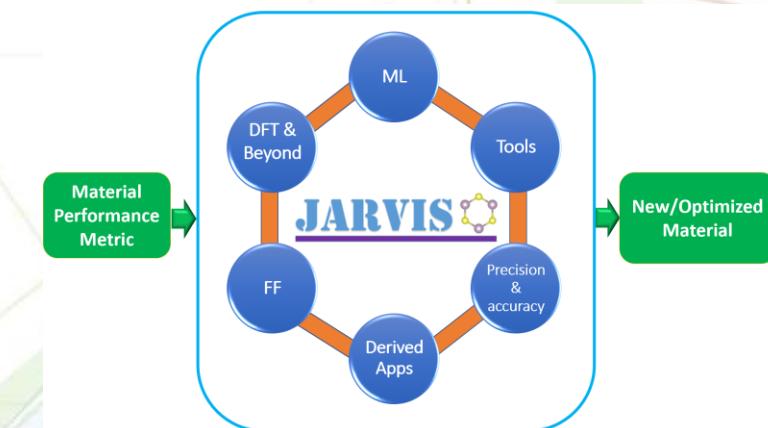


Materials Informatics: Fingerprints, Graph Neural Networks, and Transformers Based Models for Materials Design



Estd. 1901

Kamal Choudhary
Staff Scientist
NIST, Gaithersburg, MD, USA
Jan 20-25, 2025



<https://jarvis.nist.gov>

Outline

Day 1 Materials Science Basics

Modeling Materials & its applications

Programming

Quantum mechanics

Classical mechanics

MSE experiments

Day 2 AI/ML Basics

Types of ML & Models

Workflow

Databases

Classical ML (CFID, MatMiner,...)

Day 3 Deep Learning Models

CNN (AtomVision)

GNN (ALIGNN, ALIGNN-FF)

GPT (AtomGPT, DiffractGPT)

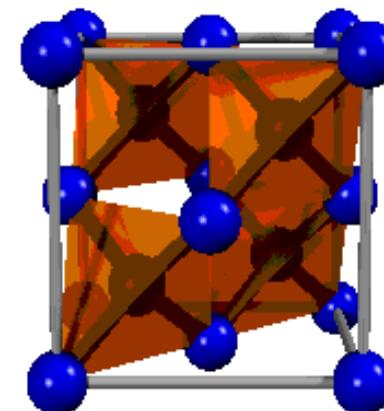
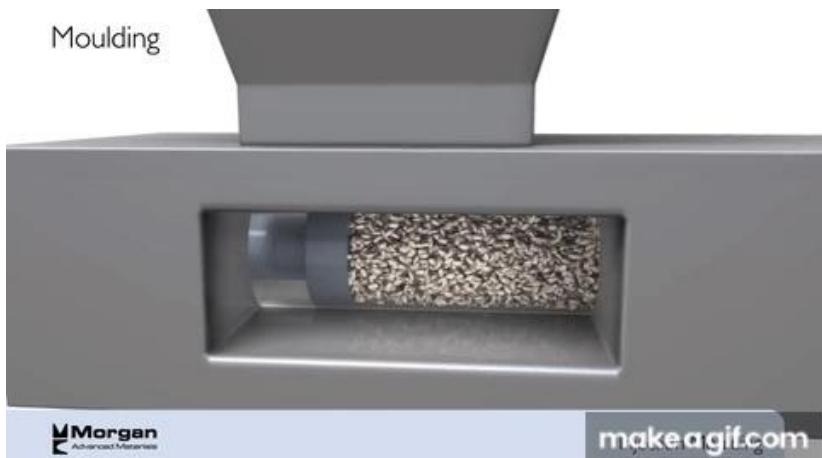
Benchmarking and reproducibility (JARVIS-Leaderboard)

Day 1 Schedule

- Fundamentals of Materials Science, Modeling, Programming
- Quantum Mechanical Calculations
- Classical Calculations

Materials Science Basics: PSPP

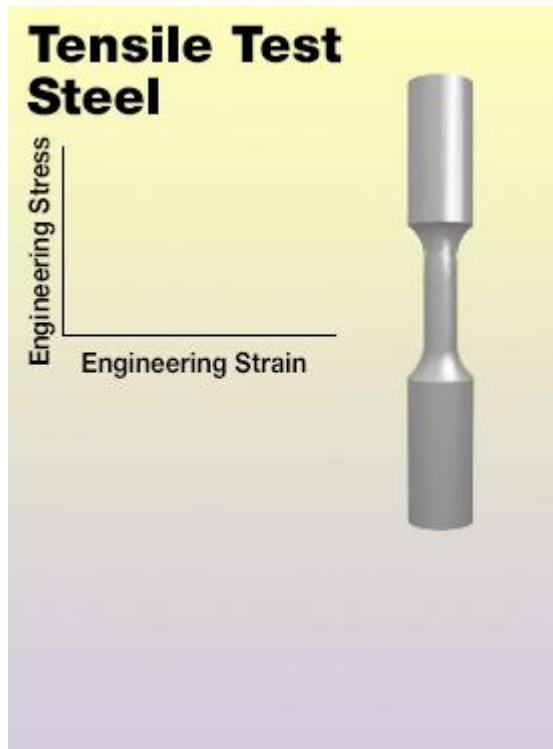
- **Material Definition:** A material is a substance or mixture of substances that forms an object. It can be pure or impure and include both living or non-living matter.
- **PSPP Relationship:** Fundamental concept in materials science linking processing, structure, properties, and performance.
- **Processing:** Methods and conditions under which a material is produced.
 - **Example:** Heat treatments like annealing and quenching alter mechanical properties such as toughness and ductility.
- **Structure:** Arrangement of atoms and molecules in a material, from nanostructures to macrostructures.
 - **Example:** Grain size in metals controlled through rolling or annealing, affecting strength via the Hall-Petch effect.



Diamond crystal structure
- Wikimedia

Materials Science Basics: PSPP

- **Properties:** Measurable material behaviors including mechanical, electrical, and thermal properties.
 - **Example:** strength, ductility, and elasticity of steel
- **Performance:** How a material behaves under real-world conditions.
 - **Example:** Tensile test results for engineers to predict the steel's performance under load



<https://www.mtu.edu/materials/k12/experiments/tensile/>

Materials Science

Elon Musk  
@elonmusk

Take Materials Science 101. You won't regret it.

7:52 PM · Sep 9, 2022

 6.3K  10K  128K  3.3K 

 **Read 6.3K replies**

Material Classes

1. Metals

- aluminum
- copper
- steel (iron alloy)
- nickel
- titanium



2. Ceramics

- clay
- silica glass
- alumina
- quartz



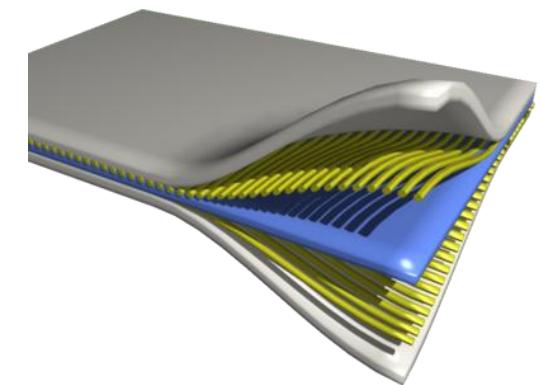
3. Polymers

- polyvinyl chloride (PVC)
- Teflon
- various plastics
- glue (adhesives)
- Kevlar



4. Composites

- wood
- carbon fiber resins
- concrete



- Solids/Liquids/Gases
- Crystalline/Amorphous

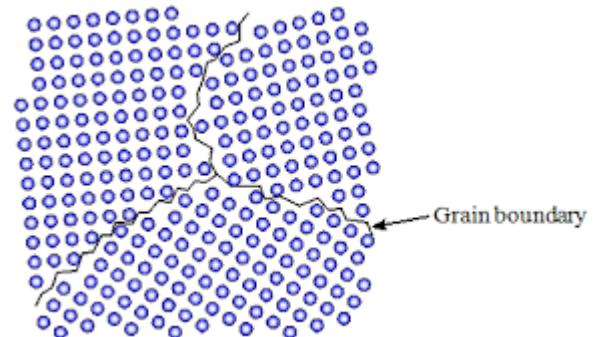
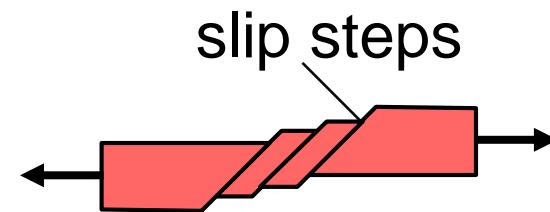
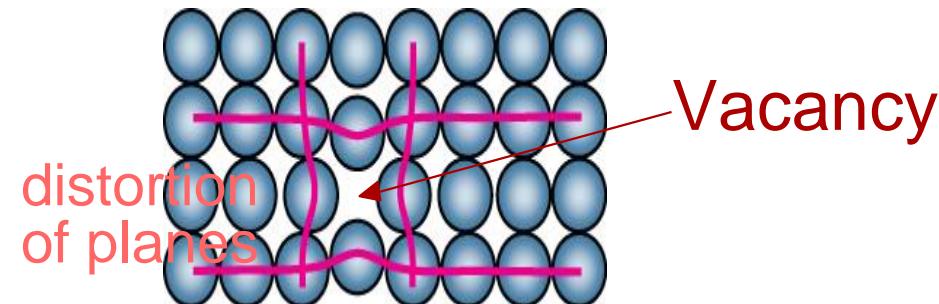
Perfect vs Defects Materials

- Vacancy atoms
- Interstitial atoms
- Substitutional atoms
- Dislocations
- Surfaces, Grain Boundaries

Point defects

Line defects

Area defects

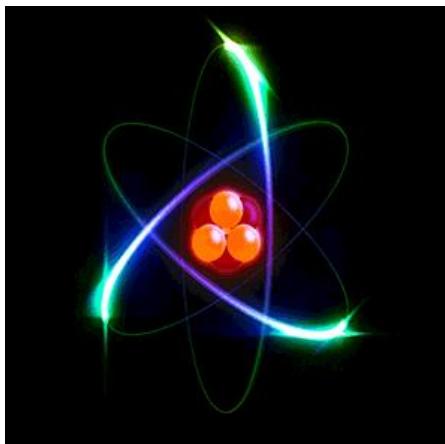
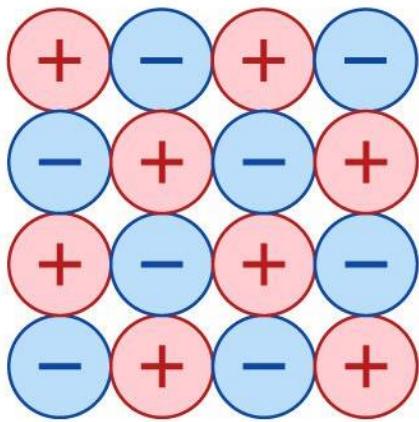


Crystal Systems

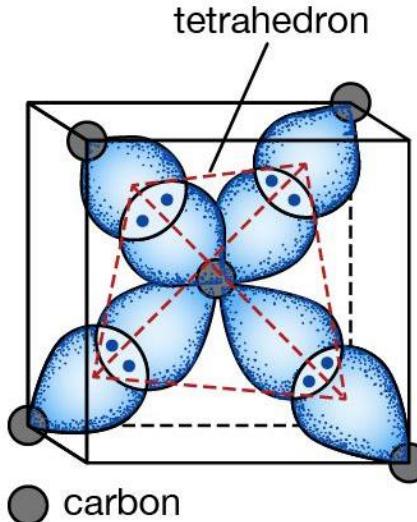
Crystal System	Bravais Lattice Types	Unit Cell Characteristics	Examples
Cubic	Simple (P), Body-Centered (I), Face-Centered (F)	$a = b = c, \alpha = \beta = \gamma = 90^\circ$	NaCl, CsCl, Diamond
Tetragonal	Simple (P), Body-Centered (I)	$a = b \neq c, \alpha = \beta = \gamma = 90^\circ$	SnO_2 (Rutile), In, TiO_2
Orthorhombic	Simple (P), Body-Centered (I), Face-Centered (F), Base-Centered (C)	$a \neq b \neq c, \alpha = \beta = \gamma = 90^\circ$	Olivine, BaSO_4 , KNO_3
Hexagonal	Simple (P)	$a = b \neq c, \alpha = \beta = 90^\circ, \gamma = 120^\circ$	Graphite, ZnO , Mg
Trigonal	Simple (P)	$a = b = c, \alpha = \beta = \gamma \neq 90^\circ$	Calcite, Quartz (α -quartz)
Monoclinic	Simple (P), Base-Centered (C)	$a \neq b \neq c, \alpha = \gamma = 90^\circ, \beta \neq 90^\circ$	Gypsum, Mica, Clinopyroxene
Triclinic	Simple (P)	$a \neq b \neq c, \alpha \neq \beta \neq \gamma \neq 90^\circ$	Kyanite, $\text{CuSO}_4 \cdot 5\text{H}_2\text{O}$, Turquoise

Chemical Bonding

ionic bond

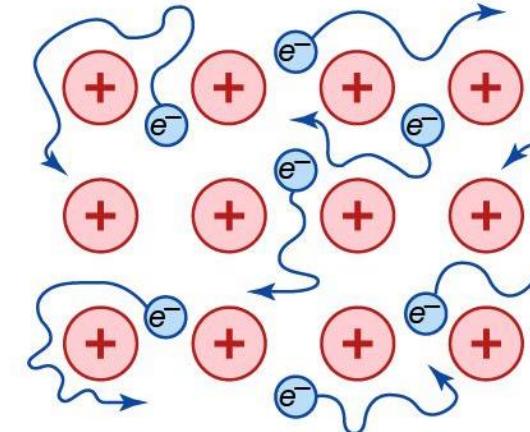


covalent bond



Covalent bonds involve electron sharing, such as between these carbon atoms when they form a diamond.

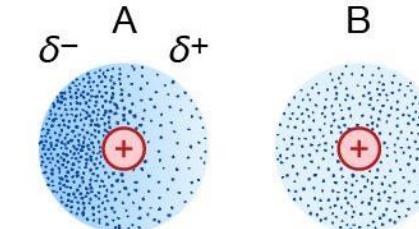
metallic bond



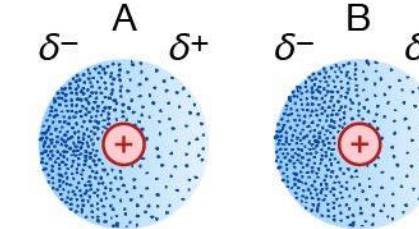
metallic structure, showing possible electron (e^-) paths around the nuclei of metal atoms (represented as spheres with a positive charge)

Metallic bonding can be thought of as a cloud of positively charged ions immersed in a cloud of valence electrons.

van der Waals bond



instantaneous dipole on A induces a dipole on B



weak dipole attraction of van der Waals bond

Neutral molecules may be held together by a weak electric force known as the van der Waals bond.

Chemical Bonding

• Metals: Metallic Bonding

- Delocalized electrons create a "sea" around positive ions.
- **Properties:** Conductivity, malleability, ductility, luster.

• Ceramics: Ionic and Covalent Bonding

- **Ionic:** Electron transfer between metal and non-metal, forming charged ions.
- **Covalent:** Electron sharing, generally between non-metals, leading to directional and strong bonds.
- **Properties:** Hardness, brittleness, high melting point, chemical stability.

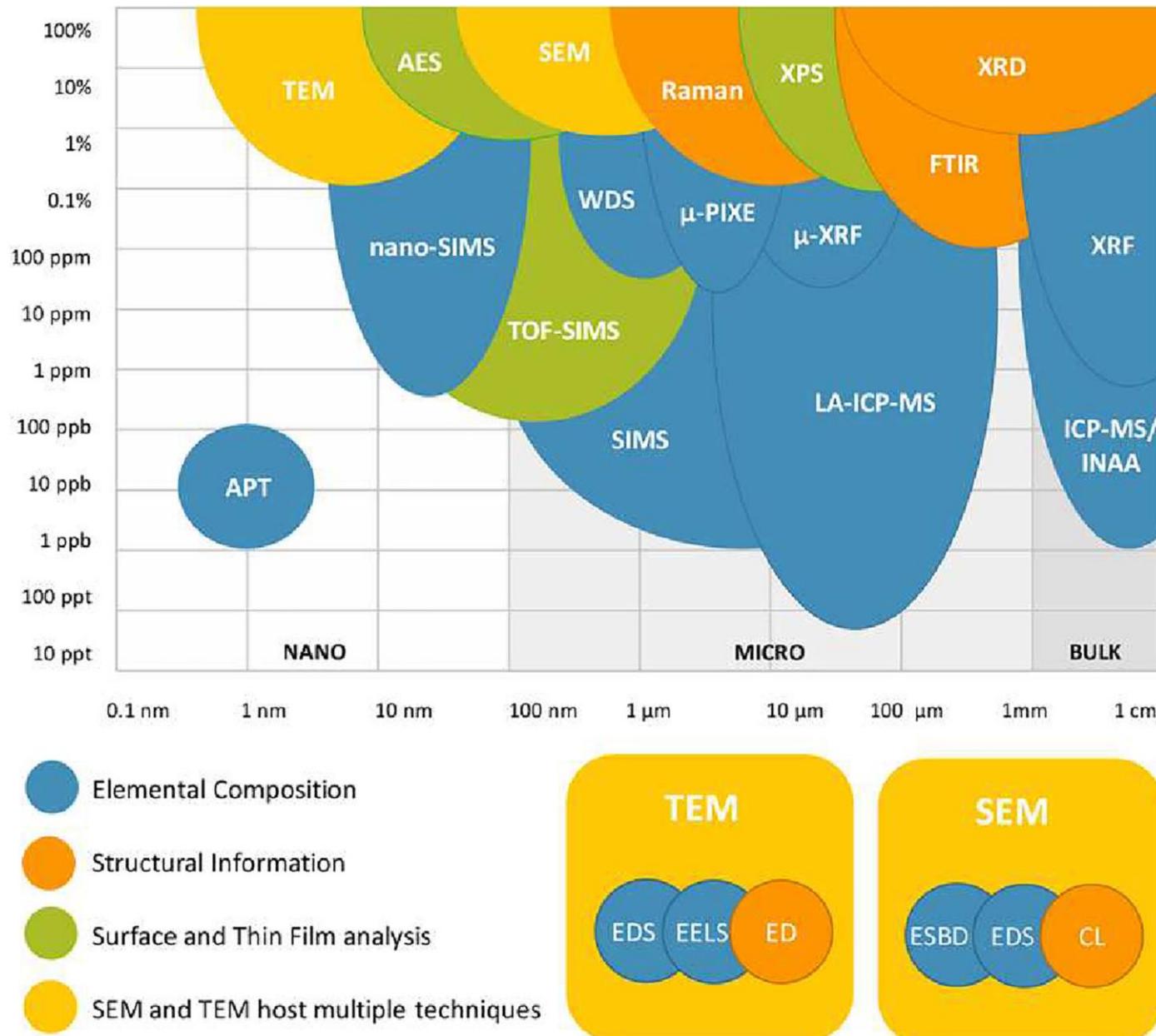
• Polymers: Covalent Bonding

- Long chains of molecules connected by strong covalent bonds.
- **Properties:** Elasticity, plasticity, tensile strength, chemical resistance.

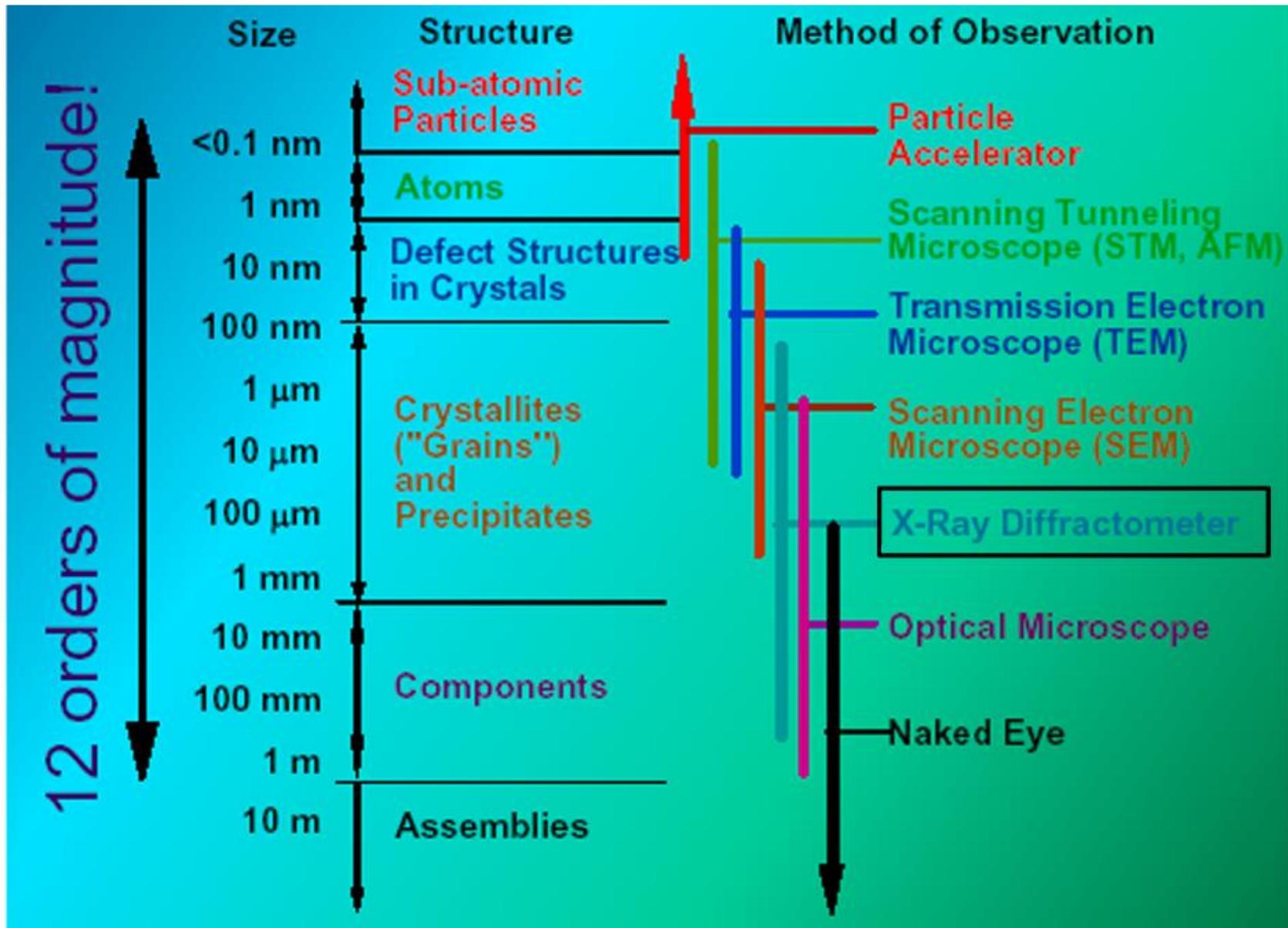
• Composites: Varied Bonding

- Combinations of different materials, each retaining its own bonding characteristics.
- **Properties:** Enhanced mechanical properties, tailored functionalities, improved durability._{r1}

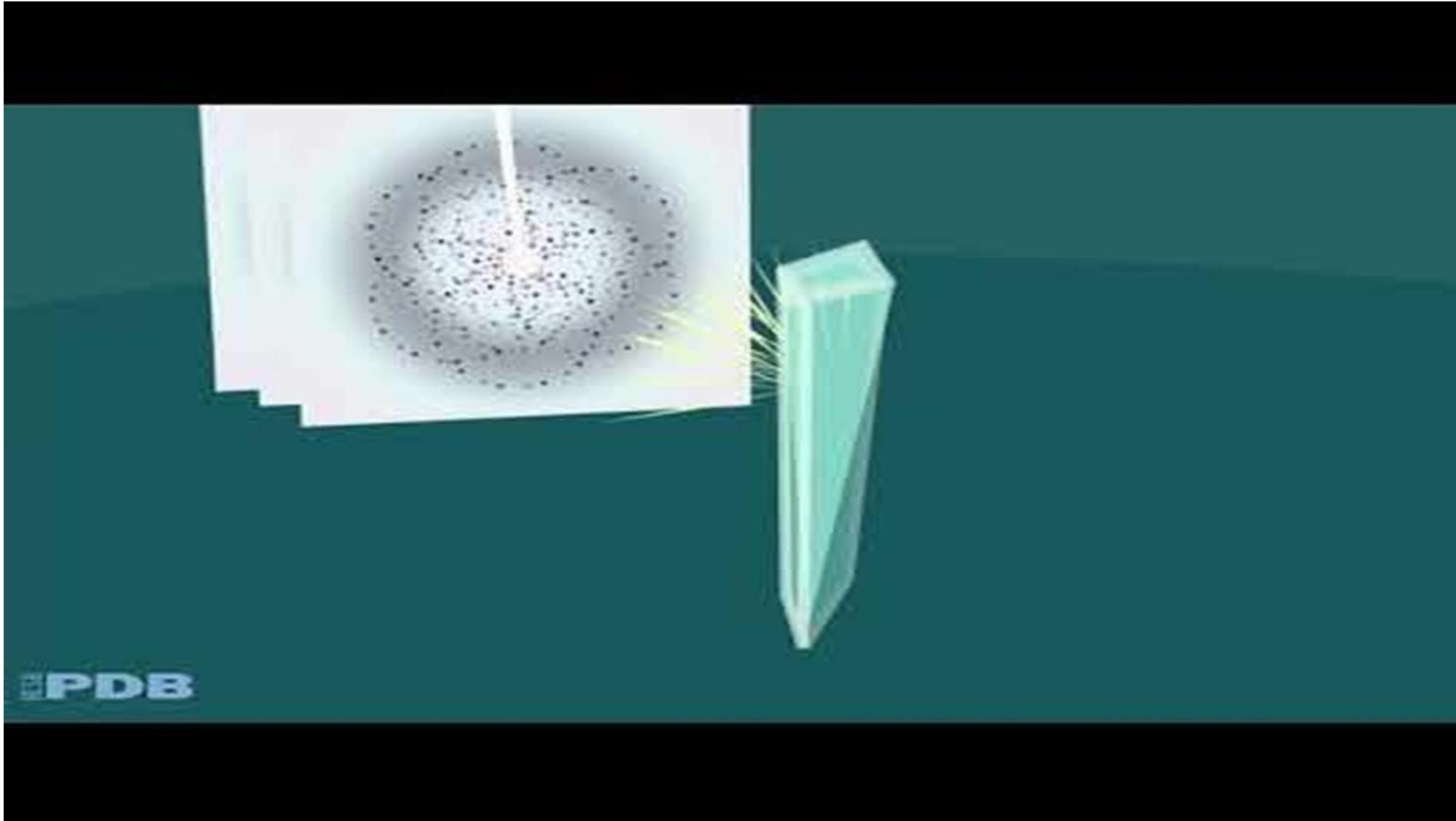
Materials Characterization



Materials Characterization



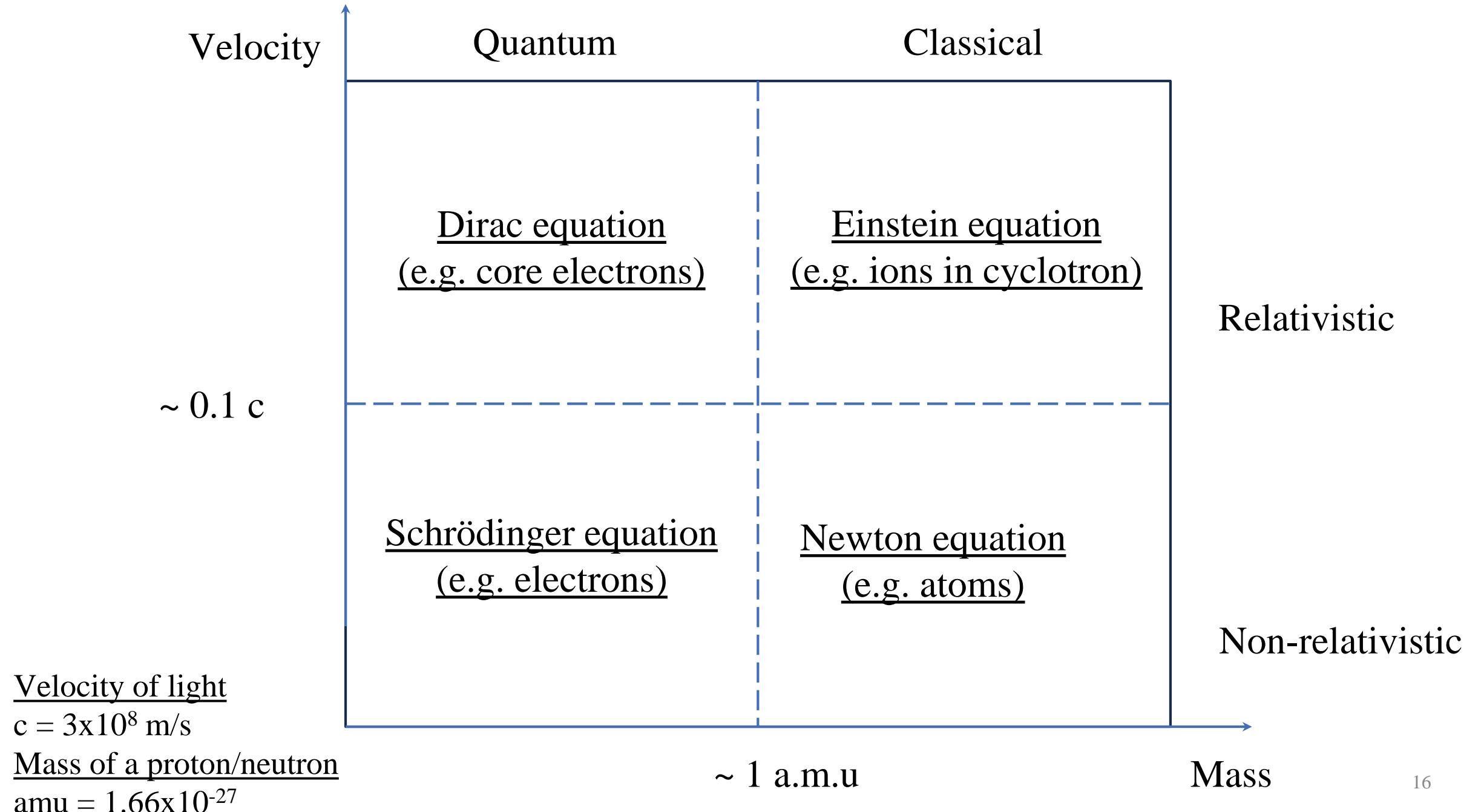
X-ray Crystallography



<https://www.youtube.com/watch?v=RUok4O9oovQ>

What is a Model?

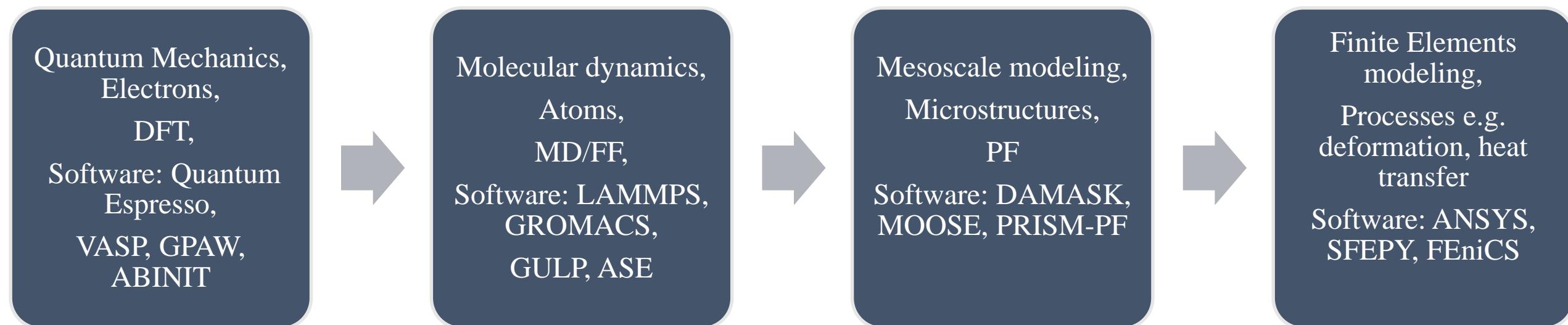
- **Definition of a Model:** A simplified representation of reality to analyze, understand, or predict real-world phenomena.
- **Purpose of Models:**
 - Simulate complex systems or phenomena.
 - Illustrate or reason about specific aspects of the real world.
 - Aid in decision-making in research, development, policy-making, and education.
- **Types of Models:**
 - **Physical Models:** Tangible, scaled-down versions of objects (e.g., globes, architectural models).
 - **Mathematical Models:** Use mathematical expressions to describe system variables (common in physics, economics, engineering).
 - **Statistical Models:** Apply statistical methods to infer and predict data patterns (e.g., linear regression, machine learning algorithms).
 - **Conceptual Models:** Diagrams or flowcharts simplifying complex ideas (used in biology, management).
 - **Computational Models:** Algorithms and simulations for studying complex system behavior (e.g., climate models, atomic simulations).



Multiscale Modeling for Mechanical Systems

Multiscale modeling is a computational approach that

- **integrates** models at different **spatial and temporal** scales to simulate complex systems.
- crucial in mechanical systems, where phenomena at one scale can **significantly influence** other scales.



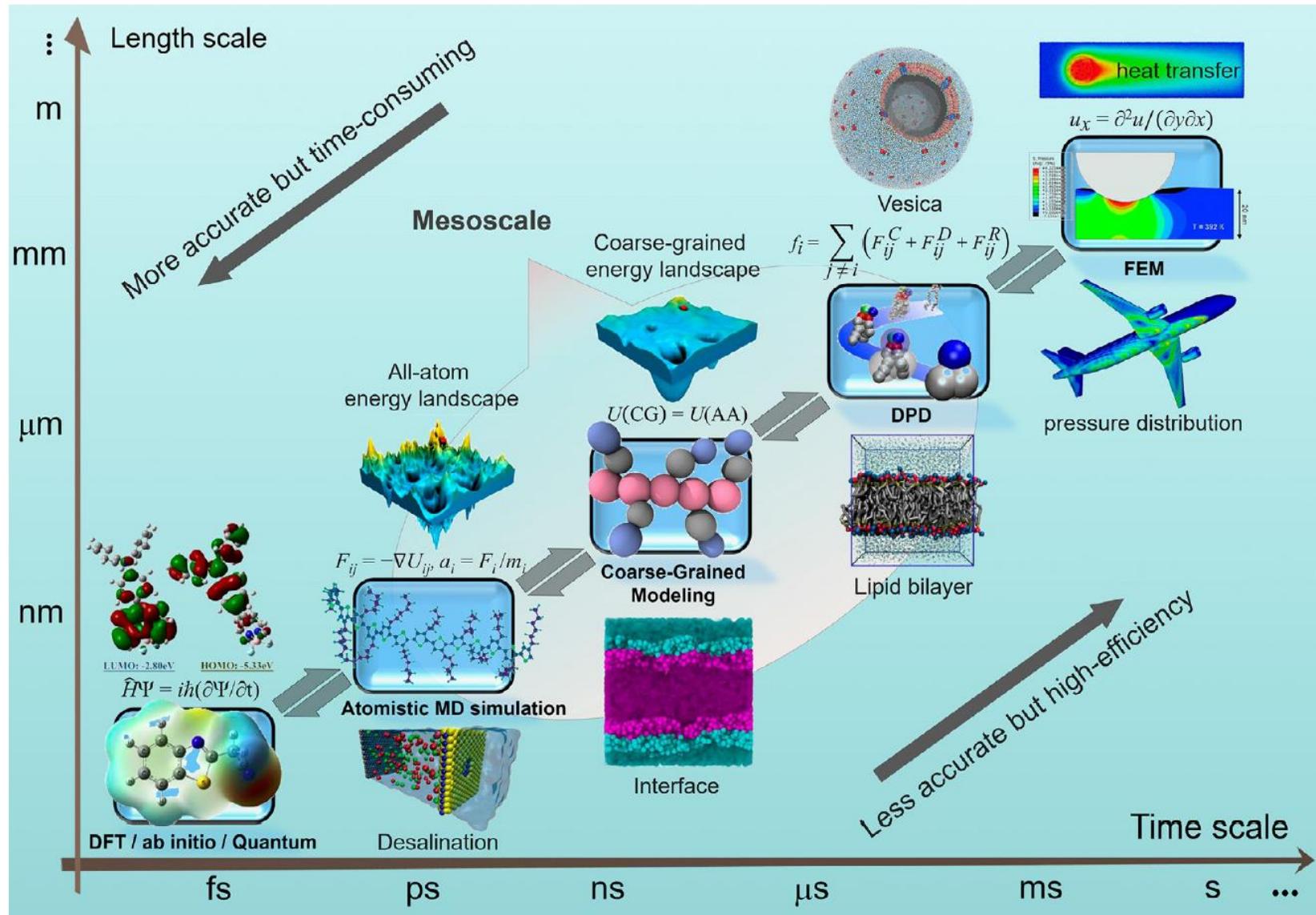
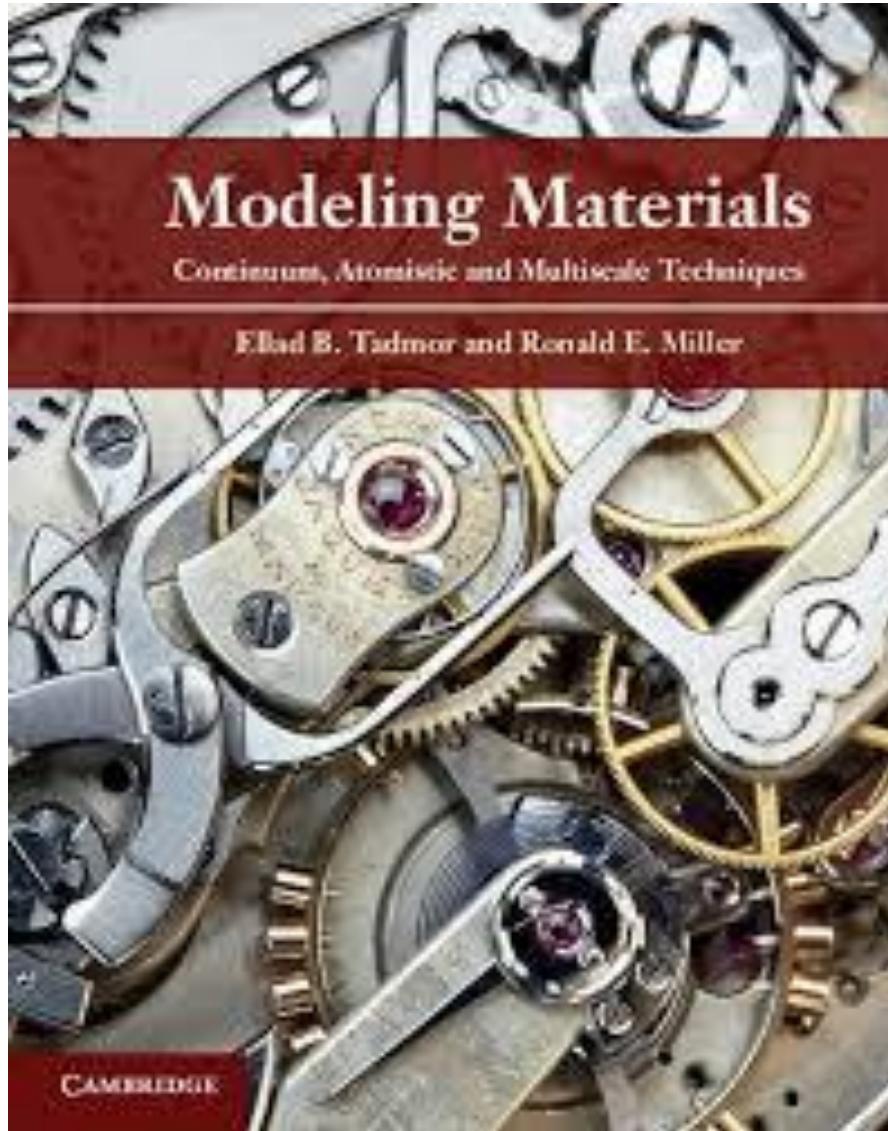


Fig. 1 Schematics of the multiscale modeling techniques across different time and length scales of material systems from quantum, all-atom, coarse-graining, mesoscale, to continuum. The paradigm shows the approximate ranges of time and length scales associated with each modeling technique along with relevant applications.

FUNDAMENTALS OF MULTISCALE MODELING OF STRUCTURAL MATERIALS

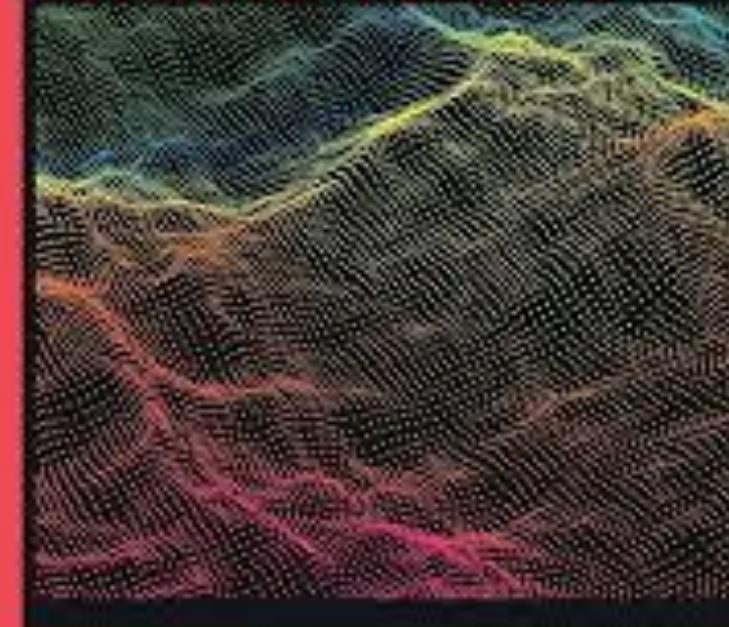


Edited by
Wenjie Xia
Luis Alberto Ruiz Pestana



ELSEVIER SERIES IN MECHANICS OF ADVANCED MATERIALS

UNCERTAINTY QUANTIFICATION IN MULTISCALE MATERIALS MODELING



Edited by
YAN WANG
DAVID L. McDOWELL

<https://www.sciencedirect.com/book/9780128230213/fundamentals-of-multiscale-modeling-of-structural-materials>

<https://www.cambridge.org/core/books/modeling-materials/7CC4027C34755637D8F641A0C8C26835#fndtn-information>

<https://www.sciencedirect.com/book/9780081029411/uncertainty-quantification-in-multiscale-materials-modeling>

Programming and Programming Languages

Programming: Programming is the process of designing and building an executable computer program to accomplish a specific computing result or to perform a particular task.

Programming Languages: A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms.

Characteristics of Programming Languages

1. Syntax and Semantics: Every programming language has a unique set of rules for usage, known as its syntax (such as : after *if* statement in Python). Semantics refer to the meaning of different elements and expressions within the language's syntax (such as run a code when it meets *if* statement).

2. High-level and Low-level Languages: High-level languages (like Python, MATLAB, Java, and C#) are more abstract, and provide a syntax that is closer to human language. Low-level languages (like Assembly or C), in contrast are closer to machine code.

3. Static vs Dynamic Typing: Languages also differ in how they handle data types. In statically typed languages (e.g., C, Java), the type of a variable is known at compile time. In dynamically typed languages (e.g., Python, JavaScript), the type is determined at runtime which adds flexibility but reduces safety and performance.

4. Compiled vs Interpreted: Some languages are compiled, i.e., their programs are directly translated into machine-readable code (like C). Others are interpreted, i.e., their code is read and executed by a runtime program without prior conversion to machine code (like Python).

Programming and Programming Languages



“In fifteen years
we’ll be teaching
programming just
like reading and
writing . . . and
wondering why we
didn’t do it sooner.”

— Mark Zuckerberg

Try an Hour of Code for
Computer Science Education Week
December 9-15.

Anybody can learn!

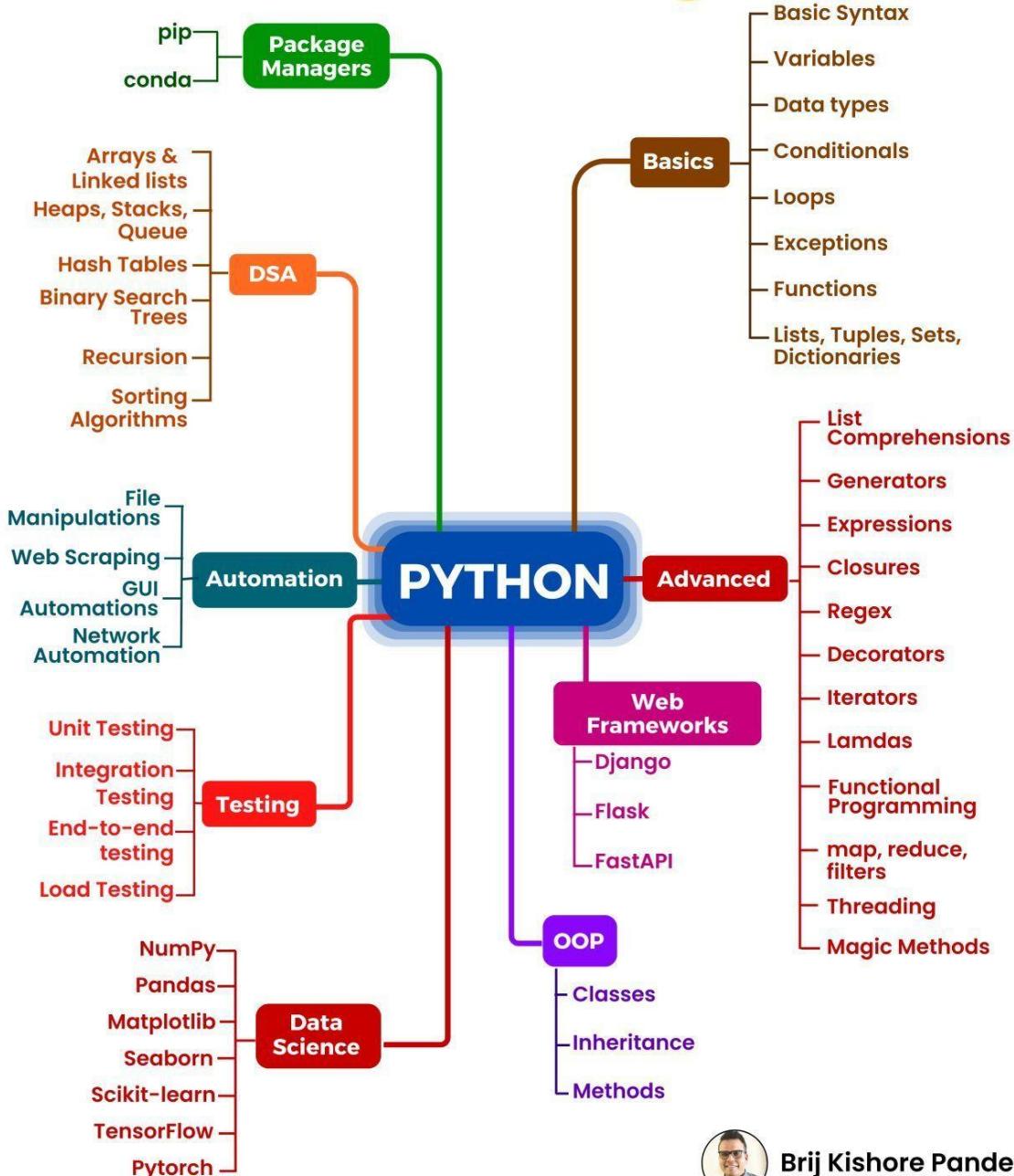
<http://code.org>



Python

PYTHON ROADMAP

DON'T FORGET TO
SAVE FOR LATER ! 



Brij Kishore Pandey

Data Structures

Variables: Variables are used to store individual data values. In a mechanical context, they could represent various physical properties or parameters.

```
# Variable to store the temperature of an engine (in degrees Celsius)
engine_temperature = 150

# Variable to store the speed of a rotating shaft (in RPM)
shaft_speed = 1200

print("Engine Temperature:", engine_temperature, "Celsius")
print("Shaft Speed:", shaft_speed, "RPM")
```

Arrays and Lists: An array is a collection of elements, useful for storing data elements of the same type. Lists are ordered collections of items that can be of different types. Lists are dynamic, meaning their size can change as elements are added or removed.

```
import numpy as np

# Using an array for numerical computation:
force_magnitudes = np.array([10, 20, 30, 40, 50]) # uniform data type (integers)
resultant_force = np.sum(force_magnitudes) # easy to compute directly
print("Resultant Force:", resultant_force)

# Using a list for general data collection:
material_properties = ['steel', 7850, 210e9, {'ductility': 'high'}] # mixed data types
print("Material Properties:", material_properties)
```

Data Structures

Sets: Sets are used for storing unique items.

```
# Set of unique component IDs in an assembly
component_ids = {'A1', 'A2', 'B1', 'A1'} # 'A1' is duplicated

# Viewing the unique components
print("Unique component IDs:", component_ids)
```

Dictionaries: Dictionaries are key-value pairs and are useful for associating unique identifiers with specific data.

```
# Dictionary to store properties of different materials
material_properties = {
    'steel': {'density': 7850, 'young_modulus': 210e9},
    'aluminum': {'density': 2700, 'young_modulus': 69e9}
}

# Accessing properties of steel
print("Steel Properties:", material_properties['steel'])
```

Data Handling and Analysis

Reading and writing data files (CSV, Excel, JSON) using Python or MATLAB.

```
timestamp, sensor1, sensor2, sensor3
2024-05-01 00:00:00, 0.1, 0.3, 0.2
2024-05-01 00:00:01, 0.2, 0.4, 0.1
2024-05-01 00:00:02, 0.15, 0.35, 0.25
2024-05-01 00:00:03, 0.18, 0.38, 0.22
2024-05-01 00:00:04, 0.12, 0.32, 0.18
```

```
% Read data from a CSV file
data = readtable('validation.csv');

% Calculate the mean of the column 'sensor1'
sensor1_mean = mean(data.sensor1);

% Display the result
disp(['The mean of sensor1 is: ', num2str(sensor1_mean)]);
```

```
# pip install pandas matplotlib
import pandas as pd
import matplotlib.pyplot as plt
# Read data from a CSV file
data = pd.read_csv('validation.csv')
sensor1_mean = data['sensor1'].mean()
# Display the result
print(f'The mean of sensor1 is: {sensor1_mean}')
#####
# Parse 'timestamp' if it's in string format and you want to use it as X-axis
data['timestamp'] = pd.to_datetime(data['timestamp'])
# Plotting
plt.figure(figsize=(10, 6)) # Set the figure size for better readability
# Plot each sensor's data with a different line
plt.plot(data['timestamp'], data['sensor1'], label='Sensor 1', marker='o')
plt.plot(data['timestamp'], data['sensor2'], label='Sensor 2', marker='x')
plt.plot(data['timestamp'], data['sensor3'], label='Sensor 3', marker='s')
# Adding titles and labels
plt.title('Sensor Readings Over Time')
plt.xlabel('Timestamp')
plt.ylabel('Sensor Values')
plt.legend() # Show legends
# Display the plot
plt.grid(True) # Add grid for easier readability
plt.show()
```

Object-Oriented Programming (OOP) Concepts

Object-Oriented Programming (OOP) is a programming paradigm that uses "objects" - data structures consisting of data fields and methods together with their interactions - to design applications and computer programs. It centers around three main concepts: classes, objects, and inheritance.

Class: A blueprint for creating objects. A class defines a type in terms of its data (attributes) and its behavior (methods).

Object: An instance of a class. When a class is defined, no memory is allocated until an object is created from the class, also known as instantiation. Each object can hold different data, but they share the structure and behavior (methods) defined by their class.

```
class MachineComponent:
    def __init__(self, material):
        self.material = material

    def display_material(self):
        print(f"Component material: {self.material}")

class Bearing(MachineComponent): # Inherits from MachineComponent
    def __init__(self, material, bearing_type):
        super().__init__(material)
        self.bearing_type = bearing_type

    def display_info(self):
        print(f"Bearing type: {self.bearing_type}, Material: {self.material}")

# Creating an object of Bearing
bearing1 = Bearing('Ceramic', 'Ball')
bearing1.display_info()
```

Parallel and High-Performance Computing

Parallel Computing: Parallel computing involves dividing large problems into smaller ones, which are then solved concurrently across multiple processing elements within a computer. It can be based on CPUs or GPUs that perform the computations. Types:

- 1. Data Parallelism:** Distributes chunks of data across multiple computing cores and performs the same operation on each chunk.
- 2. Task Parallelism:** Distributes threads or tasks across different cores, where each task can be different.

High-Performance Computing (HPC) harnesses the power of parallel processing to run advanced application programs efficiently, reliably, and quickly. This often involves using supercomputers or computer clusters.

Components:

- 1. Supercomputers:** Extremely fast computers at the cutting edge of current processing capacity.
- 2. Clusters:** Groups of linked computers that work together closely, so in many respects, they can be viewed as a single system.

Parallel Algorithms: Algorithms specifically designed to exploit parallel architectures.

Message Passing Interface (MPI): A standardized message-passing system to program parallel computers.

OpenMP (Open Multi-Processing): An API that supports multi-platform shared-memory parallel programming in C, C++, and Fortran.

