Machine Learning for Phase Transition in 1D Transverse Field Ising Model

Third Milestone: Neural Network

Yasamin Panahi*, Asra Rezafadaei*, Parsa Rangriz*

June 21, 2021

From the first part of the project, we have the data set. In the previous milestone, we used some traditional models for training and now we want to design Neural Network for our problem . This is one of the best approaches that we can use to study the phase transition point and two magnetic phases in our Quantum Ising model.

1 Introduction

Estimation is the process that we use to derive some conclusion from our data and known information. Consider a set of variables $\mathbf{x} = \{x_i\}_{i=1,\cdots,N}$ on which one is able to gain some partial observations $\mathbf{y} = \{y_\mu\}_{\mu=1,\cdots,M}$. Our aim is to infer the values of the variables $\mathbf{x}$ based on our observation. Since our model is a finite system (chain of N atoms) and we need the most possible exact numerical solution to obtain our data, we decided to use Exact diagonalization (ED) technique. It's a common numerical technique used in physics to determine the eigenstates and energy eigenvalues of a quantum Hamiltonian. After some procedures that we have mentioned in the first report we generated the data. In our problem, we have some data such as correlation function, energy differences, entanglement entropy, and we seek to find the phase transition point from them, by phase transition point we mean the h coefficient in hamiltonian that will determine the magnetic phase of system according to orientation of spins, Figure 1 shows these two magnetic phases for different ranges of h, if $h < 1$ it's Ferromagnetic and if $h > 1$ it's Paramagnetic.

For this purpose, we apply Neural Networks to our data and evaluate their efficiency by some metrics, then we plot loss vs. epochs curves to get an idea of how the Neural Network is performing. After that for the tuning part by using some well-known optimizations we will be able to chose hyper-parameters which won't end up with under-fitting or over-fitting and hopefully will give us the best result.

2 Feedforward Neural Network

2.1 What is FNN and why we used this model?

A Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The feed forward model (Figure 2) is the simplest form of neural network as information is only processed in one direction. While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards. In this model, a series of inputs enter the layer and are multiplied by the weights. Each value is then added together to get a sum of the weighted input values. Recurrent Neural Network (RNN) and Convolutional Neural Network

*Department of Physics, Sharif University of Technology, Tehran, Iran

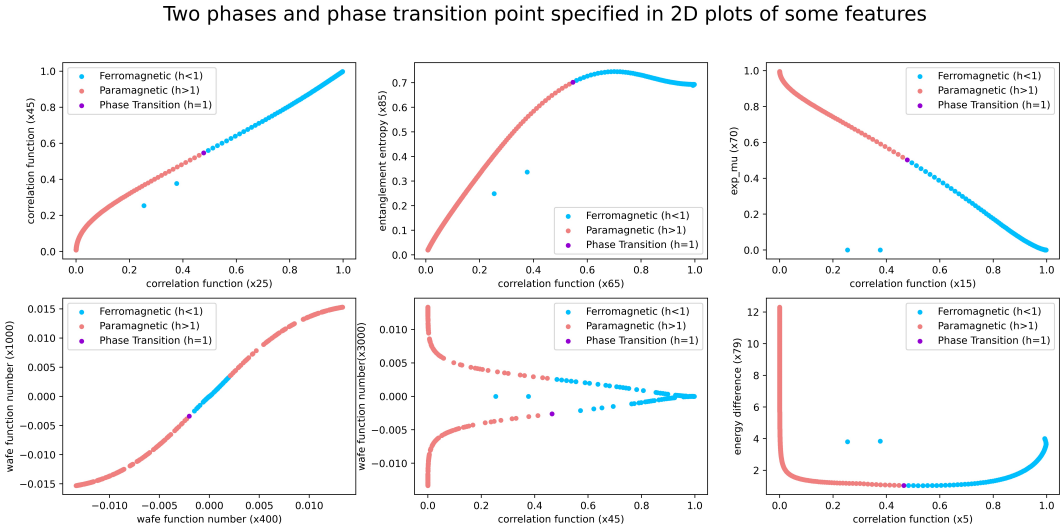


Figure 1: Two magnetic phases for different ranges of h.

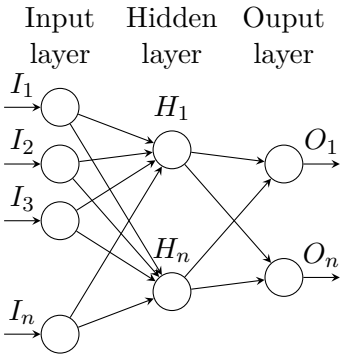


Figure 2: A Simple Feedforward Neural Network

(CNN) are the others offered and useful neural network models, RNN is usually used to analyze time-dependent data set but since our data set is time-independent, RNN isn't useful for our problem. Also CNN commonly is used to analyze visual imagery data set and again this model isn't useful for our problem.In this Report we show that a standard feed-forward neural network can be trained to detect two phase of matter and find the critical point of Quantum ising model perfectly. An important part of neural network is their activation function. The neural network activation functions, in general, are the most significant component of Machine Learning, they are fundamentally used for determining the output of deep learning models, its accuracy, and performance efficiency of the training model that can design or divide a huge scale neural network. In this project we used tanh, softmax, softsign, relu and sigmoid as activation function. For implementation of the neural network in python we used keras module from tensorflow library to make a sequential model where each layer has exactly one input tensor and one output tensor.

