Summary

Lorem ipsum

1 Introduction

2 Gross Pitaevskii Equation

2.1 Types of Potentials

2.2 Analytical Solution and Approximation

2.3 Numerical Solution and XMDS Framework

3 Problem Statements and Dataset Generation

3.1 Dataset and Dataset Generation

As mentioned, we solved NLSE for four different interaction parameters, therefore; there are four different corresponding datasets and each of them contains 10.000 elements. We used 8500 of them as training examples and 1500 of them for test. An element of a dataset involves an array containing potential values respect to the position and an another array containing interaction, kinetic, potential and total energy values respectively.

The array containing potential energy is generated according to harmonic trap expression in XMDS, thus; **randomness** of the dataset occurs in angular frequency and shift of equilibrium point. We supplied these random angular frequency and shift values to the XMDS within predetermined limits. Angular frequency can take values between 0.5 and 2. Shift of the equilibrium point is determined due to boundary conditions since **density function** must be zero at boundaries. In numerical solution of the NLSE, domain of the **transverse dimension** is between -10 to 10. Taking the maximum magnitude of the shift as ∓ 5 enough to ensure that density function goes to zero at infinity.

4 Machine Learning for NLSE

We used Pytorch Framework to build our neural networks. It allows the client codes work on both CPU and GPU via its internal python object called Tensor. If any CUDA supported GPU is available then Pytorch can use GPU without any change in the code. Code have three main parts, first one is dataloaders; it reads train and test data for specified interaction parameter from corresponding file and generates tensor dataset object. **continue**

We implemented two different types of neural network; feed forward (FNN) and convolutional neural network (CNN).

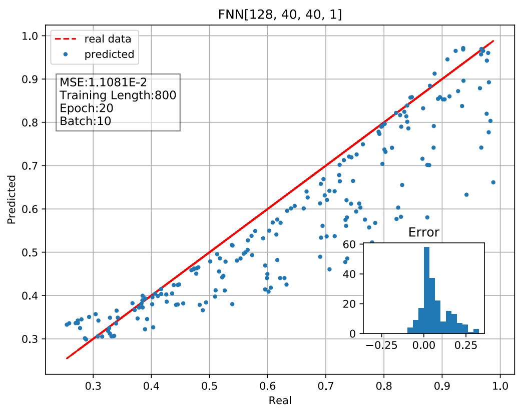
4.1 Architecture

FNN involves 128 input neurons as input layer, next layer is first hidden layer with 30 neurons, the second is same as first hidden layer, the next one is last hidden layer with 10 neurons and the last layer is output layer. Totally there are 5 layers in our FFN and it will be denoted as $FNN[128, 30, 30, 10, 4]$. Rectified linear unit (ReLU) is used for each forward except output. No operation is applied to the output neuron. Learning rate of the FNN is fixed and it is 0.001. Cost function is mean squared error (MSE) and optimization is done with Adam.

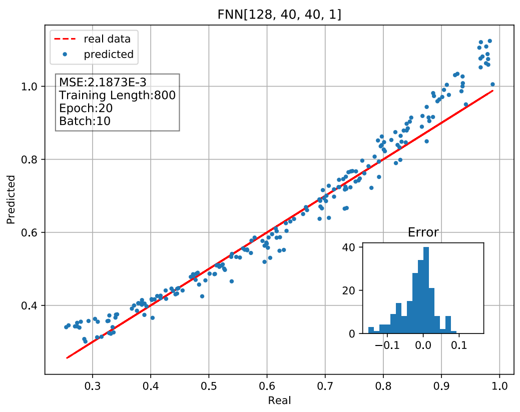
CNN has two convolution layers, two maxpool layers and three fully connected layer and last layer of the fully connected part is output layer. Maxpooling is applied to output of the first and second convolution layers. ReLu is also applied each forward except output neuron same as in the FNN. Fully connected part of the CNN is $FNN[310, 100, 20, 1]$. Learning rate, cost function and optimizer of the CNN is same as FNN which are 0.001, MSE and Adam respectively.

4.2 Hyperparameters

We firstly started with smaller dataset which involves 800 elements for trainig 200 for test to see response of the architecture and to obtain a range for learning rate.