CUDA (Compute Unified Device Architecture): A parallel computing platform and API model created by NVIDIA, allowing software developers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing.

SLURM/PBS schedulers: for submitting jobs in HPC

Parallel and High-Performance Computing

Serial Python Script for Factorial Computation

```
import math
import time

def compute_factorial(number):
    result = math.factorial(number)
    print(f"The factorial of {number} is {result}")
    return result

def serial_factorial(numbers):
    results = []
    for number in numbers:
        results.append(compute_factorial(number))
    return results

if __name__ == '__main__':
    numbers = [5, 7, 9, 10, 12, 15, 18, 20] # List of numbers to calculate factorial
    start_time = time.time()
    results = serial_factorial(numbers)
    end_time = time.time()
    #print("Factorial results:", results)
    print(f"Time taken for serial computation: {end_time - start_time} seconds")
```

Parallel Python Script for Factorial Computation

```
from multiprocessing import Pool
import math
import time

def compute_factorial(number):
    return f"The factorial of {number} is {math.factorial(number)}"

def parallel_factorial(numbers):
    pool = Pool(processes=2) # Creates a pool of 2 worker processes
    results = pool.map(compute_factorial, numbers)
    pool.close() # No more tasks
    pool.join() # Wait for completion
    return results

if __name__ == '__main__':
    numbers = [5, 7, 9, 10, 12, 15, 18, 20]
    start_time = time.time()
    results = parallel_factorial(numbers)
    end_time = time.time()
    for result in results:
        print(result)
    print(f"Time taken for parallel computation: {end_time - start_time} seconds")
    print("Number of CPUs:", multiprocessing.cpu_count())
```

Parallel and High-Performance Computing

```
import numpy as np
import pycuda.autoinit
import pycuda.driver as drv
from pycuda.compiler import SourceModule
import time

def cpu_compute_squares(data):
    """ Compute squares using the CPU """
    return data**2

def gpu_compute_squares(data):
    """ Compute squares using the GPU """
    mod = SourceModule("""
__global__ void compute_squares(float *dest, float *a)
{
    const int i = threadIdx.x + blockDim.x * blockIdx.x;
    dest[i] = a[i] * a[i];
}
""")
    compute_squares = mod.get_function("compute_squares")

    dest = np.empty_like(data)
    compute_squares(
        drv.Out(dest), drv.In(data),
        block=(1024, 1, 1), grid=(len(data) // 1024, 1)
    )
    return dest
```

```
if __name__ == "__main__":
    # Generate large array of random floats
    data = np.random.randn(10000000).astype(np.float32)

    # Compute squares on CPU
    start_time = time.time()
    cpu_result = cpu_compute_squares(data)
    cpu_time = time.time() - start_time
    print("CPU computation time:", cpu_time)

    # Compute squares on GPU
    start_time = time.time()
    gpu_result = gpu_compute_squares(data)
    gpu_time = time.time() - start_time
    print("GPU computation time:", gpu_time)
```

What is Quantum Mechanics?

In classical mechanics, systems are defined as a deterministic collection of interacting particles whose positions and velocities evolve according to Newton's equations of motion.

In contrast, in quantum mechanics systems are described by a mathematical object known as the wavefunction, which depends on the positions of all the electrons and nuclei in the system. The wavefunction of the electron evolves over time according to the Schrödinger equation (Nobel prize in Physics 1933).

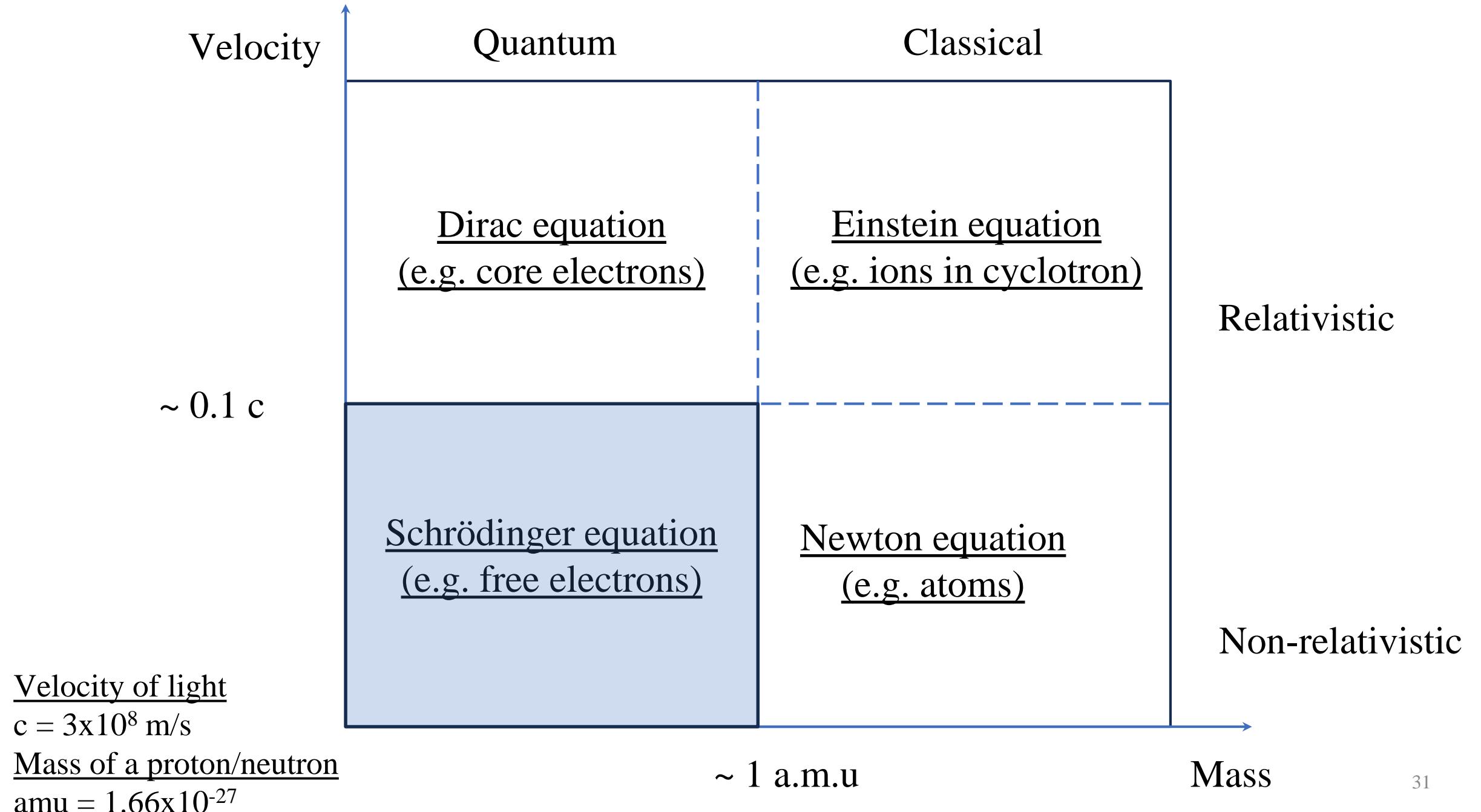
$$\frac{1}{\sqrt{2}} |\text{alive cat}\rangle + \frac{1}{\sqrt{2}} |\text{dead mouse}\rangle$$

$$\hat{H} \Psi = E \Psi$$

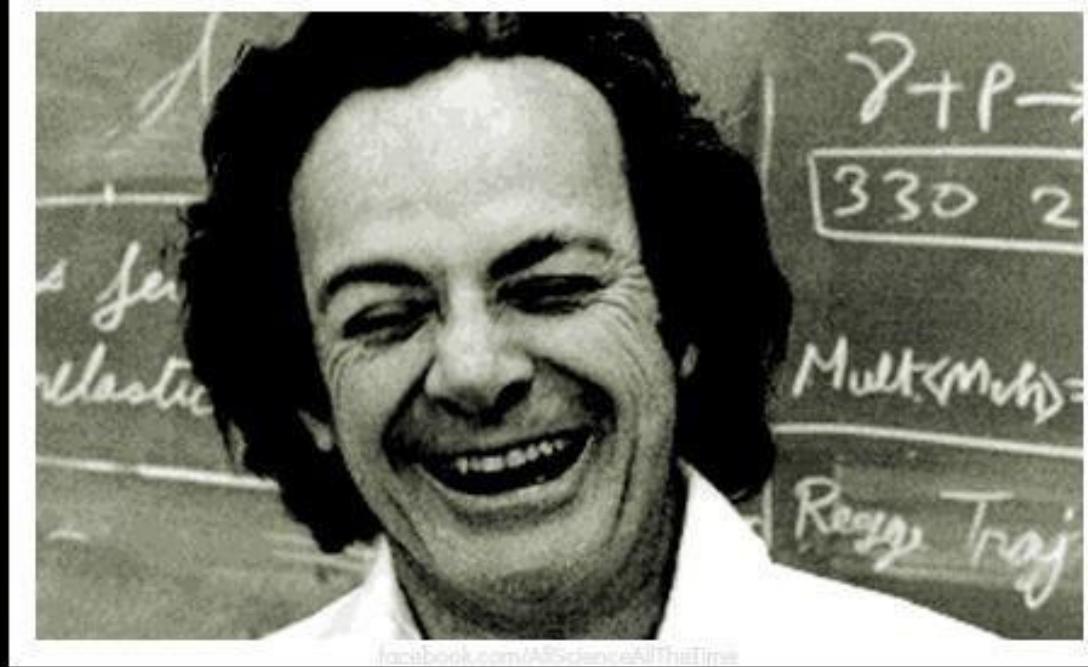
|
Hamiltonian
Operator
(Energy operator)
|
Energy
eigenvalue

$$\frac{-\hbar^2}{2m} \nabla^2 \Psi(\mathbf{r}) + V(r) \Psi(\mathbf{r}) = E \Psi(\mathbf{r})$$

$$\begin{matrix} Kinetic \\ Energy \end{matrix} + \begin{matrix} Potential \\ Energy \end{matrix} = \begin{matrix} Total \\ Energy \end{matrix}$$



Quantum Mechanics is Difficult but Useful!



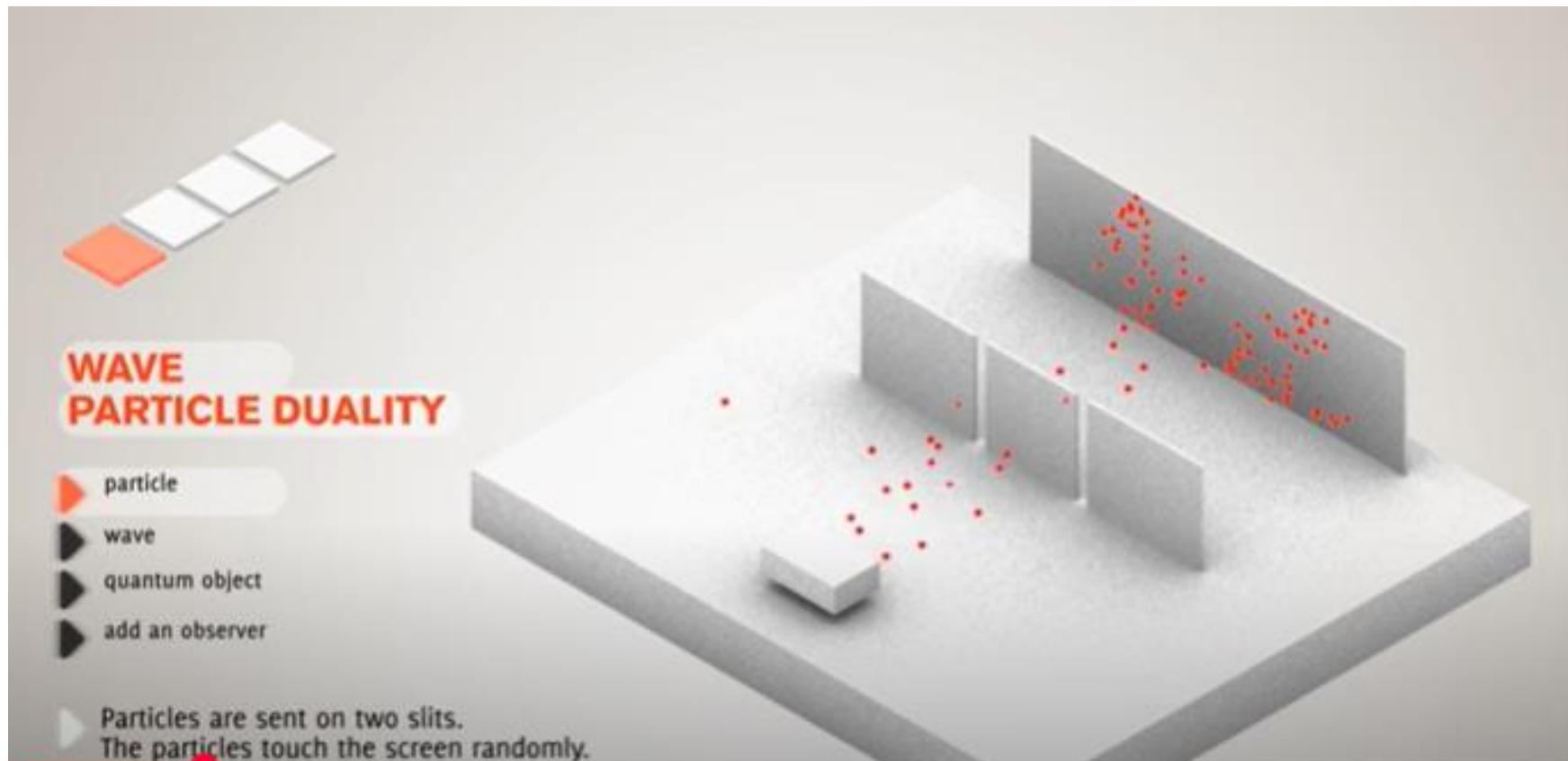
"Anyone who claims to understand quantum theory is either lying or crazy."
-Richard Feynman

Learn
Quantum Physics
in one lesson



Key Principles of Quantum Mechanics

Wave-Particle Duality describes how every particle or quantum entity can exhibit both particle-like and wave-like behavior, e.g. in the double-slit experiment. Light behaves as a wave when it creates an interference pattern and as particles (photons) when detected individually. De Broglie wavelength $\lambda=hp$, λ is the wavelength, h is Planck's constant, p is particle's momentum.



Introduction to Density Functional Theory

Many-body/electron solution of Schrodinger equation is challenging → DFT

The Nobel Prize in Chemistry 1998



Photo from the Nobel Foundation archive.
Walter Kohn

Prize share: 1/2

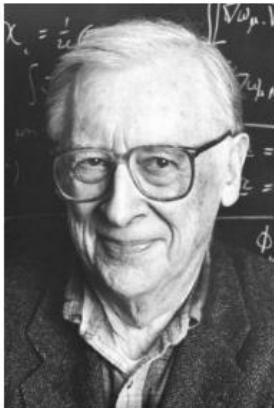


Photo from the Nobel Foundation archive.
John A. Pople

Prize share: 1/2

The Nobel Prize in Chemistry 1998 was divided equally between Walter Kohn "for his development of the density-functional theory" and John A. Pople "for his development of computational methods in quantum chemistry"

PHYSICAL REVIEW

VOLUME 136, NUMBER 3B

9 NOVEMBER 1964

Inhomogeneous Electron Gas*

P. HOHENBERG†

École Normale Supérieure, Paris, France

AND

W. KOHN‡

*École Normale Supérieure, Paris, France and Faculté des Sciences, Orsay, France
and*

University of California at San Diego, La Jolla, California

(Received 18 June 1964)

This paper deals with the ground state of an interacting electron gas in an external potential $v(\mathbf{r})$. It is proved that there exists a universal functional of the density, $F[n(\mathbf{r})]$, independent of $v(\mathbf{r})$, such that the expression $E = \int v(\mathbf{r})n(\mathbf{r})d\mathbf{r} + F[n(\mathbf{r})]$ has as its minimum value the correct ground-state energy associated with $v(\mathbf{r})$. The functional $F[n(\mathbf{r})]$ is then discussed for two situations: (1) $n(\mathbf{r}) = n_0 + \tilde{n}(\mathbf{r})$, $\tilde{n}/n_0 \ll 1$, and (2) $n(\mathbf{r}) = \varphi(\mathbf{r}/r_0)$ with φ arbitrary and $r_0 \rightarrow \infty$. In both cases F can be expressed entirely in terms of the correlation energy and linear and higher order electronic polarizabilities of a uniform electron gas. This approach also sheds some light on generalized Thomas-Fermi methods and their limitations. Some new extensions of these methods are presented.

PHYSICAL REVIEW

VOLUME 140, NUMBER 4A

15 NOVEMBER 1965

Self-Consistent Equations Including Exchange and Correlation Effects*

W. KOHN AND L. J. SHAM

University of California, San Diego, La Jolla, California

(Received 21 June 1965)

From a theory of Hohenberg and Kohn, approximation methods for treating an inhomogeneous system of interacting electrons are developed. These methods are exact for systems of slowly varying or high density. For the ground state, they lead to self-consistent equations analogous to the Hartree and Hartree-Fock equations, respectively. In these equations the exchange and correlation portions of the chemical potential of a uniform electron gas appear as additional effective potentials. (The exchange portion of our effective potential differs from that due to Slater by a factor of $\frac{2}{3}$.) Electronic systems at finite temperatures and in magnetic fields are also treated by similar methods. An appendix deals with a further correction for systems with short-wavelength density oscillations.

Introduction to Density Functional Theory

- Schrödinger equation for electrons: **wave–particle duality**,
- Schrödinger equation of a **fictitious system** (the "Kohn–Sham system") of **non-interacting particles** (typically electrons) that generate the **same density** as any given system of interacting particles
- Uses **density** vs **wavefunction** quantity

$$H\psi = E\psi$$

$$\left[-\frac{\hbar^2}{2m} \nabla^2 + V_{Eff}(r) \right] \psi_i(r) = E_i(r) \psi_i(r)$$

Exchange-correlation

$$V_{Eff} = T + V_{Ne} + V_{ee} + V_{XC}$$



Walter Kohn (2013)

- Although a **complete theory**, several approximations such as:

1) K-points, 2) vdW interactions, 3) kinetic energy deriv., 4) spin-orbit coupling, 5) e-ph coupling
(Convergence, OptB88vdW, TBmBJ, SOC topology, Superconducting prop.)

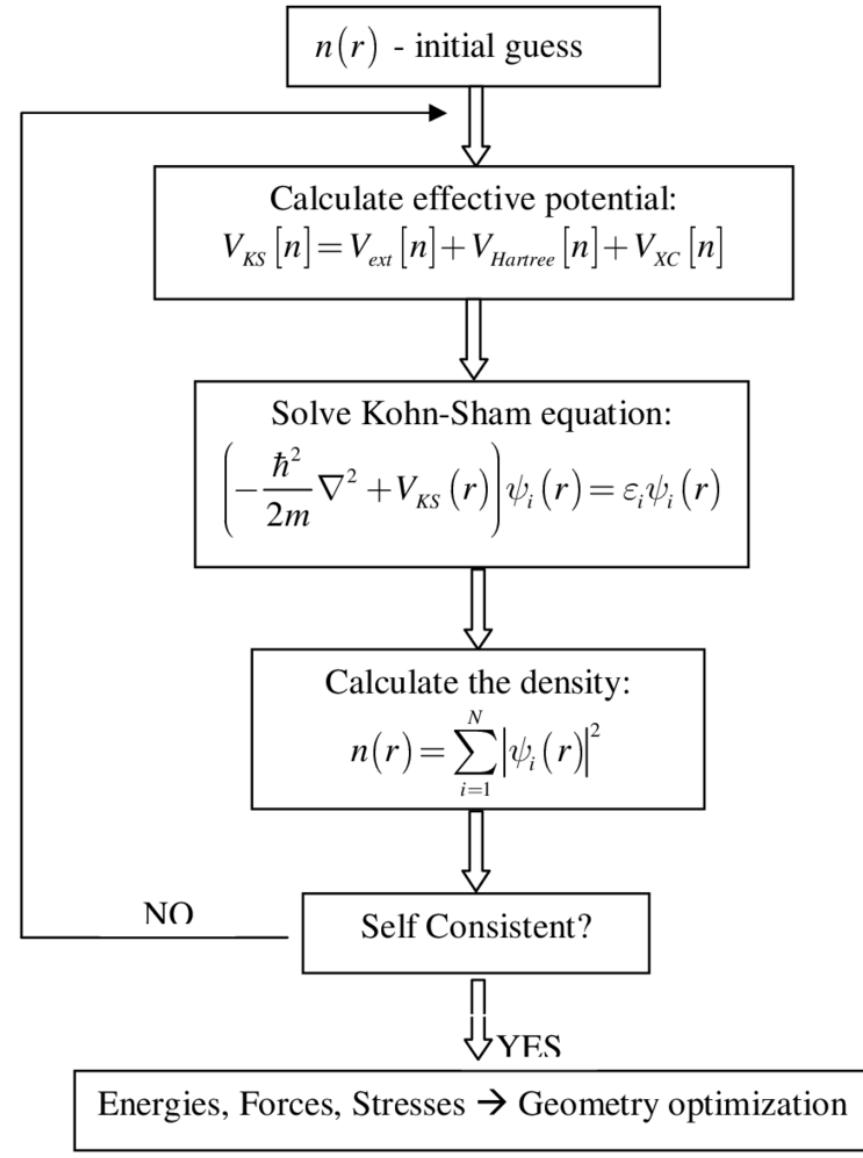
Many DFT databases with GGA-PBE, fixed k-point, no-SOC, ...

Exchange-correlation

The exchange-correlation functional aims to encapsulate all the many-body effects of a quantum system into a single term that can be added to the effective potential in which the electrons move.

1. **Exchange Interactions:** These arise due to the quantum mechanical principle that two electrons with the same spin cannot occupy the same spatial position, a consequence of the Pauli exclusion principle. This interaction results in a form of repulsion that is not purely electrostatic but quantum mechanical, influencing the spatial distribution of electrons with like spins.
 2. **Correlation Interactions:** These describe how the motion of one electron in a system is correlated with the motion of all other electrons due to their mutual electrostatic interactions. Electrons tend to avoid each other due to these repulsive forces, which is not captured by considering only the average electron density or the exchange effects alone.
- **Approximations:** Various models and approximations are used to estimate Exc , the most common being:
 - **Local Density Approximation (LDA):** Assumes the exchange-correlation energy per electron can be approximated by that of a uniform electron gas at each point in space.
 - **Generalized Gradient Approximation (GGA):** Improves upon LDA by also considering the gradient of the electron density at each point, allowing for more accurate modeling of systems where the electron density changes rapidly.
 - **Hybrid Functionals:** Include a portion of exact exchange from Hartree-Fock theory combined with DFT exchange-correlation, providing better accuracy for some systems at the cost of increased computational effort.

Overview of the DFT Framework- 1D example



```

import numpy as np
from scipy.linalg import eigh_tridiagonal
import matplotlib.pyplot as plt

def harmonic_potential(x):
    return 0.5 * (x ** 2)

def effective_potential(x, rho, alpha=0.5):
    """Combine external and Hartree potential"""
    V_ext = harmonic_potential(x)
    V_hartree = np.convolve(rho, np.exp(-alpha * np.abs(x)), mode='same') / alpha
    return V_ext + V_hartree

def solve_kohn_sham(x, num_electrons):
    dx = x[1] - x[0]
    N = len(x)

    # Approximate second derivative using finite differences
    diagonal = np.full(N, -2.0 / dx**2)
    off_diagonal = np.full(N-1, 1.0 / dx**2)

    # Initial guess for electron density
    rho = np.ones(N) * (num_electrons / N)

    for _ in range(30): # Self-consistent iterations
        V_eff = effective_potential(x, rho)

        # Hamiltonian matrix in tridiagonal form
        hamiltonian_diagonal = -0.5 * diagonal + V_eff

        # Solve the Schrodinger equation
        energies, psi = eigh_tridiagonal(hamiltonian_diagonal, off_diagonal, select='i', select_range=(0, num_electrons-1))

        # Update electron density
        rho_new = np.sum(psi**2, axis=1)
        rho = 0.7 * rho + 0.3 * rho_new # Mixing to aid convergence

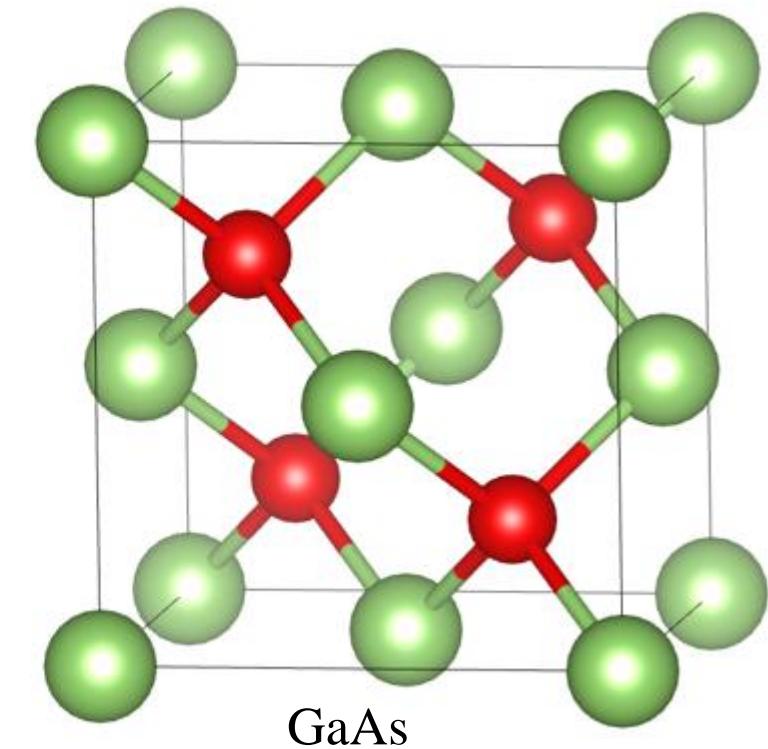
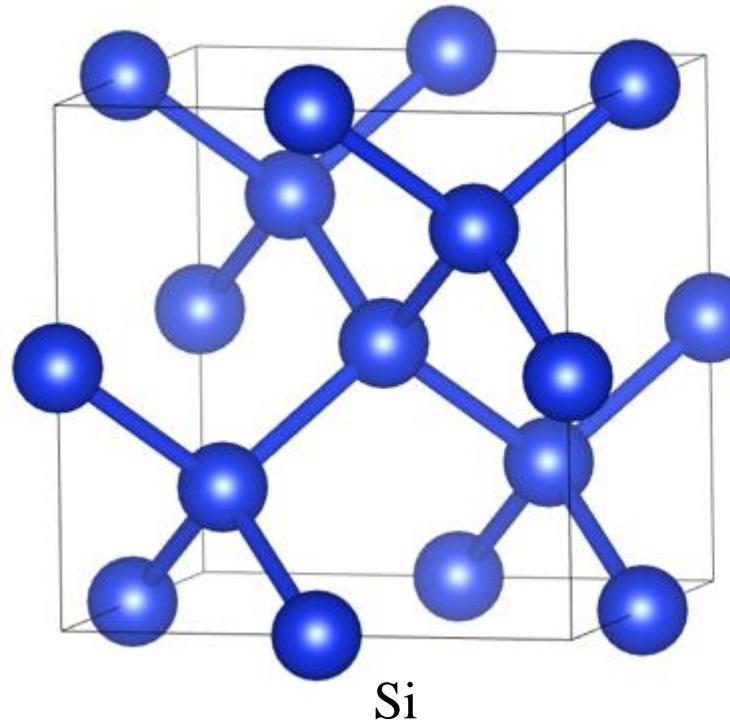
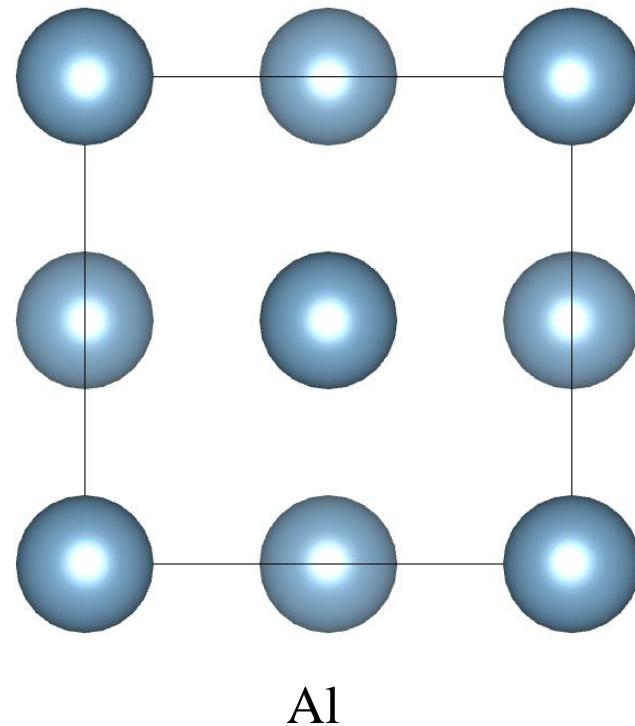
    return energies, psi, rho
  
```

The provided Python code implements the DFT framework for a single electron in a 1D harmonic oscillator potential. It includes imports for numpy, scipy, and matplotlib. The code defines three main functions: `harmonic_potential` (a simple parabolic potential), `effective_potential` (combining external and Hartree potentials), and `solve_kohn_sham` (the core iterative solver). The `solve_kohn_sham` function uses finite differences to approximate the second derivative in the Schrödinger equation. It initializes the electron density to a uniform value, then iterates 30 times to find the self-consistent solution. Each iteration involves calculating the effective potential, forming the Hamiltonian in tridiagonal form, solving the Schrödinger equation to get energy levels and wave functions, and updating the electron density using a mixing scheme (0.7 times old, 0.3 times new).

- Finite difference solution
- Single electron in a harmonic oscillator potential

Setting Up the System/Inputs

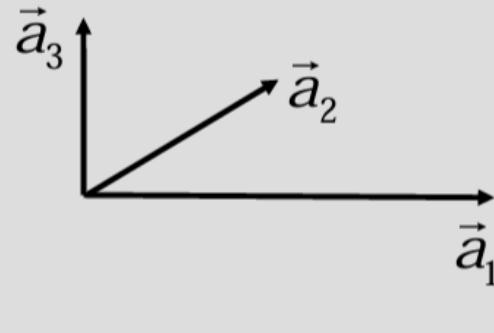
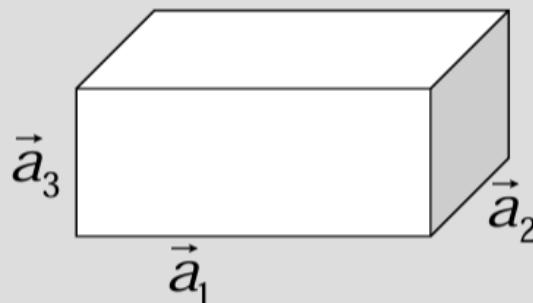
- Preparing the initial structure: atomic positions, lattice parameters
- These atomic positions can be obtained from experiments such as X-ray diffraction, NMR
- There are also large databases of such structures such as Crystallography Open Database, JARVIS-DFT, Materials Project, OQMD etc.
- Primitive (smallest) cells are preferred for atomistic simulations for computational cost reasons
- For defect/interface simulations large number of atoms could be necessary



Basis Sets

- In order to solve Kohn-Sham -equation, we need to expand the wavefunction in a basis set
- Simplify the problem of solving the Kohn-Sham equations by expanding the wavefunctions in a manageable set of basis functions.
- **Types:**
 - **Localized Orbitals:** Such as Gaussian or Slater-type orbitals; effective for molecular/isolated systems.
 - **Plane Waves:** used to model infinite systems, ideal for periodic systems; represent wave functions as sums of sinusoids. (Here $\vec{a}_1, \vec{a}_2, \vec{a}_3$ are lattice vectors)

System is assumed to be placed inside a unit cell defined by the unit vectors



The volume of the unit cell is

$$\Omega = [\vec{a}_1, \vec{a}_2, \vec{a}_3] = \vec{a}_1 \cdot (\vec{a}_2 \times \vec{a}_3)$$

$$\psi_i = \sum_{\alpha} c_{\alpha} \varphi_{\alpha}$$

Source: LANL

$$\vec{b}_1 = 2\pi \frac{\vec{a}_2 \times \vec{a}_3}{\Omega}$$

$$\vec{b}_2 = 2\pi \frac{\vec{a}_3 \times \vec{a}_1}{\Omega}$$

$$\vec{b}_3 = 2\pi \frac{\vec{a}_1 \times \vec{a}_2}{\Omega}$$

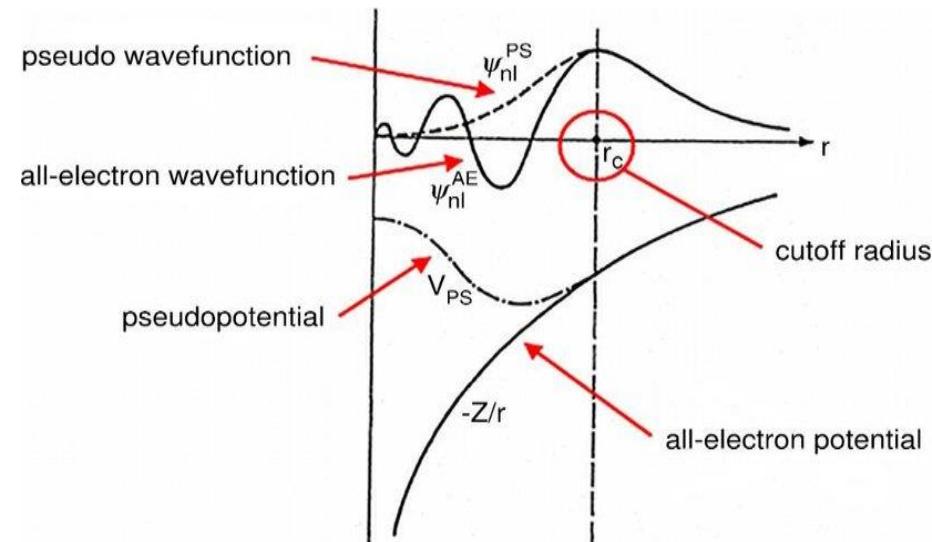
Reciprocal lattice vectors

Wave-vectors that satisfy the periodicity of the lattice

$$\vec{G}_{i_1 i_2 i_3} = \left(i_1 - \frac{N_1}{2} \right) \vec{b}_1 + \left(i_2 - \frac{N_2}{2} \right) \vec{b}_2 + \left(i_3 - \frac{N_3}{2} \right) \vec{b}_3$$

Pseudopotentials

- Effective potentials used to replace the all-electron potential near the nucleus in DFT calculations.
- Reduce computational complexity by focusing on valence electrons, effectively ignoring core electrons that do not participate in bonding
- **Types of Pseudopotentials**
 - **Norm-Conserving Pseudopotentials:** Ensure that the pseudowavefunction matches the all-electron wavefunction outside a certain cutoff radius.
 - **Ultrasoft Pseudopotentials:** Allow for greater flexibility and softer functions, reducing the plane wave basis set size needed for convergence.
 - **PAW (Projector Augmented Wave):** A method that provides the accuracy of all-electron calculations with the efficiency of pseudopotentials.



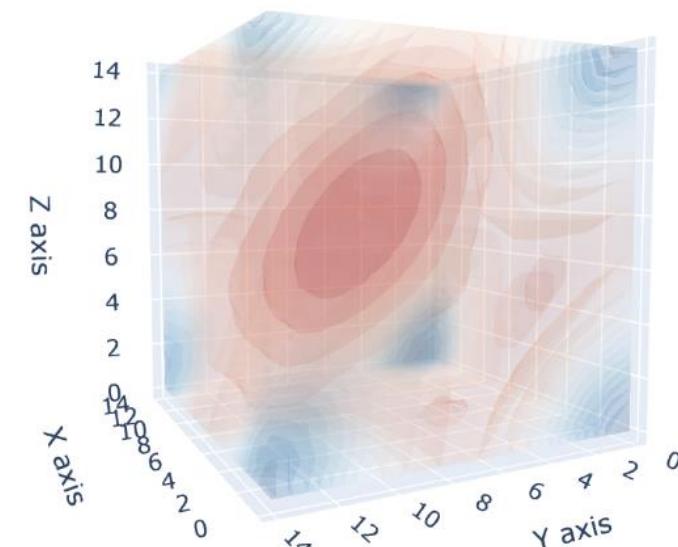
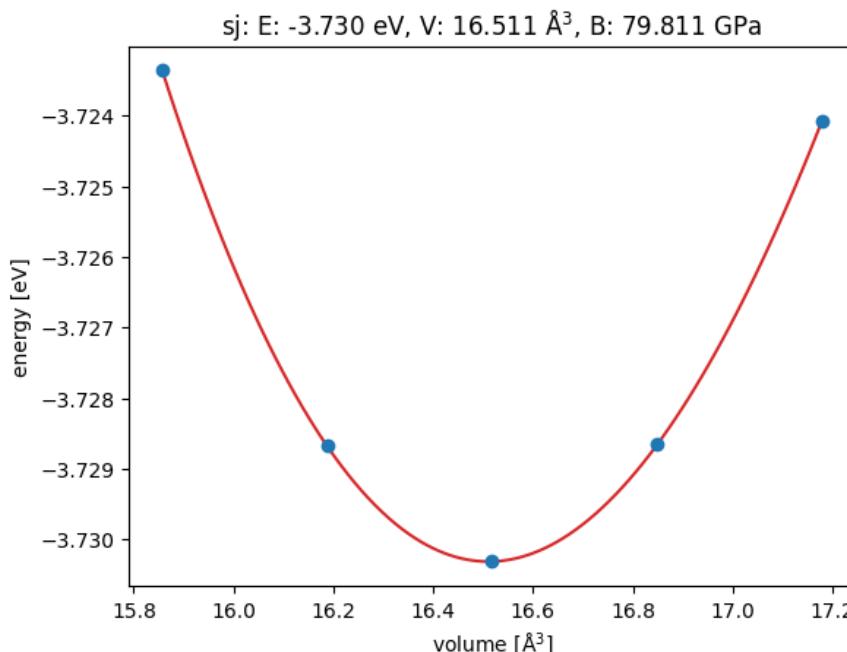
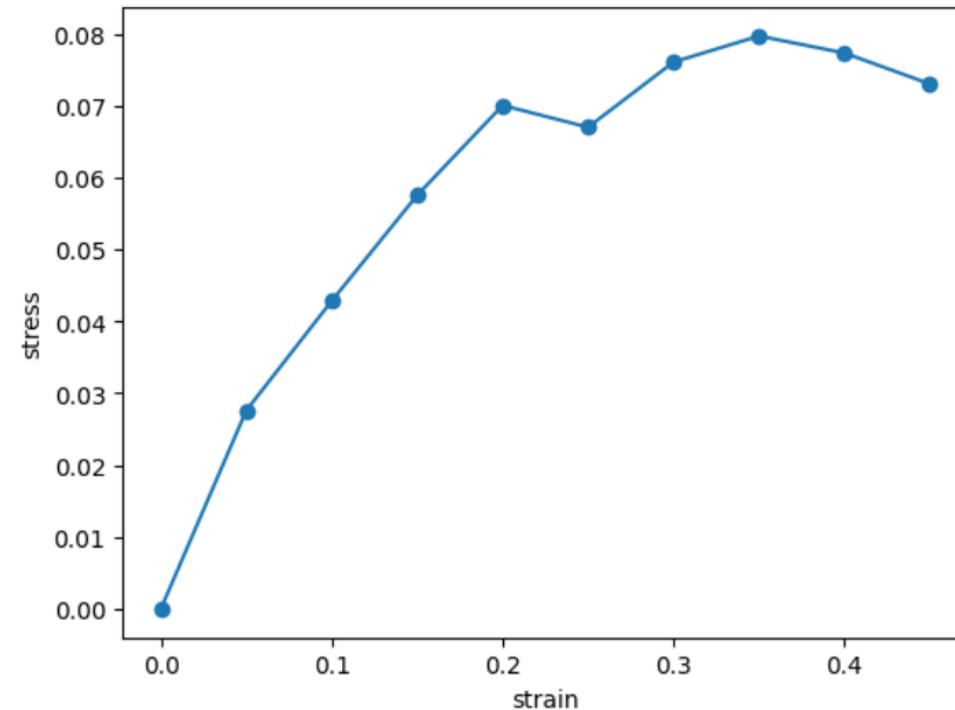
Choosing the Right Software

- Popular DFT software packages: VASP, Quantum ESPRESSO, GPAW, ABINIT
- Licensing and computational requirements
- Same Kohn-Sham equation, different format for input/output files
- Reading documentation could be useful



Outputs

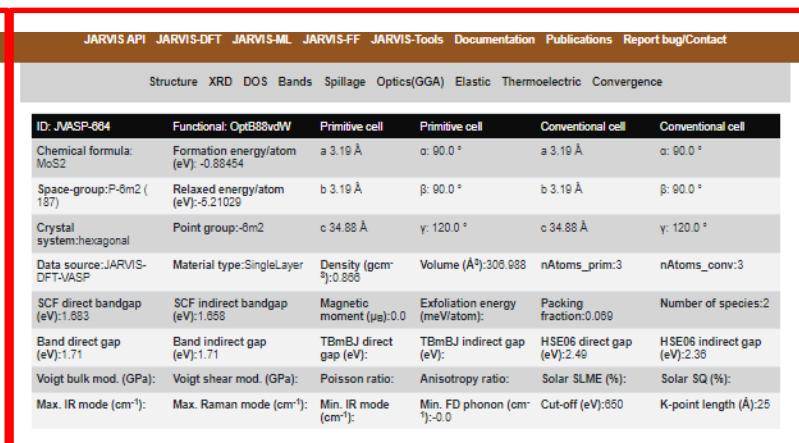
- Electronic density of states
- Charge density
- Optimized lattice constants
- Stress-strain curves
- Bulk modulus
- Elastic constants
- Thermal conductivity
- Electronic bandgap
- Magnetic moment
- Many more



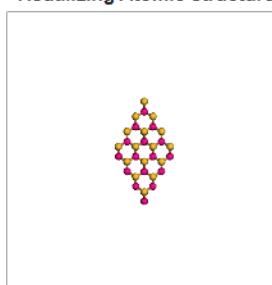
JARVIS-DFT

The screenshot shows a web browser window with a red border. The address bar contains the URL <https://jarvis.nist.gov/jarvisdft/>. The page header includes the NIST logo, navigation links for JARVIS, Publications, Colab, Downloads, Tutorials, Events, Log In / Sign Up, and Help. The main title is "Density Functional Theory (DFT)". Below the title is a search bar containing "Mo-S-" and two buttons: a blue "Search" button with a magnifying glass icon and a green "Refresh" button with a circular arrow icon. The background features a dark geometric pattern. A large blue link at the bottom center reads <https://jarvis.nist.gov/jarvisdft/>. The periodic table is partially visible on the left and right sides.

id	formula	Spg	SpgNum	crys	func	E_form	OPT_gap	MBJ_gap	hse_gap	Kv	Gv	poisson	spillage	sime	mag	type
JVASP-664	MoS2	P-6m2	187	hexagonal	OptB88vdW	-0.88454	1.658	-	-	-	-	-	0.003	-	0.0	2D
JVASP-730	MoS2	R-3m	166	trigonal	OptB88vdW	-0.60967	0.000	-	-	-	-	-	0.976	-	-0.0	2D
JVASP-100856	MoS3	P2_1/m	11	monoclinic	OptB88vdW	-0.34773	0.443	-	-	-	-	-	-	-	0.0	3D
JVASP-1IIII7	MoS2	P-3m1	164	trigonal	OptB88vdW	-0.69305	0.000	0.0	-	-	-	-	-	-	0.0	3D
JVASP-127507	MoS	Pm-3m	221	cubic	OptB88vdW	-0.18293	0.000	-	-	-	-	-	-	-	0.0	3D
JVASP-132473	MoS3I	P1	1	triclinic	OptB88vdW	0.06061	0.651	-	-	-	-	-	-	-	3.993	3D
JVASP-144515	Mo3S4	P-1	2	triclinic	OptB88vdW	-0.73826	0.000	-	-	-	-	-	-	-	0.0	3D
JVASP-228	MoS2	R-3m	166	trigonal	OptB88vdW	-0.69909	0.000	0.0	-	51.32	45.47	0.19	0.852	-	0.0	3D
JVASP-28379	MoS2	P6_3/mmc	194	hexagonal	OptB88vdW	-0.95754	0.958	1.364	-	71.48	44.27	0.26	-	33.92	-	3D
JVASP-28413	MoS2	P-3m1	164	trigonal	OptB88vdW	-0.92268	1.207	1.471	-	-	-	-	-	32.7	0.0	3D
JVASP-28733	MoS2	P6_3/mmc	194	hexagonal	OptB88vdW	-0.96160	0.921	-	-	70.62	47.69	0.23	0.013	-	0.0	3D
JVASP-34138	MoS2	P6_3/mmc	194	hexagonal	OptB88vdW	-0.96135	0.920	1.356	-	-	-	-	-	33.93	0.0	3D

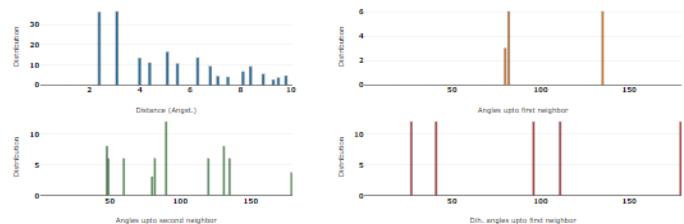


[Show POSCAR](#) [Show POSCAR-conv](#) [Show XYZ format](#) [Show CIF format](#)



Atomic Structure Analysis

The following shows the radial, angle and dihedral distribution function plots.



JARVIS-DFT

Features	JARVIS-DFT	Features	JARVIS-DFT
#Materials (Struct., E _f , E _g)	80000	2D monolayers	1011
DFT functional/methods	vdW-DFT-OptB88, TBmBJ, DFT+SOC	Raman spectra	400
K-point/cut-off	Converged for each material	Seebeck, Power Factors	23210
SCF convergence criteria	Energy & Forces	Solar SLME	8614
Elastic tensors & point phonons	17402	Spin-orbit Coupling Spillage	11383
Piezoelectric, IR spect.	4801	WannierTB	1771
Dielectric tensors (w/o ion)	4801 (15860)	STM images	1432
Electric field gradients	11865	Surfaces	300
SuperCon Tc	2200 (1058 ambient condition)	Defects	400
		Interfaces	1.4 trillion (IU), 600 (ASJ)

JARVIS-DFT

High-throughput Identification and Characterization of Two-dimensional Materials using Density functional theory

Kamal Choudhary , Irina Kalish, Ryan Beams & Francesca Tavazza

Scientific Reports 7, Article number: 5179 (2017) | [Cite this article](#)

Computational screening of high-performance optoelectronic materials using OptB88vdW and TB-mBJ formalisms

Kamal Choudhary , Qin Zhang, Andrew C.E. Reid, Sugata Chowdhury, Nhan Van Nguyen, Zachary Trau

Marcus W. Newrock, Faical Yannick Congo & Francesca Tavazza

Accelerated Discovery of Efficient Solar Cell Materials Using Quantum and Machine-Learning Methods

Kamal Choudhary*, Marnik Bercx, Jie Jiang, Ruth Pachter, Dirk Lamoen, and Francesca Tavazza

 [Cite this: Chem. Mater.](#) 2019, 31, 15, 5900–5908

Publication Date: July 17, 2019 

<https://doi.org/10.1021/acs.chemmater.9b02166>

Copyright © 2019 American Chemical Society

Article Views Altmetric Citations

4443 9 91

[LEARN ABOUT THESE METRICS](#)

Share Add to Export



RIS

High-throughput density functional perturbation theory and machine learning predictions of infrared, piezoelectric, and dielectric responses

Kamal Choudhary , Kevin F. Garrity, Vinit Sharma, Adam J. Biacchi, Angela R. Hight Walker & Francesca

Tavazza

npj Computational Materials 6, Article number: 64 (2020) | [Cite this article](#)

Data-driven discovery of 3D and 2D thermoelectric materials

Kamal Choudhary^{2,1} , Kevin F Garrity¹ and Francesca Tavazza¹

Published 27 August 2020 • © 2020 IOP Publishing Ltd

Journal of Physics: Condensed Matter, Volume 32, Number 47

Citation Kamal Choudhary et al 2020 *J. Phys.: Condens. Matter* 32 475501

High-throughput Discovery of Topologically Non-trivial Materials using Spin-orbit Spillage

Kamal Choudhary , Kevin F. Garrity & Francesca Tavazza

Scientific Reports 9, Article number: 8534 (2019) | [Cite this article](#)

Designing high- T_c superconductors with BCS-inspired screening, density functional theory, and deep-learning

Kamal Choudhary & Kevin Garrity

npj Computational Materials 8, Article number: 244 (2022) | [Cite this article](#)

High-Throughput DFT-Based Discovery of Next Generation Two-Dimensional (2D) Superconductors

Daniel Wines*, Kamal Choudhary, Adam J. Biacchi, Kevin F. Garrity, and Francesca Tavazza

 [Cite this: Nano Lett.](#) 2023, 23, 3, 969–978

Publication Date: January 30, 2023 

<https://doi.org/10.1021/acs.nanolett.2c04420>

Copyright © 2023 American Chemical Society

Article Views Altmetric Citations

3278 3 5

[LEARN ABOUT THESE METRICS](#)

Predicting anomalous quantum confinement effect in van der Waals materials

Kamal Choudhary and Francesca Tavazza

Phys. Rev. Materials 5, 054602 – Published 7 May 2021



RIS

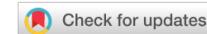
DOI: [10.1039/D4DD00031E](https://doi.org/10.1039/D4DD00031E) (Paper) [Digital Discovery](#), 2024, Advance Article

InterMat: accelerating band offset prediction in semiconductor interfaces with DFT and deep learning[†]

Kamal Choudhary  * and Kevin F. Garrity

Can a deep-learning model make fast predictions of vacancy formation in diverse materials?

Kamal Choudhary ; Bobby G. Sumpter 



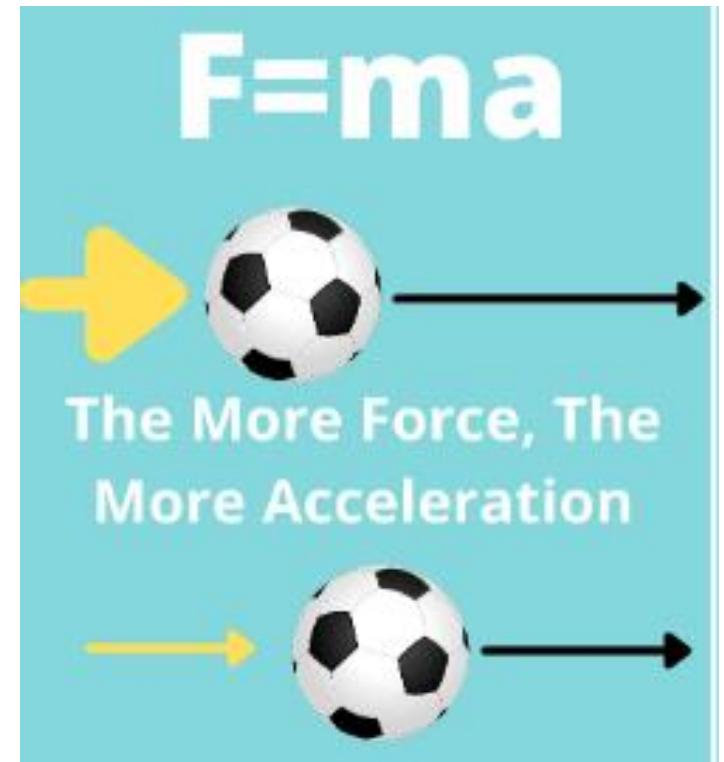
+ Author & Article Information

AIP Advances 13, 095109 (2023)



What is Classical Mechanics?

- **Definition:**
 - Study of the motion of bodies in a frame of reference
 - Deals with macroscopic scales and speeds much lower than the speed of light
- **Applications:**
 - Predicting planetary motions
 - Designing mechanical systems (e.g., vehicles, machinery)
 - Understanding structural dynamics
- **Limitations:**
 - Not applicable at atomic scales or near-light speeds
 - Replaced by quantum mechanics and relativity in such cases



Introduction to Molecular Dynamics

- **What is molecular dynamics (MD)?**

Numerical method for studying many-particle systems such as molecules, clusters, and even macroscopic systems such as gases, liquids and solids

- **First reported MD simulation:**

Alder + Wainwright (1957): Phase diagram of a hard-sphere gas

- **Basic idea of molecular dynamics:**

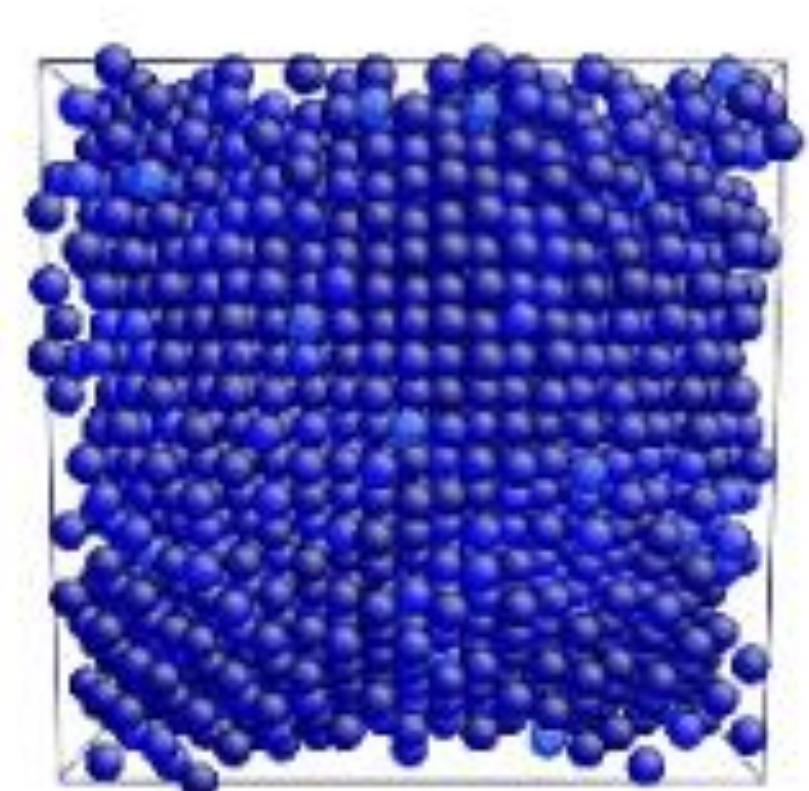
Solution of Newton's equations of motion for the individual particles (atoms, ions, ...)

