Predicting solid-state qubit material host

by

Oliver Lerstøl Hebnes

Thesis

for the degree of

Master of Science

Faculty of Mathematics and Natural Sciences
University of Oslo

February 26, 2021

Contents

Part I

Methodology and implementation

Chapter 1

Material Science Databases

There are multiple different databases for material science discovery available for every day use, some of them completely open-source while others are commercial. This chapter will give a brief overview of databases available for computational material science, and will serve as a toolbox for how to request information and what kind of python packages exist to process that information.

1.1 Fundamentals of a database

A quick search online will reveal the tremendous escalation of effort for big-data driven material science the last few years, resulting in several databases that stores ab-initio calculation details and results. We will here distinguish between a *cloud service*, which is a place to store independent databases for research and commercial purposes, and a *database*, which is an organized collection of structured information. As an example, a cloud service can store several databases, but a database cannot host a cloud service.

To limit the quest of databases, we have restricted the search for databases and cloud services to include inorganic compounds obtained experimentally or by first-principles calculations, in particular DFT-calculations using *Vienna ab initio simulation package* (VASP) [1]. VASP is a software for atomic scale materials programming. Table 1.2 and 1.3 shows a selection of databases and cloud services that meets the given criteries, respectively.

1.1.1 API and HTTP requests

To extract information from a database it is convenient to interact through an *API* (Application Programming Interface), which defines important variables such as the kind of requests to be made, how to make them and the data format for transmission. Importantly, this permits communication between different software medias. An API is entirely customizable, and can be made to extend existing functionality or tailormade for specific user-demanding modules.

The APIs that will be encountered is handled by the use of *HTTP* (Hypertext Transfer Protocol), which in its simplest form is a protocol that allows the fetching of resources. The protocol is client-server based, such as the client is requesting information and the server is responding to the request.

The most common HTTP-methods are GET, POST and HEAD, which are used to either retrieve, send, or get information about data, respectively. The latter request is usually done before a GET-method for requests considering large amount of data, since this can be a significant variable for the client's bandwith and load time. Following a request, the server normally responds with one of the status codes in table 1.1.

Status code	Description
2xx	OK - request was successful
3xx	Resource was redirected
4xx	Request failed due to either unsuccessful authentication or client error.
5xx	Request failed due to server error.

Table 1.1: Numeric status code for response. The leftmost digit decide the type of response, while the two follow-up digits depends on the implemented API.

A RESTful (Representational State Transfer) allows users to communicate with a server via a HTTP using a REST Architectural Style [2]. This enables the utilisation of Uniform Resource Identifiers (URI), where each object is represented as a unique resource and can be requested in a uniform manner. Importantly, this allows the use of both URIs and HTTP methods in an API, such that an object is represented by an unique URI whereas a HTTP-method can act on the object. This action will then return either the result of the action, or structured data that represents the object.

To provide a Python example, we can check the response by doing a GET request at the database Materials Project RESTful API in code listing 1.1. We use the preamble to version 2 of Materials Project, and add an API-check and an API-key. The response is shown in code listing 1.2. From the output, it is possible to tell that the supplied API-key is not valid, however, the request is valid.

```python
import requests
preamble = "https://www.materialsproject.org/rest/v2/"
url = preamble + "api_check"
```

```
params = {"API_KEY":"unique_api_key"}
response = requests.get(url=url, params=params)
print(response.json())
```

Listing 1.1: Practical example of getting a response from Materials Project database.

```
{"valid_response": True,
"response":
  {"api_key_valid": False,
  "details":"API_KEY is not a valid key.",
  "version":
    {"db": "2020_09_08",
    "pymatgen": "2020.8.13",
    "rest": ""2.0"}
  }
}
```

Listing 1.2: Practical example of response from Materials Project request based on 1.1. The request was done 28. january 2020.

Database	API	Free educational access	Number of entries
AFLOW	REST	True	3.27 M
OQMD [3, 4]	RESTful API (qmpy, matminer)	True	0.82 M
MP [5]	MAPI [6]	True	0.71 M
ICSD [7]	RESTful API	False	0.21 M
Jarvis-DFT	API	True	0.04 M

Table 1.2: A selection of databases of computational material science sorted after number of compounds. Abbreviations used are Novel Materials Discovery (NOMAD), Automatic-FLOW for Materials Discovery (AFLOW), Materials Project (MP), Inorganic Crystal Structure Database (ICSD) and Open Quantum Materials Database (OQMD). The number of entries can give the wrong perception of size of each respective database, as it does not visualise how many calculations have been done for each entry, nor if there might be duplicates.

1.1.2 Open-source Python libraries for material analysis

Many of the databases share convenient modules that are used to adapt, visualize, calculate or predict properties, making it easier for scientists to utilise

Cloud service	API/REST	Open educational access
NoMaD	API	True
CMR [8]	ASE	RESTful API
MatNavi	API	True
PRISMS	REST	True
Citrine	API	True
MPDS	API	False
MDF	API	False

Table 1.3: A selection of cloud services that offers database-storage. Abbreviations used are Computational Materials Repository (CMR), NIMS Materials Database (MatNavi), PRedictive Integrated Structural Materials Science (PRISMS), Materials Platform for Data Science (MPDS) and the Materials Data Fascility (MDF).

the databases. The Atomic Simulation Environment (ASE) is an environment in the Python programming language that includes several tools and modules for setting up, modifying and analyze atomistic simulations [9]. It is in particular used together with the cloud service Computational Materials Repository (CMR).