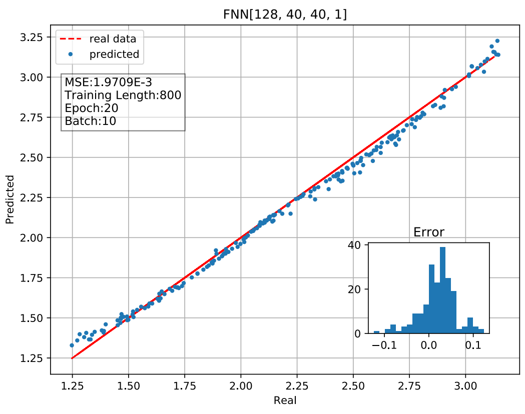


(a) FNN[128, 40, 40, 1], lr = 0.003

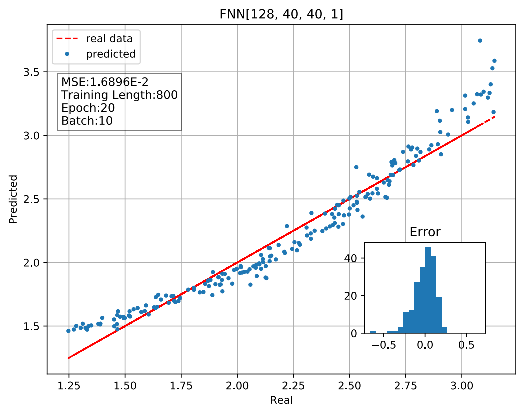


(b) FNN[128, 40, 40, 1], lr = 0.001

Figure 1: In this example interaction parameter g, is set to zero. Even learning rate 0.001 seems more accurate, it is trivial to expect that change in interaction parameter will effect the result.



(a) FNN[128, 40, 40, 1], lr = 0.003

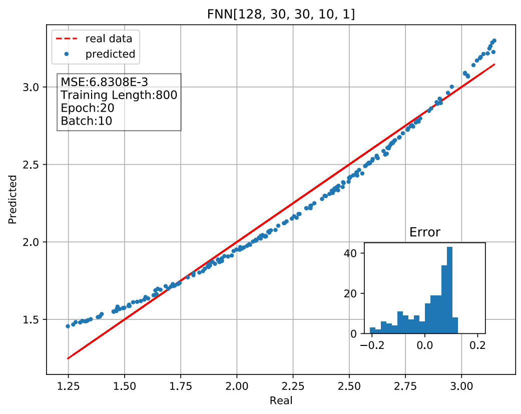


(b) FNN[128, 40, 40, 1], lr = 0.001

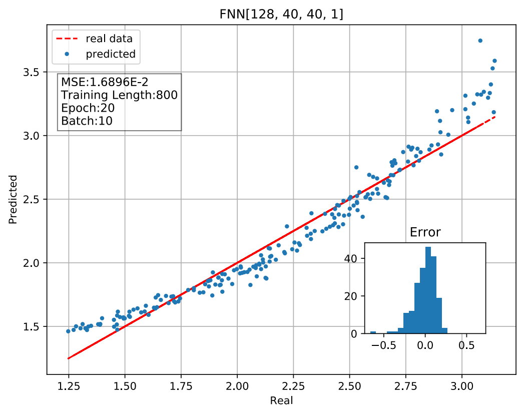
Figure 2: Here, interaction parameter, g is 10. Predictions of the network with learning rate 0.003 more accurate than 0.001.

comment about range of the learning rate ?

First architecture we used had only 4 layers and it was not sufficient.



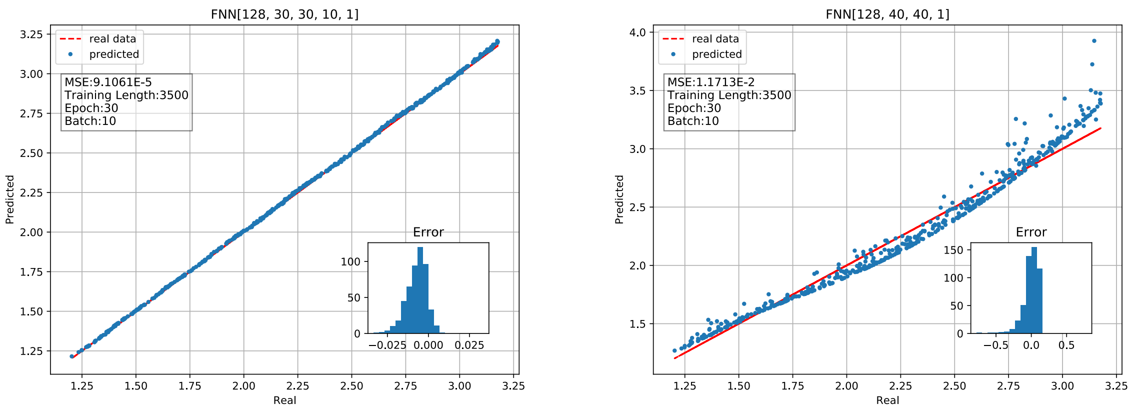
(a) FNN[128, 30, 30, 10, 1]



