AI applied to Granular Systems

Stefan Radl

> April 26, 2023, TUSAIL School “EVENT 4”

Schedule

(1) **AI Basics**

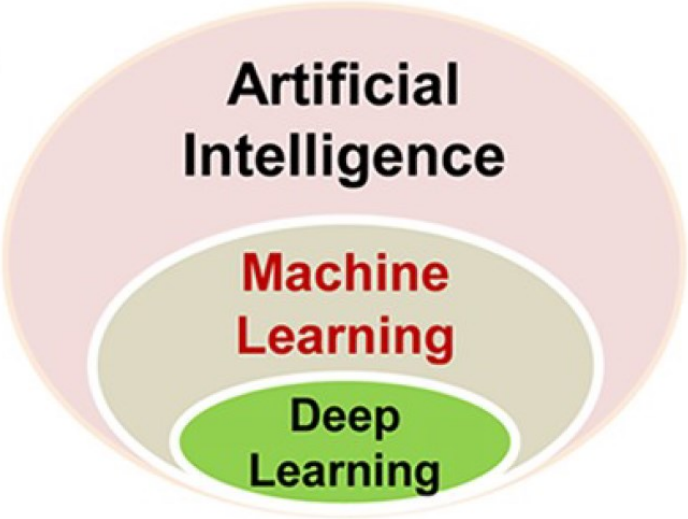
(2) **ML in Flow & Granular Systems**

(3) **Exercise (Orange3)**

(4) **Feedback**

Break(s) as you wish…

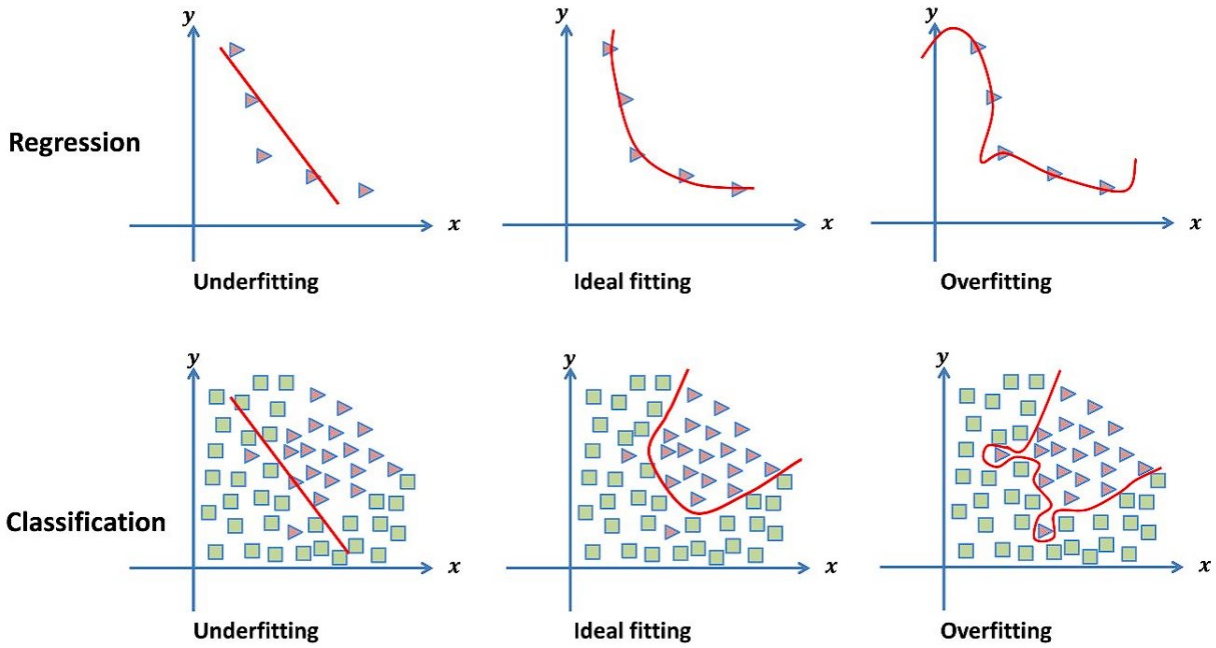
Naming Basics



- Artificial intelligence (**AI**), early „**expert systems**“ (rule-based algorithms), **machine learning** (ML), and Deep Learning (**DNN**) are separate things!
- „**data-driven**“ is often synonomous for ML. **We will focus on ML today**
- Most of the time **people mean ML** and not the larger topic AI

Zhu et al., *Ind. Eng. Chem. Res.* 2022, 61, 9901−9949

Naming Basics



- **Classification** versus **Regression**
- **Overfitting** versus **Underfitting**

Zhu et al., *Ind. Eng. Chem. Res.* 2022, 61, 9901−9949

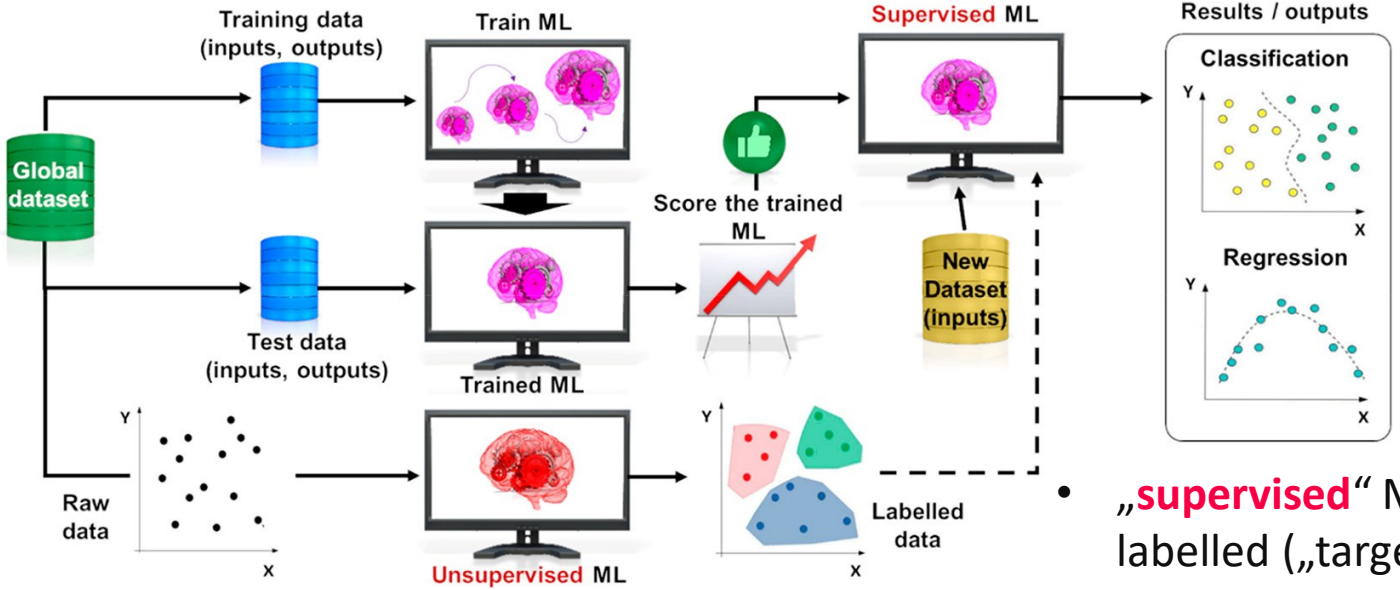
Naming Basics

	Name	Type	Role	Values
1	sepal length	N numeric	feature	
2	sepal width	N numeric	feature	
3	petal length	N numeric	feature	
4	petal width	N numeric	feature	
5	iris	C categorical	target	Iris-setosa, Iris-versicol

Orange3 „iris“ example

- an „**instance**“ is one row in the dataset. It consists of **features (marker)** and **target(s)**.
- a **feature** is a descriptor of a certain state of a system, and hence the input („**what we know**“, „**what is easy to evaluate**“)
- a **target** is what „we want to predict“. In case we have a **classification** problem, the target is a **category** (or class, or type)
- „**sample**“ is a set of features we give as an input

