Skolkovo Institute of Science and Technology

MACHINE LEARNING FOR ELASTIC STRAIN ENGINEERING

Doctoral Thesis

by

EVGENII TSYMBALOV

DOCTORAL PROGRAM IN COMPUTATIONAL AND DATA SCIENCE AND ENGINEERING

Supervisor
Professor Alexander Shapeev

Moscow - 2020

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Evgenii Tsymbalov

Prof. Alexander Shapeev

Abstract

Nanostructured materials can withstand large deformation without failure and may offer remarkable properties in the strained state: electronic, optical, thermal, and chemical features of a strained crystal may vary dramatically from the unstrained counterparts. Operating within the limits of admissible elastic strains means that these properties may be changed on demand, paving the way to the elastic strain engineering (ESE). The properties of interest are studied via the first-principles calculations, yet a large amount of computational resources is needed if one wants to explore the 6D strain space or perform an advanced optimization of any target figure of merit.

This problem is addressed by the deep elastic strain engineering, which employs the surrogate machine learning (ML) model based on the *ab initio* simulations data. This work proposes a neural-network-based machine learning framework, which is specifically tailored to the needs of ESE and exploits the interaction between the model and simulations. More specifically, we take advantage of the different sources of data provided by first-principles calculations and offer an active learning machinery for the further training of the model. The general problem of the uncertainty quantification in neural networks is also studied in detail; this work proposes a family of approaches based on the Monte-Carlo dropout uncertainty estimation. The performance of the proposed approaches is extensively studied in a number of numerical experiments.

We explore the vast 6D space of admissible strains and offer a number of discoveries made possible with the use of the ML model. These include the exploratory analysis of the six-dimensional strain space, existence of the direct bandgap, and metal transitions scenarios for the silicon and diamond crystals, as well as a comprehensive description of the topological features of the electronic band structure and FEM-assisted imitation of *in situ* experiments.

We believe that the developed approaches can offer the framework for the next-generation exploration in the field of ESE for a variety of applications in microelectronics, optoelectronics, photonics, and energy technologies.

Publications

Main author

Tsymbalov, E., Panov, M., and Shapeev, A. (2018). Dropout-based Active Learning for Regression. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 247–258. Springer

Shi, Z., Tsymbalov, E., Dao, M., Suresh, S., Shapeev, A., and Li, J. (2019). Deep Elastic Strain Engineering of Bandgap through Machine Learning. *Proceedings of the National Academy of Sciences*, 116(10):4117–4122.

Tsymbalov, E., Makarychev, S., Shapeev, A., and Panov, M. (2019). Deeper Connections between Neural Networks and Gaussian Processes Speed-up Active Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence. Main track*, pages 3599-3605. Macao, China, 2019

Tsymbalov, E., Fedyanin, K., and Panov, M. (2020). Dropout Strikes Back: Improved Uncertainty Estimation via Diversity Sampled Implicit Ensembles. arXiv preprint arXiv:2003.03274. *In preparation*.

Tsymbalov, E., Shi, Z., Dao, M., Suresh, S., Li, J., and Shapeev, A. (2020). Machine Learning for Deep Elastic Strain Engineering of Electronic Properties of Materials. *In preparation*.

Coauthor

Shi, Z., Dao, M., Tsymbalov, E., Shapeev, A., Li, J., and Suresh, S. (2020). Metallization of diamond. *Proceedings of the National Academy of Sciences*, 117(40), 24634-24639.

Shapeev A., Gubaev K., Tsymbalov E., Podryabinkin E. (2020) *Active Learning and Uncertainty Estimation*. In: Schütt K., Chmiela S., von Lilienfeld O., Tkatchenko A., Tsuda K., Müller KR. (eds) *Machine Learning Meets Quantum Physics*. Lecture Notes in Physics, vol 968. Springer, Cham

Patents

Dao, Ming, and Li, Ju, and Shi, Zhe, and Tsymbalov, Evgenii, and Shapeev, Alexander, and Suresh, Subra (2018). *ELASTIC STRAIN ENGINEERING OF MATERIALS*, WO2020076181.

Contents

List of Figures

List of Tables

List of abbreviations

AL	Active Learning
AUC	Area Under Curve
BZ	Brillouin Zone
C	Carbon
CB, CBM	Conduction Band, Conduction Band Minima
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DOB	Density Of Bandgap
DFT	Density Functional Theory
DPP	Determinantal Point Processes
EM	Effective Mass
ESE	Elastic Strain Engineering
ESED	Elastic Strain Energy Density
eV	electron-Volts
FCC	Face-Centered Cubic
FEM	Finite Element Method
FoM	Figure-of-Merit
GBR	Gradient Boosting Regression
GP	Gaussian Process
GPU	Graphics Processing Unit
GGA	Generalized Gradient Approximation
GW	Green's function and the screened Coulomb interaction W
LDA	Local Density Approximation
LL	Log-Likelihood
MAE	Mean Absolute Error
MC	Monte-Carlo
MCDUE	Monte-Carlo Dropout Uncertainty Estimation
ML	Machine Learning
NN	Neural Network
PAW	Projector Augmented Wave (method)
PBE	Perdew–Burke–Ernzerhof (functional)
RFR	Random Forest Regression
RMSE	Root Mean Squared Error
Si	Silicon
tSNE	t-Stochastic Neighbourhood Embedding
UCI	University of California, Irvine (dataset)
UE	Uncertainty Estimation
VASP	Vienna Ab Initio Package
VB, VBM	Valence Band, Valence Band Maxima
XC	eXchange–Correlation

Acknowledgements

I want to thank many people that made this thesis possible.

First of all, I would like to thank my academic advisor Alexander Shapeev for teaching me many things related to the research in general. Alex, I owe you a bit more than everything for making my doctoral studies in Skoltech as smooth as possible. I am grateful to my collaborators from MIT and NTU: Ju Li, Ming Dao, and Subra Suresh, for the fruitful work on the papers. I would like to thank Zhe (Frank) Shi for a long and close collaboration on the exciting topic of ESE.

I want to thank my collaborators from Skoltech (Maxim Panov group): Nikita Kotelevskiy, Nikita Mokrov, Kirill Fedyanin, Maxim Panov for making it possible to have another research direction in parallel to ESE. I want to thank Victor Vysotsky for his gentle support in HPC calculations. Apart from the human creatures, I would like to thank the following machines: clusters Pardus and Beowulf, Magnus cluster, and node23 in particular, Google Colab, and anonymous Russian IT startup for offering extra resources in need.

I would like to thank my research group colleagues: Evgenii Podryabinkin, Ivan Novikov, Max Hodapp, Konstantin Gubaev, Tatiana Kostiuchenko, Edgar Makarov, and Vadim Sotskov for the active participation in a long PhD journey. I am grateful to the Skoltech stuff: Polina Rumyantseva, Valentina Kostornichenko, Nadezhda Dontsu, Elena Ditte, and many others, for supporting me in times of administrative troubles. I want to thank the reviewers: Nikolai Brilliantov, Bohayra Mortazavi, Sergey Levchenko, Justin Smith, and Alexey Zaytsev for taking the time to review my thesis and for invaluable remarks and suggestions.

Finally, I would like to thank my family: younger brother for teaching me what responsibility is, my mother for raising me with a free mind, and my wife for her love and support.

1 Introduction

1.1 Motivation

One of the main objectives of materials science is materials design. To create materials with desirable properties, humankind had experimented with alloys since centuries ago. If we consider this from the methodological point of view, it was a trial-and-error approach based on thousands of experiments. Moreover, the method of choice for obtaining materials with better properties was merely mixing them in different quantities – making alloys. Different rules of thumb emerged based on materials and years of experience.

With the rise of the understanding of nature, scientists have discovered the first principles of the materials' properties. The density functional theory (DFT) (Parr, 1980) has emerged as a universal tool to solve otherwise infeasible *ab initio* equations, providing us (to some extent) with a full description of what is happening on the deep level. DFT as a tool is widely used worldwide: according to several reports (Wright, 2016; Austin et al., 2018), from 20% to 40% of supercomputing time is spend on first-principles simulations, and most of them are DFT-based. However, modern algorithms and implementations are still not scalable to the extent that they may be used for very precise calculations with tens of thousands of atoms. From the physical perspective, there are many simpler approaches that neglect some intricate effects or describe the system of interest from the global perspective. Another popular solution to this problem is surrogate modeling: instead of making tens of thousands of typical calculations with DFT, one may create a simpler model that approximates the behaviour of DFT, usually trading some accuracy for the speed. This is where machine learning comes into play: modern algorithms, such as deep neural networks (LeCun et al., 2015), may help to improve performance in not only ML-popular fields, such as computer vision (Guo et al., 2016), or natural language processing (Cambria and White, 2014) but could help with the understanding of physical effects (Zdeborová, 2017; Carrasquilla and Melko, 2017), post-process the billions of on-site experiments and help in scientific discovery (Aad et al., 2012; Derkach et al., 2020), or, as it was stated above, speed-up the calculations by analyzing the gathered data and emerge as a powerful tool in the hands of explorers (Radovic et al., 2018).

For the last few decades, a microworld became a point of interest for physicists. The quest for better materials led to the invention of graphene (Novoselov et al., 2004) and the forthcoming burst in the research of 2D materials. The industry is interested in micromaterials because of the nice theoretical properties that are put on the table, which involve better efficiency. Talking about graphene as an example: it has some intriguing properties, like

increased chemical reactivity, or remarkable electron mobility. After that, a large number of "low-dimensional" materials (2D, 1D, or even 0D) have drawn significant interest from both scientists and industry, see (Ge et al., 2016; Akinwande et al., 2017; Momeni et al., 2020) for detailed reviews.

1.2 Goal of the work

On the wave of interest to low-dimensional materials, elastic strain engineering has emerged as a promising tool to explore materials properties. Nanostructured materials can withstand much higher tensile/shear elastic strains without mechanical relaxation or failure than their conventional counterparts, opening up a huge parameter space for rational engineering of materials properties by elastic strain (Li et al., 2014). The electronic, optical, thermal, and chemical properties of crystals are functions of the 6-dimensional elastic strain tensor $\boldsymbol{\varepsilon}$, which allows us to tune materials' properties on the fly by applying a corresponding strain, and this effect is reversible in a pretty large region of the strain space. However, the 6D space of parameters is large and easy to lost in: if one would like to explore the properties for a given crystal in a range from -5% to 5% with a 0.5% step, this would require more than 85 million first-principles calculations. As a standard DFT fails to predict most of the essential properties for semiconductors (such as a bandgap) with an error of more than 50% for a silicon crystal, one also needs to work with much more intricate procedures, such as GW correction (Hedin, 1965; Shishkin and Kresse, 2006), which involves taking into account more lower-level physical concepts and slows down the calculation time by several orders of magnitude.

This problem is addressed by the deep elastic strain engineering (Shi et al., 2019), which involves the construction of special surrogate models in order to mimic the heavy GW calculations. This includes tailored training, which allows learning more from fast, but inaccurate DFT calculations, and active learning, that allows to sample and calculate fewer data. The primary goal of this work is to design a suitable surrogate model for *ab initio* calculation that can help with the elastic strain engineering-related tasks.

1.3 Thesis structure

This thesis is organized as follows.

In Chapter 2, we will provide a brief yet comprehensive introduction to the physical side of the problem, namely, a crystal physics and the band structure. We will also discuss the topic of DFT calculations important for the band structure prediction. It is followed by a part dedicated to the machine learning machinery in general and its applications and challenges related to modern materials science. We will especially focus on approaches to the *active learning* (Settles, 2012), also known as adaptive experimental design (Fedorov, 1972), as the surrogate models, including the ones used in this work, may greatly benefit from it.

The next chapter, Chapter 3, will be wholly dedicated to the review of the elastic strain engineering as the main topic of this work. We will consider both theoretical and experimental findings in this area. This chapter will conclude the introductory part of the thesis, and the next part starts from the methodology description, given in Chapter 4. Here, questions of both machine-learning and *ab initio* calculations setup will be considered. Chapter 5 focuses on the model design as well as the selected approaches to the model training and active learning.

Chapter 6 covers a detailed discussion of the conducted numerical experiments. Specific aspects of uncertainty estimation and active learning for the fully-connected neural networks are extensively examined here.

Chapter 7 will represent the quintessence of the ESE-related experiments, providing the reader with the findings and discoveries obtained from the deep machine learning model in previous chapters.

This will bring us to the concluding part of the research – discussion of both ESE and active learning results, as well as the possible future prospects, which are provided in Chapter 8.

2 Background

2.1 Crystal structure and deformation

In this section, we will briefly discuss the ideal crystal from the physical and mathematical perspective and will define the necessary concepts for later use.

2.1.1 Crystal structure

A crystal is a solid material whose microscopic components (usually atoms or molecules) are arranged in highly ordered structures. Many minerals are naturally formed as crystals, and most of the solid materials, including metals, are usually arranged in a crystal form as well.

An **ideal crystal** consists of two components: a *basis* (or *motif*), which is a group of atoms or molecules ("repeating pattern"), and a *lattice*, which describes how exactly this group of particles repeats in space. Ideal crystals do not exist in nature since the pattern is repeated infinitely in all dimensions, yet surface conditions play a significant role in determining the properties of the material, see, e.g., Nie et al. (2019). Moreover, *grains*, or microscopic crystals, are usually formed inside of the real crystals (for example, alloys), and a thorough analysis should take grains into consideration. However, an ideal crystal may be used to describe the properties of the real crystals in most of the cases and is the only accessible way to perform accurate first-principles calculations for now.

The **lattice** is defined by three *translation vectors* a_1, a_2, a_3, which preserve the translation symmetry, i.e., the atomic arrangement is exactly the same for both points r and r', defined as:

$$r' = r + n_1 a_1 + n_2 a_2 + n_3 a_3 \tag{2.1}$$

for arbitrary integers n_1, n_2, n_3. The set of points $\{r'\}$ defines a lattice, which is also referred to as a *Bravais lattice*.

A **basis** of atoms is attached to every lattice point. The position r_j of the center of atom j (of n_{atoms} atoms) of the basis relative to the associated lattice point is

$$r_j = x_j a_1 + y_j a_2 + z_j a_3, \tag{2.2}$$

where $0 \leq x_j, y_j, z_j \leq 1$, $j = 1, \ldots, n_{atoms}$. The volume spanned by the basis vectors is also referred to as a **unit cell**. A **primitive cell** a unit cell with minimal volume, meaning there is no cell of smaller volume that can serve as a building block for the crystal structure. A primitive cell may be constructed

by applying Voronoi decomposition to a Bravais lattice; it is also referred to as Wigner-Seitz cell.

In this work, we are focused on silicon and diamond as materials of interest, and the crystal structure for both materials (as well as for some others, like germanium, tin, boron nitride) is diamond cubic, which denotes two atoms in a certain relative position, repeated according to FCC Bravais lattice, see Figure 2.1 for the visual representation. The latter means that $a_1 = a_2 = a_3 = a$, where a is a lattice constant, which can be defined experimentally yet should be checked for every *ab initio* method used. The primitive cell contains two atoms, shifted by 0.25 in lattice-related coordinates. Given that, we may define the primitive basis (associated with the primitive cell) $B = \{(x_j, y_j, z_j)\}, j = 1, 2$ as

$$\begin{aligned} (x_1, y_1, z_1) &= (0, 0, 0); \\ (x_2, y_2, z_2) &= (0.25, 0.25, 0.25). \end{aligned} \tag{2.3}$$

One possible set of the corresponding lattice vectors forms a **lattice matrix**

$$L = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}. \tag{2.4}$$

From the perspective of first-principle calculations, it is much more effective to consider the primitive cell; however, in some occasions, calculations may be double-checked by using a conventional unit cell that consists of 8 atoms; see Appendix A for details.

2.1.2 Reciprocal lattice and critical points

A Fourier transform takes us from the "real" space, where a Bravais lattice of an ideal crystal exists, to the *reciprocal* space, often called **k**-space. It transforms the crystal lattice into the *reciprocal lattice*, which is also a Bravais lattice, and Voronoi decomposition shatters the reciprocal lattice into the **Brillouin zones** (also referred to as the first Brillouin zone), which are the Wigner-Seitz (primitive) cells of the reciprocal lattice. These concepts are essential since the description of the properties of an ideal crystal can be completely characterized by taking a single Brillouin zone into consideration.

Points of **k**-space (referred to as **k**-points) are used to describe electronic properties within the ideal crystal. Several **k**-points of high symmetry are of special interest – these are called critical (sometimes symmetrical) points. For the FCC lattice, where the Brillouin zone is a truncated octahedron, these are listed below (see Figure 2.1 for the graphical representation):

- Γ: center of the Brillouin zone;
- K: middle of an edge joining two hexagonal faces;
- L: center of a hexagonal face;
- U: middle of an edge joining a hexagonal and a square face;

- X: center of a square face.

Among the lines that connect the critical **k**-points in the reciprocal space, we introduce the Δ-line that connects the Γ and X **k**-points. This notation will be used later, as one of the results of this work (namely, Section 7.2) is a complete **k**-space description as the function of strain. For the visualization of the electronic properties of an ideal crystal, a path that connects several **k**-points is used as referred to as a **k**-path.

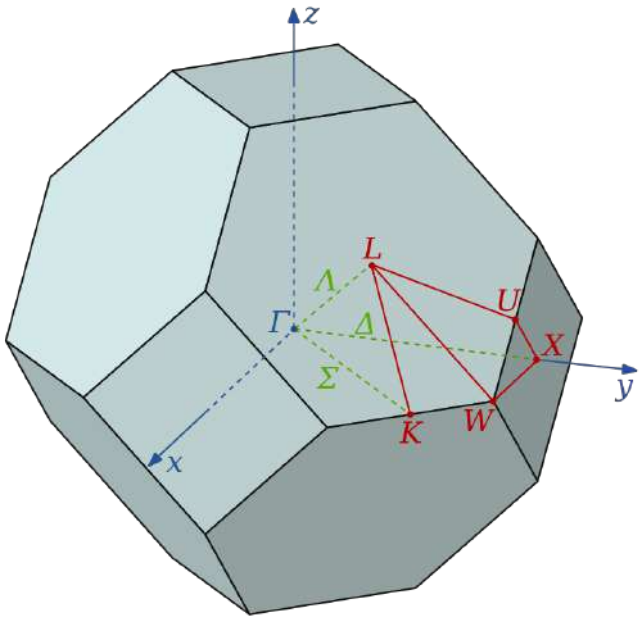


FIGURE 2.1: Brillouin zone for FCC lattice. This image is courtesy of inductiveload https://commons.wikimedia.org/wiki/User:Inductiveload user from wikipedia.org. Critical points Γ, L, and others are of special interest in this study.

2.1.3 Deformation tensor

The straining of a crystal lattice is described by applying the symmetric 3×3 tensor transformation to the primitive (or conventional unit) cell:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{12} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{13} & \varepsilon_{23} & \varepsilon_{33} \end{pmatrix}. \tag{2.5}$$

The components of elastic strain tensor $\boldsymbol{\varepsilon}$ are often referred to as

$$\varepsilon_1 = \varepsilon_{11} = \varepsilon_{xx},\ \varepsilon_2 = \varepsilon_{22} = \varepsilon_{yy},\ \varepsilon_3 = \varepsilon_{33} = \varepsilon_{zz},$$
$$\varepsilon_4 = \varepsilon_{23} = \varepsilon_{yz},\ \varepsilon_5 = \varepsilon_{13} = \varepsilon_{xz},\ \varepsilon_6 = \varepsilon_{12} = \varepsilon_{xy}.$$

We will distinguish between the special case of strains that have the **shear** (non-diagonal) components equal to zero:

$$\boldsymbol{\varepsilon}^{\star} = (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, 0, 0, 0) \in \boldsymbol{\varepsilon}^{\mathbf{3D}},$$

while the general case will be referred to as $\boldsymbol{\varepsilon}^{\mathbf{6D}}$.

However, due to crystal symmetries presented and in order to avoid redundant computations, we want to make sure that each strain we apply to the crystal has a one-to-one correspondence to a distinct deformation case. The non-translational part of a homogeneous deformation of a crystal can be defined by a second-order deformation gradient tensor $\boldsymbol{F}$, which can be viewed as the Jacobi matrix linking deformed and undeformed lattice vectors. The relationship between the symmetric strain tensor $\boldsymbol{\varepsilon}$ and $\boldsymbol{F}$ is given by

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\boldsymbol{F} + \boldsymbol{F}^{\boldsymbol{T}}) - I, \tag{2.6}$$

where I is an identity matrix.

Since the band structure does not change upon rotations of the crystal, we can eliminate the rotational degrees of freedom by adopting upper triangular $\boldsymbol{F}$ to map out all deformation cases, as in Figure 2.2.

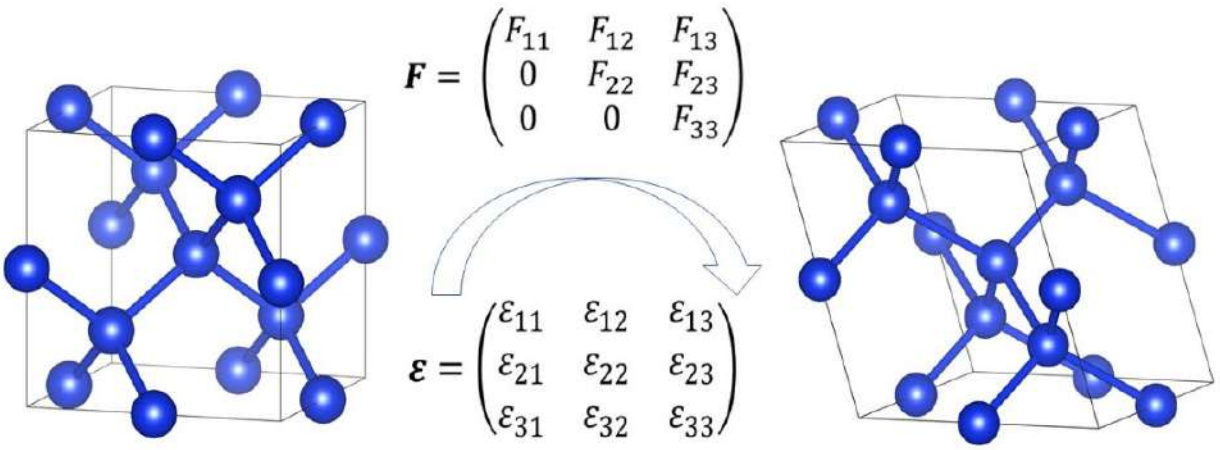


FIGURE 2.2: ESE achieved by applying a reduced deformation gradient tensor to the undeformed diamond cubic lattice of Si or C in the real space. The upper-triangular deformation tensor **F** ensures a one-to-one correspondence between the applied strains and deformation cases.

We want to note that the Brillouin zone for the deformed lattice is different from the undeformed one. The notation for the critical points, in this case, would alter a bit, see Appendix B for the additional details.

2.1.4 Elastic and inelastic deformations

The crystal deformation is called **elastic** if it transforms into the original unstrained shape after the removal of external forces; otherwise, deformation is **inelastic**. Methodologically, we are interested in the elastic deformations, as we may imagine the special devices which could strain the tiny crystal in order to control its properties on demand. However, one could design the

devices with less flexible and reversible changes of the crystal, opening the path to **inelastic strain engineering (ISE)**.

It is hard to determine whether the deformation is elastic or not based on the first-principles calculations. The real check should be performed *in situ*, in the experimental setup, as it is done in Banerjee et al. (2018). Instead, we check the structural stability of a crystal by performing the phonon stability calculations, i.e., checking the phonon spectrum (lattice vibrations) in order to estimate the stability of the crystal. In the forthcoming investigation of silicon and diamond crystals, we did our best to operate within the stable strains range.

This ends the basic introduction into the crystal structure. In the next section, we will focus on the physical equations that describe the states of the electrons in an ideal crystal, and the series of consequent approximations to them, which are necessary for the successful solution.

2.2 Density functional theory

Modern computational chemistry and computational materials science stem from two great discoveries of the twentieth century: the Schrödinger equation and the density functional theory, which paved the way towards its efficient numerical solutions. Both inventions were distinguished by Nobel prizes in physics: the first in 1933, and the second in 1998, more than half a century later.

This chapter will briefly introduce the basic concepts of density functional theory. Since this work focuses on *machine learning* with application to elastic strain engineering, this introduction will follow the steps from Sholl and Steckel (2011), focusing on the practical aspect of calculations as it is seen from the *ab initio* package user perspective.

2.2.1 From Schrödinger equation to DFT

We begin by introducing the time-independent, non-relativistic Schrödinger equation:

$$H\psi = [\frac{\hbar^2}{2m}\sum_{i=1}^{N}\nabla_i^2 + \sum_{i=1}^{N}V(\boldsymbol{r}_i) + \sum_{i=1}^{N}\sum_{j<i}U(\boldsymbol{r}_i,\boldsymbol{r}_j)]\psi = E\psi. \tag{2.7}$$

In the equation above, H is the Hamiltonian operator, E is ground-state energy of the electrons, and $\psi = \psi(\boldsymbol{r}_1,\ldots,\boldsymbol{r}_N)$ is an electronic wave function, which is a complex-valued function of each of the spatial coordinates of each of the N electrons. A complete representation of ψ should also include an electron spin, yet we omit it for the sake of clarity of presentation and the fact that spin does not affect our calculations[1]. The three terms in brackets

[1]This was tested by double-checking selected results with an *ab initio* calculations that account for the spin.

in this equation define the kinetic energy Δ of each electron, the interaction energy V between each electron and the collection of atomic nuclei, and the interaction energy U between different electrons.

The equation above is already subject to the **Born–Oppenheimer approximation**, in which the motion of atomic nuclei and electrons in a crystal can be treated separately. The main reason for the approximation is a simplification of consideration and the fact that the atomic nuclei are three orders of magnitude heavier than the electrons and thus respond to the changes in the system much slower compared to electrons. The equation (2.7) is written for fixed nuclei, and the wave function here is a function of $3N$ variables, a solution to a many-body problem. This makes it computationally intractable to consider any systems with even a moderate number of atoms.

The next step is to consider a density of electrons at a particular position in space:

$$n(\boldsymbol{r}) = 2\sum_i \psi_i^\star(\boldsymbol{r})\psi_i(\boldsymbol{r}).$$

Here, the asterisk is the complex conjugate, and the factor of two appears due to the Pauli exclusion principle, which states that each individual electron wave function can be occupied by two separate electrons with different spins. It turns out that the equation above may be described not in terms of the electronic wave function ψ but in terms of the electron density $n(\boldsymbol{r})$, which significantly reduces the number of dimensions of unknowns to 3. The two related fundamental theorems by Hohenberg and Kohn are:

- The ground-state energy E from equation (2.7) is a unique functional $\mathcal{F}$ of the electron density $n(\boldsymbol{r})$.
- The electron density $n_{\min\mathcal{F}}(\boldsymbol{r})$ that minimizes the energy of the overall functional $\mathcal{F}$ is the true electron density $n_{\text{true}}(\boldsymbol{r})$ corresponding to the full solution of the equation (2.7).

While these two theorems encourage researchers to use the electron density instead of the bulk wave functions, they do not provide any hints on how to construct the corresponding functional $\mathcal{F}$ except for the fact that given a "true" $\mathcal{F}_{\text{true}}$ we can find the electron density $n_{\min\mathcal{F}}(\boldsymbol{r})$ that minimizes its energy and this will be a solution to the equation (2.7).

The functional described by the first Hohenberg–Kohn theorem can be expressed in term of single-electron wave functions $\psi_i(\boldsymbol{r})$, involving both "known" parts, and "unknown", approximated parts, referred to as *exchange-correlation* (XC) functional (we refer to Sholl and Steckel (2011) for the full

description):

$$
\begin{aligned}
&E[n(\boldsymbol{r})] = E[\psi_i(\boldsymbol{r})] = \\
=&\frac{\hbar^2}{m}\sum_i \int \psi_i^\star \nabla^2 \psi_i d^3r + && \text{(electron kinetic energy)} \\
&+ \int V(\boldsymbol{r}) n(\boldsymbol{r}) d^3r + && \text{(Coulomb interaction between electrons and nuclei)} \\
&+ \frac{e^2}{2}\int\int \frac{n(\boldsymbol{r})n(\boldsymbol{r'})}{|\boldsymbol{r}-\boldsymbol{r'}|} d^3r\, d^3r' + && \text{(Coulomb interactions between pairs of electrons)} \\
&+ E_{\text{ion}} + && \text{(Coulomb interactions between pairs of nuclei)} \\
&+ E_{\text{XC}}[\psi_i]. && \text{(exchange} - \text{correlation functional)}
\end{aligned}
\tag{2.8}
$$

This energy can be minimized by solving the Kohn-Sham equations, which have the form:

$$
[\frac{\hbar^2}{2m}\sum_{i=1}^{N}\nabla_i^2 + V(\boldsymbol{r}) + V_H(\boldsymbol{r}) + V_{XC}(\boldsymbol{r})]\psi_i(\boldsymbol{r}) = \epsilon_i \psi_i(\boldsymbol{r}), \tag{2.9}
$$

for $i = 1, \ldots, N$. The main thing that discriminates this equation from equation (2.7) is that each equation involves only a single electron, and the main player here is an exchange-correlation potential V_{XC}, which is approximated in a number of ways, starting from the simple local density approximation (LDA) (Gross and Kohn, 1985), which assumes the ideal case of a uniform electron gas:

$$
V_{\text{XC}}^{\text{LDA}}(\boldsymbol{r}) = V_{\text{XC}}^{\text{electron gas}}[n(\boldsymbol{r})],
$$

up to the generalized gradient approximations (GGA) (Perdew et al., 1996):

$$
V_{\text{XC}}^{\text{GGA}}(\boldsymbol{r}) = V_{\text{XC}}[n(\boldsymbol{r}), \nabla(\boldsymbol{r})],
$$

and hybrid (Yanai et al., 2004; Heyd and Scuseria, 2004) or even meta-GGA (as in Zhao and Truhlar (2006)), which include more physical information or combine several levels of approximations.

In this work, we will use the Perdew–Burke–Ernzerhof (PBE) (Perdew et al., 1996) functional, which is a common choice for solid-state calculations. This functional is known for the poor performance within the semiconductors properties estimation. Hence, on the top of it, we also use the G_0W_0 correction (Shishkin and Kresse, 2006), which accounts better for excited states and results in better approximation compared to experimental data (Blase, 2003; Zanolli et al., 2007).

We can finally describe in general terms the self-consistent procedure of solving the equations (2.9) Sholl and Steckel (2011):

1. Define an initial trial electron density $n(\boldsymbol{r})$[2].

[2] Actually, in most *ab initio* packages, including VASP, this procedure starts with the trial ψ_i. After that, the electron density is calculated (step 3), and the procedure continues. The text remains unchanged since this part directly cites Sholl and Steckel (2011).

2. Solve the Kohn–Sham equations defined using the trial electron density to find the single-particle wave functions $\psi_i(\boldsymbol{r})$.

3. Calculate the electron density defined by the Kohn–Sham single-particle wave functions from the previous step, $n_{\mathrm{KS}}(\boldsymbol{r}) = 2\sum_i \psi_i^\star(\boldsymbol{r})\psi_i(\boldsymbol{r})$.

4. Compare $n(\boldsymbol{r})$ and $n_{\mathrm{KS}}(\boldsymbol{r})$. If they are the same, then this is the ground-state electron density, otherwise, $n(\boldsymbol{r})$ must be updated in some way and the process goes to step 2.

This procedure, which resembles the classic expectation-maximization approach in machine learning (Dempster et al., 1977), has many pitfalls and details we omit in this work since the majority is addressed by the Vienna Ab Initio Package (VASP) (Kresse and Furthmuller, 1996), which we use for most of our first principles experiments. In the next section, we will finally introduce the intrinsic crystal properties of our interest.

2.2.2 Electronic band structure and bandgap

Due to crystal symmetries and periodicity, solutions to the Schrödinger equation (2.7) can be written in the form of **Bloch waves**:

$$\psi_{j\boldsymbol{k}}(\boldsymbol{r}) = e^{i\boldsymbol{k}\boldsymbol{r}} u_{j\boldsymbol{k}}(\boldsymbol{r}), \tag{2.10}$$

where i is the imaginary unit, and $u_{j\boldsymbol{k}}(\boldsymbol{r})$ are periodic functions with the same periodicity as the crystal. For each value of $\boldsymbol{k}$, there are infinitely many solutions $\psi_{1\boldsymbol{k}}, \psi_{2\boldsymbol{k}}, \ldots$, that enumerate the **energy bands** – a special name for the functions that define in which positions and with which energies electron can exist in the periodic structure of a crystal. Bands are the result of the overlap of atomic orbitals, and for each band we can define a function $E_j(\boldsymbol{k})$ – the **dispersion relation** for electrons in that band.

To visualize the function $E_j(\boldsymbol{k})$, **band structure plots** are used. In these plots, the bands are drawn as $\boldsymbol{k}$ changes across the specific $\boldsymbol{k}$-path. An example is shown in Fig. 2.3. Band energies $E_j(\boldsymbol{k})$ are usually normalized by the **Fermi level** – the highest energy state occupied by electrons in a material at absolute zero temperature. The last band, occupied by the electrons, is called the **valence band** (VB), and the next to it is the **conduction band** (CB).

Arguably the most important property of the semiconductor materials is the energy **bandgap**, or simply bandgap, which is the difference between the maximal energy the valence band possesses and the minimal energy of the conduction band:

$$E_g = \min_{\boldsymbol{k}} E_{\mathrm{CB}} - \max_{\boldsymbol{k}} E_{\mathrm{VB}}.$$

As the difference of energy dispersion relations, bandgap is measured in units of electron-volts (eV).

The bandgap is a significant factor that determines the electrical conductivity: the larger the bandgap, the more energy electrons should possess to jump from the occupied valence band to the conduction band. According to the bandgap value, solids are classified as:

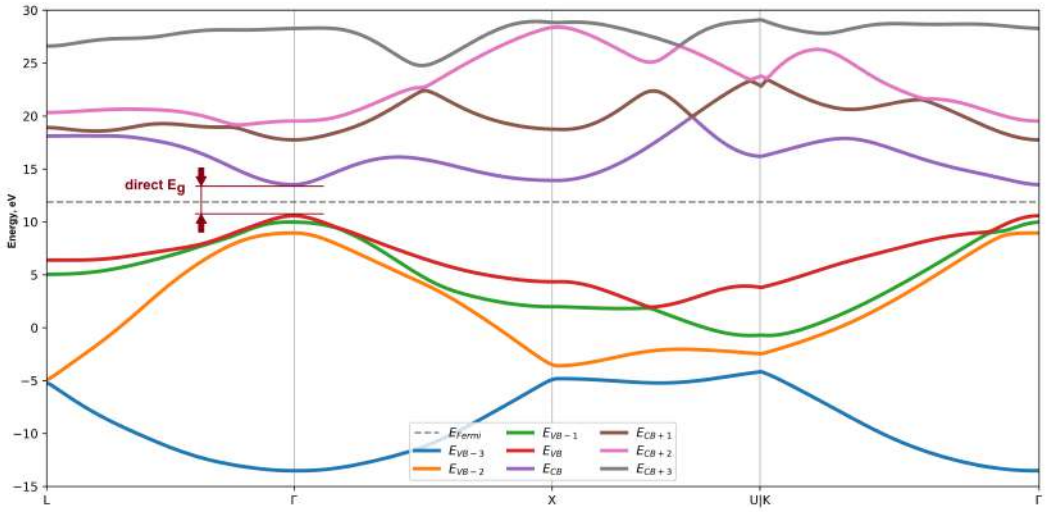


FIGURE 2.3: An example band structure along the $L - \Gamma - X - U|K - \Gamma$ $\boldsymbol{k}$-path for the diamond crystal under some strain $\boldsymbol{\varepsilon}_*$. In this case, the bandgap ($E_g = 2.94$ eV) is direct and shown on the picture; in the general case, one should check all the $\boldsymbol{k}$-points available within the calculation data. Both VB maxima and CB minima are located at the Γ $\boldsymbol{k}$-point.

- **insulators**, that have a large bandgap (e.g. ≥ 4 eV);
- **semiconductors**, that have a small bandgap (e.g. < 4 eV);
- **semimetals**, which have a negative bandgap;
- **metals**, which have a zero bandgap metals, which have a zero bandgap with the bands being filled up to the Fermi level.

The material has a **direct bandgap**, if the conduction band minimum (CBM) is situated at the same point(s) of $\boldsymbol{k}$-space as the valence band maxima (VBM):

$$\operatorname*{argmin}_{\boldsymbol{k}} E_{\mathrm{CB}} = \operatorname*{argmax}_{\boldsymbol{k}} E_{\mathrm{VB}}.$$

Otherwise, the bandgap is called **indirect**.

For materials with the direct bandgap, an electron from the valence band can be excited into the conduction band by a photon with an energy larger than the bandgap. In the case of the indirect bandgap, a *phonon*, or some vibrational motion, is required. This makes the direct bandgap a desirable property for optoelectronic materials, e.g., solar cells. Our *ab initio* calculations are restricted to the case of zero temperature (0 K); in the general case, the bandgap also depends on the temperature (Varshni, 1967), and the zero-point renormalization (Giustino et al., 2010) should be also taken into account.

Another property that could be obtained from the electronic band structure is the Hessian of the conduction band, evaluated at the conduction band

minima, called the **free electron effective mass tensor** $m^\star$:

$$\frac{1}{m^\star} = \frac{1}{\hat{h}} \begin{bmatrix} \frac{\partial^2 E_{CB}}{\partial k_1^2} & \frac{\partial^2 E_{CB}}{\partial k_1 \partial k_2} & \frac{\partial^2 E_{CB}}{\partial k_1 \partial k_3} \\ \frac{\partial^2 E_{CB}}{\partial k_2 \partial k_1} & \frac{\partial^2 E_{CB}}{\partial k_2^2} & \frac{\partial^2 E_{CB}}{\partial k_2 \partial k_3} \\ \frac{\partial^2 E_{CB}}{\partial k_3 \partial k_1} & \frac{\partial^2 E_{CB}}{\partial k_3 \partial k_2} & \frac{\partial^2 E_{CB}}{\partial k_3^2} \end{bmatrix} \Bigg|_{CBM}. \tag{2.11}$$

This tensor is used in transport calculations, e.g., electron transport or carrier density estimates. Both second derivative and inversion require calculations and corresponding surrogate models to be very precise.

We want to note that this tensor can also be calculated for the valence band at the valence band maxima, accessing the hole conductivity. However, in this work, we focus on the $m^\star$.

This work is mostly dedicated to the silicon crystal, which is a typical semiconductor in the unstrained state, and the carbon diamond crystal, which is an insulator in the unstrained state. Both silicon and diamond crystals have an indirect bandgap. In the next chapters, we will demonstrate that the strain may change the bandgap significantly as well as its type, opening new ways to smart engineering of materials. On this path, the main methodological instrument we rely on in the world of complicated and computationally expensive calculations is the **machine learning**.

2.3 Machine learning

2.3.1 General formulation

The term "machine learning" (ML) usually refers to the automatic construction of the models that can infer predictions based on the existing data or introduce specific labels to it (in case of clustering or anomaly detection algorithms). It usually differs from the statistics in terms of making fewer assumptions about the data and thus leading to more generalizable, "general-purpose" models (Bzdok et al., 2018).

In this work, we will be mostly focusing on the **supervised learning** – a setting in which we have a tabulated relation between the properties that describes the data (features) and the properties of our interest (targets, or labels), which we want to infer from the data not presented before. More specifically, we are focused on the regression problem, where the target variables are the subset of real numbers.

Let

$$y = f(x),\ x \in \mathcal{X} \subset \mathbb{R}^{M_{\text{input}}},\ y \in \mathbb{R}^{M_{\text{output}}}$$

be some unknown function which we want to approximate using the values from the training set

$$D_{\text{train}} = \{x_j^{\text{train}}, f(x_j^{\text{train}})\},\ j = 1, \ldots, N_{\text{train}}.$$

Suppose we have a model $\hat{f}$ that approximates the f:

$$\hat{f}\colon \mathcal{X} \to \mathbb{R}^{M_{\text{output}}}.$$

We train this model on D_{train} with a loss function

$$L(\hat{f}, D_{\text{train}}) = \sum_{j=1}^{N_{\text{train}}} ||f(x_j^{\text{train}}) - \hat{f}(x_j^{\text{train}})|| \tag{2.12}$$

as a fitting (optimization) criterion. Here, $||\cdot||$ is usually an L_2-norm.

If we denote the set of parameters that describes the model as ω, then the optimization problem of finding the optimal model parameters ω_{optimal}, which describe the training data D_{train} best (in terms of (2.12)), could be written in the following form:

$$\omega_{\text{optimal}} = \underset{\omega}{\operatorname{argmin}}\, L(\hat{f}(\omega), D_{\text{train}})). \tag{2.13}$$

The problem (2.13) is then solved using suitable optimization algorithms. A typical solution is sub-optimal (i.e., ω_{optimal} is a local minimum of $\hat{f}(\omega, D_{\text{train}})$, not the global one), since the obtaining of an optimal solution is an NP-complete problem for most of the modern models (see, e.g., Blum and Rivest (1992)). However, in most cases, practitioners rely on the sub-optimal solutions, as the training time is limited and modern models are powerful enough to describe the existing data. The training procedure for deep learning models is empirically shown (Mishkin and Matas, 2015) to benefit from the weight initialization (Glorot and Bengio, 2010; Klambauer et al., 2017). As the optimization algorithms are iterative (i.e., they perform the same procedure until convergence or stopping criteria meeting), one often tracks the change of the loss function on the separate set of the data, called the validation set:

$$D_{\text{val}} = \{x_j^{\text{val}}, f(x_j^{\text{val}})\},\ j = 1, \ldots, N_{\text{val}}.$$

Apart from the ω, model $\hat{f}$ usually has other properties to tune, called **hyperparameters**. Typical examples include the kernel choice for the kernel-based methods, a number of models in ensemble and ensembling type for the ensembling methods, and the architecture for the case of neural networks. In most of the cases (except for the Gaussian regression), the choice of hyperparameters is done via separate experiments on the validation set as well.

To measure the performance of different models, the residual between the prediction and the correct answer is done on the separate set, which was not used for the training or validation:

$$D_{\text{test}} = \{x_j^{\text{test}}, f(x_j^{\text{test}})\},\ j = 1, \ldots, N_{\text{test}}. \tag{2.14}$$

The three sets are chosen so they do not intersect:

$$D_{\text{train}} \cap D_{\text{test}} = D_{\text{train}} \cap D_{\text{val}} = D_{\text{val}} \cap D_{\text{test}} = \varnothing.$$

This alters a bit from the empirical risk minimization principle (Vapnik, 2013), according to which one should use the training set D_{train} to estimate the *generalization error*. Indeed, in line with the statistical learning theory, input data is assumed to be sampled from some probability distribution ($x \sim \mathcal{X}$), and the goal is to find the parameters ω that minimize the loss expectation, similar to (2.12):

$$\mathbb{E}_{X \sim \mathcal{X}} L(f(\hat{\omega}), X). \tag{2.15}$$

Nevertheless, (2.14) is still a good and unbiased estimate of the generalization error (2.15) given $D_{\text{test}}, D_{\text{train}}, D_{\text{val}} \sim \mathcal{X}$. As a matter of fact, the main reason for holding a separate set of data is the **overfitting** – an empirically observable or expected difference between the training loss (2.12) and the generalization error (2.15).

2.3.2 Overfitting and regularization

In the last two decades, the exponential growth of the computational power and the "rise of the data" (Cukier and Mayer-Schoenberger, 2013), expressed as the burst in the amount of information humankind generates, stores, and processes every year, have shifted machine learning trends to the models with the large expressive power. Theoretically, the richness of the class of functions $\hat{f}$ comes from may be measured in terms of the Vapnik–Chervonenkis dimension (Vapnik and Chervonenkis, 2015) for the binary classification problem, and more recent Rademacher and Gaussian complexities (Bartlett and Mendelson, 2002) for the regression. From the practical point of view, one could compare the complexities of two models from the same class based on the number of parameters ($\dim \omega$) and several hyperparameters.

Unless one knows the most suitable algorithm, general-purpose models are of great use. The downside of these models is their ability to replicate the training data nearly perfectly while failing to achieve the desired level of accuracy on the test set, resulting in overfitting. In some cases, this problem could be tracked on the training stage by comparing the loss values for the training and validation sets; in this case, the aforementioned stopping criteria based on the validation loss serves as the regularization (Zhang et al., 2005; Yao et al., 2007; Raskutti et al., 2014). Yet the most straightforward and widely accepted mechanism is the introduction of the special terms to the loss function, which are called **regularizers**:

$$L(\hat{f}, D_{\text{train}}) = \sum_{j=1}^{N_{\text{train}}} ||f(x_j^{\text{train}}) - \hat{f}(x_j^{\text{train}})|| + \lambda ||\omega||_R, \tag{2.16}$$

where $||\cdot||_R$ is usually an L_1- or L_2-norm. λ is used to tune the regularization level and is also considered as a hyperparameter.

Another popular option is the incorporation of the noise into the input or output data (noisy training or data augmentation (Van Dyk and Meng, 2001; Wang and Perez, 2017)) or into the model parameters (like dropout (Srivastava et al., 2014) or Gaussian noise injection (Wager et al., 2013)). In

these cases, one could also regulate the noise level and adjust it to the needs of the model.

A less popular option is a structural decrease in the model's complexity. For the neural networks, this includes decreasing the number of weights: removing unnecessary layers, decreasing the layer's width for the fully-connected parts, and removing extra filters and introducing dilation (Yu and Koltun, 2015) for the convolutional part. Kernel-based methods could reduce the overfitting by removing unnecessary parts of kernel or kernels. Decision trees could be regularized by the pruning (Mehta et al., 1995) or other structural diminishments (Friedman et al., 2001).

Thinking outside of the model, one could also come up with new data, resulting in a better sampling from the generative distribution, from which data comes. While effective and straightforward by itself, this method could be further improved by active learning, as will be described later.

In our models, we rely on the weight regularization and dropout in the case of the neural networks. For other algorithms, we would use the standard means of regularization as well.

2.3.3 Modern models in machine learning

Below we will briefly describe the classes of modern machine learning models, which are widely used in computational material science, computational chemistry, and in general. For a more comprehensive overview of the presented methods, please refer to Bishop (2006).

Kernel-based methods

In general, kernel methods are based upon the idea of *similarity*, which is defined on the pairs of points $(x, x') \in \mathcal{X}$ by the **kernel** function k:

$$k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R},$$
$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_V,$$

where feature map Φ maps into some dot product space V, and $\langle \cdot, \cdot \rangle_V$ is an inner product. Most of the popular algorithms avoid the explicit mapping and do operate within the feature space directly, thus enabling the linear algorithms to have a non-linear decision boundary (in case of classification); this is also known as the *kernel trick*. Algorithms such as Supporting Vector Machines (SVM) (Cortes and Vapnik, 1995) and Kernel Ridge Regression (KRR) (Vovk, 2013) learn a linear function in the space induced by the respective kernel k and the data; if k is a non-linear mapping, the resulting function in the original space $\mathcal{X}$ would be non-linear as well.

