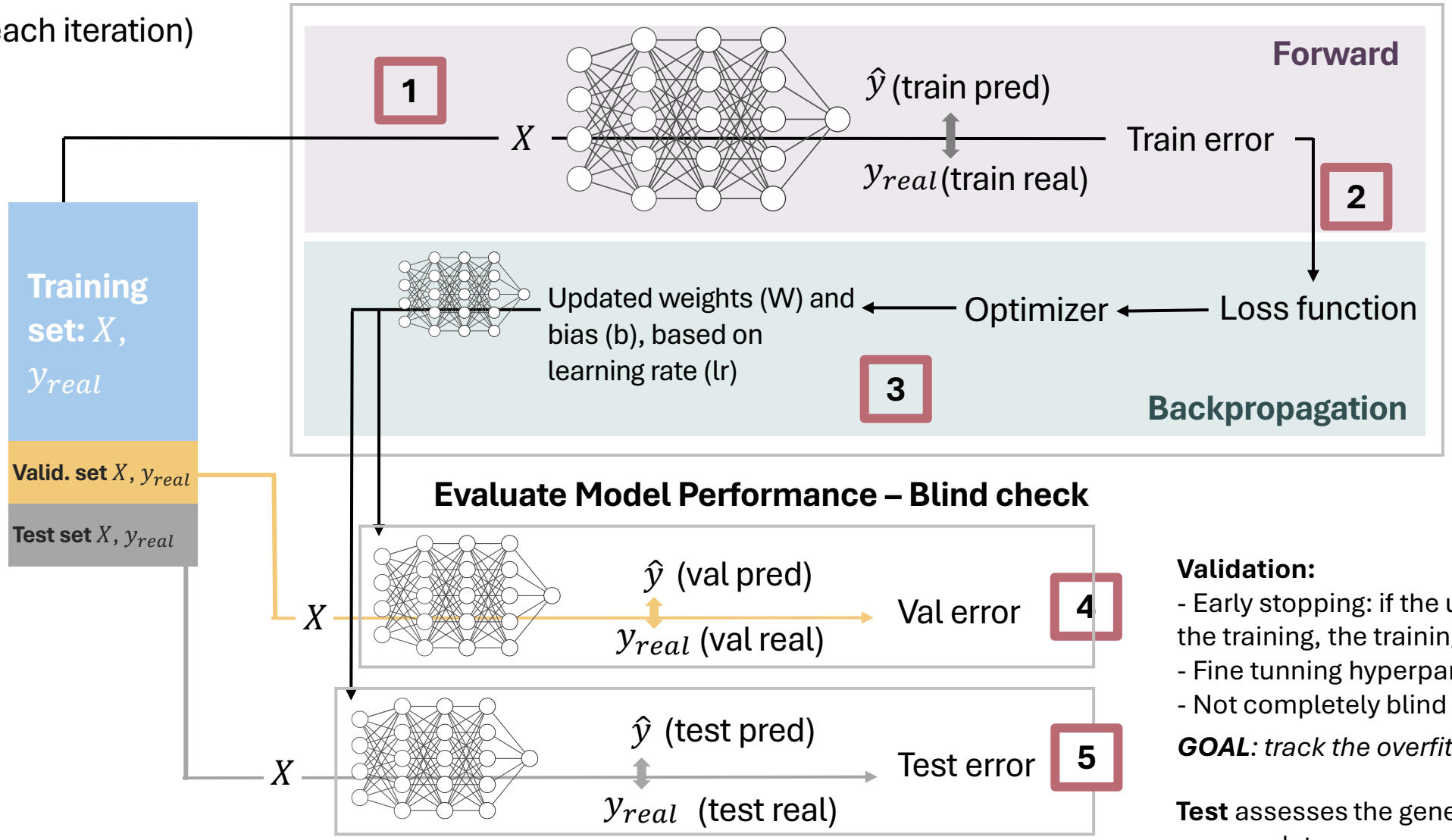
Validation:
- Early stopping: if the updated W and b are worse after the training, the training is stopped.
- Fine tunning hyperparameters: η
- Not completely blind

GOAL: *track the overfitting*

NN training

EPOCH 1
(each iteration)

Training of the model



Loss function: quantifies the error of the model. $\hat{y} - y_{real}$
E.g. mean squared error (MSE)