Interactions are approximated by classical **model potentials** constructed by comparison with experiment (empirical potentials)

Leads to simulation of purely classical many-particle problem

- **Example:**

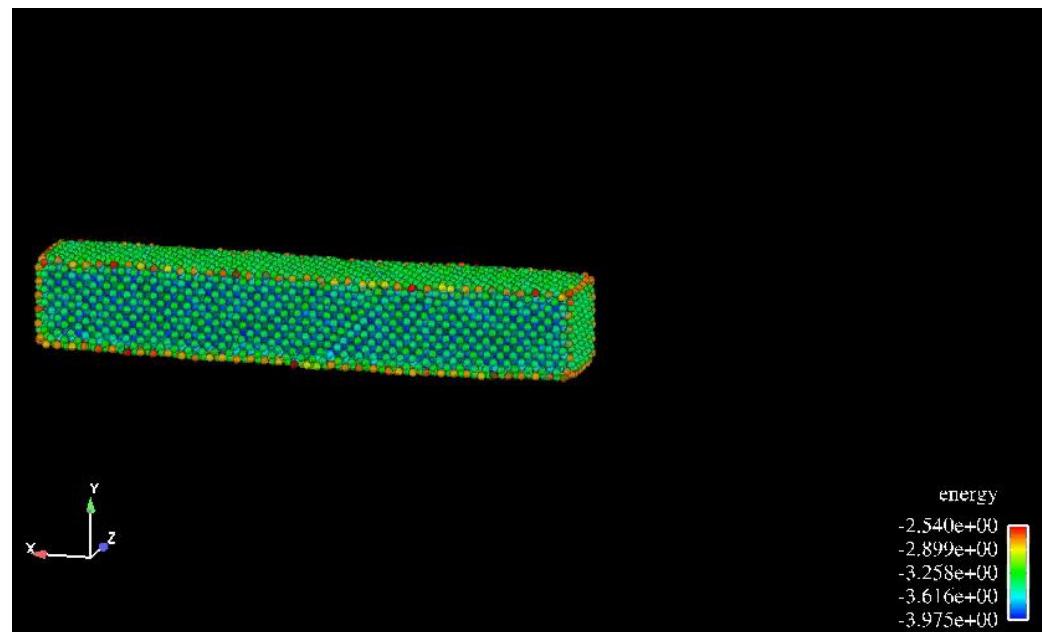
Tensile Loading of an Aluminum Single Crystal



*Movie showing deformation of single crystal aluminum loaded in the <100> direction at a strain rate of 10^{10} s^{-1} and a temperature of 300 K.
https://www.cavs.msstate.edu/icme/code/lammps/tutorials/lammps/uniaxial_tension.php*

Overview of MD

- Pick atoms/particles, masses and forces (or potential) **fitted to DFT/experiments**
- Initialize positions and momentum (boundary conditions in space and time)
- Solve $\mathbf{F} = m \mathbf{a}$ to determine $\mathbf{r}(t)$, $\mathbf{v}(t)$, **the integrator**, and note time is discrete (t_k) not continuous
- Compute properties along the trajectory
- Estimate errors
- Try to use the simulation to answer physical questions



Phase Transition for a Hard Sphere System

B. J. ALDER AND T. E. WAINWRIGHT

University of California Radiation Laboratory, Livermore, California
(Received August 12, 1957)

A CALCULATION of molecular dynamic motion has been designed principally to study the relaxations accompanying various nonequilibrium phenomena. The method consists of solving exactly (to the number of significant figures carried) the simultaneous classical equations of motion of several hundred particles by means of fast electronic computers. Some of the details as they relate to hard spheres and to particles having square well potentials of attraction have been described.^{1,2} The method has been used also to calculate equilibrium properties, particularly the equation of state of hard spheres where differences with previous Monte Carlo³ results appeared.

Interatomic Potentials and Force Fields

Interatomic potentials are **mathematical functions** that describe the potential energy between atoms or molecules as a **function of their positions**. These potentials are crucial for determining the forces in an MD simulation.

Types of Force Fields (FFs)

- **Pairwise Potentials:** **Simplest form**, considering only interactions between pairs of atoms. Examples include Lennard-Jones and Coulombic potentials.
- **Many-Body Potentials:** Consider interactions among **three or more atoms simultaneously**, providing a more accurate representation of atomic interactions. Examples include Embedded Atom Method (EAM) and Tersoff potentials.
- **Bonded Potentials:** Used for molecules, accounting for bond stretching, angle bending, and torsional interactions.

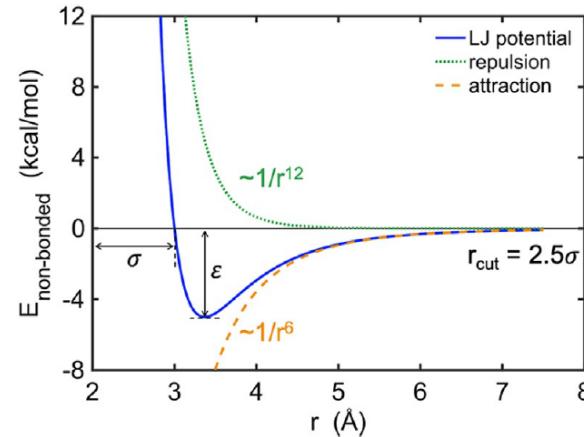


Fig. 5 The Lennard-Jones (LJ) potential. In the figure, r is the distance between a pair of atoms, r_{cut} is the cutoff distance, 2.5σ is a typical value, ϵ is the depth of the minimum energy, and σ is a length scale parameter related to the equilibrium distance between atoms. The dotted and dashed lines show the repulsion and attraction branches of the LJ potential, respectively.

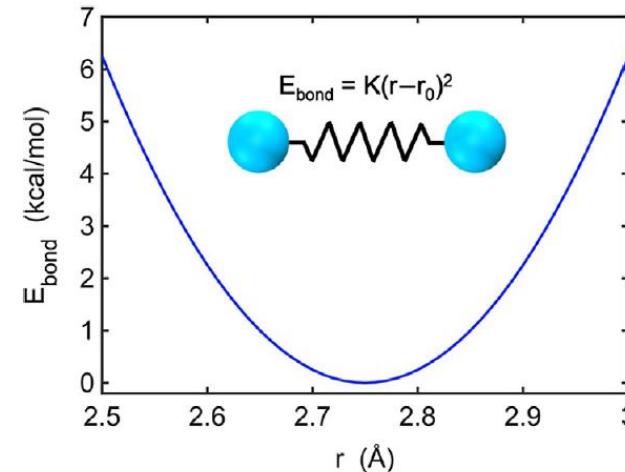


Fig. 3 Harmonic potential, $E_{\text{bond}}(r) = K(r - r_0)^2$, where $K = 100$ kcal/mol/Å 2 is the bond spring constant, and $r_0 = 2.75$ Å is the equilibrium bond length.

Types of FF

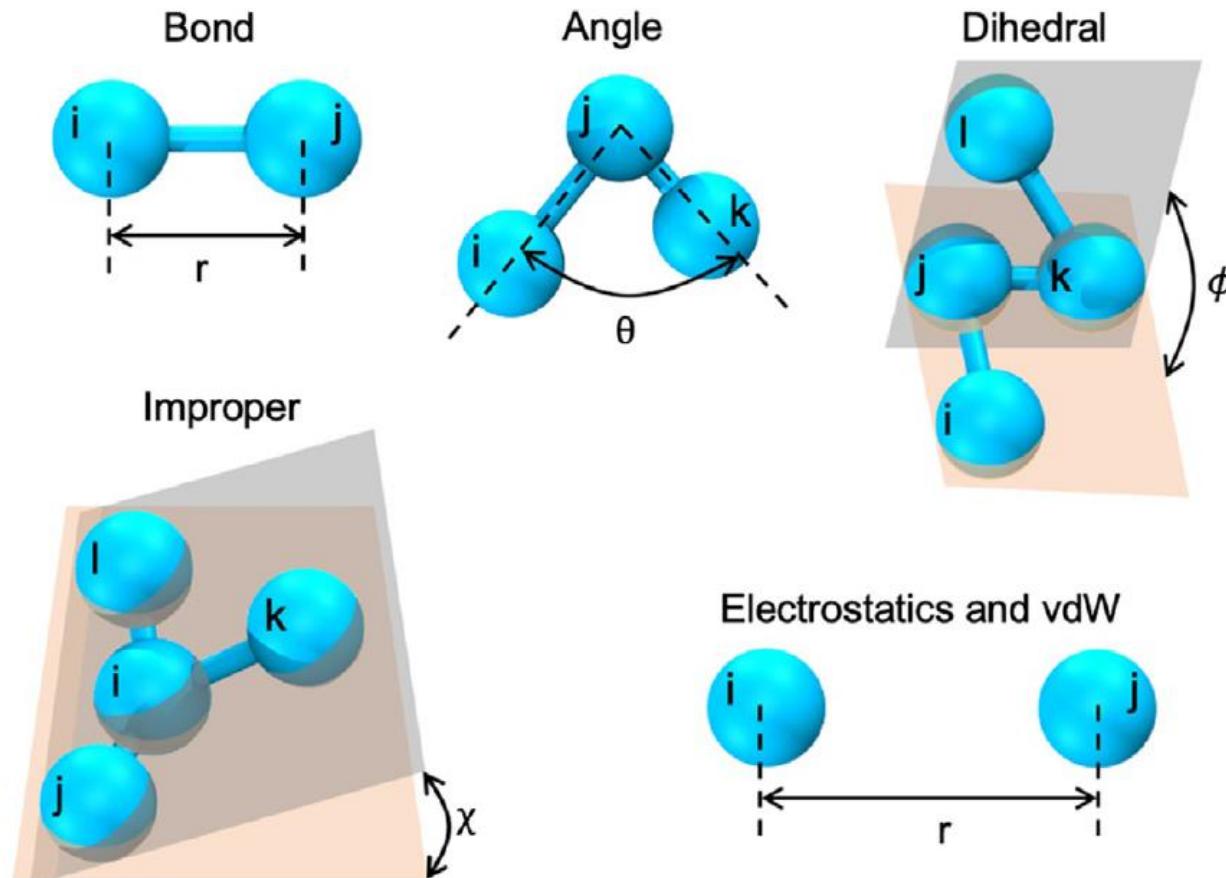


Fig. 2 Different types of interactions typically included in empirical force fields. Bond, angle, dihedral, and improper interactions are considered bonded, and electrostatics and van der Waals nonbonded.

Table 1 List of commonly used force fields in MD or MC simulations.

Force field family	Variants	Most applicable system
CHARMM [1]	CGenFF, CHARMM22, CHARMM27, and CHARMM36.	Organic molecules, solutions, polymers, biochemical molecules, etc.
AMBER [2]	GAFF, GLYCAM, ff94, ff96, ff98, ff99, ff02, ff02EP, etc.	Biochemical molecules (proteins, nucleic acids, polysaccharides, etc.)
DREIDING [15]	–	Organic, biological, polymeric, and main-group inorganic molecules
OPLS [3,4]	OPLS-AA and OPLS-UA.	Biological macromolecules, solutions, etc.
MMX [9,10]	MM2, MM3, MM4, MM+, etc.	Organic chemical compounds, free radicals, ions, etc.
CFF [14]	COMPASS, COMPASS II, CFF95, CFF91, and PCFF.	Organic small molecules, biomolecules, molecular sieves, polymers, metals, etc.
ReaxFF [16]	–	Metals, ceramics, silicon, polymers
ClayFF [17]	–	Hydrated and multicomponent mineral systems and their interfaces with aqueous solutions.

Classical Force-Fields

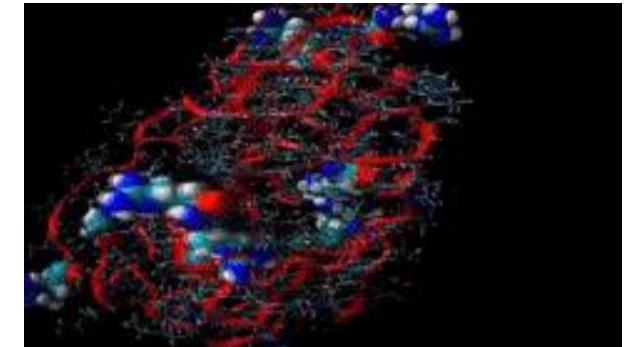
- Coulomb potential
- Lennard-Jones potential
- Morse-potential

$$V(q_1(\vec{r}_1), q_2(\vec{r}_2)) = \frac{kq_1q_2}{|\vec{r}_2 - \vec{r}_1|}$$

$$V_{LJ} = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right]$$

$$V(r) = D_e(e^{-2a(r-r_e)} - 2e^{-a(r-r_e)})$$

One parameter to fit/optimize



These potentials lack angular information hence not able to capture elastic constants well

- Stilinger-Weber potential $V_{TOT} = \sum_{i,j}^N V_2(r_{ij}) + \sum_{i,j,k}^N V_3(r_{ij}, r_{ik}, \theta_{ijk})$ Uses angle but transferability problem
- EAM and MEAM potential $V_{TOT} = \sum_i^N F_i \left(\sum_j \rho(r_{ij}) \right) + \frac{1}{2} \sum_{i,j}^N V_2(r_{ij})$ Uses electron density, for metallic systems
- Bond-order/Tersoff/Brenner $V_{ij}(r_{ij}) = V_{repulsive}(r_{ij}) + b_{ijk} V_{attractive}(r_{ij})$ Uses bond information, for covalent systems

These potentials lack charge information

- Fixed charge potentials: Coulomb-Buckingham
- Other FFs: ReaxFF, COMB, AMBER, CHARMM, OPLS etc.

$$F_i = m_i a_i, \quad F_i = -\nabla_i V, \quad -\frac{dV}{dr_i} = m_i \frac{d^2 r_i}{dt^2}$$



Berni Alder

JARVIS-FF



NIST IPR Interatomic Potentials Repository



Force-field (FF)

Search Elements

Speak or Type Element to search..

Scroll Down ▾

Advanced search

Elements

Ni-Al-

JARVIS-ID

H	
Li	Be
Na	Mg



Search

Downloads

Release notes

Installation guide

User guide

Known issues affecting users of GROMACS

Getting started

System preparation

Managing long simulations

Answers to frequently asked questions (FAQs)

Force fields in GROMACS

Molecular dynamics parameters (.mdp options)

Useful mdrun features

Getting good performance

Force fields in GROMACS

AMBER

AMBER (Assisted Model Building and Energy Refinement) refers to a suite of programs for the simulation of biomolecules and a package of molecular dynamics force fields.

GROMACS supports the following AMBER force fields natively:

- AMBER94
- AMBER96
- AMBER99
- AMBER99SB
- AMBER99SB-ILDN
- AMBER03
- AMBERGS

Information concerning the force field can be found using the following links:

- [AMBER Force Fields](#) - background about the AMBER force fields
- [AMBER Programs](#) - information about the AMBER suite of programs
- [ANTECHAMBER/GAFF](#) - Generalized Amber Force Field (GAFF) v.2 - parameters suitable for small molecules that are compatible with GROMACS

Statistical Ensembles

Real-world experiments often operate under **specific conditions** (constant temperature, pressure, volume, etc.)

Statistical ensembles (hypothetical copies) allow us to relate the multitude of possible microstates ((e.g., Avogadro's number, 10^{23}) to macroscopic thermodynamic quantities by **averaging over all microstates** according to certain probabilities

Ways to describe a system of particles with specific constraints and conditions:

- **Microcanonical Ensemble (NVE)**: Fixed N, V, E
- **Canonical Ensemble (NVT)**: Fixed N, V, T
- **Grand Canonical Ensemble (μ VT)**: Fixed μ, V, T
- **Isothermal-Isobaric Ensemble (NPT)**: Fixed N, P, T

Number of particles (N), Chemical potential (μ), Volume (V), Pressure (P), Temperature (T)

Overcoming Challenges

- **Hybrid Approaches:** Combining classical force fields with **quantum mechanical** methods (QM/MM) to accurately model systems where electronic effects are non-negligible.
- **Development of New Force Fields:** Ongoing research into more versatile and robust force fields that can handle a broader range of interactions and conditions such as **Machine Learning Force-Fields**.

Martin Karplus Facts



Martin Karplus

The Nobel Prize in Chemistry 2013

Born: 15 March 1930, Vienna, Austria

Affiliation at the time of the award: Université de Strasbourg, Strasbourg, France; Harvard University, Cambridge, MA, USA

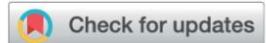
Prize motivation: “for the **development of multiscale models for complex chemical systems**”

Prize share: 1/3

© Nobel Media AB. Photo: A. Mahmoud

Digital Discovery

PAPER



Cite this: *Digital Discovery*, 2023, 2, 346

Received 12th September 2022
Accepted 12th January 2023

DOI: 10.1039/d2dd00096b

rsc.li/digitaldiscovery



[View Article Online](#)

[View Journal](#) | [View Issue](#)

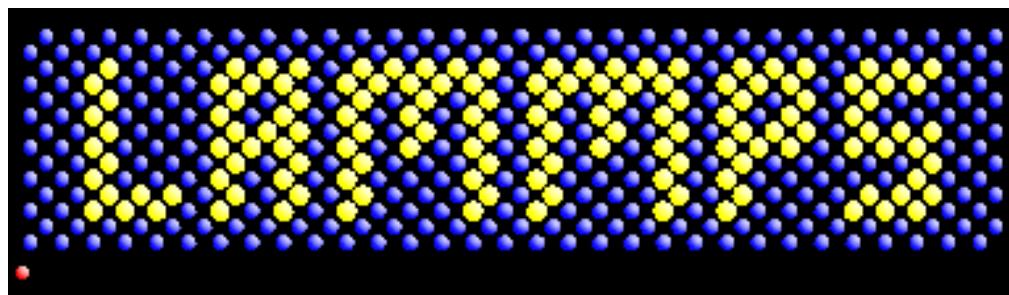
Unified graph neural network force-field for the periodic table: solid state applications

Kamal Choudhary, ^{ab} Brian DeCost, ^c Lily Major, ^{de} Keith Butler, ^e Jeyan Thiyagalingam ^e and Francesca Tavazza ^c

Classical force fields (FFs) based on machine learning (ML) methods show great potential for large scale simulations of solids. MLFFs have hitherto largely been designed and fitted for specific systems and are not usually transferable to chemistries beyond the specific training set. We develop a unified atomistic line graph neural network-based FF (ALIGNN-FF) that can model both structurally and chemically diverse solids with any combination of 89 elements from the periodic table. To train the ALIGNN-FF model, we use the JARVIS-DFT dataset which contains around 75 000 materials and 4 million energy-force entries, out of which 307 113 are used in the training. We demonstrate the applicability of this method for fast optimization of atomic structures in the crystallography open database and by predicting accurate crystal structures using a genetic algorithm for alloys.

Choosing the Right Software

- Popular MD software packages: LAMMPS, GROMACS, AMBER, CHARMM, OpenMM
- Choose based on available force-fields/implementations
- Same Newton's equation, different format for input/output files
- Reading documentation could be useful



The Amber Home Page

Tools for Molecular Simulations

AmberTools24 Amber24 Manuals Tutorials Force Fields Contacts History

Useful links:

Download Amber Installation Amber Citations GPU Support Updates Mailing Lists



What is Continuum Mechanics?

Key Areas of Study

- **Elasticity:** Behavior of materials that return to their original shape after deformation when external forces are removed.
- **Plasticity:** Behavior of materials that undergo permanent deformation under external forces.
- **Viscoelasticity:** Materials exhibit both viscous (material's resistance to flow) and elastic characteristics when undergoing deformation.

Governing Equations

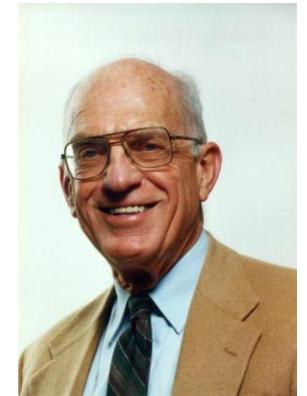
- **Conservation of Mass**
- **Conservation of Momentum**
- **Conservation of Energy**

Applications

- Predicting the effects of stress and strain on structures and materials.
- Designing materials and structures in civil, aerospace, automotive, and mechanical engineering.
- Understanding geological phenomena and biomaterial behaviors

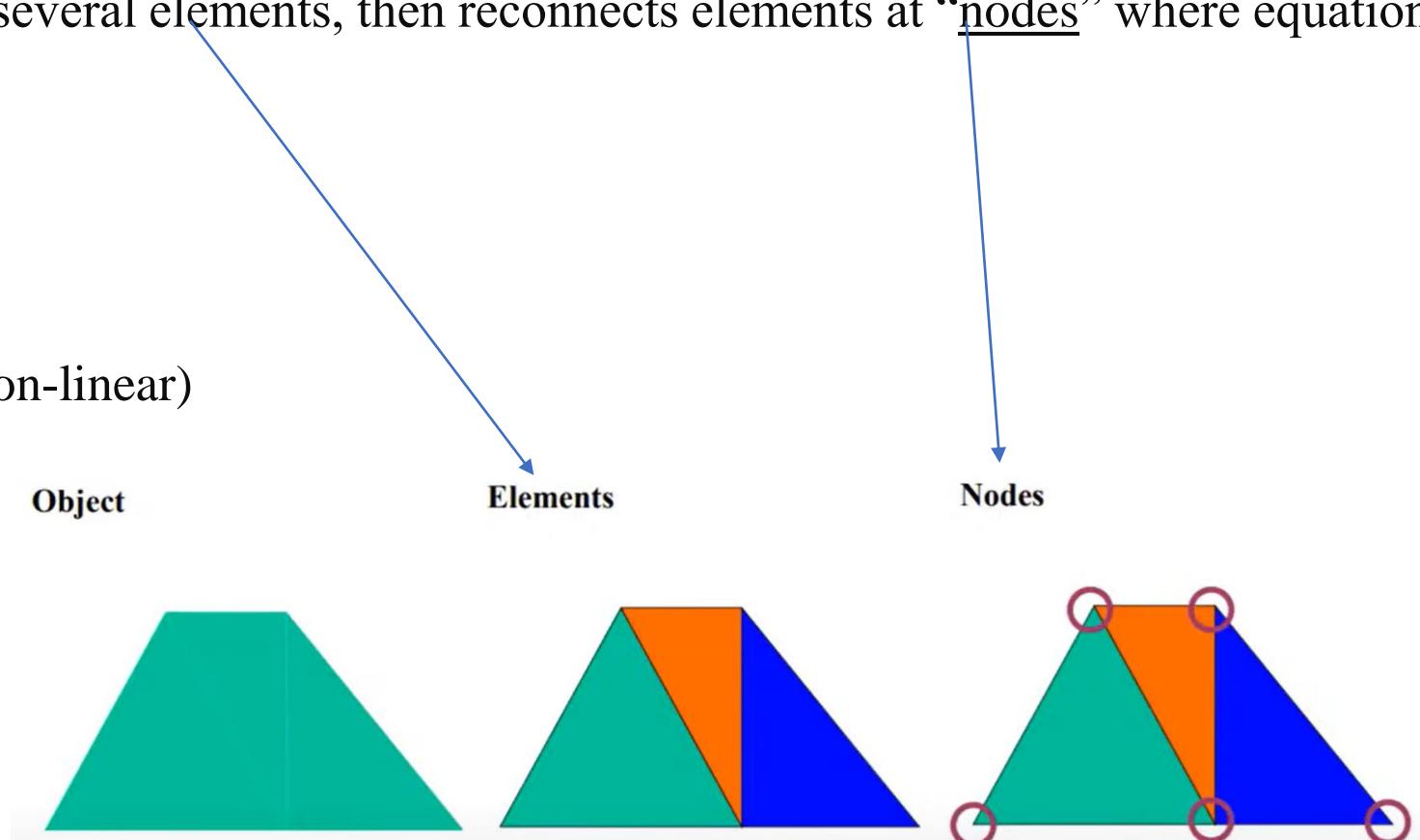
What is Finite Element Method (FEM)?

- Continuum: Infinite
- FEM: Finite (origin of the name FEM)
- FEM divides a structure into several elements (pieces of the structure)
- Then reconnects elements at “nodes” as if nodes were pins or drops of glue that hold elements together
- This process results in a set of simultaneous algebraic equations
- Finite Element Method (FEM) is a numerical technique for finding approximate solutions to boundary value problems for partial differential equations. It is used predominantly in engineering, especially for structural, thermal, and fluid analysis.
- The term FEM was first coined by Ray William Clough (“Legend of Earthquake Engineering”) in 1960.
- Most commercial FEM packages (such as ANSYS, Adina, Abaqus) originated in 1970.



Foundations of FEM

- Engineering phenomenon can be expressed by governing equations and boundary conditions
- FEM approximates PBEs into a set of simultaneous algebraic equations:
 $[K]\{u\} = \{F\}$, where K are material properties, u is the behavior, F is the action applied;
- **Behavior u is unknown and should be solved/analyzed/plotted; $\{u\} = [K]^{-1}\{F\}$**
- FEM divides a structure into several elements, then reconnects elements at “nodes” where equations are solved
- **Applications:**
 - Mechanical/Aerospace
 - Civil/Automotive
 - Structural/Stress analysis
(Static/Dynamic, Linear/Non-linear)
 - Fluid flow
 - Heat transfer
 - Electromagnetic fields
 - Soil mechanics
 - Acoustics
 - Biomechanics



Foundations of FEM

Analysis	Property [K]	Behavior {u}	Action {F}
Elastic	Stiffness	Displacement	Force
Thermal	Conductivity	Temperature	Heat source
Fluid	Viscosity	Velocity	Body force
Electrostatic	Permittivity	Electric potential	Charge

- Behavior \underline{u} is unknown
- Knowing appropriate material properties is important; experimental/DFT/MD... data
- We will try to run one/a few examples of each of the above in the next module

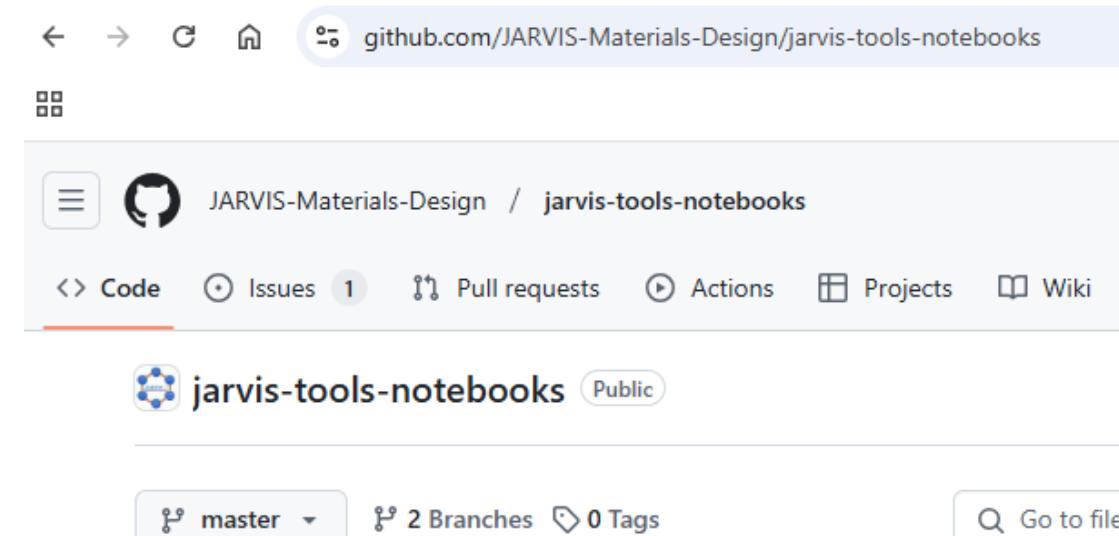
$$[\mathbf{K}] \{\mathbf{u}\} = \{\mathbf{F}\} \quad \Rightarrow \quad \{\mathbf{u}\} = [\mathbf{K}]^{-1} \{\mathbf{F}\}$$

↑ ↑ ↑

Property **Behavior** **Action**

Day 1 Hands-on Session

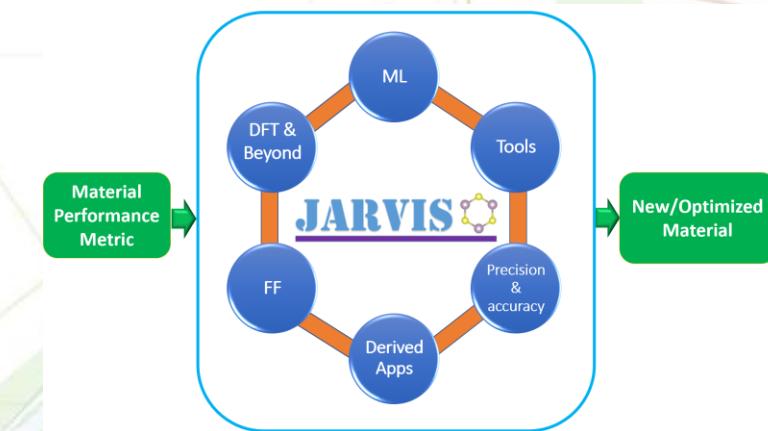
1. Go to website: <https://jarvis.nist.gov/>
2. Make an account, Go to one of the databases/interactive apps
3. On JARVIS website, click on Tutorials >> JARVIS-Tools-Notebooks link
4. B.1 HPC basics
5. B.2 Python beginners notebook
6. B.4 Silicon atomic structure and analysis example
7. ES. 1 Analyzing JARVIS-DFT data
8. ES.29 GPAW Colab
9. FF.1 JARVIS-LAMMPS



Materials Informatics: Fingerprints, Graph Neural Networks, and Transformers Based Models for Materials Design



Kamal Choudhary
Staff Scientist
NIST, Gaithersburg, MD, USA
Jan 20-25, 2025



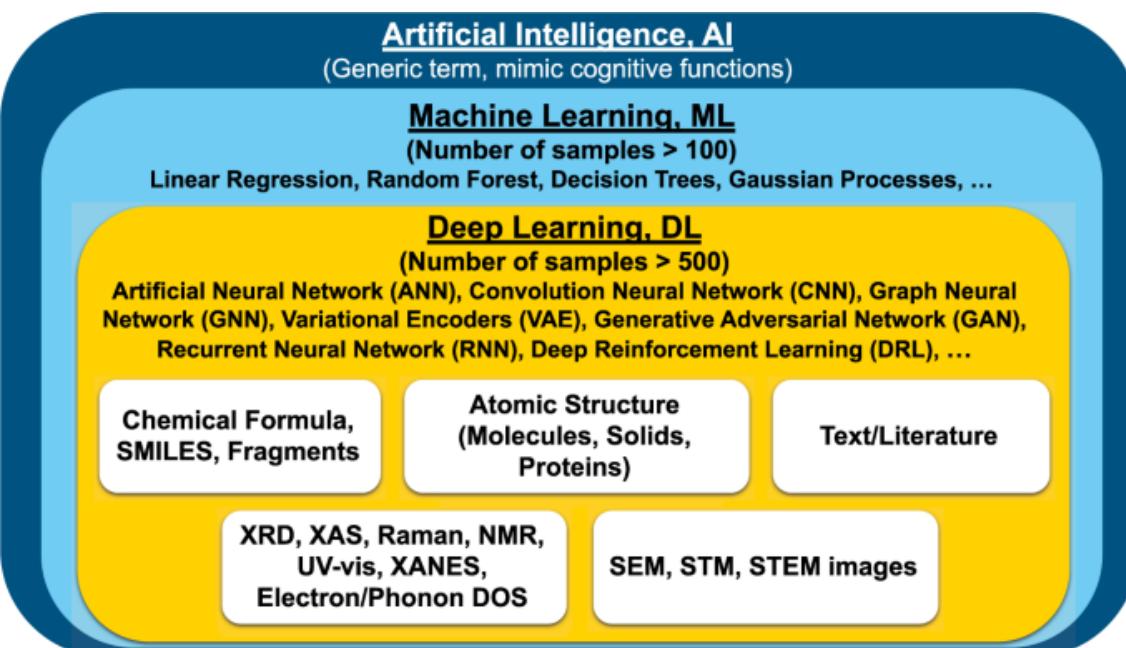
<https://jarvis.nist.gov>

Day 3 Schedule

- Fundamentals of AI/ML
- Types of ML
- AI/ML ready Materials Databases
- Fingerprints/Descriptors, Conventional ML algorithms for Materials

What is AI/ML/DL?

- Artificial intelligence (AI) is a broad field of computer science that's concerned with building intelligent machines capable of performing tasks typically requiring human intelligence.
- Machine learning (ML) is a type of artificial intelligence (AI) that allows computers to learn without being explicitly programmed.
- Machines don't really know Materials/Mechanical Engineering etc., they deal with/know numbers.
- If multi-scale models guide experiments, ML methods can guide multi-scale modeling methods



[nature](#) > [npj computational materials](#) > [review articles](#) > [article](#)

Review Article | [Open access](#) | Published: 05 April 2022

Recent advances and applications of deep learning methods in materials science

[Kamal Choudhary](#)✉, [Brian DeCost](#), [Chi Chen](#), [Anubhav Jain](#), [Francesca Tavazza](#), [Ryan Cohn](#), [Cheol Woo Park](#), [Alok Choudhary](#), [Ankit Agrawal](#), [Simon J. L. Billinge](#), [Elizabeth Holm](#), [Shyue Ping Ong](#) & [Chris Wolverton](#)

[npj Computational Materials](#) **8**, Article number: 59 (2022) | [Cite this article](#)

71k Accesses | **37** Altmetric | [Metrics](#)

<https://www.nature.com/articles/s41524-022-00734-6>

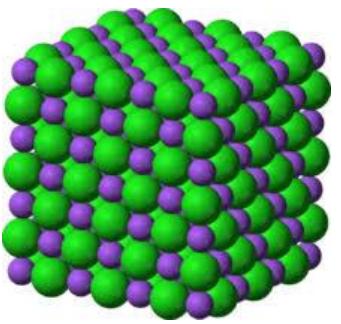
<https://www.deeplearningbook.org/>

Motivation

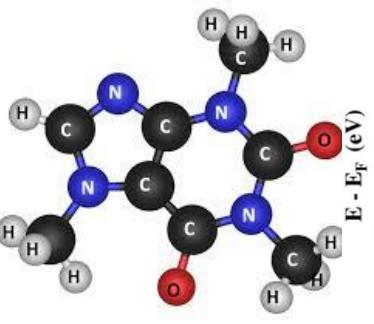
Materials Genome Initiative 2011, \$400 million



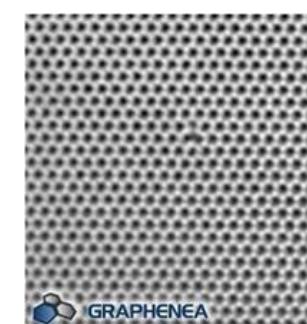
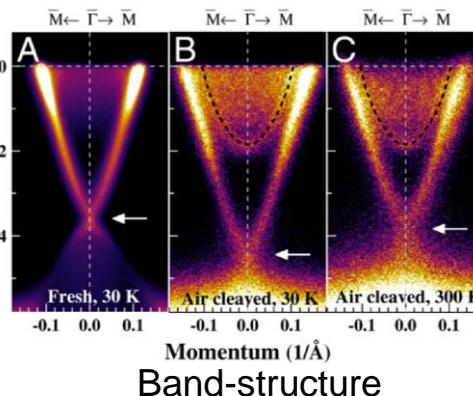
US CHIPS Act 2022, \$52 billion



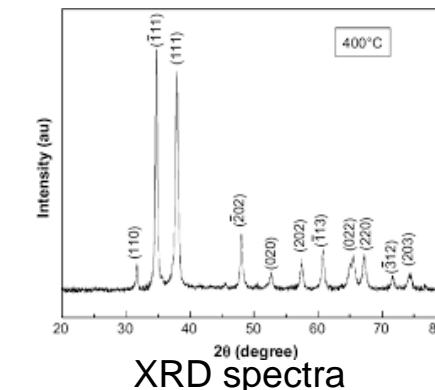
Crystals



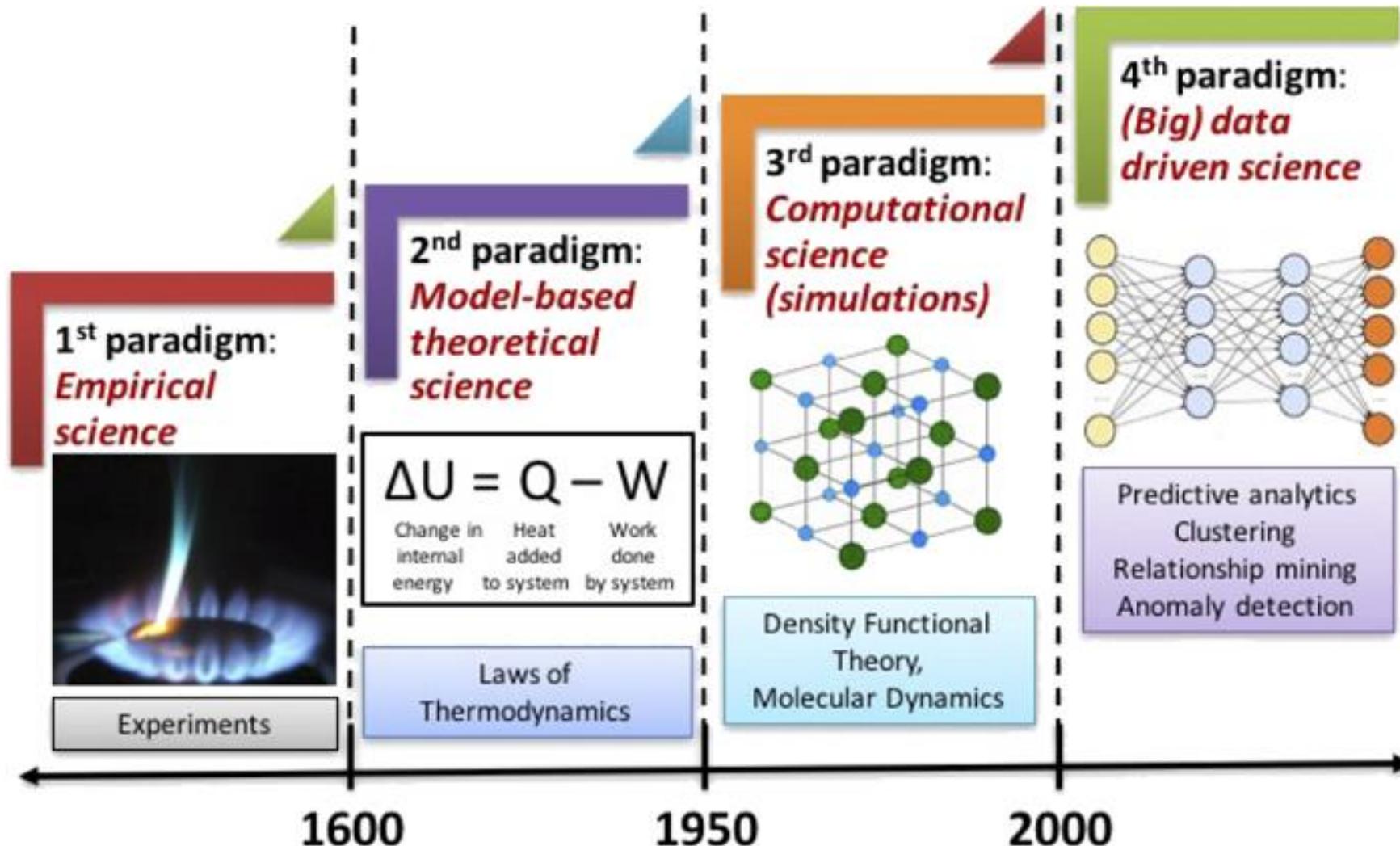
Molecules

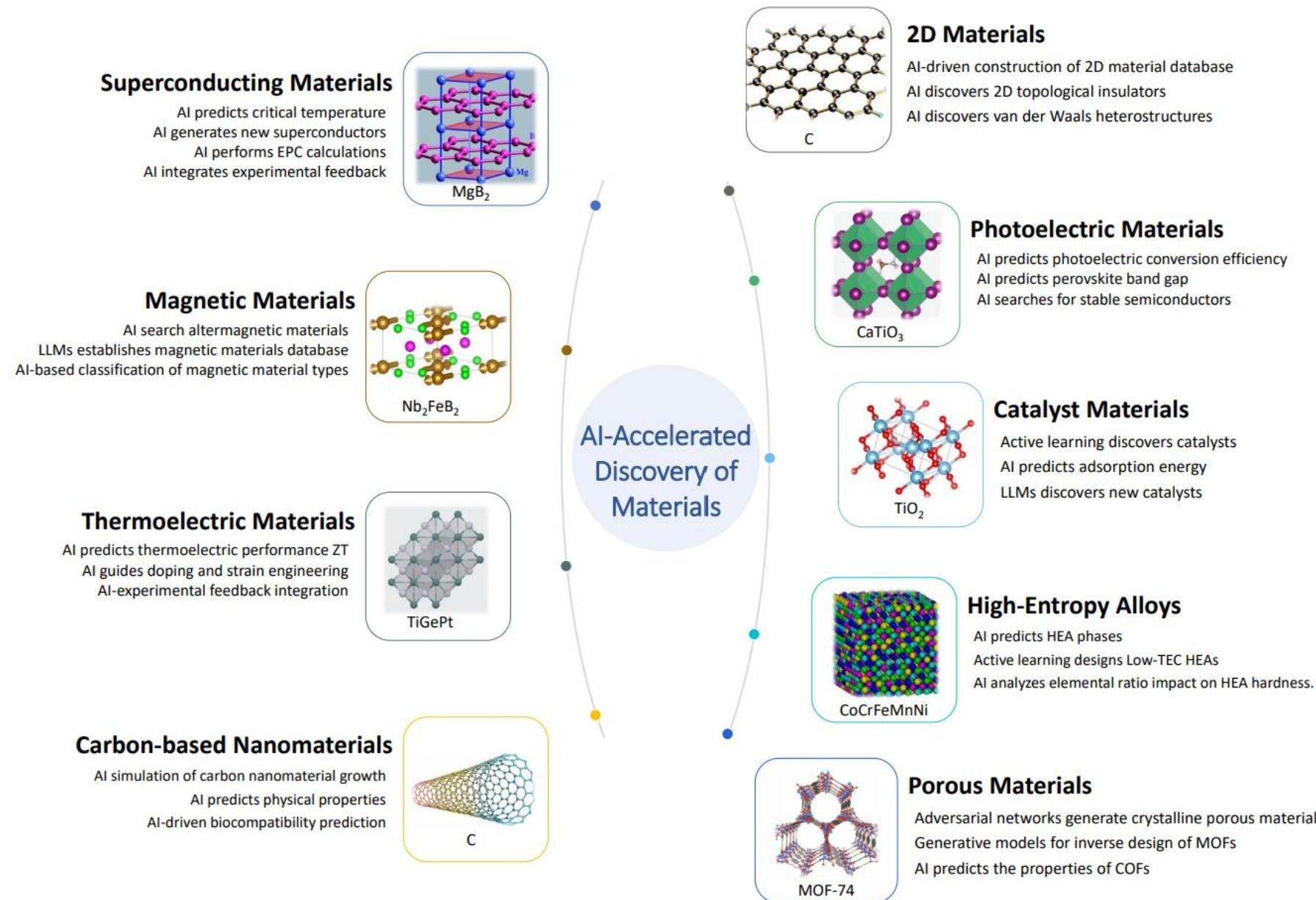


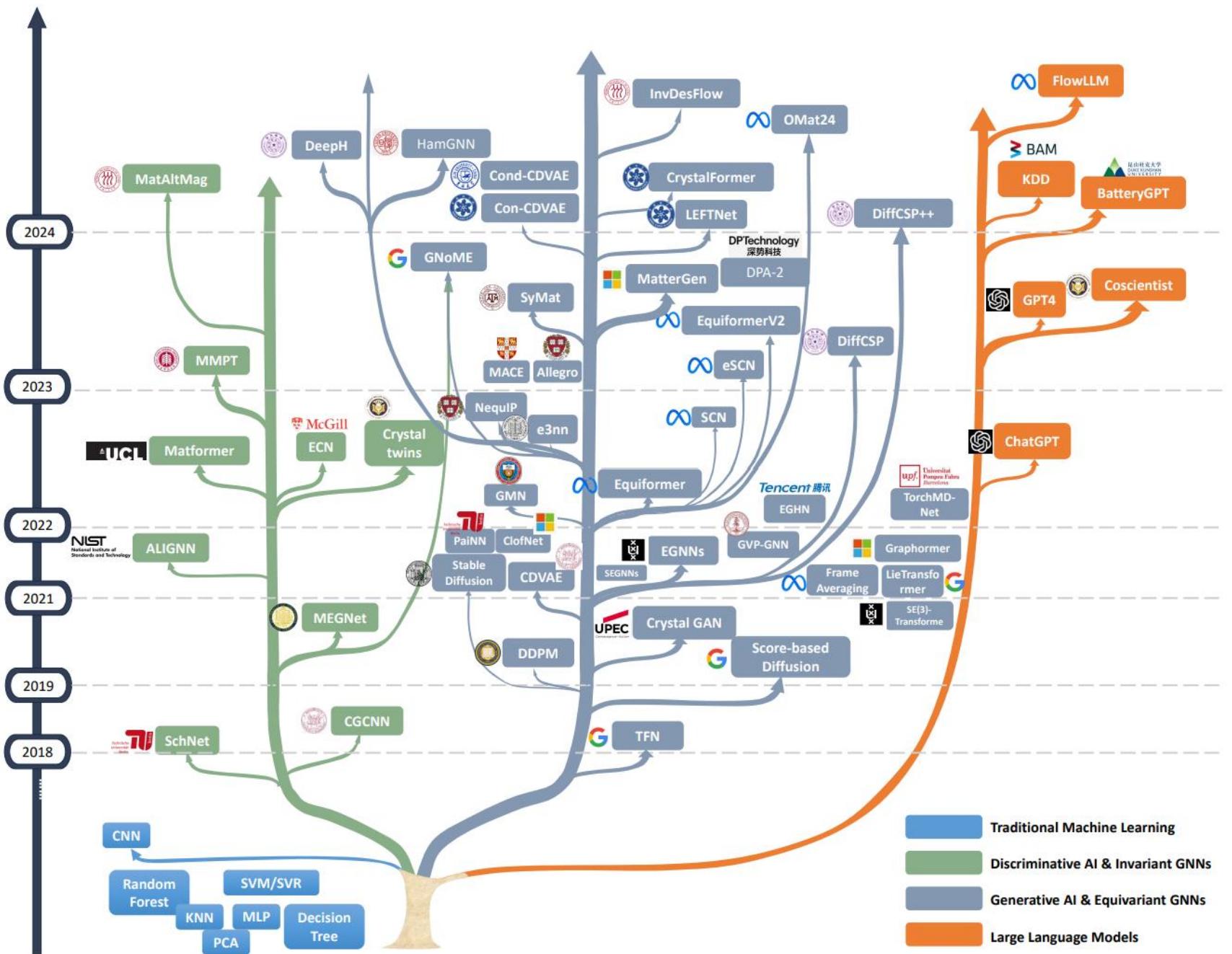
TEM images

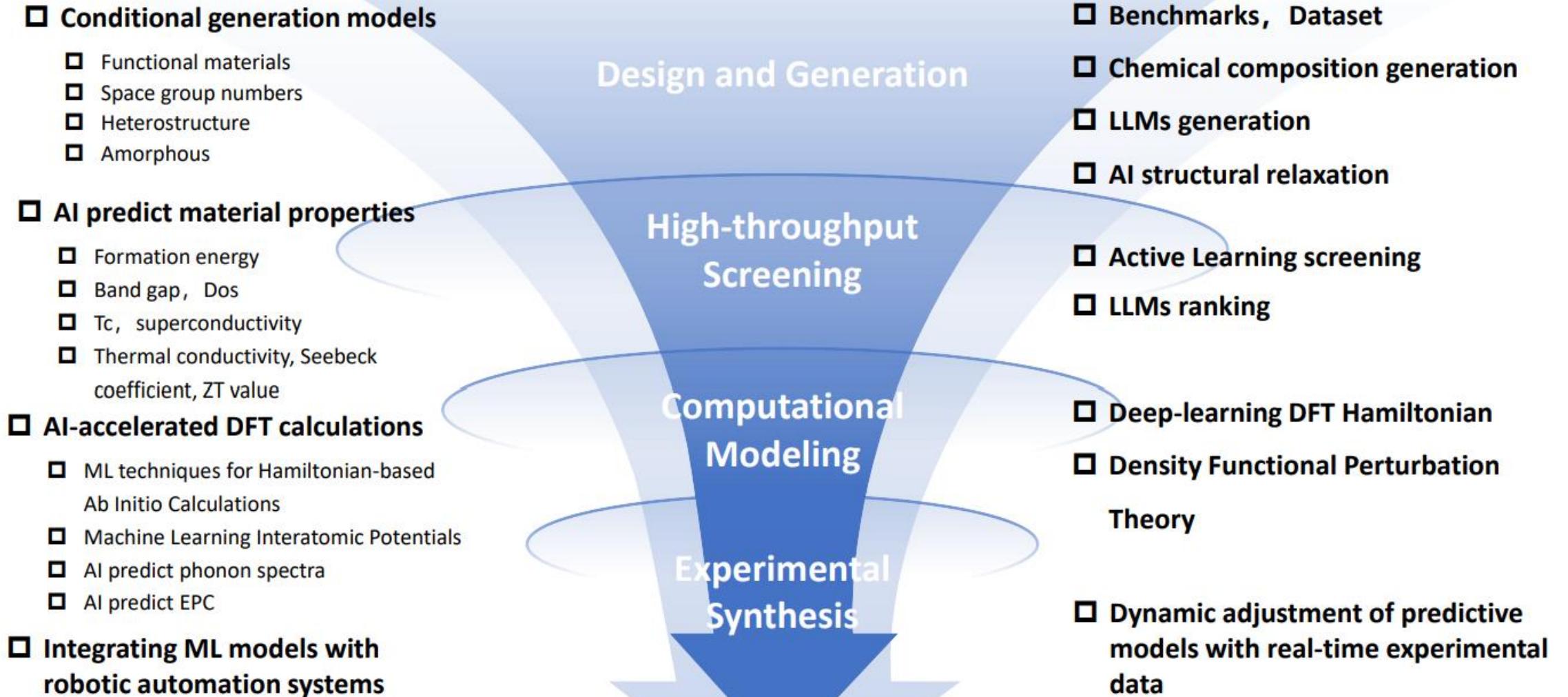


AI/ML as a Fourth Paradigm

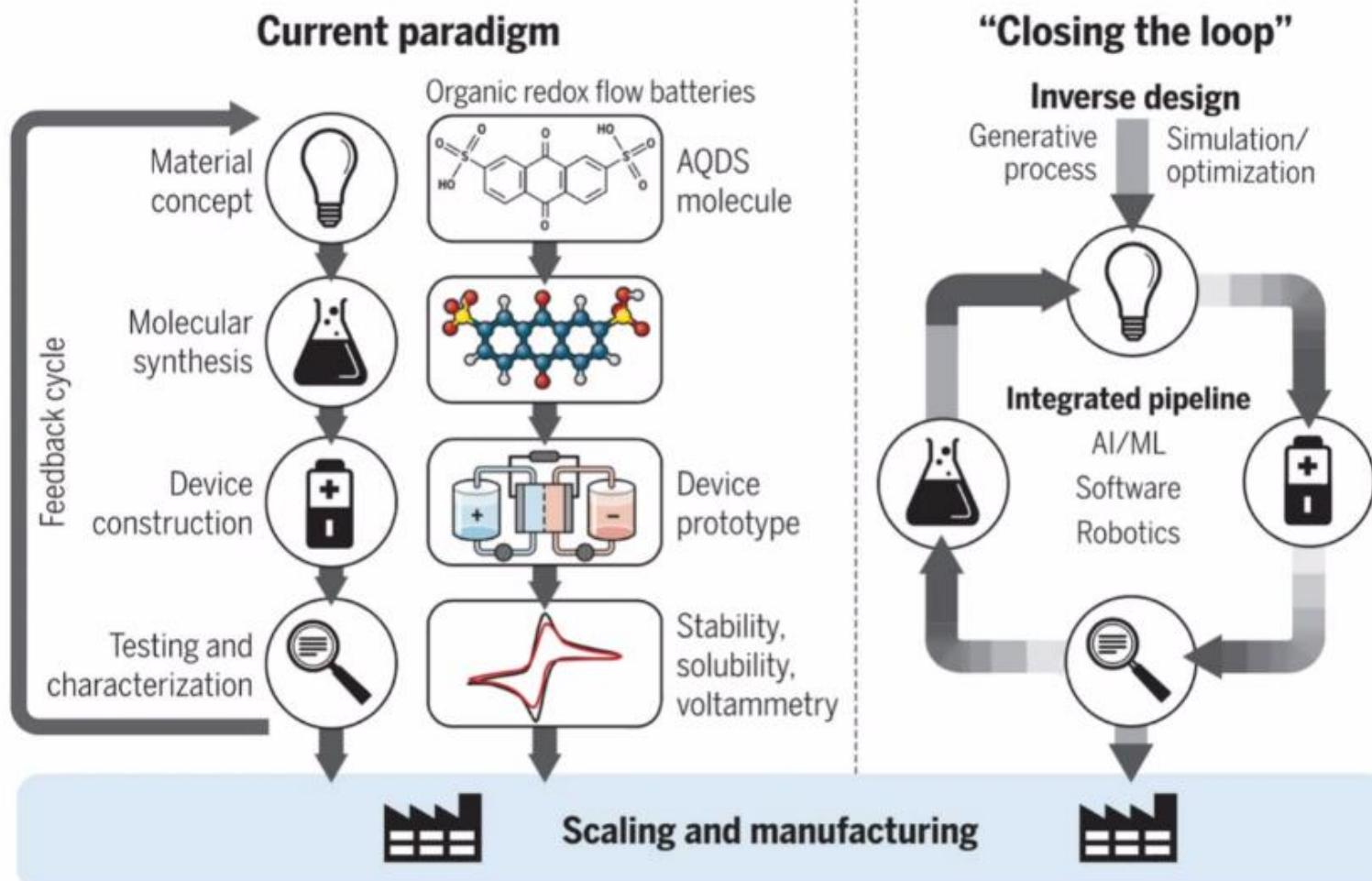








CLOSING THE LOOP WITH SELF-DRIVING LABS



Sánchez-Lengeling and Aspuru-Guzik, **Science**, 361, 360, (2018).

