Applying Machine Learning Methods to Laser Acceleration of Protons: Lessons Learned from Synthetic Data

Presented in Partial Fulfillment of the Requirements for graduation "with Honors Research Distinction in Physics" in the undergraduate colleges of The Ohio State University

by

Tom Zhang

The Ohio State University

April 2023

Project Advisor: Professor Chris Orban, Department of Physics

Abstract

Neural networks and other machine learning algorithms are beginning to be used in ultra-intense laser physics. Compared to other machine learning applications, ultra-intense laser systems are typically data poor because of limitations on the number of shots per second. An important concern is that the machine learning method being used remains accurate even when trained on a relatively small number of data points. By using synthetic data based on a model proposed by Fuchs et al. 2006, we seek to explore the speed and accuracy of three different machine learning methods including a neural network with two hidden layers. The results of this project can potentially be applied to training neural networks on real experimental data sets.

Contents

1 Introduction

The field of ultra-intense laser physics features many non-linear physical systems in which it is either difficult to analytically solve, or computationally expensive to simulate. As such, laser physicists are increasingly turning to machine learning techniques to optimize their laser systems. For example, a recent paper by Loughran et al. (2023) used Gaussian Process regression to conduct Bayesian optimization to optimize the maximum proton energy of their laser pulses automatically[1].

There are questions, however, over how easily machine learning models can be applied to laser systems in general. While Loughran et al. saw success in using machine learning to optimize their laser, the laser system that Loughran et al. used had operated at 1 Hz. Laser systems with kHz repetition rates may produce more data than some machine learning methods may be able to process in quasi-real time.

Furthermore, there are many different machine learning models that can be utilized. Bethke et al (2021), for example, had used invertible neural networks to predict the result of Laser-Wakefield Acceleration simulations[2]. Comparing the performance of different machine learning models with each other may better aid laser physicists into selecting the appropriate machine learning model given their needs. Thus, the goal of this thesis is two-fold: We will use a simple, fully connected neural network and explore its performance given different constraints to its training length and data set size, and we will also compare the performance of this neural network to other machine learning models.

2 Experimental Description

2.1 The Dataset

We use synthetic data that was primarily based on a physical model introduced by Fuchs et al. (2005)[3]. Some modifications to the relationship between the laser period and acceleration timescale for protons were introduced to the model based on the work done by Djordjevic et al (2021)[4] and to match the model to the experimental results found by Morrison et al. (2018)[5] In this model, we have five input parameters: Laser intensity, pulse duration, target thickness, focal distance, and spot size, which combine to result in the maximum proton energy as well as the number of protons generated. This model can be further extended to calculate the average energy per proton, as well as the total proton energy generated from the laser interaction.

The synthetic data is based on a laser that can generate a laser with a maximum energy of 10 mJ, a pulse duration of 40 femtoseconds, a spot size of 2 microns Full-Width Half-Max, a minimum target thickness of 0.45 microns and a maximum target thickness of 15 microns, a minimum possible laser intensity of 10^{17} W cm^{-2}, a maximum possible laser intensity of 10^{19} W cm^{-2}, and a maximum focal distance offset of 30 microns.

2.2 Network Architecture

For our neural network, we'll be using a fully connected neural network with two hidden layers. The first hidden layer will have 64 nodes while the second hidden layer will have 16 nodes. Our neural network will be optimizing for the mean squared error. The PyTorch library was used to implement this neural network. Figure 1 depicts our neural network's architecture.

