Machine Learning: The Hubbard model

Niels Billiet

May 16, 2018

Contents

1 Research overview **3**

2 The Hubbard model **4**

3 Theoretical background: Neural networks **7**

3.1 Mathematical description of neural networks 7

3.1.1 Hidden units 8

3.1.2 Activation function 8

3.2 Training of the neural network 9

3.2.1 Initialization phase 10

3.2.2 Training phase 11

4 Two site Hubbard model **14**

4.1 Training data 15

4.2 Test data 15

4.3 Neural networks 15

4.3.1 General assessment of the hyper parameter sweep 17

4.3.2 Reliability and reproducability of the networks . . . . .

5 Four site Hubbard model

5.1 Training data . . . .

5.1.1 Randomized dataset . . . .

5.1.2 Data augmentation . . . .

5.2 Test dataset . . . .

5.2.1 Test set based upon relevant chemical structures . . . .

5.2.2 Test data based upon quantumchemical effects . . . . .

5.2.3 Test data based on structures with less complexity . . .

5.3 Neural networks . . . .

5.3.1 Architectures: General overview and general performance assesment . . . .

5.3.2 Network assesment through the visualization of the potential energy surface . . . .

6 Conclusions

1 Research overview

2 The Hubbard model

A well established a approximative methode in quantum chemistry is the H$\ddot{u}$ckel method or the Hckel molecular orbital theory. In this method a simplified description of π systems is given through a linear combination of p_z orbitals. Given the orthonormal basis $\{p_{z,i}\}$ where the size of the basis is given by the amount of conjugated carbon atoms we can construct the matrix $\mathbf{H}$ as follows

$$H_{ij} \begin{cases} \alpha & \text{i=j} \\ \beta & \text{i and j are nearest neighbouring atoms} \\ 0 & \text{i and j are not nearest neighbouring atoms} \end{cases}$$

Using this very simplified model we can already get a understanding of the electronic structure of conjugated π system without having to resolve to complex computational methods. A more generalized approach to the H$\ddot{u}$ckel method can be found in the Hubbard model. Indeed, the Hubbard model finds its origin within the solid state physics and performs a description in a similar fashion to the H$\ddot{u}$ckel method?

- The Hubbard model gives a simple description of a system in which electrons are free to delocalize in the system. It can thus be used to describe conductors and electron delocalization in molecules
- Integration of a electrostatic repulsion term provides some form of electron correlation in the system and thus offers some improvement over mean field methods such as Hartree-Fock
- On a computational level the full CI wavefunction of the system can be computed with a finite basis.

The Hubbard Hamiltonian is constructed using fermionic creation and annihilation operators.

- $\hat{c}^{\dagger}_{i,\sigma}$ is the creation operator, which creates an electron on site i with spin σ
- $\hat{c}_{i,\sigma}$ is the annihilation operator, which creates an electron on site i with spin σ

Where these operators adhere to the following anticommutation relation

$$\{\hat{c}_{i,\sigma}, \hat{c}^{\dagger}_{j,\sigma}\} = \hat{c}_{i,\sigma}\hat{c}^{\dagger}_{j,\sigma} + \hat{c}^{\dagger}_{j,\sigma}\hat{c}_{i,\sigma} = \delta_{ij}$$

$$\{\hat{c}^{\dagger}_{i,\sigma}, \hat{c}^{\dagger}_{j,\sigma}\} = \{\hat{c}_{i,\sigma}, \hat{c}_{j,\sigma}\} = 0$$

Using these operators the Hubbard Hamiltonian is then given by

$$\hat{H} = \sum_{<i,j>,\sigma}^{N} -t_{ij}\left(\hat{c}^{\dagger}_{i,\sigma}\hat{c}_{j,\sigma} + c.c.\right) + \sum_{i=1}^{N} U_i \hat{n}_{i\downarrow}\hat{n}_{i\uparrow}$$

Where

- t_{ij} is the hopping parameter, this parameter is proportional with the internuclear distance between the atoms and with the overlap of the orbitals in question
$$t_{ij} \propto \frac{1}{R_{ij}} \propto S_{ij}$$

- U is the electrostatic interaction term, expressed in terms of the hopping parameter. A positive U indicates repulsion between two electrons situated on the same site

- $\hat{n}_{i,\sigma}$ is the counting operator working on site i. The counting operator is defined as the product of the annihilation operator and creation operator on site i with spin σ. This operator has 2 possible values when applied to a basis function i.e. 1 when there is an electron on site i with spin σ or 0.

The Hamiltonian matrix $\mathbf{H}$ is consequently constructed by applying $\hat{H}$ to every basis function of the system. In this research we will focus on Hubbard systems where there are as many electrons present as there are sites (i.e. half filled systems). In order to provide a complete description of the eigen spectrum of the molecule in question we have to take all possible spin states into account. For a system containing N electrons we can describe the system with the following spin components
$$|S_{z,tot}| \in \left[0, \frac{1}{2}, \frac{1}{2}, \dots, \frac{N}{2}\right]$$
Each of these spin sectors has a subbasis associated with them which corresponds to all possible combinations of electrons across the available sites (and taking the Pauli principle into consideration). For the spinsector where there are N electrons in total present of which N_α have a α spin and N_{beta} electrons have a β spin we get a total number of basis functions equal to
$$\binom{N}{N_\alpha}\binom{N}{N_\beta}$$
The complete eigenspectrum of the molecule can consequently be computed by combining the subbasis of each spin sector into the complete basis and diagonalizing $\mathbf{H}$. In general this comes down to diagonalizing each individual spinsector submatrix in the Hamiltonian as there is no spin inversion component present in the Hamiltonian operator. Thus preventing the transformation of a basisfunction from one spin sector to another spin sector.

However the simplest representation of the any Hubbard system can be given by the adjacency matrix. Indeed suppose we have a Hubbard system consisting of N sites then we can construct a adjacency matrix $\mathbf{A}$ as follows

$$\begin{bmatrix} U_1 & t_{12} & t_{13} & \dots & t_{1N} \\ t_{21} & U_2 & t_{23} & \dots & t_{2N} \\ t_{31} & t_{32} & U_3 & \dots & t_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{N1} & t_{N2} & t_{N3} & \dots & U_N \end{bmatrix}$$

This adjacency matrix is a Hermitian matrix and thus contains all the necessary information concerning the system in its upper or lower triangle. It is this adjacency matrix that also forms the link back to the H$\ddot{u}$ckel Hamiltonian. Indeed if one replaces the U value by α and the t value by β one forms the Hamiltonian matrix in the H$\ddot{u}$ckel method easily.

3 Theoretical background: Neural networks

Many problems present in everyday life and within the scientific community can be described as a non-linear problem which are on their own approachable through complex statistical methods. However, over the recent years the interest in developing artificial intelligence which could potentially analyze and solve said problems has increased and has lead to the development of a wide array of AI tools.

Among these tools, a very powerful method called Deep Neural Networks (DNN) has emerged and has shown a lot of promise with regard to solving complex problems such as picture recognition, speech recognition, etc. These neural networks take a lot of inspiration from biological brain structure

- The brain consists of a large collection of neurons that form a strongly connected netwerk that cooperates in order to solve a problem, perform a complex action, etc.
  $\rightarrow$ **Strong interconnectivity**
- A desired action arises only when the brain receives a specific signal for example under the form of a neurotransmitter. This signal is transmitted starting from a small number neurons to a large collection.
  $\rightarrow$ **Starting from a simple input signal an output is generated through amplification and transformation of the original signal**
- In order for a signal to be transmitted through a neuron, a certain threshold membrane polarization must occur within the neuron's cell membrane.
  $\rightarrow$ **Only input signals of a certain strength will result in an output**
- In the brain specific regions are specified towards performing or interpreting specific actions.
  $\rightarrow$ **Presence of local specialisation in the network**

The DNN attempts to capture all the aforementioned characteristics with it's algorithm to approximate the behavior of a biological brain

3.1 Mathematical description of neural networks

The deep neural network is generally described through three different mathematical structures

1. **Input layer**, The first layer of the neural network consists of a representation of the given input data. This data is represented as a tensor and will be used a basis input for consecutive computations in the following layers to transform it to the desired output.
2. **Hidden layer(s)**, a data structure consisting of **hidden units**, also referred to as **nodes** or neurons, which perform a computation on the output of the previous layer. Additionaly there is also a **bias unit** present

in each hidden layer. After the computation within the node it's output gets transformed by a **activation function** and will serve as the input for the following layer

3. **Output layer**, postioned after the last hidden layer and represents the output. This data is represented through a tensor as well

Looking at this summary it becomes clear that as the input information enters the network it is transformed and combined in such a fashion that it mirrors the output data. As the information flows through the network it becomes more correlated to the output.
Figuur van een simpel neuraal network

3.1.1 Hidden units

The fundamental unit of the neural network is the hidden units or nodes. These nodes represent a linear combination of a set outputs from the previous layer. Depending on the interconnectivity of the layers with respect to each other we can go from a one to one mapping to a mapping of the whole previous layer in each node of the current layer. Suppose that the previous layer, numbered as (n-1), in the network consists of M nodes each characterized by a output z, then we can express the form a of each node in the current layer n as follows

$$a_i^n = \sum_j^M w_{ij}^n z_j^{n-1} + w_{i0}^n \tag{1}$$

Where in this equation we have assumed that the network is a dense network, i.e. a fully connected network. In this equation the w_{i0}^n represents the bias of the node. More about this bias value shall be discussed in the following section. From this equation it becomes apparent quickly that depending on the amount of nodes in the previous layer and in the current layer the possibility to create different linear combinations of information increases with the layer size.

3.1.2 Activation function

The next important component of the neural network and some moght argue the most important component is the activation unit. In the paragraphs above we defined the node of the current layer as the linear combination of the outputs of the previous layer. This output however is not simply the linear combination, this combination of outputs is first transformed using a non linear activation function h

$$z_i^n = h(a_i^n) \ \textit{where h() is a non-linear function} \tag{2}$$

The use of the activation function allows us to use linear combinations of input variables to simulate a non-linear function. It is this property that grants the neural network algorithms it's strength when it comes to modeling. Indeed if the activation function would be absent than the neural network would be reduced to a consecutive linear regression algorithm and would fail to properly process non linear data. The approximative properties of these networks are so strong that they are sometimes referred to as universal approximators, however this is only the case if a suitable combination of weights is found that fits the training data.

The choice of different activation functions is often linked to the problem off interest. Indeed, generally speaking neural networks are used in complex classification problems or regression problems. Solely based on the nature of these problems we can intuitively tell that the nature of the non linearity will generally be different for both and thus will preferably use specific types of activation functions.

A common choice of activation function for regression like problems, and the ones that will be utilized in this research, are those of the ReLU family of functions. The ReLU type functions or *Rectified Linear activation Unit* consists of three different functions: ReLU, Leaky ReLU and Parametric ReLU

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$$

$$LReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0,01x & \text{if } x < 0 \end{cases}$$

$$PReLU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\text{x} & \text{if } x < 0 \quad \text{with } \alpha \in \text{]0,1[} \end{cases}$$

The non linearity of these functions are centered around the origin and perform a sign discrimination. It should be apparent however that the three different activation functions presented here represent a stepwise modification in degree of signal dampening. Indeed where the ReLU function dampens a signal beyond a certain threshold ($x < 0$) the Leaky ReLU function allows for a small negative signal to be passed throughout the network. The parametric ReLU in comparison allows for a fine tuned signal dampening when it concerns a negative signal with the only boundary being $\alpha \neq 1$ as this would revert the activation function to a linear function

FIGUUR VAN DE VERSCHILLENDE ACTIVATIE FUNCTIES

The bias node present in each layer serves the purpose of causing a better fit to the training. The presence of this value allows the activation to be shifted from it's original origin. In terms of the ReLU type activation function the bias allows the dampening segment of the function to be shifted in order to filter data more appropriately. This bias parameter thus represents a extra degree of freedom which requires optimization during the training phase of the nueral network

3.2 Training of the neural network

In order for the neural network to capture any features present in the data a learning process is required. In order for the network to learn something we require three different datasets

1. **Training dataset**, used as the source of knowledge which the network will use to adjust its weights. Via weight update the network will attempt to capture any features present in the training data

2. **Validation dataset**, used a reference to check if the network is learning the features correctly. This dataset is constructed similarly to the training data but should not be present in the actual training data. Correct interpretation of this data is considered to be an indication that the network has captured the important features of the data

3. **Test dataset**, final evaluation dataset after the training is done. Performance on this data is considered as a clear indication concerning the robustness and accuracy of the network

Training a network generally consists of three phases and will be discussed in the following sections.

1. Initialization phase
2. Training phase
   (a) Feed-forward phase
   (b) Back propagation phase
   (c) Validation phase
3. Test phase

3.2.1 Initialization phase

After the network architecture has been decided upon the initialization phase has commenced. Indeed, making the structure of the network does not suffice for the training phase to be initiated. Before training can be initiated we must first assign a initial value to the weights of the network. Many different methods of drawing weights exist

1. **Zero/one initializer** Apply a uniform weight of zero or one across the whole weight space. Standardly this is not done due too the heavy initial oscillations of the network evaluation function

2. **Uniform initializer** Randomly draw weight from a predefined or custom continuous uniform distribution.

3. **Guassian initializer** Randomly draw weight from a Gaussian distribution with a predefined mean and standard deviation.
   - Adaptive variance algorithm, which adjust the width of the weight distribution based on the weight space dimension
   - Truncated normal sampling, sampling within one or two standard deviations of the mean and redrawing extreme values

4. **Orthogonal initializer** Randomly generate a orthogonal weight matrix

Different initialization algorithms will have a effect on the training length and in case of a complex error surface possibly affects the eventual outcome of the neural network. The standard initializer however is the uniform distribution and will be determined by the interval and the random seed.

3.2.2 Training phase

During the training phase the input data gets fed to the network. this is referred to as the feed forward phase as the original data moves down the network and undergoes consecutive non linear transformation until a predicted outcome y is achieved. This outcome is on one hand dependent on the original input vector $\mathbf{x}$ but on the other hand mainly upon the weights and biases as these are the unique variables to the network itself. The network can thus be written down as a function $y(\mathbf{x}, \mathbf{w})$

The next step in the training would be too evaluate the predicted outcome of the network, which is performed through the means of a loss function E. Let $\mathbf{t}_n$ be the target vector of the input vector $\mathbf{x}_n$ which is part of a set $\mathbf{X}$ of size N and $y(\mathbf{x}, \mathbf{w})$ be the network function then we can define the error function E as follows

$$E = \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}, \mathbf{w}) - \mathbf{t}_n)^2$$

From this equation it becomes clear that the error function is only dependent on the network weights and thus optimization of the network and its error function will require minimization of the derivatives with respect too the weights

$$\nabla E(w) = 0$$

However one should note that finding the global minimum for this function is not a easy task. Indeed the weight space itself is a multidimensional space on which the error function has a non linear dependence which results in a high amount of local minima. On top of this we know that the weight space itself has inherent symmetry related to the amount of nodes in a layer and the activation function itself. For a layer consisting of M hidden units in which each node is characterized by a unique combination of weights there will be a M! permutational symmetry that can lead to a equivalent minimum.

Generally speaking the optimization of the neural network will be performed using a gradient descent optimization algorithm which is performed during the back propagation phase of the network. As was mentioned above our intent is too minimize the global error function which it self is a function of the network weights. After computation of the error function for a specific input vector the back propagation algorithm utilizes this error to compute changes in the weights and will send these errors through the network in a backwards function to perform a layer wise update of the weight space. The general scheme for such a back propagation algorithm goes as follows

- Compute the respons for a given input vector $\mathbf{x}_n$. The output vector $\mathbf{y}_n$ will be dependent upon the activation outputs of the previous layer
- Evaluate the derivative of the error function in function of the weights. The derivative cannot be computed directly as the weights themselves are function of the activation function. Using the chain rule we get the following derivative for the weight i of node j

$$\frac{\delta E}{\delta w_{ji}} = \frac{\delta E}{\delta z_j}\frac{\delta z_j}{\delta a_j}\frac{\delta a_j}{\delta w_{ji}} \tag{3}$$

Which requires the derivative of the activation function and thus is the reason why we require the activation function to be differentiable. We can see that this equation becomes more complex as we advance further back in the neural network. Indeed optimization of the network for nodes that are not connected to the output will require knowlegde of the derivatives of the layers closer too the output.

- After computing all the derivatives the weights are updated using the following update rule
$$w_{ji}^{\tau+1} = w_{ji}^{\tau} - \eta \frac{\delta E}{\delta w_{ji}} \tag{4}$$
Where τ represents the iteration cycle and η the learning rate of the network.

The back propagation algorithm can be fine tuned with respect to how much data gets processed in each back and forth cycle. Adjusting the batch size, i.e. the amount of data points that are used to evaluate the loss function, will have a effect on how the network weights are update in each cycle. Point wise iteration can adjusts the weight with respect to each input vector $\mathbf{x}_n$ can be useful when the dataset is very homogeneous and similarly structured. However if the data has a lot of structural variety this can cause the error function to display irregular behavior and inconsistent weight changes during a training cycle. This is why training in batches of data can offer a advantage as the loss function will evaluate the population error and will minimize this error, ensuring generality of the network.

The back propagation algorithm is repeated until all data points have been passed through the network. After these update cycles are completed the loss function is evaluated for the validation dataset which will be used a monitor to asses the network training progress. These training cycles continue for a set amount of cycles called epochs untill the preset monitor has reached a minimal value.

The afore mentioned back propagation scheme is the general algorithm by which the weights are updated in a neural network. However many different variations on this algorithm exist and these are related to the way the weights update is calculated. In this research we will use the Adaptive Moment Estimation or ADAM algorithm.

As can be seen from the algorithm pseudocode we notice that the ADAM algorithm is a variation of the stochastic gradient descent (SGD) algorithm that is the standard in the back propagation algorithm however there afre some significant differences. Unlike the SGD algorithm, ADAM utilizes first and second order moments, $\mathbf{m}$ en $\mathbf{v}$, which decay over time due too the β^t in a exponential fashion in order too accelerate the convergence of the objective function too a minimum.

These moments represent the mean of the gradient (first order moment) and the uncentered variance (second order moment) of the gradient. Acceleration of the converge is then achieved by remembering the gradient from the previous timestep in the optimization procedure and subtracting the current and previous gradient from each other. Should the gradient of the previous timestep be significantly larger than that of the current then the algorithm will force the update too happen in the direction of the largest descent.