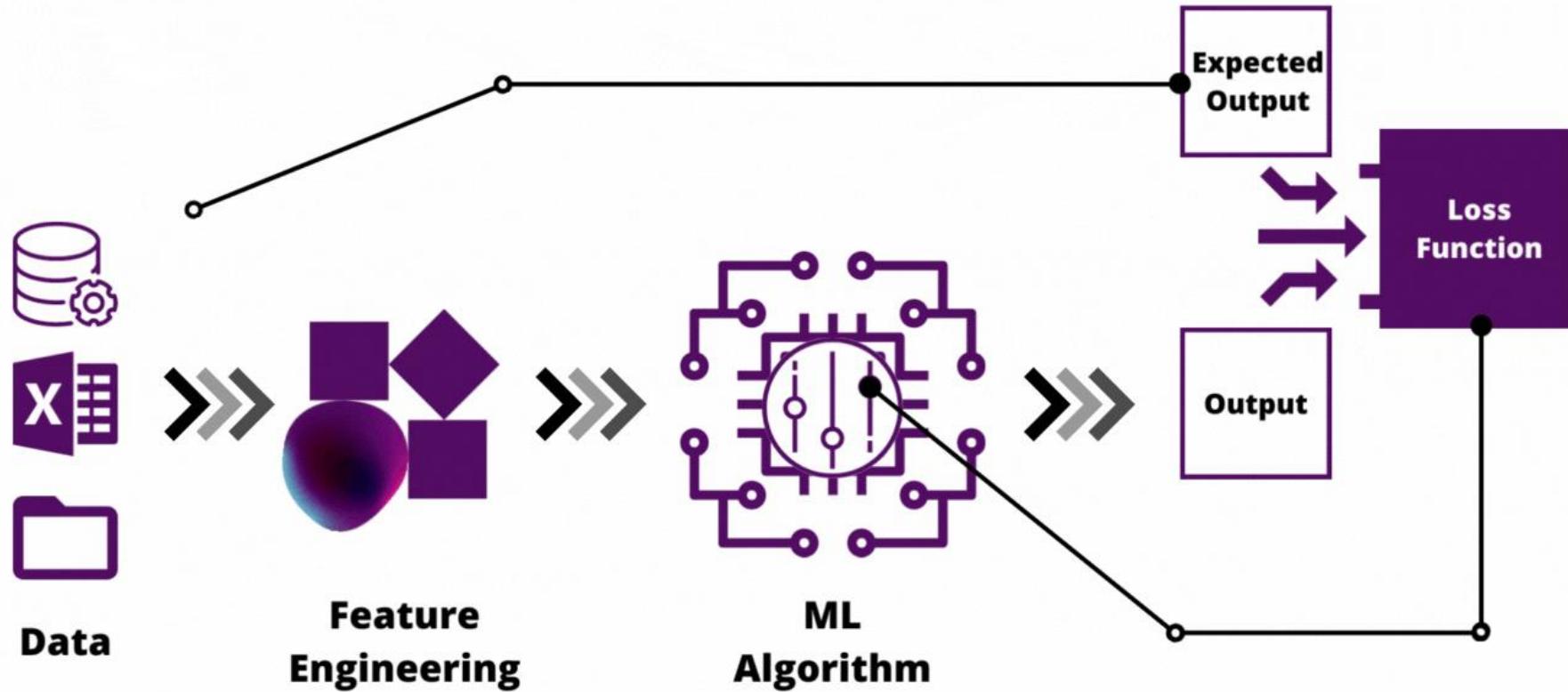
Types of ML

- **Supervised Learning:**
 - Involves training a model on a labeled dataset, where the desired output is known.
 - Common algorithms: Linear Regression, Support Vector Machines (SVM), Neural Networks.
 - Example: Predicting material strength based on composition/structure and processing conditions.
 - Types:
 - Classification: categorical output (e.g. cat vs dog classification of images)
 - Regression: continuous output (e.g. weight of a cat)
- **Unsupervised Learning:**
 - Involves training a model on an unlabeled dataset, where the desired output is unknown.
 - Common algorithms: K-means Clustering, Principal Component Analysis (PCA).
 - Example: Identifying patterns in manufacturing defects without prior knowledge of defect types.
- **Reinforcement Learning:**
 - Involves training a model to make a sequence of decisions by rewarding desirable actions.
 - Common algorithms: Q-learning, Deep Q-Networks (DQN).
 - Example: Optimizing the control parameters of a robotic arm in real-time.

Applications of ML in Materials Engineering

- **Material Property Prediction:**
 - Use ML models to predict properties such as yield strength, elasticity, and thermal conductivity.
 - Example: Using neural networks to predict the stress-strain behavior of novel alloys.
- **Design Optimization:**
 - Apply ML algorithms to optimize design parameters for mechanical components.
 - Example: Using genetic algorithms to optimize the shape of an airfoil for maximum lift-to-drag ratio.
- **Fault Detection and Diagnosis:**
 - Implement ML models to detect and diagnose faults in mechanical systems.
 - Example: Using SVM to classify different types of wear in bearings based on vibration data.
- **Predictive Maintenance:**
 - Utilize ML for predictive maintenance by analyzing historical data to predict future failures.
 - Example: Using time-series analysis to predict the remaining useful life of machinery.
- **Process Control and Automation:**
 - Develop intelligent control systems using reinforcement learning.
 - Example: Optimizing the welding process parameters in real-time to ensure consistent quality.

General Procedure of ML



1. Data (Define problem,
Data collection &
Visualize)

2. Model
selection

3. Training

4. Evaluation &
Deployment

Define the problem

Determine objective: Clearly define the problem you want to solve and the goals you want to achieve with machine learning.

- **Understand the Problem:** Identify the specific issue or opportunity that machine learning can address.
- **Define Objectives:** Set clear, measurable objectives for the project (e.g., improve prediction accuracy, reduce operational costs).
- **Scope the Project:** Determine the scope, including the data to be used, the timeline, and the resources required.

Determine type of task:

- Supervised,
- Unsupervised,
- Reinforcement learning,
- A combination?



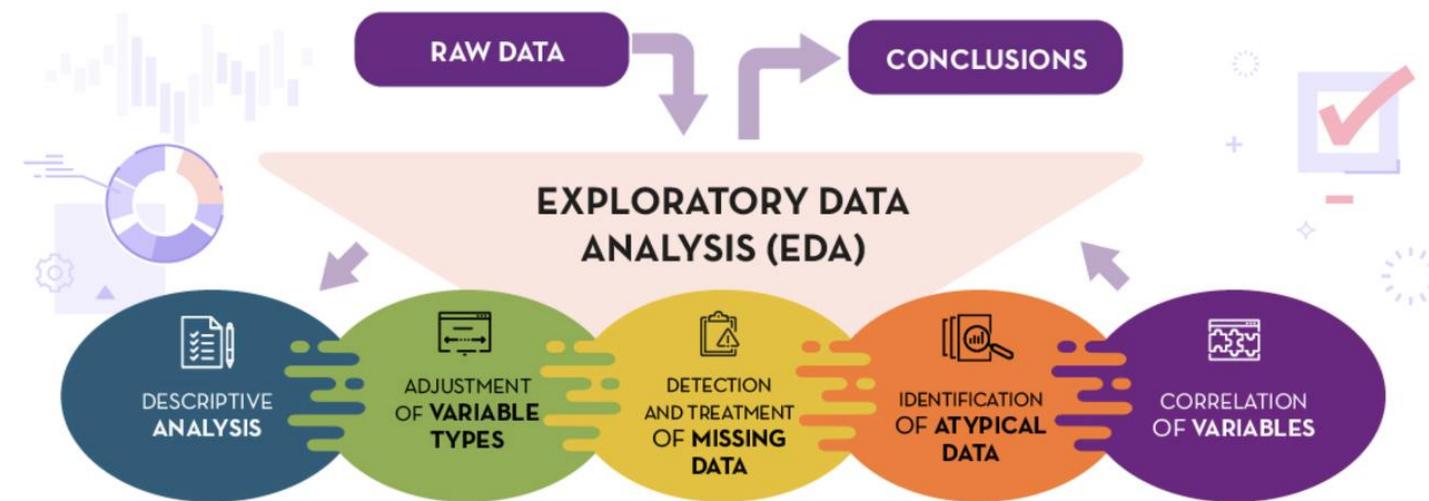
Source: pinterest

Data collection

- **Objective:** Gather the data needed to train and evaluate the machine learning model, Garbage In, Garbage Out (GIGO).
- **Identify Data Sources:** Determine where the data will come from (e.g., sensors, databases, public datasets).
- **Collect Data:** Gather the data, ensuring it is relevant and sufficient for the problem at hand.
- **Data Storage:** Store the collected data in a structured format suitable for analysis.
 - Atomic structure (RCSB PDB, JARVIS-DFT, Materials Project, OQMD, AFLOW),
 - Coarse-grain: CGProt, MSCG , ENM-DB,
 - Phase field dataset: PFHub, CALPHAD-DB,
 - FigShare, Kaggle, Zenodo, GitHub & many more ...

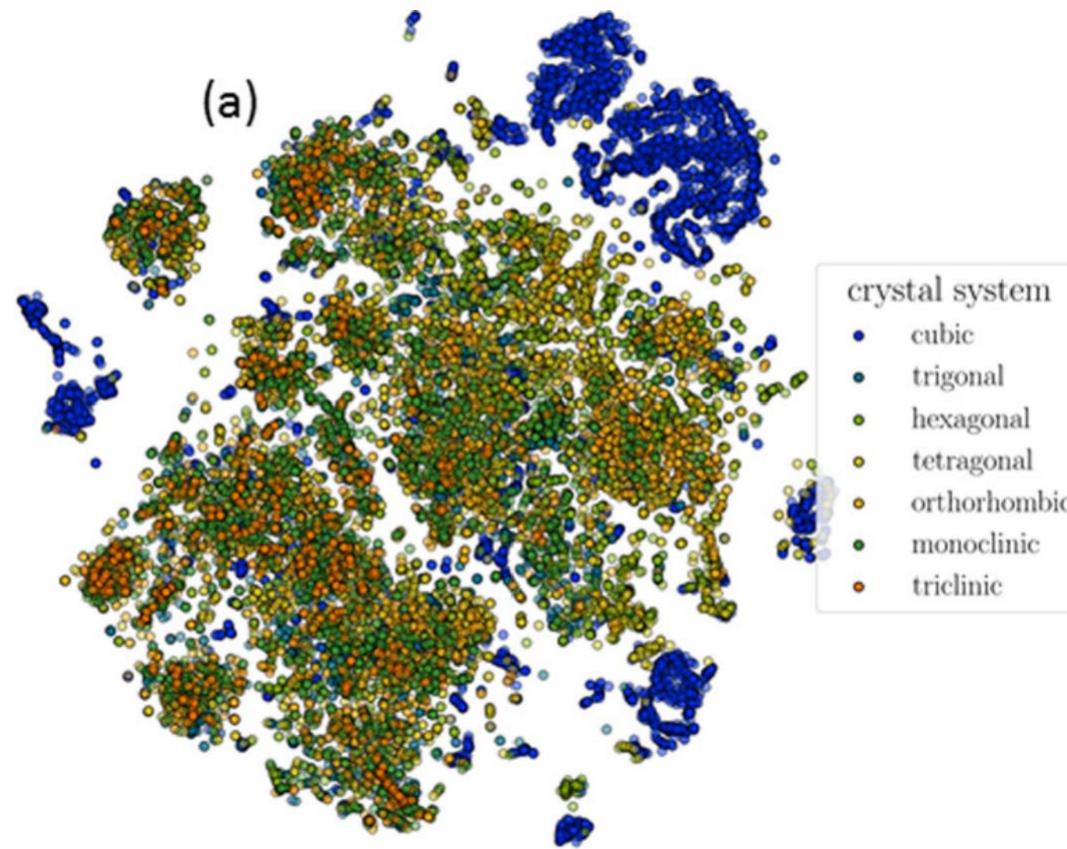
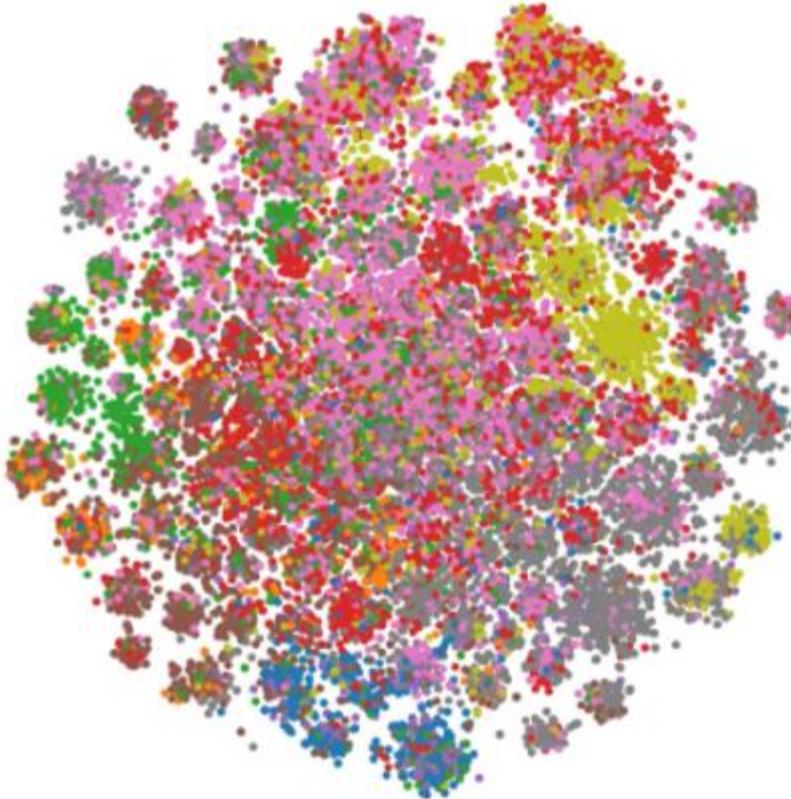
Exploratory Data Analysis (EDA)

- **Descriptive Statistics:** Calculate summary statistics to understand the distribution and central tendencies of the data.
- **Data Visualization:** Use plots (e.g., histograms, scatter plots, box plots) to visualize data distributions and relationships.
- **Correlation Analysis:** Identify correlations between features to understand their relationships and potential multi-collinearity issues.



PCA/t-SNE/UMAP/Histograms

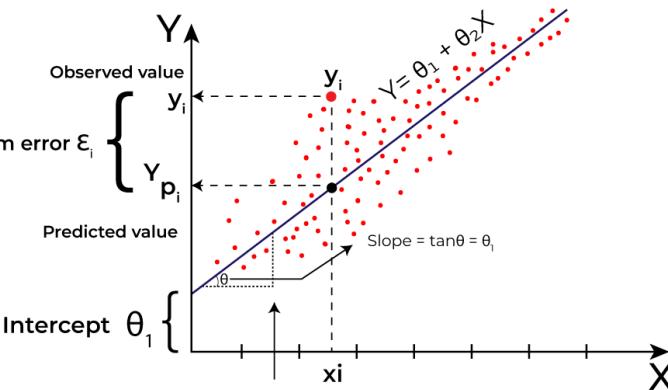
(a) arXiv: cond-mat



UMAP (Uniform Approximation and Projection)
t-SNE (t-distributed stochastic neighbor embedding)
PCA (Principal component analysis)
... (using Scikit-Learn package, CFID, ChemNLP)

ML Models

Linear Regression



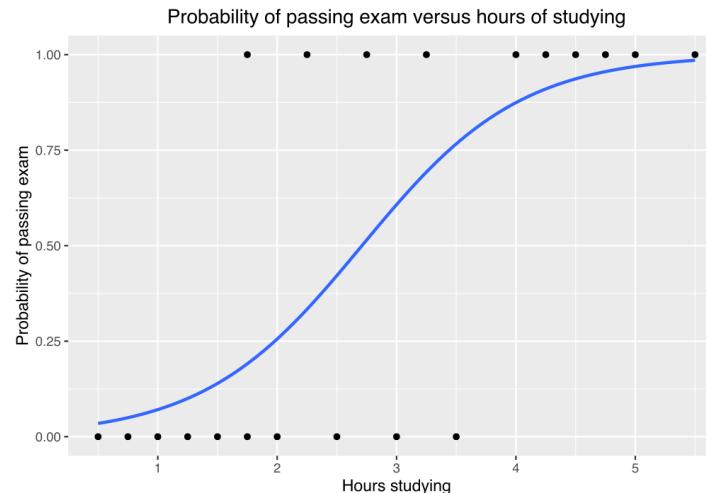
$$y = \beta_0 + \beta_1 x$$

- y is the dependent variable.
- x is the independent variable.
- β_0 is the y-intercept of the line.
- β_1 is the slope of the line.
- To estimate β_0 and β_1 , we minimize the sum of the squared residuals:

$$\text{RSS} = \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i))^2$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

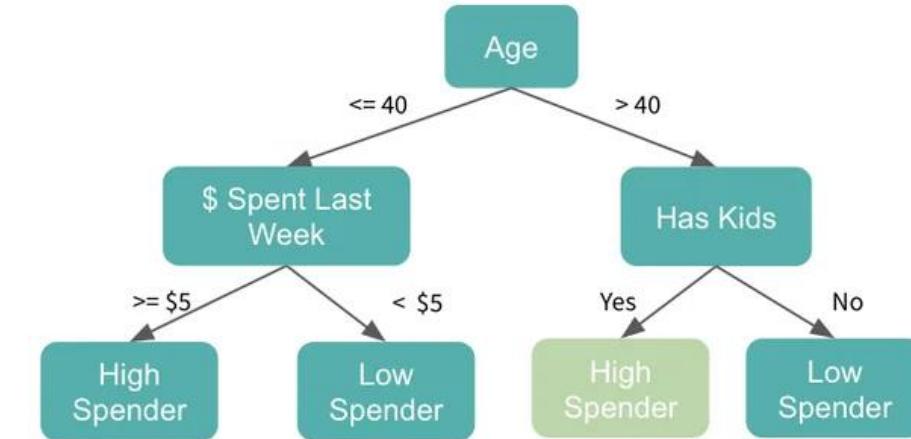
Logistic Regression



- Log-odds of an event as a linear combination of one or more independent variables
- Used for predicting a binary or categorical dependent variable.

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

Tree based methods



Tree-based models use a series of if-then rules to generate predictions from one or more decision trees: RandomForest, GBM

Gini Impurity:

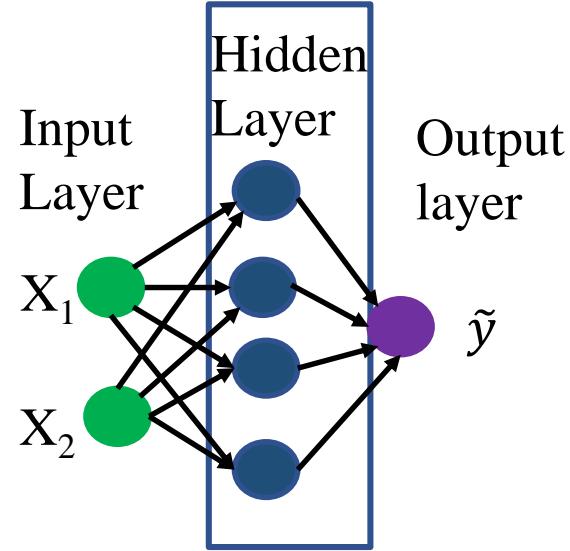
$$Gini = 1 - \sum_{i=1}^n p_i^2$$

Where p_i is the proportion of samples belonging to class i in the node.

$$Entropy = - \sum_{i=1}^n p_i \log_2(p_i)$$

ML Models

Standard NN



1) Forward propagation

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

$$a^{[1]} = \sigma(z^{[1]}); \quad a^{[0]} = X$$

2) Cost, $J(W, b) = f(y - \hat{y})$

3) Gradient descent (∇J):
minimize cost with W, b

4) Backpropagation:
chain rule to get, $\frac{\partial J}{\partial W}$

ConvolutionNN

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

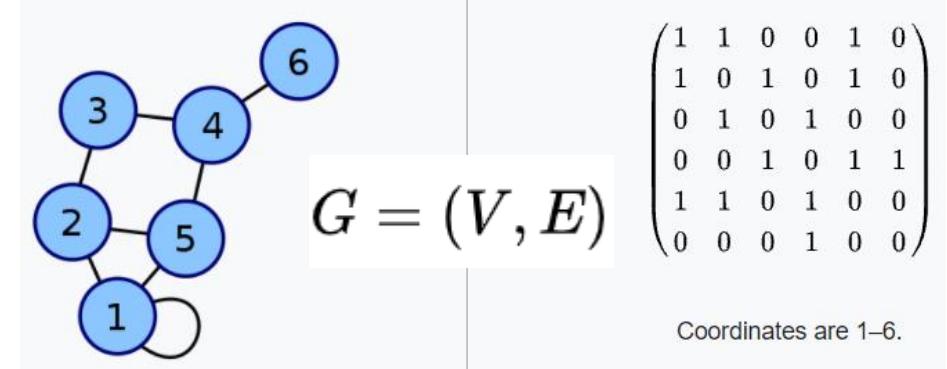
4		

Convolved Feature

1	0	1
0	1	0
1	0	1

- 1) Convolution:
element-wise multiplication & sum
- 2) Pool: Max, Average, Sum
- 3) Fully Connected: Standard NN
Shared weights (Learnable filters),
regularized version of NNs

GraphConvNN



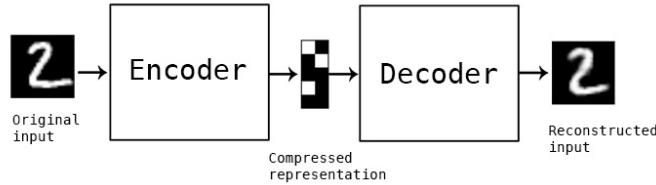
Types: un/weighted, un/directed, line, Hetero/Homogenous, Multigraph

- 1) Adjacency matrix, $N \times N$ (N : #nodes),
- 2) D: degree of node
- 3) Update node representation using message passing, GPU efficient
- 4) Update equation is local, neighborhood of a node only, independent of graph size

$$h_i^{\ell+1} = f(h_i^\ell, \{h_j^\ell\}_{j \in \mathcal{N}_i})$$

ML Models

AutoEncoders



Used for Dimensionality reduction or feature learning:

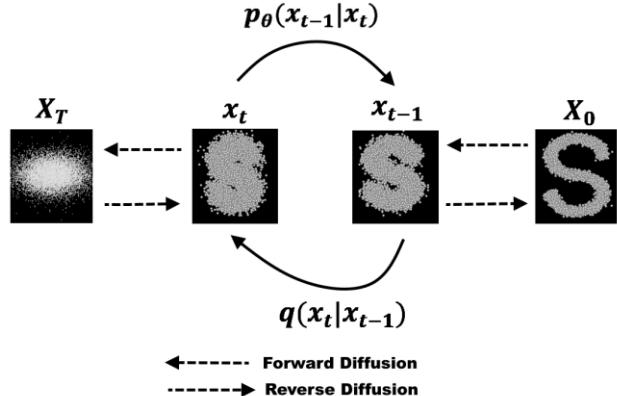
- 1) Encoding functions: x to a latent dimension z
 $\mathbf{z} = \sigma(\mathbf{Wx} + \mathbf{b})$

- 2) Decoding functions: z to x
 $\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}')$

3) Loss function

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{x}') &= \|\mathbf{x} - \mathbf{x}'\|^2 \\ &\quad \| \mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{Wx} + \mathbf{b})) + \mathbf{b}') \|^2 \end{aligned}$$

Diffusion Models



Reverse diffusion process, starting from random noise

1) Forward diffusion: gradually adds noise over time steps

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) := \prod_{t=1}^T \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

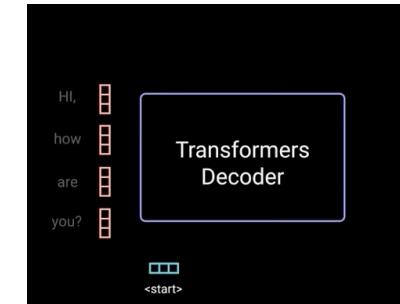
2) Reverse Diffusion Process:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

3) Training Objective and sampling:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} [\|\epsilon - \epsilon \theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2]$$

Transformers



- 1) Self-attention: calculates the relevance between each pair of input tokens.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- 2) Multi-head attention: focus on different parts of the input sequence simultaneously

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- 4) Positional Encoding: instead of RNN, use PE to convey the order of elements in the sequence

$$\text{PE}_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$\text{PE}_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

Model Training

- **Split Data:** Divide the data into training and testing (or validation) sets, because it's hard to generate new data for testing models; usually 80 % training, 20 % testing data
- **Cross-validation:** Perform train-test split multiple times (say 5 or 10 times) to avoid data-split bias
- **Train Model:** Use the training set to train the model, adjusting parameters as needed.
- **Hyperparameter Tuning:** Optimize hyperparameters using techniques such as grid search or random search.

scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Install User Guide API Examples Community More

Section Navigation

- sklearn
- sklearn.base
- sklearn.calibration
- sklearn.cluster
- sklearn.compose
- sklearn.covariance
- sklearn.cross_decomposition
- sklearn.datasets
- sklearn.decomposition
- sklearn.discriminant_analysis
- sklearn.dummy
- sklearn.ensemble
- sklearn.exceptions
- sklearn.experimental
- sklearn.feature_extraction

train_test_split

`sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)` [source]

Split arrays or matrices into random train and test subsets.

Quick utility that wraps input validation, `next(ShuffleSplit().split(X, y))`, and application to input data into a single call for splitting (and optionally subsampling) data into a one-liner.

Read more in the [User Guide](#).

Parameters:

***arrays : sequence of indexables with same length / shape[0]**
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : float or int, default=None
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://scikit-learn.org/stable/modules/cross_validation.html

scikit-learn.org/stable/modules/cross_validation.html

Install User Guide API Examples Community More

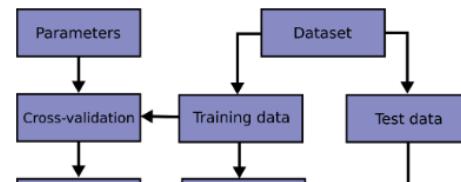
Section Navigation

- 1. Supervised learning
- 2. Unsupervised learning
- 3. Model selection and evaluation
 - 3.1. Cross-validation: evaluating estimator performance
 - 3.2. Tuning the hyper-parameters of an estimator
 - 3.3. Tuning the decision threshold for class prediction
 - 3.4. Metrics and scoring: quantifying the quality of predictions
 - 3.5. Validation curves: plotting scores to evaluate models
- 4. Inspection
- 5. Visualizations

3.1. Cross-validation: evaluating estimator performance

Learning the parameters of a prediction function and testing it on the same data is a methodologic mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **overfitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a **test set** `X_test, y_test`. Note that the word "experiment" is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally. Here is a flowchart of typical cross validation workflow in model training. The best parameters can be determined by [grid search](#) techniques.

```
graph TD; Parameters[Parameters] --> CV[Cross-validation]; Dataset[Dataset] --> CV; CV --> TrainingData[Training data]; CV --> TestData[Test data]; TrainingData --> CV; TestData --> CV;
```



Objective/Loss Function Optimization & Gradient Descent

- Gradient descent is a fundamental optimization algorithm used for minimizing the cost function in machine learning and deep learning models.
- Types of GD:
 - Batch gradient descent: cost function is computed using the entire training dataset
 - Mini-batch gradient descent: computes the gradient using a small random subset (mini-batch) of the training data at each iteration
 - Stochastic (S)GD: the gradient of the cost function is computed using a single training example at each iteration
 - Gradient descent with Momentum: accumulates a velocity vector in directions of persistent reduction in the cost function and uses it to update the parameters
 - Adagrad (Adaptive Gradient Algorithm): adapts the learning rate for each parameter based on past gradients, making larger updates for infrequent parameters and smaller updates for frequent parameters
 - RMSprop (Root Mean Square Propagation): adaptive learning rate method that addresses Adagrad's diminishing learning rate issue
 - Adam (Adaptive Moment Estimation): Adam combines the ideas of momentum and RMSprop by maintaining a moving average of both the gradient and its square

Model Evaluation

- **Performance Metrics:** Evaluate the model using appropriate metrics:
 - Classification: accuracy, precision, recall, F1-score
 - Regression: R^2 , mean absolute error, root mean squared error.
- **Baseline/Random guessing model:**
 - For classification ($1/N$; N : number of classes),
 - For Regression: take average of training samples and use it as predictions for the test set (average guessing model); MAD/MAE should be > 1
- **Cross-Validation:** Use cross-validation to ensure the model generalizes well to unseen data.
- **Compare/Benchmark Results:** Compare the performance of the model to baseline models or previous iterations.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

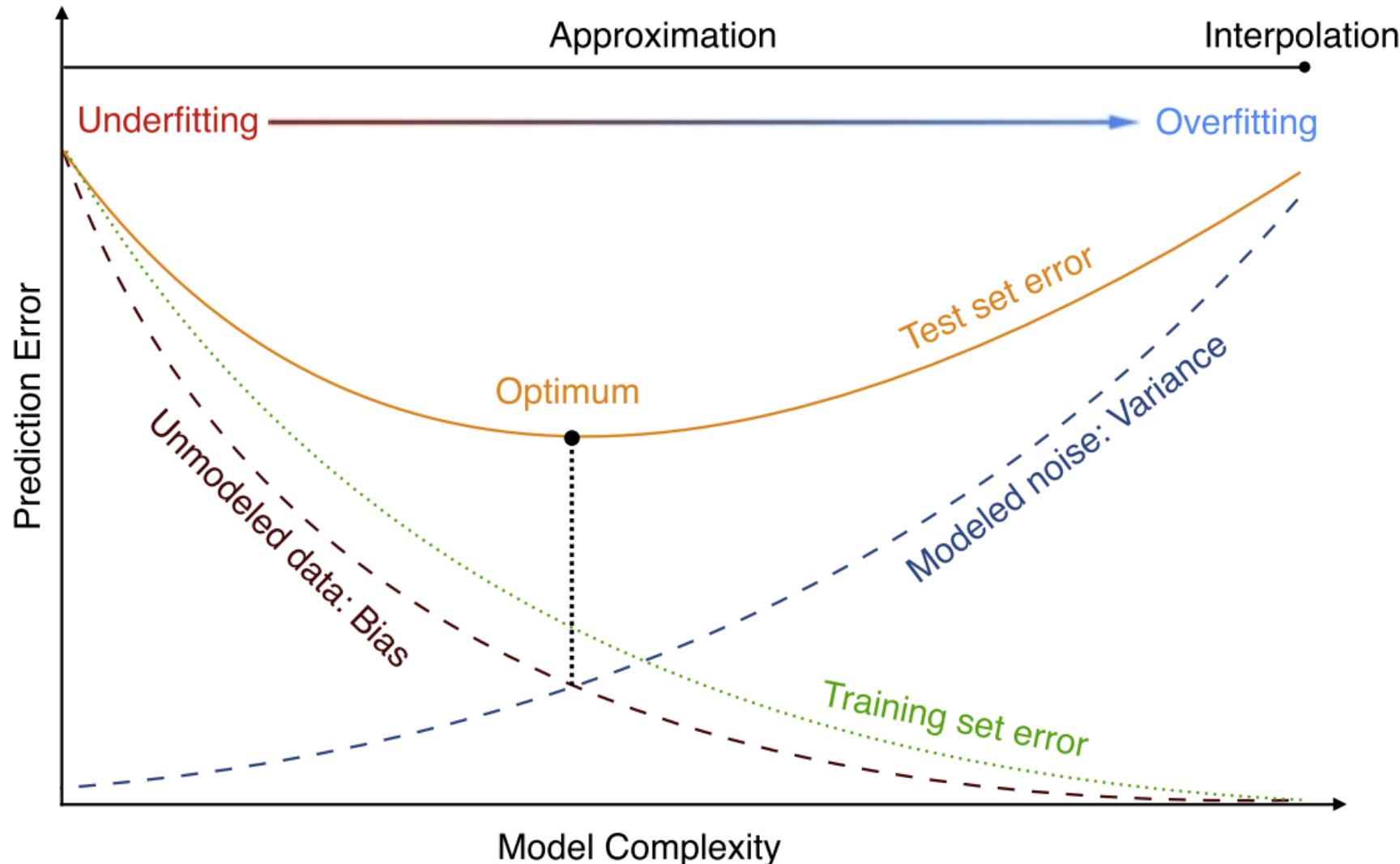
Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

$$R = \frac{\sum_{i=1}^N (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^N (y_i - \bar{y})^2 \sum_{i=1}^N (\hat{y}_i - \bar{\hat{y}})^2}}$$

$$MAE = \bar{e} = \frac{1}{N} \sum_N |y - \hat{y}|$$

$$RMSE = \sqrt{\frac{1}{N} \sum_N (y - \hat{y})^2}$$

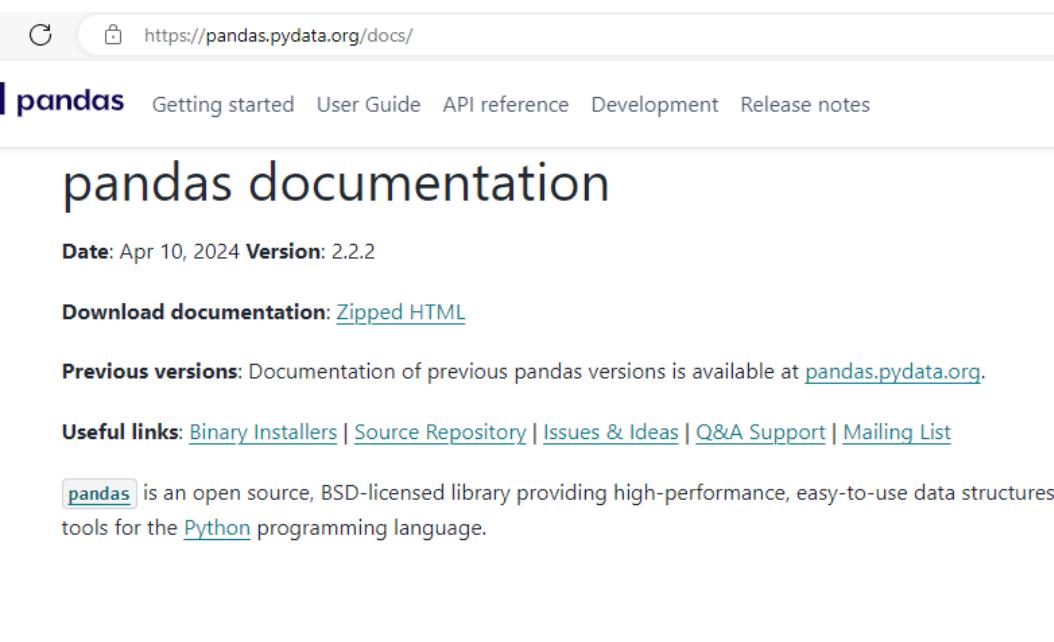
Bias Variance Trade off



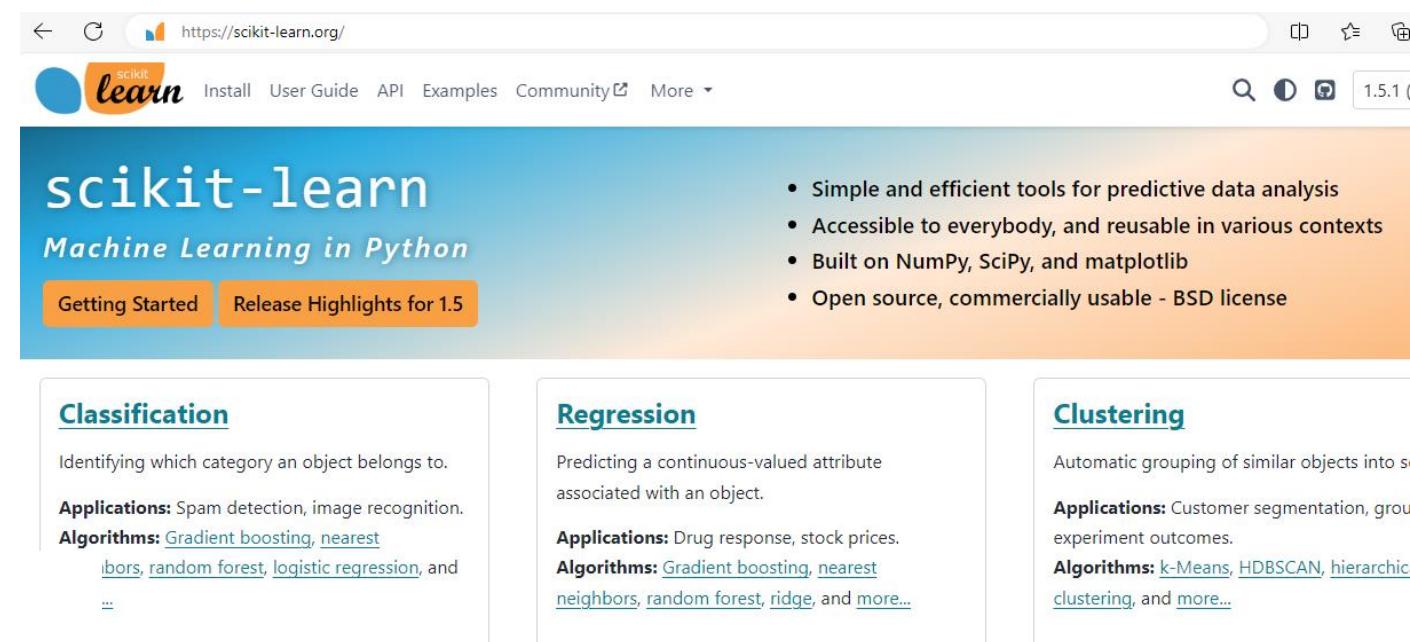
Model Deployment (for developers)

- **Deployment Options:** Choose how to deploy the model (e.g., as a web service, embedded in an application).
 - Flask, Django, Fast-API, StreamLit: e.g. Instagram developed with Django
 - Cloud based hosting: Amazon Web Services, Azure, Google Cloud
- **Integration:** Integrate the model with existing systems and workflows.
 - CI/CD (continuous integration/continuous development)
 - PyTest, CodeCoverage for checking code health everytime you make a change
- **Monitoring:** Set up monitoring to track the model's performance and detect any issues.
 - CRON job: automatically test every minute/hour/day/month

Software Libraries



The screenshot shows the pandas documentation homepage. At the top, there's a navigation bar with links for "Getting started", "User Guide", "API reference", "Development", and "Release notes". Below the navigation, the title "pandas documentation" is displayed. A date stamp "Date: Apr 10, 2024 Version: 2.2.2" is present. There are links to "Download documentation: Zipped HTML" and "Previous versions". A section titled "Useful links" lists "Binary Installers", "Source Repository", "Issues & Ideas", "Q&A Support", and "Mailing List". A paragraph at the bottom explains that pandas is an open source library for Python, providing high-performance data structures and tools.



The screenshot shows the scikit-learn documentation homepage. The header includes a logo, a search bar, and a version indicator "1.5.1". The main title is "scikit-learn Machine Learning in Python". Below the title are two buttons: "Getting Started" and "Release Highlights for 1.5". To the right, a list of bullet points highlights the library's features: "Simple and efficient tools for predictive data analysis", "Accessible to everybody, and reusable in various contexts", "Built on NumPy, SciPy, and matplotlib", and "Open source, commercially usable - BSD license". The page is divided into three main sections: "Classification", "Regression", and "Clustering", each with a brief description and a list of applications and algorithms.



Other Libraries:

- PyTorch, TensorFlow, JAX,
- Scikit-Image, OpenCV, Pillow,
- HuggingFace, JARVIS-Leaderboard etc.

AI/ML Review Articles/Books

JPhys Materials

TOPICAL REVIEW

From DFT to machine learning: recent approaches to materials science—a review

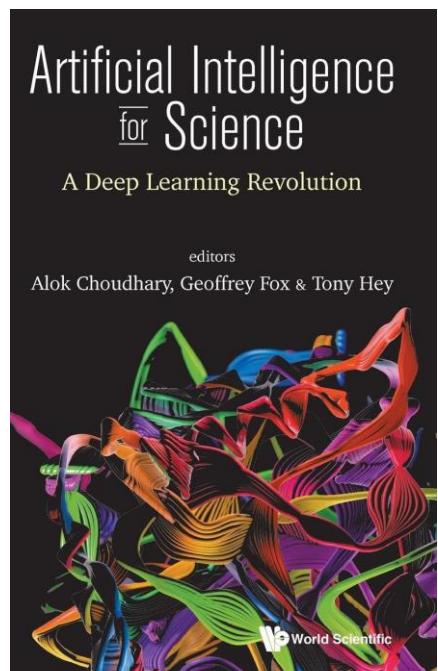
Gabriel R Schleder^{1,2,3} , Antonio C M Padilha² , Carlos Mera Acosta^{1,2} , Marcio Costa²  and Adalberto Fazzio^{1,2,3} 

¹ Center for Natural and Human Sciences, Federal University of ABC, 09210-580, Santo André, São Paulo, Brazil

² Brazilian Nanotechnology National Laboratory/CNPEM, 13083-970, Campinas, São Paulo, Brazil

³ Authors to whom any correspondence should be addressed.

E-mail: gabriel.schleder@ufabc.edu.br and adalberto.fazzio@lnnano.cnpm.br



npj | computational materials

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [npj computational materials](#) > [review articles](#) > [article](#)

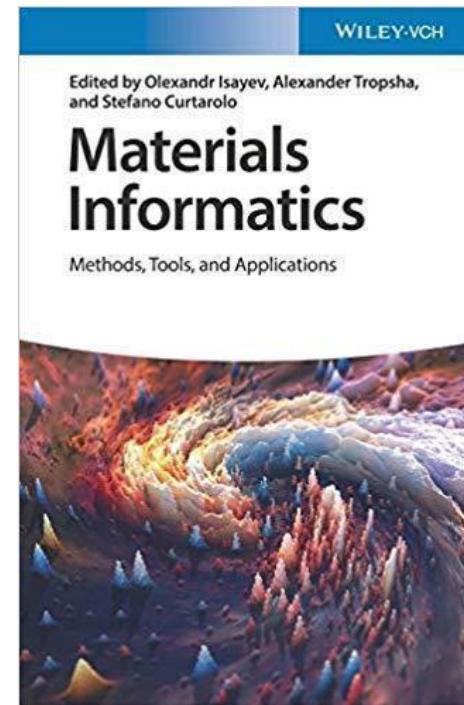
Review Article | [Open access](#) | Published: 05 April 2022

Recent advances and applications of deep learning methods in materials science

[Kamal Choudhary](#) , [Brian DeCost](#), [Chi Chen](#), [Anubhav Jain](#), [Francesca Tavazza](#), [Ryan Cohn](#), [Cheol Woo Park](#), [Alok Choudhary](#), [Ankit Agrawal](#), [Simon J. L. Billinge](#), [Elizabeth Holm](#), [Shyue Ping Ong](#) & [Chris Wolverton](#)

npj Computational Materials **8**, Article number: 59 (2022) | [Cite this article](#)

89k Accesses | 437 Citations | 36 Altmetric | [Metrics](#)

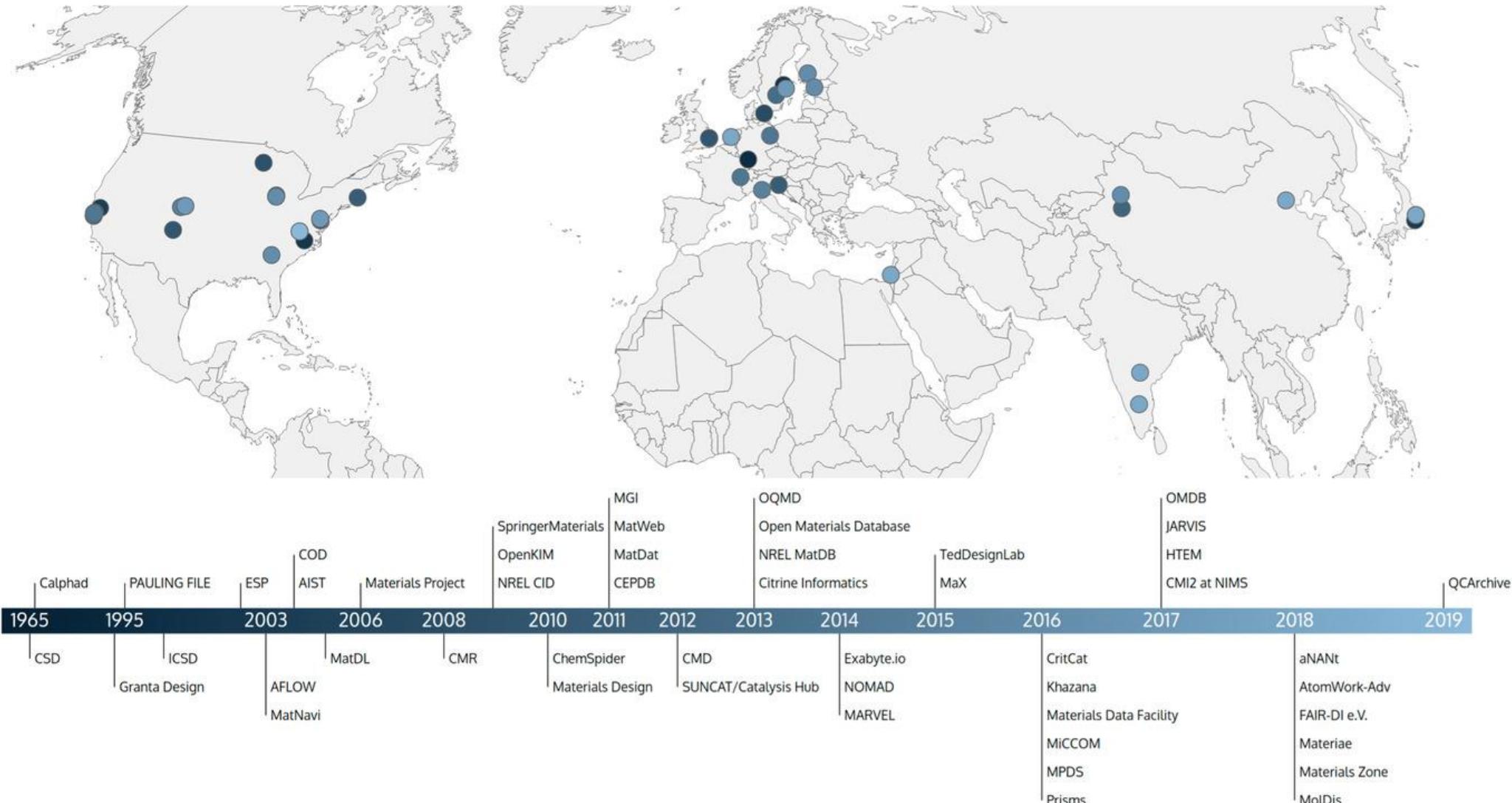


Databases

Table 1. Databases and software for DL atomistic design ('k', 'mil' = thousand, million).

Databases			
DB name	Datasize	Link	Ref.
JARVIS-DFT	56k	https://jarvis.nist.gov/jarvisdft/	3
JARVIS-FF	2.5k	https://jarvis.nist.gov/jarvisff/	3
MP	144k	https://materialsproject.org/	5
OQMD	816k	http://oqmd.org/	4
AFLOW	3.5mil	http://www.aflowlib.org/	6
QM9	134k	http://quantum-machine.org/datasets/	7
ANI	20mil	https://github.com/isayev/ANI1_dataset	96
MD17	1mil	http://quantum-machine.org/datasets	308
Tox21	760k	https://tox21.gov/resources/	309
CCCBDB	2069	https://cccbdb.nist.gov/	310
HOPV15	350	https://doi.org/10.6084/m9.figshare.1610063	311
C2DB	4000	https://cmr.fysik.dtu.dk/c2db/c2db.html	312
FreeSolv	504	https://github.com/MobleyLab/FreeSolv	313
NOMAD	11mil	https://nomad-lab.eu/prod/rae/gui/search	8
OPTIMADE	18mil	http://www.optimade.org/providers-dashboard/	314
<i>Open catalyst</i>			
project	1.2mil	https://opencatalystproject.org	315
MatBench	200k	https://matbench.materialsproject.org/	316
MCloud	22mil	https://www.materialscloud.org/home#statistics	317
CoreMOF	163k	https://mof.tech.northwestern.edu/	318
QMOF	22k	https://github.com/arosen93/QMOF	124
PDB	183k	https://www.rcsb.org/	319
PDBBind	23k	http://www.pdbbind.org.cn/	9
MOAD	39k	http://www.bindingmoad.org/	320

Databases



Software

Software packages			
Model name	Applications	Link	
Ref.			
ALIGNN	Mol, Sol	https://github.com/usnistgov/alignn	65
SchNetPack	Mol, Sol	https://github.com/atomistic-machine-learning	69
CGCNN	Sol	https://github.com/txie-93/cgcnn	67
MEGNet	Mol, Sol	https://github.com/materialsvirtuallab/megnet	33
DimeNet	Mol	https://github.com/klicperajo/dimenet	68
MPNN	Mol	https://github.com/priba/nmp_qc	108
MatDeepLearn	Sol	https://github.com/vxfung/MatDeepLearn	321
GATGCNN	Sol	https://github.com/superlouis/GATGNN	322
ANI	Mol	https://github.com/isayev/ASE_ANI	96
Amp	Sol	https://bitbucket.org/andrewpeterson/amp	323
TensorMol	Mol	https://github.com/jparkhill/TensorMol	324
TorchMD	Mol	https://github.com/torchmd/torchmd	325
PROphet	Sol	https://github.com/biklooost/PROphet	326
DeepMD	Mol	https://github.com/deepmodeling/deepmd-kit	101,327
ænet	Sol	https://github.com/atomisticnet/ænet	328
E3NN	Mol	https://github.com/e3nn/e3nn	329
Neural			
fingerprint	Mol	https://github.com/HIPS/neural-fingerprint	330
DeepChemSt.	Mol	https://github.com/MingCPU/DeepChemStable	331
MoleculeNet	Mol, Sol	https://github.com/deepchem/deepchem	332
dgl-lifesci	Prot	https://github.com/awslabs/dgl-lifesci	66
gnina	Prot	https://github.com/gnina/gnina	110

Table 2. Software to apply DL to chemical formula, SMILES, and fragment representations.

