Manual for: Extensible Structure-Informed Prediction of Formation Energy with Improved Accuracy and Usability employing Neural Networks

Adam M. Krajewski[a], Jonathan W. Siegel [b]
Jinchao Xu [b], Zi-Kui Liu[a]

a. Department of Materials Science and Engineering, The Pennsylvania State University, USA

b. Department of Mathematics, The Pennsylvania State University, USA

Corresponding author: A. M. Krajewski, 216-456-1534, ak@psu.edu

October 15, 2020

Contents

1 Introduction

1.1 Publication Abstract

Pre-Print: arxiv.org/abs/2008.13654 [1]

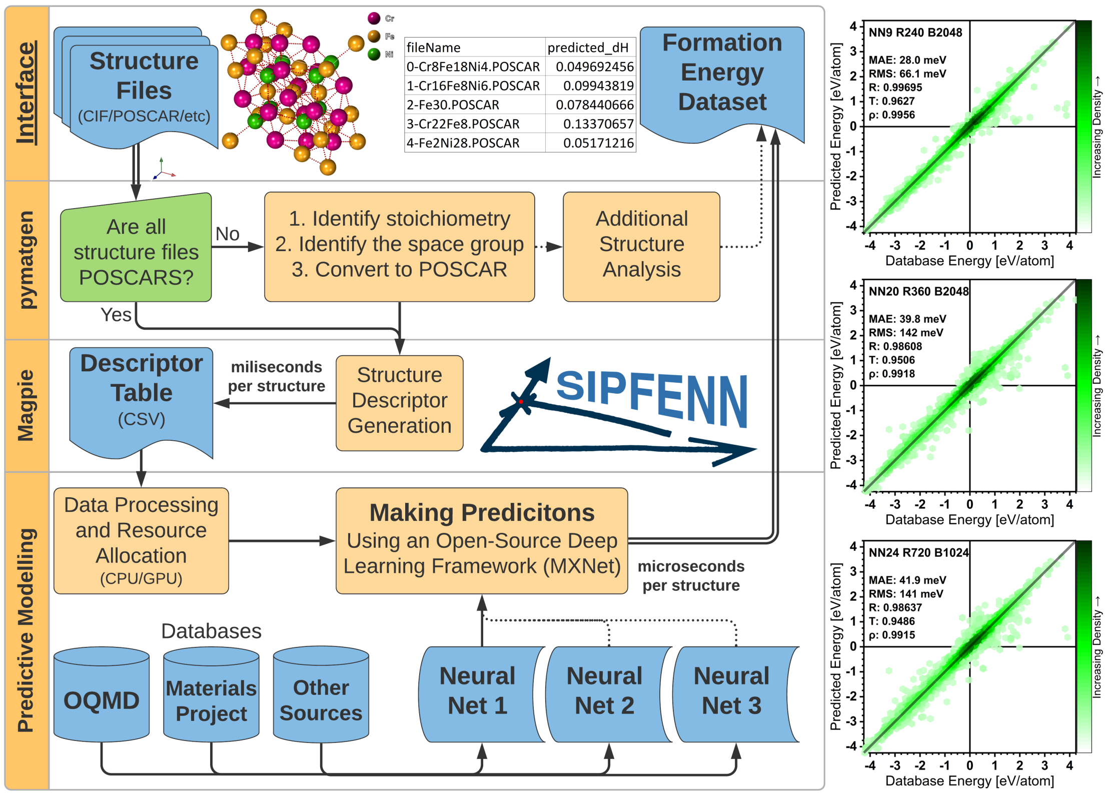
In recent years, numerous studies have employed machine learning (ML) techniques to enable orders of magnitude faster high-throughput materials discovery by augmentation of existing methods or as standalone tools. In this paper, we introduce a new neural network-based tool for the prediction of formation energies based on elemental and structural features of Voronoi-tessellated materials. We provide a self-contained overview of the ML techniques used. Of particular importance is the connection between the ML and the true material-property relationship, how to improve the generalization accuracy by reducing overfitting, and how new data can be incorporated into the model to tune it to a specific material system.

In the course of this work, over 30 novel neural network architectures were designed and tested. This lead to three final models optimized for (1) highest test accuracy on the Open Quantum Materials Database (OQMD), (2) performance in the discovery of new materials,

and (3) performance at a low computational cost. On a test set of 21,800 compounds randomly selected from OQMD, they achieve mean average error (MAE) of 28, 40, and 42 meV/atom respectively. The second model provides better predictions on materials far from ones reported in OQMD, while the third reduces the computational cost by a factor of 8.

We collect our results in a new open-source tool called SIPFENN (Structure-Informed Prediction of Formation Energy using Neural Networks). SIPFENN not only improves the accuracy beyond existing models but also ships in a ready-to-use form with pre-trained neural networks and a user interface.

1.2 Graphical Abstract (Operation Schematic)



fileName	predicted_dH
0-Cr8Fe18Ni4.POSCAR	0.049692456
1-Cr16Fe8Ni6.POSCAR	0.09943819
2-Fe30.POSCAR	0.078440666
3-Cr22Fe8.POSCAR	0.13370657
4-Fe2Ni28.POSCAR	0.05171216

Figure 1.1: (Left) Schematic of SIPFENN operation depicting how user can interact with the software and what processes happen within it. (Right) Performance (on a random OQMD-subset) of 3 neural networks described in the publication, top-to-bottom: Standard Materials Model, Light Model, and Novel Materials Model

1.3 Software Access

The most recent version of SIPFENN code is available through Penn State's Phases Research Lab website at www.phaseslab.com/sipfenn in (1) a minimal version that can be run on precomputed descriptors in CSV format as well as (2) ready-to use version with pre-compiled Magpie [2]. SIPFENN contains hard coded links to neural networks stored in cloud that can be downloaded at a single-click (see Section 3.2) or directly from psu.box.com/v/SIPFENN-NeuralNets. All neural networks are stored both in (1) open-source MXNet format maintained by the Apache Foundation and used within SIPFENN, and in (2) closed-source WLNet format maintained by Wolfram Research and having advantage of even easier deployment, as well as guaranteed forward compatibility with future versions of Wolfram Language. For further ensured longevity of results, SIPFENN neural networks are also stored at the CERN's Data Centre through the courtesy of Zenodo.org service under doi:10.5281/zenodo.4006803.

2 Installation

At the time of writing, SIPFENN is shipped as a versioned ZIP file. There are two versions available, (1) containing SIPFENN source code only, that allows user to run predictions based on descriptor files (see Section 3.4), and (2) also including a pre-compiled Magpie [2] software, thus having all of the code required for its operation including descriptor generation (see Section 3.3). Option number 2 is preferred for simplicity, and it should run regardless of operating system, however may not work on every machine (e.g. ARM-based computers such as smartphones). Please remember to cite accordingly.