3 Evaluation

For using neural network, we have to set some layers and activation functions. But this won't be possible unless we evaluate the performance of each neural network we have used. In the following section, we will explain how we did this process for both Regression and Classification Problems.

This Part includes the Loss vs Epoch curves for regression and Loss & Accuracy vs Epoch curves for classification the proper metrics for each problem are also discussed. In the first part, our goal was only to design a Neural Network to predict our desired results well, so we didn't tune the model, we just played with different hyperparameters, altough we didn't use tuning in this section the results of our designed Neural Network's were quite satisfactory.

3.1 Regression

The main purpose of our problem was to find h which determines phase of the system. Our regression problem is not a regression for vlaues between 0 and 1 as for this case the sigmoid function for the last layer of Neural Network would be a good choice but since the 'y' label in our data set can be a decimal value between 0 and 7, our probelm is regression to arbitrary values, so the proper choice for the last layer of the Feedforward Neural Network would be a linear function with single node (since it's regression). We trained different Feedforward Neural Networks for different options and situations. First we traind three FNN's with same activation function for all hidden layers, One FNN which all hidden layers are relu, one with tanh and another one with sigmoid. Next we designed a FNN with differnt activation functions containing Relu, Tanh, Softsign and Softmax. We tried optimizers like SGD, Adam and RMSprop. After trying all these three options we concluded that the Adam optimizer works significantly better than others for our problem, so when compiling the model with keras, we set optimizer to be Adam. In order to see the predictions of our Neural Networks for 'h', we ploted 'h' vs some of our features which are shown in Figures 4, 5, 6. As we see in the figures the prediction vs. actual data curves are nearly fitted and neural networks work well.

3.1.1 Metrics and Loss vs Epoch Curve

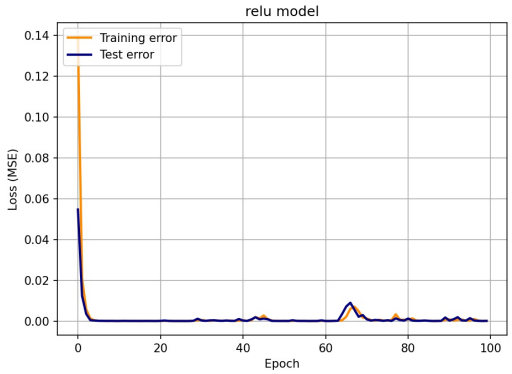
As we mentioned in the privious report, we chose MSE as the main metric for our problem so for compiling the model we set loss to be mean squared error. After triaing and testing the model we used all four main regression metrics expressed in privious report (MSE, MAE, RMSE and R^2 score) to evaluate the prediction of our designed Neural Networks. We also plot the Loss vs Epoch curve for our train and test data using MSE as loss fucntion (Figure 3). This curve is quite informative, by looking at this curve we can see how the model is working on our test and train data and whether the model overfits or not, Fortunately we did not come across any over-fitting or under-fitting.

3.2 Classification

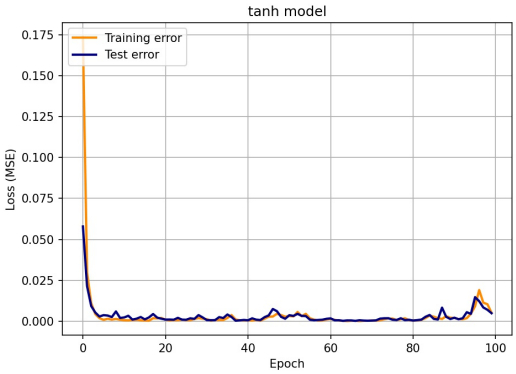
The main goal in our project was to find the 'y' lable of our data set to determine the phases of system, we treated this goal as a regressrion problem in the previous milestone.But now for the third milestone with Neural Network we decided to solve both regression and classification problems. In order to solve our problem from classification point of view we delete the phase transition point in our data set and assumed that we have only samples with $h < 1$ and $h > 1$ labels which we want to classify. For this purpose we assign 0 to all $h < 1$ samples (Ferromagnetic Phase) and 1 to all $h > 1$ samples (Paramagnetic Phase), so it's a binary classification problem. We designed a FNN with different activation functions containing Relu, Tanh, Softsign and Softmax, the optimizer was Adam and the loss function and metric were respectively binary cross entropy and accuracy. In order to see how the model worked for classification we ploted the classification results for different features which can be seen in the notebook of our code.