Chemical formula		
Model name	Link	Ref.
MatMiner	https://github.com/hackingmaterials/matminer	151
MagPie	https://bitbucket.org/wolverton/magpie	150
DScribe	https://github.com/SINGROUP/dscribe	158
ElemNet	https://github.com/NU-CUCIS/ElemNet	141
IRNet	https://github.com/NU-CUCIS/IRNet	152,153
Roost	https://github.com/CompRhys/roost	154
CrabNet	https://github.com/anthony-wang/CrabNet	333
CFID-Chem	https://github.com/usnistgov/jarvis	90
Atom2vec	https://github.com/idocx/Atom2Vec	334
CrossPropertyTL	https://github.com/NU-CUCIS/CrossPropertyTL	157
<i>SMILES and fragments</i>		
DeepSMILES	https://github.com/baoilleach/deepsmiles	335
ChemicalVAE	https://github.com/aspru-guzik-group/chemical_vae	336
CVAE	https://github.com/jaechanglim/CVAE	133
DeepChem	https://github.com/deepchem/deepchem	332
DeepFRAG	https://git.durrantlab.pitt.edu/jdurrant/deepfrag/	337
DeepFRAG-k	https://github.com/yaohangli/DeepFragK	338
CheMixNet	https://github.com/NU-CUCIS/CheMixNet	339
SINet	https://github.com/NU-CUCIS/SINet	340

JARVIS-Database Overview

← → C jarvis.nist.gov



JARVIS Databases WebApps Tools Tutorials Outreach

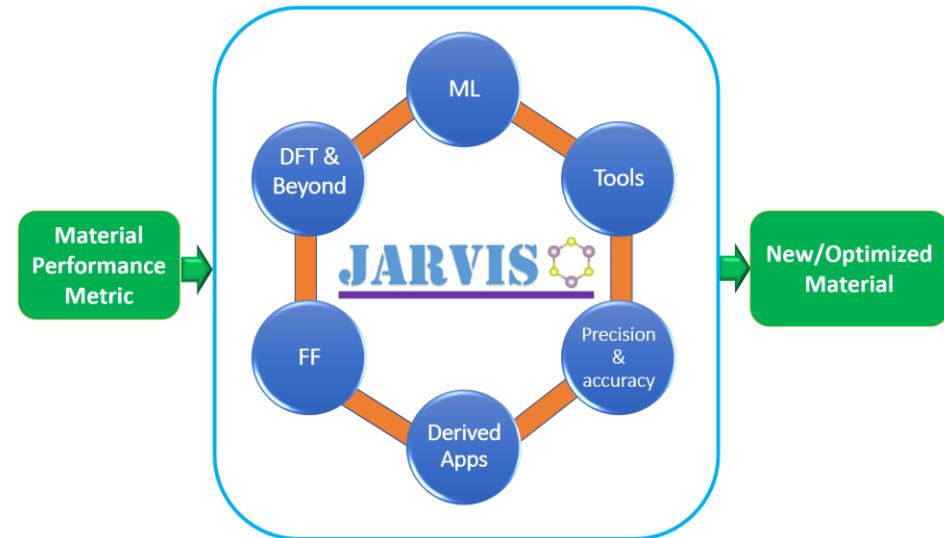
The JARVIS (Joint Automated Repository for Various Integrated Simulator discovery and optimization using classical force-field, density functional th calculations and experiments.

Collection of Databases, WebApps, Tools, Tutorials,...

Materials Genome Initiative 2011, \$400 million



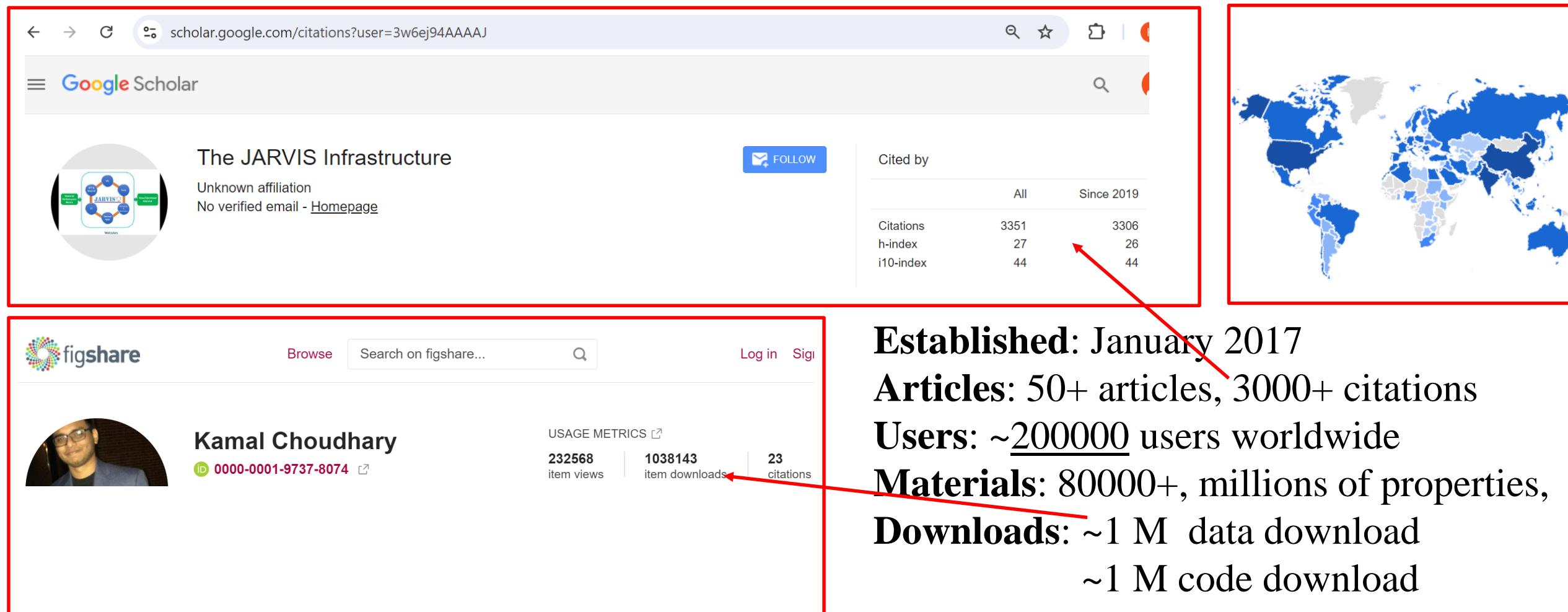
<https://jarvis.nist.gov> (Estd. 2017)



US CHIPS Act 2022, \$52 billion



Impact



Established: January 2017
Articles: 50+ articles, 3000+ citations
Users: ~200000 users worldwide
Materials: 80000+, millions of properties,
Downloads: ~1 M data download
~1 M code download

[Choudhary et al., Nature: npj Computational Materials 6, 173 \(2020\)](#)
[Wines et al., Applied Physics Reviews 10 \(4\) \(2023\)](#)

Databases

Name	Database/WebApp	Description
1. JARVIS-DFT	https://jarvis.nist.gov/jarvisdft	Density functional theory DB for 80000+ materials, >million properties
2. JARVIS-FF	https://jarvis.nist.gov/jarvisdft	Classical DB for ~2000 materials, >100 Classical FFs
3. JARVIS-CFID	https://jarvis.nist.gov/jarvisml	Classical force-field inspired (CFID) descriptors for conventional ML
4. ALIGNN, FF	https://github.com/usnistgov/alignn/	Atomistic Line Graph Neural Network for fast property prediction, unified FF for periodic table, structure optimization
5. AtomGPT	https://github.com/usnistgov/atomgpt	Atomistic Generative Pretrained transformer for forward and inverse materials design
6. InterMat & DefectMat	https://github.com/usnistgov/intermat & https://github.com/usnistgov/defectmat	Interface & Defect Materials Design Toolkit with DFT and AI methods
7. ChemNLP	https://github.com/usnistgov/chemnlp	Natural language processing for materials chemistry for arXiv/pubchem data
8. DAC	https://jarvis.nist.gov/jdac	Direct air capture/CO ₂ capture with AI trained on GCMC data
9. Leaderboard	pages.nist.gov/jarvis_leaderboard	Large scale benchmark platform with >300 benchmarks, 9 million data points
10. AtomVision	https://github.com/usnistgov/atomvision	Computer vision models and dataset for materials science, STEM/STM
11. AtomQC	https://github.com/usnistgov/atomqc	Quantum computation library for molecules and solids
12. Tb3Py	https://github.com/usnistgov/tb3py	Three-body tight-binding model for the periodic table
13. Solar	https://jarvis.nist.gov/jsolar	Solar cell materials design with DFT, AI

How to Get Started

github.com/JARVIS-Materials-Design/jarvis-tools-notebooks

JARVIS-Materials-Design / jarvis-tools-notebooks

Table of Contents

- [Introduction](#)
- [Basics](#)
- [ES tutorial](#)
- [FF/MC related tutorial](#)
- [JARVIS-School](#)
- [AI tutorial](#)
- [QC tutorial](#)
- [References](#)
- [How to contribute](#)
- [Correspondence](#)
- [Funding support](#)
- [License](#)

Collection of Jupyter Notebooks & Recorded Videos

JARVIS-Tools Notebooks (Introduction)

The JARVIS-Tools Notebooks is a collection of Jupyter/ Google-Colab notebooks to provide on various methods for materials design. It consists of several types of applications such as

jarvis.nist.gov/events/

JARVIS 

JARVIS-NIST Events

- AIMS
- QMMS
- JARVIS-School

JARVIS-School is accompanied by We plan to first g topics such as:

nist.gov/news-events/events/2023/07/artificial-intelligence-materials-science-aims-workshop

2023 Artificial Intelligence for Materials Science (AIMS) Workshop
3 videos

- 20230725-27_AIMS_Day1 5:49:39
- 20230725-27_AIMS_Day1 5:49:39
- 20230725-27_AIMS_Day2 4:44:20
- 20230725-27_AIMS_Day2 4:44:20
- 20230725-27_AIMS_Day3 4:37:50
- 20230725-27_AIMS_Day3 4:37:50

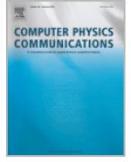
Feature engineering/descriptors

1. Element descriptors
2. MagPie
3. Coulomb matrix,
4. Ewald sum matrix,
5. sine matrix,
6. Many-body Tensor Representation (MBTR),
7. Atom-centered Symmetry Function (ACSF),
8. Smooth Overlap of Atomic Positions (SOAP),
9. Classical force-field inspired descriptors (CFID),
10. Sure independence screening
and sparsifying operator (SISSO),...



Computer Physics Communications

Volume 247, February 2020, 106949



Dscribe: Library of descriptors for
machine learning in materials
science ☆

Lauri Himanen ^a  , Marc O.J. Jäger ^a, Eiaki V. Morooka ^a,
Filippo Federici Canova ^{a b}, Yashasvi S. Ranawat ^a, David Z. Gao ^{b c}, Patrick Rinke ^{a f},
Adam S. Foster ^{a d e}



Computational Materials Science

Volume 152, September 2018, Pages 60-69



Matminer: An open source toolkit for
materials data mining

Logan Ward ^{a b}  , Alexander Dunn ^{c d}, Alireza Faghaninia ^c, Nils E.R. Zimmermann ^c,
Saurabh Bajaj ^{c e}, Qi Wang ^c, Joseph Montoya ^c, Jiming Chen ^f, Kyle Bystrom ^d,
Maxwell Dylla ^g, Kyle Chard ^{a b}, Mark Asta ^d, Kristin A. Persson ^c, G. Jeffrey Snyder ^g,
Ian Foster ^{a b}, Anubhav Jain ^c  

Chemical fraction descriptors

	H	Li	Be	B	C	N	O	F	...
LiH	1	1	0	0	0	0	0	0	...
LiF	0	1	0	0	0	0	0	1	...
BeO	0	0	1	0	0	0	1	0	...
BN	0	0	0	1	0	1	0	0	...
:	:	:	:	:	:	:	:	:	...

Fig. 1.1 Binary elemental descriptors representing the presence of chemical elements. The number of binary elemental descriptors corresponds to the number of element types included in the training data

Isao Tanaka *Editor*

Nanoinformatics

OPEN

 Springer

MagPie descriptors

Element Property statistics of Magpie	Additional Predictors of Magpie	Added Predictors in this work
Atomic Number	Stoichiometry p-norm ($p = 0, 2, 3, 5, 7$)	Total Atom Number
Mendeleev Number	Elemental Fraction	Maximum Atom Number
Atomic Weight	Fraction of Electrons in each Orbital	Minimum Atom Number
Melting Temperature	Band Center	Average Atom Number
Periodic Table Row and Column	Ion Property (possible to form ionic compound, ionic charge)	Specific Value
Covalent Radius		Atom Number Variance
Electronegativity		
The number of Valence e in each orbital(s, p, d, f, total)		
The number of unfilled e in each orbital (s, p, d, f, total)		
Ground State Volume		
Ground State Band Gap Energy		
Ground State Magnetic Moment		
Space Group Number of elements		

npj | computational materials

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [npj computational materials](#) > [articles](#) > article

Article | [Open access](#) | Published: 26 August 2016

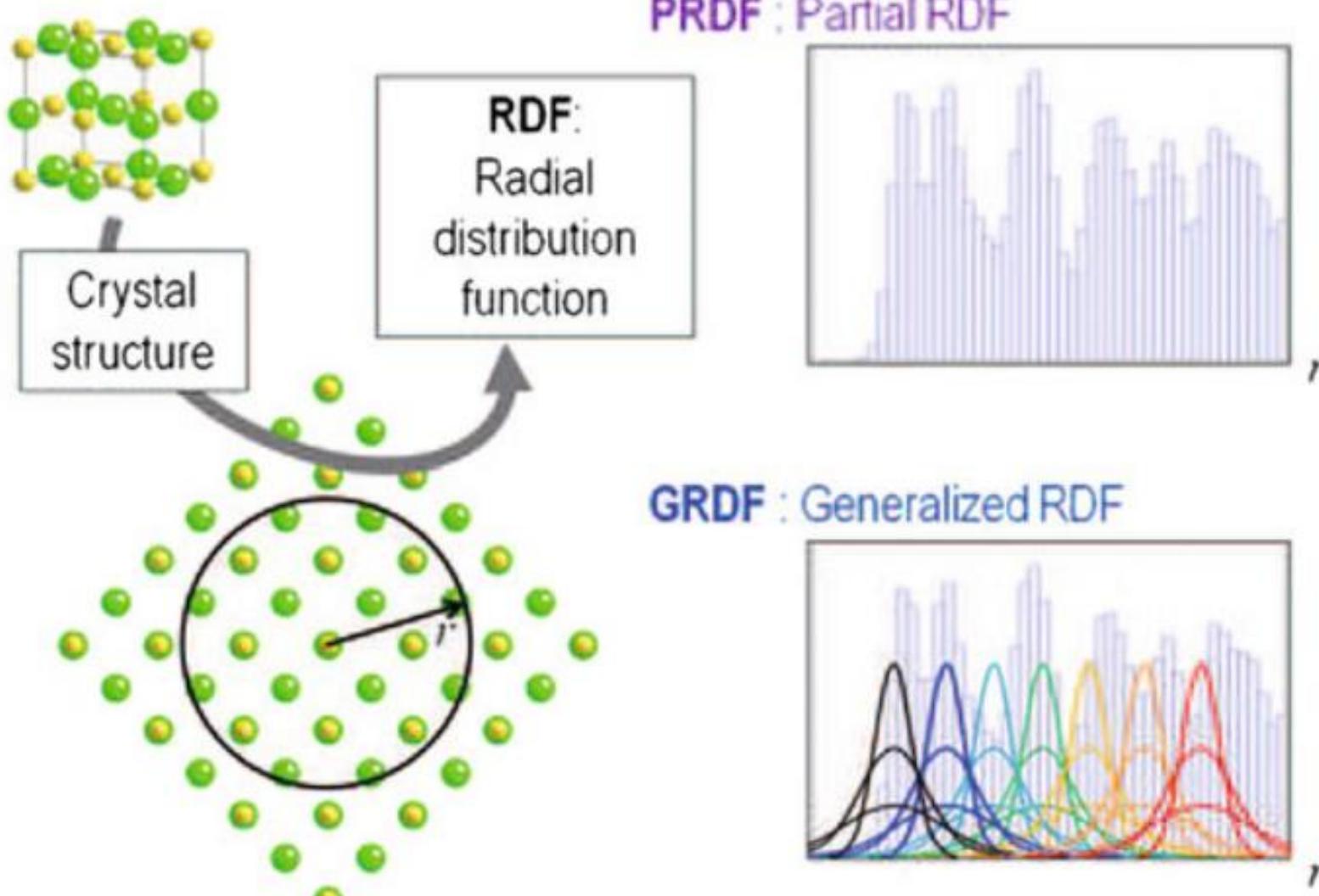
A general-purpose machine learning framework for predicting properties of inorganic materials

[Logan Ward](#), [Ankit Agrawal](#), [Alok Choudhary](#) & [Christopher Wolverton](#) 

[npj Computational Materials](#) 2, Article number: 16028 (2016) | [Cite this article](#)

60k Accesses | 1067 Citations | 18 Altmetric | [Metrics](#)

Structural descriptors (RDF/PRDF)



Coulomb matrix descriptors

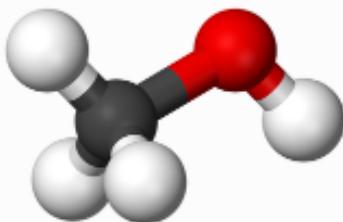
Coulomb matrix is calculated with the equation below.

$$M_{ij}^{\text{Coulomb}} = \begin{cases} 0.5Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{for } i \neq j \end{cases}$$

The diagonal elements can be seen as the interaction of an atom with itself and are essentially a polynomial fit of the atomic energies to the nuclear charge Z_i . The off-diagonal elements represent the Coulomb repulsion between nuclei i and j .

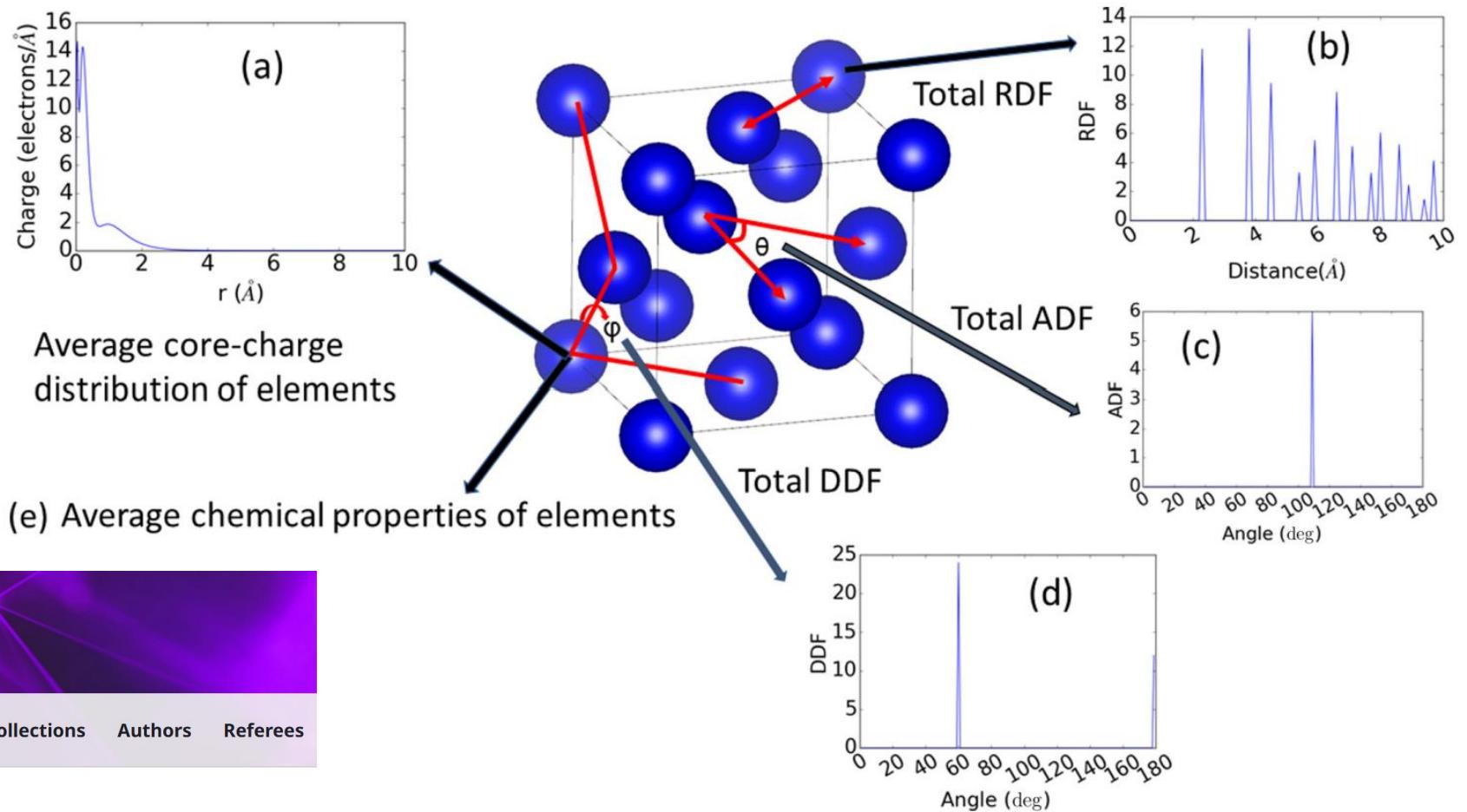
Let's have a look at the CM for methanol:

https://singroup.github.io/dscribe/0.3.x/tutorials/coulomb_matrix.html



36.9	33.7	5.5	3.1	5.5	5.5
33.7	73.5	4.0	8.2	3.8	3.8
5.5	4.0	0.5	0.35	0.56	0.56
3.1	8.2	0.35	0.5	0.43	0.43
5.5	3.8	0.56	0.43	0.5	0.56
5.5	3.8	0.56	0.43	0.56	0.5

CFID Descriptors



GO MOBILE » | ACCESS BY NIST

Machine learning with force-field-inspired descriptors for materials: Fast screening and mapping energy landscape

Kamal Choudhary, Brian DeCost, and Francesca Tavazza

Other Descriptors

Behler-Parinello

Behler-Parinello (BP) also called "Gaussian" fingerprints are local fingerprints based on symmetry functions. Two of such symmetry functions are given by the radial and angular components G^{rad} and G^{ang} below where summations run over all neighbors j and k separated by distances R_{ij} and R_{ik} with respect to atom i within a cutoff distance R_c around i . θ_{ijk} is the angle between atoms i, j and k . η, R_s, λ and ζ are parameters whose values are chosen by the user. f_c is a cutoff function used to ensure a smooth transition to zero at the R_c . For more information, see: [\[BP\]](#)

$$G_i^{rad} = \sum_j e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij})$$

$$G_i^{ang} = 2^{1-\zeta} \sum_{j,k \neq i} (1 + \lambda \cos \theta_{ijk})^\zeta e^{-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)^2} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk})$$

$$f_c = \begin{cases} 0.5[\cos(\frac{\pi R_{ij}}{R_c}) + 1] & \text{for } R_{ij} \leq R_c \\ 0 & \text{for } R_{ij} > R_c \end{cases}$$

Zernike

Zernike fingerprints are similar to Bispectrum fingerprints in the sense that they are based on decomposition of a local atomic density wrt basis sets. However, Zernike fingerprints are based on decomposition wrt zernike polynomials and 3D spherical harmonics. The general formula is given below. Please consult this paper which describes the [\[Zernike\]](#) method.

$$\rho(\tilde{r}, \theta, \phi) = \sum_{n=0}^{\inf} \sum_l \sum_{m=-l}^l c_{nl}^m Z_{nl}^m(\tilde{r}, \theta, \phi) \quad \text{for } n - l \geq 0$$

AGNI

The [\[AGNI\]](#) method was developed as a framework for machine learning force field development in which the forces are calculated directly without going through energy predictions. The associated fingerprint is given by $V_{i,\alpha,k}$ below where α denotes the direction (x,y or z) of the force between atoms i and j separated by distance r_{ij} . The parameter w corresponds to the width of Gaussians placed at positions a_k within a cutoff distance R_c . Similarly to the BP fingerprint, f_c is the cutoff function ensuring a smooth transition to zero at R_c .

$$V_{i,\alpha,k} = \sum_{j \neq i} \frac{r_{ij}^\alpha}{r_{ij}} \frac{1}{\sqrt{2\pi w}} e^{-0.5(\frac{r_{ij}-a_k}{w})^2} f_c(r_{ij})$$

Bispectrum

Bispectrum fingerprints for representation of chemical environments were proposed by Bartok et al. and are based on the decomposition of a local atomic density function with respect to 4D spherical harmonics. The bispectrum representation is then built based on the coefficients $c_{m'm}^j$ of the decomposition given below. For more information, please consult the original paper on the development of [\[Bispectrum\]](#) fingerprints.

$$B_{j_1,j_2,j} = \sum_{m'_1,m_1=-j_1}^{j_1} \sum_{m'_2,m_2=-j_2}^{j_1} \sum_{m',m=-j}^j c_{m'm}^j C_{j_1 m_1 j_2 m_2}^{jm} C_{j_1 m'_1 j_2 m'_2}^{jm'} c_{m'_1 m_1}^{j_1} c_{m'_2 m_2}^{j_2}$$

Comparison of Descriptors

```
# Convert my_data to numpy array using CFID-Chem, Magpie and elemental fraction descriptors

X_cfid = []
X_Mag = []
X_elff = []

Y = []
IDs = []

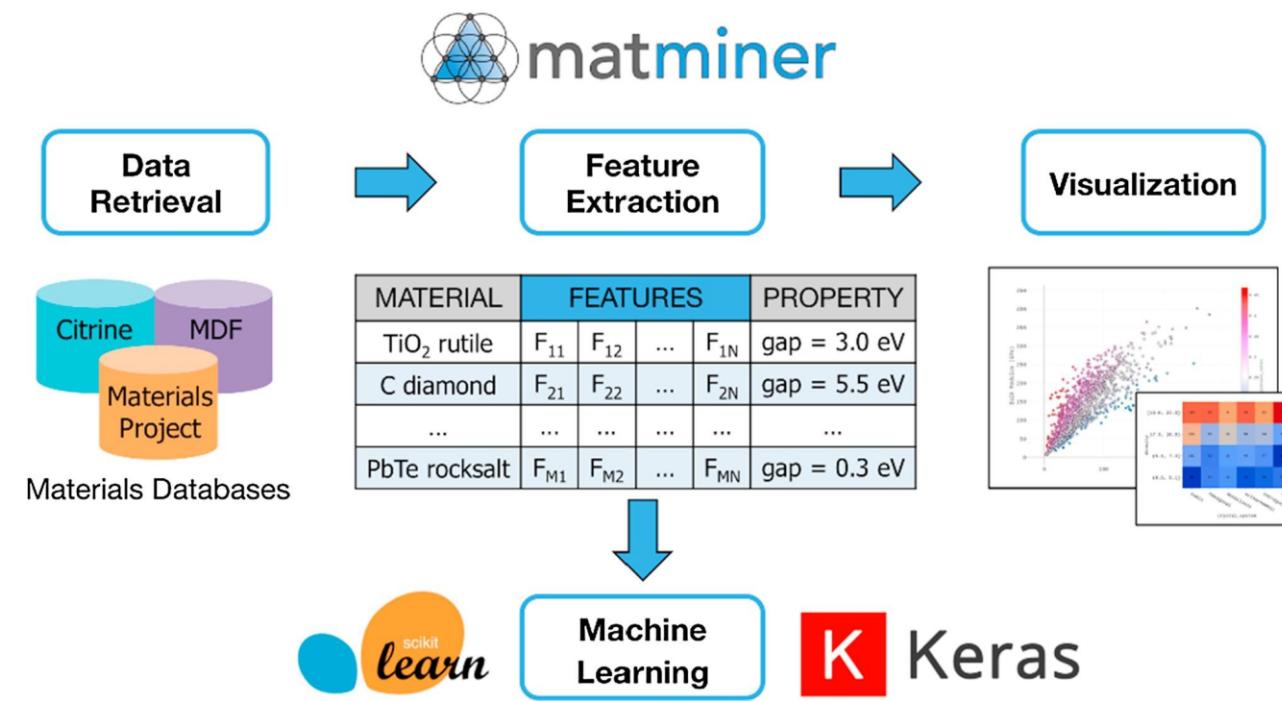
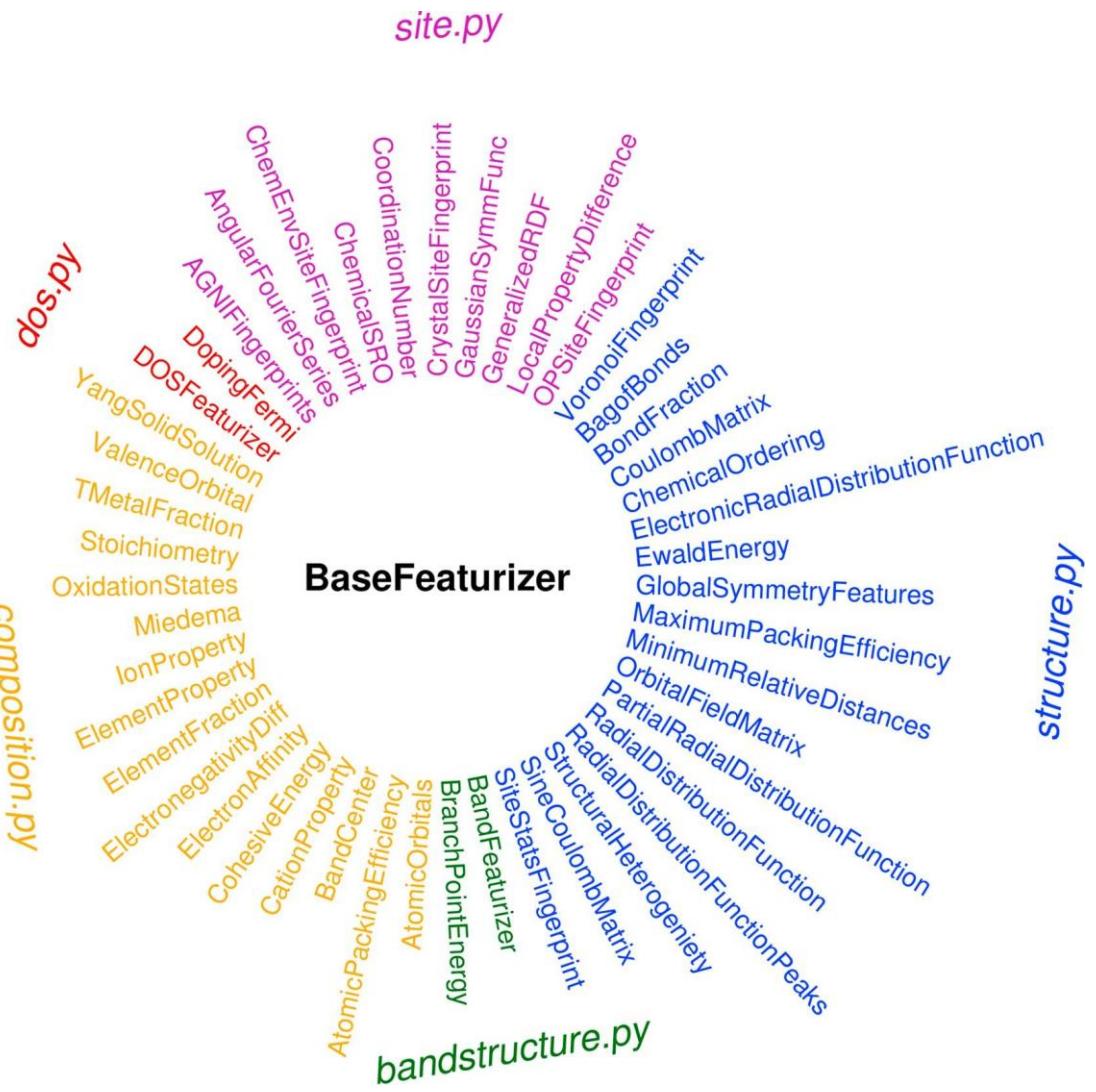
for ii, i in enumerate(my_data):
    comp = i[0]
    desc_magpie,magpie_names = get_chem_only_descriptors(formula=comp, source="magpie")
    val = i[1]
    X_Mag.append(desc_magpie)
    Y.append(val)
    IDs.append(ii)

X_Mag = np.array(X_Mag)

Y = np.array(Y).reshape(-1, 1)

...
Print out the mean absolute deviation, which captures the spread of values in
the target property distribution
...
IDs = np.array(IDs)
mad = mean_absolute_deviation(Y)
print("MAD:", mad)
```

MatMiner



Day 3 Hands on

AI 1 ML Sklearn Steel Fatigue

AI 2 ML_Chem_Formula_Descriptors

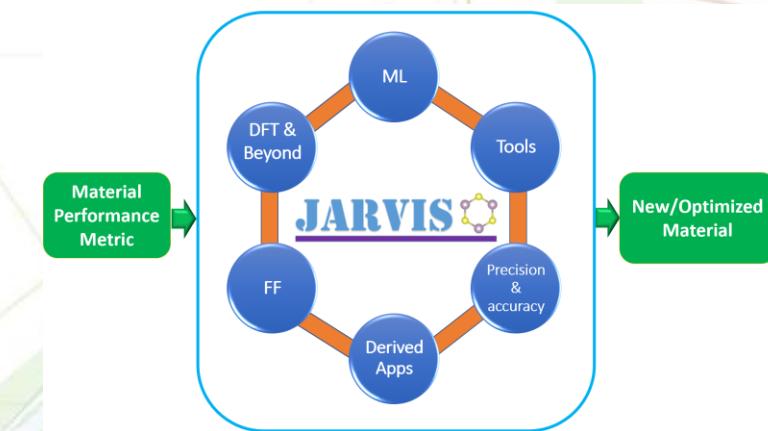
AI 3 Simple_Machine_learning_training_example_with_CFID_descriptors

AI 5 JARVIS_CFID_LightGBM_GPUvsCPU

Materials Informatics: Fingerprints, Graph Neural Networks, and Transformers Based Models for Materials Design



Kamal Choudhary
Staff Scientist
NIST, Gaithersburg, MD, USA
Jan 20-25, 2025



<https://jarvis.nist.gov>

Day 4

- Deep learning models (CNN, GNN, GPT)
- AtomVision, ALIGNN, AtomGPT, ChemNLP
- JARVIS-Leaderboard
- Benchmarking and reproducibility

AtomVision

A deep learning framework for atomistic image data

usnistgov / atomvision Public

generated from usnistgov/opensource-repo

Code Issues Pull requests Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

Author	Commit Message	Date	Commits
knc6	Add github action.	ec56094 2 hours ago	14 commits
	.github	Add github action.	2 hours ago
	atomvision	Remove unnecessary files.	2 hours ago
	CODEMETA.yaml	Initial commit	26 days ago
	LICENSE.md	Initial commit	26 days ago
	README.md	Update README.md	2 hours ago
	setup.py	Fix requirements.	2 hours ago

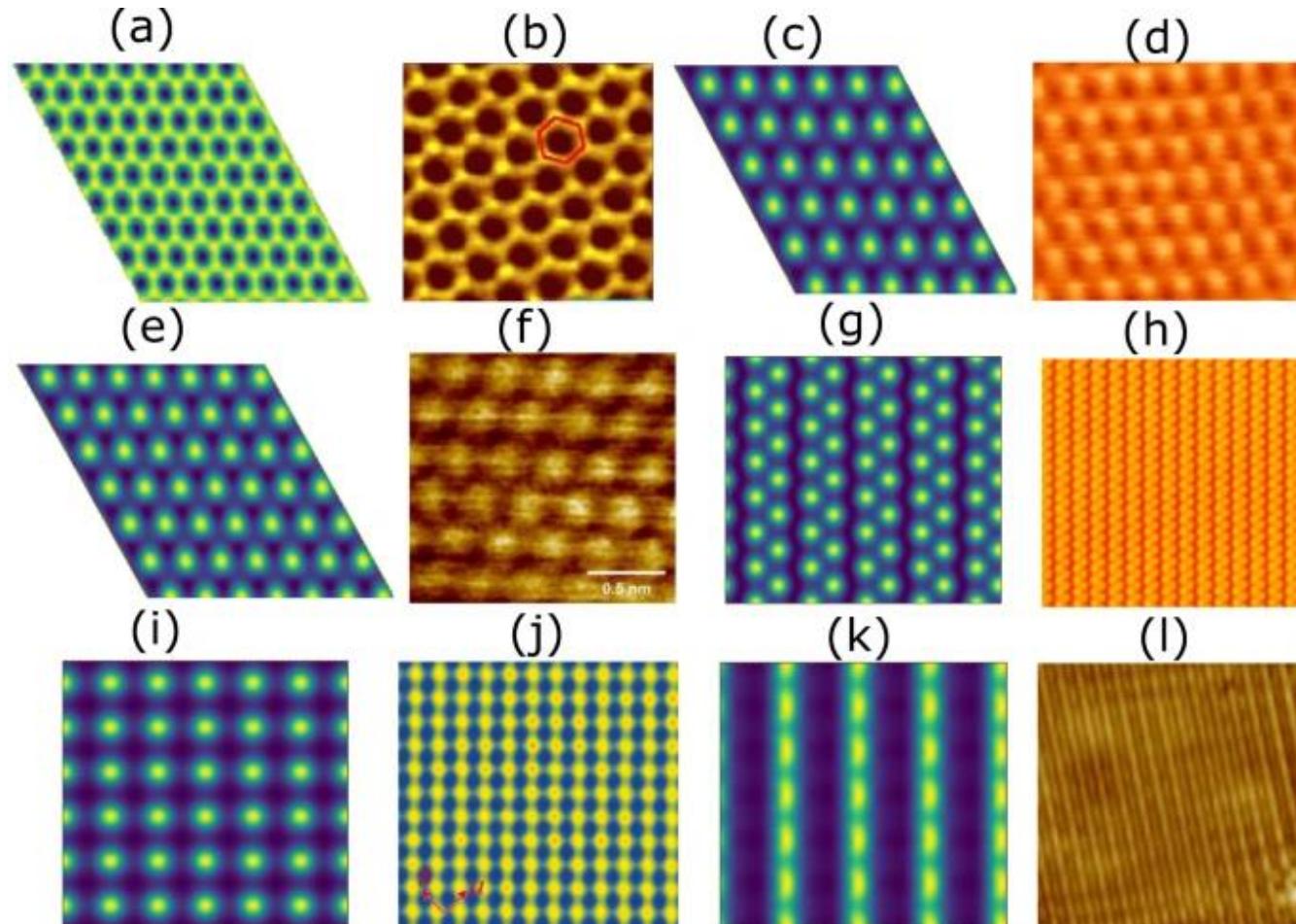
README.md

Open in Colab

Scanning Tunneling Microscope Image

Tersoff-Hamman approach, constant height/current images

Tunneling current is proportional to the integrated local density of states (ILDOS)



$$n(r, E) = \sum_{\mu} |\psi_{\mu}(r)|^2 \delta(\varepsilon_{\mu} - E)$$

$$I(r, V) \propto \int_{E_F}^{E_F + eV} dE n(r, E)$$

[nature](#) > [scientific data](#) > [data descriptors](#) > [article](#)

Data Descriptor | [Open Access](#) | Published: 11 February 2021

Computational scanning tunneling microscope image database

Kamal Choudhary✉, Kevin F. Garrity, Charles Camp, Sergei V. Kalinin, Rama Vasudevan, Maxim Ziatdinov & Francesca Tavazza

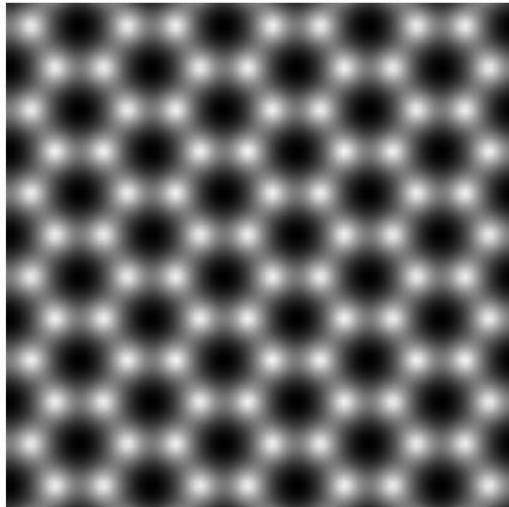
[Scientific Data](#) 8, Article number: 57 (2021) | [Cite this article](#)

1594 Accesses | 1 Citations | 4 Altmetric | [Metrics](#)

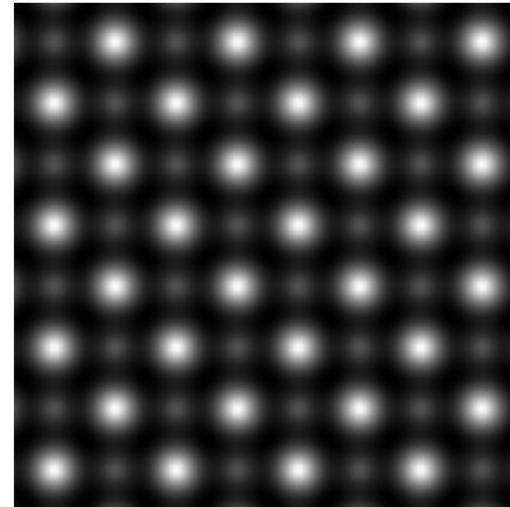
Scanning Transmission Electron Microscope Image

Convolution approximation: accurate for **thin films** mainly
Based on Rutherford scattering model

$$I(\mathbf{r}) = R(\mathbf{r}, Z) \otimes \text{PSF}(\mathbf{r}), \quad R(\mathbf{r}, Z) = \sum_{i=1}^N Z_i^{1.7} \delta(\mathbf{r} - \mathbf{r}_i)$$



C: JVASP-667

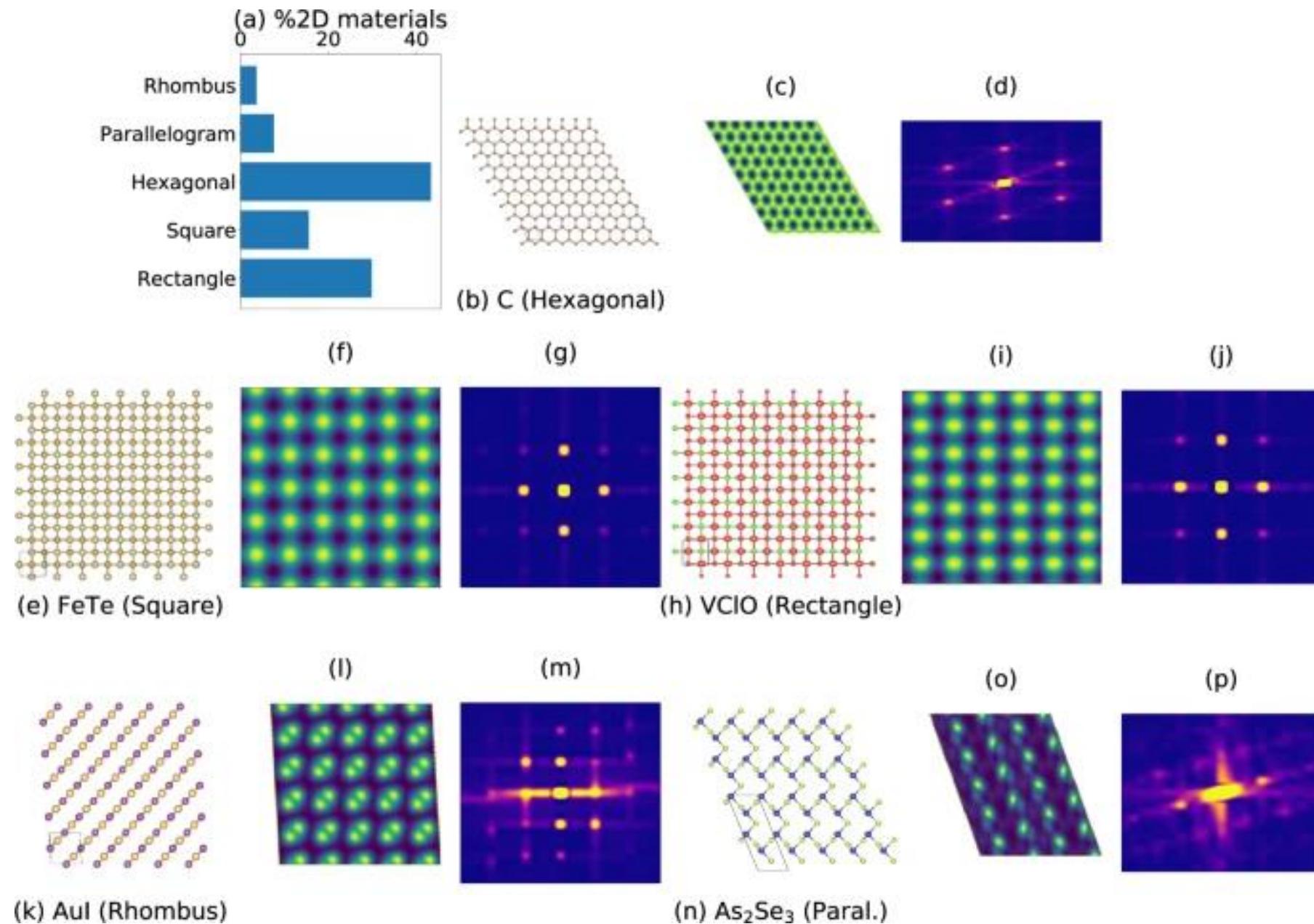


FeTe: JVASP-6667



PPdSe: JVASP-6316

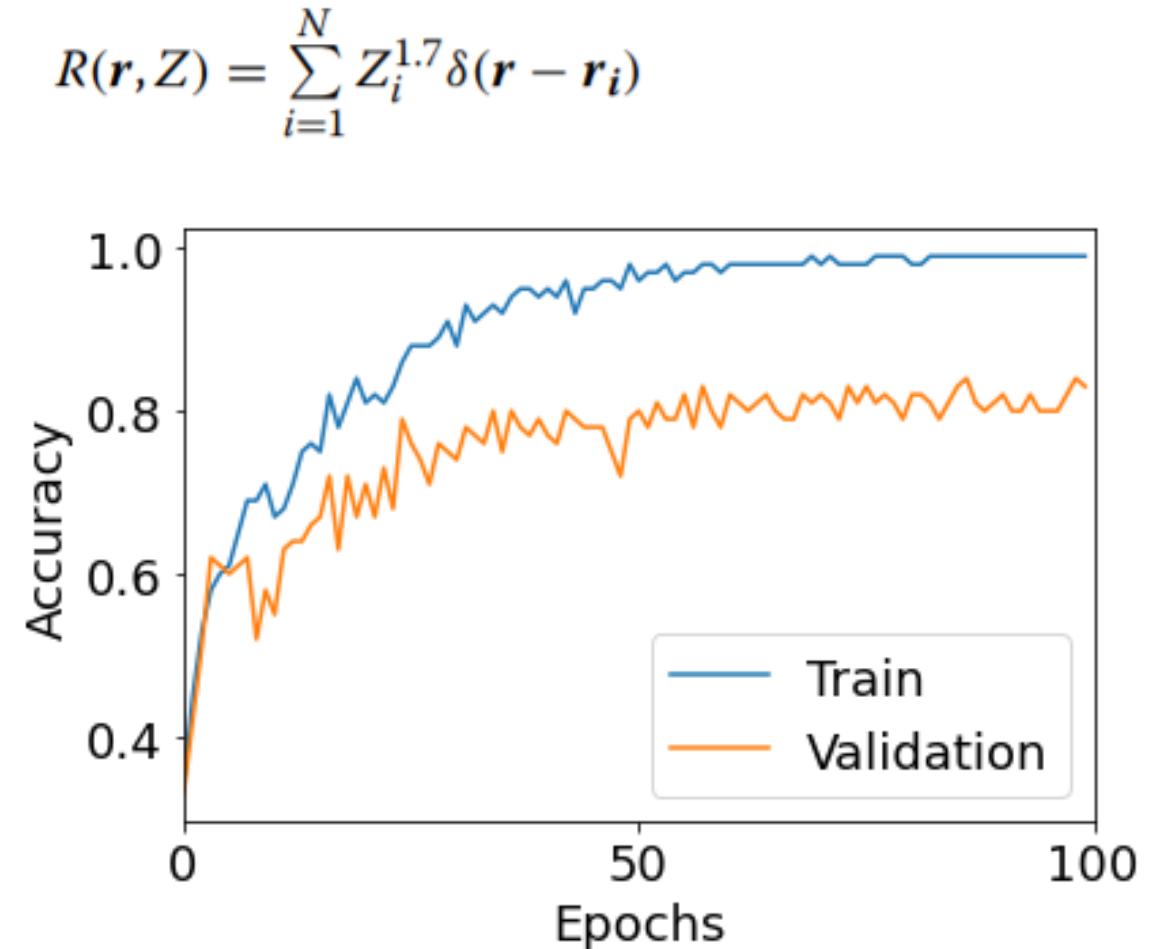
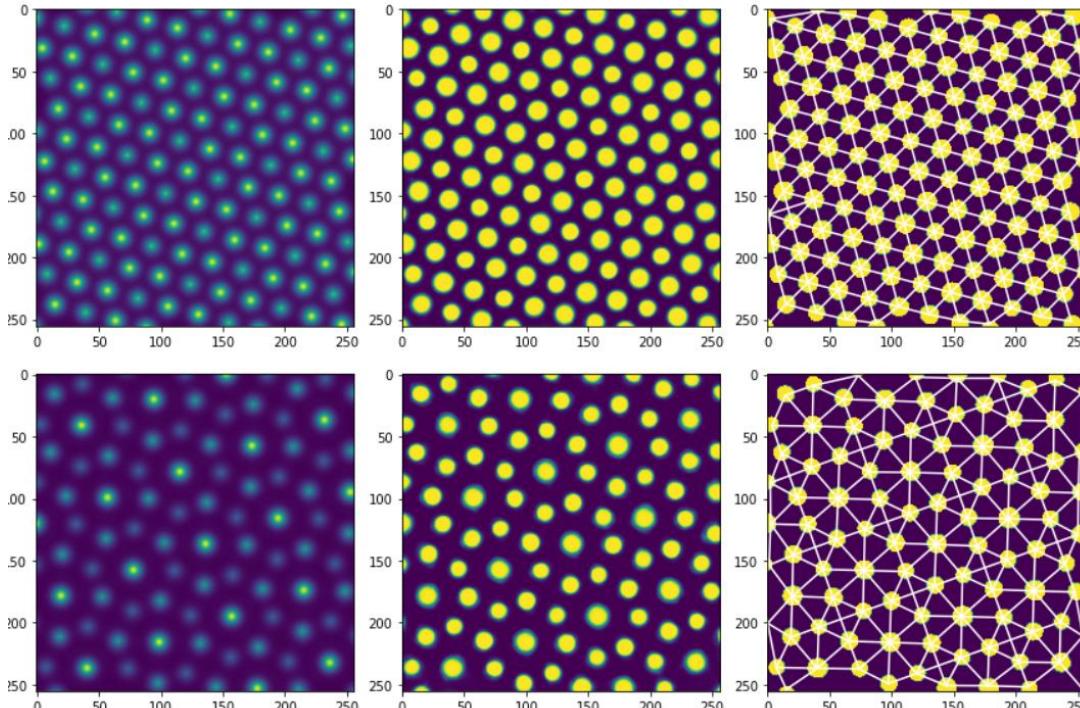
2D Lattice Classification



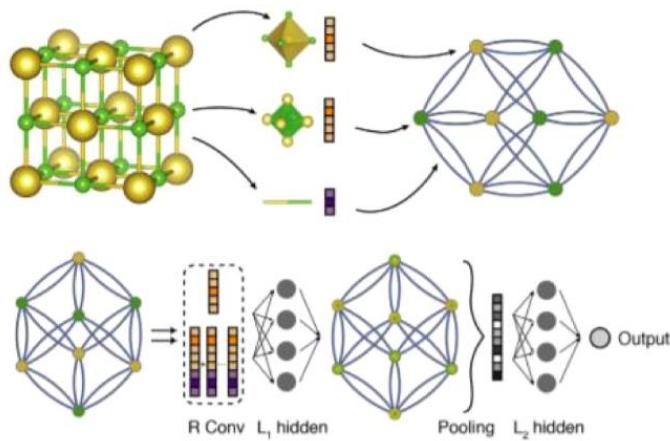
Applications: Integrating with STEM imaging

Convolution approximation: accurate for **thin films** mainly
Based on Rutherford scattering model

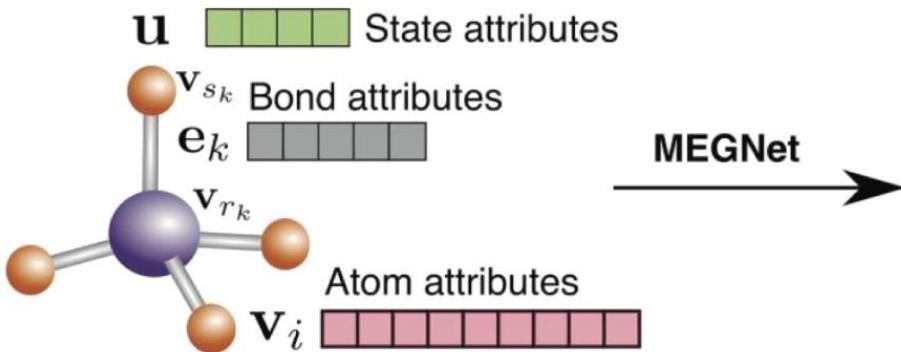
$$I(\mathbf{r}) = R(\mathbf{r}, Z) \otimes \text{PSF}(\mathbf{r}), \quad R(\mathbf{r}, Z) = \sum_{i=1}^N Z_i^{1.7} \delta(\mathbf{r} - \mathbf{r}_i)$$