GOAL: *reduce loss during the training*

Optimizer determines the direction in which to optimize the trainable parameters. Updates W and b based on the learning rate (η)

$$w_{new} = w - \eta * \frac{\partial out}{\partial w}$$

GOAL: *smoothly reduce the performance error*

Validation:
- Early stopping: if the updated W and b are worse after the training, the training is stopped.
- Fine tunning hyperparameters: η
- Not completely blind

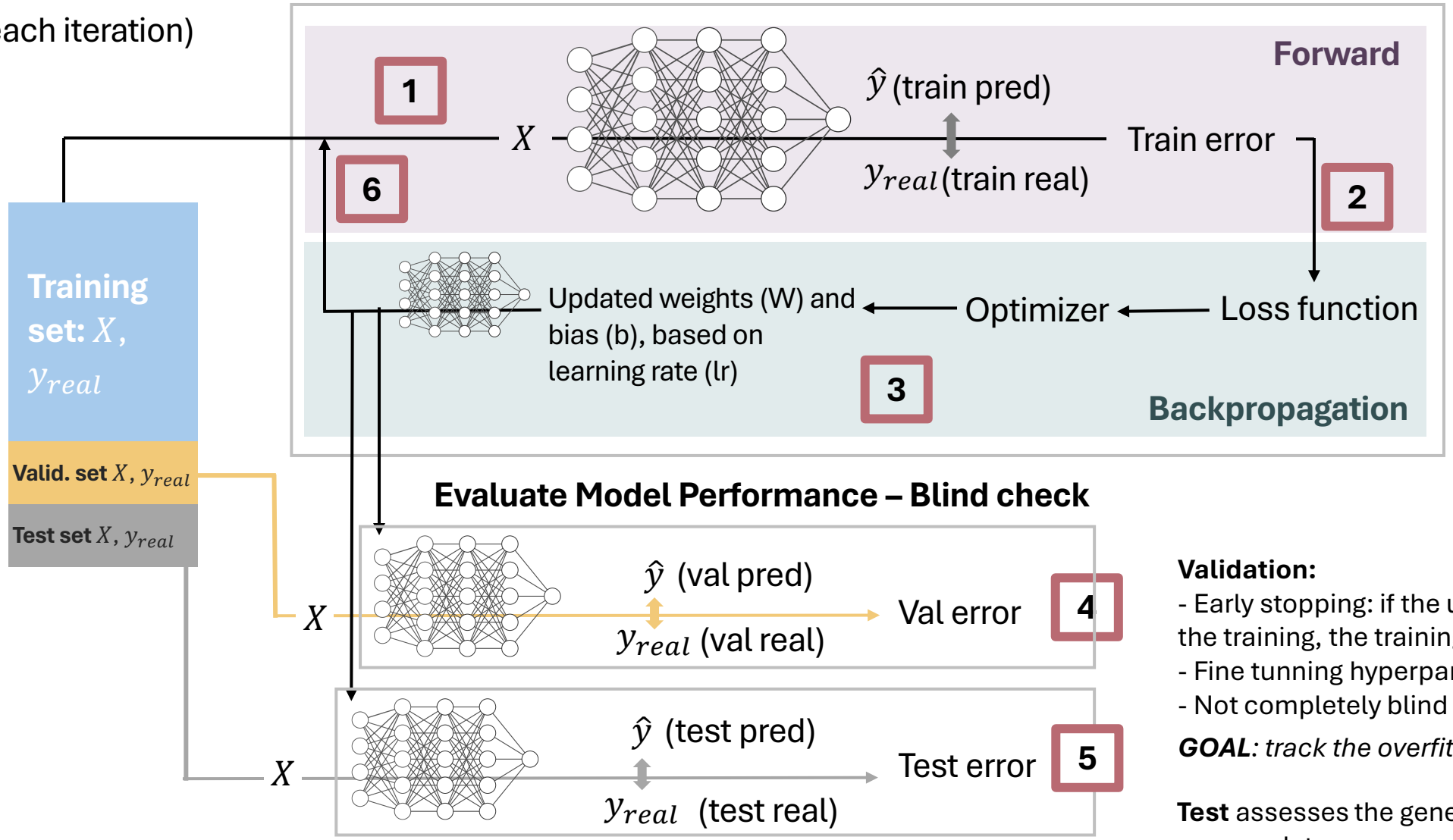
GOAL: *track the overfitting*

Test assesses the generalization of the model to unseen data.