3.2.1 Loss & Accuracy vs Epoch Curve and Confusion Matrix

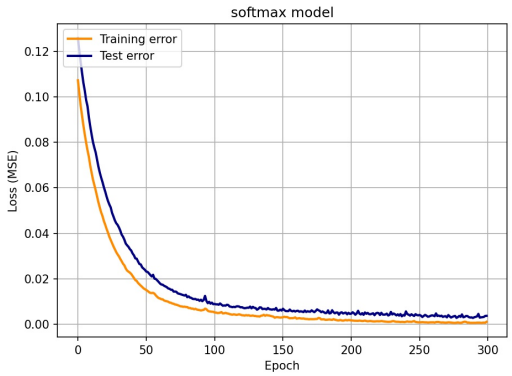
Like the regression part we also plot the Loss vs Epoch curve for out train and test data, this time using binary cross entropy loss fucntion, since we have classification we also plot the Accuracy vs Epoch plot (Figure 14) one of these plots is before tuning and the other two are after tuning, as we can see in these plots there isn't any over-fitting or under-fitting here. One of the best and most



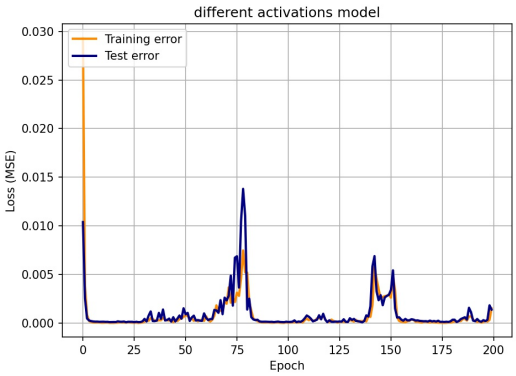
(a) Relu



(b) Tanh



(c) Softmax



(d) Mixed

Figure 3: Loss vs. Epoch Curves for Regression

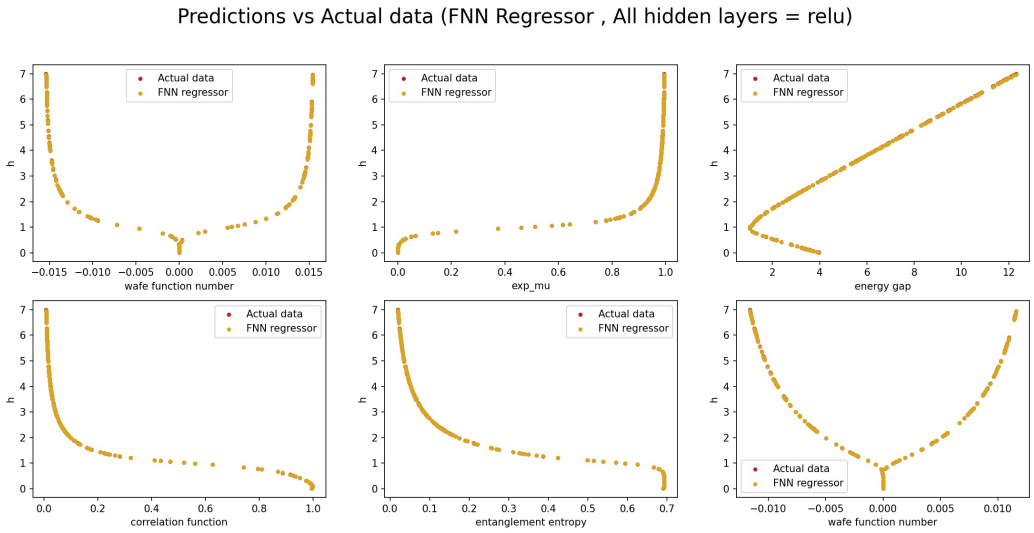


Figure 4: Actual Data vs. Prediction Data (Only Relu as Activation)