Another commonly used module is the Python Materials Genomics (pymatgen) [10]. This is a well-documented open module with both introductory and advanced use case examples written in Jupyter Notebook for easy reproducibility, and is integrated with the Materials Project RESTful API.

Another exceedingly popular library is matminer [11], which is an open-source toolkit for material analysis written in Python. Matminer is powered by a group known as *Hacking Materials Research Group* [1]. Matminer provides modules to extract data sets from many cloud-services and databases, with examples in table 1.2 and 1.3. Additionally, they provide the tools to extract possibly thousands of features from calculations based on DFT and more, and have modules for visualization and automatic machine learning. These tools will be examplified in the next chapter.

TODO : Add paragraph about sklearn.

A full selection of python libraries used and their versions can be found in the Github page (TODO: Add github page.)

[1]Project's Github site: https://github.com/hackingmaterials.

1.2 Databases and cloud services

Every database has its own speciality, and no two databases are the same. There exists entries that are fundamentally identical in several databases, but with different properties as a consequence of parameters used, such as the functional utilised in VASP or the relaxation scheme. This section digs up what exactly is each respective database's claim to fame.

1.2.1 Novel Materials Discovery

The Novel Materials Discovery (NOMAD) [12] Repository is an open-access platform for sharing and utilizing computational materials science data. NOMAD also consists of several branches such as NOMAD Archieve, which is the representation of the NOMAD repository parsified into a code-independent format, NOMAD Encyclopedia, which is a graphical user interface (GUI) for characterizing materials, and lastly NOMAD Analytics Toolkit, which includes early-development examples of artificial-intelligence tools [12].

Databases that are a part of NOMAD data collection includes Materials Project, the Open Quantum Materials Database and AFLOW. They are all based on the underlying quantum engine VASP.

1.2.2 Materials project

Materials project [5] is an open source project that offers a variety of properties of over one hundred thousand of inorganic crystalline materials. It is known as the initiator of materials genomics and has as its mission to accelerate the discovery of new technological materials, with an emphasis on batteries and electrodes, through advanced scientific computic and innovative design.

Every compound has an initial relaxation of cell and lattice parameters performed using a $1000k$-point mesh to ensure that all properties calculated are representative of the idealized unit cell for each respective crystal structure. The functional GGA is used to calculate band structures, while for transition metals it is applied $+U$ correction to correct for correlation effects in d- and f-orbital systems that are not addressed by GGA calculations [13]. The thermodynamic stability for each phase with respect to decomposition, is also calculated. This is denoted as E Above Hull, with a value of zero is defined as the most stable phase at a given composition, while larger positive values indicate increased instability.

Each material contains multiple computations for different purposes, resulting in different 'tasks'. The reason behind this is that each computation has a purpose, such as to calculate the band structure or energy. Therefore, it is possible to receive several tasks for one material which results in more features per material.

1.2.3 AFLOW

The AFLOW[14–16] repository is an automatic software framework for the calculations of a wide range of inorganic material properties. They utilise the GGA-PBE functional within VASP with projector-augmented wavefunction (PAW) potentials to relax twice and optimize the ICSD-sourced structur. They are using a $3000 - 6000$ k-point mesh, indicating a more computationally expensive calculation compared to the Materials Project. Next, the band structure is calculated with an even higher k-point density, in addition to the $+U$ correction term for most occupied d- and f-orbital systems, resulting in a standard band gap [17]. Furthermore, they apply a standard fit gathered from a study of DFT-computed versus experimentally measured band gap widths to the initial calculated value, obtaining a fitted band gap [18].

AFLOW-ML [19] is an API that uses machine learning to predict thermomechanical and electronic properties based on the chemical composition and atomic structure alone, which they denote as *fragment descriptors*. They start with applying a classification model to predict if a compound is either a metal or an insulator, where the latter is confirmed with an additional regression model to predict the band gap width. To be able to predict properties on an independent data set, they utilise a fivefold cross validation process for each model. They report a 93% prediction success rate of their initial binary classification model, whereas the majority of the wrongful predictions are narrow-gap semiconductors. The authors does not compare their predicted band gap to experimental values, but it is found that 93% of the machine-learning-derived values are within 25% of the DFT $+U$-calculated band gap width [20].

1.2.4 Open Quantum Materials Database

The Open Quantum Materials Database (OQDM) [3, 4] is a free and available database of DFT-calculations. It has included thermodynamic and structural properties of more than 600.000 materials, including all unique entries in the Inorganic Crystal Structure Database (ICSD) consisting of less than 34 atoms.

The DFT calculations are performed with the VASP software whereas the electron exchange and correlation are described with the GGA-PBE, while using the PAW potentials. They relax a structure using $4000 - 8000$ k-point mesh, indicating an even increasing computational expensive calculation than AFLOW again. Several element-specific settings are included such as using the $+U$ extension for various transition metals, lanthanides and actinides.

