# MPhys Project - Machine Learned Potentials For MD Simulations


## University of Exeter EMPS

**Natan Szczepaniak**
**Supervisor: Prof. Saverio Russo**


## Week 2 (01/06/20-05/06/20)

## Contents


- **Day 5**
  - 5.0 - [3] Machine Learning Potentials for atomistic simulations (continuation)
    - 5.0.2 - Atom centered symmetry functions
    - 5.0.3 - ML Potentials Overview
    - 5.0.4 - Discussion
  - 5.1 - Summary
  - 5.2 - Correspondence with Prof. Saverio Russo
  - 5.3 - Correspondence with Prof. Matthew Bates
  - 5.4 - NNP Implementation Search
  - 5.5 - Correspondence with Dr. Jörg Behler
- **Day 6**
  - 6.1 - Response from Dr. Jörg Behler
  - 6.2 - Brief for Today
  - 6.3 - RuNNer-Rev1_1-Slides.pdf Documentation
    - 6.3.1 - Mode 1
    - 6.3.2 - Mode 2
    - 6.3.3 - Mode 3
- **Day 7**
  - 7.1 - SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials
    - 7.1.1 - Background
    - 7.1.2 - Theory
    - 7.1.3 - Neural Network Optimisation
    - 7.1.4 - Code
    - 7.1.5 - SiO2 Example
  - 7.2 - Meetings
- **Day 8**
  - 8.1 - Simple-NN (Behler-Method Python Implementation) Setup
    - 8.1.1 - Running SIMPLE-NN on Windows
    - 8.1.2 - Running SIMPLE-NN on Raspberry Pi 4
    - 8.1.3 - Running SIMPLE-NN on AWS EC2
  - 8.2 - Email to Prof. Russo about access to University Servers
- **Day 9**
  - 9.1 - VirtualBox Virtual Machine SIMPLE-NN (Ubuntu 20.04)
  - 9.2 - VirtualBox Virtual Machine SIMPLE-NN (Ubuntu 18.04)
- **Day 9.5** (Finished Running)
  - 9.5.1 - Finished Running the SIMPLE-NN example code

# Day 5 - 01/06/20

# 5.0 [3] Machine Learning Potentials for atomistic simulations Analysis

## 5.0.1 The Role of a Descriptor

The output of the ML potential method is a PES that is invariant under translation and rotation of the system. Cartesian coordinates do not are not invariant under translations. This requires us to find a way to describe the system in an invariant way. The answer is descriptors.

Since the inception of NN Potentials nearly 25 years ago, the biggest roadblock the field faced were suitable input descriptors for a large number of atoms. In the recent years, beginning with Behler's 2007 proposition there has been a few more that have been thought of. The requirement for a descriptor is to

- Atom Centered Symmetry Functions
- Bispectrum of the neighbour density
- Smooth Overlap of atomic positions
- Coulomb matrix

I am omitting the bottom 3 for simplicities sake for now. I would rather understand the underlying concept of the most simple and fundamental one (ACSF) before moving onto the others.

## 5.0.2 Atom centered symmetry functions

The method of Atom centered symmetry functions is exactly the same as described in the 2007 paper analysed earlier. Specifics of this method are outlined in Section 3.1.2, the same exact equations are mentioned.

This paper labels these symmetry functions as (ACSF) Atom Centered Symmetry Functions. These symmetry functions depend on positions of neighbouring atoms up to a cut off radius $R_c$.

This paper is more clear about the $R_s$ parameter in the Radial Function which is not defined in the original paper. $R_s$ is a "shift" vector. The $\eta$ parameter is the width of the gaussian of the symmetry function. Typically a set of $\eta$ values are used to obtain a radial fingerprint.

The paper describes the Angular symmetry function parameters as: $\zeta$ - angular resolution, $\lambda$ - defines positions of extrema of cosine function. (Still unsure what the use of $\lambda$ is)

Each atom has around 50-100 symmetry functions describing it with differing values of $\eta$, $\zeta$, $R_s$ and $\lambda$. The typical cutoff radius $R_c$ is around 6-9$\text{Å}$.

The paper also discusses **"pair centered symmetry functions" (PCSF)** which can also be used and are "equally suitable for obtaining high quality potentials". We will not do that as I dont see any benefits.

## 5.0.3 ML Potentials Overview

In addition to different descriptors that can be used for the input to the ML algorithm, the actual algorithm can also be changed. The main one used by Behler in 2007 was an ANN (Artificial Neural Network). This is quite a primitive ML method however the rise of Deep NNs is bringing it back to relevance.

Different methods that can be used and are being implemented in contemporary research are:

- **Gaussian approximation potentials (GAPs) and kernel methods**
  - These use the Bisepecrum of the neighbour density descriptor and optimise the potential using Gaussian process regression. GAPs are a type of kernel methods (they are a linear combination of some basis functions).
- **Support Vector Machines**
  - This is one of the most popular ML methods however it is usually implemented for classification tasks so it is rarely used in the ML PES field. Instead, Support Vector Regression (SVR) is used instead.
- **Spectral Neighbourhood analysis potential**
  - Same as GAP but bispectrum components are linearly related to the atomic energies. This is similar to the Moment Tensor Potential Example from Section 1.4.1. (Compare with Equation 24 in paper).

### 5.0.4 Discussion

This paper outlined a lot about the field of ML potentials in a very general form. A lot was learned in terms of terminology of the (ACSF) though.

This will be a very good paper to reference back to in the future when we decide which methods we want to look at. For now however, we will stick the the Atom Centered Symmetry Function and Neural Network algorithm. Once explored one implementation of this method and test it for some molecule of interest, we will explore other methods.

## 5.1 Summary

- Write up analysis of "[3] Machine Learning Potentials for atomistic simulations"
- "Admin Day"
- Called Saverio
  - Proposition of another topic denied
  - Find implementation of code and explain for wednesday
- Called Bates
  - Topic Set in stone
  - OneDrive Unclear
  - Viva unclear
  - Relieved that we could hypothetically work on this in the summer
  - Should get an email soon with specfics
- Found PES-Learn (Born-Oppenheimer approximation code)
- Found Simple-NN implementation which is Behlers method created in python using tensorflow and mpi4py. It uses LAMMPS for simulation. (IMPORTANT) this is the code we should try understand.
- Found RuNNer which is Behler's First implementation of the method using code. This However is privately owned by Behler
- Sent email to Jorg Behler
- Formatted last week's Lab book
- Sent Saverio List of papers we used

## 5.2 Correspondence with Prof. Saverio Russo

We started off the day with an MS Teams meeting with our Supervisor Prof. Russo. Due to earlier discussion, Prof. Russo suggested we could change the topic of our project to something more "hands-on" however we argued against this as we agree as a group this project is doable.

We discussed how we will use the rest of the time left during these next few days in more detail.

Here are some notes written during the meeting:

- Proposition of investingating HfOx decided not suitable as there is more resources available for current project and don't want to switch topics again.
- Explained current progess by showing notebook
- Final Goal for 3 weeks should be to find and understand a method for doing the task
- "Present the problem then Provide steps to solve problem" very good to keep in mind
- By next meeting try to present an in-depth look at a method (Behler-Parrinello)
- Next meeting Wednesday 11:00

After the meeting, I sent an email to Professor Russo outlining the field we will be researching and papers that we have focused on to bring us on the same page to make communication easier. Here is the email.

> **Dear Saverio,**
>
> I am sending you the list of papers promised during our meeting today.
>
> I know this may find you quite late however I decided to work a bit more on compiling a list of good papers that represent the topic well. I have ordered them in order of complexity as I understand them.
>
> - **Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces http://cacs.usc.edu/education/cs653/Behler-NNPES-PRL07.pdf**
> - **Machine Learning Potentials for atomistic simulations https://aip.scitation.org/doi/full/10.1063/1.4966192**
> - **Machine learning for interatomic potential models https://aip.scitation.org/doi/10.1063/1.5126336**
>
> Additionally, I found 2 implementation of the method we would like to replicate in the form of a python library using tensor >flow that outputs into a LAMMPS simulation. One is privately owned and I have sent the request to get it and this is the other >one. The paper associated is at the top of the page. It is very similar to Behlers method and is documented quite well. As it >stands, this will be the code we will attempt to understand by the end of these 3 weeks.
>
> - **GitHub Repository for SIMPLE-NN (SNU Interatomic Machine-learning Potential package) : https://github.com/MDIL-SNU/SIMPLE-NN**
>
> The Paper we will try to understand for Wednesday's meeting will be Paper 1 from the above list as it is the most accessible >in the shortest time and covers quite a range.
>
> **Kind Regards,**
> **Natan Szczepaniak**

## 5.3 Correspondence with Prof. Matthew Bates

In the morning I got a response from Prof. Bates offering an MS Teams meeting to clarify the details of our concerns.