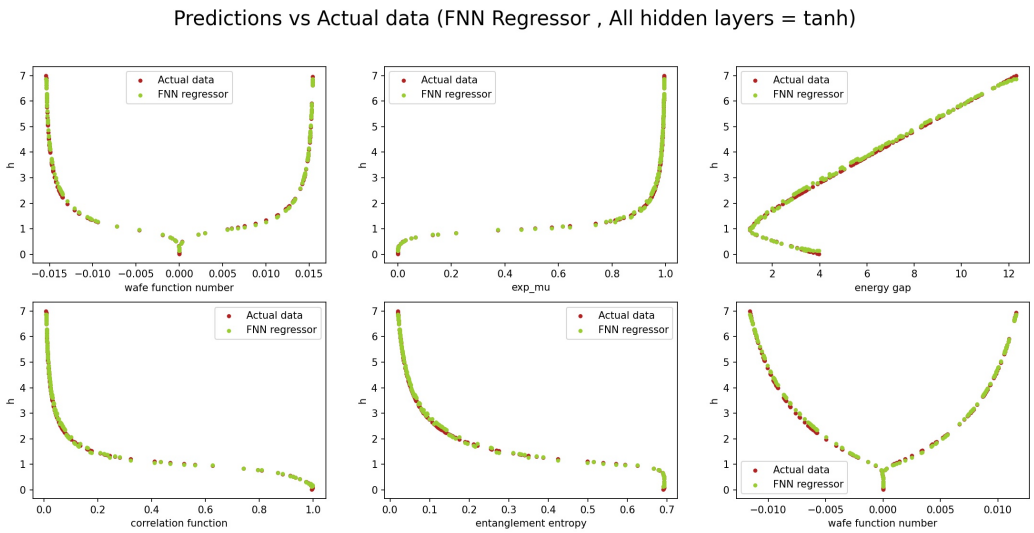


Figure 5: Actual Data vs. Prediction Data (Only Tanh as Activation)

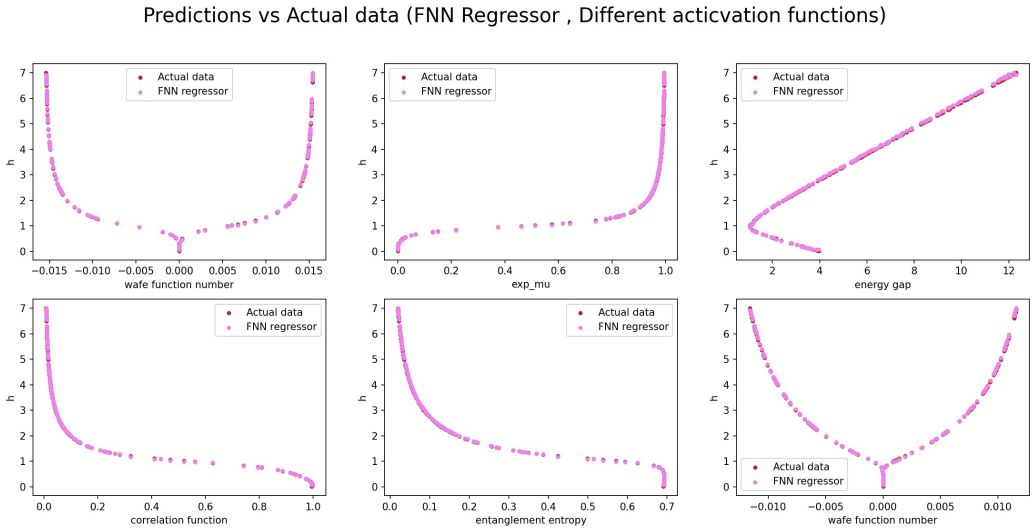
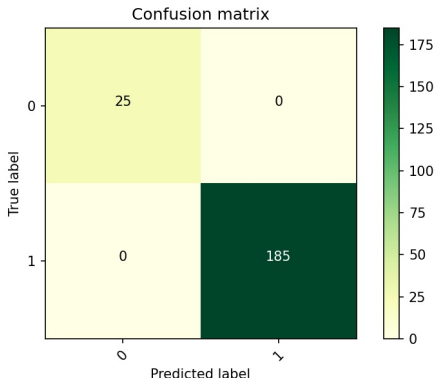
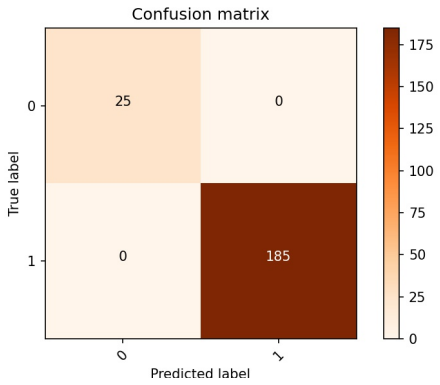


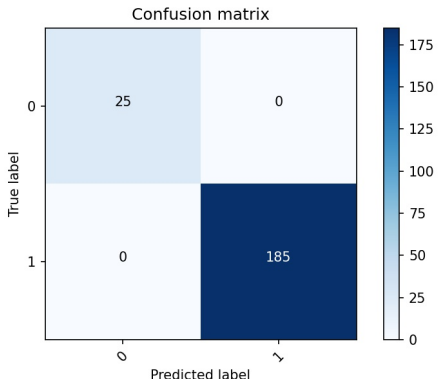
Figure 6: Actual Data vs. Prediction Data (Different Activations)



(a) FNN after tuning with Random Search Opt.



