01 Introduction to Deep Learning

Attila Cangi

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

01.1 Background and Theory

- Overview of Neural Networks
- Perceptron Model
- Binary Classification with Perceptrons

01.2 Interactive Session

- Neural Networks (Multi-Layer Perceptrons)
- Learning Colors

01.3 Research Application *(in equilibrium)*

- Scalable Neural Networks for Predicting the Electronic Structure of Matter

IMPRS for Quantum Dynamics and Control

Summer School 2024: Machine Learning and Many Body Systems in or out of Equilibrium

Wroclaw, Poland

July 30, 2024

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

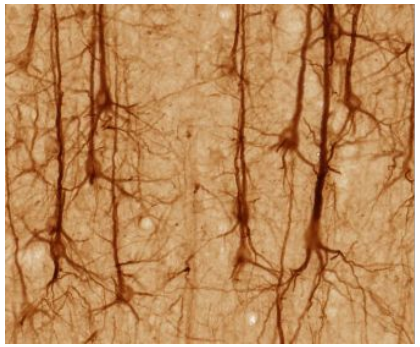
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

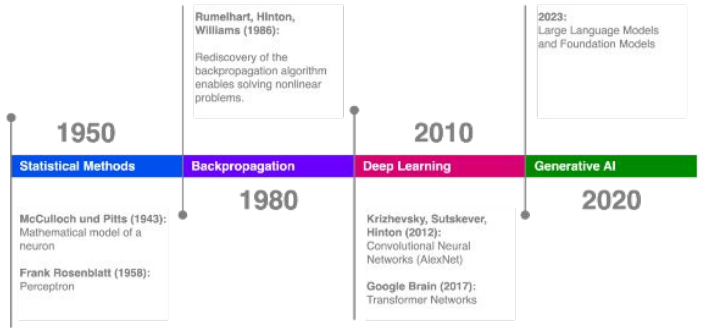
01 Introduction to Deep Learning

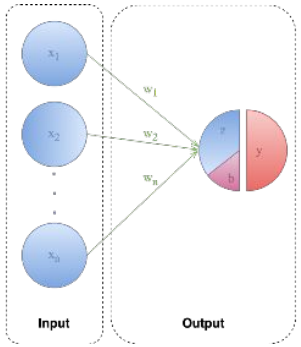
01.1 Background and Theory

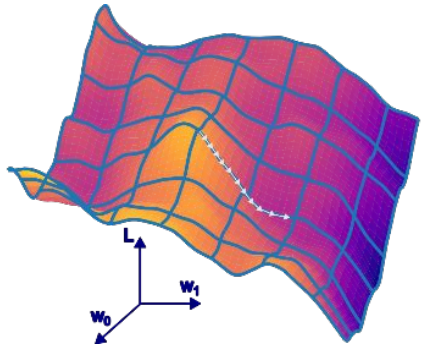
CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

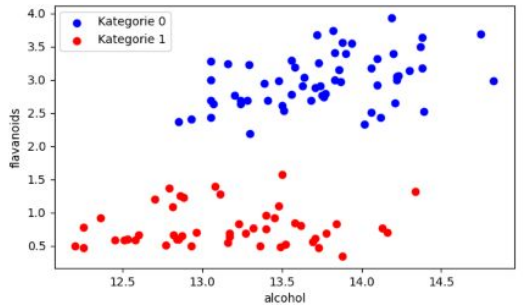
www.casus.science











IMPRS for Quantum Dynamics and Control

Summer School 2024: Machine Learning and Many Body Systems in or out of Equilibrium

Wroclaw, Poland

July 30, 2024

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

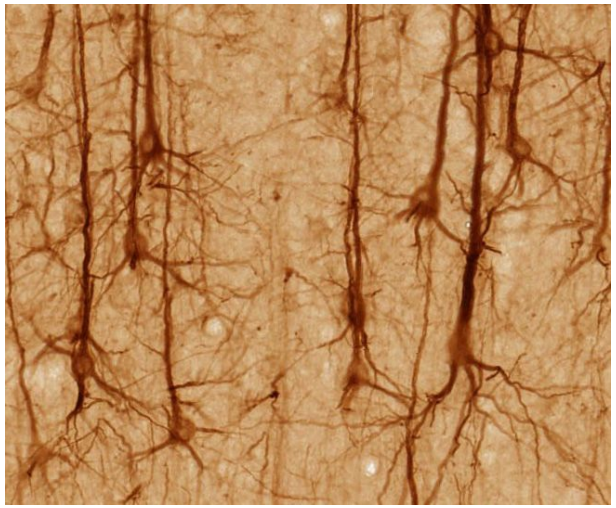
Neural Networks

Biological Neurons

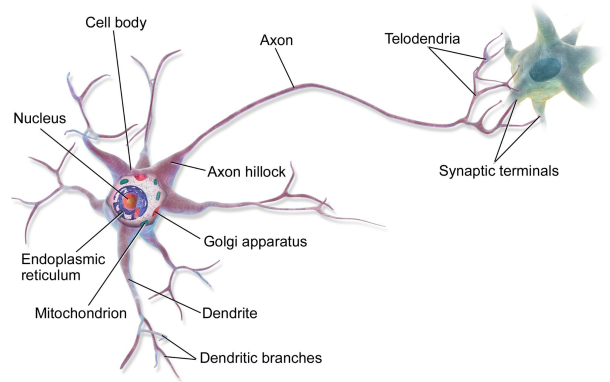
A neuron, also known as a nerve cell, is the basic functional unit of the nervous system. Neurons are specialized cells that receive, process and transmit information through electrical and chemical signals.

A neuron is made up of three main parts:

- **Cell body** (soma)
  Contains the nucleus and other organelles necessary for basic cell functions.
- **Dendrites**
  Tree-like projections that receive signals from other nerve cells and transmit them to the cell body.
- **Axon**
  A long, thin extension that conducts electrical signals away from the cell body to other nerve cells, muscles or glands. The end of the axon branches into axon terminals that release neurotransmitters to transmit signals.



Neurons in the cerebral cortex

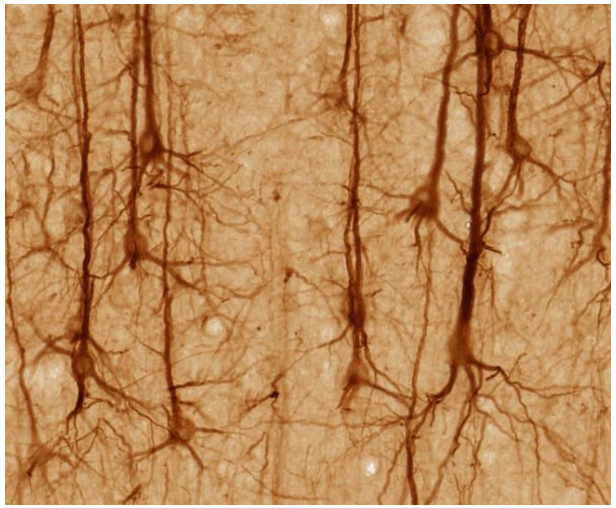


Neural Networks

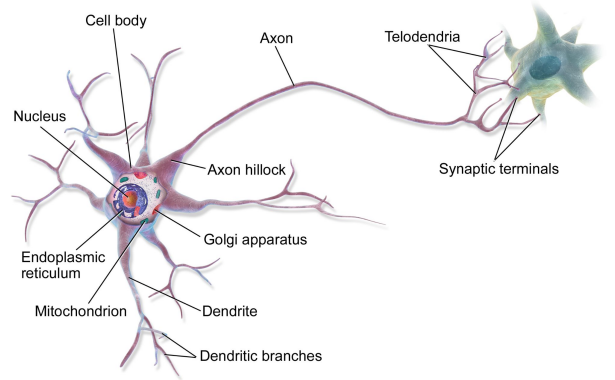
Biological Neurons

The function of a neuron can be divided into several steps:

- **Reception of signals**
  Dendrites receive chemical signals from neighboring neurons. These signals lead to changes in the membrane potential of the neuron.
- **Generation of an action potential**
  When the membrane potential reaches a certain threshold, an action potential is triggered. This is a rapid change in the membrane potential that spreads along the axon.
- **Transmission of the action potential**
  The action potential travels along the axon to the axon terminals. This occurs through a series of depolarizations and repolarizations of the cell membrane.



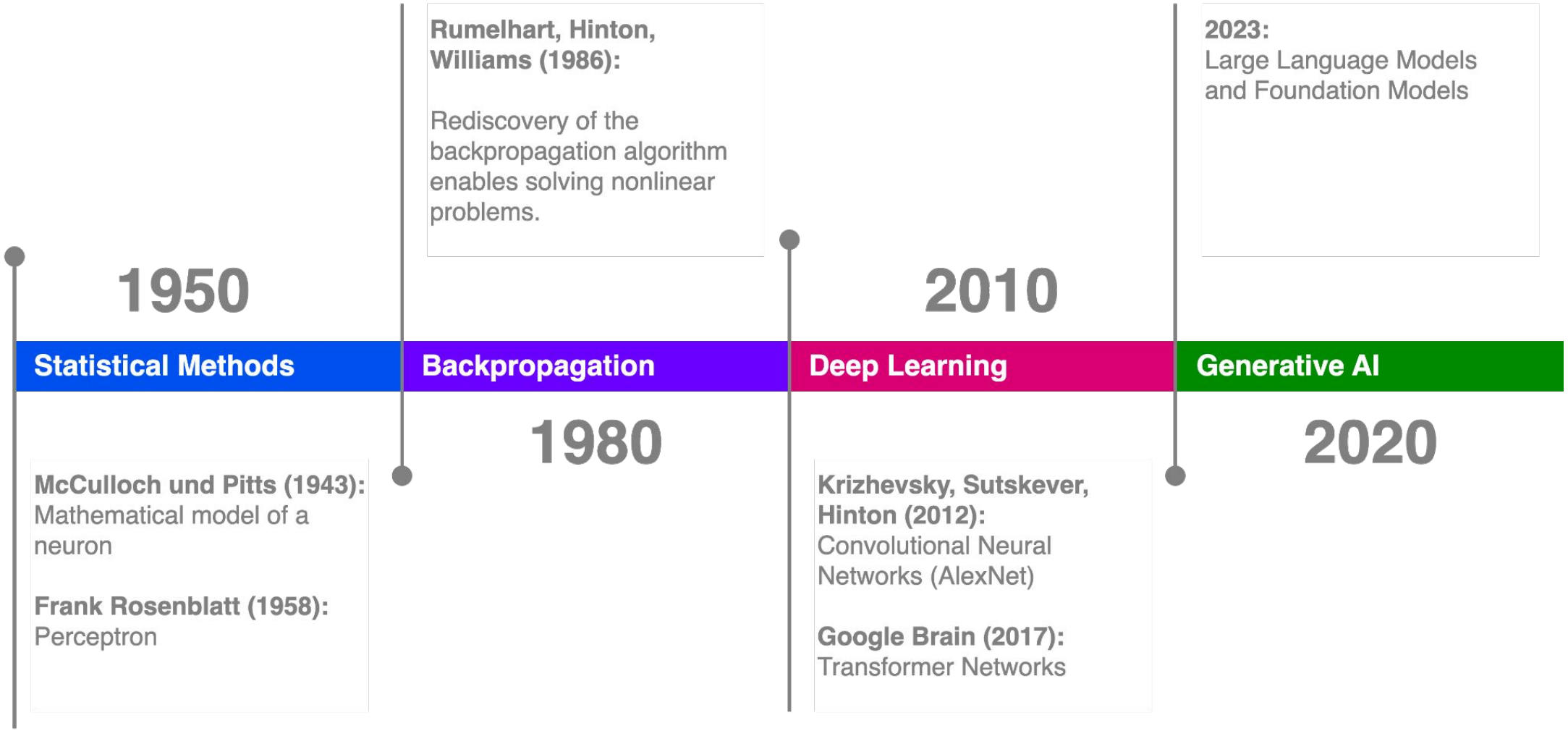
Neurons in the cerebral cortex



This variability of information transmission is reflected in the weights of artificial neural networks.

Neural Networks

Short History of Artificial Neural Networks



Perceptron

Structure

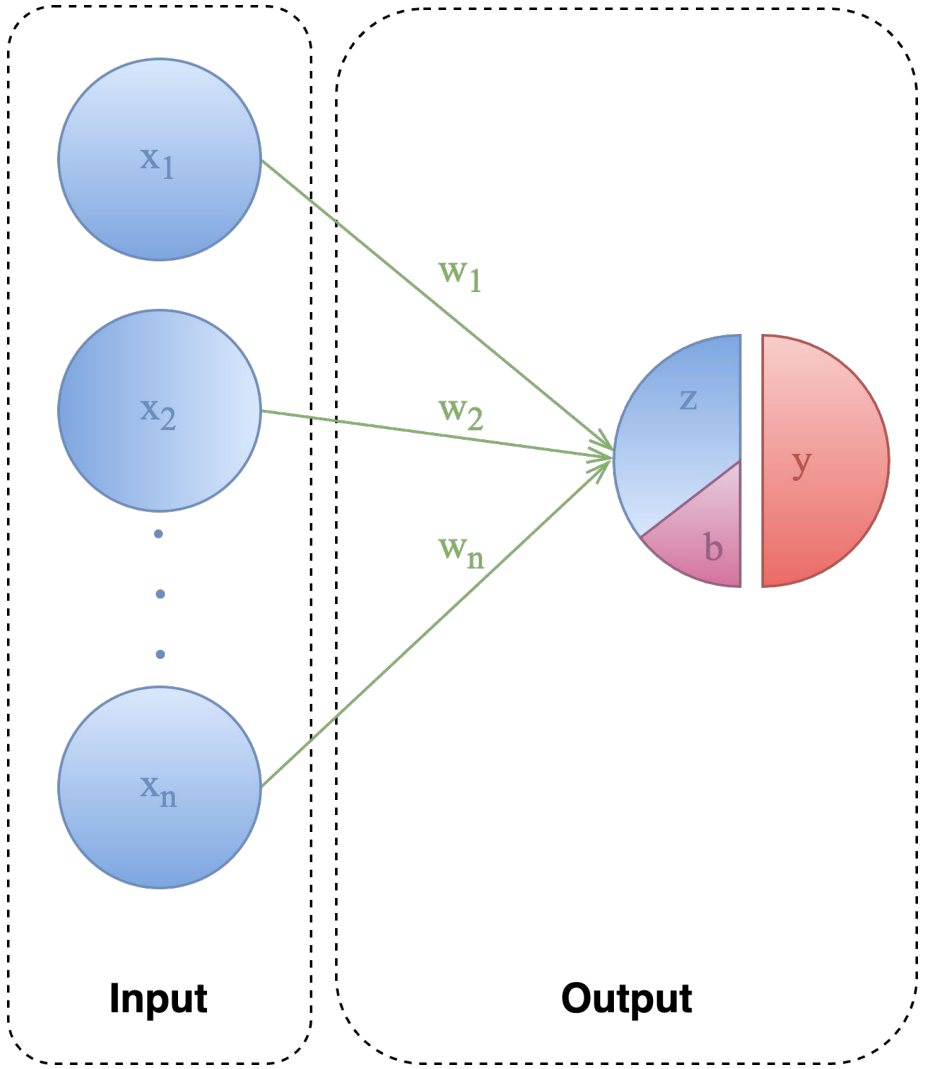
The simple perceptron model consists of an input layer and an output layer.

In the input layer, the input is $(x_1, x_2, \ldots, x_n)$.

The output layer consists of a single neuron. It contains the network input (z) and the output value (y).

This network can be used for binary classification, i.e. the network can decide for an input whether it belongs to a certain category.

The classification is expressed by the output value (y).



Perceptron

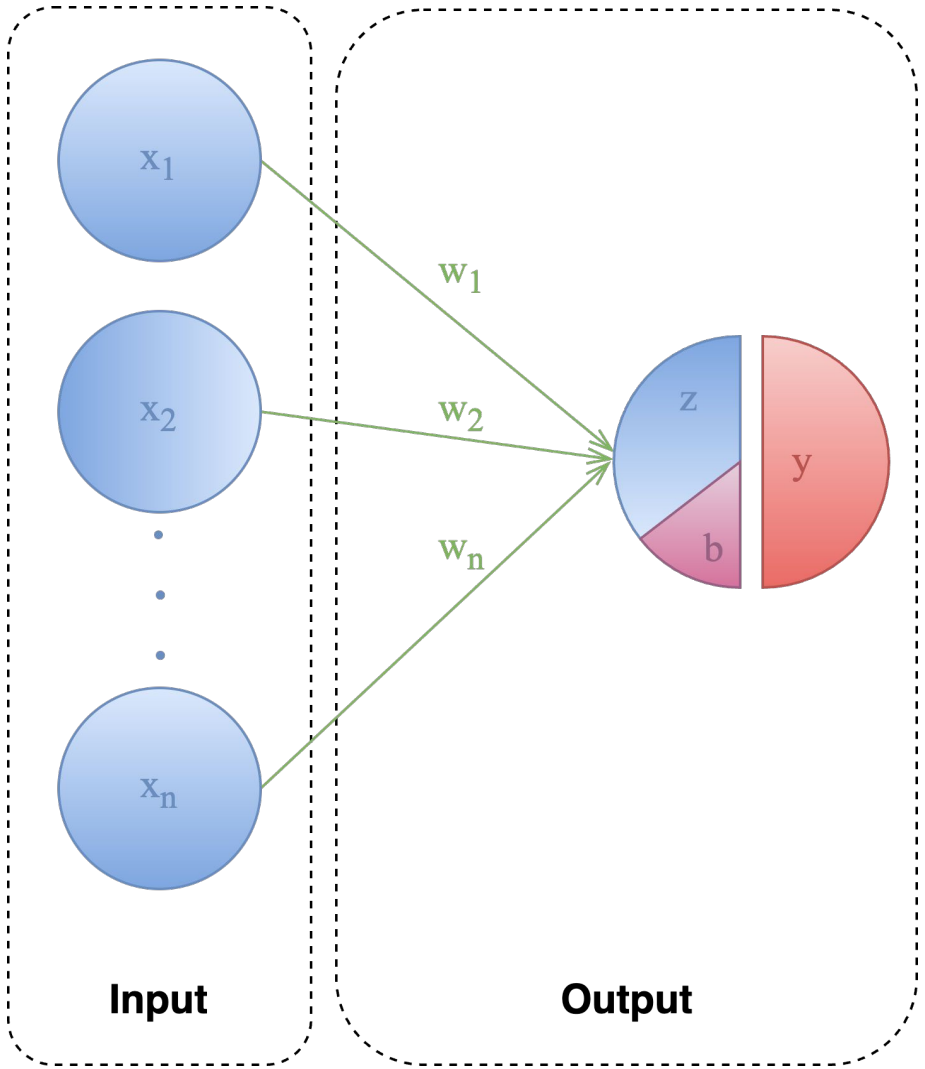
Forward Propagation

Input

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad n \in \mathbb{R}, \quad x_i \in \mathbb{R}$$

Weights

$$\vec{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}, \quad w_i \in \mathbb{R}$$

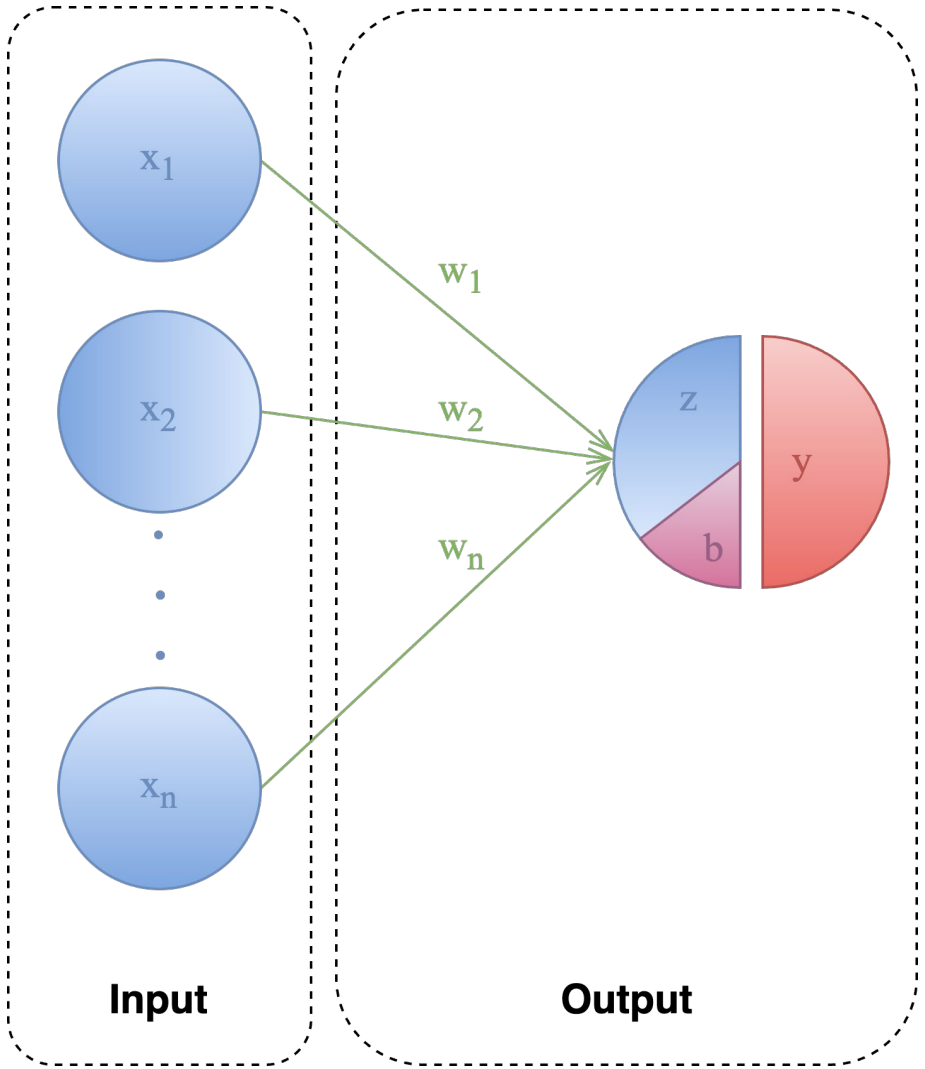


Perceptron

Forward Propagation

Net input

$$
\begin{aligned}
z &= w_1 x_1 + \cdots + w_n x_n \\
&= \vec{w}^T \cdot \vec{x} \\
&= (w_1, \ldots, w_n) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}
\end{aligned}
$$