In addition, any calculation containing 3d or actinide elements are spin-polarized with a ferromagnetic alignment of spins to capture possible magnetism. However, the authors note that this approach does not capture complex magnetic, such as antiferromagnetism, which has been found to result in substantial errors for the formation energy [21].

1.2.5 JARVIS

Joint Automated Repository for Various Integrated Simulations (JARVIS) [22] - DFT is an open database based based on the VASP software to perform a variety of material property calculations. It consists of roughly 40.000 3D and 1.000 2D materials using the vdW-DF-OptB88 van der Waals functional, which was originally designed to improve the approximation of properties of two-dimensional van der Waals materials, but has also shown to be effective for bulk materials [23, 24]. The functional has shown accurate predictions for lattice-parameters and energetics for both vDW and non-vdW bonded materials [25].

Structures included in the data set are originally taken from the materials project, and then re-optimized using the OPT-functional. Finally, the combination of the OPT and modified Becke-Johnson (mBJ) functionals are used to obtain a representative band gap of each structure, since both have shown unprecedented accuracy in the calculation of band gap compared to any other DFT-based calculation methods [26].

The JARVIS-DFT database is part of a bigger platform that includes JARVIS-FF, which is the evaluation of classical forcefield with respect to DFT-data, and JARVIS-ML, which consists of 25 machine learning to predict properties of materials. In addition, JARVIS-DFT also includes a data set of 1D-nanowire and 0D-molecular materials, yet not publically distributed.

Chapter 2

Extraction and featurization of data

The main work of this thesis can visualised through the flowchart in figure (TODO: add flowchart figure). Initially, we start by extracting all entries in the Materials Project that matches a specific query. Thereafter, we apply matminer's featurization tools to make thousands of features of the data. In a parallel step, entries that are deemed similar to the entries from the initial Materials Project query are extracted from AFLOW, AFLOW-ML, JARVIS-DFT and OQMD. Finally, we combine the steps together and prepare the data for further analysis.

The initial query has the requirement that all entries has to be derived from an experimental ICSD entry, and is reasoned by that we can identify equivalent entries in other databases. Furthermore, all entries in the Materials Project needs to have a band gap larger than 0.1eV. Recall that Materials Project applies the functional GGA in estimating the band gap, which is known to severely underestimate the given electronic property. Therefore, we have a chosen a low value to not rule out any potential candidates but high enough to leave out all materials that can be considered metallic.

2.1 Practical data extraction with Python-examples

For this section, we will show practical examples of how to extract data that fulfill the criteria for a material to host a qubit candidate given in the theory part.

We will begin with the database of Materials Project, and then restrict the query thereafter. This data mining process is reproducable as a jupyter notebook[1] and the databases in question are the ones refered to in the previous section.

Instead of setting up a multiple HTTP-methods, we will here take a look at

[1]add and insert DOI for JN data-mining.ipynb

the easiest method at obtaining data from each database. This includes looking into the APIs that supports data-extraction and that are recommended by each respective database.

The range of data in a database can consist of data from a few entries up to an unlimited amount of entries with even further optional parameters, and has limitless use in applications. However, the amount of data in a database is irrelevant if the data is inaccessible. Therefore, we provide a qualitative guide in how to extract information in the easiest way possible. The guide will focus on three main bulletpoints; accessibility, speed of extraction and versatility of API. Additionally, the guide will make the reader aware of the current state of documentation that exists of each database.

2.1.1 Materials Project

The most up-to-date version of Materials Project can be extracted using the python package pymatgen, which is integrated with Materials Project REST API. Other retrievel tools that is dependent on pymatgen includes matminer, with the added functionality of returning a pandas dataframe. Copies of Materials Project are added frequently to cloud services such as Citrine Informatics, but the latest added entries to Materials Project cannot be guaranteed in such a query.

Entries in Materials Project are characterized using more than 60 features[2], some features being irrelevant for some materials while fundamental for others. The data is divided into three different branches, where the first can be described as basic properties of materials including over 30 features, while the second branch describes experimental thermochemical information. The last branch yields information about a particular calculation, in particular information that's relevant for running a DFT script.

To extract information from the database, we will be utilising the module pymatgen. This query supports MongoDB query and projection operators[3], resulting in an almost instant query. Thus, Materials Project is regarded as a highly accessible database.

1. Register for an account[4], and generate a secret API-key.
2. Install pymatgen in the current environment
3. Import pymatgen.
4. Set the required critera.