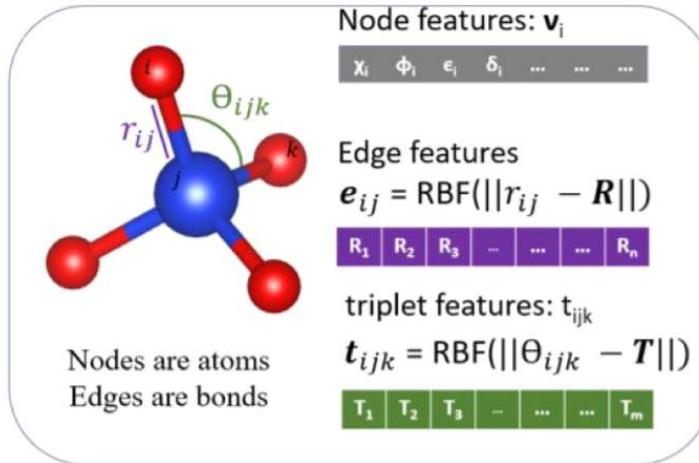
Graph Neural Networks



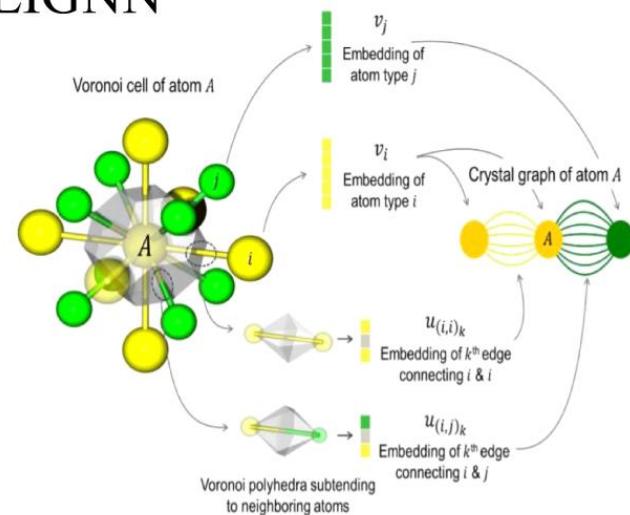
(a) CGCNN



(c) MEGNet



(b) ALIGNN



(d) iCGCNN

ALIGNN

Atomistic Line Graph Neural Network

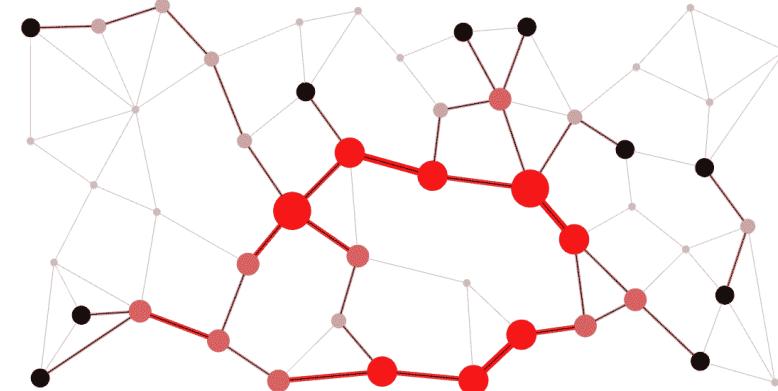
github.com/usnistgov/alignn

usnistgov / alignn

Type / to

Code Issues 23 Pull requests 5 Discussions Actions Projects

NIST alignn Public Edit Pins ▾



npj | computational materials

Explore content ▾ About the journal ▾ Publish with us ▾

nature > npj computational materials > articles > article

Article | Open Access | Published: 15 November 2021

Atomistic Line Graph Neural Network for improved materials property predictions

Kamal Choudhary & Brian DeCost

npj Computational Materials 7, Article number: 185 (2021) | Cite this article

Digital Discovery

PAPER

Check for updates

Cite this: DOI: 10.1039/d2dd00096b



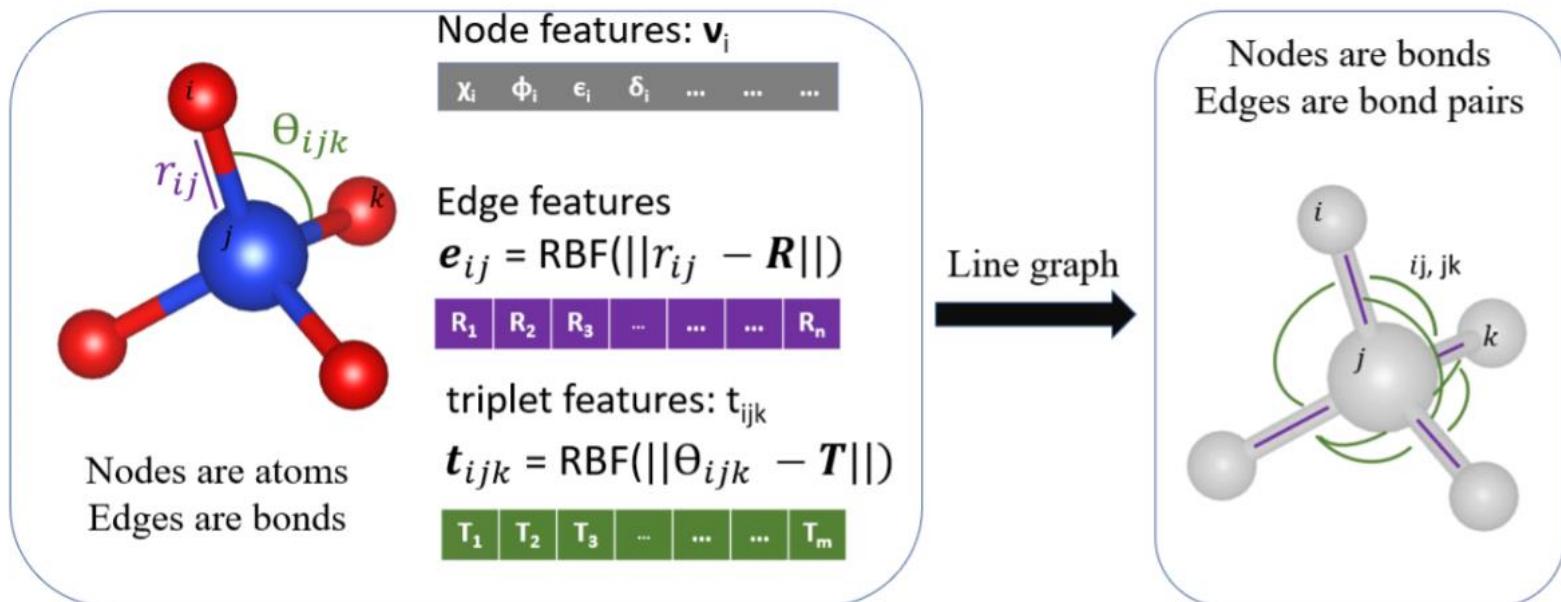
View Article Online
View Journal

Unified graph neural network force-field for the periodic table: solid state applications

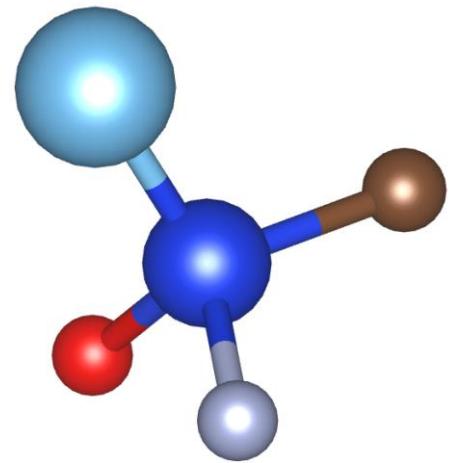
Kamal Choudhary, *ab Brian DeCost, c Lily Major, de Keith Butler, e Jeyan Thiyyagalingam e and Francesca Tavazza c

Atomistic Graph & Line Graph

Explicitly represent pairwise and triplet (bond angle) interactions using line graph
Possible to extend for n-body, e.g. line graph of line graph



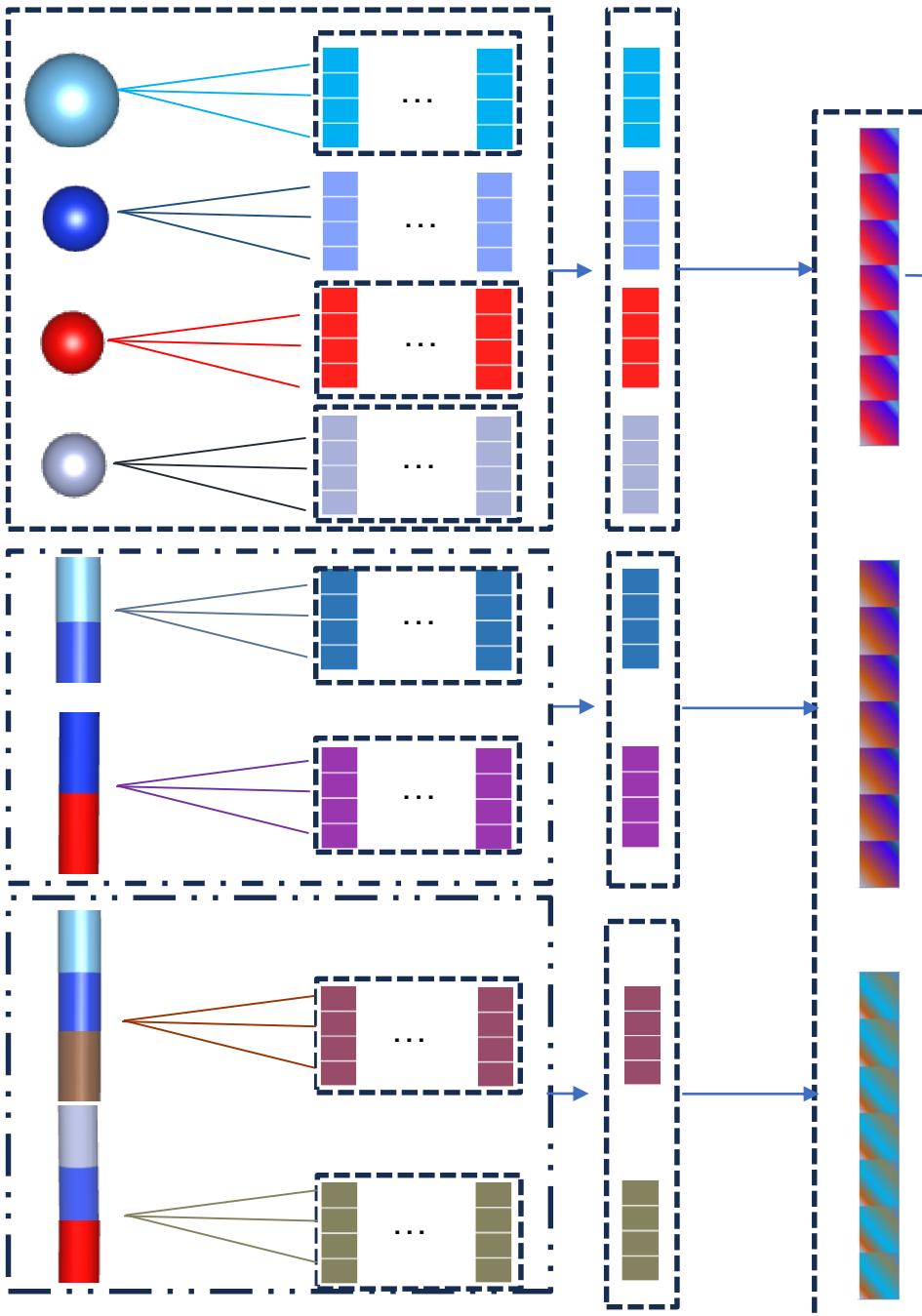
- Graph level prediction, e.g. energy
- Node level predictions, e.g. charges
- Node level derivatives, e.g. forces
- Edge level predictions, e.g. LJ params



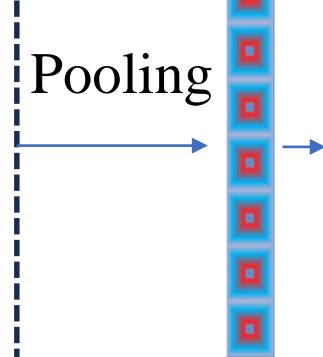
Bond features
(y)

Angle features
(z)

Atom features (x) Embedding ALIGNN $f(x,y,z)$
& GCN $f(x,y)$



Node wise outputs $f(x)$
(Charge, Magnetic moment,...)



Graph wise outputs
(Energy, Bandgap,
Magnetic moment,
& 20+ properties)

Node wise derivatives
of energy (forces)

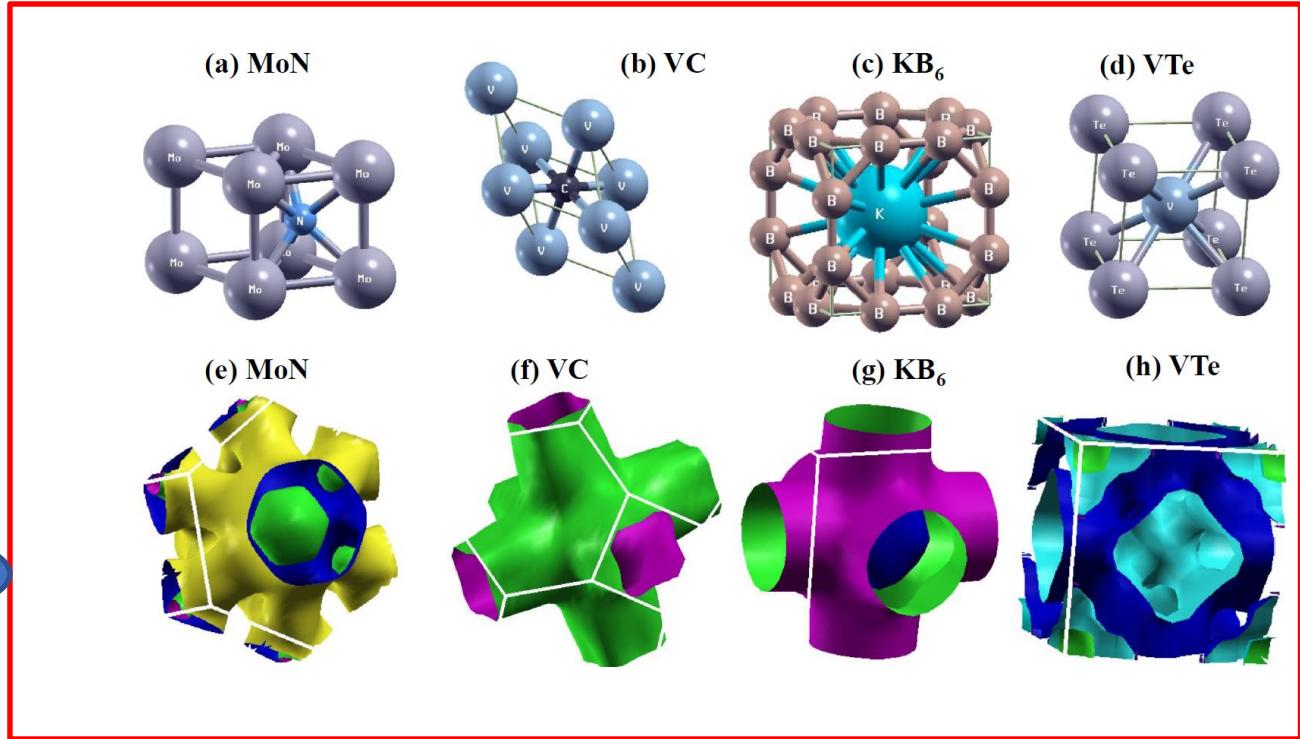
Performance on the JARVIS-DFT Dataset

Property	Units	MAD	CFID	CGCNN	ALIGNN	MAD: MAE
Formation energy	eV(atom) ⁻¹	0.86	0.14	0.063	0.033	26.06
Bandgap (OPT)	eV	0.99	0.30	0.20	0.14	7.07
Total energy	eV(atom) ⁻¹	1.78	0.24	0.078	0.037	48.11
Ehull	eV	1.14	0.22	0.17	0.076	15.00
Bandgap (MBJ)	eV	1.79	0.53	0.41	0.31	5.77
Kv	GPa	52.80	14.12	14.47	10.40	5.08
Gv	GPa	27.16	11.98	11.75	9.48	2.86
Mag. mom	μB	1.27	0.45	0.37	0.26	4.88
SLME (%)	No unit	10.93	6.22	5.66	4.52	2.42
Spillage	No unit	0.52	0.39	0.40	0.35	1.49
Kpoint-length	\AA	17.88	9.68	10.60	9.51	1.88
Plane-wave cutoff	eV	260.4	139.4	151.0	133.8	1.95
ϵ_x (OPT)	No unit	57.40	24.83	27.17	20.40	2.81
ϵ_y (OPT)	No unit	57.54	25.03	26.62	19.99	2.88
ϵ_z (OPT)	No unit	56.03	24.77	25.69	19.57	2.86

Trained on ~55k materials

- Total energy, Formation energy , Ehull
- Bandgap (OPT), Bandgap (MBJ)
- Kv, Gv
- Mag. mom
- ϵ_x (OPT/MBJ), ϵ_y (OPT), ϵ_z (OPT), ϵ (DFPT:elec+ionic)
- Max. piezo. stress coeff (e_{ij})
- Solar-SLME (%)
- Topological-Spillage
- 2D-Exfo. energy
- Kpoint-length
- Plane-wave cutoff
- Max. Electric field gradient
- avg. m_e , avg. m_h
- n-Seebeck, n-PF, p-Seebeck, p-PF

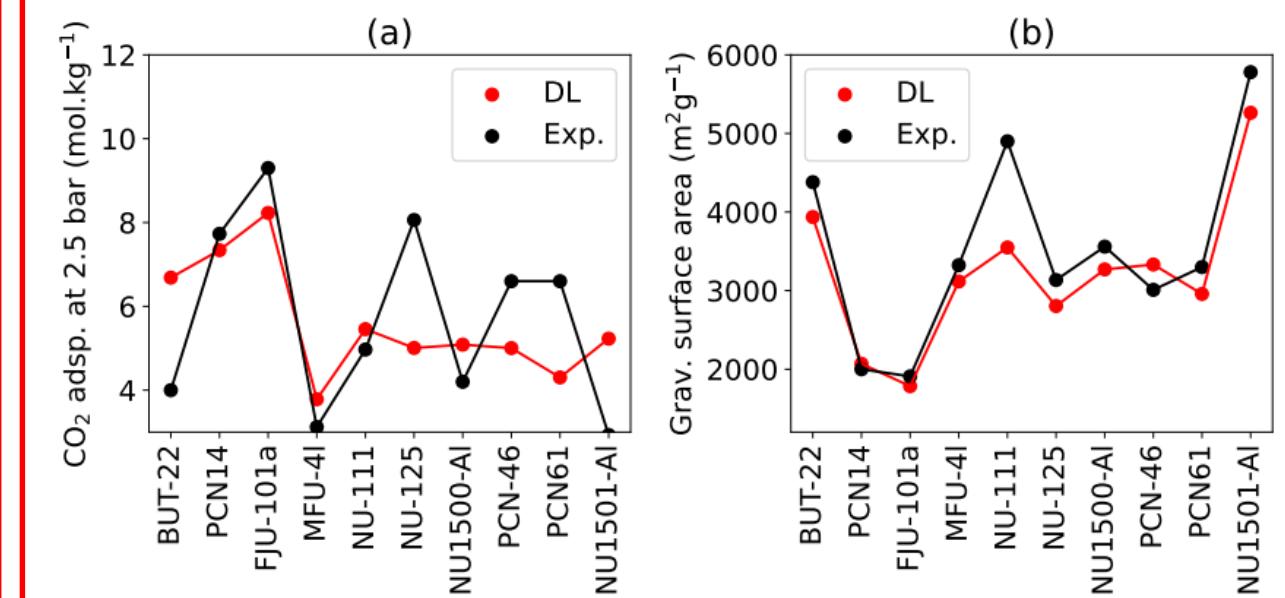
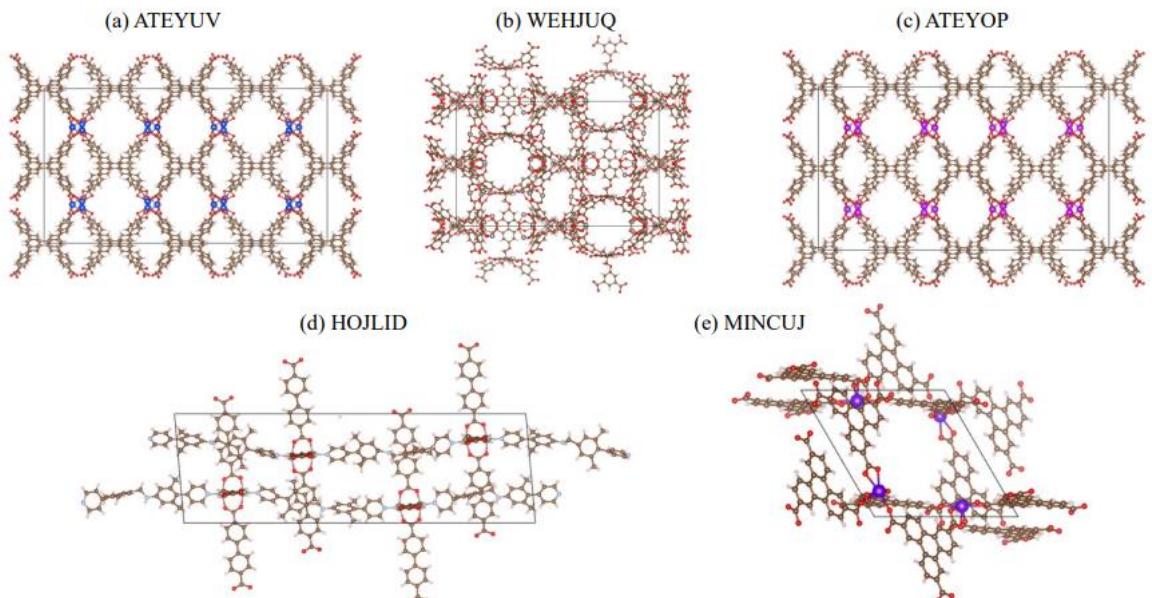
BCS Superconductors



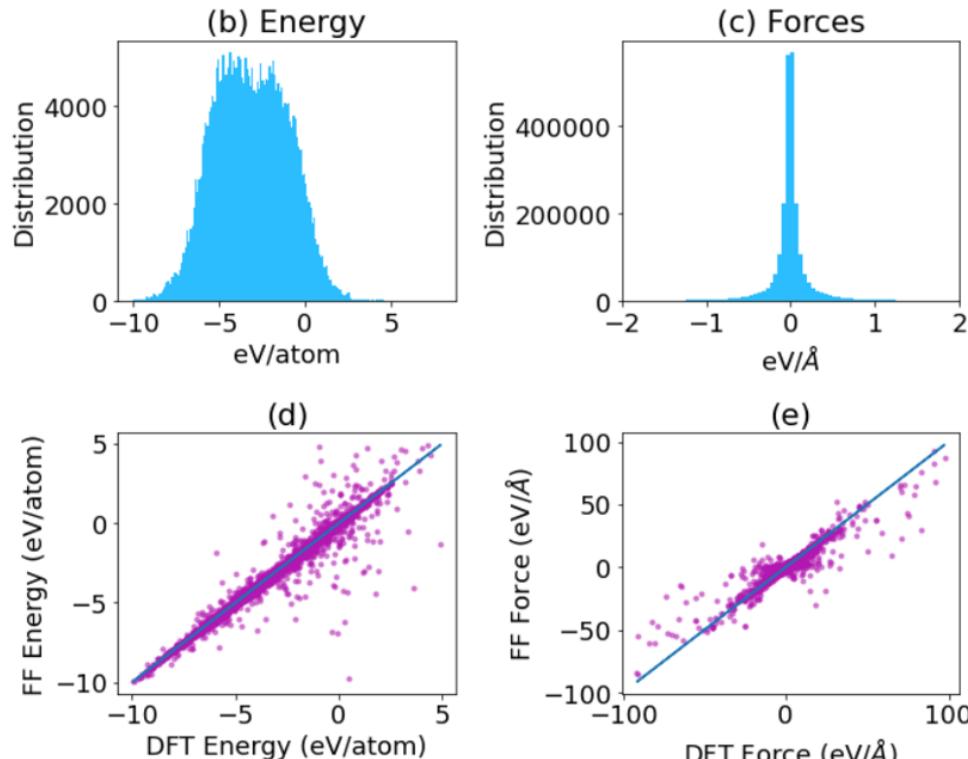
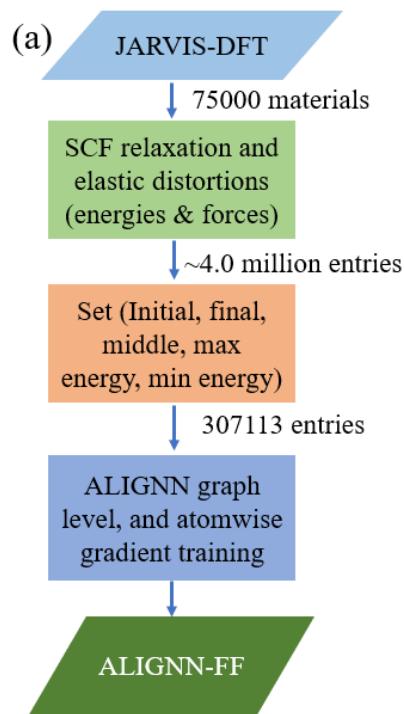
- Prediction on 10 % test data
- 8293 out of 431778 materials in COD as superconductors
- First predicting Eliashberg function, then $T_c \rightarrow 6\%$ improvement
- ALIGNN for both scalar and spectral learning

CO_2 capture & MOFs

DL model for predicting CO_2 adsorption in MOFs (using hMOF GCMC data)



Unified GNN Force-field



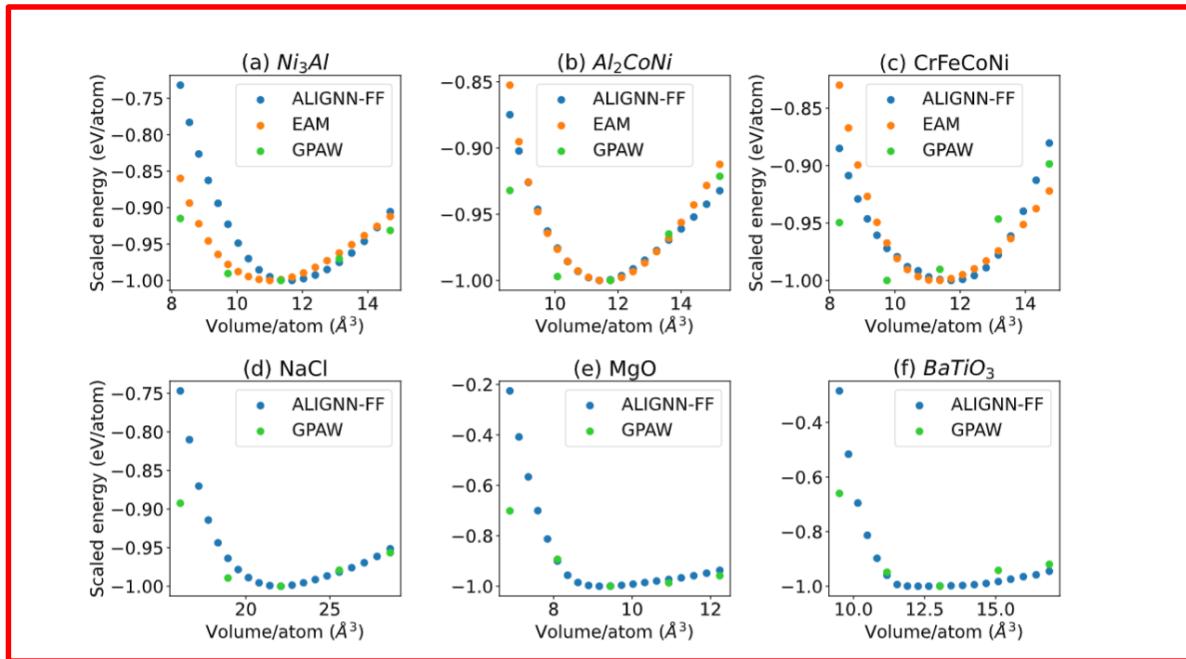
Weight	MAE-Energies (eV/atom)	MAE-Forces (eV/Å)
0.1	0.034	0.092
0.5	0.044	0.089
1.0	0.051	0.088
5.0	0.082	0.054
10.0	0.086	0.047

- Simulate any combination of 89 elements from the periodic table
- Structure optimization & Phonon property predictions

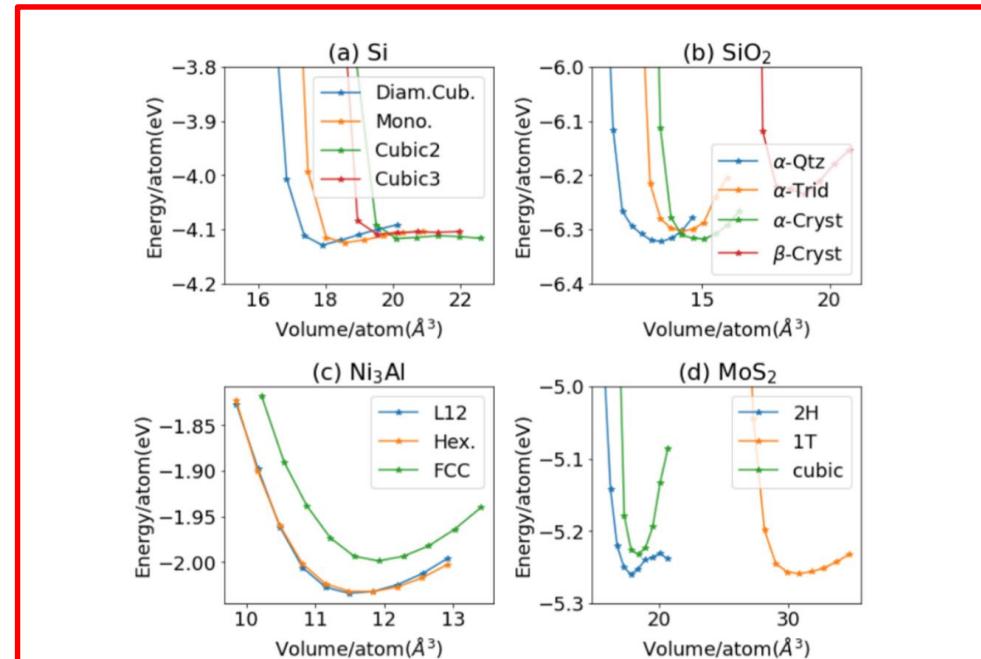
Examples

Notebooks	Google Colab	Descriptions
Regression model	Open in Colab	Examples for developing single output regression model for exfoliation energies of 2D materials.
MLFF	Open in Colab	Examples of training a machine learning force field for Silicon.
ALIGNN-FF Relaxer+EV_curve+Phonons+Interface gamma_surface+Interface separation	Open in Colab	Examples of using pre-trained ALIGNN-FF force-field model.
Miscellaneous tasks	Open in Colab	Examples for developing single output (such as formation energy, bandgaps) or multi-output (such as phonon DOS, electron DOS) Regression or Classification (such as metal vs non-metal), Using several pretrained models.

Unified GNN Force-field



Genetic algorithm



Digital
Discovery

PAPER

Check for updates

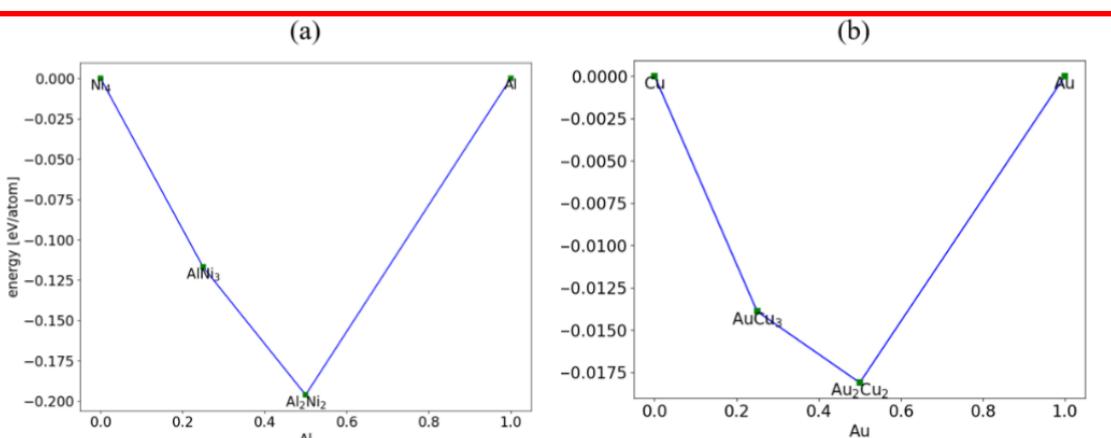
Cite this: DOI: 10.1039/d2dd00096b



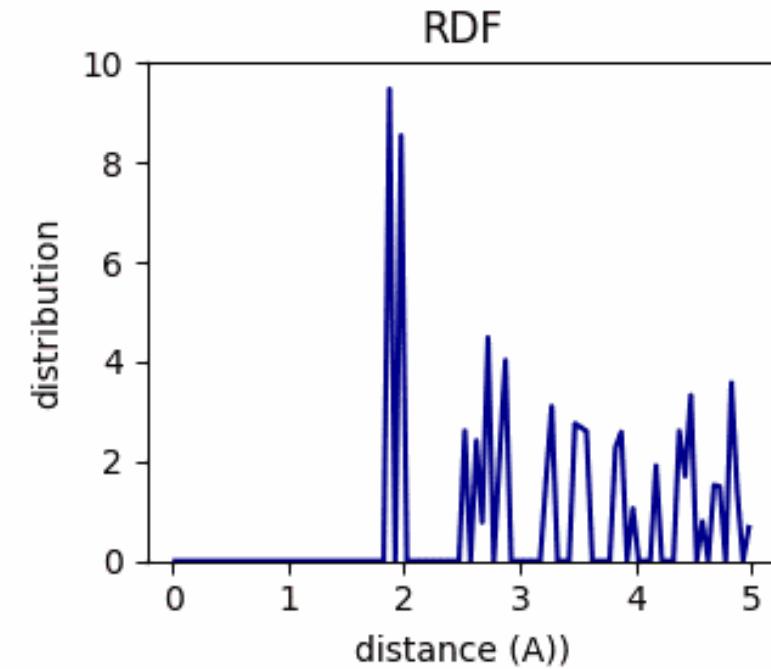
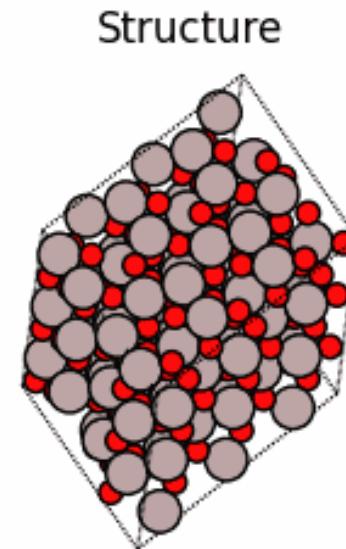
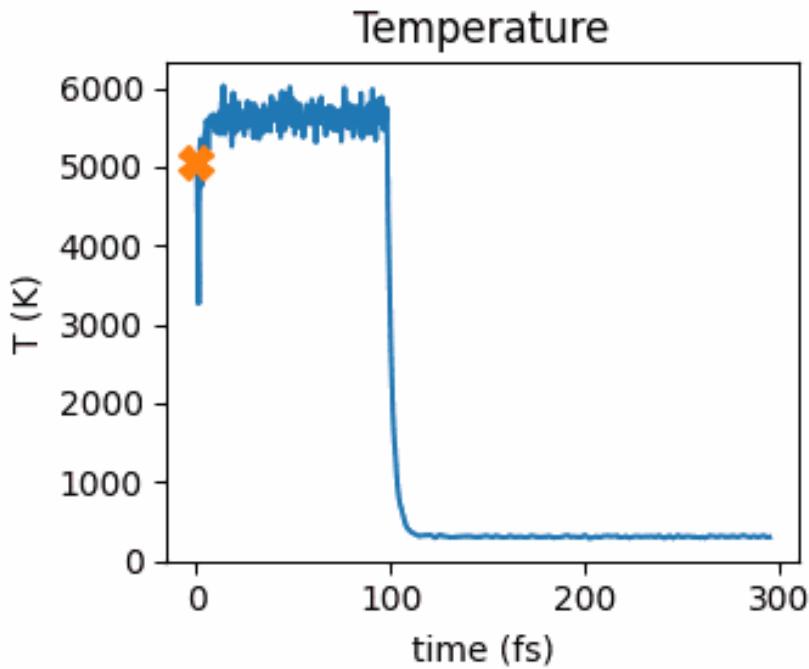
[View Article Online](#)
[View Journal](#)

Unified graph neural network force-field for the periodic table: solid state applications

Kamal Choudhary, ^{a,b} Brian DeCost, ^c Lily Major, ^{de} Keith Butler, ^e Jeyan Thiyagalingam ^e and Francesca Tavazza ^c

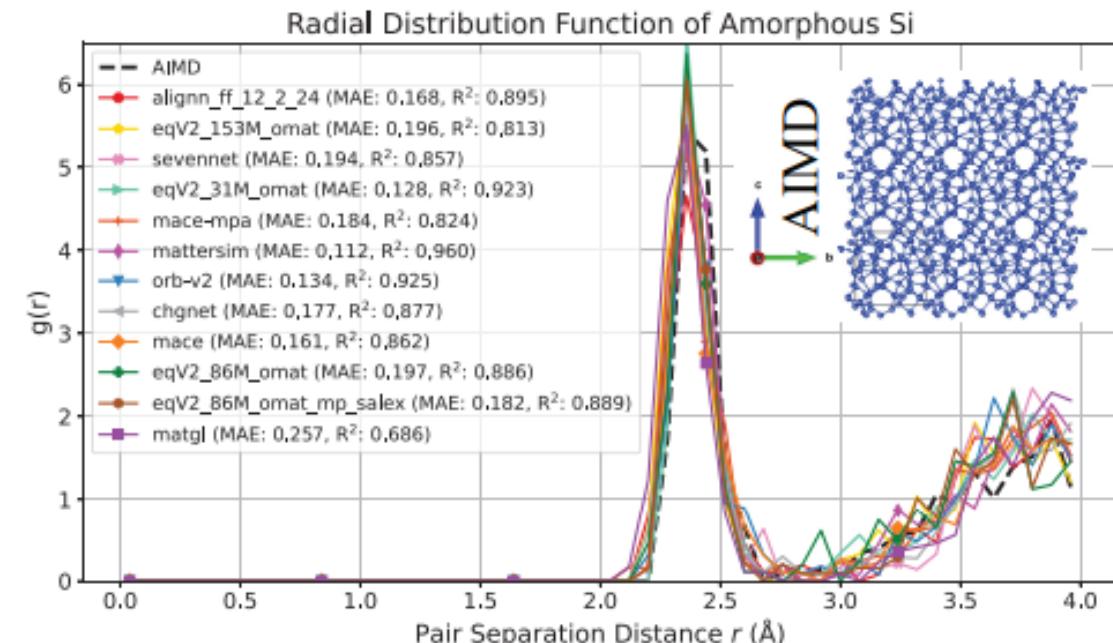


Al_2O_3 Melt-Quench Simulation Example



CHIPS-FF

uMLFF Type	err_a Å	err_b Å	err_c Å	err_vol Å ³	err_kv GPa	err_c11 GPa	err_c44 GPa	err_phonon cm ⁻¹	Time seconds
alignn_ff	0.087	0.102	0.137	9.95	122	196	74	150	8.8 x 10 ⁴
chgnnet	0.046	0.049	0.109	3.58	85	59	45	68	3.5 x 10 ⁴
eqV2_153M_omat	0.036	0.040	0.111	2.90	117	70	32	49	1.6 x 10 ²
eqV2_31M_omat	0.029	0.032	0.099	3.05	115	78	37	48	3.4 x 10 ⁵
eqV2_31M_omat_mp_salex	0.028	0.030	0.096	3.17	112	47	41	48	2.7 x 10 ⁵
eqV2_86M_omat	0.033	0.040	0.094	3.41	116	85	35	49	9.7 x 10 ⁵
eqV2_86M_omat_mp_salex	0.027	0.030	0.100	3.14	114	45	35	48	7.2 x 10 ⁵
mace	0.035	0.038	0.084	3.00	94	43	38	58	7.2 x 10 ⁴
mace-alexandria	0.081	0.086	0.206	6.05	76	96	45	81	2.5 x 10 ⁵
mace-mpa	0.044	0.046	0.098	3.57	110	35	34	49	8.4 x 10 ⁴
matgl	0.052	0.057	0.128	3.50	83	64	41	78	1.7 x 10 ⁴
matgl-direct	0.044	0.046	0.106	2.88	84	73	50	85	2.5 x 10 ⁴
mattersim	0.031	0.033	0.110	3.18	110	32	32	48	5.9 x 10 ⁴
orb-d3-v2	0.031	0.030	0.048	1.90	130	74	74	52	2.7 x 10 ⁴
orb-v2	0.023	0.025	0.159	3.38	119	80	74	49	2.4 x 10 ⁴
sevennet	0.035	0.038	0.100	3.43	103	46	37	53	5.7 x 10 ⁴



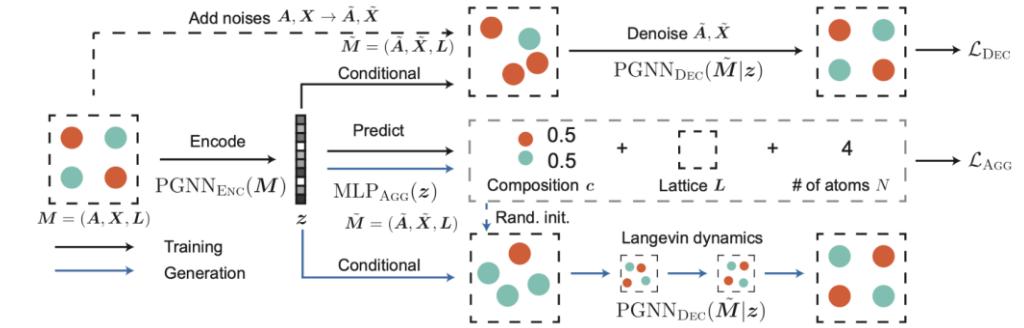
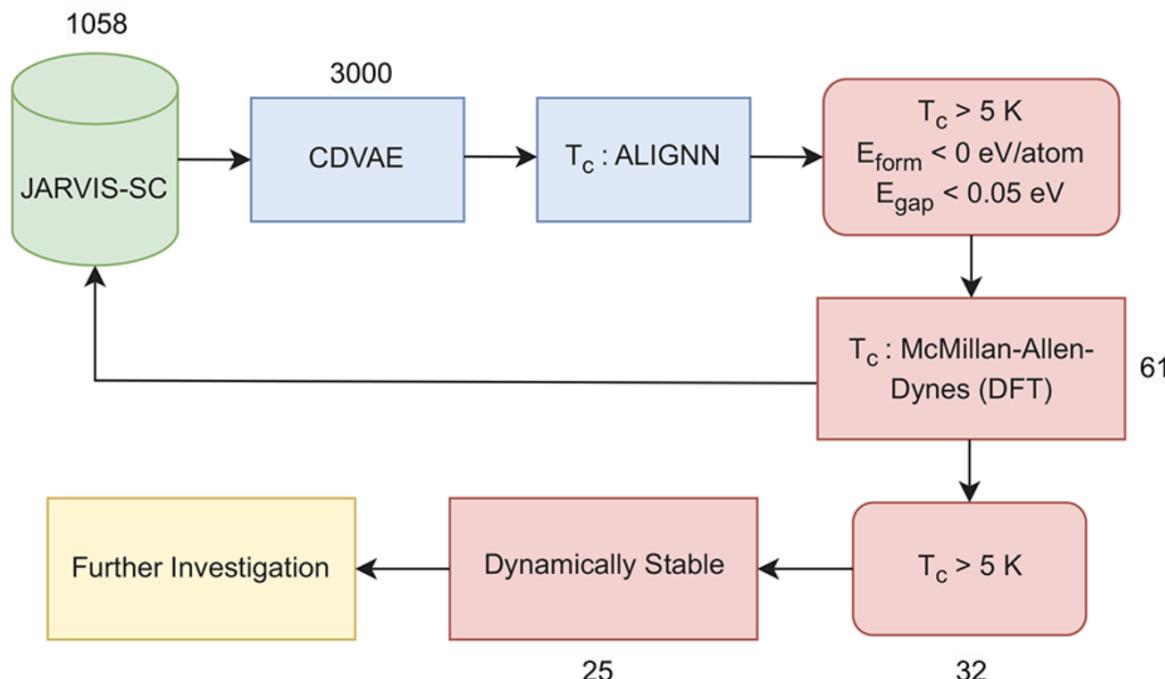
CHIPS-FF: Evaluating Universal Machine Learning Force Fields for Material Properties

Daniel Wines, Kamal Choudhary

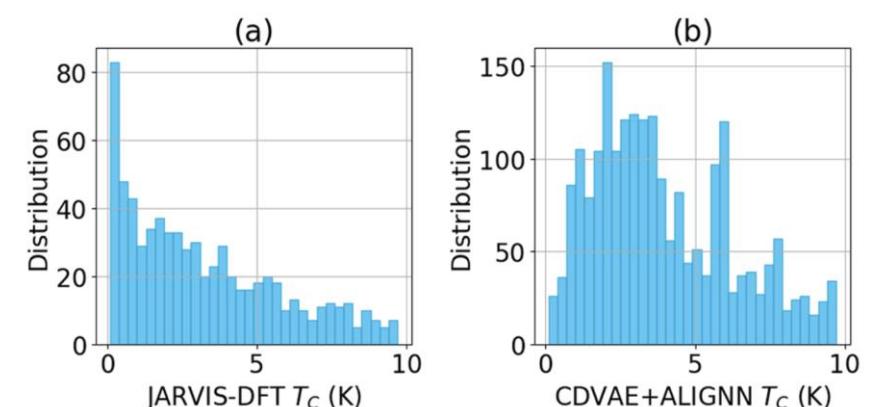
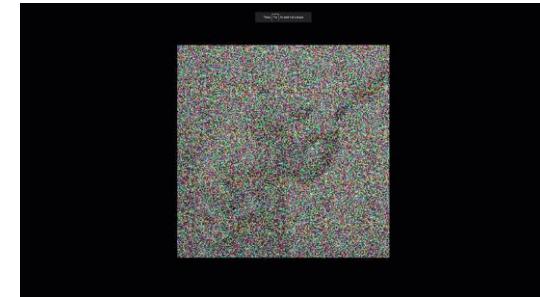
<https://arxiv.org/abs/2412.10516>

Inverse Design (superconductors)

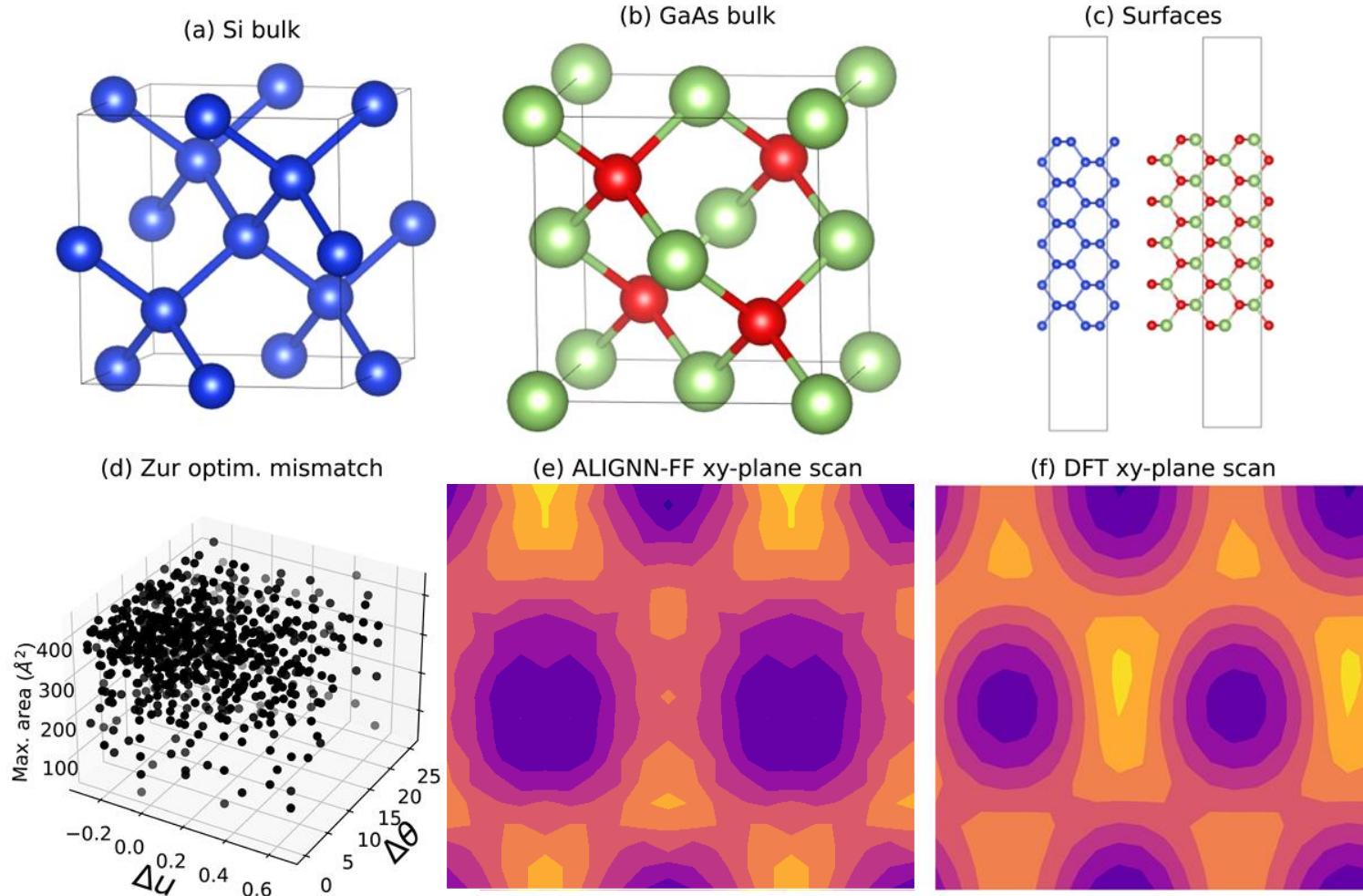
- Crystal diffusion variational autoencoder (CDVAE) for inverse design
- Trained on JARVIS-SC data
- Properties screened with ALIGNN, verified with DFT



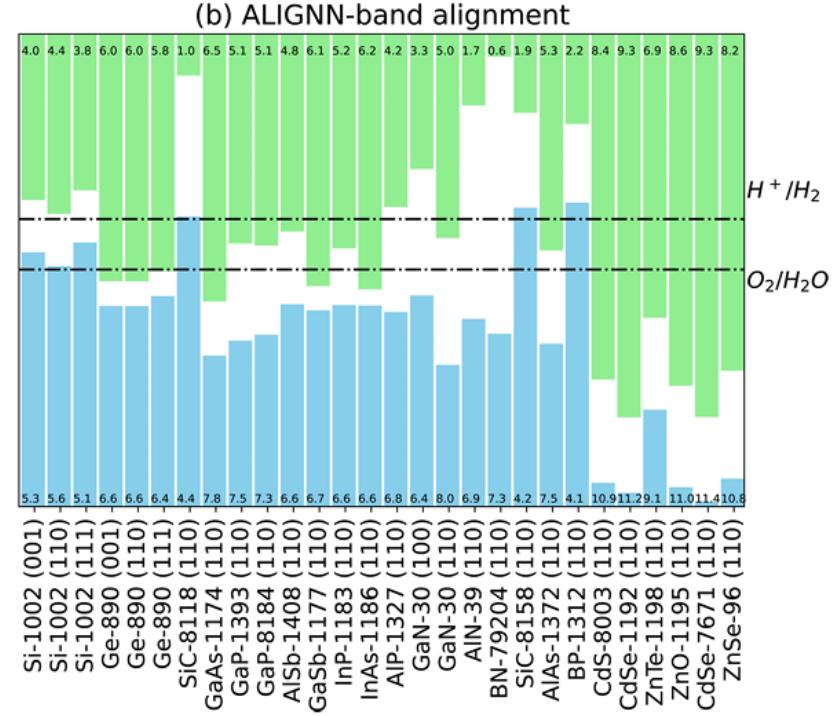
Xie et al. arXiv:2110.06197 (2022)