Algorithm 1 ADAM

```
Require: $\alpha$: Predetermined step size or learning rate
Require: $\beta_1, \beta_2 \in [0, 1[$]: exponential decay rates for the estimated moments
    of the first order and second order moment
Require: $f(\theta)$: A stochastic objective function in terms of the parameters $\theta$
Require: $\theta_0$: The initial parameter vector
 1: $m_0 \leftarrow 0$
 2: $v_0 \leftarrow 0$
 3: t $\leftarrow$ 0
 4: while $\theta_t$ not converged do
 5:     $t \leftarrow$ t+1
 6:     $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
 7:     $m_t \leftarrow \beta_1.m_{t-1} + (1-\beta_1)g_t$
 8:     $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)g_t^2$
 9:     $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
10:     $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
11:     $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$
12: end while
13: return $\theta_t$
```

4 Two site Hubbard model

In this first section concerning the actual research itself we will begin with the study of the simplest of all the Hubbard models, being the two site system. Specifically we will focus ourselfs on the system that is half filled. As was mentioned in the section concerning the Hubbard model itself this system is characterized by the following adjacency matrix

$$\begin{bmatrix} U_1 & t_{12} \\ t_{21} & U_2 \end{bmatrix}$$

Due too the hermicity of this adjacency matrix we can characterize the system by its upper or lower triangle leading to a total of three parameters which will describe the energy of the ground state

$$\begin{bmatrix} U_1 & t_{12} & U_2 \end{bmatrix}$$

Indeed one can verify easily that the ground state will be dependent on these parameters only. For a system consisting of two sites and two electrons there is a total of three spin sectors $[-1, 0, 1]$. The basis functions for these spin sector are the following

$$|S_Z = 1\rangle = \{|\uparrow, \uparrow\rangle\}$$
$$|S_Z = -1\rangle = \{|\downarrow, \downarrow\rangle\}$$
$$|S_Z = 0\rangle = \{|\uparrow\downarrow, 0\rangle, |\uparrow, \downarrow\rangle, |\downarrow, \uparrow\rangle, |0, \uparrow\downarrow\rangle\}$$

The Hubbard Hamiltonian thus forms a six by six matrix which can be diagonilized easily

$$\begin{bmatrix}
\langle \uparrow\downarrow, 0|\hat{H}|\uparrow\downarrow, 0\rangle & \langle \uparrow\downarrow, 0|\hat{H}|\uparrow, \downarrow\rangle & \langle \uparrow\downarrow, 0|\hat{H}|\downarrow, \uparrow\rangle & \langle \uparrow\downarrow, 0|\hat{H}|0, \uparrow\downarrow\rangle & \langle \uparrow\downarrow, 0|\hat{H}|\uparrow, \uparrow\rangle & \langle \uparrow\downarrow, 0|\hat{H}|\downarrow, \downarrow\rangle \\
\langle \uparrow, \downarrow|\hat{H}|\uparrow\downarrow, 0\rangle & \langle \uparrow, \downarrow|\hat{H}|\uparrow, \downarrow\rangle & \langle \uparrow, \downarrow|\hat{H}|\downarrow, \uparrow\rangle & \langle \uparrow, \downarrow|\hat{H}|0, \uparrow\downarrow\rangle & \langle \uparrow, \downarrow|\hat{H}|\uparrow, \uparrow\rangle & \langle \uparrow, \downarrow|\hat{H}|\downarrow, \downarrow\rangle \\
\langle \downarrow, \uparrow|\hat{H}|\uparrow\downarrow, 0\rangle & \langle \downarrow, \uparrow|\hat{H}|\uparrow, \downarrow\rangle & \langle \downarrow, \uparrow|\hat{H}|\downarrow, \uparrow\rangle & \langle \downarrow, \uparrow|\hat{H}|0, \uparrow\downarrow\rangle & \langle \downarrow, \uparrow|\hat{H}|\uparrow, \uparrow\rangle & \langle \downarrow, \uparrow|\hat{H}|\downarrow, \downarrow\rangle \\
\langle 0, \uparrow\downarrow|\hat{H}|\uparrow\downarrow, 0\rangle & \langle 0, \uparrow\downarrow|\hat{H}|\uparrow, \downarrow\rangle & \langle 0, \uparrow\downarrow|\hat{H}|\downarrow, \uparrow\rangle & \langle 0, \uparrow\downarrow|\hat{H}|0, \uparrow\downarrow\rangle & \langle 0, \uparrow\downarrow|\hat{H}|\uparrow, \uparrow\rangle & \langle 0, \uparrow\downarrow|\hat{H}|\downarrow, \downarrow\rangle \\
\langle \uparrow, \uparrow|\hat{H}|\uparrow\downarrow, 0\rangle & \langle \uparrow, \uparrow|\hat{H}|\uparrow, \downarrow\rangle & \langle \uparrow, \uparrow|\hat{H}|\downarrow, \uparrow\rangle & \langle \uparrow, \uparrow|\hat{H}|0, \uparrow\downarrow\rangle & \langle \uparrow, \uparrow|\hat{H}|\uparrow, \uparrow\rangle & \langle \uparrow, \uparrow|\hat{H}|\downarrow, \downarrow\rangle \\
\langle \downarrow, \downarrow|\hat{H}|\uparrow\downarrow, 0\rangle & \langle \downarrow, \downarrow|\hat{H}|\uparrow, \downarrow\rangle & \langle \downarrow, \downarrow|\hat{H}|\downarrow, \uparrow\rangle & \langle \downarrow, \downarrow|\hat{H}|0, \uparrow\downarrow\rangle & \langle \downarrow, \downarrow|\hat{H}|\uparrow, \uparrow\rangle & \langle \downarrow, \downarrow|\hat{H}|\downarrow, \downarrow\rangle
\end{bmatrix}$$

Which is reduced too

$$\begin{bmatrix}
U_1 & t_{12} & t_{12} & 0 & 0 & 0 \\
t_{12} & 0 & 0 & t_{12} & 0 & 0 \\
t_{12} & 0 & 0 & t_{12} & 0 & 0 \\
0 & t_{12} & t_{12} & U_2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

When the two atoms are the same the solution of the eigenvalue equation simplifies to the following eigenvalues

$$\lambda_1 = 0$$
$$\lambda_2 = 0$$
$$\lambda_3 = 0$$
$$\lambda_4 = U$$
$$\lambda_5 = \frac{1}{2}\left(U + \sqrt{U^2 + 16t^2}\right)$$
$$\lambda_6 = \frac{1}{2}\left(U - \sqrt{U^2 + 16t^2}\right)$$

In case of the conventional chemical system, i.e. U¿0 and t¡0, λ_6 represents the groundstate energy. Even though this equation of the groundstate energy does not represent the PES of the whole parameter space we can already note some facts of importance.

- The t parameter will be symmetric due too the quadratic term. We expect the t parameter to contribute to the stabilization of the system whether the value is positive or negative.
- The U parameter is not symmetric due to the first order term and the second order term. Negative U values will stabilize the system and positive values destabilizes the system
- In case of t=0 the PES surface of the homonuclear system will be a discontinuous function of U. When $U < 0$ the energy will decrease linearly while the energy becomes zero for $U > 0$

These characteristics of the PES will serve as decent criteria in the visualization of the PES.

4.1 Training data

For the analysis of the two site Hubbard model we will use the upper triangle of adjacency matrix as was defined in the previous section. The sampling will be done at random for all the parameters involved. For the two types of parameters we will both use a continuous uniform distribution albeit defined in a different sampling interval.

$$\text{U: } \mathcal{U}(\text{-3,3})$$
$$\text{t: } \mathcal{U}(\text{-10,10})$$

The uniform distribution is employed here to provide a unbiased and equal sampling of the complete parameter space of interest. In total we will include 5.10^5 data points where each one is defined by the aforementioned distributions.

4.2 Test data

Due too the limited size of the parameter space of the two site model it is relatively easily to visualize the potential energy surface of the two site model as a 3D plot with iso surfaces corresponding to the energy.

For this means we construct a parameter grid that samples the parameter space in a constant fashion. Each parameter is partitioned into sections with step size 0.1 and thus consists of a grid with dimension $200 \times 60 \times 60$ or a total of 720000 points.

4.3 Neural networks

Due too the limited complexity of the system of interest we can utilize the two site Hubbard model to test the properties of the neural networks themselves. From the theory of machine learning we know that these networks are dependent on a large variety of hyper parameters. The following characteristics will influence the overall performance of the neural network

- The amount of layers present in the network, which determines the amount of non linear transformations the original data undergoes
- The amount of nodes per layer present in the network, the amount of different combinations that will undergo the non linear transformation
- The activation function, which determines the type of non linear transformation that is performed throughout the network
- The random seed, during the initialization phase of the neural network the algorithm assigns random weights to all the nodes present in the network. Due too the nature of the back propagation algorithm we expect that the initial value of these weight will have some influence on the minimum this network will converge too.
- The optimization algorithm which determines the method through which the network converges
- The batch size, determining the amount of training data that is used in each back propagation cycle
- The size of the validation data set
- The amount of epochs will determine the final point too which the network has converged

All of these factors will eventually contribute too the overall performance of the network as they determine where the eventual minimum in the error surface is achieved. Due too the complexity of this hyper parameter space and the limited amount of time we are unable to perform a complete scan of this hyper parameter space. We will however focus on a couple of these to perform a superficial search as we expect that some will have a larger impact than others.

The hyper parameters of interest in this research will be the following

- The activation function, we will perform a comparison between the different activation functions of the ReLU type of functions. These functions each introduce a extra degree of freedom which the network can utilize to perform a better training on the data
- The amount of layers in the network, an essential quantifier in the complexity of the neural network
- The amount of nodes per layer in the network, a supplemental qualifier for network complexity
- The seed of the random initialization, due too the decision of too train the networks using a GPU accelareted algorithm we expect some fluctuations on the network performance. It is therefore essential to get a idea too which degree we can expect thus too happen

As such we will train networks with the following combinations of hyper parameters

- A variation of one layer to a total of ten layers
- Three nodes per layer (the amount of input variables), six nodes per layer and nine nodes per layer
- Three different activation functions being the ReLU, LReLU and PReLU
- The seed of initialization for the networks will be done for a random seed of one and a random seed of two

In addition to this we will also provide a superficial assessment of the reproducibility of these networks by selecting the optimal network from all these models and perform the same optimization with different seeds. Followed by a comparison of the predicted values of these networks to the exact values.

4.3.1 General assessment of the hyper parameter sweep

Network performace for the hyper parameter sweep at random seed 1 The table belows details the mean squared error of the neural networks trained using different architectures at the random seed of 1. As desriptors of these networks of these networks we have tabulated the training, validation, training set and test error. The values are displayed using a heatmap with a color range between $[0, 0.02]$ as the majority of the networks that show a good performance are centered between these two values. Blue cells are indicative of errors that lie near the lower bound of this interval, and the red those that are near the upper bound (or above). The white cells are those that have a intermediate between these two values.

A first observation we can make is that the networks that are trained using the ReLU activation function tend to display a large error for networks defined by having a rather minimalistic architecture, i.e. ten nodes per layer. This behavior is however not visible for the networks that utilize the LReLU and PReLU activation function. For these two activation functions we see that the performance in these setups does not display the overal best performance but they do have a mean squared error below one (and some even displaying performance within the range of the more complex networks). Meaning that even at a less complex architectures these two activation functions create the possibility to create a network that is significant in the analysis of the data(?)

Another general observation that can be made is that increasing the network complexity through increasing the amount of nodes present in the layers has a beneficial effect. Indeed we see a general decrease in mean square error when increasing the amount of nodes per layer and that this effect is at its largest when going from three nodes per layer too six nodes per layer. This transition generally creates a decrease in the mean square error by a factor ten too hundred. The further increase in error occuring at the transition form sic to nine nodes per layer however is not off this magnitude and generally is only by a factor two (in some cases less and in other more).

What is remarkable however is that although the increase of nodes consistently provides improvement upon the network performance we do see that this is not necessarily the case for the networks. Indeed for neither of the three different activation functions do we see that network of the highest complexity offers the best performance. The increase of amount of layers has no apparent detrimental effect but they do not increase the performance. This might be indicative that for the given problem there might be a limit too the network complexity that offers improvement of the performance.

Despite this fact we do see however that the best performing network is one utilizing the ReLU activation function. Indeed looking at the mean squared error for the network defined by six layers and nine nodes per layer we see that the architecture utilizing the ReLU activation function out performs the LReLU and PReLU with the same architecture. Indeed when comparing too the best performing LReLU network (six layers with nine nodes per layer) and the best PReLU network (seven layers and nine nodes per layer) we see that this ReLU network that has a mean squared error that is two to three times lower than that of said networks.

Network performance for the hyper parameter sweep at random seed 2 A similar table was constructed for the hyper parameter sweep performed at a random seed of 2. When looking at the distribution through the color coding we can already observe a significant difference with the parameter sweep performed at random seed 1. Indeed when looking at the networks trained with the PReLU activation function we see that the networks of the simplest complexity display large errors compared to the other networks. These large error are also present at the networks trained with the ReLU activation function but not at the networks trained with the LReLU activation function

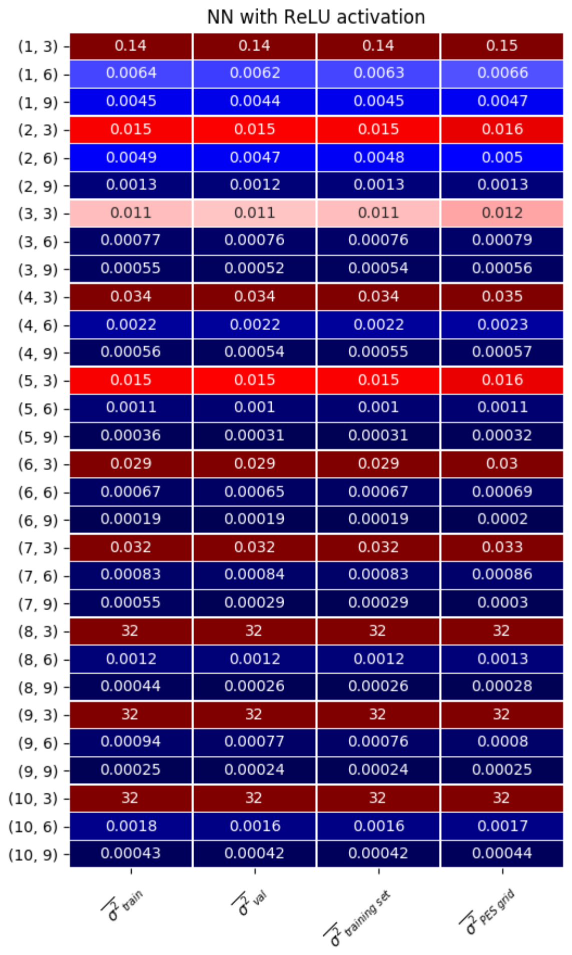
We do notice that in general the errors for the networks having only three nodes per layer tend to have a more volatile error when the seed is changed. This is the most apparent in the case of the ReLU and PReLU networks but also in the LReLU networks. Indeed, even though these networks do not display these large error spikes for neither seeds in case of the LReLU activation function we do note that they are the ones that shows the largest oscillation in there error. The other networks consisting of more nodes per layer also show variations on their mean squared error but these variations tend to be within the same magnitude across the different seeds.

As was the case when the random seed was set to one, the overall best performing network is a network using the ReLU activation function. Indeed looking at the network with nine layers and nine nodes per layer we see that this network provides the overall best performance (of both seeds that is). This is followed by the LReLU network consisting of eight layers and nine nodes per layer and last but not least the PReLU network of four layers and nine nodes per layer. This last result is interesting as this is one of the simpler networks and in comparison too the other optimal networks this one has a significantly lower amount of layers.

Overall conclusion of the hyper parameter sweep Comparing the same hyper parameter sweep over the two different seeds has illustrated quiet a few interesting things.

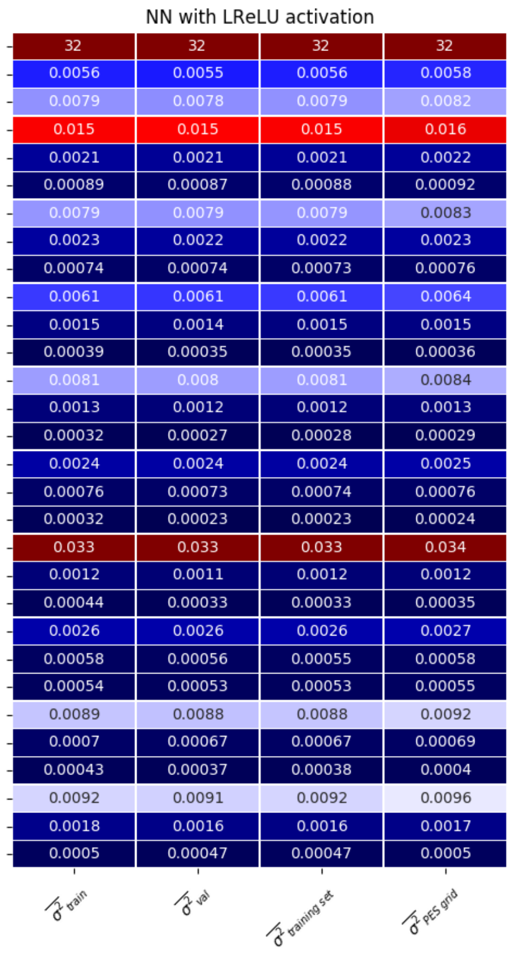
- Though the ReLU activation function is the simplest of the activation functions, it appears to have the general best performance. This is however accompanied by a rather volatile behavior of some of the networks when it comes to assessment of the errors.
- Even though the LReLU activation function does not show the same performance as that of the ReLU function, it's error are very close to that of the best ReLU networks. In addition these networks have (so far) displayed a more robustness to them when it concerns networks of lesser complexity
- The PReLU networks have shown intermediare behavior between the two afore mentioned activation functions.
- In the case of these two hyper parameter sweeps it appears that a network of larger complexity in terms of amount of layers present is not necessarily a guarantee for better performance
- Concerning the amount of nodes per layer we see that even though three amount of nodes is in some way enough to capture the features of the data, we see that this is also accompanied by a certain degree of unreliability. Increasing the amount of nodes appears to be beneficial

Due too its general performance we shall continue with the study of the LReLU functions as robustness is a desired feat in these networks.



