MSIAM M2 Thesis Defense
on
27/August/2021

Ghimire Suraj
M2 MSIAM 2020-21
Grenoble INP Ensimag

Variable Selection of the Parrinello-Behler Descriptors for Neural-Network Potentials in Material Science

Research project performed at Grenoble INP SiMaP

Suraj Ghimire

Grenoble INP Ensimag (Master MSIAM)

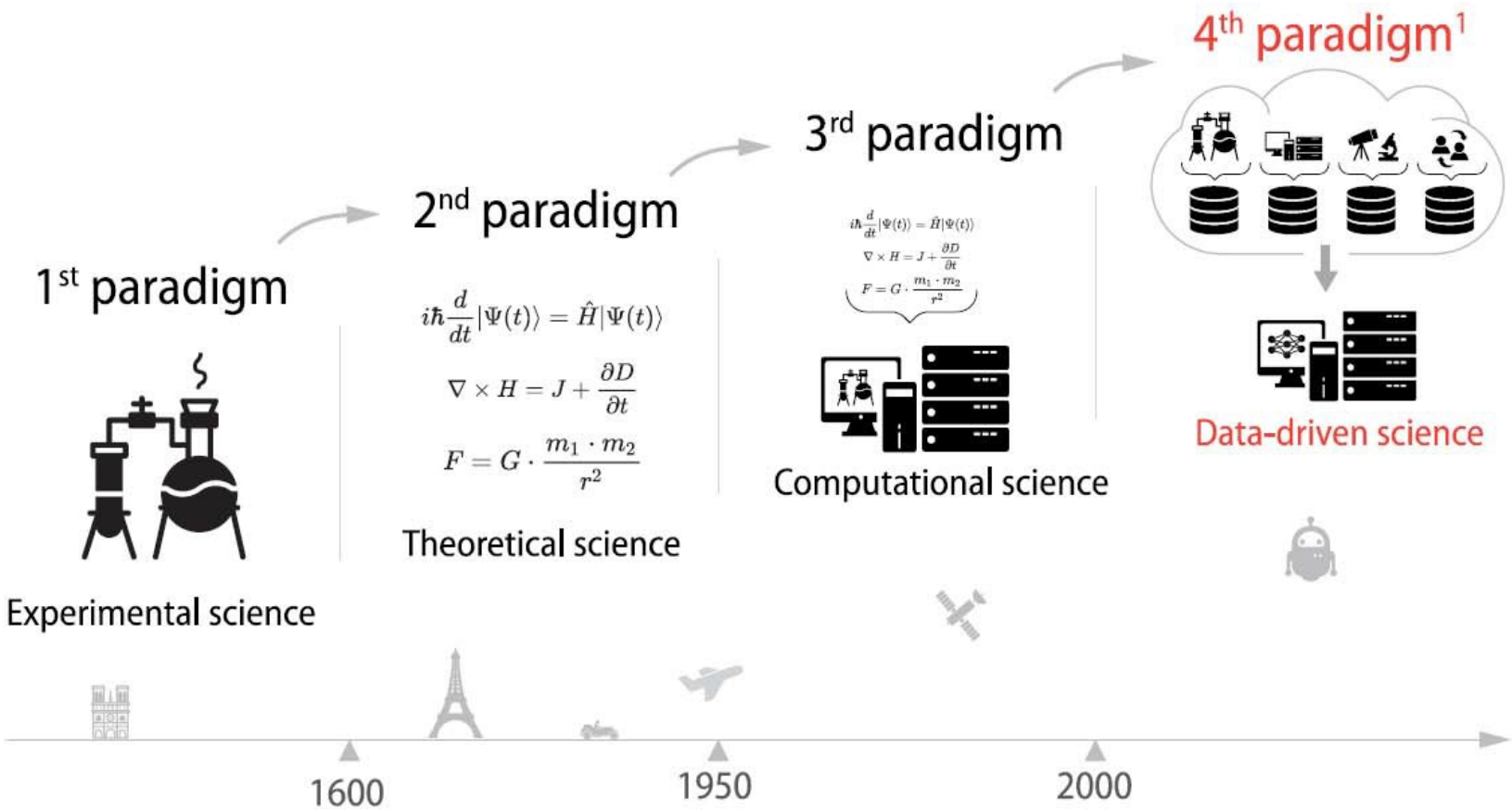
August 27th, 2021

Under the supervision of

Noel JAKSE
SIMAP, Grenoble

Emilie DEVIJVER
LIG, Grenoble

Evolving of Molecular Dynamics



[1] Jim Gray, 2007 [GRAY]

Fig 1: Paradigm of Molecular Science

Evolving of Molecular Dynamics

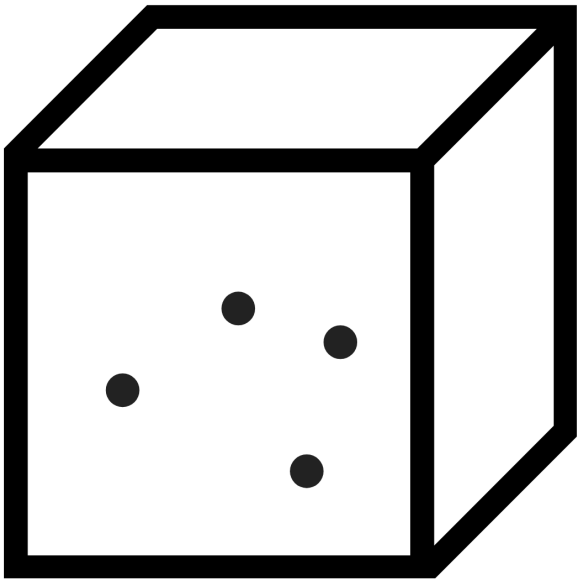


Fig 2: Representation of Atom within the box

Molecular dynamics (MD) is a computer simulation technique where the time evolution of a set of interacting atoms is followed by integrating their equations of motion.
In molecular dynamics we follow the laws of classical mechanics, and most notably Newton's law:

$$\mathbf{F}_i = m_i \mathbf{a}_i$$

for each atom i in a system constituted by N atoms.

Single layer perceptron

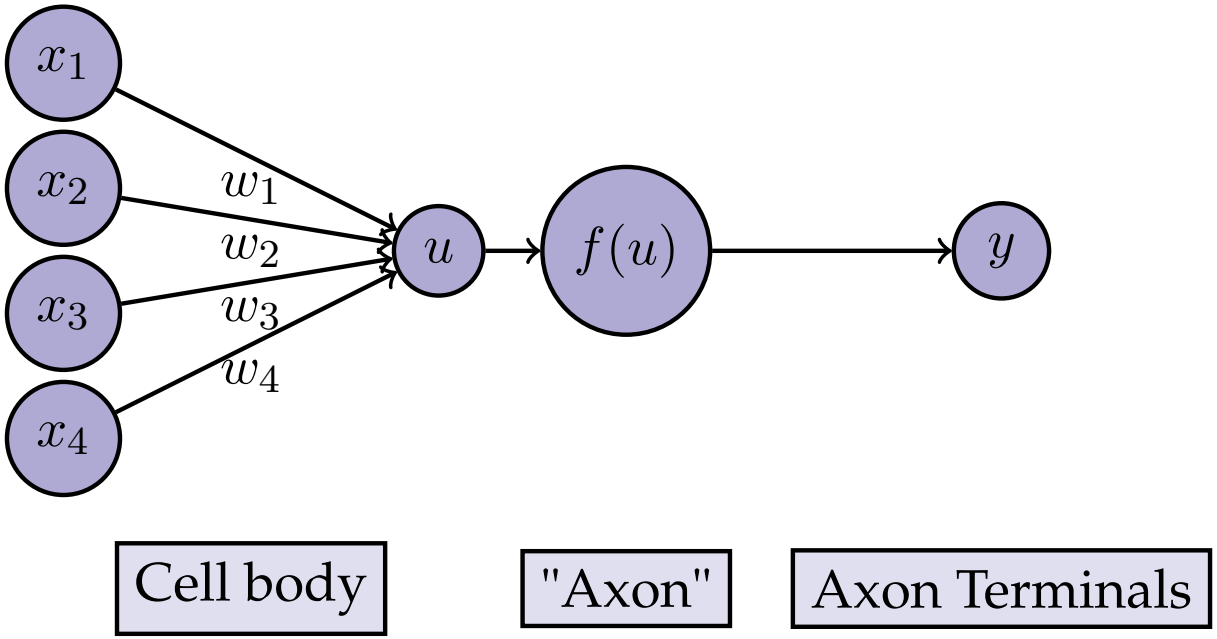


Fig: 3: Single-layer perceptron network

Single layer perceptron

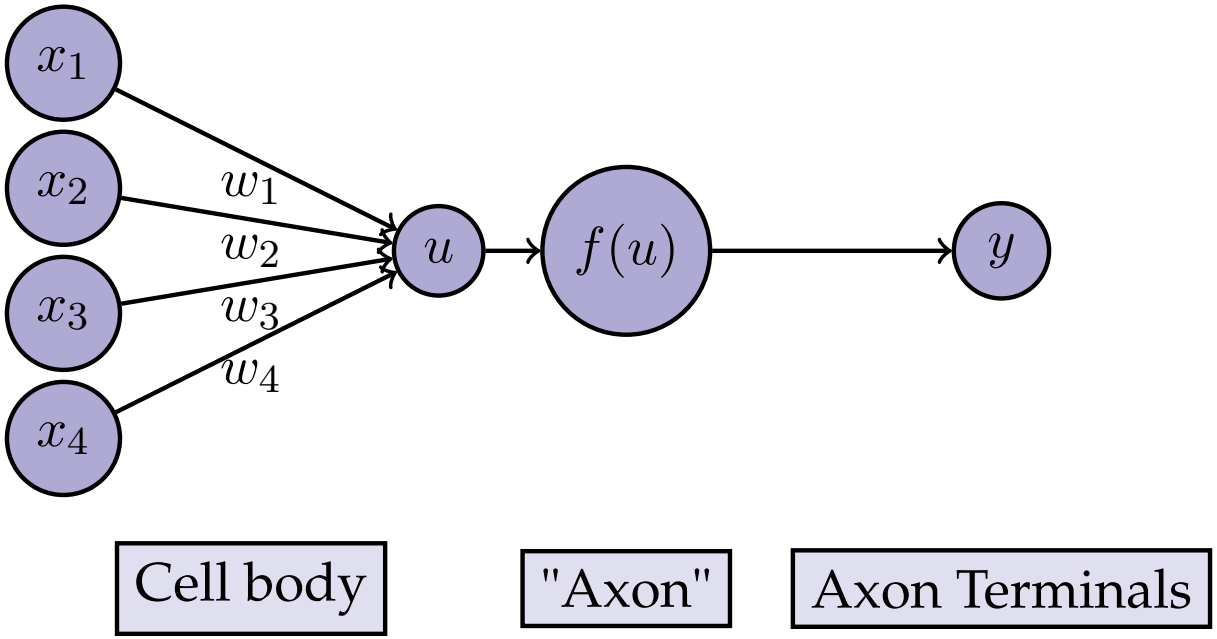


Fig: 3: Single-layer perceptron network

$$y = f(u) = f\left(\sum_{i=1}^{N_{\text{inputs}}} w_i x_i + b_i\right) \tag{1}$$