Kernel methods are arguably the popular choice for the Bayesian-based methods, especially Gaussian Processes (Rasmussen, 2004). The downside of these approaches is the need of kernel selection, which requires the user to know the data well, and near-cubic scaling with the training data, which is imposed by the need of operations on the kernel matrix $K \in \mathbb{R}^{N_{\text{train}} \times N_{\text{train}}}$ in

the inference scenario. However, several approaches to complexity reduction exist, such as the choice of anchor points (Liu et al., 2010) and approximate inference (Rasmussen, 2004).

Ensembles on trees

Another popular candidate for a general-purpose machine learning model is an ensemble-powered tree classifier, such as Random Forest (Ho, 1995) and Gradient Boosting (Friedman, 2001). Popular implementations, which provide a remarkable off-the-shelf functionality, include XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). These models consist of the ensembles of simple regressors (or classifiers), namely decision trees (Breiman et al., 1984), which are empowered by the bagging and boosting (see Bishop (2006) for the detailed explanation), as well as a number of empirical and performance tricks. Another neat property is that the base model – decision tree – could naturally work with the categorical data, which reduces the complexity overhead connected with the feature transformation into the numerical type.

Two main drawbacks of these models are the imposed piece-wise linearity of the feature space separation, which prevent the models to effectively address the non-linear relations in the small data scenario, and the connected memory overhead, which may appear on large data sets. The last problem could be addressed by imposing additional regularization, e.g., tree pruning (Mehta et al., 1995).

Neural networks and deep learning

The term "deep learning" is typically used to describe the complex models, which usually consist of several different parts (LeCun et al., 2015). However, due to the overwhelming popularity of neural networks in machine learning, it is now somewhat synonymous with the usage of neural networks within the model.

Most of the neural network architectures consist of several structural parts, called **layers**, which are connected to each other in a consecutive fashion (in simple cases) until they produce an answer. All the parts are usually **end-to-end** trained – the gradients from the loss function at the end of the network are passed backward to update the layers' parameters (weights); this process is referred to as *backpropagation*, and the corresponding models are thus fully differentiable. A simple, fully connected layer in the neural network may be described as:

$$S_i^h = \sum_{j=1}^{N_{h-1}} \omega_{ij}^h O_j^{h-1}, \quad i = 1, \dots, N_h, \tag{2.17}$$

where O_i^h is an output of the h-th layer of the neural network given by a non-linear transformation $\sigma(\cdot)$ of the corresponding pre-activation S_i^h:

$$O_i^h = \sigma\left(S_i^h\right), \quad i = 1, \dots, N_h.$$

One may construct the network using the fully connected layers only; we utilized this simple yet powerful model in Section 5.1.1 and in Shi et al. (2019) for a silicon crystal and use it as a baseline or supportive model for the diamond crystal case. For the special problems or the special data types (such as images, video, audio, text, sequences of variable length), special layers are developed. For instance, to reduce the number of weights to process the image as an input (which consists of millions of parameters!), the **convolutional** layers (LeCun et al., 1989) are used, which exploit the data structure (the pixels in the image are similar to their neighbours) and lead to the dramatic decrease in the parameter number, thus enabling rapid training. There are more special tricks to deal with images, as well as special layers and architectures for the different tasks, yet they are out of the scope of this work.

Different kinds of layers, together with symbolic differentiation, are implemented in a number of free-to-use packages, such as PyTorch (Paszke et al., 2019), Tensorflow (Abadi et al., 2016), and MXNet (Chen et al., 2015), to name a few. One of the great advantages is that the architectures, especially convolutional ones, are well-suitable to be accelerated on GPU (Oh and Jung, 2004).

One of the substantial disadvantages is that the neural networks are mostly "black-box" – it is hard to determine what influenced the decision making due to the complex structure of the typical deep learning model. Another disadvantage is the absence of a straightforward mechanism for the uncertainty quantification within the popular models. This problem will be addressed in detail in Sections 5.2.1 and 5.2.2, and is discussed in the next section as well.

2.3.4 Uncertainty estimation

In the modern machine learning, problems of estimating the uncertainty of model predictions usually arise in the context of applying such estimations for solving problems of adaptive design of experiments (active learning) and Bayesian optimization, while researchers pay relatively little attention to the accuracy of such estimates. Existing approaches to the uncertainty estimation usually work with a specific class of models based on probabilistic assumptions, such as linear regression and Gaussian processes-based regression models. However, despite the active development of Bayesian approaches for model training, the question of how to obtain uncertainty estimations for the most popular and accurate models, such as neural networks, is still being discussed.

The main methods for uncertainty quantification in neural networks include the following approaches:

- **Uncertainty quantification using model ensemble**

  This model for estimating uncertainty was proposed in the 1990s (Cohn, 1994; Cohn et al., 1996). The method itself, which consists in training of several models and estimating the variance (standard deviation) of their predictions, does not depend on the type of model; however, it is appealing in this particular case due to the large number of parameters

of the neural network, as well as the stochasticity of the learning process. Ensembles of neural networks (see Li et al. (2018b) for a detailed review) often boil down to an independent training of several models, which works well in some applications (Beluch et al., 2018), but is computationally expensive for the large-scale applications. There are many works aimed at the further diversification of the ensemble - models are made different from each other structurally (using different architectures (Opitz and Shavlik, 1996) and models (Lu and Bongard, 2009)), use different data subsets (bagging), (Zhou et al., 2002). Recently, forcing models in ensembles to be more diverse was shown to improve results even further in Jain et al. (2019). This approach is being actively used at the present time (Smith et al., 2018; Pearce et al., 2018) due to the simplicity of both the method and its implementation.

The main disadvantage of this approach is the increased amount of computational resources needed: using N neural networks trained independently increases the time required for training by N times and also requires N times more memory to store weights. This leads to a significant drop in the performance of the modern large neural network architectures. One of the options for speeding up this approach is to use previous weights (snapshot ensemble as in Huang et al. (2017)), but this reduces the diversification of models and imposes additional costs on the model weights storage.

- **Uncertainty estimation based on Bayesian approach**

  This approach is applied to Bayesian neural networks (Kononenko, 1989; Bishop, 1997), whose weights are random variables with an explicitly defined distribution (prior) that changes during the training process. In some cases, this technique makes it possible to explicitly express the variance of the model output (Richard and Lippmann, 1991) and is also suitable for the rapid generation of an ensemble of models on the fly. Unfortunately, the straightforward use of the Bayesian approach is very costly in terms of computational resources due to a large number of neural network parameters and large amounts of data, often accompanying the use of deep learning models. Most of the works in this field are aimed at reducing this complexity and theoretically justifying the existing heuristic methods for training models from the Bayesian point of view (Neklyudov et al., 2017; Molchanov et al., 2017; Teye et al., 2018; Matthews et al., 2018). It is worth noting that finding suitable prior weighting distributions of scales is especially important in the context of this approach (Hafner et al., 2018; Malinin and Gales, 2018).

  Another direction in uncertainty estimation is the incorporation of elements of random processes in the architecture and/or loss function for a neural network (Lee et al., 2017; Garnelo et al., 2018; Sun et al., 2018). This approach is associated with modern theoretical work in the field of Bayesian methods for neural networks but focuses on the practical component instead.

Connections between neural networks and Gaussian processes recently gain significant attention, see Matthews et al. (2018); Lee et al. (2017), which study random, untrained NNs, and show that such networks can be approximated by Gaussian processes (GP) in the infinite network width limit. Another direction is the incorporation of the GP-like elements into the NN structure, which may provide theoretical guarantees on the uncertainty estimates while preserving the power of NN, see (Sun et al., 2018; Garnelo et al., 2018) for some recent contributions. In Bradshaw et al. (2017), authors propose an NN+GP pipeline that may be trained end-to-end; however, the GP part slows down both training and inference, whereas in Sections 5.2.3 and 6.4 we use the GP for the inference only.

- **Dropout-based uncertainty quantification**

  Dropout (Srivastava et al., 2014) was first introduced as an empirical method to fight the correlation of weights of a neural network, and then found its theoretical justification as stochastic averaging of an ensemble of models (Srivastava et al., 2014), the realization of a Bayesian neural network with Bernoulli weights distribution (Gal and Ghahramani, 2016), and the hidden state model (Maeda, 2014).

  The main idea of dropout is to omit ("drop out") part of the activations of the hidden layer (while preserving the sample mean) during the training time. This allowed state of the art models to reach better accuracy at the cost of increased training time. Nowadays, most of the modern deep learning architectures use dropout.

  The breakthrough came with the idea of the use of a dropout not only at the training stage but also at the inference stage; this approach appeared recently in the works of Gal (see Gal (2016)), where uncertainty estimates based on the Bayesian approach and the dropout were proposed and analyzed. Most of the works in this direction are focused on the classification task (Kampffmeyer et al., 2016; Gal et al., 2017). However, most of the tasks of the so-called “physics of processes” are devoted to regression. The dropout-based approach is attractive due to the simplicity of its implementation and the possibility of application to the already trained models and established architectures. Dropout-based uncertainty often estimated as less computationally costly compared to the Bayesian analogues, and we exploit it as the main method for uncertainty estimation (UE) throughout the work.

The uncertainty quantification can be used to construct the confidence intervals of the model prediction (Cohn et al., 1996; Zhang et al., 2019; Pearce et al., 2018). In some cases, it makes sense to calibrate model predictions on a separate data subset (Kuleshov et al., 2018) to obtain more robust uncertainty estimates.

In the context of this work, we use the uncertainty estimation in order to rank the samples of the unlabelled data for the consequent first-principles calculations. The results of these calculations do enrich the training data and,

with further additional training, may lead to a more accurate model. This approach, in general, is called an **active learning** and is described in detail in the next section.

2.3.5 Pool-based active learning

Active learning (Settles, 2012) is a subfield of machine learning that operates under the following assumptions. Let

$$f_{\text{oracle}} : x \mapsto y,\ x \in \mathcal{X} \subset \mathbb{R}^{M_{\text{input}}},\ y \in \mathbb{R}^{M_{\text{output}}}$$

be an **oracle** function that, although its unknown, black-box nature, may be used on demand to get the predictions. Although the user and algorithms have access to this function, its usage implies some cost, which may be associated with the calculation time, cost of computational resources, or wage of human annotators. Luckily, a limited amount of **labeled data** (often associated with the training set) is usually available:

$$D_{\text{labeled}} = \{x_j^{\text{labeled}}, f_{\text{oracle}}(x_j^{\text{labeled}})\},\ j = 1, \dots, N_{\text{labeled}},$$

as well as an unlabeled data set

$$\mathcal{P} = \{x_j\},\ j = 1, \dots,$$

either finite or infinite, from which new instances may be drawn. In the finite unlabeled set setting, to which we will stick in this work, it is called **pool**. As before, we want to construct an approximator $\hat{f}_{\text{oracle}}$ that mimics the underlying oracle (in terms of generalization error), and also minimizes the total cost, i.e., uses the oracle as few times as possible. A general problem in active learning is to construct a sampling algorithm (called an **acquisition function**, or **querying function**) that ranks the samples from the pool based on the surrogate model:

$$\mathcal{A}(\hat{f}_{\text{oracle}}, \mathcal{P}) : \mathcal{P} \to \mathcal{R}. \tag{2.18}$$

Based on this mapping, data would be sampled (either single point at the time, or in batches), annotated by the oracle, and added to the training set, see the typical active learning cycle at Fig. 2.4. This process is usually continued until the needed accuracy of the surrogate model $\hat{f}_{\text{oracle}}$ is reached, new samples do not increase the accuracy for a few consecutive active learning iterations, or one is out of resources to use the oracle anymore.

In our setting, the role of an oracle is played by the VASP package that calculates the values of interest for us. Therefore, while the active learning framework is somewhat close to the machine learning framework described in Section 2.3.1, we should benefit from the opportunity of having an oracle at hand.

The main hypothesis of active learning is that one could come up with an acquisition function that outperforms a random sampling. In practice,

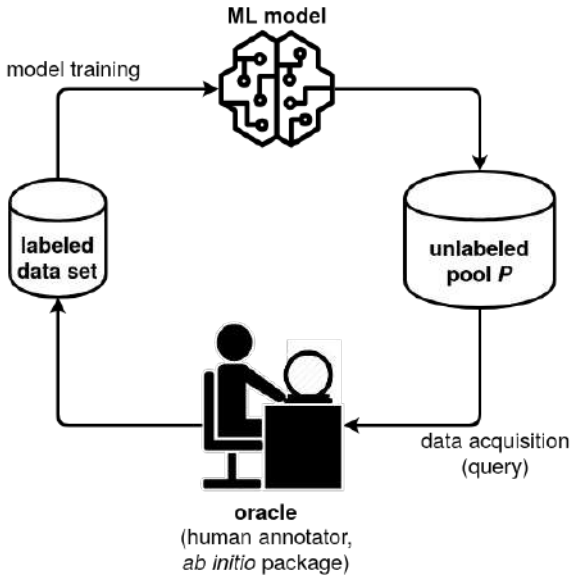


FIGURE 2.4: Pool-based active learning cycle. The machine learning model queries samples from the pool $\mathcal{P}$, which are then annotated by the oracle. Labeled samples are added to the training set, and the model is trained on the updated training set. Adapted from Settles (2012).

there is a number of scenarios for which we cannot be sure that there a better solution exists: for example, it is hard to come up with a suitable model in case of a very noisy oracle, which has some uncertainty in its predictions.

Another key assumption is connected to the construction of the acquisition function (2.18). It is natural to expect that the samples with the largest model error (the difference between the predictions and true answers) could be of great use if annotated by the oracle; we will later (in Section 5.2.2) refer to this as an **ideal error sampling**. This leads to the proposition, which is widely used by the active learning practitioners: uncertainty estimates, which are expected to correlate with the model's error, are the best candidates for an acquisition function. However, this is not true for a number of problems; we will provide the reader with examples in the experimental Section 6.3.

The last yet very important component of the active learning framework is the ability of the model to be trained on the additional data, which is called **warm-start training**. One could also train the model from scratch; however, this may require additional time and may affect the acquisition function (2.18), which depends on the approximator $\hat{f}_{\text{oracle}}$. Moreover, data-hungry ensemble models and neural networks would not benefit from a few additional points; active learning for them is performed in the batch scenario, which has its own drawbacks (like sampling two close points from the same region of features space). In general, for the modern machine learning methods, several approaches exist:

- In the case of the Gaussian Processes, two scenarios may appear:

1. If active learning results in a few new data points added, one could just recalculate the kernel's parameters for a prediction.
2. If a significant amount of data is added, one could fix the hyperparameters and recalculate the covariance matrix in an effective way (Burnaev and Panov, 2015). The hyperparameters may be updated later.

- For the ensembles on trees, one could simply add more trees to the ensembles and possibly re-weight them in a fashion of making the recent prediction more important.
- In the case of neural networks, one could simply run more epochs (full sweeps throught the training set) on the updated data set. Shim et al. (2020) propose to emphasize the new samples by amplifying the gradients. Still, Ash and Adams (2019) claim that warm-start training can hurt generalization.

This concludes the general introduction to the machine learning models, which are used in this work. In the next section, we will review the existing works and approaches to the elastic strain engineering and surrogate modeling in the materials science.

3 Related work

3.1 Elastic strain engineering

3.1.1 Industrial perspective and motivation

One of the first questions that may appear is a choice of materials to study with the elastic strain engineering: why do we focus on the silicon and diamond as the primary candidates for investigation?

The answer for the silicon is pretty straightforward: it is a cheap and universal semiconductor, which appears in the transistor industry; main use cases include the CMOS (Complementary metal-oxide semiconductor) and MOSFET (metal-oxide semiconductor field-effect transistor) production. A nice review of the history of strained silicon from the industrial point of view can be found in Chidambaram et al. (2006). The main point igniting the researching interest starts from the early 2000s research and patents by IBM (Huang, 2001), AMD (Xiang, 2003), and later Intel (Bohr, 2007) (all connected to the old 1993 patents by IBM and Amberwave Systems (Brasen et al., 1993)) on the strained silicon technology in a variety of chips and transistors. A 1% uniaxial strain in a certain direction offered a large boost in the carrier mobility – up to 50% (Bedell et al., 2014) and increased central processing unit (CPU) clock speed correspondingly. Strain engineering remained a hot topic for a long time: see Figure 3.1 for the statistics on the strained silicon patents for the last 30 years. A more technical description of the strained Si technology can be found in Mistry et al. (2007).

As for the diamond crystal, it has an ultra-wide bandgap of 5.5 eV in the unstrained state; moreover, superior properties in terms of durability and melting temperature makes it quite challenging to bend. However, in Banerjee et al. (2018), the possibility of a diamond straining was proven experimentally; the strain amount both materials can withstand is fantastic: a 9% local tensile strain for the diamond and 16% for the silicon (Zhang et al., 2016) opens a window of opportunity to explore the vast strain space in the quest for better materials.

We would like to note that from the experimental perspective, it is important to measure the actual strain of the deformed material. This is usually either estimated via theoretical finite-element simulations, as in Banerjee et al. (2018), or experimentally via transmission electron microscopy (Hÿtch and Minor, 2014).

A comprehensive overview of the state-of-the-art techniques and results for the elastic strain engineering can be found in Li et al. (2014); we will discuss some important milestones from it below.

FIGURE 3.1: A general overview of the patents for the strained silicon in 1990–2020. Data is taken from the Google Patents system on the "strained silicon" query: https://patents.google.com/?q=strained+silicon&oq=strained+silicon. One can observe a significant rise in the number of patents together with patent-holders diversification in the 00s.

3.1.2 Notable materials for ESE

A tremendous number of works is dedicated to the straining of semiconductors and materials. A short overview of the leading research directions is provided below.

- **1D and 2D materials,** primarily carbon- and silicon-based, deserve a separate chapter with an overwhelming number of both theoretical and experimental works on it; we would restrict ourselves with a few selected works related to the elastic strain engineering.

  In Guinea et al. (2010), authors investigate how strain can lead to the pseudomagntic field in graphene without changing the bandgap; they also discuss various scenarios on how to strain the nanosheets. Peng and De (2014) studies the elastic limits for silicene (2D silicon), silicane (silicene with the hydrogen group) for a wide range of strains (up to 20%) via PBE calculations. Topsakal et al. (2010) explores the bandgap change and plastic transition for graphane via GW calculations. In Minamisawa et al. (2012), an industrial technology of silicon nanowires is described. In Kumar et al. (2015), a comprehensive description of monolayer graphene membranes is provided.

  As for other 2D materials, Fei and Yang (2014) explores the phosphorene (2D P) changing conducting direction under 6% uniaxial and 5% biaxial strains. Guo et al. (2015) explores Bi_2Te_3 nanosheets for the engineering of topological surface states. In Wang et al. (2013), a reversible lattice deformation for Ni nanowires is described.

Another interesting direction is the research on "quantum dots" (also referred to as 0D materials); in Kuklewicz et al. (2012), authors investigate exciton energy tuning with strain, using glue for the elastic deformation of InGaAs.

- **Germanium (Ge)**. Germanium was shown to have a direct bandgap under specific biaxial strain (El Kurdi et al., 2010), which makes it another desirable alternative for solar cell development and other optoelectronic applications (Nam et al., 2012). Sukhdeo et al. (2014) describes a direct bandgap Ge on Si sheet provided by 5.7% uniaxial strain (epitaxy).

- **Ferroic ($SrTiO_3$, $BiFeO_3$, $EuTiO_3$, ...) and other oxides**. A nice compilation of both theoretical and practical results for ferroic oxides can be found in Schlom et al. (2014). Yildiz (2014) studies effects on electrocatalysis and diffusion, discovering the tremendous influence of strain on the molecule adsorption energy, dissociation barrier, and other properties. Jang et al. (2008) explores polarization direction tuning via deformation. In Cao et al. (2009), a Mott transition for VO_2 under elastic strain is described. Fu et al. (2014) explore -5% to 5% strains for the ZnO semiconductor in terms of exciton dynamics.

- **Aluminium and other metal alloys**. In Reddy (2004), authors demonstrate the strain-stress curves (obtained via *in situ* experiments) for the Al-Si-C alloys under the uniaxial strain up to 5%. Another study (Cao et al., 2015) measures elastic strain energy density, similar to proposed in Section 4.2.2, and derive fracture criteria based on the experimental analysis of more than 7000 specimens of aluminium alloy. A shape memory of NiTi alloys and investigation of its properties under inelastic strain is described in Du et al. (2015). Mosca et al. (2008) investigates the magnetocaloric effect and magnetic phase transition under elastic strain. In Yan et al. (2016), authors show how strained films change the catalytic activity in hydrogen evolution reaction.

- **Molybdenum disulfide (MoS_2)** has attracted notable attention due to nice properties in the layered form. While bulk MoS_2 crystal has an indirect bandgap of 1.8 eV, its layered form shows a 1.2 eV direct bandgap, making it a notable candidate for the various microelectronic applications. Experimental works show the possibility of creating microprocessors (Wachter et al., 2017), as well as memristors (Li et al., 2018a; Xu et al., 2019) with MoS_2. In Conley et al. (2013), authors use both GW_0 calculations and experimental results to estimate the effect on the optical bandgap (which differs from the usual bandgap by excitonic binding energy) for small (up to 2.2%) uniaxial strains. Experimental continuous tuning of the electronic band structure was demonstrated in He et al. (2013). Castellanos-Gomez et al. (2013) demonstrates the experimental measurements combined with the tight-binding model calculations for the strains up to 2.5%. Li et al. (2016) considers MoS_2 with vacancies of S (from 0.1 to 5%) and a small range of strains (up to 1%).

A large strain (up to 9% biaxial strain) appears in Feng et al. (2012), where the idea of a solar funnel is investigated via G_0W_0 calculations.

In the next section, we will take a step away from the topic of elastic strain engineering and will discuss the related work for the machine learning-*ab initio* simulation interaction.

3.2 ML-assisted simulation

This section is related to a brief overview of the machine learning models used for materials science. More specifically, we will discuss the models that are used on the top of (or based on) the first-principles calculations, with a detailed description of the approaches similar to the ones we design in this work.

3.2.1 ML-simulation taxonomy

We start with a concise description of both means and methods of machine learning and simulation, and will refer to (von Rueden et al., 2019, 2020) for a detailed discussion. This section aims to answer the general methodological question: in what ways machine learning models and simulators can interact and benefit from each other? This allows our approach to find a place among similar approaches used in materials science and surrogate modeling.

While **machine learning** starts from the data and the end goal is to construct the model, which could be used for inference or prediction, **simulation** starts from the model, which generates the data that supports decision-making. In other words, machine learning produces **inductive** models (that generalize from the data) by its nature, while simulations use a **deductive** model, which is usually provided by the physical laws or its approximations. A typical example of the machine learning model is the prediction of handwritten digits: given the labeled data of images and corresponding digits, one may train a model (taking, for instance, the CNN as a model type) to infer the class of a new image; this model can be used, for example, to enhance the text recognition for post offices, archives, etc. A typical simulation, in turn, is based on the physical model of a particular phenomenon and is used to derive the data; for instance, the model for weather forecasting includes a (data-free) system of partial differential equations that characterizes the development of air flow parameters (such as pressure, air velocity and others) over time. This model is fed with the data from the weather towers as an input, and the prediction is used to describe the weather in a particular region. The general overview that summarizes the aforementioned taxonomy, adapted from (von Rueden et al., 2020), is shown in Figure 3.2.

Both machine learning and simulation could benefit from each other on every step of the model construction process and during the inference; however, as this work is dedicated to the machine learning enhancement of the

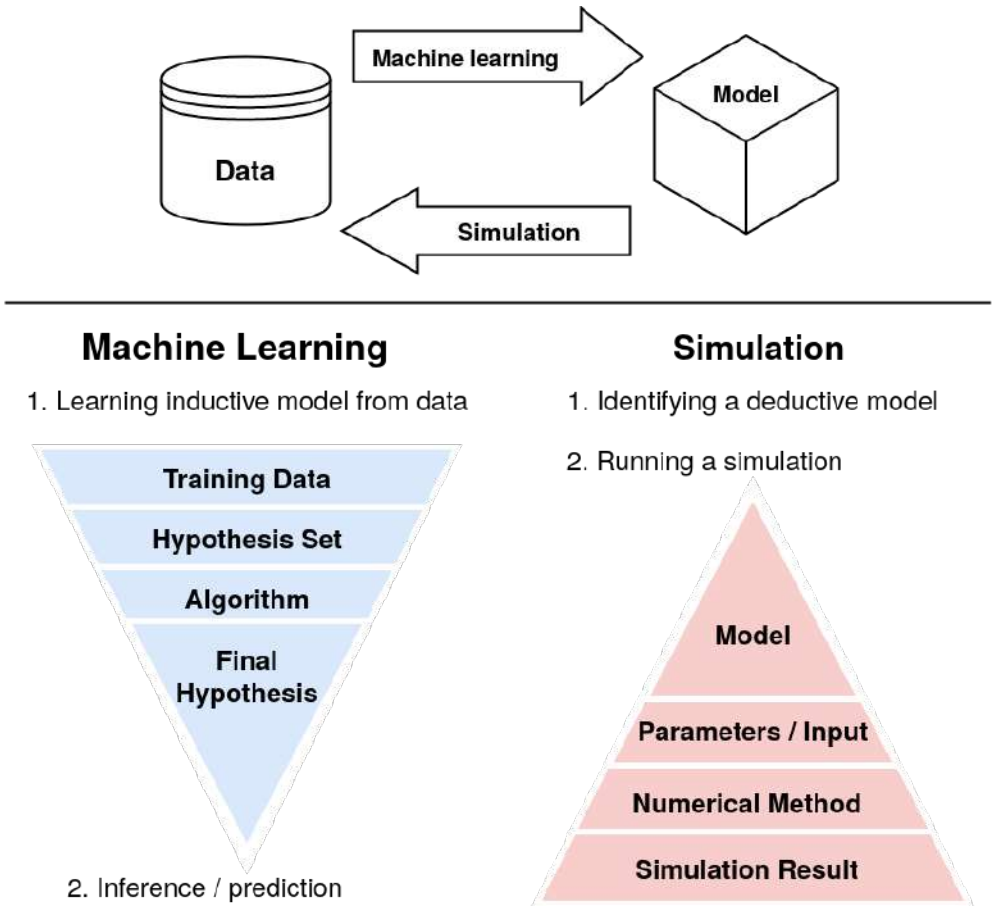


FIGURE 3.2: Components of machine learning and simulation. **Top:** machine learning produces inductive models (that generalize from the data) by its nature, while simulations use a deductive model, which is usually provided by the physical laws or its approximations. **Bottom:** typical components of machine learning and simulation. Adapted from (von Rueden et al., 2020).

first-principles calculations (=simulations), we will focus on this particular scenario. In terms of taxonomy shown in Figure 3.2, machine-learning assisted simulation can be performed in the following non-mutually exclusive cases (as described in von Rueden et al. (2020)):

- **Model reduction and surrogate modeling**. The most widely used scenario is a replacement of a heavy underlying model with an approximate yet fast ML model, which is known as **surrogate modeling**. These models, as an example, could be trained on cheaper data from other simulations, or experimental results. Another possibility is the model order reduction (see Schilders et al. (2008) for review), which is usually based on the matrix or tensor decomposition of the underlying high-order formulation. It should be noted that, depending on the context, the full replacement of the simulation routine with the ML model is also referred to as surrogate modeling.

- **Input data choice**. Machine learning offers a variety of techniques for the intelligent data sampling and experimental design, starting from space decomposition (e.g., using the Latin hypercube sampling (McKay et al., 1979)) up to Bayesian models-assisted sampling (Burnaev and Panov, 2015). An active learning scenario for the surrogate model construction also belongs to this category, setting up the inputs for the next simulations.

- **Numerical methods enhancement**. Another important integration type is the replacement of numerical techniques with ML analogues. A typical example is the injection of interatomic potentials within the process of molecular dynamics or atom relaxation (Podryabinkin and Shapeev, 2017).

- **Result processing for scientific discovery**. The last yet arguably the most significant usage of machine learning lies within the interpretation of simulation results: pattern recognition and assisted optimization, which could be performed using a rich set of tools like clustering and manifold learning.

We would like to note that the ML machinery we design may also benefit from the physical knowledge and insights incorporated; see Sections 4.3 and 5.1.2 for details. Moreover, the most relevant results are verified with the simulations, like all the particular deformation cases listed in Section 7.1. The general view on the methodology used for the ESE learning is discussed in Section 4.1.

In the next part, we will review the important works on the machine-learning-assisted *ab initio* calculations.

3.2.2 Selected works in ML-assisted simulation

As we are focused on the works related to the first-principles calculations and simulations, we begin with a few words on the main problems, and challenges in this field. This section can be considered as a short overview of main data sources, problems, models and challenges in the field of machine learning applications to chemoinformatics and materials science, as many of these models influenced the methodology described in this work.

Databases

We start with the data; in terms of machine-learning like, ready-to-go dataset, we should mention the various datasets collected through extensive *ab initio* calculations. First of all, it is QMX (QM7, QM9, etc.) and GDB-X (GDB-7, GDB-9, GDB-11) datasets, which provide a mapping from the organic molecules to its properties, which were calculated using DFT-like XC-functional (B3LYP (Yanai et al., 2004) in most of the cases). These databases serve as a benchmark for different ML-based methods to compare. We would like to note that there exists a tremendously large database of 166

billion molecules (GDB-17, (Ruddigkeit et al., 2012)), which does not provide any properties yet may be suitable for the large unsupervised exploration in future by, e.g., learning the lower-dimensional representations, or grouping molecules into the clusters.

Among other notable databases, we would like to mention an AFLOW database (Setyawan and Curtarolo, 2010; Curtarolo et al., 2012; Rose et al., 2017) of more than 3.2 million compounds with over 588 million calculated properties, which also provides a friendly interface for easy access (Taylor et al., 2014). The creation and existence of such databases boosted the development of computational materials science and slightly pushed it to the ML side in the 2010s.

While these databases offer a tool to explore and validate the machine-learning models that may assist in simulation, a real use case is to somehow connect these to the "oracle" that performs the first-principles simulations. Therefore, some authors do not test their models extensively on these databases and report the performance on open problems instead, marrying their ML machinery with the simulation software.

Problems

Most of the properties of interest require not a single calculation (as for the bandgap or electronic band structure case) but a series of calculations and approximations; examples include thermoconductivity, reaction rate estimation, and so on. Two important types of multi-step *ab initio* calculations are molecular dynamics (MD) that describes the system evolution in time and is usually related to the interaction of molecules, and relaxation, that describes the system evolution under external or internal forces, and is closely related to materials science. Both procedures require additional QM calculations, and one of the most exploited ways to enhance the speed is related to "simulating" the call to a first-principle calculation by replacing the result with the ones produced by an ML model.

Another challenge is connected to the simulation of large-scale systems. When the system size reaches thousands and tens of thousands of atoms, most of the calculations break due to the memory limitations or require too much time. Therefore, another approach machine learning can help with is connected to breaking such a system into parts (which are pretty similar to each other) and then assembling the contribution of each into the final answer. To that end, most of the modern models related to simulation impose **locality** – an ability to operate within a certain range for each atom.

Notable models

As can be seen in the Schrödinger equation (2.7), the properties of the system of the atoms are fully described by the atom types and the distances between them. The main question is how to operationalize this knowledge and embed it within a model – in other words, how to extract useful features.

A classic approach requires the calculation of pairwise distances between pairs of atoms, and angles between the triples (quadruples, etc.) of atoms,

within a certain cutoff radius of each atom. These quantities are then combined in a way to represent the local environment for each atom, and the final answer is parametrized over the features within each local environment and all the environments in total. An important note is related to the various symmetries such a model should impose; usually, it is a translational and rotational symmetries, as well as the permutational symmetry (you can swap any two atoms of the same type, and the system would be the same). Popular models could be roughly divided into the following categories:

- BoB-based (Bag of Bonds) models and its modifications are usually constructed on the aforementioned type of features and are KRR-based (Hansen et al., 2015; Brockherde et al., 2017; Chmiela et al., 2017; Hansen et al., 2013; Chmiela et al., 2018). A comprehensive review can be found in Huang and Von Lilienfeld (2016).

- NN-based models that inspired by the Behler–Parinello networks (an extensive review on early models is provided in Behler (2011)) do construct similar types of features automatically of semi-automatically and are usually fully-differentiable, that enables easy end-to-end training. The downside is that for most of the cases, such networks need an "interaction" (or message-passing) steps that are launched in advance to the usual parameter training by gradient descent; the performance of the network is usually dictated by the number of such steps, which ensures additional hyperparameter manipulation in order to find the most suitable model. Another notable downside is a moderate performance on small-scale datasets. Modern models of this class include message-passing neural networks (MPNNs, described and classified in Gilmer et al. (2017)), ANI-1(x) models (Smith et al., 2017; Zubatyuk et al., 2019), and SchNet-related models (Schutt et al., 2017; Schütt et al., 2018).

- A curious approach based on the gradient boosting trees is presented in (Isayev et al., 2017), where authors propose a graph-based dissection of a crystal system.

- Another important approach is GAP (Gaussian Approximation Potentials), developed in Bartók et al. (2010), where authors propose to use Gaussian Processes on the basis set of 4D spherical harmonics. Among non-NN based approaches, this approach is by far among the most accurate approximations yet is the most computationally expensive one.

- An interesting approach based on the Moment Tensor Potentials was proposed in Shapeev (2016). It offers a feature representation based on the invariant polynomials and could be classified as a generalized linear model from the ML point of view. A small number of trainable parameters coupled with a suitable feature set results in the most effective performance among the non-NN models (Zuo et al., 2020).

These models (referred to as **interatomic potentials** as they take into account interaction on the atomic level) are universal in the sense that they

could learn any property of a system. A lot of different approaches, which are tailored to the specific figures-of-merit (like the one described in this work) are not included in this small review. For a more detailed review of the application to energy materials, we refer to Chen et al. (2020).

Steps to discovery

Of course, one could solve the problems of material optimization and chemical discovery without such potentials, on the pure DFT (or with the help of inaccurate semi-empirical potentials). One of the most famous examples is USPEX (Universal Structure Predictor: Evolutionary Xtallography, Oganov and Glass (2006)), which is based on the genetic algorithm, and led to many discovered structures that were then proven experimentally. Among the other works, we would like to note a similar approach to the bandgap engineering in Chang and von Lilienfeld (2018), where authors use a combination of gradient descent and genetic algorithm to generate the crystal structures with a large direct bandgap. Both (and many more) exploration methods can benefit from the ML model incorporated into the discovery algorithm.

The main purpose of the interatomic potentials mentioned in a previous section is not to fit a given dataset perfectly but to deliver useful and explainable results for a given problem. In terms of ML-assisted simulations, all of these models can be used as a surrogate to the *ab initio* calculations, but only a few can provide a mechanism for the active learning or estimate its own error.

One important approach is related to ANI-1 type networks. In the subsequent works, the authors propose two notable extensions to the simulation process. In Smith et al. (2018), they develop an active learning mechanism based on the ensemble of ANI-1 networks and demonstrate that the corresponding uncertainty estimate actually correlates with the model error and is also useful in an active learning scenario. They also show that the data fusion scenario, implemented via transfer learning, is capable of reaching better accuracy with a small amount of high-fidelity data in Smith et al. (2019). Both introduced features – transfer learning and active learning – are exploited in a similar manner in our work. Compared to other NN-based potentials, works on ANI-1 demonstrate effectiveness and applicability to the ML-assisted simulation.

Another outstanding approach is explored in the consecutive works on MTP. In (Podryabinkin and Shapeev, 2017; Gubaev et al., 2018), authors develop an active learning methodology that estimates the uncertainty based on the distance from the training set (with the D-optimality criterion), and evaluate its effectiveness in different scenarios. These include the study of diffusion processes (Novoselov et al., 2019), application to the chemical reaction rate estimation (Novikov et al., 2018), accelerating the alloy discovery (Gubaev et al., 2019), and combination with the USPEX machinery for the boron structures prediction (Podryabinkin et al., 2019).

Apart from the direct simulation acceleration, we would like to note a few works that use the ML machinery to explore the intrinsic properties of the chemical space. These include a high-level review on how similar models can be used for materials properties exploration (von Lilienfeld et al., 2019), an explainable analysis of "feature importance" on the molecular level (Schutt et al., 2017), and tSNE-like deep exploration of the chemical space (Karlov et al., 2019), to name a few.

Related works

We would like to conclude this chapter with a brief discussion of works especially related to ours. Apart from the aforementioned "alchemical" approach to the bandgap engineering, described in (Chang and von Lilienfeld, 2018), bandgap prediction with the help of neural networks was first addressed in (Zhaochun et al., 1998) and later appeared in a number of works; we would refer to the high-level reviews in (Schmidt et al., 2019; Chen et al., 2020). A more complicated take on the dispersion relation and band structure are made in (Malheiros-Silveira and Hernandez-Figueroa, 2012; Pilozzi et al., 2018); a tight-binding model functional approach is discussed in Peano et al. (2019). Since the band structure prediction is only the part of the problem in some cases, some authors propose a more target-oriented approach to the prediction of band-crossing and other topological features (Zhang et al., 2018; Claussen et al., 2019).

The method presented in this work is focused on the electronic band structure prediction for a very specific problem – elastic strain engineering of semiconductor crystals – and offers a number of ideas on the physical-informed model construction as well as develops a framework for an effective interaction with the *ab initio* simulations. In the next chapter, we will discuss the details on both model design and the overall methodology of ML-assisted simulation applied to the specific problem of ESE.

4 Methodology

4.1 General ESE methodology

We begin the methodology chapter with the high-level formulation on how exactly the machine learning model for ESE and the corresponding first-principles calculations affect and complement each other. This section serves the purpose of both a general view on the ESE models we design and a brief overview of the upcoming methodology sections.

From the ML-assisted simulation perspective, mentioned in Section 3.2.1, the components of the whole *ab initio* simulation process for ESE may be described in the following way:

- **Model** is the density functional theory, which, in turn, is a handy approximation described in Section 2.2.
- **Parameters and input data** are the VASP settings, described in Section 4.2 and listed in Appendix C, and the deformation tensor $\boldsymbol{\varepsilon}$, correspondingly. In terms of ML-simulation interaction, we will refer to the strain value $\boldsymbol{\varepsilon}$ only.
- **Numerical methods** are the set of internal VASP routines for convergence of band energies.
- **Simulation result** is ultimately an electronic band structure, represented as a rank-4 tensor, or a set of these calculations. Three out of four dimensions represent the **k**-space coordinates, and the last dimension denotes the band number, see Section 5.1.2 for details.

From the machine learning side, the process parts are as follows:

- **Training data** is a set of strain-band structure pairs $(\boldsymbol{\varepsilon_j}, E(\boldsymbol{\varepsilon_j}))$ in the general case or set of strain-bandgap pairs $(\boldsymbol{\varepsilon_j}, E_g(\boldsymbol{\varepsilon_j}))$ for the simplified case of silicon crystal.
- **Hypothesis set** is an underlying model class we use for training: NNs and CNNs, with a particular variants described in detail in Sections 5.1.1 and 5.1.2, correspondingly.
- **Algorithm** is a standard gradient descent algorithm with variations tailored to the NN architectures, such as Adam (Kingma and Ba, 2014).
- There is plenty of **final hypotheses** we test with the developed ML machinery, starting from the extreme bandgap values (see Section 7.1) up to the exploration of the band structure topology (Section 7.2).

The methodological approach implemented in this work fuses both ML-assisted simulation (see Section 3.2.1) and simulation-assisted machine learning, resulting in a rapid exploration of the strain space. In particular, two following parts of simulation benefit from the ML model:

- **Input data** for the simulation may be provided by the active learning algorithm within the CNN, see Section 5.1.2 for details and Section 6.1.2 for the empirical estimate of active learning applicability for the ESE project. As a side note, not related to strain engineering, the developed active learning algorithms (Sections 5.2.1, 5.2.2, 5.2.3) were tested in a number of problems and scenarios, with quantified results presented in Sections 6.2, 6.3, and 6.4. Of special importance are the numerical experiments that emulate the real experience of interaction with the simulator: active learning with the SchNet (Section 6.4.3) and surrogate model for the fluid flow in a wellbore in drilling scenario (Appendix E).

- A machine learning-based approach with the surrogate modeling makes it possible to detect patterns and test hypotheses for **scientific discovery**. Optimization routines that are looking for the best combination of figures-of-merit and straining effort require tens of thousands of calculations. Large maps describing how the figures-of-merit evolve within the vast strain space, like the ones presented in Section 7.1 require millions of calculations; discoveries presented in this work would simply be inaccessible for plain DFT calculations.

The surrogate machine learning model we design benefits from the calculations in the following ways:

- Apart from the obvious benefit of having an "oracle" (in terms of Section 2.3.5) for querying data by demand, *ab initio* simulations offer another possibility of obtaining fast approximations of the electronic band structure by using cheap yet inaccurate PBE calculations without the GW approximation step. This "low-fidelity" data enables the option of fast preliminary training for the corresponding model, see Section 5.1.2 for details for the diamond ESE model; throughout the text, we refer to this approach as *data fusion*. Another possibility is the so-called "Δ-ML" approach when one model is trained on the inaccurate data, and another is fitted either on the difference between inaccurate and accurate data answers, or having the prediction for an inaccurate data as an input. This scenario was implemented in the silicon model case and described in Section 5.1.1.

- The **hypothesis set**, which corresponds to the class of ML models we use, was specifically tailored to the known physical symmetries and properties, including ones derived from the data. Section 5.1.2 is dedicated to this translation from the data- and physics-derived properties into the language of ML machinery.

- Last yet not the least, is thorough **verification of the selected results** discovered using a machine learning model with additional simulations. As an example, all the particular deformation cases listed in Section 7.1 were double-checked by separate GW calculations.

In the next section, we will take a close look at how the first-principles calculations are organized.

4.2 First-principles calculations

4.2.1 General settings

We used the Perdew-Burke-Ernzerhof (PBE) (Perdew et al., 1996) exchange-correlation functional and the projector augmented wave method (PAW) (Blochl, 1994) in our DFT simulations implemented in the Vienna Ab initio Simulation Package (Kresse and Furthmuller, 1996) with spin-polarization incorporated. Computations that invoke GW corrections were run based upon the same PBE-PAW settings; we utilized the following three-step procedure:

1. **Preliminary ground-state calculation**. This is a standard preparation step[1], as GW calculations always require a one-electron basis set. We will refer to the results obtained on this step as PBE results: PBE band structure, PBE bandgap, etc. This step is computationally cheap.

2. **DFT virtual orbitals calculation**. Another step on the way on GW calculation preparation, we need to enrich the band structure with a reasonable number of virtual orbitals (50-100 per atom, as recommended in VASP tutorials).

3. **GW calculation**. In order to save time (after all, we need thousands of calculations!), we utilize the "single-shot" quasi-particle energies method, referred to as G_0W_0. This method operates on the diagonal matrix elements of the self-energy only, with a Taylor expansion of the self-energy around the DFT energies as an approximation step.

The listings for the input INCAR files are provided in the Appendix C.

In all calculations, a plane wave basis set with an energy cutoff of 600 eV was adopted to expand the electronic wavefunctions. The Brillouin zone integration was conducted on an $8 \times 8 \times 8$ ($6 \times 6 \times 6$ for silicon calculations) Monkhorst-Pack (Monkhorst and Pack, 1976) **k**-point mesh. We had deliberately reduced the symmetry setting to include the **k**-space reflection symmetry only (`ISYM = 0`) to unify the data processing pipeline. Atomic coordinates in all the structures were relaxed until the maximum residual force was below 0.0005 eV Å^{-1}.

Density functional perturbation theory (DFPT) calculations were performed to determine the phonon stability on a 16-atom diamond supercell with a

[1] Not in the case of self-consistent GW.

$2 \times 2 \times 2$ **k**-mesh. The same pseudopotential and convergence criteria as above were kept.

We want to note that even this setup and methods may be far from the truth, according to the various studies. For instance, Giustino et al. (2010) suggests that zero-point renormalization of the bandgap for the diamond crystal may be as large as 0.6 eV. We also do not consider the Varshni effect (Varshni, 1967), which suggests that the bandgap for the non-zero temperature is, in fact, smaller. Another significant limitation is connected with the possible defects of the crystal and surface effects, which may alter the result (see Nie et al. (2019)); to address these challenges, one needs to consider larger supercell in the calculation, including vacuum or defects. Last but not least, is a DFT convergence error: we did not use large parameters of energy cutoff or dense **k**-mesh in order to find a balance between calculation time and accuracy. A thorough discussion on the accuracy of *ab initio* approaches is out of scope for this work; however, ML machinery developed here may be applied to more rigorous data as well.

4.2.2 Elastic strain energy density

Elastic strain engineering seeks to identify metastable states of matter for optimizing functional properties and performance. A strained material is in a state of higher energy than when it is in a stress-free state, characterized by the **elastic strain energy density** (ESED) h, which we define as:

$$h(\varepsilon) \equiv \frac{E_{total}(\varepsilon) - E^0_{total}}{V^0}, \tag{4.1}$$

where $E_{total}(\varepsilon)$ is the total energy of the cell deformed by strain ε, and E^0_{total} and V^0 are the total energy and volume of the undeformed cell, respectively. Elastic strain energy density is naturally measured in units of $\mathrm{meV}/\mathrm{\AA}^3$.

Addressing the following question is at the heart of ESE: What is the energy cost (h) to achieve the desired property change? In the stress-free equilibrium state, silicon has a bandgap of 1.1 eV; with an increase in strain energy density, a variety of possible bandgaps emerge. Even silicon with as little strain energy density as 0.2 $\mathrm{meV}/\mathrm{\AA}^3$ can become quite a different material from the stress-free silicon.

Mathematically, we can define the cumulant "density of states" of bandgap (cDOB) as the following:

$$\begin{aligned} c(E'_g; h') \equiv \int_{h(\boldsymbol{\varepsilon})<h'} d^6\boldsymbol{\varepsilon}\, \delta(E'_g - E_g(\boldsymbol{\varepsilon})) = \\ \int d^6\boldsymbol{\varepsilon}\, \delta(E'_g - E_g(\boldsymbol{\varepsilon})) H(h' - h(\boldsymbol{\varepsilon})), \end{aligned} \tag{4.2}$$

where $d^6\boldsymbol{\varepsilon} \equiv d\varepsilon_1 d\varepsilon_2 d\varepsilon_3 d\varepsilon_4 d\varepsilon_5 d\varepsilon_6$ is the measure in the 6D strain-space, $\delta(\cdot)$ is the Dirac delta function, and $H(\cdot)$ is the Heaviside step function. We then define the "density of states" of bandgap (DOB) at h' by taking the derivative

of the cumulant with respect to h':

$$\rho(E_g'; h') \equiv \frac{\partial c(E_g'; h')}{\partial h'} = \int d^6\boldsymbol{\varepsilon}\delta(E_g' - E_g(\boldsymbol{\varepsilon}))\delta(h' - h(\boldsymbol{\varepsilon})). \tag{4.3}$$

The meaning of DOB is as follows: provided one is willing to consider elastically strained states within the $(h - \frac{dh}{2}, h + \frac{dh}{2})$ energy interval, the distribution of bandgaps that these states provide. The DOB function $\rho(E_g; h)$ offers a blueprint to what bandgaps are accessible at what energy cost. One can use the definition (4.3) not only for the electronic bandgap, but also generally for any scalar properties (thermoelectric figure of merit zT, Baliga's figure of merit, Curie temperature, etc.), that will provide an essential road map for ESE, as will be demonstrated in Section 7.1. We can define an upper-envelope function $E_g^{\text{upper}}(h)$ and lower-envelope function $E_g^{\text{lower}}(h)$ based on $\rho(E_g; h)$ also:

$$\begin{aligned} E_g^{\text{upper}}(h) &\equiv \max \operatorname{supp}_{E_g}(\rho(E_g, h)), \\ E_g^{\text{lower}}(h) &\equiv \min \operatorname{supp}_{E_g}(\rho(E_g, h)). \end{aligned} \tag{4.4}$$

In deep ESE, $E_g^{\text{lower}}(h)$ also indicates the 6D steepest descent strain direction to obtain a certain figure of merit. The whole bandgap envelope is analogous to the "flight envelope" used in aerodynamics to describe the allowable Mach number at a given atmospheric density (altitude) for an aircraft. In this work, the shape of the "entire iceberg" is revealed for silicon and diamond, and we can visualize the entire range of possibilities that deep ESE can achieve.