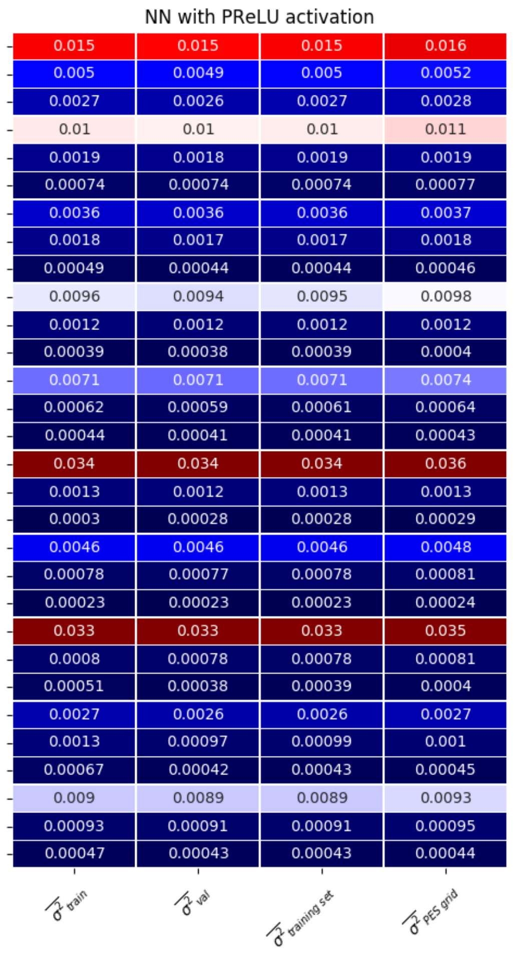
NN with ReLU activation

	$\overline{\sigma^2}_{\text{train}}$	$\overline{\sigma^2}_{\text{val}}$	$\overline{\sigma^2}_{\text{training set}}$	$\overline{\sigma^2}_{\text{PES grid}}$
(1, 3)	0.14	0.14	0.14	0.15
(1, 6)	0.0064	0.0062	0.0063	0.0066
(1, 9)	0.0045	0.0044	0.0045	0.0047
(2, 3)	0.015	0.015	0.015	0.016
(2, 6)	0.0049	0.0047	0.0048	0.005
(2, 9)	0.0013	0.0012	0.0013	0.0013
(3, 3)	0.011	0.011	0.011	0.012
(3, 6)	0.00077	0.00076	0.00076	0.00079
(3, 9)	0.00055	0.00052	0.00054	0.00056
(4, 3)	0.034	0.034	0.034	0.035
(4, 6)	0.0022	0.0022	0.0022	0.0023
(4, 9)	0.00056	0.00054	0.00055	0.00057
(5, 3)	0.015	0.015	0.015	0.016
(5, 6)	0.0011	0.001	0.001	0.0011
(5, 9)	0.00036	0.00031	0.00031	0.00032
(6, 3)	0.029	0.029	0.029	0.03
(6, 6)	0.00067	0.00065	0.00067	0.00069
(6, 9)	0.00019	0.00019	0.00019	0.0002
(7, 3)	0.032	0.032	0.032	0.033
(7, 6)	0.00083	0.00084	0.00083	0.00086
(7, 9)	0.00055	0.00029	0.00029	0.0003
(8, 3)	32	32	32	32
(8, 6)	0.0012	0.0012	0.0012	0.0013
(8, 9)	0.00044	0.00026	0.00026	0.00028
(9, 3)	32	32	32	32
(9, 6)	0.00094	0.00077	0.00076	0.0008
(9, 9)	0.00025	0.00024	0.00024	0.00025
(10, 3)	32	32	32	32
(10, 6)	0.0018	0.0016	0.0016	0.0017
(10, 9)	0.00043	0.00042	0.00042	0.00044



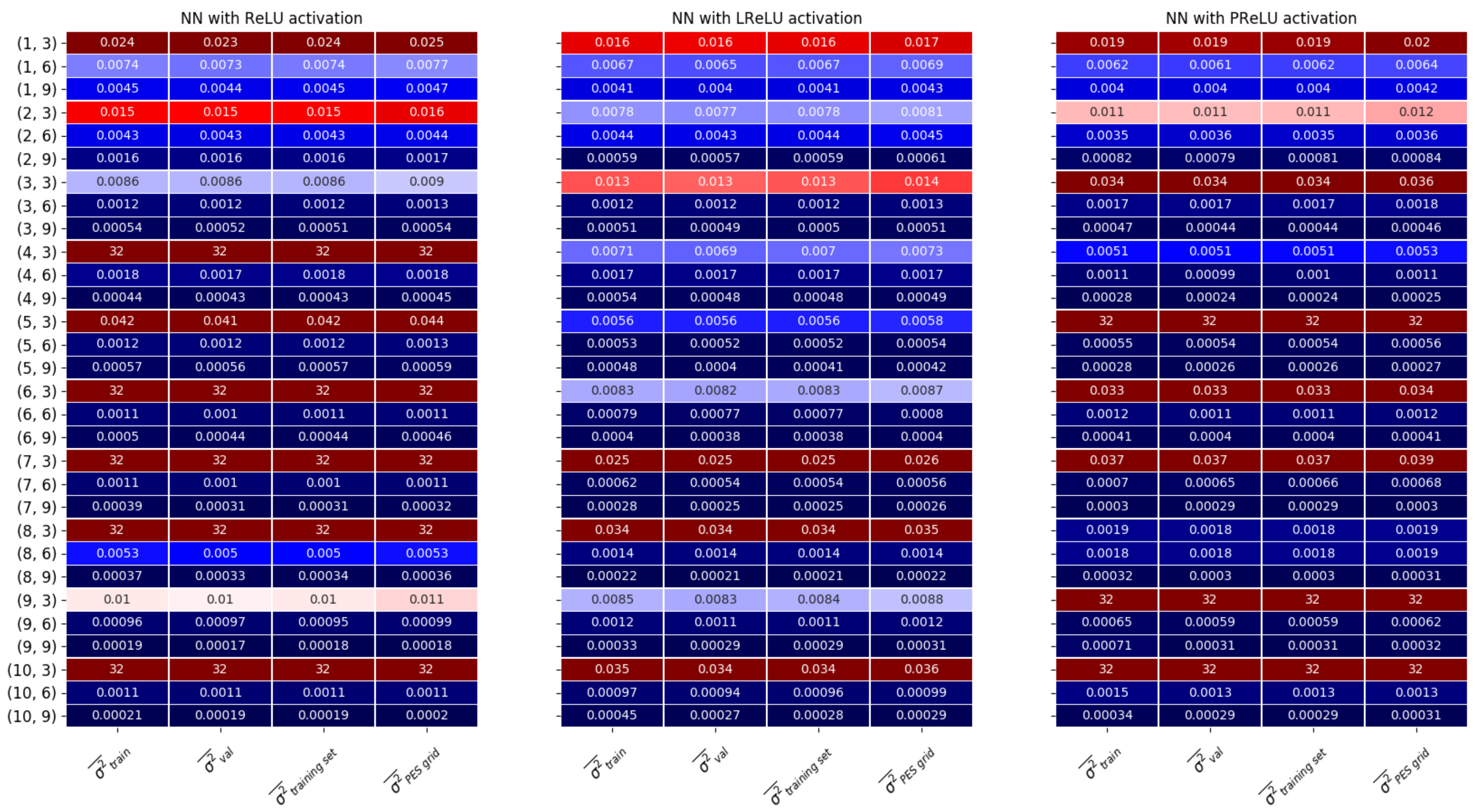
NN with LReLU activation

	$\overline{\sigma^2}_{\text{train}}$	$\overline{\sigma^2}_{\text{val}}$	$\overline{\sigma^2}_{\text{training set}}$	$\overline{\sigma^2}_{\text{PES grid}}$
(1, 3)	32	32	32	32
(1, 6)	0.0056	0.0055	0.0056	0.0058
(1, 9)	0.0079	0.0078	0.0079	0.0082
(2, 3)	0.015	0.015	0.015	0.016
(2, 6)	0.0021	0.0021	0.0021	0.0022
(2, 9)	0.00089	0.00087	0.00088	0.00092
(3, 3)	0.0079	0.0079	0.0079	0.0083
(3, 6)	0.0023	0.0022	0.0022	0.0023
(3, 9)	0.00074	0.00074	0.00073	0.00076
(4, 3)	0.0061	0.0061	0.0061	0.0064
(4, 6)	0.0015	0.0014	0.0015	0.0015
(4, 9)	0.00039	0.00035	0.00035	0.00036
(5, 3)	0.0081	0.008	0.0081	0.0084
(5, 6)	0.0013	0.0012	0.0012	0.0013
(5, 9)	0.00032	0.00027	0.00028	0.00029
(6, 3)	0.0024	0.0024	0.0024	0.0025
(6, 6)	0.00076	0.00073	0.00074	0.00076
(6, 9)	0.00032	0.00023	0.00023	0.00024
(7, 3)	0.033	0.033	0.033	0.034
(7, 6)	0.0012	0.0011	0.0012	0.0012
(7, 9)	0.00044	0.00033	0.00033	0.00035
(8, 3)	0.0026	0.0026	0.0026	0.0027
(8, 6)	0.00058	0.00056	0.00055	0.00058
(8, 9)	0.00054	0.00053	0.00053	0.00055
(9, 3)	0.0089	0.0088	0.0088	0.0092
(9, 6)	0.0007	0.00067	0.00067	0.00069
(9, 9)	0.00043	0.00037	0.00038	0.0004
(10, 3)	0.0092	0.0091	0.0092	0.0096
(10, 6)	0.0018	0.0016	0.0016	0.0017
(10, 9)	0.0005	0.00047	0.00047	0.0005



NN with PReLU activation

	$\overline{\sigma^2}_{\text{train}}$	$\overline{\sigma^2}_{\text{val}}$	$\overline{\sigma^2}_{\text{training set}}$	$\overline{\sigma^2}_{\text{PES grid}}$
(1, 3)	0.015	0.015	0.015	0.016
(1, 6)	0.005	0.0049	0.005	0.0052
(1, 9)	0.0027	0.0026	0.0027	0.0028
(2, 3)	0.01	0.01	0.01	0.011
(2, 6)	0.0019	0.0018	0.0019	0.0019
(2, 9)	0.00074	0.00074	0.00074	0.00077
(3, 3)	0.0036	0.0036	0.0036	0.0037
(3, 6)	0.0018	0.0017	0.0017	0.0018
(3, 9)	0.00049	0.00044	0.00044	0.00046
(4, 3)	0.0096	0.0094	0.0095	0.0098
(4, 6)	0.0012	0.0012	0.0012	0.0012
(4, 9)	0.00039	0.00038	0.00039	0.0004
(5, 3)	0.0071	0.0071	0.0071	0.0074
(5, 6)	0.00062	0.00059	0.00061	0.00064
(5, 9)	0.00044	0.00041	0.00041	0.00043
(6, 3)	0.034	0.034	0.034	0.036
(6, 6)	0.0013	0.0012	0.0013	0.0013
(6, 9)	0.0003	0.00028	0.00028	0.00029
(7, 3)	0.0046	0.0046	0.0046	0.0048
(7, 6)	0.00078	0.00077	0.00078	0.00081
(7, 9)	0.00023	0.00023	0.00023	0.00024
(8, 3)	0.033	0.033	0.033	0.035
(8, 6)	0.0008	0.00078	0.00078	0.00081
(8, 9)	0.00051	0.00038	0.00039	0.0004
(9, 3)	0.0027	0.0026	0.0026	0.0027
(9, 6)	0.0013	0.00097	0.00099	0.001
(9, 9)	0.00067	0.00042	0.00043	0.00045
(10, 3)	0.009	0.0089	0.0089	0.0093
(10, 6)	0.00093	0.00091	0.00091	0.00095
(10, 9)	0.00047	0.00043	0.00043	0.00044



	NN with ReLU activation				NN with LReLU activation				NN with PReLU activation			
	$\overline{\sigma^2}_{train}$	$\overline{\sigma^2}_{val}$	$\overline{\sigma^2}_{training\ set}$	$\overline{\sigma^2}_{PES\ grid}$	$\overline{\sigma^2}_{train}$	$\overline{\sigma^2}_{val}$	$\overline{\sigma^2}_{training\ set}$	$\overline{\sigma^2}_{PES\ grid}$	$\overline{\sigma^2}_{train}$	$\overline{\sigma^2}_{val}$	$\overline{\sigma^2}_{training\ set}$	$\overline{\sigma^2}_{PES\ grid}$
(1, 3)	0.024	0.023	0.024	0.025	0.016	0.016	0.016	0.017	0.019	0.019	0.019	0.02
(1, 6)	0.0074	0.0073	0.0074	0.0077	0.0067	0.0065	0.0067	0.0069	0.0062	0.0061	0.0062	0.0064
(1, 9)	0.0045	0.0044	0.0045	0.0047	0.0041	0.004	0.0041	0.0043	0.004	0.004	0.004	0.0042
(2, 3)	0.015	0.015	0.015	0.016	0.0078	0.0077	0.0078	0.0081	0.011	0.011	0.011	0.012
(2, 6)	0.0043	0.0043	0.0043	0.0044	0.0044	0.0043	0.0044	0.0045	0.0035	0.0036	0.0035	0.0036
(2, 9)	0.0016	0.0016	0.0016	0.0017	0.00059	0.00057	0.00059	0.00061	0.00082	0.00079	0.00081	0.00084
(3, 3)	0.0086	0.0086	0.0086	0.009	0.013	0.013	0.013	0.014	0.034	0.034	0.034	0.036
(3, 6)	0.0012	0.0012	0.0012	0.0013	0.0012	0.0012	0.0012	0.0013	0.0017	0.0017	0.0017	0.0018
(3, 9)	0.00054	0.00052	0.00051	0.00054	0.00051	0.00049	0.0005	0.00051	0.00047	0.00044	0.00044	0.00046
(4, 3)	32	32	32	32	0.0071	0.0069	0.007	0.0073	0.0051	0.0051	0.0051	0.0053
(4, 6)	0.0018	0.0017	0.0018	0.0018	0.0017	0.0017	0.0017	0.0017	0.0011	0.00099	0.001	0.0011
(4, 9)	0.00044	0.00043	0.00043	0.00045	0.00054	0.00048	0.00048	0.00049	0.00028	0.00024	0.00024	0.00025
(5, 3)	0.042	0.041	0.042	0.044	0.0056	0.0056	0.0056	0.0058	32	32	32	32
(5, 6)	0.0012	0.0012	0.0012	0.0013	0.00053	0.00052	0.00052	0.00054	0.00055	0.00054	0.00054	0.00056
(5, 9)	0.00057	0.00056	0.00057	0.00059	0.00048	0.0004	0.00041	0.00042	0.00028	0.00026	0.00026	0.00027
(6, 3)	32	32	32	32	0.0083	0.0082	0.0083	0.0087	0.033	0.033	0.033	0.034
(6, 6)	0.0011	0.001	0.0011	0.0011	0.00079	0.00077	0.00077	0.0008	0.0012	0.0011	0.0011	0.0012
(6, 9)	0.0005	0.00044	0.00044	0.00046	0.0004	0.00038	0.00038	0.0004	0.00041	0.0004	0.0004	0.00041
(7, 3)	32	32	32	32	0.025	0.025	0.025	0.026	0.037	0.037	0.037	0.039
(7, 6)	0.0011	0.001	0.001	0.0011	0.00062	0.00054	0.00054	0.00056	0.0007	0.00065	0.00066	0.00068
(7, 9)	0.00039	0.00031	0.00031	0.00032	0.00028	0.00025	0.00025	0.00026	0.0003	0.00029	0.00029	0.0003
(8, 3)	32	32	32	32	0.034	0.034	0.034	0.035	0.0019	0.0018	0.0018	0.0019
(8, 6)	0.0053	0.005	0.005	0.0053	0.0014	0.0014	0.0014	0.0014	0.0018	0.0018	0.0018	0.0019
(8, 9)	0.00037	0.00033	0.00034	0.00036	0.00022	0.00021	0.00021	0.00022	0.00032	0.0003	0.0003	0.00031
(9, 3)	0.01	0.01	0.01	0.011	0.0085	0.0083	0.0084	0.0088	32	32	32	32
(9, 6)	0.00096	0.00097	0.00095	0.00099	0.0012	0.0011	0.0011	0.0012	0.00065	0.00059	0.00059	0.00062
(9, 9)	0.00019	0.00017	0.00018	0.00018	0.00033	0.00029	0.00029	0.00031	0.00071	0.00031	0.00031	0.00032
(10, 3)	32	32	32	32	0.035	0.034	0.034	0.036	32	32	32	32
(10, 6)	0.0011	0.0011	0.0011	0.0011	0.00097	0.00094	0.00096	0.00099	0.0015	0.0013	0.0013	0.0013
(10, 9)	0.00021	0.00019	0.00019	0.0002	0.00045	0.00027	0.00028	0.00029	0.00034	0.00029	0.00029	0.00031

4.3.2 Reliability and reproducability of the networks

Many interesting features were discovered solely by looking at the mean squared error of the different architectures of the networks at different random initialization seeds. We have noticed however that the network that displays the overall best performance is not a easily predictable feat and that variation on the errors does occur across the different architectures with different activation functions and different random seeds.

As such we wish to focus more on the reproducibility and reliability of these networks. Though the errors for the different networks with nine nodes per layer all lie within the same interval of $[0.001, 0.0001]$ it is desirable to achieve an error that is more or less the same across the different seeds. To this mean we shall aim our focus in this section more towards the variation of this error.

Looking at the errors for the LReLU network we note that the network consisting of nine layers and nine nodes per laye in both have errors that are relatively close to one and another across the different seeds. As such we shall perform the following tests to provide a general assessment of the stability of the network

- Starting from the same seed we shall perform the network training multiple times under this condition. The assessment of the stability of the network in question shall be done by comparing the mean squared error of the prediction of these networks on the test data. This provides a reference too what degree we can expect the network too be stable.
- Starting from a number of different seeds we shall perform the network training using the same condition. As in the previous suggested test, comparison of the mean squared error shall be done between the predictions on the testing data. This provides a reference too the degree of stability between different seeds

As a reference we shall also perform the same inter and intra seed comparison for a model with less complexity. For this purpose we shall use a neural network with nine layers and three nodes per layer.

We shall define a network that is stable as one that converges too a same solution when training is repeated different times or using a different seed. Too asses whether a network has converged too the same solution we propose the following criteria

1. If the networks converge to the same minimum across different training cycles than the training loss and the validation loss should converge too the same minimum
2. If the networks converge to the same minimum then the difference between the prediction surfaces should be relatively equivalent too each other. Too asses this we shall use a error matrix Δ in which the elements are calculated as follows for

$$\Delta_{ij} = \begin{cases} \frac{1}{N}\sum_{n=1}^{N}\left(t_i - y_i(x_n)\right)^2 & \text{if } i = j \\ \\ \frac{1}{N}\sum_{n=1}^{N}\left(y_i(x_n) - y_j(x_n)\right)^2 & \text{if } i \neq j \end{cases}$$

where t_n is the target vector of input vector x_n and y_i is the network function of network i. The diagonal will contain the mean square error of the network compared too the exact solution and will provide a reference to compare the differences between the networks

Discussion of the nine layer and three nodes per layer network When looking at the loss evolution of the nine layer and three nodes per layer network we already notice that this graph consists of very irregularly shaped curves. Though the training loss an the validation loss evolve in the same fashion for the different training cycles we do note that they differ significantly from each other. Indeed in both cases, i.e. training performed on the same seed multiple times and training on different seeds, we see that none of the curves evolve too a common minimal loss after extensive training. This is already indicative that these networks, though they share a similar architecture, converge too different minima on the error surface.

Our suspicion that they do not describe the same minimum is also confirmed when looking at the error matrices of both the inter seed and intra seed comparison. The diagonal of both cases display a relative large variability when it comes too the mean squared error compared too the exact PES surface. Though we do expect some variability in this error as even in the best case scenario we expect that the networks will not land at exactly at the same location on the error surface should they converge too the same minimum, the fact that some errors differ by the a factor ten is troubling too say the least.