[2]All features can be viewed in the documentation of the project: https://github.com/materialsproject/mapidoc/master/materials

[3]https://docs.mongodb.com/manual/reference/operator/query/

[4]https://materialsproject.org

5. Set the wanted properties.

6. Apply the query.

The code nippet in code listing 2.1 resembles steps $3-6$, with the additional usage of the library Pandas. For this particular query we have set the filter as four items. Firstly, we would like to exclude all spin zero isotopes using the MongoDB operator that matches non of the values specified in the array. Thereafter, we would like to have a compound that is deemed similar to an ICSD entry. All of the resulting entries have be deemed non-magnetic (NM), and lastly, all compounds with polar space groups will be excluded.

```python
from pymatgen import MPRester
from pandas import DataFrame

# Insert secret API-key inside MPRester in quotes
with MPRester() as mpr:

    criteria = {"elements":{"$nin":exclude_spin_zero_isotopes
    },
                    "icsd_ids": {'$gte': 0},
                    "magnetic_type": {"$eq": "NM"},
                    "spacegroup.number": {"$nin":polar_spacegroups
    }}

    props = ["material_id", "icsd_ids",
             "spacegroup", "band_gap"]

    entries = DataFrame(mpr.query(criteria=criteria,
                                  properties=props))
```

Listing 2.1: Practical example of extracting information from Materials Project using pymatgen, resulting in a Pandas DataFrame named entries that contains the properties given after performing a filter on the database. The criteria is given as a JSON, and supports MongoDB operators.

2.1.2 Citrine Informatics

Citrine Informatics is a cloud service, which means that the spectrum of stored information varies broadly. We will access research through open access for institutional and educational purposes. Information in Citrine can be stored using a scheme that is broken down into two sections, with private properties for each entry in addition to common fields that are the same for all entries. However, the query happens swiftly and is noted as highly accessible.

In this example, we will gather experimental data using the module matminer. The following steps are required to extract information from Citrine Informatics.

1. Register for an account[5], and generate a secret API-key.
2. Install matminer in the current environment.
3. Import matminer.
4. Set the required critera.
5. Set the wanted properties and common fields.
6. Apply the query.

The code listed in code listing 2.2 gives an easy example to steps $2-4$ with experimental data as filter. The resulting query will be returned as a Pandas DataFrame, but it is not neccessary to include the pandas since it is already implemented in the module matminer.

```
# Insert secret API-key inside CitrineDataRetrieval in quotes
cdr = CitrineDataRetrieval()

criteria = {'data_type': 'EXPERIMENTAL'}
properties = ['Band gap']
common_fields = ['chemicalFormula', "references", "Structure",
    "Crystallinity", "Crystal structure", "uid"]

experimental_entries = cdr.get_dataframe(criteria = criteria,
                                         properties = properties,
                                         common_fields = common_fields)
```

Listing 2.2: Practical example of extracting information from Citrine Informatics using matminer, resulting in a Pandas DataFrame named experimental_entries that contains the properties given after performing a filter on the database. The criteria is given as a JSON.

2.1.3 AFLOW

The query from AFLOW API [14] supports lazy formatting, which means that the query is just a search and does not return values but rather an object. This object is then used in the query when asking for values. For every object it is neccessary to request the desired property, consequently making

[5]https://citrination.com

the query process significantly more time-demanding than similar queries using APIs such as pymatgen or matminer for Citrine Informatics. Hence, the accessibility is strictly limited to either searching for single compounds or if the user possess sufficient time.

Matminer's data retrievel tool for AFLOW is currently an ongoing issue [27], thus we present in code listing 2.3 a function that extracts information from AFLOW and returns a Pandas DataFrame. In contrast to Materials Project and Citrine Informatics, AFLOW does not require an API-key for a query, which reduces the amount of steps to obtain data.

```python
# Tables
import pandas as pd

# Visual representation of the query process
from tqdm import tqdm

# Query library
from aflow import *

import os
if not os.path.exists('data'):
    os.makedirs('data')

def get_dataframe_AFLOW(compound_list, keys, batch_size):
    """
    A function used to make a query to AFLOW.
    ...
    Args
    ----------
    compound_list : list (dim:N)
        A list of strings containing full formula, eg. ["H2O1
    ", "Si1C1"]
    keys : list (dim:M)
        A list containing the features of the compound one
    wants to extract,
        eg. Egap
    batch_size : int
        Number of data entries to return per HTTP request

    Returns
    -------
    pd.DataFrame (dim:MxN)
        A DataFrame containing the resulting matching queries.
     This can result
        in several matching compounds
    """
    index = 0
    for compound in tqdm(compound_list):
        print("Current query: {}".format(compound))

        results = search(catalog='icsd', batch_size=batch_size)\
            .filter(K.compound==compound)

        # If search returns matching query
        if len(results)>0:
            for result in tqdm(results):
                for key in keys:
                    try:
                        aflow_dict[key].append(getattr(result, key))
                    except:
                        aflow_dict[key].append("None")

                pd.DataFrame.from_dict(aflow_dict)
                  .loc[[index]].to_csv("data/AFLOW_DATA_temp.csv",
                sep=",",
                index=False,
                header=False,
                mode='a')
                index += 1
        else:
            print("No compound is matching the search")
            continue

    return pd.DataFrame.from_dict(aflow_dict)
```

Listing 2.3: Practical example of extracting information from AFLOW. The function can extract all information in AFLOW for a given list of compounds, however, it is a slow method and requires consistent internet connection.

2.1.4 AFLOW-ML

In this part, we will be using a machine learning algorithm named AFLOW-ML Property Labeled Material Fragments (PLMF) [19] to predict the band gap of structures. This algorithm is compatible with a POSCAR of a compound, which can be generated by the CIF (Crystallographic Information File) that describes a crystal's generic structure. It is possible to download a structure as a poscar by using Materials Project front-end API, but is a cumbersome process to do so individually if the task includes many structures. Extracting the feature of POSCAR is yet to be implemented in the RESful API of pymatgen, thus we demonstrate the versatility of pymatgen with a workaround.