Certainly, for optical applications, another huge question is whether a bandgap is direct or indirect. This direct bandgap envelope will be a subset embedded within the DOB. We can define the density of direct bandgaps (cDOD) in parallel to (4.2), (4.3), (4.4), but with E_{directg} instead of E_g, to obtain cDOD $\rho_d(E_{\text{directg}}, h)$ and its bounds $E_{\text{directg}}^{\text{upper}}(h)$, $E_{\text{directg}}^{\text{lower}}(h)$. Obviously, if direct bandgaps exist at any strain (Corkill and Cohen, 1993; Saladukha et al., 2018; Inaoka et al., 2015), for that strain, there will be

$$(E_{\text{directg}}^{\text{lower}}(h), E_{\text{directg}}^{\text{upper}}(h)) \subseteq (E_g^{\text{lower}}(h), E_g^{\text{upper}}(h)). \tag{4.5}$$

The elastic strain energy density, as well as the other quantities, would be thoroughly analysed on the data available from calculations in the next section.

4.3 Exploratory data analysis of diamond crystal data

Understanding data is a very first and crucial step for solving any machine learning problem. Various visualizations often help to get deep insights into the problem and act correspondingly. In this section, we look into some specific projections of the data given by the first-principles calculations for

a strained diamond crystal case. Although data for silicon crystal is mostly omitted here, we would like to emphasize that the same insights do hold for Si as well.

4.3.1 Stability estimates

In order to first roughly estimate the stability boundaries for the diamond crystal, we have conducted a series of phonon stability experiments (conducted by Zhe Shi). We have sampled $> 10\,000$ strains within the following limits:

$$|\varepsilon_{jj}| < 30\%, j \in \{x, y, z\};$$
$$|\varepsilon_{ij}| < 20\%, i, j \in \{x, y, z\}, i \neq j.$$

A bivariate distribution plot is shown at Figure 4.1. Here, inspired by the results of stability region estimates in 2D materials (Kumar et al., 2015), we used the sums of absolute values for diagonal and off-diagonal elements of the strain tensor as quantities for axes:

$$\Sigma_{diag} = |\varepsilon_{xx}| + |\varepsilon_{yy}| + |\varepsilon_{zz}|;$$
$$\Sigma_{shear} = |\varepsilon_{xy}| + |\varepsilon_{yz}| + |\varepsilon_{xz}|.$$

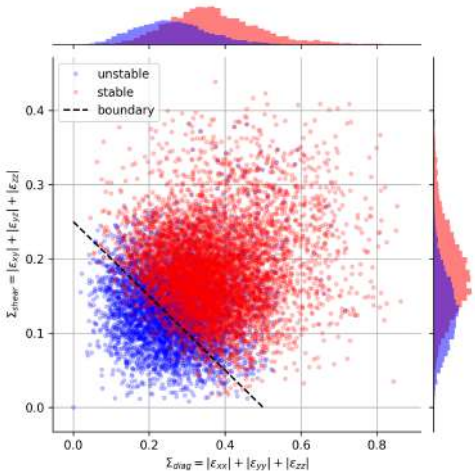


FIGURE 4.1: Bivariate distribution plot for the strains sampled for the stability boundaries estimation. Two classes are not linearly separable; a conservative estimate (4.6) visualized as a dashed line; see Figure 4.2 for the details on this estimate. The stability here is phonon stability accessed from the *ab initio* calculations.

While Figure 4.1 suggests that there is no linearly separable model to distinguish the stable cases from the unstable ones, we have decided to come up with a simple rough estimate in these terms. This is connected to the accuracy limits in the phonon stability calculations: these were done for the Γ critical point only and hence reliably indicate the unstable cases only. We

have done a separate stability estimation for the actual GW calculations we performed later; on the initial stage, we used the following linear estimate:

$$\Sigma_{shear} < 0.25 - 0.5\Sigma_{diag}. \tag{4.6}$$

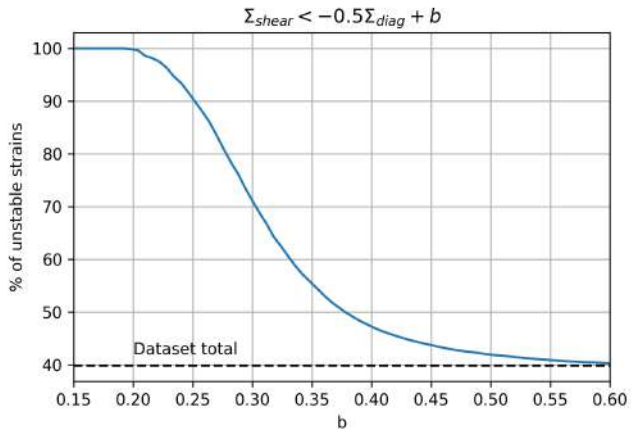


FIGURE 4.2: The percent of unstable strains as a function of the line translation parameter in inequality (4.6). The chosen parameter of $b = 0.25$ corresponds to the 90% confidence estimated on the data set. Regardless of this estimate, all major results were double-checked for stability.

The details on the estimation of the line translation parameter are shown in Figure 4.2; we have taken the value for a 90% confidence estimation of true negative rate (TPR). The slope has been estimated as an orthogonal vector to the line that connects the centers of distributions in Figure 4.1.

4.3.2 Electronic band structure correlations

We want to start by pointing out that there are two kinds of symmetries that hold by construction within the electronic band structure regardless of the strain value:

- **k**-space reflection, or time-reversal symmetry:

$$E_n(\boldsymbol{k}) = E_n(-\boldsymbol{k}), \tag{4.7}$$

  which holds for materials with no strong magnetic properties. This includes all the semiconductors of our possible interest: silicon, germanium, gallium arsenide, and many other materials, including carbon diamond.

- **k**-space periodicity:

$$E_n(\boldsymbol{k} + \boldsymbol{k}^\star) = E_n(\boldsymbol{k}), \tag{4.8}$$

  where $\boldsymbol{k}^\star = (a\boldsymbol{i}, b\boldsymbol{j}, c\boldsymbol{m})$, $a, b, c \in \mathbb{Z}$, $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{m}$ are reciprocal space lattice vectors.

This may be viewed as a periodic boundary in **k**-space.

To demonstrate the intrinsic connections within the electronic band structure data from the GW calculations, we have studied the relationship for the adjacent points in **k**-space. We have considered two metrics: median intraband Pearson correlation:

$$\rho_{\text{intra}}(n,d) = \underset{\substack{j=1,\dots,N,\\ ||\boldsymbol{k_1}-\boldsymbol{k_2}||_{L_1^*}=d}}{\text{med}} \rho(E_n^j(\boldsymbol{k_1}), E_n^j(\boldsymbol{k_2})), \tag{4.9}$$

and median interband Pearson correlation:

$$\rho_{\text{inter}}(n_1,n_2) = \underset{j=1,\dots,N}{\text{med}} \rho(E_{n_1}^j(\boldsymbol{k_1}), E_{n_2}^j(\boldsymbol{k_2})), \tag{4.10}$$

where

- $\rho(\cdot,\cdot)$ is a Pearson correlation coefficient between the two samples. It measures the power of a linear relationship.
- med is a sample median. We take median instead of average for a more robust estimate.
- $||\cdot||_{L_1^*}$ is a Manhattan distance. It seems to have a straightforward distance choice in the **k**-space given the discrete nature of **k**-space grid. The asterisk here shows that the **k**-space periodicity (4.8) is taken into account.
- $E_n^j(\boldsymbol{k})$ is a dispersion relation for electrons of n^{th} band at the wavevector **k**, defined in Section 2.2.2.
- N is a number of samples.

Each of the energy bands evolves piecewise-smoothly with changes in **k**, and the information within the energy dispersion of a specific band includes intraband correlations with respect to **k**. In other words, we do expect adjacent points from the same band of **k**-space to correlate in terms of dispersion relation energies $E_n(\boldsymbol{k})$, just like intensities of adjacent pixels for a gray-scale image do correlate with each other. Figure 4.3 shows that for most of the bands, the dependence is really close to linear, with the conduction and upper bands ($n \geq n_{CB}$) showing a stronger relationship.

As suggested by solid-state physics, the energy bands are not "independent" from each other. Instead, they collectively describe the physical nature of the crystal. For example, if we consider single electron in a periodic potential resulting from the interaction of the electron with the ions and other electrons, by solving the Schrödinger equation, we get a series of Bloch waves as solutions (see Section 2.2.2), each of which has a predicted dispersive form. Through the first-principles method, the quantized energy levels are determined altogether at once, meaning that the n^{th} band is not calculated alone but is affected and determined by its neighbouring bands, including the adjacent $(n-1)^{\text{th}}$ and $(n+1)^{\text{th}}$ bands as well as other non-adjacent bands

FIGURE 4.3: Heatmap for the median intraband Pearson correlation coefficient (4.9) between the energies in two points separated by a given Manhattan distance in **k**-space. Unsurprisingly, the correlation is strongest in the case of two adjacent **k**-points.

(Perdew et al., 1996). In other words, there is information regarding the interband correlations against n that is included in the band structure of a crystal. As the exploratory analysis of the median interband correlation (Figure 4.4) suggests, this connection is strong and evident in the case of the valence and higher bands ($n \geq n_{VB}$).

This again resembles the image structure, where different color channels are not independent of each other. To reveal and preserve the internal formation of the band structure data in the means of our model, we incorporate into our ML scheme convolutional structure to carry out the band structure prediction task. The CNN is known for its ability to extract hierarchical patterns in images and assembling more complex patterns out of smaller ones, e.g., the "jittered-clustered" image dataset NORB (Ciresan et al., 2011; Ciregan et al., 2012). Due to the pictorial nature of the band structure, CNN can be expected as a good tool in extracting and exploiting useful "image patterns", or band correlations, in our task.

4.3.3 Bandgap and band extrema

PBE vs GW bandgap

We start the exploration of one of the most significant electronic properties we explore – the bandgap E_g. The relationship is shown in Figure 4.5. As an *ad hoc* solution, one could come up with a linear estimate of the GW bandgap given the (much computationally cheaper) PBE one:

$$E_g^{GW} \approx 1.178 E_g^{PBE} + 0.982. \tag{4.11}$$

FIGURE 4.4: Median interband Pearson correlation coefficient (4.10) between the energies in the same points in **k**-space but different bands. The correlation is strong in the case of adjacent bands: top VB ($n = n_{VB}$), lowest CB ($n = n_{CB}$) and its adjacent band ($n = n_{CB+1}$).

The relation (4.11) demonstrates that the so-called "scissors-cut", or constant shift estimate that connects both methods, is not very far from the truth. However, we can also see that for a specific region of $E_g^{PBE} < 0$, the relationship loses its linearity. For a more accurate estimate, one should adopt a more advanced ML model.

Another important reason for the data visualization is that one could detect the anomalies in the data. We have essentially improved our data sampling and processing pipeline with a similar visualization, avoiding the "garbage in – garbage out" scenario.

Conduction band minima

Within the strain space we sampled from, the valence band maxima is always located at the Γ point. The situation is different for the conduction band minima (CBM), for which we have discovered the following approximate distribution:

- 'Δ'-line: 75% of cases. This is also a CBM for the unstrained crystal.
- Γ **k**-point: 14% of cases. The similar rate for a direct bandgap for silicon was $< 1\%$.
- 'X' **k**-point: 11% of cases.

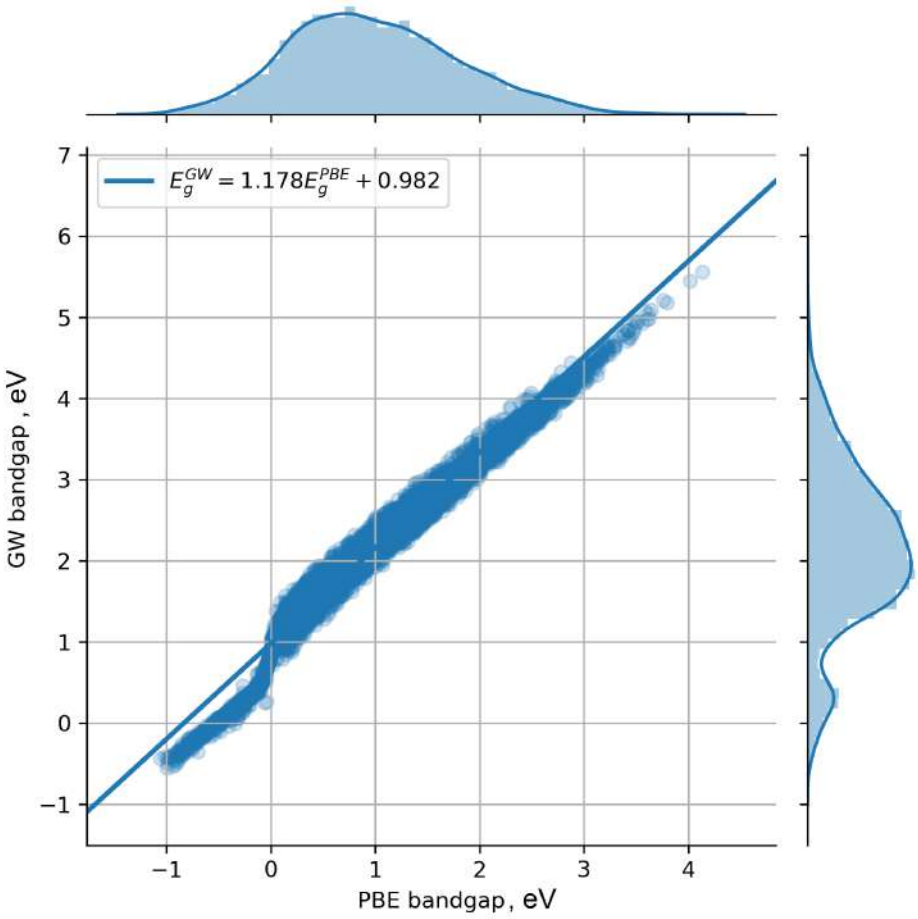


FIGURE 4.5: PBE vs GW bandgap along with the histograms for both. The relationship is very similar to the linear for the most of the population.

Direct bandgap

The ability of elastic strain engineering to turn the indirect bandgap of an unstained diamond crystal into the direct one could be exploited in a number of ways, possibly leading to the new optoelectronic devices. The direct bandgap is not of a rare occasion in terms of the 6-dimensional strain space, and we will fully explore it not by mere sampling from the strains but using the power of the surrogate model we design. Nevertheless, we would like to explore some statistics of the direct bandgap data subset.

It is of interest to explore the regions in the strain space that correspond to the direct bandgap. Unfortunately, the direct visualization of the strain space is difficult, and we start by introducing the pairwise projections on the strain components axes in Figure 4.6. This plot indicates that the typical requirement for a direct bandgap to appear involves the strain cases with the shear (off-diagonal) components being far from zero: $|\varepsilon_{xy}|, |\varepsilon_{yz}|, |\varepsilon_{xz}| \gg 0$.

Figure 4.6 also indicates that it is hard to locate the exact regions of the direct bandgap. tSNE also indicates that the cases are indistinguishable from the general sample. We can try to answer the following question: is the subset

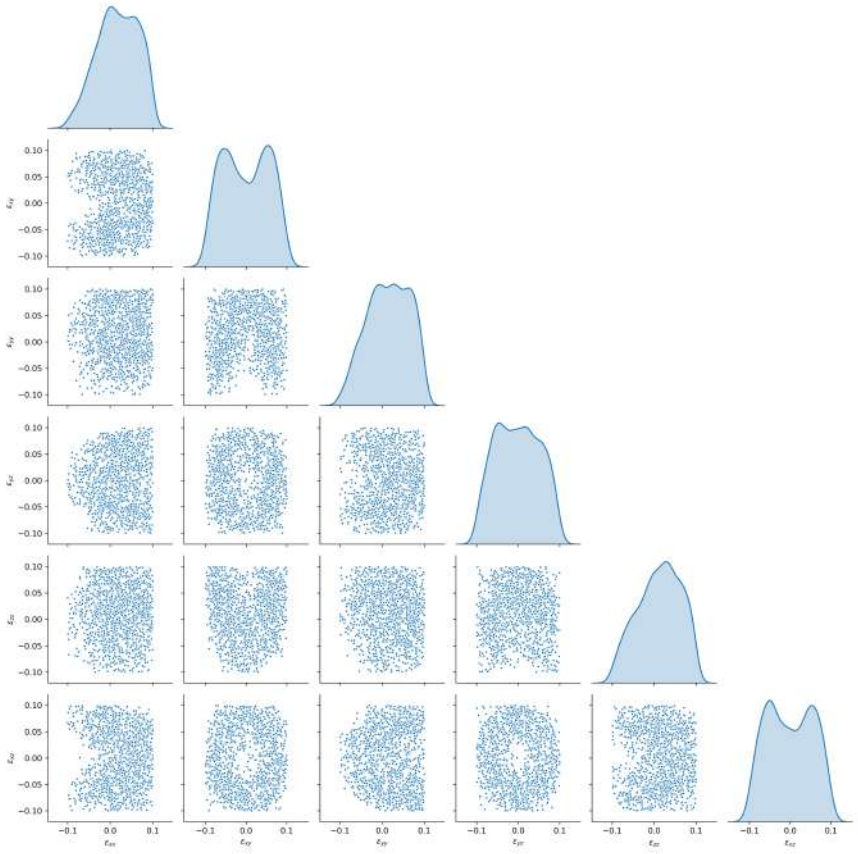


FIGURE 4.6: Pairwise scatter plot for direct bandgap cases. Pairwise plots for $\varepsilon_{xy}, \varepsilon_{yz}, \varepsilon_{xz}$ suggest that these strain components are unlikely to hit zero simultaneously in the direct bandgap case. At the end of each row, density plots, or smoothed histograms, are provided for the strain components.

of direct bandgap strains contained within the manifold of a smaller dimension? And the answer is yes.

One of the standard techniques in the *intrinsic dimension* estimation is the maximum likelihood estimation (MLE) applied to the distances between close neighbours (Levina and Bickel, 2005). Since it is stochastic by its nature, we conducted several runs of intrinsic dimension MLE for the whole sample, random samples, and direct bandgap subset. The results are shown in Figure 4.7. It illustrates that the direct bandgap strains are likely to form a 5-dimensional manifold within the 6-dimensional manifold of admissible strains.

4.3.4 Strain-bandstructure symmetries

For the bandgap as an electronic bandstructure property, the symmetry group of the crystal permutations consists of 48 elements. However, due to the usage of the specific strain tensor (see Section 2.1.3 for details) the corresponding symmetries are not held anymore. In this section, we will explore some symmetry cases and their possible utilization.

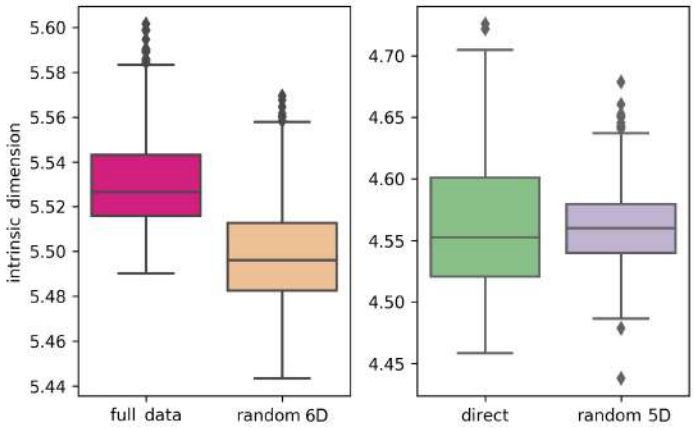


FIGURE 4.7: Intrinsic dimension MLE for the strain data: full data set, random points in 6-dimensional space, direct bandgap strains, and random points in 5-dimensional space. Intrinsic dimension for a direct bandgaps is less than one for the general sample.

Let us consider four arbitrary strain cases:

$$\begin{aligned}
\varepsilon_A &= (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{yz}, \varepsilon_{xz}, \varepsilon_{xy});\\
\varepsilon_B &= (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{yz}, -\varepsilon_{xz}, -\varepsilon_{xy});\\
\varepsilon_C &= (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, -\varepsilon_{yz}, \varepsilon_{xz}, -\varepsilon_{xy});\\
\varepsilon_D &= (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, -\varepsilon_{yz}, -\varepsilon_{xz}, \varepsilon_{xy}).
\end{aligned}$$

For these, there is a complicated one-to-one correspondence between the dispersion energies in k-points of the corresponding electronic band structures we found empirically:

$$\begin{aligned}
E_n(k_1^B, k_2^B, k_3^B) &= E_n(k_1^A, k_1^A - k_3^A, k_1^A - k_2^A);\\
E_n(k_1^C, k_2^C, k_3^C) &= E_n(k_2^A - k_3^A, k_2^A, k_2^A - k_1^A);\\
E_n(k_1^D, k_2^D, k_3^D) &= E_n(k_3^A - k_2^A, k_3^A - k_1^A, k_3^A).
\end{aligned} \tag{4.12}$$

The mean absolute deviation for these is near 7e-5 eV, which looks like a rounding error of VASP routines since we used 4 decimal digits after a comma in the standard OUTCAR output file.

This equivalence matters a lot since we now have 4 times more data without the corresponding loss in accuracy. The scheme is close to the data augmentation, which is widely used for the CNN training.

These findings conclude the exploratory analysis section. In the next section, we will show how the explored properties influenced the model design.

5 Methodology development

5.1 Model design

By that point, when the preliminary analysis is done, we can finally raise some realistic expectations about our surrogate model and discuss it in detail:

- **Band structure prediction.** Instead of focusing on the single property (like the bandgap E_g or its type), we aim to gather the full electronic band structure information at once due to a number of reasons. First, Section 4.3 have demonstrated that the data is intensively interconnected, and exploiting these connections may be beneficial in terms of accuracy of the whole procedure. Second, obtaining a full bandstructure at once offers a more comprehensive description of what is going on in terms of effects caused by the band structure change, and is required for a proper bandgap estimation. For example, with a band structure at hand, one could estimate how stable is the direct bandgap in terms of other minima in **k**-space being close. Another example is a possible prediction of a electronic mass tensor (2.11), which requires the position of conduction band minima and the values in several **k**-points. Moreover, with the help of other additional calculations, it offers the prediction of other properties and figures-of-merit (FoMs), such as the absorption coefficient. Nevertheless, we would compare the performance of the "general" model, which predicts the whole band structure, with a "specialized" model focused on a single property, in experimental Section 6.1.2.

  In this work, we are focusing on the approach that fixes a **k**-point mesh and predicts the energy dispersion relations on its nodes based on the six-dimensional strain $\boldsymbol{\varepsilon}$. Another possible solution could be the model that is designed to take both strain and **k**-point coordinates as input and predict the energy relation for a given band. This model was designed for the case of PBE data, and the accuracy results are reported in Section 6.1.1. However, switching to the next accuracy level with the GW calculations is hard due to computational constraints and specific requirements to the **k**-mesh generation – one just be lacking data to train the model on.

- **Warm start (incremental fitting).** We definitely want our surrogate models to learn successfully from several data sets and assimilate them. This capability is becoming increasingly important with the spread of materials property databases that collect data from different studies

(Jain et al., 2013). Luckily, most of the modern models may theoretically be used in this fashion, as discussed in Section 2.3.5. The ability of models to handle the warm start in practice is accessed in Section 6.1.1; in particular, we are looking forward to the utilization of the vast PBE data we can generate for cheap.

- **Active learning.** As mentioned in Section 2.3.5, we would like to take advantage of the access to the oracle (i.e., VASP) that could calculate the electronic band structure for a given strain value $\boldsymbol{\varepsilon}$. To this extent, we want to access the uncertainty estimates and use them as an acquisition function.

- **High accuracy and fast inference.** Last, but not the least, are the performance requirements our model should meet. Of course, one could use rough linear-like estimates similar to (4.6) for a fast approximation, yet for a number of applications, the required accuracy is not enough. The situation gets worse for a small bandgap materials, such as silicon crystal: the mean absolute error of 0.1 eV is 1-2 % relative error for the diamond crystal yet close to 6-7 % for the silicon crystal and could reach 10-15 % for the narrow-bandgap germanium.

  The high-throughput requirement is connected with an exploration of the vast six-dimensional strain space. While optimization routines may require tens of thousands of calculations, the approaches aimed at the full description may require millions or even billions of runs. To that end, we want our model to be fast in terms of the inference time, and this is also the main reason we do not learn separate models for the energy dispersion relation for each band and each $\mathbf{k}$-point, as this increases the inference time by the three orders of magnitude (4 (energy bands) $\times$ 260 ($\mathbf{k}$-points for 8^3 mesh, excluding time-reversal symmetry (4.7))).

We could think of two families of models that meet these requirements:

- **Gaussian Processes and kernel-based methods.** The high accuracy and uncertainty estimation are natural to these models, and they could be generalized to work on multi-output fashion. Unfortunately, setting up such a model requires intensive kernel search, and fast inference coupled with the large output dimension is challenging, as well as the active learning on large amounts of data.

- **Neural networks**. We have ended up using these models due to the following reasons. Most of the properties are natural to these models, and there exist specialized architectures. Moreover, on the implementation side, deep learning libraries, such as Tensorflow Abadi et al. (2016) or Torch Paszke et al. (2019) are pretty mature and popular, which means extensive support may be found online. The situation with the Gaussian processes is not that bright; it is connected with the lesser popularity of this approach in general.

One of the main drawbacks of neural networks are still-developing techniques of uncertainty estimation. To overcome this difficulty, we do propose various methods of uncertainty estimation for the neural networks in general, and evaluate them within the course of the ESE project and for standard ML-related problems and benchmarks as well in Chapter 6.

We would like to point out that our understanding of approaches suitable for the elastic strain engineering has changed along with the research course. For a silicon crystal, we took advantage of simple models, probing the surrogate model landscape that is described in the next Section 5.1.1 in detail and in Shi et al. (2019) in short. A more thorough analysis was performed for the diamond crystal, and this resulted in a fundamental redesign of the neural network architecture on top of the previous method to incorporate more physics at the very beginning. Based on an improved understanding, we further proposed a better data representation scheme and a more advanced algorithm based upon convolutional neural network (CNN) structure capable of bringing about significantly improved ML outcomes that invoke physical insights, see Section 5.1.2. The fast and reliable inference of the proposed model opens a path towards analyzing and scrutinizing the unexplored corners of the vast 6D strain space and find the most energy-efficient ways for FoM optimization by ESE.

5.1.1 NN description for Si crystal

Neural networks were implemented within the Tensorflow (Abadi et al., 2016) framework. To predict the bandgap for a silicon crystal, we used architecture with four hidden layers with a (64 – 128 – 256 – 256) in the case of three-normal-strains strains ($\varepsilon_{xy} = \varepsilon_{yz} = \varepsilon_{xz}$) and a (512 – 256 – 256 – 256) architecture for the general case with shear strains, as shown in Figure 5.1. For the more complicated task of band energy prediction at a single **k**-point, the (512 – 256 – 256 – 256) architecture was used. The leaky rectified linear unit (Maas et al., 2013) was chosen as an activation function. We used the Adam stochastic optimization method (Kingma and Ba, 2014), the orthogonal weight initialization (Saxe et al., 2013), and the dropout (Srivastava et al., 2014) technique to prevent overfitting.

Training procedure: data fusion

Data fusion represents the concept of combining different data sources in order to improve the model (Khaleghi et al., 2013). We adopted this approach to further improve the learning outcome for E_g, the most technically important property for electronic material. While the data fusion model prediction in Ramakrishnan et al. (2015) corresponds to a baseline value plus a correction, our data fusion approach is more advanced. More specifically, given E_g^{PBE} computed using an approximate baseline level of theory (PBE) at a particular query strain case, a related E_g^{PBE} value corresponding to a more accurate and more demanding target level of theory (GW) can be estimated

as a function of both E_g^{PBE} and elastic strain $\boldsymbol{\varepsilon}$. Therefore, the E_g^{GW} consistent with the query strain case is learned using exclusively ε and E_g^{PBE} as input, as illustrated in Figure 5.1.

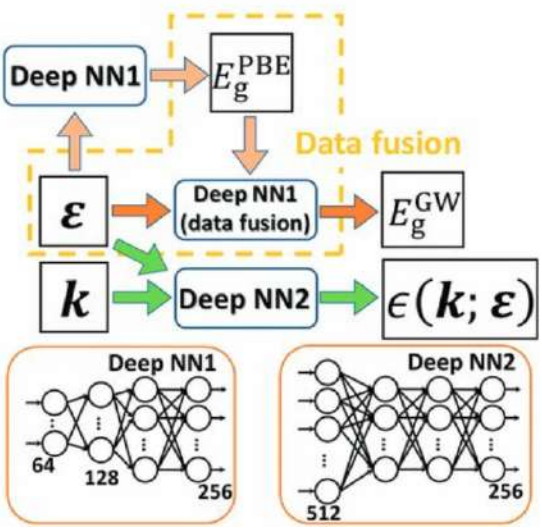


FIGURE 5.1: Machine learning workflow with a fully-connected neural networks. For a typical bandgap-prediction task, the input contains the strain information only, and the target is either E_g^{PBE} or E_g^{GW}. In the data fusion process, the bandgap predicted from fitting the PBE dataset is also taken in as an input to fit the GW bandgap. For the whole band structure fitting task, the input contains both strain information and the **k**-point coordinates, and the target is the energy dispersion $E_n(\boldsymbol{k}, \varepsilon)$, where n is the band index, **k** is the wavevector, and ε is the crystal strain tensor. The hidden-layer structures of the two associated deep NNs are also depicted.

5.1.2 CNN description

We are focused on the accurate prediction of the electronic band structure, from which most of the FoMs may be directly derived. In the search of a concept that better represents the nature of energy dispersion as an object, we, inspired by the tremendous success of deep learning in the field of image processing LeCun et al. (2015), found the desired analogy in an idea of image. Indeed, the "pictorial" view, with concepts like RGB color channels that do correlate with each other, and the pixel-like discretization as the tool to perceive the continuous nature of the phenomena, imparts an important knowledge of the true essence of the band structure. Now we can rightly regard the energy dispersions as some sort of stacked 3D “images”, with the reciprocal coordinates $\boldsymbol{k} = (k_1, k_2, k_3)$ representing the “voxels” (i.e., 3D “pixels”) of an image and E_n denoting the “color-scale” (similar to the RGB or grayscale of a real image) at each voxel for a particular 3D image, n (band

index). As we demonstrated in Section 4.3.2, energy bands evolve piecewise-smoothly with changes in **k**, and the information within the energy dispersion of a specific band includes intraband correlations with respect to **k**. An illustration of this pictorial nature of band structure can be found in Figure 5.2.

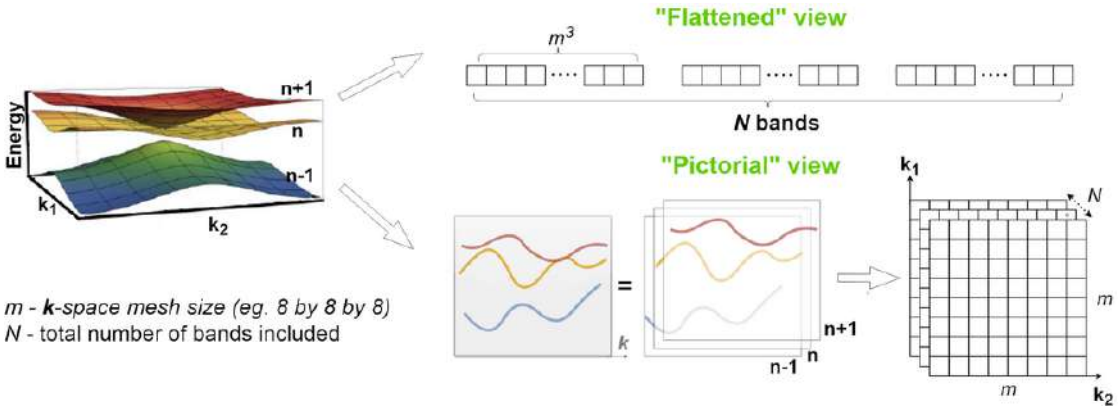


FIGURE 5.2: Different representations of a band structure. In the "flattened" view, a band structure is represented as N stacked flattened arrays (vectors) and processed like independent values. Each array is m^3 in length. In the "pictorial" view, the band structure is considered as N 3D images stacked together, each of which has a "voxel" dimension of $m \times m \times m$. The eigenvalues on an energy band can be thought of as the "color-scale" of the voxels. The bands are shown in 2D for visualization purposes.

It is noted that the baseline here is the simpler and straightforward ML scheme, which is based on feed-forward neural network that treats an energy band as a flattened array of independent values, will ignore the information of intraband correlation (shown in Figure 4.3). Figure 5.3 summarizes the key differences between the baseline approach and the proposed CNN.

As was highlighted earlier in Section 4.3.2, we rely on the convolutional neural networks as an essential piece of our model. The general setup of the proposed CNN-based model is illustrated in Figure 5.4.

It consists of two consecutive blocks: the fully-connected part and the convolutional part. In the beginning, the strain tensor $\boldsymbol{\varepsilon}$ is taken as the input and transformed into a feature vector through a series of fully-connected layers, as depicted in Figure 5.4. This feature vector has a length of Nm^3, where m^3 equals the number of **k**-points sampled in the Brillouin zone, and N is the number of bands we aim to approximate altogether. Depending on the **k**-mesh density, the feature vector can be made very long and adopted as a rich representation of the intraband information for a band structure. For instance, if we aim to use a total of four bands ($N = 4$) to describe the energy dispersion near the bandgap under a particular deformation, we could come up with 4 vectors, each of which has a length of m^3, to represent, respectively, the band energy dispersion for the top valence band ($VB, n = n_{VB}$), the lowest conduction band ($CB, n = n_{CB}$), and their adjacent two bands ($n = n_{VB-1}$

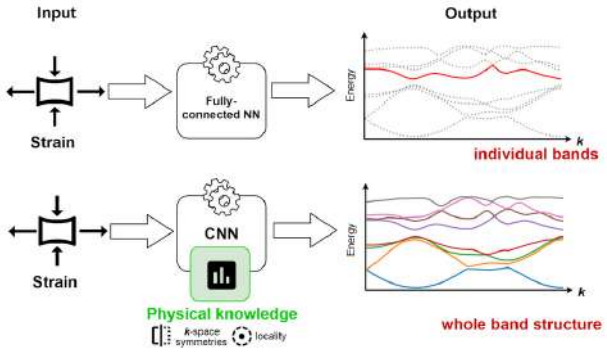


FIGURE 5.3: Comparison of the two different approaches to machine learning the ESE. One can predict the eigenvalues on an energy band independently by directly utilizing the "flattened" band structure representation, or take into account the inner relations between the values provided by the physical nature of the data and result in getting the entire band structure at once.

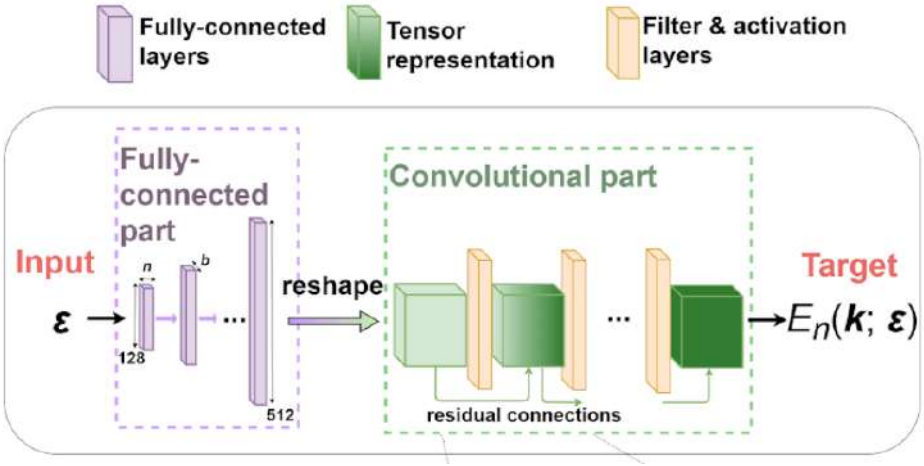


FIGURE 5.4: CNN architecture for the band structure prediction. The strain components are passed through a few fully-connected layers, with the last layer reshaped into a rank-5 tensor. After a few convolutional layers (with residual connections He et al. (2016) that improve convergence by avoiding the gradient vanishing), the network produces the band structure as the output, which is fitted against the targeted DFT-computed band structure. Here, b is a batch size, and n is a number of bands.

and $n = n_{CB+1}$). Stacking them together, we can build an $m \times m \times m \times N$ tensor representation of the band structure for any individual strain data, as illustrated in Figure 5.5. This is then fed into the next block of convolutions.

The convolutional block consists of several layers that update this tensor representation until the final output – the band structure. We expect the tensor representations closer to the final layer are more relevant to the output,

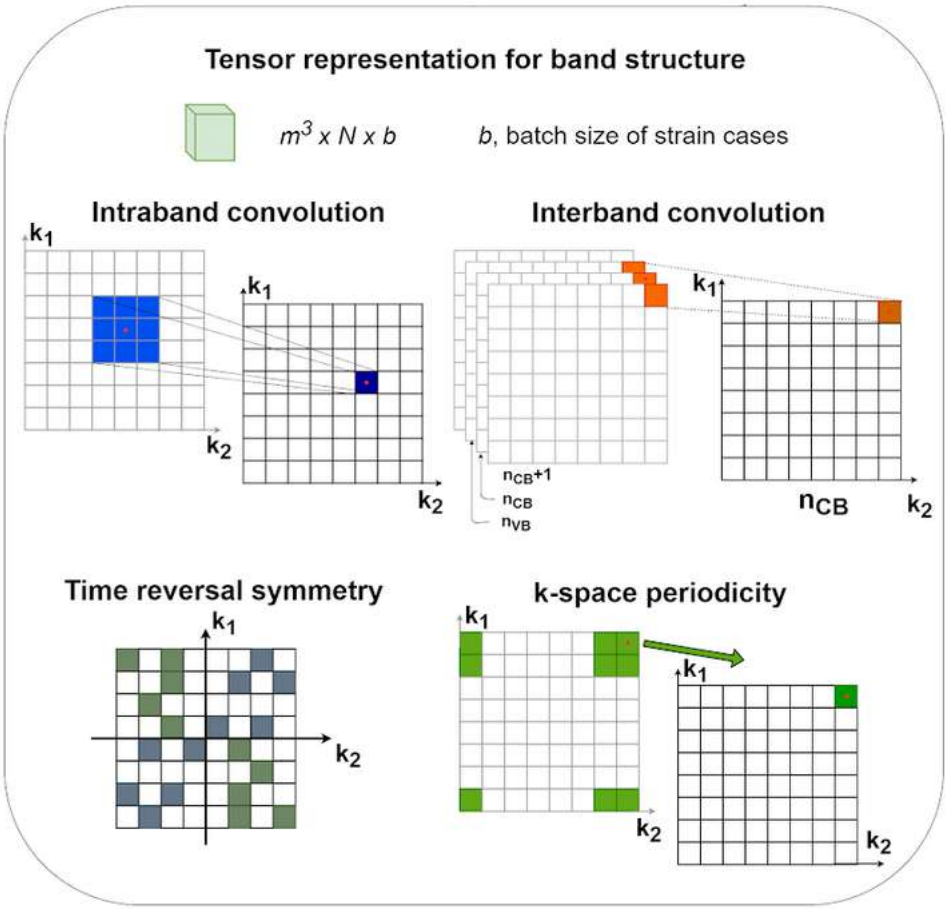


FIGURE 5.5: Tensor representation and physical insights incorporated into the CNN model: time-reversal symmetry, **k**-space periodicity, and inter- and intraband convolutions.

as in the case of the image processing models. Note that the output tensor retains the same dimension of the band structure, i.e., $m \times m \times m \times N$. This extraction process can proceed for many layers and a band structure tensor full of detailed features involving deep intra- and inter-band information will be delivered to fit the eventual target, which is the true band structure obtained by DFT calculations, and thus complete the ML inference. The power of our model lies in the architecture of the proposed CNN model, which is tailored to the known physical structure and exploratory data analysis results (see Section 4.3.2 for details) in order to simplify the training and speed up the inference. In particular, it takes advantage of:

- **The time-reversal symmetry** (4.7), which holds for the diamond crystal. Corresponding tensor representation holds this property.
- The correlation between energy dispersion relations at the same **k**-point of different bands (**interband correlation**, see Figure 4.4). An interband convolution between the bands is applied to each **k**-point so bands could affect each other.

- The correlation between the energy eigenvalues associated with adjacent **k**-points of the same band (**intraband correlation**, see Figure 4.3)), which represents the fact that the band energy is a piecewise-smooth function of the **k**-space coordinates. We run intraband convolutions for several cycles (computational blocks) so that the underlying physics of how energy eigenvalues from adjacent **k**-points affect each other can be well-learned.

- Band structure calculations take advantage of the periodic nature of a crystal lattice (4.8), exploiting its symmetry. The band structure plot resulting from restricting **k** to the first Brillouin zone, also known as the reduced zone scheme, is used most typically by physicists. This reciprocal lattice periodicity is represented in our model using a **special padding technique for periodic boundary condition** that follows the reduced zone scheme.

Training procedure

The training of our model consists of three parts: preliminary training, data fusion, and active learning. In the first part, we trained our model on the large dataset ($\approx$ 35 000 strain values) of computationally inexpensive PBE calculations[1]. After a reasonable accuracy was achieved, in the second part, we performed training on the much smaller amount ($\approx$ 6 000 samples) of the accurate GW data (sampled according to the Latin Hypercube Sampling (McKay et al., 1979)), starting from the NN parameters we learned in a previous stage. This approach is known as the *knowledge transfer* as some of the knowledge that the NN has gathered from the low-fidelity PBE data is exploited to ease the training on the costly yet reliable GW data, see Figure 5.6 for a schematic representation of this process. The third integral part of the training is the active learning procedure. In our case, we used the dropout-based uncertainty estimation to sample the most "uncertain" strain cases for further improvement of the model. Specifically, after the first round of the training on the GW data, we performed an estimation over a large set of random strains in 6D and chose a small amount of 200 strain cases with the largest expected error as evaluated by this temporary model (uncertainty measurement). These strain cases were added to the training set for the next round of training, as illustrated in Figure 5.6. Our study indicates that 5-10 cycles of the above active learning may enable the trained CNN to reach the same level of accuracy with twice or three times less additional data, thus considerably reducing the total amount of ab initio calculations without compromising the robustness of our ML model, see 6.1.2.

[1] We used all the data calculated on the exploratory analysis part of the research; in practice, the model may be pre-trained on a much smaller amount of data ($\approx$ 5 000 strains).

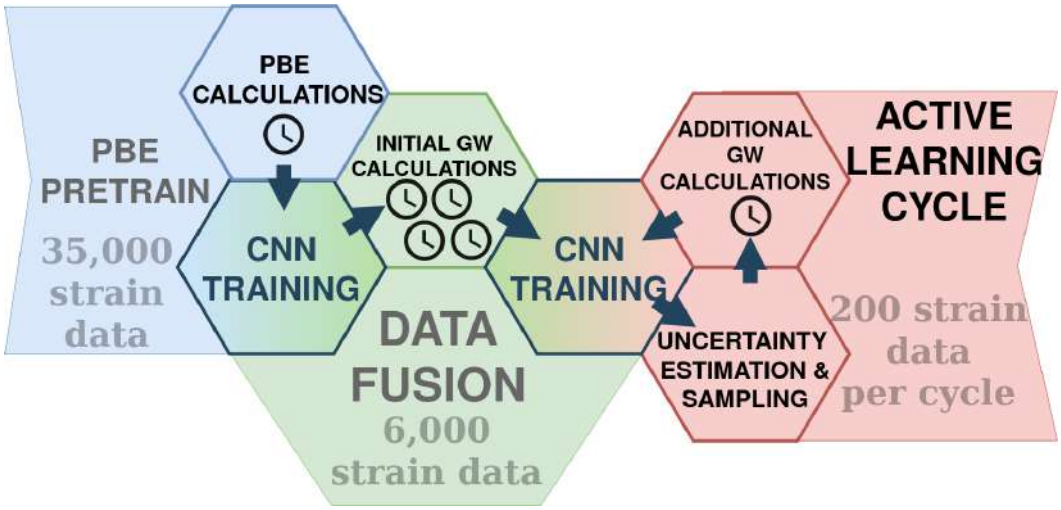


FIGURE 5.6: The whole ML scheme involving pre-train, data fusion, and active learning. The solid arrows show the workflow, and clock symbols indicate the time (in log scale) required for *ab initio* calculations. This framework is designed to benefit from both "oracle" existence and low-fidelity data by introducing the active learning and pre-training parts, correspondingly.

5.2 Uncertainty estimation and active learning in neural networks

This section is dedicated to the technical details on the uncertainty estimation for the neural networks. We will step off a bit from the elastic strain engineering and will focus on the more general case of the fully-connected neural networks. While generally feasible methods were described in brief in Sections 2.3.4 and 2.3.5, here we will focus on the dropout-based approaches and its derivatives as an original content produced in search of both low-effort and reliable uncertainty estimates for the neural networks. We will begin with a description of the basic model and then will turn into possible and tested improvements.

5.2.1 Basic dropout-based UE and active learning

Using dropout at the prediction stage allows us to generate stochastic predictions, and, consequently, to estimate the variance of these predictions. Our approach is based on the hypothesis that data samples with higher standard deviations have larger errors of true function predictions. Although this is not always the case, concerning a neural network of a reasonable size trained on a reasonable number of samples, we can possibly observe a correlation between dropout-based variance estimates and prediction errors. It should be noted that the result does vary (like any other result of neural network training) depending on several factors: architecture and size of the neural network, samples used for initial training, and hyperparameters, such as regularization, learning rate, and dropout probability. Following the notation

introduced in Sections 2.3.4 and 2.3.5, we propose an MCDUE (Monte-Carlo Dropout Uncertainty Estimate)-based active learning algorithm as follows:

1. **Initialization**. Choose a trained neural network $\hat{f}(x) = \hat{f}(x, \omega)$, where ω is a vector of weights. Set the dropout probability π. Set the number of stochastic runs T.

2. **Variance estimation**. For each sample x_j from the pool $\mathcal{P}$:

   (a) Make T stochastic runs of the model $\hat{f}$ using dropout and collect outputs $y_k = \hat{f}_k(x_j) = \hat{f}(x_j, \omega_k), k = 1, \ldots, T$, where ω_k are sampled from Bernoulli distribution with parameter π.