2.1 Basic

SIPFENN installation has been made very simple and doesn't require administrator privileges, as to simplify the deployment on high performance computing (HPC) machines. One simply unpacks the ZIP file into a desired location, later called "SIPFENN folder".

Before software can be run, a Python environment, in which it will operate, will require other dependencies. They are listed in the "requirements.txt" file and can be quickly fetched using the standard Python package manager **pip** by opening the command line (a.k.a. terminal / command prompt), going to the SIPFENN folder, and typing:

```
pip install –r requirements.txt
```

This should install all dependencies and allow the user to move directly to running the software (see Section 3). If there are any issues, they often stem from how the Python environment was configured on the given machine and can often be quickly solved by doing a clean installation (see Section 2.2).

2.2 With Anaconda

To do a clean install of SIPFENN, the user is recommended to use Anaconda. It is a free Python distribution available at https://www.anaconda.com, which allows the creation of environments managed by the conda package manager.

Once Anaconda is installed, to install SIPFENN dependencies, go to the SIPFENN folder and type in the following commands into the command line to create a new environment (named e.g. myenv):

```
conda create –name myenv
conda install -n myenv pip
conda activate myenv
conda install –channel conda-forge pymatgen
pip install –r requirements.txt
```

On some Mac computers, some extra steps described in Section 5.2 may be needed to run the GUI correctly.

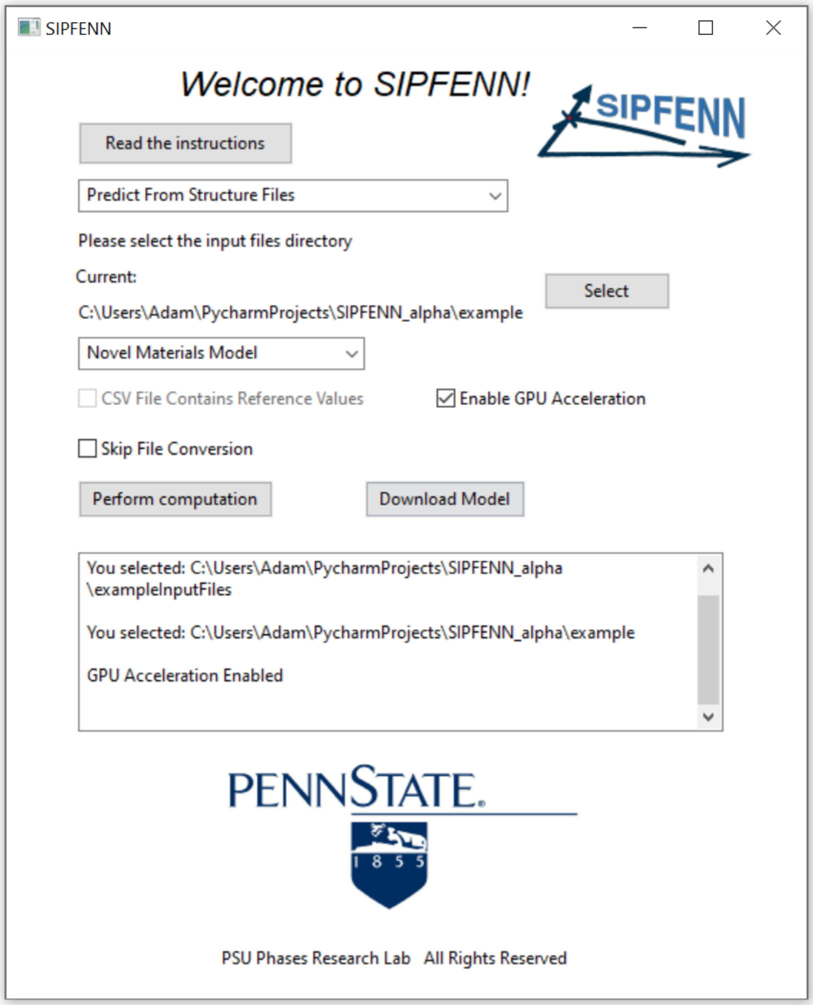
3 Use

3.1 Start

Once SIPFENN (with required packages) is installed, it can be run by going to SIPFENN folder within command line (a.k.a terminal) and running:

```
python SIPFENN.py
```

This should bring up the SIPFENN Graphical User Interface (GUI) such as the one in Figure 3.1 below. The exact look will depend on the machine and operating system.



Predict From Structure Files

Please select the input files directory

Current:
C:\Users\Adam\PycharmProjects\SIPFENN_alpha\example

Novel Materials Model

☐ CSV File Contains Reference Values ☑ Enable GPU Acceleration

☐ Skip File Conversion

```
You selected: C:\Users\Adam\PycharmProjects\SIPFENN_alpha
\exampleInputFiles

You selected: C:\Users\Adam\PycharmProjects\SIPFENN_alpha\example

GPU Acceleration Enabled
```

Figure 3.1: SIPFENN Graphical User Interface (GUI)

3.2 Models Download

The final step before predictions can be made is the network download. The easiest way to accomplish this is to (1) select the desired network from a drop-down menu, and (2) download the model, as depicted in Figure 3.2.

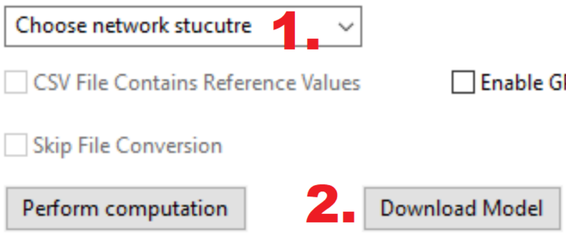
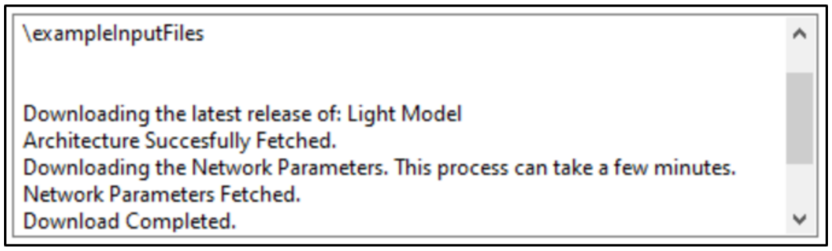


Figure 3.2:

The network download process will now begin, as indicated in the SIPFENN log window (e.g. Figure 3.3).