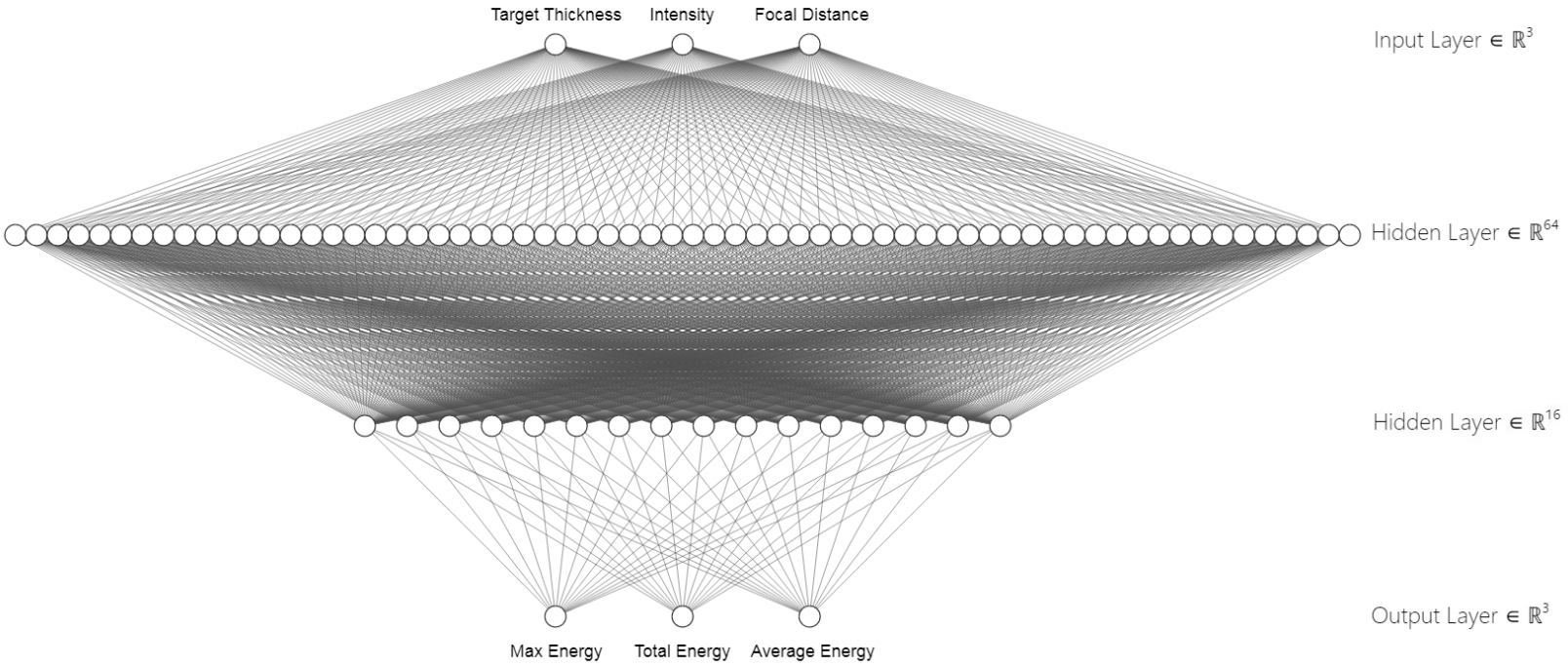

Figure 1: Architecture for our neural network

2.3 Neural Network Optimization

We first train our neural network for 40 epochs on varying amounts of data points from our synthetic data set before plotting the average relative error of our three outputs for the neural network for the testing portion of our data points. The purpose of this is to allow us to observe how much improvement our neural network gains as we add more data points. Because training a neural network is computationally expensive, it is important to examine whether training on additional data points is giving diminishing performance gains. Average relative error is used rather than mean squared error as average relative error is a more intuitive metric to gauge a neural network's accuracy compared to mean squared error.

Once we have determined the optimal number of points to train our neural network with this method, we now train our neural network again using this optimal number of points. This time, however, we will train our neural network on differing numbers of epochs before plotting our neural network's accuracy for each of these epochs. The number of epochs a neural network is trained for is also of interest for us, as if a neural network is trained for too few epochs, the neural network would have poor accuracy for our data. Training a neural network for too many epochs, however, is both computationally expensive and can lead to the neural network overfitting to the training data rather than discerning general patterns within the training data.