Sub-networks and networks

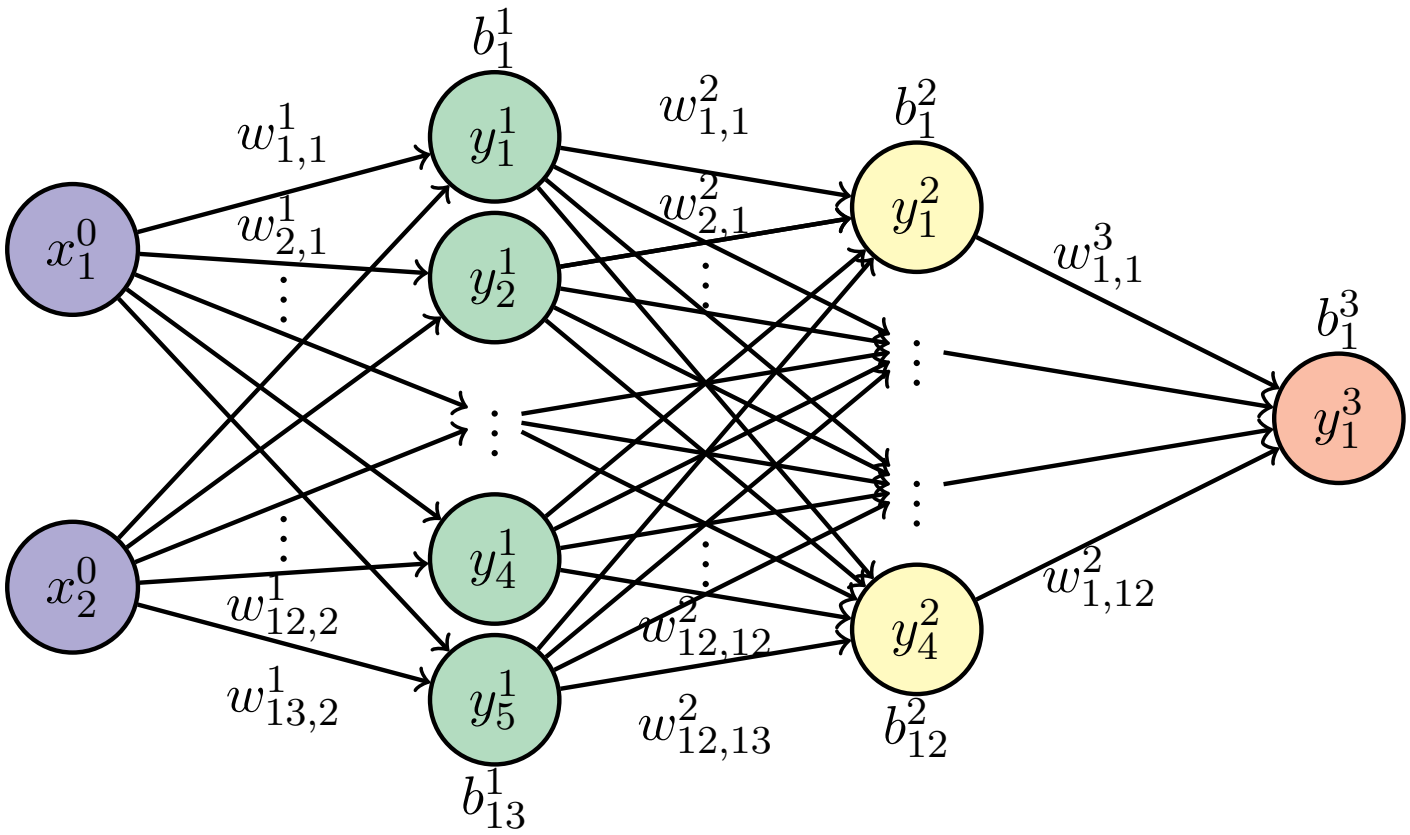


Fig 4: Example of Multi-layer perceptron network

Mathematical representation of the network

Mathematical Equation for fig 4 is

$$y_i^3 = f^3\left(\sum_{j=1}^{4} w_{ij}^3 f^2\left(\sum_{k=1}^{5} w_{jk}^2 f^1\left(\sum_{l=1}^{2} w_{kl}^1 x_l^0 + b_k^1\right) + b_j^2\right) + b_i^3\right) \tag{2}$$

The generalized equation of ANNs can be written as

$$y_i^l = f^l\left(u_i^l\right) = f^l\left(\sum_{j=1}^{N_{l-1}} w_{ij}^l y_j^{l-1} + b_i^l\right) \tag{3}$$

This Neural Network structure is written as $2-5-4-1$

Behler Parrinnelo Features

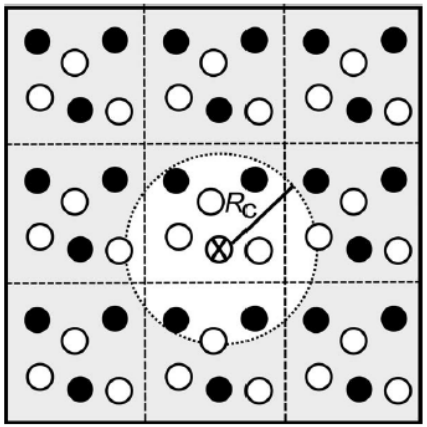


Fig 5: cut off function

Cutoff Function

$$f_{\mathrm{c}}\left(R_{ij}\right)=\begin{cases}0.5\cdot\left[\cos\left(\frac{\pi R_{ij}}{R_{\mathrm{c}}}\right)+1\right] & \text{for } R_{ij}\leqslant R_{\mathrm{c}}\\ 0 & \text{for } R_{ij}>R_{\mathrm{c}}\end{cases} \tag{4}$$

where R_{ij} refers to the distance between an atom i and an atom j and R_c is the cutoff radius.

Behler Parrinnelo Features [1] [2]

Radial Function

$$G_i^2 = \sum_j e^{-\eta(R_{ij}-R_s)^2} \cdot f_c\left(R_{ij}\right) \tag{5}$$

Angular Function

$$G_i^5 = 2^{1-\zeta} \sum_{j,k\neq i}^{\text{all}} \left(1+\lambda\cos\theta_{ijk}\right)^\zeta \cdot e^{-\eta\left(R_{ij}^2+R_{ik}^2\right)} \cdot f_c\left(R_{ij}\right) \cdot f_c\left(R_{ik}\right) \tag{6}$$

where η is a Gaussian width, $\theta_{ijk} = \text{acos}\left(\mathbf{R}_{\text{ij}} \cdot \mathbf{R}_{\text{ik}}/R_{ij} \cdot R_{ik}\right)$

Sub-networks and networks

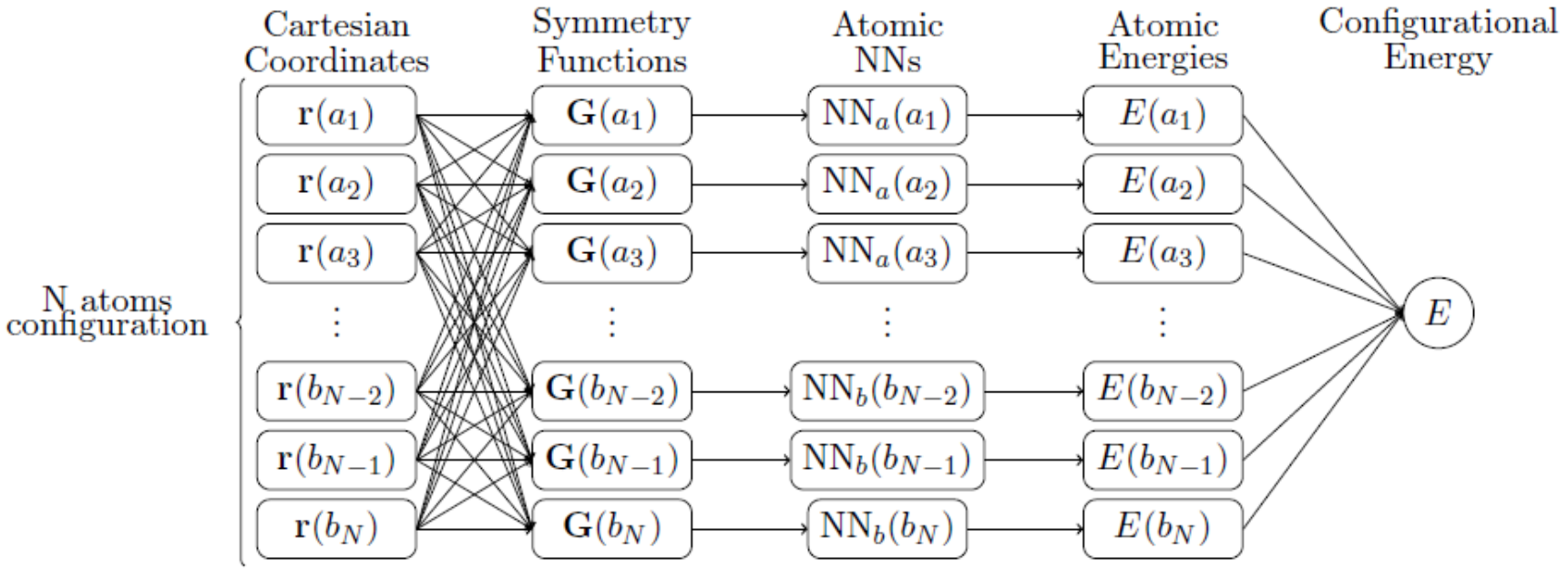


Fig 6: Symmetry functions on NN [[3]

An example of HDNN of 22-10-10-1

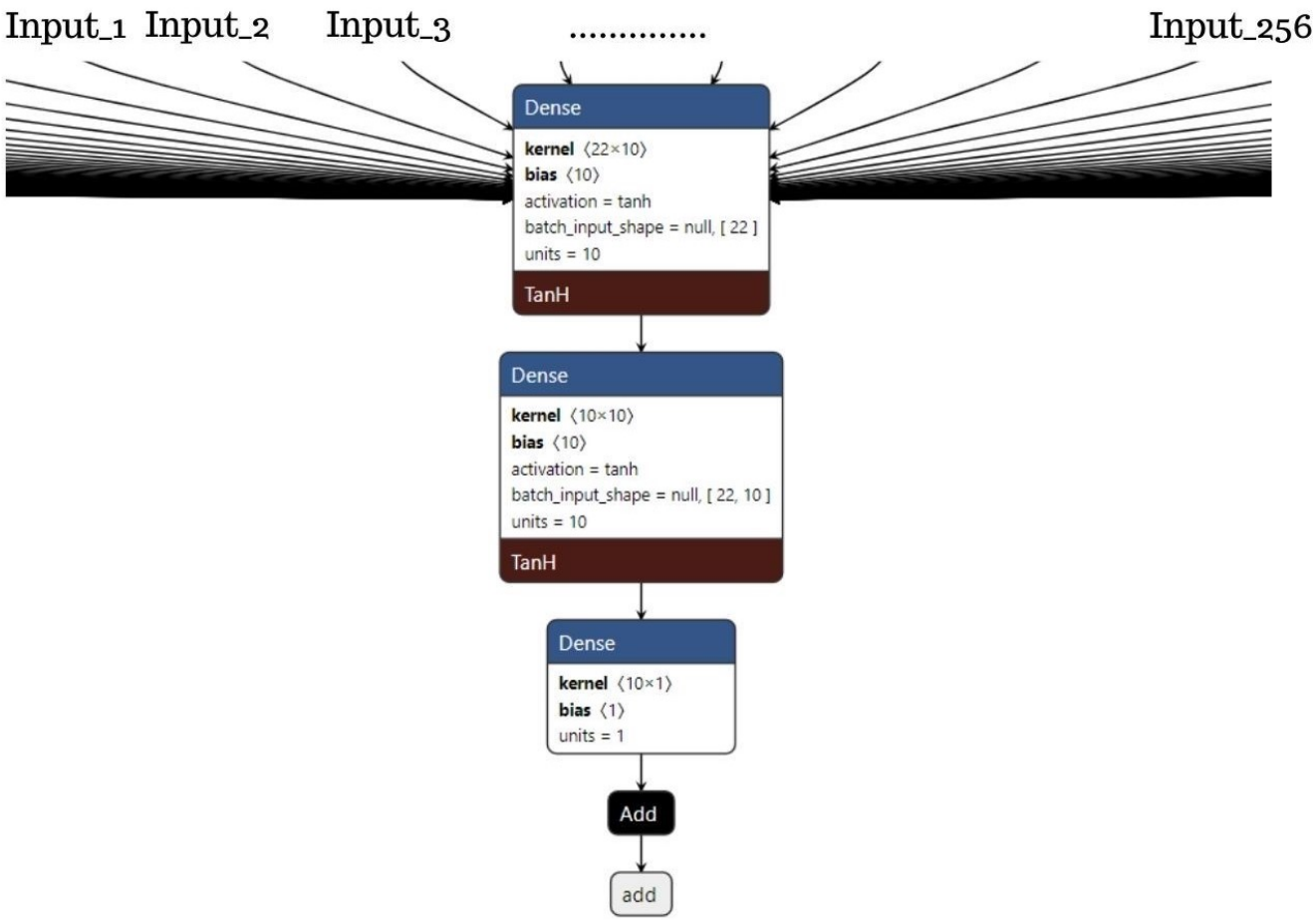


Fig 7 An example of HDNN of 22-10-10-1

Our Objective: To select sets of variables in Neural Network

Our Objective: To select sets of variables in Neural Network

- That gives prediction of potential with highest precision

Our Objective: To select sets of variables in Neural Network

- That gives prediction of potential with highest precision

How do we do measure precision?