Naming Basics



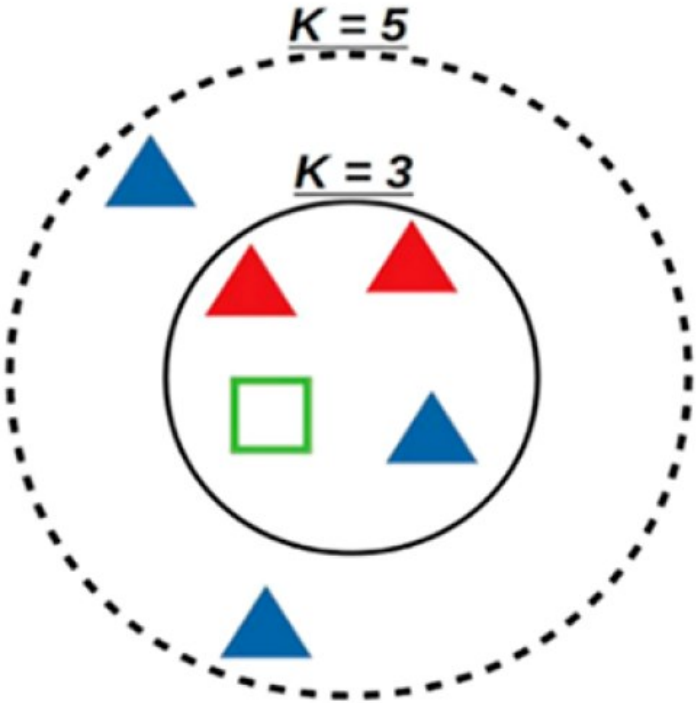
- „**supervised**“ ML uses already labelled („targeted“) data.
- „**unsupervised**“ ML aims at the identification of labels. It attempts to label the „unseen“

Lombardo et al., *Chem. Rev., 122, 2022*

Naming Basics

- the simple **k-Nearest Neighbors („kNN“) algorithm** is based on a 2-step procedure: (i) identify a number of k nearest neighbors in the original dataset, (ii) calculate mean (or median) of the neighbors‘ target value from the **feature(s) of our input**

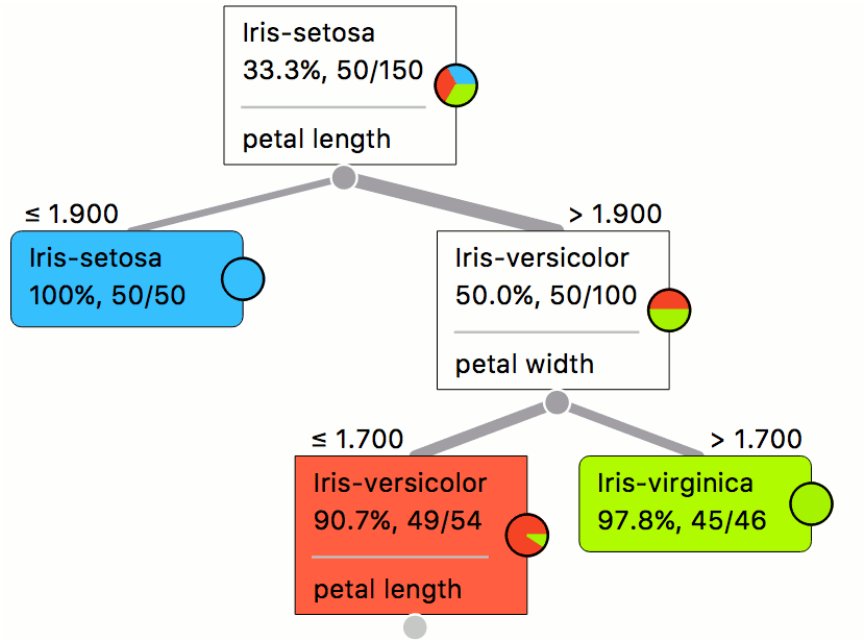
- The kNN algorithm is a **non-learning** („untrained“) AI, and essentially a **simple „interpolation“ algorithm.** It is **AI without explicit ML.**



Lombardo et al., *Chem. Rev., 122, 2022*

Naming Basics

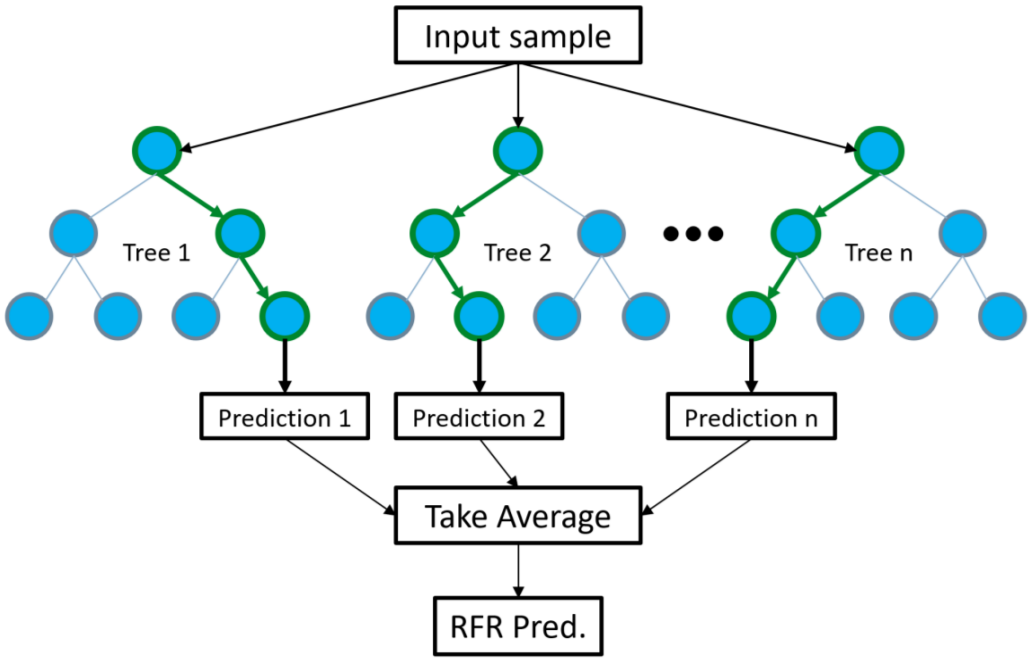
- „Trees“ and random „forests“ are a **sub-type of ML** algorithms.
- “(decision) **trees**“ are **if-then decision** structures.
- They are typically used for **classification** problems



Orange3 „iris“ example

Naming Basics

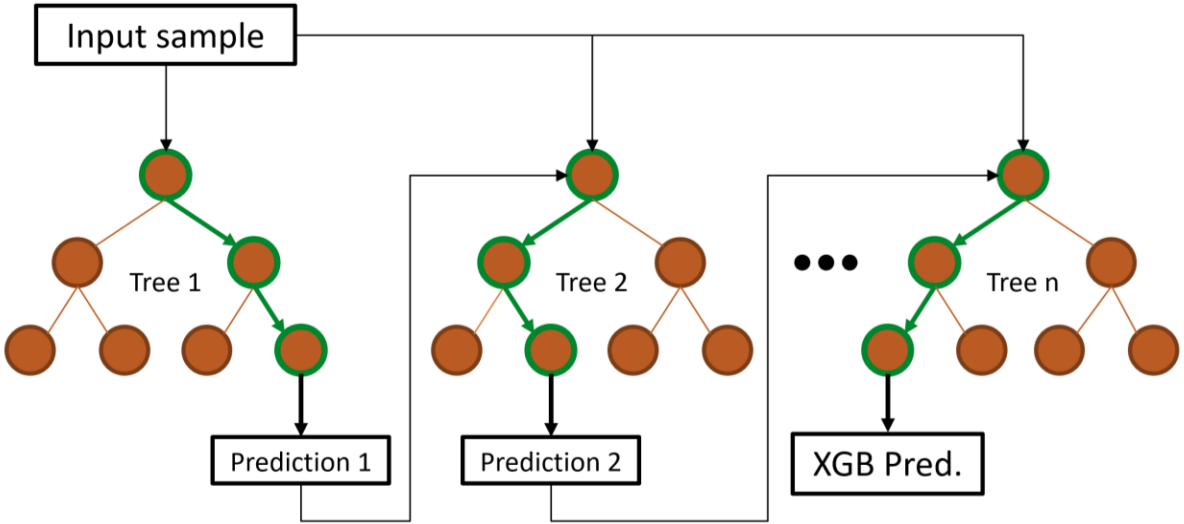
- a „(decision) **forest**“ is **an ensemble of trees**. This makes the prediction more stable and accurate. Also, it can be used as a **regressor** and not only as a classifier
- Since forests are often „randomized“ (i.e., random feature selection for individual trees), they are called „**random forests**“ (RFs), or **Random Forest Regressors** (RFRs)