> **Dear Natan,**
>
> I contacted Saverio on Friday. He said that he told you that Steven Hepplestone and GP Srivastava would be able to help with the programming aspects. Have you tried contacting them?
>
> Regarding the Lab Book assessment next week, the lab book will be assessed (i.e. PDF record of what you have been doing for these three weeks as described in an earlier email). Your supervisor will be one of the two assessors for this Lab Book assessment, and there will be a via which will essentially cover: what the project is about, what progress you have made so far, and aspects of your record keeping in the lab book. The criteria for the lab book are at:
>
> http://newton.ex.ac.uk/handbook/PHY/PhysLogbookAssmnt.html (http://newton.ex.ac.uk/handbook/PHY/PhysLogbookAssmnt.html)
>
> Would you like to have a MS/Teams meeting with me today (or soon)?
>
> Best wishes,
> **Matthew**

Later on in the day we held a meeting with Prof. Bates and the other group members. We discussed that we had a very informative meeting with Prof. Russo and are much more clear about where we stand with the direction of the project.

Another issue brought up was about the hand-in and criteria of the laboratory book and the storage/submission. Prof. Bates himself said that its a thing that he has to "sort out in the next 24 hours".

There is nothing wrong with Lab book entries in July/August.

Should get an email soon with specfics of viva/assessment.

## 5.4 NNP Implementation Research

Implementations of the NNP (Neural-Network Potentials) methods found:

- **PES-Learn**
  - Python implementation of the Born-Oppenheimer approximation using neural networks (research this)
- **SIMPLE-NN**
  - Python implementation of the same method used in the 2007 paper but much more approachable as it utilises the TensorFlow library with the output being in LAMMPS which is a simulation software we were planning on using.
- **RuNNer**
  - Original code written by Behler himself, hosted on GitLab on Behler's private repository. Code dates back to the first implementation of the method described in the 2007 paper.

There are many more however for brevity I decided to include these ones.

## 5.5 Correspondence with Dr. Jörg Behler

After all of the meetings I decided to look for the original implementation of the Behler-Parrinello method described in the 2007 paper (https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.98.146401). After some reserach I found that it was being hosted on a private repository owned by Jörg Behler so I decided to send him an email requesting access.

**Dear Dr. Jörg Behler,**

Hope you are doing well.

I am writing to you regarding your code for the implementation of high-dimensional NN potentials, RuNNer.

My name is Natan Szczepaniak studying for a Physics Master's degree at the University of Exeter in the UK. My research group and I are currently studying the field of Machine Learned Potentials and are in awe of your contribution in this area. We would love to dissect and learn more about the implementation of your method to learn from it in order to hopefully progress the field further. If at all possible, we would like to request a copy of the RuNNer code as from what we have seen, it is the foundation in the field.

Here are my details outlined on the University of Göttingen website:

Name: Natan Szczepaniak
Institution: University of Exeter
Institutional Email Address: ns496@exeter.ac.uk
Gitlab username: @natan.szczepaniak

Kind Regards,
**Natan Szczepaniak**

# Day 6 - 02/06/20

## 6.1 Response from Dr. Jörg Behler

I started off the day by recieving a reply from Dr. Jörg Behler thanking for the interest in RuNNer however there was an issue with the new GitLab account I created for this project. I quickly fixed the issue and sent a follow up email. Here is the correspondence.

Dear Natan,

You should have access now. Any feedback, suggestions or even contributions are very welcome. Enjoy using RuNNer.

Best wishes,
Jörg

Prof. Dr. Jörg Behler
Theoretische Chemie
Institut für Physikalische Chemie
Georg-August-Universität Göttingen
Tammannstr. 6, D-37077 Göttingen
Tel.: +49 (0)551 39-23133
E-mail: joerg.behler @uni-goettingen.de

>

Dear Dr. Jörg Behler,

The account should be working now.

I don't usually use Gitlab and after making the account I didn't complete the email verification which is probably why it wasn't coming up

username: @natan.szczepaniak

Apologies for any inconveniences.

Kind Regards,
Natan Szczepaniak

> Dear Natan,
>
> thank you very much for your interest in RuNNer.
> Unfortunately, I cannot find your username in gitlab. Could it
> be that you registered for github, and not gitlab?
>
> Best wishes, Jörg
>
> Prof. Dr. Jörg Behler
> Theoretische Chemie
> Institut für Physikalische Chemie
> Georg-August-Universität Göttingen
> Tammannstr. 6, D-37077 Göttingen
> Tel.: +49 (0)551 39-23133
> E-mail: joerg.behler @uni-goettingen.de

In the files provided by Dr. Behler was a pdf file with a presentation explaining in detail the specifics of the inner workings of the code described in the 2007 paper. We will use this to try understand the Behler-Parrinello approach in order to either recreate it using python or use this pre-existing and apply it to Peroxides in Dr. Hepplestone's research mentioned in the meeting with our supervisor.

The code is written in Fortran which is a language I nor my partners are familiar with however the documentation is nothing but amazing and clear so this is a very useful resource for our research.

## 6.2 Brief for Today

After yesterday's meeting and correspondance we agreed with the other team members that we will utilise today to prepare for tomorrow's meeting with Professor Russo. As shown previously in the email exchange with Saverio Russo **Section 5.2** the focus of this will be explaining our understanding of the Behler-Parrinello method.

Resources we will focus on are:

- RuNNer-Rev1_1-Slides.pdf
- Generalized Neural-Network Representation of High-Dimensional Potential-EnergySurfaces (https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.98.146401)
- SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials (https://www.sciencedirect.com/science/article/pii/S0010465519301298?via%3Dihub)

The main concept to understand will be the angular symmetry function $G_i^2$ from Section 3.1.

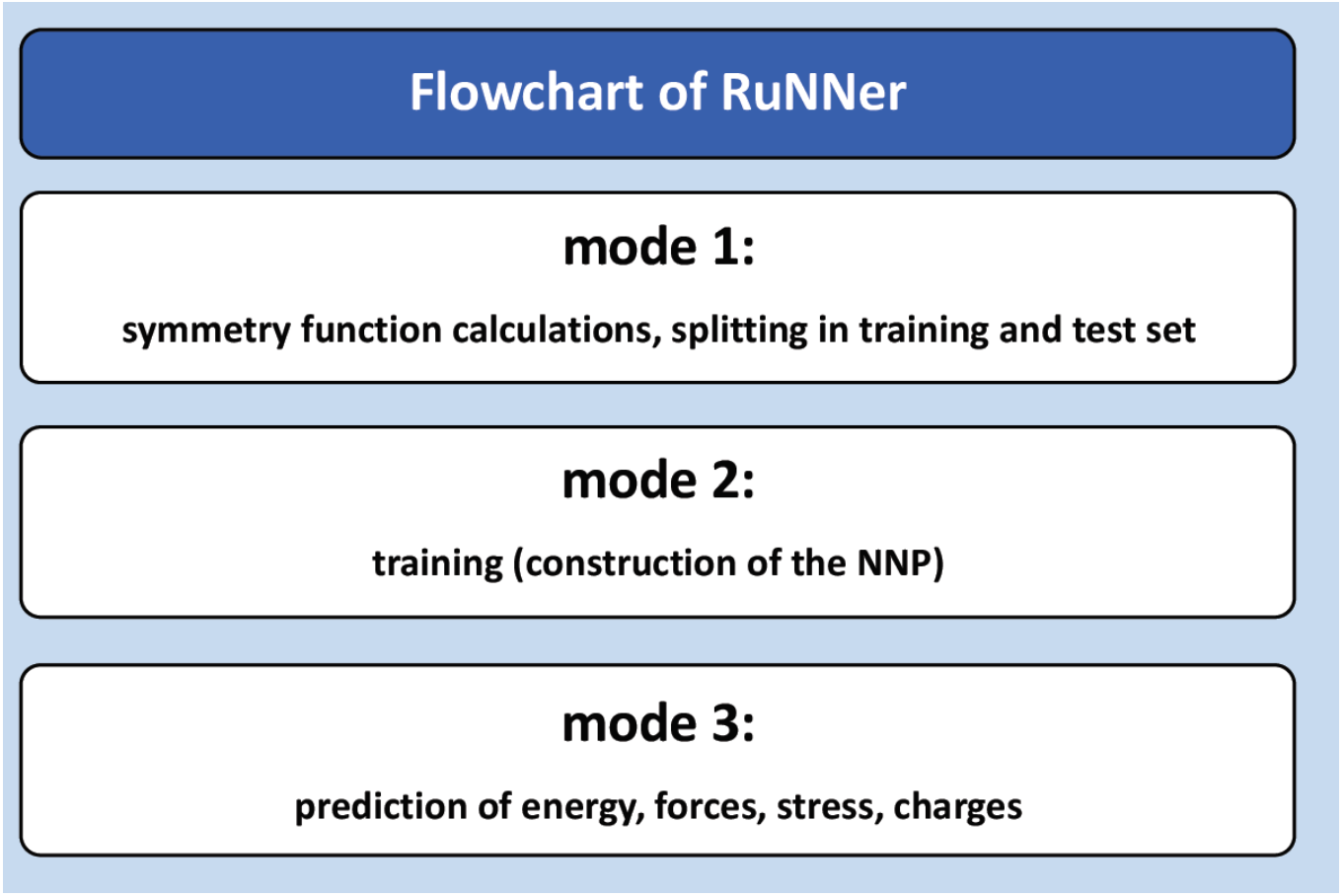
## 6.3 RuNNer-Rev1_1-Slides.pdf

For the use of this documentation, Behler specifies in the manual to cite the following papers whenever using his work.