By minimizing the validation loss

Our Objective: To select sets of variables in Neural Network

- That gives prediction of potential with highest precision

How do we do measure precision?

By minimizing the validation loss

- That has low computation cost

Our Objective: To select sets of variables in Neural Network

- That gives prediction of potential with highest precision

How do we do measure precision?

By minimizing the validation loss

- That has low computation cost

How do we do it?

By selecting smaller sets of observations as possible

Our Objective: To select sets of variables in Neural Network

- That gives prediction of potential with highest precision

How do we do measure precision?

By minimizing the validation loss

- That has low computation cost

How do we do it?

By selecting smaller sets of observations as possible

By selecting smaller structure as possible

Our Objective: To select sets of variables in Neural Network

- That gives prediction of potential with highest precision

How do we do measure precision?

By minimizing the validation loss

- That has low computation cost

How do we do it?

By selecting smaller sets of observations as possible

By selecting smaller structure as possible

Possible way for variable/feature selections

Possible way for variable/feature selections

On the basis of weights

Comparing weights of a variables within the model

Possible way for variable/feature selections

On the basis of weights

Comparing weights of a variables within the model

Random selection of variables

Randomization of available features

Possible way for variable/feature selections

On the basis of weights

Comparing weights of a variables within the model

Random selection of variables

Randomization of available features

Least cost

through least cost of the model

Possible way for variable/feature selections

On the basis of weights

Comparing weights of a variables within the model

Random selection of variables

Randomization of available features

Least cost

through least cost of the model

Measuring Criteria

Comparing Least loss of various across models

Cost function

$$\Gamma \equiv \Gamma(\mathbf{w}, X, y) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_i^*)^2 \tag{7}$$

Adam Optimisation Algorithm

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{8}$$

where

$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t,$ $\qquad v_t = \beta_2 v_{t_1} + (1 - \beta_2)\, g_t^2$

$\hat{m}_t = \frac{m_t}{1-\beta_1}$ $\qquad \hat{v}_t = \frac{v_t}{1-\beta_2}$

Adam on Keras

```
optimizer_adam = optimizers.Adam
(learning_rate=use.learning_rate, beta_1=0.9,
beta_2=0.999, epsilon=1e-08, amsgrad=False)
```

[4]

A little about Dataset

Temperature	State	#NO. of obs
10 K	Solid	1000 obs
300 K	Solid	342 obs
500 K	Solid	344 obs
800 K		1000 obs
1000 K	Liquid	1000 obs
1250 K	Liquid	1000 obs
1350 K	Liquid	1000 obs
1500 K	Liquid	1000 obs
1600 K	Liquid	1000 obs
1700 K	Liquid	1000 obs
1500 KH	Liquid	346 obs
Total		9032 obs

The initial Dataset [3]

256 atoms
Total No. of rows
$\longrightarrow 2,312,192 rows$

A little about Dataset

Variables of Dataset

- Input variables:-G2 and G5 variables
- Forces in each direction
- Energy Predictions

A little about Dataset

Method of Simulation

- Trained a dataset of 22 variables by changing structures eg. Number of layers, depth, validation split, lrr, Number of observations
- Selected the best possible combination
- Extended the data to 329 variables
- Replicated the training with the best combination
- Selected the smallest set of variables that gives prediction with high precision.

Train and validation loss per atom (G2)

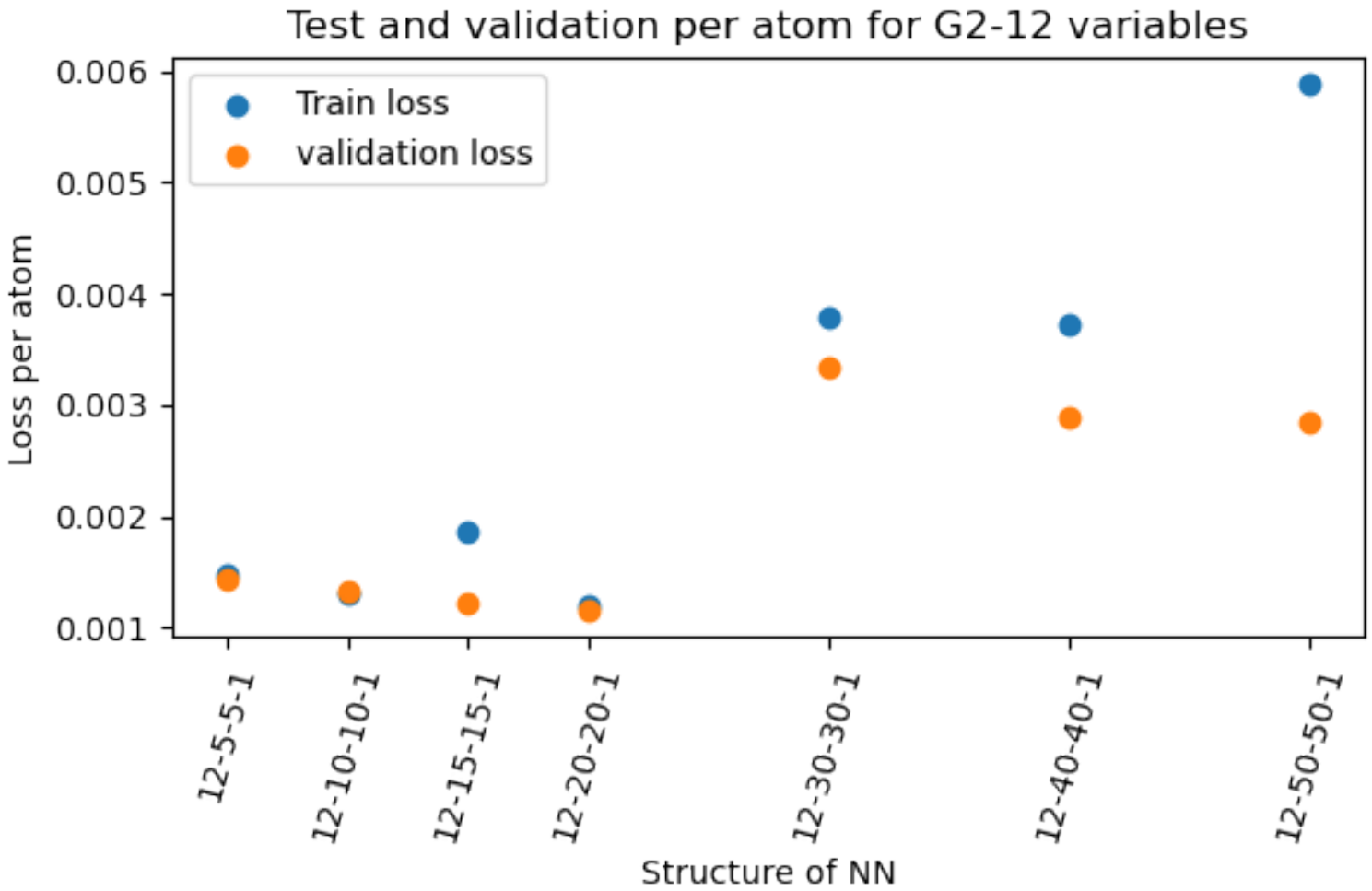


Fig 8: Loss per atom with 12 radial (G2) features

Physical representations of 12 radial (G2) features

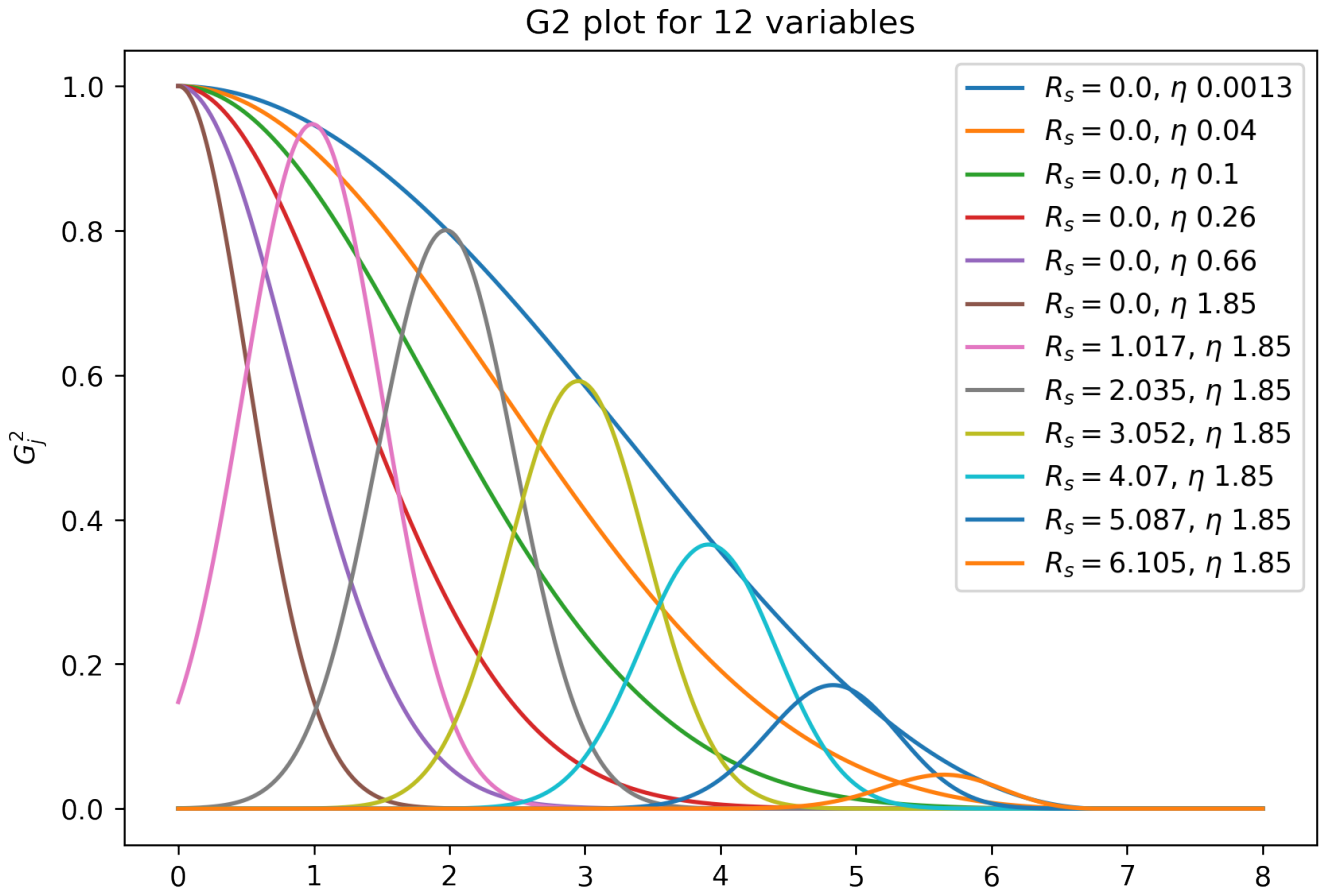


Fig 9: Physical representations of 12 radial (G2) features

Train and validation loss per atom (G2+G5)