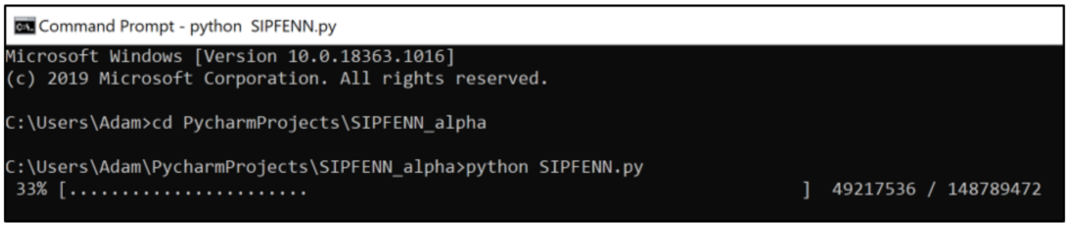


```
\exampleInputFiles


Downloading the latest release of: Light Model
Architecture Succesfully Fetched.
Downloading the Network Parameters. This process can take a few minutes.
Network Parameters Fetched.
Download Completed.
```

Figure 3.3:

Progress of the download can be traced in real time in the command line window which was used to start SIPFENN (e.g. Figure 3.4). In the authors' experience, download speeds from the default server have been on the order of 5 MB/s, what corresponded to 30 seconds download time for Light Model network and 4 minutes for the other two models.



```
Command Prompt - python  SIPFENN.py
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Adam>cd PycharmProjects\SIPFENN_alpha

C:\Users\Adam\PycharmProjects\SIPFENN_alpha>python SIPFENN.py
 33% [.......................                                  ]  49217536 / 148789472
```

Figure 3.4:

If there is any issue with either running the above procedure or the GUI is not accessible, there are two other sources of neural networks used within SIPFENN as described in Section 1.3. The most reliable solution is to download networks from CERN-hosted Zenodo repository, where networks are published under doi:10.5281/zenodo.4006803. An alternative (and often faster) option is to fetch them from psu.box.com/v/SIPFENN-NeuralNets. For normal SIPFENN operation, download and use files with ".json" and ".param" extensions.

If networks are manually downloaded, their names need to be manually adjusted to work with the GUI. The required name format is: architecture-trainingRounds-batchSize.extension (e.g. "NN20-R720-B2048.json"). The command line always requires a manual network name input, so in this case a name change is not required.

3.3 Predictions from Structure Files

Performing predictions from structure files is the fundamental functionality of SIPFENN. It can accept any periodic arrangement of atoms and then calculate the descriptor (see [3] for details) using Magpie [2].

3.3.1 Accepted Formats

SIPFENN imports structure files using the Pymatgen [4] python package and accepts any format supported by it. For a complete list of over 100 supported (or possibly supported) formats refer to https://pymatgen.org. Table 1 lists a few common structure files that are a subset of those accepted by SIPFENN out-of-the-box.

.cif .mcif
.POSCAR .CONTCAR .vasp .CHGCAR
.json .mson
.yaml
.xsf
.cssr

Table 1: Selection of supported structure file formats. For a complete list refer to https://pymatgen.org

Within SIPFENN, all of these structure file formats are interpreted and then exported into a unified dataset of structures in **POSCAR** format. A POSCAR file contains all information defining a crystal in the following form:

```
Zn4 Pd22
1.0
-4.551100 4.551100 4.551100
4.551100 -4.551100 4.551100
4.551100 4.551100 -4.551100
Pd Zn Pd
10 4 12
direct
0.344379 0.343775 0.343863 Pd
0.666830 0.005434 0.005443 Pd
0.005788 0.666491 0.005671 Pd
0.005510 0.005617 0.666819 Pd
0.008960 0.359592 0.359530 Pd
0.007361 0.651440 0.652104 Pd
0.359350 0.359500 0.008882 Pd
0.651953 0.651326 0.007384 Pd
0.359296 0.008562 0.359166 Pd
0.652068 0.007640 0.651957 Pd
0.788173 0.789024 0.788552 Zn
0.199957 0.011813 0.011755 Zn
```

```
0.011331 0.200140 0.011360 Zn
0.011793 0.011940 0.199951 Zn
0.657594 0.657697 0.390615 Pd
0.344909 0.742695 0.008495 Pd
0.742442 0.344405 0.008912 Pd
0.280319 0.280563 0.612913 Pd
0.657686 0.389927 0.657728 Pd
0.344873 0.009207 0.742790 Pd
0.280662 0.612568 0.280595 Pd
0.742771 0.009147 0.344883 Pd
0.390318 0.657618 0.657616 Pd
0.612900 0.280432 0.280268 Pd
0.008974 0.344526 0.742312 Pd
0.008560 0.742640 0.345022 Pd
```

The detailed description of meaning of each line can be found in the VASP Guide at https://cms.mpi.univie.ac.at/vasp/guide/node59.html.

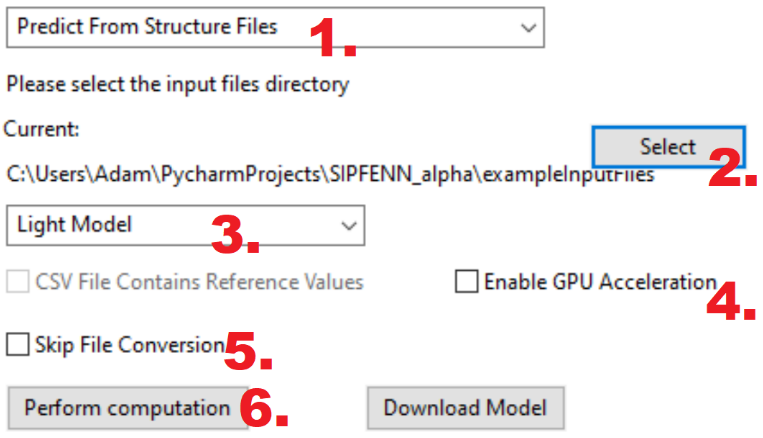
3.3.2 How to Create Structure Files From Scratch

Given intuitive and straightforward formatting of structure files like POSCARs (see Section 3.3.1 above), they can be relatively easily created even in a text editor, and then saved into a text file with a correct extension. Alternatively, one can use some more robust software such as CrystalMaker [5] (paid and popular in academia) or VESTA [6] (free for non-commercial users).

3.3.3 Running Predictions from GUI

Running predictions from structure files using GUI is very straightforward. First, create a folder with only the structure files of interest. It needs to contain only structure files and/or ".txt" files describing the contents, as SIPFENN will attempt to run predictions on all files other than ".txt", since accepted structure file formats are very numerous.

Once files are prepared, SIPFENN can be run on structure files through GUI by: (1) selecting the "Predict From Structure Files" mode, (2) selecting the folder with structure files, (3) selecting the model (neural network) to be used, (4) if CUDA-capable GPU is available, possibly enabling it, (5) if all structure files are POSCAR, possibly skipping the file conversion for time savings, and (6) engaging the predictions, as depicted in Figure 3.5 below.