Tausendschön, *PhD thesis, 2023*

Naming Basics

- **gradient boosting** is an approach similar to RFs
- However, decisions are **arranged sequentially** rather than in parallel
- Typically, **trees are** used as sub-units, but other decision units can be used as well

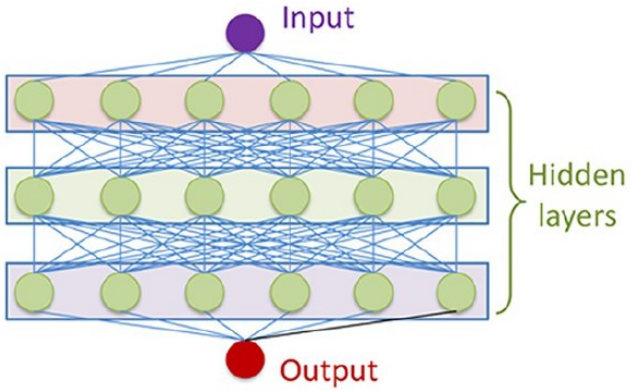


Tausendschön, *PhD thesis, 2023*

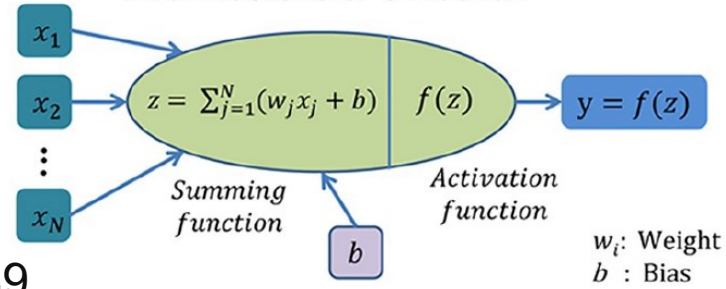
Naming Basics

- A Neural network (NN) is **a system of neurons** that is organized in multiple layers
- NNs are an important **sub-type of ML** algorithms
- A neuron is characterized by (i) a **weight-bias**-summation operation, and (ii) and an **activation function** (or just the **„activation“**). More complex neurons (or „**cells**“) are possible (e.g., LSTM, RNN)
- The set of **weights w_j and biases b** (offset) for each neuron, and the **NN structure** defines a NN

Artificial neural network



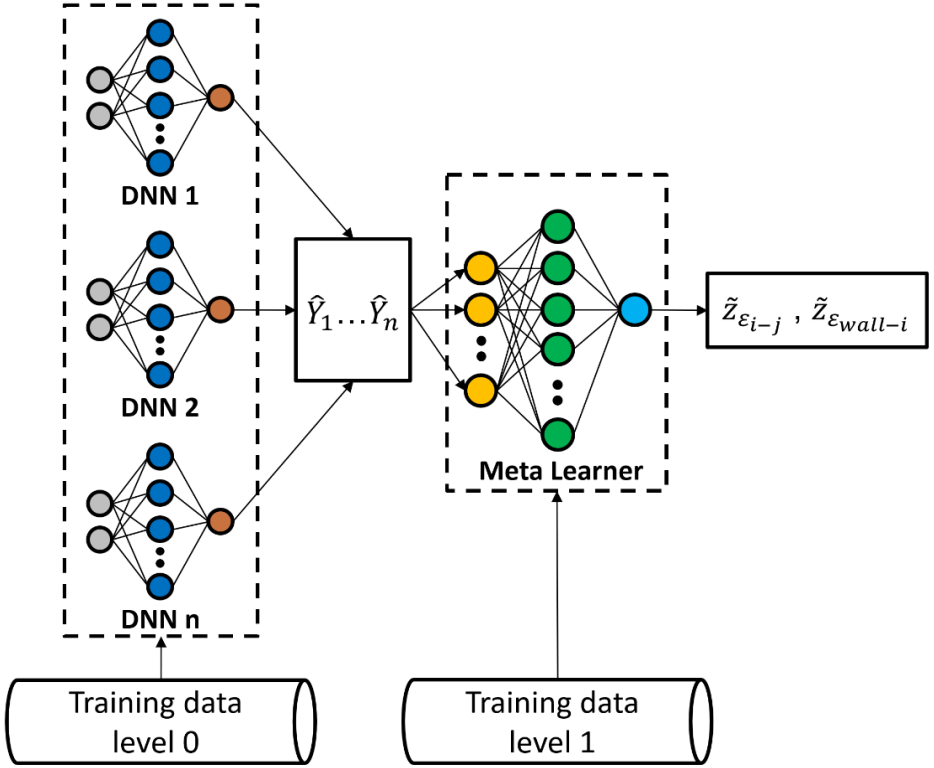
Architecture of a neuron



Zhu et al., *Ind. Eng. Chem. Res.* 2022, 61, 9901−9949

Naming Basics

- „ensemble (deep) NNs“ (**eDNNS**) can be used as well
- NNs are used as **level 0** models. These models are trained independently first.
- a **level 1 „meta learner“** is subsequently introduced. This is a separate NN and is **trained by a separate data set**. During „meta training“ level 0 models are not changed.



Tausendschön, *PhD thesis, 2023*

Naming Basics

- **Training** is the process of parameterizing an ML-algorithm. This is an **optimization** problem.

- Training is often the most **time-consuming part of the solution** (i.e., the computational work). Hence, the optimizer is often just called „**solver**“.

- „**hyperparameter search**“ is the process of searching (i.e., optimizing) the settings of an ML-algorithm. It is often „**an optimization of the optimizer**“ and of course of the ML-algorithm details (e.g., the tree depth). Hyperparameters are the „**architecture**“ of an ML model. The hyperparameter search uses a separate **validation data set.**

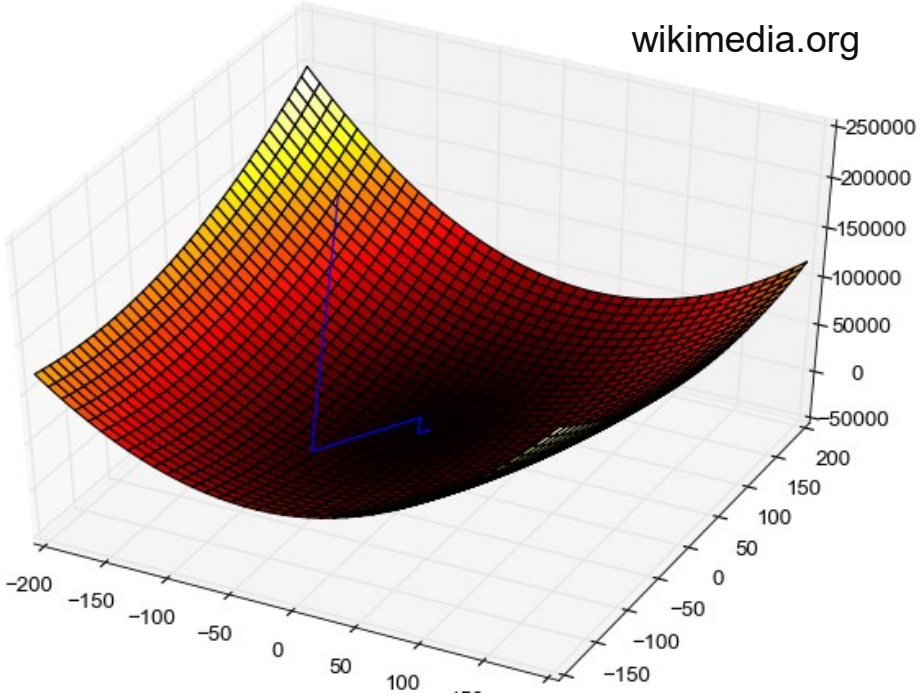
Neural Network Details - Optimizers

- The optimizer (solver) is the **central algorithmic element** in a NN model.
- It determines the **speed**, **convergence** behavior, **memory** requirement, and computational (resource) **efficiency** of the training process.
- It implements the **logic (i.e., the true „artificial intelligence“),** i.e., how the parameters of the NN are updated. It is the **„learner“ of the network**

Neural Network Details - Optimizers

- The **Gradient Descent (GD)** method works by initial randomization of the weights and biases.
- Then, the gradient of the **response surface** („**loss curve**“, „**loss**“ ***L***) is calculated, and the new parameter set is updated.
- How strong the gradient is weighted in the update is controlled with the „**learning rate**“ α parameter.