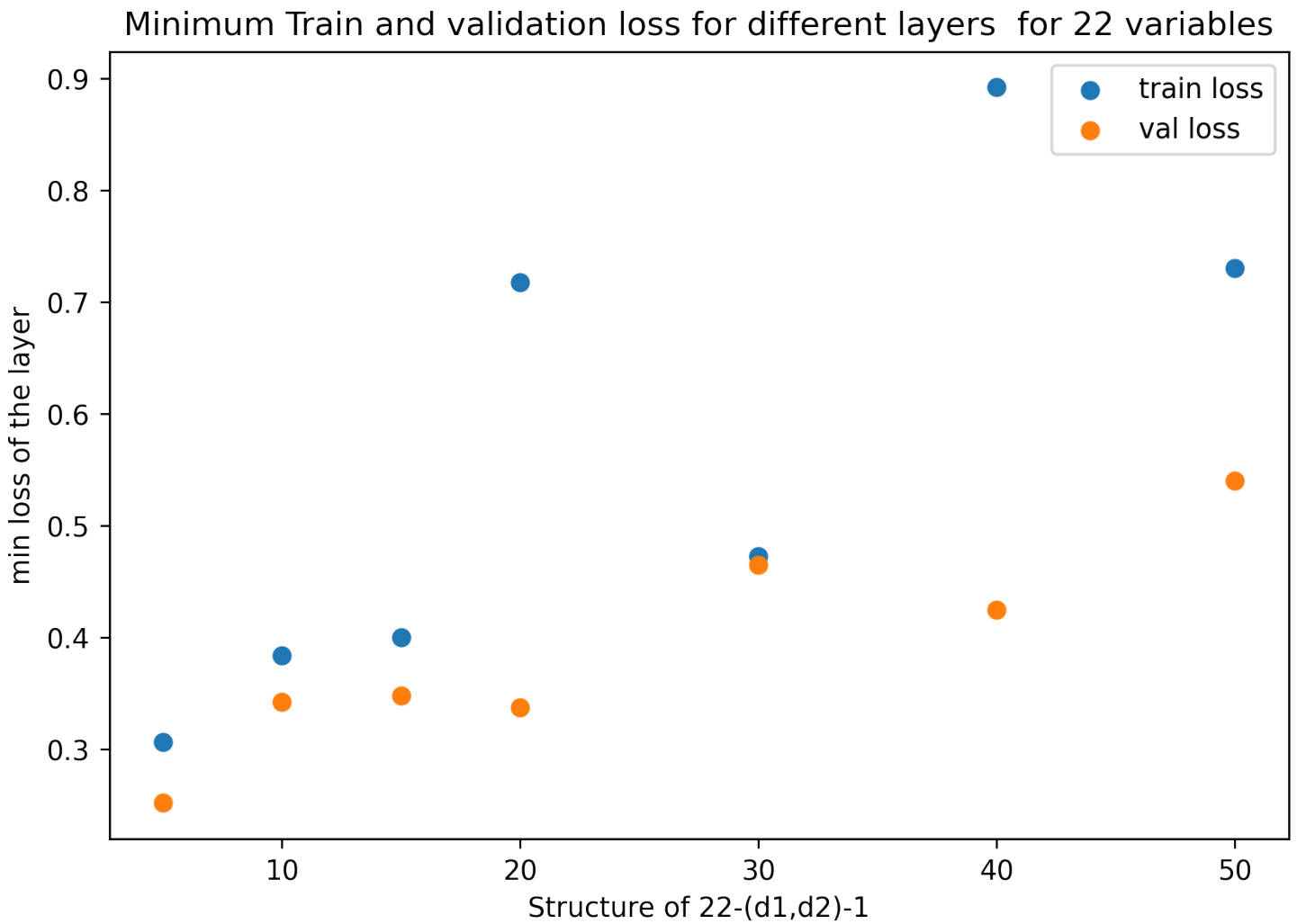


Fig 10: Training loss and validation loss for different layers

Selecting other hyperparameters

- **Train Validation split** :- 0.8
- **Number of layers** :- 2
- **Depth of inner layer** : - $(10, 10)$
- **Activation Function**:- $tanh$

Train-validation curve of loss (MSE) of of 22-10-10-1

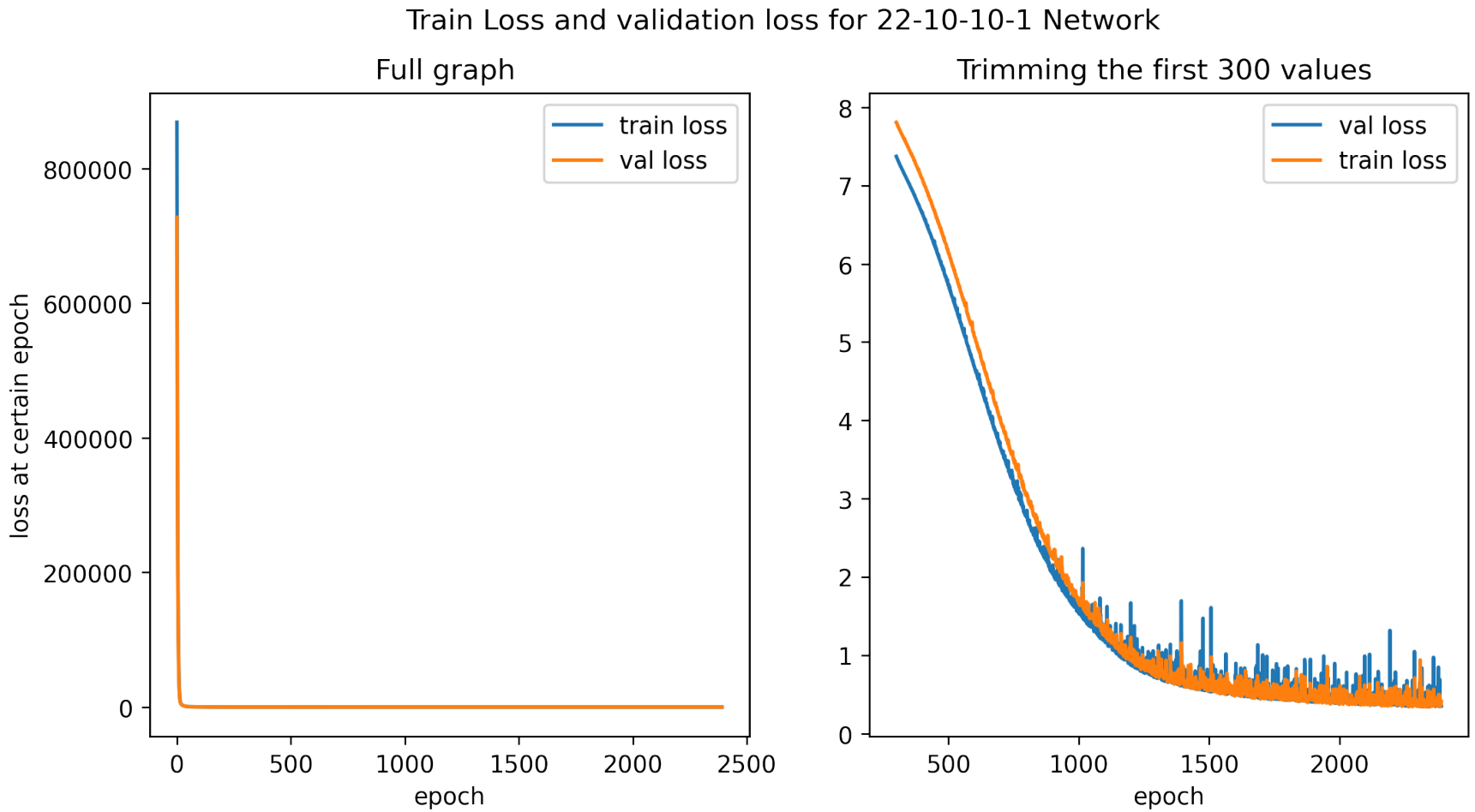


Fig 11: Training loss and validation loss for structure 22-10-10-1. Least validation loss 0.3481