(b) FNN after tuning with Bayesian Opt



(c) FNN before tuning

Figure 7: Confusion matrices

informative classification metrics is Confusion Matrix. It is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. It gives us an insight not only into the errors being made by the classifier but more importantly the types of errors that are being made, so for this part we must include confusion matrix as an important metric, Figure 7 shows the confusion matrices for each Neural Network that we applied, one before tuning and two others after tuning. There is also a classification report for each Neural Network in the code notebook, it's a text report showing the main classification metrics such as precision, recall and f1-score.

4 Tuning (Hyperparameter Optimization)

There are some Hyperparameters in the Neural Network like Activation functions in hidden layers, Number of nodes, Number of hidden layers and Learning rate for optimizer etc. We can tune these Hyper-parameters in order to get a better prediction. For Hyperparameter Optimization we used three techniques: Random Search optimization, Bayesian optimization and Hyperband optimization. The main process of what we did in the regression problem is the following: For Random Search optimization and Bayesian optimization we had some candidates for each hyperparameter, Number of hidden layers between 2 to 10, Number of nodes between 20 and 220, Learning rate to be 0.1 or 0.001 or 0.0001 and Activation functions to be relu, softmax and tanh. Then we decided to see what will happen if our Neural Network has only one hidden layer? We used Hyperband optimization to find the best Number of nodes for hidden layer and best Learning rate, we designed three single hidden layer Neural Networks with three different activation function: Relu, Tanh and Softmax, for these three Neural Networks we also used l2 loss for regularization. For the classification problem we followed the same precedure for Random Search optimization and Bayesian optimization, we did

not use Hyperband optimization for classification part. Next we will explain briefly how these three optimization methodes work.

Random search [1] and Hyperband optimization [3] (a specific variation of random search) is a kind of optimization method that do not work with gradient and some well-known algorithms. The method is first select a random variable for cost function in the relative space and then choose another random variable and check that if the previous one is larger or not and by repeating this method, one may select the best value. But the Bayesian optimization [2] uses Bayesian rule for the base of the procedure. In a short description, it tries to minimize some cost function using prior distribution.

If we tune our neural network by applying these three optimizations, the loss vs. epoch curves for regression (Figure 8) show that for each one, there are some fluctuations impact on the loss function. But since these fluctuations are near zero, there is nothing problematic to conside.

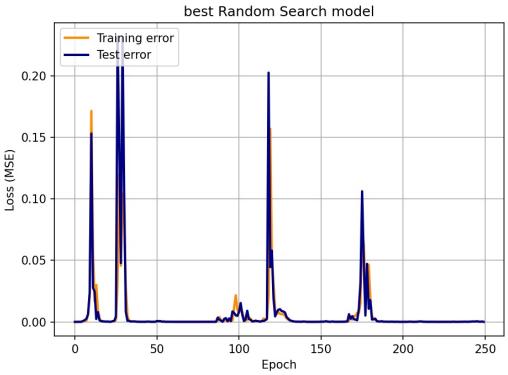
Finally, for regression problem we can compare the actual data with predicted values by our models after tuning. (Figure. 9, 10, 11, 12, 13)

5 Conclusion

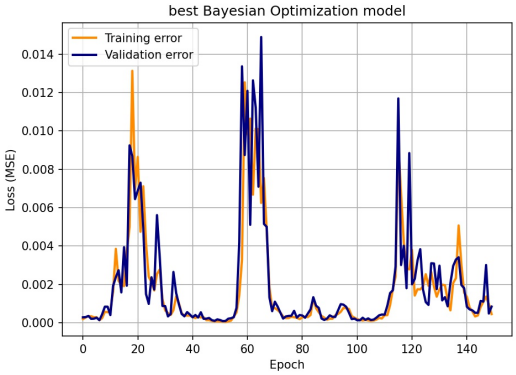
We are looking for the best and fastest neural network for our probelm. From the following table we can see that if we want to consider the overall error, the Neural Network which is tuned with Random Search Optimization is the best model. As we can see although the fit time of this model is larger than most of other models but the MSE is about $4.7497e-05$ and so it is very optimal Neural Network and it is as good as the best traditional models like GPR and KRR that we used in the privious part of this project. If we concentrate on the prediction of models for phase transition point (h_{PT}) which is one of our important aims, we can see that the model which is tuned by Bayesian optimization is the best for predicting this particular point because it is near to the exact theoretical value ($h = 1$). As we expected the Neural Networks with single hidden layer are not performing well in compare to others although they're tuned with Hyperband Optimizer.