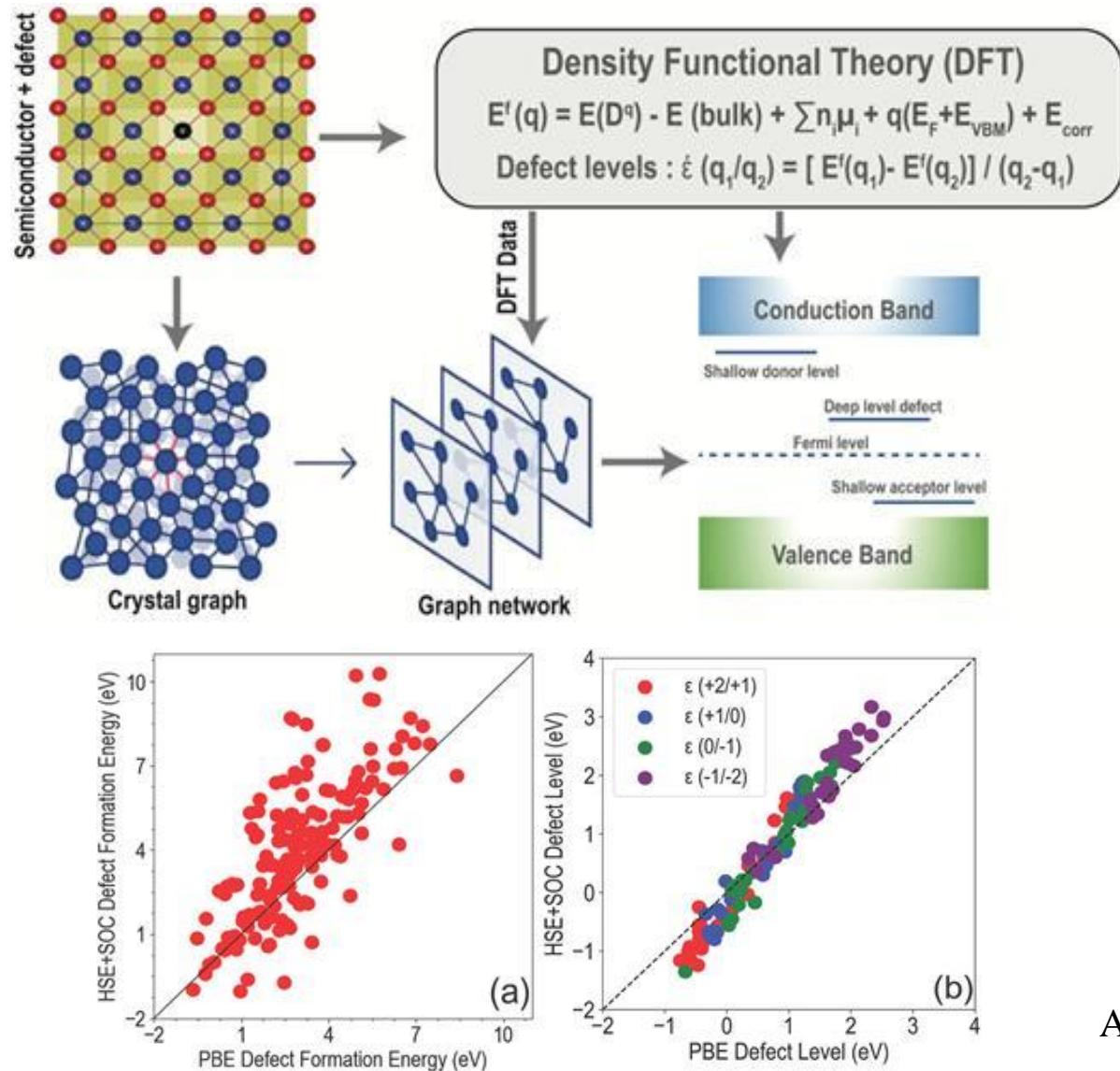
InterMat: Interface Materials Design (Semicons)



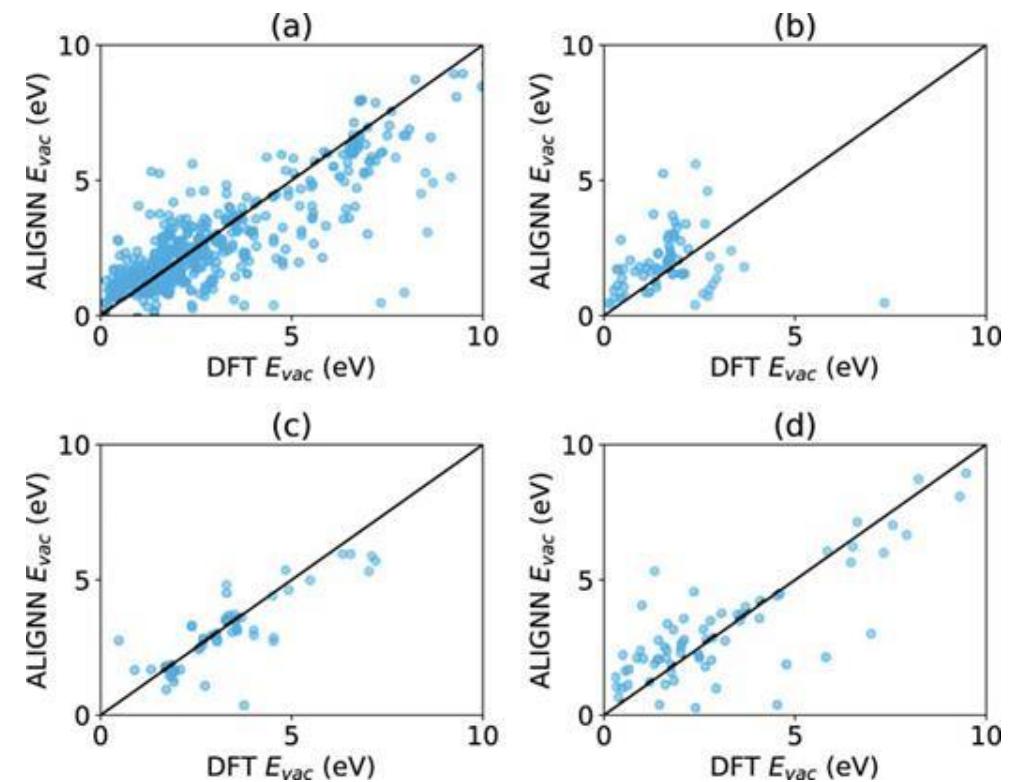
- ALIGNN+DFT for accelerated interface design
- Benchmarked band-offset predictions
- General workflow for materials design



DefectMat: Defect Materials Design

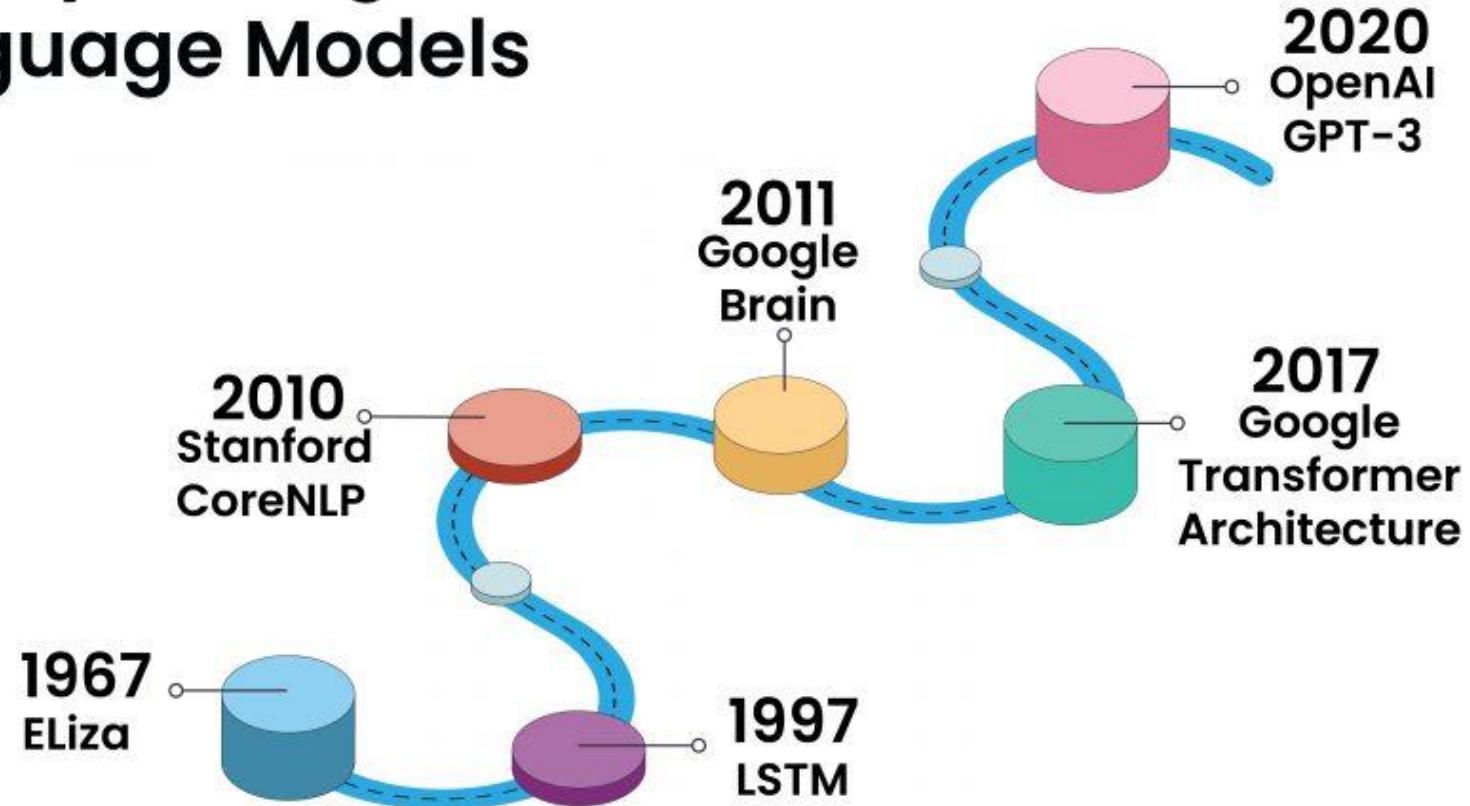


- ALIGNN+DFT for accelerated defect design
- Benchmarked vacancy formation predictions
- Neutral and charged defects

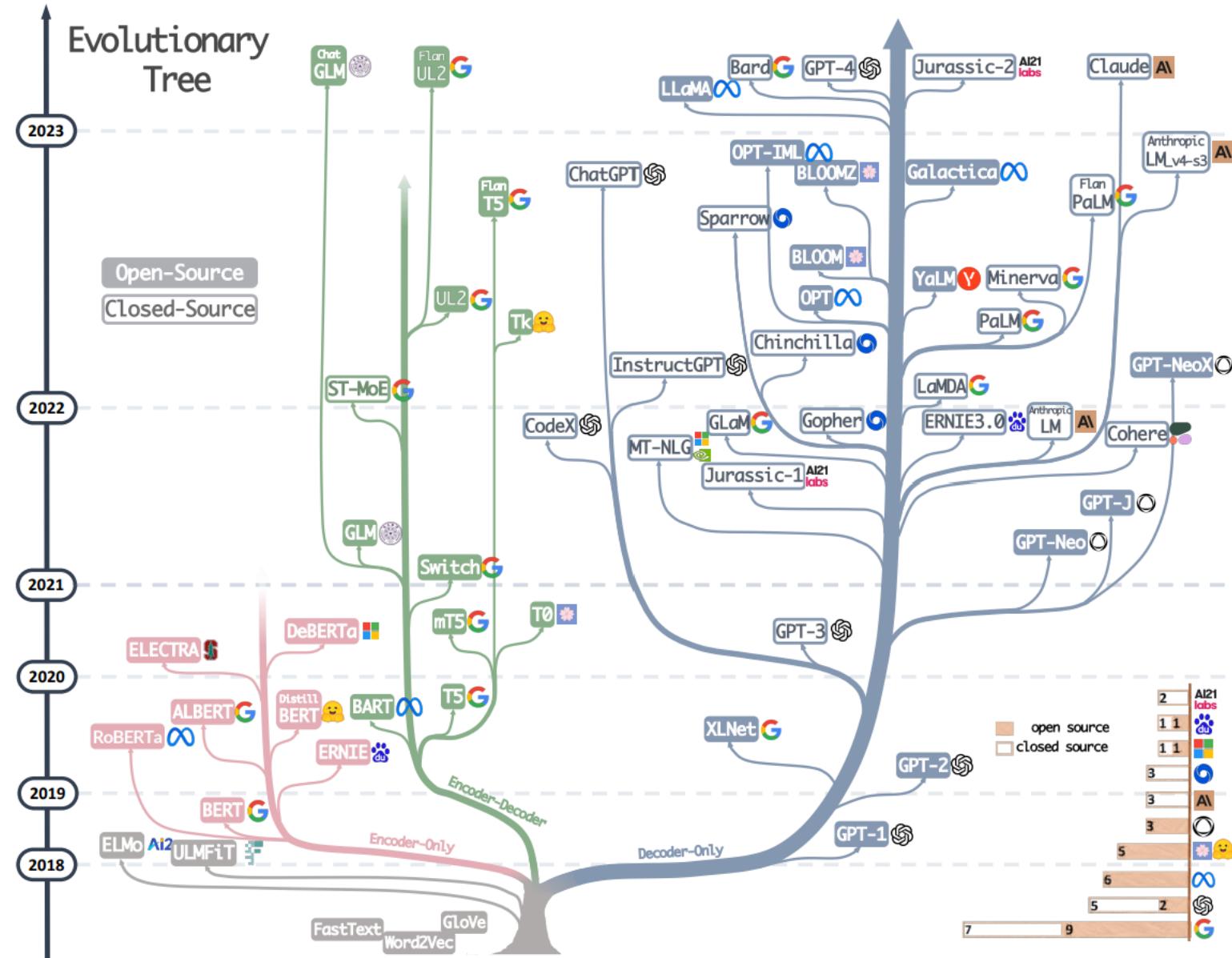


Large Language Models

History of Large Language Models



LLM Evolution



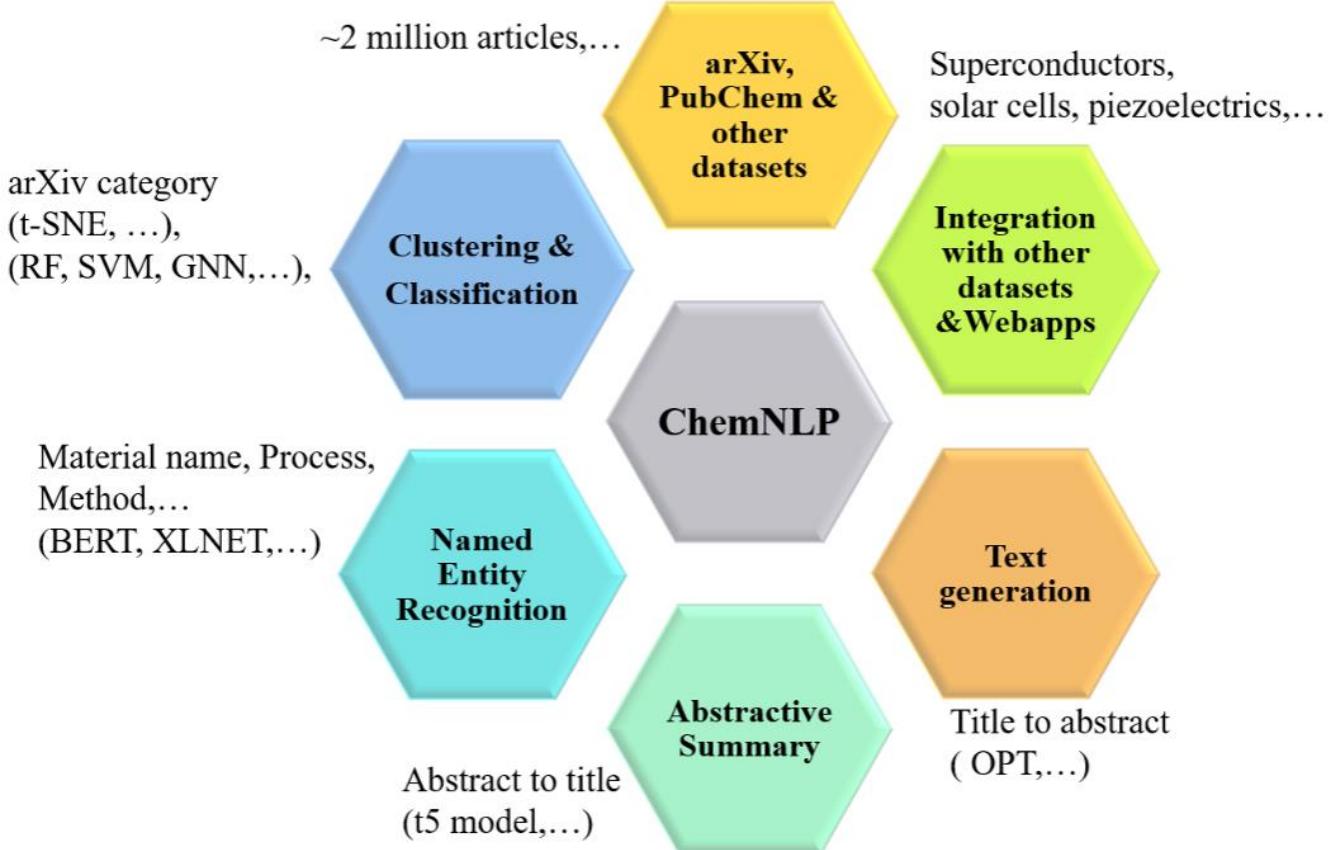
LLM Applications for MSE

DOI: [10.1039/D3DD00113J](https://doi.org/10.1039/D3DD00113J) (Perspective) *Digital Discovery*, 2023, 2, 1233-1250

14 examples of how LLMs can transform materials science and chemistry: a reflection on a large language model hackathon[†]

Kevin Maik Jablonka  *^a, Qianxiang Ai  ^b, Alexander Al-Feghali  ^c, Shruti Badhwar  ^d, Joshua D. Bocarsly  ^e, Andres M. Bran  ^{fg}, Stefan Bringuer  ^h, L. Catherine Brinson ⁱ, Kamal Choudhary ^j, Defne Circi ⁱ, Sam Cox ^k, Wibe A. de Jong ^l, Matthew L. Evans ^{mn}, Nicolas Gastellu ^c, Jerome Genzling ^c, María Victoria Gil ^o, Ankur K. Gupta ^l, Zhi Hong ^p, Alishba Imran ^q, Sabine Kruschwitz ^r, Anne Labarre ^c, Jakub Lála ^s, Tao Liu ^c, Steven Ma ^c, Sauradeep Majumdar ^a, Garrett W. Merz ^t, Nicolas Moitessier ^c, Elias Moubarak ^a, Beatriz Mourino ^a, Brenden Pelkie ^u, Michael Pieler ^{vw}, Mayk Caldas Ramos ^k, Bojana Ranković ^{fg}, Samuel G. Rodrigues ^s, Jacob N. Sanders ^x, Philippe Schwaller ^{fg}, Marcus Schwarting ^y, Jiale Shi ^b, Berend Smit ^o, Ben E. Smith ^e, Joren Van Herck ^a, Christoph Völker ^r, Logan Ward ^z, Sean Warren ^c, Benjamin Weiser ^c, Sylvester Zhang ^c, Xiaoqi Zhang ^a, Ghezal Ahmad Zia ^r, Aristana Scourtas ^{aa}, K. J. Schmidt ^{aa}, Ian Foster ^{ab}, Andrew D. White ^k and Ben Blaiszik ^{*aa}

ChemNLP



usnistgov / chemnlp Public

Code Issues Pull requests Actions Projects

main 5 branches 1 tag

The Journal of Physical Chemistry C > Vol 127/Issue 35 > Article
Subscribed

C: PHYSICAL PROPERTIES OF MATERIALS AND INTERFACES | August 25, 2023

ChemNLP: A Natural Language-Processing-Based Library for Materials Chemistry Text Data

Kamal Choudhary*, and Mathew L. Kelley

Open PDF open URL

Cite Share Jump

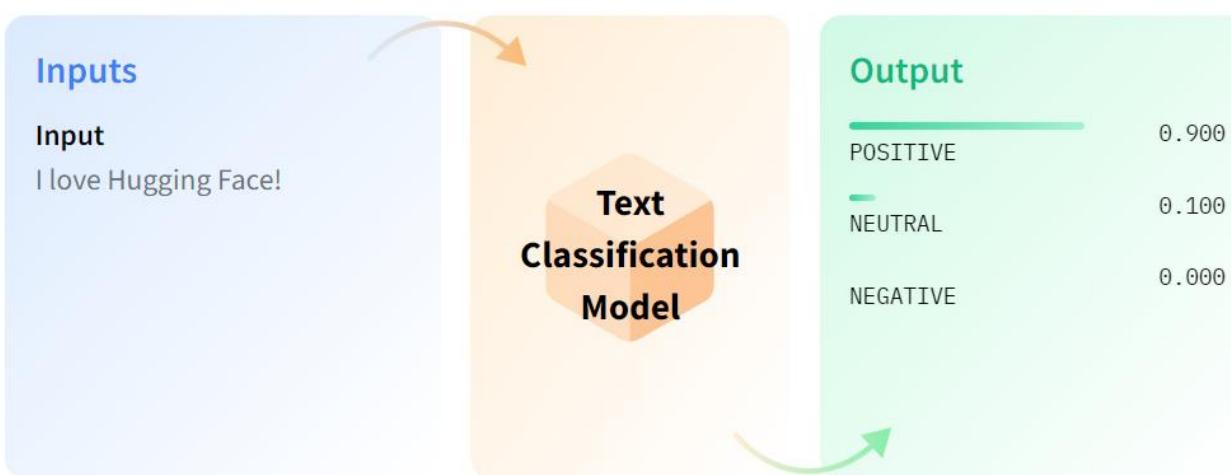
Text classification & Token classification

← → ⌂ huggingface.co/tasks/text-classification

Tasks

Text Classification

Text Classification is the task of assigning a label or class to a given text. Some use cases are sentiment analysis, natural language inference, and assessing grammatical correctness.

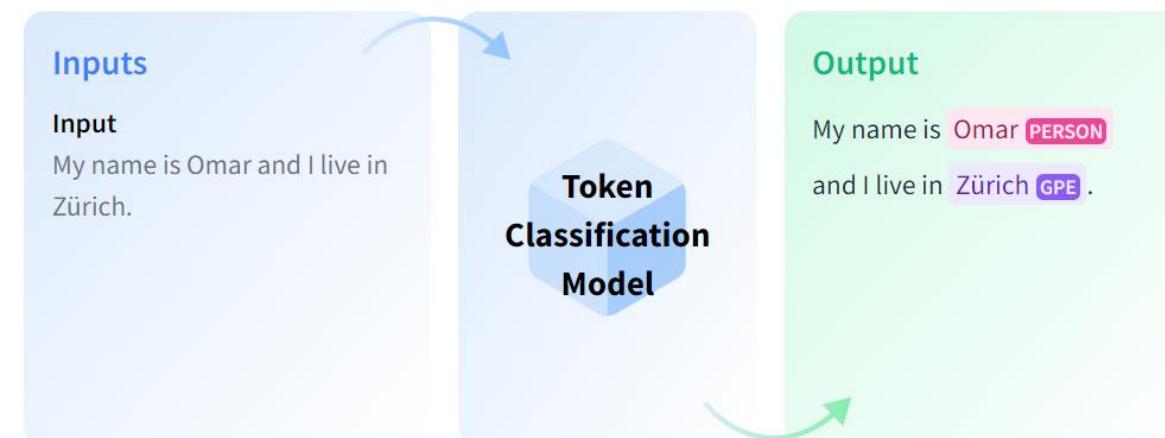


← → ⌂ huggingface.co/tasks/token-classification

Tasks

Token Classification

Token classification is a natural language understanding task in which a label is assigned to some tokens in a text. Some popular token classification subtasks are Named Entity Recognition (NER) and Part-of-Speech (PoS) tagging. NER models could be trained to identify specific entities in a text, such as dates, individuals and places; and PoS tagging would identify, for example, which words in a text are verbs, nouns, and punctuation marks.



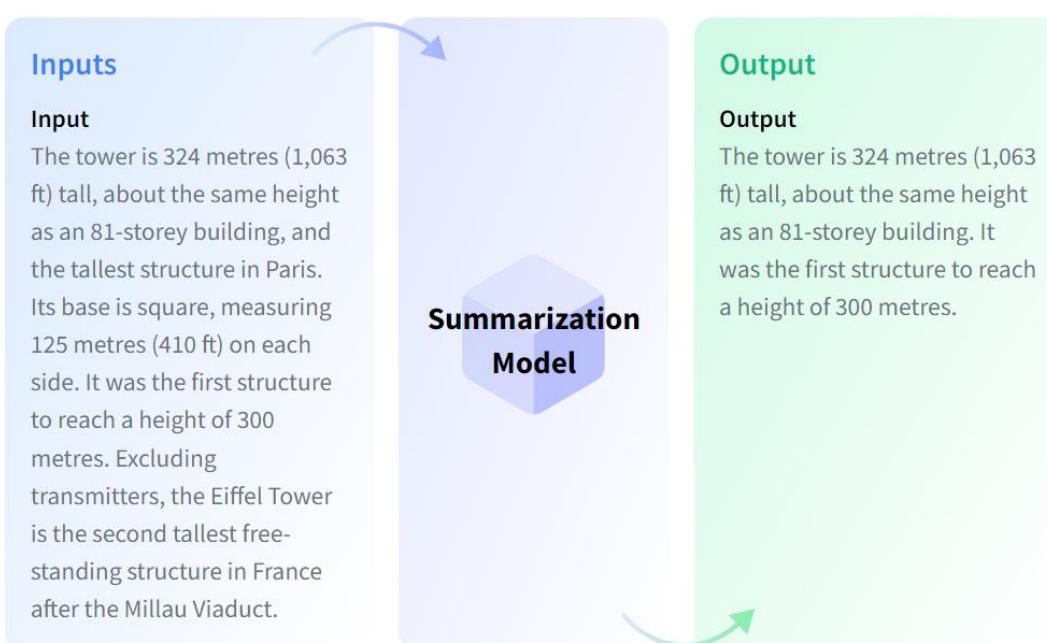
Text summarization & Text-generation

← → ⌛ huggingface.co/tasks/summarization

< Tasks

📄 Summarization

Summarization is the task of producing a shorter version of a document while preserving its important information. Some models can extract text from the original input, while other models can generate entirely new text.

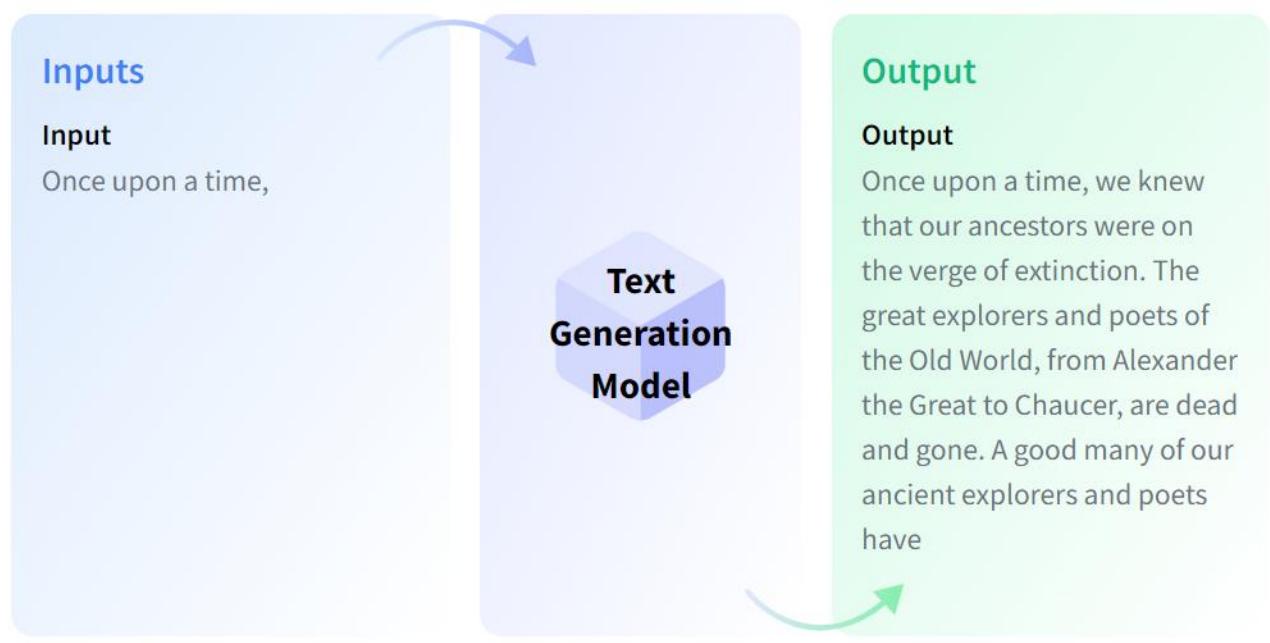


← → ⌛ huggingface.co/tasks/text-generation

< Tasks

📝 Text Generation

Generating text is the task of producing new text. These models can, for example, fill in incomplete text or paraphrase.



Conventional NLP: TFIDF

← → 🔍 scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

scikit learn Install User Guide API Examples Community More ▾

Prev Up Next

scikit-learn 1.3.0 Other versions

Please cite us if you use the software.

sklearn.feature_extraction.text.
TfidfVectorizer
TfidfVectorizer
Examples using
sklearn.feature_extraction.text.T

sklearn.feature_extraction.text.TfidfVectorizer

```
class sklearn.feature_extraction.text.TfidfVectorizer(*, input='content', encoding='utf-8',  
decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word',  
stop_words=None, token_pattern='(\\u)\\b\\w\\w+\\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None,  
vocabulary=None, binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True,  
sublinear_tf=False) ↴
```

[source]

Convert a collection of raw documents to a matrix of TF-IDF features.

In [21]:

```
sentence = "The oldest human fossil is the skull discovered in the Cave of Aroeira in Almonda."  
  
TfidfVec = TfidfVectorizer()  
tfidf = TfidfVec.fit_transform([sentence])  
  
cols = TfidfVec.get_feature_names()  
matrix = tfidf.todense()  
pd.DataFrame(matrix, columns = cols, index=["Tf-Idf"])
```

Out[21]:

	almonda	aroeira	cave	discovered	fossil	human	in	is	of	oldest	skull	the
Tf-Idf	0.208514	0.208514	0.208514	0.208514	0.208514	0.208514	0.417029	0.208514	0.208514	0.208514	0.208514	0.625543

TFIDF score for term i in document j = $TF(i,j) * IDF(i)$

where

$IDF = \text{Inverse Document Frequency}$

$TF = \text{Term Frequency}$

$$TF(i,j) = \frac{\text{Term } i \text{ frequency in document } j}{\text{Total words in document } j}$$

$$IDF(i) = \log_2 \left(\frac{\text{Total documents}}{\text{documents with term } i} \right)$$

and

$t = \text{Term}$

$j = \text{Document}$

<https://www.kaggle.com/code/ashoksrinivas/nlp-with-tfidf-neural-networks>

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

<https://i.stack.imgur.com/mtmP6.png>

Transformers & “Attention Is All You Need”

arXiv > cs > arXiv:1706.03762

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

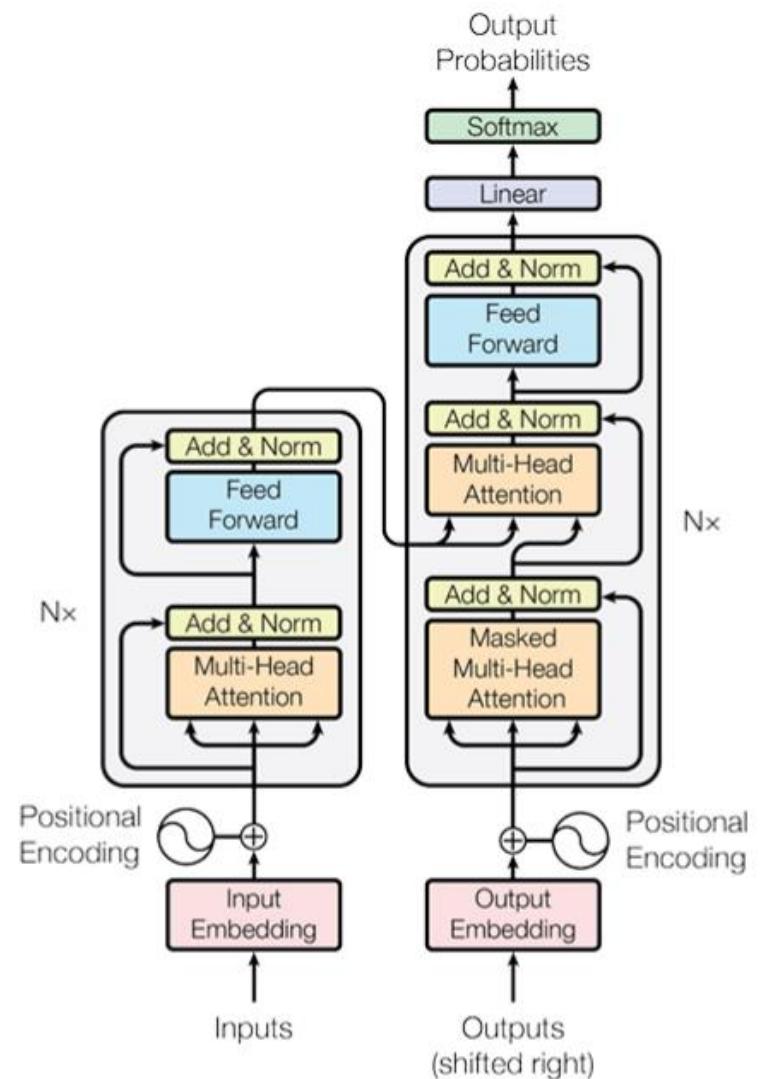
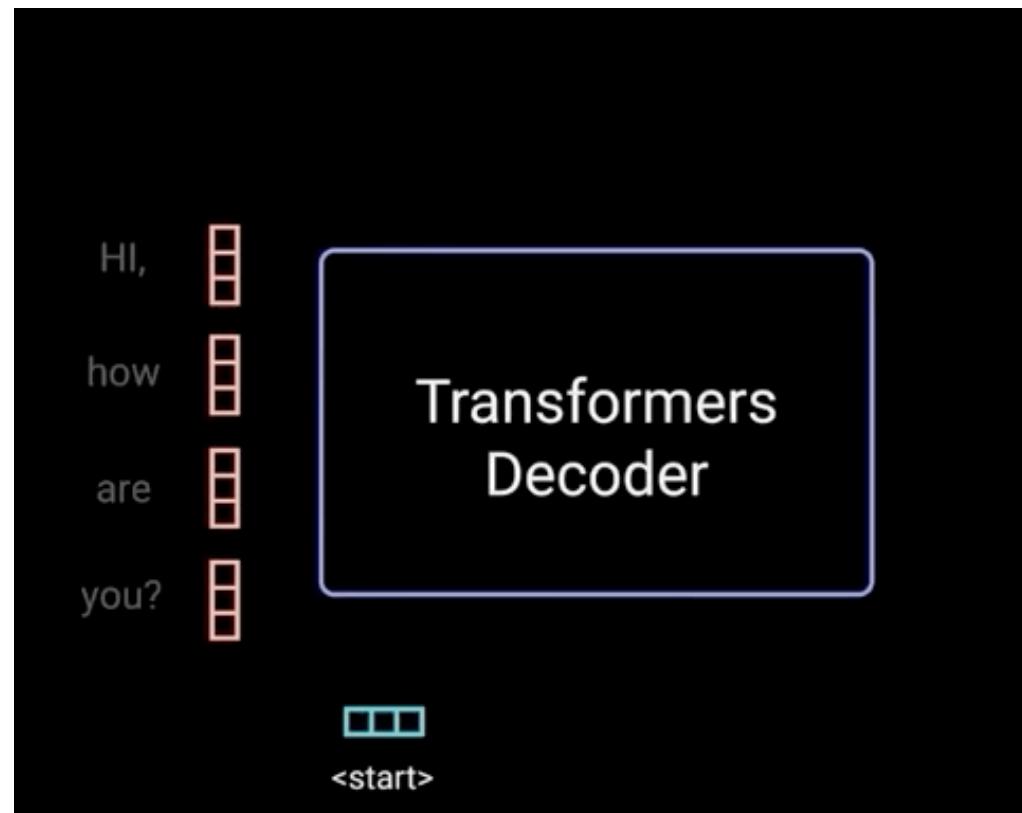
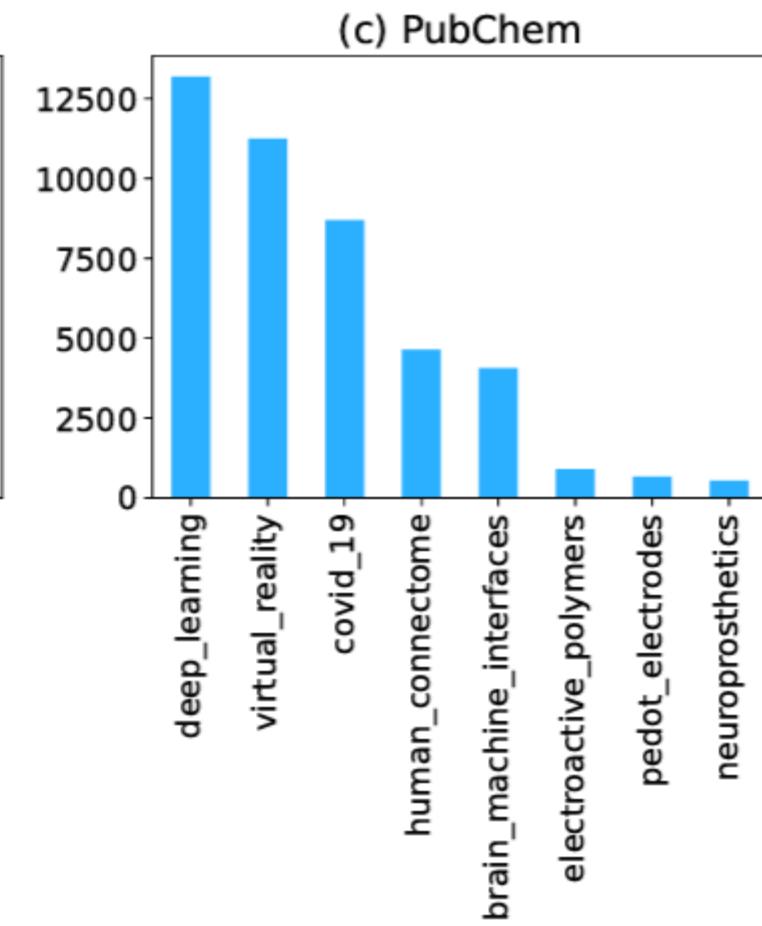
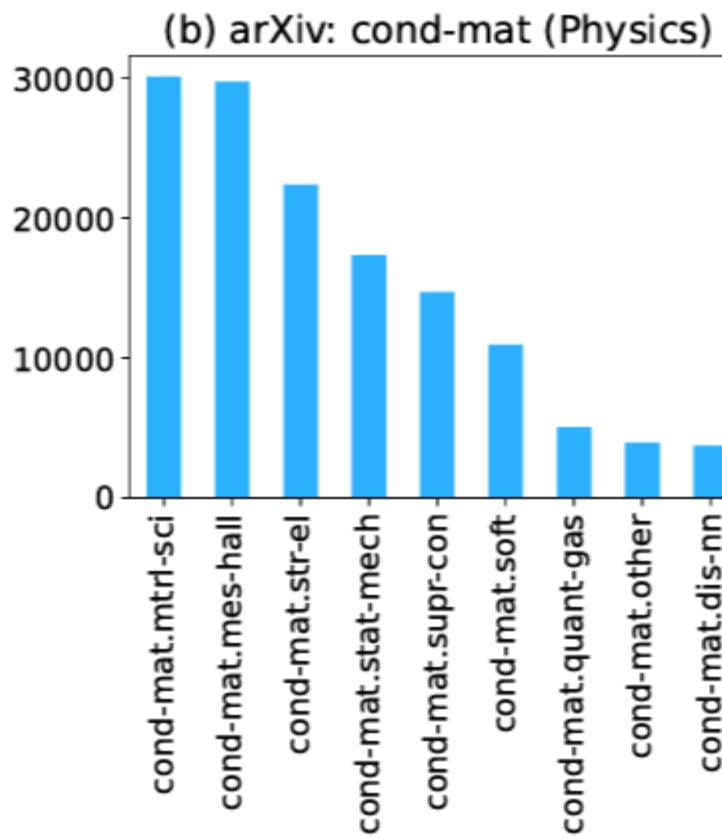
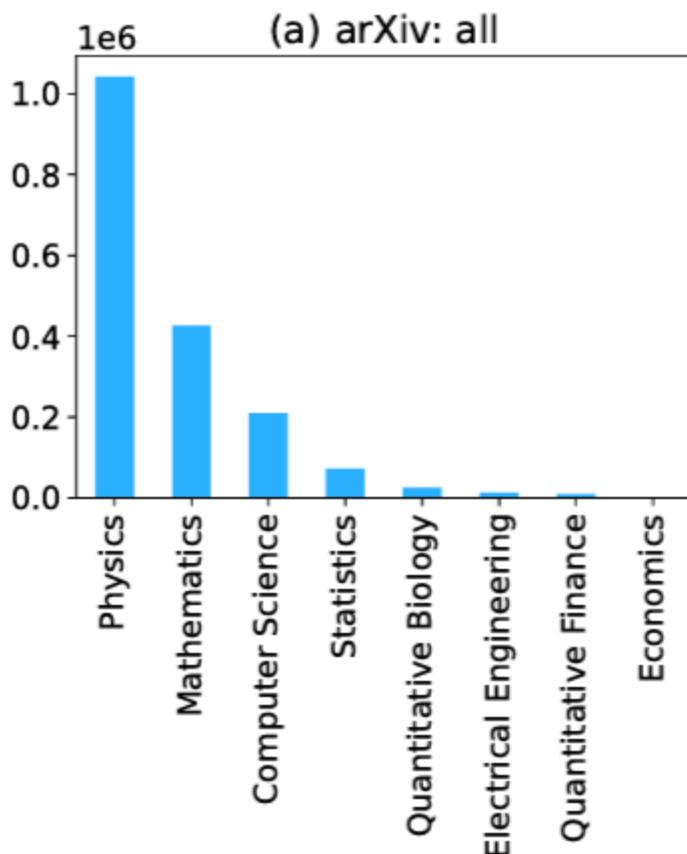


Figure 1: The Transformer - model architecture.

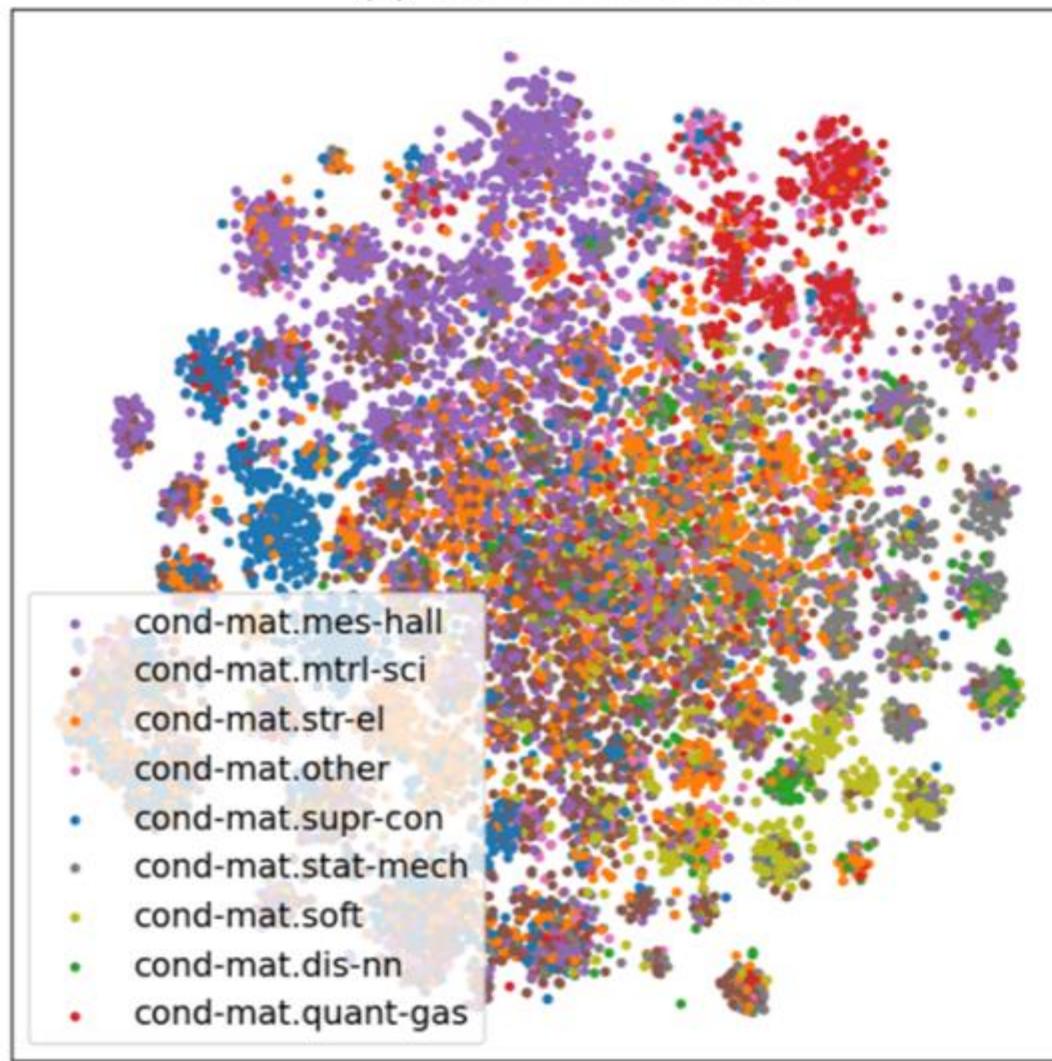
Much better than RNN, LSTM etc.
Attention: extremely long-term memory

ChemNLP Datasets

Exploratory data analysis (EDA)



(a) arXiv: cond-mat



(b) PubChem

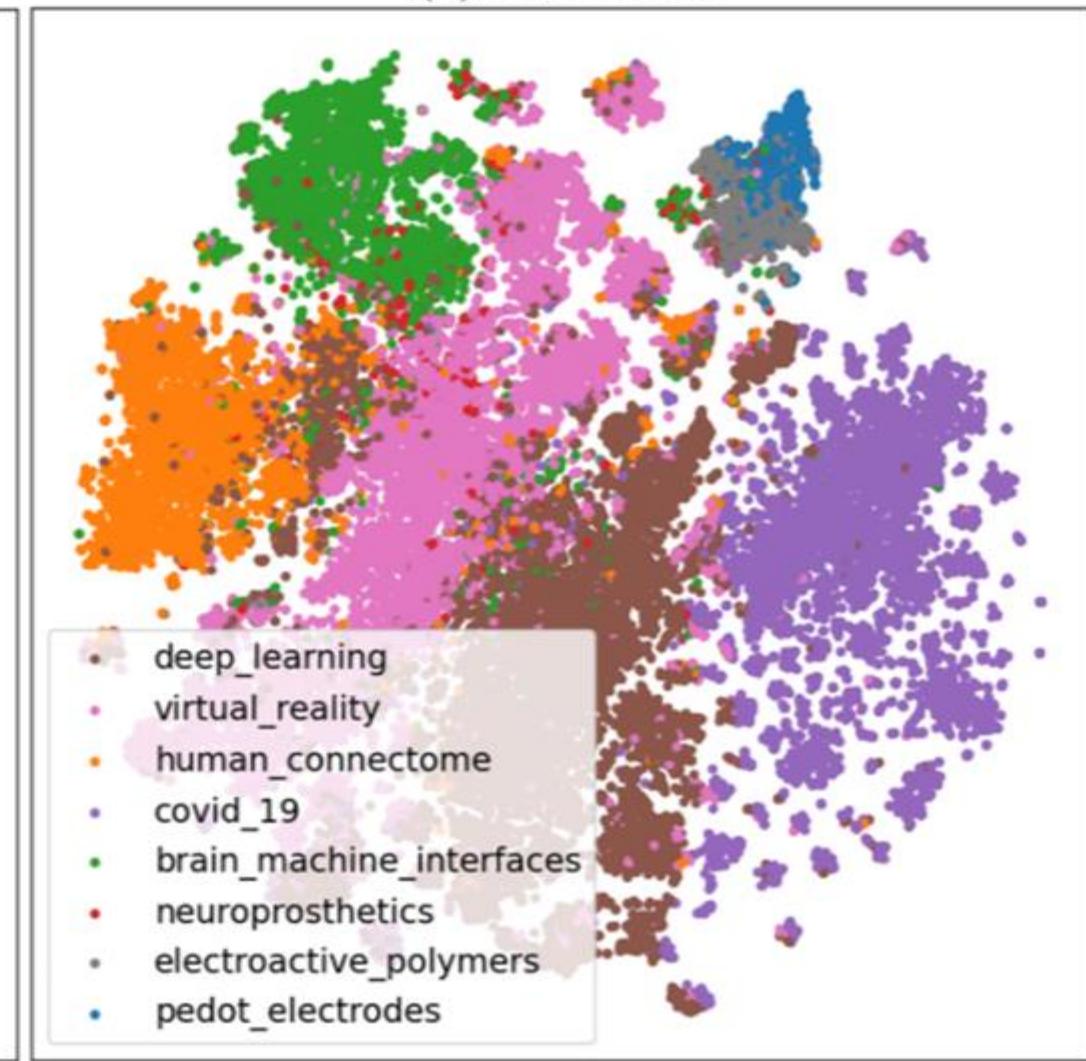


Figure 4: The t-SNE visualisations of the cond.mat articles in the arXiv and PubChem datasets.

Text classification

Accuracy

Table 1: Comparison of models for classifying articles.

arXiv			
Model	Title only	Abstract only	Title+Abstract
SVM	84.8	90.8	91.3
DistilBert	86.7	89.9	90.0
RF	86.9	88.4	88.6
LR	79.2	85.0	86.0
GNN	76.0	81.0	82.0

PubChem			
SVM	94.5	94.0	97.6
DistilBert	94.5	97.0	97.5
RF	94.3	94.0	96.7
LR	92.0	93.0	96.7
GNN	85.0	91.0	92.0

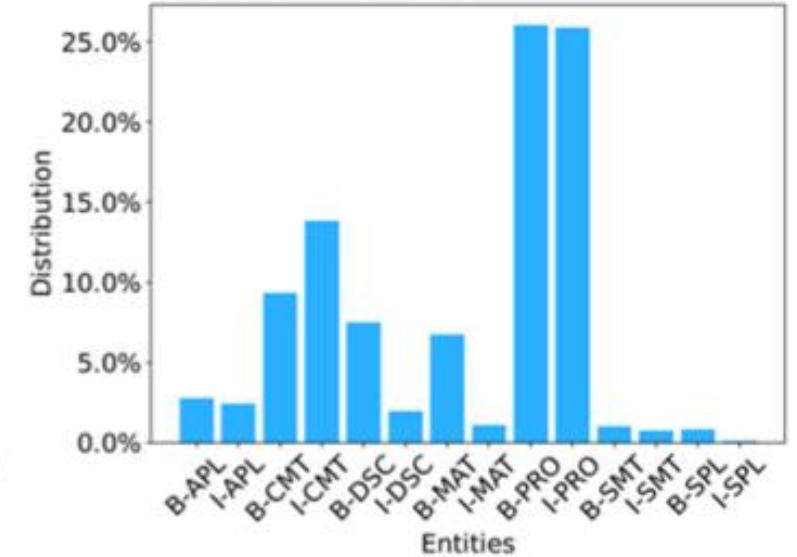
$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Named Entity Recognition TokenName classification

(a) Example NER application

The electrochemical **B-PRO** lithium **B-MAT**-ion extraction/insertion **I-PRO** properties **I-PRO** of two spinel **B-SPL** lithium **B-MAT** manganese **I-MAT** oxides **I-MAT**, stoichiometric **B-DSC**, Li_{0.99}Mn₂O₄ **B-MAT** and lithium **B-MAT**-rich Li_{1.04}Mn_{1.93}O₄ **B-MAT** were studied. Their voltage **B-PRO** profile **I-PRO** and cyclic **B-CMT** voltammetry **I-CMT** all show two-step reaction for lithium **B-MAT**-ion extraction/insertion **I-PRO** at their 4 V plateau. The high-resolution **B-CMT** in situ X-ray **B-CMT** diffraction **I-CMT** (XRD) studies show both materials experience three cubic **B-SPL** phases, cubic **B-SPL** I, II, and III, during cycling at 4 V plateaus. Two two-phase coexistence regions were observed. These results show that phase transitions between cubic **B-SPL** I and II and between cubic **B-SPL** II and III are first-order transitions. The in situ XRD data also reveal that lithium **B-MAT**-rich composition does not suppress the phase transitions of spinel **B-SPL** between their three cubic **B-SPL** phases, but leads to the increase of variation range of the lattice parameters of the cubic **B-SPL** phases. This in turn reduces the jumping step of the lattice parameter between adjacent phases and increases the overlap of the adjacent phase. This study shows conclusively that the three-phase process is the intrinsic feature of spinel **B-SPL** Li_{1+y}Mn₂O₄ during lithium **B-MAT**-ion extraction/insertion **I-PRO** on the 4 V plateau.

(b) NER on arXiv cond-mat



The performance of the model is measured in terms of F1 score given by:

$$F_1 = \frac{2TP}{2TP + FP + TN} \quad (5)$$

87 % F1 score

where, TP, TN, FN, FP are True Positive, True Negative, False Negative, and False Positive instances respectively. Similar to accuracy, the maximum value of F1 is 100 %.