Test-Predict Curve of the Model

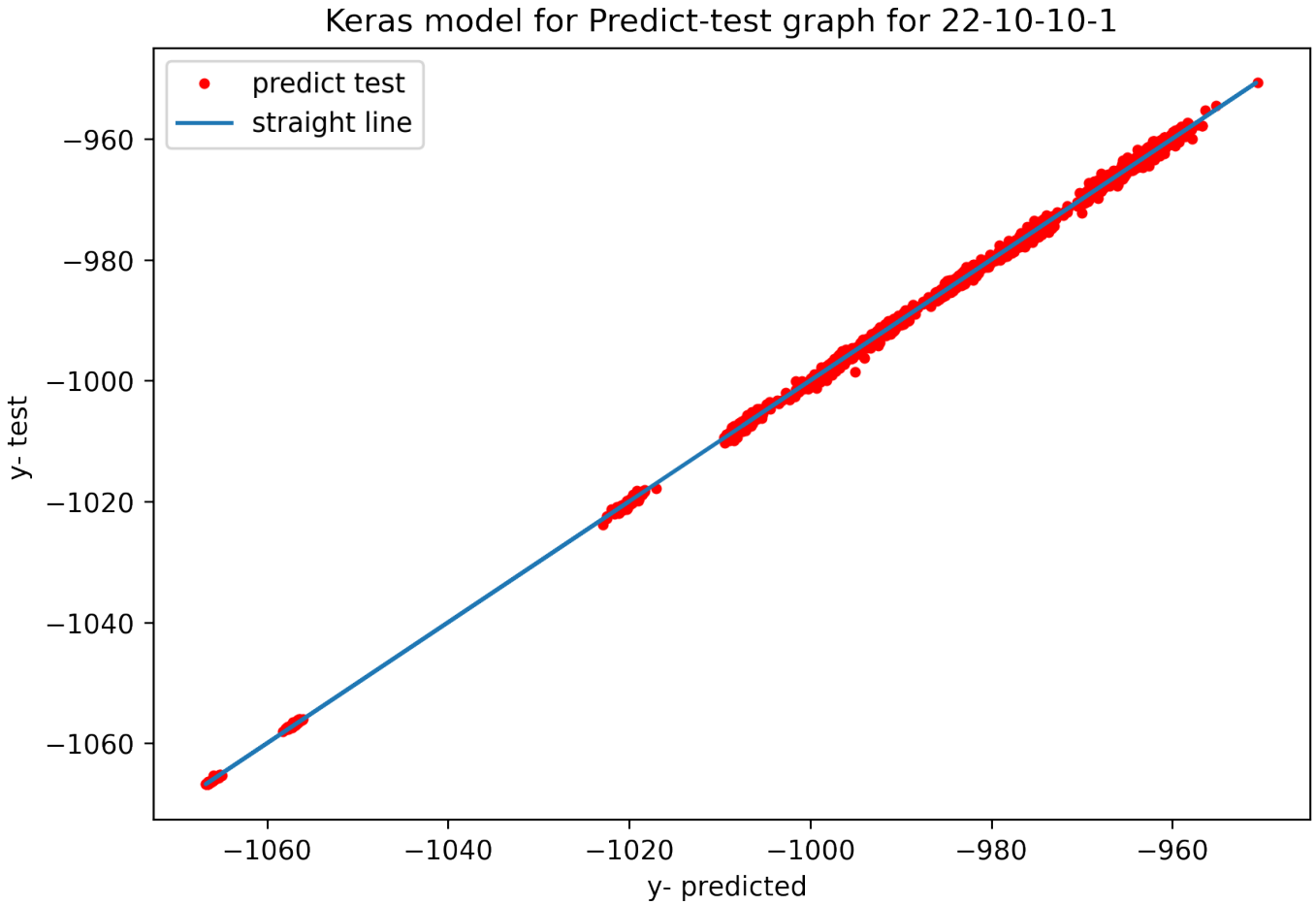


Fig 12: Test-Predict Curve of the Model for structure 22-10-10-1

Predicting Potentials with Random Forest

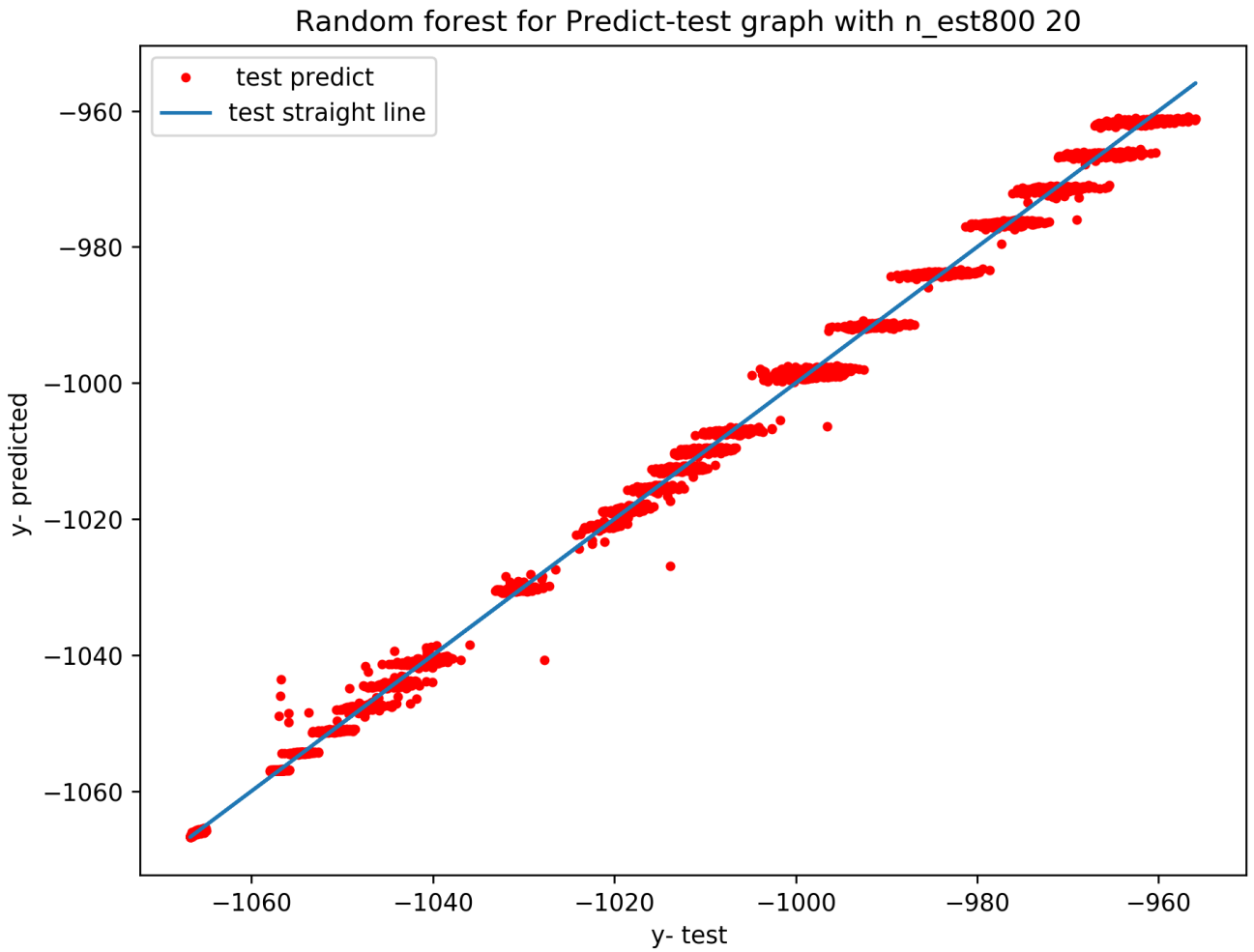
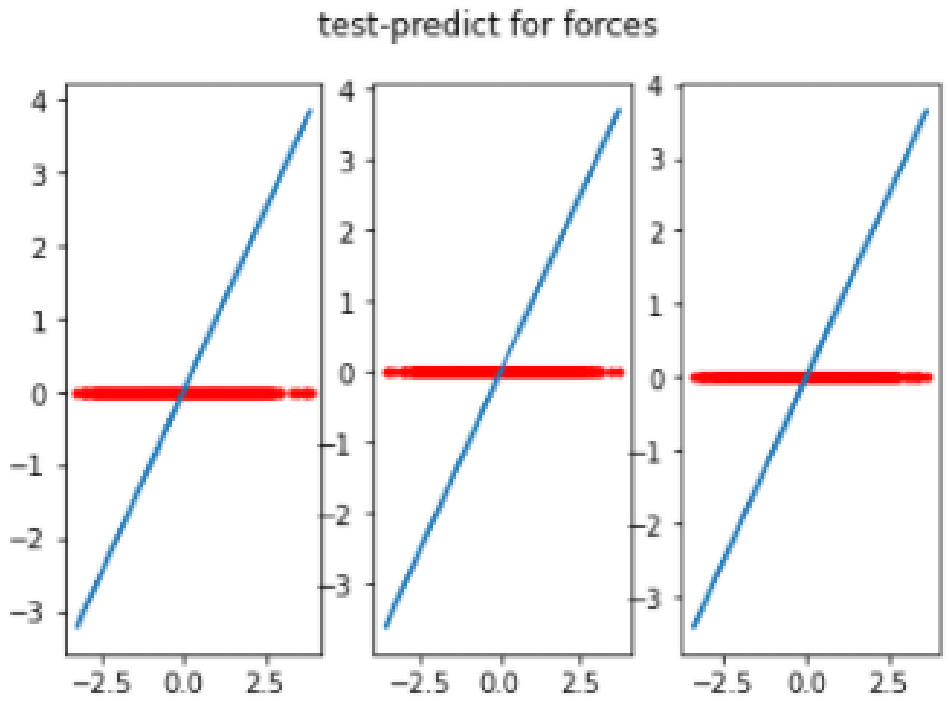


Fig 13: Predicting potentials with random forest

Predicting Forces with Random Forest



	max_depth	mse_x	mse_y	mse_z
0	20	0.344343	0.34433	0.344338

Fig 14: Predicting forces with random forest

Extension of G2 variables

```
#format r_c,eta,r_s
x = np.linspace(0.0,2.0,201)
x1 = np.ones(68)*1.85
e1 = np.linspace(0.05,6.8,68)
e =  np.zeros(201)
r = np.ones(269)*6.8
x = np.append(x,x1)
e = np.append(e,e1)
x[0] = 0.0013
features_G2_=[]
for i in range (269):
    features_G2_.append([r[i],x[i],e[i]])
```