Perceptron

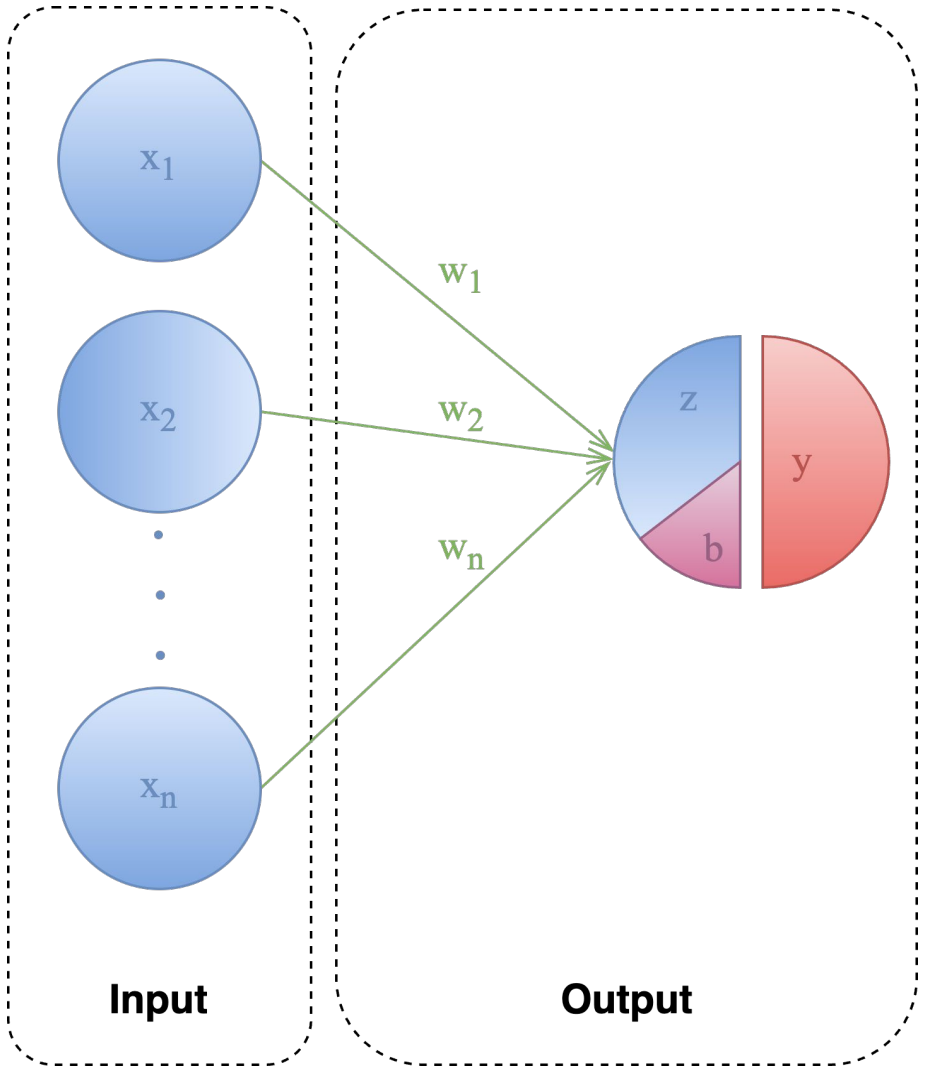
Forward Propagation

Activation and output

In the second and final step, we calculate the activation of the output neuron, which also corresponds to the output of the perceptron model.

An activation function is applied to the network input:

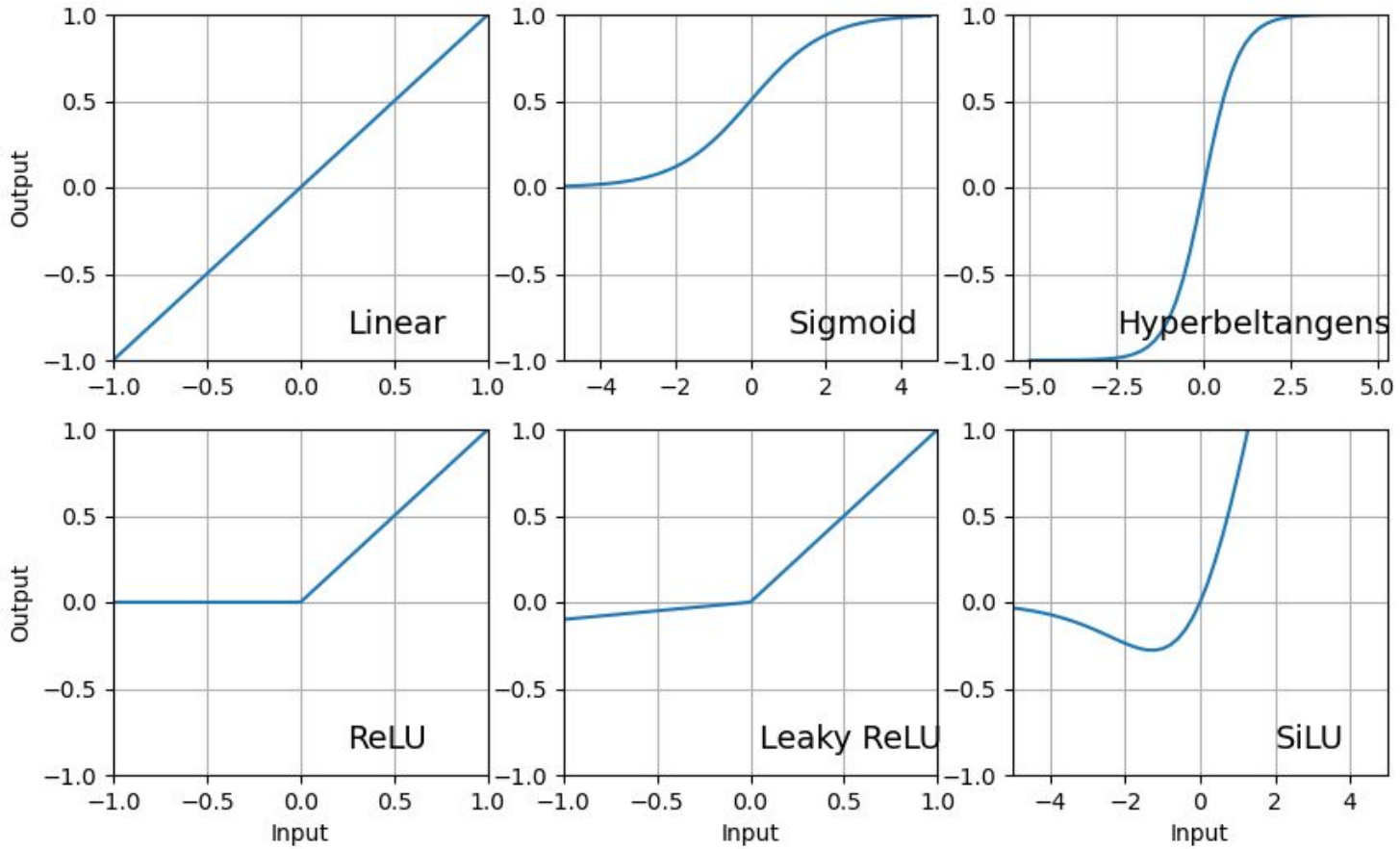
$$y = g(z + b)$$



Perceptron

Forward Propagation

Examples of Activation Functions



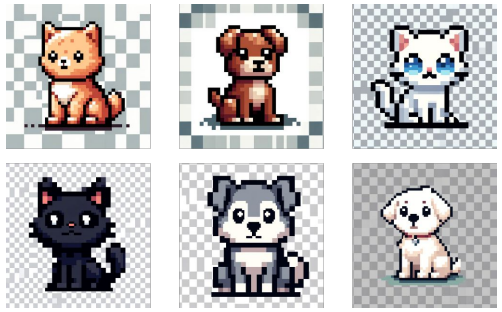
Perceptron

Learning Process

In the context of the perceptron model, we understand learning as the **gradual adaptation of the weights to the desired target function** with the help of training data.

Labeled Data

Training data is, for example, a series of data with a label.



Data labeled as cat and non-cat.

The training data can therefore be written as pairs of feature vectors and labels:

$$(\vec{x}^k, y^k) \quad k \in \{1, \ldots, N\}$$

With a neural network, we must distinguish between the calculated output of the current network and the correct output of a training example.

Perceptron

Learning Process

Learning Step

Learning means that we calculate the output for each training example and then adjust the weights. This is called a **learning step**.

We can therefore adjust the weight vector for the pairs of feature vectors and labels in each learning step by adding a change of all weights to the current value:

$$w_i := w_i + \Delta w_i$$

We will define a concrete rule for adjusting the weights in a moment. First let's look at the properties of the weight update:

- If the calculated output is greater than the reference value, the weight update should be negative, i.e. the weight of this neuron should be weakened.

- If the calculated output is smaller than the reference value, the weight update should be positive, i.e. the weight is increased.

- If the calculated output and the reference value are the same, no weighting update should take place.

Perceptron

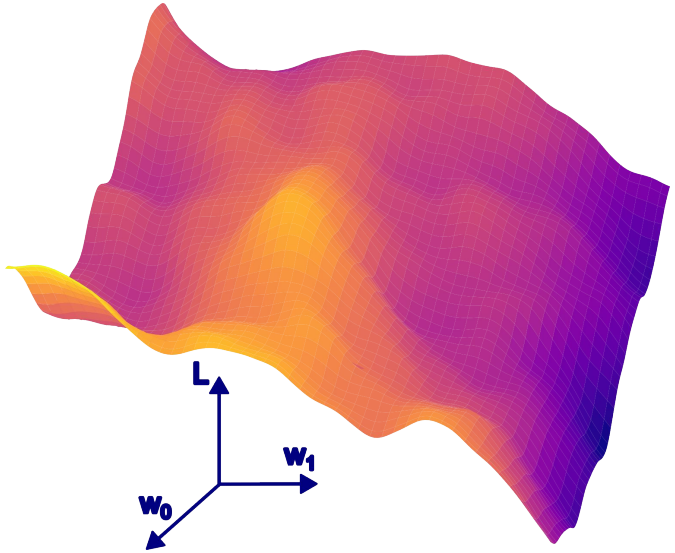
Gradient Descent

Loss Function

In order to derive a suitable learning rule, we must first define the term loss function:

$$L(w) = \frac{1}{2N} \sum_{k=1}^{N} \left[y^k - \bar{y}^k \right]^2$$

It is a multidimensional function of the weights or the weight vector

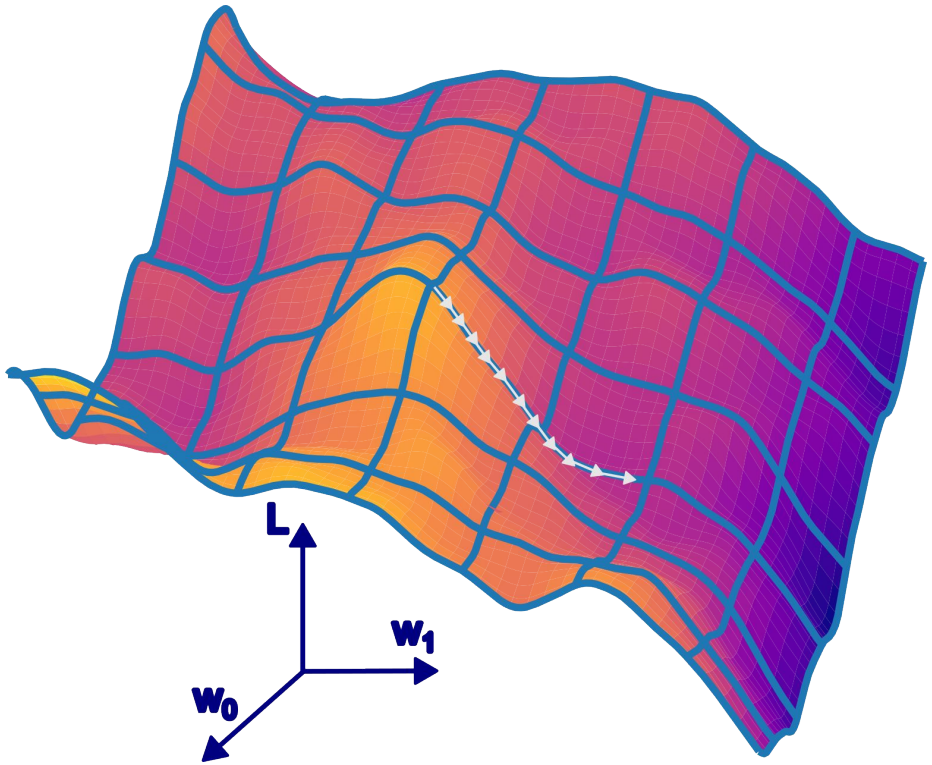


Perceptron

Gradient Descent

Learning Rule: Minimum of the Loss Landscape

We can now use the loss function to derive a learning rule by setting ourselves the goal of minimizing the value of the loss function. This means that we look for valleys in the parameter landscape.



Perceptron

Weight Update

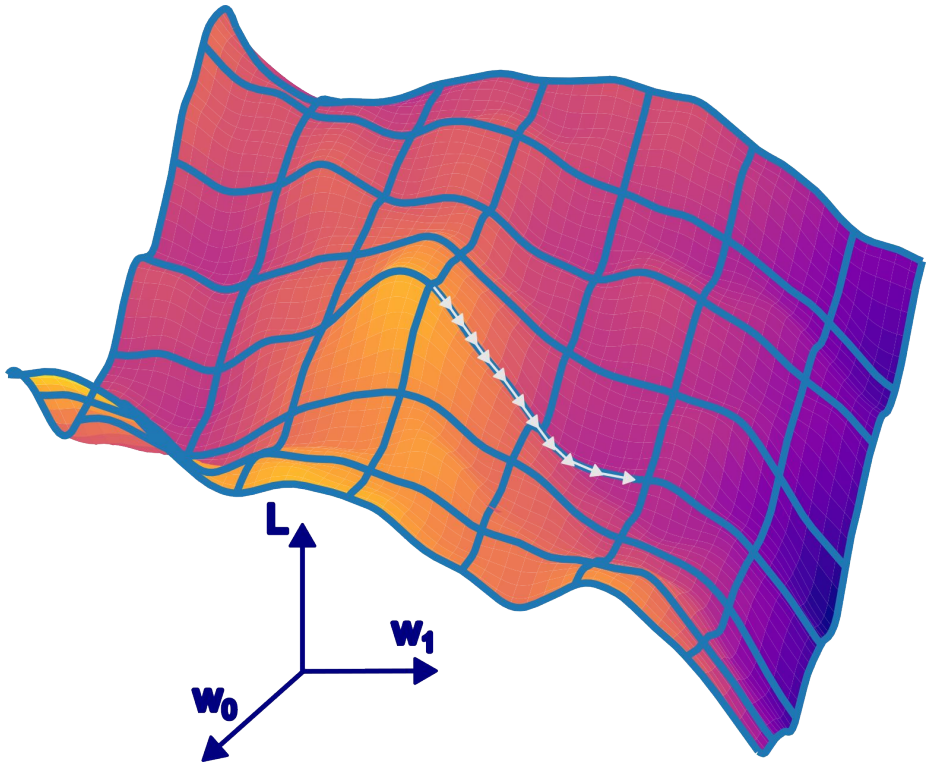
Use the gradient of the loss landscape, which informs us about the largest increase in the error landscape. We therefore take the negative gradient and thus obtain our learning rule:

$$\Delta\vec{w} = -\alpha \nabla L(w)$$

$$= -\alpha \begin{pmatrix} \frac{\partial L(w)}{\partial w_1} \\ \vdots \\ \frac{\partial L(w)}{\partial w_n} \end{pmatrix}$$

where we have also introduced the learning rate.

As can be seen in the figure, updating the weights in the direction of the negative gradient causes us to run in the direction of the minimum of the loss landscape.

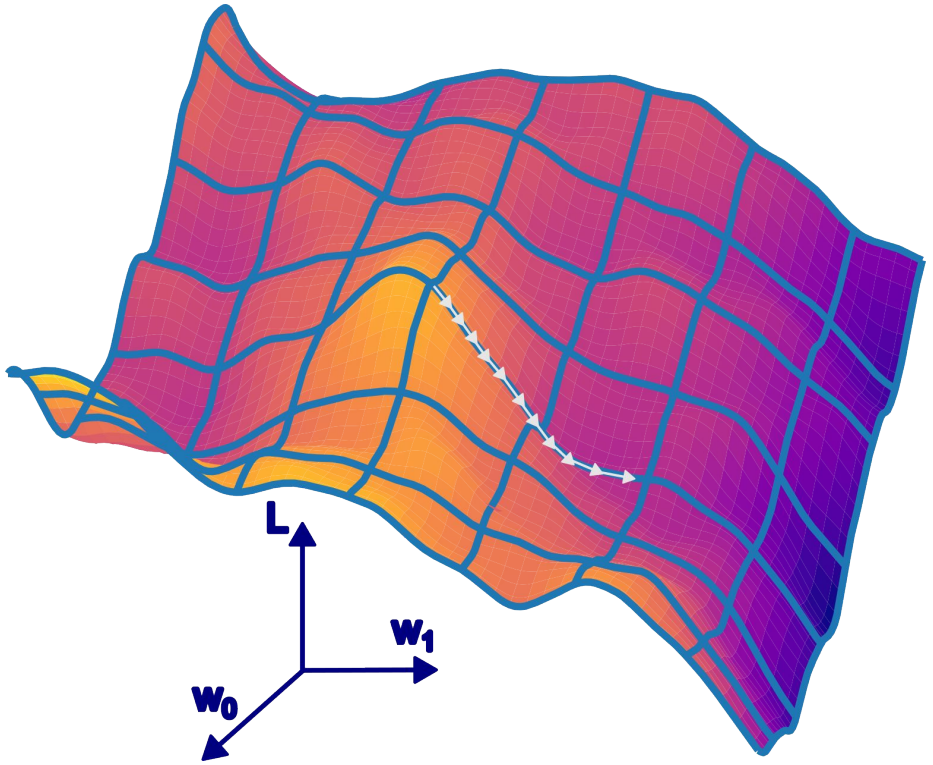


Perceptron

Weight Update

Rewrite the loss terms, so we can take a derivative with respect to the weights:

$$
\begin{aligned}
L(w) &= \frac{1}{2N}\sum_{k=1}^{N}\left[y^k - \bar{y}^k\right]^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\left[y^k - g(z^k)\right]^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\left(y^k - z^k\right)^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\left(y^k - \vec{w}^T\vec{x}^k\right)^2 \quad .
\end{aligned}
$$