wikimedia.org

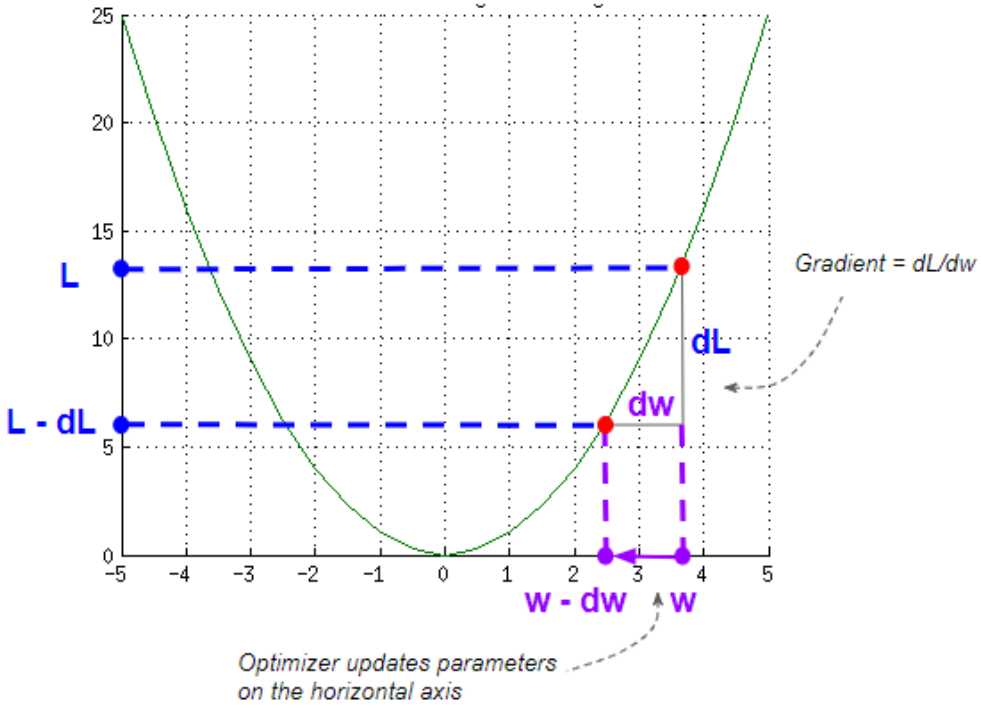


Neural Network Details - Optimizers

https://ketanhdoshi.github.io/Optimizer-Techniques/

$$w_j^{(t+1)} = w_j^{(t)} - \alpha \frac{\partial L}{\partial w_j}$$

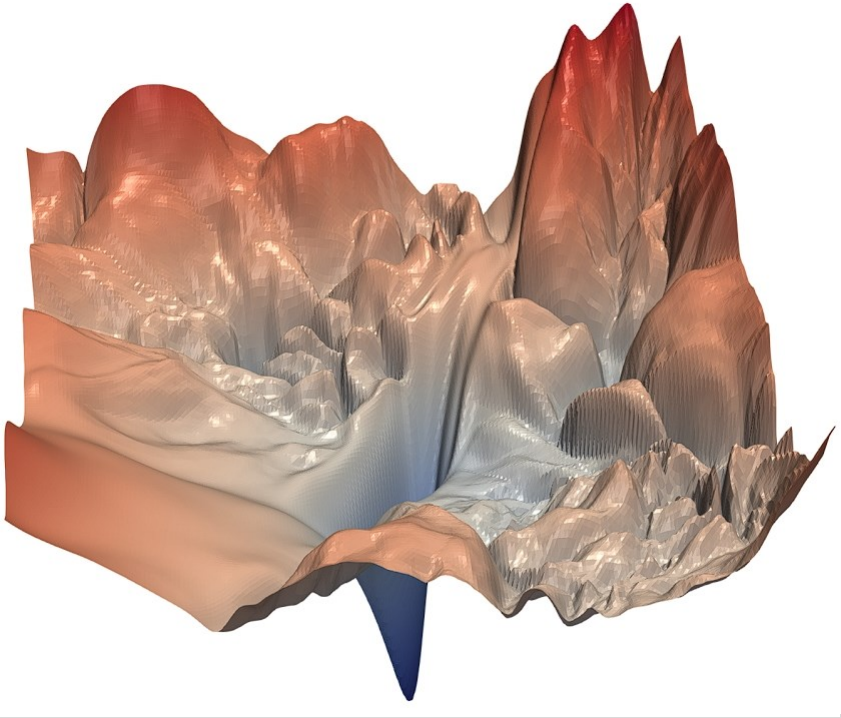
- this update process (for a single weight) can be visualized as shown on the right: the update to w_j is **proportional** to the **slope of L!**
- Note: this algorithm is different from the **Newton method** (or Newton-Raphson algorithm)



Neural Network Details - Optimizers

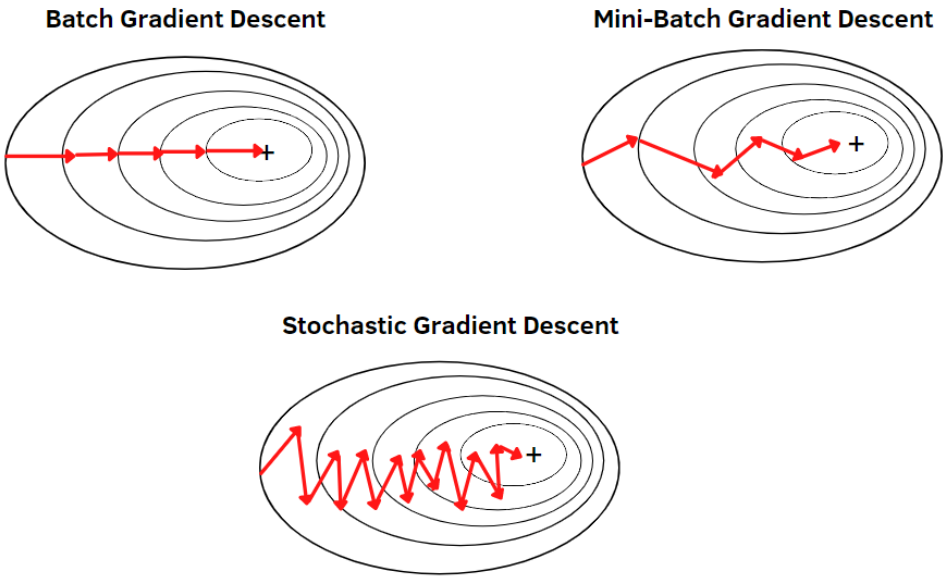
https://arxiv.org/pdf/1712.09913.pdf

- The practical challenge is that the loss function has a complex shape **with (possibly many) local minima.**
- Other challenges are **saddle points**, or narrow **ravines (valeys)** in the loss function topology.
- Thus, **naive GD methods** are **unsuitable** as training methods of complex NNs



Neural Network Details - Optimizers

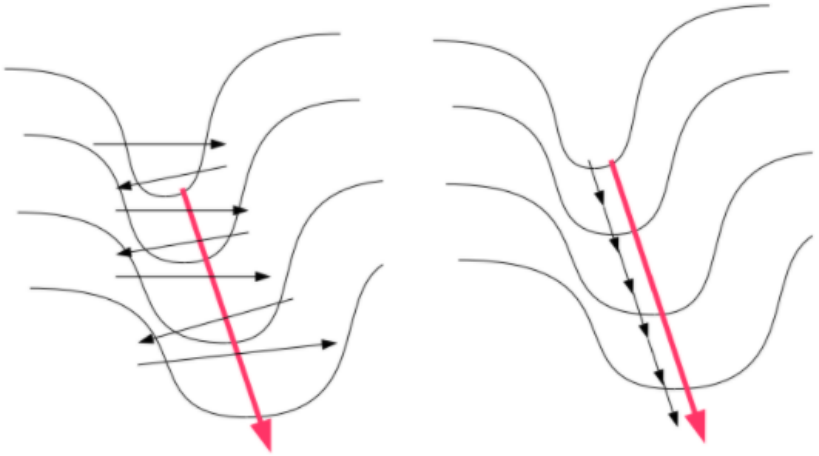
- **Stochastic** (and other **„mini-batch“**) **Gradient Descent (SGD)** methods use just a few randomly selected instances („**mini-batch**“) of the dataset to calculate the loss curve.
- This mini-batch of instances changes for each training step („**iteration**“), and hence the loss function. This allows to **„jump“ out** of local minima
- Note: an entire (training) **epoch** is completed once the **whole dataset is passed thru a NN**.



www.analyticsvidhya.com

Neural Network Details - Optimizers

- **AdaM** (**Ada**ptive **M**oment estimation) is a stochastic gradient-based optimizer. It was highly influential (#1 most cited paper of Kingma and Ba https://arxiv.org/abs/1412.6980. >**143k** citations since 2015).
- It is based on **parameter-individual learning rates** and uses the idea of „**momentum**“: it considers the history of past derivates of the loss function.
- AdaMax is a variant of AdaM (it uses a different norm for the weight udpate).