Please select the input files directory

Current:

C:\Users\Adam\PycharmProjects\SIPFENN_alpha\exampleInputFiles

Light Model

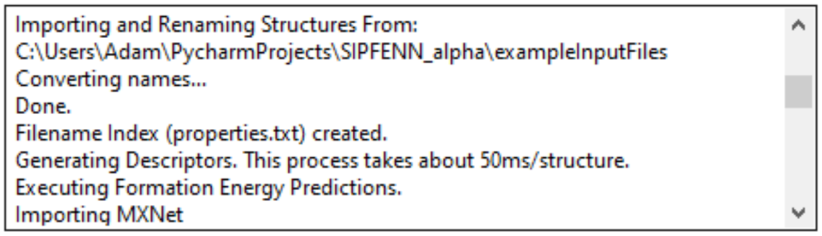
CSV File Contains Reference Values

Enable GPU Acceleration

Skip File Conversion

Figure 3.5:

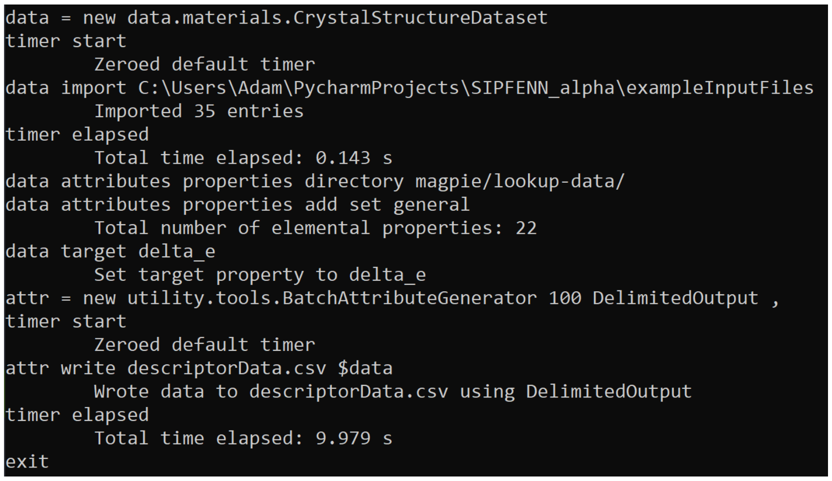
Once engaged, SIPFENN will import structures, analyze them, and convert them to POSCAR, giving feedback like that in Figure 3.6.



```
Importing and Renaming Structures From:
C:\Users\Adam\PycharmProjects\SIPFENN_alpha\exampleInputFiles
Converting names...
Done.
Filename Index (properties.txt) created.
Generating Descriptors. This process takes about 50ms/structure.
Executing Formation Energy Predictions.
Importing MXNet
```

Figure 3.6:

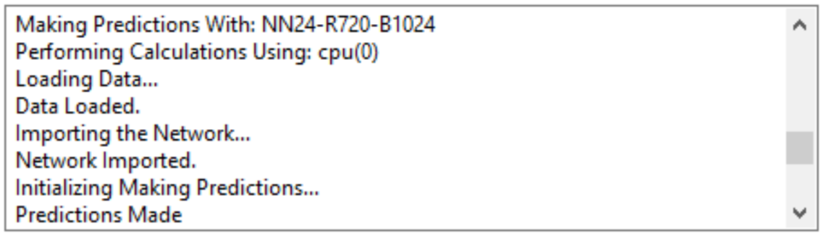
Once converted, SIPFENN will run Magpie to generate the CSV descriptor file. The time intensity of this process scales roughly linearly with the number of atoms in the structure (with some constant per-structure overhead). On the testing laptop PC, this process took approximately 50 ms per average OQMD structure and 1500 ms per 500-atom super-cell. Progress (or issues) can be tracked in the command line window. Expected output is shown in Figure 3.7. It is important to verify whether this step was completed successfully, as it can often fail if Java is not installed or up-to-data (see Section 5.1 for troubleshooting).



```
data = new data.materials.CrystalStructureDataset
timer start
        Zeroed default timer
data import C:\Users\Adam\PycharmProjects\SIPFENN_alpha\exampleInputFiles
        Imported 35 entries
timer elapsed
        Total time elapsed: 0.143 s
data attributes properties directory magpie/lookup-data/
data attributes properties add set general
        Total number of elemental properties: 22
data target delta_e
        Set target property to delta_e
attr = new utility.tools.BatchAttributeGenerator 100 DelimitedOutput ,
timer start
        Zeroed default timer
attr write descriptorData.csv $data
        Wrote data to descriptorData.csv using DelimitedOutput
timer elapsed
        Total time elapsed: 9.979 s
exit
```

Figure 3.7:

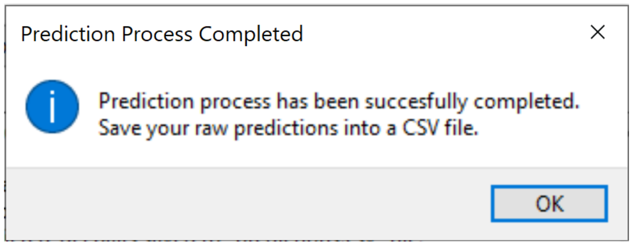
In the last step, SIPFENN will run one of the neural networks on the generated descriptor CSV table. Progress can be tracked in the SIPFENN log window, and should look like what's shown in Figure 3.8.



```
Making Predictions With: NN24-R720-B1024
Performing Calculations Using: cpu(0)
Loading Data...
Data Loaded.
Importing the Network...
Network Imported.
Initializing Making Predictions...
Predictions Made
```

Figure 3.8:

If all SIPFENN operations are completed successfully, predicted values of formation energy (in eV/atom) will be generated and SIPFENN will prompt the user to (as in Figure 3.9) save them into a CSV file in a desired location.



Prediction Process Completed

Prediction process has been succesfully completed.
Save your raw predictions into a CSV file.

Figure 3.9:

While the user is prompted to save the file with the minimum amount of generated information (i.e. the predicted formation energies, described in Section 4.1), two other files are generated and saved inside the SIPFENN folder. The "predictions.csv" (described in Section 4.2) containing corresponding file names, and the relatively large "descriptor.csv" (described in Section 4.3) containing a table will descriptors of all structures from the folder selected earlier.

3.3.4 Running Predictions from Command Line

One of the main reasons behind creation of SIPFENN was to enable formation energy predictions within other software, thus almost all functionalities can be run from command line in a very efficient way. Furthermore, for some users running SIPFENN on a server or other remote machine, a command line interface may be a necessity.