2.4 Neural Network Comparison with Other Machine Learning Methods

We'll be comparing our neural network's performance to two different machine learning algorithms: A Support Vector Regression (SVR) and a Gaussian Process regressor (which is considered to be a Bayesian method). In contrast to a neural network, both of these machine learning algorithms first apply what's known as a "kernel" function to map data

to a higher dimensional space. The SVR uses the kernel function in an attempt to make our data linearly separatable while the Gaussian Process uses the kernel function to represent the similarity between two data points.

For each machine learning model, we will train them on a varying number of data points from our data set, starting from 5000 data points and ending with 19,000 data points. We use less data points for this experiment compared to when we were optimizing our neural network due to memory limitations as a result of our Gaussian process model. While there are Gaussian process models such as Stochastic Variational Gaussian Process[6] which are supposed to be used to mitigate the space requirements of Gaussian process, these models operate by determining a subset of the data set which can represent the data set as a whole, and training the Gaussian process model on that subset, making it difficult for us to properly compare this version of Gaussian process to the other machine learning models in terms of time and space complexity relative to data set size. We also include a 10% Gaussian Noise to our data set for this part to better simulate uncertainty in measurements.

For each set of data points we train our three machine learning models on, we will evaluate our machine learning models on three metrics: Mean Squared Error for Max Proton Energy, Average Percent Error for Max Proton Energy, and Training Time in seconds. We use the first two metrics to evaluate our machine learning models to determine how accurate they are, while the third metric is used to determine how fast these machine learning models work relative to each other.

To ensure that any differences in performance between the machine learning models are due solely to the algorithms rather than the machine they are running on, all machine learning models were trained on the same machine. For this study, we used the Pitzer cluster from the Ohio Supercomputer Cluster which uses a NVIDIA Volta V100 GPU, while the Pitzer cluster also has a dual-GPU configuration, we only used a single GPU for our study as the library we used to implement our SVR only supports a single GPU. Our SVR was implemented by the cuDF library while our Gaussian Process algorithm was

implemented through the GPFlow library. Both of these libraries along with the PyTorch library allow for GPU acceleration, enabling a fairer comparison between our machine learning models in terms of time spent. We also record the max memory allocated for the GPU for each machine learning model to gain a better idea of how much memory each machine learning model requires.

3 Results

3.1 Neural Network Performance

For the measurement of our neural network performance, we measured the average relative error of our neural network for each set of points. We empirically observe that the average relative error begins to converge at a different value for each of our three predicted outputs beginning at 20,000 points. Training a neural network with more points costs more time, so it is important to determine when we're getting diminishing returns for giving our neural network more data points to train on. We determined that 20,000 points is the optimal size for the data set we use for training our neural network (See figure 2).

Figure 2: Plot of average relative error of testing data for the three outputs on a neural network trained for 40 epochs for different amounts of training data

After determining the optimal number of points, we then attempted to determine the optimal number of epochs to train our neural network on. As expected, when we train for only a handful of epochs, the accuracy of the neural network predictions is very low. As seen from figure 3, the average relative error for all predicted outputs fell below 10% by the time we were training with 30 epochs, We observed that at 45 epochs, the average error began to increase again. This is more clearly seen from figure 4. This suggests that we are beginning to overfit at 45 or more epochs.

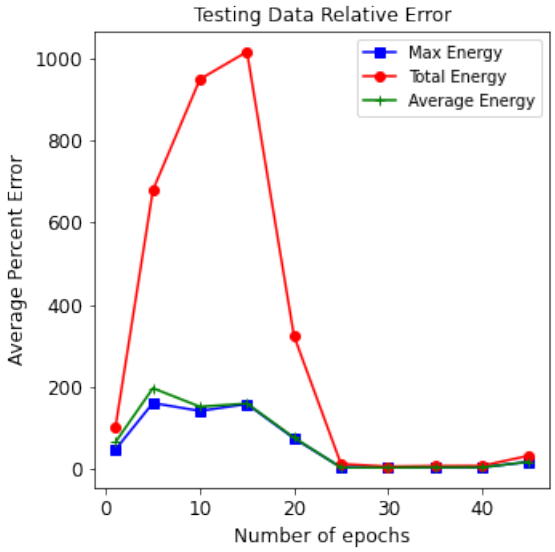

Figure 3: Plot of average relative error of testing data for the three outputs on a neural network trained on 20,000 data points for different epochs