- J. Behler, Int. J. Quant. Chem. 115, 1032 (2015).
- J. Behler, Angew. Chem. Int. Ed. 56, 12828 (2017).

This file is private and had to be obtained directly from Jorg Behler. It goes through how the RuNNer Behler-Parrinello method was first implemented. This implementation was written in Fortran. I dont have much experience with Fortran so my understanding of this will be quite surface level. This pdf does cover a lot of how the theory meets the program which is worthwhile to look at.

Firstly, it shows the Flowchart of RuNNer, it is rather vague meaning it will apply to other code implementations of this method written in different languages.




The program is seperated into 3 modes. Preparation of data and symmetry functions, training and then predictions of energies, forces, stress and charges which are outputted in a file.

The program has main 2 input files, input.nn and input.data.

- input.nn
  - Control files which contain all parameters for the NN and the symmetry functions.
- input.data
  - Contains structural information such as cartesian coordinates combined with energies, forces and charges. This input data is obtained from DFT based ab-initio method calculations output by by simulators like VASP. Output of these electronic structure programs must be formatted into input.data.

The documentation then outlines the variables used within the input files such as species type and others. Also the use of "grep" is mentioned showing it is meant to be run on a UNIX based system. (I may want to try run this code in the future.)

### 6.3.1 Mode 1

Cutoff function is hardcoded into the program with one input parameter -> cutoff radius. This function is used in the radial and angular function. This cut off radius is always chosen for the right purpose.

Radial function takes parameters: gaussian width $\eta$, radial shift $R_s$ and cutoff radius $R_c$.

Angular function takes parameters: gaussian width $\eta$, angular resolution $\zeta$, elements of neighbouring atoms $\lambda$ (+1 or -1) and cutoff radius $R_c$.

(Use the same $R_c$ for both)

**IMPORTANT FACT ABOUT SYMMETTRY FUNCTIONS**

Symmetry functions $G$ serve as input for the Neural Networks. Number of $G$ needed to describe a system grows with the number of elements N:

**Example: Binary System molecule AB**

# Day 7 - 03/06/20

*"The accuracy of the NN is limited only by that of the training data."*

*"The potential can be systematically improved by properly extending the training set."*

*~ 2007 paper*

# 7.1 SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials

## 7.1.1 Background

**SIMPLE-NN uses the ASE python library found and mentioned last week in Section 2.3 to handle output from ab initio programs like VASP or Quantum espresso**

The abstract of the paper includes a very useful Nature of problem -> Solution method schematic which clarifies the workflow of the project.

**Nature of problem:** Inferencing the potential energy surface for the given system with accuracy comparable to ab initio methods but with much lower computational costs.

**Solution method:** Calculate descriptor vectors that encode local chemical environment. High-dimensional neural network is used to predict the total energy from the descriptor vectors. The trained neural network can be used for molecular dynamics simulations.

The paper is very similar to the Behler-Parinello Paper but written in python and explained in more detail. It gives an example of the code being run for SiO2.

> *Recently, machine learning (ML) based interatomic potentials are gaining attention as they can reproduce potential energy surfaces (PES) of ab initio calculations, with a much lower computational cost. Therefore, an efficient code for training ML potentials and inferencing PES in new configurations would widen the application range of MD simulations.*

The following is undoubtedly the best description of the problem at hand I have read so far

> *Depending on whether the electronic structure is explicitly calculated or not, there are two types of MD: ab initio and classical. The ab initio MD, which is usually based on the density functional theory (DFT), gives accurate and reliable results. However, the method is limited to a system size and timescale less than a few hundreds of atoms and picoseconds, respectively. It is because of the heavy computational costs in solving the Kohn–Sham equation and O(N3) scaling with respect to the number of atoms N. In contrast, the classical MD describes interatomic interactions with a model potential that consists of analytic functions. [11–13] The method is suitable for largescale simulations owing to fast evaluation of the potential energy surface and its gradients, with a linear scaling with the system size. However, construction of the proper model function is a formidable task as it requires long experience in development, as well as deep understanding on the given system.*

Computational cost of ML Potentials is much smaller than the ab initio method but with a similar accuracy.

Two main leaders in the field of ML potentials are artificial neural networks (ANNs) and Gaussian Process Regression (GPR). Currently, ANNs are better suited to create potentials for simulating more complex phenomena however, recent developments in the field have theorised ways of making GPR based packages less computationally demanding.

The paper highlights current implementations of NNP software for MD simulations. As of the publishing of this paper (21/04/19) there are 6 main ones available:

- **RuNNer (Mentioned) - (Dr. Behler's Private GitLab)**
- **aenet (Mentioned) - http://ann.atomistic.net/**
- **Amp - https://bitbucket.org/andrewpeterson/neural/src/master/**
- **DeePMD-kit - https://github.com/deepmodeling/deepmd-kit**
- **ANI-1 - https://github.com/isayev/ASE_ANI**
- **PROPhet - https://biklooost.github.io/PROPhet/**

Each one of these are worth having a look at as they all have something to offer. ANI-1 seems to have a very large dataset of DFT calculations worth having a look at (Mentioned previously). This paper demonstrates another implementation using the same mathematical method with the addition of a gaussian density function $\rho(\mathbf{G})$ done in python using tensorflow. This package is called SIMPLE-NN.

## 7.1.2 Theory

> *Regression model on the relationship between atomic configurations and total energies.*

It uses the same functions as the previous paper. The theory is unchanged from Behler-Parinello approach. The difference in this implementation is that Gaussian Density Function (GDF) is used which resolves the sampling bias. (More on this later)

The cutoff function is defined as previously:

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

The radial symmetry function and angular functions are also the same as previously shown in Section 3.1.

$$G_i^1 = \sum_{j \neq 1}^{all} e^{-\eta(R_{ij} - Rs)^2} f_c(\mathbf{R}_{ij})$$

An addition to this explanation of symmetry functions being used is that for training the neural network, every component of G is scaled into [-1,1]. This is to make the NN inputs cleaner.

## 7.1.3 Neural Network

The neural network architecture is the exact same as the one shown in Section 3.1.2. This is quite nicely generalised into an equation for the propagation of values in the k-th layer to the next.

$$x_j^{k+1} = f\left(\sum_{i=1}^{N^k} x_i^k w_{ij}^k + b\right)$$

where like before, $x_i^k$ is the i-th node in the k-th layer and w^k_{ij} is connection weight between $x_i^k$ and $x_j^{k+1}$. $f$ is the activation function (once again) mentioned before.

Just like in Section 3.1, the total energy is given by summation of all of the atomic energies. The forces are then obtained by differentiating this wrt. position.

The way the NN trains is by minimising the Root Mean Squared Error (RMSE) in the energies and forces scaled by $M$ (total number of structures in training set), $N_i$ number of atoms in the i-th structure and a scaling parameter $\mu$. The loss function looks $\Gamma$ like this:

$$\Gamma = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{E_i^{\mathrm{DFT}} - E_i^{\mathrm{NNP}}}{N_i}\right)^2 + \frac{\mu}{3\sum_{i=1}^{M} N_i}\sum_{i=1}^{M}\sum_{j=1}^{N_i}|\mathbf{F}_{ij}^{\mathrm{DFT}} - \mathbf{F}_{ij}^{\mathrm{NNP}}|^2$$

$$= \mathrm{RMSE(energy)}^2 + \frac{\mu}{3}\mathrm{RMSE(force)}^2,$$

The accuracy of the NN is undermined by biased data that is inputted.

> For example, during MD simulations, atoms vibrate around equilibrium positions most of the time, and so the training set constructed from the MD trajectories is concentrated around specific G's. For another example, defects are usually modeled together with a large number of bulk atoms, so the training set is heavily weighted toward the bulk configuration although description on the defect is also crucial. Atomic NNs trained over such biased datasets retain large errors for the under-sampled configurations, which often lead to catastrophic failure in MD.

This is why the Gaussian density function is introduced. It is used to "cure the sampling bias". I will leave this for now as I don't have enough time to cover this topic however when I get the SIMPLE-NN code to run, I will try to trace the steps of what the code is doing to understand what is exactly happening. It will be better to gain an understanding when I have the code somewhat understood. I doubt I will have time to do this during this 3 week period though.