NN training

EPOCH 1
(each iteration)

Training of the model



Loss function: quantifies the error of the model. $\hat{y} - y_{real}$
E.g. mean squared error (MSE)

GOAL: *reduce loss during the training*

Optimizer determines the direction in which to optimize the trainable parameters. Updates W and b based on the learning rate (η)

$$w_{new} = w - \eta * \frac{\partial out}{\partial w}$$

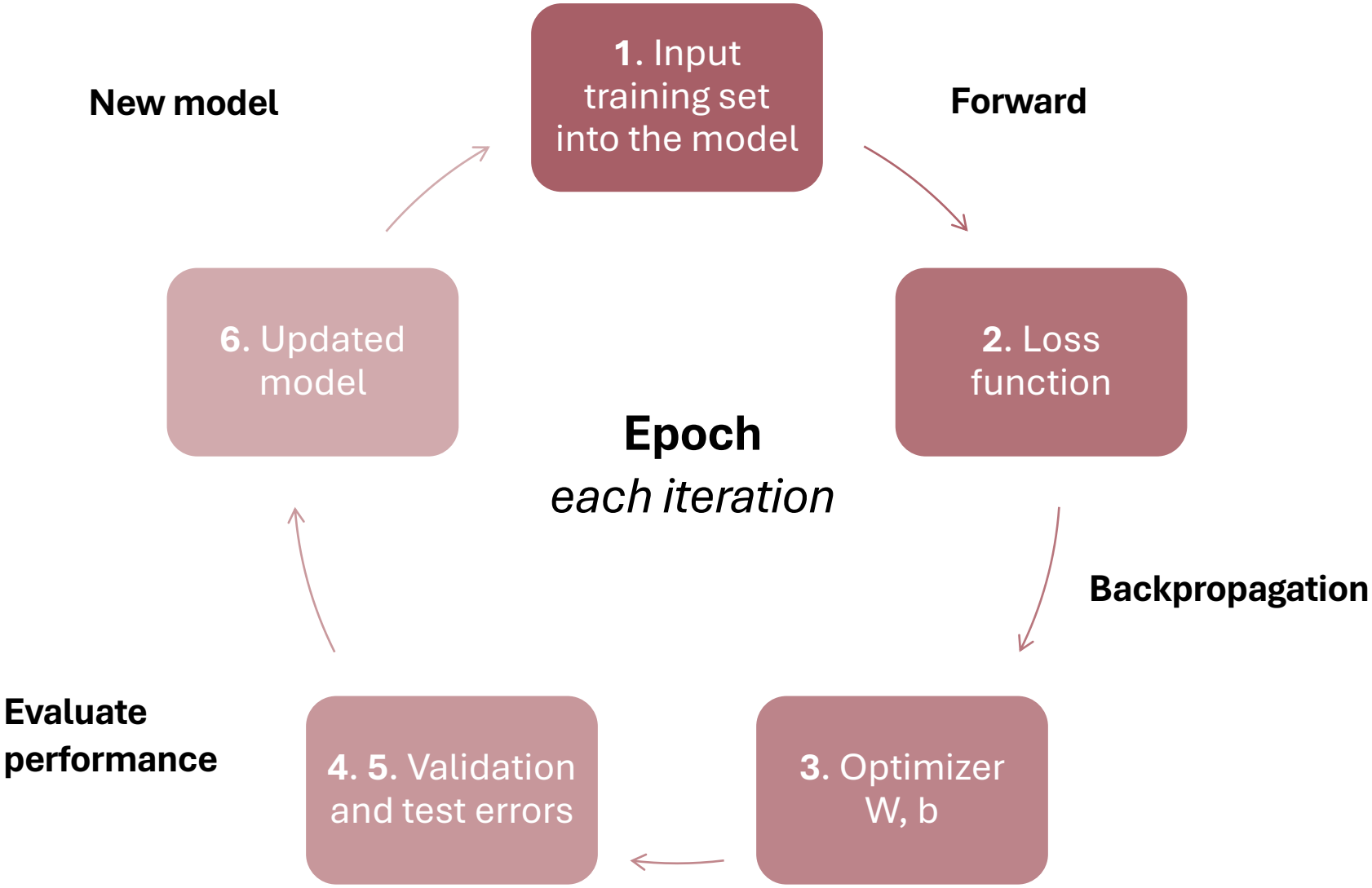
GOAL: *smoothly reduce the performance error*

Validation:
- Early stopping: if the updated W and b are worse after the training, the training is stopped.
- Fine tunning hyperparameters: η
- Not completely blind

GOAL: *track the overfitting*

Test assesses the generalization of the model to unseen data.