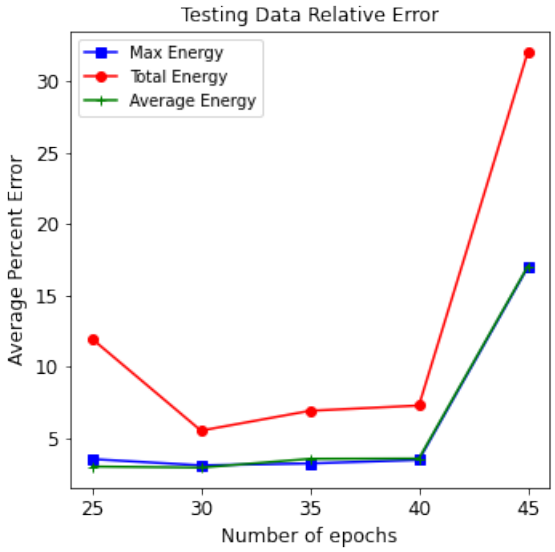

Figure 4: Plot of average relative error of testing data for the three outputs on a neural network trained on 20,000 data points for different epochs for epochs 25-45 for a better view of the average percent error

3.2 Performance Compared to Other Machine Learning Algorithms

As seen from figures 5 and 6, our neural network is generally more accurate compared to a support vector regression using both our MSE and our average percent error accuracy metrics, but less accurate in most cases compared to our Gaussian process algorithm. The accuracy of both our Support Vector Regression and Gaussian process algorithm remains generally much more stable relative to our neural network.

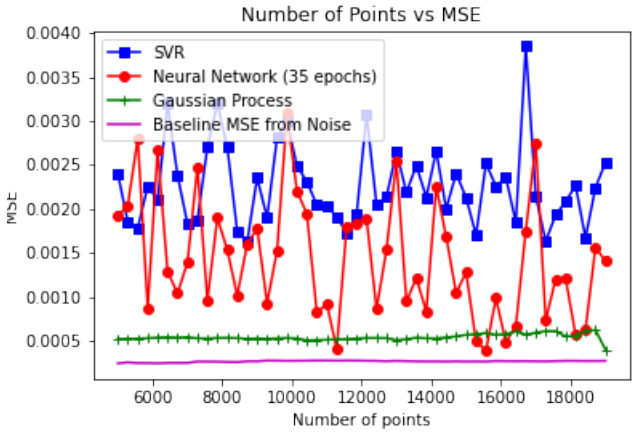

Figure 5: Mean squared error comparison for max proton energy between SVR, Gaussian Process, and our Neural Network. The MSE between the noisy dataset and the dataset's real values is indicated by the magenta line.

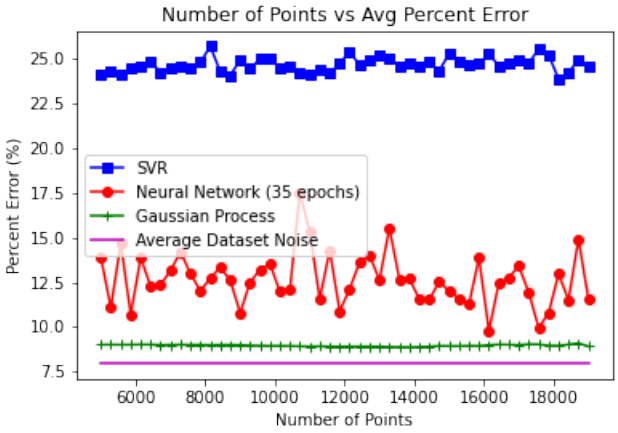

Figure 6: Average relative error comparison for max proton energy between SVR, Gaussian Process, and our Neural Network

While our neural network is overall more accurate than SVR, the time it takes to train a neural network is also consistently longer than a SVR, as seen from figure 7. Similarly, while the Gaussian process is much more consistently accurate than a neural network, we can see that the Gaussian process algorithm takes much more time to finish training compared to the neural network. Both the neural network and SVR appear to scale linearly with the training data set size while the Gaussian Process algorithm appears to scale quadratically with the training data set size. Interestingly, the MSE for both our neural network and the SVR are roughly equal but the average relative error, however, is much smaller for the neural network.