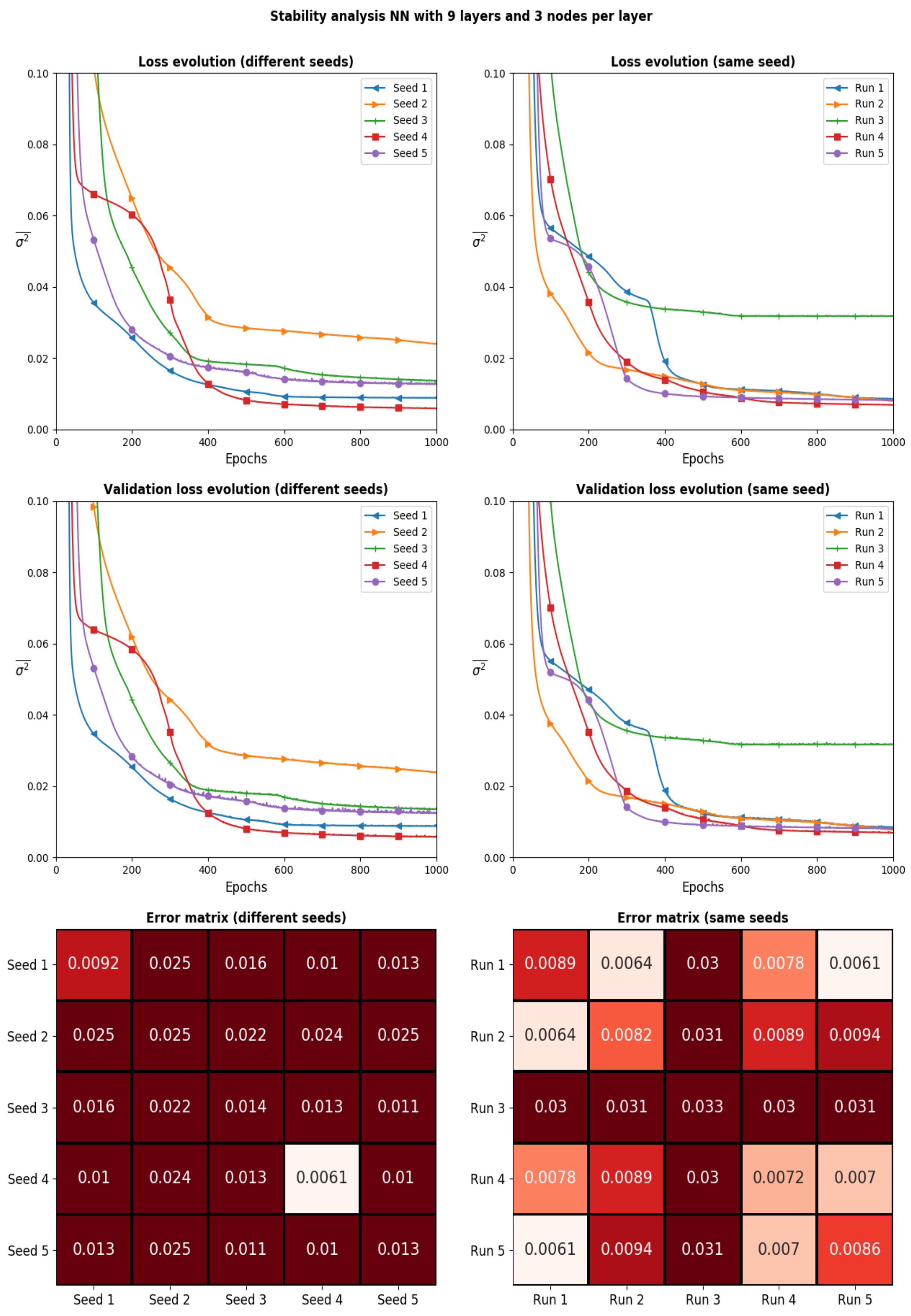
When looking at the off diagonal elements we see that this error between the different surfaces also shows inconsistencies in comparison to each other. This error between the prediction surfaces of networks trained on different seeds shows some form of consistency as they are all centered around a similar value. However when looking at the error of networks trained on the same seed this consistencies is no longer present. Indeed, looking at the different network comparisons we see that some tend to be very similar to each other there is one (run 3) that appears to be very different compared to the other networks.

Discussion of the nine layer and nine nodes per layer network The difference between the stability analysis of the three nodes per layers network and the nine nodes per layer network is quiet significant. Indeed when looking at both the training loss evolution and validation loss evolution of the different training cycles we already notice that independent of the seed or the run of the same seed that the models appear to be converging to the same minimum in loss. This is already a positive indicator as this increases the chance that the networks all converged to the same minimum on the error surface. We do take note that at the lowest value of the loss function there also appears to be some oscillatory behavior which is probably indicative that the network has converged to a minimum but has trouble reaching the actual exact minimum. From this observation we can expect that the networks will not be exactly the same but will display minor differences when comparing their prediction surfaces.

Looking at the diagonal of the error matrix we note that all the mean squared errors with respect to the exact PES lie in the same magnitude of error, i.e. 10^{-4}, and appear to be centered in the interval of $[0.0002, 0.0005]$. The off diagonal elements also lie within this magnitude of 10^{-4} but tend to be larger than the diagonal elements (roughly 2 times in many cases).

Conclusion As is visible in the comparison between the two networks we take note that the network that are less complex tend to have a more volatile error surface. The difference between seeds and within the same seed are not insignificant and thus give rise too the idea that either these networks converge to a different minimum on the error surface or they are in the same region of the error surface but this error surface has "superficial" minima and causes the algorithm to not converge to a specific minimum.

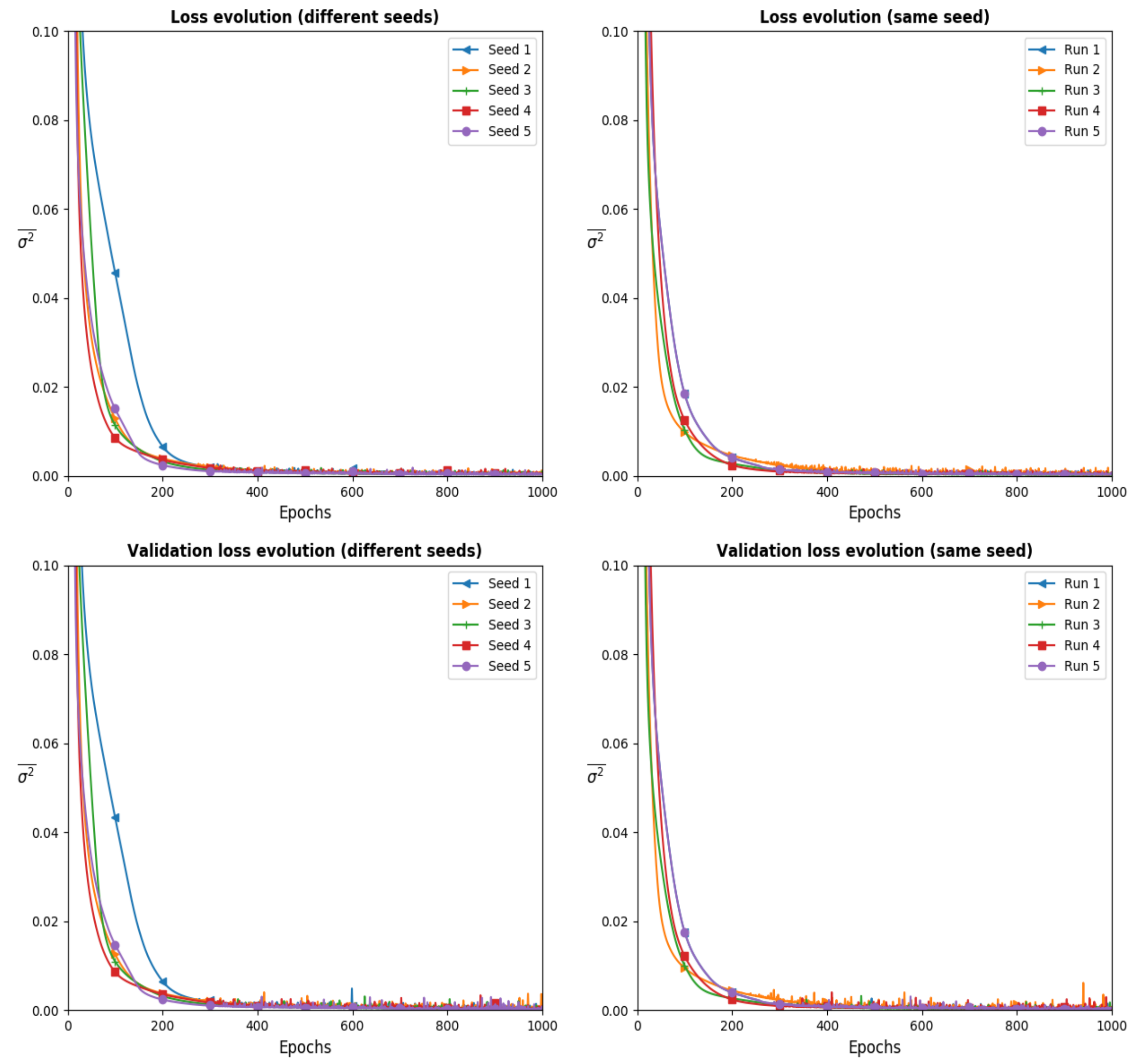
Starting from a certain level of network complexity we see that these networks converge to a stable solution. Though we cannot asses directly if the minima they converge too are the same, we do note that they minima they converge too are relatively similar too each other. Thus based on these criteria we can conclude that a certain level of reliability and reproducibility can be obtained from these networks but only on the condition that they have a certain level of network complexity

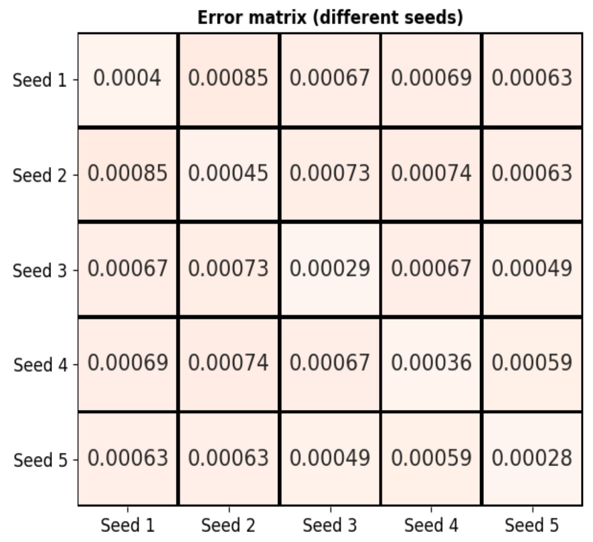


	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
Seed 1	0.0092	0.025	0.016	0.01	0.013
Seed 2	0.025	0.025	0.022	0.024	0.025
Seed 3	0.016	0.022	0.014	0.013	0.011
Seed 4	0.01	0.024	0.013	0.0061	0.01
Seed 5	0.013	0.025	0.011	0.01	0.013

	Run 1	Run 2	Run 3	Run 4	Run 5
Run 1	0.0089	0.0064	0.03	0.0078	0.0061
Run 2	0.0064	0.0082	0.031	0.0089	0.0094
Run 3	0.03	0.031	0.033	0.03	0.031
Run 4	0.0078	0.0089	0.03	0.0072	0.007
Run 5	0.0061	0.0094	0.031	0.007	0.0086

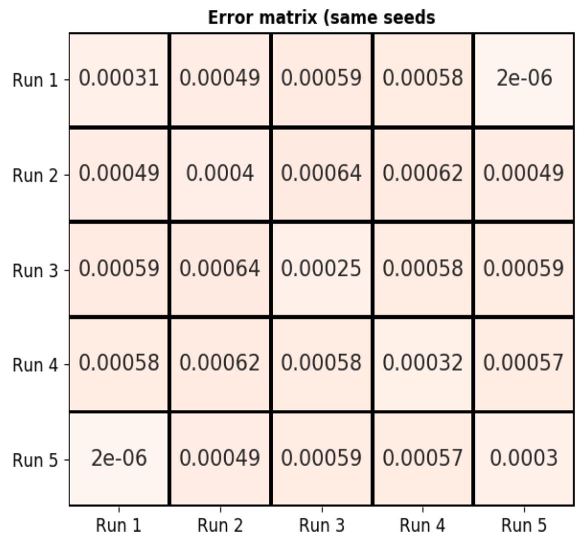
Stability analysis NN with 9 layers and 9 nodes per layer





Error matrix (different seeds)

	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
Seed 1	0.0004	0.00085	0.00067	0.00069	0.00063
Seed 2	0.00085	0.00045	0.00073	0.00074	0.00063
Seed 3	0.00067	0.00073	0.00029	0.00067	0.00049
Seed 4	0.00069	0.00074	0.00067	0.00036	0.00059
Seed 5	0.00063	0.00063	0.00049	0.00059	0.00028



Error matrix (same seeds

	Run 1	Run 2	Run 3	Run 4	Run 5
Run 1	0.00031	0.00049	0.00059	0.00058	2e-06
Run 2	0.00049	0.0004	0.00064	0.00062	0.00049
Run 3	0.00059	0.00064	0.00025	0.00058	0.00059
Run 4	0.00058	0.00062	0.00058	0.00032	0.00057
Run 5	2e-06	0.00049	0.00059	0.00057	0.0003

5 Four site Hubbard model

In this section we will study the effect of increasing the Hubbard system upon network performance and predictive capabilities. In the previous section we have seen that neural networks with a relatively simple architecture i.e. the amount of nodes per layer equal to the input dimension size, was technically already enough to provide a correct representation of the potential energy surface of the two site Hubbard model.

However one has to consider that in terms of chemical complexity the two site model represents the base of complexity possible in the Hubbard model (without taking isolated atoms into consideration of course). For the two site structure we can only come up with one structural arrangement i.e. a connection between two adjacent sites. This situation is completely different when looking at the four site model. For the four site Hubbard model we can look at twenty six different structural arrangements with a different amount of total bonds (the four site model has a minimum of 3 bonds and a maximum of 6 bonds). **Figuur van de verschillende structurele mogelijkheden van de 4 site modellen**

This structural diversity of the foursite Hubbard model also implies that the chemistry behind these models will be more complex. Indeed when one looks at the analytical solutions of the two site Hubbard model characterized by the upper triangle of the adjancency matrix

$$[U_1, t_{12}, U_2]$$

we get a polynomial that is only dependent upon three different parameters. This leads to a PES that is limited in its complexity and that is easily visualized (as was demonstrated in the previous section). The four site Hubbard model however is dependent upon the following general adjacency matrix upper triangle

$$[U_1, t_{12}, t_{13}, t_{14}, U_2, t_{23}, t_{24}, U_3, t_{34}, U_4]$$

which will results in a ground state energy which is dependent on ten different parameters. Not only is the size of the parameter space much larger than that of the two site model, the possibility for different parameter interactions also increases which will give rise to different unique chemical effects.

The main problem which will present itself in the evaluation of the neural networks is the evaluation of the performance of said networks. In the case of the two site model we had a ground state energy which is dependent on only three parameters and thus lends itself to being visualized quiet easily. The ten dimensional space of the PES of the four site Hubbard model ground state on the other hand is not easily assesed through visual means.

5.1 Training data

5.1.1 Randomized dataset

Similary to the two site model we will construct dense neural networks which perform a non linear mapping between the upper triangle of the adjacency matrix to the groundstate energy. In the case of the foursite model we have four different U parameters and six different t parameters each chosen from a continuous random uniform distribution. The training dataset is constructed according to the following distributions

$$U : \mathcal{U}(-3, 3)$$

$$t : \mathcal{U}(-10, 10)$$

This is done to assure an even sampling of the complete parameter space and train the network without any inherent chemical knowledge embedded in the training dataset. In total 10^6 point are used during the training procedure of which 5 percent is used to validate the networks evolution across the error surface.

5.1.2 Data augmentation

Due too the nature of the four site Hubbard model we also have the oppertunity to asses the networks capability to incorporate the inherent symmetry of the Hamiltonian operator. Indeed, consider a symmetry operator $\hat{\mathcal{R}}$ and the Hamiltonian operator $\hat{H}$ it can be shown that these two operators commute with each other

$$[\hat{\mathcal{R}}, \hat{H}] = 0$$

Thus implying that they share eigenvectors, meaning that the energy is unchanged before and after a symmetry operations that does not change the molecule itself. Using this fundamental property we can already asses a part of the networks performance and robustness. Consider the general input vector of the neural network

$$\begin{bmatrix} U_1 & t_{12} & t_{13} & t_{14} \\ t_{21} & U_2 & t_{23} & t_{24} \\ t_{31} & t_{32} & U_3 & t_{34} \\ t_{41} & t_{42} & t_{43} & U_4 \end{bmatrix}$$

It is easily possible to perform symmetry operations upon this adjacency matrix by performing an equal permutation of the rows and columns. Let (1,2,3,4) be the indices for the rows R and the columns C then any permutation of these indices will result in a equivalent structure. For a system consisting of N sites we have the possibility of augmenting the dataset with N! new points that don't need to be computed with a Hubbard solver to participate in the training procedure.

Starting from the unaugmented dataset we will construct two additional datasets each augmented with a specific amount of symmetry operations. We will however keep the number of datapoints in these datasets constant relative to the original dataset. In total we will have the following datasets

1. **Unaugmented completely randomized dataset**, consisting of 10^6 upper triangle adjacency matrix where the parameters are defined according to the aforementioned continuous uniform distributions
2. **N symmetry augmented dataset**, starting from the randomized dataset we choose a one fourth of the original points and perform four permutation operations upon each of these points. The dataset thus consists of $2,5.10^5$ unique datapoints and $7,5.10^5$ symmetry equivalent datapoints
3. **N! symmetry augmented dataset**, starting from the randomized dataset we choose a one twenty-fourth of the original points and perform twenty-four permutation operations upon each of these points. The dataset thus consists of $4,2.10^4$ unique datapoints and $9,66.10^5$ symmetry equivalent datapoints

Each of these datasets will be trained in the same fashion and will be evaluated according to the same test set, this will hopefully give a idea on two different factors

- What is the effect of data augmentation upon the model performance? Is the introduction of this physical knowledge necessary to obtain a robust model?
- Is there a minimal limit to the amount of points necessary to obtain a overall decetn performance. As mentioned in the summary above the augmented data contains less "unique" data in comparisson to the unaugmented data. One could argue that in order to provide a general description an increasing amount of unique data will be required to capture all the important information present in the parameter space
- Can the property of inherent symmetry of the Hamiltonian emerge through the analysis of completely random and assymetric data or does it need to be present in the training data it self?

5.2 Test dataset

5.2.1 Test set based upon relevant chemical structures

As was mentioned before the assessment of the performance of the networks trained on the four site Hubbard model will prove to be a greater challenge than that of the two site Hubbard model. The dimension of the parameter space prohibits us from visualizing the complete PES of the ground state of the system in question and thus will require a intelligent choice to obtain a qualitative descriptor.

In contrasts to the two site model, we have trained these networks on data that in a strict sense does not correspond to systems of large chemical/physical importance. Indeed by randomizing all the parameters in the input vector we have created a set of molecular structures that corresponds to a completely distorted cyclic structure that is connected with every possible atom in its neighborhood. When we classicaly think about structures of chemical relevance we generally think about molecular structures that posses a inherent symmetry.

The first structures that come too mind when thinking about molecules that are defined by four atoms is the butadiene molecule and the cyclobutadiene molecule. Both these molecules are well understood and well studied in classical research subjects and thus will provide a adequate first descriptor. The corresponding adjacency matrix input vector will have the following shape

Butadiene as the linear Hubbard model

$$\begin{bmatrix} U_1 & t_{12} & 0 & 0 & U_2 & t_{23} & 0 & U_3 & t_{34} & U_4 \end{bmatrix}$$

Cyclobutadiene as the cyclic Hubbard model

$$\begin{bmatrix} U_1 & t_{12} & 0 & t_{14} & U_2 & t_{23} & 0 & U_3 & t_{34} & U_4 \end{bmatrix}$$

For both these molecular systems we will construct a test dataset that correspond to a homo nuclear system meaning that we will impose a symmetry in the parameter space. For the homo nuclear system this implies a universal U and a universal t across the whole system, which will be varied across in the whole training range in steps of 0,1 resulting in a test set of size 12000 data points each.