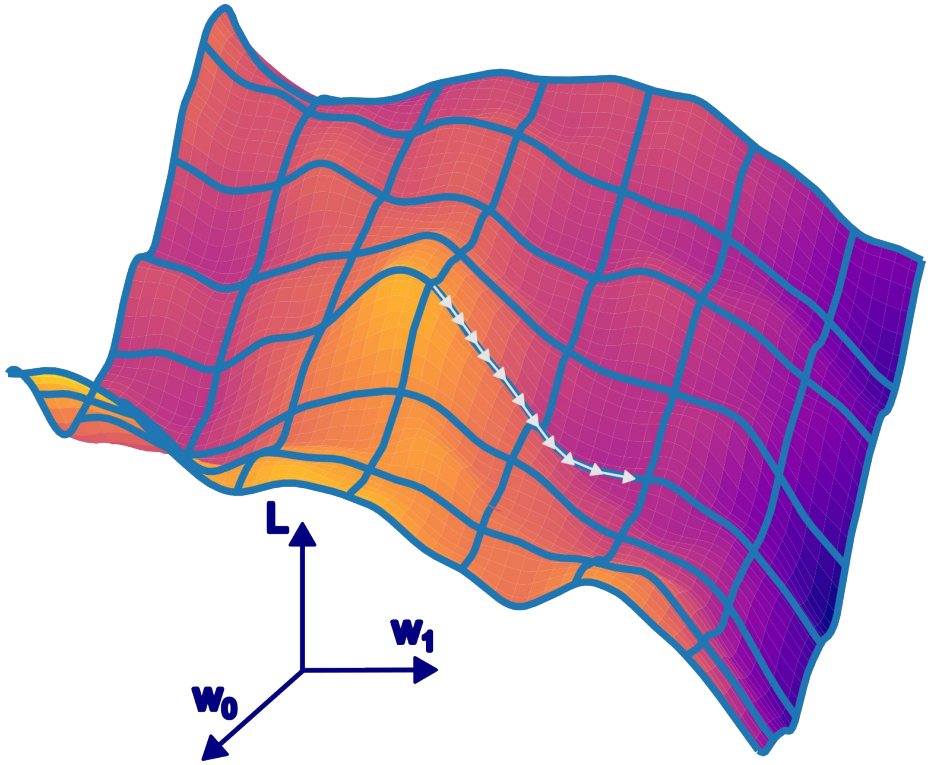


Perceptron

Weight Update

Now we can evaluate the derivative:

$$
\begin{aligned}
\frac{\partial L}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2N}\sum_{k=1}^{N}\left(y^k-\vec{w}^T\vec{x}^k\right)^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\frac{\partial}{\partial w_i}\left(y^k-\vec{w}^T\vec{x}^k\right)^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}2\left(y^k-\vec{w}^T\vec{x}^k\right)\frac{\partial}{\partial w_i}\left(y^k-\vec{w}^T\vec{x}^k\right) \\
&= \frac{1}{N}\sum_{k=1}^{N}\left(y^k-\vec{w}^T\vec{x}^k\right)\frac{\partial}{\partial w_i}\left[-(w_1,\ldots,w_i,\ldots,w_n)\begin{pmatrix}x_1^k\\ \vdots\\ x_i^k\\ \vdots\\ x_n^k\end{pmatrix}\right] \\
&= -\frac{1}{N}\sum_{k=1}^{N}\left(y^k-\vec{w}^T\vec{x}^k\right)\frac{\partial}{\partial w_i}\left[(w_1x_1^k+\cdots+w_ix_i^k+\cdots+w_nx_n^k)\right] \\
&= -\frac{1}{N}\sum_{k=1}^{N}\left(y^k-\vec{w}^T\vec{x}^k\right)x_i^k\ .
\end{aligned}
$$



Perceptron

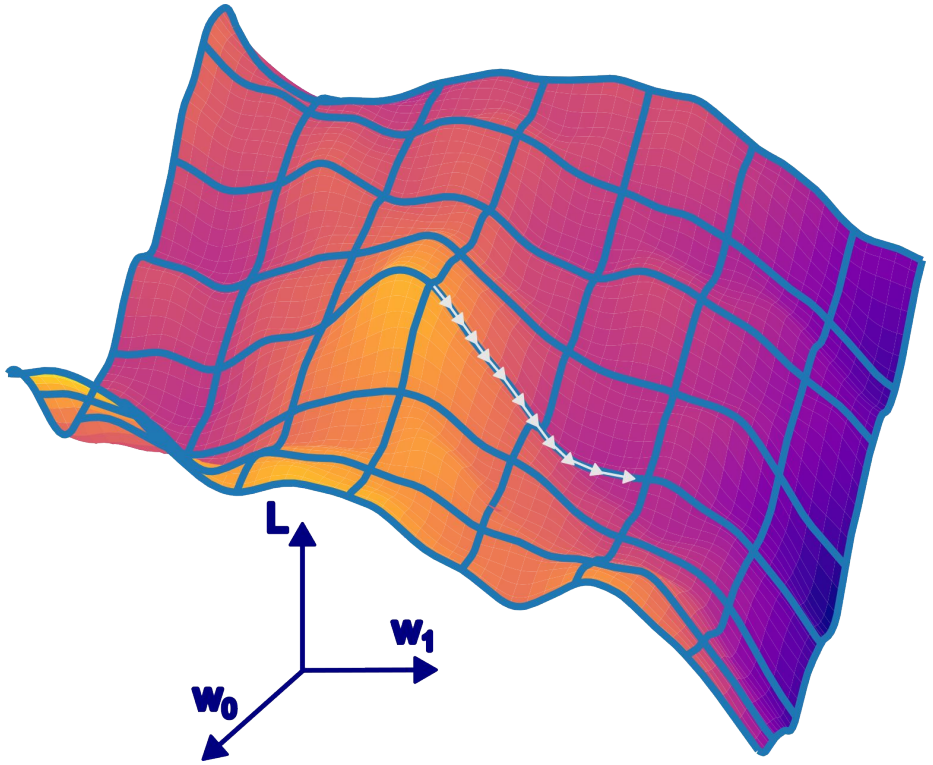
Weight Update

At the end we resubstitute the the definition of the activation function and the output:

$$\Delta w_i = -\alpha \frac{\partial L(w)}{\partial w_i}$$

$$= \frac{\alpha}{N} \sum_{k=1}^{N} \left[y^k - g(z^k) \right] x_i^k$$

$$= \frac{\alpha}{N} \sum_{k=1}^{N} \left(y^k - \bar{y}^k \right) x_i^k$$



Thus, we have derived our learning rule using gradient descent.
As we can see, we first need to process the sum over all training data (N times) to calculate our weight update for a learning step.

Binary Classification with the Perceptron

Learning Algorithm

With the help of the derived learning rule, we can now formulate a learning algorithm.

1. initialize all weights $\vec{w} = (w_0, \ldots, w_n)$.
2. for each epoch:
   - Set $\Delta w_i = 0$
   - For each set of training data $(x^k, y^k), k = 1, \ldots, N$:
     - Calculate output y^k.
     - Calculate weight update: $\Delta w_i^k = \Delta w_i^k + (y^k - \bar{y}^k) x_i^k$.
   - Calculate the mean of all weight updates over the training data: $\Delta w_i = \frac{\alpha}{N} \sum_{k=1}^{N} \Delta w_i^k$.
   - Update all weights $w_i = w_i + \Delta w_i$

Binary Classification with the Perceptron

Batch Processing of the Training Data

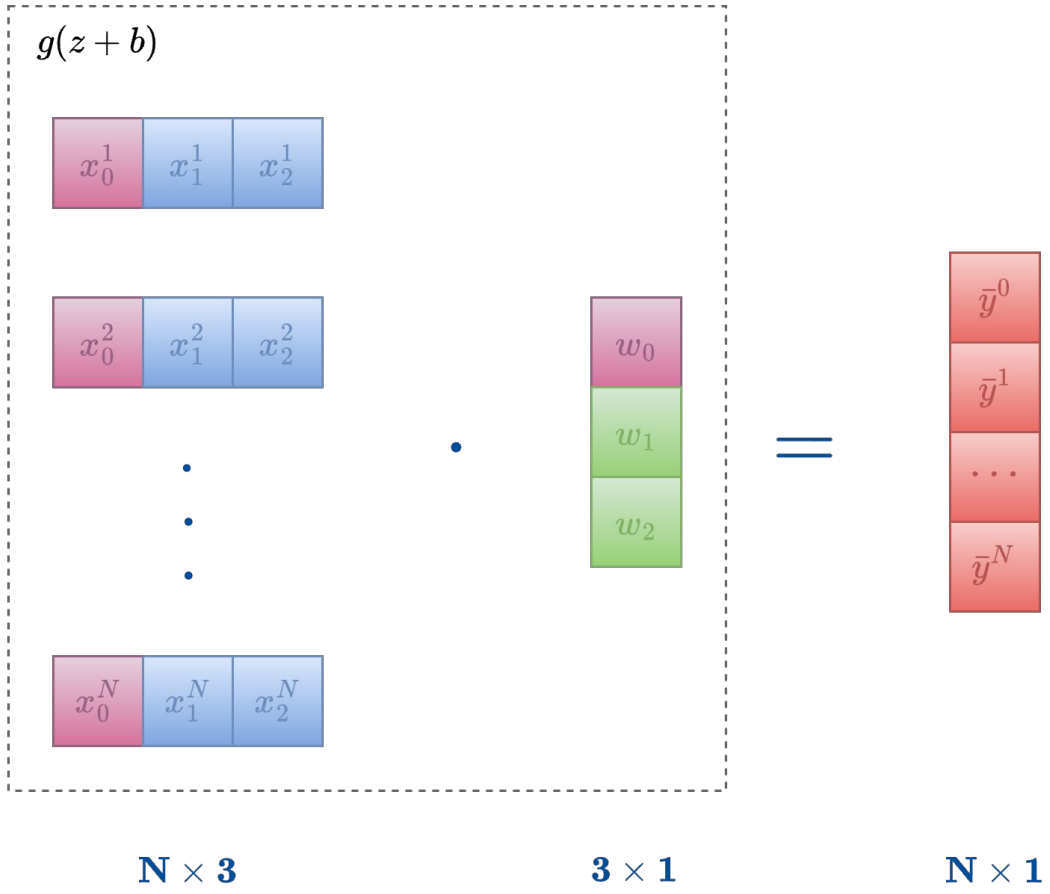
We have implemented the perceptron in such a way that the matrix x contains all feature vectors.

This ensures that all training data is run through before the weights are updated. The feature vectors are stacked on top of each other to form an N x 3 input matrix.

It should be noted that the first column of the matrix represents the bias neuron, i.e. contains only ones.

By matrix multiplying the N x 3 input matrix with the 3 x 1 weight vector, we obtain an N x 1 output vector that contains all outputs for all training examples.

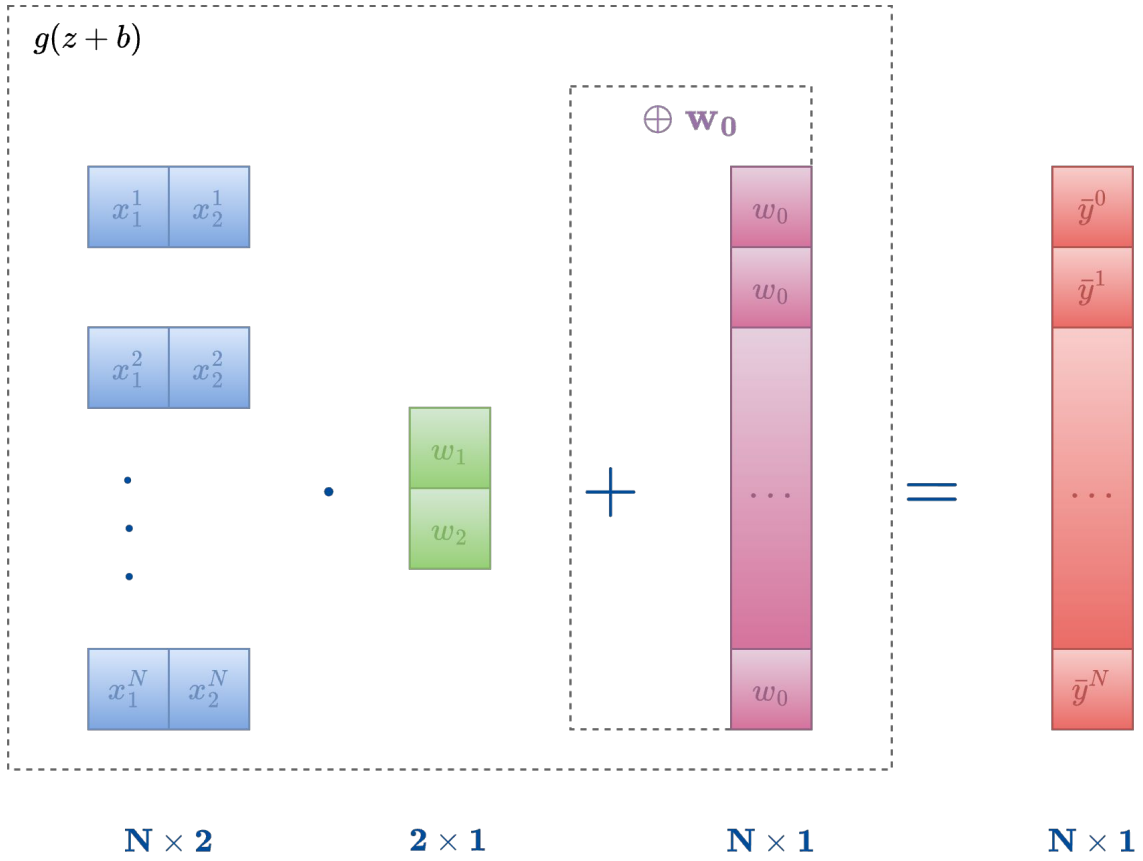
The input vectors are actually row vectors and stacked on top of each other.



Binary Classification with the Perceptron

Batch Processing of the Training Data

However, we can make the implementation more efficient by “broadcasting” the weight of the bias neuron.



Binary Classification with the Perceptron

Implementation of the Weight Update

Recall the definition of the weight update:

$$\vec{w} := \vec{w} + \Delta\vec{w}$$

$$\Delta w_i = -\alpha \frac{\partial L(w)}{\partial w_i} = \frac{\alpha}{N} \sum_{k=1}^{N} \left(y^k - \bar{y}^k\right) x_i^k$$

$$\begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^N \\ x_2^1 & x_2^2 & & x_2^N \end{bmatrix} \cdot \begin{bmatrix} y^1 - \bar{y}^1 \\ y^2 - \bar{y}^2 \\ \cdots \\ y^N - \bar{y}^N \end{bmatrix} = \begin{bmatrix} x_1^1(y^1 - \bar{y}^1) + \cdots + x_1^N(y^N - \bar{y}^N) \\ x_2^1(y^1 - \bar{y}^1) + \cdots + x_2^N(y^N - \bar{y}^N) \end{bmatrix}$$

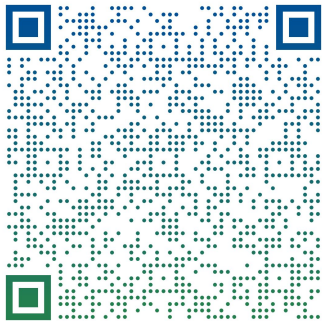
$\mathbf{2 \times N}$ $\quad$ $\mathbf{N \times 1}$ $\quad$ $\mathbf{2 \times 1}$

01 Introduction to Deep Learning

01.2 Interactive Session: Machine Learning Colors

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science



IMPRS for Quantum Dynamics and Control

Summer School 2024: Machine Learning and Many Body Systems in or out of Equilibrium

Wroclaw, Poland

July 30, 2024

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

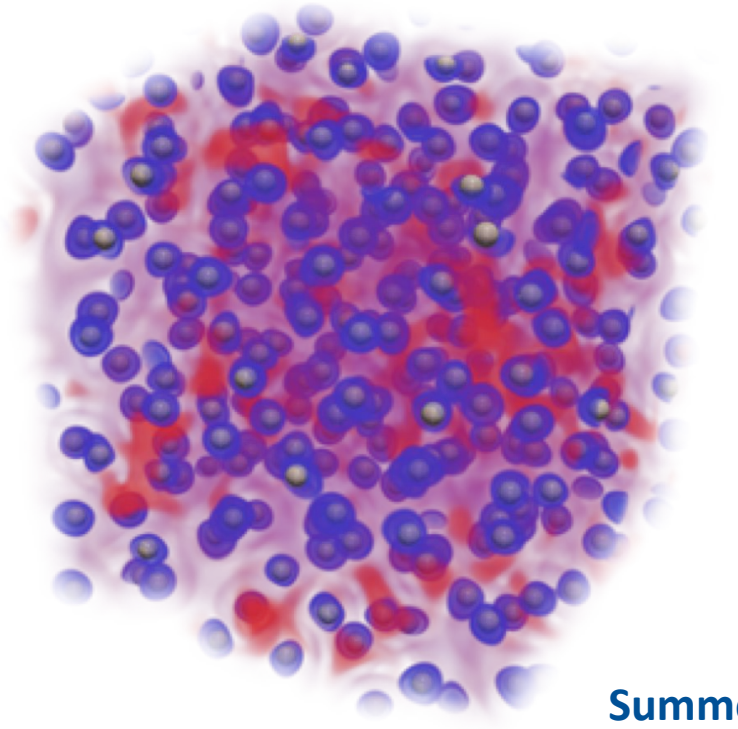
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

01 Introduction to Deep Learning

01.3 Research Application:
Scalable Neural Networks for Predicting the Electronic Structure of Matter

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science



MALA
Materials Learning Algorithms

IMPRS for Quantum Dynamics and Control

Summer School 2024: Machine Learning and Many Body Systems in or out of Equilibrium

Wroclaw, Poland

July 30, 2024

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

Outline

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

- Theoretical Background
  - Electronic Structure Problem
  - Density Functional Theory
- Motivation
  - Electronic Structures at Scale from Machine Learning
- Neural Network Model for Density Functional Theory
- Outlook

Outline

Electronic Structure Problem

Non-relativistic Schrödinger equation

$$\hat{H}(\underline{\mathbf{r}}, \underline{\mathbf{R}})\Psi(\underline{\mathbf{r}}, \underline{\mathbf{R}}) = E\Psi(\underline{\mathbf{r}}, \underline{\mathbf{R}})$$

$$\hat{H}(\underline{\mathbf{r}}; \underline{\mathbf{R}}) = \hat{T}_e(\underline{\mathbf{r}}) + \hat{V}_{ee}(\underline{\mathbf{r}}) + \hat{V}_{ei}(\underline{\mathbf{r}}; \underline{\mathbf{R}}) + \hat{V}_{ii}(\underline{\mathbf{R}})$$

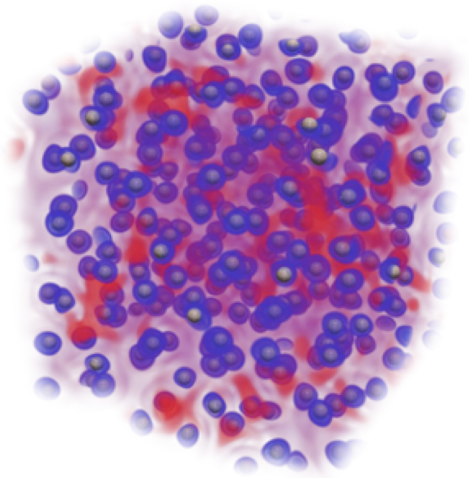
$$\hat{T}_e(\underline{\mathbf{r}}) = \sum_i^{N_e} -\frac{\nabla_i^2}{2}$$

$$\hat{V}_{ee}(\underline{\mathbf{r}}) = \sum_i^{N_e} \sum_{j \neq i}^{N_e} \frac{1}{2} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}$$

$$\hat{V}_{ei}(\underline{\mathbf{r}}; \underline{\mathbf{R}}) = -\sum_i^{N_e} \sum_\alpha^{N_i} \frac{Z_\alpha}{|\mathbf{r}_i - \mathbf{R}_\alpha|}$$

$$\hat{V}_{ii}(\underline{\mathbf{R}}) = -\sum_\alpha^{N_i} \sum_{\beta \neq \alpha}^{N_i} \frac{1}{2} \frac{Z_\alpha Z_\beta}{|\mathbf{R}_\alpha - \mathbf{R}_\beta|}$$

Molecular and materials properties



Molecular structure, Crystal structure, Charge density, Cohesive energy, Elastic properties, Vibrational properties, Magnetic order, Dielectric susceptibility, Magnetic susceptibility, Phase transitions, Bond dissociation, Enthalpies of formation, Ionization potential, Electron affinity, Band gaps, Equation of state