   (b) Calculate the standard deviation (as an acquisition function):

$$s_j = A^{\text{MCDUE}}(x_j) = \sqrt{\frac{1}{T-1}\sum_{k=1}^{T}(y_k - \bar{y})^2}, \; \bar{y} = \frac{1}{T}\sum_{k=1}^{T} y_k.$$

3. **Sampling**. Pick m samples with the largest standard deviations s_j.

Step 2 here essentially obtains the uncertainty estimate. In literature, this estimate is often referred to as MC Dropout, or MCD, and used as a baseline. The computational cost per sample is $O(TN_{\text{pool}})$. However, on modern GPU-based implementations, the sampling can be done in parallel. One could also decrease T to speed up the procedure. It should be emphasized that we can start with a pre-trained neural network from the previous iteration if we train the model on the extended training set; such a method may significantly speed-up retraining.

We would like to note that the basic dropout-based UE and the corresponding active learning procedure have several drawbacks, including biased estimation, greedy sampling algorithm, and loose theoretical guarantees, which will be discussed in detail in the next sections. In order to address these drawbacks, two approaches are proposed: GP-based enhancing of a trained NN, described in Section 5.2.3, and dropout mask diversification, introduced in the next section.

5.2.2 Diversified dropout

Another view on the dropout-powered uncertainty estimation is to treat a neural network with dropout as an implicit ensemble of models. To further strengthen it, we propose to sample the most diverse models from it to improve the uncertainty estimation. As a particular realization of the general idea, we suggest sampling masks using the machinery of determinantal point processes (DPP) (Macchi, 1975; Kulesza et al., 2012), which are known to give diverse samples. Inference with dropout for single models can be combined with ensembles to increase the quality of uncertainty estimation even further.

This methodology was developed in collaboration with Maxim Panov and Kirill Fedyanin.

Dropout as a set of random masks

An application of dropout to neurons results in the following formula for the output:

$$O_i^h = \sigma\left(\sum_{j=1}^{N_{h-1}} \omega_{ij}^h m_j^h O_j^{h-1}\right), \quad i = 1, \ldots, N_h,$$

where

- ω^h are a set of weights on the h-th layer of the neural network;
- O_i^h is an output of the h-th layer of the neural network;
- $\sigma(\cdot)$ is a non-linear transformation (activation function);
- m_j^h are Bernoulli random variables with the probability of 0 equal to p.

We omitted the bias term here just to simplify the representation. Note that if an input variable of the neural network is denoted by x, then the output of every layer is a function of x, i.e. $O_i^h = O_i^h(x)$.

Let us denote the vector of dropout weights m_j^h for the h-th layer by $m_h = (m_1^h, \ldots, m_{N_h}^h)^{\mathrm{T}}$ and the full set of dropout weights by $M = (m_1, \ldots, m_K)$.

Thus, any neural network $\hat{f}(x)$ with dropout layers essentially has 2 sets of parameters: the full set of learnable weights W and the set of dropout weights M:

$$\hat{f}(x) = \hat{f}(x \mid W, M).$$

Let us have a neural network with dropout, which was trained on some dataset giving weight estimates $\hat{W}$. Then the dropout weights M remain the free parameters and require selection at the time of inference:

$$\hat{f}(x \mid M) = \hat{f}(x \mid \hat{W}, M).$$

The originally proposed by Hinton et al. (2012) and currently the standard choice is to take

$$\hat{M} = \frac{1}{1-p} I,$$

where p is the dropout rate used during training and I is the identity matrix of the corresponding shape. Such an approach gives the fixed function $\hat{f}(x \mid \hat{M})$, which is known to give reasonably good performance despite being a heuristic choice.

The MC Dropout may be inferred if one considers Bernoulli distribution as a prior over elements of a matrix M (Gal and Ghahramani, 2016; Nalisnick et al., 2019), which results in a Bayesian model:

$$\hat{f}(x) = \hat{f}(x \mid M), \quad M \sim Bernoulli(1-p).$$

Within this approach, one can sample some number T of i.i.d. realizations $M_1, \ldots, M_T$ from the prior distribution and compute approximate posterior

mean

$$\bar{f}_T(x) = \frac{1}{T}\sum_{i=1}^{T}\hat{f}(x \mid M_i)$$

and variance

$$\bar{\sigma}_T^2(x) = \frac{1}{T}\sum_{i=1}^{T}\big(\hat{f}(x \mid M_i) - \bar{f}_T(x)\big)^2, \tag{5.1}$$

with the last equation being nothing less than Monte-Carlo Dropout uncertainty estimate (MCDUE) we introduced above.

We suggest a different approach, namely we treat $\hat{f}(x \mid M)$ as an ensemble of models indexed by dropout masks M. Such a view allows us to decouple inference from training and pose an intuitive question: what set of masks $M_1, \ldots, M_T$ should one choose in order to obtain the best uncertainty estimate $\bar{\sigma}_T^2(x)$? Importantly, here we do not limit the selection of masks to be samples from standard dropout distribution, which in principle should allow to obtain better estimates. However, the design of the mask selection procedure is a non-trivial problem, which we discuss below in detail.

The first incentive is to directly sample the masks as the most diverse set of vectors in the mask space, and it may seem that we can do this with approaches like Hamming code (Hamming, 1950) or Latin Hypercube Sampling (McKay et al., 1979). One of the main challenges that stop us from doing so is the curse of dimensionality: the random set of such masks will be really close to the optimal set in terms of the average (maximal) distance between the masks. Therefore, we do not expect these data-agnostic approaches to strictly beat the random sampling since many neurons in the network are highly correlated in practice. For example, consider a correlation matrix of neurons in a linear layer of a convolutional neural network, trained on CIFAR-10 (see Figure 5.7). The correlation was computed on 10 000 samples and clearly shows groups of highly correlated neurons. It suggests that sampling masks for such a layer completely at random might lead to many highly correlated neurons in the sample. This property is non-desirable as it leads to more correlated models in the ensemble. Therefore are focusing on the data-based approaches instead, since fighting the correlation between the neurons is best done by first measuring it.

Another piece of motivation comes from the fact that sampling masks for the layer like shown in Figure 5.7 uniformly at random might result in high variance of pre-activations. As a result, the estimates for the whole network may require a significant number of samples (stochastic passes through the NN) T to converge. We illustrate this behaviour in Figure 5.8, where we observe that several hundreds of simple MC dropout estimates are required for the convergence of the log-likelihood values. It is clearly seen that a larger number of samples improves the values of log-likelihood, yet may impose computational cost too large to be used in real-world applications due to computational restrictions. However, one may expect that the knowledge about the correlations between neurons can help to sample more diverse neurons and improve the estimates.

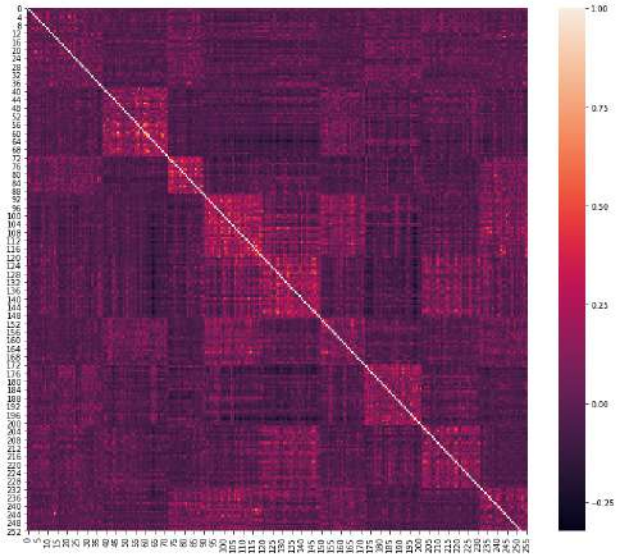


FIGURE 5.7: Correlation matrix C between the outputs of the neurons in a linear layer of the convolutional NN, trained on the CIFAR-10 dataset. Correlations are computed based on a set of 10 000 samples; columns and rows are reshuffled with the biclustering. Groups of correlating neurons can be clearly seen.

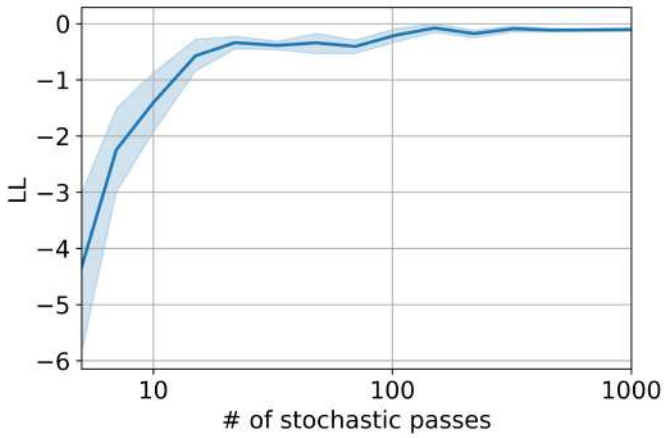


FIGURE 5.8: Log-likelihood computed via MC dropout increases with an increase of the number of stochastic passes T. More than 100 samples are needed to reach convergence.

In what follows, we consider the probabilistic generation of masks m_h from some distribution $P^{(h)}$ with possibly non-i.i.d. distributions of components. Similarly to the case of dropout, we suggest using an unbiased estimate of the layer-wise mean. Our main motivation is to approximately preserve the average performance of the trained network. The construction of the unbiased estimator is non-trivial and is given by celebrated Horvitz-Thompson (HT) estimator Horvitz and Thompson (1952):

$$S_i^h = \sum_{j=1}^{N_{h-1}} \frac{1}{\pi_j^h} m_j^h w_{ij}^h O_j^{h-1}, \quad i = 1, \ldots, N_h, \tag{5.2}$$

where π_j^h is the marginal probability of value 1 for the random variable m_j^h.

Decorrelation Approaches

Let us consider a certain layer h of the network with dropout. We assume that we have access to the correlations $C_{ij}^{(h)} = \mathrm{corr}_x\{O_i^h(x), O_j^h(x)\}$, $i, j = 1, \ldots, N_h$. In practice, we compute an empirical correlation based on some set of points which represents the data distribution well enough, obtaining the correlation matrix:

$$C^{(h)} \in \mathbb{R}^{N_h \times N_h} \tag{5.3}$$

between the neurons of the h-th hidden layer. Below we discuss several approaches to sampling neurons in a way that the correlation between sampled neurons is as small as possible. We note that instead of the correlation matrix $C^{(h)}$ one may consider the covariance matrix $K^{(h)}$ in any of the approaches described below. The properties of the methods significantly depend on the choice of the matrix, and we will perform the empirical evaluation of the methods based on each of them in the experiments.

- **Leverage score sampling.** A basic approach for non-uniform sampling of rows and columns in kernel matrices is the so-called *leverage score sampling* Alaoui and Mahoney (2015). In this approach, the neurons are sampled independently with different probabilities π_j^h:

$$\pi_j^h \sim \ell_\lambda^{(h)}(j) = \left[C^{(h)}\left(C^{(h)} + \lambda I\right)^{-1}\right]_{jj}, \; j = 1, \ldots, N_h,$$

  where the quantities $\ell_\lambda^{(h)}(j)$ are called *leverage scores*.

  This approach makes neurons from large and highly correlated clusters to be sampled less frequently. In Section 6.3, we show that leverage score sampling indeed allows obtaining better uncertainty estimates for out-of-distribution data in regression tasks compared to MC dropout. However, its performance for in-domain data is even inferior to uniform sampling.

- **Sampling with Determinantal Point Processes.** Determinantal Point Processes (DPPs) (Kulesza et al., 2012) are specific probability distributions over configurations of points that encode diversity through a kernel function. They were introduced in (Macchi, 1975) for the needs of statistical physics and were used for a number of ML applications, see Kulesza et al. (2012) for an overview. DPP can be seen as a probabilistic MaxVol algorithm (Goreinov et al., 2010; Çivril and Magdon-Ismail, 2009) of finding a maximal-volume submatrix.

  We use correlation matrix $C^{(h)}$ as the likelihood kernel for DPP. Then, given a set S of selected points for a mask distribution $m_h \sim DPP\left(C^{(h)}\right)$, we obtain

$$\mathbb{P}[m_h = S] = \frac{\det C_S^{(h)}}{\det[C^{(h)} + I]}, \;\; h = 1, \ldots, K,$$

where $C_S^{(h)} = \left[C_{ij}^{(h)}\right]_{i,j\in S}$, i.e., a square submatrix of $C^{(h)}$ obtained by keeping only rows and columns indexed by S. Finally, we sample T masks for each layer based on corresponding correlation matrices.

To better understand the DPP, let us come back to the correlation matrix depicted in Figure 5.7. The probability for DPP to take highly correlated neurons into the sample S is low as in such case the corresponding determinant $\det C_S^{(h)}$ will have a small value. Thus, DPP tends to sample neurons from different clusters, increasing the diversity of the ensemble.

From a computational point of view, DPP-sampling requires $O(N_h^3)$ operations for generating each sample. It is quite expensive but completely viable even for modern large networks, which usually have up to 1024 neurons in fully-connected layers. Importantly, masks can be precomputed once, and then the same masks are used on the inference stage for every test sample with no additional overhead. Also, computations in last fully-connected layers with dropout usually require only a few percents of the total computational budget in ImageNet-size networks. Therefore, a computational overhead caused by the DPP-sampling does not have a significant impact on the inference time.

- **k-DPP.** The k-DPP is a variation of the DPP, conditioned to produce samples of fixed size $|S| = k$. With the cost of introducing an additional parameter, it allows to tune the sampling procedure as the choice of k apparently has a significant influence on the result. Motivated by clustered behaviour observed in real-world correlation matrices (see again Figure 5.7), we suggest taking k dependent on the rank of matrix $C^{(h)}$ and set

$$k = \pi \cdot \operatorname{rank}\left(C^{(h)}\right)$$

for some value $0 < \pi \leq 1$. In practice, matrix$C^{(h)}$ is never exactly low-rank, and some notion of efficient rank should be used.

Importantly, if $\pi = 1$ and matrix $C^{(h)}$ is indeed block-structured and thus low-rank, the resulting sampling procedure for k-DPP will always give one neuron sampled from every block giving the same resulting approximation for all the samples. This behaviour contradicts our initial goals of diversifying samples; therefore, we took $\pi = 0.5$ in most of the experiments. In the case of k-DPP, the computation of the marginal probabilities π_j^h for HT-estimator 5.2 is non-trivial and requires the separate optimization procedure, see the details in Amblard et al. (2018). We also note that DPP-based sampling may be considered from a fully Bayesian perspective, with DPP being a special prior on masks promoting diversity.

5.2.3 GP-based enhancing of dropout NNs

Bayesian introduction

The uncertainty estimate, defined in Section 5.2.1, may be accessed through the Bayesian point of view: we can treat neural networks as probabilistic models $p(y \mid x, \omega)$. The vector of neural network weights ω is assumed to be a random variable with some prior distribution $p(\omega)$. The likelihood $p(y \mid x, \omega)$ determines the distribution of network output at a point x given specific values of parameters ω. In the regression case, the likelihood can be simply assumed to be Gaussian with the mean value given by the function $\hat{f}(x, \omega)$, which is the neural network itself. There is a vast literature on training Bayesian networks (see Graves (2011) and Paisley et al. (2012) among many others), which mostly targets the so-called variational approximation of the intractable posterior distribution $p(\omega \mid D)$ by some easily computable distribution $q(\omega)$.

The approximate posterior predictive distribution reads as:

$$q(y \mid x) = \int p(y \mid x, \omega)\, q(\omega)\, d\omega.$$

The simple way to generate random values from this distribution is to use the Monte-Carlo approach, which allows estimating the mean:

$$\mathbb{E}_{q(y \mid x)}\, y \approx \frac{1}{T} \sum_{t=1}^{T} \hat{f}(x, \omega_t),$$

where the weight values ω_t are i.i.d. random variables from distribution $q(\omega)$. Similarly, one can use Monte-Carlo to estimate the approximate posterior variance $\hat{\sigma}^2(x \mid \hat{f})$ of the prediction y at a point x and use it as an acquisition function:

$$A(x \mid \hat{f}, D) = \hat{\sigma}^2(x \mid \hat{f}).$$

We note that the considered acquisition function formally doesn't depend on the dataset D except for the fact that D was used for training the neural network $\hat{f}$.

The standard active learning approaches rely on the greedy point selection by design. If one tries to obtain several samples with the same acquisition function, it usually results in obtaining several nearby points from the same region of design space, see Figure 5.9. Such behaviour is typically undesirable as nearby points are likely to have very similar information about the target function. Moreover, neural network uncertainty predictions are sometimes overconfident in out-of-sample regions of design space.

There are several approaches to overcome these issues each having its drawbacks:

1. One may retrain the model after each point addition, which may result in a significant change of the acquisition function and lead to the selection of a more diverse set of points. However, such an approach is

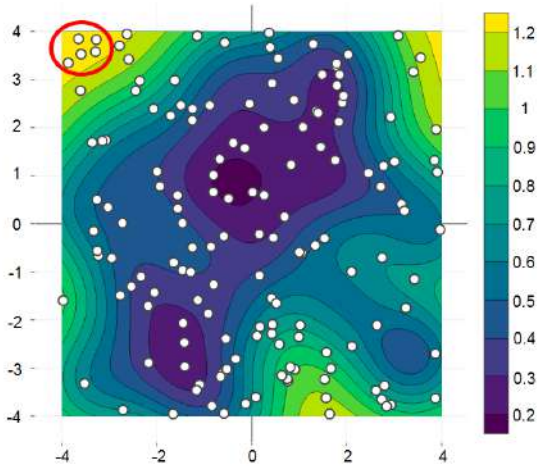


FIGURE 5.9: Contour plot of neural network variance prediction for bivariate problem and pool points (white). Five points from the pool with maximum values of variance lie in the same region of design points (upper-left corner).

usually very computationally expensive, especially for neural network-based models.

2. One may try to add a distance-based heuristic, which explicitly prohibits sampling points which are very close to each other and increase values of acquisition function for points positioned far from the training sample. Such an approach may give satisfactory results in some cases yet usually requires fine-tuning towards particular applications (like the selection of specific distance function or choice of the parameter value which determines whether two points are near or not), while its performance may degrade in high-dimensional problems.

3. One may treat specially normalized vector of acquisition function values at points from the pool as a probability distribution and sample the desired number of points based on their probabilities (the higher the acquisition function value, the point is more likely to be selected). This approach usually improves over the greedy baseline procedure. However, it still gives many nearby points.

GP approximation of Bayesian NN

We propose to overcome the difficulties mentioned above by considering the full approximate posterior distribution. Effectively, the random function

$$\hat{f}(x, \omega) = \mathbb{E}_{p(y \mid x, \omega)}\, y$$

is the stochastic process indexed by x.

The covariance function of the process $\hat{f}(x, \omega)$ is given by

$$k(x, x') = \mathbb{E}_{q(\omega)} \big(\hat{f}(x, \omega) - m(x)\big) \big(\hat{f}(x', \omega) - m(x')\big),$$

where $m(x) = \mathbb{E}_{q(\omega)} \hat{f}(x, \omega)$.

As was shown in Matthews et al. (2018) and Lee et al. (2017), neural networks with random weights converge to Gaussian processes in the infinite layer width limit. However, one is not limited to asymptotic properties of purely random networks as Bayesian neural networks trained on real-world data exhibit near Gaussian behaviour, see the example in Figure 5.10.

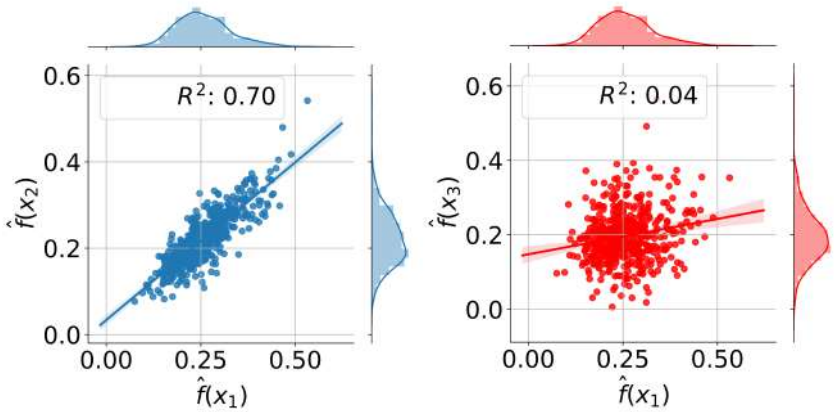


FIGURE 5.10: Bivariate distribution plots for the stochastic NN output at points x_1, x_2 and x_3, where x_1 is much closer to x_2 in feature space than to x_3. Both univariate and bivariate distributions are Gaussian-like, while the correlation between function values is much higher for closer points.

We aim to make the Gaussian process approximation $\hat{g}(x \mid \hat{f})$ of the stochastic process $\hat{f}(x, \omega)$ and compute its posterior variance $\hat{\sigma}^2(x \mid \hat{f}, X)$ given the set of anchor points $X = \{x_i\}_{i=1}^N$. Though the set X can be chosen arbitrarily, in practice, the approximation quality benefits from X being uniformly spread over the design space. Typically, X is a subset of the training sample. Given X, Monte-Carlo estimates $\hat{k}(x', x'')$ of the covariance function $k(x', x'')$ for every pair of points $x', x'' \in X \cup x$ allow computing

$$\hat{\sigma}^2(x \mid \hat{f}, X) = \hat{k}(x, x) - \hat{k}^{\mathrm{T}}(x) \hat{K}^{-1} \hat{k}(x), \tag{5.4}$$

where $\hat{K} = \left[\hat{k}(x_i, x_j)\right]_{i,j=1}^N$ and $\hat{k}(x) = \left(\hat{k}(x_1, x), \ldots, \hat{k}(x_N, x)\right)^{\mathrm{T}}$.

We note that only the trained neural network $\hat{f}(x, \omega)$ and the ability to sample from the distribution $q(\omega)$ is needed to compute $\hat{\sigma}^2(x \mid \hat{f}, X)$.

The benefits of the Gaussian process approximation and the usage of the formula (5.4) are not evident as one might directly estimate the variance of neural network prediction $\hat{f}(x, \omega)$ at any point x by sampling from $q(\omega)$ and use it as an acquisition function. However, the approximate posterior variance $\hat{\sigma}^2(x \mid \hat{f}, X)$ of Gaussian process $\hat{g}(x \mid \hat{f})$ has an important property that is has large values for points x lying far from the points from the training set X. Thus, out-of-sample points are likely to be selected by the active learning procedure.

Moreover, the function $\hat{\sigma}^2(x \mid \hat{f}, X)$ depends solely on covariance function values for points from a set X (and not on the output function values). Such property allows updating uncertainty predictions by just adding sample points to the set X. More specifically, if we decide to sample some point x, then the updated posterior variance $\hat{\sigma}^2(x \mid \hat{f}, X')$ for $X' = X \cup x'$ can be easily computed:

$$\hat{\sigma}^2(x \mid \hat{f}, X') = \hat{\sigma}^2(x \mid \hat{f}, X) - \frac{\hat{k}^2(x, x' \mid \hat{f}, X)}{\hat{\sigma}^2(x' \mid \hat{f}, X)}, \tag{5.5}$$

where $\hat{k}(x, x' \mid \hat{f}, X) = \hat{k}(x, x') - \hat{k}^{\mathrm{T}}(x)\hat{K}^{-1}\hat{k}(x')$ is the posterior covariance function of the process $\hat{g}(x \mid \hat{f})$ given X.

Importantly, $\hat{\sigma}^2(x \mid \hat{f}, X')$ for points x in some vicinity of x' will have low values, which guarantees that further sampled points will not lie too close to x' and other points from the training set X. The resulting NNGP active learning procedure is depicted in Figure 5.11 and described in Algorithm 1.

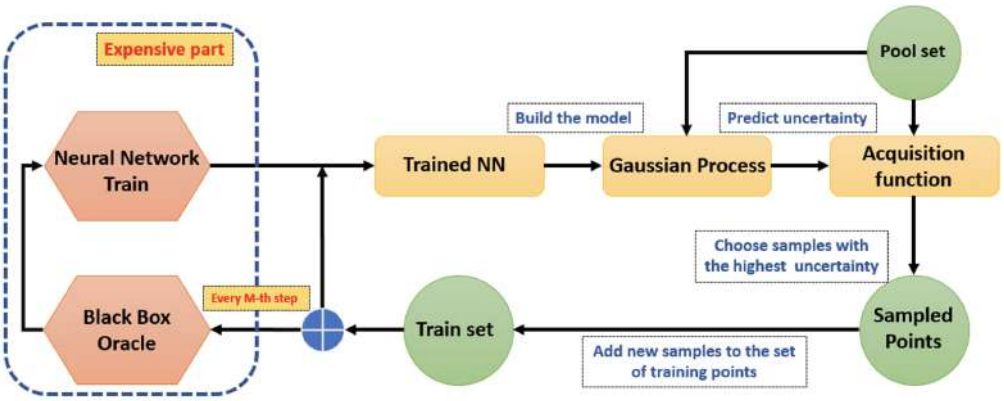


FIGURE 5.11: Schematic representation of the NNGP approach to active learning. GP is fitted on the data from a stochastic output of NN, and the posterior variance of GP is used as an acquisition function for sampling. The most computationally expensive part (function evaluation at sampled points and neural network retraining) is done only every M steps of sampling, while all the intermediate iterations are based solely on trained neural network and corresponding GP approximation.

Let us also consider the multi-step active learning process, which should be computationally efficient, i.e., provide high approximation quality with a minimum number of points selected and also minimize the number of other computationally expensive operations, such as neural network retraining. It is summarized in Algorithm 2 and also shown in Figure 5.11

5.2.4 Baselines

We use the following baselines to compare the performance of the proposed UE and AL machinery:

Algorithm 1 NNGP

Require: Number of samples to generate N_s, pool $\mathcal{P}$, the set X_* of inducing points for Gaussian process model, neural network model $\hat{f}(x, \omega)$, dropout probability π and regularization parameter λ.

Ensure: Set of points $X_s \subset \mathcal{P}$ with $|X_s| = N_s$.

for $t = 1, \ldots, T$ **do**
  $\omega_t \sim \text{Bern}(\pi)$.
  $\omega_t = \hat{\omega}(\omega_t)$.
  $y_t^i = \hat{f}(x_i, \omega_t)$ for each $x_i \in \mathcal{P}$.
  $z_t^j = \hat{f}(x_j, \omega_t)$ for each $x_j \in X_*$.
end for
Calculate the covariance matrix $\hat{K} = \left[cov(z^i, z^j)\right]_{i,j=1}^{N}$.
for each $x_j \in \mathcal{P}$ **do**
  $\hat{k}_j = \left[cov(z^i, y^j)\right]_{i=1}^{N}$.
  $v_j = var(y^j)$.
  $\hat{\sigma}_j^2 = v_j - \hat{k}_j^{\mathrm{T}}(\hat{K} + \lambda I)^{-1}\hat{k}_j$.
end for
Return N_s points from pool $\mathcal{P}$ with largest values of the variance $\hat{\sigma}_j^2$.

Algorithm 2 M-step NNGP

Require: Number of samples to generate N_s, number of samples per active learning iteration M, pool $\mathcal{P}$, the set X_* of inducing points for Gaussian process model, neural network model $\hat{f}(x, \omega)$, dropout probability π and regularization parameter λ.

Ensure: Set of points $X_s \subset \mathcal{P}$ with $|X_s| = N_s$.

Initialize sets $X_s := \emptyset, \mathcal{P}^* := \mathcal{P}$.
for $m = 1, \ldots, M$ **do**
  Run NNGP procedure with parameters $N_s/M, \mathcal{P}^*, X_* \cup X_s, \hat{f}, \pi, \lambda$, which returns a set of points X_o.
  $X_s = X_s \cup X_o$.
  $\mathcal{P}^* = \mathcal{P}^* \setminus X_o$.
end for

- **Random sampling.** This algorithm samples random points x from the pool. Computational cost is $O(1)$ in this case, which makes this algorithm the fastest compared to all the others. This approach may be used for both classification and regression tasks.

- **Greedy max-min sampling.** To sample a point, this algorithm takes the point from the pool most distant from the training set (in the l_2 sense) and adds it to the training set. This process continues until the required number of points is added to the training set. The acquisition function for max-min sampling is

$$A_{\text{MM}}(x) = \min_k \|x - x_k^{\text{train}}\|^2.$$

  In this case, computational cost can be estimated as $O(N_{train}N_{pool})$.

- **Batch max-min sampling.** Although straightforward and intuitive, the max-min sampling is also computationally expensive in the case of a large number of dimensions and pool/training set size since it requires that a full distance matrix is calculated on every stage of active learning. We propose the batch version that has the same acquisition function but samples K points which are the most distant from the training set. Although it does seem less optimal, this solution speeds up sampling up to K times, assuming K is the number of samples to be sampled on each iteration. In our experiments, we set K equal to 4.

- We also consider **oracle sampling**, which simply make uses of the actual model errors at test points. Of course, the real errors are not available in practice; however, it is natural to assume that good uncertainty estimates should correlate with model's errors. Thus, oracle errors can be used in experiments to compare the performance with considered approaches for both classification and regression tasks.

5.2.5 Metrics

To measure the performance of the uncertainty estimation and active learning we either use the Dolan-More performance curves (see Section 5.3.3) or the following metrics:

- **Uncertainty estimation.** In general, for uncertainty estimates, we, following (Hernández-Lobato and Adams, 2015; Jain et al., 2019), compute log-likelihood of Gaussian distribution with uncertainty estimates plugged in place of standard deviation. Standard deviation of multiple stochastic runs (or predictions, in case of ensembles) is used as an uncertainty estimate and in acquisition function as well.

- **Active learning.** During the neural network training, we usually optimize the mean squared error (MSE, l_2) metric, and it is natural to report this loss as an error on the test set that the actively trained model reaches. However, we also report the mean absolute error (MAE, l_1)

and maximum absolute error (MaxAE, l_∞). We believe that the actual task of regression is more general than optimization of one given metric (e.g., MSE); thus, the choice of a particular metric is merely an operationalization of the real problem behind the regression task. Moreover, several applications exist, in which the maximal error is a much more appropriate accuracy metric (like chemistry or physical simulations) than the mean error. Unfortunately, it is hard to use the l_∞ loss function for training neural networks since it is non-differentiable. In case one deals with two algorithms that have a similar MSE and significantly different MaxAE, the algorithm with a smaller maximal error should be preferred.

5.3 Numerical experiments setup

In this section, we introduce the methodology of measuring the performance of our models, as well as the details on the training procedure and the architecture of large-scale experiments on active learning and uncertainty estimation.

5.3.1 Models for ESE

As the NN-based ESE models were defined above with the architecture and training procedure specifically tailored to the corresponding problem, we will, therefore, focus on the fine training details, mostly for other baseline models, and optimization specifics.

Silicon crystal

Data split. In all the cases involving the model accuracy measurement, the train-test data split was done in 9:1 ratio. For the models that optimized hyperparameters, the validation set was typically chosen over 10-fold cross-validation.

Models optimization metric. All the models optimized the MSE (mean squared error) metric.

Tree-based ensemble baselines. Ensemble algorithms, based on decision trees (Breiman et al., 1984), are extensively studied and widely used in modern machine learning tasks. Most of the algorithms are prone to overfitting, can work with various types of features, and thus require no or little data preprocessing. For our regression task, we used the following two types of ensembling on decision trees: the random forest algorithm (Breiman, 2001), and the gradient boosting regression (Friedman, 2001).

One of the major drawbacks of most of these algorithms is the memory size used for storing the trained model. The memory size rapidly grows as the depth of the trees increases. Moreover, the size of the models increases

linearly with the increasing number of estimators (trees) for the random forest and extra trees algorithms. Thus, to obtain good accuracy, one needs to use a lot of memory on both hard disk drive and RAM.

We did a cross-validation over the following hyperparameters, with the best ones marked with a bold font:

- random forest: maximal tree depth (3, **10**, 15), number of estimators (50, 100, 200, 300, **500**), minimal number of samples for splitting (**2**, 5, 10).
- gradient boosting: maximal tree depth (3, **5**, 10), learning rate (0.01, 0.03, **0.1**, 0.3), number of estimators (30, 100, 200, **300**), minimal number of samples for splitting (2, **5**, 10).

Although the random forest regression algorithms with 500 estimators show slightly better results than the one with 300 estimators, we used 300 estimators to avoid memory problems. We used the subsampling technique (see details in (Friedman, 2002)) for the gradient boosting regression to prevent overfitting.

Incremental fitting. We also show that our NN-based surrogate models can successfully learn from several datasets and assimilate them. This capability is becoming increasingly important with the spread of materials property databases that collect data from different studies (Jain et al., 2013). The incremental training of the NN starts from the same weights but is done on the extended dataset with the additional data included. We also increase the learning rate of stochastic gradient descent algorithm and regularizers (dropout rate and weight regularization) to circumvent limitations arising from the same local minima of the loss function established during the training on the initial dataset. This allows the model to not only handle additional training on the incoming data appended to a database but to do it much faster than from scratch.

It is important to note that the joint data set is imbalanced in the sense of having more values from data drawn from $\boldsymbol{\varepsilon}^{\mathbf{6D}}$. In order to balance the number of samples, one may use various approaches for over-sampling and under-sampling, see Batista et al. (2004); we used random over-sampling for the models.

Optimization. The optimizations performed in this part of the research were performed using the SciPy package (Virtanen et al., 2020) implementations. Namely, finding the shortest pathways to the metallic behaviour of the silicon crystal and the location of bandgap isosurfaces' edges used an upgraded version of differential evolution (Storn and Price, 1997) for the global optimization, and the adaptive Nelder-Mead method (Gao and Han, 2012) for the local optimization.

Elastic strain energy density model. The model for the energy density estimation (h, defined in Section 4.2.2) was based on the kernel ridge regression, implemented in Scikit-learn Python library (Pedregosa et al., 2011). The mean absolute error measured on the test set was not exceeding 0.001 eV.

Diamond crystal models

Data split. In all the cases involving the model accuracy measurement, the train-test data split was done in 9:1 ratio in random. For the models that optimized hyper-parameters, the validation set was typically chosen over 5-fold cross-validation.

Inference time measurements. The performance was measured using the `timeit` function from the standard Python library. At least 10 cycles were performed, and the average value is reported together with the standard deviation.

Models optimization metric. For all the regression tasks, the MSE was used as an optimization metric, and a cross-entropy was used for the classification tasks. We would like to note that the implementation of the CNN we provide is capable of optimizing more complex metrics in order to tailor the model for a specific metric, e.g., bandgap; yet we were not exploiting it in the reported experiments.

Elastic strain energy density model. A feed-forward neural network with (128-256-256-512) architecture was used to provide the predictions for the elastic strain energy density h (see Section 4.2.2). It uses a rectified linear unit (Glorot et al., 2011) as an activation function, was trained for 20 000 epochs using the Adam optimizer (Kingma and Ba, 2014), and demonstrated the mean absolute error of 0.2978 eV on the separate test set.

Effective mass estimation. The partial derivatives used in the effective mass estimation (2.11) were approximated using central differences with the mesh size proportional to the k-grid internal distance:

$$\begin{aligned}\left.\frac{\partial^2 E_{\mathrm{CB}}}{\partial k_1 \partial k_2}\right|_{(k_1^\star,k_2^\star,k_3^\star)} \approx \frac{1}{4\delta^2}(&E_{\mathrm{CB}}(k_1^\star+\delta,k_2^\star,k_3^\star)- \\ &E_{\mathrm{CB}}(k_1^\star-\delta,k_2^\star,k_3^\star)+ \\ &E_{\mathrm{CB}}(k_1^\star,k_2^\star+\delta,k_3^\star)- \\ &E_{\mathrm{CB}}(k_1^\star,k_2^\star-\delta,k_3^\star)).\end{aligned} \tag{5.6}$$

In a $8\times8\times8$ **k**-mesh, $\delta = \frac{\sqrt{2}}{8}(\frac{2\pi}{a})$ when computing $m^\star$ positions along the 'Δ'-line and is in the unit of the reciprocal of length, where a is the lattice constant of a carbon diamond. All the other derivatives are approximated in the same way to achieve the effective mass tensor approximation.

Kernel ridge regression baselines. We were using the kernel ridge regression with an RBF kernel implemented in Scikit-learn Python library (Pedregosa et al., 2011). Hyper-parameters were optimized with an adaptive Nelder-Mead optimization method (Gao and Han, 2012) over the validation set; the best of three separate runs starting from random points were chosen. A linear kernel was used for the classification task.

Feed-forward NN baselines. We have used a deep architecture (128-256-256-512-512-1024-1024) so the number of parameters of a feed-forward neural network will be larger than the number of parameters for the complex CNN model, thus resulting in a more fair comparison. The leaky rectifier unit (Maas et al., 2013) was used as an activation function, and the same

stopping criteria as for CNN were used, as well as the adaptive learning rate and l_2-regularization schedule.

5.3.2 Active learning and uncertainty estimation

We focus on the non-Bayesian methods as comparable in terms of the computational time for large datasets and models.

MCDUE-based experiments

For the experiments described in Section 6.2, we used the following experimental setup:

1. Initialization of the initial dataset I, training pool $\mathcal{P}$, number of samples added on each step m, the final size of the dataset f, network architecture, and learning parameters.

2. The network is trained on the initial dataset I. Its weights are copied to the networks corresponding to each active learning algorithm.

3. For each active learning algorithm:

   (a) While $|I| < f$:

      i. Obtain the rank r_j for every $x_j \in \mathcal{P}$ using an acquisition function A.
      ii. Sample point set $S \subset \mathcal{P}$, $|S| = m$ with maximal ranks r_j.
      iii. Add S to I: $I := I \bigcup S$.
      iv. Exclude S from the corresponding $\mathcal{P}$: $\mathcal{P} := \mathcal{P}/S$.
      v. Train the neural network on I.

   (b) Calculate the metrics.

For each experiment, the number of training epochs was set to 10 000. We used the l_2-regularization of the weights with the regularization parameter $\alpha = 10^{-5}$, a five-layer fully-connected network with the $256 - 128 - 64$ architecture and leaky linear rectifier (Maas et al., 2013) with leakiness $\beta = 0.01$ as an activation function. We used the Theano library (Al-Rfou et al., 2016) and Lasagne framework (Dieleman et al., 2016). Data points were shuffled and split in the following ratio: 20% on a training set, 60% on a pool, 20% on the test set. Also, if the network was trained from scratch (re-initialized weights), the number of training epochs is doubled (to catch up with the initial training). Since the neural network training procedure is stochastic by its nature, we conducted 20 experiments shuffling the dataset and re-initializing weights each time.

Experiments on the dropout diversity masks

All the regression models were trained with RMSE as a loss function. For DPP-based methods, we use the DPPy implementation provided in (Gautier et al., 2019). On top of single models, we also consider a straightforward ensemble approach with NNs trained exactly the same way as single models from different initializations.

The specific details for each experiment are as follows:

- **Uncertainty Estimation for Regression Datasets.** For this experiment, we used feed-forward NNs with leaky ReLU activation function (Maas et al., 2013) and (128-128-64) architecture. For each dataset, 50% was used for the training and 50% for testing. Multiple experiments are done via 2-fold cross-validation, and multiple runs / UE runs. See Appendix D.2 for the list of datasets used.

  Neural networks were trained for 10 000 epochs maximum, with checking the error on the validation set every 100 iterations: early stopping triggers if the error did not decrease for five consecutive checks (patience = 5). Batch size equals to 500, dropout applied after the hidden layers only (except for the last layer) with rate equal to 0.5, except for the experiment C (see below). MSE was used as a loss function, and optimization was performed with the standard settings of PyTorch Adadelta optimizer.

  For in-domain uncertainty estimation: for each dataset, random 50% of points were used for training and other 50% for testing. The log-likelihood values are averaged over testing set. Multiple experiments are done via 5 random train-test splits, 2-fold cross-validation and 5 runs of the training procedures for every model (resulting in 50 average log-likelihood values contributing to each boxplot). Uncertainty estimates were computed for the different number of stochastic passes $T = 10, 30$, and 100 for every model.

  Ensembles of models were trained separately on the same data from different random weight initializations.

- **Regression Datasets: NN Configurations Effect.** In order to testify UE approaches on slightly different settings of NN architecture, we settled out three more experiments with variations in:

  - **architecture**. Different problems require different fully-connected layers of NNs to be used in order to being able both train on data successfully and do not overfit.
  - **activation function**. It was shown in (Hein et al., 2019) that the choice of activation function might alter the confidence of the predictions on the out-of-distribution data. To that end, we considered both linear and non-linear rectifiers.
  - **dropout rate**. While in classic papers the most robust dropout rate $p = 0.5$ is often considered, for real problems, lesser values of p

are used in order to speed up the convergence and smaller NNs to be used. Initially (Gal, 2016), the dropout rate is proposed to be chosen in a cross-validation round together with other hyperparameters, such as regularization term, learning rate, etc.

The variations in the settings are provided in Table 5.1.

TABLE 5.1: Settings for UCI experiments.

Index	Architecture	Activation	p
A (main text)	128-128-64	leaky ReLU	0.5
B	32-32-16	leaky ReLU	0.5
C	128-128-256	CELU	0.2
D	256-256-512	CELU	0.5

NNGP-based experiments

Setup details for each provided experiment are as follows:

- **Airline delays dataset and NCP comparison.** Following Hafner et al. (2018), we use the airline delays dataset (see Hensman et al. (2013)) and NN consisting of two layers with 50 neurons each, leaky ReLU activation function, and trained with respect to NCP-based loss function. We took a random subset of 50 000 data samples from the data available on the January – April of 2008 as a training set, and we chose 100 000 random data samples from May of 2008 as a test set. We used the following variables as input features *PlaneAge*, *Distance*, *CRSDepTime*, *AirTime*, *CRSArrTime*, *DayOfWeek*, *DayofMonth*, *Month*, and *ArrDelay* + *DepDelay* as a target. We would like to note that we used (as in the original paper) an NCP-based (noise contrastive prior-based, see details in Hafner et al. (2018) loss function different from classic MSE.

- **Experiments on UCI datasets.** We conducted a series of experiments with active learning performed on the data from the UCI ML repository Dua and Taniskidou (2017). All the datasets represent real-world regression problems with 15+ dimensions and 30000+ samples, see Section D.1 of Appendix D. The datasets were chosen to be hard-to-fit-with-GP.

  For every experiment, data are shuffled and split in the following proportions:

  - training set D_{train}: 10%;
  - test set D_{test}: 5%;
  - validation set $(D_{val}$ (for early-stopping): 5%;
  - pool $\mathcal{P}$: 80%.

We used a simple neural network with three hidden layers of sizes 256, 128 and 128. We performed 16 active learning iterations with 200 points picked at each iteration. The learning rate started at 10^{-3}, its decay was set to 0.97 and changed every 50 000 epochs. The minimal learning rate was set to 10^{-5}. We reset the learning rate for each active learning algorithm in the hope of beating the local minima problem. The training dropout rate was set to 0.1. L_2 regularization was set to 10^{-4}. Batch size set to 200.

- **SchNet training.** We tested MCDUE and NNGP on the problem of predicting the internal energy of the molecule at 0K from the QM9 data set (Ramakrishnan et al., 2014). We used a Tensorflow implementation of a SchNet (Schütt et al., 2018) with the same architecture as in the original paper except for an increased size of hidden layers (from 64 and 32 units to 256 and 128 units, respectively) and dropout layer placed in between of them and turned on during an inference only. It is expected that wider hidden layers lead to better uncertainty estimation, as the infinite-width layer will theoretically result in an unbiased estimate of both mean and variance for the corresponding Bayesian neural network (Gal, 2016).

  In our experiment, we separate the whole dataset of 133 885 molecules into the initial set of 10 000 molecules, the testing set of 5 000 molecules, and the rest of the data allocated as the pool. On each active learning iteration, we perform 100 000 training epochs and then calculate the uncertainty estimates using either MCDUE or NNGP approach, or random sampling as a baseline. We then select 2 000 molecules with the highest uncertainty from the pool, add them to the training set and perform another active learning iteration.

5.3.3 Performance curves

To compare the performance of the algorithms across the different datasets, the initial number of training samples and training samples themselves, we will use Dolan-More curves, which, following Dolan and Moré (2002), may be defined as follows. Let q_a^p be an error measure of the a-th algorithm on the $\mathcal{P}$-th problem. Then, defining the performance ratio $r_a^p = \frac{q_a^p}{\min_x(q_x^p)}$, we can define the Dolan-More curve as a function of the performance ratio factor τ:

$$\rho_a(\tau) = \frac{\#(p : r_a^p \leq \tau)}{n_p}, \tag{5.7}$$

where n_p is a total number of evaluations for the problem p. Thus, $\rho_a(\tau)$ defines the fraction of problems in which the a-th algorithm has the error not more than τ times bigger than the best competitor in the chosen performance metric.

Dolan-More performance curves are a natural and popular choice for the benchmarking. However, Gould and Scott (2016) states that this tool should

be used with care, providing an example of possible misinterpretation of comparison results.

5.4 Summary

This chapter was dedicated to a detailed description of the developed methodology. Starting from the portrayal of the desirable properties of the future model, in Section 5.1.1, we have described simple fully-connected NN models we used for the fitting of silicon crystal's properties. Here, we also provided insights on how to combine the data with different fidelity into a single model.

Based on the insights provided in Section 4.3, we have redesigned a straightforward fully-connected property-based model into a powerful convolutional neural network that mimics some of the intrinsic properties of the output data by special convolutional layers. In Section 5.1.2, we also provided details on the intricate training procedure, which involves both pre-training on the PBE data and a few active learning cycles.

In the following Section 5.2, three dropout-based algorithms for the neural networks are suggested: a simple MC dropout, first described in Gal (2016) (yet extensively tested further in this work), its modification that aims to the diversification of the masks using determinantal point processes (DPPs), and GP-based approximation of the neural network's output. The explanation is followed by a brief description of baselines and metrics. This methodology was developed in collaboration with Maxim Panov and Kirill Fedyanin.

At the end of this chapter, in Section 5.3, we provide more technical details on how the experiments for ESE-related models and UE approaches were designed. We also introduce the Dolan-More performance curves to measure the performance across different algorithms and datasets.

In the next chapter, we present the results of numerical experiments that evaluate the accuracy of the designed ESE machinery and provide the comparison of the designed AL and UE methods for a wide range of problems.

6 ML experiments

6.1 Numerical experiments for ESE models

This section is dedicated to the numerical experiments that estimate the performance of the ESE models for silicon and diamond crystals. The supporting information is provided in Section 5.3.1.

6.1.1 Silicon crystal model

PBE data model

We start with the results on a cheap yet inaccurate data from the PBE calculations. These preliminary experiments are important due to the following:

- PBE-PAW data could be sampled on a much larger scale since it is 100-1000 times cheaper than G_0W_0 data in terms of computational time of first-principles calculations. Therefore, one may check the potential large-scale applications and estimate the model performance in this scenario.
- Low-fidelity data could be used to pre-train the model in a variety of ways. The most simple scenario is "Δ-ML" (Ramakrishnan et al., 2015), when another model fits the discrepancy between the low-fidelity-data model's predictions and more credible data; we will make use of this scenario for the diamond crystal experiments. Another possible approach is transfer learning when the low-fidelity-data model requires additional training on more accurate data; it is often considered with the "freezing" of some model's parameters (Samala et al., 2017).