(b) FNN[128, 40, 40, 1]

Figure 3: Here interaction paramater g, is again set to 10. Batch size is 10 and total number of epoch is 20 for this dataset.

Then we increased the number of layers to 5 and tried different number of neuron combinations. Each of these tried network behaved different for different g values. For example, MSE of the FNN[128, 40, 40, 20, 1] for g = 10 is $5.11x10^{-3}$ but for g = 0 it is $2.00x10^{-3}$



(a) FNN[128, 30, 30, 10, 1] (b) FNN[128, 40, 40, 1]

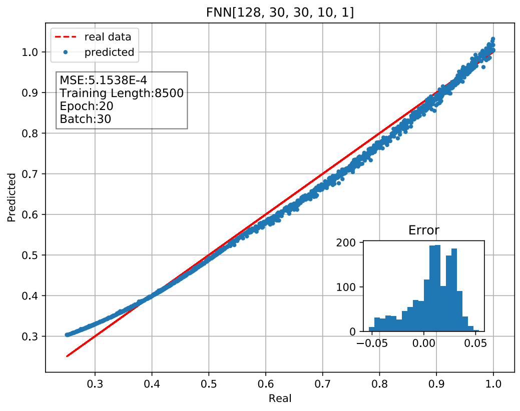
Figure 4: 3500/500

After narrowing the number of candidate architectures with small dataset, we increased the size of the dataset to 3500/500 to get more information about distribution of the predictions and to determine other two hyperparameters; mini batch size and epoch. **Add results of this arch**.

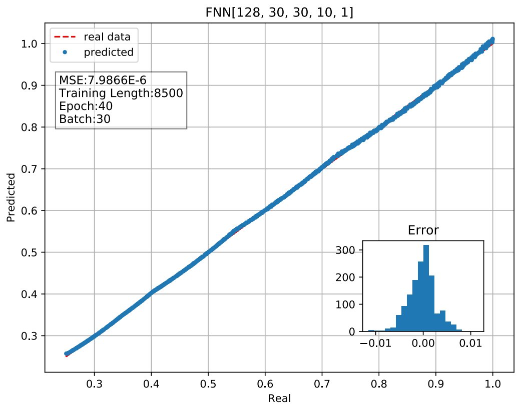
4.3 Training Results

4.3.1 Non-interacting System

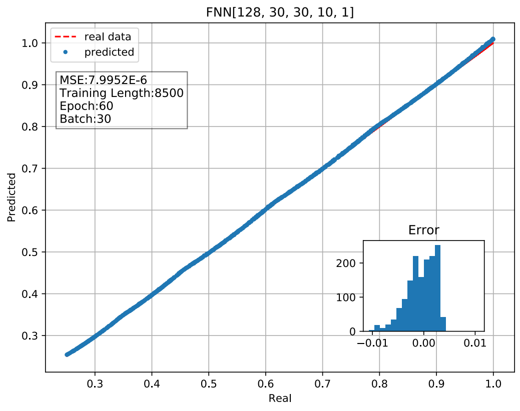
In non-interacting system, GPE reduces to SE which does not involve non-linear term.



(a) FNN[128, 30, 30, 10, 1]



(b) FNN[128, 30, 30, 10, 1]



(c) FNN[128, 40, 40, 1]

Figure 5: Since the system has no interaction parameter, problem is relatively easy when it is compared to systems that involve interaction. In (a) it is obvious network requires more training example and it reaches a satisfactory level in (b). Supplying 20 more epochs nearly does not effect the prediction accuracy.
asdasd asda

4.3.2 Interacting Systems