ORIGINAL ARTICLE

# Applying Machine-Learning Methods to Laser Acceleration of Protons: Lessons Learned From Synthetic Data

Ronak Desai[1] | Thomas Zhang[1] | John J. Felice[1] | Ricky Oropeza[1] | Joseph R. Smith[2] | Alona Kryshchenko[3] | Chris Orban[1] | Michael L. Dexter[4] | Anil K. Patnaik[4]

[1]Department of Physics, The Ohio State University, Columbus, Ohio, USA | [2]Department of Physics, Marietta College, Marietta, Ohio, USA | [3]Department of Mathematics, California State University Channel Islands, Camarillo, California, USA | [4]Department of Engineering Physics, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA

**Correspondence:** Ronak Desai (desai.458@osu.edu)

**Received:** 16 July 2024 | **Revised:** 4 September 2024 | **Accepted:** 16 October 2024

**Funding:** This work was supported by the National Science Foundation (2109222), U.S. Department of Energy (89243021SSC000084), and Air Force Office of Scientific Research (23AFCOR004).

**Keywords:** laser-driven ion acceleration | machine learning | normal sheath acceleration | optimization | target

## ABSTRACT

In this study, we consider three different machine-learning methods—a three-hidden-layer neural network, support vector regression, and Gaussian process regression—and compare how well they can learn from a synthetic data set for proton acceleration in the Target Normal Sheath Acceleration regime. The synthetic data set was generated from a previously published theoretical model by Fuchs et al. 2005 that we modified. Once trained, these machine-learning methods can assist with efforts to maximize the peak proton energy, or with the more general problem of configuring the laser system to produce a proton energy spectrum with desired characteristics. In our study, we focus on both the accuracy of the machine-learning methods and the performance on one GPU including memory consumption. Although it is arguably the least sophisticated machine-learning model we considered, support vector regression performed very well in our tests.

## 1 | Introduction

The field of ultra-intense laser science is increasingly beginning to embrace machine learning (ML) and Bayesian optimization methods [1, 2]. This is especially true as the repetition rates of ultra-intense laser systems increase and data acquisition systems improve (e.g., Heuer et al. [3]). Although there has been notable work applying these methods to laser wakefield [4–7] and free electron lasers (e.g., [8, 9]), to date, there has been limited use of these methods to enhance and control proton acceleration from ultra-intense laser systems. Recently, Loughran et al. [10] provided results from training an ML model on proton acceleration data using a laser system that operates at a 1 Hz repetition rate using Bayesian optimization. Ma et al. [11] also describe high-level ongoing efforts toward similar goals on laser systems operating at a repetition rate of a few Hz using a neural network (NN) approach.

There are ultra-intense laser systems that operate at higher than 1 Hz repetition rates, and these systems are already being used in efforts to accelerate protons. Morrison et al. [12], for example, accelerated protons to ~2 MeV energies using a few mJ ultra-intense laser pulses at a repetition rate of 1 kHz. This experiment could potentially be replicated on the many mJ class, ~kHz repetition rate laser systems that exist today (e.g., Cao et al. [13]). Even kHz-rate neutron generation has been reported recently at the Extreme Light Laboratory [14]. It is also true that future industrial or defense applications of ultra-intense laser systems will likely operate closer to this repetition rate regime [15]. Ideally, we would like to train ML models on these systems in quasi-real time. A natural question is therefore, of the ML models that are being used today, can they be quickly trained on tens of thousands of shots or more? Can this be achieved with modest computational resources such as a single GPU, or would it require a supercomputer or GPU cluster? The ability to train an ML model in quasi-real time could greatly assist efforts to optimize and/or control the properties of ion beams resulting from the laser interaction. Specifically, the ML model could help discover ways to increase the max proton energy, or it could help with the inverse problem of wanting a particular ion energy distribution and needing to know the laser parameters to produce that distribution.

In our study, we will use three different ML methods to train models using up to 20,000 synthetic data points with different levels of added noise in order to predict aspects of the ejected proton kinetic energy distribution including the peak proton energy, the total kinetic energy in ejected protons and the average kinetic energy of ejected protons. The ML models also predict the conversion efficiency from laser energy to proton energy because it is straightforward to take the predicted total energy in ejected protons and divide it by the total laser energy, which is one of the inputs. Although NNs can approximate highly complex functions, it is not obvious that this method will be more accurate than other methods because NNs require large amounts of data to train.

Our synthetic dataset is generated from a theoretical model that we modified from the theory model in Fuchs et al. [16]. Although this synthetic data model is smoother and likely less featured than real experimental data sets, it allows us to generate a practically unlimited number of data points in order to test different ML models and examine how they perform with different assumptions about the noise.

Another approach, that some groups have investigated [17–20], would be to perform large numbers of 1D PIC simulations and to use this as training data for ML models. This is very different from our approach in that the noise comes from the finite number of particles in the PIC simulations whereas in our study the noise is added intentionally to represent what a real experiment might see. Another reason why we did not perform large numbers of 1D PIC simulations is that these simulations do not (without further assumption) capture the behavior of putting the target closer or further away from peak focus, unlike our model. Placing the target off of peak focus is a potentially important strategy for controlling the proton energy spectrum (e.g., Morrison et al. [12], Loughran et al. [10]).