NN training

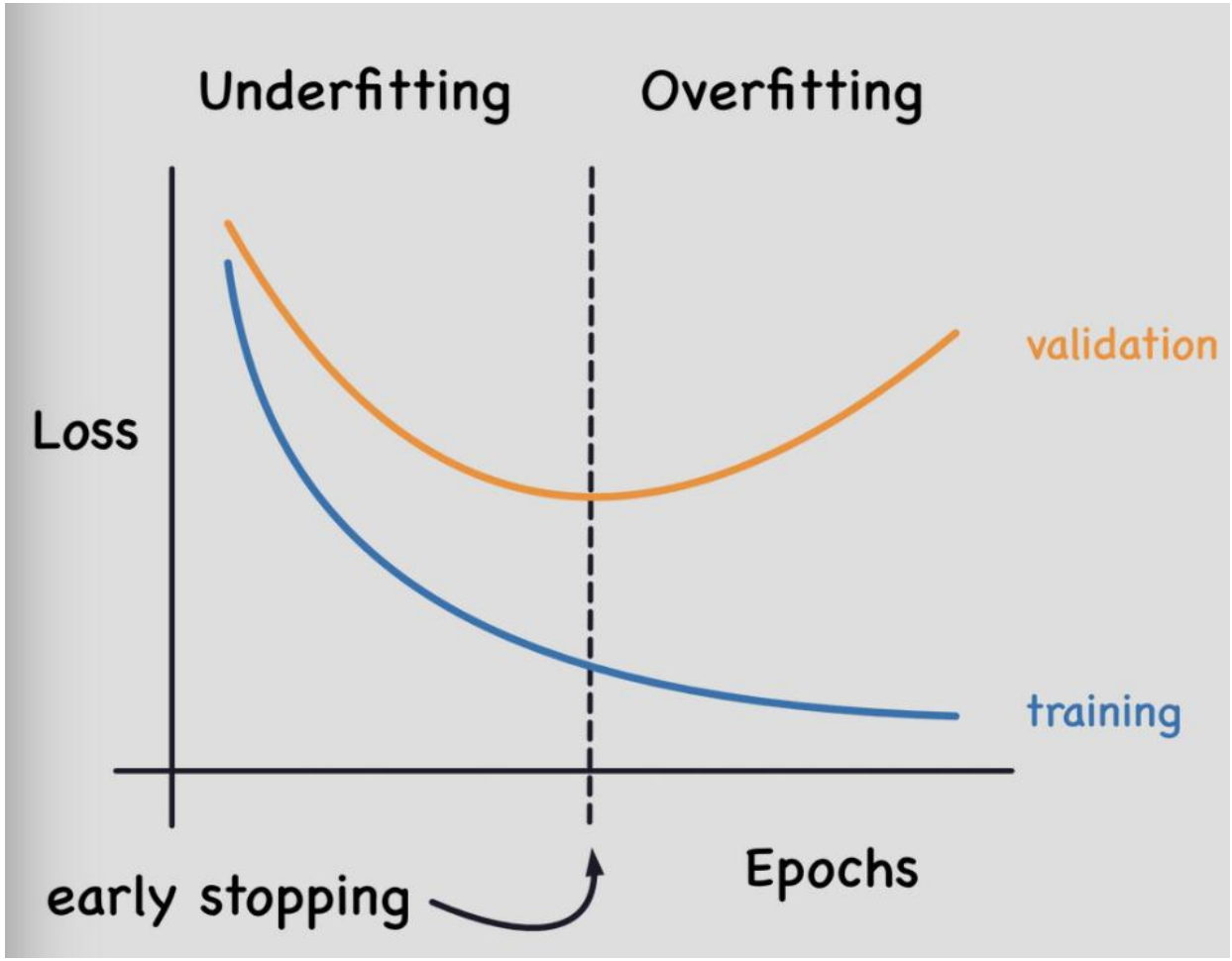

1. Input training set into the model
Forward
2. Loss function
Backpropagation
3. Optimizer W, b
4. 5. Validation and test errors
Evaluate performance
6. Updated model
New model
Epoch
each iteration