Feedforward Neural Network	MSE	h_{PT} Prediction	h_{PT} MAE	Fit time	Predict Time
All hidden layers Relu	1.22e-04	1.0086	8.6e-03	10.9 s	102 ms
All hidden layers Tanh	4.87e-03	1.0151	1.5e-02	10.2 s	102 ms
All hidden layers Softmax	3.79e-03	0.9882	1.1e-02	21.1 s	81.5 ms
Different hidden layers	1.39e-03	1.0114	1.1e-02	20.5 s	103 ms
Random Search Opt.	4.74e-05	0.9921	7.9e-03	55 s	48.4 ms
Bayesian Opt.	8.39e-04	1.0023	2.3e-03	13.2 s	95.2 ms
Hyperband Opt. (Relu)	2.13e-02	1.0305	3.05e-02	52 s	77.8 ms
Hyperband Opt. (Tanh)	7.73e-03	0.9541	4.5e-02	61 s	67.2 ms
Hyperband Opt. (Softmax)	2.13e-02	1.0306	3.06e-02	55 s	48.4 ms

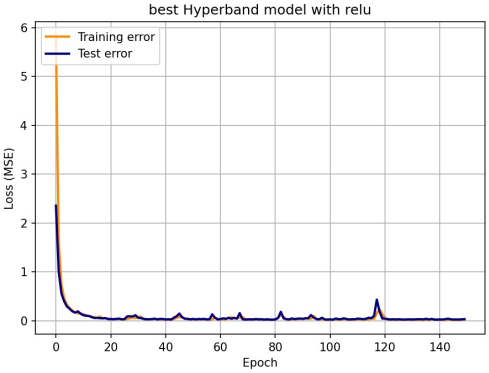
The reason that some models gave us a better prediction for Phase transition point although they have larger MSE and vice versa is the fact that MSE is calculated by averaging over all 701 samples but Phase transition point is only one sample in our data and it may be predicted more precisely by a model with larger MSE. For general conclusion on models we think that the MSE is a much better metric to look at than the Phase transition point, since the Phase transition point (h_{PT}) maybe only accidentally predicted better in a particular run of the model while that model is not working very well on the overall regression problem of predicting h and determining phase of the system. One may ask, why we have this difference between the numerical and theoretical solutions? The answer is that in theoretical case, we assume that the number of atoms are infinite but in numerical way we considered only finite numbers of spins (12 atoms) and this leads us to this difference in our results.



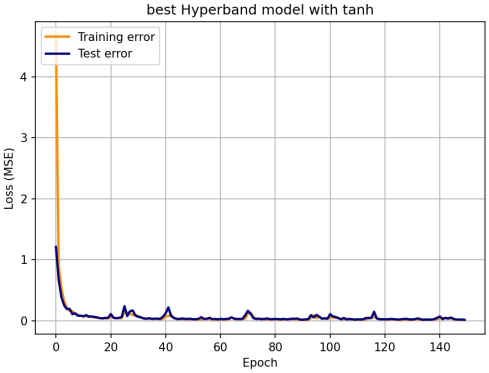
(a) Random Search Optimization



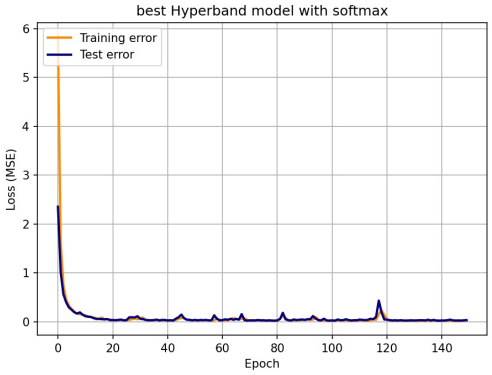
(b) Bayesian Optimization



(c) Hyperband Optimzation (Relu)



(d) Hyperband Optimization (Tanh)



(e) Hyperband Optimization (Softmax)

Figure 8: Loss vs. Epoch Curves for Regression

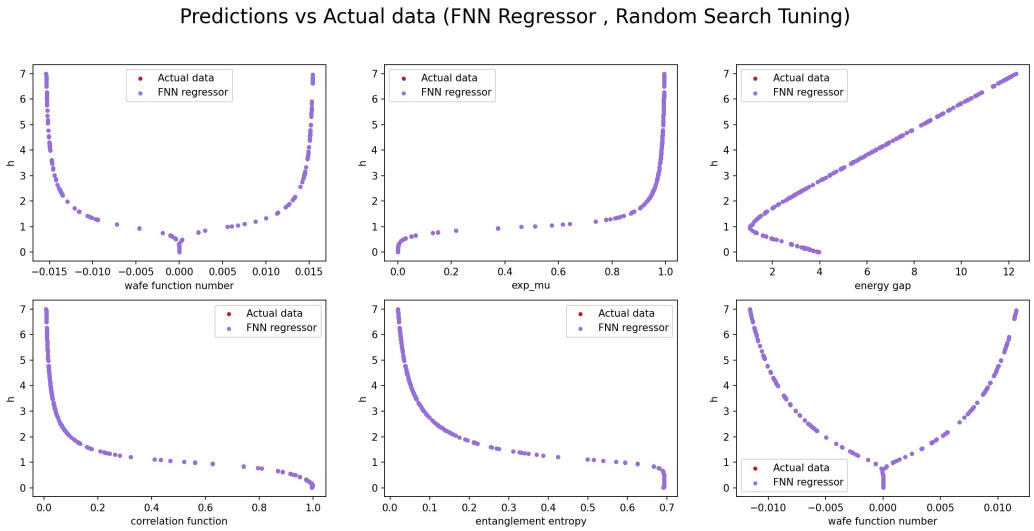


Figure 9: Actual Data vs. Prediction Data (Random Search Optimization)