Whatever the reason, one first imports the script designed for predicting from structure files called **predictFromPOSCAR.py** by opening Python Console inside the SIPFENN folder or another python script (within SIPFENN folder), and typing:

```
import predictFromPOSCAR
```

Now, four key settings need to be set in the same fashion to what has been described in Section 3.3.3, namely path to the folder with structures, neural network (model) name, gpu usage setting, and whether structure need to be converted to POSCARs. Examples of all four are:

```
path = 'C:/Users/Adam/Projects/SIPFENN_Beta/test5_NN20'
net = 'NN20-R720-B2048'
gpu = 0
conv = 1
```

Now, predictions can be made by calling the **run()** function of **predictFromPOSCAR**. This can be done, with settings predefined as above, by running:

```
predictions = predictFromPOSCAR.run(path,net,gpu,conv)
```

or by passing settings explicitly

```
predictions = predictFromPOSCAR.run('C:/Users/Adam/Projects/SIPFENN
_beta/test5_NN20','NN20-R720-B2048',0,1)
```

This should perform all predictions, while printing feedback similar to what was shown in Section 3.3.3 in Figures 3.6, 3.7, and 3.8. Some additional information may also be generated, such as warnings regarding versions of packages, depending on how system console is configured.

Running the above code returns a table with minimum amount of generated information (i.e. the predicted formation energies, described in Section 4.1). This table can be used by another software (if running within another script) or saved by the user. A quick way to save the raw results as a table that can be later opened in, e.g. Excel, is to import **savetxt** from **numpy**:

```
from numpy import savetxt
```

and save predictions with a quick command

savetxt('myAmazingPredictions.csv', predictions, delimiter=',')

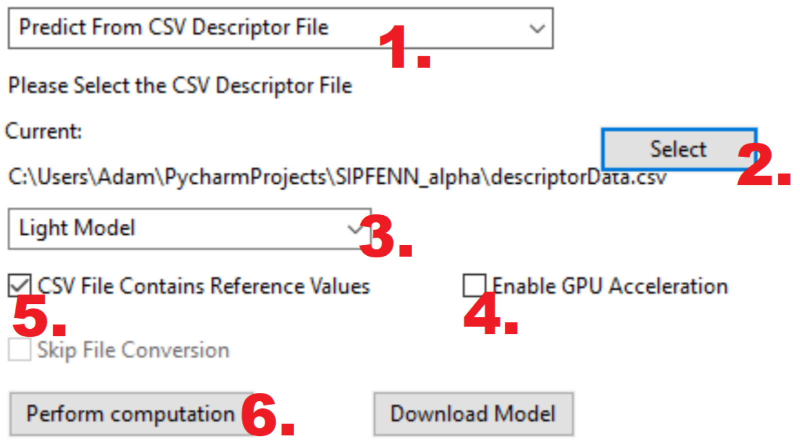
Furthermore, in the background, two files are generated and saved inside the SIPFENN folder. The "predictions.csv" (described in Section 4.2) containing corresponding file names, and relatively large "descriptor.csv" (described in Section 4.3) containing a table will descriptors of all structures from the selected folder.

3.4 Predictions from Pre-Calculated Descriptor Files

As stressed in the publication [1] and indicated in the graphical abstract in Figure 1.1, descriptor calculation is much more computationally expensive than making predictions using a neural network. Therefore, it is often desirable to re-use calculated descriptors to speed up calculations, where, for example different models are used, or the descriptor space is to be explored for inverse design. Like the core functionality of predictions from structure files, this one can also be run from both GUI and command line.

3.4.1 Running Predictions from GUI

Predictions in this mode are run like in Section 3.3.3, by (1) selecting the "Predict From CSV Descriptor File" mode, (2) selecting the CSV descriptor file, (3) selecting the model (neural network) to be used, (4) if CUDA-capable GPU is available, selecting whether to enable it, (5) selecting whether the CSV file contains reference values for formation energy. They are located in the last column of the file, namely "delta_e", and can be added manually to a descriptor file generated when running predictions from structure files (see Section 3.3.3 or 3.3.4). Lastly, predictions are engaged by (6) pressing "Perform Computation". All steps are depicted in Figure 3.10.


Predict From CSV Descriptor File

Please Select the CSV Descriptor File

Current:

C:\Users\Adam\PycharmProjects\SIPFENN_alpha\descriptorData.csv

Light Model

[✓] CSV File Contains Reference Values

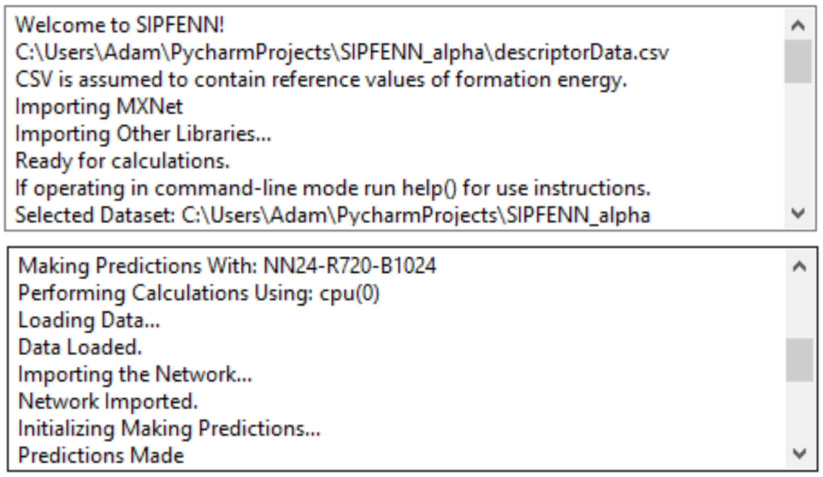
[ ] Enable GPU Acceleration

[ ] Skip File Conversion

Figure 3.10:

Like when running predictions based on structures, feedback is printed in the log window.