NN architecture

NN training

Hyperparameter	Chemistry n>1000	Comments
Number of hidden layers	2 - 3	
Number of hidden nodes	32, 64, 128, 256 based on heuristics	Between the input number of features and the output feature. **NB!** Over- and under- fitting
Activation function	ReLu	
Random state	42	Initialize the Weights (W) and biases (b)
Number of epochs	50-200	Training iterations
Loss function	Mean-squared error	Measure the error
Learning rate (η)	0.0001 to 0.1	Adam: 0.001
Solver	Adam	Commonly used
Batch size (optional)	32, 64, 128, 256	Divide training set to speed up the training process
Validation (optional)	Early stopping	Stop if validation loss does not improve for 10 consequtive epochs

NN training



Validation:

- Early stopping: if the updated W and b are worse after the training, the training is stopped.
- Fine tunning hyperparameters: η
- Not completely blind

***GOAL**: track the overfitting*

Play around – NN model

https://playground.tensorflow.org/

Playground.tensorflow.org

ML API

	scikit learn	PyTorch	TensorFlow
Use	Traditional ML algorithms Deep learning with MLP	Deep learning and AI	Deep learning
Ease of use	Simple	Pythonic syntax	Keras helps
Performance	Small to medium datasets No GPU	GPU acceleration	GPU acceleration
Developers	Extension of SciPy	Meta AI, now Linux Foundation	Google

Types of NN

- Feed forward NN (FNN) : fixed-length vector

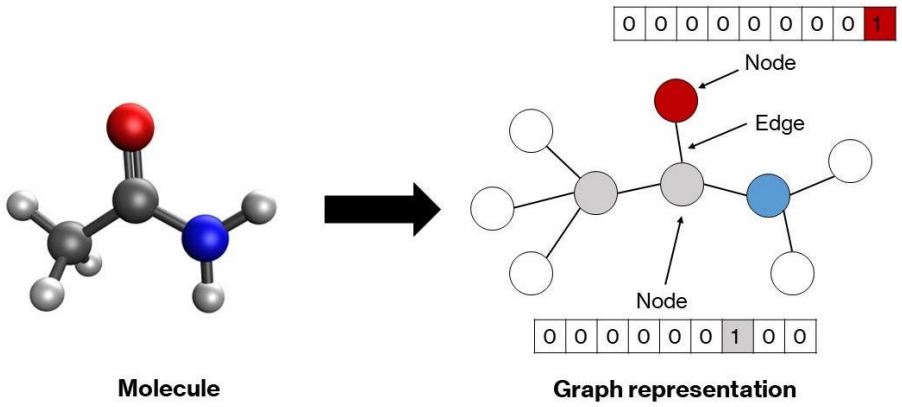
[10, 3, 5, 20, 82] = molecule

Types of NN

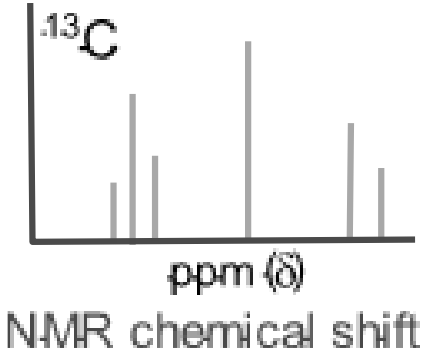
- Feed forward NN (FNN) : fixed-length vector

[10, 3, 5, 20, 82] = molecule

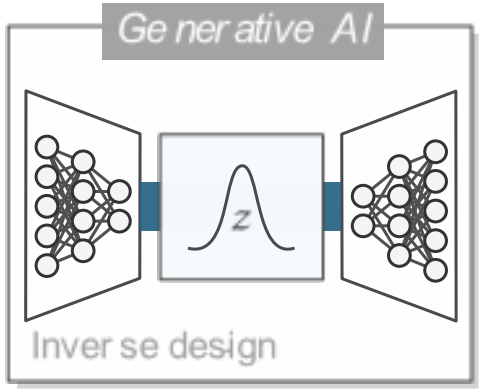
- Graph NN (GNN) : input is a graph – molecular graph (consider all the connections)