Ensemble methods on decision trees, including gradient boosting regression (GBR) and random forest regression (RFR), Lagrange interpolation, and artificial NN were adopted for ML fitting. Table 6.1 shows the results on the accuracy of these models in the case of non-shear deformation ($\boldsymbol{\varepsilon^{3D}}$). The best results were attained by the NN, which have fast evaluation time and thus are more preferable.

GW data model

After obtaining a very accurate model for the PBE data by the neural network, we fit the difference between this model and GW calculations as a function of a strain and PBE bandgap, resulting in a model with the accuracy comparable to experimental data (Δ-ML). It is shown that GW bandgap in

TABLE 6.1: Root mean squared error (RMSE) for various ML algorithms for the bandgap prediction and energy prediction tasks from PBE data (in units of eV). Lagrange polynomial of degree 8 is used.

ML input	Lagrange	GBR	RFR	NN	ML target
$\boldsymbol{\varepsilon}^{3D}$	0.0150	0.0367	0.0247	**0.0049**	E_g
$\boldsymbol{\varepsilon}^{6D}$	-	0.0743	0.0781	**0.0264**	E_g
$\boldsymbol{k}$ and $\boldsymbol{\varepsilon}^{3D}$	-	0.1125	0.1078	**0.0131**	$E_n(\boldsymbol{k};\boldsymbol{\varepsilon})$ at VB
$\boldsymbol{k}$ and $\boldsymbol{\varepsilon}^{6D}$	-	0.1593	0.1555	**0.0184**	$E_n(\boldsymbol{k};\boldsymbol{\varepsilon})$ at CB

$\boldsymbol{\varepsilon}^{\boldsymbol{3D}}$ strain case can be approximated within an accuracy of 8 meV, see Table 6.2. Our model yields better fitting results compared to a model based on the GW data only, suggesting that our data fusion technique is successfully learning out the GW theory by itself in fitting the electronic bandstructure of a material.

TABLE 6.2: MAE and RMSE (in units of eV) for ML algorithms for bandgap prediction with or without the Δ-ML model. Here, the Lagrange polynomial of degree 8 is used. Relative error: norm of the difference between the true value and the prediction divided by the norm of the true value.

	GW		GW+PBE (Δ-ML)	
ML algorithms	MAE	RMSE	MAE	RMSE
Lagrange	0.0211	0.0274	0.0186	0.0241
GBR	0.0334	0.0521	0.0135	0.0209
RFR	0.0434	0.0596	0.0145	0.0215
NN	0.0099	0.0144	0.0080	0.0118
NN relative error	1.72%	2.78%	1.38%	2.05%

Incremental fitting

The proposed model is, in fact, also able to fuse the learning outcomes from the different datasets. Numerical experiments demonstrate that incremental fitting of the models effectively reduces the error on a new dataset, see Table 6.3. Such incrementally fitted models are, thus, equally applicable to the bandgap approximation and various optimization tasks. Moreover, these models may be reused when shifting to other materials such as Ge, since the implicit insights about symmetries, transitions, and extreme cases are stored in the parameters of NN. Training the model for the other material starting from the weights for Si can significantly reduce the time and amount of data needed due to knowledge transfer, leading to the rapid development of versatile surrogate models for ESE.

The successful combination of the quantitative advantage of PBE and the qualitative advantage of GW results in a bandgap-prediction model with a level of accuracy comparable to experiments.

TABLE 6.3: Si bandgap prediction errors, RMSE and MAE (in units of eV), for the incremental fitting scenario on reduced datasets. The error in both metrics is reduced for both $\boldsymbol{\varepsilon}^{\mathbf{3D}}$ and $\boldsymbol{\varepsilon}^{\mathbf{6D}}$ datasets after the incremental fitting.

	$\boldsymbol{\varepsilon}^{\mathbf{3D}}$		$\boldsymbol{\varepsilon}^{\mathbf{6D}}$	
Metric	before	after	before	after
RMSE	0.0403	0.0069	0.0264	0.0253
MAE	0.0167	0.0052	0.0179	0.0167

6.1.2 Diamond crystal model

In this section, we compare the designed CNN model to the baselines: kernel ridge regression (KRR), and simple feed-forward NN (with the total number of trainable parameters close to one of CNN). We also provide the comparison between the specialized, property-based models (trained to predict a single feature, such as the bandgap or the CBM position as a class), and the band structure models that predict all the bands at once like CNN does.

Property-based models

Table 6.4 shows the results of the comparison between the specialized NN and KRR models and the general-purpose CNN model. The results suggest that the simple NN is better in terms of both value prediction as well as the inference speed. However, the CNN accuracy is close to the NNs, and in the more complicated task of the conduction band minima prediction, the proposed model shows better accuracy, let aside the ability to predict a new class (a CBM not presented in the training set).

We also visualize the error distribution for the bandgap and Γ gap prediction in Figure 6.1.

Bandstructure models

For the more complicated task of the band structure prediction (1040 values with the time-reversal symmetry taken into account), we have set the NN with 1040 values on the last layer and an ensemble of 1040 KRR models as the baselines. Table 6.5 shows the results of numerical experiments. In most of the cases, CNN demonstrates the superior accuracy with the KRR model having a similar error. It should be noted that in the current implementation, the ensemble of KRR models has three orders of magnitude larger inference time. This problem could possibly be mitigated by transferring the implementation from CPU to GPU. The proposed CNN model offers a nice combination of both accuracy and inference time.

TABLE 6.4: Accuracy comparison among specialized models. This means that all the models (except for the CNN) were trained for the selected task only. Bold font indicates the best result in a category.

Metric	CNN	NN	KRR
Bandgap prediction			
RMSE, eV (relative error, %)	0.108214 (2.22%)	**0.096223** **(1.83%)**	0.168535 (3.21%)
MAE, eV (relative error, %)	0.072424 (1.38%)	**0.062605** **(1.19%)**	0.122125 (2.33%)
Γ gap prediction			
RMSE, eV (relative error, %)	0.088265 (1.62%)	**0.085658** **(1.57%)**	0.146379 (2.68%)
MAE, eV (relative error, %)	**0.053497** **(0.98%)**	0.055520 (1.02%)	0.115819 (2.12%)
CBM prediction (classification)			
Error	**2.34%**	5.70%	33.8%[1]
Inference time			
Time	14.4 ms ± 59.2 µs	**1.29 ms ± 14.6 µs**	48.8 ms ± 287 µs

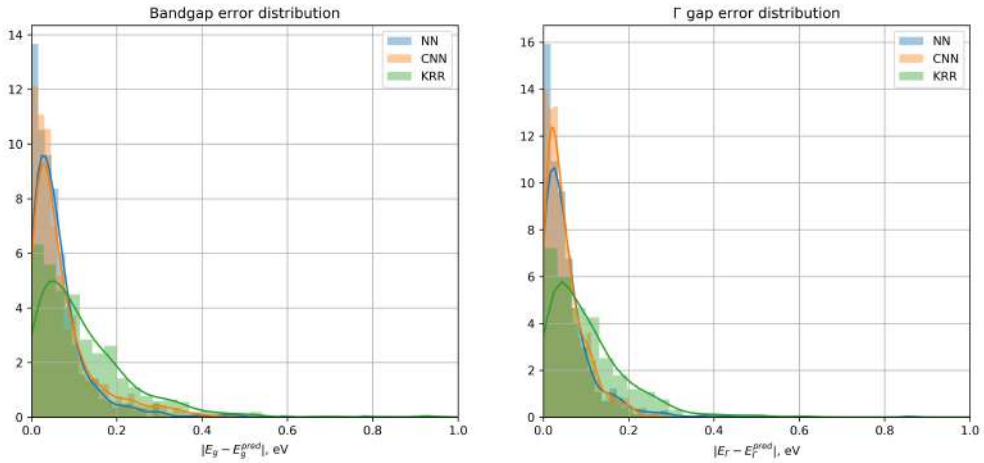


FIGURE 6.1: Error distribution for different algorithms for the bandgap and Γ gap prediction tasks. CNN and NN demonstrate similar performance, while KRR shows the long tail error distribution.

Active learning

We investigate the active learning scenario on the PBE data, as the GW calculations are far too expensive to be used in the comprehensive experiment. Our study indicates that 5-10 cycles of the above active learning may enable the trained CNN to reach the same level of accuracy with less additional data, thus reducing the total amount of ab initio calculations without compromising the robustness of our ML model, see Fig. 6.2.

TABLE 6.5: Accuracy comparison among specialized models. This means that all the models (except for the CNN) were trained for the selected task only. Bold font indicates the best result in a category.

Metric	CNN	NN	KRR
Valence band prediction			
RMSE, eV (relative error, %)	**0.038464** **(0.23%)**	0.052195 (0.31%)	0.043643 (0.26%)
MAE, eV (relative error, %)	**0.031379** **(0.19%)**	0.042052 (0.25%)	0.035710 (0.21%)
Conduction band prediction			
RMSE, eV (relative error, %)	**0.045981** **(0.30%)**	0.111479 (0.72%)	0.059352 (0.38%)
MAE, eV (relative error, %)	**0.035453** **(0.23%)**	0.091714 (0.59%)	0.042620 (0.27%)
Inferenced bandgap prediction			
RMSE, eV (relative error, %)	0.108214 (2.22%)	0.158525 (3.02%)	**0.101998** **(2.13%)**
MAE, eV (relative error, %)	**0.072424** **(1.38%)**	0.120696 (2.30%)	0.082020 (1.56%)
Inferenced Γ gap prediction			
RMSE, eV (relative error, %)	**0.088265** **(1.63%)**	0.149067 (2.73%)	0.097539 (1.79%)
MAE, eV (relative error, %)	**0.053497** **(0.98%)**	0.105048 (1.92%)	0.063617 (1.17%)
Inferenced CBM prediction (classification)			
Error	**2.34%**	4.50%	6.25%
Inference time			
Time	14.4 ms $\pm$ 59.2 µs	**2.48 ms** $\pm$ **32.2 µs**	25 s $\pm$ 105 ms

As for the GW part of the training cycle described in Section 5.1.2, we made 10 iterations of 200 samples each to extend the training set. The error curve is shown on the Figure 6.3, and indicates that the last three AL cycles did not improve the model in terms of accuracy.

EM estimation

To further demonstrate the capability of our ML framework in physics investigation and exploration, we approximate the electron effective mass, $m^\star$. It is a quantity used to model the behavior of a free electron with that mass and is an important basic parameter that influences measurable properties of a solid, from the efficiency of a solar cell to the speed of an integrated circuit. If we denote the conduction band energy dispersion as $E_{\mathrm{CB}}(\boldsymbol{k})$, then the corresponding free electron effective mass tensor can be defined in terms of the Hessian matrix $H(E_{\mathrm{CB}}(\boldsymbol{k}))$ consisting of second partial derivatives against $\mathbf{k}$.

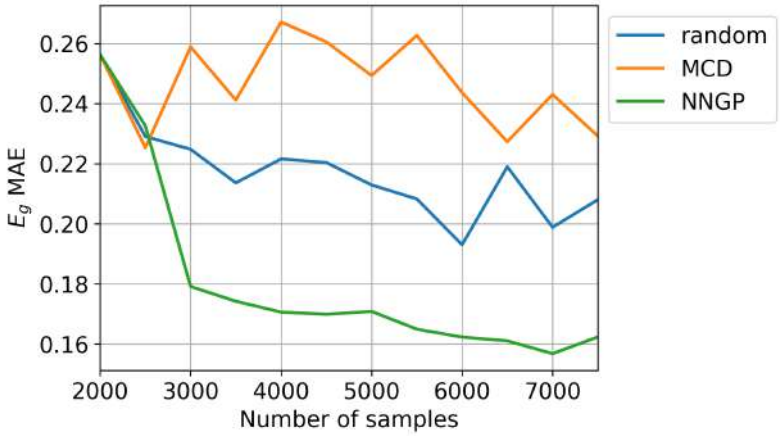


FIGURE 6.2: Steady improvement of model performance during active learning with and without uncertainty estimation on PBE data. MCDUE-based active learning showed results inferior to the random sampling, yet NN+GP approach is showing promising results and thus was used for the GW data part.

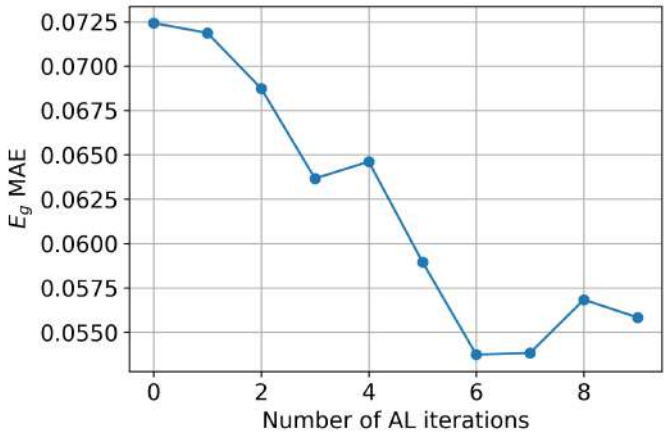


FIGURE 6.3: Improvement of model performance during active learning with NN+GP approach on GW data. 200 samples were drawn at each AL step. The last three cycles did not improve the model in terms of accuracy.

Based upon the values drawn from our ML model, we obtain the $m^\star$ tensor for an undeformed diamond at CBM through fitting the band structure.

Given the second-order derivative nature of $m^\star$, it reveals not only the shape of an energy band but also provides more detailed information of energy dispersion. The anisotropy at CBM is characterized by a longitudinal mass ($m_l = 1.55m_0$, where m_0 is the free electron mass) along the corresponding equivalent (100) reciprocal space direction and two transverse masses ($m_t = 0.31m_0$) in the plane perpendicular to the longitudinal direction. Our results for m_l and m_t are close to both the GW and experimental values (Table 6.6), offering more evidence for the reliability of our electronic band structure fitting machinery. Using this model, we can accurately predict the $m^\star$ tensor

for every **k**-point at every strain level, allowing us to further study the electronic transport properties of a material and gain an insight on how these properties can be tuned by elastic strain.

TABLE 6.6: Longitudinal and transverse electron effective mass at CBM in undeformed diamond (in units of m_0). The results obtained through our CNN model are compared with experiments, our previous feed-forward NN model, and explicit calculations using existing methods including GW, linear muffin-tin-orbital (LMTO) model, G_0W_0, and quasiparticle self-consistent GW (QSGW).

Method	Source	m_l	m_t	m_l/m_t
CNN	this work	1.55	0.31	5.0
NN	Shi et al. (2019)	1.63	0.31	5.16
GW	this work	1.44	0.31	4.61
LMTO	Willatzen et al. (1994)	1.5	0.34	4.41
G_0W_0	Lofas et al. (2011)	1.1	0.22	5.0
QSGW	Lofas et al. (2011)	1.2	0.22	5.45
Experiment	Nava et al. (1980)	1.4	0.36	3.89

To demonstrate the accuracy and applicability of our model, we also show the results on the EM difference for the case of hydrostatic compression and tension ($\varepsilon_{xx} = \varepsilon_{yy} = \varepsilon_{zz}, \varepsilon_{xy} = \varepsilon_{yz} = \varepsilon_{xz}$). The reciprocal components of the effective mass tensor diagonal are shown in Figure 6.4, which shows that our advanced CNN model is capable of predicting intricate derivative properties with suitable accuracy.

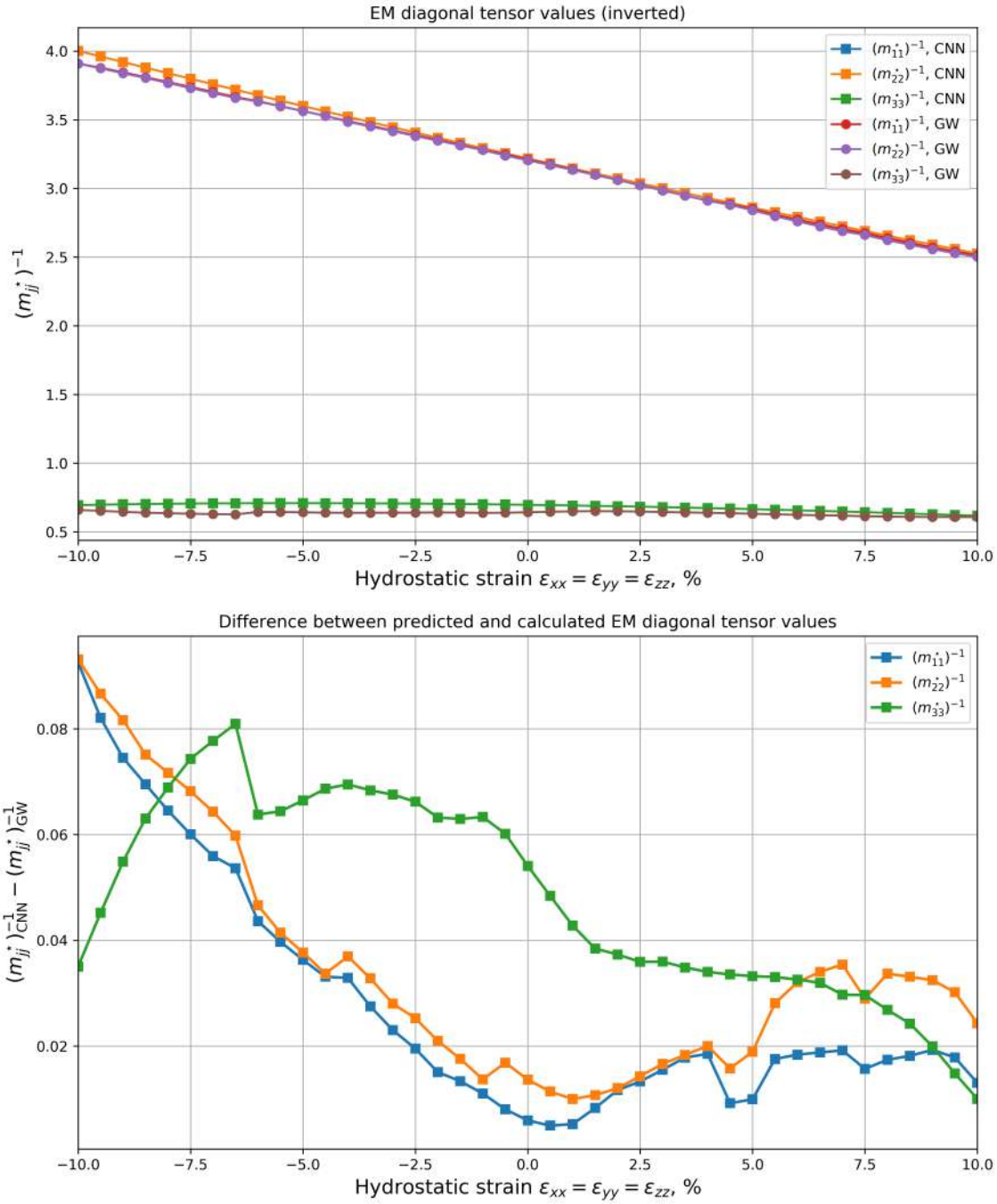


FIGURE 6.4: **Top**: Reciprocals to the effective mass tensor values; GW calculations versus CNN predictions. Both predicted and real EM estimates are close to each other, so we visualized the difference on the **bottom** plot.

6.2 Active learning with MC dropout

This section is dedicated to the results of initial active learning experiments with a simple dropout uncertainty estimate, which were also published in Tsymbalov et al. (2018). For the data sets list used in this section, please refer to the Section D.1 of Appendix D.

6.2.1 Correlation plots

We would start with the demonstration of the possible relations between the simple MC Dropout uncertainty estimate, defined in Section 5.2.1, and the real error on the points UE produced for. The performance of the algorithm depends on the variety of factors, starting from the NN architecture, training procedure, and active learning protocol, up to the data in hand and the amount of it available. So the results may vary from the ones shown in Figure 6.5, where no observable correlation is found, up to the ones in Figure 6.6, where the relationship is evident. The latter is usually observed in the case of very close data distribution of train and test data, and is rather close to the ideal case. However, ever for the case shown in Figure 6.5, the median error for the randomly selected samples is lower than the median error for the selected by the uncertainty estimation 1% of the data.

6.2.2 Ratio plots

First, we compare our MCDUE-based approach with the baseline approaches using the ratio of errors in various metrics. Figures 6.7 and 6.8 show that our active learning approach has a better performance than random sampling in RMSE and MaxAE metrics, and a small accuracy increase as compared to a max-min algorithm. It should be noted that as the number of active learning iterations (new data gathering and learning on the top of it) increases (thus leaving the data pool empty), the ratio turns to 1.

6.2.3 Dolan-More plots

We conducted a number of experiments with a single iteration of active learning performed on the datasets listed in Section D.1 of Appendix D, see Figure 6.9 for Dolan-More curves (defined in Section 5.3.3). Note that $\rho_a(1)$ is the ratio of problems on which the a-th algorithm performance was the best, and it is always the case of the MCDUE-based algorithm. Judging by the area under curve (AUC) metric, the MCDUE-based approach outperforms the random sampling and is slightly better than a batch max-min sampling.

6.3 Diversified dropout masks

In this section, we present the results of the numerical experiments on the DPP-powered dropout approach described in Section 5.2.2. For the details on the setup and dataset processing, please refer to Section 5.3.2. Most of these experiments were published in Tsymbalov et al. (2020). The code reproducing the majority of experiments is available at `https://github.com/stat-ml/dpp-dropout-uncertainty`.

We also tested this approach for the problems of image classification, please refer to Appendix F for details.

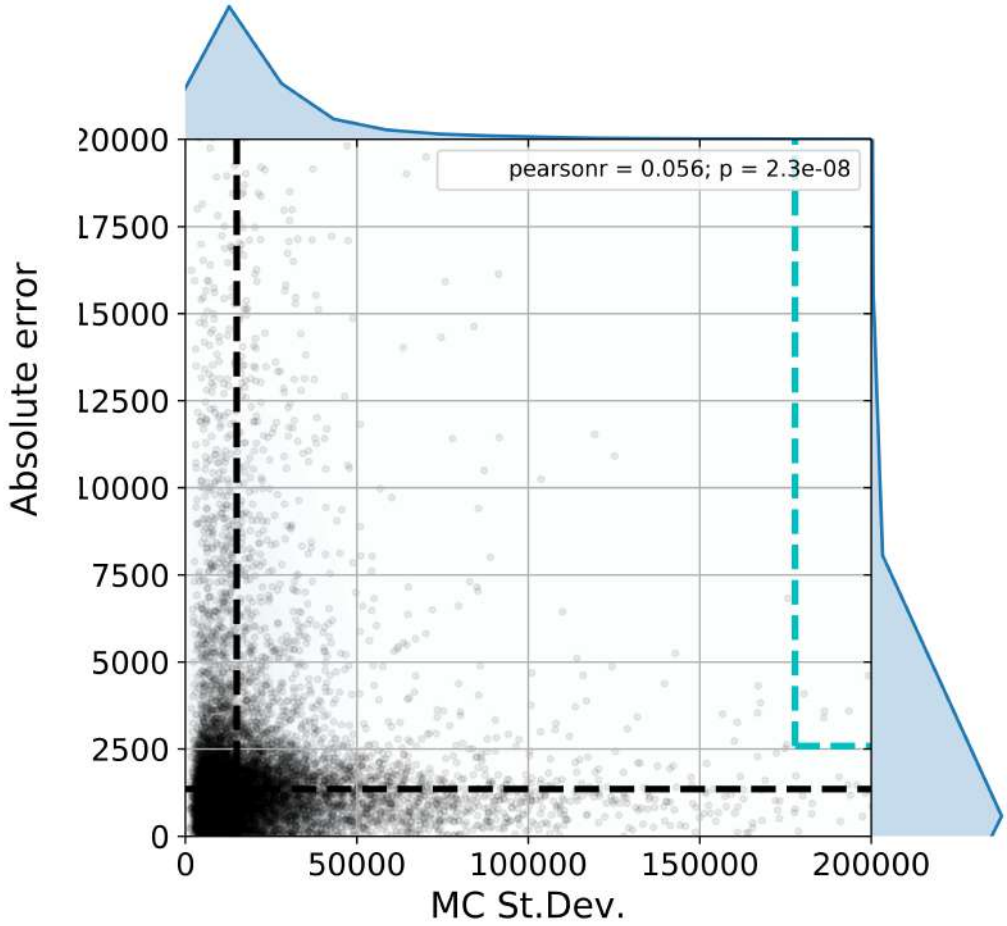


FIGURE 6.5: Scatter plot shows the relation between the MC standard deviation and the absolute error for test samples. The black dashed lines correspond to the medians of distributions, the vertical blue line corresponds to the 0.99 percentile of the MC standard deviation distribution, while the horizontal blue line shows the median percentile of the absolute error distribution of corresponding samples, which is equal to 0.783 in this case. Five-layer neural network with a 256-128-64 structure was used on the Online News Popularity dataset (Fernandes et al., 2015). The Pearson correlation coefficient equals to 0.056, thus showing no linear relation between the absolute error and the MC standard deviation.

6.3.1 Uncertainty Estimation for Regression Datasets

Similarly to Jain et al. (2019), we run a series of experiments on various regression datasets, see Appendix D.2 for the full list of datasets. We show the resulting distributions of log-likelihood values for each dataset in Figure 6.10. We observe that either DPP or k-DPP always show the best results. Most importantly, DPP works very well already for a small number of stochastic passes $T = 10$ and consistently has low variance, which is extremely important for practical usage.

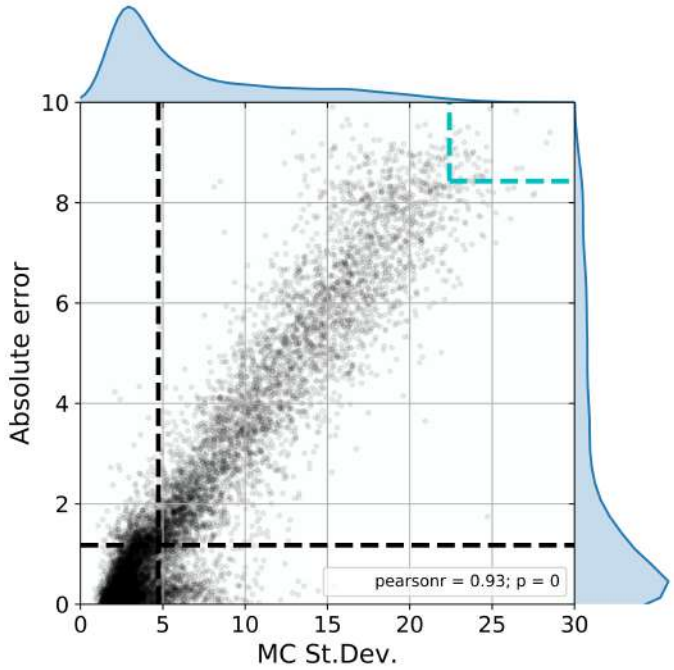


FIGURE 6.6: Scatter plot shows the relation between the MC standard deviation and the absolute error for test samples. The black dashed lines correspond to the medians of distributions, the vertical blue line corresponds to the 0.99 percentile of the MC standard deviation distribution, while the horizontal blue line shows the median percentile of the absolute error distribution of corresponding samples, which is equal to 0.975 in this case. Five-layer neural network with a 256-128-64 structure was used on the CT slices dataset (Graf et al., 2011). The Pearson correlation coefficient equals to 0.93, thus showing an almost linear relation between the absolute error and the MC standard deviation, so if we choose a sample with a relatively high MC standard deviation, it will probably have large absolute error.

We also performed an experiment with out-of-distribution (OOD) data. To generate OOD data we pick a random feature and split the data into the training set and OOD set by the median value on this feature. The experiments were run for 5 different splits. For OOD data, good uncertainty estimates should have in average higher values compared to in-domain data. Tables 6.7, 6.8, 6.9 provide the percentages of OOD points with UE values higher than α percentile of UE distribution for training data ($\alpha = 80\%, 90\%, 95\%$). The resulting numbers should be considered with a significant grain of salt due to their high variance but still DPP and k-DPP show the best results based on average values.

6.3.2 Ensembling for Regression Datasets

We consider ensembles of 5 models and combine them with the different inference methods for individual models. We visualize the log-likelihood

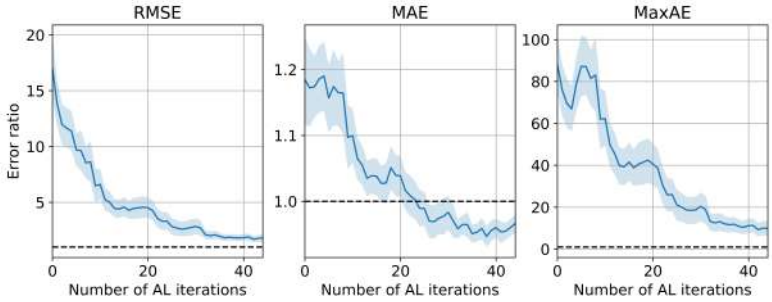


FIGURE 6.7: Comparison of MCDUE-based algorithm and random sampling algorithm: the ratio of errors (training curves) for various metrics on the KEGG Network dataset (Shannon et al., 2003). A ratio bigger than 1 (dashed line) shows the superiority of the MCDUE-based algorithm. The blue line shows the mean over 25 experiments, the standard deviation is also shown. One can see that the proposed algorithm outperforms the random sampling significantly on RMSE and MaxAE metrics.

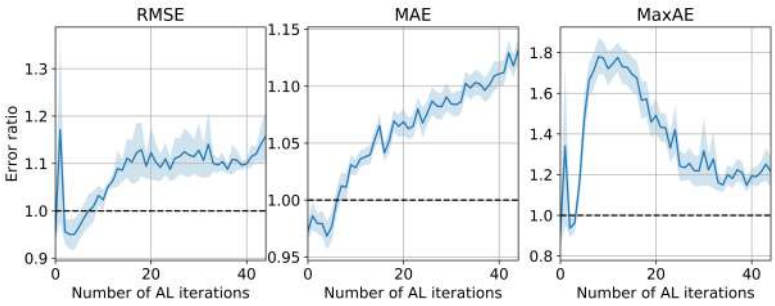


FIGURE 6.8: Comparison of MCDUE-based algorithm and max-min sampling algorithm: the ratio of errors (training curves) for various metrics on KEGG Network dataset (Shannon et al., 2003). A ratio bigger than 1 (dashed line) shows the superiority of the MCDUE-based algorithm. The blue line shows the mean over 25 experiments, the standard deviation is also shown. One can see that the proposed algorithm slightly outperforms the max-min sampling across all the metrics by up to 20%.

metric for each dataset, see Figure 6.11. There is no single method that gives the best results uniformly over the considered datasets, yet DPP-based methods show superior performance more often than other approaches. Also, it is clearly seen that pure ensembling without sampling in individual models is usually inferior even to the plain MC dropout, while the combination of ensembling with sampling consistently improves the quality of uncertainty estimation.

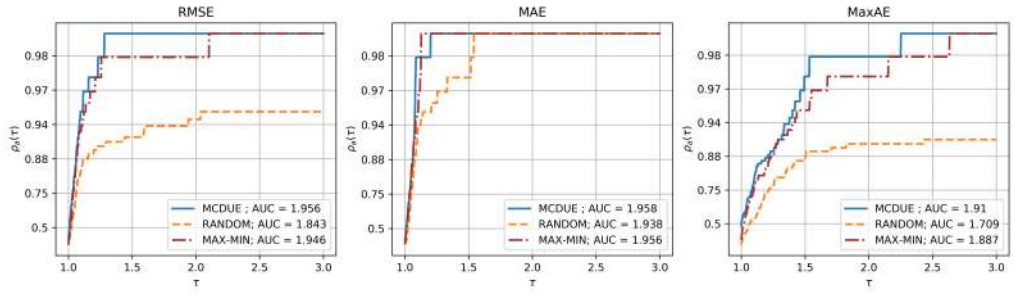


FIGURE 6.9: Dolan-More curves for various metrics and acquisition functions. Figures on the legend indicate the area under the curve (AUC) metric for each algorithm. The number of training samples was chosen randomly from 1000 to the 20% of the training set, the number m of points to sample from the pool $\mathcal{P}$ was chosen randomly from 100 to 1100, with a 140 experiments conducted in total. The MCDUE-based approach outperforms the random sampling and is slightly better than batch max-min sampling.

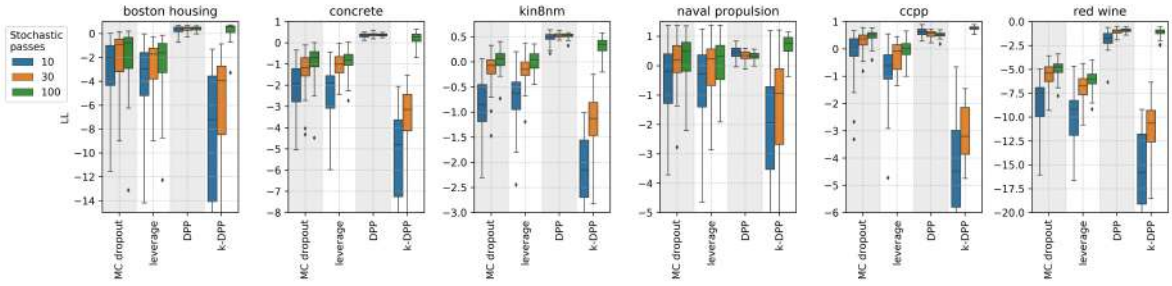


FIGURE 6.10: Log-likelihood metric across various UCI datasets for NN UE models with a different number of stochastic passes $T = 10, 30, 100$. DPP and k-DPP give better results compared to other methods with DPP working well already for $T = 10$ and consistently showing lower variance.

TABLE 6.7: Percentages of OOD points with UE values higher than specified percentile of UE distribution for training data for *concrete* dataset. DPP and k-DPP show the best results based on average values (top-2 average values are put in **bold**). For all the methods, $T = 100$.

percentile	MC dropout	leverage	DPP	k-DPP
80	55.0±27.6	61.3±27.7	**70.4**±26.0	**71.9**±28.0
90	46.0±30.7	52.9±30.8	**59.6**±30.1	**60.8**±33.7
95	40.6±32.1	46.5±33.1	**52.1**±32.9	**51.8**±36.3

6.3.3 Regression Datasets: NN Configurations Effect

In order to testify UE approaches on slightly different settings of NN architecture, we settled out three more experiments with variations in architecture, activation function, and dropout rate; for the details on it as well as the

TABLE 6.8: Percentages of OOD points with UE values higher than specified percentile of UE distribution for training data for *Boston housing* dataset. DPP and k-DPP show the best results based on average values (top-2 average values are put in **bold**). For all the methods, $T = 100$.

percentile	MC dropout	leverage	DPP	k-DPP
80	49.6±26.9	68.1±23.7	**69.2**±29.3	**83.1**±23.2
90	36.9±27.9	53.6±26.9	**59.6**±31.5	**63.7**±29.4
95	28.2±26.7	40.7±30.0	**53.5**±32.5	**50.9**±36.0

TABLE 6.9: Percentages of OOD points with UE values higher than specified percentile of UE distribution for training data for *red wine* dataset. DPP and k-DPP show the best results based on average values (top-2 average values are put in **bold**). For all the methods, $T = 100$.

percentile	MC dropout	leverage	DPP	k-DPP
80	50.4±26.9	53.1±22.3	**73.5**±23.9	**60.9**±26.5
90	36.6±28.0	39.1±23.5	**61.8**±29.2	**45.0**±31.1
95	27.3±27.7	30.1±22.6	**51.0**±31.9	**34.7**±34.2

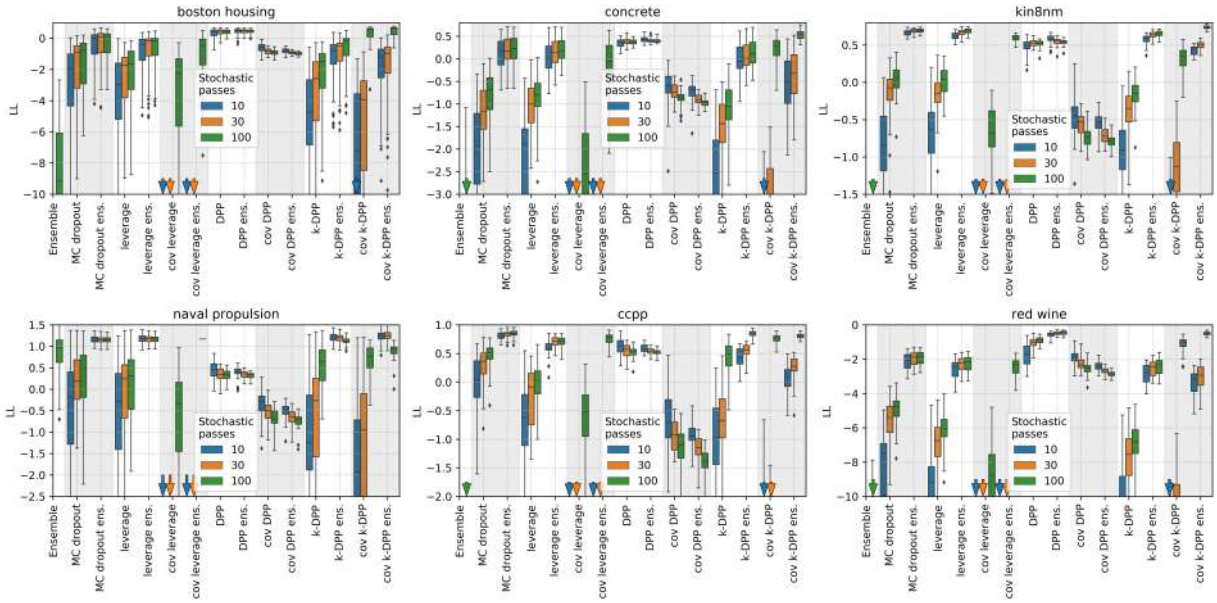


FIGURE 6.11: Log-likelihood across various UCI datasets for single models and ensembles of NN UE models. Arrows on the bottom indicate box plots being below the bottom boundary. DPP constantly shows good results in the single model scenario, being not far from ensemble-based methods.

introduction of additional metrics, please refer to Section 5.3.2.

We have visualized the results for other experiments in Figures 6.12, 6.13, and 6.14. DPP-based methods show the best performance for the majority of cases. For the very large (relative to the size of the datasets) NN architecture,

the leverage score-based approach shows promising performance as well.

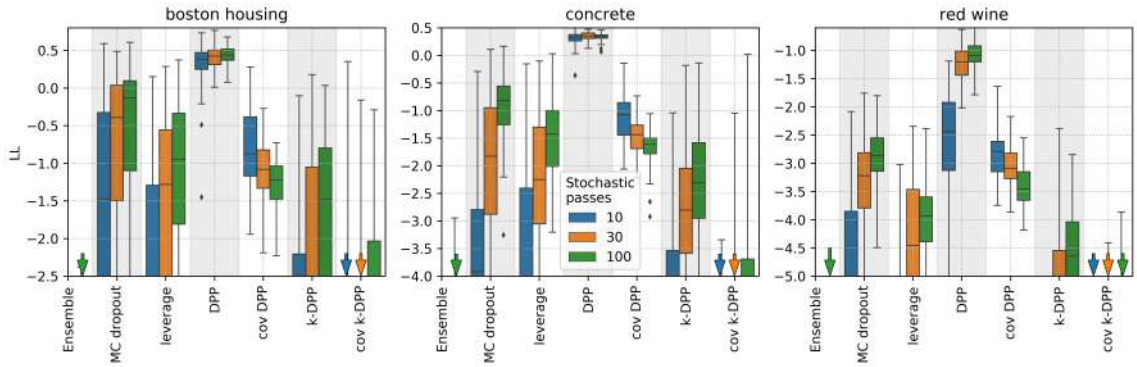


FIGURE 6.12: Log-likelihood across various UCI datasets for single models of NN UE for the small-NN experiment B, see Section 5.3.2 for setup details. DPP shows outstanding performance; it also demonstrates the most stable results in terms of the variance between the runs. Arrows on the bottom indicate box plots being below the bottom boundary.

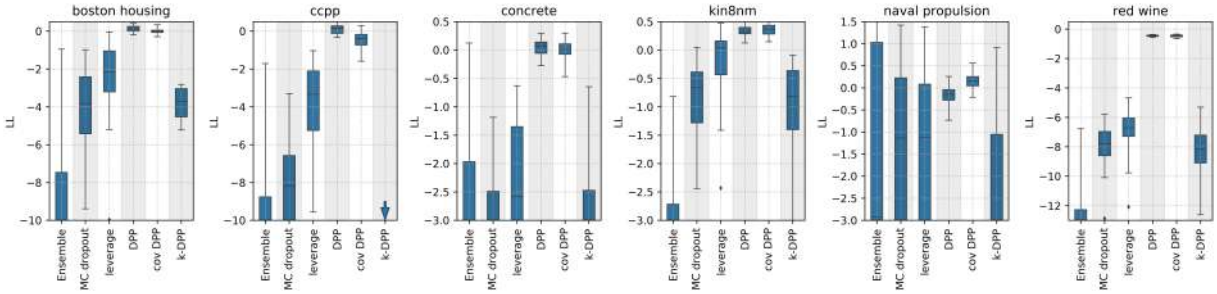


FIGURE 6.13: Log-likelihood across various UCI datasets for single models of NN UE for the large-NN experiment C with the reduced dropout rate, see Section 5.3.2 for setup details. A larger number of stochastic runs (30, 100) demonstrate the performance inferior to the 10 runs approach. DPP and k-DPP methods show stable dominance over other approaches. Arrows on the bottom indicate box plots being below the bottom boundary.

6.4 Enhancement with Gaussian Processes

In this section, we are presenting the results of the numerical experiments on the NN+GP approach described in Section 5.2.3. For the details on the setup and dataset processing please refer to Section 5.3.2. Please also refer to the Appendix E for the experiments with the hydraulic simulator. Most of these experiments were published in Tsymbalov et al. (2019).

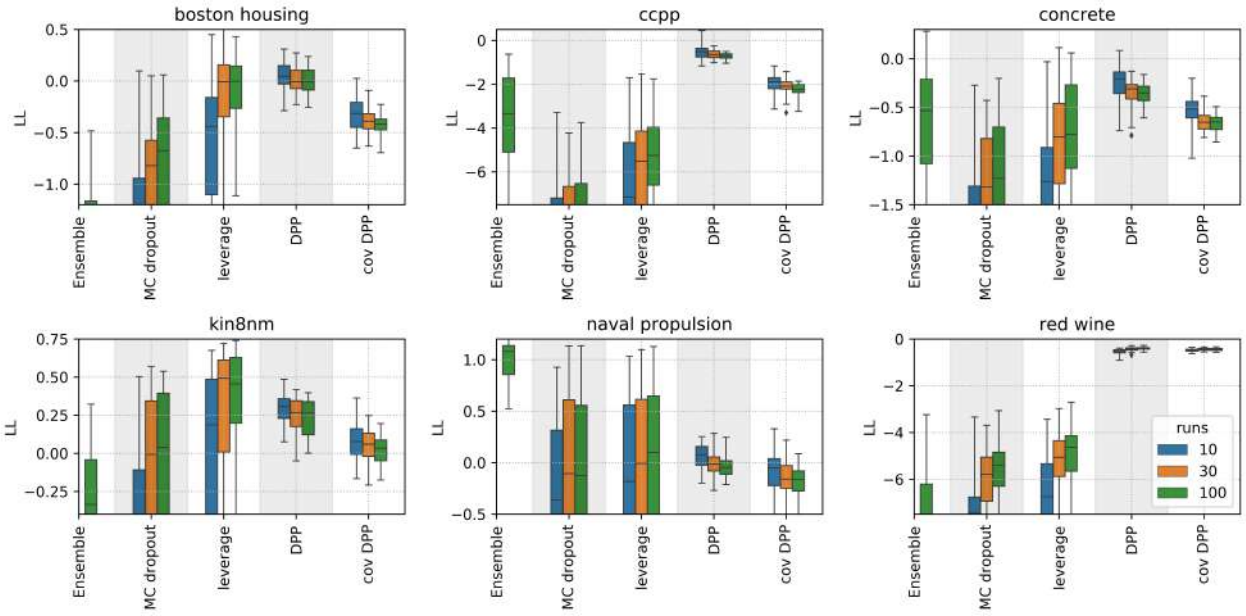


FIGURE 6.14: Log-likelihood across various UCI datasets for single models of NN UE for the large-NN experiment D, see Section 5.3.2 for setup details. DPP and leverage mask decorrelation shows the best results. Arrows on the bottom indicate box plots being below the bottom boundary.

6.4.1 Airline delays dataset and NCP comparison

We start the NNGP experiments by comparing the proposed approach with the one based on uncertainty estimates obtained from a Bayesian neural network with *Noise Contrastive Prior (NCP)*, see Hafner et al. (2018). Following this paper, we use the airline delays dataset (see Hensman et al. (2013)).

The results for the test set are shown in Figure 6.15. The proposed NNGP approach demonstrates a comparable error with respect to the previous results and outperforms (on average) other methods in the continuous active learning scenario.

6.4.2 Performance curves for UCI dataset

To compare the performance of the algorithms across the different data sets (see Section D.1 of Appendix D) and different choices of training samples, we constructed the Dolan-More curves for the errors of approximation for considered problems after the 16th iteration of the active learning procedure, see Figure 6.16. We see that the NNGP and M-step NNGP procedures are superior in terms of RMSE compared to MCDUE and random sampling.

6.4.3 SchNet training

To demonstrate the power of our approach, we conducted a series of numerical experiments with the state-of-the-art neural network architecture in

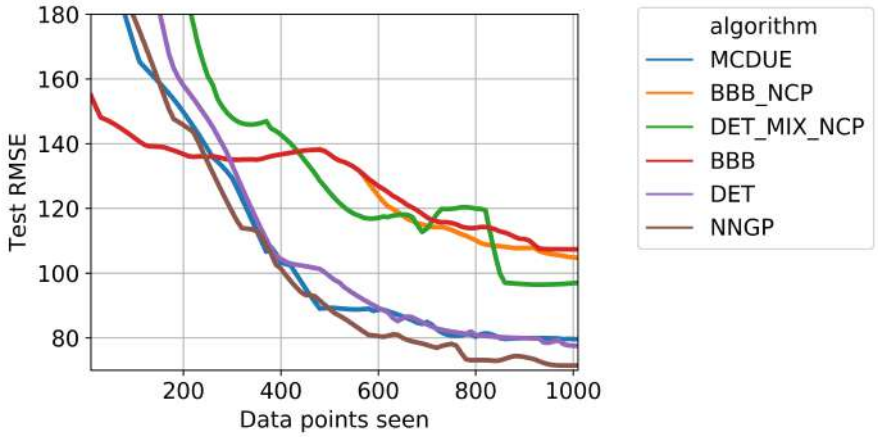


FIGURE 6.15: Root mean squared errors as functions of active learning iteration for different methods on the Airline delays data set. Plots show the median of the errors over 25 runs. NNGP initially has a much higher error, but shows the rapid improvement and becomes the best method near iteration 300.