### 7.1.4 Code

The code is mostly written in python however more computationally demanding parts of it such as calculating descriptors is written in C/C++ to enhance performance. The python side of the code is written with the assistance of Tensorflow which makes is very easy to implement a neural network and tweak its parameters. In addition to this, parallelisation is used by the program via MPI to utilise multiple cores. Tensorflow also makes things more efficient by using the CPU to prepare data and using the GPU to train the network. This way, whilst the GPU is being used to train the network in one epoch, the CPU is preparing the data for the next epoch.

The program takes input from ab initio simulations, processes them using ASE, then passes this information on to create descriptors, descriptors are then split into train/test and fed into the neural network which minimises the RMSE and outputs coefficients of a potential energy surface that can be used in LAMMPS. To make the connection between the two work, SIMPLE-NN offers a new "pair_style" to include into the LAMMPS installation.

Scalability of the system follows the computing cost equation linearly $O(N)$ compared to the ab initio method $O(N^3)$ as mentioned in the previous papers analysed.

The input.yaml file looks like this:

```
generate_features: true
preprocess: true
train_model: true
atom_types:
- Si
- O
symmetry_function:
params:
Si: params_Si
O: params_O
neural_network:
method: Adam
nodes: 30-30
```

In addition to this input file is also the ab-initio calculations used for training as well as the symmetry function parameters for specfic elements "params_Si" and "params_O". (Units used in the SIMPLE-NN output are the equivalent of LAMMPS's "metal" units).

### 7.1.5 SiO2 Example

As a demonstration of the code, the authors of the paper trained the neural net on data obtained from SiO2 simulations using VASP. In the data training set, they included simulations of: three types of crysals, amorphous & liquid phases, crystal structures distorted by isotropic compression/expansion, volume conserving mono-axial strain and shear strain. In total there were 3048 SiO2 structures in the data set used.

For each type of atom, 70 symmetry functions are used. This is shown in the params_Si files which consist of 70 lines of parameters for the symmetry functions. With 30 nodes each per layer, we end up with 3030 weights and 61 biased nodes.

The "Adam" optimisation method within Tensorflow is used and 10% of the dataset is used for validation of the data. This part is what makes it supervised learning. In this part you can also judge how close the fit is to the original data. This is usually a threshold that is embedded in the code which stops the code from running once a specified minimum errror is reached.

Next the output is tested further by using it to run MD simulations. A common tool of comparison in this field is the radial distribution function. Below is an example of how the SIMPLE-NN code compares to ab intitio methods for SiO2 radial and angular functions.

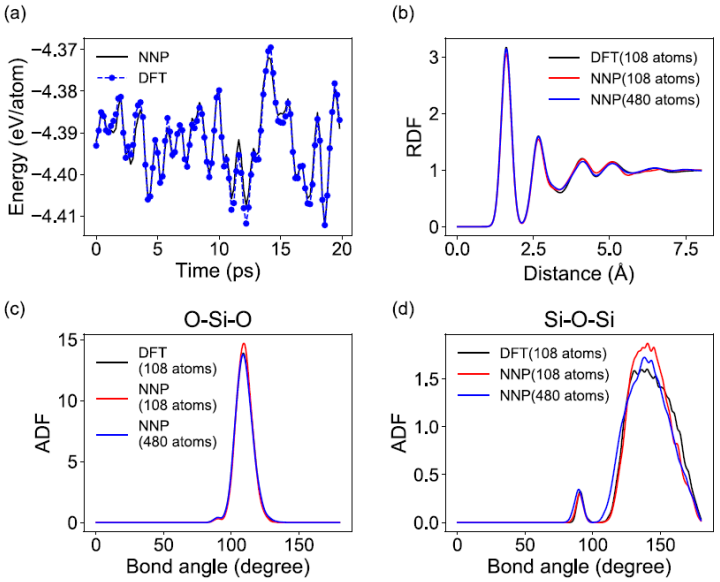



**Fig. 6.** (a) Comparison of NNP (solid line) and DFT (solid disks) energies along the MD trajectory of amorphous $SiO_2$ at 500 K. The trajectory is calculated by NNP and configurations for DFT calculations are sampled every 200 fs. (b)–(d) Comparison of structural properties of amorphous $SiO_2$ between DFT and NNP. The structures are generated by independent melt-quench simulations. (b) Total radial distribution function (RDF), angle distribution functions (ADFs) for (c) O-Si-O and (d) Si-O-Si. For NNP, supercells of two different sizes (108 and 480 atoms) are used.

It is clear that the more atoms used in the NNP code, the more it resembles the DFT ab initio data. For more complex structures however, the authors found that the Adam Optimiser is not sufficient and that the use of L-BFGS is preffered.

In summary, this is a good level of complexity for us to attempt to run this code and implement for our structure of choice.

## 7.2 Meetings

Unfortunately, the meeting at 11:00 had to be moved. As Nathan was unavailable for the rest of the day we moved it until tomorrow.

Max and I held a brief meeting with Prof. Russo at 2 pm where we explained that we are making progress

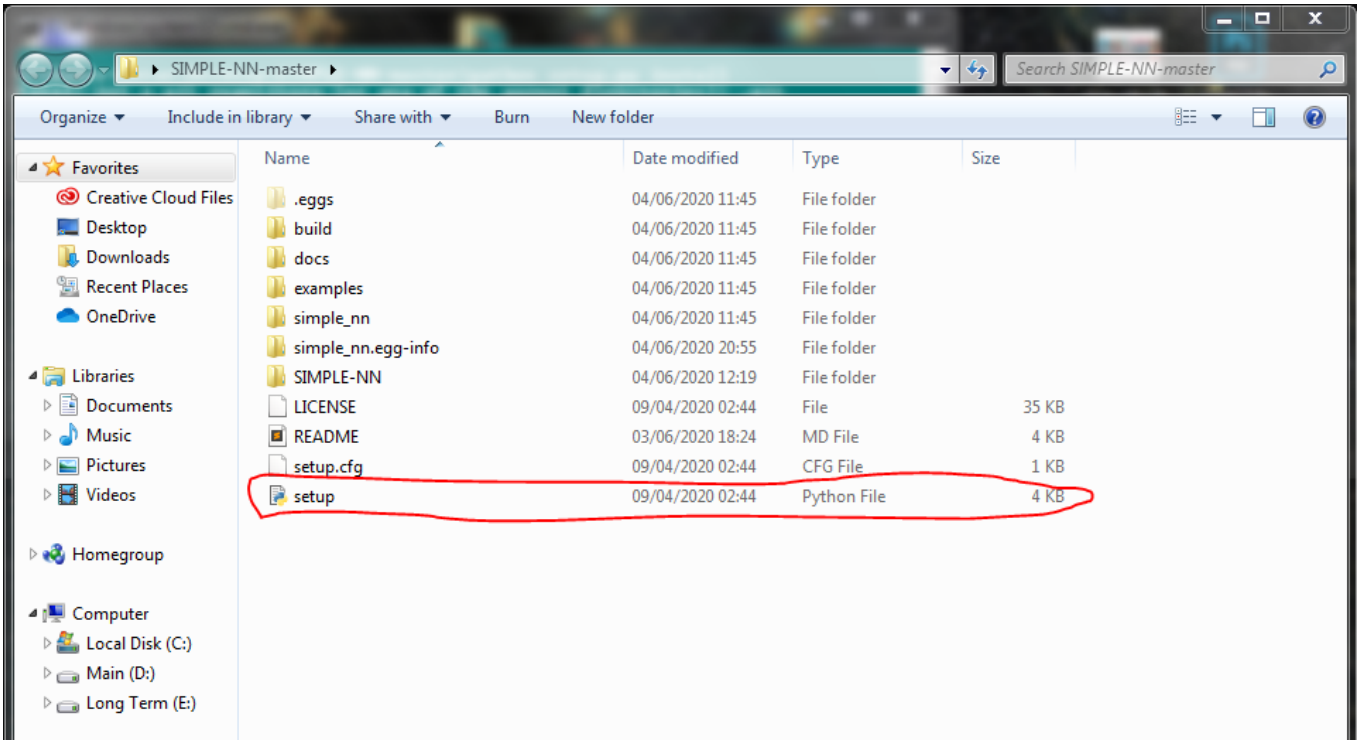
# Day 8 - 04/06/20

## 8.1 Simple-NN (Behler-Method Python Implementation) Setup

Today I decided to try and move on from looking at equations and instead take initiative to run the python implementation of Behler's method in order to understand how the method works in terms of code.