As a additional test set we will define a set which will resemble structurally on the original training input. For these data points we will also use a uniform U and t parameter and define the bond along the diagonal (opposing atoms) based on geometric means. Due too the definition of the t parameter we know that the magnitude of this hopping term scales inversely with the internuclear distance. Treating the fully connected cyclic structure as a square, we can calculate the distance between opposing atoms in terms of the vertex length easily through the Pythagorean theorem as $\sqrt{2}$. Following this reason the input vector for the fully connected four site Hubbard model will have the following input vector

$$\begin{bmatrix} U & t & \frac{1}{\sqrt{2}}t & t & U & t & 1\frac{}{\sqrt{2}}\mathrm{t} & U & t & U \end{bmatrix}$$

5.2.2 Test data based upon quantumchemical effects

Study of the potential energy surface of the homo nuclear system will already provide much insight in the working and performance of the neural networks. However we know from previous research and study that the potential energy surface also exhibits characteristic behavior that we could possibly utilize to analyze the network performance. One such effects that is well known is the Jahn-Teller effect which states the following

Jahn-Teller effect

Consider a potential energy surface of a molecular system that is degenerate, *then the system will undergo a geometrical distortion that will cause the degeneracy to be lifted. This distortion of the molecular geometry will often result in a transition from a state that is higher in symmetry to one that is lower in symmetry.*

For each of the molecular test systems defined in the previous section we can create a dataset that will test the presence (or absence) of this effect in the neural networks. As was mentioned in the theory section concerning the Hubbard model, we know that the t parameter is inversely proportional to the internuclear distance between the sites in question. Assessing the presence of stabilization through geometrical distortion would correspond in this case to increasing or decreasing the t parameter of the bond in question.

For the four site Hubbard model this Jahn-Teller distortion is easily visualized by splitting up the bonds in a symmetric fashion and plotting the PES in function of varying t pairs. For the linear four site Hubbard model we will look at the energy in function of the variation in bond distance between the outer bonds and the central bond that connects these two bonds. The dataset will be constructed in the following fashion (with U=1)

Jahn-Teller variant of the linear four site Hubbard model

$$\begin{bmatrix} U & t_1 & 0 & 0 & U & t_2 & 0 & U & t_1 & U \end{bmatrix}$$

For the cyclic four site model a similar dataset can be constructed albeit with a different choice of bond pairs. In the cyclic system and fully connected system the distortion will be performed along the ring elongating two vertices while contracting two others, i.e. breaking the C_4 symmetry (square) to a C_2 symmetry (rectangle). The bond pairs are chosen by groupnig each bond with its opposing bond. The input vectors of these structures thus will be

Jahn-Teller variant of the cyclic four site Hubbard model

$$\begin{bmatrix} U & t_1 & 0 & t_2 & U & t_2 & 0 & U & t_1 & U \end{bmatrix}$$

Jahn-Teller variant of the fully connected four site Hubbard model

$$\begin{bmatrix} U & t_1 & t_3 & t_2 & U & t_2 & t_3 & U & t_1 & U \end{bmatrix}$$

$$t_3 = \frac{1}{\sqrt{\frac{1}{t_1^2} + \frac{1}{t_2^2}}}$$

5.2.3 Test data based on structures with less complexity

In order to assess the generality of the neural network we could formulate datasets that represent molecular systems that are less complex than the original four site model. Indeed one could write any molecular system that has less sites than the original training data structure. For the four site system we can formulate three additional datasets that represent a test set for the two site

Hubbard model, the linear three site Hubbard model and the cyclic three site Hubbard model

Two site Hubbard model in the four site basis

$$\begin{bmatrix} U_1 & t_{12} & 0 & 0 & U_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Linear three site Hubbard model in the four site basis

$$\begin{bmatrix} U_1 & t_{12} & 0 & 0 & U_2 & t_{23} & 0 & U_3 & 0 & 0 \end{bmatrix}$$

Cyclic three site Hubbard model in the four site basis

$$\begin{bmatrix} U_1 & t_{12} & t_{13} & 0 & U_2 & t_{23} & 0 & U_3 & 0 & 0 \end{bmatrix}$$

Each of these input vectors are sampled at random according to the uniform distribution used to define the original training dataset.

Not only do these networks provide a assessment of the generality of the nueral network they also present an additional opportunity to test the general understanding of symmetry. Indeed, when performing the same operation as the data augmentation procedure we obtain a extended dataset that describe the same molecule in different "spatial" arrangements.

5.3 Neural networks

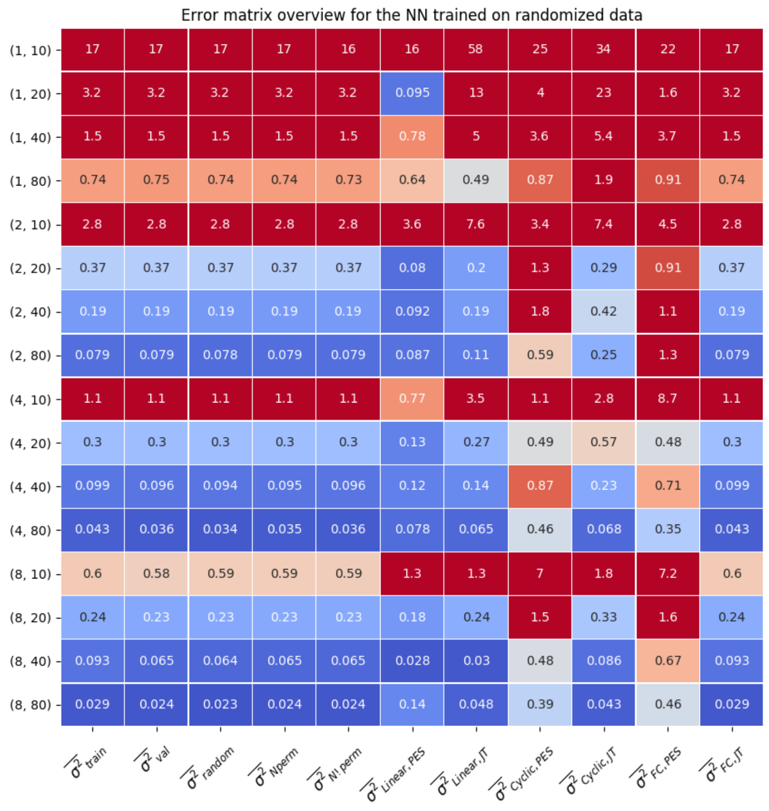
5.3.1 Architectures: General overview and general performance assesment

For the four site Hubbard model we will perform a limited parameter sweep based on the knowledge from the study of the two site Hubbard model networks. For each of the aforementioned training datasets i.e. completely randomized, N symmetry augmented and N! symmetry we will maken sixteen different networks that differ in complexity. The amount of layers will be varied from one, two, four and eight layers and the amount of nodes will be varied from ten, twenty, fourty and eighty nodes per layer. In each of these networks the different layers are densely connected, employ the LReLU activation function and the Adam optimizer. Random initialization of the network weights is controlled by the random seed of one. Each model will be trained for 1000 epochs with training done on batches of size twenty-thousand employing a a partition of 5 percent of the training set as a validation set. As a monitor for optimal network selection we will use the validation error as the criterium.

In the following figures the different mean squared errors for the relevant descriptors are summarized in a error matrix. These errors are highlighted by a divergent colourscheme centered around a mean squared error from zero to one meaning that the lower the error for the descriptor in question the bluer the cell will be in colour. The error matrix has been constructed for the networks trained on the completely random data, the N symmetry augmented and the N! symmetry augmented data and already highlights some interesting points concerning the effects of data augmentation on the networks performance.

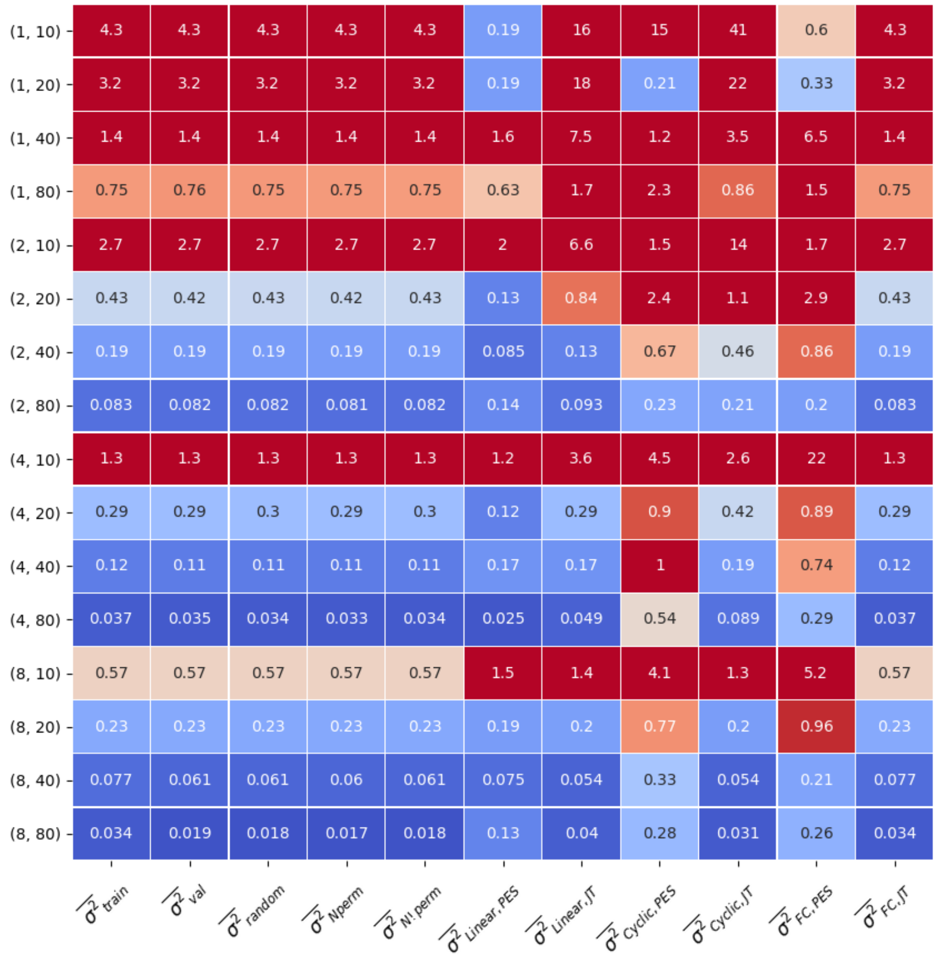
In general we already observe that the errors are more or less equally distributed for the different networks albeit with some differences. In all instances we see that the less complex networks, i.e. those defined by having one layer and/or having ten nodes per layer, are not robust enough to handle both the training data and the test data adequatly. However comparing the networks that were trained on random data alone to those that were trained on symmetry augmented data we notice that the later does display some desired characteristics. Indeed when looking at the N symmetry augmented data errors we see that the network existing of one layer with twenty nodes does seem to capture some feats of the homo nuclear PES of the linear, cyclic and fully connected test models. This might be an indicator that the introduction of data augmentation with known properties might reduce the need for complexity in the neural network. Though this can not be claimed with absolute certainty as one would expect that this effect should be amplified with addition of more nodes in the networks of simple complexity. However one also needs to take into account that although the networks were trained on an equal amount fo points, the augmented datasets both contain less unique data points compared too the random dataset.

Another clear observations is that both with the scaling of the amount of layers present in the networks and the amount of nodes that these layers contain the networks overall performance tends to increase. Indeed, comparing the values of the three setups we notice that the models defined by the presence of eighty nodes per layer show the overall best performance on both the training data and the test data. We see that going from one layer too two layers (both with eighty nodes per layer) cause a reduction in the mean squared error by roughly a factor of ten. Further increasing of the amount of layers from two to four further causes a reduction more or less by a factor two (in some cases less than a factor two). The further reduction of the error by the transition of four layers to eight still causes a reduction of the error although in a small increment. However there does appears to be a interesting difference between the different networks. In the network trained on random data we observe that the error from the homonuclear PES of both the linear and fully connected test models seems to increase with the increase of layers. This occurs aswell in the both the symmetry augmented networks but only for the linear test system. And this shift in error is smaller for the network trained on the N! symmetry augmented data.



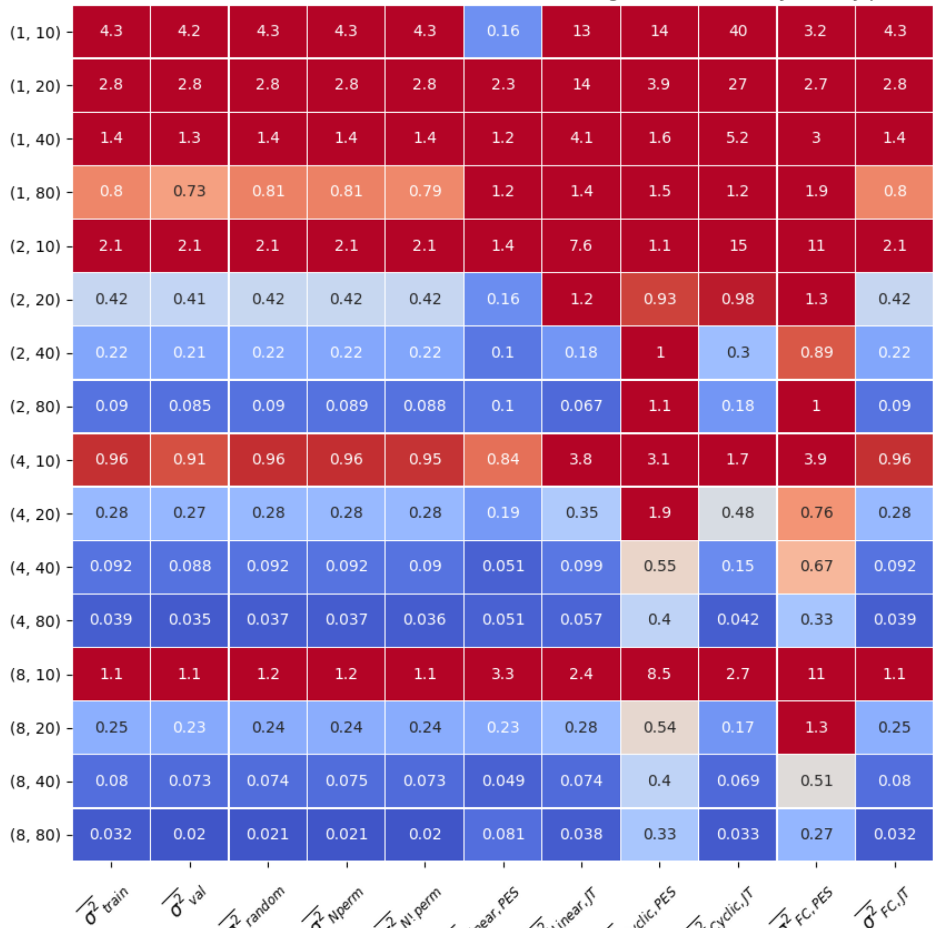
	$\overline{\sigma^2}_{train}$	$\overline{\sigma^2}_{val}$	$\overline{\sigma^2}_{random}$	$\overline{\sigma^2}_{Nperm}$	$\overline{\sigma^2}_{N!perm}$	$\overline{\sigma^2}_{Linear,PES}$	$\overline{\sigma^2}_{Linear,JT}$	$\overline{\sigma^2}_{Cyclic,PES}$	$\overline{\sigma^2}_{Cyclic,JT}$	$\overline{\sigma^2}_{FC,PES}$	$\overline{\sigma^2}_{FC,JT}$
(1, 10)	17	17	17	17	16	16	58	25	34	22	17
(1, 20)	3.2	3.2	3.2	3.2	3.2	0.095	13	4	23	1.6	3.2
(1, 40)	1.5	1.5	1.5	1.5	1.5	0.78	5	3.6	5.4	3.7	1.5
(1, 80)	0.74	0.75	0.74	0.74	0.73	0.64	0.49	0.87	1.9	0.91	0.74
(2, 10)	2.8	2.8	2.8	2.8	2.8	3.6	7.6	3.4	7.4	4.5	2.8
(2, 20)	0.37	0.37	0.37	0.37	0.37	0.08	0.2	1.3	0.29	0.91	0.37
(2, 40)	0.19	0.19	0.19	0.19	0.19	0.092	0.19	1.8	0.42	1.1	0.19
(2, 80)	0.079	0.079	0.078	0.079	0.079	0.087	0.11	0.59	0.25	1.3	0.079
(4, 10)	1.1	1.1	1.1	1.1	1.1	0.77	3.5	1.1	2.8	8.7	1.1
(4, 20)	0.3	0.3	0.3	0.3	0.3	0.13	0.27	0.49	0.57	0.48	0.3
(4, 40)	0.099	0.096	0.094	0.095	0.096	0.12	0.14	0.87	0.23	0.71	0.099
(4, 80)	0.043	0.036	0.034	0.035	0.036	0.078	0.065	0.46	0.068	0.35	0.043
(8, 10)	0.6	0.58	0.59	0.59	0.59	1.3	1.3	7	1.8	7.2	0.6
(8, 20)	0.24	0.23	0.23	0.23	0.23	0.18	0.24	1.5	0.33	1.6	0.24
(8, 40)	0.093	0.065	0.064	0.065	0.065	0.028	0.03	0.48	0.086	0.67	0.093
(8, 80)	0.029	0.024	0.023	0.024	0.024	0.14	0.048	0.39	0.043	0.46	0.029

Error matrix overview for the NN trained on randomized data augmented with N symmetry permutations



	$\overline{\sigma^2}_{\text{train}}$	$\overline{\sigma^2}_{\text{val}}$	$\overline{\sigma^2}_{\text{random}}$	$\overline{\sigma^2}_{\text{Nperm}}$	$\overline{\sigma^2}_{\text{N! perm}}$	$\overline{\sigma^2}_{\text{Linear, PES}}$	$\overline{\sigma^2}_{\text{Linear, JT}}$	$\overline{\sigma^2}_{\text{Cyclic, PES}}$	$\overline{\sigma^2}_{\text{Cyclic, JT}}$	$\overline{\sigma^2}_{\text{FC, PES}}$	$\overline{\sigma^2}_{\text{FC, JT}}$
(1, 10)	4.3	4.3	4.3	4.3	4.3	0.19	16	15	41	0.6	4.3
(1, 20)	3.2	3.2	3.2	3.2	3.2	0.19	18	0.21	22	0.33	3.2
(1, 40)	1.4	1.4	1.4	1.4	1.4	1.6	7.5	1.2	3.5	6.5	1.4
(1, 80)	0.75	0.76	0.75	0.75	0.75	0.63	1.7	2.3	0.86	1.5	0.75
(2, 10)	2.7	2.7	2.7	2.7	2.7	2	6.6	1.5	14	1.7	2.7
(2, 20)	0.43	0.42	0.43	0.42	0.43	0.13	0.84	2.4	1.1	2.9	0.43
(2, 40)	0.19	0.19	0.19	0.19	0.19	0.085	0.13	0.67	0.46	0.86	0.19
(2, 80)	0.083	0.082	0.082	0.081	0.082	0.14	0.093	0.23	0.21	0.2	0.083
(4, 10)	1.3	1.3	1.3	1.3	1.3	1.2	3.6	4.5	2.6	22	1.3
(4, 20)	0.29	0.29	0.3	0.29	0.3	0.12	0.29	0.9	0.42	0.89	0.29
(4, 40)	0.12	0.11	0.11	0.11	0.11	0.17	0.17	1	0.19	0.74	0.12
(4, 80)	0.037	0.035	0.034	0.033	0.034	0.025	0.049	0.54	0.089	0.29	0.037
(8, 10)	0.57	0.57	0.57	0.57	0.57	1.5	1.4	4.1	1.3	5.2	0.57
(8, 20)	0.23	0.23	0.23	0.23	0.23	0.19	0.2	0.77	0.2	0.96	0.23
(8, 40)	0.077	0.061	0.061	0.06	0.061	0.075	0.054	0.33	0.054	0.21	0.077
(8, 80)	0.034	0.019	0.018	0.017	0.018	0.13	0.04	0.28	0.031	0.26	0.034