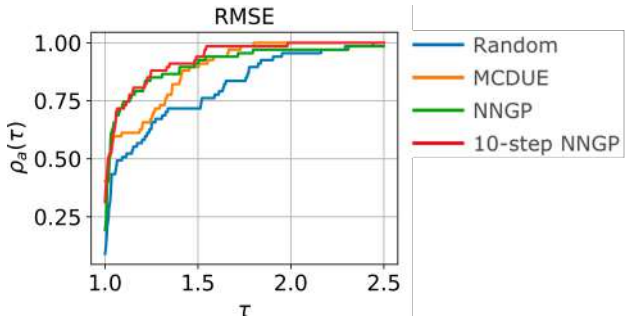


FIGURE 6.16: Dolan-More curves for UCI datasets and different active learning algorithms after 16 active learning iterations. Root mean squared error (RMSE) on an independent test set is considered. NNGP- and M-step NNGP-based algorithms show better performance compared to MCDUE and random sampling.

the field of chemoinformatics "SchNet" (Schütt et al., 2018). This network takes information about an organic molecule as an input, and, after special preprocessing and complicated training procedure, outputs some properties of the molecule (like energy). Despite its complex structure, SchNet contains fully connected layers, so it is possible to use a dropout in between them.

We tested our approach on the problem of predicting the internal energy of the molecule at 0K from the QM9 data set (Ramakrishnan et al., 2014), and conducted two experiments on training. For the details on the SchNet modification, data splitting, and differences in experiments, please refer to Section 5.3.2.

The results are shown in Figures 6.17. Both NNGP and MCDUE approaches demonstrate a steady decrease in error and are superior to the random sampling. Such improvement is very significant in terms of the time

savings for the computationally expensive quantum-mechanical calculations. For example, to reach the RMSE of 1 kcal/mol (a state of the art error reported in Schutt et al. (2017)) starting from the SchNet trained on 10 000 molecules, one need to additionally sample 15 000 molecules in case of random sampling or just 10 500 molecules using the NNGP or MC dropout uncertainty estimation procedure.

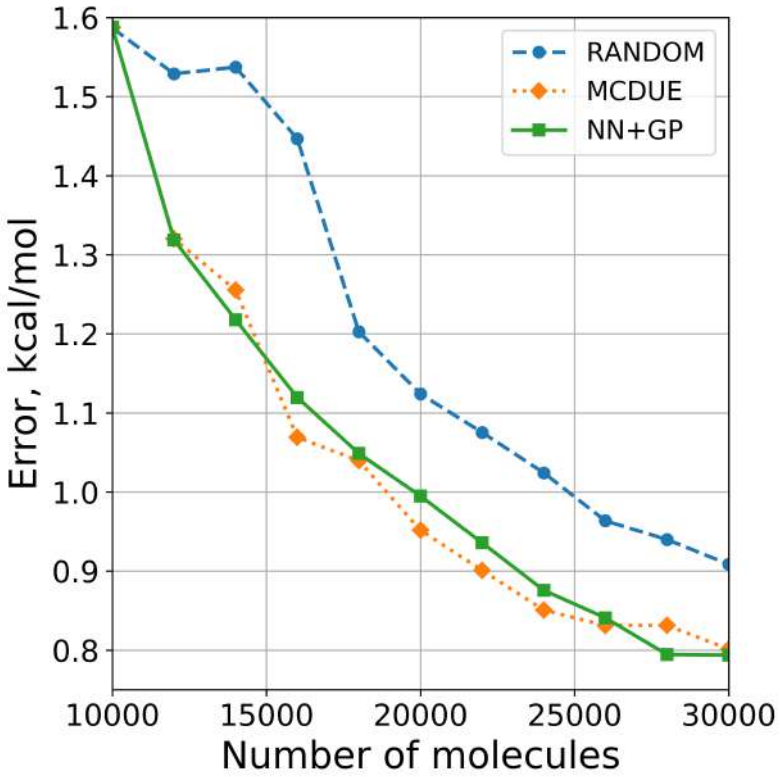


FIGURE 6.17: Training curves for the active learning scenario for SchNet (see Section 5.3.2 for details). Starting from 10 000 random molecules we pick 2 000 based on the uncertainty estimate. Dropout-based algorithms result in the 15% decrease in mean absolute test error compared to random sampling. The results are averaged over three independent runs with random choices of the starting 10 000 molecules.

6.5 Discussion

This section was dedicated to the results of numerical experiments on machine learning models. We cover not only the performance of the models for elastic strain engineering but also access the accuracy of various uncertainty estimation and active learning approaches on a separate set of problems.

In Section 6.1.1, we examine NN-based models for a silicon crystal. Numerical experiments on PBE data demonstrate that the error for the neural network is significantly lower than one for the baseline methods (Lagrange

interpolation, GBR, RFR). We also show that in the case of GW data, one can increase the accuracy by utilizing predictions produced by the PBE model in $\Delta - ML$ fashion. Another important point is that mixing the different datasets (one on the large subspace of three-dimensional data and another on the smaller subspace of six-dimensional data) increases accuracy for both tasks, paving the way for a combination of results produced for different simulations.

In next Section 6.1.2, we address the question of performance of the derived CNN model in comparison to the NN and KRR baselines. While NN shows slightly better results for the prediction of a single property (e.g., bandgap E_g or Γ gap), it shows worse results for the tasks of band structure prediction and conduction band minima classification. This is also demonstrated for the advanced problem of effective mass estimation, where the model should both locate CBM correctly and then produce accurate values for the estimation of the second derivative provided by finite differences. Therefore, the CNN model is more preferable, although it is possible to use a simpler NN if one aims to predict only a single feature, such as the bandgap value. We also show how the active learning for CNN works in a case of PBE data, where it outcompetes the random sampling baseline, and discovered that a simple MCDUE approach is inapplicable here. For the GW data, we demonstrate a moderate accuracy improvement with a learning curve using NN-GP approach.

The rest of the section is dedicated to the general active learning and uncertainty estimation for neural networks. Section 6.2 presents the results of a simple dropout-based active learning, and also reveals possible problems with its use. We enhance the MCDUE baseline in the next Section 6.3, where the improvement caused by the use of diversification of dropout masks via DPPs is demonstrated on a number of tasks. We would like to note that not all of these methods were used for the ESE problems; this research direction is left for the future work.

While MCDUE (even with the diversification add-on) may show acceptable results for the uncertainty estimation, various active learning caveats, such as sampling of close points within a single batch, do exist. We address them in Section 6.4, where the approach of post-processing with Gaussian processes is proposed. It is tested on a number of problems, including the interaction with simulators in Sections 6.4.3 and Appendix E, where it shows promising results.

To sum up, all of the proposed approaches are (or close to) state-of-the-art in MC dropout-based uncertainty estimation for neural networks. However, the general mechanism of obtaining both robust and simple UE is unknown and yet to be explored. Similarly, a question of how to organize active learning for the case of NN is mostly unanswered, yet there is a certain rise of interest to this topic in ML community.

As for the ESE, numerical experiments demonstrate the ability of developed models to mimic the DFT calculations within a reasonable accuracy level; this statement is further checked in the next section, where proposed machinery is used to explore the strain space, and selected results were

checked and confirmed by a separate *ab initio* calculations.

7 Strain-induced properties

This section represents the quintessence of this work – namely, results and insights discovered by the use of high-throughput machine learning models. We will first provide the readers with the description of the bandgap "envelope" – a density plot that represents the many-to-many relation between the bandgap and elastic strain energy density, defined in Section 4.2.2. This will be followed by the detailed analysis of the bandgap topology in the strain space, with examples of the distinguished strains. On top of that, we will provide an example of how this machinery may be used for assisting *in situ* experiment.

This part of the research was carried out in close collaboration with Zhe Shi and other coauthors.

7.1 Bandgap optimization and "envelope"

7.1.1 Silicon crystal case

The many-to-many relation between $h(\varepsilon)$ and the bandgap $E_g(\varepsilon)$ is shown in Figure 7.1. In the stress-free equilibrium state, silicon has a bandgap of 1.1 eV; with an increase in strain energy density, a variety of possible bandgaps emerge. Even silicon with as little strain energy density as 0.2 meV/Å^3 can become quite a different material from the stress-free silicon. As h further increases, the largest allowable bandgap drops and an "envelope" forms, as evidenced by the change of maximal and minimal bandgap reachable under a fixed h. The shading of the envelope regions in Figure 7.1 reflects the distribution of the available bandgap. A darker shading qualitatively indicates that the number of possible strains to achieve a specific bandgap at a given h is higher. Outside the envelope, the shading color is white, meaning that the corresponding bandgap is not attainable.

An upper-envelope function E_g^{upper} and lower-envelope functions E_g^{lower}, defined in 4.4 and rendered as the black and red dotted lines in Figure 7.1 indicate the path to obtain the fastest change in E_g. For instance, if the goal is to reduce the bandgap of silicon from 1.1 eV as fast as possible, with the least cost of elastic energy, the red-dotted line in Figure 7.1 offers the best design of the strain tensor to achieve this goal. This is further detailed on Figure 7.2, which illustrates that silicon's "fastest path to metallization" is actually a curved path in the strain space: the initial fastest-descent direction for E_g (at $h = 0$) is quite different from when E_g hits zero, and linear perturbation theory such as the deformation potential theory (Bardeen and Shockley,

1950) is not expected to work well in deep-strain space. At 1.35 meV/Å^3 the bandgap of Si can vanish, corresponding to the minimum energy required for semiconductor-to-metal transition in the whole 6D strain space (open red circle on the horizontal axis of Fig. 7.1). The deformation case is

$$\begin{aligned} \varepsilon_1 = -5.4928\%,\ \varepsilon_2 = 2.416\%,\ \varepsilon_3 = 1.3348\%, \\ \varepsilon_4 = 1.1057\%,\ \varepsilon_5 = -1.096\%,\ \varepsilon_6 = 0.5024\%. \end{aligned} \tag{7.1}$$

Both fastest descent path and zero bandgap strain were found using the differential evolution optimization algorithm by Storn and Price (1997), which was launched on the deep ESE model. The zero bandgap strain was verified by a separate GW calculation; its electronic band structure is shown in Figure 7.3.

Our deep ESE model found within experimentally accessible strain range that the indirect-to-direct bandgap transition takes place in silicon in the high h region and a minimum strain energy density h_d^{min} around 15.4 meV/Å^3 exists for the direct bandgap to appear. The little red direct bandgap "island" of DOD (density of direct bandgaps, see 4.2.2) can be achieved by applying $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 \geq 9.3\%$.

We will discuss the other ways to reach the zero bandgap in the Section 7.2, yet there is only a single way to obtain the maximum bandgap of 1.24 eV reachable by strained silicon, which is to apply a hydrostatic tensile strain of 6.5%. It should be noted that silicon strained to such an extent can nearly reach the maximum theoretical efficiency, known as the Shockley–Queisser limit (Ruhle, 2016) of a single p-n junction solar cell, demonstrating the possible application of ESE in solar energy conversion devices.

7.1.2 Diamond crystal case

The DOB plot for the case of a diamond crystal is shown in Figure 7.4. The maximal bandgap reachable is 6.5 eV, which could be realized by the hydrostatic compression of 10%. The lowest energy density cost for reaching the direct bandgap is h = 20 meV/Å^3 at the (1.1%, 2.4%, 0.6%, -3.0% , -3.3%, 2.7%) strain. A complete ranking of the common crystal directions regarding their ability to reducing the bandgap can be found in the Appendix G.

In the case of a diamond, deep ESE provides an opportunity to reduce its bandgap to a level comparable to that of InAs. Our results propose that by straining diamond in the most optimal way, it can be transformed to mimic the properties of a lower-bandgap semiconductor while almost preserving its own uniqueness such as high strength and thermal conductivity, thereby paving the way for designing hitherto unexplored combinations of material characteristics.

While the direct bandgaps in the strain space could be relatively easily located and described as a distinct subregion for the silicon crystal case, the situation is more tricky for the diamond crystal. An exploratory study conducted in Section 4.3.3 suggests that there may exist a 5-dimensional manifold within the 6-dimensional manifold of admissible strains; this, however,

does not say anything on how to locate or visualize the manifold. One of the insights drawn from both data and model is that one needs to use the shear components of the strain; we visualize 1000 most distinct indirect bandgap cases and 150 most distinct direct bandgap cases (to conserve the relative share) in Figure 7.5.

The conventional way to modulate electronic properties in semiconductors is the so-called compositional grading technique. Through varying the stoichiometry of a semiconductor through, for example, molecular beam epitaxy, a graded bandgap can be produced (Capasso, 1988). This means of tweaking the material property is conceptually based on traditional chemical alloying, whereby the chemical composition is tuned in an alloy melt to produce desirable strength or ductility. Invoking this approach, conventional bandgap engineering resorted to chemical alloying such as $GaAl_{1-x}As_x$ or $Ga_{1-x}In_xAs$. However, we have demonstrated here that the stress-free situation is usually not the optimal state for a figure-of-merit, and elastic strains allow the bandgap to exhibit many more possible values so that each pure material candidate should occupy a much larger hyperspace enabled through the achievable 6D strain space. The more general bandgap engineering approach should utilize gradients in both composition and strain to achieve the desired band alignment.

We also data-mined the 6D strain space to study the conduction related properties and the elastic strain energy density against ε. Here, we adopted our ML model to acquire the many-to-many relation between conductivity effective mass for the conduction electron $m^{\star}_{cond}(\varepsilon)$ and $h(\varepsilon)$, as shown in Figure 7.6a. The values of scalar $m^{\star}_{cond}$ are obtained by averaging individual longitudinal and transverse effective masses, as in Van Zeghbroeck (2010):

$$m^{\star}_{cond} = \frac{3}{\frac{1}{m_{11}} + \frac{1}{m_{22}} + \frac{1}{m_{33}}}. \tag{7.2}$$

The purple shading in Figure 7.6a reveals the distribution of the available $m^{\star}_{cond}$, with darker shading implying more strains are able to reach a specific value of $m^{\star}_{cond}$ at a given h. In principle, by using our model, we can accurately predict any components of the $m^{\star}$ tensor and their arithmetic averages for every **k**-point at every strain level.

In the design of photovoltaic cells and scintillators, it is desirable to adopt a semiconductor material with a direct bandgap and small effective mass to allow for a combining high light yield and conductivity. When ESE is used to modulate the bandgap and effective mass together, a lower elastic strain energy density is often preferable than a higher energy density for reaching the same property design. However, in our case of materials properties optimization, the best solution that simultaneously minimizes all objectives (namely E_g, $m^{\star}_{cond}$, and h) does not exist. Instead, we found out Pareto-efficient solutions that cannot be better off (decreased) in any of the three values without worsening off (increasing) at least one of the other two values. As shown in Figure 7.6b, the 3D Pareto front of minimized E_g, $m^{\star}_{cond}$, and h indicates a trade-off must be made in simultaneously having a small

bandgap and conductivity effective mass, where h could increase up to more than 120 meV/Å^3. One cannot achieve, for example, a near-zero bandgap and $m^\star_{cond} < 0.25 m_e$ without paying a considerable price in h by deforming diamond, as indicated by the "infeasible region" in Figure 7.6b. Also, one can usually find higher h values that correspond to the same (E_g, $m^\star_{cond}$) combination. The strain cases with such h values are in the "feasible region" in Figure 7.6b. In addition, if one would like to access to all possible combinations of (E_g, $m^\star_{cond}$) achieved by straining diamond and to find the lowest elastic strain energy density ($h_{\min}$) for each combination, Figure 7.6c could be a blueprint for this purpose. Note that it is not a 2D projection of the 3D Pareto front of Figure 7.6b where only minimized E_g and $m^\star_{cond}$ are present.

7.2 Bandgap topology

7.2.1 Silicon crystal case

As was shown above, the effect of different deformations on the bandgap value may be degenerate. Here we choose silicon crystal as an example to demonstrate the power of our deep ESE model in investigating the bandgap topology and bandstructure-related physics. While a set of the reachable bandgap values in the full 6D strain space does not allow for an explicit visualization, we can explore the accessible electronic bandgap range directly if we restrict ourselves to tensile and compressive strains ($\varepsilon_4 = \varepsilon_5 = \varepsilon_6 = 0$). Figure 7.7 illustrates the Si bandgap isosurfaces, i.e., the set of points in the strain space where the bandgap is equal to some given values for different levels obtained by our high-throughput NN model. Since both the crystal structure and deformation tensor have some symmetries, and the bandgap as a function of strain is invariant with respect to some of them, the isosurfaces appear to have the shape of a convex polyhedron for every strain having the following symmetric structure:

1. The μ and χ points lie on the $\varepsilon_1 = \varepsilon_2 = \varepsilon_3$ line. We thus denote their coordinates by (a, a, a) and (b, b, b), respectively.

2. The α_j $(j = 1, 2, 3)$ points form a regular triangle which lies in a plane orthogonal to the $\varepsilon_1 = \varepsilon_2 = \varepsilon_3$ line. Their coordinates are denoted by (c, d, d), (d, c, d), and (d, d, c), respectively.

3. The β_j $(j = 1, 2, 3)$ points also form a regular triangle which lies in a plane orthogonal to the $\varepsilon_1 = \varepsilon_2 = \varepsilon_3$ line. Their coordinates are denoted by (e, e, f), (f, e, e), and (e, f, e), respectively.

The shape of the isosurface is similar for both PBE and GW bandgaps, although the specific strain values may differ for the same PBE and GW bandgap levels. It was found that the easiest way (with the least $h(\boldsymbol{\varepsilon}^{\mathbf{3D}})$) to obtain the 0-eV bandgap without any shear strain is to apply a normal strain of -3.86 and 4.36% along any two of the three $\langle 100 \rangle$ directions while

leaving the third $\langle 100 \rangle$ direction undeformed. Therefore, there are six strain cases that are equivalent, as indicated by red dots in Fig. 7.9. The position of the vertices of the E_g isosurface in the strain space is the function of selected bandgap value, and the detailed relationship between the bandgap and the strains is shown in Fig. 7.10.

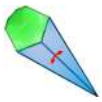
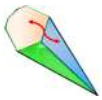
The formation of the E_g isosurfaces, such as the ones in Fig. 7.7, is due to the relative position of the valence band maximum (VBM) and the conduction band minimum (CBM). Despite different shape variations of the two energy bands, modulating elastic strain provides possibilities for the VBM and CBM to differ by the same amount with respect to the vacuum level. For undeformed silicon with a bandgap of 1.1 eV, the VBM is located at the Γ point, and the CBM lies on the straight line (the Δ-line) in the **k**-space and is positioned at about 85% of the way from the Brillouin zone center to the zone boundary (Jenkins, 1956). Under 3D deformation, the cubic crystal symmetry of Si is lifted and we follow the **k**-point labeling scheme explained in the Appendix B to describe band extrema positions. It is found that VBM remains at Γ for both silicon and diamond crystal irrespective of deformation, whereas the position of CBM can be greatly affected by external strains. Using the geometry of the E_g isosurface as a visualization tool, we identify four types of **k**-space transition in CBM that may happen across the ridgelines on the isosurface.

Starting with the strain points on the lower faces separated by μ-α_j ridgelines of the E_g isosurface in Fig. 7.7, we found that the CBM retains roughly the same relative position along the "Δ"-type line as in the undeformed case, and that crossing the ridgelines only switches CBM among $\Delta_1 = (0, k_1, k_1)$, $\Delta_2 = (k_1, 0, k_1)$, and $\Delta_3 = (k_1, k_1, 0)$ where $k_1 \approx 0.425$. In other words, μ-α_1 ridgeline corresponds to Δ_2/Δ_3 transition, μ-α_2 ridgeline corresponds to Δ_1/Δ_3 transition, μ-α_3 ridgeline corresponds to Δ_1/Δ_2 transition, and we can indeed label each carapace by its CBM character $\Delta_1, \Delta_2, \Delta_3$. We term this transition occurring in the small strain region as the Δ-switching. In this case, the linear deformation potential theory can be used to describe the strain effects on the band extremum (Bardeen and Shockley, 1950). However, investigation of the large deformation points on its upper faces in Fig. 7.7 reveals that the CBM would not retain its location and major changes would happen.

Our ML model captures the occurrence of "$L - \Delta$" transition across the $\beta_i - \alpha_j$ ridgelines where the CBM changes to "L" points in **k**-space: $L_1 = (0.5, 0, 0)$, $L_2 = (0, 0.5, 0)$, $L_3 = (0, 0, 0.5)$; see Fig. 7.11A and B, where, for example, "Δ_3 carapace" changes to "L_1 carapace" across the $\alpha_1 - \beta_3$ ridgeline, and "Δ_3 carapace" changes to "L_2 carapace" across the $\alpha_2 - \beta_3$ ridgeline. None of the ridgelines or carapaces (e.g., Δ_3 carapace bound by $\mu - \alpha_1 - \beta_3 - \alpha_2 - \mu$) are truly flat. The large, nonperturbative deformation makes the conventional theory ineffective in predicting it. Moving further toward χ in the strain space, CBM would remain at L and a cross-over of the $\chi_2 - \beta_j$ ridgelines is referred to as an L-switching. Indirect-to-direct bandgap transition occurs near the upper tip of the paleolith-like isosurface where CBM appears at Γ, as shown in Figure 7.11C. This can be explained by the competition between drops of different band edges. In general, as strain increases, the band

edge at both Γ and L would decrease. As a result of high strains, the energy decrease at Γ is faster and eventually the bandgap becomes direct, as shown in Figure 7.11D. In this case, we transition, for example, from the L_1 carapace ($\alpha_1 - \beta_3 - \chi_3 - \chi_2 - \beta_2 - \alpha_1$ in Figure 7.7) to "Γ carapace" ($\chi_1 - \chi_2 - \chi_3 - \chi_1$ in Figure 7.7) across the $\chi_2 - \chi_3$ ridgeline. When the strained Si turns into a direct-bandgap semiconductor, it will exhibit a significant enhancement in its optical transitions around the fundamental adsorption edge compared with an undeformed Si, due to the elimination of phonon involvement to facilitate adsorption or emission. As absorbance increases exponentially with thickness in a material, a solar cell based on direct bandgap Si with a high adsorption coefficient would require much less thickness to absorb the same amount of light, paving the way for the design of lightweight high-efficiency solar cells. Table 7.1 summarizes all of the details of the **k**-space transitions, thus resolving the conduction band properties exhaustively for a wide range of strains.

TABLE 7.1: **k**-space CBM transitions. Each of 12 separating edges of the polyhedron is tabulated. The constants k_1 and k_2 are approximately equal to 0.425 and 0.5, corresponding to points on Δ and L, respectively.

Type		Plane 1 ↔ Plane 2	**k**-coordinate of CBM
'Δ'-switching		$\alpha_2\beta_1\mu \leftrightarrow \alpha_2\beta_2\mu$	$(k_1, k_1, 0) \leftrightarrow (0, k_1, k_1)$
		$\alpha_3\beta_2\mu \leftrightarrow \alpha_3\beta_3\mu$	$(0, k_1, k_1) \leftrightarrow (k_1, 0, k_1)$
		$\alpha_1\beta_3\mu \leftrightarrow \alpha_1\beta_1\mu$	$(k_1, 0, k_1) \leftrightarrow (k_1, k_1, 0)$
'L'-switching		$\beta_1\beta_3\chi \leftrightarrow \beta_2\beta_3\chi$	$(k_2, 0, 0) \leftrightarrow (0, 0, k_2)$
		$\beta_2\beta_3\chi \leftrightarrow \beta_1\beta_2\chi$	$(0, 0, k_2) \leftrightarrow (0, k_2, 0)$
		$\beta_1\beta_2\chi \leftrightarrow \beta_1\beta_3\chi$	$(0, k_2, 0) \leftrightarrow (k_2, 0, 0)$
'L-to-Δ' transition		$\beta_1\beta_2\alpha_2 \leftrightarrow \alpha_2\beta_1\mu$	$(0, k_2, 0) \leftrightarrow (k_1, k_1, 0)$
		$\beta_1\beta_2\alpha_2 \leftrightarrow \alpha_2\beta_2\mu$	$(0, k_2, 0) \leftrightarrow (0, k_1, k_1)$
		$\beta_2\beta_3\alpha_3 \leftrightarrow \alpha_3\beta_2\mu$	$(0, 0, k_2) \leftrightarrow (0, k_1, k_1)$
		$\beta_2\beta_3\alpha_3 \leftrightarrow \alpha_3\beta_3\mu$	$(0, 0, k_2) \leftrightarrow (k_1, 0, k_1)$
		$\beta_1\beta_3\alpha_1 \leftrightarrow \alpha_1\beta_1\mu$	$(k_2, 0, 0) \leftrightarrow (k_1, k_1, 0)$
		$\beta_1\beta_3\alpha_1 \leftrightarrow \alpha_1\beta_3\mu$	$(k_2, 0, 0) \leftrightarrow (k_1, 0, k_1)$
Indirect-to-direct E_g transition		$\beta_1\beta_3\alpha_1 \leftrightarrow \chi_1\chi_2\chi_3$	$(k_2, 0, 0) \leftrightarrow (0, 0, 0)$
		$\beta_1\beta_2\alpha_2 \leftrightarrow \chi_1\chi_2\chi_3$	$(0, k_2, 0) \leftrightarrow (0, 0, 0)$
		$\beta_2\beta_3\alpha_3 \leftrightarrow \chi_1\chi_2\chi_3$	$(0, 0, k_2) \leftrightarrow (0, 0, 0)$

7.2.2 Diamond crystal case

For the case of a diamond crystal, the corresponding bandgap isosurface in $\varepsilon_{xx} - \varepsilon_{yy} - \varepsilon_{zz}$ space exposes less interesting structure, as there are the conduction band minima only and no transition to the direct bandgap, see Figure 7.12(b-d) for the details. Key features of this bandgap isosurface in 3D include carapaces that are piecewise smooth, ridgelines where two smooth carapaces meet, and corners where three ridgelines meet. The multifaceted

nature of the isosurfaces is attributed to the switch of the CBM **k**-space position. As a consequence of strain tensor and crystal symmetries, this isosurface has the following strain and electronic properties:

- Three carapaces labeled as Δ_1, Δ_2, and Δ_3 correspond to strain cases with the same value of indirect bandgap but different CBM positions of (0, 0.375, 0.375), (0.375, 0, 0.375), and (0.375, 0.375, 0), respectively.
- Three ridgelines labeled as r_1, r_2, and r_3 with the strain relations $\varepsilon_{yy} = \varepsilon_{zz}$, $\varepsilon_{xx} = \varepsilon_{zz}$, and $\varepsilon_{xx} = \varepsilon_{yy}$, respectively.
- The corner μ is the intersection of r_1, r_2, and r_3 is the most "tensile" hydrostatic strain point on the bandgap isosurface, i.e., $\varepsilon_{xx} = \varepsilon_{yy} = \varepsilon_{zz}$.

However, the "shear-only" strain space $\varepsilon_{xy} - \varepsilon_{yz} - \varepsilon_{xz}$ shows more promising picture, see Figure 7.12e. Besides three different indirect bandgap CBM positions at X_1:(0, 0.5, 0.5), X_2:(0.5, 0, 0.5), and X_3:(0.5, 0.5, 0), 3D shear strains can also give rise to direct bandgap in diamond where CBM is at the Γ point. The change from the carapace labeled as X_1 to the carapace labeled as Γ thus indicate an indirect-to-direct bandgap transition in a diamond (yellow arrow in Figure 7.12e). The corresponding band structures are shown in Figure 7.12f and g, respectively.

7.3 Diamond metallization case

In this section, we will briefly demonstrate the use case of the proposed ML machinery for the simulation of the *in situ* experiment described in Banerjee et al. (2018). This part of the research was mostly carried out by Zhe Shi and Ming Dao. The top-level scheme of the experiment is shown in Figure 7.13: diamond nanoneedles grown on silicon substrate are bent by the diamond nanoindenter tip. Different parts of the diamond nanoneedle are subject to the different deformation; one can use the finite element modeling to estimate the strain in the different points (as done in the original paper). Moreover, based on the obtained values of strain, we can estimate the local bandgap via the machine learning model.

The results of this multistage simulation suggest that a metallization occurs, see the bottom right part of Figure 7.13. The corresponding deformations were double-checked by the independent first-principles calculations. The elastic strain energy density values, accessible via the experimental setup, are thus estimated to be near 80-85 meV/Å^3, which is a bit larger than the suggested lower bound, presented in Figure 7.4, yet is more robust and explainable. For further details, see Shi et al. (2020).

The abovementioned experiment with the FEM-ML coupling demonstrates the applicability of the proposed ML scheme to the real-world engineering problems and scientific discoveries. This part of the research was mostly carried out by Ming Dao and Zhe Shi.

7.4 Discussion

This chapter effectively demonstrated ESE-related use cases for the ML machinery developed throughout the work. Brand new knowledge of the nature of silicon and diamond crystals under strain is brought into existence by the combined effort of *ab initio* calculations and machine learning.

DOB plots in Section 7.1 shows a general map of ESE capabilities in the E_g–h plane, unveiling possible (and most probable, in terms of randomly sampled strain) values of bandgap for the silicon and diamond crystals. These values show that the options of widening the bandgap are limited, yet one could effectively reduce the bandgap by controlling the deformation in a special way.

We would like to point out that presented optimization results (of zero and direct bandgaps) were double-checked by separate calculations. However, theoretically, more optimal values, omitted by the ML machinery, can exist. Moreover, due to the limitations of the underlying first-principles calculations, namely a limited $\mathbf{k}$-point mesh, the proposed values are actually upper-bound estimates of the theoretical bandgap.

Section 7.2 is dedicated to a deeper analysis of the electronic band structure, namely, conduction band minima transitions, that are studied and visualized on the example of reduced shear-free strain space ($\boldsymbol{\varepsilon}^{\mathbf{3D}}$). This kind of theory was before addressed in a smaller scale (e.g. considering uniaxial or biaxial strains) and usually by means of $\mathbf{k} \cdot \mathbf{p}$ perturbation theory, which considers the band structure at the vicinity of critical points only. Another option is to use imprecise PBE calculations, with the results being far from the ground truth. In this work, we made it possible on the level of GW calculations, which are much closer to the experimental results. To put this into computational perspective, Figure 7.7 requires 8 000 000 (200^3 calculations to produce (this number may be decreased by one order of magnitude if symmetries are taken into account) and took less than a minute to both calculate and render on a laptop using the ML surrogate model. We would like to note that while this part of the research was limited to the 3D case for the visualization properties, a more comprehensive description in 6D space is yet to be described, and the corresponding ML machinery for it is already developed.

The last part of the chapter, Section 7.3 demonstrates how developed models may be used to support simulations of real-life experiments on the elastic strain engineering. We use FEM to access the deformation on the particular parts of the nanoneedle; however, our model (due to the features of underlying *ab initio* calculations) is focused on the bulk materials and does not takes into account the surface effects, which may alter the values a lot, especially in the diamond case, see Nie et al. (2019) for a detailed discussion. Nevertheless, it is still a solid approximation to start from. This part of the research was mostly carried out by Zhe Shi and Ming Dao.

This section concludes the main part of the thesis. In the next chapter, an extended summary of the work will be provided together with the outlined extensions of current research planned for future work.

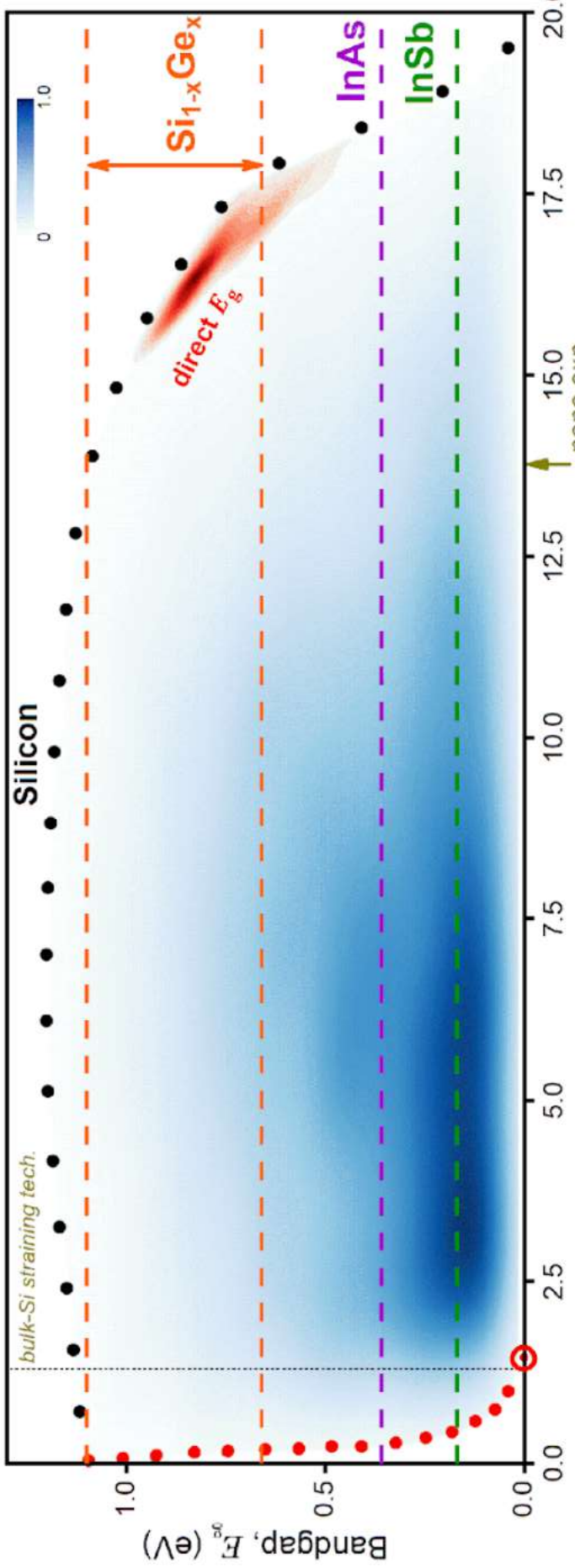


FIGURE 7.1: Reachable bandgap values for various h within the whole deformation space for silicon. The region where the strained silicon has a direct bandgap is colored in red. The circle at $h = 1.35$ meV/Å^3 indicates the lowest energy penalty for the semiconductor-to-metal transition. The arrow on the horizontal axis indicate reachable h by the *in situ* experiments (Zhang et al., 2016). Red circles indicate the fastest bandgap decrease straining path.

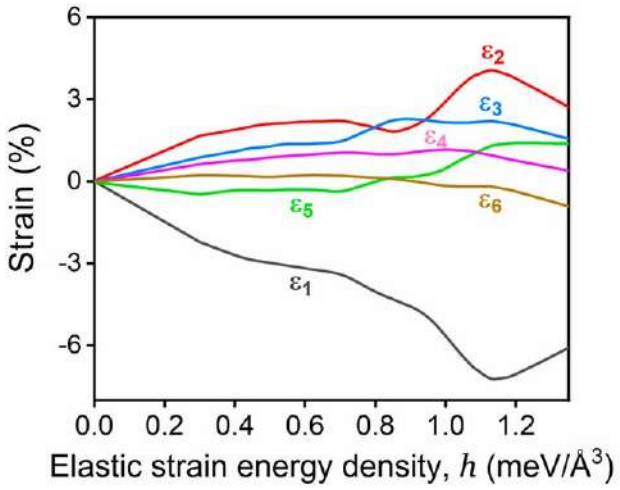


FIGURE 7.2: The most energy-efficient strain pathway to reach the zero-bandgap state (end of $\varepsilon_1 - \varepsilon_6$ curves), i.e., the lower-envelope function $E_g^{\text{lower}}(h)$ in silicon corresponding to the red-dotted line in Fig. 7.1. The zero-bandgap state (open red circle on the horizontal axis of Fig. 7.1) corresponds to the deformation case of ε_1 = –5.4928%, ε_2 = 2.416%, ε_3 = 1.3348%, ε_4 = 1.1057%, ε_5 = –1.096%, and ε_6 = 0.5024% in [001],[010],[001] coordinate frame.

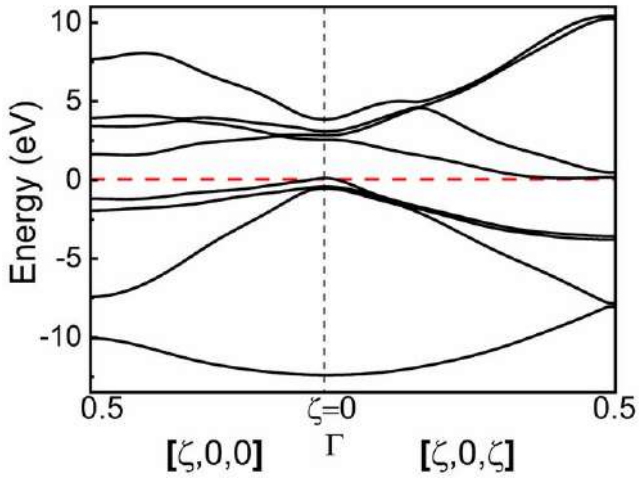


FIGURE 7.3: GW band structure associated with deformation 7.1. The fractional coordinates for the three high-symmetry points along the selected **k**-path are (0.5, 0, 0), (0, 0, 0), and (0.5, 0, 0.5), respectively.

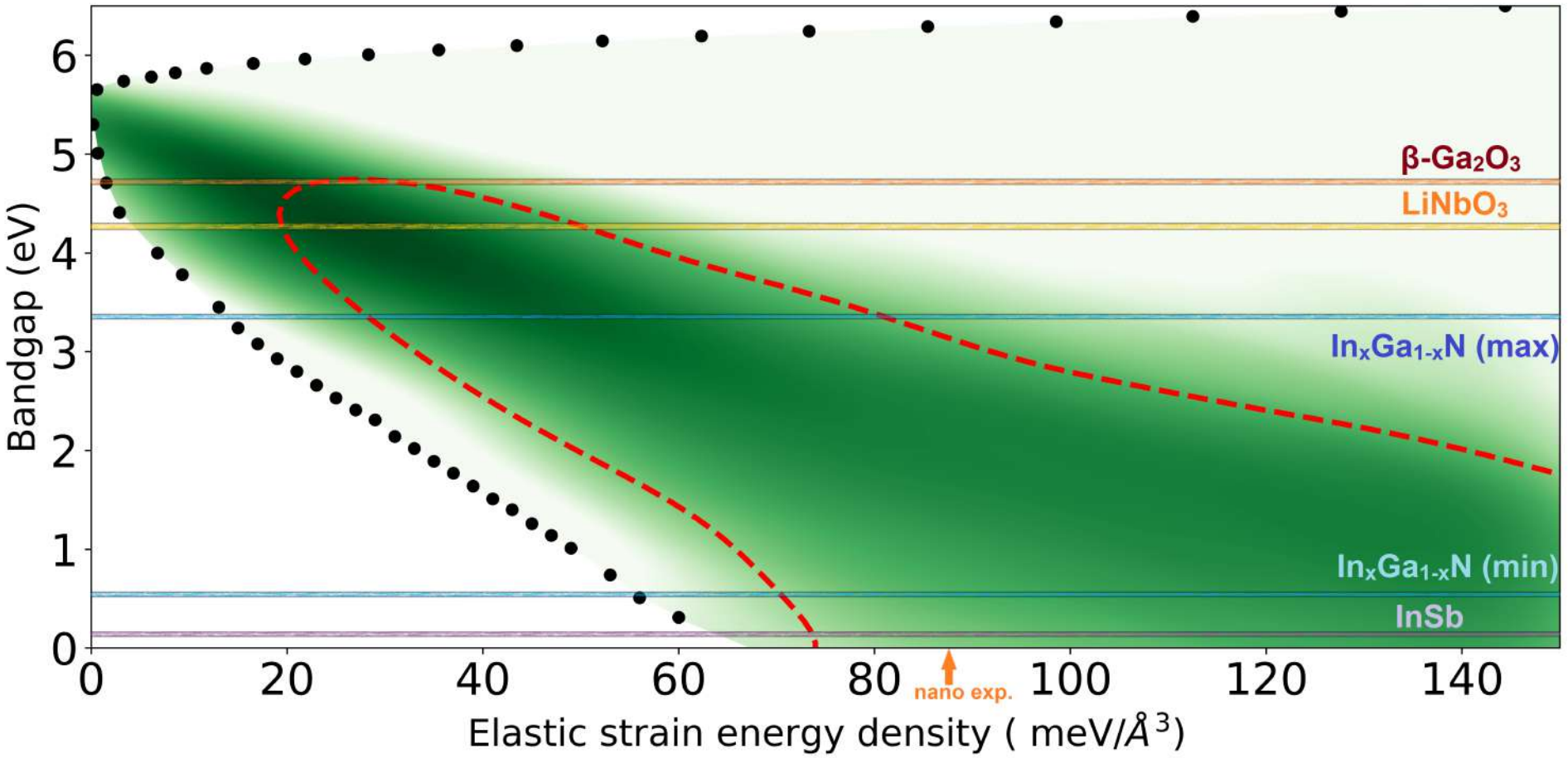


FIGURE 7.4: Reachable bandgap values for various h within the whole deformation space for diamond crystal. The boundary of the strained crystal having a possible direct bandgap is denoted by the red dashed line. The arrow on the horizontal axis indicate reachable h by the *in situ* experiments (Banerjee et al., 2018).

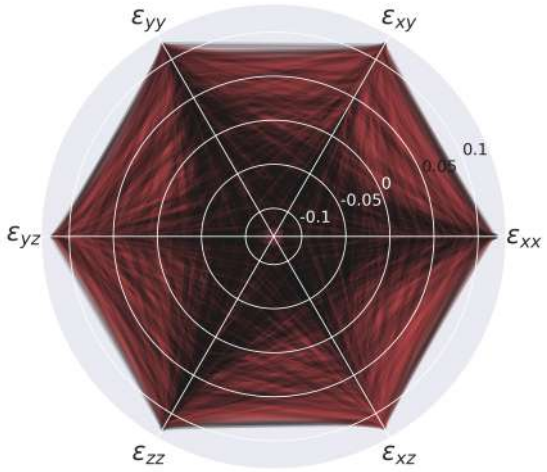


FIGURE 7.5: Indirect and direct bandgap cases for the diamond crystal. Every Green-Lagrange strain is represented here as a hexagon with vertices on the corresponding axes. Black hexagons (filled the background) correspond to random 6D strains; red hexagons ($\sim$ 15% in amount) correspond to the direct bandgap strains generated by our ML model. White circles denote the axis ticks in polar coordinates.

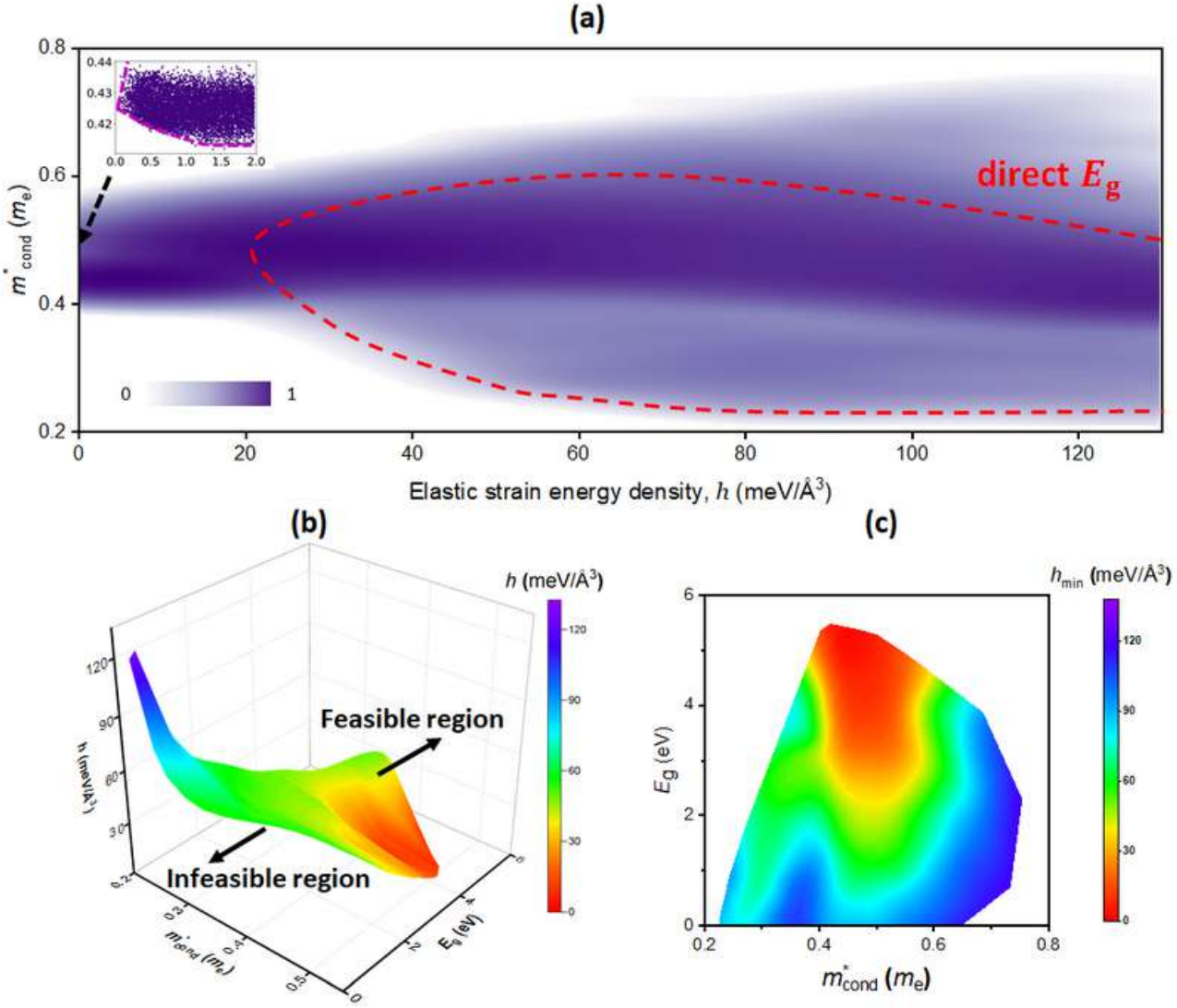


FIGURE 7.6: ML of electron effective mass. **(a)** Distribution and density of states of conductivity effective mass. Strain region where direct bandgap may appear is bound by the red dashed line. Inset is the zoomed-in plot near $h = 0$ of the $m^\star_{cond}$ distribution. **(b)** Pareto front for minimizing $m^\star_{cond}$, bandgap, and h. The color contours indicate different elastic strain energy densities h. Points within the Pareto front are feasible while those beyond the Pareto front (under the colored surface) are infeasible. **(c)** Color contour plot of the lowest elastic strain energy density (hmin) required for achieving any combinations of bandgap and $m^\star_{cond}$. (c) contains more (E_g, $m^\star_{cond}$) points and is not a 2D projection of (b).