It is however shorter, since steps like structure file conversion and descriptor generation are skipped, and thus not reported. Example log from successful computation is shown in Figure 3.11

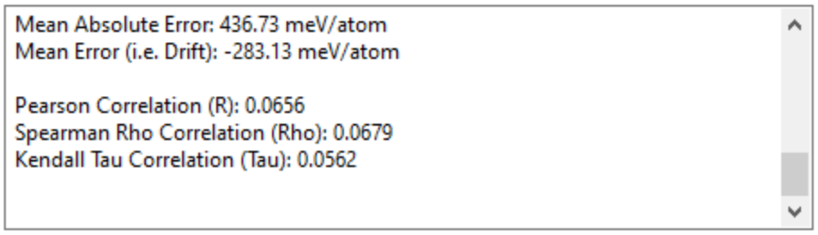


```
Welcome to SIPFENN!
C:\Users\Adam\PycharmProjects\SIPFENN_alpha\descriptorData.csv
CSV is assumed to contain reference values of formation energy.
Importing MXNet
Importing Other Libraries...
Ready for calculations.
If operating in command-line mode run help() for use instructions.
Selected Dataset: C:\Users\Adam\PycharmProjects\SIPFENN_alpha
```

```
Making Predictions With: NN24-R720-B1024
Performing Calculations Using: cpu(0)
Loading Data...
Data Loaded.
Importing the Network...
Network Imported.
Initializing Making Predictions...
Predictions Made
```

Figure 3.11:

If the "CSV File Contains Reference Values" has been checked (see pt.5 in Figure 3.10), model performance will be evaluated with regard to the reference data in the descriptor file and performance information will be reported. Example of the reported statistics (for simulated data) are shown in Figure 3.12.

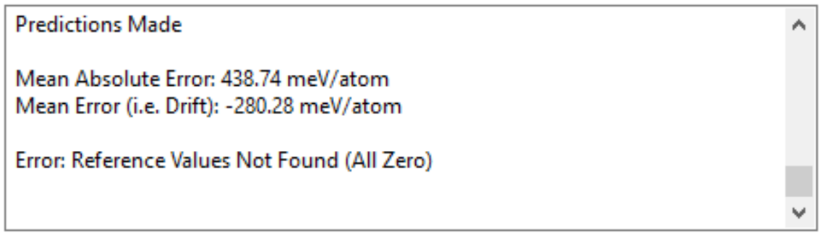


```
Mean Absolute Error: 436.73 meV/atom
Mean Error (i.e. Drift): -283.13 meV/atom

Pearson Correlation (R): 0.0656
Spearman Rho Correlation (Rho): 0.0679
Kendall Tau Correlation (Tau): 0.0562
```

Figure 3.12:

If "CSV File Contains Reference Values" has been checked, but SIPFENN detects that all reference values are *exactly* 0, indicating that reference values were not inserted to the descriptor file, it will calculate the error (in this case mean formation energy across the dataset), but will skip undefined correlation statistics and return a message like in Figure 3.13.



```
Predictions Made

Mean Absolute Error: 438.74 meV/atom
Mean Error (i.e. Drift): -280.28 meV/atom

Error: Reference Values Not Found (All Zero)
```

Figure 3.13:

3.4.2 Running Predictions from Command Line

Predictions in this mode are run like in Section 3.3.4. In this case the script to be ran is called **predictFromCSV.py**, so one opens the Python Console inside the SIPFENN folder, or another python script (within SIPFENN folder) and types:

```
import predictFromCSV
```

Now, four key settings need to be set in the same fashion to what has been described in Section 3.3.4. Like there, the first three are: path to the folder with structures, neural network (model) name, gpu usage setting. In this case, however, the last setting relates to whether the descriptor file contains reference values (explained in Section 3.4.1. Examples for all four are:

```
path = 'C:/Users/Adam/Projects/SIPFENN_Beta/descriptorData.csv'
net = 'NN20-R720-B2048'
gpu = 0
ref = 0
```

Now, predictions can be made by calling the **run()** function of **predictFromCSV**. This can be done, with settings predefined as above, by running:

```
predictions = predictFromCSV.run(path,net,gpu,ref)
```

or by passing settings explicitly

```
predictions = predictFromCSV.run('C:/Users/Adam/Projects/SIPFENN
_beta/descriptorData.csv','NN20-R720-B2048',0,1)
```

This should perform all predictions, while printing feedback similar to what was shown in Section 3.4.1 in Figure 3.11. Some additional information may also be generated, such as warnings regarding versions of packages, depending on how system console is configured.

Running the above code returns a table with minimum amount of generated information (i.e. the predicted formation energies, described in Section 4.1). This table can be used by another software (if running within another script) or saved by the user. A quick way to save the raw results as a table that can be later opened in, e.g. Excel, is to import **savetxt** from **numpy**:

```
from numpy import savetxt
```

and save predictions with a quick command

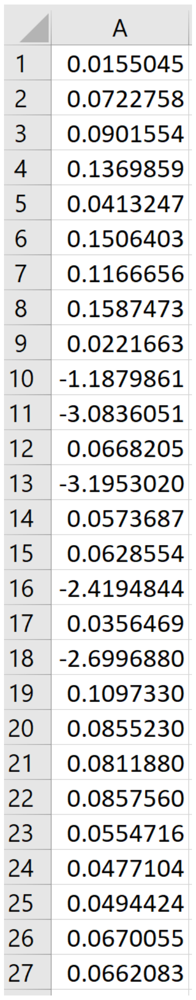
```
savetxt('myAmazingPredictions.csv', predictions, delimiter=',')
```

Unlike in the case of running predictions on structure files, no additional files are generated, since the only input was a descriptor representation of materials.

4 Prediction Results

4.1 Raw Neural Network Output ("rawResults.csv")

The main result of SIPFENN operation, regardless of how it's used, is the result table containing formation energies predicted for each atomic structure file or descriptor table entry. In the default settings, this table of 32-bit floating point numbers (see Figure 4.1) is saved into a CSV file "rawResults.csv", so that it can later be used by user or some software.

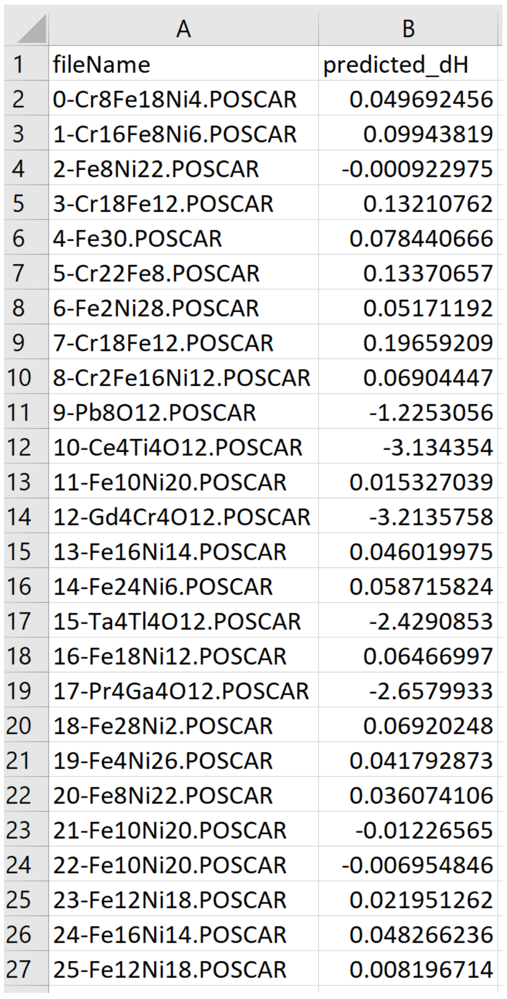


	A
1	0.0155045
2	0.0722758
3	0.0901554
4	0.1369859
5	0.0413247
6	0.1506403
7	0.1166656
8	0.1587473
9	0.0221663
10	-1.1879861
11	-3.0836051
12	0.0668205
13	-3.1953020
14	0.0573687
15	0.0628554
16	-2.4194844
17	0.0356469
18	-2.6996880
19	0.1097330
20	0.0855230
21	0.0811880
22	0.0857560
23	0.0554716
24	0.0477104
25	0.0494424
26	0.0670055
27	0.0662083