Error matrix overview for the NN trained on randomized data augmented with N! symmetry permutations



	$\overline{\sigma^2}_{\text{train}}$	$\overline{\sigma^2}_{\text{val}}$	$\overline{\sigma^2}_{\text{random}}$	$\overline{\sigma^2}_{\text{Nperm}}$	$\overline{\sigma^2}_{\text{N! perm}}$	$\overline{\sigma^2}_{\text{Linear, PES}}$	$\overline{\sigma^2}_{\text{Linear, JT}}$	$\overline{\sigma^2}_{\text{Cyclic, PES}}$	$\overline{\sigma^2}_{\text{Cyclic, JT}}$	$\overline{\sigma^2}_{\text{FC, PES}}$	$\overline{\sigma^2}_{\text{FC, JT}}$
(1, 10)	4.3	4.2	4.3	4.3	4.3	0.16	13	14	40	3.2	4.3
(1, 20)	2.8	2.8	2.8	2.8	2.8	2.3	14	3.9	27	2.7	2.8
(1, 40)	1.4	1.3	1.4	1.4	1.4	1.2	4.1	1.6	5.2	3	1.4
(1, 80)	0.8	0.73	0.81	0.81	0.79	1.2	1.4	1.5	1.2	1.9	0.8
(2, 10)	2.1	2.1	2.1	2.1	2.1	1.4	7.6	1.1	15	11	2.1
(2, 20)	0.42	0.41	0.42	0.42	0.42	0.16	1.2	0.93	0.98	1.3	0.42
(2, 40)	0.22	0.21	0.22	0.22	0.22	0.1	0.18	1	0.3	0.89	0.22
(2, 80)	0.09	0.085	0.09	0.089	0.088	0.1	0.067	1.1	0.18	1	0.09
(4, 10)	0.96	0.91	0.96	0.96	0.95	0.84	3.8	3.1	1.7	3.9	0.96
(4, 20)	0.28	0.27	0.28	0.28	0.28	0.19	0.35	1.9	0.48	0.76	0.28
(4, 40)	0.092	0.088	0.092	0.092	0.09	0.051	0.099	0.55	0.15	0.67	0.092
(4, 80)	0.039	0.035	0.037	0.037	0.036	0.051	0.057	0.4	0.042	0.33	0.039
(8, 10)	1.1	1.1	1.2	1.2	1.1	3.3	2.4	8.5	2.7	11	1.1
(8, 20)	0.25	0.23	0.24	0.24	0.24	0.23	0.28	0.54	0.17	1.3	0.25
(8, 40)	0.08	0.073	0.074	0.075	0.073	0.049	0.074	0.4	0.069	0.51	0.08
(8, 80)	0.032	0.02	0.021	0.021	0.02	0.081	0.038	0.33	0.033	0.27	0.032

5.3.2 Network assesment through the visualization of the potential energy surface

The next step in the assesment of the neural networks would be to look at the test set data (both their predicted values and the squared error surfaces). the figure below provides a visualization of the exact values of these test sets and gives a clear indication of what we need to look for when analyzing the networks.

- The homonuclear PES of the molecular systems all are shaped in a same fashion but will differ in energy values. A requisite for a well behaved neural network would thus be that each of these PES are shape similarly and that a distinction in energy should be present based upon the internal structure of the system in question
- The Jahn-Teller PES of the linear test system has the shape of a concentric oval that starts in the $t_1 = t_2 = 0$ are and spreads across the whole t space
- The Jahn-Teller PES of the cyclic test systems is symmetric around the $t_1 = t_2 = 0$ point and is split up in four different sections which are separated along the diagonal of the graph, i.e. where the absolute values of both the t parameters is equal.

Due too the large amount of data and the rather large visual representation of the data evolution in terms of network complexity we will restrict ourselves to only analyze the results from the networks that are trained on the completely random data and from one molecular structure.

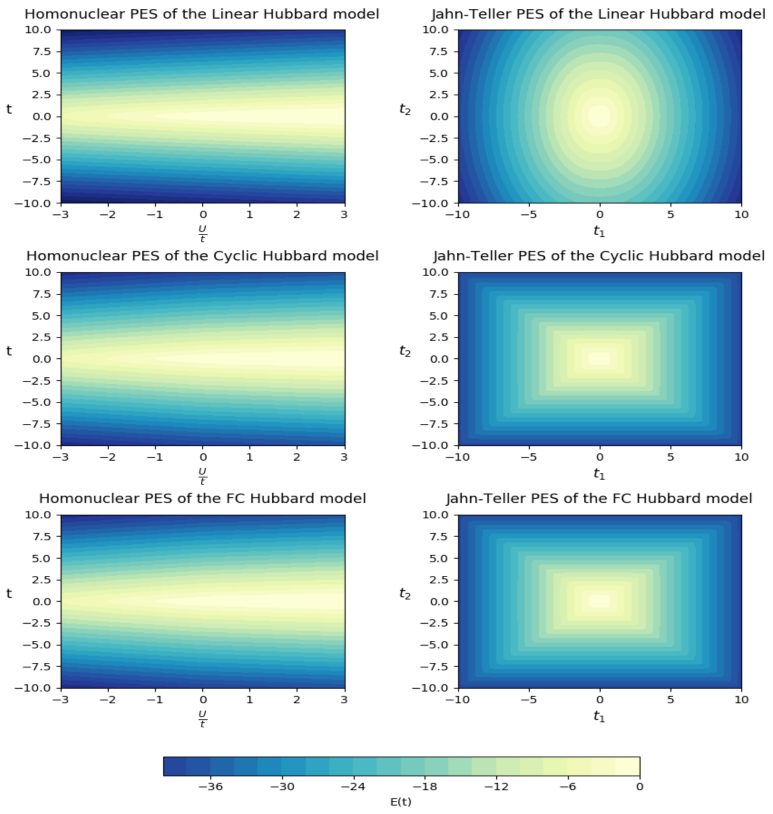
The system of our choice will be the linear test system for two different reasons.

1. Compared to the original training input this testing system differs the most on a structural level. Indeed the training data is the four site Hubbard model in which the maximum amount of bonds is present in the system, compared to the linear Hubbard model which has the minimal amount of bonds
2. The sampling was done according to a continuous uniform distribution for both the t parameter and the U parameter of the Hubbard system. From statistics we know that the probability density function of a uniform distribution is given by

   $$f(x) = \frac{1}{b-a} \text{ for the uniform distribution } \mathcal{U}(a,b) \text{ with } x \in [a,b]$$

   Where in the case of a bond not existing between two atoms would correspond to a value of t=0. Statistically speaking we have a chance of this occuring with 5 % chance. And so the estimated chance of the inclusion of a system which has 3 bonds would correspond to $0,05^3$ as there are 3 bond that each have to be zero. On top of that there is a total of sixteen structures that have three bonds of which twelve corresponds to the linear Hubbard model. This bring us too a total estimated chance of 0,009375%

chance of the inclusion of a linear structure in the training data. Taking the size of the dataset into account thus would (at best) roughly come down to 9375 linear structures of the million input structures



Discussion of the homonuclear PES Looking at the evolution of the prediction of the homo nuclear PES test set in function of the network complexity we immediately notice that the shape of the PES is already present in the simplest of networks . Indeed starting from the top left corner of the grid, i.e. one layer and ten nodes per layer, we already see that the energy is centered around the t=0 value and decreases when we increase this parameter. However we do note that the energy tends to decrease quicker in comparison to the other graphs. When increasing the network complexity wether it be the amount of layers or the amount of nodes per layer we notice that the network quickly evolves to the desired shape of the exact PES.

Closer inspection of the evolution of the squared error visualization grants us additional insight of potential problem areas and how the network adjusts it's predictions. Indeed when looking at the simplest network we immediately notice a large concentration of faulty predictions centered around the t=0 value. We can also see that this error is some what present in all the networks but definitely improves with the increasing complexity of the network. When looking at the looking at the most complex model of the set we see that the error is almost completely centered around the area of t=0 and $U > 0$. As was mentioned in the section concerning the analysis of the two site models this possibly finds the origin in the underrepresented sampling of the t=0 area and the anomalous behavior of the PES around this area as the energy shows a discontinuity around this area.

Discussion of the Jahn-Teller PES The evolution of the Jahn-Teller PES of the linear test system displays a far more drastic change in comparison to the homo nuclear PES. Examining the least complex network already shows a remarkable difference to the expected PES as the contour plot displays a more diamond shape instead of the expected oval. We see this shape change in a similar fashion when we increase the amount of layers from one to eight (with 10 nodes per layer) and when we increase the amount of nodes per layer from ten to eighty (in one layer). Indeed the diamond shape goes to hexagon like contour plot followed up by a hexagon that is some what smoothened and finally to crude version of the desired oval shape. Upon increasing the complexity of the network even further we see that curde oval shape smoothens further out too the expected shape of the exact solution too the symmetric variation in bonds.

Looking at the error graphs in function of the evolution of the network complexity we see a the evolution of the values themselves. Looking at the first column and row (the first level of increasing network complexity) we find similar results to the visual analysis of the predicted results. Upon increasing the complexity the error tends to be reduced but this does not occur in a equal fashion when comparing the increasing amount of layers too the increasing amount of nodes in one layer.

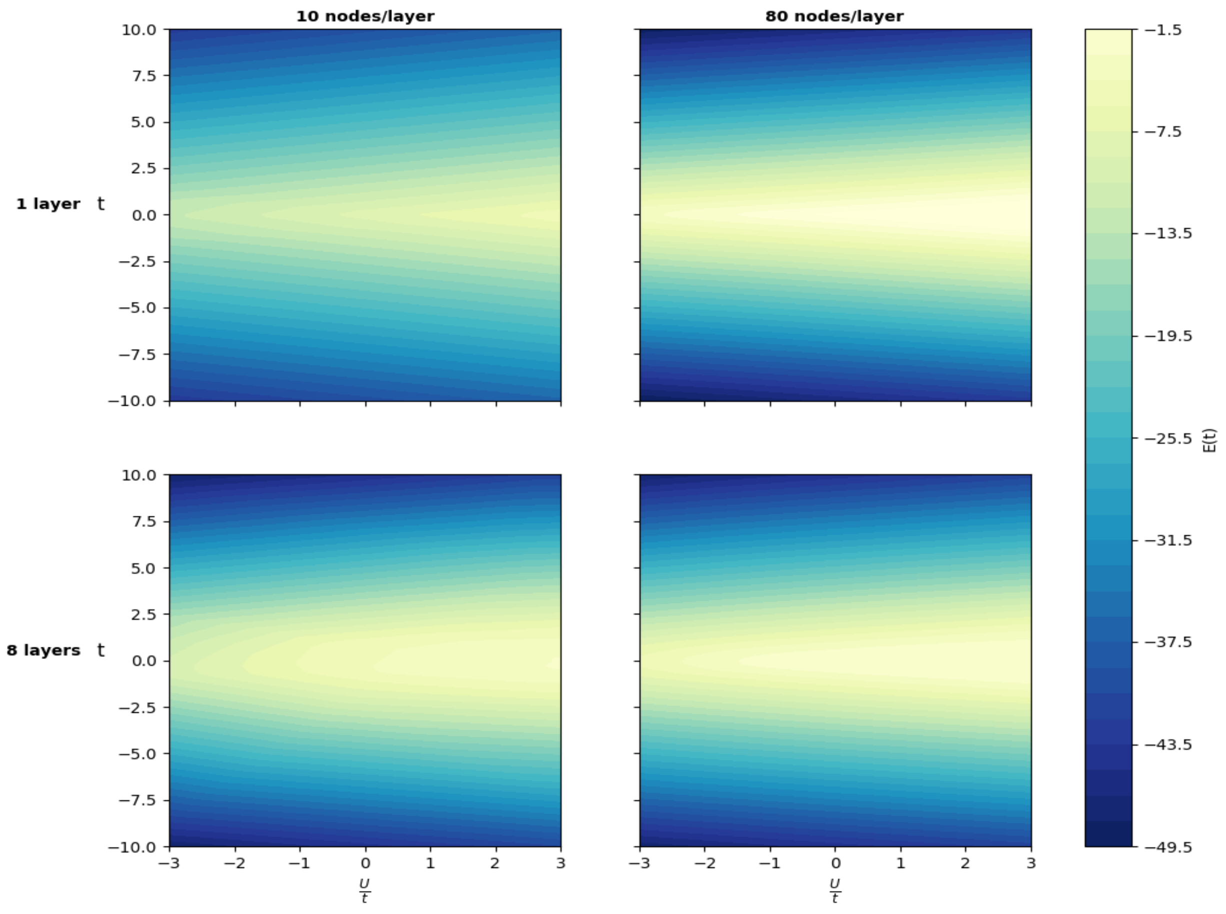
Indeed when increasing the amount of nodes in the one layer network we see thath the original symmetrical error start to perform better along the diagonal of the variation of t values. This means that the network becomes better at predicting at the energy of structures where the outer bonds and the inner bond are in a region where they are (relatively) close to each other. This well behaved region expands the more nodes we add to the one layer model. When comparing the situation where the amount of layers increases as too when the amount of nodes increases it becomes apparent that the overal error in the case of increasing amount of layers is lower. This highlights the some what non-linear relationship between increasing complexity due too more layers and due too more nodes per layer. Although the amount of connections is increased in a linear equally in both cases the network performance behaves very different. What is different however is the amount of non-linear mapping that occurs in the networks with increasing amount of layers, as more layers implies more consecutive transformations with the activation function. Thus highlighting the importance of the foundation upon which the neural networks are build.

One has to however be cautious to state this relationship when moving further down into the complexity evolution. Indeed it becomes apparent that the complexer network tend to have a error evolution which does not necesarily improve stepwise upon further increasing the network complexity. The two and four layer models appear to have a relatively volatile error surface when it comes to the increasing amount of nodes per layer. The eight layer models do show a more consecutive update of the error surface and eventually become the overall best model but with a similar error centered around the $t_1 = t_2 \approx 0$.

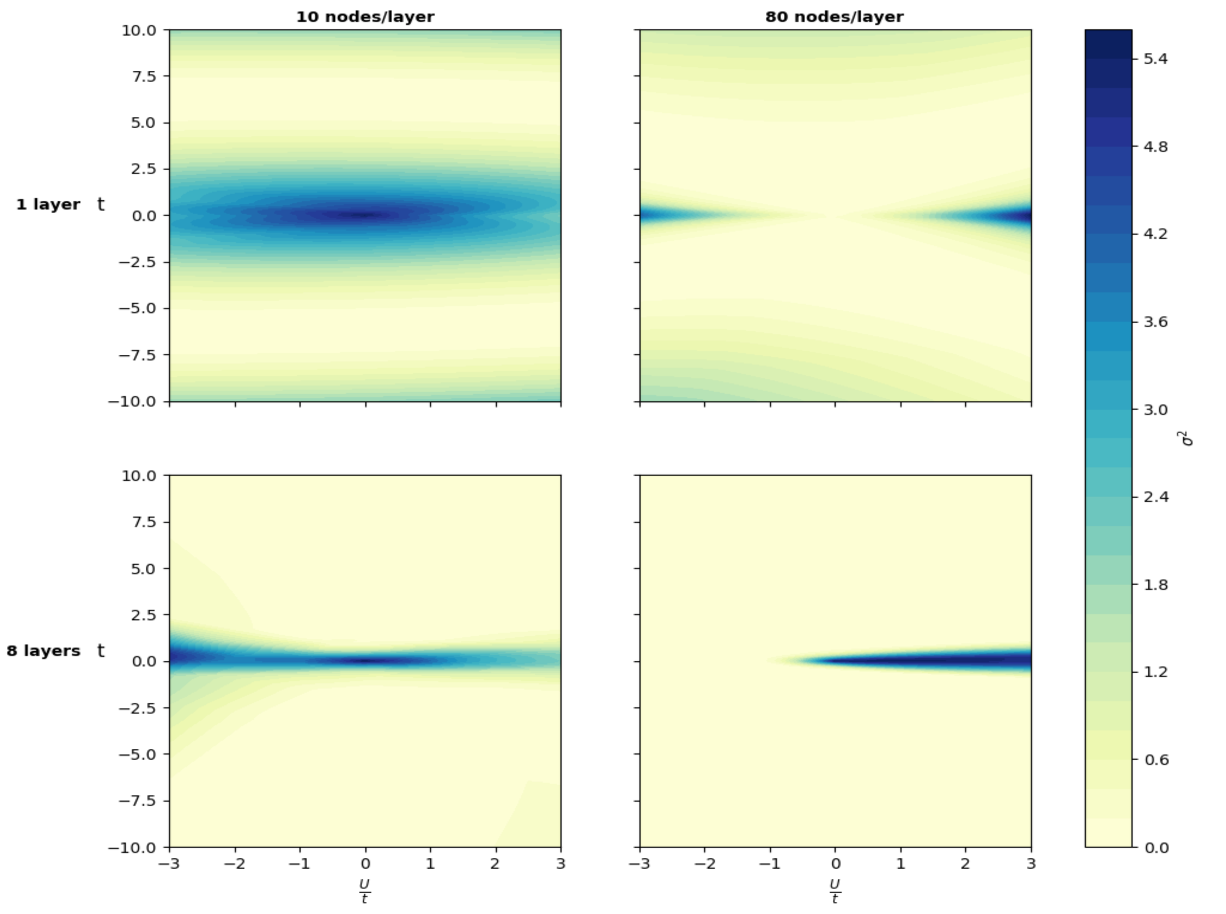
General conclusion In general we see that increasing the network complexity has a benificial effect upon network performance. The predictive power scales with both the amount of layers in the network and the amount of nodes per layer in the network, albeit in a unequal way. The main problem area in the large part of these models appears to be centered around the limit case when t parameters are centered around the zero value.

As too the other test structures, i.e. the cyclic and fully connected Hubbard model we get a result that is similar too the linear Hubbard model. These systems however do display a more well defined improvement upon increasing network complexity. The error surfaces of the Jahn-Teller PES does not display the same kind of "volatility" as that of the linear system but this can somewhat be expected as they resembles the majority of the training data the best. If the reader is interested in examining the evolutions of performance themselves they are referred to the addendum where they will be included as additional information.