We begin with extracting the desired compounds formula, its material_id for identification, and their respectful structure in CIF-format from Materials Project. In an iterative process, each CIF-structure is parsed to a pymatgen structure, where pymatgen can read and convert the structure to a POSCAR stored as a Python dictionary. Finally, we can use the POSCAR as input to AFLOW-ML, which will return the predicted band gap of the structure. This iterative process parsing and converting, but is an undemanding process. The function that handles this is presented in code listing 2.4.

A significant portion of the process is tied up to obtaining the input-file for AFLOW-ML, and fewer structures will result in an easier process. Nevertheless, we present the following steps in order to receive data from AFLOW-ML.

1. Download AFLOWmlAPI[6].
2. Getting POSCAR from MP.
   (a) Apply the query from Materials Project with "CIF", "material_id" and "full_formula" as properties.
   (b) Insert resulting DataFrame into function defined in code listing 2.4.
3. Insert POSCAR to AFLOW-ML.

```python
# Tables
import pandas as pd

# Visual representation of the query process
from tqdm import tqdm

# ML library and structural library
from aflowml.client import AFLOWmlAPI

from pymatgen import Structure
from pymatgen.io.vasp.inputs import Poscar
from pymatgen.io.cif import CifParser

import os
if not os.path.exists('data'):
    os.makedirs('data')

def get_dataframe_AFLOW_ML(entries, pathAndFileName = False):
    """
        A function used to initialise AFLOW-ML with appropiate inputs.
        ...
        Args
        ----------
        entries : Pandas DataFrame
        {
            "cif": {}
```

[6]http://aflow.org/src/aflow-ml/ to the same directory as code listing 2.4

```
            - Materials Project parameter "cif", which is a dict
        "compound": []
            - list of strings
        "material id": []
            - list of strings
    }
    fileName : str
        Path to file, e.g. "data/aflow_ml.csv"
        Writing to a file during iterations. Recommended for large entries.

    Returns
    -------
    Pandas DataFrame
        A DataFrame containing features as compound and material id,
        as well as the keys in the AFLOW-ML algorithm Property
        Labeled Material Fragments.
    """
    def writeToFile(fileName, row):
        pd.DataFrame.from_dict(AFLOW_ML).loc[[index]].to_csv(fileName,
            sep=",",
            index=False,
            header=False,
            mode='a')

    firstIteration = True
    for index, entry in tqdm(entries.iterrows()):

        struc = CifParser.from_string(entry["cif"]).get_structures()[0]

        poscar = Poscar(structure=struc)

        ml = AFLOWmlAPI()
        prediction = ml.get_prediction(poscar, 'plmf')

        if firstIteration:
            AFLOW_ML = {k: [] for k in prediction.keys()}
            AFLOW_ML["full_formula"]    = []
            AFLOW_ML["material_id"] = []
            firstIteration = False

        for key in prediction.keys():
            AFLOW_ML[key].append(prediction[key])

        AFLOW_ML["full_formula"].append(entry["full_formula"])
```

```
                AFLOW_ML["material_id"].append(entry["material_id"
    ])

                if (pathAndFileName):
                    writeToFile(pathAndFileName, row=pd.DataFrame.
    from_dict(AFLOW_ML).loc[[index]])

        return pd.DataFrame.from_dict(AFLOW_ML)
```

Listing 2.4: Practical example of extracting information from AFLOW-ML. The function will convert a CIF-file (from e.g. Materials Project) to a POSCAR, and will use it as input to AFLOW-ML. In return, one will get the structure's predicted band gap. It should be noted that this requires the AFLOW-ML library in the same directory.

2.1.5 JARVIS-DFT

The newest version of the JARVIS-DFT dataset can be obtained by requesting an account at the official webpage, but with the drawback that an administrator has to either accept or deny the request. Thus, the accessibility of the database is dependent on if there is an active administrator paying attention to the requests. Another approach is to download the database through matminer, however with the limitation of not neccessarily having the latest version of the database. The following steps describes the process of extracting JARVIS-DFT using matminer's convenience loader module, and can be regarded as easily accessible with few lines of code and instantanous download.

1. Install matminer in the current environment.
2. Import matminer.
3. Load the dataset using code listing 2.5.

```
from matminer.datasets.convenience_loaders import
    load_jarvis_dft_3d

JARVIS_entries = load_jarvis_dft_3d(drop_nan_columns=["gap
    tbmbj"])
```

Listing 2.5: Practical example of extracting information from JARVIS-DFT. For this example, we exclude all metals by removing all non-measured band gaps.

We can observe that there is no advanced search filter when loading the database from matminer. The author of matminer regards this as the user's task, which is easily done through the use of the python library Pandas.