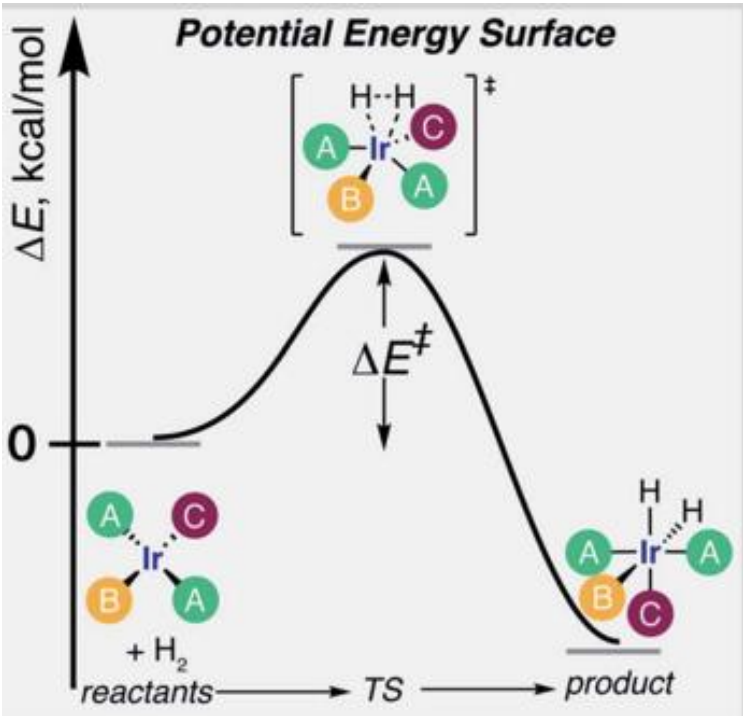
- Convolutional NN: grid of pictures

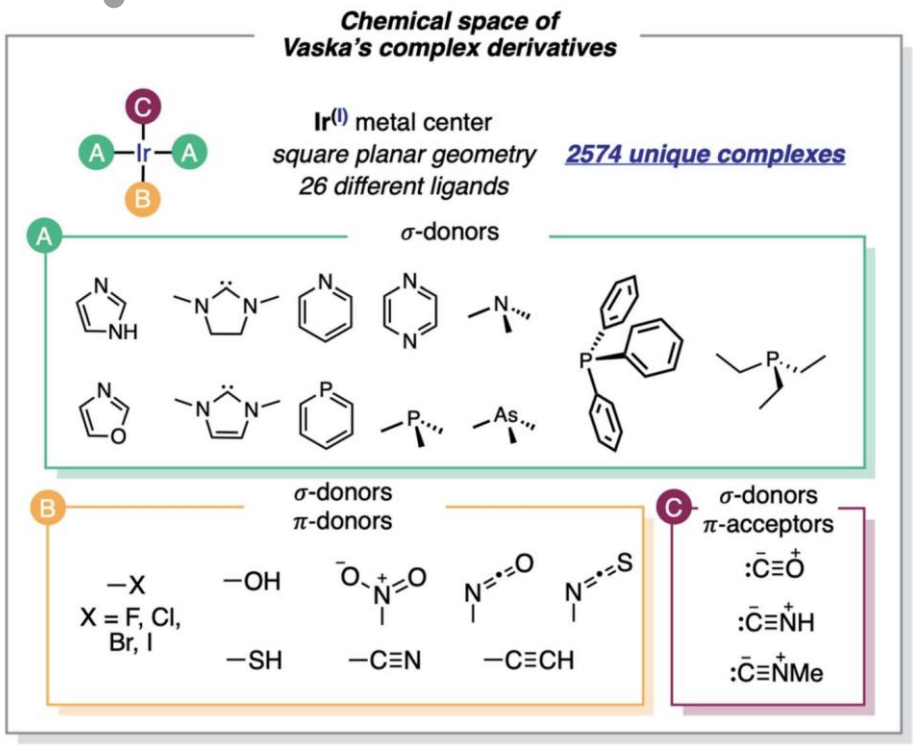


- Variational autoenconder (VAEs)



Example in chemistry



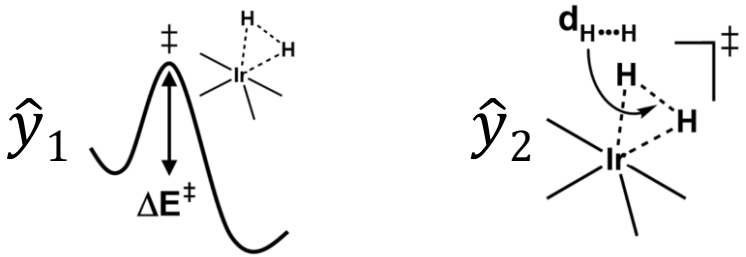


Chemical space
1947 Ir complexes

Too computationally demanding

Prediction with NN model

Target properties (y)



Input properties (X)

Generic properties

$$x_1, x_2, \ldots, x_n$$

Chem. Sci., **2020**, *11*, 4584–4601 | 4587