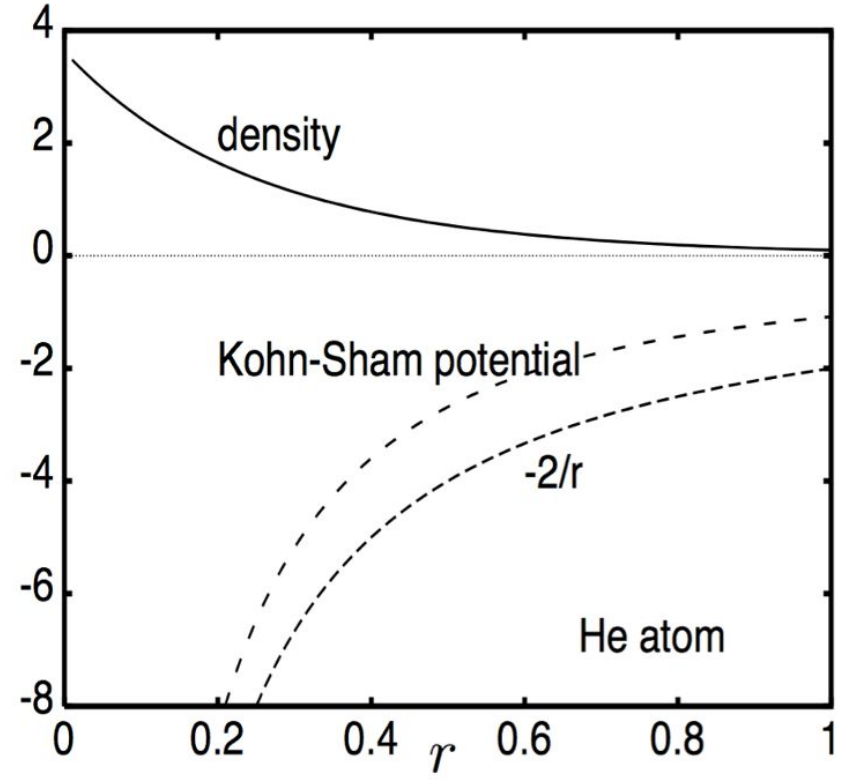
Density Functional Theory

$$\left[-\frac{1}{2}\nabla^2 + v_{\mathrm{S}}(\mathbf{r};\underline{\mathbf{R}})\right]\phi_j(\mathbf{r};\underline{\mathbf{R}}) = \epsilon_j\phi_j(\mathbf{r};\underline{\mathbf{R}})$$

$$v_{\mathrm{S}}(\mathbf{r};\underline{\mathbf{R}}) = \frac{\delta U[n]}{\delta n(\mathbf{r};\underline{\mathbf{R}})} + \frac{\delta E_{\mathrm{XC}}[n]}{\delta n(\mathbf{r};\underline{\mathbf{R}})} + v_{ei}(\mathbf{r};\underline{\mathbf{R}})$$

$$n(\mathbf{r};\underline{\mathbf{R}}) = \sum_j \phi_j^*(\mathbf{r};\underline{\mathbf{R}})\phi_j(\mathbf{r};\underline{\mathbf{R}})$$

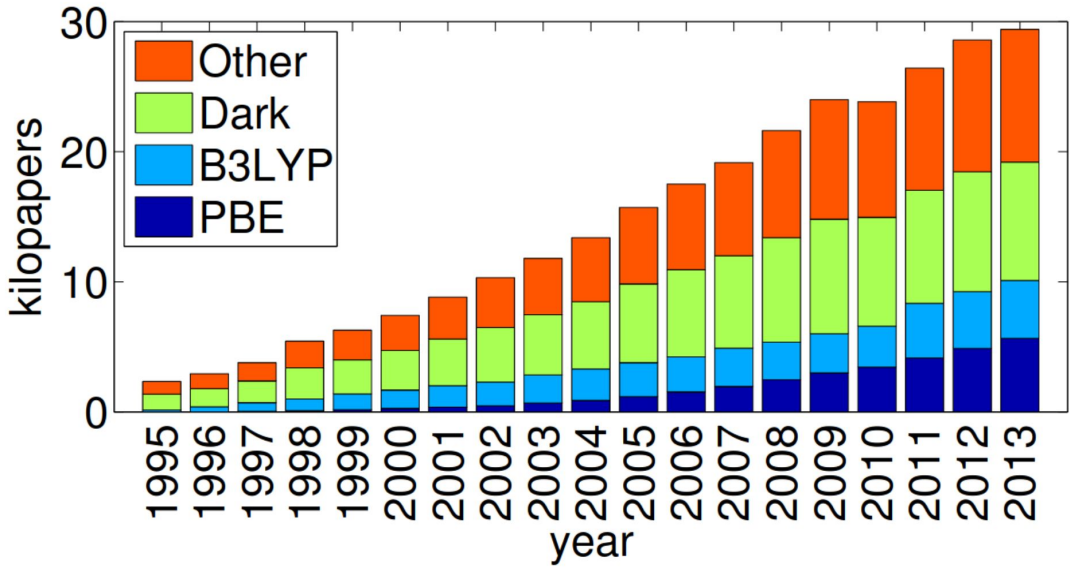
$$E[n] = T_{\mathrm{S}}[n] + U[n] + E_{\mathrm{XC}}[n] + \int d^3r\; n(\mathbf{r};\underline{\mathbf{R}})v_{ei}(\mathbf{r};\underline{\mathbf{R}})$$



K. Burke, „The ABC of DFT“.

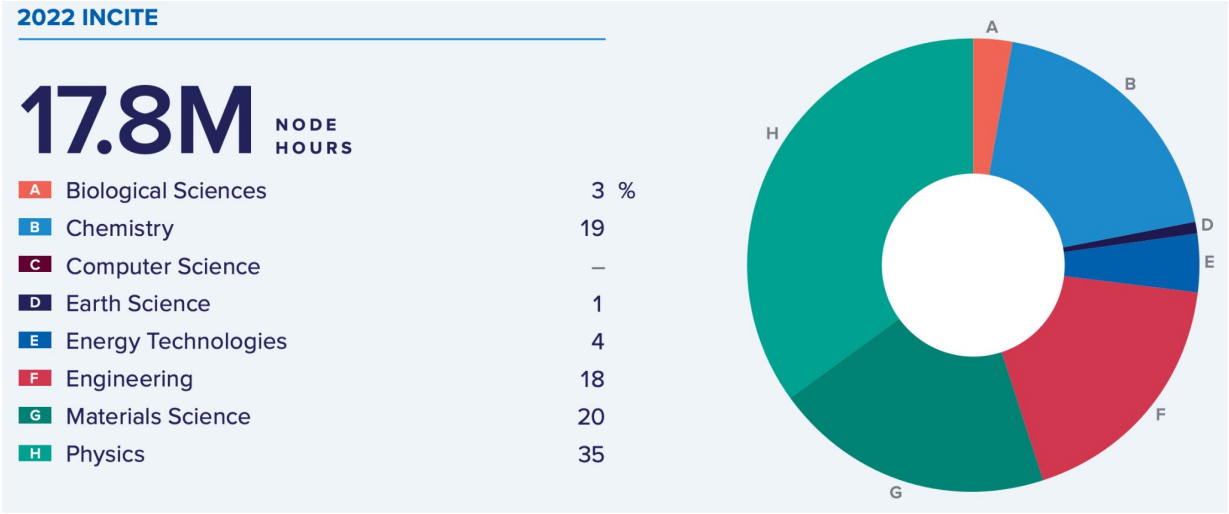
Density Functional Theory

Most popular method for solving the electronic structure problem



Pribram-Jones et al.,
https://doi.org/10.1146/annurev-physchem-040214-121420

One of the world's largest computational expenses



2022 ALCF Annual report, https://ar22.alcf.anl.gov/

Outline

Theoretical Background

Electronic Structure Problem

Density Functional Theory

Motivation

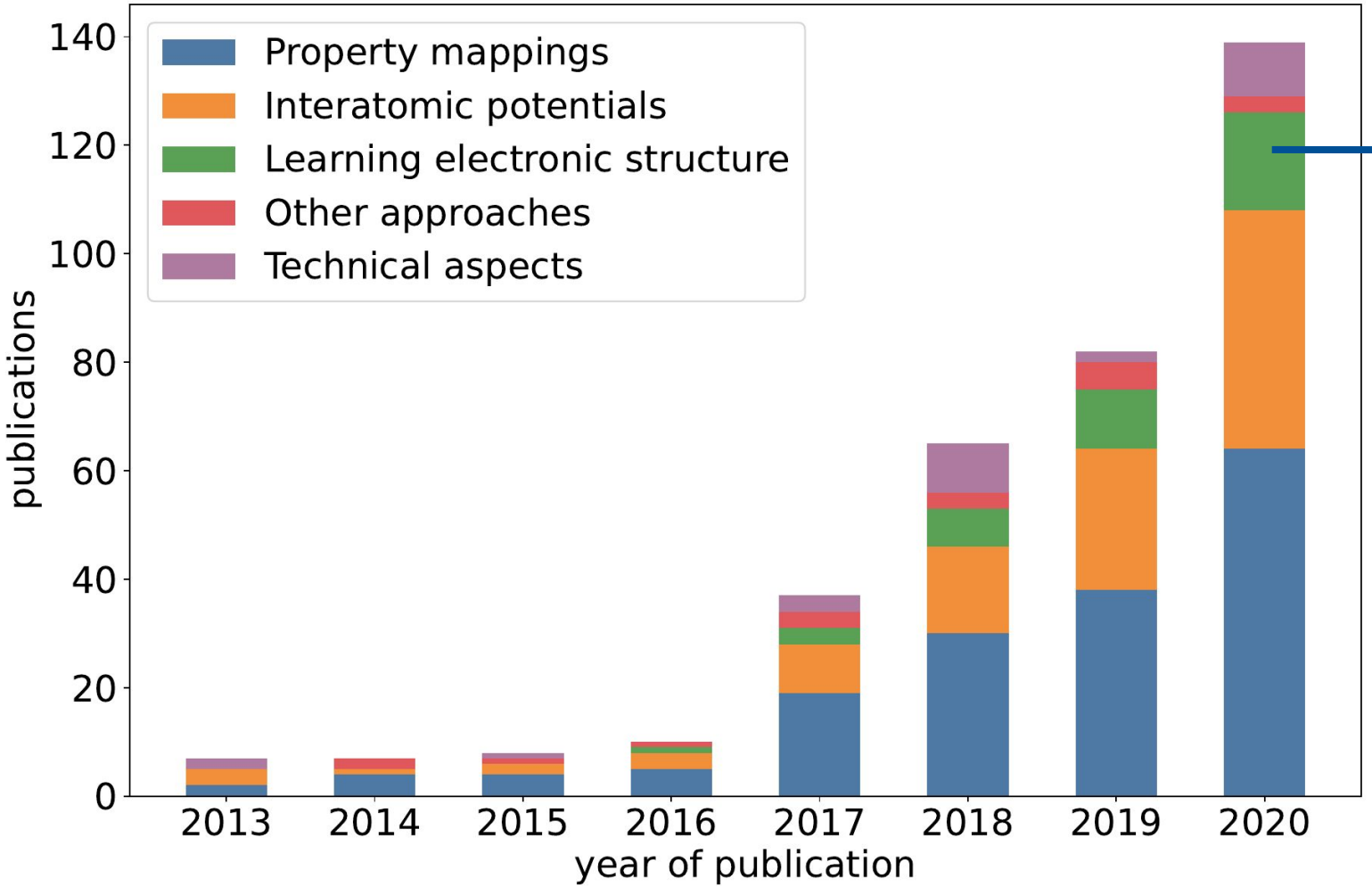
Electronic Structures at Scale from Machine Learning

Neural Network Model for Density Functional Theory

Outlook

Motivation

State of the art in combining electronic structure theory with machine learning



Machine learning the electronic structure (this work)

Meta study analyzing 370 research articles

L. Fiedler, K. Shah, M. Bussmann, A. Cangi, Phys. Rev. Mater. 6, 040301 (2022).

Outline

Theoretical Background

Electronic Structure Problem

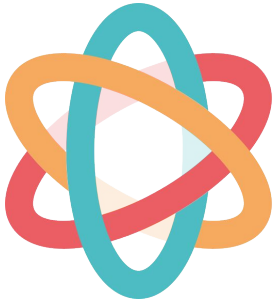
Density Functional Theory

Motivation

Electronic Structures at Scale from Machine Learning

Neural Network Model for Density Functional Theory

Outlook



M Λ L Λ

Materials Learning Algorithms

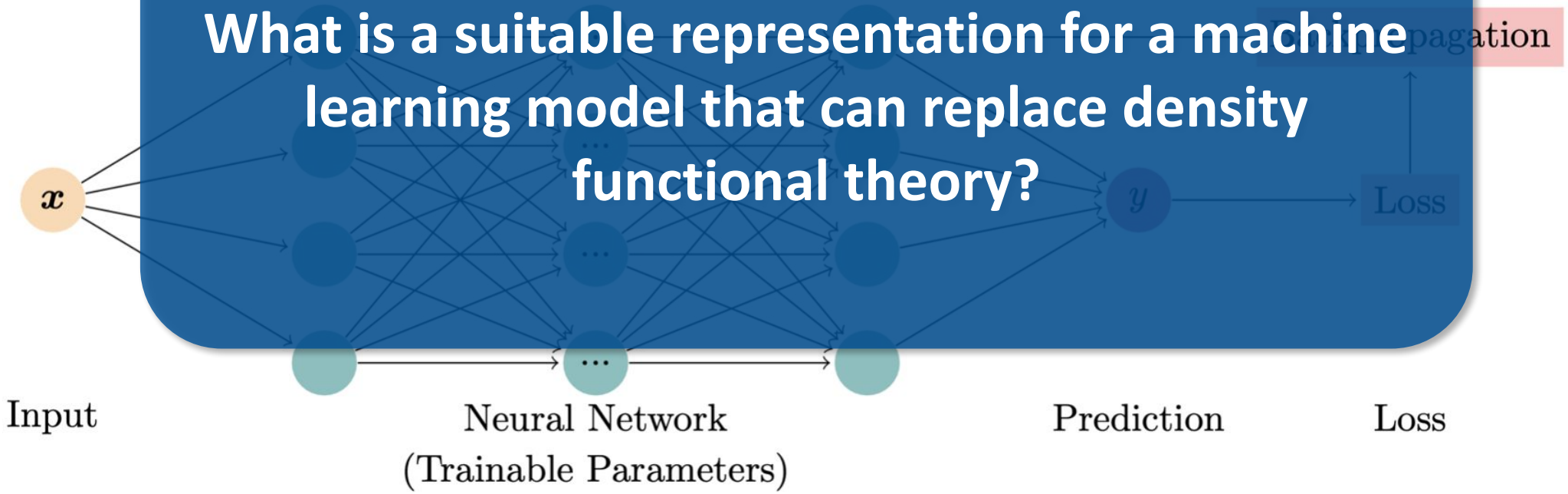
Machine Learning

Neural Networks

Optimize loss function based on the difference between true and predicted values

$$L_{NN} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - y^{*(i)}\right)^2$$

What is a suitable representation for a machine learning model that can replace density functional theory?



Materials Learning Algorithms

Finding a suitable representation for machine learning

Need a suitable ML model yielding total energies. The KS kinetic energy is problematic.

$$E[n] = T_S[n] + U[n] + E_{XC}[n] + \int d^3r\; n(\mathbf{r})\, v_{ei}(\mathbf{r})$$

Express the total energy as

$$E[n] = E_b - U + E_{XC}[n] - \int d^3r\; n(\mathbf{r})\, v_{XC}(\mathbf{r})$$

With the band energy

$$E_b = \sum_j \epsilon_j = T_S + \int d^3r\; n(\mathbf{r})\, v_S(\mathbf{r}) = T_S + \int d^3r\; n(\mathbf{r})\; [v_{ei}(\mathbf{r}) + v_H(\mathbf{r}) + v_{XC}(\mathbf{r})]$$

Materials Learning Algorithms

Finding a suitable representation for machine learning

Consider the local density of states:

$$d(\epsilon, \mathbf{r}) = \sum_j \phi_j^*(\mathbf{r})\phi_j(\mathbf{r})\delta(\epsilon - \epsilon_j) = G_S(\epsilon, \mathbf{r}, \mathbf{r}' = \mathbf{r})$$ **(Diagonal of the single-particle Green function)**

$$n(\mathbf{r}) = \int d\epsilon \; d(\epsilon, \mathbf{r})$$ **(Electron density)**

$$D(\epsilon) = \int d^3r \; d(\epsilon, \mathbf{r})$$ **(Density of states)**

$$E_b = \sum_j \epsilon_j = \int d\epsilon \; \epsilon \, D(\epsilon)$$ **(Band energy)**

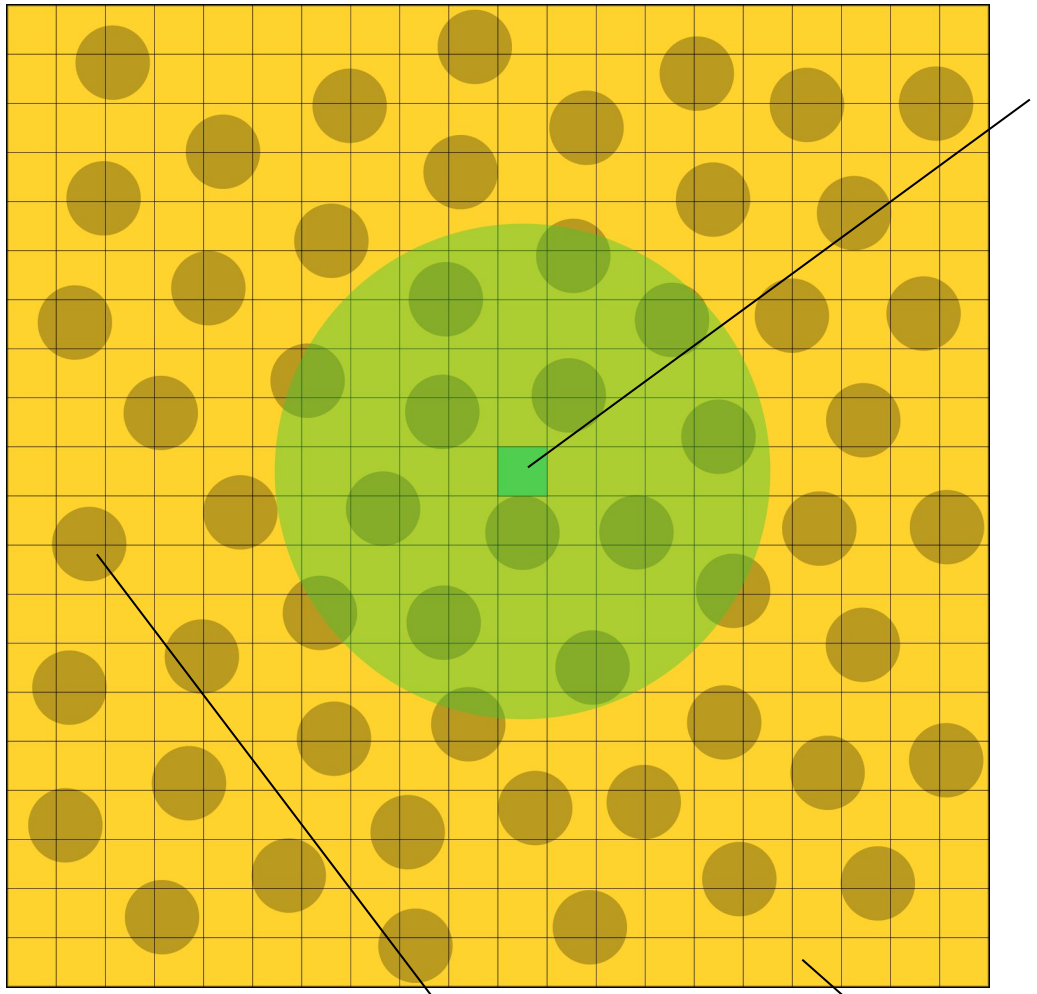
This allows us to evaluate the total energy from just one quantity – the local density of states.

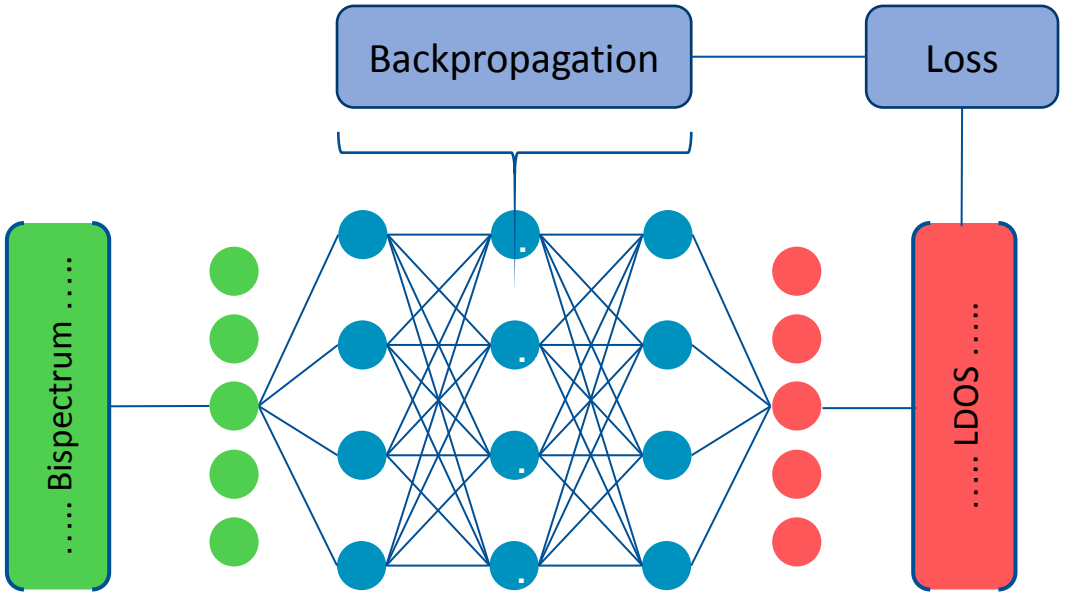
$$E[n] = E_b - U + E_{XC}[n] - \int d^3r \; n(\mathbf{r}) \, v_{XC}(\mathbf{r})$$

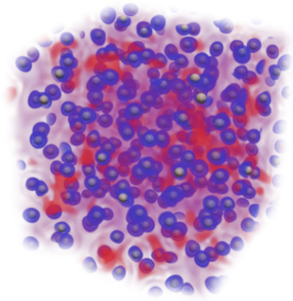
We have also extended this to finite electronic temperature.