Physical representation of G2 269 variables

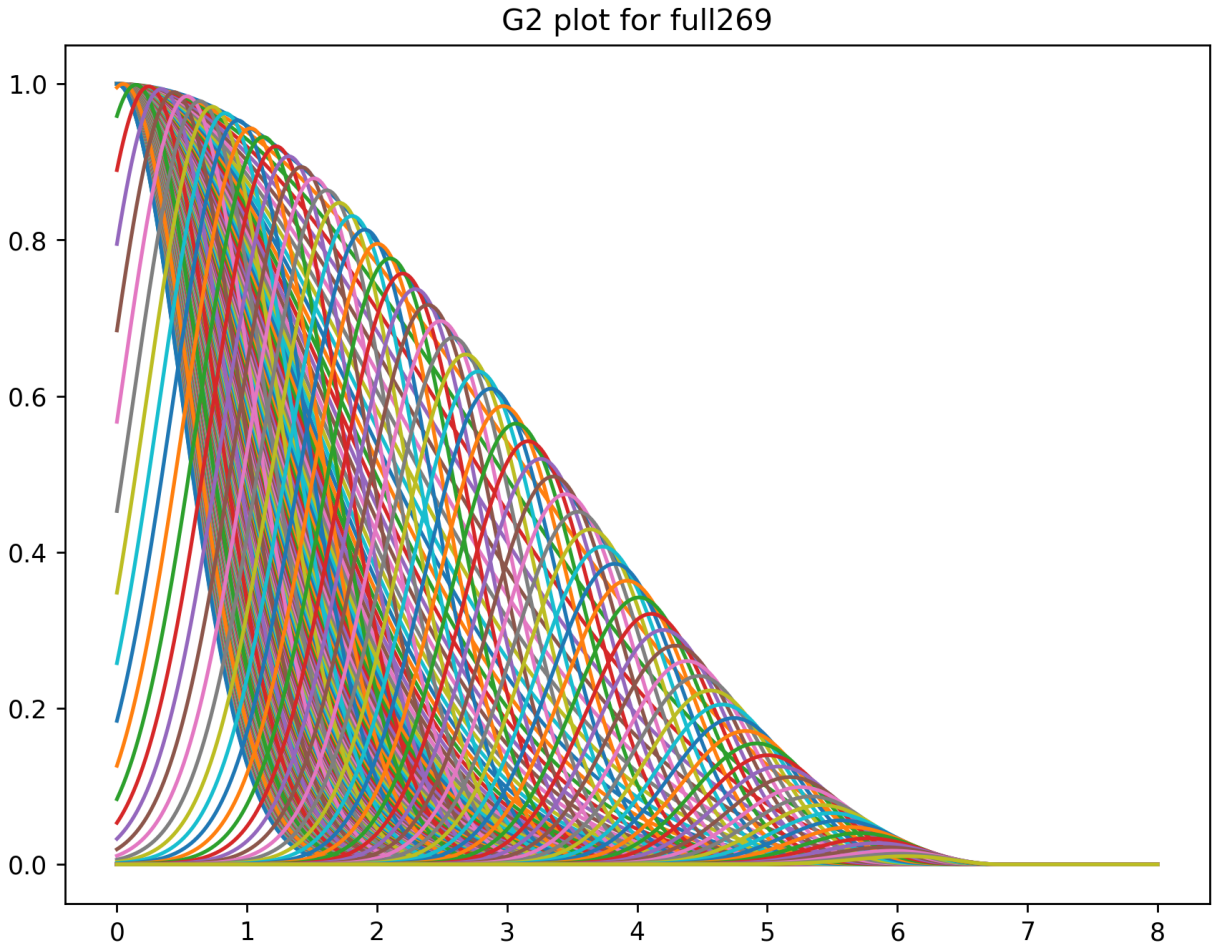


Fig 15: Physical representation of G2 269 variables

Verifying the data

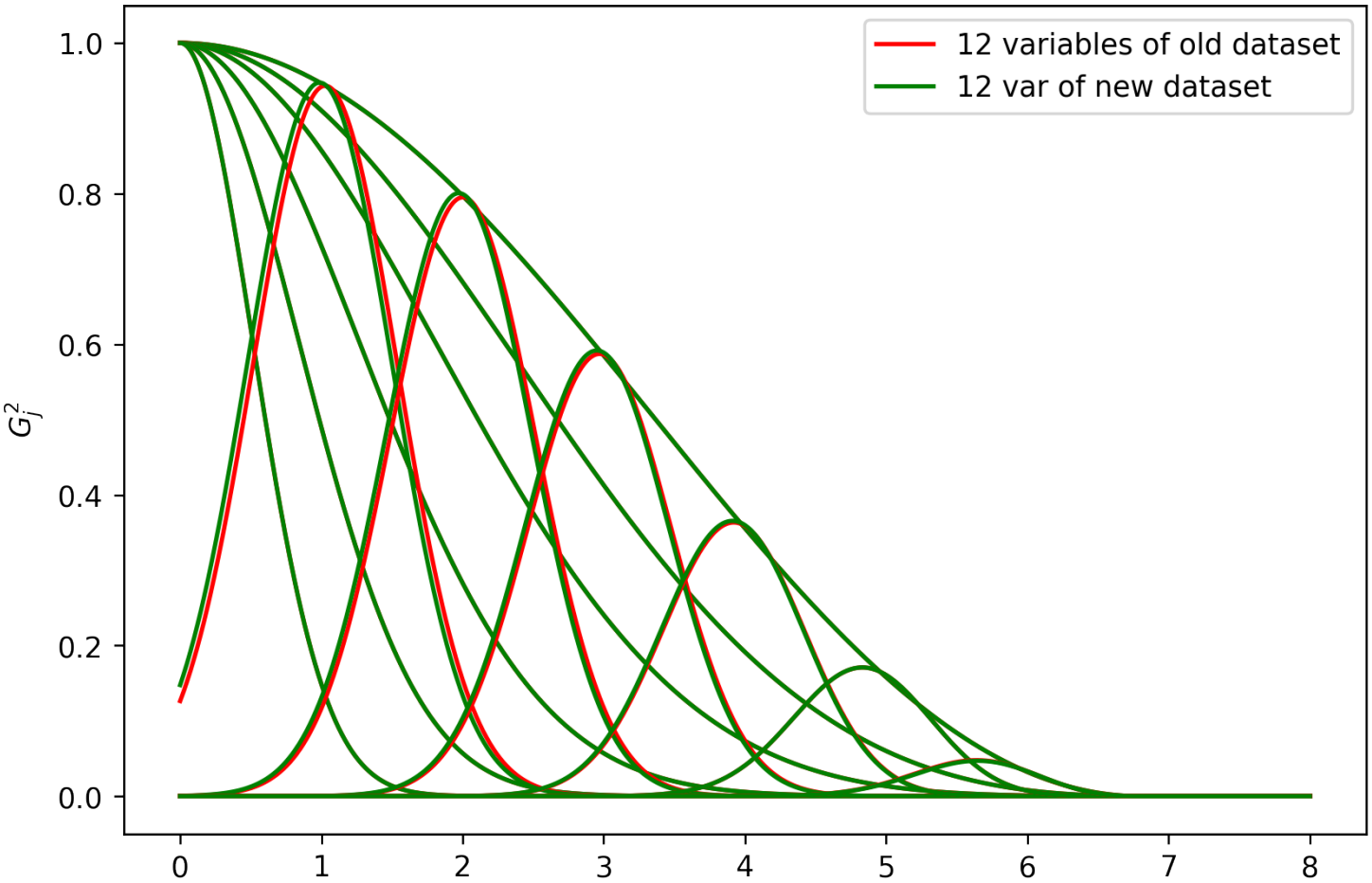


Fig 16: Matching G2 features of both case

Verifying the data

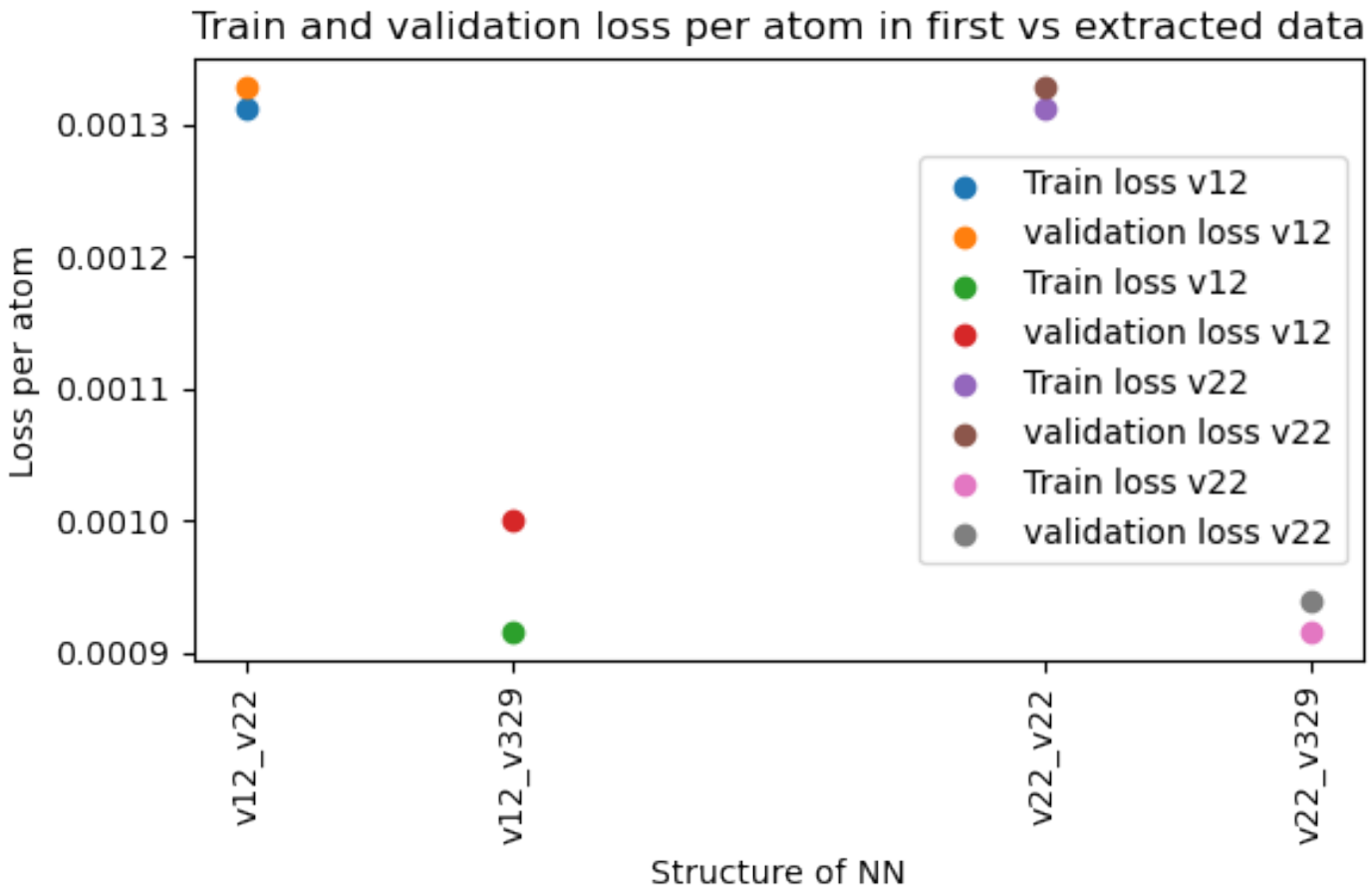


Fig 17: Comparing least loss of 12 and 22 variables in the case of old data and extracted data