### 8.1.1 Running SIMPLE-NN on Windows

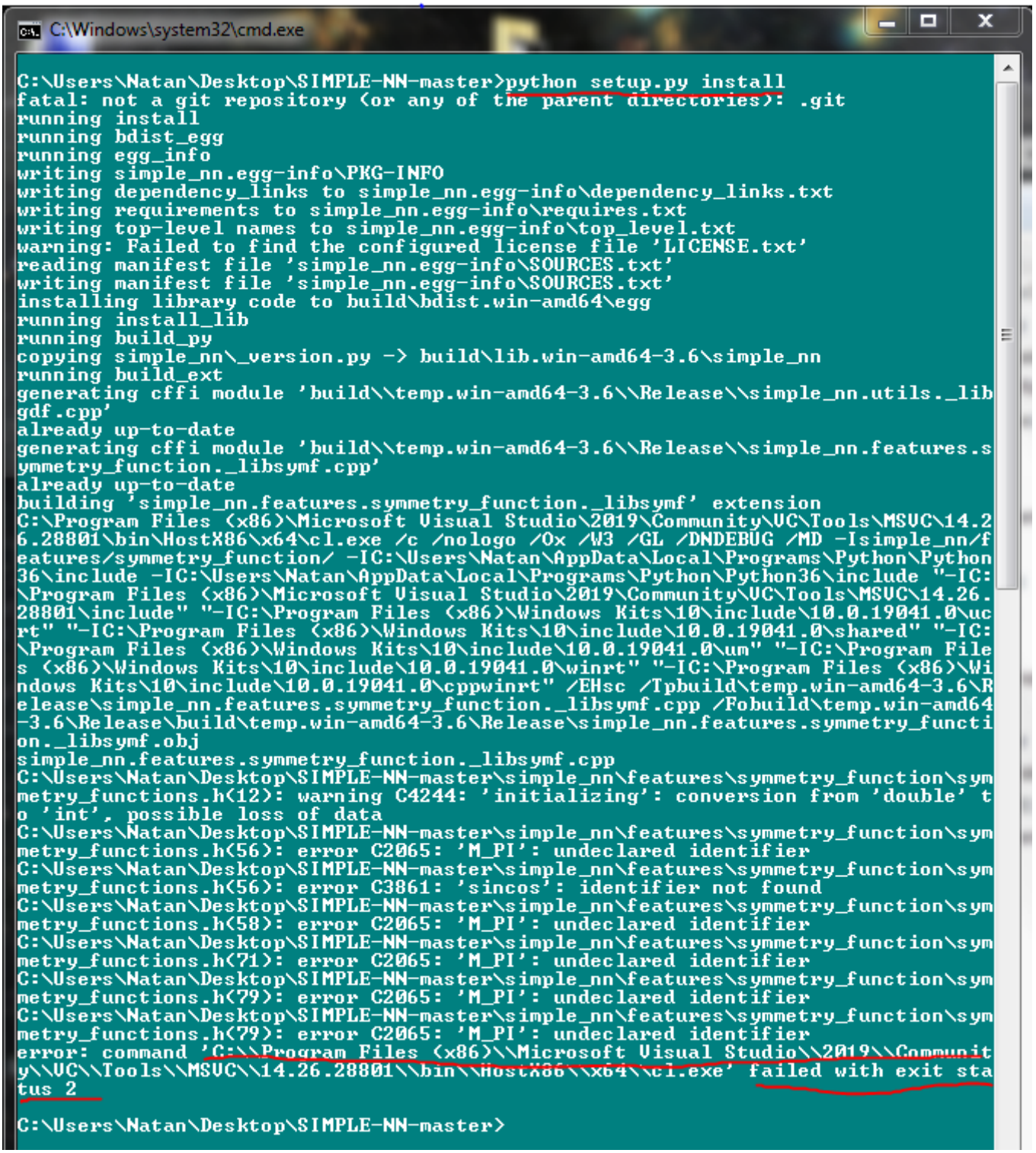
At first, we tried running the code using a Windows based machine. We began by opening the directory we downloaded it to.




As instructed by the README.md file. We ran the setup.py file from the command line by running:

```
python setup.py install
```

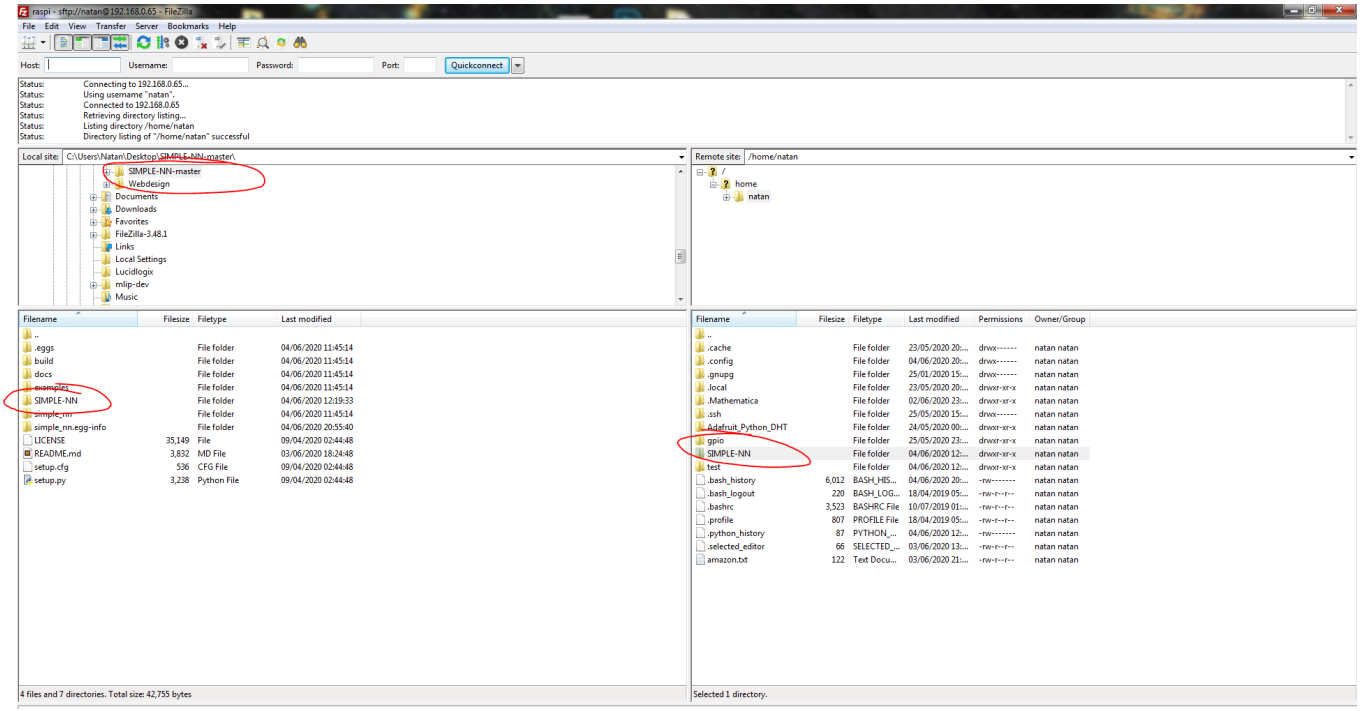
However we were greeted with a funky error message telling us we need Microsoft Visual Tools 2019 Build Tools. After solving this we still got this error



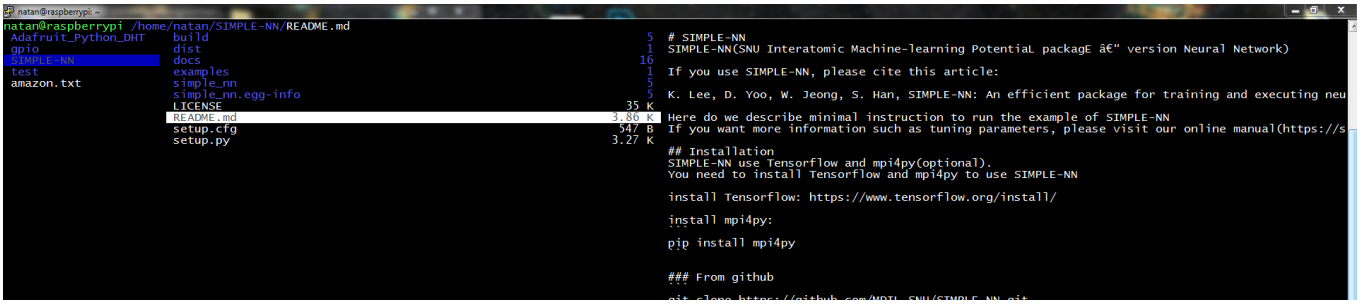

After further investigation of the program, I discovered some files that are typical of a Unix system. This prompted me to try run this program on a Unix-based operating system. (No-where in the documentation it mentioned this). The only linux machine available to me at the time was my personal Raspberry Pi which I SSH into for my personal projects. I realise this is extremely underpowered for the purpose however I just wanted to get a proof of concept working with zero errors, i was not worried about how long the simulation would take as I wasn't planning on it finishing.

### 8.1.2 Running SIMPLE-NN on Raspberry Pi 4

I transferred the Simple-NN file from my computer using Filezilla through using the SFTP protocol. I also installed the python libaries required: "mpi4py" and "Tensorflow".