Materials Learning Algorithms

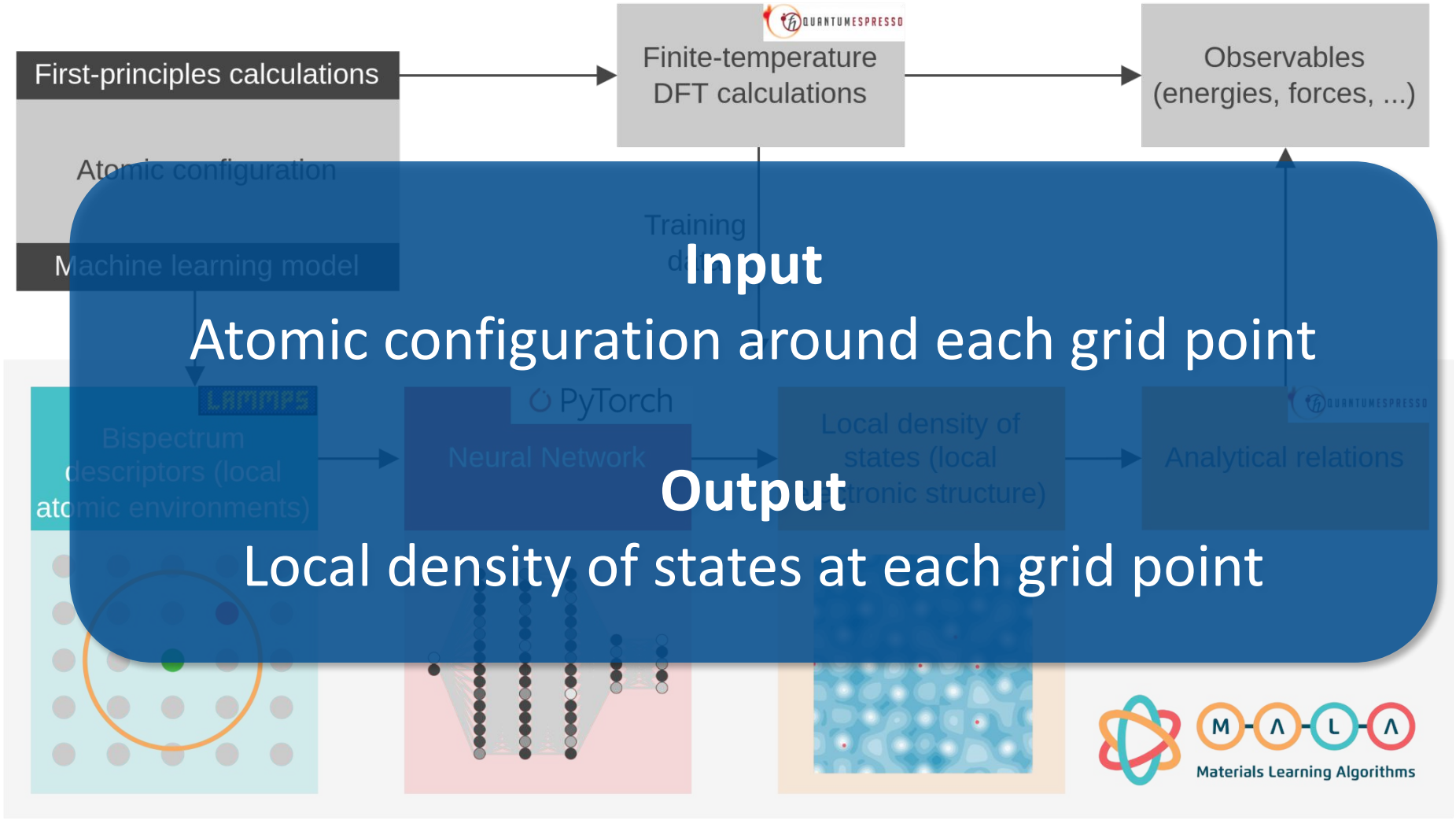
Spatially resolved neural-network model for the local density of states







Materials Learning Algorithms



github.com/mala-project

2023 R&D 100 WINNER

Materials Learning Algorithms

Spaces of Interpolation

Phase boundaries, i.e., ionic configurations
J. A. Ellis, L. Fiedler, G. A. Popoola, N. A. Modine, J. A. Stephens, A. P. Thompson, A. Cangi, S. Rajamanickam, Accelerating Finite-temperature Kohn-Sham Density Functional Theory with Deep Neural Networks, Phys. Rev. B 104, 035120 (2021).

Temperatures
L. Fiedler, N. A. Modine, K. D. Miller, A. Cangi, Machine learning the electronic structure of matter across temperatures, Phys. Rev. B 108, 125146 (2023).

Number of atoms
L. Fiedler, N. A. Modine, S. Schmerler, D. J. Vogel, G. A. Popoola, A. P. Thompson, S. Rajamanickam, A. Cangi, Predicting electronic structures at any length scale with machine learning, Npj Comput. Mater. 9, 115 (2023).

Mass density, Temperature-mass density, Atomic species, …

Materials Learning Algorithms

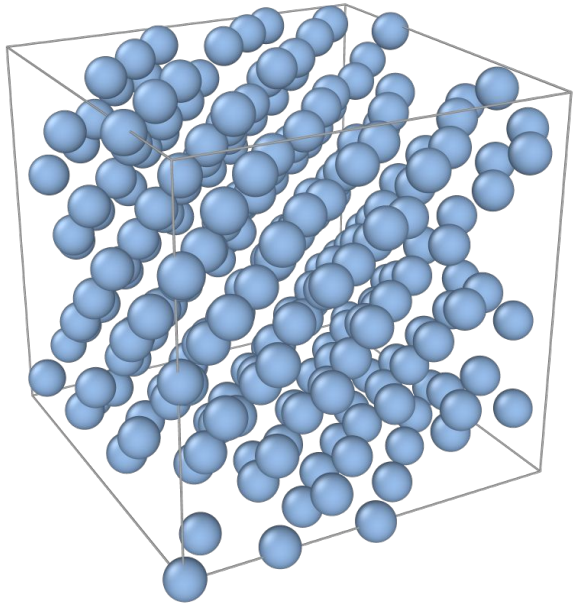
Interpolation across phase boundaries

Materials Learning Algorithms

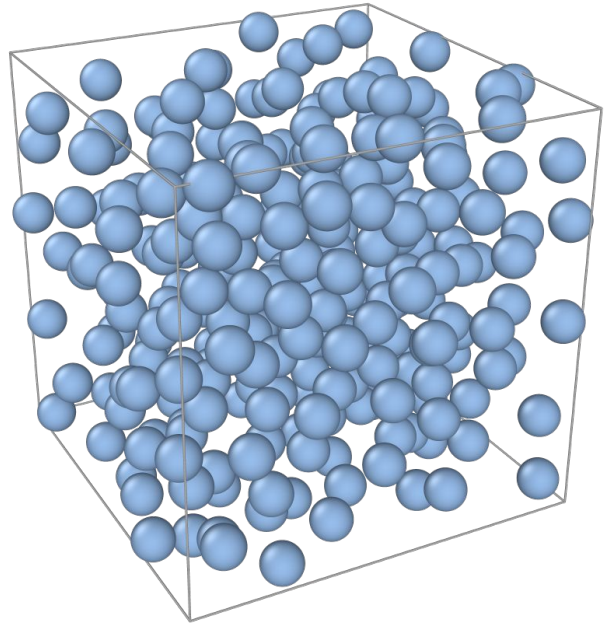
Interpolation across phase boundaries

Aluminum at the melting point (933 K) and ambient mass density (2.7 g/cc)

Solid snapshot



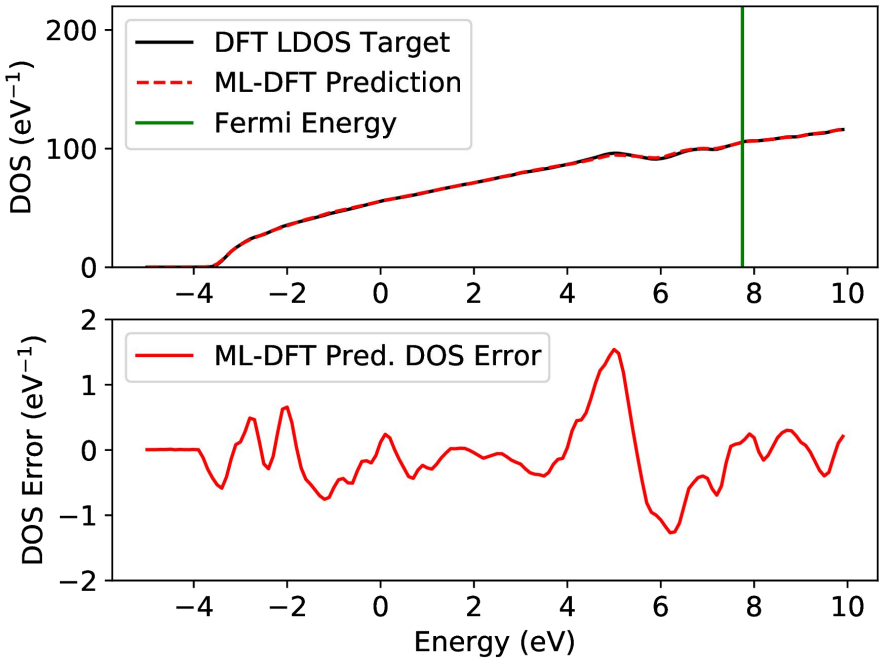
Liquid snapshot

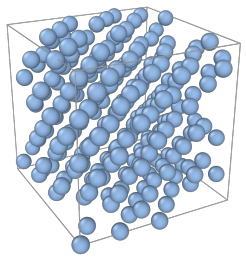


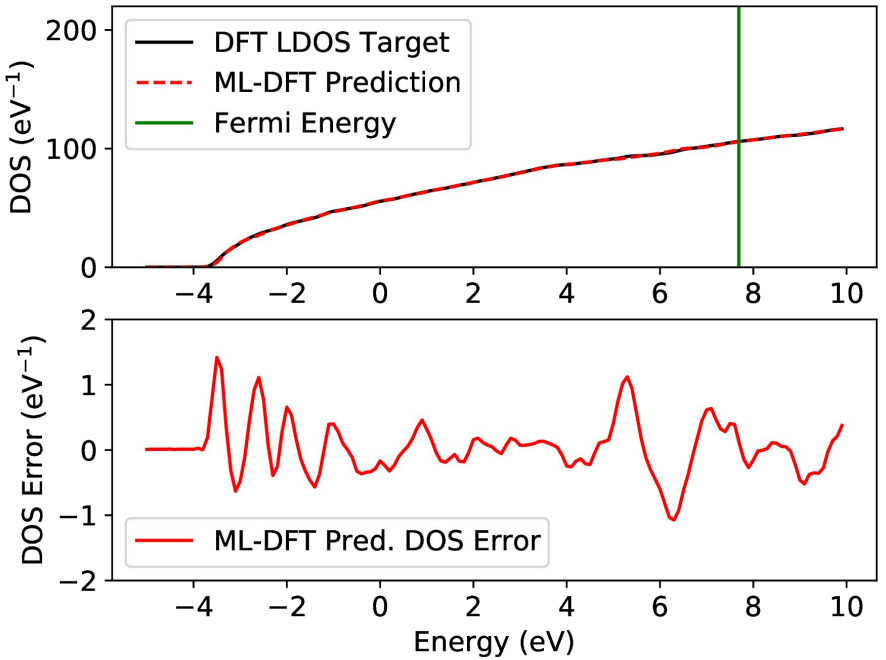
Materials Learning Algorithms

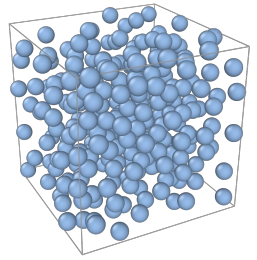
Interpolation across phase boundaries

Density of states





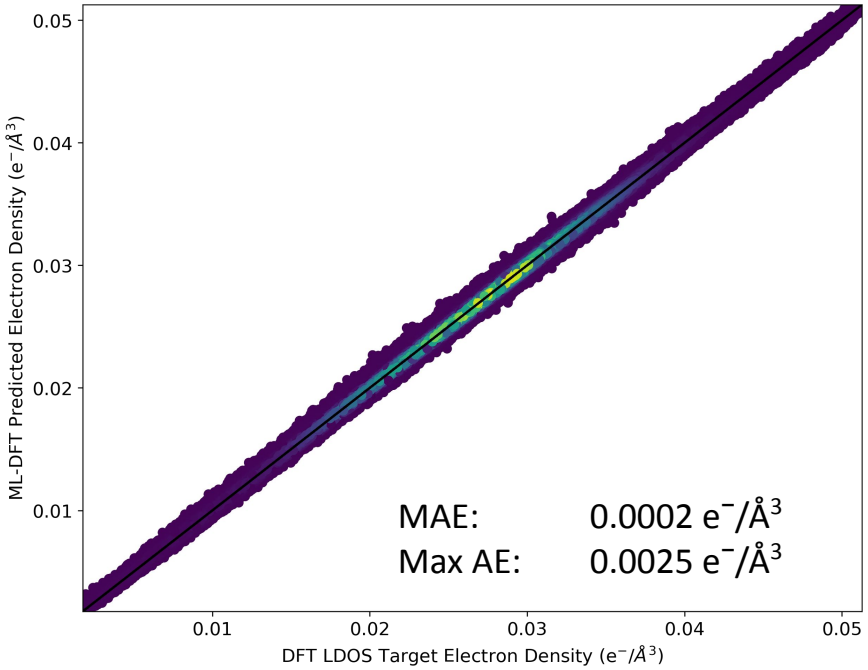


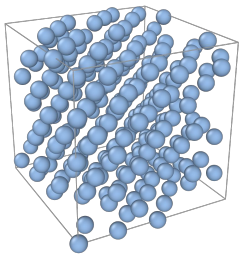


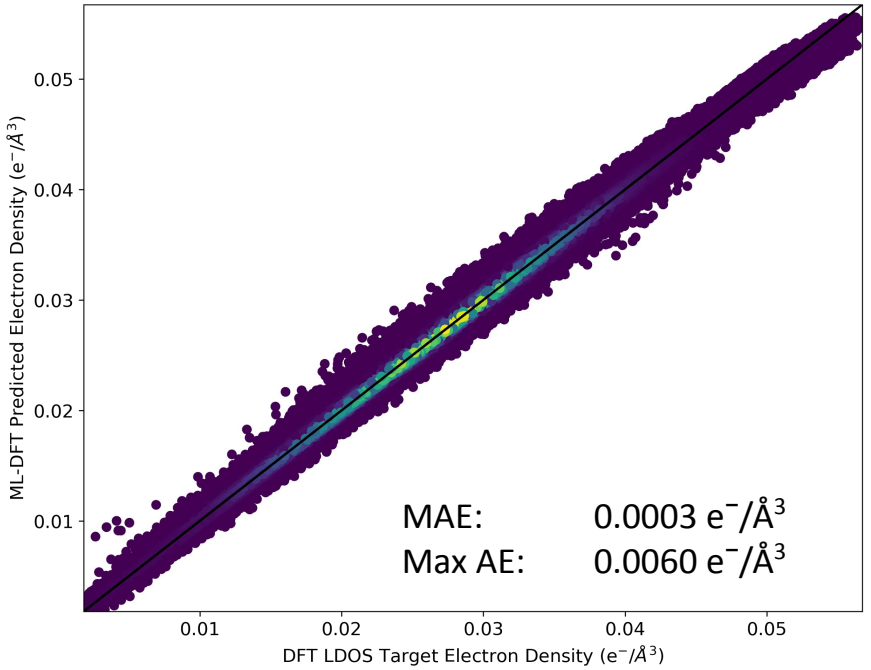
Materials Learning Algorithms

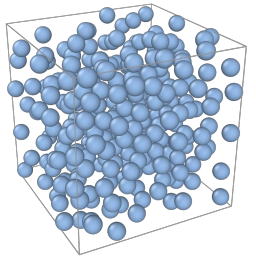
Interpolation across phase boundaries

Electron density







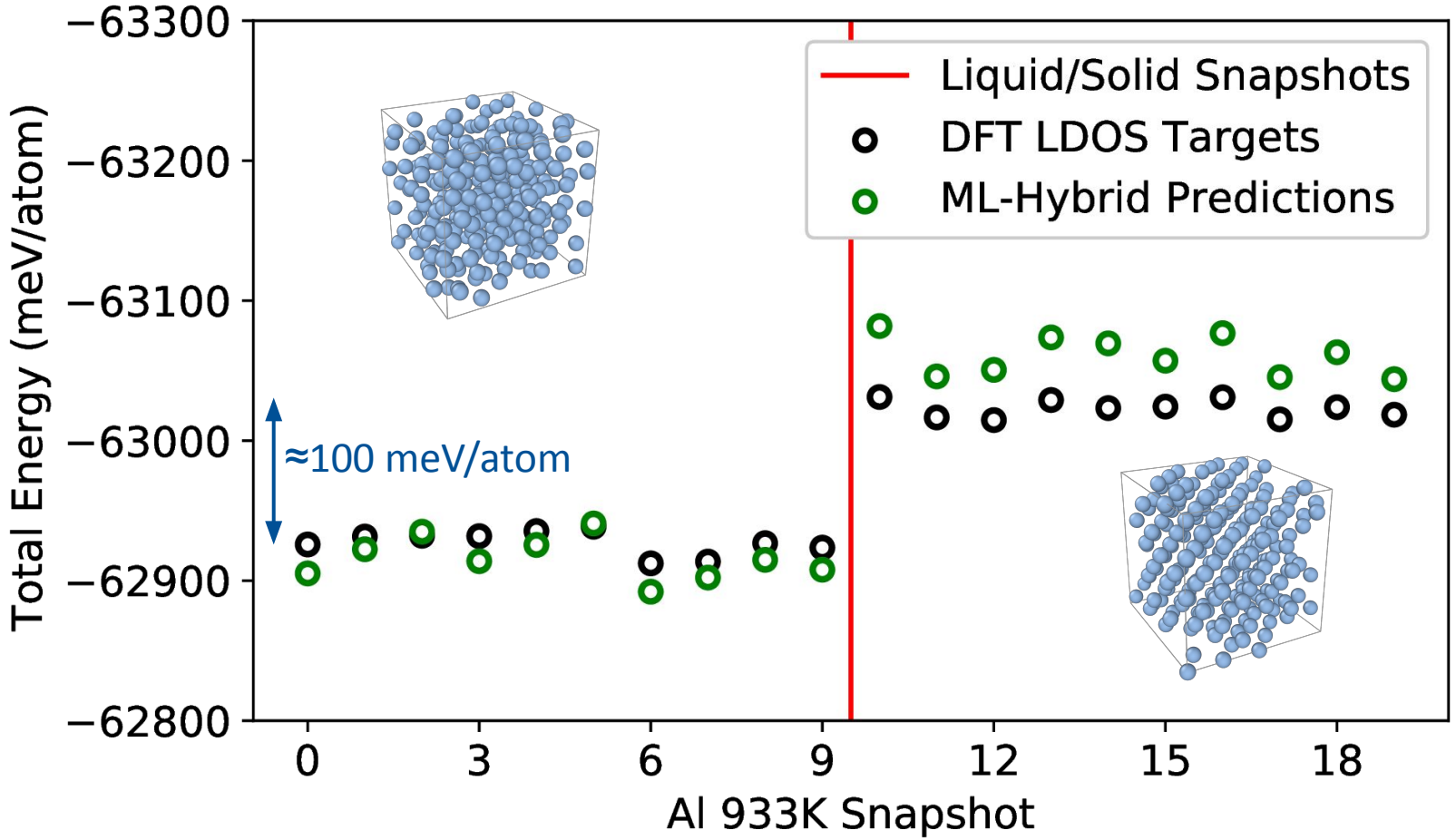


Materials Learning Algorithms

Interpolation across phase boundaries

Total energy

ML model can resolve the energy gap between the two phases.