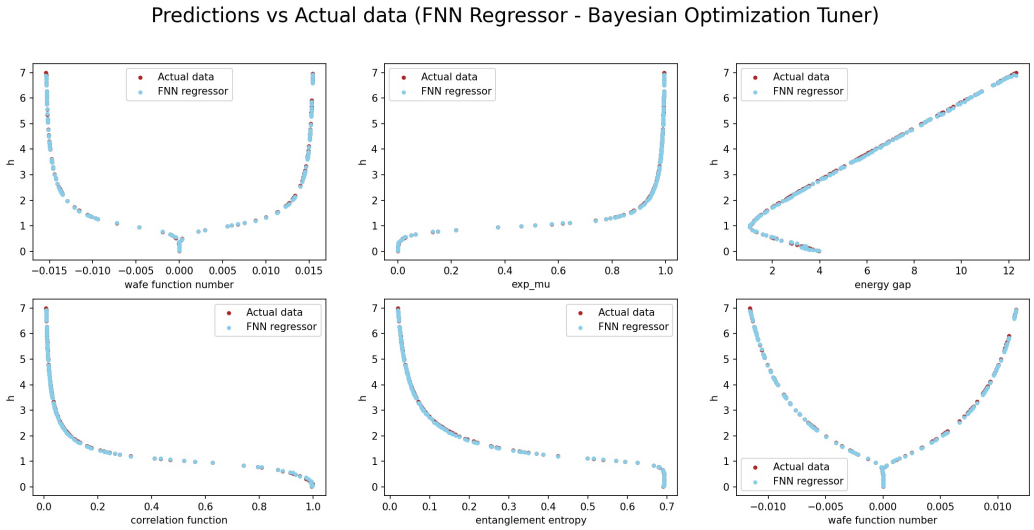


Figure 10: Actual Data vs. Prediction Data (Bayesian Optimization)

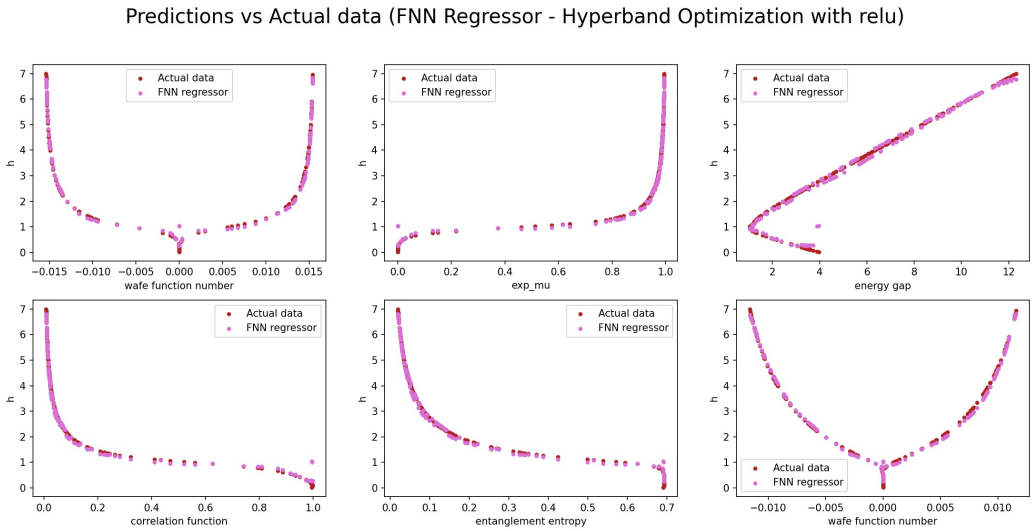


Figure 11: Actual Data vs. Prediction Data (Hyperband - Relu)