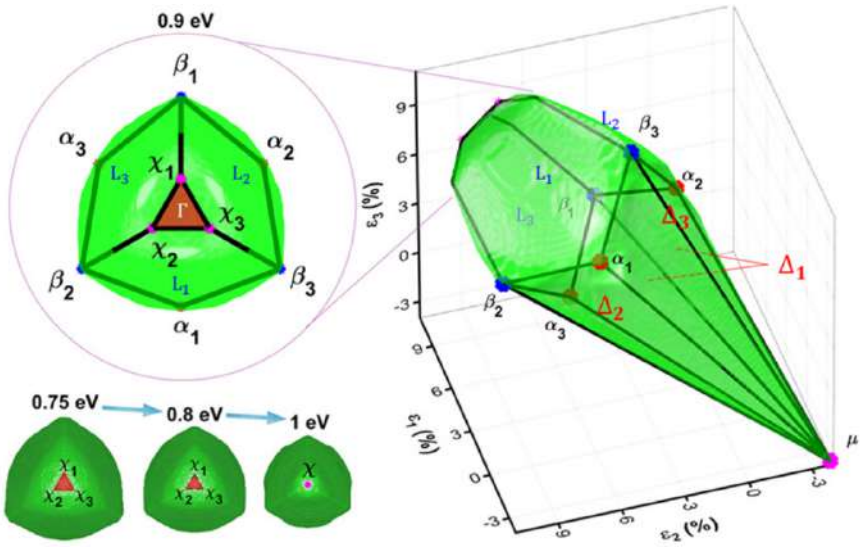


FIGURE 7.7: Bandgap isosurfaces for silicon in the $\varepsilon_1\varepsilon_2\varepsilon_3$ strain space appear to have the paleolith shape for every bandgap level. The main corners $(\chi, \mu, \alpha_j, \beta_j)$ of an isosurface at E_g = 0.9 eV are indicated by different colors and the "carapaces" are distinguished by their associated **k**-space CBM labels. The red triangular faces indicate the direct-bandgap region at different E_g levels. As bandgap increases, the area for the red triangle eventually shrinks to a single χ point.

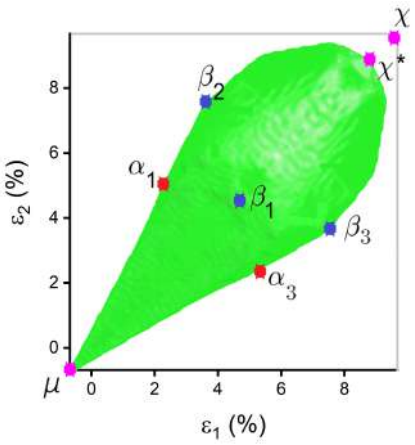


FIGURE 7.8: Bandgap isosurface shown through the $\varepsilon_1 - \varepsilon_2$ projection of Si at 1 eV level. The χ point corresponds to the direct-bandgap case and it splits into three at small E_g as shown in Figure 7.7.

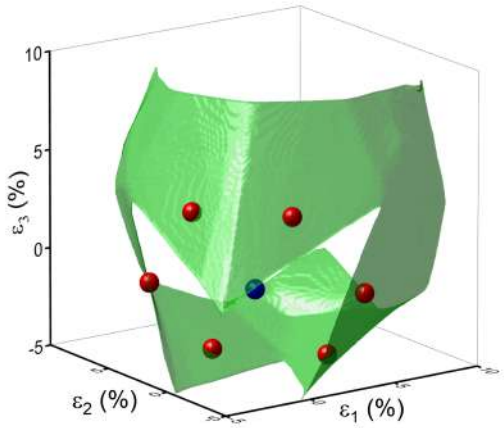


FIGURE 7.9: Zero-bandgap isosurface in the strain space. The blue point corresponds to the strain-free state; red points are strains with the least h of 1.65 meV/Å on this isosurface.

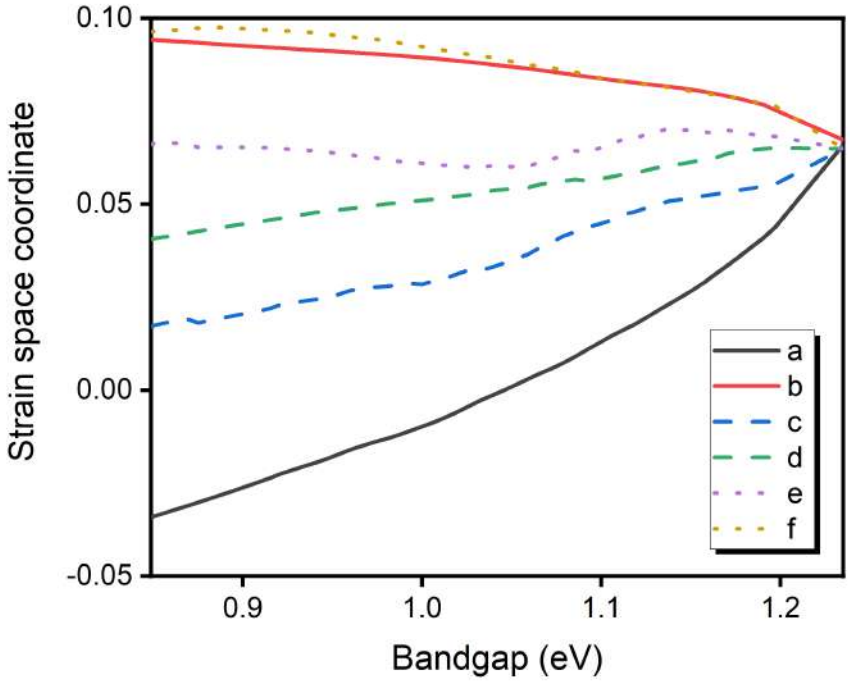


FIGURE 7.10: Strain-space coordinates of the bandgap isosurface corners (defined as in Figure 7.7) as a function of the bandgap level. The maximum bandgap possible in this strain space is about 1.24 eV, and it is reached at the triaxial strain of 6.5%. In the cases where three χ-type points exist, b equals the average coordinate of them. See Section 7.2.1 for the detailed description of the presented values.

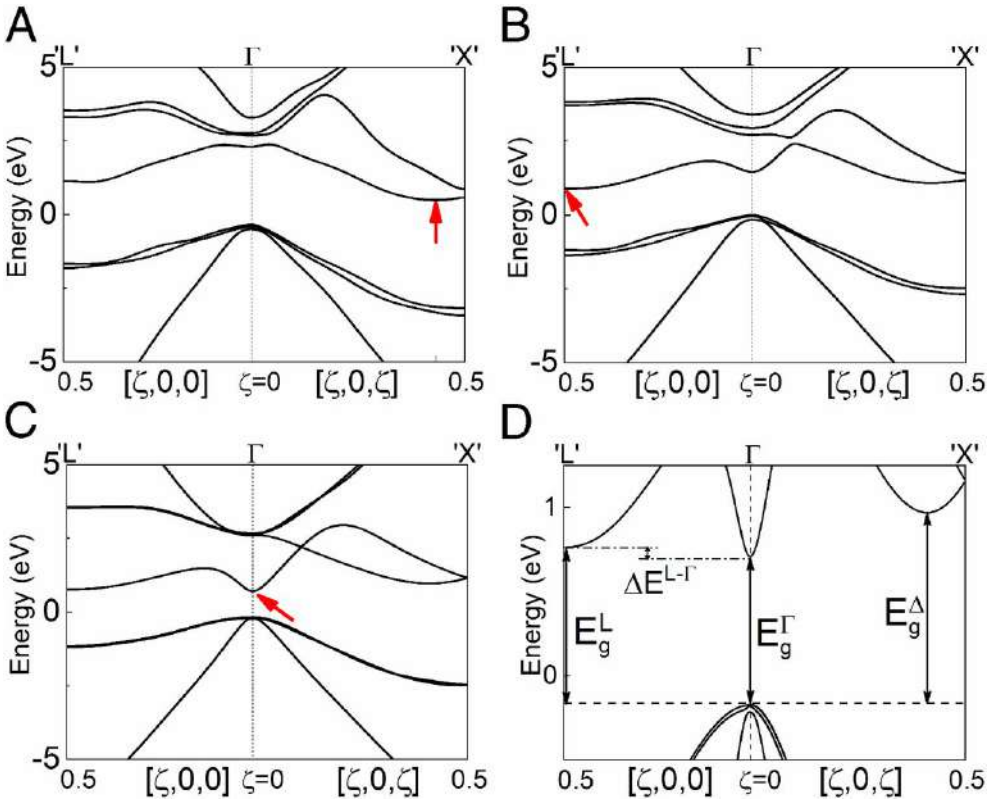


FIGURE 7.11: Illustration of **k**-space transition in Si predicted by our model. All the transitions are verified by DFT calculations with GW_0 correction. (**A** $\rightarrow$ **B**) represents the 'Δ-L' transition and (**B** $\rightarrow$ **C**) shows the indirect-to-direct transition. The CBM (red arrows) locates at **k**-point (0.433, 0.433, 0), (0.5, 0, 0), and (0, 0, 0) respectively. (**D**) The enlarged bandstructure around Fermi energy shows the competition of the three possible CBM positions. The three non-shear strain cases for (**A**-**C**) are (-0.23%, 1.84%, 3.45%), (4.63%, 8.23%, 9.22%), and (9.85%, 9.31%, 9.4%), corresponding to points on the different faces of the bandgap isosurface in Figure 7.7; this sequence of bandstructure plots represents the "journey" on the isosurface: it starts at the Δ_3 face **(A)**, then moves to the L_1 face **(B)** and ends at the Γ "tip" **(C)**.

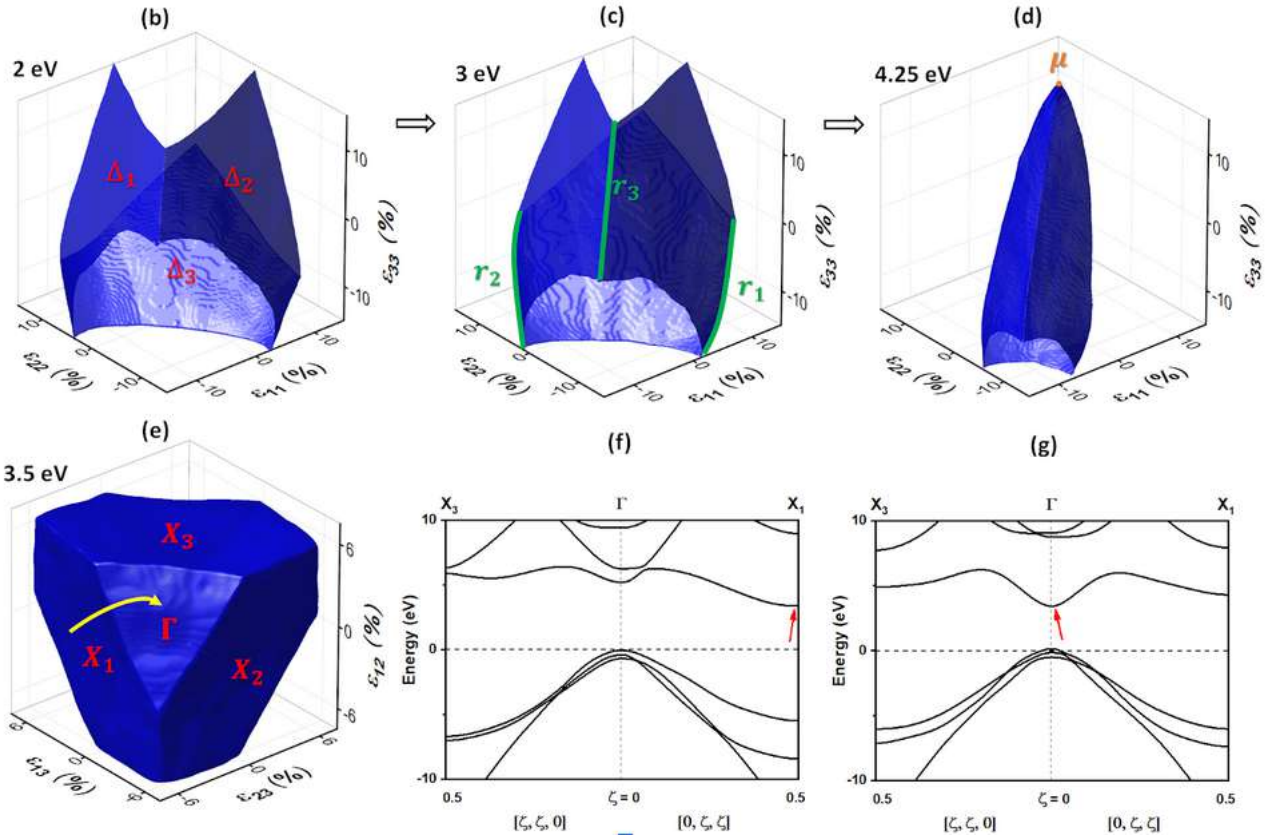


FIGURE 7.12: Bandgap isosurfaces for the diamond crystal case. (b-d) "Paleolith"-like bandgap isosurfaces in the $\varepsilon_{xx} - \varepsilon_{yy} - \varepsilon_{zz}$ (normal only) strain space at 2 eV, 3 eV, and 4.25 eV levels, respectively. The carapaces, ridgelines, and corners are indicated in red, green, and brown letters, respectively. (e) Bandgap isosurface in the $\varepsilon_{xy} - \varepsilon_{yz} - \varepsilon_{xz}$ (shear only) strain space at 3.5 eV. The yellow arrow indicates a change of carapaces on this isosurface pertaining to indirect-to-direct bandgap transition in diamond. The corresponding change of CBM k-space coordinate from X_1(0, 0.5, 0.5) to Γ (0, 0, 0) is shown in band structure plots (f) and (g), respectively. Red arrows in both plots indicate the CBM.

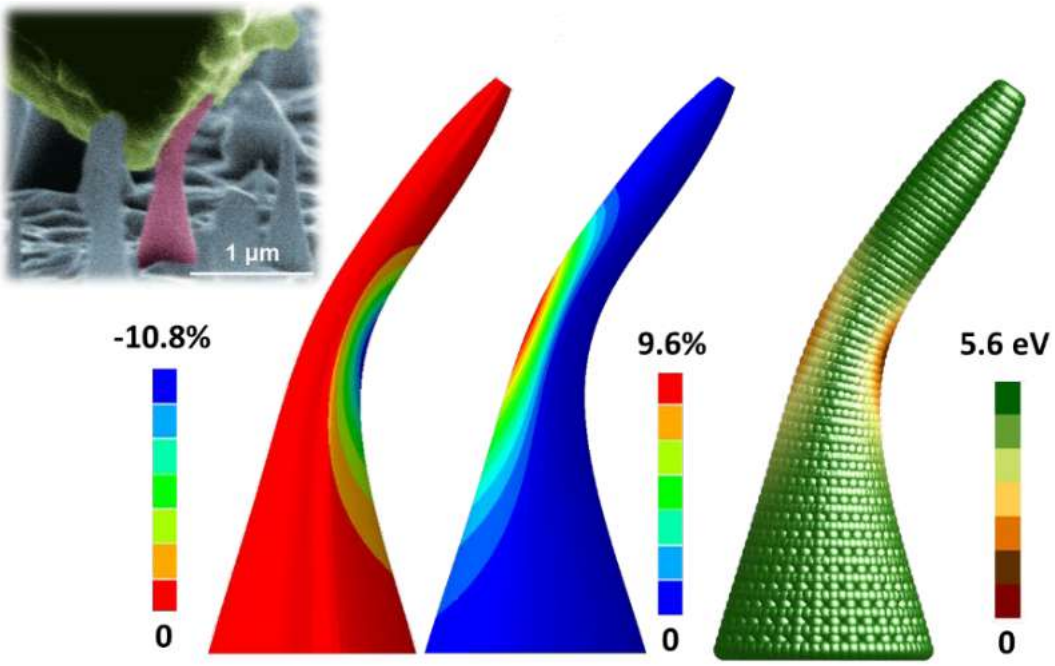


FIGURE 7.13: FEM predictions of the local compressive and tensile strain distributions and predictions of the NN and machine learning algorithm of the distribution of bandgap for a diamond nanoneedle with its <110> crystallographic direction aligned with the needle axis. The inset shows a scanning electron micrograph of the deformed nanoneedle during the bending experiment, from Banerjee et al. (2018).

8 Conclusions

8.1 Summary

In this work, an ML-assisted simulation pipeline was developed for the problems of elastic strain engineering, namely, prediction of the electronic band structure.

Elastic strain engineering offers superior performance to the existing semiconductor materials. Yet electronic properties of deformed crystals are estimated by costly *ab initio* simulations, and the thorough exploration or optimization within the vast strain space may require millions of such calculations. To cope with this problem, an approach involving a tailored NN-based surrogate model was designed, together with an intricate protocol for the interaction with the underlying *ab initio* model, which involves a combination of different data sources and active-learning-powered additional training. The resulting model is demonstrated to be both accurate compared to the initial data, and fast, reducing the calculation time by four orders of magnitude. This opens up a number of opportunities for the optimization of figures of merit (FoM), as well as the coupling with the finite-element methods.

Instead of focusing on the prediction of a given FoM, our model is designed to predict the whole electronic band structure, from which necessary properties may be easily derived. This corresponding regression to the high-dimensional domain is not only capable of predicting all the properties at once but also shows an accuracy superior to one of the specialized model in a number of tasks, such as the conduction band minima classification. The resulting data structure – the electronic band – is a subject to the number of intrinsic features and symmetries; these characteristics were thoroughly analysed and taken into account during the model design and helped to both reduce the number of parameters by adopting the CNN-based architecture and preserve the internal structure in a physically-informed ML fashion.

The specifics of data being produced by the first-principles simulations open up the opportunity for continuous learning. An important question is how to sample and how to surpass the random sampling baseline, if possible: can we query the data our model is least certain at? Therefore, an uncertainty estimation methodology based on the dropout-based Monte-Carlo sampling was proposed for the fully-connected neural networks in general, and a number of approaches to enhance the dropout baseline were suggested. The resulting methods were tested in a variety of problems, not limited to the elastic strain engineering, and showed the performance superior to the one of random sampling and other baselines. This easy-to-implement approach enhanced the surrogate model for the ESE as well, showing promising results in the active learning scenario.

Another important feature of the *ab initio* data is the possibility to compute the answer on the different levels of theory: one can adopt fast PBE-PAW calculations for an imprecise estimate of the electronic band structure, or use the costly GW approximation, which provides answers closer to experimental values and is used as a reference. This aspect of the first-principles calculations is exploited to help the data-hungry NN model to fight the lack of GW data (that takes up to three orders of magnitude more time to obtain) by pre-training on the larger amount of PBE-PAW data. The models trained on the aforementioned "data fusion" scenario demonstrate more robust performance compared to the models fed with the GW data only.

All the above-mentioned components were combined together to construct surrogate ML models that help in the process of exploring the vast possibilities of elastic strain engineering of silicon and diamond crystals. With the help of designed machinery, we performed an exploratory analysis of the six-dimensional strain space and discovered the indirect-to-direct bandgap transitions as well as semiconductor-to-metal transitions, together with the effort estimates required to induce these effects experimentally. Our models suggest that ESE offers a large variety of crystal states, surpassing the possibilities provided by the mere compounding of elements like $Si_{1-x}Ge_x$ or $In_xGa_{1-x}N$ in terms of bandgaps available. We also performed a deep analysis of the band structure topology and corresponding transitions induced by the strain and used our model in the FEM-assisted imitation of *in situ* experiments. Selected results discovered with the help of surrogate models were double-checked by a separate *ab initio* calculations.

In summary, this work extends the frontier of knowledge in the field of elastic strain engineering with the help of machine learning, allowing to increase the capability of existing first-principles methods by approximating them with the blazing speed. In the next section, we will emphasize the possible future directions of this work.

8.2 Future work and research extensions

8.2.1 ESE development

The applicability of the proposed approaches is not limited to the examined examples of silicon and diamond crystals. The proposed NN architecture is scalable to include more bands and larger **k**-meshes, and many handy symmetries we exploit may be easily omitted or turned off in the current implementation. For instance, in the case of ionic materials, the $E(\boldsymbol{k}) = E(-\boldsymbol{k})$ symmetry may break, and one would need to account for it in the corresponding model. Another example may include larger **k**-meshes, which may demonstrate stronger interference in the **k**-space; increasing the receptive field of the convolutional layer may help.

To further increase the accuracy, one generally needs either more data or reduce the model class in order to reduce the number of parameters. The first way is possible if one will loose the deformation tensor restrictions put in Section 2.1.3. This will lead to the many-to-many relationship between the

tensor deformation and electronic band structure, i.e., there would be many deformations corresponding to a single band structure. However, this will open many opportunities for data augmentation, as the 48-fold symmetry group will result in a tremendous amount of data. As for the other opportunity, we believe that the Gaussian Processes with the careful kernel choice (ideally accounting for all the symmetries) may lead to the superior results in most of the cases.

One of the possible future directions is connected to the finer band structures. It inevitably means more time for the *ab initio* calculation to be done yet may be necessary for certain cases of multicomponent materials. A simple yet powerful way to do it is to try a super-resolution approach (as, for instance, in Tai et al. (2017)), fitted on cheap PBE data.

As for the possible extensions in terms of the values to tune and predict, we would like to mention the phonon bandstructure fitting, deeper exploration of the calculated properties (e.g., exploring the hole conductivity and enhancement of carrier mobilities, which involves tuning more Hessians like the one for the effective mass tensor). Another desirable application is the coupling of the proposed models with the FEM simulators, which was tackled in Section 7.3 yet can be used for the broader range of materials science tasks.

Active learning, demonstrated as a proactive interaction between the surrogate model and first-principles simulator, is clearly a trend now (Podryabinkin and Shapeev, 2017; Podryabinkin et al., 2019; Smith et al., 2018). One of the options is indeed to use MLIP (Podryabinkin et al., 2019) as an additional driver for the NN: to use it between the complex NN and *ab initio* calculations, as an intermediate model that is powerful enough to provide the NN with the approximate answers instead of VASP. We have proposed three dropout-based approaches to the active learning yet tested only two of them for the problems of ESE, where only NN-GP approach showed moderate performance. One may think of the various improvements depending on the model and task at hand. However, the first and the most important possible step is taking into account the cost of future calculations; this field is known as *cost-sensitive active learning*, see, e.g., Donmez and Carbonell (2008) for details. We put effort into the development of the machine-learning style, ensemble-agnostic approach, and tested it against the simple random baselines for ESE problems. In the next section, we will briefly sum up the part of this work related to the uncertainty estimation and active learning for neural networks.

8.2.2 Active learning development

A variety of approaches to uncertainty estimation and active learning exists. This topic becomes more and more popular as the amount of data increases dramatically every year, yet the abilities for processing, cleaning, and selection are not. Another factor is the increasing popularity of machine learning as the topic for the traditionally simulation-based fields, such as fluid dynamics, geospatial processing, and others.

A simple active learning procedure may be performed using a committee of models, and this is a solid baseline in the cases of sufficient computational resources available. In Section 6.3, we demonstrated one way of further performance improvement via diverse dropout means; other traditional ways include data and model diversification.

While we ultimately avoided the thorough discussion of the Bayesian neural networks, the field is now actively developing, and one may expect BNNs to become a handy and available tool in the future with the development and popularization of corresponding libraries.

In this work, we described an easy-to-implement, ensemble-free approach to the uncertainty estimation and its basic modifications. For robust system design, one may need to derive better and more empirically straightforward criterion on whether it can trust the current prediction or not. While we mostly used the uncertainty estimates to rank the samples within a pool, a more intuitive measure, such as a confidence interval, may be more useful in practice. The simplest way is an uncertainty calibration, e.g., Platt-based one (Guo et al., 2017), can be used. In general, a better approach may be connected with the explicit or implicit tolerance to uncertainty incorporated into the loss function, for instance, a Noise Contrastive Prior loss (Hafner et al., 2018), mentioned in Section 6.4.

8.3 Author's contribution

The results presented in this work would not be possible without the extensive collaboration. This section lists the author's contribution to each research project showed in this work.

For the projects related to the elastic strain engineering, all the machine learning models (including training and active learning methodology) were designed and implemented by the author, as well as approximately half of the *ab initio* calculations and thorough data analysis. The choice of the *ab initio* setup, processing, and discussion of the results, as well as the writing the core of the corresponding articles were made in co-authorship with the Zhe Shi from Massachusetts Institute of Technology, not to mention top-level guidance from other co-authors: Alexander Shapeev (Skolkovo Institute of Science and Technology), Ju Li and Ming Dao (Massachusetts Institute of Technology) and Subra Suresh (Nanyang Technological University). As for the results presented in Section 7.3, related to the FEM-coupling for the diamond metallization exploration, the author solely performed machine-learning-based calculations (using designed models) and was partially involved in the data analysis.

As for the projects related to the dropout-based active learning for the neural networks, the author performed the model design, implementation, and run experiments solely for the cases of MCDUE-based models (Sections 5.2.1 and 6.2) and NN-GP coupling (Sections 5.2.3 and 6.4). As for the part with dropout mask diversification (Sections 5.2.2 and 6.3), the author holds credits for the early implementation of the decorrelation method and performed all the regression-related experiments. Together with the co-authors

Kirill Fedyanin and Maxim Panov (Skolkovo Institute of Science and Technology), the author extensively participated in the code debugging, data analysis, and results interpretation.

Bibliography

Aad, G., Abajyan, T., Abbott, B., Abdallah, J., Khalek, S. A., Abdelalim, A. A., Abdinov, O., Aben, R., Abi, B., Abolins, M., et al. (2012). Observation of a new particle in the search for the standard model higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 716(1):1–29.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Akinwande, D., Brennan, C. J., Bunch, J. S., Egberts, P., Felts, J. R., Gao, H., Huang, R., Kim, J.-S., Li, T., Li, Y., et al. (2017). A review on mechanics and mechanical properties of 2D materials—graphene and beyond. *Extreme Mechanics Letters*, 13:42–77.

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., et al. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 472:473.

Alaoui, A. and Mahoney, M. W. (2015). Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, pages 775–783.

Amblard, P.-O., Barthelmé, S., and Tremblay, N. (2018). Subsampling with k determinantal point processes for estimating statistics in large data sets. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 313–317. IEEE.

Ash, J. T. and Adams, R. P. (2019). On the difficulty of warm-starting neural network training. *arXiv preprint arXiv:1910.08475*.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. (2020). Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*.

Austin, B., Daley, C., Doerfler, D., Deslippe, J., Cook, B., Friesen, B., Kurth, T., Yang, C., and Wright, N. J. (2018). A metric for evaluating supercomputer performance in the era of extreme heterogeneity. In *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pages 63–71. IEEE.

Banerjee, A., Bernoulli, D., Zhang, H., Yuen, M.-F., Liu, J., Dong, J., Ding, F., Lu, J., Dao, M., Zhang, W., et al. (2018). Ultralarge elastic deformation of nanoscale diamond. *Science*, 360(6386):300–302.

Bardeen, J. and Shockley, W. (1950). Deformation potentials and mobilities in non-polar crystals. *Physical Review*, 80(1):72.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Bartók, A. P., Payne, M. C., Kondor, R., and Csányi, G. (2010). Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29.

Bedell, S., Khakifirooz, A., and Sadana, D. (2014). Strain scaling for CMOS. *MRS Bulletin*, 39(2):131–137.

Behler, J. (2011). Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. *Physical Chemistry Chemical Physics*, 13(40):17930–17955.

Beluch, W. H., Genewein, T., Nurnberger, A., and Kohler, J. M. (2018). The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Ismir*, volume 2, page 10.

Bishop, C. M. (1997). Bayesian neural networks. *Journal of the Brazilian Computer Society*, 4(1).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Blase, X. (2003). Quasiparticle band structure and screening in silicon and carbon clathrates. *Physical Review B*, 67(3):035211.

Blochl, P. E. (1994). Projector augmented-wave method. *Physical review B*, 50(24):17953.

Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127.

Bohr, M. (2007). The invention of uniaxial strained silicon transistors at intel. *Intel Corporation*, pages 1–4.

Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. (2017). Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv:1707.02476*.

Brasen, D., Fitzgerald Jr, E. A., Green, M. L., and Xie, Y.-H. (1993). Method for making low defect density semiconductor heterostructure and devices made thereby. US Patent 5,221,413.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Brockherde, F., Vogt, L., Li, L., Tuckerman, M. E., Burke, K., and Muller, K.-R. (2017). Bypassing the Kohn-Sham equations with machine learning. *Nature communications*, 8(1):1–10.

Burnaev, E. and Panov, M. (2015). Adaptive design of experiments based on Gaussian processes. In *International Symposium on Statistical Learning and Data Sciences*, pages 116–125. Springer.

Buza, K. (2014). Feedback prediction for blogs. In *Data analysis, machine learning and knowledge discovery*, pages 145–152. Springer.

Bzdok, D., Altman, N., and Krzywinski, M. (2018). Points of significance: statistics versus machine learning.

Cambria, E. and White, B. (2014). Jumping NLP curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57.

Cao, J., Ertekin, E., Srinivasan, V., Fan, W., Huang, S., Zheng, H., Yim, J., Khanal, D., Ogletree, D., Grossman, J., et al. (2009). Strain engineering and one-dimensional organization of metal–insulator domains in single-crystal vanadium dioxide beams. *Nature nanotechnology*, 4(11):732–737.

Cao, J., Li, F., Li, P., Ma, X., and Li, J. (2015). Analysis of ductile–brittle competitive fracture criteria for tension process of 7050 aluminum alloy based on elastic strain energy density. *Materials Science and Engineering: A*, 637:201–214.

Capasso, F. (1988). Compositionally graded semiconductors and their device applications. In *Electronic Structure of Semiconductor Heterojunctions*, pages 70–98. Springer.

Carrasquilla, J. and Melko, R. G. (2017). Machine learning phases of matter. *Nature Physics*, 13(5):431–434.

Castellanos-Gomez, A., Roldán, R., Cappelluti, E., Buscema, M., Guinea, F., van der Zant, H. S., and Steele, G. A. (2013). Local strain engineering in atomically thin MoS_2. *Nano letters*, 13(11):5361–5366.

Chang, K. S. and von Lilienfeld, O. A. (2018). $Al_xGa_{1-x}As$ crystals with direct 2 eV band gaps from computational alchemy. *Physical Review Materials*, 2(7):073802.

Chen, C., Zuo, Y., Ye, W., Li, X., Deng, Z., and Ong, S. P. (2020). A critical review of machine learning of energy materials. *Advanced Energy Materials*, 10(8):1903242.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Chidambaram, P., Bowen, C., Chakravarthi, S., Machala, C., and Wise, R. (2006). Fundamentals of silicon material properties for successful exploitation of strain engineering in modern CMOS manufacturing. *IEEE Transactions on Electron Devices*, 53(5):944–964.

Chmiela, S., Sauceda, H. E., Muller, K.-R., and Tkatchenko, A. (2018). Towards exact molecular dynamics simulations with machine-learned force fields. *Nature communications*, 9(1):1–10.

Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schutt, K. T., and Muller, K.-R. (2017). Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015.

Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Çivril, A. and Magdon-Ismail, M. (2009). On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47-49):4801–4811.

Claussen, N., Bernevig, B. A., and Regnault, N. (2019). Detection of topological materials with machine learning. *arXiv preprint arXiv:1910.10161*.

Cohn, D. A. (1994). Neural network exploration using optimal experiment design. In *Advances in neural information processing systems*, pages 679–686.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Conley, H. J., Wang, B., Ziegler, J. I., Haglund Jr, R. F., Pantelides, S. T., and Bolotin, K. I. (2013). Bandgap engineering of strained monolayer and bilayer MoS_2. *Nano letters*, 13(8):3626–3630.

Coraddu, A., Oneto, L., Ghio, A., Savio, S., Anguita, D., and Figari, M. (2014). Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Journal of Engineering for the Maritime Environment*, –(–):–.

Corkill, J. L. and Cohen, M. L. (1993). Band gaps in some group–IV materials: A theoretical analysis. *Phys. Rev. B*, 47:10304.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553.

Cukier, K. and Mayer-Schoenberger, V. (2013). The rise of big data: How it's changing the way we think about the world. *Foreign Aff.*, 92:28.

Curtarolo, S., Setyawan, W., Wang, S., Xue, J., Yang, K., Taylor, R. H., Nelson, L. J., Hart, G. L., Sanvito, S., Buongiorno-Nardelli, M., et al. (2012). AFLOWLIB.ORG: A distributed materials properties repository from high-throughput *ab initio* calculations. *Computational Materials Science*, 58:227–235.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Derkach, D., Kazeev, N., Ratnikov, F., Ustyuzhanin, A., and Volokhova, A. (2020). Cherenkov detectors fast simulation using neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 952:161804.

Dieleman, S., Schlter, J., Raffel, C., Olson, E., Snderby, S. K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., et al. (2016). Lasagne: First release. *URL http://dx. doi. org/10.5281/zenodo*, 27878.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Donmez, P. and Carbonell, J. G. (2008). Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 619–628.

Du, M., Cui, L., Cao, Y., and Bard, A. J. (2015). Mechanoelectrochemical catalysis of the effect of elastic strain on a platinum nanofilm for the orr exerted by a shape memory alloy substrate. *Journal of the American Chemical Society*, 137(23):7397–7403.

Dua, D. and Taniskidou, E. K. (2017). UCI machine learning repository [http://archive. ics. uci. edu/ml].

El Kurdi, M., Fishman, G., Sauvage, S., and Boucaud, P. (2010). Band structure and optical gain of tensile-strained germanium based on a 30 band k·p formalism. *Journal of Applied Physics*, 107(1):013710.

Fedorov, V. V. (1972). *Theory of optimal experiments*. Elsevier.

Fei, R. and Yang, L. (2014). Strain-engineering the anisotropic electrical conductance of few-layer black phosphorus. *Nano letters*, 14(5):2884–2889.

Feng, J., Qian, X., Huang, C.-W., and Li, J. (2012). Strain-engineered artificial atom as a broad-spectrum solar energy funnel. *Nature Photonics*, 6(12):866.

Fernandes, K., Vinagre, P., and Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer.

Freeman, L. C. (1965). *Elementary applied statistics: for students in behavioral science*. John Wiley & Sons.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378.

Fu, X., Su, C., Fu, Q., Zhu, X., Zhu, R., Liu, C., Liao, Z., Xu, J., Guo, W., Feng, J., et al. (2014). Tailoring exciton dynamics by elastic strain-gradient in semiconductors. *Advanced Materials*, 26(16):2572–2579.

Gal, Y. (2016). Uncertainty in deep learning. *University of Cambridge*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proc. ICML'16*, pages 1050–1059.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep Bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*.

Gao, F. and Han, L. (2012). Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. (2018). Conditional neural processes. *arXiv preprint arXiv:1807.01613*.

Gautier, G., Polito, G., Bardenet, R., and Valko, M. (2019). DPPy: DPP sampling with Python. *JMLR*, 20(180):1–7.

Ge, M., Cao, C., Huang, J., Li, S., Chen, Z., Zhang, K.-Q., Al-Deyab, S., and Lai, Y. (2016). A review of one-dimensional TiO_2 nanostructured materials for environmental and energy applications. *Journal of Materials Chemistry A*, 4(18):6772–6801.

Gerritsma, J., Onnink, R., and Versluis, A. (1981). Geometry, resistance and stability of the delft systematic yacht hull series. *International shipbuilding progress*, 28(328):276–297.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

Giustino, F., Louie, S. G., and Cohen, M. L. (2010). Electron-phonon renormalization of the direct band gap of diamond. *Physical review letters*, 105(26):265501.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Gogotsi, Y. G., Kailer, A., and Nickel, K. G. (1999). Transformation of diamond to graphite. *Nature*, 401(6754):663–664.

Goreinov, S. A., Oseledets, I. V., Savostyanov, D. V., Tyrtyshnikov, E. E., and Zamarashkin, N. L. (2010). How to find a good submatrix. In *Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub*, pages 247–256. World Scientific.

Gould, N. and Scott, J. (2016). A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–5.

Graf, F., Kriegel, H.-P., Schubert, M., Polsterl, S., and Cavallaro, A. (2011). 2d image registration in CT images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 607–614. Springer.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356.

Gross, E. and Kohn, W. (1985). Local density-functional theory of frequency-dependent linear response. *Physical review letters*, 55(26):2850.

Gubaev, K., Podryabinkin, E. V., Hart, G. L., and Shapeev, A. V. (2019). Accelerating high-throughput searches for new alloys with active learning of interatomic potentials. *Computational Materials Science*, 156:148–156.

Gubaev, K., Podryabinkin, E. V., and Shapeev, A. V. (2018). Machine learning of molecular properties: Locality and active learning. *The Journal of chemical physics*, 148(24):241727.

Guinea, F., Katsnelson, M., and Geim, A. (2010). Energy gaps and a zero-field quantum Hall effect in graphene by strain engineering. *Nature Physics*, 6(1):30–33.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

Guo, L., Yan, H., Moore, Q., Buettner, M., Song, J., Li, L., Araujo, P. T., and Wang, H.-T. (2015). Elastic properties of van der Waals epitaxy grown bismuth telluride 2D nanosheets. *Nanoscale*, 7(28):11915–11921.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48.

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., and Davidson, J. (2018). Reliable uncertainty estimates in deep neural networks using noise contrastive priors.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160.

Hansen, K., Biegler, F., Ramakrishnan, R., Pronobis, W., Von Lilienfeld, O. A., Muller, K.-R., and Tkatchenko, A. (2015). Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *The journal of physical chemistry letters*, 6(12):2326–2331.

Hansen, K., Montavon, G., Biegler, F., Fazli, S., Rupp, M., Scheffler, M., Von Lilienfeld, O. A., Tkatchenko, A., and Muller, K.-R. (2013). Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419.

Harrison Jr, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air.

He, K., Poole, C., Mak, K. F., and Shan, J. (2013). Experimental demonstration of continuous electronic structure tuning via strain in atomically thin MoS_2. *Nano letters*, 13(6):2931–2936.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hedin, L. (1965). New method for calculating the one-particle green's function with application to the electron-gas problem. *Physical Review*, 139(3A):A796.

Hein, M., Andriushchenko, M., and Bitterwolf, J. (2019). Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. *arXiv:1309.6835*.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

Heyd, J. and Scuseria, G. E. (2004). Efficient hybrid density functional calculations in solids: Assessment of the Heyd–Scuseria–Ernzerhof screened Coulomb hybrid functional. *The Journal of chemical physics*, 121(3):1187–1192.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Horvitz, D. G. and Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Huang, B. and Von Lilienfeld, O. A. (2016). Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity.

Huang, F.-Y. (2001). Silicon-germanium BiCMOS on SOI. US Patent 6,288,427.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*.

Hÿtch, M. J. and Minor, A. M. (2014). Observing and measuring strain in nanostructures and devices with transmission electron microscopy. *MRS bulletin*, 39(2):138–146.

Inaoka, T., Furukawa, T., Toma, R., and Y., S. (2015). Tensile-strain effect of inducing the indirect-to-direct band-gap transition and reducing the band-gap energy of Ge. *J. Appl. Phys.*, 118:105704.

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., et al. (2019). Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597.

Isayev, O., Oses, C., Toher, C., Gossett, E., Curtarolo, S., and Tropsha, A. (2017). Universal fragment descriptors for predicting properties of inorganic crystals. *Nature communications*, 8(1):1–12.

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., et al. (2013). Commentary: The materials project: A materials genome approach to accelerating materials innovation. *Apl Materials*, 1(1):011002.

Jain, S., Liu, G., Mueller, J., and Gifford, D. (2019). Maximizing overall diversity for improved uncertainty estimates in deep ensembles. *arXiv preprint arXiv:1906.07380*.

Jang, H., Baek, S., Ortiz, D., Folkman, C., Das, R., Chu, Y., Shafer, P., Zhang, J., Choudhury, S., Vaithyanathan, V., et al. (2008). Strain-induced polarization rotation in epitaxial (001) $BiFeO_3$ thin films. *Physical review letters*, 101(10):107602.

Jenkins, D. (1956). Calculations on the band structure of silicon. *Proceedings of the Physical Society. Section A*, 69(7):548.

Kampffmeyer, M., Salberg, A.-B., and Jenssen, R. (2016). Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2016 IEEE Conference on*, pages 680–688. IEEE.

Karlov, D. S., Sosnin, S., Tetko, I. V., and Fedorov, M. V. (2019). Chemical space exploration guided by deep neural networks. *RSC advances*, 9(9):5151–5157.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154.

Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, Inc.

Kononenko, I. (1989). Bayesian neural networks. *Biological Cybernetics*, 61(5):361–370.

Kresse, G. and Furthmuller, J. (1996). Efficiency of *ab-initio* total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational materials science*, 6(1):15–50.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kuklewicz, C. E., Malein, R. N., Petroff, P. M., and Gerardot, B. D. (2012). Electro-elastic tuning of single particles in individual self-assembled quantum dots. *Nano letters*, 12(7):3761–3765.

Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. *arXiv:1807.00263*.

Kulesza, A., Taskar, B., et al. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286.

Kumar, S. et al. (2015). *Multi-scale mechanics of monolayer graphene membranes: elasticity, fracture, and mechanochemistry*. PhD thesis, Massachusetts Institute of Technology.

LeCun, Y. (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as Gaussian processes. *arXiv preprint arXiv:1711.00165*.

Levina, E. and Bickel, P. J. (2005). Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems*, pages 777–784.

Li, D., Wu, B., Zhu, X., Wang, J., Ryu, B., Lu, W. D., Lu, W., and Liang, X. (2018a). MoS_2 memristors exhibiting variable switching characteristics toward biorealistic synaptic emulation. *ACS nano*, 12(9):9240–9252.

Li, H., Wang, X., and Ding, S. (2018b). Research and development of neural network ensembles: a survey. *Artificial Intelligence Review*, 49(4):455–479.

Li, J., Shan, Z., and Ma, E. (2014). Elastic strain engineering for unprecedented materials properties. *MRS Bulletin*, 39(2):108–114.

Li, M., Wan, Y., Tu, L., Yang, Y., and Lou, J. (2016). The effect of V_{MoS_3} point defect on the elastic properties of monolayer MoS_2 with REBO potentials. *Nanoscale research letters*, 11(1):1–7.

Liu, W., He, J., and Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 679–686.

Lofas, H., Grigoriev, A., Isberg, J., and Ahuja, R. (2011). Effective masses and electronic structure of diamond including electron correlation effects in first principles calculations using the GW-approximation. *AIP Advances*, 1(3):032139.

Lu, Z. and Bongard, J. (2009). Exploiting multiple classifier types with active learning. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1905–1906.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30.

Macchi, O. (1975). The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122.

Maeda, S.-i. (2014). A Bayesian encourages dropout. *arXiv preprint arXiv:1412.7003*.

Malheiros-Silveira, G. N. and Hernandez-Figueroa, H. E. (2012). Prediction of dispersion relation and PBGs in 2D PCs by using artificial neural networks. *IEEE Photonics Technology Letters*, 24(20):1799–1801.

Malinin, A. and Gales, M. (2018). Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pages 7047–7058.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

Mehta, M., Rissanen, J., Agrawal, R., et al. (1995). MDL-based decision tree pruning. In *KDD*, volume 21, pages 216–221.

Minamisawa, R. A., Süess, M. J., Spolenak, R., Faist, J., David, C., Gobrecht, J., Bourdelle, K. K., and Sigg, H. (2012). Top-down fabricated silicon nanowires under tensile elastic strain up to 4.5%. *Nature communications*, 3(1):1–6.

Mishkin, D. and Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*.

Mistry, K., Allen, C., Auth, C., Beattie, B., Bergstrom, D., Bost, M., Brazier, M., Buehler, M., Cappellani, A., Chau, R., et al. (2007). A 45nm logic technology with high-k+ metal gate transistors, strained silicon, 9 cu interconnect layers, 193nm dry patterning, and 100% pb-free packaging. In *2007 IEEE International Electron Devices Meeting*, pages 247–250. IEEE.

Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org.

Momeni, K., Ji, Y., Wang, Y., Paul, S., Neshani, S., Yilmaz, D. E., Shin, Y. K., Zhang, D., Jiang, J.-W., Park, H. S., et al. (2020). Multiscale computational understanding and growth of 2D materials: a review. *npj Computational Materials*, 6(1):1–18.

Monkhorst, H. J. and Pack, J. D. (1976). Special points for brillouin-zone integrations. *Physical review B*, 13(12):5188.

Mosca, D., Vidal, F., and Etgens, V. H. (2008). Strain engineering of the magnetocaloric effect in MnAs epilayers. *Physical review letters*, 101(12):125503.

Nalisnick, E., Hernandez-Lobato, J. M., and Smyth, P. (2019). Dropout as a structured shrinkage prior. In *International Conference on Machine Learning*, pages 4712–4722.

Nam, D., Sukhdeo, D., Cheng, S.-L., Roy, A., Chih-Yao Huang, K., Brongersma, M., Nishi, Y., and Saraswat, K. (2012). Electroluminescence from strained germanium membranes and implications for an efficient Si-compatible laser. *Applied physics letters*, 100(13):131112.

Nava, F., Canali, C., Jacoboni, C., Reggiani, L., and Kozlov, S. (1980). Electron effective masses and lattice scattering in natural diamond. *Solid State Communications*, 33(4):475–477.

Neklyudov, K., Molchanov, D., Ashukha, A., and Vetrov, D. P. (2017). Structured Bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Nie, A., Bu, Y., Li, P., Zhang, Y., Jin, T., Liu, J., Su, Z., Wang, Y., He, J., Liu, Z., et al. (2019). Approaching diamond's theoretical elasticity and strength limits. *Nature Communications*, 10(1):1–7.

Novikov, I. S., Suleimanov, Y. V., and Shapeev, A. V. (2018). Automated calculation of thermal rate coefficients using ring polymer molecular dynamics and machine-learning interatomic potentials with active learning. *Physical Chemistry Chemical Physics*, 20(46):29503–29512.

Novoselov, I., Yanilkin, A., Shapeev, A., and Podryabinkin, E. (2019). Moment tensor potentials as a promising tool to study diffusion processes. *Computational Materials Science*, 164:46–56.

Novoselov, K. S., Geim, A. K., Morozov, S. V., Jiang, D., Zhang, Y., Dubonos, S. V., Grigorieva, I. V., and Firsov, A. A. (2004). Electric field effect in atomically thin carbon films. *science*, 306(5696):666–669.

Nugteren, C. and Codreanu, V. (2015). Cltune: A generic auto-tuner for OpenCL kernels. In *Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2015 IEEE 9th International Symposium on*, pages 195–202. IEEE.

Oganov, A. R. and Glass, C. W. (2006). Crystal structure prediction using *ab initio* evolutionary techniques: Principles and applications. *The Journal of chemical physics*, 124(24):244704.

Oh, K.-S. and Jung, K. (2004). Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314.

Opitz, D. W. and Shavlik, J. W. (1996). Generating accurate and diverse members of a neural-network ensemble. In *Advances in neural information processing systems*, pages 535–541.

Paisley, J., Blei, D. M., and Jordan, M. I. (2012). Variational Bayesian inference with stochastic search. In *Proc. ICML'12*, pages 1363–1370.

Parr, R. G. (1980). Density functional theory of atoms and molecules. In *Horizons of Quantum Chemistry*, pages 5–15. Springer.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Peano, V., Sapper, F., and Marquardt, F. (2019). Rapid exploration of topological band structures using deep learning. *arXiv preprint arXiv:1912.03296*.

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. (2018). High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. *arXiv preprint arXiv:1802.07167*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peng, Q. and De, S. (2014). Elastic limit of silicane. *Nanoscale*, 6(20):12071–12079.

Perdew, J. P., Burke, K., and Ernzerhof, M. (1996). Generalized gradient approximation made simple. *Physical review letters*, 77(18):3865.

Pilozzi, L., Farrelly, F. A., Marcucci, G., and Conti, C. (2018). Machine learning inverse problem for topological photonics. *Communications Physics*, 1(1):1–7.

Podryabinkin, E., Rudyak, V., Gavrilov, A., and May, R. (2013). Detailed modeling of drilling fluid flow in a wellbore annulus while drilling. In *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, volume 6.

Podryabinkin, E. V. and Shapeev, A. V. (2017). Active learning of linearly parametrized interatomic potentials. *Computational Materials Science*, 140:171–180.