Materials Learning Algorithms

Interpolation across temperatures

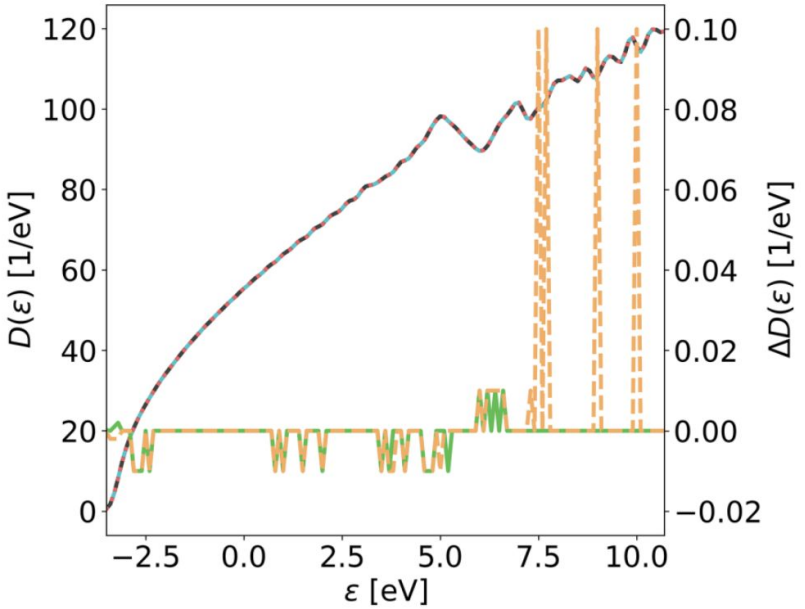
Materials Learning Algorithms

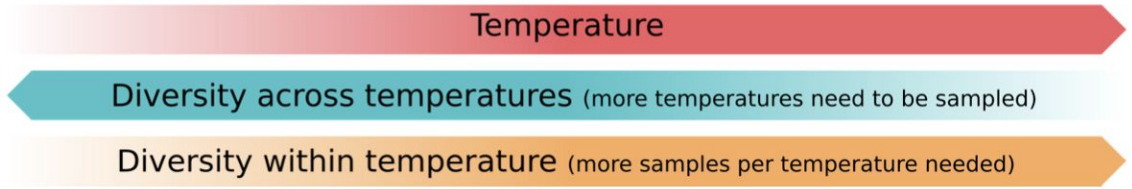
Interpolation across temperatures

Separate treatment of electronic and ionic temperature possible (laser-heated matter).

High accuracy across range of temperatures.

Extrapolation in electronic temperature domain trivially possible due to properties of the local density of states (infinite sum over energy eigenstates).





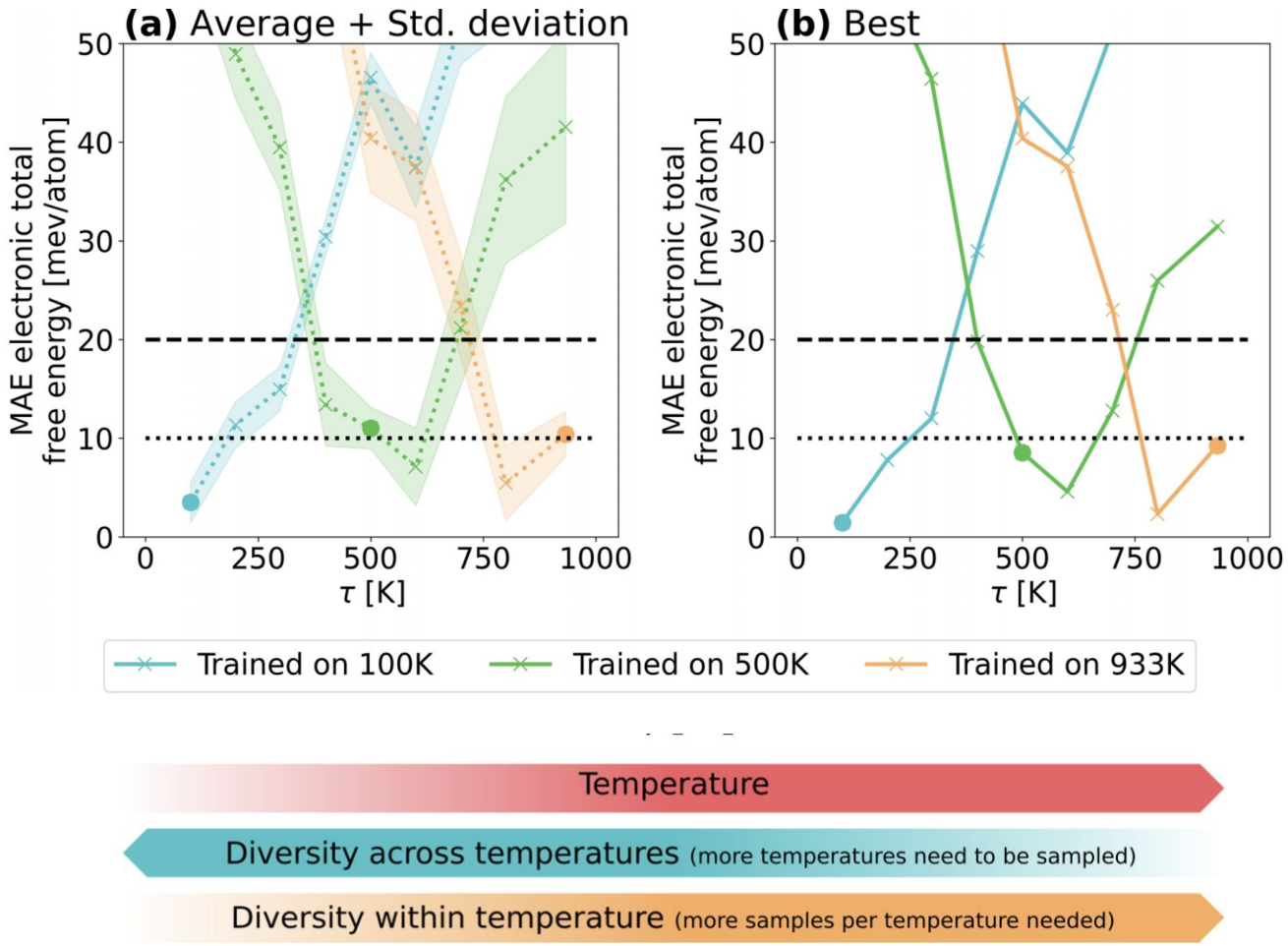
Materials Learning Algorithms

Interpolation across temperatures

Separate treatment of electronic and ionic temperature possible (laser-heated matter).

High accuracy across range of temperatures.

Extrapolation in electronic temperature domain trivially possible due to properties of the local density of states (infinite sum over energy eigenstates).



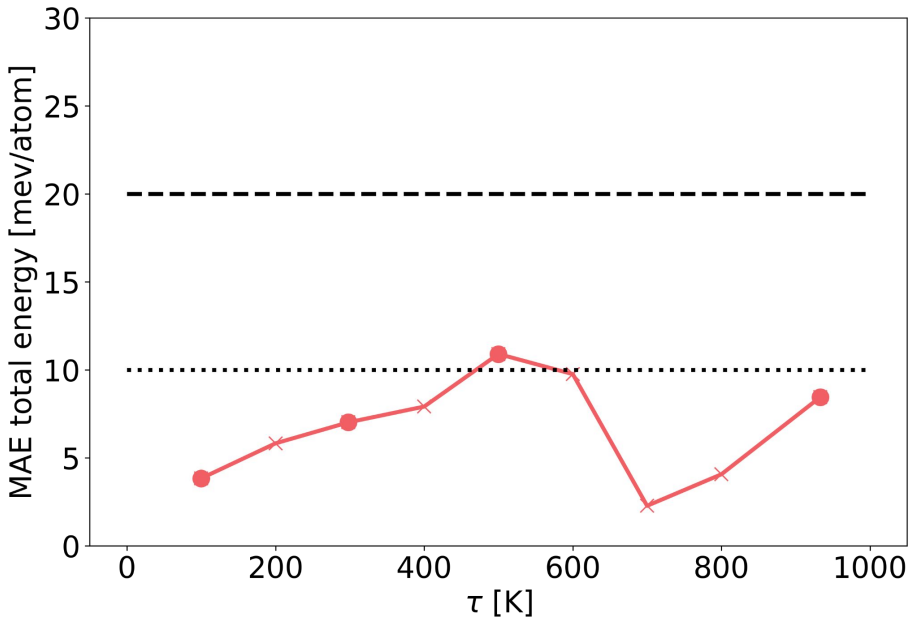
Materials Learning Algorithms

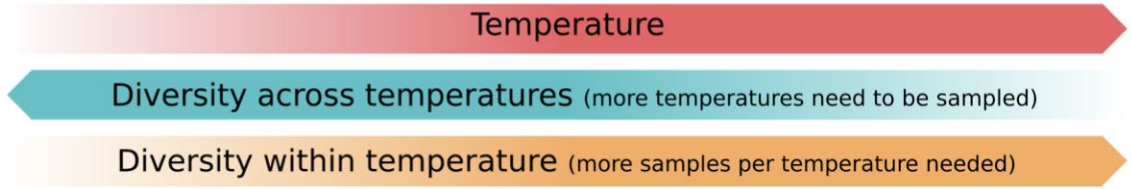
Interpolation across temperatures

Separate treatment of electronic and ionic temperature possible (laser-heated matter).

High accuracy across range of temperatures.

Extrapolation in electronic temperature domain trivially possible due to properties of the local density of states (infinite sum over energy eigenstates).





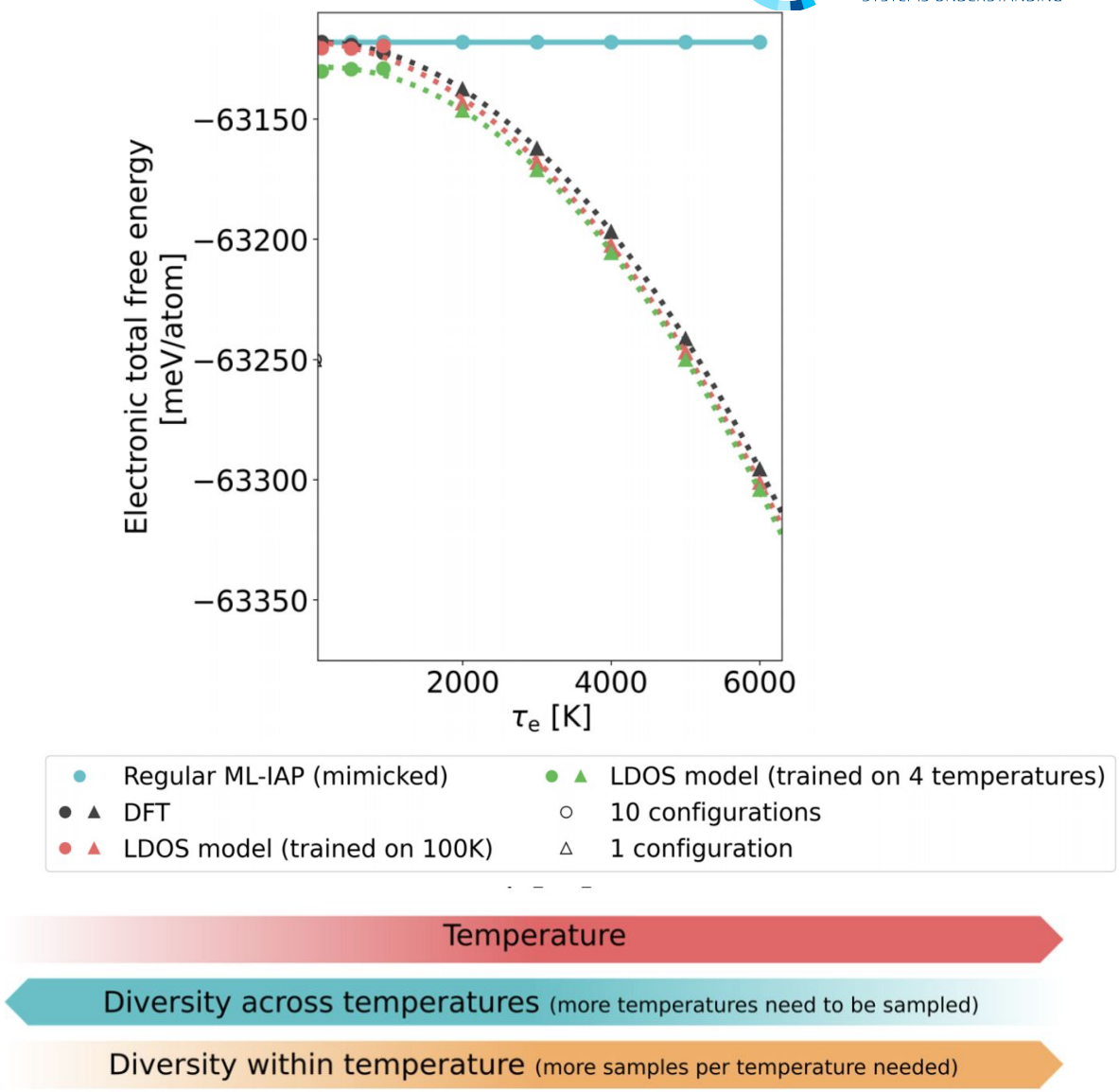
Materials Learning Algorithms

Interpolation across temperatures

Separate treatment of electronic and ionic temperature possible (laser-heated matter).

High accuracy across range of temperatures.

Extrapolation in electronic temperature domain trivially possible due to properties of the local density of states (infinite sum over energy eigenstates).

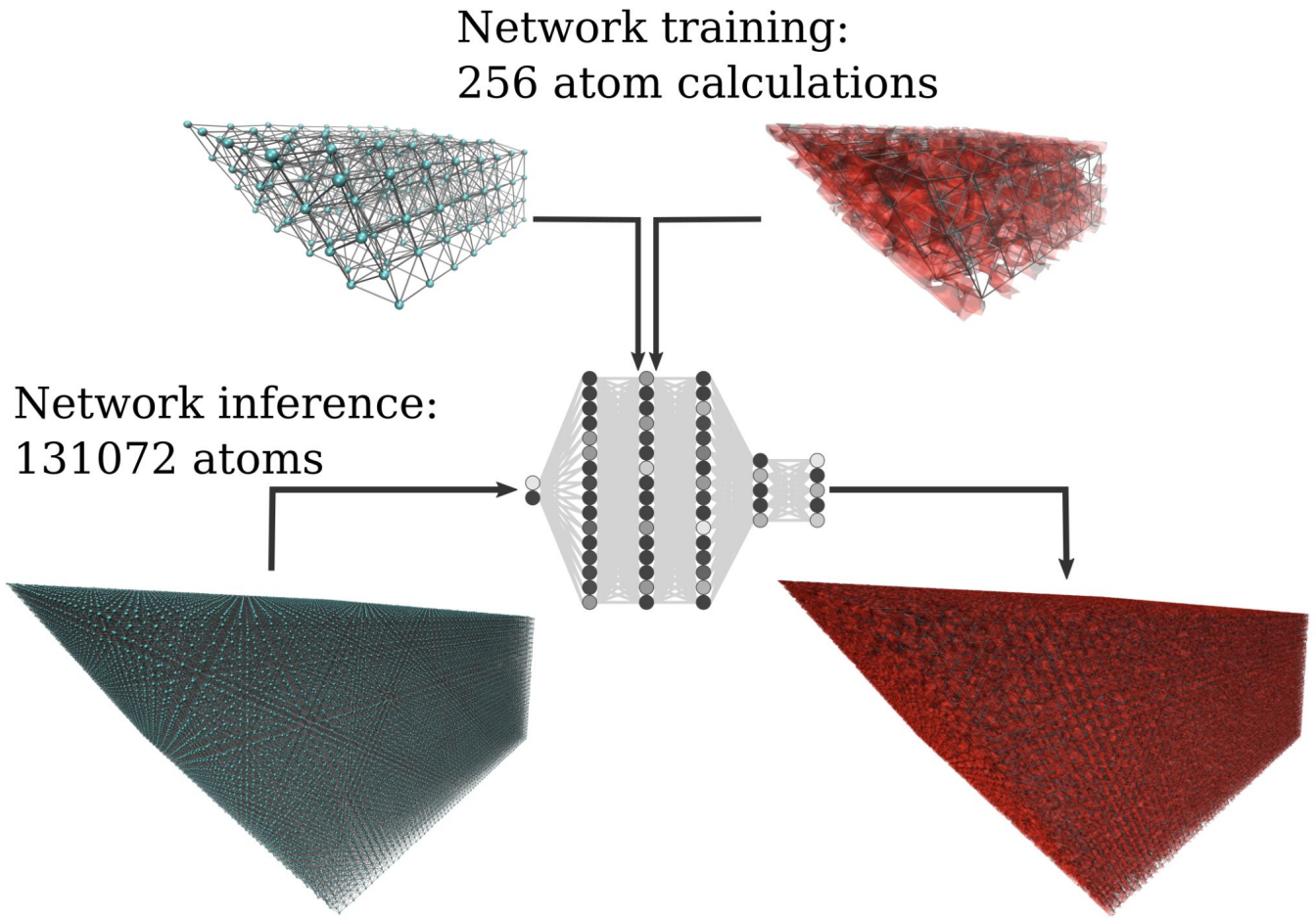


Materials Learning Algorithms

Interpolation across the number of atoms

Materials Learning Algorithms

Interpolation across the number of atoms



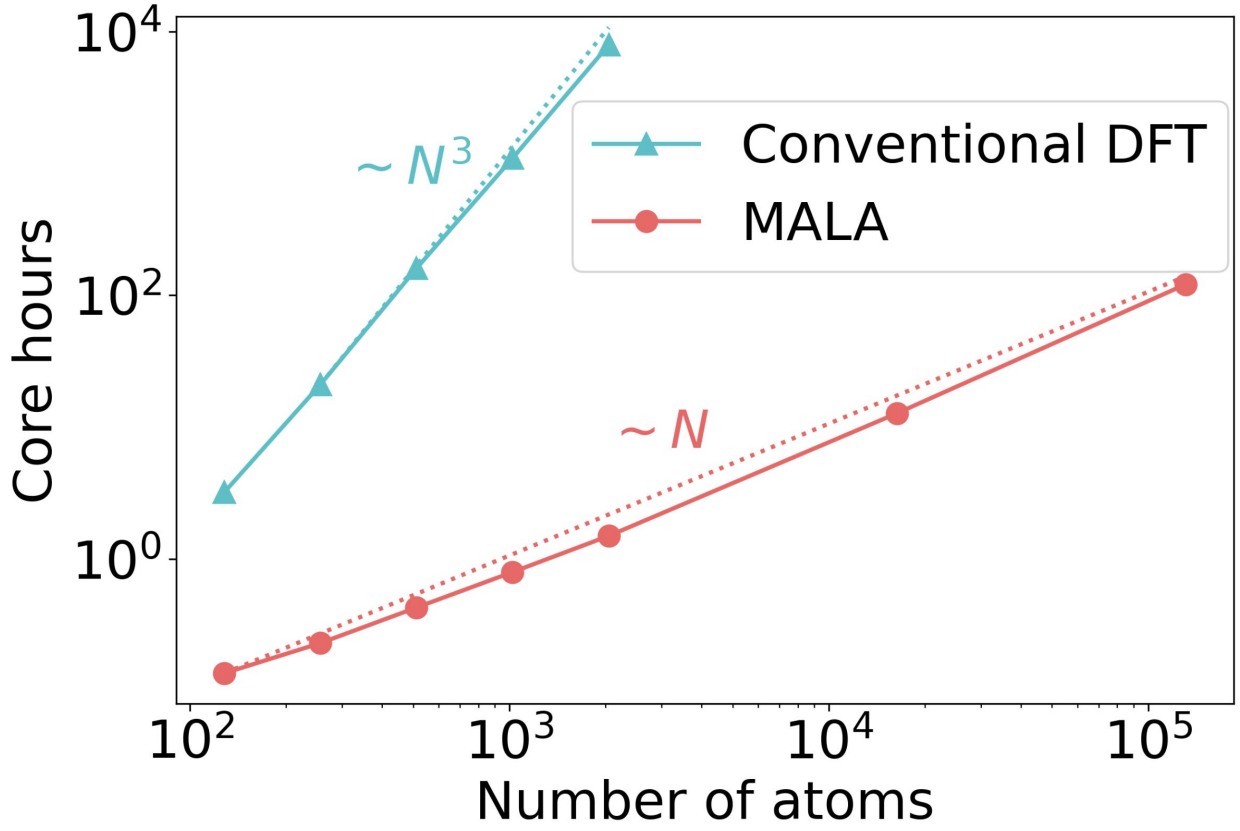
Materials Learning Algorithms

Interpolation across the number of atoms

Computational cost

Conventional DFT scales with the cube of the systems size.

We achieve linear scaling and enable large-scale DFT calculations.



Materials Learning Algorithms

Interpolation across the number of atoms

Computational cost

Conventional DFT scales with the cube of the systems size.

We achieve linear scaling and enable large-scale DFT calculations.

Table 2. Complete report of timings for DFT and ML calculations.