SGD bounces back and forth from one side of the valley to the other

Using Momentum the zig-zag cancels out, while the direction along the valley is reinforced

https://icml.cc/Conferences/2010/papers/458.pdf

Neural Network Details - Optimizers

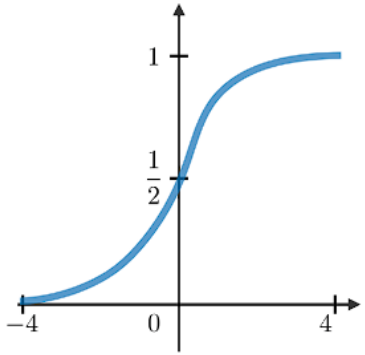
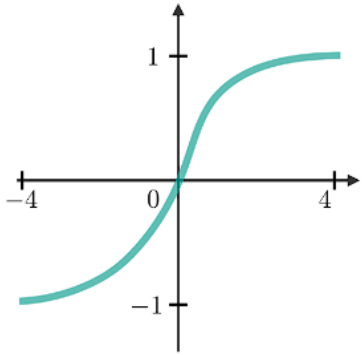
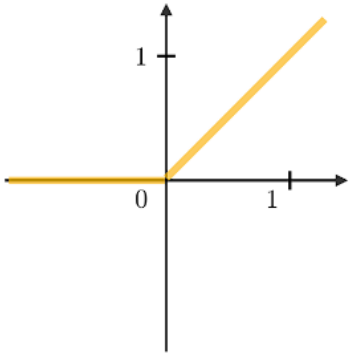
- **RMSProp** (**r**oot **m**ean **s**quare **prop**agation) and **AdaGrad** (**Ada**ptive **grad**ient) are predecessors of AdaM. They belong to the „**SGD advanced** family“ of algorithms.

- „**L-BFGS-B**“ is an advanced algorithm and stands for **Limited-memory B**royden-**F**letcher-**G**oldfarb-**S**hanno with **Bound constraints**. It is a **quasi-Newton method**. It is memory sparse (i.e., efficient) and allows bounding of weights and biases. The latter reduces the **overfitting tendency**.

Neural Network Details – Activation Functions

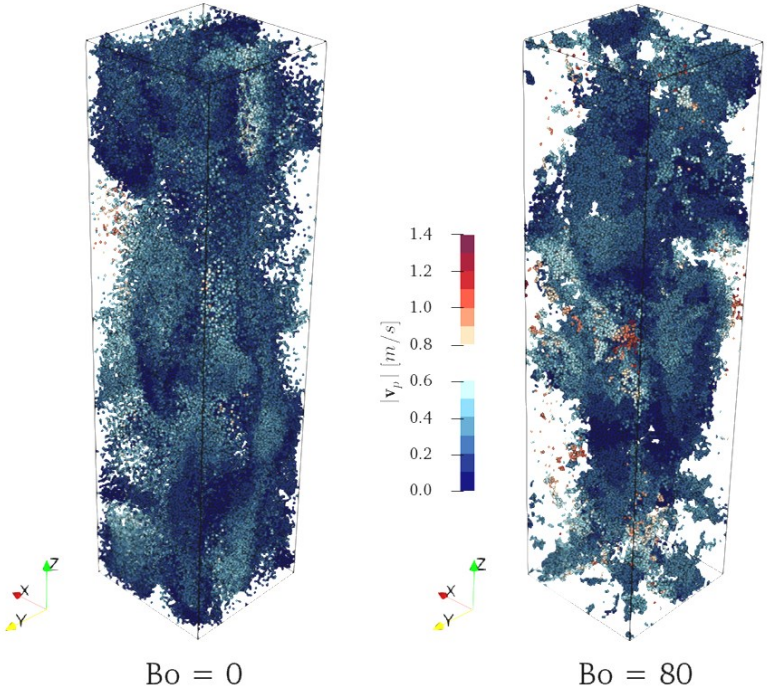
- ...should be simple to compute, **but nonlinear** (most problems cannot be modeled well with linear relationships)
- **ReLU** is the standard. „Leaky ReLU“ improves performance by adding a shallow slope for z<0.
- **Use others** only if you need **zero-centered functions**

also „logistic“

Sigmoid	Tanh	ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

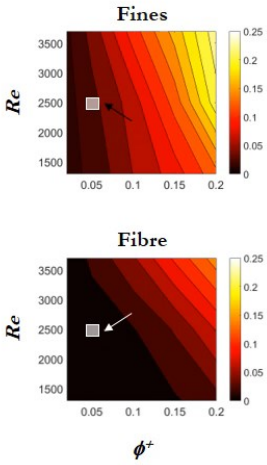
https://stanford.edu/~shervine/teaching/

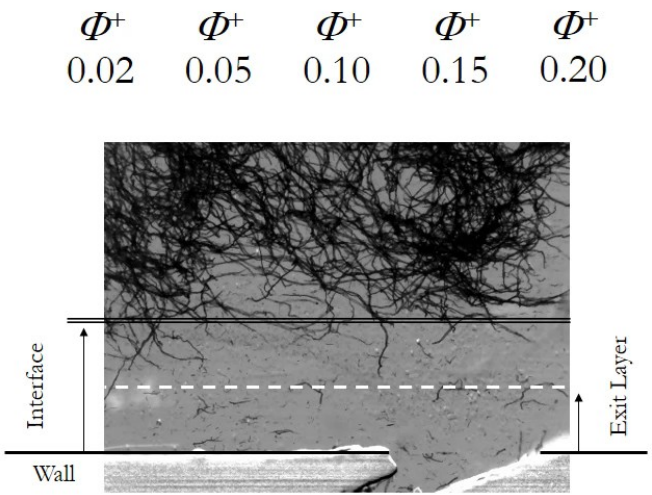
Granular & Fluid-Particle Flow Systems



Tausendschön, *PhD thesis, 2023*

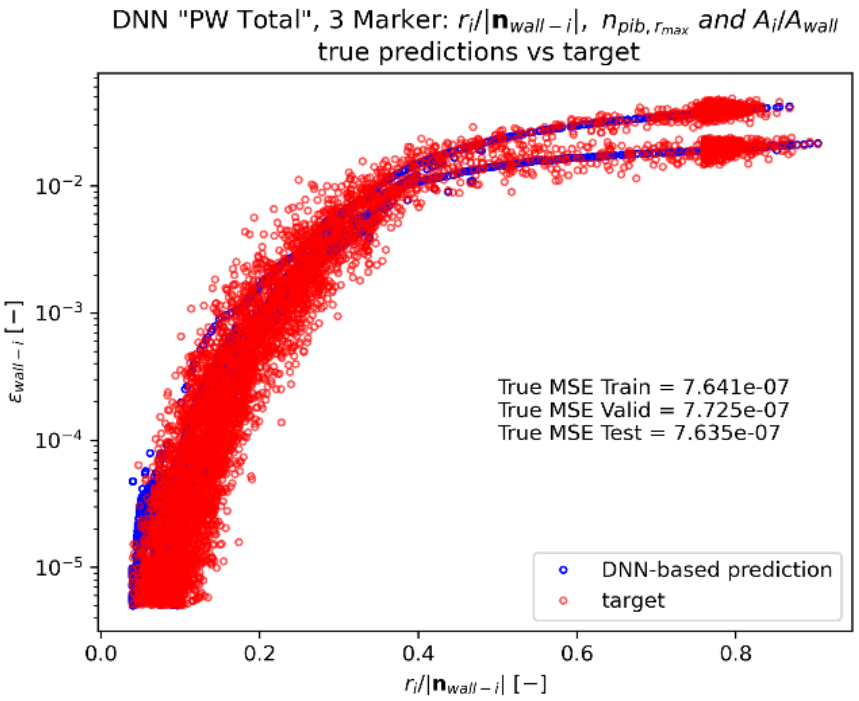
Fibre Suspension Flow





Why and When?