Digging 329 variables

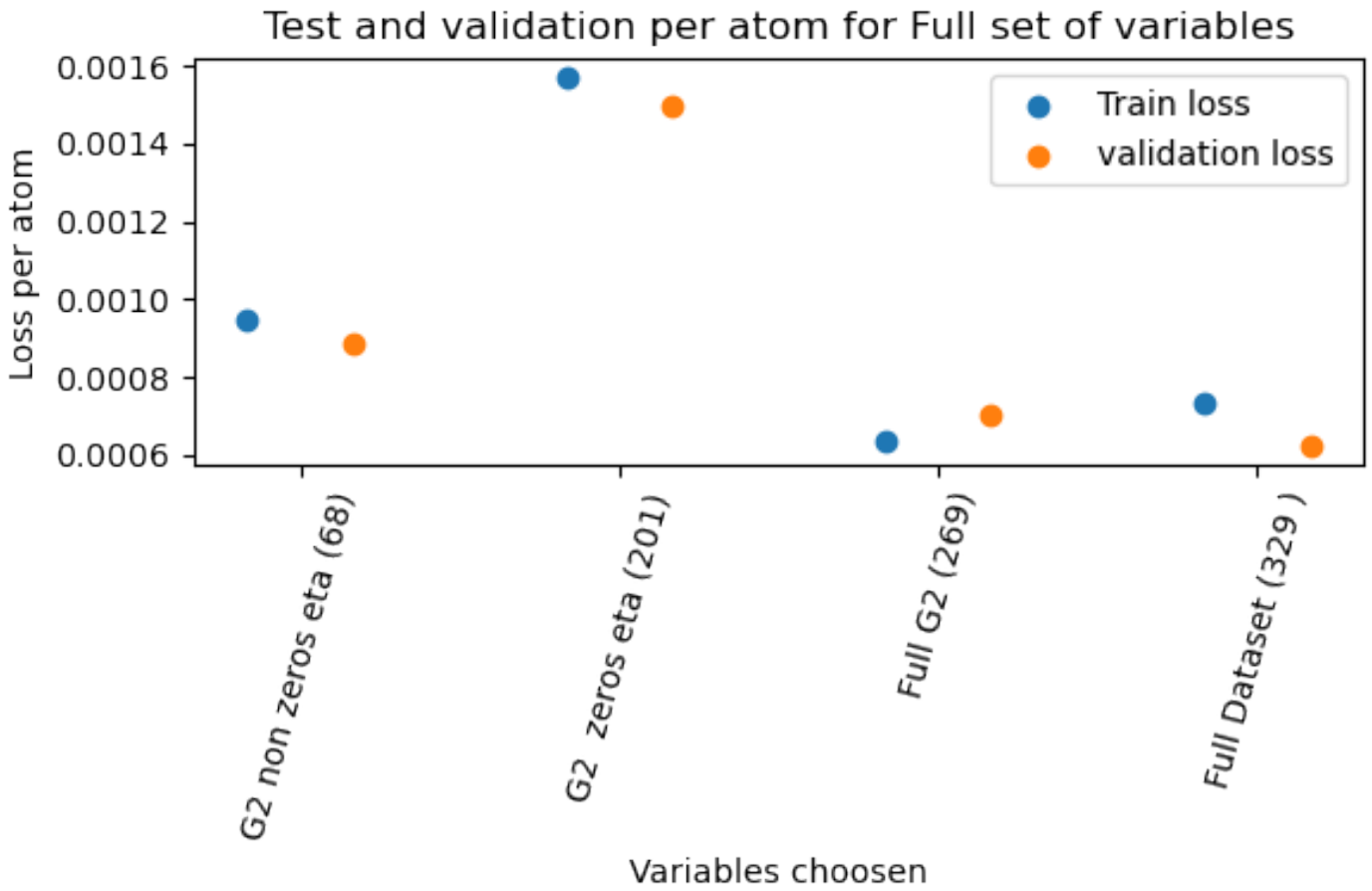


Fig 18: Training with variables defined by G_s

Randomly selecting variables

Training with randomly selecting variables

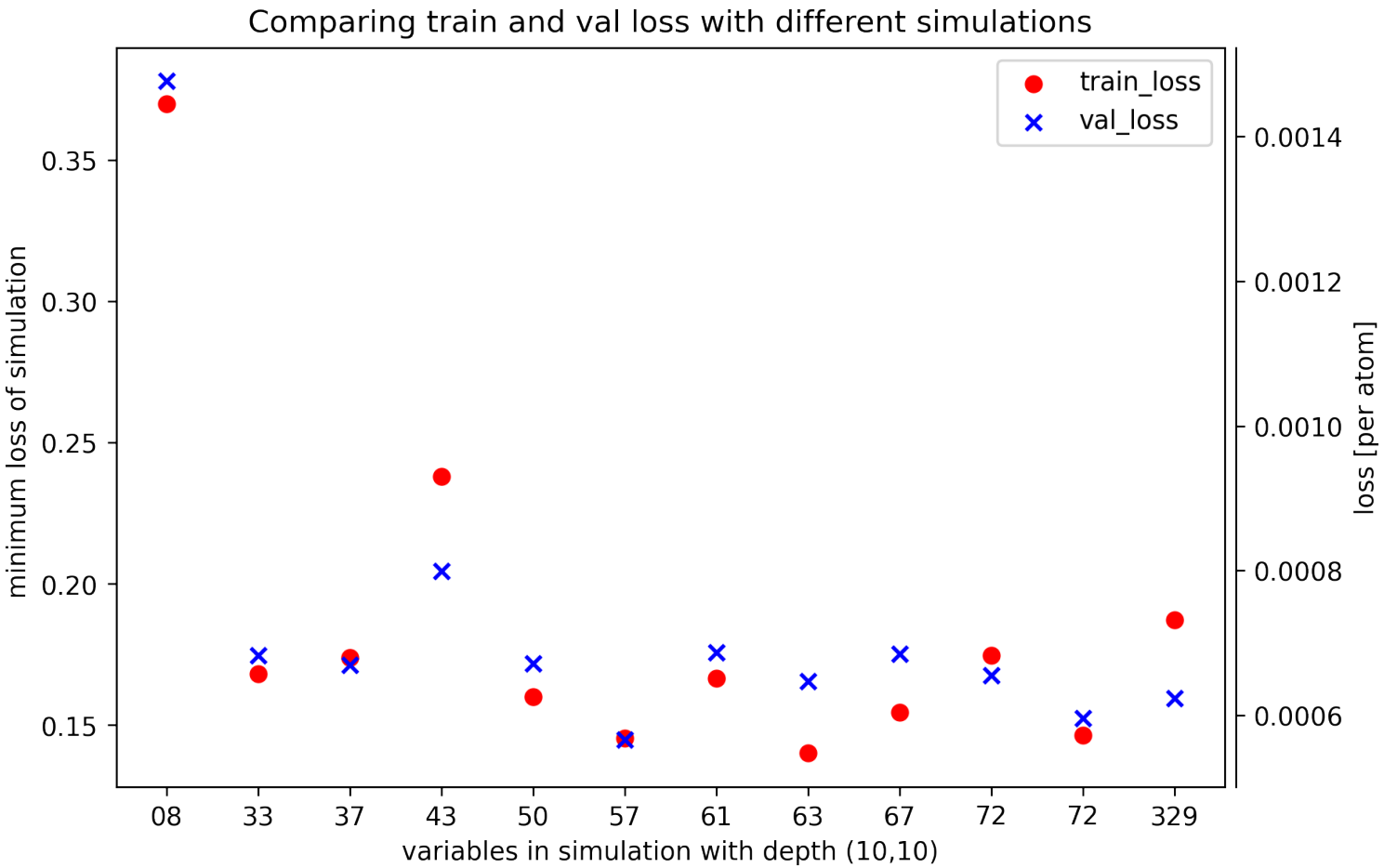
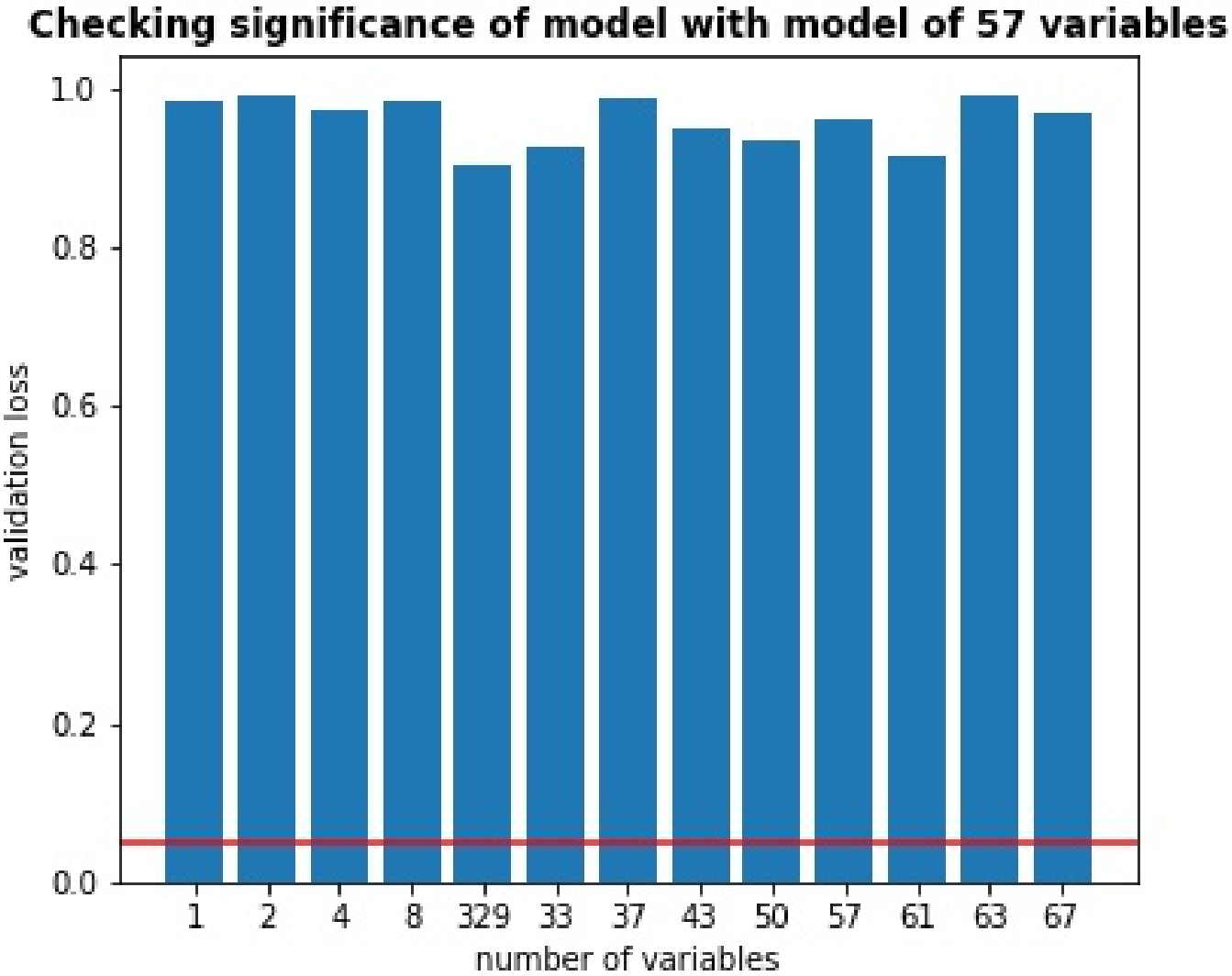


Fig 19: Training with various number of variables

Statistical Significance using t-test



Further Digging 57 variables

Digging with respect to G_s

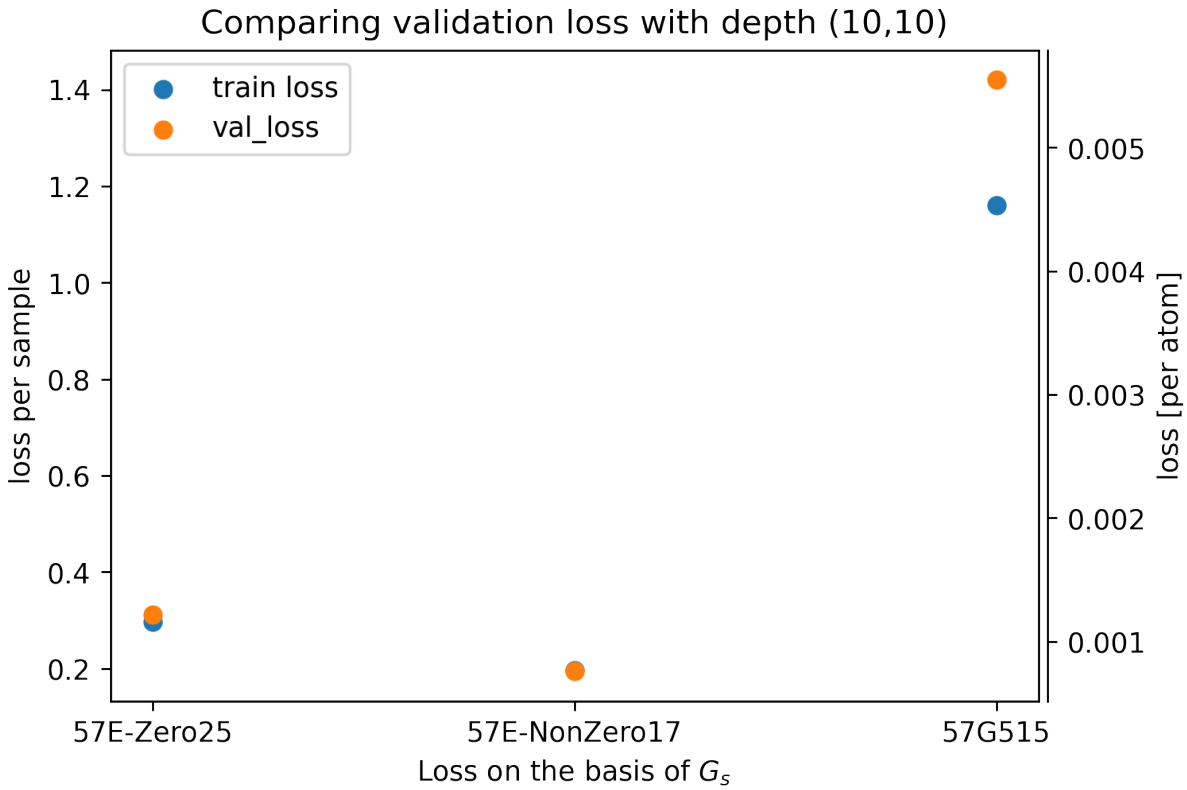
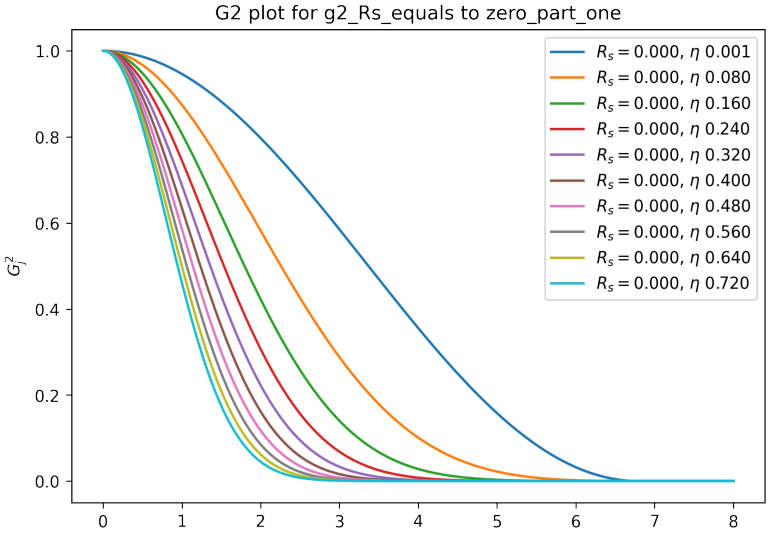
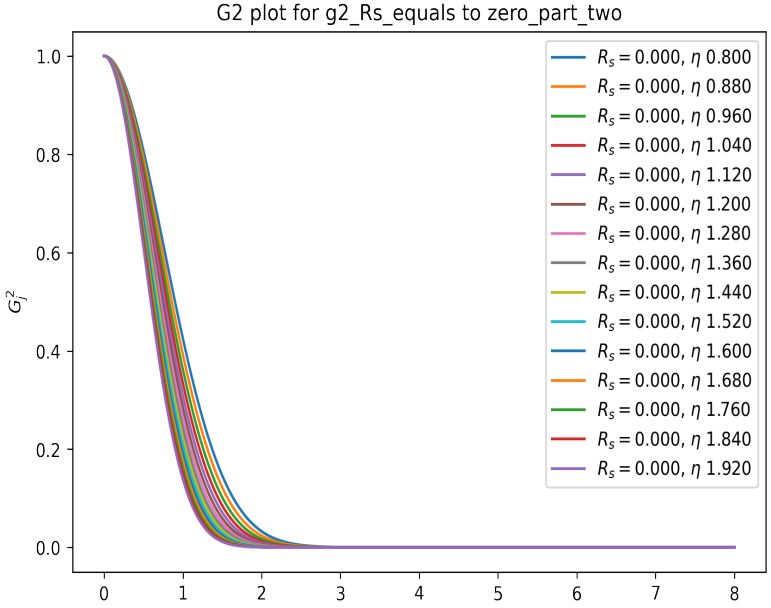
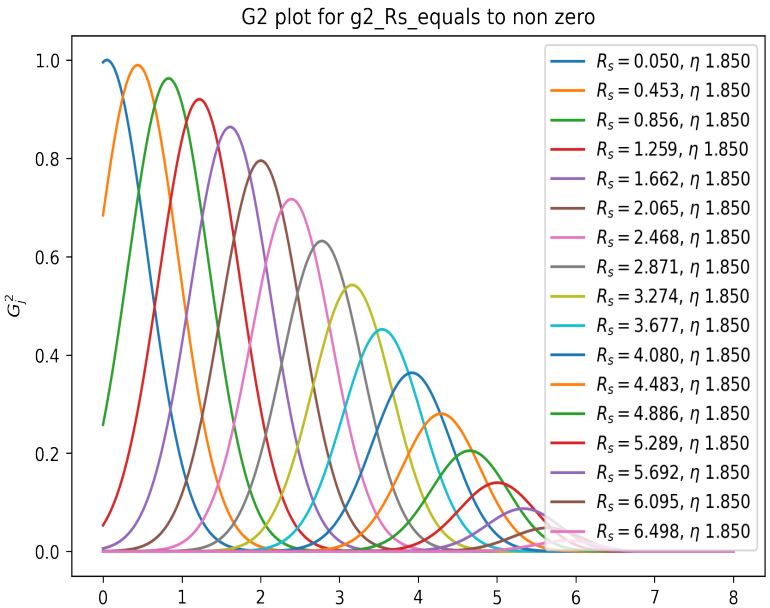