Number of atoms	DFT (Quantum ESPRESSO)			ML (MALA)		
	Wall time [h]	No. CPUs	CPU time [h]	Wall time [h]	No. CPUs	CPU time [h]
128	0.033	96	3.2	0.006	24	0.14
256	0.109	192	21.0	0.010	24	0.23
512	0.417	384	160.0	0.018	24	0.43
1024	1.400	768	1075.2	0.018	45	0.80
2048	16.500	480	7920.0	0.033	45	1.50
16,384	-	-	-	0.085	150	12.78
131,072	-	-	-	0.808	150	121.21

CPU time refers to the total number of hours all CPUs employed for a particular calculation, i.e., it is the product of the wall time and the number of cores.

Materials Learning Algorithms

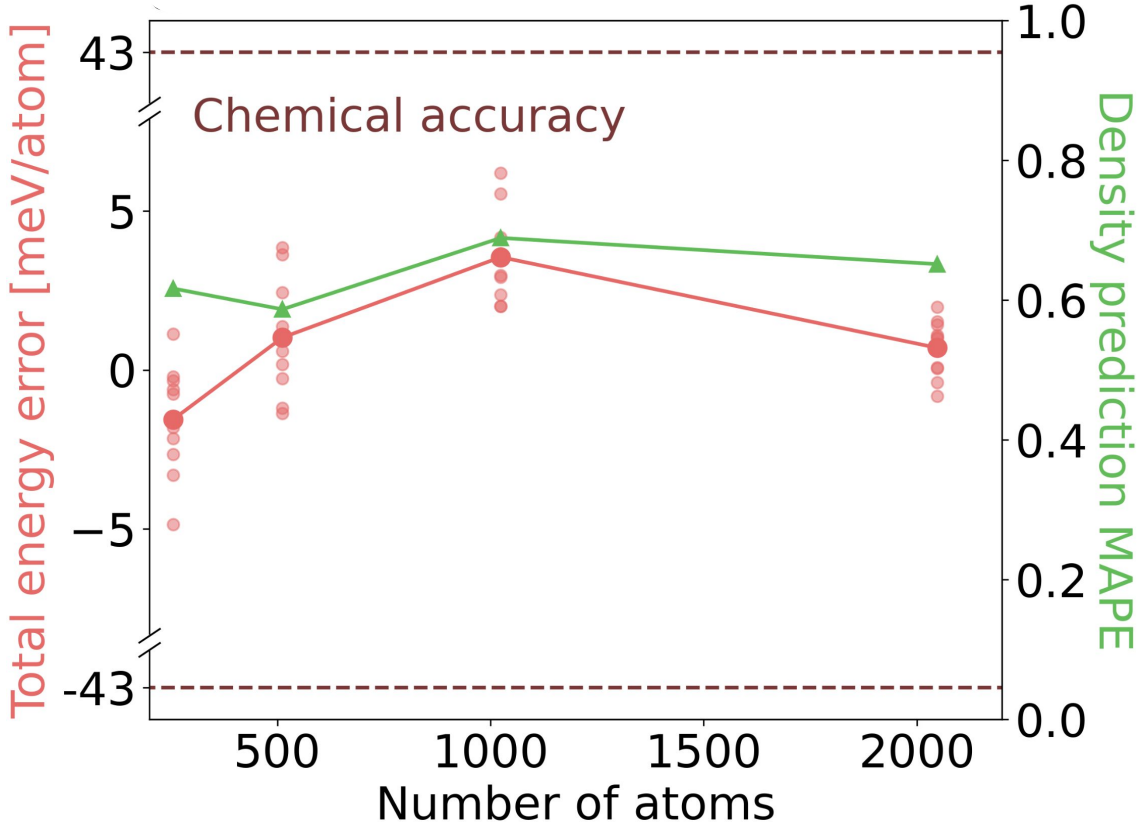
Interpolation across the number of atoms

Accuracy

Benchmarks at DFT scales (up to 2000 atoms).

Total energies are very accurate and errors do not increase with system size.

Also density prediction is highly accurate.



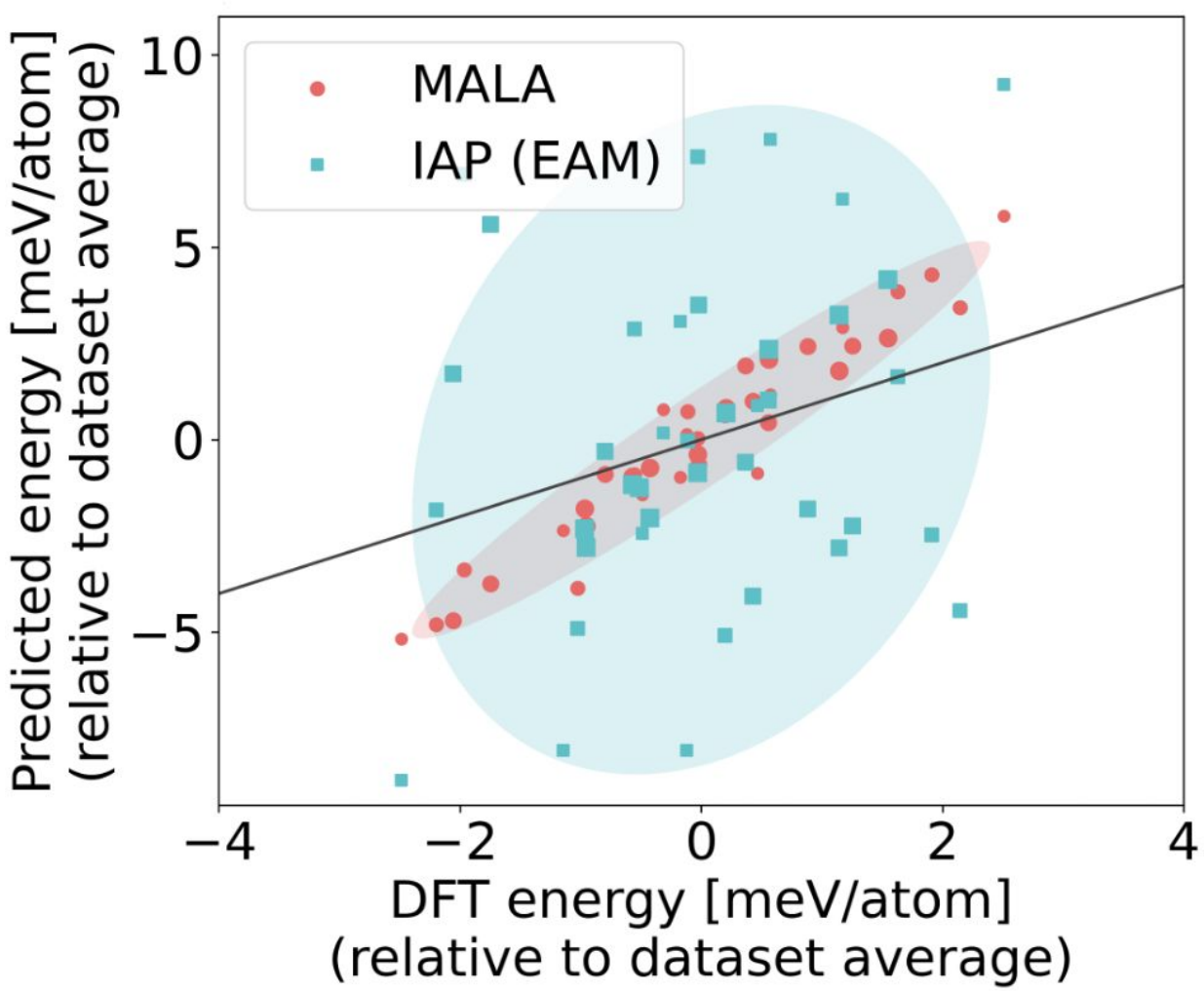
Materials Learning Algorithms

Interpolation across the number of atoms

Accuracy

Benchmarks at DFT scales (up to 2000 atoms).

Compare predicted energies with a conventional interatomic potential (EAM potential).



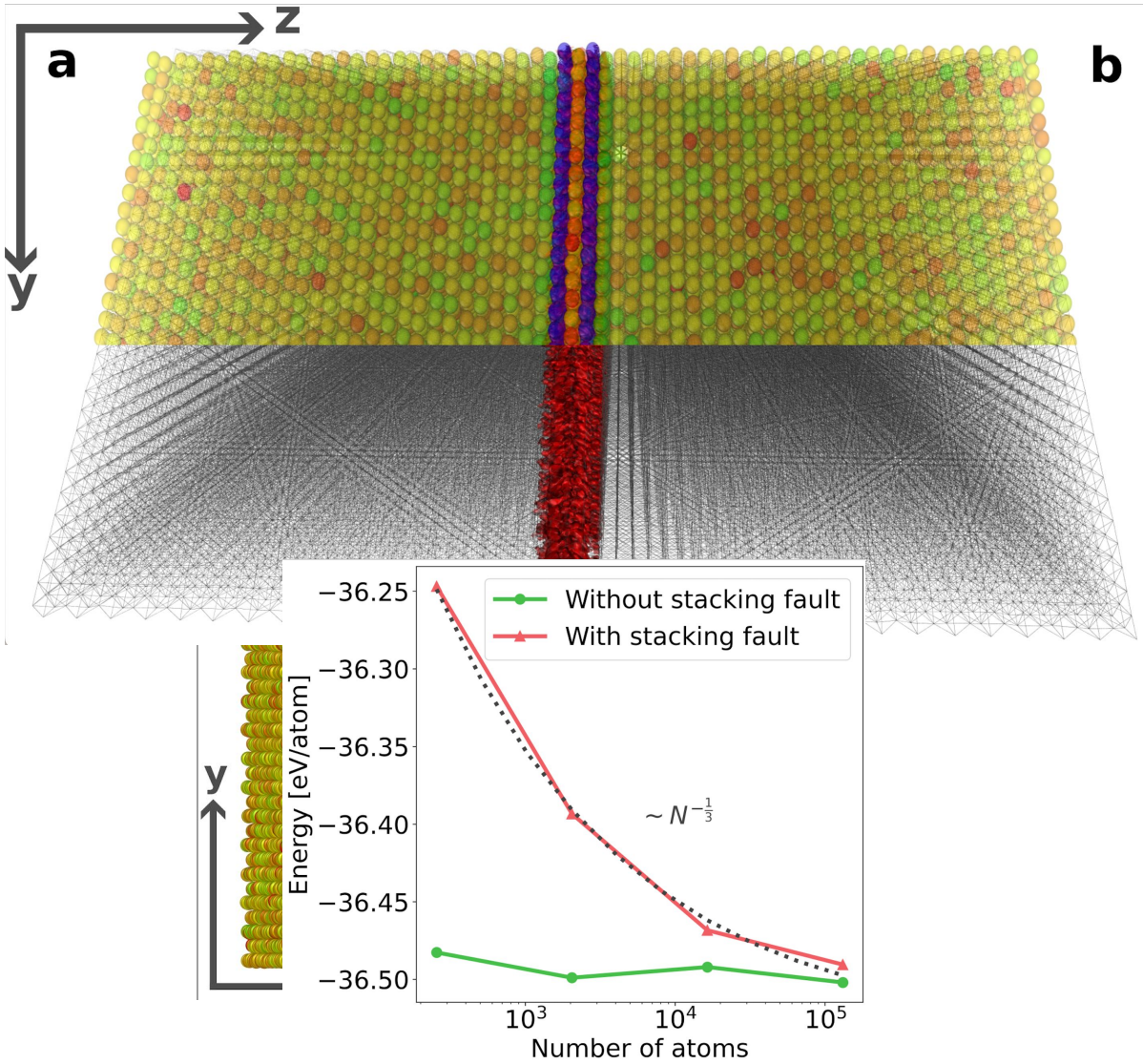
Materials Learning Algorithms

Interpolation across the number of atoms

Stacking fault

Introduce a stacking fault into a slab of Beryllium (change local crystal structure from hcp to fcc).

Predict the electronic structure for this system that contains 131,072 atoms.



Outline

Theoretical Background

Electronic Structure Problem

Density Functional Theory

Motivation

Electronic Structures at Scale from Machine Learning

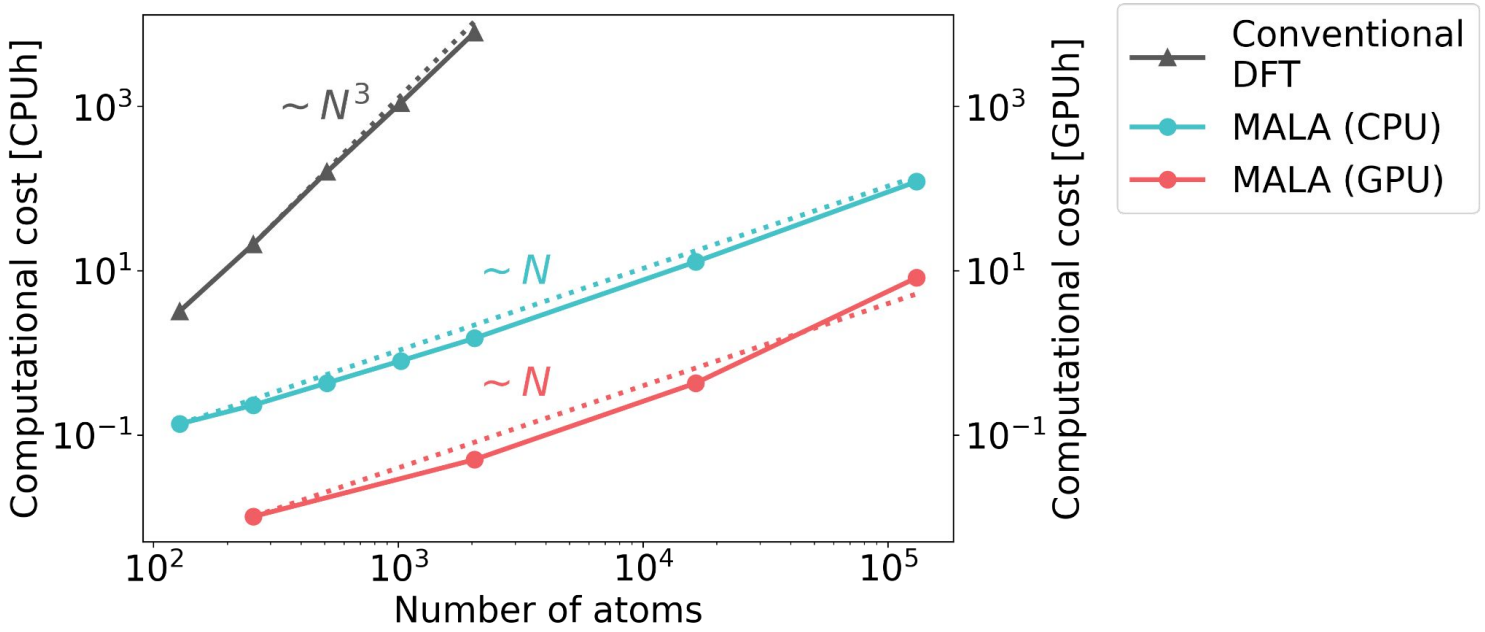
Neural Network Model for Density Functional Theory

Outlook

Summary and Outlook

Development goals

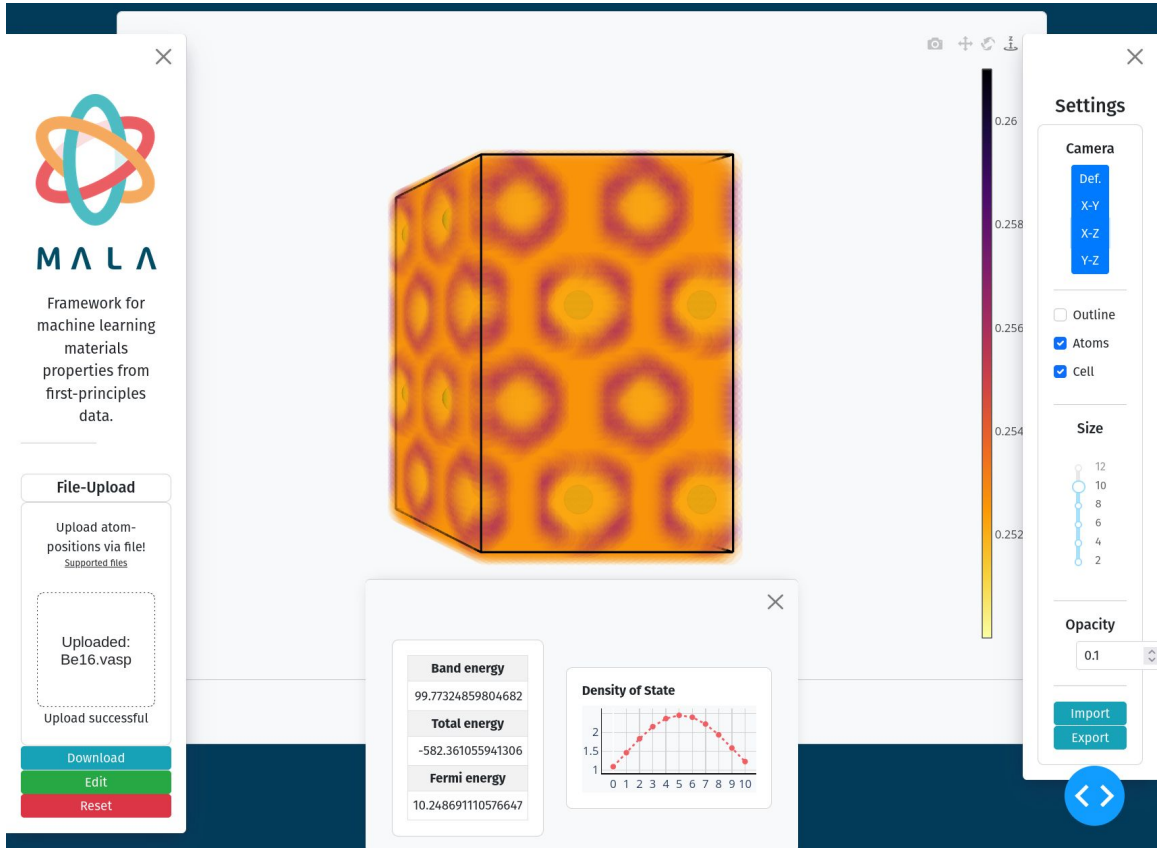
Scalability



Summary and Outlook

Development goals

- Scalability
- Web user interface for deploying models



Summary and Outlook

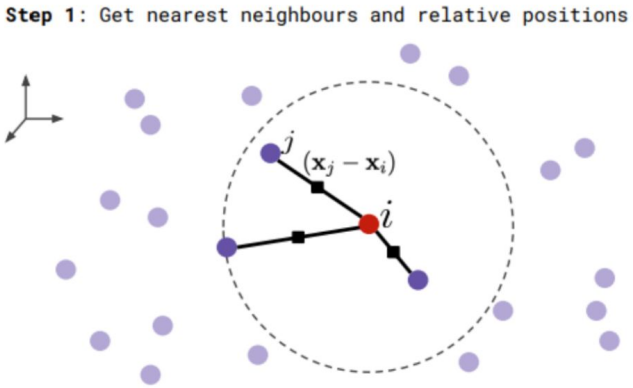
Development goals

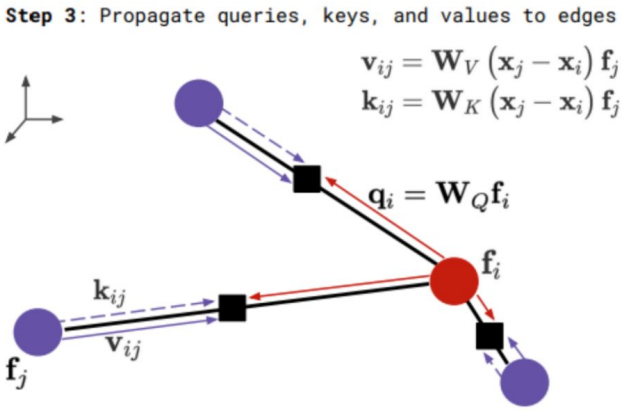
Scalability

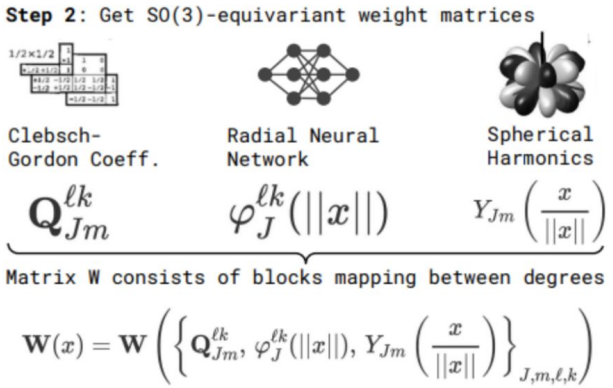
Web user interface for deploying models

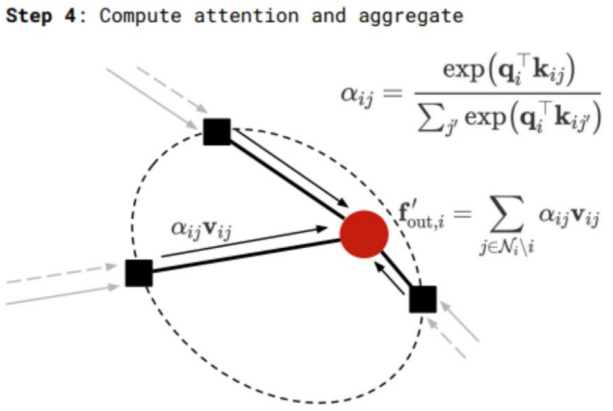
Graph neural networks

- SE(3) - Symmetry group of 3D rotations and translations
- Equivariant - The vector output changes with the same group transformation as the input
- Graph - Works on point-clouds, is entirely grid-free
- Attentional - Can selectively focus on relevant pieces of information
- Neural Networks - Go fast on a GPU









Fuchs et al., arXiv:2006.10503 (2020).