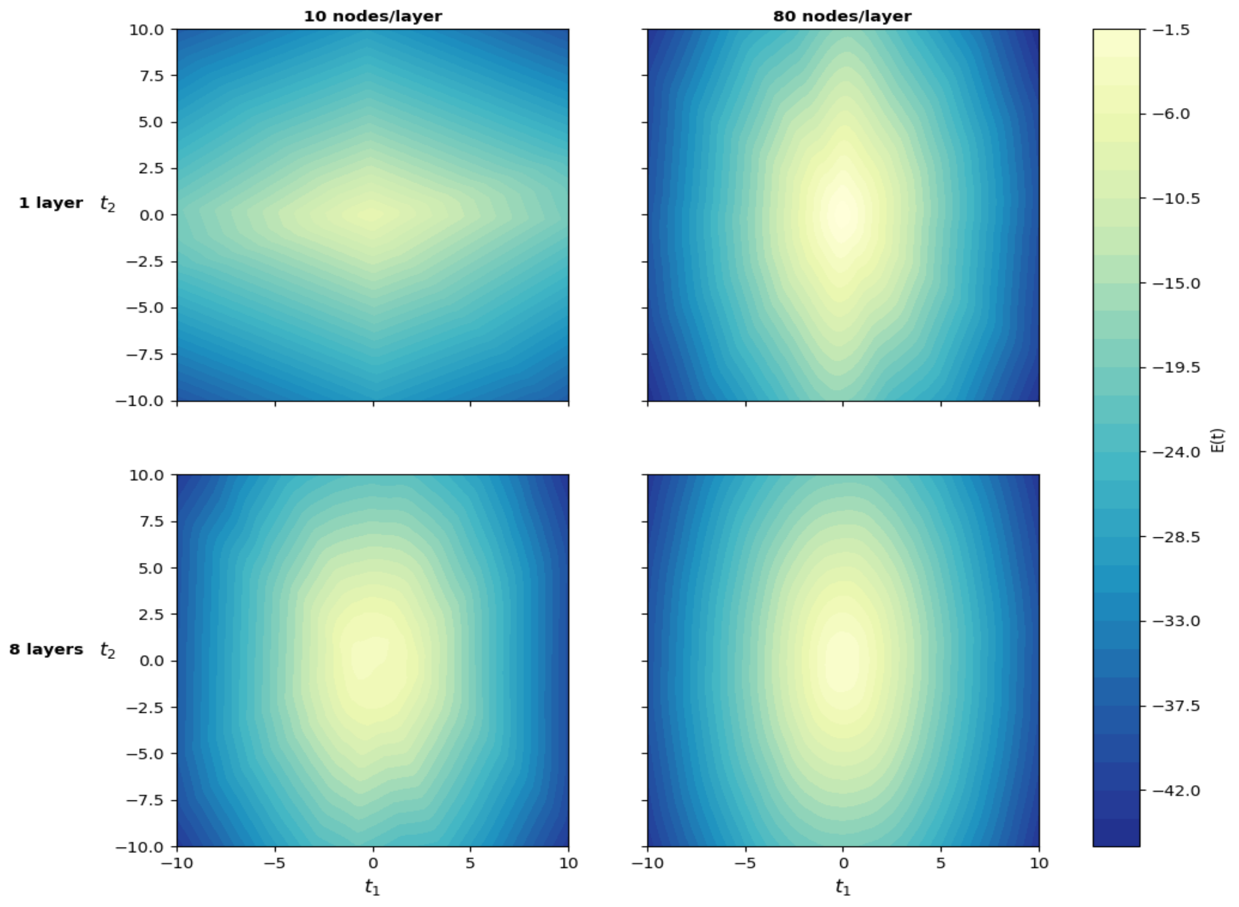
Evolution of the Linear Hubbard model Homonuclear PES in function of network complexity



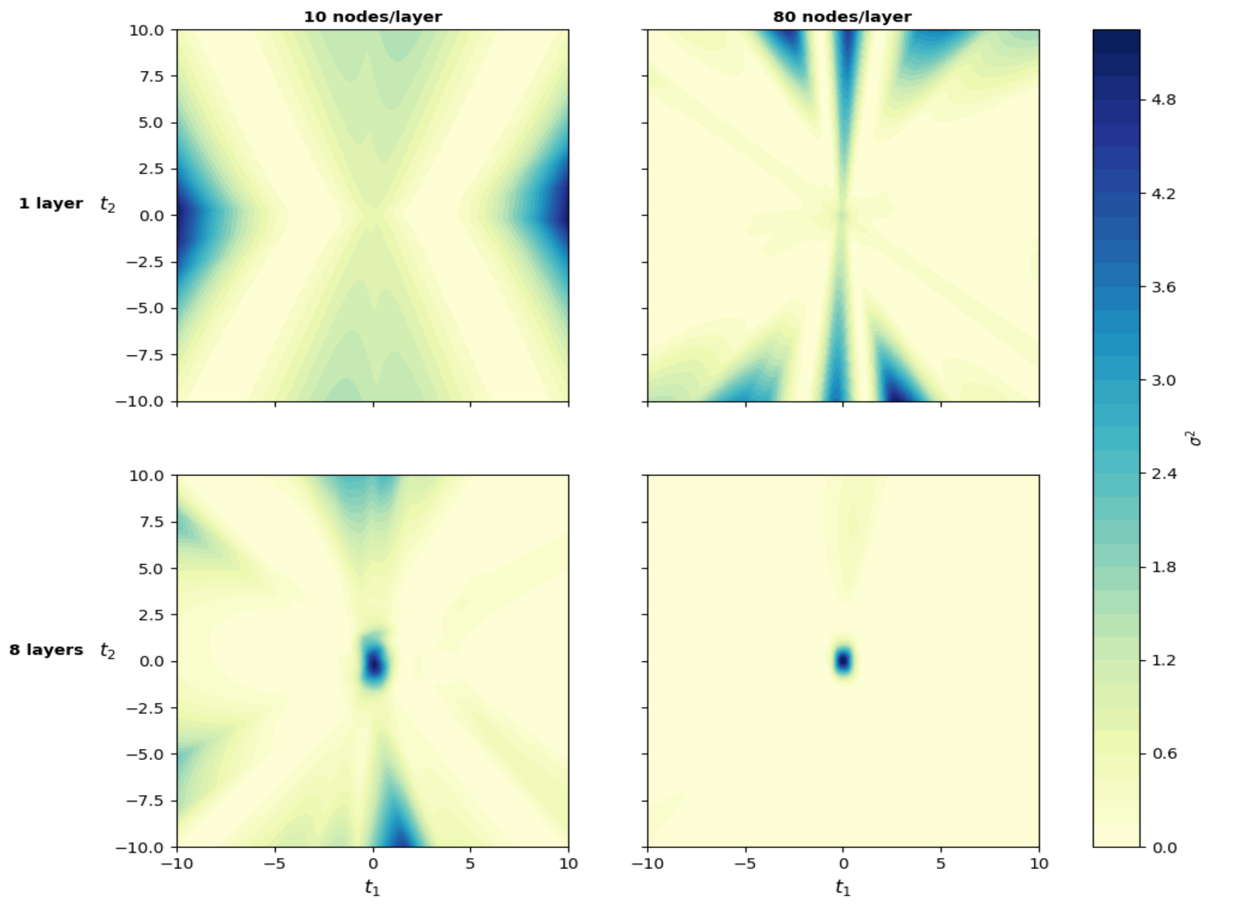
Evolution of the MSE of the Linear Hubbard model Homonuclear PES in function of network complexity



Evolution of the Linear Hubbard model Jahn-Teller PES in function of network complexity



Evolution of the MSE of the Linear Hubbard model Jahn-Teller PES in function of network complexity



Comparisson with networks utilizing augmented data After processing all the visual data from the linear test system in function of the network complexity we came to the conclusion that the network which shows the best overall performance was the one which has eight layers and eighty nodes per layer. One can ask the question if this would differ for the networks that were trained on data that has been augmented with established physical knowledgee. As a comparison we will analyse the same grid of complexity evolution for the networks that have been trained on the N! symmetry augmented data.

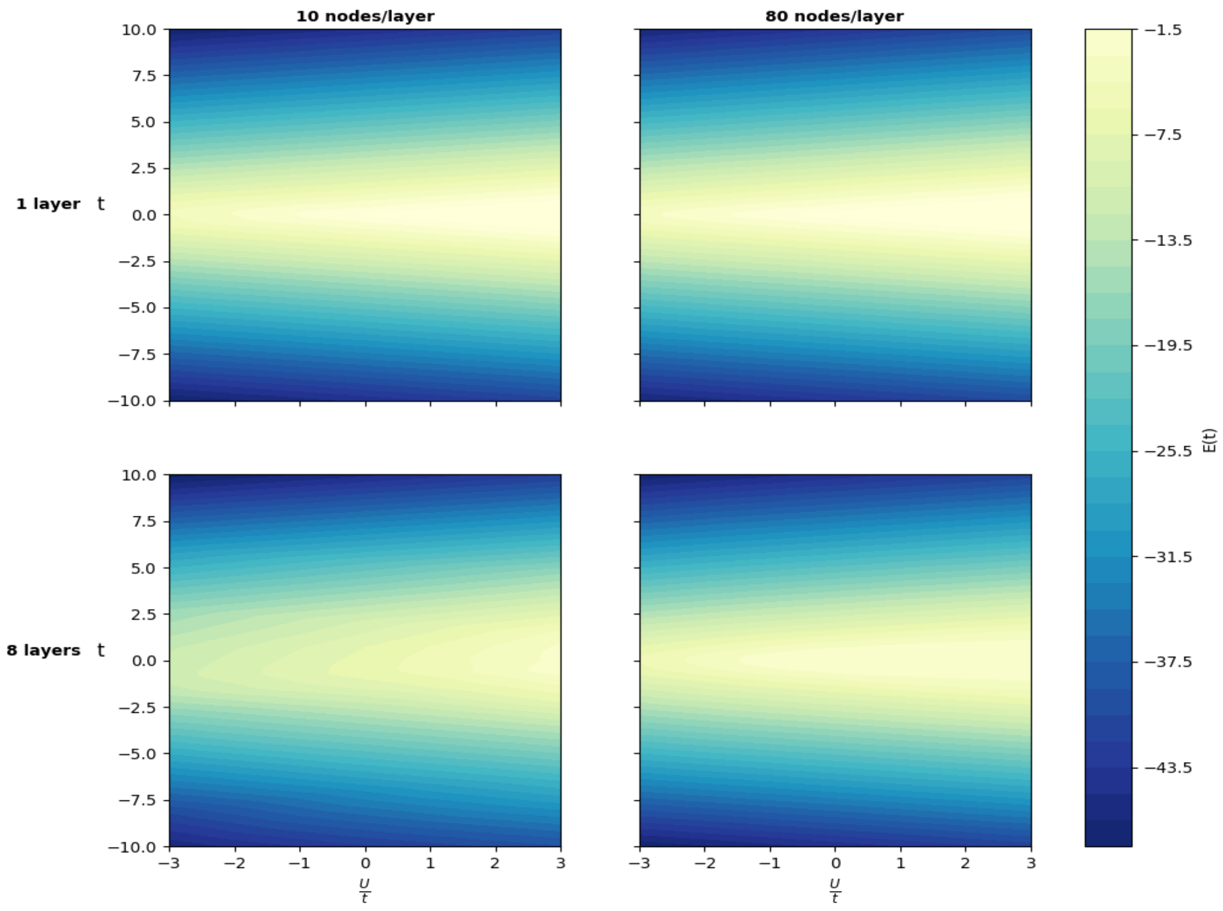
As was visible in the visual analysis of the networks trained on the completely randomized data we see that the N! symmetry augmented data comes to the same conclusion that being that the network of eight layers and eighty nodes per layer produces the overall best performance. However the evolution of the networks in terms of complexity does provide a different result than the random trained networks.

Indeed when looking at the PES of the homo nuclear test systems we see a significant difference in the error surfaces of the networks of low complexity that being the one layer network with ten nodes per layer and twenty nodes per layer and the two layer network with ten nodes per layer. Upon closer inspection of the error surfaces we that the main difference between these error surfaces occurs in the region of t=0 where a lot of variation occurs in both the models.

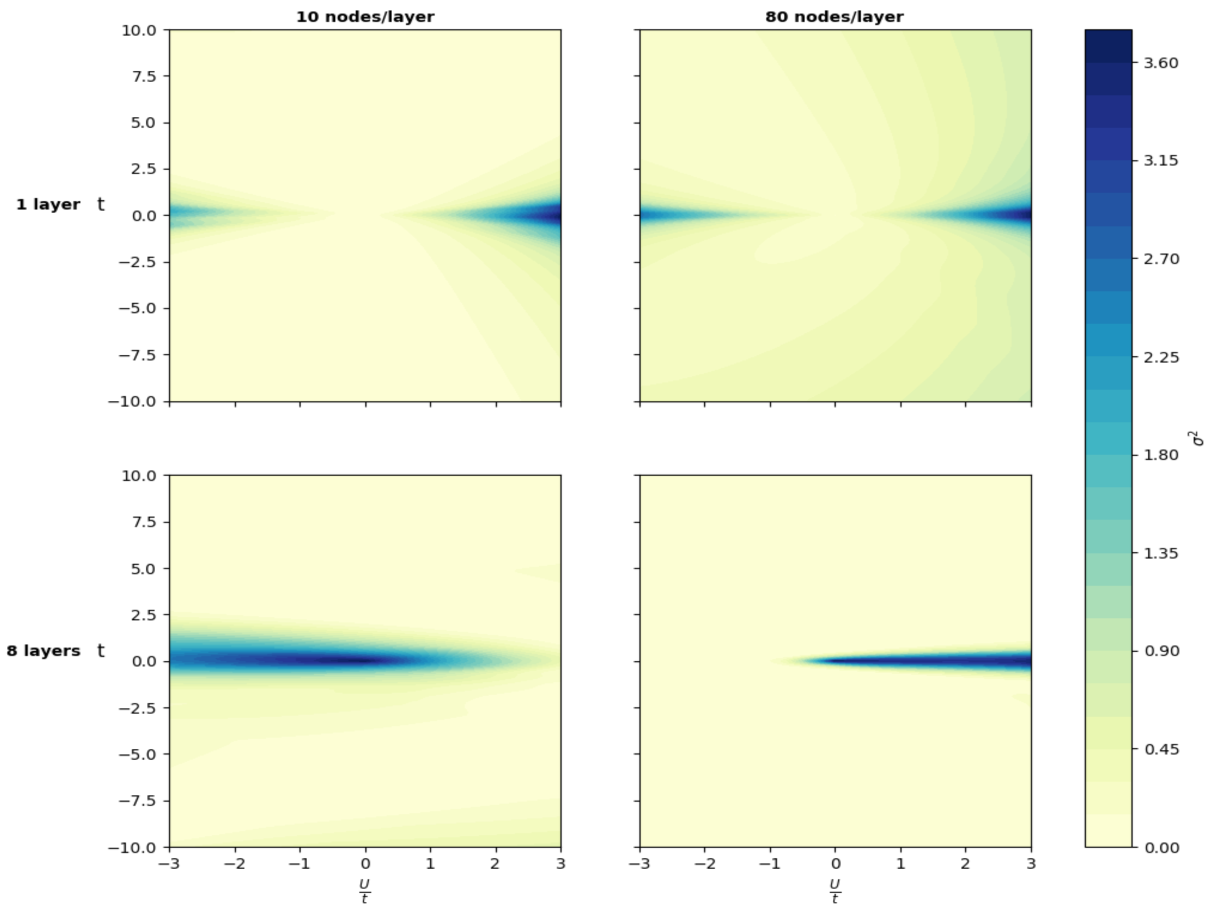
Where the homo nuclear PES was more or less the same for both these sets of models there is a significant difference present in the Jahn-Teller PES. Indeed when looking at the simplest network we already see the largest difference between these two sets of networks. In the case of the random trained networks the PES of the simplest model started out as a diamond shaped surface. In the case of the network trained on the symmetry augmented data we obtain the hexagon shaped PES which was visible in the random trained networks upon addition of more layers and/or more nodes per layer. This difference continues itself throughout the entire evolution in terms of complexity as the error surfaces of the symmetry augmented networks tend to be more defined.

As a general assessment we can state that the use of augmented data has some benefit when it concerns networks of lesser complexity. However given a certain level of network complexity the network comes to a similar conclusion for both the completely random training data and the N! symmetry augmented data

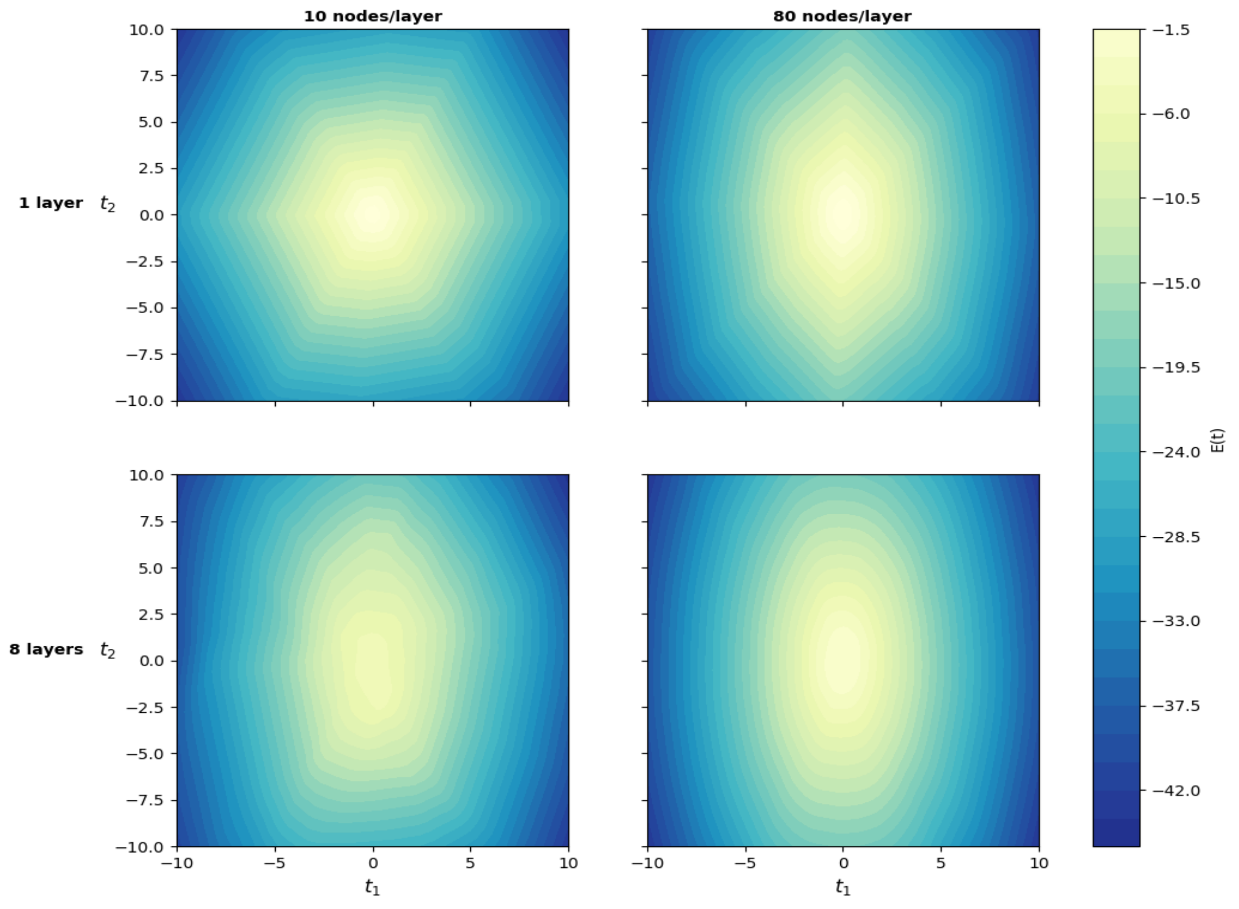
Evolution of the Linear Hubbard model Homonuclear PES in function of network complexity (N! symmetry)