Fig 21: Training with various number of variables


G2 plot for g2_Rs_equals to zero_part_one
G_j^2
$R_s = 0.000, \eta\ 0.001$
$R_s = 0.000, \eta\ 0.080$
$R_s = 0.000, \eta\ 0.160$
$R_s = 0.000, \eta\ 0.240$
$R_s = 0.000, \eta\ 0.320$
$R_s = 0.000, \eta\ 0.400$
$R_s = 0.000, \eta\ 0.480$
$R_s = 0.000, \eta\ 0.560$
$R_s = 0.000, \eta\ 0.640$
$R_s = 0.000, \eta\ 0.720$


G2 plot for g2_Rs_equals to zero_part_two
G_j^2
$R_s = 0.000, \eta\ 0.800$
$R_s = 0.000, \eta\ 0.880$
$R_s = 0.000, \eta\ 0.960$
$R_s = 0.000, \eta\ 1.040$
$R_s = 0.000, \eta\ 1.120$
$R_s = 0.000, \eta\ 1.200$
$R_s = 0.000, \eta\ 1.280$
$R_s = 0.000, \eta\ 1.360$
$R_s = 0.000, \eta\ 1.440$
$R_s = 0.000, \eta\ 1.520$
$R_s = 0.000, \eta\ 1.600$
$R_s = 0.000, \eta\ 1.680$
$R_s = 0.000, \eta\ 1.760$
$R_s = 0.000, \eta\ 1.840$
$R_s = 0.000, \eta\ 1.920$


G2 plot for g2_Rs_equals to non zero
G_j^2
$R_s = 0.050, \eta\ 1.850$
$R_s = 0.453, \eta\ 1.850$
$R_s = 0.856, \eta\ 1.850$
$R_s = 1.259, \eta\ 1.850$
$R_s = 1.662, \eta\ 1.850$
$R_s = 2.065, \eta\ 1.850$
$R_s = 2.468, \eta\ 1.850$
$R_s = 2.871, \eta\ 1.850$
$R_s = 3.274, \eta\ 1.850$
$R_s = 3.677, \eta\ 1.850$
$R_s = 4.080, \eta\ 1.850$
$R_s = 4.483, \eta\ 1.850$
$R_s = 4.886, \eta\ 1.850$
$R_s = 5.289, \eta\ 1.850$
$R_s = 5.692, \eta\ 1.850$
$R_s = 6.095, \eta\ 1.850$
$R_s = 6.498, \eta\ 1.850$

Further Digging 57 variables

Different possibilities of groups of 57 variables

```
#Randomly selecting even spaced variables
a,b,c = 8,4,4 #for 57 variables
x,y,z= 1,202,1 #set A
#x,y,z= 2,203,2 #set B
#x,y,z = 3,204,3 #set C
#x,y,z = 4,205,4 #set D
g2_4_1 = np.array(range(x,201,a))
g2_4_2 = np.array(range(y,270,b))
g2_4 = np.concatenate((g2_4_1, g2_4_2), axis= None)
g5_4 = np.array(range(z,61,c))
```

Further Digging 57 variables

Different possibilities of groups of 57 variables.

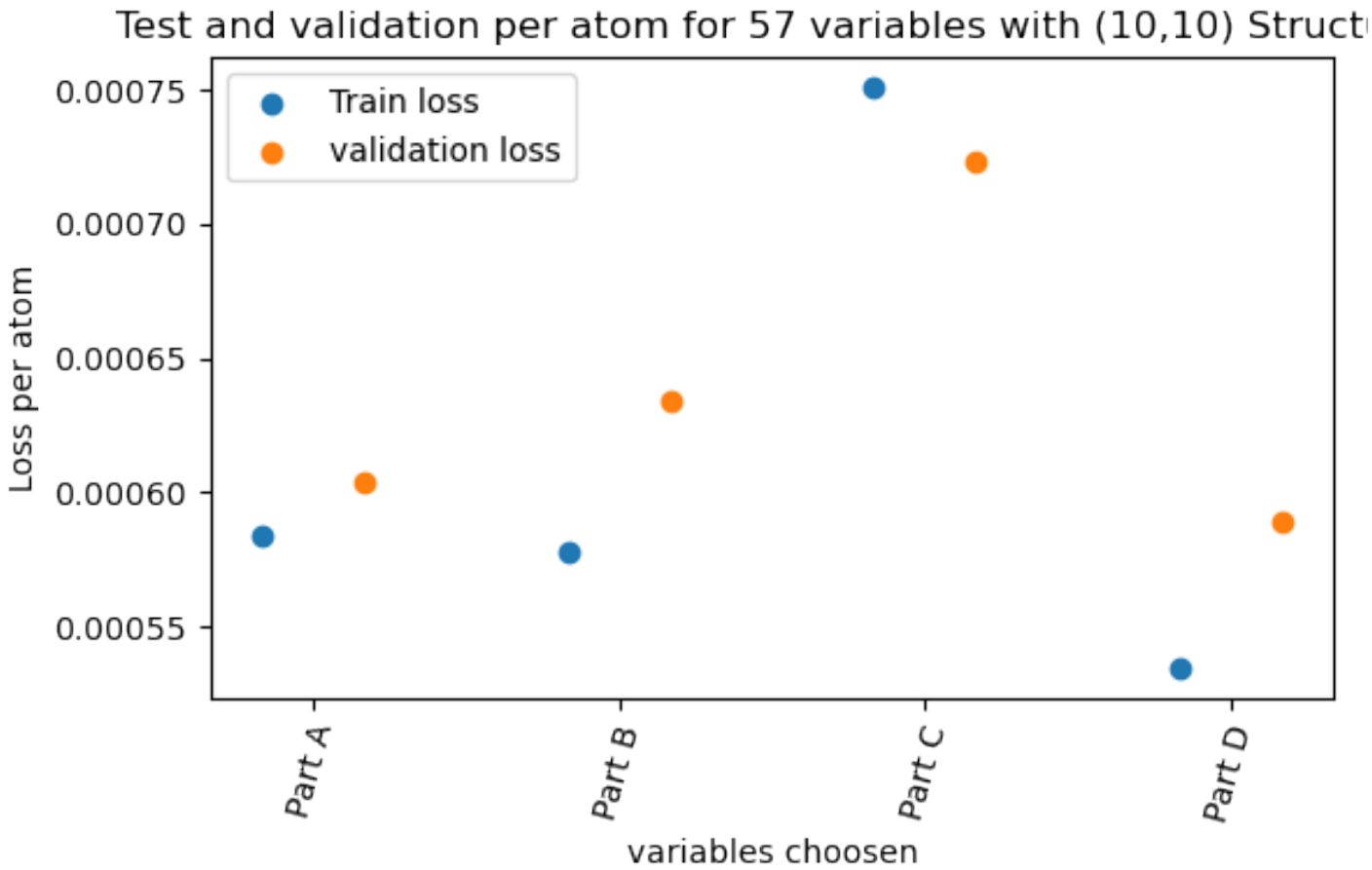


Fig 22: Training with various groups of 57 variables, part D gives the least validation loss of 0.150827 per configuration (0.00059 per atom).

Dropping one variable at a time

Different possibilities of groups of 57 variables

- Training with 56 variables
- Compared the least loss of all 57 models
- Retrained with variables after dropping 5 variables .

Dropping variables whose absence gives a better loss

Different possibilities of groups of 57 variables

- Starting from 57, dropped a variable at a time
- If loss increases, retake that variable again
- If loss decreases, permanently drop that variable
- At the end dropped four variables.

Dropping variables with respect to weight

Different possibilities of groups of 57 variables

- Dropped variables that have least absolute sum weight/ sum of weights .

Conclusion

Our Result

- The least loss decreased from 0.28 for 22 variables to 0.14478 for 57 variables.
- Dataset with 57 features gave the best precision.
- There can be one or more sets of variables that give better precision.
- Adding new variables with G_s non-zero values can give higher precision.

Conclusion

Hurdles during training process

- Least loss is not consistent
- Different Results at a different time
- Not feasible to replicate the training for every possibility.

Conclusion

Further possibilities

Lasso Net Regressor [5]

- A penalty is introduced to the original empirical risk minimization that encourages feature sparsity. The formulation transforms the combinatorial search to a continuous search by varying the level of the penalty.
- A proximal gradient algorithm is applied in a mathematically elegant way so that it admits a simple and efficient implementation. The method can be implemented by adding just a few lines of code to a standard neural network.

Conclusion

Any Questions Please?

Conclusion

Any Questions Please?

Thank you

- Heartful Thanks to my supervisors Noel JAKSE & Emilie DEVIJVER
- Thanks to my Professors and Friends for your support in 3 years of journey
- Thanks to Ensimag, UGA, and SIMAP administration Staff for your support
- Thanks to my family.

[1] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *Journal of Chemical Physics*, 134(7), 2011. ISSN 00219606. doi: 10.1063/1.3553717.

[2] Jörg Behler. Constructing high-dimensional neural network potentials: A tutorial review. *International Journal of Quantum Chemistry*, 115(16):1032–1050, 2015. ISSN 1097461X. doi: 10.1002/qua.24890.

[3] Anthony Saliou. Kungliga Tekniska Höngskolan, Building neural network potentials for Lennard-Jones and Aluminium systems Anthony Saliou Master Thesis for the MSc in Engineering Physics KTH , Royal Institute of Technology. (July), 2020.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*, Jan 2017. URL `http://arxiv.org/abs/1412.6980`. arXiv: 1412.6980.

[5] Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021. URL `http://jmlr.org/papers/v22/20-848.html`.