Summary and Outlook

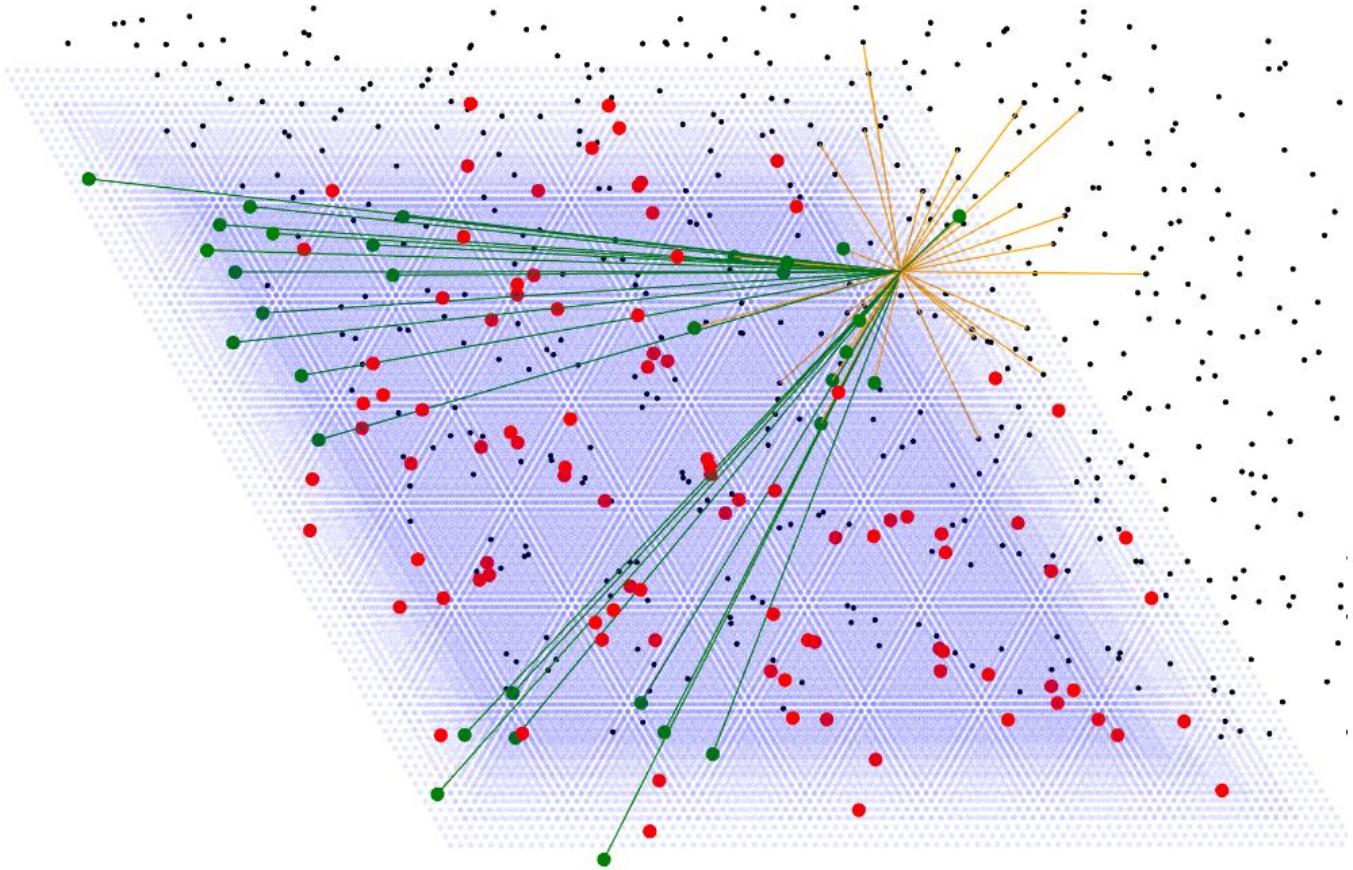
Development goals

Scalability

Web user interface for deploying models

Graph neural networks

- SE(3) - Symmetry group of 3D rotations and translations
- Equivariant - The vector output changes with the same group transformation as the input
- Graph - Works on point-clouds, is entirely grid-free
- Attentional - Can selectively focus on relevant pieces of information
- Neural Networks - Go fast on a GPU



Summary and Outlook

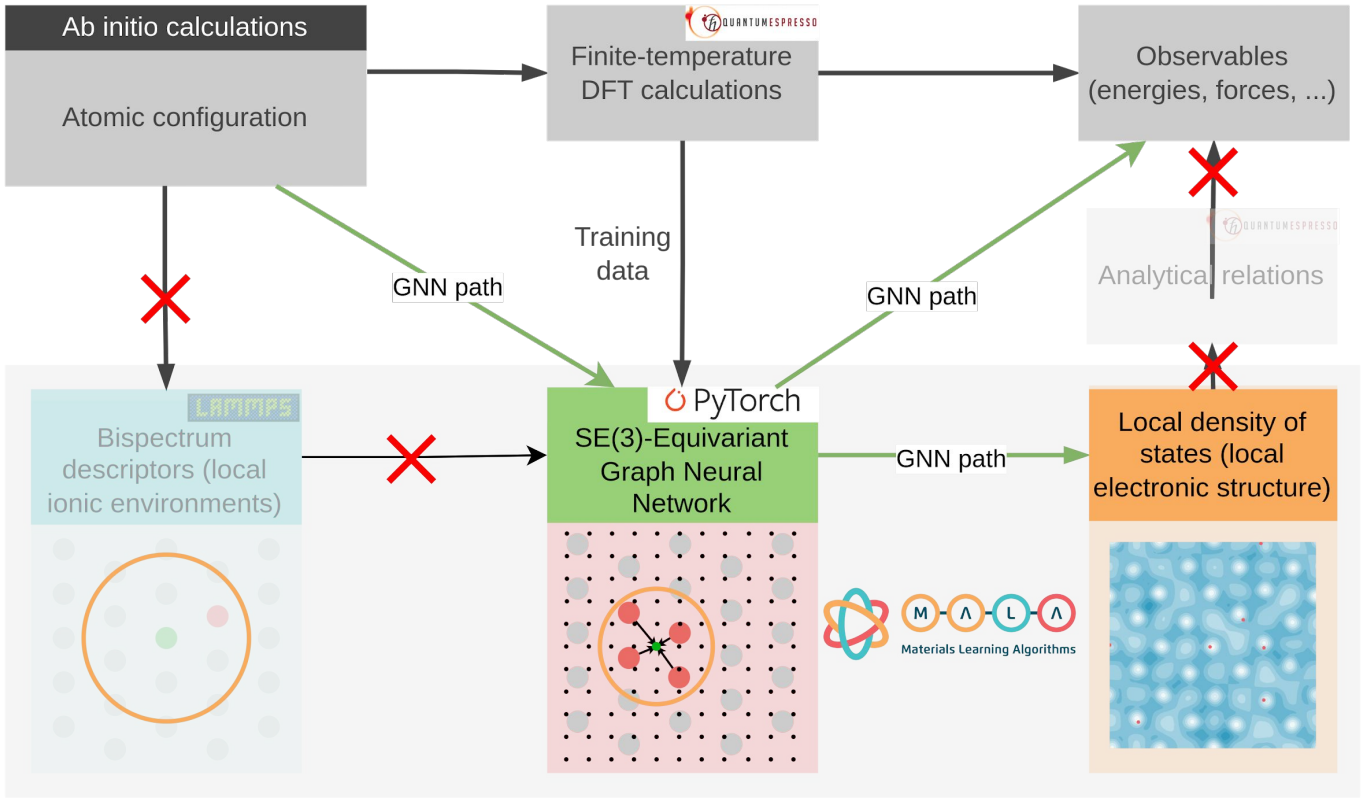
Development goals

Scalability

Web user interface for deploying models

Graph neural networks

- SE(3) - Symmetry group of 3D rotations and translations
- Equivariant - The vector output changes with the same group transformation as the input
- Graph - Works on point-clouds, is entirely grid-free
- Attentional - Can selectively focus on relevant pieces of information
- Neural Networks - Go fast on a GPU

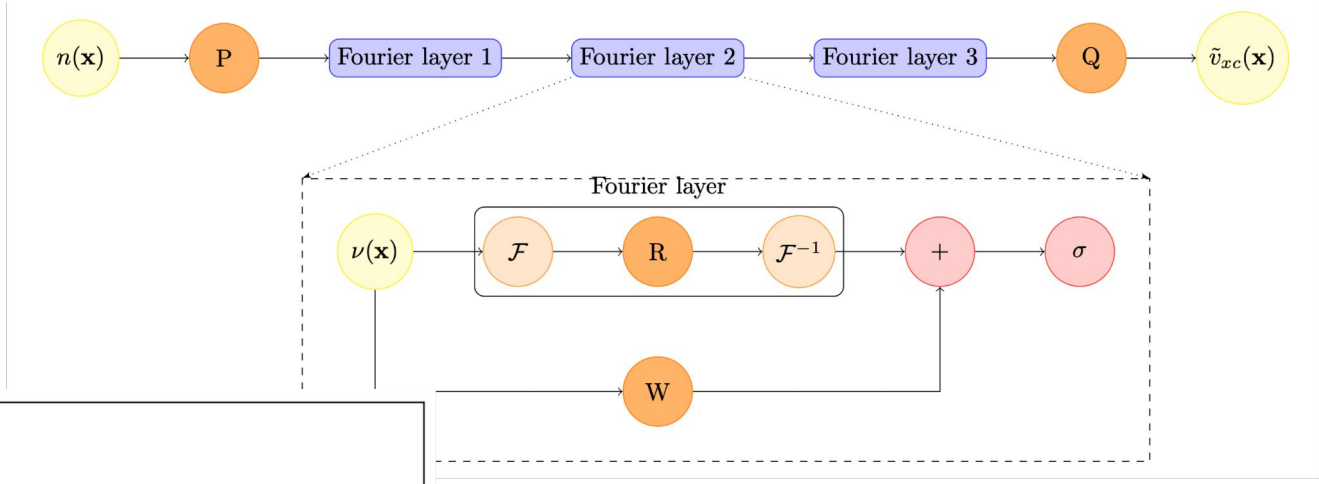


Summary and Outlook

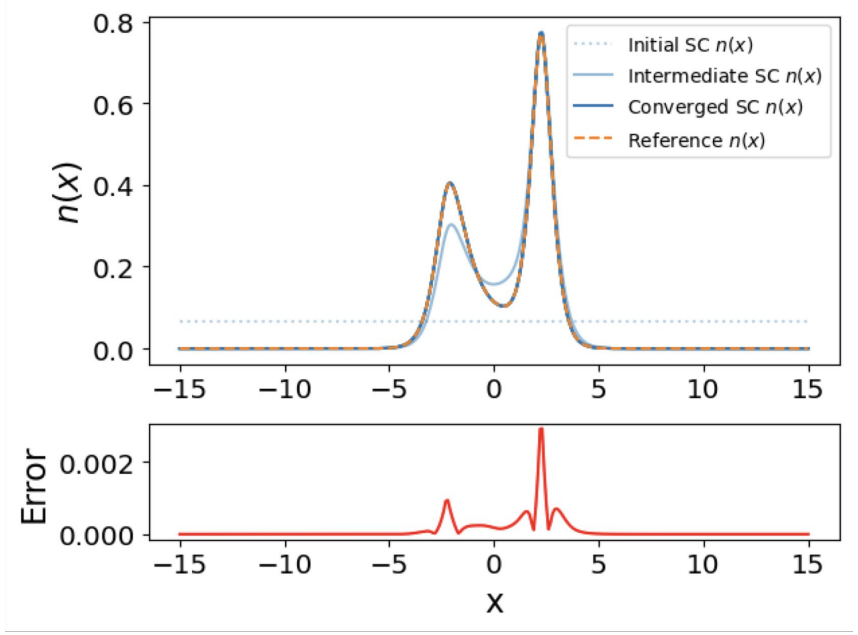
Development goals

- Scalability
- Web user interface for deploying models
- Graph neural networks
- Physics-informed machine learning
  - Inverting the KS equations

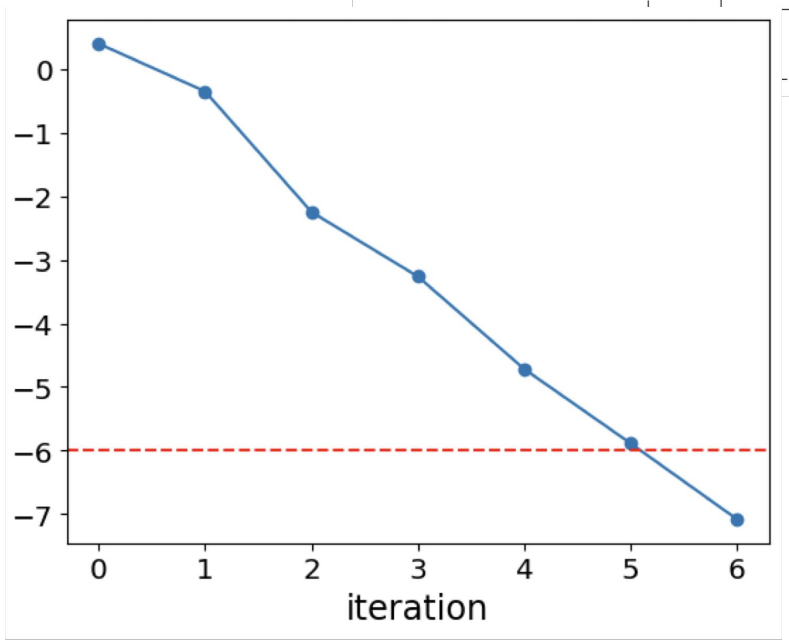
Fourier neural operators



FNO Architecture



Forward calculation using FNO v_{xc}



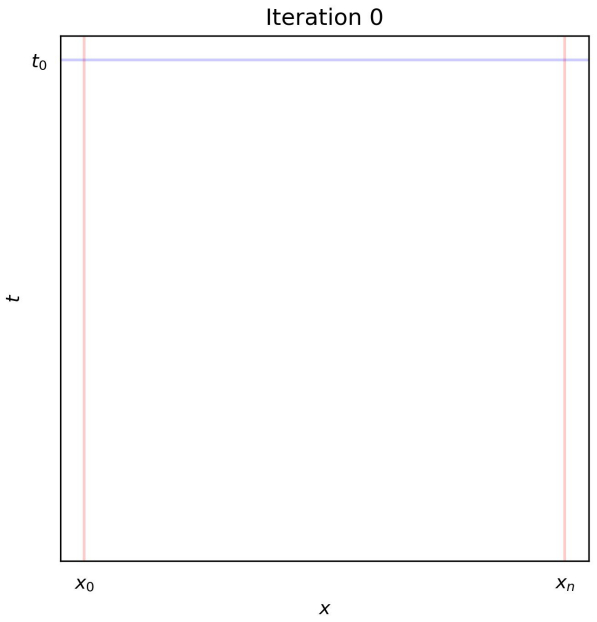
Norm of density through iterations

V. Martinetto, K. Shah, A. Cangi, A. Pribram-Jones, Mach. Learn.: Sci. Technol. 5 015050 (2024).

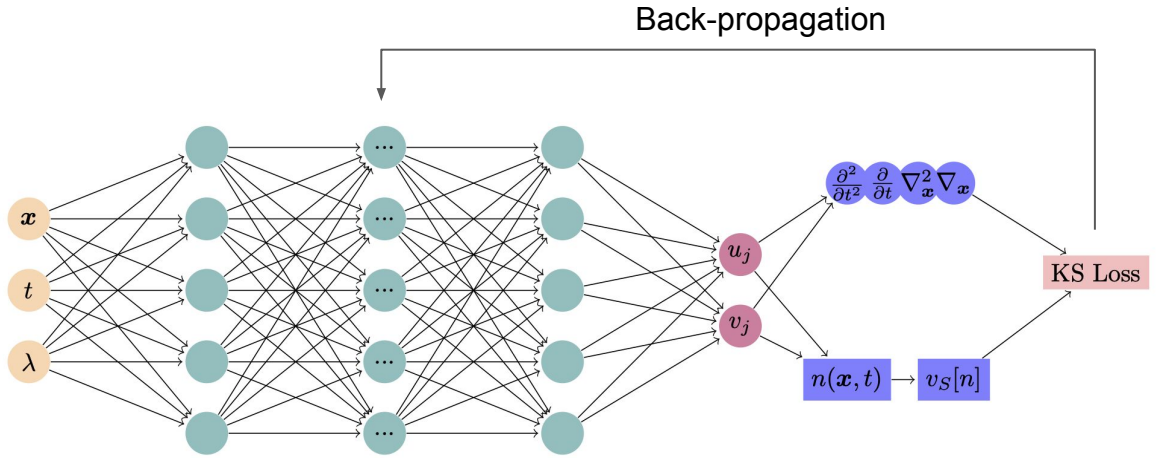
Summary and Outlook

Development goals

- Scalability
- Web user interface for deploying models
- Graph neural networks
- Physics-informed machine learning
  - Machine learning electron dynamics



Physics-informed neural networks



$$i\frac{\partial}{\partial t}\phi_i(\mathbf{r},t) = \left[-\frac{1}{2}\nabla^2 + v_S(\mathbf{r},t)\right]\phi_i(\mathbf{r},t)$$

$$n(\mathbf{r},t) = \sum_i \phi_i^*(\mathbf{r},t)\phi_i(\mathbf{r},t)$$

K. Shah, P. Stiller, N. Hoffmann, A. Cangi, NeurIPS Workshop Machine Learning and the Physical Sciences (2022).

Summary and Outlook

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Development goals

- **Scalability**
- **Web user interface for deploying models**
- **Graph neural networks**
- **Physics-informed machine learning**
- Multi-modal output
- Foundation models for electronic structures

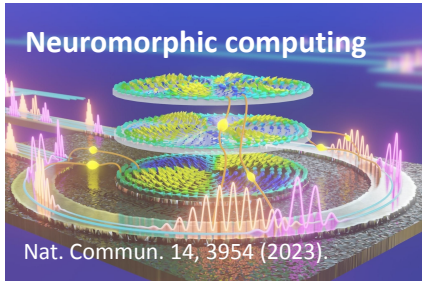
Science goals

Transferability for other parameter spaces

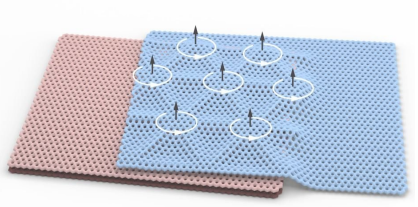
- Mass density, electric fields, electronic spin, heterogeneous and complex materials

Dynamics

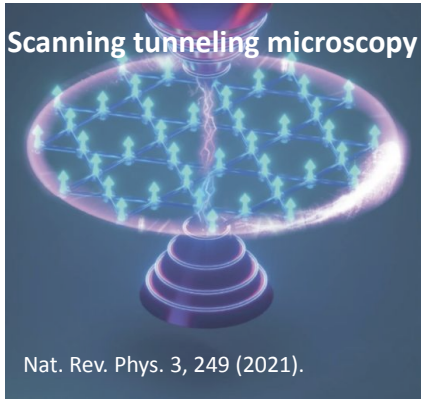
- Machine-learning driven first-principles molecular dynamics with access to the electronic structure



Twisted Moiré layers



Nat. Phys. 17, 374 (2021).



Scalable simulations of challenging systems, such as semiconductor devices

Thanks for your attention

Thanks to my team and collaborators!

Attila Cangi, Tom Jungnickel, Bartosz Brzoza
Karan Shah, Lenz Fiedler, Timothy Callow
Kushal Ramakrishna, Uwe Hernandez-Acosta, Hossein Tahmasbi

Mani Lokamani, Steve Schmerler, Peter Steinbach, Guido Juckeland

Daniel Kotik, Jiri Vyskocil, Franz Pöschl

Normand A. Modine, Dayton J. Vogel, Kyle D. Miller, Aidan P. Thompson, Sivasankaran Rajamanickam

CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

TECHNISCHE UNIVERSITÄT DRESDEN

CBG Max Planck Institute of Molecular Cell Biology and Genetics

UFZ HELMHOLTZ Centre for Environmental Research

Uniwersytet Wrocławski

SPONSORED BY THE

Federal Ministry of Education and Research

STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS

Freistaat SACHSEN