2.2 Matminer featurization

Before applying any machine learning algorithm, raw data needs to be transformed into a numerical representation that reflects the relationship between the input and output data. This transformation is known as generating descriptors or features, however, we will in this work adapt the name *featurization*. The open source library of Matminer provides tools to featurize existing features extracted from Materials Project. In this section we will describe how to extract the features from an initial Materials Project query result (see subsection. 2.1.1), and the resulting features. It is beyond the scope of this work to go in-depth of each feature since the resulting dataset contains a quantity of several thousand features, but we will here take the liberty to serve a brief overview of the features and refer to each respective citation for more information.

Table 2.1: Feasible triples for highly variable Grid, MLMMH.

1a	**2a**	**3a**	**4a**
Features	Description	Original reference	Feature reference
Composition features			
AtomicOrbitals	Highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO).	[28]	-
AtomicPacking-Efficiency	Packing efficiency.	[29]	-
BandCenter	Estimation of absolute position of band center using geometric mean of electronegativity.	[30]	-
ElementFraction	Fraction of each element in a composition.	-	-

Continued on next page

Table 2.1 – continued from previous page

1	2	3	4
ElementProperty	Statistics of various element properties. (From preset("magpie"))	[10, 31, 32]	-
IonProperty	Maximum and average ionic character.	[31]	-
Miedema	Formation enthalpies of intermetallic compounds, solid solutions, and amorphous phases using semi-empirical Miedema model.	[33]	-
Stoichiometry	L^p norm-based stoichiometric attributes.	[31]	-
TMetalFraction	Fraction of magnetic transition metals.	[32]	-
ValenceOrbital	Valence orbital attributes such as the mean number of electrons in each shell.	[31]	-
YangSolidSolution	Mixing thermochemistry and size mismatch terms of Yang and Zhang (2012).	[34]	-
Oxid composition features			

Continued on next page

Table 2.1 – continued from previous page

1	**2**	**3**	**4**
Electronegativity-Diff	Statistics on electronegativity difference between anions and cations.	[32]	-
OxidationStates	Statistics of oxidation states.	[32]	-
Structure features			
DensityFeatures	Calculate density, volume per atom and packing fraction.	-	-
GlobalSymmetry-Features	Determines spacegroup number, crystal system (1-7) and inversion symmetry.	-	-
RadialDistribution-Function	Calculates the radial distribution function of a crystal system.	-	-
CoulombMatrix	Generate the Coulomb matrix, which is a representation of the nuclear coulombic interaction of the input structure.	[35]	-
PartialRadial-Distribution-Function	Compute the partial radial distribution function of a crystal structure	[36]	-

Continued on next page

Table 2.1 – continued from previous page

1	2	3	4
SineCoulomb-Matrix	Computes a variant of the coulomb matrix developed for periodic crystals.	[37]	-
EwaldEnergy	Computes the energy from Coulombic interactions based on charge states of each site.	[38]	-
BondFractions	Compute the fraction of each bond in a structure, based on nearest neighbours.	[39]	-

2.3 Preprocessing

kaffe

2.4 Screen procedure

kaffe

Part II

Results

Chapter 3

Validation

A thorough testing procedure is important to find out if the code is working as intentionally. The procedure might reveal the presence or absence of bugs, and as a project grows, it can give an indication if a new implementation breaks the original project. Therefore, we present a test-case scenario to test if the two machine learning algorithms are able predict the correct label. It is the same algorithm that will be used in the following chapters, and it will provide us the opportunity to understand how the algorithm works and to draw parallells between the separate works.

The validation process is a reproduction of Ref [40]. To be able to draw any parallell to their work, we use the exact same dataset in the beginning phase. It should be noted that even if the computational aspects of the validation is closely related to Ref. [40], the work eventually diverges in terms of focus. In their work they include a stability analysis using convex hull analysis in DFT calculations from OQMD, however, we will in this thesis not decide whether a compound is considered stable or not in an atomic configuration.

3.1 The perovskite dataset

The dataset in question contains 390 experimentally reported ABO_3 compounds. All compounds are charged balanced, and for every compound there is a feature explaining which structure the compound takes, either being a cubic perovskite, perovskite, or not a perovskite at all. Off the 390 compounds, there are 254 perovskites and 136 non-perovskites. Of the 254 perovskites, 232 takes a non-cubic perovskite structure while only 22 takes the cubic perovskite structure. Consequently, this will be visualized by two columns named Perovskite, which represents if a compound is either perovskite (1) or not perovskite (0), and Cubic, which represents if a compound is cubic perovskite (1), non-cubic perovskite (-1), or not perovskite(0).

3.1.1 Features

There are in total 9 features we can train a model on. Many of the features are based on the Shannon ionic radii [41], which are estimates of an element's ionic hard-sphere radii extracted from experiment. They are dimensionless numbers, and are frequently used in studies involving perovskite structures of materials since they can be a measurement of the ionic misift of the B atom. This can be used to find the deviation of the structure from an ideal cubic geometry. The octahedral factor for an ABO_3 solid is known as

$$O = \frac{r_b}{r_O}, \tag{3.1}$$

where r_b and r_O are the Shannon radii for the B-atom and oxygen ($r_O = 1.4Å$), respectively. If the octahedral factor is $O = 0.435$, it corresponds to a hard-sphere closed-packed arrangement where B and O ions are touching, while a six-fold coordination appear to require $0.414 < O < 0.732$ according to empirical studies [42]. O, r_A and r_b are represented as features in our data set. We can also compute the Goldschmidt tolerance factor, which is defined as

$$t = \frac{r_A + r_O}{\sqrt{2}(r_A + r_O)}. \tag{3.2}$$

The tolerance factor favors the following structures in the interval:

- $t > 1$: Hexagonal nonperovskite.
- $0.9 < t < 1.0$: Cubic perovskite.
- $0.75 < t < 0.9$: Orthorombic perovskite.
- $t < 0.75$: Not a perovskite.