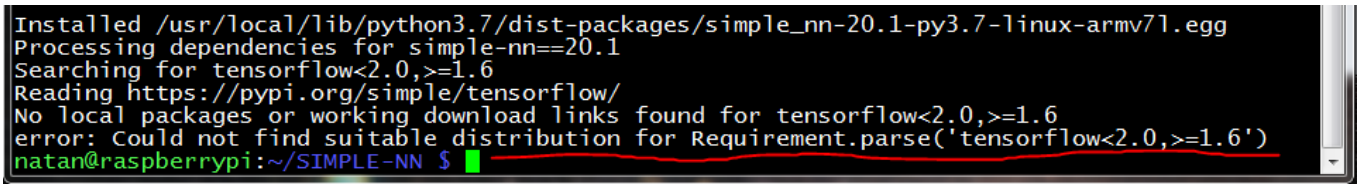


Then navigated to the directory on the rasperry pi using an SSH command line using Putty.exe. (I'm using package called "ranger" for easier navigation)



I then ran the exact same setup command

```
sudo python3 setup.py install
```



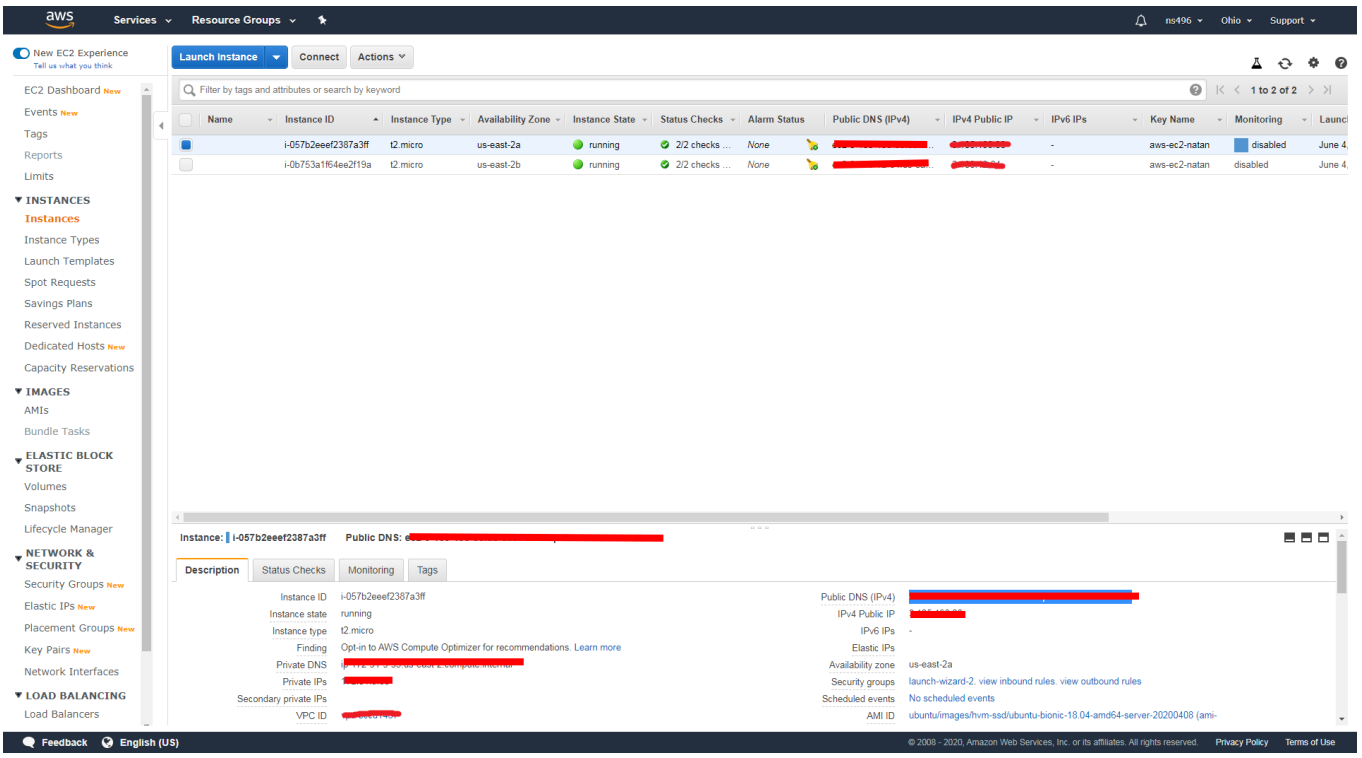

Unfortunately, this did not work. The Raspberry Pi uses a processor of the ARM architecture that is incompatible with the version of tensorflow required (tensorflow==1.6) by this program.

This in turn means that for this program to run properly we need a linux distribution running on an x86 processor architecture. As I do not have one available I was left with two options. A virtual box on my current machine or an AWS server with the correct OS.

As I wanted to share this code with my collegues I thought it might be a better option to host an AWS server with a Ubuntu x86 distribution so that we can all work on the same files at once. I chose the 12 month free trial of EC2 micro servers which I realise that are extremely underpowered for this type of calculation (as is the raspberry pi) however right now I was only trying to get a proof of concept to work.

## 8.1.3 Running SIMPLE-NN on AWS EC2

I started off by creating a free tier EC2 Ubuntu x86 instance on the Amazon Web Services platform. I generated the private and public keys and SSHd into the server. I gave access to my collegues aswell. The screenshot below shows the AWS dashboard for the instances running.

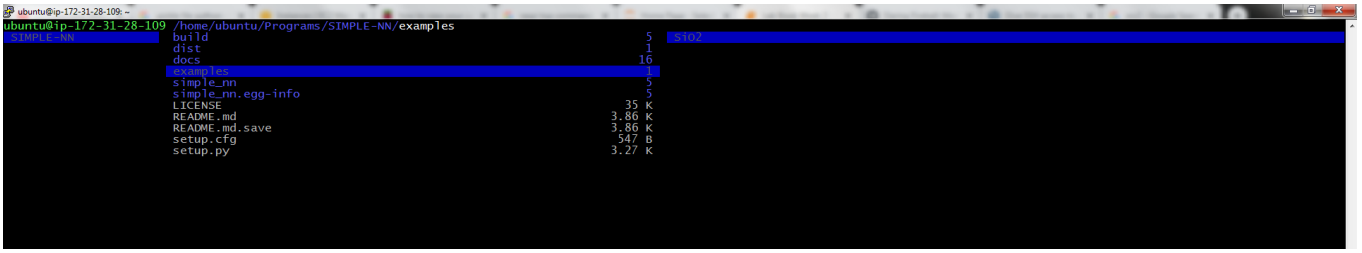



After some initial setup installing "pip" and downloading my preffered tools I also installed the dependancies for the program:

```
pip install tensorflow==1.6
pip install mpi4py
```

(for mpi4py installation to work i had to run: "sudo apt install libopenmpi-dev")

I transferred the SIMPLE-NN files using Filezilla through SFTP the same way as the raspberry pi. I then ran the setup.py again and finally the setup was sucessful with one adjustment (Downgrading "ase" library to 3.17 from 3.18). Now that the program was installed on this AWS instance, I navigated to the examples file which contains example code for Silicon Dioxide (60 atoms).




I then ran the run.py in the generate_NNP file. The code gave a few warnings about wrong filetypes used however python being python dealt with it well and succeeded in producing output files. The code crashed when the instance ran out of disk space (I used a 10GB disk not thinking ahead.)

I should have most likely used a virutal environment for running all of this code as this would have made versioning of the libraries easier. This will definitely come useful when trying to run fortran programs.

I then tried the exact same method using another instance I created now with 30GB storage, still not ideal but better. Turns out after running, the program wants a version of tensorflow that is higher than 1.6 because it needs a certain command not presented in 1.9.

## 8.2 Email to Prof. Russo about access to University Servers

Clearly we are going to need more computing power if we want to run this properly, this was just a test to get familiar with the software. I decided to write Prof. Russo an email about access to the University computers to which we can VPN or SSH into.