- Equations describing **phenomena not available or unclosed**
- A numerical solution to describe the phenomen at hand is **time consuming**
- **Non-trivial cause-effect relationships** (e.g., hysteresis, bifurcation) exist that rule out a simple correlation
- **Multiple features** influence the target
- You need a **large number of correlations** that are time-consuming to develop



Tausendschön, *PhD thesis, 2023*

Why and When?

3 main application areas for flow & process topics

(1) closure construction (e.g., continuum stress models, drag models, force/stress corrections, etc.)

(2) directly simulate motion („learn **particle movement**“, or at least correct from a simple **motion rule**) or overall process performance

(3) accelerate a fundamental **solution algorithm** (e.g., linear solver used in CFD)

What cannot be done (fully) by ML?

- **meaningful (re)scaling** of features and targets
- **feature selection** and benchmarking (e.g., hysteresis, bifurcation)
- Generation and curation of **data**
- **Splitting** of training, validation, and test data
- (data) **workflow** management
- …and of course everything **you do with the ML-empowered model** (shape optimization, etc.)

- **identification** of a meaningful (dimensionless) target (unless you want to to unsupervised learning…)

What **should not** be done with ML?

- **Replace a few simple correlations** that only require a few (1…2) markers
- Attempt to predict **dimensional quantities**, or absolute quantities
- Directly use **uncurated/unreviewed raw data** for training and benchmarking
- Skip the **validation or testing** step (i.e., a "**biased evaluation**" = evaluation with training data)

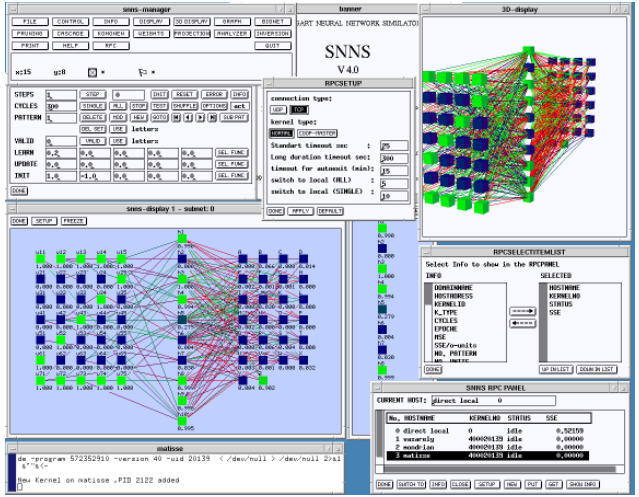
Note that…

validation = evaluation of model performance during hyperparameter search

testing = evaluation of final model performance

Some Relevant Studies: NN history

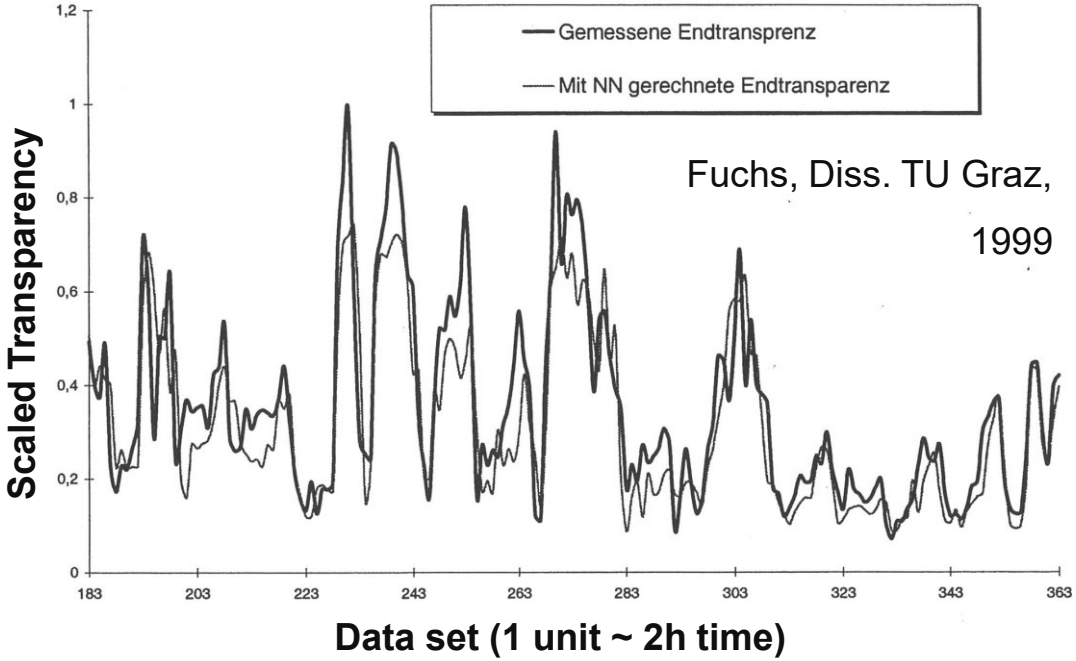
- Early **1990's**
- www.ra.cs.uni-tuebingen.de/SNNS
- **No NN-specific optimization algorithms**
  (simulated annealing, ConjGrad, Monte Carlo…)



Fuchs, Diss. TU Graz, 1999

Some Relevant Studies: NN history

- Fuchs uses SNNS to predict **sedimentation performance** of sludge based on training with **182 (!) data points**
- 8 features (input):
  - pH, **COD**, conductivity
  - **$temp_{feed}$, $temp_{amb}$**
  - **Flow rate**, **O_2 content in aerator**
  - "sludge management"



Some Relevant Studies: accelerated DEM

- **Overall Goal: Speed up DEM simulations**
- Idea: do a **naive evolution** of position and velocity ("intermediate step"), and then use a **NN to correct for collision events** (base idea of Ummenhofer et al.)
- Target: **position correction Δx** after intermediate step
- Convolutional NN with many **features: velocity of neighbors.**
- **"multi-scale"** loss function: micro (position error) and macro (center of mass error)

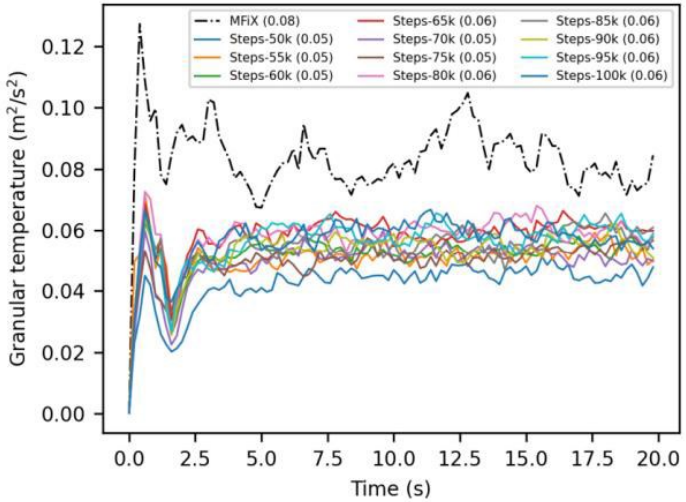
$$L^{n+1} = \alpha \frac{1}{N} \sum_{i=1}^{N} \left\| x_i^{n+1} - \hat{x}_i^{n+1} \right\|_2 + (1-\alpha) \left\| \frac{1}{N} \sum_{i=1}^{N} x_i^{n+1} - \frac{1}{N} \sum_{i=1}^{N} \hat{x}_i^{n+1} \right\|_2$$

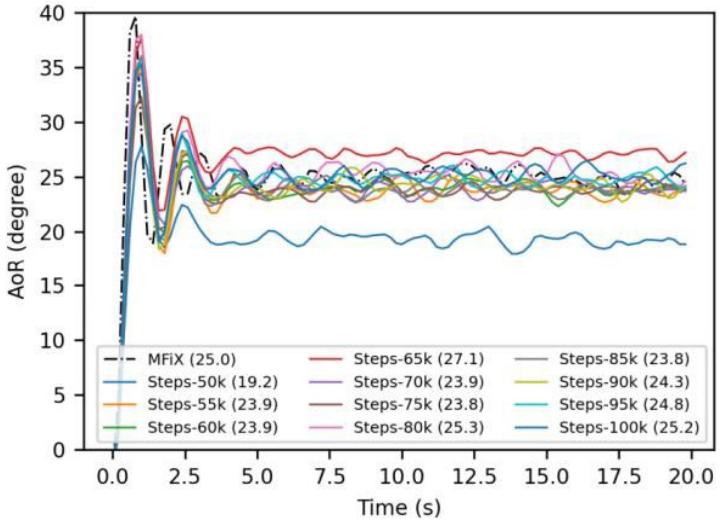
Lu et al., *Chem Eng Sci., 245, 2021.*

Ummenhofer et al., ICLR 2020.