If the tolerance factor is exactly $t = 1$, the structure is known as perfectly cubic and is free for any structural alterations.

Furthermore, the Shannon radii r_A and r_B can be directly correlated with the structure. Perovskites require $r_A > r_B$, and that A-atoms are in a 12-fold coordinated site if $r_A > 0.9Å$. A-atoms also occur in a sixfold coordinated site if $r_A < 0.8Å$ and $r_B > 0.7Å$.

From bond valence theory we can find the valence of an ion to be the sum of valences, that is

$$V_i = \sum_i \nu_{ij} \tag{3.3}$$

$$= \sum_i \frac{\exp(d_o - d_{ij})}{b}, \tag{3.4}$$

where d_{ij} is the bond length while d_0 and b are parameters from experimental data. The bond length can be found from 3.4 given the general value $b = 1.4Å$ and d_0, that can be found from Zhang *et al.* database [42]. The valence of an ion is associated with its neighboring ions and the chemical bonds, and therefore the band length d_{AO} and d_{BO} are included in the data set.

The two last features originates from the Mendeleev numbers of Villars *et al.* [43] for the A- and B atom. The given values positions the elements in structurally similar groups. This means that he groups the elements in the following interval.

- s-block $\in \{1, 10\}$.
- $Sc = 11$.
- $Y = 12$.
- f-block $\in \{13, 42\}$.
- d-block $\in \{43, 66\}$.
- p-block $\in \{67, 10\}$.

Bibliography

1. Kresse, G. & Furthmüller, J. Efficient iterative schemes forab initiototal-energy calculations using a plane-wave basis set. *Physical Review B* **54,** 11169–11186 (Oct. 1996).
2. Battle, R. & Benson, E. Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). *Journal of Web Semantics* **6,** 61–69 (Feb. 2008).
3. Kirklin, S. *et al.* The Open Quantum Materials Database (OQMD): assessing the accuracy of DFT formation energies. *npj Computational Materials* **1.** doi:10.1038/npjcompumats.2015.10 (Dec. 2015).
4. Saal, J. E., Kirklin, S., Aykol, M., Meredig, B. & Wolverton, C. Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD). *JOM* **65,** 1501–1509 (Sept. 2013).
5. Jain, A. *et al.* Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials* **1,** 011002 (July 2013).
6. Ong, S. P. *et al.* The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles. *Computational Materials Science* **97,** 209–215 (Feb. 2015).
7. Levin, I. *NIST Inorganic Crystal Structure Database (ICSD)* en. 2020. doi:10.18434/M32147.
8. Landis, D. D. *et al.* The Computational Materials Repository. *Computing in Science & Engineering* **14,** 51–57 (Nov. 2012).
9. Larsen, A. H. *et al.* The atomic simulation environment—a Python library for working with atoms. *Journal of Physics: Condensed Matter* **29,** 273002 (June 2017).
10. Ong, S. P. *et al.* Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science* **68,** 314–319 (Feb. 2013).

11. Ward, L. *et al.* Matminer: An open source toolkit for materials data mining. *Computational Materials Science* **152,** 60–69 (Sept. 2018).

12. Draxl, C. & Scheffler, M. The NOMAD laboratory: from data sharing to artificial intelligence. *Journal of Physics: Materials* **2,** 036001 (May 2019).

13. Wang, L., Maxisch, T. & Ceder, G. Oxidation energies of transition metal oxides within the GGA+U framework. *Physical Review B* **73.** doi:10.1103/physrevb.73.195107 (May 2006).

14. Curtarolo, S. *et al.* AFLOWLIB.ORG: A distributed materials properties repository from high-throughput ab initio calculations. *Computational Materials Science* **58,** 227–235 (June 2012).

15. Curtarolo, S. *et al.* AFLOW: An automatic framework for high-throughput materials discovery. *Computational Materials Science* **58,** 218–226 (June 2012).

16. Calderon, C. E. *et al.* The AFLOW standard for high-throughput materials science calculations. *Computational Materials Science* **108,** 233–238 (Oct. 2015).

17. Setyawan, W. & Curtarolo, S. High-throughput electronic band structure calculations: Challenges and tools. *Computational Materials Science* **49,** 299–312 (Aug. 2010).

18. Setyawan, W., Gaume, R. M., Lam, S., Feigelson, R. S. & Curtarolo, S. High-Throughput Combinatorial Database of Electronic Band Structures for Inorganic Scintillator Materials. *ACS Combinatorial Science* **13,** 382–390 (June 2011).

19. Isayev, O. *et al.* Universal fragment descriptors for predicting properties of inorganic crystals. *Nature Communications* **8.** doi:10.1038/ncomms15679 (June 2017).

