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 transverse dimension and a number which is total energy value.

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 in predetermined limits. Angular frequency can take values between 0.5 and 2. Shift of the equilibrium point is determined due to boundary conditions since potential must be zero at infinity. 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 potential 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 we will denote as $FNN[128, 30, 30, 10, 1]$. 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 which is a single neuron. 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 optimum learning rate. The first architecture we used had only 4 layers and it was not sufficient. **(Add result graphs of this architecture and comment about it)** Then we increased the number of layers to 5 and tried different number of neuron combination. **Add results of FNN[128 40 40 10 1] and comment about decrement of the total neuron number**