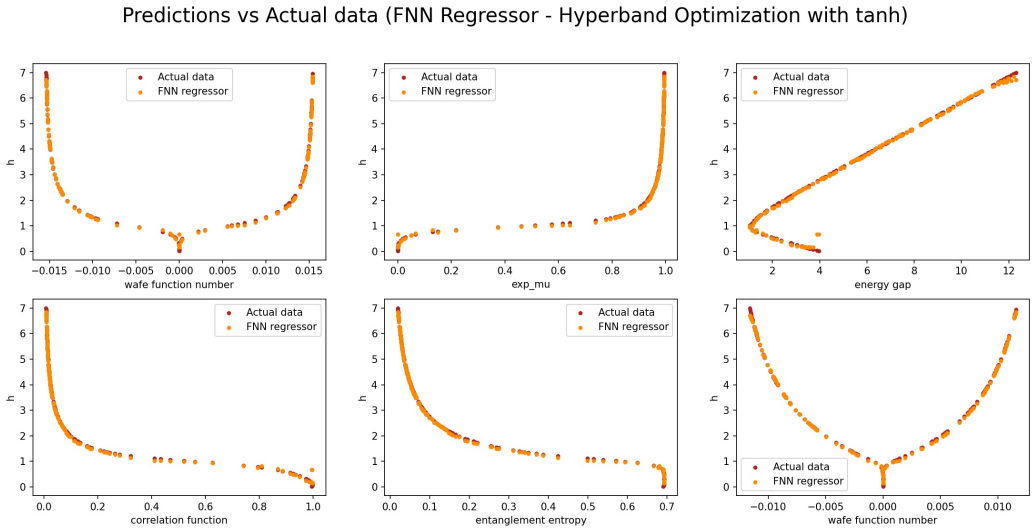


Figure 12: Actual Data vs. Prediction Data (Hyperband - Tanh)

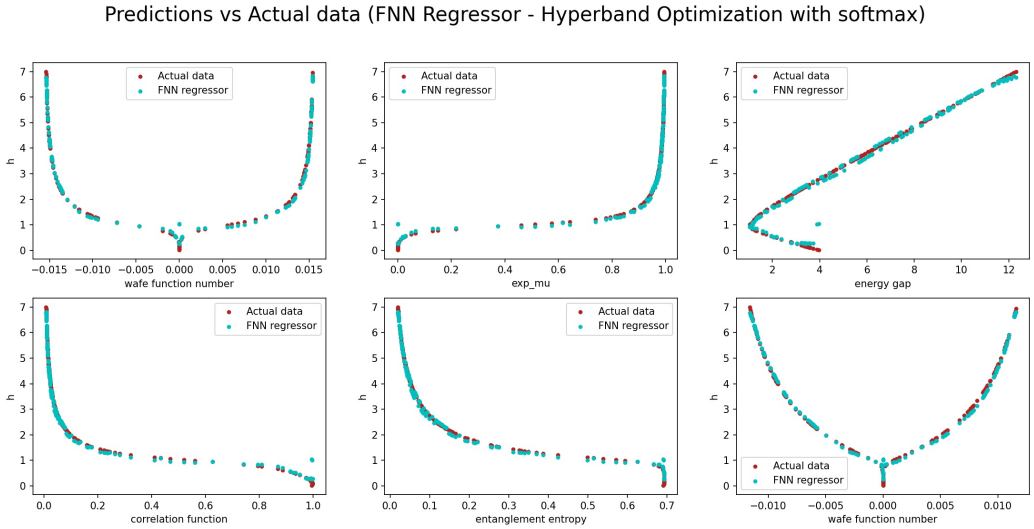
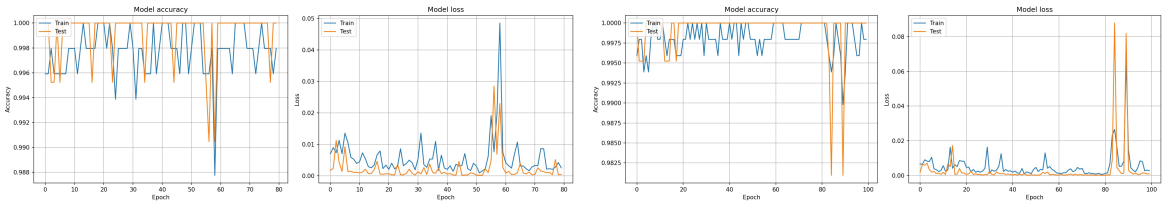
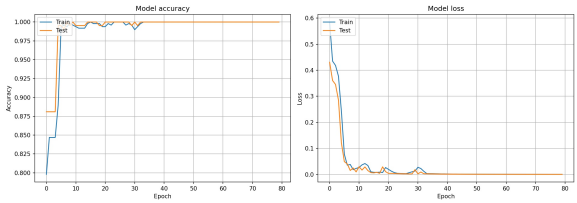


Figure 13: Actual Data vs. Prediction Data (Hyperband - Softmax)



(a) FNN after tuning with Random Search Opt

(b) FNN after tuning with Bayesian Opt



(c) FNN before tuning

Figure 14: Loss& Accuracy vs Epoch Curve for Classification

References

[1] M. Schumer, K. Steiglitz, *Adaptive Step Size Random Search.* IEEE Transactions on Automatic Control, Vol 13, Issue 3 (1968)

[2] Jonas Mockus, *Bayesian Approach to Global Optimization (Theory and Applications).* Kluwer Academic Publisher (1989)

[3] Lisha Li, et. al., *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.* Journal of Machine Learning Research, Vol 18, p.1-52 (2018)