20. Ferrenti, A. M., de Leon, N. P., Thompson, J. D. & Cava, R. J. Identifying candidate hosts for quantum defects via data mining. *npj Computational Materials* **6.** doi:10.1038/s41524-020-00391-7 (Aug. 2020).

21. Stevanović, V., Lany, S., Zhang, X. & Zunger, A. Correcting density functional theory for accurate predictions of compound enthalpies of formation: Fitted elemental-phase reference energies. *Physical Review B* **85.** doi:10.1103/physrevb.85.115104 (Mar. 2012).

22. Choudhary, K. *et al.* JARVIS: An Integrated Infrastructure for Data-driven Materials Design. arXiv: `2007.01831v1 [cond-mat.mtrl-sci]` (July 3, 2020).

23. Thonhauser, T. *et al.* Van der Waals density functional: Self-consistent potential and the nature of the van der Waals bond. *Physical Review B* **76.** doi:10.1103/physrevb.76.125112 (Sept. 2007).

24. Klimeš, J., Bowler, D. R. & Michaelides, A. Van der Waals density functionals applied to solids. *Physical Review B* **83.** doi:10.1103/physrevb.83.195131 (May 2011).

25. Choudhary, K., Cheon, G., Reed, E. & Tavazza, F. Elastic properties of bulk and low-dimensional materials using van der Waals density functional. *Physical Review B* **98.** doi:10.1103/physrevb.98.014107 (July 2018).

26. Choudhary, K. *et al.* Computational screening of high-performance optoelectronic materials using OptB88vdW and TB-mBJ formalisms. *Scientific Data* **5.** doi:10.1038/sdata.2018.82 (May 2018).

27. Rosenbrock, C. W. A Practical Python API for Querying AFLOWLIB. arXiv: 1710.00813v1 [cs.DB] (Sept. 28, 2017).

28. Kotochigova, S., Levine, Z. H., Shirley, E. L., Stiles, M. D. & Clark, C. W. Local-density-functional calculations of the energy of atoms. *Physical Review A* **55,** 191–199 (Jan. 1997).

29. Laws, K. J., Miracle, D. B. & Ferry, M. A predictive structural model for bulk metallic glasses. *Nature Communications* **6.** doi:10.1038/ncomms9123 (Sept. 2015).

30. Butler, M. A. & Ginley, D. S. Prediction of Flatband Potentials at Semiconductor-Electrolyte Interfaces from Atomic Electronegativities. *Journal of The Electrochemical Society* **125,** 228–232 (Feb. 1978).

31. Ward, L., Agrawal, A., Choudhary, A. & Wolverton, C. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials* **2.** doi:10.1038/npjcompumats.2016.28 (Aug. 2016).

32. Deml, A. M., O'Hayre, R., Wolverton, C. & Stevanović, V. Predicting density functional theory total energies and enthalpies of formation of metal-nonmetal compounds by linear regression. *Physical Review B* **93.** doi:10.1103/physrevb.93.085142 (Feb. 2016).

33. Weeber, A. W. Application of the Miedema model to formation enthalpies and crystallisation temperatures of amorphous alloys. *Journal of Physics F: Metal Physics* **17,** 809–813 (Apr. 1987).

34. Yang, X. & Zhang, Y. Prediction of high-entropy stabilized solid-solution in multi-component alloys. *Materials Chemistry and Physics* **132,** 233–238 (Feb. 2012).

35. Rupp, M., Tkatchenko, A., Müller, K.-R. & von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Physical Review Letters* **108.** doi:10.1103/physrevlett.108.058301 (Jan. 2012).

36. Schütt, K. T. *et al.* How to represent crystal structures for machine learning: Towards fast prediction of electronic properties. *Physical Review B* **89.** doi:10.1103/physrevb.89.205118 (May 2014).

37. Faber, F., Lindmaa, A., von Lilienfeld, O. A. & Armiento, R. Crystal structure representations for machine learning models of formation energies. *International Journal of Quantum Chemistry* **115,** 1094–1101 (Apr. 2015).

38. Ewald, P. P. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik* **369,** 253–287 (1921).

39. Hansen, K. *et al.* Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space. *The Journal of Physical Chemistry Letters* **6,** 2326–2331 (June 2015).

40. Balachandran, P. V. *et al.* Predictions of new ABO3 perovskite compounds by combining machine learning and density functional theory. *Physical Review Materials* **2.** doi:10.1103/physrevmaterials.2.043802 (Apr. 2018).

41. Shannon, R. D. Revised effective ionic radii and systematic studies of interatomic distances in halides and chalcogenides. *Acta Crystallographica Section A* **32,** 751–767 (Sept. 1976).

42. Zhang, H., Li, N., Li, K. & Xue, D. Structural stability and formability of ABO3-type perovskite compounds. *Acta Crystallographica Section B Structural Science* **63,** 812–818 (Nov. 2007).

43. Villars, P., Cenzual, K., Daams, J., Chen, Y. & Iwata, S. Data-driven atomic environment prediction for binaries using the Mendeleev number. *Journal of Alloys and Compounds* **367,** 167–175 (Mar. 2004).