We also found, contrary to our expectations, the cheapest of the three machine learning models in terms of memory allocation, allocating only 1.18 gigabytes of memory compared to the 3.025 gigabytes of memory that SVR utilized and the 16.42 gigabytes of memory that Gaussian process used.

One interesting thing to note is that for a low number of data points, a Gaussian process model appears to be both more accurate and faster relative to a neural network as better seen with figures 8 and 9. This suggests that it may perhaps be more effective to use a

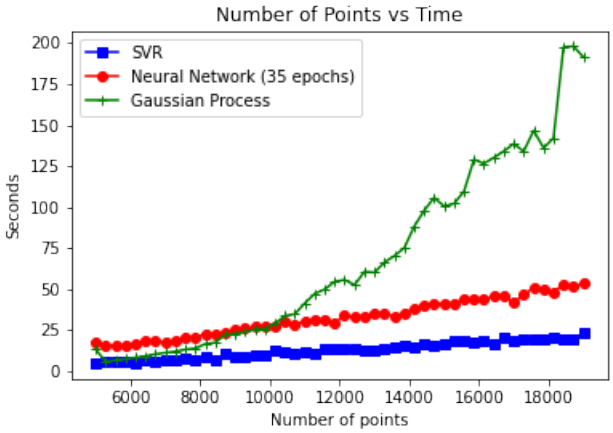

Figure 7: Comparison between SVR, Gaussian Process, and our Neural Network in terms of computation time

Table 1: Memory Metrics for each Machine Learning Model

	SVR	Neural Network	Gaussian Process
Max GPU Memory Allocated (Gigabytes)	3.025	1.18	16.42

Gaussian process model when we expect to have a relatively low number of data points, and to swap to a neural network machine learning model as our data set grows in size.

Another discrepancy we note but are unable to explain is the fact that the time complexity of a Gaussian process scales quadratically with data set size as previously noted. This is notable as theoretically speaking, a Gaussian process model should scale cubically, as a Gaussian process model typically has a $\mathcal{O}(n^3)$ running time[7]. It is possible, however, that the implementation GPFlow uses for a Gaussian process model uses some approximations to achieve better performance. In fact, as Tsang et al. (2005) [8] notes, a similar phenomenon has been observed for Support Vector Machines where the theoretically running time is $\mathcal{O}(n^3)$ but state-of-the-art algorithms have achieved running times between $\mathcal{O}(n)$ and $\mathcal{O}(n^{2.3})$, often through approximating the optimal solution rather than achieving

the optimal solution exactly[8].

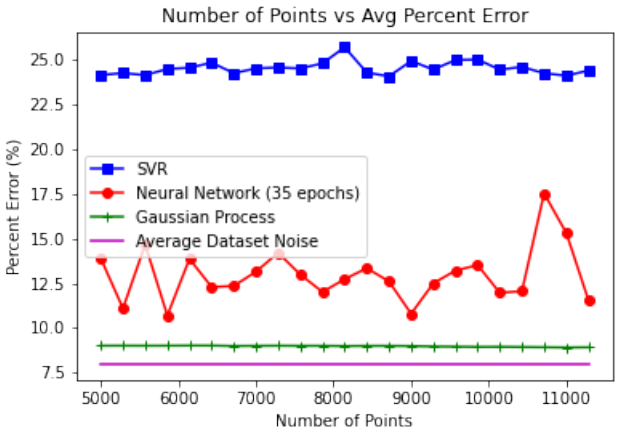

Figure 8: Average relative error for max proton energy comparison between SVR, Gaussian Process, and our Neural Network for low number of data points