Some Relevant Studies: accelerated DEM

- Performance for macro-quantities **(angle of repose) satisfactory after ~70k frames**





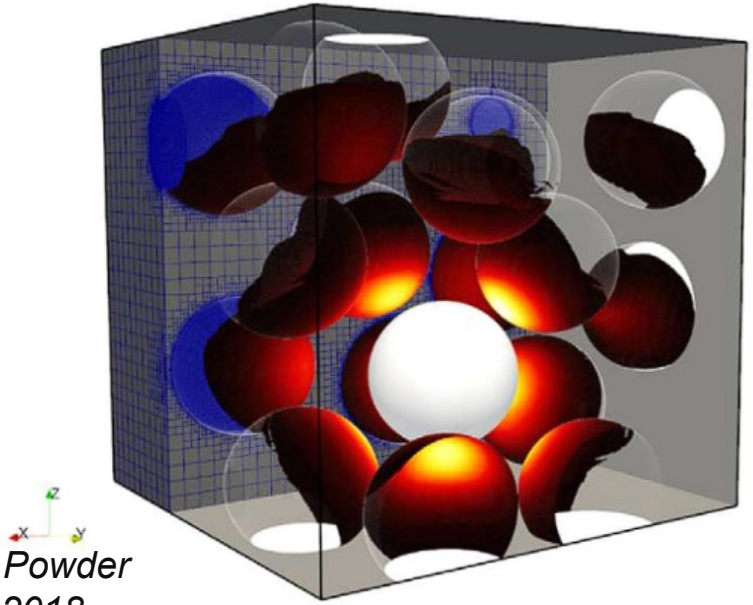
...however, **persistent underprediction of T_g**

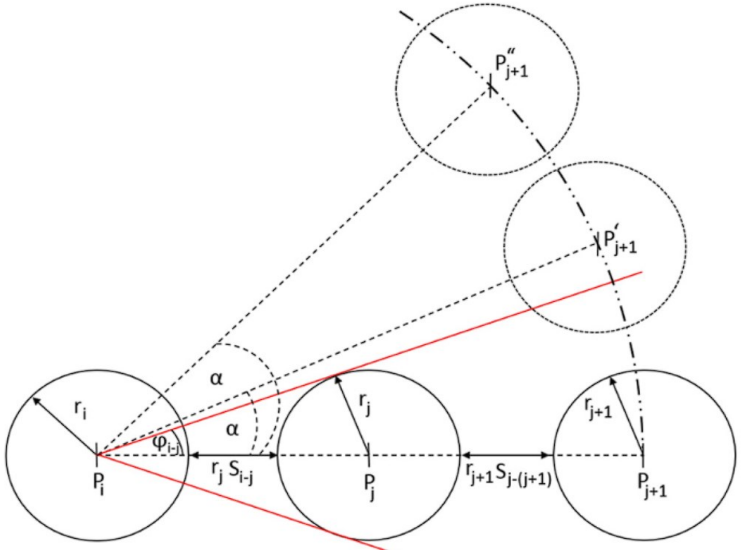
Overall: ~**80x speedup @ training for each setup with ~100k frames. Huge potential** for improvement...

Lu et al., *Chem Eng Sci., 245, 2021.*

Some Relevant Studies: closures for view factors

- **Earlier study:** estimate of view factors between particles and particle-walls considering shadowing using simplified „point ray“ model

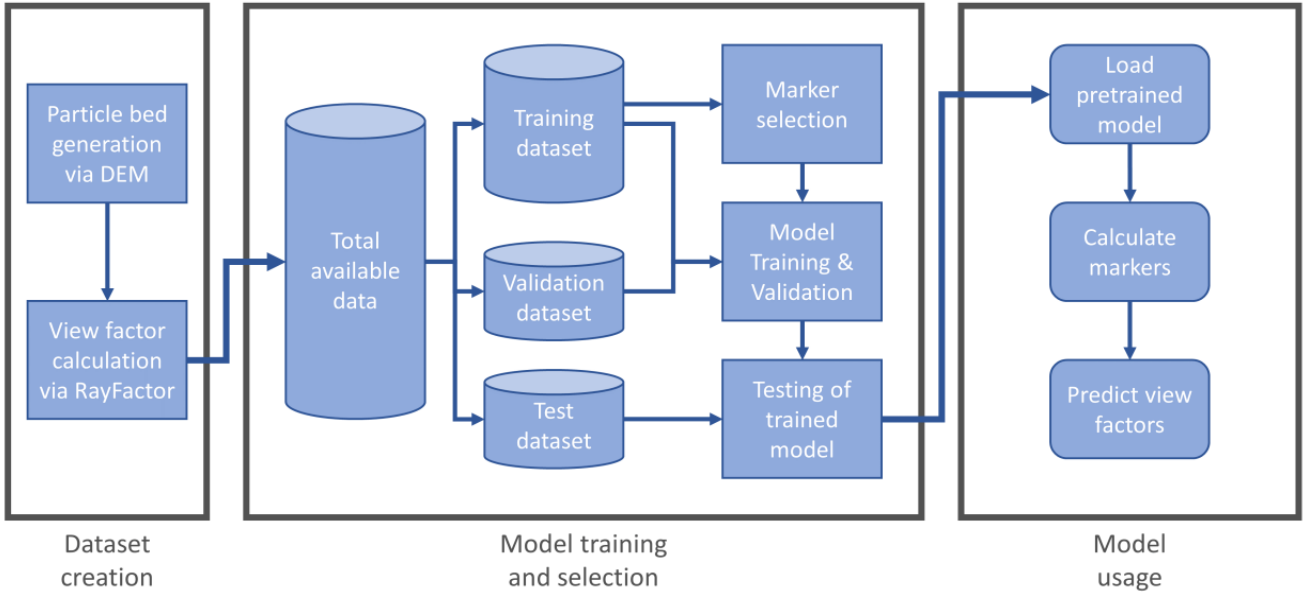




Forgber & Radl, *Powder Technol, 323, 2018*

Some Relevant Studies: closures for view factors

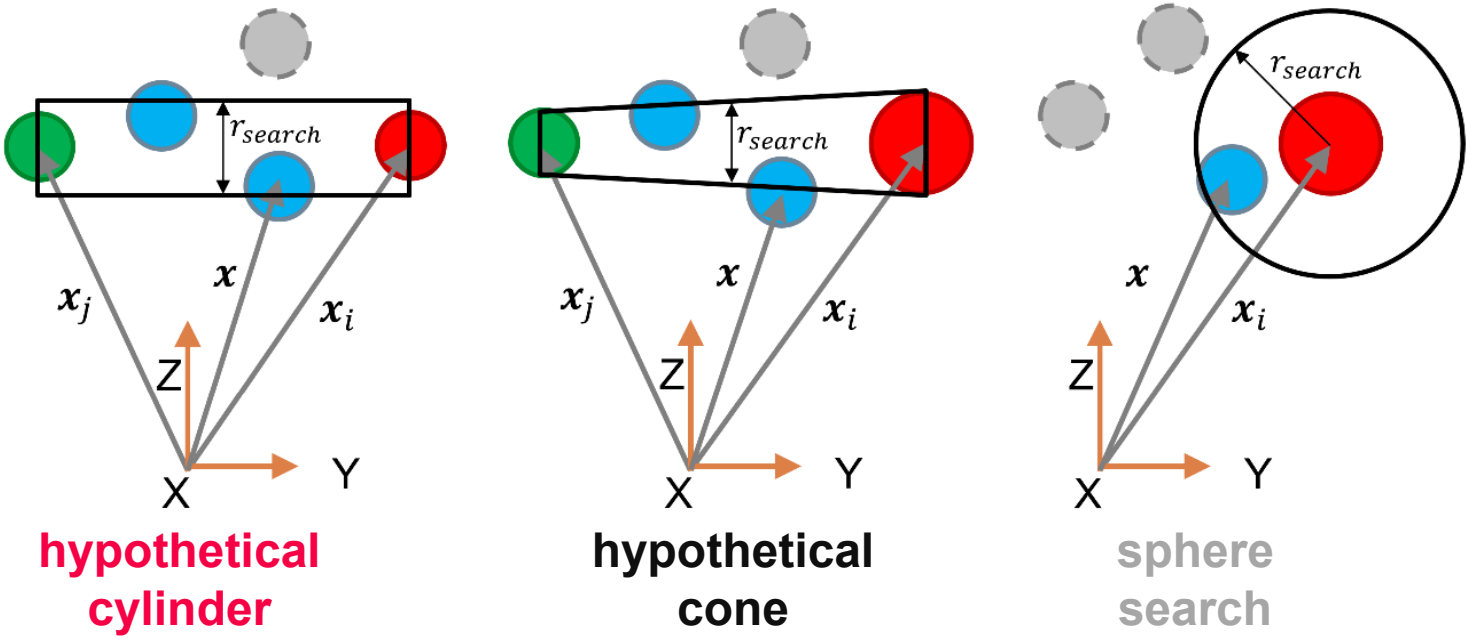
- **New Goal: faster** and accurate estimate of view factors between particles and particle-walls