Figure 4.1:

4.2 Predictions File (”predictions.csv”)

A more robust representation of the results is stored automatically in ”predictions.csv” file. In the default configuration, as shown in Figure 4.2, it contains both the filenames of structure files used to make predictions and the corresponding formation energies.



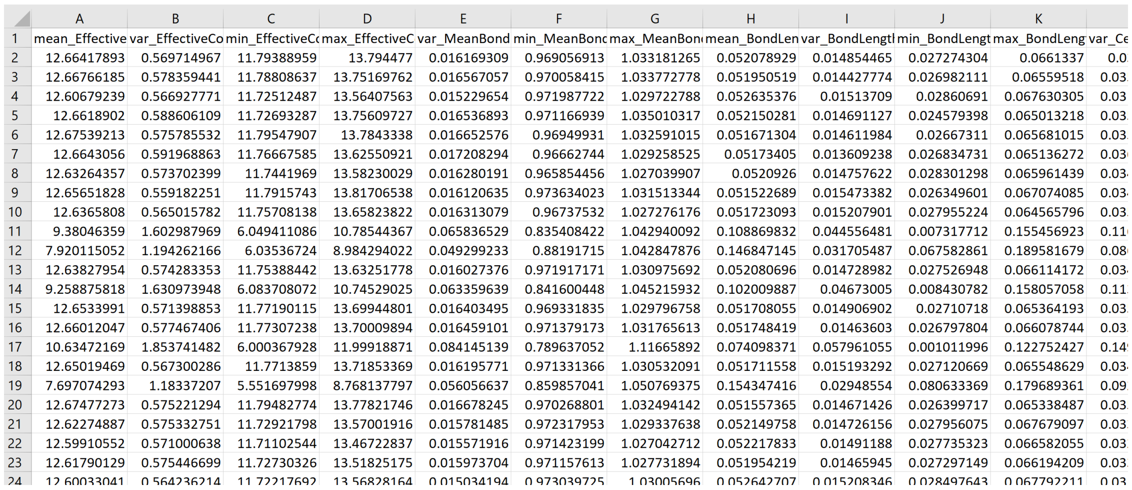
	A	B
1	fileName	predicted_dH
2	0-Cr8Fe18Ni4.POSCAR	0.049692456
3	1-Cr16Fe8Ni6.POSCAR	0.09943819
4	2-Fe8Ni22.POSCAR	-0.000922975
5	3-Cr18Fe12.POSCAR	0.13210762
6	4-Fe30.POSCAR	0.078440666
7	5-Cr22Fe8.POSCAR	0.13370657
8	6-Fe2Ni28.POSCAR	0.05171192
9	7-Cr18Fe12.POSCAR	0.19659209
10	8-Cr2Fe16Ni12.POSCAR	0.06904447
11	9-Pb8O12.POSCAR	-1.2253056
12	10-Ce4Ti4O12.POSCAR	-3.134354
13	11-Fe10Ni20.POSCAR	0.015327039
14	12-Gd4Cr4O12.POSCAR	-3.2135758
15	13-Fe16Ni14.POSCAR	0.046019975
16	14-Fe24Ni6.POSCAR	0.058715824
17	15-Ta4Tl4O12.POSCAR	-2.4290853
18	16-Fe18Ni12.POSCAR	0.06466997
19	17-Pr4Ga4O12.POSCAR	-2.6579933
20	18-Fe28Ni2.POSCAR	0.06920248
21	19-Fe4Ni26.POSCAR	0.041792873
22	20-Fe8Ni22.POSCAR	0.036074106
23	21-Fe10Ni20.POSCAR	-0.01226565
24	22-Fe10Ni20.POSCAR	-0.006954846
25	23-Fe12Ni18.POSCAR	0.021951262
26	24-Fe16Ni14.POSCAR	0.048266236
27	25-Fe12Ni18.POSCAR	0.008196714

Figure 4.2:

The main purpose of this file is to allow easy extension of what is analyzed and stored. Thanks to the numerous functions implemented withing pymatgen [4], with a simple modification of the ”PredictFromPOSCAR.py” script, this file can be populated with many attributes of materials that are interesting to the user, granted that file conversion (resulting in file analysis and conversion) is enabled. In general, additional attributes should be extracted in similar fashion to the Structure.formula (at line 30) and then included in the csv writer (around line 70).

4.3 Descriptor FIle (”descriptorData.csv”)

If SIPFENN is operated in the ”Predict From Structure Files” mode, it will generate a descriptor table and store it in the CSV format in ”descriptorData.csv”. Such descriptor table, part of which is depicted in Figure 4.3, contains all 271 descriptor features (i.e. columns) calculated for every material in the dataset (i.e. row). It also contains an additional column ”delta_e” at the end, which, when generated by SIPFENN, is filled with 0s. It can, however, be later populated by known reference values of formation energy, obtained through DFT or other method, and then used for model accuracy testing, as mentioned in Section 3.4.1.