ChemNLP Webpage for Composition Search

<https://jarvis.nist.gov/jarvischemnlp/>

JARVIS-ChemNLP

Search Elements

Advanced search

Elements Ni-Al-
ArXiv-ID XYZ
Author cubic

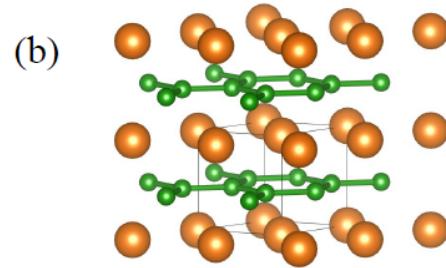
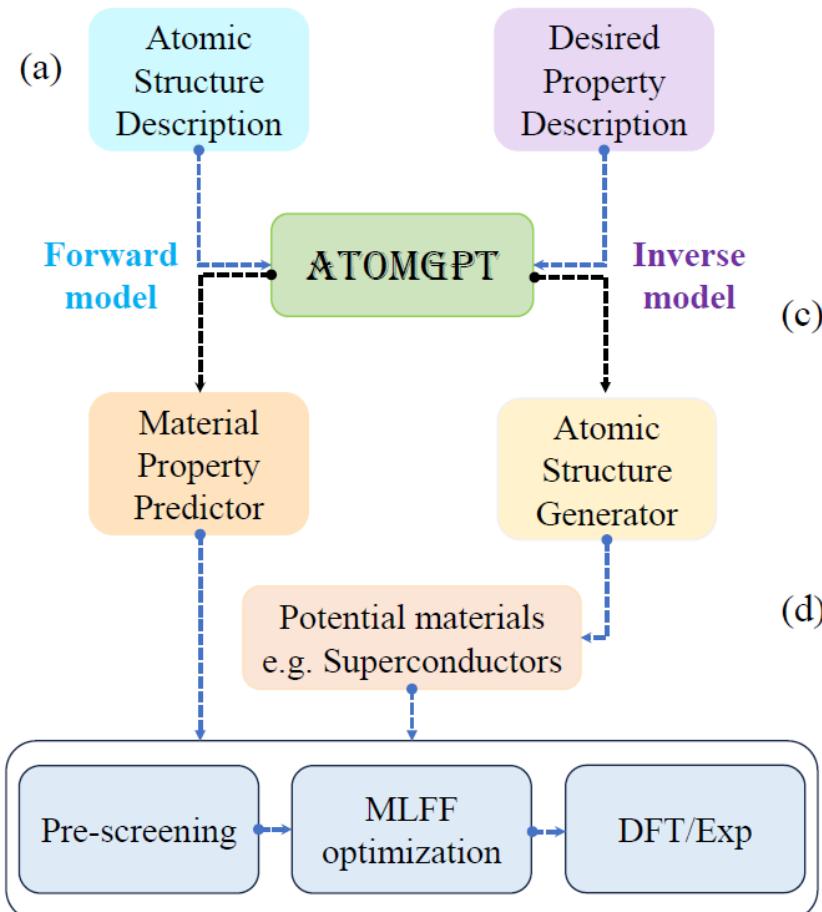
Results Found: 36

ArXiv-ID	Link	Title	Year	Authors
1411.3006	https://arxiv.org/abs/1411.3006	Mn as surfactant for the self-assembling of Al _x G _{3-x} O ₃ N _y heterostructures	2014	Thibaut Devillers, U Tian, Rajdeep Adhikari, Giulio Capuzzo and Alberto Bonanni
1102.4977	https://arxiv.org/abs/1102.4977	Polarization-balanced design of AlN/GaN heterostructures: Application to double-barrier structures	2011	Kristian Berland, Thorvald O. Andersson, Per Hyldegaard

The periodic table shows the following color mapping for the search query 'Al-Ga-N-':

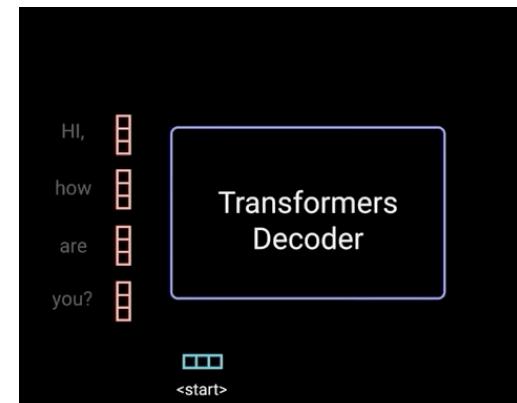
- Hydrogen (H) is green.
- Lithium (Li) is red.
- Boron (B) is yellow.
- Aluminum (Al) is purple.
- Gallium (Ga) is grey.
- Nitrogen (N) is light green.
- Phosphorus (P) is light green.
- Sulfur (S) is light green.
- Chlorine (Cl) is orange.
- Neon (Ne) is red.
- Scandium (Sc) is blue.
- Titanium (Ti) is blue.
- Vanadium (V) is blue.
- Chromium (Cr) is blue.
- Manganese (Mn) is blue.
- Iron (Fe) is blue.
- Cobalt (Co) is blue.
- Nickel (Ni) is blue.
- Copper (Cu) is blue.
- Zinc (Zn) is blue.
- Rubidium (Rb) is red.
- Sr is orange.
- Yttrium (Y) is blue.
- Zirconium (Zr) is blue.
- Niobium (Nb) is blue.
- Molybdenum (Mo) is blue.
- Tantalum (Tc) is blue.
- Ruthenium (Ru) is blue.
- Rhenium (Re) is blue.
- Osmium (Os) is blue.
- Iridium (Ir) is blue.
- Pt is blue.
- Au is blue.
- Hg is blue.
- In is pink.
- Sn is pink.
- Sb is yellow.
- Te is yellow.
- Iodine (I) is orange.
- Xe is red.
- Cesium (Cs) is orange.
- Ba is orange.
- Hf is blue.
- Ta is blue.
- W is blue.
- Re is blue.
- Os is blue.
- Ir is blue.
- Pt is blue.
- Au is blue.
- Hg is blue.
- Tl is pink.
- Pb is pink.
- Bi is pink.
- Po is pink.
- At is yellow.
- Rn is red.
- Fr is red.
- Ra is orange.
- Rf is blue.
- Db is blue.
- Sg is blue.
- Bh is blue.
- Hs is blue.
- Mt is blue.
- Ds is blue.
- Rg is blue.
- Cn is blue.
- Nh is light green.
- Fl is light green.
- Mc is light green.
- Lv is light green.
- Ts is light green.
- Og is light green.
- La is cyan.
- Ce is cyan.
- Pr is cyan.
- Nd is cyan.
- Pm is cyan.
- Sm is cyan.
- Eu is cyan.
- Gd is cyan.
- Tb is cyan.
- Dy is cyan.
- Ho is cyan.
- Er is cyan.
- Tm is cyan.
- Yb is cyan.
- Lu is cyan.
- Ac is teal.
- Th is teal.
- Pa is teal.
- U is teal.
- Np is teal.
- Pu is teal.
- Am is teal.
- Cm is teal.
- Bk is teal.
- Cf is teal.
- Es is teal.
- Fm is teal.
- Md is teal.
- No is teal.
- Lr is teal.

AtomGPT

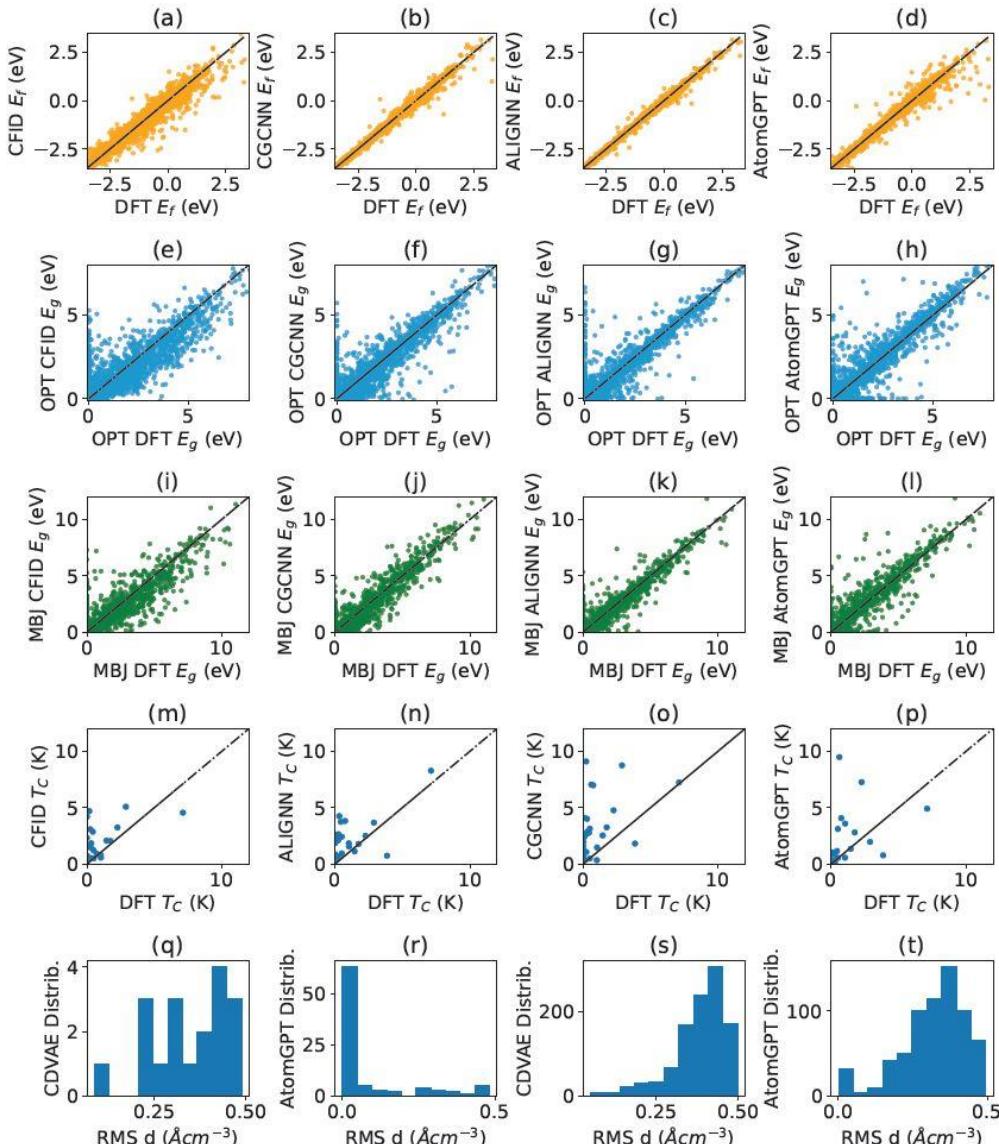


ChemNLP describer input:
MgB₂ is crystallizes in the hexagonal P6/mmm space group. There is one shorter (2.5 Å) and one longer (3.96 Å) B-Mg bond lengths. Mg is bonded in edge-sharing MgB₁₂ Cuboctahedral...
Output: 32.685

Alpaca input:
Instruction: Below is a description of a superconductor material.,
Input: The chemical formula is MgB₂. The T_c_supercon is 32.685. Generate atomic structure description with lattice lengths, angles, coordinates and atom types.,
Output: 3.07 3.07 3.51\n90 90 120\nMg 0.000 0.000 0.000\nB 0.667 0.333 0.500\nB 0.333 0.667 0.500}



$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$



	Forward models			
Prop/MAE	CFID	CGCNN	ALIGNNN	AtomGPT
E_{form} (eV/atom)	0.142	0.063	0.033	0.072
OPT E_g (eV)	0.299	0.199	0.142	0.139
MBJ E_g (eV)	0.531	0.407	0.310	0.319
T_c (K)	1.99	2.60	2.03	1.54
	Inverse models			
Database/RMSE	CDVAE	AtomGPT		
SuperConDB	0.24	0.08		
Carbon24	0.36	0.32		

AtomGPT: Atomistic Generative Pretrained Transformer for Forward and Inverse Materials Design

Kamal Choudhary*

Cite this: *J. Phys. Chem. Lett.* 2024, 15, XXX, 6909–6917

Publication Date: June 27, 2024

<https://doi.org/10.1021/acs.jpclett.4c01126>

Not subject to U.S. Copyright. Published 2024 by American Chemical Society

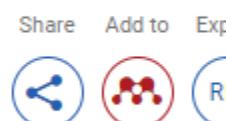
[Request reuse permissions](#) [Subscribed](#)

Article Views

Altmetric

Citations

LEARN ABOUT THESE METRICS



Generated structures relaxed with ALIGNNN-FF before DFT-superconductivity workflow

AtomGPT Architecture

arXiv > cs > arXiv:1706.03762

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

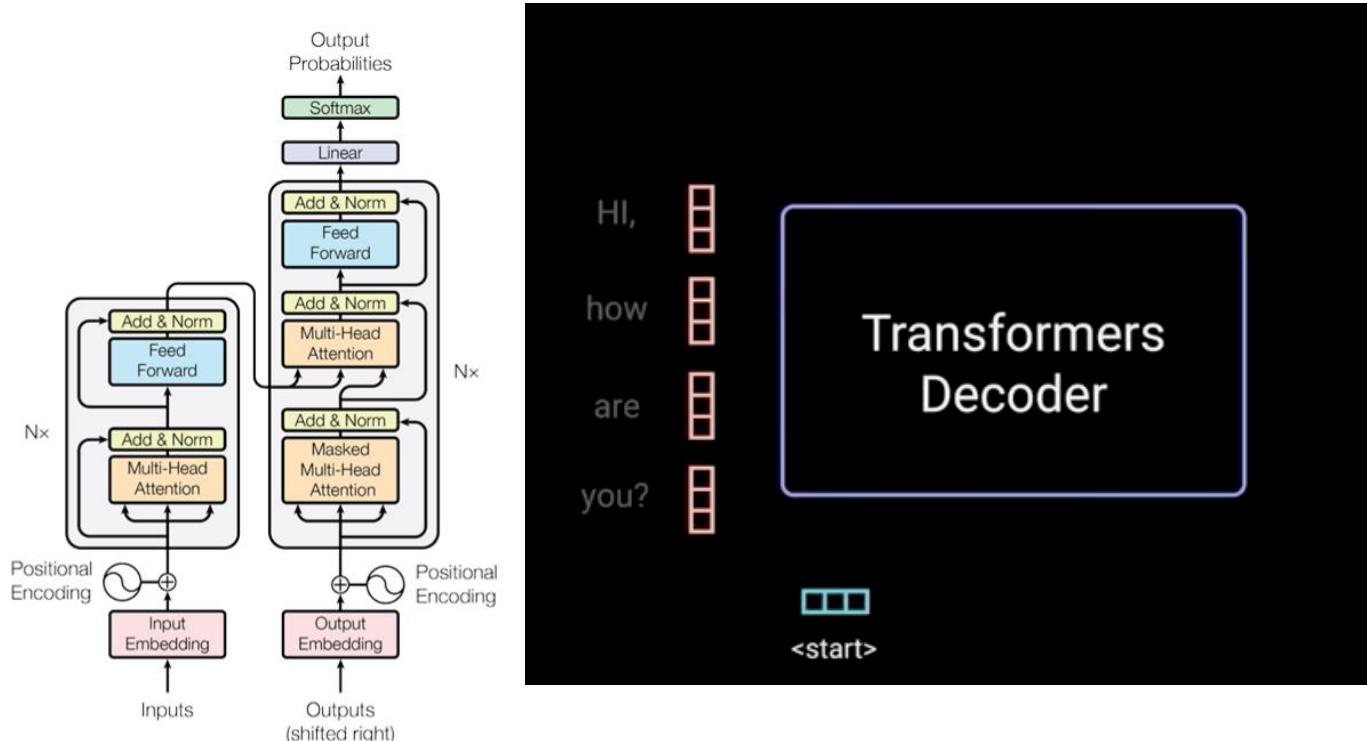


Figure 1: The Transformer - model architecture.

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Much better than RNN, LSTM etc.
Attention:~ extremely long-term memory

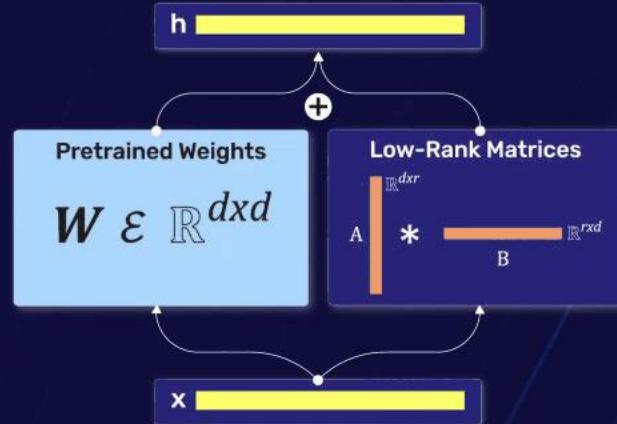
- Uses HuggingFace ecosystem <https://huggingface.co/>
- Uses Mistral AI-7B
- Modified language model head for forward models
- Low-rank adaptation (LoRA) for parameter-efficient fine-tuning (PEFT)
- Rotary position embedding (RoPE)
- Transformer reinforcement learning (TRL)

A screenshot of the Hugging Face platform showing the model card for 'atomgpt_mistral_tc_supercon'. The card includes the model name, a preview of generated text ('HI, how are you?'), and various metadata such as 'Text Generation', 'PEFT', 'Safetensors', 'English', 'chemistry', and 'text-generation'. It also shows 2 likes and links to 'Model card', 'Files and versions', 'Community', and 'Settings'.

Fine-tune models for atomic structure generation of superconductor

AtomGPT Architecture

Fine-Tuning LLMs using PEFT



- **Parameter-Efficient Fine-Tuning (PEFT)**: adjusting only a subset of parameters while keeping the majority of the model frozen
- **Low-Rank Adaptation (LoRA)**: A specific form of PEFT that inserts low-rank matrices into the model's architecture. These matrices capture task-specific knowledge and are fine-tuned while keeping the majority of the model's original parameters fixed.
- **Automatic Mixed Precision (AMP)**: Trains LLMs with mixed-precision data types for faster computations without loss of accuracy.
- **Gradient Accumulation**: Accumulates gradients over multiple batches to reduce memory usage.
- **Transformer Reinforcement Learning (TRL)**: transformer architectures based agents for Reinforcement Learning (RL) instead of simple NNs
- **Rotary Position Embedding (RoPE)**: encodes the absolute position with a rotation matrix and meanwhile incorporates the explicit relative position dependency in self-attention formulation

arXiv > cs > arXiv:2104.09864

Computer Science > Computation and Language

[Submitted on 20 Apr 2021 (v1), last revised 8 Nov 2023 (this version, v5)]

RoFormer: Enhanced Transformer with Rotary Position Embedding

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, Yunfeng Liu

Future work:

Integrate AtomGPT with
Retrieval-Augmented Generation (RAG)

<https://learnopencv.com/fine-tuning-langs-using-peft/>

Try AtomGPT on Google Colab Today!



Google colab/Jupyter notebook

Notebooks	Google Colab	Descriptions
<u>Forward Model training</u>	 Open in Colab	Example of forward model training for exfoliation energy.
<u>Inverse Model training</u>	 Open in Colab	Example of installing AtomGPT, inverse model training for 5 sample materials, using the trained model for inference, relaxing structures with ALIGNN-FF, generating a database of atomic structures.
<u>HuggingFace model inference</u>	 Open in Colab	AtomGPT Structure Generation/Inference example with a model hosted on Huggingface.

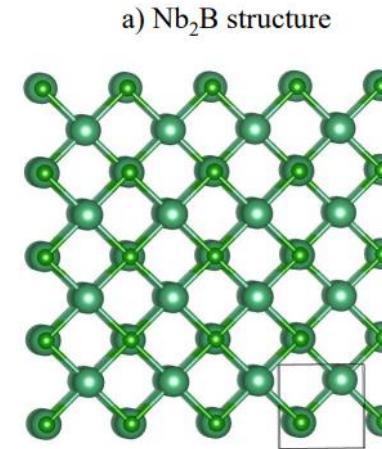
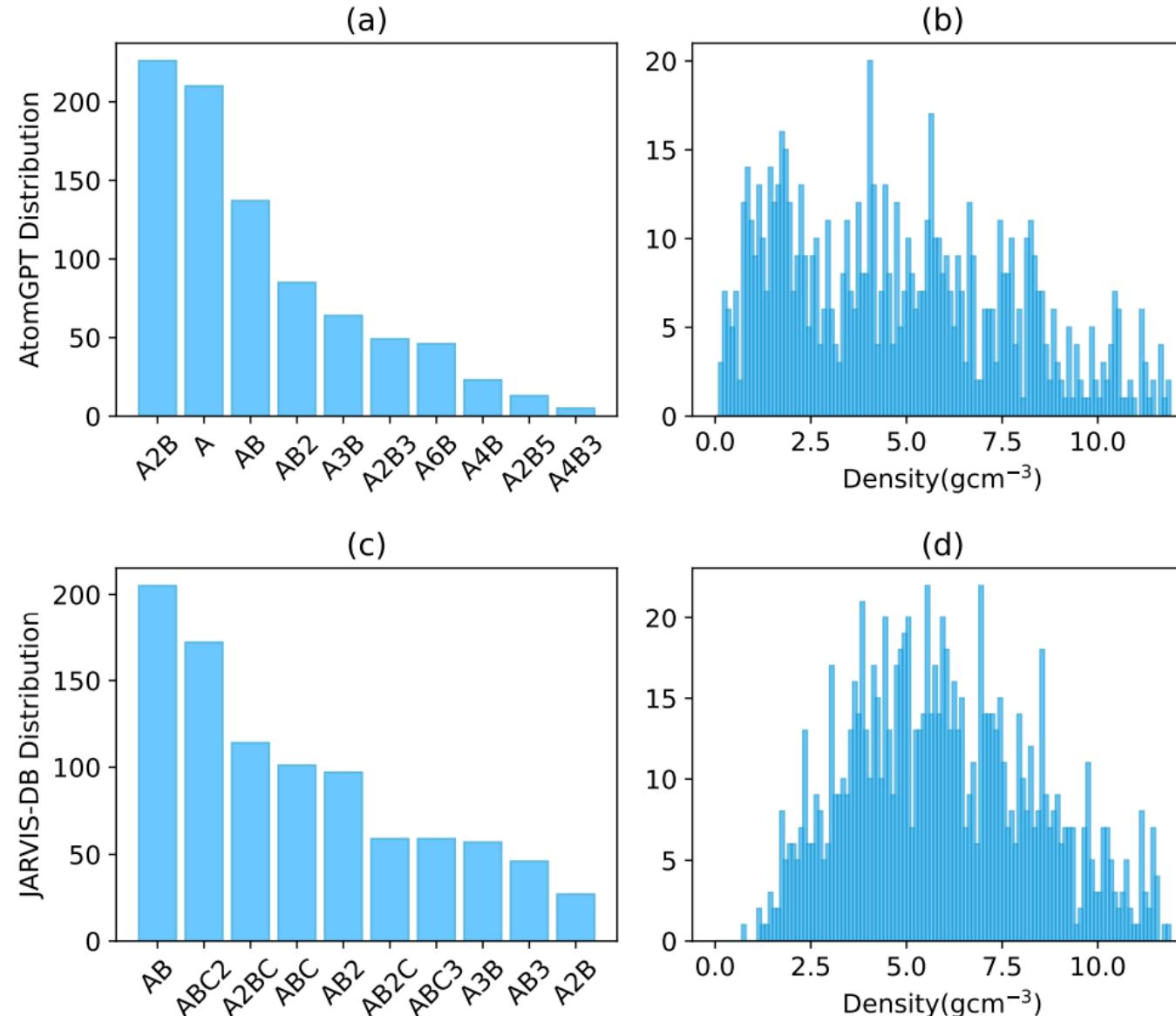
For similar other notebook examples, see [JARVIS-Tools-Notebook Collection](#)

HuggingFace link 😊

<https://github.com/usnistgov/atomgpt>

<https://huggingface.co/knc6>

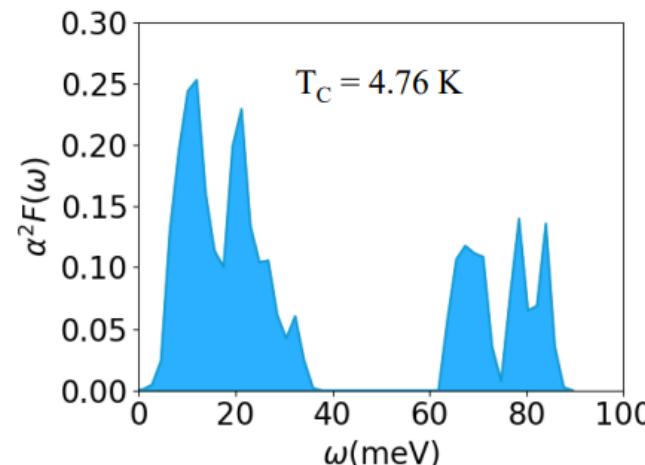
AtomGPT



$$T_c = \frac{\omega_{log}}{1.2} \exp \left[-\frac{1.04(1+\lambda)}{\lambda - \mu^*(1+0.62\lambda)} \right]$$

$$\omega_{log} = \exp \left[\frac{\int d\omega \frac{\alpha^2 F(\omega)}{\omega} \ln \omega}{\int d\omega \frac{\alpha^2 F(\omega)}{\omega}} \right]$$

b) Eliashberg function



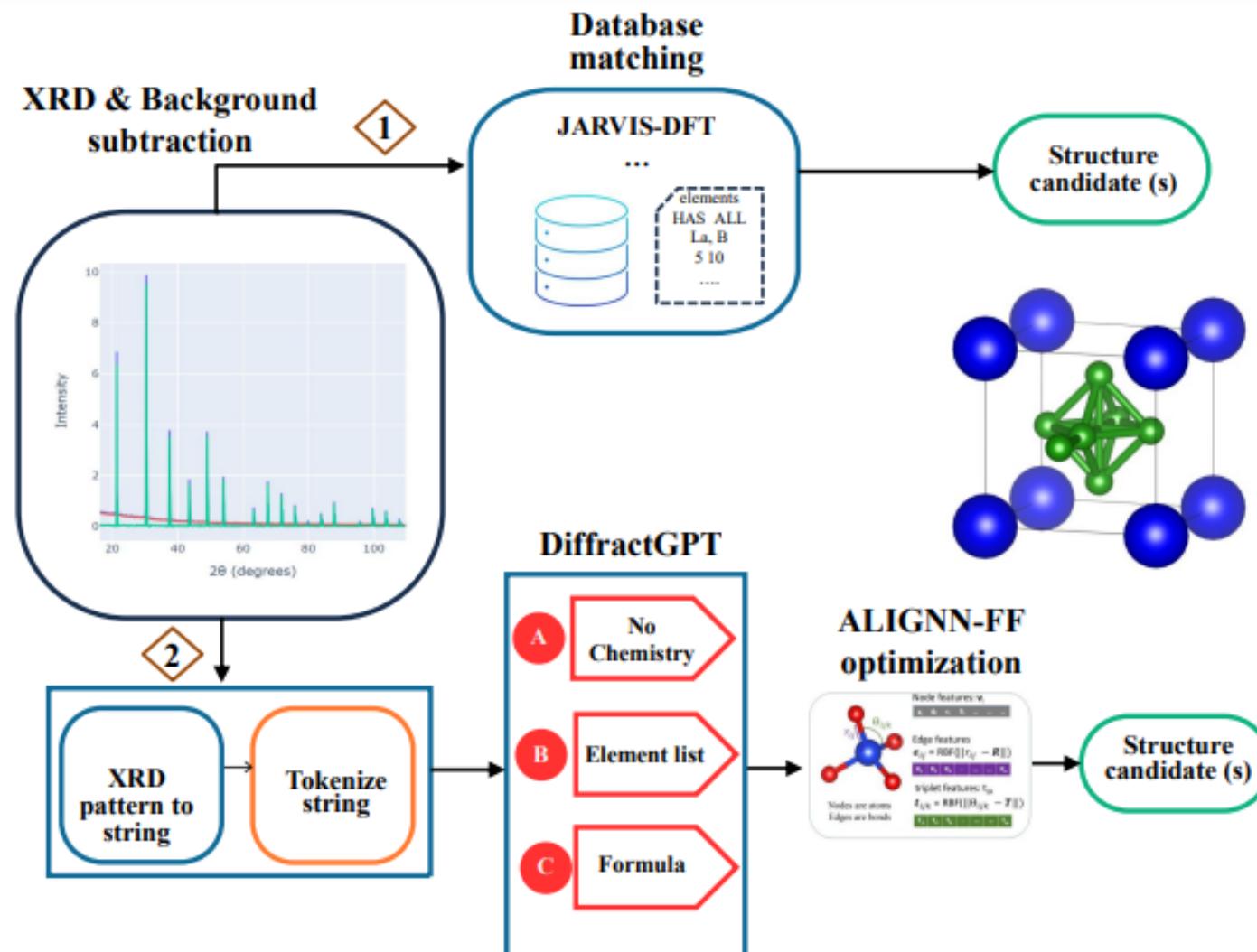


Table 1: Performance measurement in terms of mean absolute error (MAE) for predicting lattice constants (\AA) using gradient boosting regression (GBR), convolutional neural network (CNN), and varieties of DiffractGPT (DGPT) models. We also compare root mean square distance in predicted vs target structures using DGPT models.

Prop/MAE	GBR	CNN	DGPT-no formula	DGPT-element list	DGPT-formula
a	1.03	0.28	0.25	0.18	0.17
b	0.99	0.27	0.26	0.20	0.18
c	1.27	0.28	0.38	0.28	0.27
RMS-d	-	-	0.23	0.21	0.07

Reproducibility & Benchmarking



Challenges in materials science community:

- Reproducibility
 - Transparency
 - Validation
 - Fidelity
 - Data vs. metadata
 - What is the ground truth/reference?
- Synergy of computational and experimental databases

1,500 scientists lift the lid on reproducibility

[Monya Baker](#)

[Nature](#) 533, 452–454 (2016) | [Cite this article](#)

171k Accesses | 1952 Citations | 5176 Altmetric | [Metrics](#)

https://pages.nist.gov/jarvis_leaderboard/

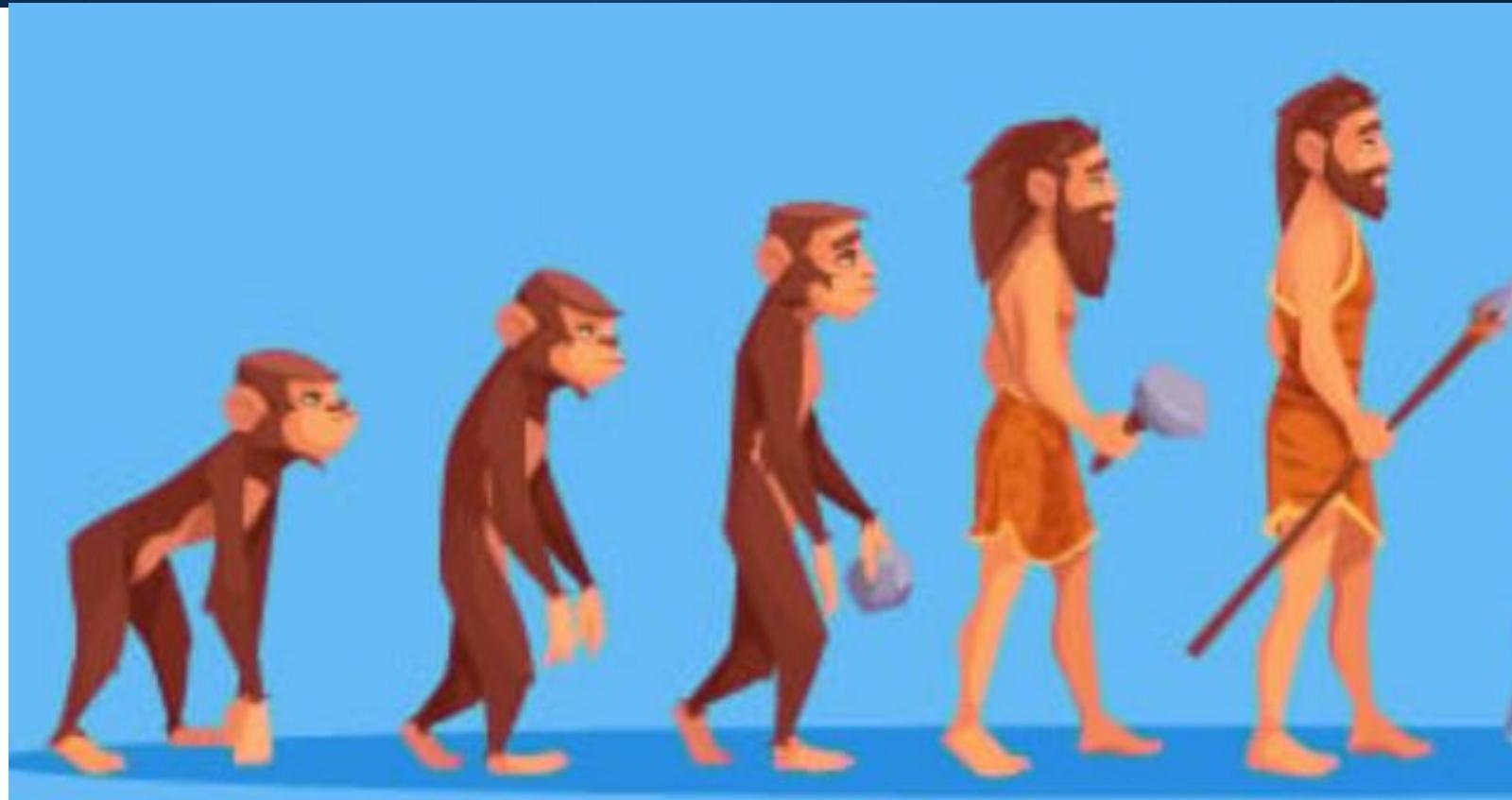
The screenshot shows the JARVIS-Leaderboard homepage. At the top, there's a navigation bar with the NIST logo, a search bar, and a version info box (v2024.1.26). The main heading is "Explore State-of-the-Art Materials Design Methods". Below this, there are four rows of cards representing different methods:

Artificial intelligence (AI)	Electronic Struct. (ES)	Force-field (FF)/potentials	Quantum Comput. (QC)
Contributions: 607 See All Benchmarks	Contributions: 786 See All Benchmarks	Contributions 282 See All Benchmarks	Contributions: 6 See All Benchmarks
Experiments (EXP)	Example Notebooks	Methodologies	Contribution Guide
Contributions: 25 See All Benchmarks	Notebooks: 16 See All Notebooks	Available Methods: 385 Learn more	Contributors: 21 Learn more

Goals

1. Flexibility to add new benchmarks
2. Experimental round-robin studies
3. Multi-modality: benchmarks beyond AI/ML/atomistic property
4. Easy to use/ get started
5. worked out examples (Jupyter/Colab notebooks)

Benchmark evolution in MSE



<1900s
Illustrative example

1900's
UCI, MNIST & other repos

2000's
Multi-modal data

2010's
Big data

Today
Designed benchmarks

Categories of benchmarks

JARVIS-Leaderboard: a large scale benchmark of materials design methods

Kamal Choudhary , Daniel Wines, Kangming Li, Kevin F. Garrity, Vishu Gupta, Aldo H. Romero, Jaron T. Krogel, Kayahan Saritas, Addis Fuhr, Panchapakesan Ganesh, Paul R. C. Kent, Keqiang Yan, Yuchao Lin, Shuiwang Ji, Ben Blaiszik, Patrick Reiser, Pascal Friederich, Ankit Agrawal, Pratyush Tiwary, Eric Beyerle, Peter Minch, Trevor David Rhone, Ichiro Takeuchi, Robert B. Wexler, Arun Mannodi-Kanakkithodi, Elif Ertekin, Avanish Mishra, Nithin Mathew, Mitchell Wood, Andrew Dale Rohskopf, Jason Hattrick-Simpers, Shih-Han Wang, Luke E. K. Achenie, Hongliang Xin, Maureen Williams, Adam J. Baciotti & Francesca Tavazza  Show fewer authors

npj Computational Materials 10, Article number: 93 (2024) | [Cite this article](#)

Category	General name	Method Specification
ES	DFT ⁶⁴	VASP ^{52,53} (PBE ⁵⁴ , LDA ⁶⁴ , OptB88vdW ⁶⁵ , Opt86BvdW ⁶⁶ , TBmBJ ^{67,68} , SCAN ⁶⁹ , r2SCAN ⁷⁰ , HSE06 ⁷¹) QE ⁷² (PBE ⁵⁴ , PBEsol ⁷³) ABINIT ^{74–76} (PBE ⁵⁴) GPAW ⁷⁷ (PBE ⁵⁴ , LDA ⁶⁴ , GLLB-sc ⁷⁸)
	QMC ⁶¹	QMCPACK ⁷⁹ (DMC ⁶¹)
	GW ⁶³	VASP ^{52,53} (G ₀ W ₀ ⁶³ , GW ₀ ⁶³)
	TB ⁵⁷	ThreeBodyTB.jl ⁸⁹ (Wannier90 ⁸⁰)
AI	Descriptor	CFID ⁸¹ , MagPie ⁸² , MatMiner ^{55,83} , crystal feature model ⁸⁴ , ElemNet ^{85–87} , IRNet ^{88–90} , BRNet ^{91,92} , SNAP ⁹³
	Graph-based	ALIGNN ⁹⁴ , CGCNN ⁹⁵ , SchNet ⁹⁶ , AtomVision ⁹⁷ , ChemNLP ⁹⁸ , DimeNet ^{99,100} , CHGNet ¹⁰¹ , M3GNET ¹⁰² , kgcnn_coGN ¹⁰³ , Potnet ¹⁰⁴ , Matformer ¹⁰⁵
	Transformers	OPT ¹⁰⁶ , GPT ¹⁵ , T5 ¹⁰⁷
FF	LJ ¹⁰⁸	LAMMPS ¹⁰⁹ (2D-Liquid)
	EAM ¹¹⁰	LAMMPS ¹⁰⁹ (FCC-Al)
	REBO ¹¹¹	LAMMPS ¹⁰⁹ (Diamond-Si)
	AMBER99sb-ildn ¹¹²	GROMACS ¹¹³ (Alanine dipeptide)
QC	CHARMM36m ¹¹⁴	GROMACS ¹¹³ (α -aminoisobutyric acid)
	Algorithms	Qiskit ¹¹⁵ (VQE ¹¹⁶ , VQD ¹¹⁷) PennyLane ^{118,119} (VQE ¹¹⁶ , VQD ¹¹⁷)
	Circuits	Qiskit ¹¹⁵ (PauliTwo Design ¹¹⁵ , SU(2) ¹¹⁵)
EXP	Diffraction	XRD (Bruker D8)
	Manometry	CO ₂ adsorption FACT lab ¹²⁰
	Vibroscopy	Kevlar FAVIMAT ¹²¹
	Magnetometry	Susceptibility (PPMS) ¹²¹

- 1) Electronic Structure
- 2) Artificial Intelligence
- 3) Force Field
- 4) Quantum Computation
- 5) Experiment

Types of Data:

- **Atomic structure** (Molecule, Crystal)
- **Material Property** (Bandgap, bulk modulus)
- **Images** (Microscopy: SEM, TEM, STM)
- **Spectra** (Diffraction: X-ray, Neutron, PL)
- **Text** (Research articles, notebooks, blogs)
- **Eigensolver** (Quantum Computation algorithms)

Categories of benchmarks

Summary table

Category/Sub-cat.	SinglePropertyPrediction	SinglePropertyClass	MLFF	TextClass	TokenClass	TextSummary	TextGen	AtomGen	ImageClass	Spectra	EigenSolver
AI	418	21	119	18	1	1	3	3	2	1	-
ES	776	-	-	-	-	-	-	-	-	10	-
FF	50	-	-	-	-	-	-	-	-	-	-
QC	-	-	-	-	-	-	-	-	-	-	6
EXP	7	-	-	-	-	-	-	-	-	18	-

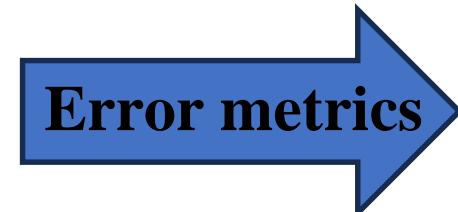
- Number of contributors: 20
- Number of methods: 159
- Number of benchmarks: 296
- Number of contributions: 1453
- Number of datapoints: 8731126

- Currently:
- ~100 Colab tutorial notebooks
 - 296 benchmarks (reference)
 - 1705 contributions
 - ~9 million datapoints

Contributions & Benchmarks

Contributions

- 1) Electronic Structure
- 2) Artificial Intelligence
- 3) Force Field
- 4) Quantum Computation
- 5) Experiment

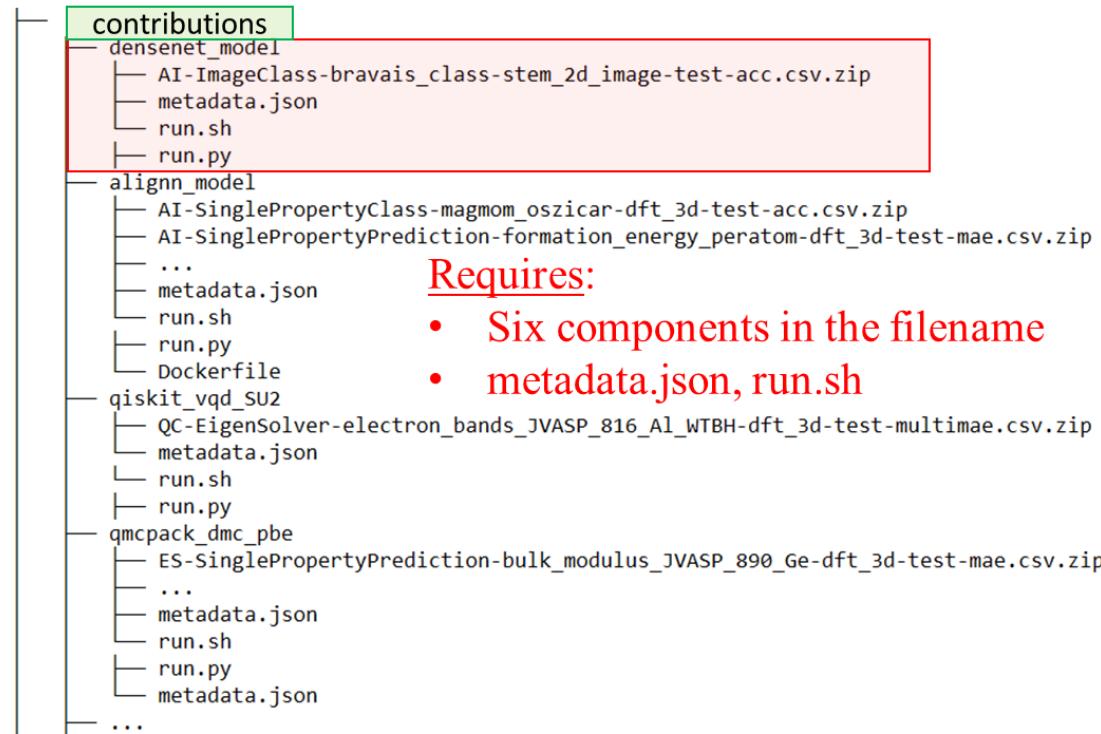


Benchmarks (reference point)

- 1) Experiment(s)
- 2) Test dataset
- 3) Electronic Structure
- 4) Analytical results
- 5) Other Experiments

*Benchmarks must be well-defined with an associated DOI

Contributions & Benchmarks



Requires:

- Six components in the filename
- metadata.json, run.sh

Benchmarks (json.zip)

```
{"train":
```

```
    {"JVASP-57695": 1.55,
     "JVASP-39": 1.55,
     "JVASP-7648": 0.38,
```

```
    ...}
```

Contributions (csv.zip)

id,prediction

JVASP-1408,0.54107
JVASP-1195,0.78746
JVASP-20778,0.58677

....

```
"test":
```

```
    {"JVASP-1408": -0.13,
     "JVASP-1195": 1.55,
     "JVASP-20778": 0.1,
```

```
    ...}
```

```
}
```

Example: AI Formation energy per atom

Model for formation_energy_peratom

Home Guide Notebooks AI ES EXP FF QC

Search jarvis.leaderboard v2024.1.26 44 28

AI

- Model for avg_elec_mass
- Model for avg_hole_mass
- Model for bandgap
- Model for bulk_modulus_kv
- Model for dft_piezo_max_dielectric
- Model for dfpt_piezo_max_dij
- Model for ehull
- Model for encut
- Model for epsx
- Model for epsy
- Model for epsz
- Model for exfoliation_energy
- Model for formation_energy_peratom**
- Model for kpoint_length_unit
- Model for magmom_oscicar
- Model for max_efg
- Model for mbj_bandgap
- Model for mepsx
- Model for mepsy
- Model for mepsz
- Model for n-Seebeck
- Model for n-powerfact
- Model for optb88vdw_bandgap
- Model for optb88vdw_total_energy
- Model for heat capacity
- Model for shear_modulus_gv
- Model for slme
- Model for spillage
- Model for halide_perovskite
- HSE decomposition energy

Description: This is a benchmark to evaluate how accurately an AI model can predict the formation energy per atom using the JARVIS-DFT (dft_3d) dataset. The dataset contains different types of chemical formula and atomic structures. Here we use mean absolute error (MAE) to compare models with respect to DFT accuracy.

AI-SinglePropertyPrediction-formation_energy_peratom-dft_3d-test-mae

MAE (formation_energy_peratom)

kgcnn_coGN, kgcnn_coNGN, potnet, matformer_256, alignnn_model, kgcnn_schnet, kgcnn_megnet, kgcnn_dimenetpp, cgcnn_cgcn, cgcnn_model, matminer_xgboost, matminer_rf, matminer_lgbm, crfd, crfd_ch6

Reference(s): <https://www.nature.com/articles/s41524-023-01012-9>; <https://hackingmaterials.lbl.gov/matminer>, <https://hackingmaterials.lbl.gov/matminer/>, <https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.120.145301>, <https://www.nature.com/articles/s41524-021-00650-1>, <https://github.com/divelab/AIRS/tree/main/OpenMat/Matformer>, <https://doi.org/10.1103/PhysRevMaterials.2.083801>, <https://doi.org/10.48550/arXiv.2305.11842>, <https://www.nature.com/articles/s41524-020-00440-1>, https://github.com/aimat-lab/gcnn_keras, <https://github.com/divelab/AIRS/tree/main/OpenMat/PotNet>

Model benchmarks

Model name	Dataset	MAE	Team name	Dataset size	Date submitted	Notes
kgcnn_coGN	dft_3d	0.0271	kgcnn	55713	05-06-2023	CSV, JSON, run.sh, Info
kgcnn_coNGN	dft_3d	0.0291	kgcnn	55713	05-06-2023	CSV, JSON, run.sh, Info
potnet	dft_3d	0.0293	DIVE@TAMU	55713	06-02-2023	CSV, JSON, run.sh, Info

A

A	B	C	D	E	F	G	H	I	J
1	id	prediction							
2	JVASP-386	-0.06485							
3	JVASP-697	-0.53265							
4	JVASP-120	-0.07311							
5	JVASP-833	-1.86553							
6	JVASP-229	-4.28398							
7	JVASP-120	-0.9024							
8	JVASP-120	-1.56657							
9	JVASP-181	-0.41461							

dft_3d_formation_energy_peratom.json - Notepad

```
File Edit Format View Help
[{"train": {""JVASP-21450": -0.50496, "JVASP-66320": 0.417, "JVASP-42000": -0.38536, "JVASP-115193": -0.74961, "JVASI02245, "JVASP-18480": -2.9647, "JVASP-107218": -0.66397, "JVASP-123687": -0.31591, "JVASP-82029": -0.31863, "JVASPS-52188": -2.30026, "JVASP-75687": 0.4939, "JVASP-41702":
```

```
#!/bin/bash
pip install alignnn
python run.py
```

```
{
  "model_name": "ALIGNNN",
  "project_url": "https://www.nature.com/articles/s41524-021-00650-1",
  "date_submitted": "01-14-2023",
  "author_email": "knc6@nist.gov",
  "database_version": "12-12-2022",
  "team_name": "ALIGNNN",
```

Day 4 Hands on

AI 1

AI 28

AI 29

AI 30

Day 4 Hands on

AI 7 JARVIS_Leaderboard_MatMiner

AI 9 JARVIS_ALIGNNN_Basic_Training_example

AI 12 JARVIS_Leaderboard_MLFF/ALIGNNN-FF for Silicon

Melt Quench

Our team and Collaboration



K. Choudhary
(NIST)

D. Wines
(NIST)

F. Tavazza
(NIST)

B. DeCost
(NIST)

K. F. Garrity
(NIST)

A. J. Biacchi
(NIST)

Conclusion and Contact Information

1. JARVIS Overview

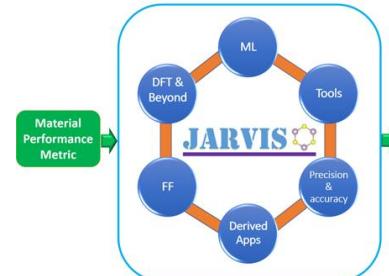
2. JARVIS-DFT

3. ALIGNN

4. AtomGPT

5. Leaderboard

- NIST-JARVIS is a comprehensive database and toolset for materials science research
- A one-stop resource for materials design and discovery
- It is not just a tool—it's a collaborative platform
- Thank you for your time. If you're interested in learning more or collaborating, please feel free to reach out!



<https://jarvis.nist.gov>

Email: kamal.choudhary@nist.gov

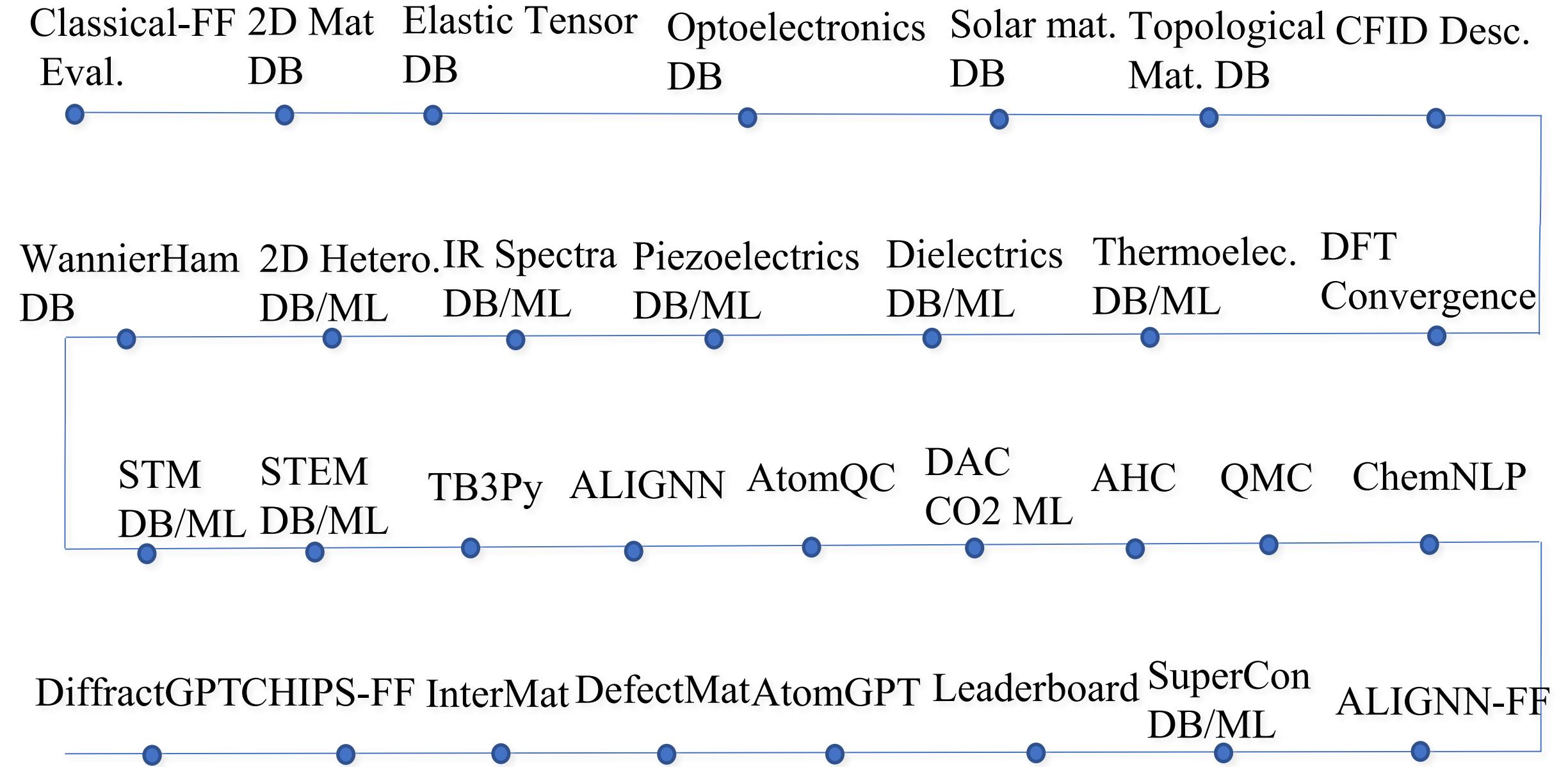


@knc6



@dr_k_choudhary





Hands-on session



10:05 AM to 12:00 noon

1. Go to website: <https://jarvis.nist.gov/>
2. Make an account, Go to one of the databases/interactive apps
3. On JARVIS website, click on Tutorials >> JARVIS-Tools-Notebooks link [10 min]
4. Running a simple DFT calculations with QE/GPAW [20 min]
5. Analyzing data in JARVIS-DFT [10 min]
6. Training an ALIGNN model [20 min]
7. Training an ALIGNN-FF model [20 min]
8. Using Pre-trained ALIGNN/ALIGNN-FF models [10 min]
9. Train/Infer AtomGPT models [20 min]
10. Run an FF calculation with LAMMPS (optional walk through) [10 min]
11. Train AtomVision model with STEM/STM image data (optional)
12. Develop AomQC circuit/models (optional)