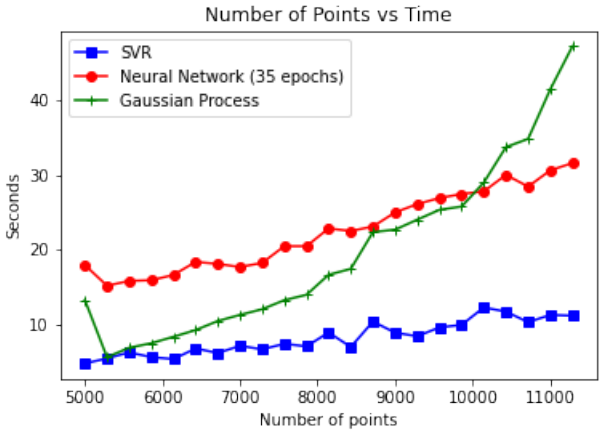

Figure 9: Comparison between SVR, Gaussian Process, and our Neural Network in terms of computation time for low number of data points

4 Conclusion

In conclusion, we found that out of our three machine learning models, Gaussian process was the most accurate. However, Gaussian process also used the most GPU memory and it performed much slower than the other two machine learning methods for large data sets. In contrast, a completely connected neural network with two hidden layers, though slightly less accurate than a neural network, was much faster and required less memory. This suggests that for larger data sets, a neural network would be a more appropriate choice to use as it scales well time-wise with number of data points. A future direction this work can go is to see if the results of this study hold up to real-world data.

5 Acknowledgements

I would like to thank Prof. Chris Orban for welcoming me to his research group and providing me this opportunity to work on this project. I'd also like to thank Joseph Smith and Ricky Oropeza for their aid in generating the synthetic dataset that was used for this paper. In addition, I'd like to thank Pedro Gaxiola for providing his code for implementing a GPU-accelerated support vector machine for regression for the synthetic data used for this project.

6 Works Cited

References

[1] B Loughran, MJV Streeter, H Ahmed, S Astbury, M Balcazar, M Borghesi, N Bourgeois, CB Curry, SJD Dann, S DiIorio, et al. Automated control and optimisation of laser driven ion acceleration. *High Power Laser Science and Engineering*, pages 1–11, 2023.

[2] Friedrich Bethke, Richard Pausch, Patrick Stiller, Alexander Debus, Michael Bussmann, and Nico Hoffmann. Invertible surrogate models: Joint surrogate modelling and reconstruction of laser-wakefield acceleration by invertible neural networks, 2021.

[3] J Fuchs, Patrizio Antici, Emmanuel D'Humières, E. Lefebvre, Marco Borghesi, E Brambrink, C Cecchetti, M Kaluza, Victor Malka, M. Manclossi, S. Meyroneinc, Paola Mora, Jörg Schreiber, Toma Toncian, Henri Pépin, and Patrick Audebert. Laser-driven proton scaling laws and new paths towards energy increase. *Nature Physics*, 2, 12 2005.

[4] B Djordjević, A Kemp, J Kim, J Ludwig, R Simpson, S Wilks, T Ma, and D Mariscal. Mev proton acceleration at khz repetition rate from ultra-intense laser liquid interaction. *Plasma Physics and Controlled Fusion*, 8 2021.

[5] J Morrison, S Feister, K Frische, D Austin, G Ngirmang, N Murphy, C Orban, E Chowdhury, and W Roquemore. Mev proton acceleration at khz repetition rate from ultra-intense laser liquid interaction. *New Journal of Physics*, 2 2018.

[6] J Hensman, Fusim N, and Lawrence N. Gaussian processes for big data. 9 2013.

[7] M Belyaev, E Burnaev, and Kapushev Y. Exact inference for gaussian process regression in case of big data with the cartesian product structure. 3 2014.

[8] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(13):363–392, 2005.