Tausendschön, *PhD thesis, 2023*

Some Relevant Studies: closures for view factors

- **Feature („Marker“) analysis**: number of particles as simple **shadowing indicator**



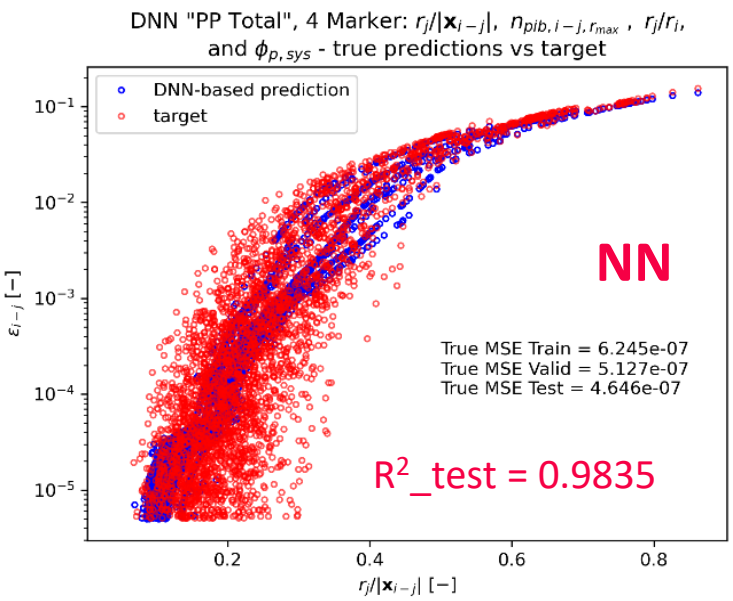
hypothetical cylinder | **hypothetical cone** | **sphere search**

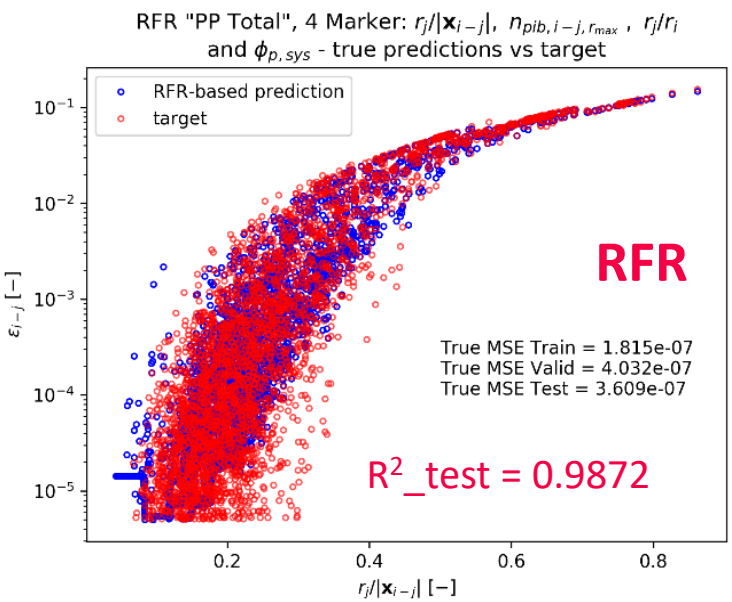
Tausendschön, *PhD thesis, 2023*

Some Relevant Studies: closures for view factors

- **Results for 4 markers, particle-particle, bi- & polydisperse**

$$\psi_{PP,4M} = \left(\frac{r_j}{|\boldsymbol{x}_{i-j}|}, n_{pib,i-j,r_{max}}, \frac{r_j}{r_i}, \phi_{p,sys} \right)$$



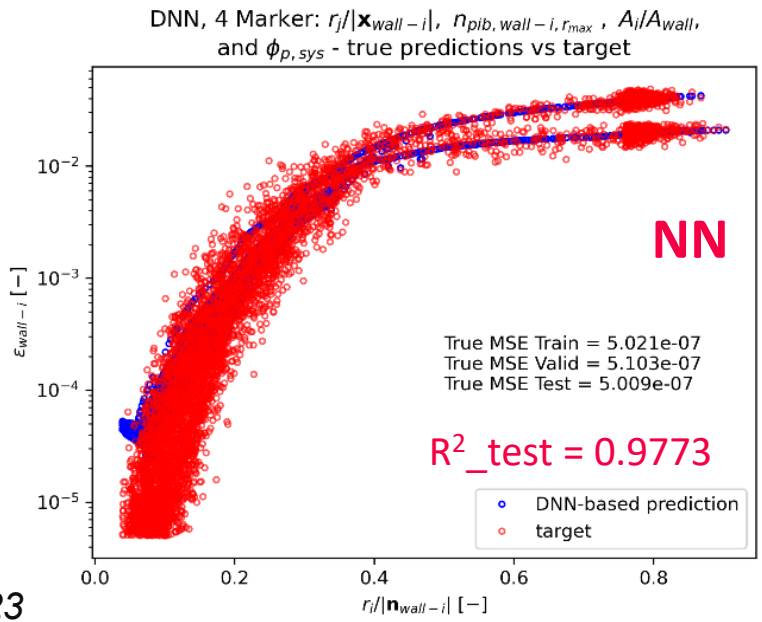


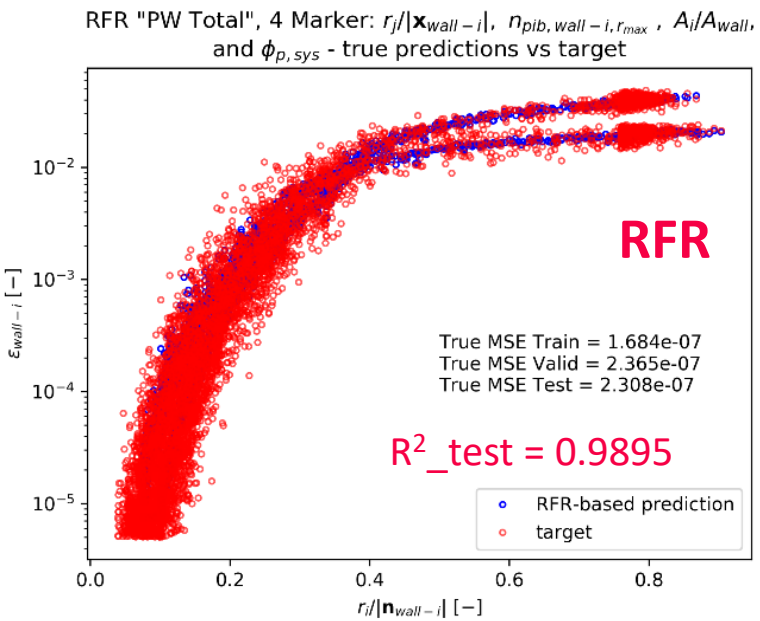
Tausendschön,
PhD thesis, 2023

Some Relevant Studies: closures for view factors

- **Results for 4 markers, particle-wall, bi- & polydisperse**

$$\psi_{PW,4M} = \left(\frac{r_i}{|\mathbf{n}_{wall-i}|}, \frac{A_i}{A_{wall}}, n_{pib,wall-i,r_{max}}, \phi_{p,sys}\right)$$





Tausendschön,
PhD thesis, 2023

AI applied to Granular Systems

Stefan Radl

> April 26, 2023, TUSAIL School “EVENT 4”