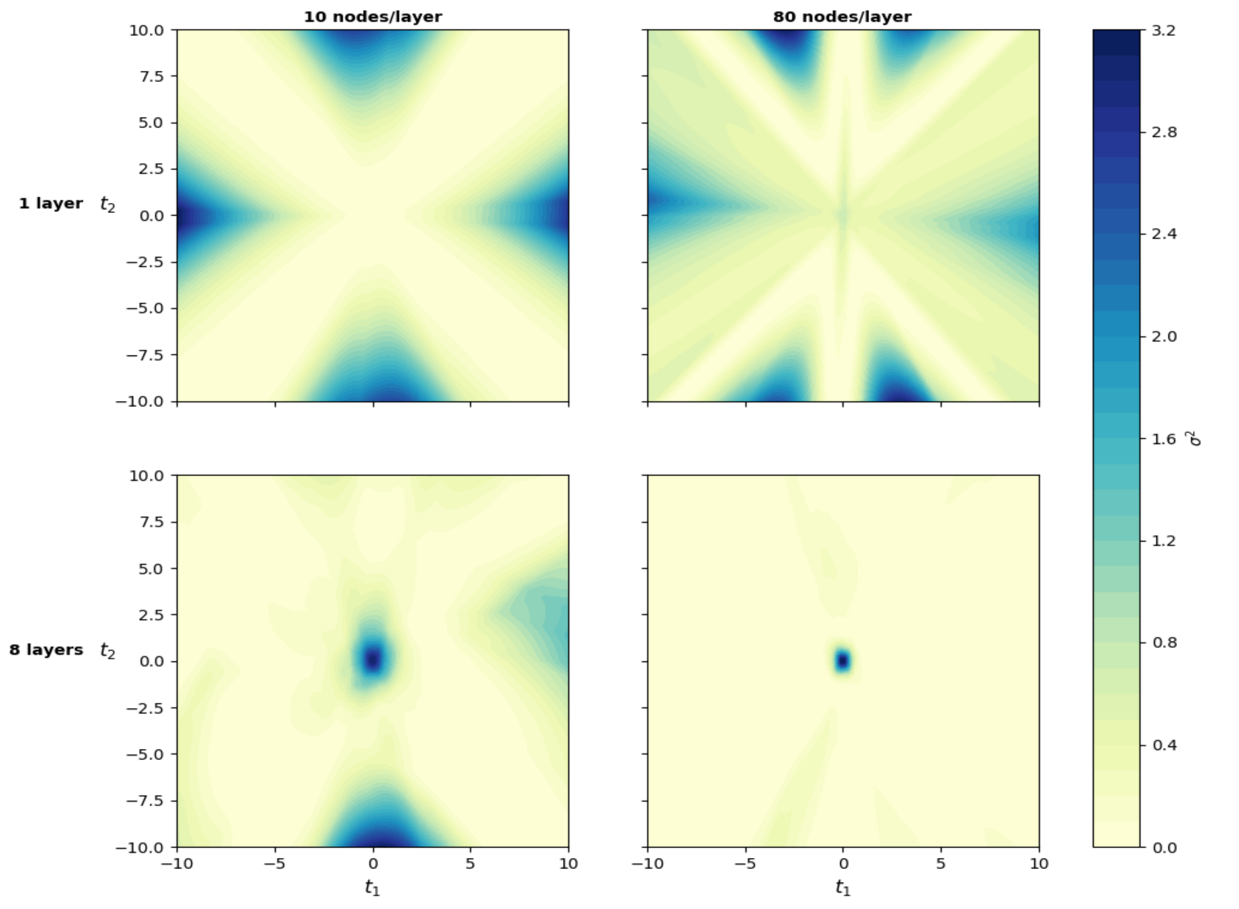
Evolution of the MSE of the Linear Hubbard model Homonuclear PES in function of network complexity (N! symmetry)



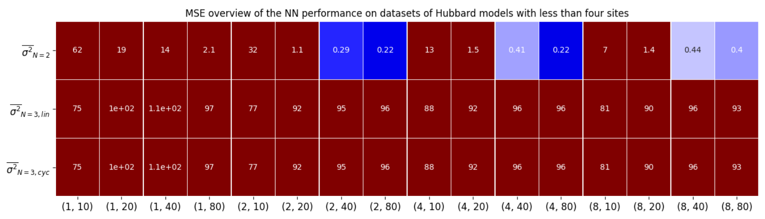
Evolution of the Linear Hubbard model Jahn-Teller PES in function of network complexity (N! symmetry)



Evolution of the MSE of the Linear Hubbard model Jahn-Teller PES in function of network complexity (N! symmetry)



Performance on structures of lesser complexity The table below contains the networks that were trained on random data their general performance on the randomized datasets of the two, linear three and cyclic three site Hubbard models defined in the four site Hubbard model basis. As is clearly visible the overal performance of these networks is not up too par with that on the four site training data and test data. The only expection of these appears to be the two site testing data which has a decent performance for the networks of the following complexity of two, four and eight layers either consisting of fourty or eighty nodes per layer.



MSE overview of the NN performance on datasets of Hubbard models with less than four sites

	(1, 10)	(1, 20)	(1, 40)	(1, 80)	(2, 10)	(2, 20)	(2, 40)	(2, 80)	(4, 10)	(4, 20)	(4, 40)	(4, 80)	(8, 10)	(8, 20)	(8, 40)	(8, 80)
$\overline{\sigma^2}_{N=2}$	62	19	14	2.1	32	1.1	0.29	0.22	13	1.5	0.41	0.22	7	1.4	0.44	0.4
$\overline{\sigma^2}_{N=3,lin}$	75	1e+02	1.1e+02	97	77	92	95	96	88	92	96	96	81	90	96	93
$\overline{\sigma^2}_{N=3,cyc}$	75	1e+02	1.1e+02	97	77	92	95	96	88	92	96	96	81	90	96	93

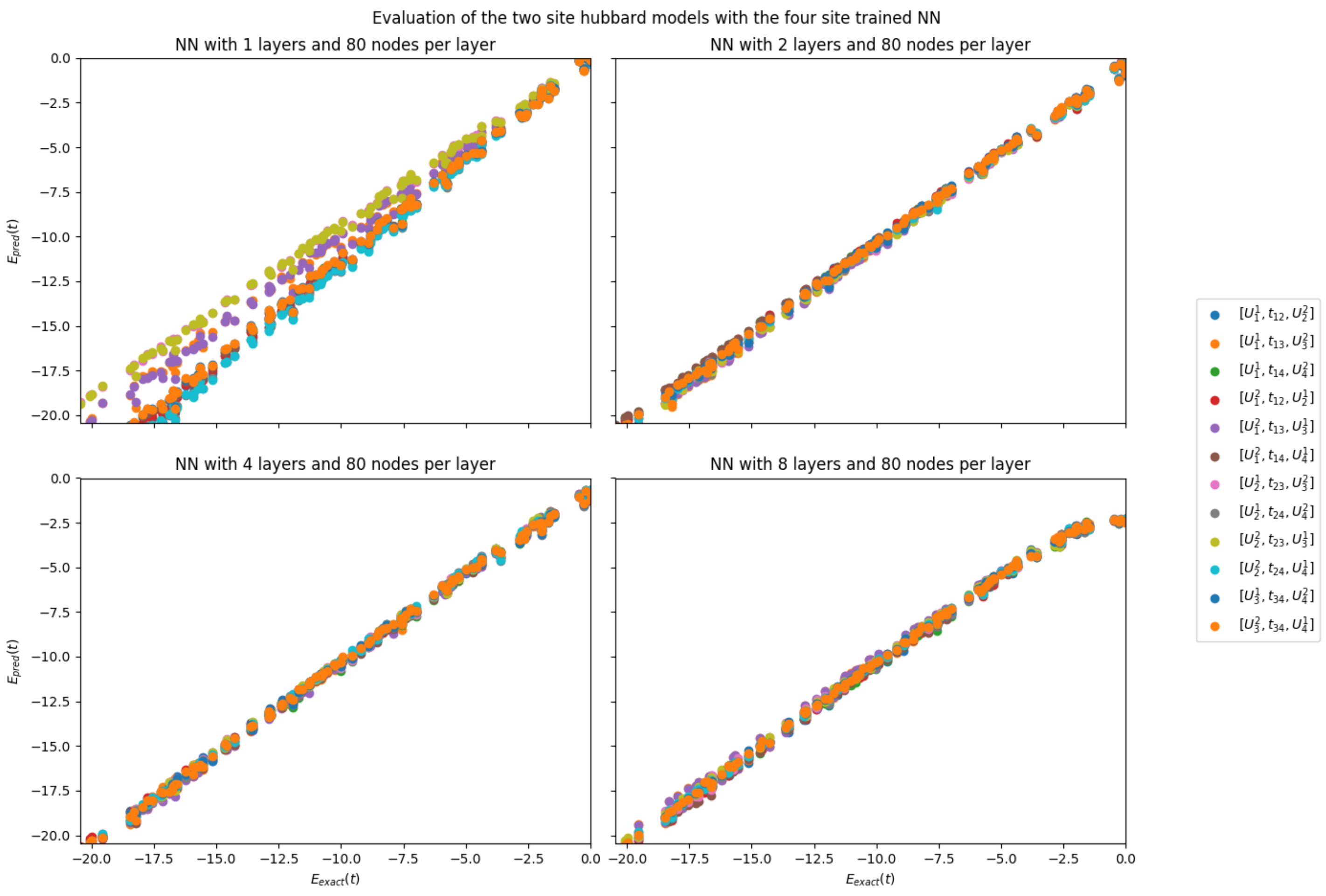
Indeed looking at figure (nr) a visual depiction of the predicted energy versus the exact energy has been made for a hundered points and their permutation across the input vector. This was done for the optimal models for each layer of the random trained neural networks. Looking at the two site predictions we notice that the one layer model with eighty nodes per layer has trouble with the inherent symmetry of the Hubbard system resulting in a distribution of points that do not lie on a straight line. For the two, four and eight layer model however the different permutations all lie on the same line and coincide with the exact energy relatively well.

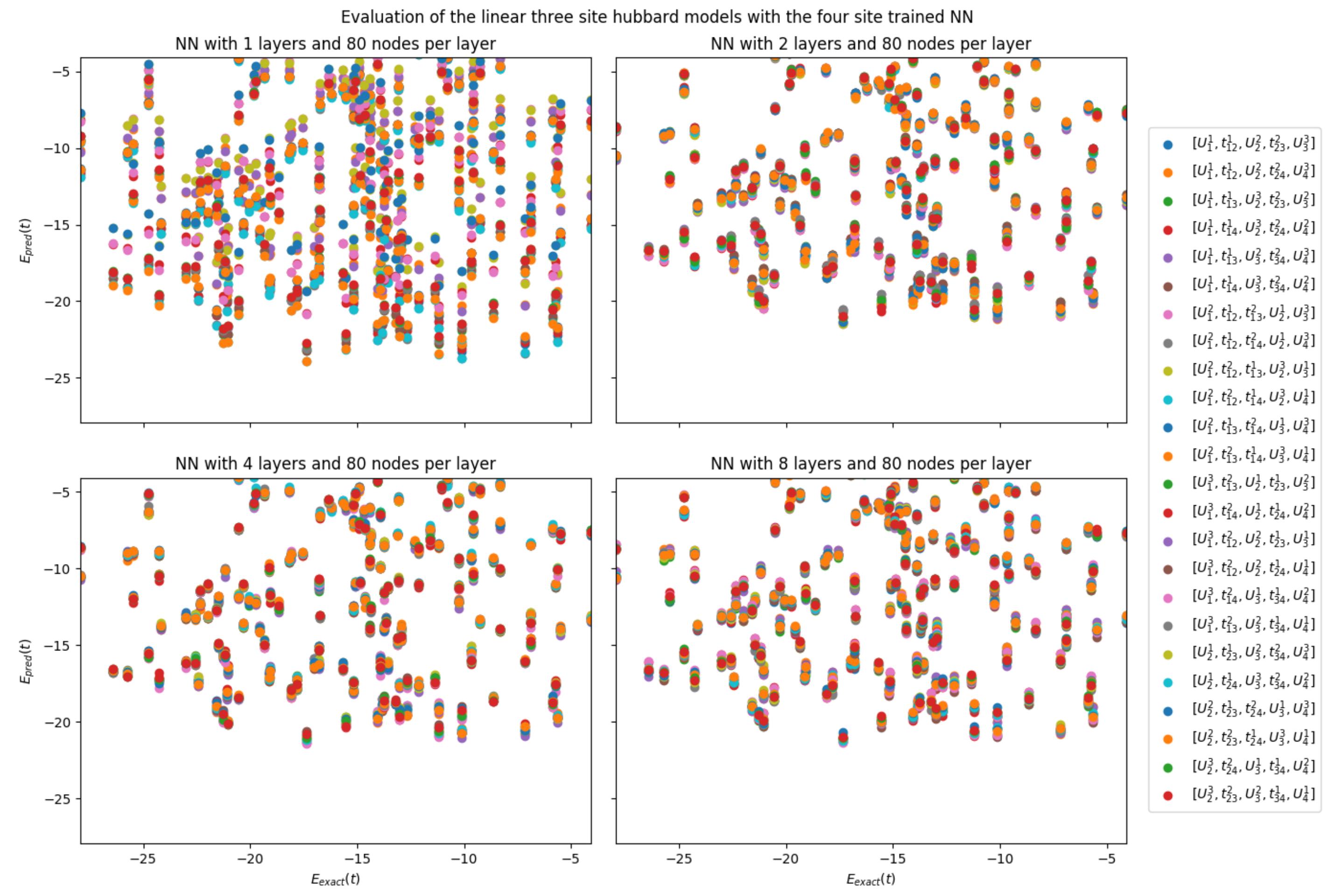
The linear three site testing model however does not show any form of consistency in its predictive capabilities. We hypothesize that the two site systems are fit for predictions with the four site due to being implicitly present in the Jahn-Teller effect for the four site system. Indeed if we stretch the middle bond in a linear four site model or one of the pairs of the cyclic four site model to a "infinite" distance the t value should go to zero in this case, when this occurs we technically have two two site models. With the Hubbard model being a full CI method we expect that it includes size consistency meaning that for two molecules A and B that are not connected

$$E\left(A+B\right)=E\left(A\right)+E\left(B\right)$$

For two molecular fragments that are the same this would imply double the energy of the molecular fragment. The occurrence of this phenomenon implies that though examples of dissociation in molecular fragments were not included in the original training data, the network was able to learn this characteristic through the analysis of unrelated data.

The anomalous behavior of the eight layer model in the low energy limit could also be explained following this reasoning. Indeed looking back at the graph that displays the error surface of the eight layer network with eighty nodes we see that the area of minimal performance is the section where t approaches zero. In the low energy limit of the two site Hubbard model we know that the t parameter approaches zero.


Evaluation of the two site hubbard models with the four site trained NN
NN with 1 layers and 80 nodes per layer
NN with 2 layers and 80 nodes per layer
NN with 4 layers and 80 nodes per layer
NN with 8 layers and 80 nodes per layer
$E_{pred}(t)$
$E_{exact}(t)$
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
−17.5
−20.0
$[U_1^1, t_{12}, U_2^2]$
$[U_1^1, t_{13}, U_3^2]$
$[U_1^1, t_{14}, U_4^2]$
$[U_1^2, t_{12}, U_2^1]$
$[U_1^2, t_{13}, U_3^1]$
$[U_1^2, t_{14}, U_4^1]$
$[U_2^1, t_{23}, U_3^2]$
$[U_2^1, t_{24}, U_4^2]$
$[U_2^2, t_{23}, U_3^1]$
$[U_2^2, t_{24}, U_4^1]$
$[U_3^1, t_{34}, U_4^2]$
$[U_3^2, t_{34}, U_4^1]$


Evaluation of the linear three site hubbard models with the four site trained NN
NN with 1 layers and 80 nodes per layer
NN with 2 layers and 80 nodes per layer
NN with 4 layers and 80 nodes per layer
NN with 8 layers and 80 nodes per layer
$E_{pred}(t)$
$E_{exact}(t)$
−5
−10
−15
−20
−25
$[U_1^1, t_{12}^1, U_2^2, t_{23}^2, U_3^3]$
$[U_1^1, t_{12}^1, U_2^2, t_{24}^2, U_4^3]$
$[U_1^1, t_{13}^1, U_2^3, t_{23}^2, U_3^2]$
$[U_1^1, t_{14}^1, U_2^3, t_{24}^2, U_4^2]$
$[U_1^1, t_{13}^1, U_3^2, t_{34}^2, U_4^3]$
$[U_1^1, t_{14}^1, U_3^3, t_{34}^2, U_4^2]$
$[U_1^2, t_{12}^1, t_{13}^2, U_2^1, U_3^3]$
$[U_1^2, t_{12}^1, t_{14}^2, U_2^1, U_4^3]$
$[U_1^2, t_{12}^2, t_{13}^1, U_2^3, U_3^1]$
$[U_1^2, t_{12}^2, t_{14}^1, U_2^3, U_4^1]$
$[U_1^2, t_{13}^1, t_{14}^2, U_3^1, U_4^3]$
$[U_1^2, t_{13}^2, t_{14}^1, U_3^3, U_4^1]$
$[U_1^3, t_{13}^2, U_2^1, t_{23}^1, U_3^2]$
$[U_1^3, t_{14}^2, U_2^1, t_{24}^1, U_4^2]$
$[U_1^3, t_{12}^2, U_2^2, t_{23}^1, U_3^1]$
$[U_1^3, t_{12}^2, U_2^2, t_{24}^1, U_4^1]$
$[U_1^3, t_{14}^2, U_3^1, t_{34}^1, U_4^2]$
$[U_1^3, t_{13}^2, U_3^2, t_{34}^1, U_4^1]$
$[U_2^1, t_{23}^1, U_3^2, t_{34}^2, U_4^3]$
$[U_2^1, t_{24}^1, U_3^3, t_{34}^2, U_4^2]$
$[U_2^2, t_{23}^1, t_{24}^2, U_3^1, U_4^3]$
$[U_2^2, t_{23}^2, t_{24}^1, U_3^3, U_4^1]$
$[U_2^3, t_{24}^2, U_3^1, t_{34}^1, U_4^2]$
$[U_2^3, t_{23}^2, U_3^2, t_{34}^1, U_4^1]$

6 Conclusions