	A	B	C	D	E	F	G	H	I	J	K	
1	mean_Effective	var_EffectiveCo	min_EffectiveC	max_EffectiveC	var_MeanBond	min_MeanBonc	max_MeanBon	mean_BondLen	var_BondLengtl	min_BondLengt	max_BondLeng	var_Ce
2	12.66417893	0.569714967	11.79388959	13.794477	0.016169309	0.969056913	1.033181265	0.052078929	0.014854465	0.027274304	0.0661337	0.0
3	12.66766185	0.578359441	11.78808637	13.75169762	0.016567057	0.970058415	1.033772778	0.051950519	0.014427774	0.026982111	0.06559518	0.03
4	12.60679239	0.566927771	11.72512487	13.56407563	0.015229654	0.971987722	1.029722788	0.052635376	0.01513709	0.02860691	0.067630305	0.03
5	12.6618902	0.588606109	11.72693287	13.75609727	0.016536893	0.971166939	1.035010317	0.052150281	0.014691127	0.024579398	0.065013218	0.03
6	12.67539213	0.575785532	11.79547907	13.7843338	0.016652576	0.96949931	1.032591015	0.051671304	0.014611984	0.02667311	0.065681015	0.03
7	12.6643056	0.591968863	11.76667585	13.62550921	0.017208294	0.96662744	1.029258525	0.05173405	0.013609238	0.026834731	0.065136272	0.03
8	12.63264357	0.573702399	11.7441969	13.58230029	0.016280191	0.965854456	1.027039907	0.0520926	0.014757622	0.028301298	0.065961439	0.03
9	12.65651828	0.559182251	11.7915743	13.81706538	0.016120635	0.973634023	1.031513344	0.051522689	0.015473382	0.026349601	0.067074085	0.03
10	12.6365808	0.565015782	11.75708138	13.65823822	0.016313079	0.96737532	1.027276176	0.051723093	0.015207901	0.027955224	0.064565796	0.03
11	9.38046359	1.602987969	6.049411086	10.78544367	0.065836529	0.835408422	1.042940092	0.108869832	0.044556481	0.007317712	0.155456923	0.11
12	7.920115052	1.194262166	6.03536724	8.984294022	0.049299233	0.88191715	1.042847876	0.146847145	0.031705487	0.067582861	0.189581679	0.08
13	12.63827954	0.574283353	11.75388442	13.63251778	0.016027376	0.971917171	1.030975692	0.052080696	0.014728982	0.027526948	0.066114172	0.03
14	9.258875818	1.630973948	6.083708072	10.74529025	0.063359639	0.841600448	1.045215932	0.102009887	0.04673005	0.008430782	0.158057058	0.11
15	12.6533991	0.571398853	11.77190115	13.69944801	0.016403495	0.969331835	1.029796758	0.051708055	0.014906902	0.02710718	0.065364193	0.03
16	12.66012047	0.577467406	11.77307238	13.70009894	0.016459101	0.971379173	1.031765613	0.051748419	0.01463603	0.026797804	0.066078744	0.03
17	10.63472169	1.853741482	6.000367928	11.99918871	0.084145139	0.789637052	1.11665892	0.074098371	0.057961055	0.001011996	0.122752427	0.14
18	12.65019469	0.567300286	11.7713859	13.71853369	0.016195771	0.971331366	1.030532091	0.051711558	0.015193292	0.027120669	0.065548629	0.03
19	7.697074293	1.18337207	5.551697998	8.768137797	0.056056637	0.859857041	1.050769375	0.154347416	0.02948554	0.080633369	0.179689361	0.09
20	12.67477273	0.575221294	11.79482774	13.77821746	0.016678245	0.970268801	1.032494142	0.051557365	0.014671426	0.026399717	0.065338487	0.03
21	12.62274887	0.575332751	11.72921798	13.57001916	0.015781485	0.972317953	1.029337638	0.052149758	0.014726156	0.027956075	0.067679097	0.03
22	12.59910552	0.571000638	11.71102544	13.46722837	0.015571916	0.971423199	1.027042712	0.052217833	0.01491188	0.027735323	0.066582055	0.03
23	12.61790129	0.575446699	11.72730326	13.51825175	0.015973704	0.971157613	1.027731894	0.051954219	0.01465945	0.027297149	0.066194209	0.03
24	12.60033041	0.564236214	11.72217692	13.56828164	0.015034194	0.973039725	1.03005696	0.052642707	0.015208346	0.028497643	0.067792211	0.03

Figure 4.3:

5 Possible Issues and Solutions

5.1 Descriptor Generation / Java

In most cases, issues with the descriptor generation step can be traced back to issues with Java installation or configuration. The first step that one should try, is to go to https://www.oracle.com/java/technologies/javase-downloads.html, select the ”JDK Download”, select an installer compatible with the operating system (such as ”Windows x64 Installer”), and install Java JDK.

If issues with Java persist, one can refer to the Magpie publication [2] and related code repository issues page.

5.2 Running GUI on Mac

On some Mac machines, there may be issue with running windowed graphics from the command line (terminal). This issue can be solved by running a clean install described in Section 2.2, but with additional command at the end

conda install python.app

and then, instead of the command from Section 3.1, starting the software with

pythonw SIPFENN.py

5.3 Multi-Sublattice Models

Magpie [2] is implemented in a way that does not recognize atoms of the same element as the same species, if they are given separately within the POSCAR structure file. Example file in 3.3.1 will result in an error, since there are two separate Pd sublattices with occupancies of 10 and 12, rather than a single one with occupancy of 22.

While not radical, this difference will affect predictions, usually on the order of 10 meV/atom, and more fundamentally, the calculated descriptor, since attributes such as "WC_Magnitude" depend on whether two palladiums are recognized as different species.

With any format other than POSCAR the issue will be solved automatically, since file format conversion is forced, and processed files will be formatted as required. With POSCAR, however, conversion is skipped to save time for often pointless import+export operation. This can he changed, thus solving problem, by modifying string '.POSCAR' in line 33 of "PredictFromPOSCAR.py" to any string ''.

6 Acknowledgements

This work was financially supported by the ICDS Seed Grant from the Pennsylvania State University, the Office of Naval Research (ONR) via Contract No. N00014-17-1-2567, the National Science Foundation (NSF) via Grant No. CMMI-1825538, and the Department of Energy (DOE) via Award Nos. DE-FE0031553 and DE-EE0008456.

We would like to thank Zhengqi Liu for his help implementing the graphical user interface, Dr. ShunLi Shang for providing the Fe-Cr-Ni σ-phase dataset, and Brandon Bocklund for providing the SQS dataset.

7 References

References

[1] A. M. Krajewski, J. W. Siegel, J. Xu, and Z.-K. Liu, "Extensible Structure-Informed Prediction of Formation Energy with Improved Accuracy and Usability employing Neural Networks," Tech. Rep., 2020.

[2] L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton, "A general-purpose machine learning framework for predicting properties of inorganic materials," *npj Computational Materials*, vol. 2, 8 2016.

[3] L. Ward, R. Liu, A. Krishna, V. I. Hegde, A. Agrawal, A. Choudhary, and C. Wolverton, "Including crystal structure attributes in machine learning models of formation energies via Voronoi tessellations," *Physical Review B*, vol. 96, no. 2, 7 2017.

[4] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, "Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis," *Computational Materials Science*, vol. 68, pp. 314–319, 2 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025612006295

[5] CrystalMaker Software Ltd, "CrystalMaker," Oxford, England. [Online]. Available: www.crystalmaker.com

[6] K. Momma and F. Izumi, "VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data," *Journal of Applied Crystallography*, vol. 44, no. 6, pp. 1272–1276, 12 2011. [Online]. Available: http://scripts.iucr.org/cgi-bin/paper?db5098