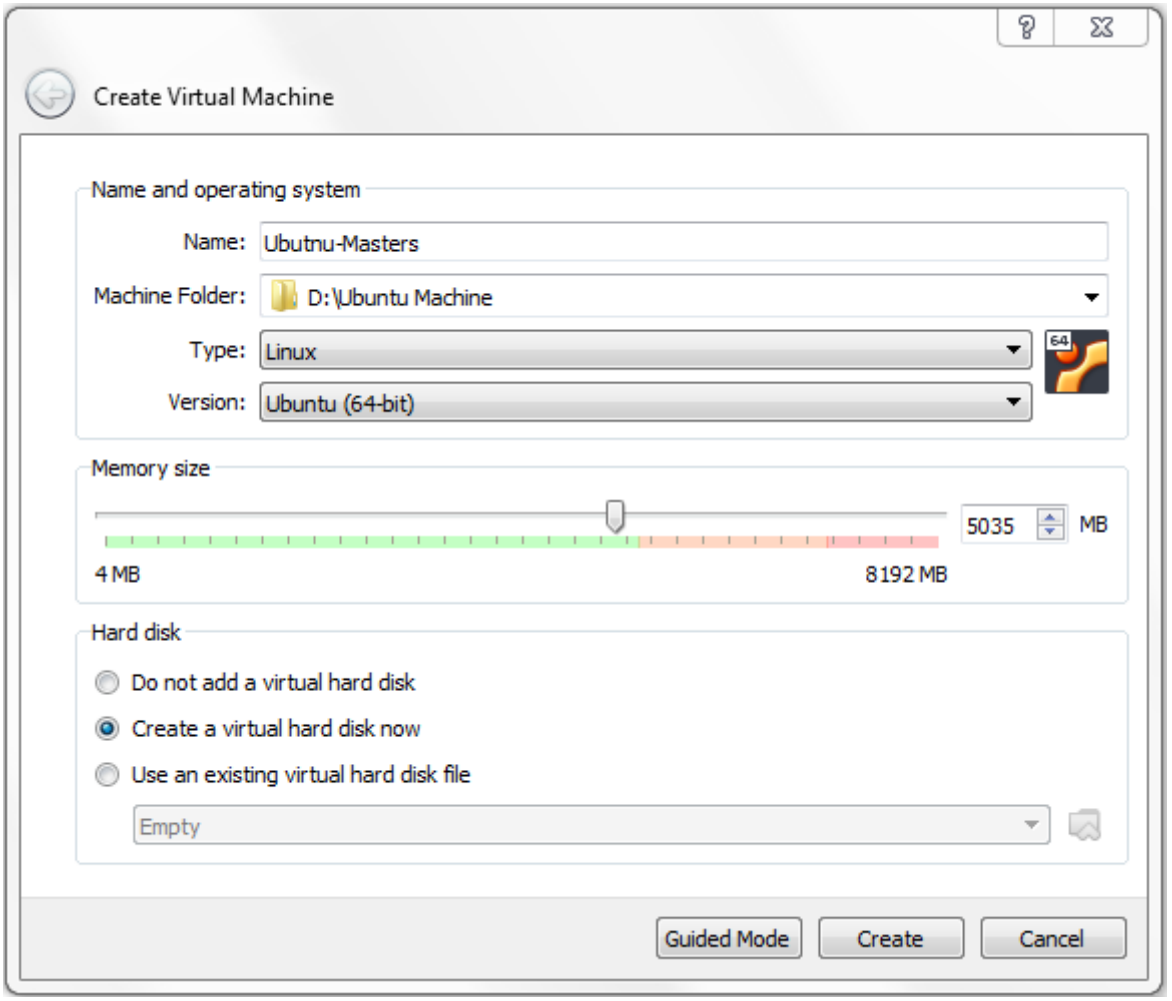
> Dear Prof. Russo,
>
> Hope you are doing well. Sorry you couldn't make the meeting today but I understand how busy you must be.
>
> We used today to study more the python implementation of the Behler method but unfortunately couldn't run it on windows. After further investigation I decided to try running it on a unix based system. Because the only unix based system at my disposal is a Raspberry Pi I tried running it on there. Unfortunately some of the requirements don't support the processor architecture of the Raspberry Pi.
>
> I then decided to create a mini-server Linux Ubuntu x86 instance using amazon web services to try and run the code. This way I also made it easy for the other group members to collaborate as I shared this server with them. After a few adjustments I was finally able to run the code without any errors. The only problem being is that there was not enough space on this Free Tier mini server I set up and we werent able to find a Potential Energy Surface for SiO2 as we literally ran out of space.
>
> What I wanted to ask you is whether you would or anyone else in the department know whether we could get access to a server at the uni that would be able to use for our experimentation. I know we mentioned Prof. Hepplestone but I wanted your opinion of where I could find access to these kinds of resources.
>
> Kind Regards,
> Natan

# Day 9 - 05/06/20

## 9.1 VirtualBox Virtual Machine SIMPLE-NN (Ubuntu 20.04)

Majority of today was spent on setting up the right environment for the software to run in a virtual machine.

I started off with running the same code on a virtual machine locally on my own computer. I started off by installing VirtualBox and installing it. (Firstly I installed it on my (D:) drive to save space on my ssd however that ended up with errors so i had to reinstall.)




I made a virtual machine with 5GB or RAM and 80GB of storage. (Still underpowered but should do the task given enough time in my opinion.) I then downloaded an Ubuntu 20.04 .iso disk image from the Ubuntu website and installed it on there. I then attempted to install the dependancies (tensorflow==1.9 and mpi4py) on the machine but quickly found that Ubuntu 20.04 comes preinstalled with Python 3.8 and pip 20.1.1. This is too new to install tensorflow==1.9 as it only works with tensorflow2.

I then decided to set up a virtual envrionment within the virtual machine using the "virtualenv" package that ran python3.6 where i could then download and install the proper requirements.

```
virtualenv --python=python3.6 myvenv
```

despite this working and all of the requirements being downloaded, SIMPLE-NN did not like being ran inside a virtual environment on a virtual machine as it threw errors about missing system files.

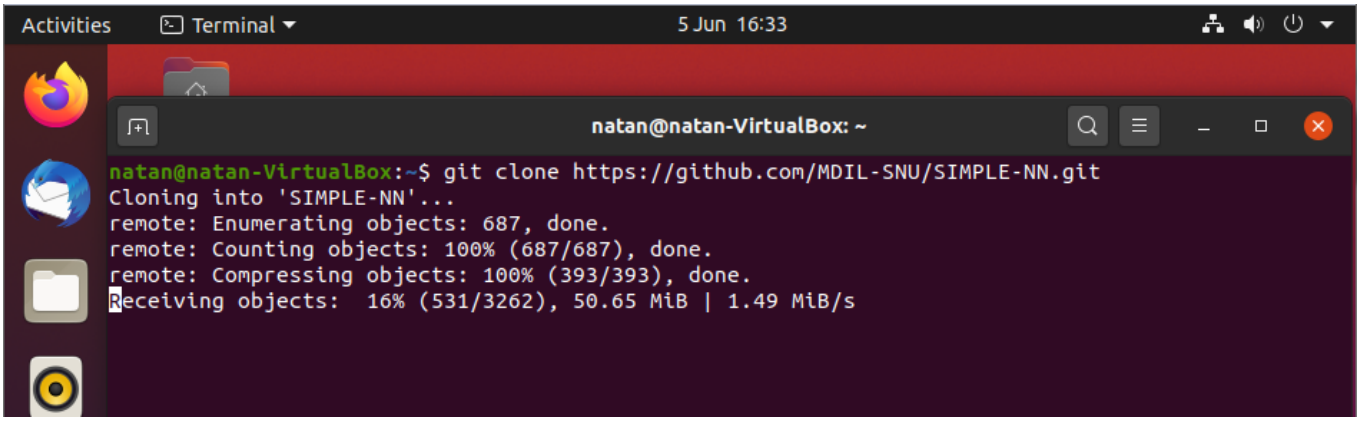
In retrospect I probably should have used Anaconda, a different distribution of python which makes it easier to switch between virtual environments but I personally don't like using it as it includes a lot of libraries that are unnecessary.

## 9.2 VirtualBox Virtual Machine SIMPLE-NN (Ubuntu 18.04)

After further investigation I noticed that the version of Ubuntu this program did work on on the AWS server was Ubuntu 18.04. The version of python pre-installed on there was Python 3.6 and pip 9.0.1. (Installing different versions of python on the same machine with completely isolated package managment tools is very cumbersome, this was easier). I then downloaded the disk image and created an Ubuntu instance. I made sure to do everything extremely precisely this time. This is the commands I ran to get the program running (excluding setup of the machine itself eg. "sudo apt-get update") for future reference:

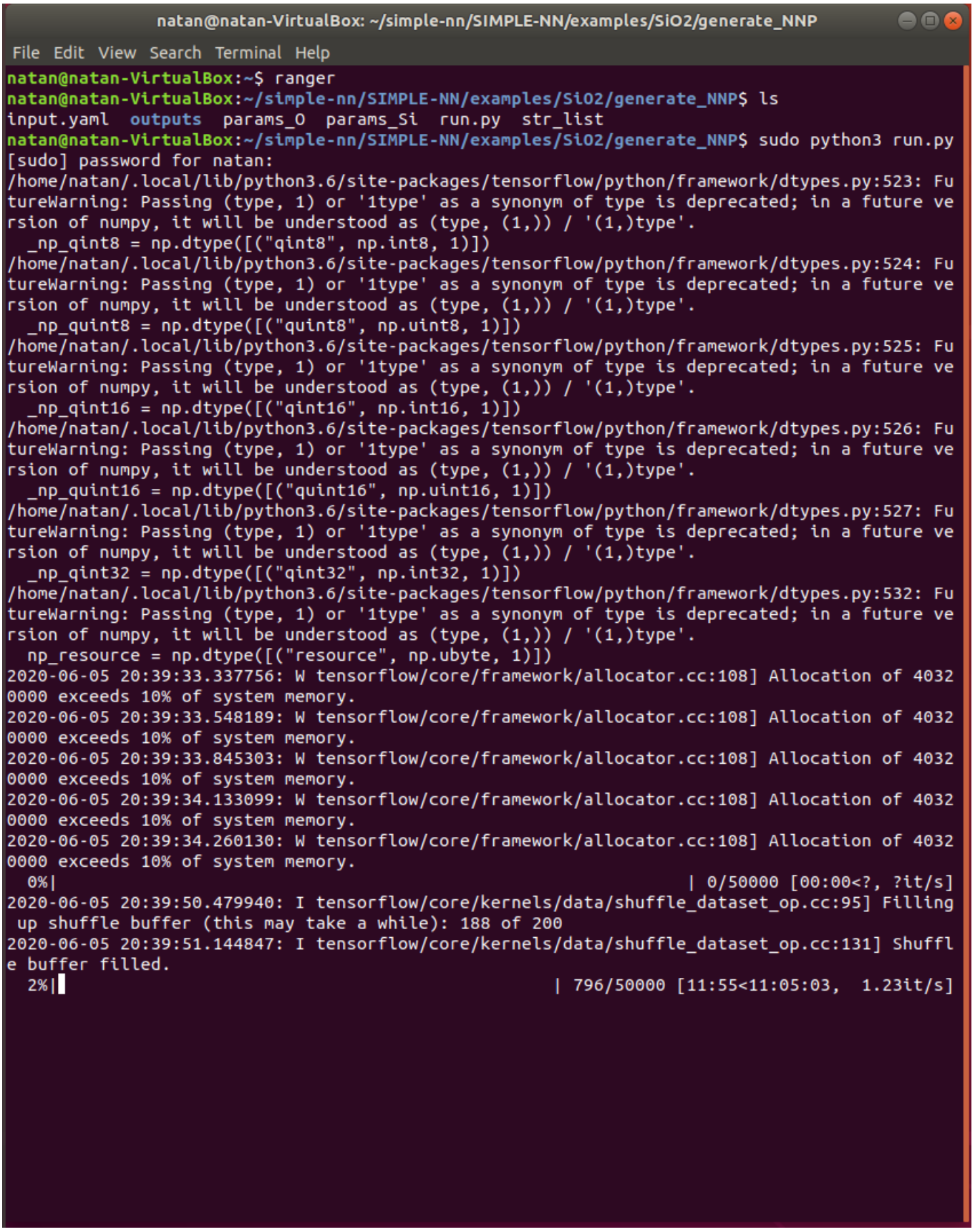
```
# Dependancies
sudo pip3 install tensorflow==1.9
sudo apt install libopenmpi-dev
sudo pip3 install mpi4py
sudo apt-get install git

# Downlading files and changing directory
git clone https://github.com/MDIL-SNU/SIMPLE-NN.git
cd SIMPLE-NN

# Running setup and changing ase version 3.19=>3.17
sudo python3 setup.py install
sudo pip3 uninstall ase
pip3 install ase==3.17

# Changing to example directory and running NNP code to get PES
cd examples/SiO2/generate_NNP
sudo python3 run.py
```

This is the directory that the code was installed in:




And this is the current screen after running run.py. There are a few warnings about the wrong datatypes however this is not an issue. There are no errors and the code is running. Judging from the progressbar this will take around 10 hours. I will let this run throughout the night.

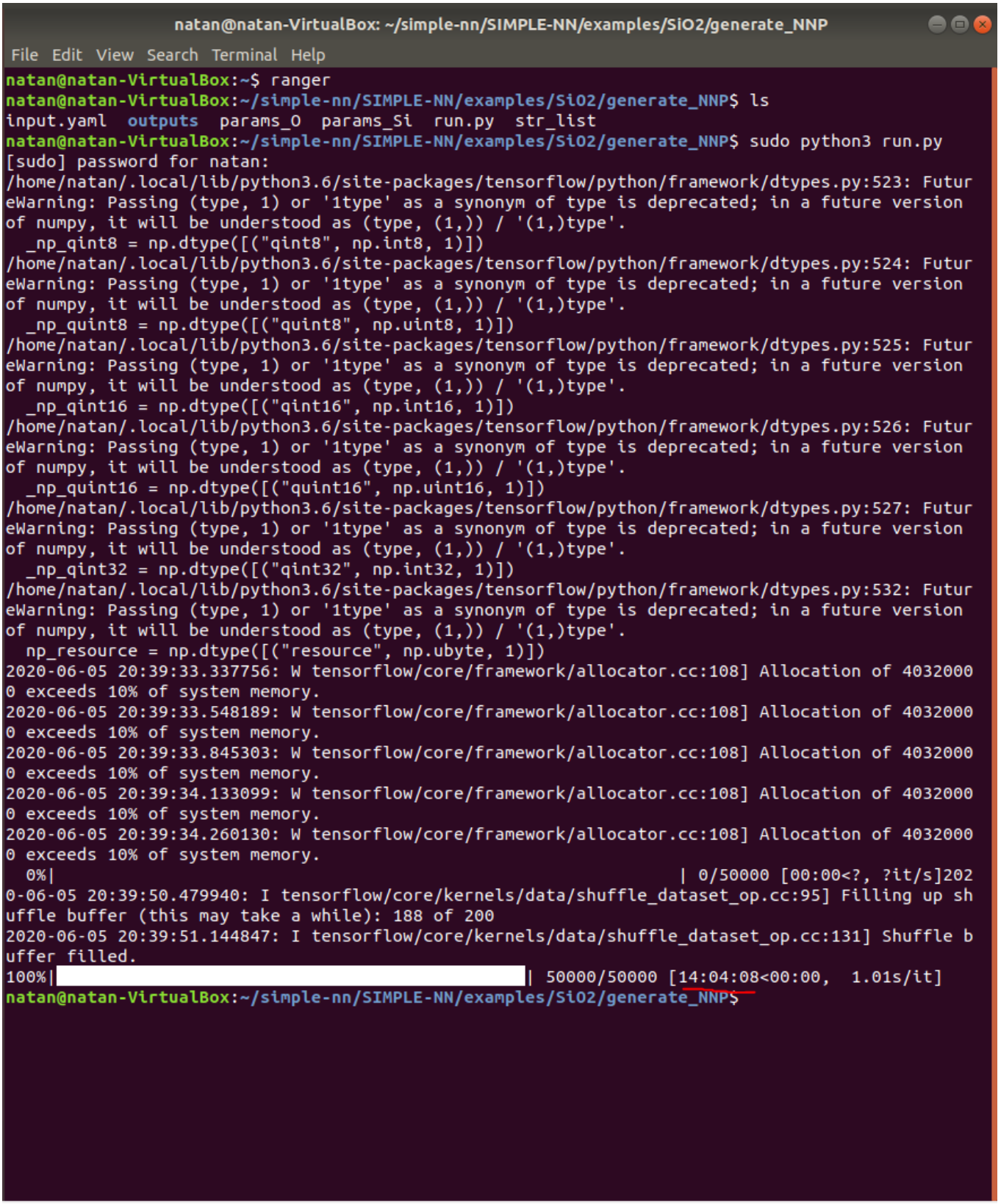
One silly thing I did was I forgot to increase the processor cores of the virtual machine before running the code as I didnt realise it would work and only realised this when the process was running for a while. Another thing may be increasing the video memory as I know Tensorflow is designed to utilise graphical memory.




After this is done, I will try to get the output data and feed it into LAMMPS for further analysis and hopefully a nice visualisation for the viva.

# Day 9.5 - 09/06/20 (Finished Running)

The SiO2 example PES generation code has finished running after 14 hours , 4 minutes and 8 seconds.




This code outputs a file "potential_saved_iterationXXXX" which contains the generated potential which then can be implemented in LAMMPS by copying over files from the SIMPLE-NN features folder into the LAMMPS installation folder:

```
cp /directory/of/simple-nn/features/symmetry_function/pair_nn.* /directory/of/la
mmps/src/
cp /directory/of/simple-nn/features/symmetry_function/symmetry_function.h /direc
tory/of/lammps/src/
```

Then in the LAMMPS input script making sure the right pair_style and pair_coeff are selected:

```
pair_style nn
pair_coeff * * /path/to/potential_saved Si O
```

I made another virutalbox to test and try out LAMMPS installation and run some examples on it while keeping SIMPLE-NN files seperate to avoid corruption.

Next week I will try to run an example simulation using LAMMPS, see the commonality between the input files and try to run the PES data generated by SIMPLE-NN in the simulator to then pass over to OVITO a visualisation software to confirm it is working.