Podryabinkin, E. V., Tikhonov, E. V., Shapeev, A. V., and Oganov, A. R. (2019). Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning. *Physical Review B*, 99(6):064114.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pages 6638–6648.

Radovic, A., Williams, M., Rousseau, D., Kagan, M., Bonacorsi, D., Himmel, A., Aurisano, A., Terao, K., and Wongjirad, T. (2018). Machine learning at the energy and intensity frontiers of particle physics. *Nature*, 560(7716):41–48.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. (2015). Big data meets quantum chemistry approximations: the δ-machine learning approach. *Journal of chemical theory and computation*, 11(5):2087–2096.

Raskutti, G., Wainwright, M. J., and Yu, B. (2014). Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366.

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Reddy, A. C. (2004). Experimental evaluation of elastic lattice strains in the discontinuously SiC reinforced Al-alloy composites. In *National Conference on Emerging Trends in Mechanical Engineering, Nagapur*, page 81.

Richard, M. D. and Lippmann, R. P. (1991). Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483.

Rose, F., Toher, C., Gossett, E., Oses, C., Nardelli, M. B., Fornari, M., and Curtarolo, S. (2017). Aflux: The LUX materials search API for the AFLOW data repositories. *Computational Materials Science*, 137:362–370.

Rosenbrock, H. (1960). An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184.

Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of chemical information and modeling*, 52(11):2864–2875.

Ruhle, S. (2016). Tabulated values of the Shockley-Queisser limit for single junction solar cells. *Solar Energy*, 130:139–147.

Saladukha, D., Clavel, M. B., Murphy-Armando, F., Greene-Diniz, G., Grüning, M., Hudait, M. K., and Ochalski, T. J. (2018). Direct and indirect band gaps in Ge under biaxial tensile strain investigated by photoluminescence and photoreflectance studies. *Phys. Rev. B*, 97:195304.

Samala, R. K., Chan, H.-P., Hadjiiski, L. M., Helvie, M. A., Cha, K. H., and Richter, C. D. (2017). Multi-task transfer learning deep convolutional neural network: application to computer-aided diagnosis of breast cancer on mammograms. *Physics in Medicine & Biology*, 62(23):8894.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Schilders, W. H., Van der Vorst, H. A., and Rommes, J. (2008). *Model order reduction: theory, research aspects and applications*, volume 13. Springer.

Schlom, D. G., Chen, L.-Q., Fennie, C. J., Gopalan, V., Muller, D. A., Pan, X., Ramesh, R., and Uecker, R. (2014). Elastic strain engineering of ferroic oxides. *Mrs Bulletin*, 39(2):118–130.

Schmidt, J., Marques, M. R., Botti, S., and Marques, M. A. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36.

Schutt, K. T., Arbabzadah, F., Chmiela, S., Muller, K. R., and Tkatchenko, A. (2017). Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):1–8.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. (2018). SchNet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722.

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Setyawan, W. and Curtarolo, S. (2010). High-throughput electronic band structure calculations: Challenges and tools. *Computational materials science*, 49(2):299–312.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504.

Shapeev, A. V. (2016). Moment tensor potentials: A class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173.

Shi, Z., Dao, M., Tsymbalov, E., Shapeev, A., Li, J., and Suresh, S. (2020). Metallization of diamond. *Proceedings of the National Academy of Sciences*, 117(40):24634–24639.

Shi, Z., Tsymbalov, E., Dao, M., Suresh, S., Shapeev, A., and Li, J. (2019). Deep elastic strain engineering of bandgap through machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4117–4122.

Shim, J., Kang, S., and Cho, S. (2020). Active learning of convolutional neural network for cost-effective wafer map pattern classification. *IEEE Transactions on Semiconductor Manufacturing*.

Shishkin, M. and Kresse, G. (2006). Implementation and performance of the frequency-dependent GW method within the PAW framework. *Physical Review B*, 74(3):035101.

Sholl, D. and Steckel, J. A. (2011). *Density functional theory: a practical introduction*. John Wiley and Sons.

Smith, J. S., Isayev, O., and Roitberg, A. E. (2017). ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203.

Smith, J. S., Nebgen, B., Lubbers, N., Isayev, O., and Roitberg, A. E. (2018). Less is more: Sampling chemical space with active learning. *The Journal of chemical physics*, 148(24):241733.

Smith, J. S., Nebgen, B. T., Zubatyuk, R., Lubbers, N., Devereux, C., Barros, K., Tretiak, S., Isayev, O., and Roitberg, A. E. (2019). Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nature communications*, 10(1):1–8.

Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado, Z. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Sukhdeo, D. S., Nam, D., Kang, J.-H., Brongersma, M. L., and Saraswat, K. C. (2014). Direct bandgap germanium-on-silicon inferred from 5.7% <100> uniaxial tensile strain. *Photonics Research*, 2(3):A8–A13.

Sun, S., Zhang, G., Wang, C., Zeng, W., Li, J., and Grosse, R. (2018). Differentiable compositional kernel learning for Gaussian processes. *arXiv preprint arXiv:1806.04326*.

Tai, Y., Yang, J., and Liu, X. (2017). Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3147–3155.

Taylor, R. H., Rose, F., Toher, C., Levy, O., Yang, K., Nardelli, M. B., and Curtarolo, S. (2014). A RESTful API for exchanging materials data in the AFLOWLIB.org consortium. *Computational materials science*, 93:178–192.

Teye, M., Azizpour, H., and Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. *arXiv preprint arXiv:1802.06455*.

Topsakal, M., Cahangirov, S., and Ciraci, S. (2010). The response of mechanical and electronic properties of graphane to the elastic strain. *Applied Physics Letters*, 96(9):091912.

Tsanas, A. and Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567.

Tsymbalov, E., Fedyanin, K., and Panov, M. (2020). Dropout strikes back: Improved uncertainty estimation via diversity sampled implicit ensembles. *arXiv preprint arXiv:2003.03274*.

Tsymbalov, E., Makarychev, S., Shapeev, A., and Panov, M. (2019). Deeper connections between neural networks and gaussian processes speed-up active learning. In *IJCAI*.

Tsymbalov, E., Panov, M., and Shapeev, A. (2018). Dropout-based active learning for regression. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 247–258. Springer.

Tufekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140.

Van Dyk, D. A. and Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50.

Van Zeghbroeck, B. V. (2010). *Principles of semiconductor devices and heterojunctions*. Prentice Hall.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. N. and Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer.

Varshni, Y. P. (1967). Temperature dependence of the energy gap in semiconductors. *physica*, 34(1):149–154.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3):261–272.

von Lilienfeld, O. A., Tkatchenko, A., and Muller, K.-R. (2019). Exploring chemical compound space with quantum-based machine learning. *arXiv preprint arXiv:1911.10084*.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., et al. (2019). Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems. *arXiv: 1903.12394*.

von Rueden, L., Mayer, S., Sifa, R., Bauckhage, C., and Garcke, J. (2020). Combining machine learning and simulation to a hybrid modelling approach: Current and future directions. In *International Symposium on Intelligent Data Analysis*, pages 548–560. Springer.

Vovk, V. (2013). Kernel ridge regression. In *Empirical inference*, pages 105–116. Springer.

Wachter, S., Polyushkin, D. K., Bethge, O., and Mueller, T. (2017). A microprocessor based on a two-dimensional semiconductor. *Nature communications*, 8(1):1–6.

Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359.

Wang, J. and Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, page 11.

Wang, L., Liu, P., Guan, P., Yang, M., Sun, J., Cheng, Y., Hirata, A., Zhang, Z., Ma, E., Chen, M., et al. (2013). In situ atomic-scale observation of continuous and reversible lattice deformation beyond the elastic limit. *Nature communications*, 4(1):1–7.

Willatzen, M., Cardona, M., and Christensen, N. (1994). Linear muffin-tin-orbital and k·p calculations of effective masses and band structure of semiconducting diamond. *Physical Review B*, 50(24):18054.

Wright, N. J. (2016). NERSC-9. Presentation, see slide 19.

Xiang, Q. (2003). Cmos with strained silicon channel NMOS and silicon germanium channel PMOS. US Patent 6,600,170.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms.

Xu, R., Jang, H., Lee, M.-H., Amanov, D., Cho, Y., Kim, H., Park, S., Shin, H.-j., and Ham, D. (2019). Vertical MoS_2 double-layer memristor with electrochemical metallization as an atomic-scale synapse with switching thresholds approaching 100 mV. *Nano letters*, 19(4):2411–2417.

Yan, K., Maark, T. A., Khorshidi, A., Sethuraman, V. A., Peterson, A. A., and Guduru, P. R. (2016). The influence of elastic strain on catalytic activity in the hydrogen evolution reaction. *Angewandte Chemie International Edition*, 55(21):6175–6181.

Yanai, T., Tew, D. P., and Handy, N. C. (2004). A new hybrid exchange–correlation functional using the Coulomb-attenuating method (CAM-B3LYP). *Chemical Physics Letters*, 393(1-3):51–57.

Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315.

Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808.

Yildiz, B. (2014). "stretching" the energy landscape of oxides—effects on electrocatalysis and diffusion. *Mrs Bulletin*, 39(2):147–156.

Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zanolli, Z., Fuchs, F., Furthmuller, J., von Barth, U., and Bechstedt, F. (2007). Model GW band structure of InAs and GaAs in the wurtzite phase. *Physical Review B*, 75(24):245121.

Zdeborová, L. (2017). Machine learning: New tool in the box. *Nature Physics*, 13(5):420–421.

Zhang, D., Lu, L., Guo, L., and Karniadakis, G. E. (2019). Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850.

Zhang, H., Tersoff, J., Xu, S., Chen, H., Zhang, Q., Zhang, K., Yang, Y., Lee, C.-S., Tu, K.-N., Li, J., et al. (2016). Approaching the ideal elastic strain limit in silicon nanowires. *Science advances*, 2(8):e1501382.

Zhang, P., Shen, H., and Zhai, H. (2018). Machine learning topological invariants with neural networks. *Physical review letters*, 120(6):066401.

Zhang, T., Yu, B., et al. (2005). Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579.

Zhao, Y. and Truhlar, D. G. (2006). A density functional that accounts for medium-range correlation energies in organic chemistry. *Organic Letters*, 8(25):5753–5755.

Zhaochun, Z., Ruiwu, P., and Nianyi, C. (1998). Artificial neural network prediction of the band gap and melting point of binary and ternary compound semiconductors. *Materials Science and Engineering: B*, 54(3):149–152.

Zhou, Z.-H., Wu, J., and Tang, W. (2002). Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263.

Zubatyuk, R., Smith, J. S., Leszczynski, J., and Isayev, O. (2019). Accurate and transferable multitask prediction of chemical properties with an atoms-in-molecules neural network. *Science advances*, 5(8):eaav6490.

Zuo, Y., Chen, C., Li, X., Deng, Z., Chen, Y., Behler, J., Csányi, G., Shapeev, A. V., Thompson, A. P., Wood, M. A., et al. (2020). Performance and cost assessment of machine learning interatomic potentials. *The Journal of Physical Chemistry A*, 124(4):731–745.

Appendices

A Conventional unit cell

The conventional lattice matrix would be an identity matrix: $L_{\text{conventional}} = I$, and the basis $B_{\text{conventional}} = (x_j, y_j, z_j), j = 1, \ldots, 8$ as

$$\begin{aligned}
(x_1, y_1, z_1) &= (0, 0, 0); \\
(x_2, y_2, z_2) &= (0.25, 0.75, 0.75); \\
(x_3, y_3, z_3) &= (0, 0.5, 0.5); \\
(x_4, y_4, z_4) &= (0.75, 0.75, 0.25); \\
(x_5, y_5, z_5) &= (0.5, 0, 0.5); \\
(x_6, y_6, z_6) &= (0.75, 0.25, 0.75); \\
(x_7, y_7, z_7) &= (0.75, 0.25, 0.75); \\
(x_8, y_8, z_8) &= (0.5, 0.5, 0).
\end{aligned} \tag{A.1}$$

B On the critical point notation for the Brillouin zone of a strained crystal

Under general 3D three-normal-strains deformation, the original O_h crystal point group of Si (diamond C) turns into a D_{2h} point group. The Brillouin zone for for deformed crystal case is shown in Figure B.1. In general, it is not anymore a regular truncated octahedron with equilateral hexagonal and square faces, as in Figure 2.1. The reciprocal space lattice vectors are adjusted by the inverse transpose of the deformation gradient tensor in real space, i.e., $\boldsymbol{F}^{-T}$, as a result of the deformation. The center of any type of Brillouin zone is labeled as Γ, and we keep this tradition. In undeformed Si, the centers of the square and regular hexagonal surfaces on the Brillouin zone boundary are completely degenerate and labeled as X and L, respectively. For the simplicity of comparison, we follow the same spirit and still denote the 'X'-type points as the centers of the tetragon surfaces and 'L'-type points as the centers of the regular/non-regular hexagonal surfaces. The lines that connect the Γ point to the 'X'-type points are labeled as 'Δ'-type. This way, the six 'X'-and 'L'-type points, though non-degenerate, would keep the correct fractional coordinates of $\langle 0.5, 0, 0.5\rangle$-and $\langle 0.5, 0, 0\rangle$-type, and the **k**-points along the Γ-'X' line would all have the $\langle \zeta, 0, \zeta\rangle$-type coordinates. **k**-points of our concern always appear on either the center of the Brillouin zone, the center of the zone boundary surfaces, or the line connecting the zone center and surface center; thus, our notations are sufficient.

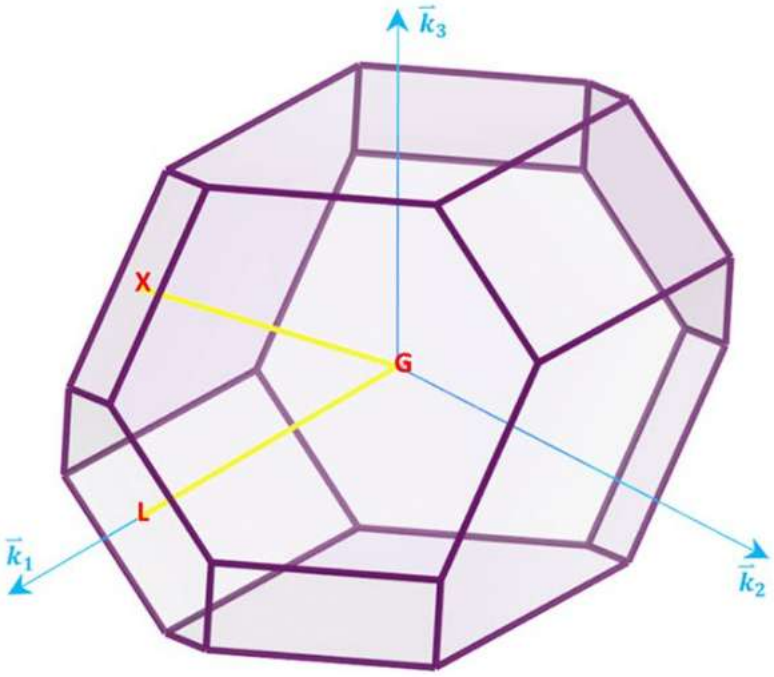


FIGURE B.1: Brillouin zone of diamond cubic crystal under three-normal-strains deformation. It is a tetradecahedron with 8 hexagonal and 6 quadrilateral faces.

C VASP calculation settings

C.1 Silicon calculations on a primitive cell

C.1.1 Preliminary ground state calculation

```
ENCUT = 600
ISMEAR =  0
SIGMA  =  0.05
EDIFF  = 1E-8
NPAR = 4
ISIF = 2
ISYM = 0
```

C.1.2 DFT virtual orbitals calculation

```
ENCUT = 600
ALGO = Exact
NBANDS  = 128
LOPTICS = .TRUE. ;
CSHIFT = 0.1
NEDOS = 2000

ISMEAR =  0
SIGMA  =  0.05
EDIFF  = 1E-8
NPAR = 4
ISYM = 0
```

C.1.3 GW calculation

```
ENCUT = 600
ALGO = GW0 ;
LSPECTRAL = .TRUE. ;
NOMEGA = 50
NBANDS = 128
ISYM = 0
```

C.2 Diamond calculations on a primitive cell

C.2.1 Preliminary ground state calculation

```
ENCUT = 600
ISMEAR =  0
SIGMA  =  0.05
EDIFF  = 1E-8
NPAR = 4
ISYM = 0

ISIF = 2
IBRION = 2
NSW = 1000
```

C.2.2 DFT virtual orbitals calculation

```
ENCUT = 600
ALGO = Exact
NBANDS  = 96
LOPTICS = .TRUE. ;
CSHIFT = 0.1
NEDOS = 2000

ISMEAR =  0
SIGMA  =  0.05
EDIFF  = 1E-8
NPAR = 4
ISYM = 0
```

C.2.3 GW calculation

```
ENCUT = 600
ALGO = GW0 ;
LSPECTRAL = .TRUE. ;
NOMEGA = 50
NBANDS = 96
ISYM = 0
```

D Data sets for NN experiments

D.1 Active learning with vanilla dropout

We took the data from UCI ML repository Dua and Taniskidou (2017), see the Table D.1 for more details. All the datasets represent real-world problems with 15+ dimensions and 30000+ samples. The exception is the synthetic Rosenbrock 2000D dataset, which has 10000 samples and 2000 dimensions.

TABLE D.1: Summary of the datasets used in our experiments.

Dataset name	# of samples	# of attributes	Feature to predict
BlogFeedback Buza (2014)	60021	281	Number of comments
SGEMM GPU Nugteren and Codreanu (2015)	241600	18	Median calculation time
YearPredictionMSD Bertin-Mahieux et al. (2011)	515345	90	Year
Relative location of CT slices Graf et al. (2011)	53500	386	Relative location
Online News Popularity Fernandes et al. (2015)	39797	61	Number of shares
KEGG Network Shannon et al. (2003)	53414	24	Clustering coefficient
Rosenbrock 2000D Rosenbrock (1960)	10 000	2000	Function value

D.2 Diversified dropout

TABLE D.2: Summary of the datasets used in UCI experiments section, see Dua and Taniskidou (2017).

Dataset name	Samples	Columns
boston Harrison Jr and Rubinfeld (1978)	506	13
concrete Yeh (1998)	1030	8
energy Tsanas and Xifara (2012)	768	8
kin8nm [1]	8192	8
naval Coraddu et al. (2014)	11934	16
ccpp Tufekci (2014)	9568	4
red wine Cortez et al. (2009)	1599	11
yacht Gerritsma et al. (1981)	308	6

E Hydraulic simulator

This part of the research was carried out in collaboration with Maxim Panov and Evgeny Podryabinkin.

In the oil industry, to determine the optimal parameters and control the drilling process, engineers carry out hydraulic calculations of the well's circulation system, which are usually based on either empirical formulas or semi-analytical solutions of the hydrodynamic equations. However, such semi-analytical solutions are known just for a few individual cases, while in the other ones, only very crude approximations are usually available.As a result, such calculations have relatively low precision. On the other hand, the full-scale numerical solution of the hydrodynamic equations describing the flow of drilling fluids can provide a sufficient level of accuracy, but it requires significant computational resources and subsequently is very costly and time-consuming. The possible solution to this problem is the use of a surrogate model.

We used a surrogate model for the fluid flow in the wellbore while drilling, see Podryabinkin et al. (2013). We consider the following key parameters of this system, varied in the following ranges:

- Diameter ratio of tubes, $0.5 \leq \frac{D_S}{D_H} \leq 0.9$;
- Reynolds number characterizing the rate of pumping of drilling fluid $0 \leq R_{\text{trip}} \leq 20000$;
- Axial Reynolds number, characterizing the axial flow relevance $0 \leq R_{\text{fr}} \leq 20000$;
- Rotational Reynolds number, characterizing the rotational flow relevance $0 \leq R_{\text{Rot}} \leq 20000$;
- Bingham number, characterizing the impact of the yield stress effects $0 \leq B_n \leq 50$;
- Power index of fluid $0.6 \leq n \leq 1$.

In this experiment, we used a two-layer neural network with 50 neurons per each layer and LeakyReLU activation function. Initial training and pool sets had 50 and 20 000 points, respectively. We completed 10 active learning iterations, adding 50 points per each iteration.

The results are shown in Figure E.1. Clearly, NNGP is superior in terms of RMSE and MAE, while the maximum error for NNGP is 10 times lower.

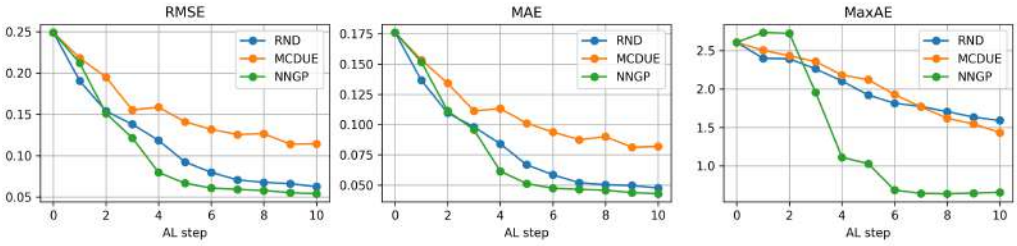


FIGURE E.1: Training curves for the active learning scenario for the hydraulic simulator case. The NNGP sampling algorithm outperforms the random sampling and MCDUE with a large margin in terms of maximal error. Here, RND refers to the random sampling algorithm.

F Diversified masks for image classification

This part of my research was carried out in collaboration with Maxim Panov and Kirill Fedyanin.

For regression, the variance of prediction is a standard uncertainty measure. However, uncertainty estimation for classification is, in some sense, more challenging than for regression as there is no obvious candidate for uncertainty measure.

Let us define the average probability for the class prediction by ensemble members $\bar{p}_T(y = c \mid x) = \frac{1}{T}\sum_{i=1}^{T} p(y = c \mid x, M_i)$. The standard uncertainty measure usually considered in the literature is

$$s(x) = 1 - \max_c \; \bar{p}_T(y = c \mid x),$$

which is based solely on the mean probabilities predicted by the ensemble. While providing good results in practice Snoek et al. (2019); Ashukha et al. (2020) it doesn't use the information about the variation of predictions between ensemble members.

In our work, we consider *BALD* Houlsby et al. (2011) uncertainty measure and combine it with different sampling schemes considered above. BALD is equal to the mutual information between outputs and model parameters:

$$I(x) = H(x) - \frac{1}{T}\sum_{c=1}^{C}\sum_{i=1}^{T} -p(y = c \mid x, M_i) log\big(p(y = c \mid x, M_i)\big),$$

where $H(x) = -\sum_{c=1}^{C} \bar{p}_T(y = c \mid x) \log\big(\bar{p}_T(y = c \mid x)\big)$ is an entropy of the ensemble mean. Importantly, BALD values are directly linked with the diversity of the ensemble members, and therefore are well suited for combination with our approach. The other popular uncertainty measures in the literature (see, e.g. Freeman (1965)) are

- **1 - maximum probability:** simple averaging of probabilities over the ensemble and taking the one with maximum value: $s(x) = 1 - \max_c \; \bar{p}_T(y = c \mid x)$
- **Variation ratio:** $v(x) = 1 - \frac{f_m(x)}{T}$, where $f_m(x)$ is the number of predictions for the most frequently chosen class by distinct runs of the model or members of the ensemble.

We consider three datasets: MNIST, which is a toy dataset of handwritten digits LeCun (1998), CIFAR-10, which is a 10-class image dataset with simple

objects Krizhevsky et al. (2009), and ImageNet Deng et al. (2009), the large scale image classification dataset. Importantly, for MNIST, we use only 500 train samples – otherwise models would have too good accuracy and uncertainty estimation for in-domain data would not be relevant. For CIFAR-10, we use 50 000 samples for training and 10 000 for testing. For the MNIST dataset, we use a simple convolutional neural network with two convolutional layers, max-pooling, and two fully connected layers. For the CIFAR-10, we use a more powerful network with 6 convolutional layers and batch normalization. Finally, for ImageNet, we use the pre-trained ResNet-18 neural network He et al. (2016) from PyTorch Paszke et al. (2019). Dropout with rate $p = 0.5$ is used before the last fully-connected layer in all the cases. $T = 100$ stochastic passes were made for every model. The experiments are repeated three times with different seeds for the models.

In the classification tasks, neural networks were trained for 100 epochs maximum, with checking the error on the validation set every epoch: early stopping triggers if the error did not decrease for three consecutive checks (patience = 3). The batch size equals to 128, and the dropout applied after the hidden linear layer with a rate equal to 0.5. Cross entropy was used as a loss function, and optimization was performed with the standard setting of PyTorch Adam optimizer. For each dataset, we use different models. Below we denote the convolutional layer with i input channels, j output channels as $\mathrm{conv(i, j)}$. Kernel size is 3x3 for each convolution.

- For the MNIST, we used simple convolutional network with layers conv(1, 16) - maxpool - conv(16, 32) - maxpool - linear(1152, 256) - dropout - linear(256, 10)
- For the CIFAR, we used VGG-alike network with layers conv(3, 16) - conv(16, 16) - maxpool(2, 2) - conv(16, 32) - conv(32, 32) - maxpool(2, 2) - conv(32, 64) - conv(64, 64) - maxpool(2, 2) - linear(1024, 128) - dropout - linear(128, 10)
- For the ImageNet, we used ResNet-18 with implementation and pre-trained weights from PyTorch.

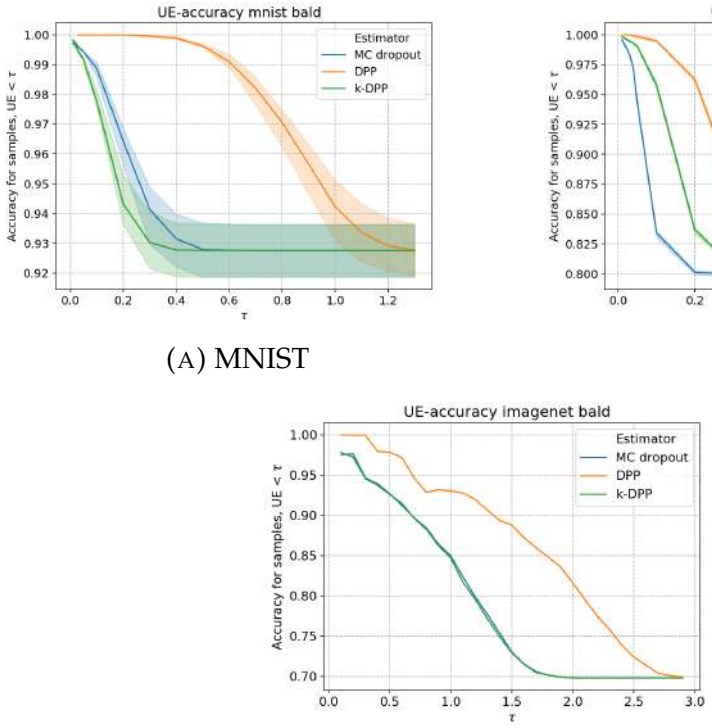
Ensembles of models for experiments below were trained separately on the same data from different weight initializations.

In this section, we aim to show the applicability of the proposed methods to the classification tasks, computer vision problems in particular.

We follow the experimental setup from Snoek et al. (2019), where it is proposed to assess the quality of UE models by confidence-accuracy curve, where confidence as $c(x) = 1 - s(x)$ with $s(x)$ being an uncertainty estimate.

For in-domain uncertainty estimation, results are presented via the UE-accuracy curve, see Figure F.1. It assumes that samples with lower uncertainty will be classified with higher average accuracy. It can be clearly seen that DPP significantly outperforms all the competitors on every dataset.

We should emphasize that the superiority of DPP is especially strong for ImageNet, where the usage of DPP required only 2% computational overhead compared to MC dropout according to our experiments. One of our



(A) MNIST

(B) CIFAR

(C) ImageNet

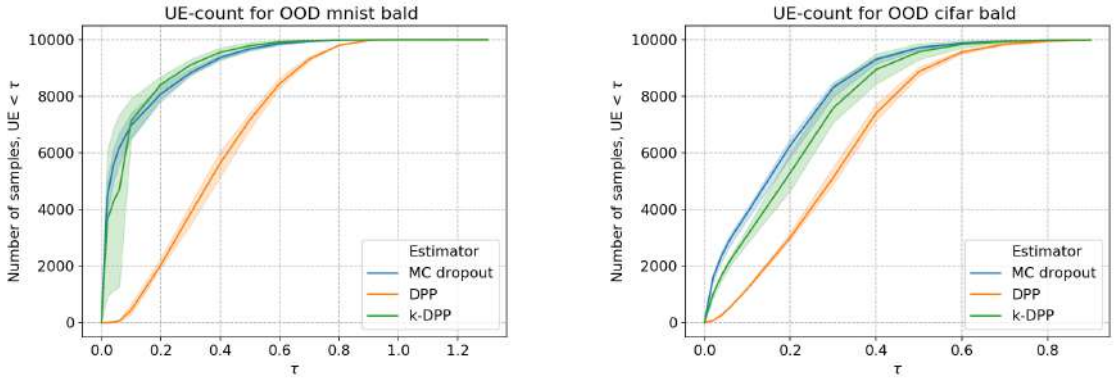
FIGURE F.1: UE-accuracy curve (the higher curve – the better). We select the samples with low uncertainty to ensure that the accuracy is higher for them.

initial concerns was the scalability of the methods because DPP sampling complexity is up to N^3, where N is the size of the layer. In practice, the overhead appears to be relatively small for real-world models because the number of operations to compute dropout masks is negligible compared to the normal forward propagation in a large network. For the ImageNet experiment, the difference was less than a few percent comparing to the Monte-Carlo dropout, see Table F.1.

TABLE F.1: Time to calculate uncertainty with different methods on 5 000 sample images from ImageNet

	MC dropout	DPP	k-DPP
Inference time, s	125.5±0.03	129.9±0.14	132.2±0.03

We also consider the detection of out-of-distribution samples, which is one of the important problems for the uncertainty estimation. We use fashion-MNIST Xiao et al. (2017) and SVHN images Netzer et al. (2011) as OOD samplesfor MNIST and CIFAR-10 correspondingly. We use the count-vs-uncertainty curve and expect fewer points with the low uncertainty for good uncertainty estimation methods. The results are presented in Figure F.2. We see that the DPP-based approach allows us to detect OOD samples better for both considered datasets.



(A) MNIST (B) CIFAR

FIGURE F.2: Count-vs-uncertainty curve for out-of-distribution data (the lower curve – the better).

We also provide the results for the two baseline sampling methods: 1 - maximum probability, and variation ratio, see Figure F.3). For the probability, we average the predictions for all dropout masks. We also report the result for a single network without a dropout and for an ensemble of 20 independently trained networks. The same methods are applied to the OOD experiment (Figure F.4). Note, here we provide the results for our methods both with correlation and covariance (cov) kernels. We can see that for these measures, our DPP approach based on the correlation kernel outperforms the other approaches as well.

Finally, we want to present the out-of-distribution experiment for the ImageNet dataset here. As OOD images, we took the 50 000 samples from the CheXpert Small(chest radiography dataset) Irvin et al. (2019). We used BALD uncertainty measure. The results are presented in Figure F.5. The DPP method performs very well, while k-DPP is inferior to the MC Dropout.

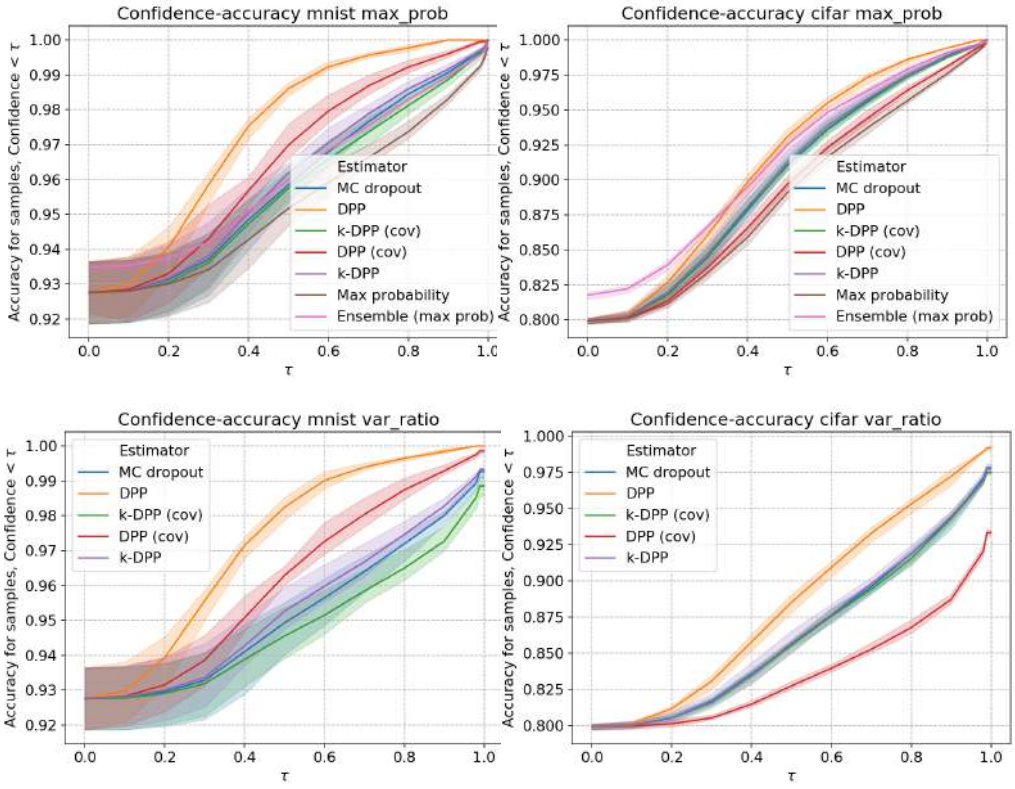


FIGURE F.3: Accuracy-vs-confidence curve (the higher curve – the better). We provide results for max probability and variation ratio measures for MNIST and CIFAR datasets.

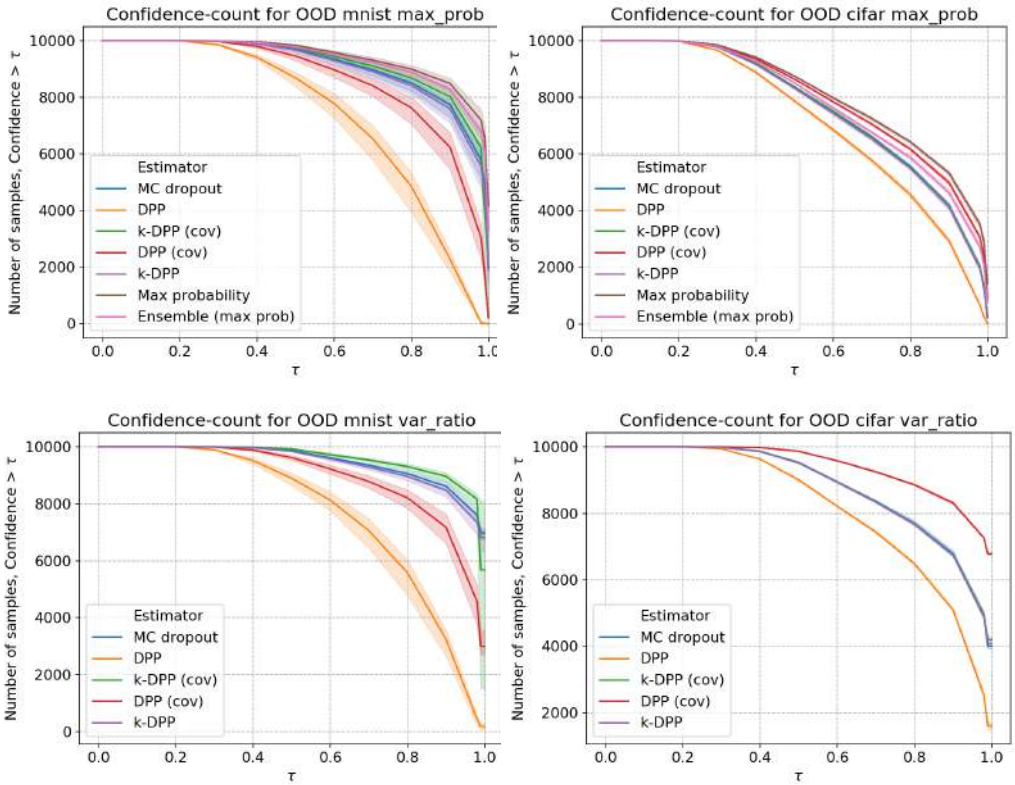


FIGURE F.4: Count-vs-confidence curve for out-of-distribution data (the lower curve – the better). We provide results for max probability and variation ratio measures for MNIST and CIFAR datasets.

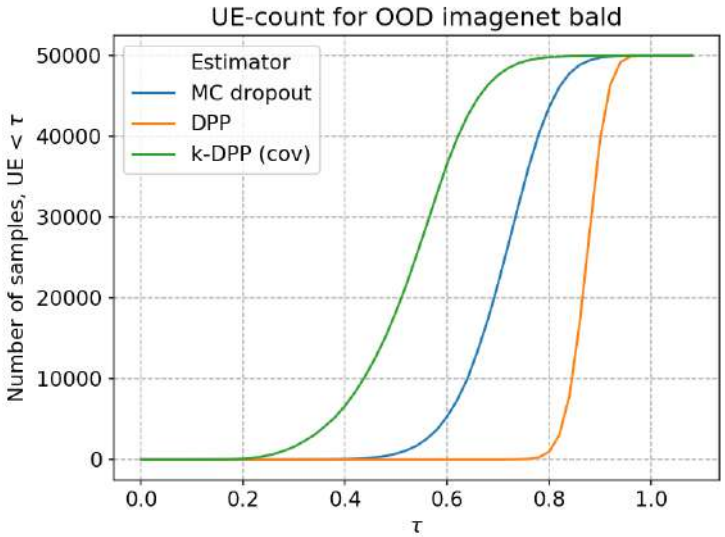


FIGURE F.5: Count-vs-uncertainty curve for out-of-distribution data on ImageNet (the lower curve – the better).

G Crystal orientation effects study

According to the thorough study showcased in Figures G.1, G.2, G.3, G.4, the ranking of common diamond or silicon crystal direction families for obtaining the same target bandgap through uniaxial tensile or compressive straining (i.e., constrained straining without allowing for the Poisson effect) can be varied at different strain levels. For example, in order to achieve a 5 eV bandgap in diamond, uniaxial tensile straining along <100> direction requires a smaller strain magnitude than along <111> direction; whereas to achieve 4 eV bandgap in diamond, uniaxial tensile straining along <111> direction requires a smaller strain magnitude than along <100> direction, as depicted in Figure G.2. It is also found out that allowing atomic internal relaxation during straining results in evident structural reconfiguration, especially in large deformation cases. Some of the diamond straining cases may even facilitate graphitization Gogotsi et al. (1999).

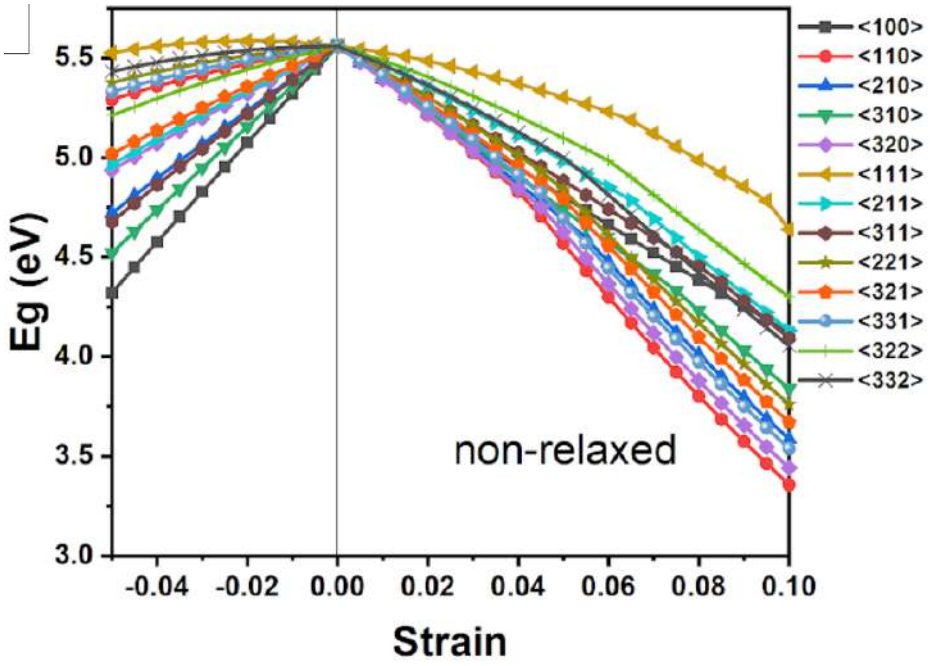


FIGURE G.1: Bandgap change as a function of strain for uniaxial straining along different crystal orientations in diamond with a non-relaxed atomic structure.

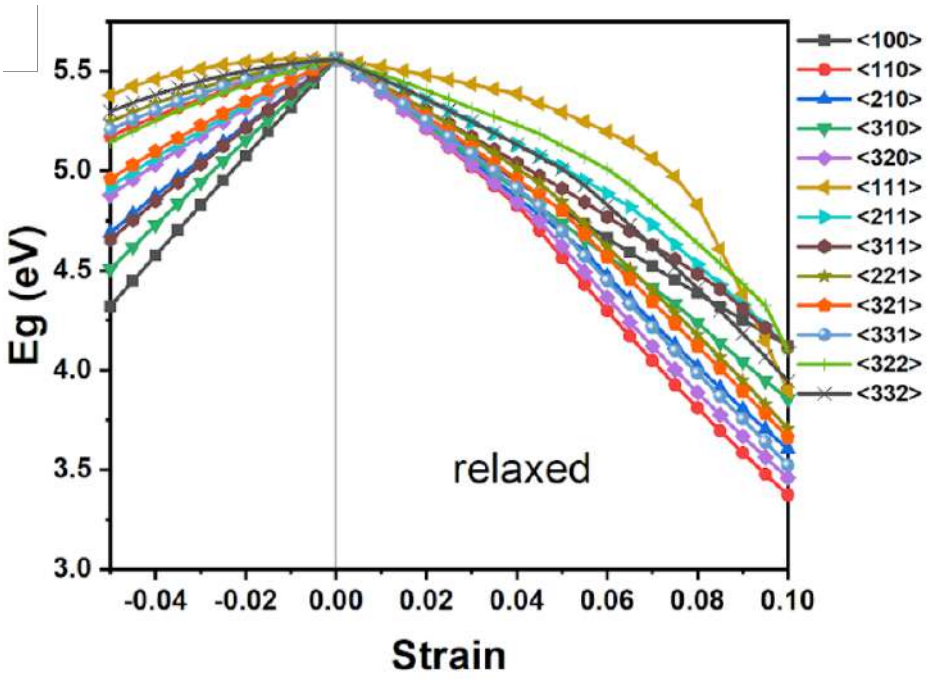


FIGURE G.2: Bandgap change as a function of strain for uniaxial straining along different crystal orientations in diamond with a relaxed atomic structure.

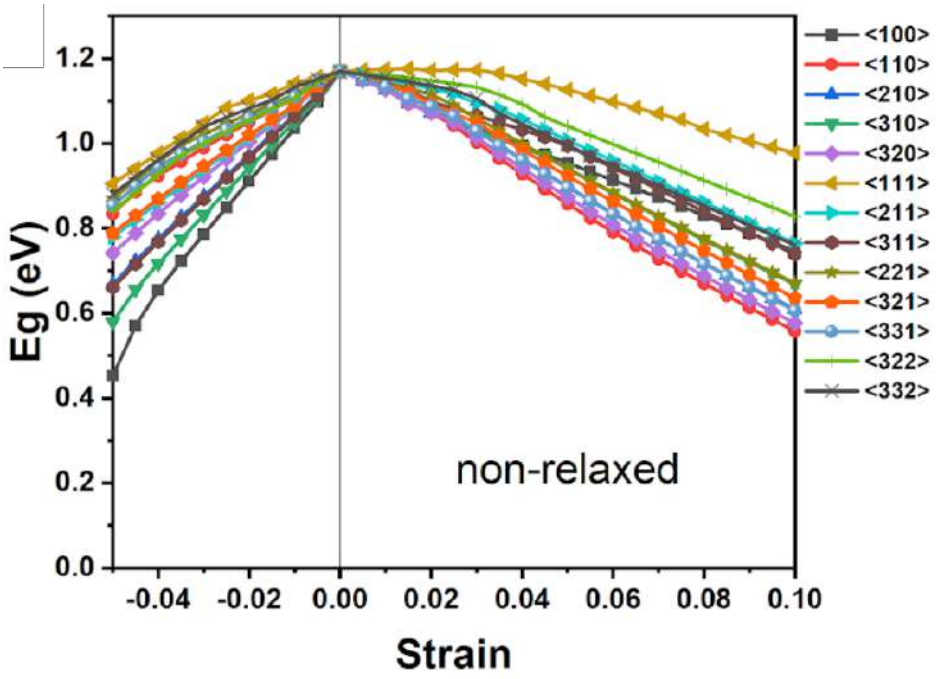


FIGURE G.3: Bandgap change as a function of strain for uniaxial straining along different crystal orientations in silicon with a non-relaxed atomic structure.

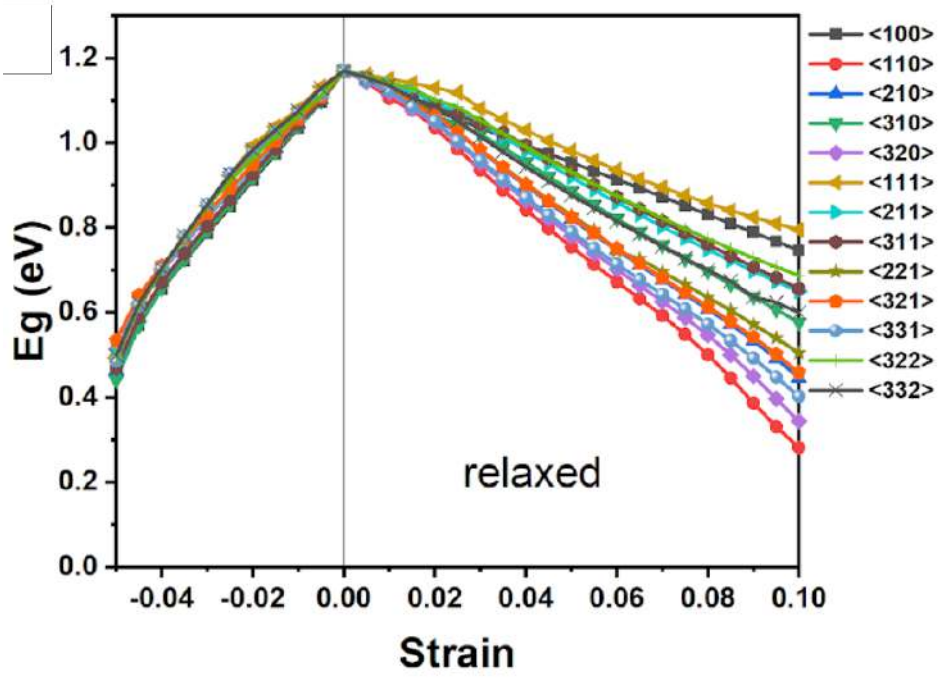


FIGURE G.4: Bandgap change as a function of strain for uniaxial straining along different crystal orientations in silicon with a relaxed atomic structure.