In §2, we discuss our modified Fuchs et al. [16] model and the noise added to it when generating the synthetic data set. In §3, we describe the three different ML models used in this study. In §4, we show our results for the accuracy of the trained models. §5 considers the computational performance and memory consumption of the trained models. In §6, we use these trained models with a simple optimization task. In §7 and §8, we summarize and conclude.

## 2 | Synthetic Data

### 2.1 | Modified Fuchs et al. Model

We extend the model (in a trivial way) by adding focal distance as an input and using the ideal Gaussian beam formula to determine the effective spot size and intensity. Had we used large numbers of 1D or even 2D PIC simulations instead of a theoretical model, a similar prescription would need to be added to model the diminished intensity of the laser pulse as the target moves away from peak focus.

As will be discussed later, in our study we kept the pulse duration and wavelength fixed. The spot size was also “fixed” in the sense that the spot size at peak focus was always the same, but for non-zero focal distance, the effective spot size on the target depends on the distance from the target to peak focus. This approach of keeping the pulse duration and laser wavelength fixed while moving the target through the focus of the laser is similar to experimental studies like Morrison et al. [12] and Loughran et al. [10], which both found interesting features in the proton acceleration results even when the target was not at the peak focus. We generate synthetic data based on a physical model described by Fuchs et al. [16] which introduces a cutoff time to the plasma expansion model by Mora [21]. This model has five input parameters: laser intensity, wavelength, pulse duration, target thickness, and spot size. This model uses empirical formulae to estimate quantities such as laser absorption and hot electron temperature.

The Fuchs et al. [16] model provides an estimate of the maximum proton energy (given in Equation (1) in the study) as

$$E_{\max} = 2T_{\text{hot}}\left[\ln\left(t_p + \left(t_p^2+1\right)^{1/2}\right)\right]^2 \tag{1}$$

where $T_{\text{hot}}$ is the hot electron temperature and $t_p$ is a normalized acceleration time directly proportional to the acceleration timescale of the protons $\tau_{\text{acc}}$. The Fuchs et al. [16] model assumes the acceleration timescale is on the order of pulse duration of the laser (i.e., $\tau_{\text{acc}} \approx 1.3\,\tau_{\text{FWHM}}$) where the multiplier was determined empirically. We made a modification to this model by increasing the multiplier from 1.3 to 4.0 in order to agree with experiments better in the intensity regime of $10^{18}-10^{19}$ W cm$^{-2}$. This modification is inspired by Djordjević, Kemp, Kim, Ludwig, et al. [22], who in their §5.3 treated the multiplier between the laser pulse duration and the proton acceleration timescale as a free parameter to be constrained by 1D particle-in-cell simulation results. Without this modification (i.e., the unmodified Fuchs et al. model), we found that the max proton energies in this intensity regime were too low compared to experiments in this intensity regime (e.g., Morrison et al. [12]).

The total and average proton energies can be obtained by integrating the number of accelerated protons per unit energy (given by Equation (2) in Fuchs et al. [16])

$$\frac{dN}{dE} = \left[\frac{n_{e0}c_s t_{\text{acc}} S_{\text{sheath}}}{(2ET_{\text{hot}})^{1/2}}\right]\exp\left[-\left(\frac{2E}{T_{\text{hot}}}\right)^{1/2}\right] \tag{2}$$

which is dependent on the surface area of the hot electron sheath $S_{\text{sheath}}$ with number density $n_{e0}$ and the proton sound speed $c_s$. In this way, the Fuchs model provides three outputs. To be clear, these are the protons accelerated by the laser interaction, not necessarily the protons in the bulk target that do not gain significant energy. Note that in computing the total proton energy and the average proton energy from the Fuchs et al. [16] model, which involves integrating $dN/dE$ in Equation (2), we assumed the minimum kinetic energy to be zero for simplicity rather than using a non-zero cutoff as they did in Fuchs et al. [16]. This choice only weakly affects the total and average proton energies when the energy cutoff is below ~50 keV. There can be situations where higher energy cutoffs would be important to consider, for example, if there is a detector that is insensitive to lower energy protons. The interested reader can modify the included code [23] to change this minimum cutoff energy if desired.

There are many semi-analytic models that exist that could have been used for this purpose (e.g., Schreiber et al. [24], Passoni and Lontano [25], Passoni et al. [26], Zimmer et al. [27]) and that are potentially more accurate than Fuchs et al. [16], especially at high intensity. We used Fuchs et al. [16] because it is well known in the field, it is relatively simple, and because we are restricting our domain of interest to protons accelerated by the Target Normal Sheath Acceleration (TNSA) mechanism (e.g., Clark et al. [28], Hatchett et al. [29], Snavely et al. [30], Passoni et al. [26]).

## 2.2 | Range of Synthetic Data Generated

We generate synthetic data based on a Ti:Sapphire kHz laser system at the Wright-Patterson Air Force Base in Dayton, Ohio that can generate 40 femtosecond pulses, a spot size of 1.8 µm Full-Width Half-Max, and target thicknesses between 0.5 and 10 µm (see Morrison et al. [12]). We generated synthetic data with total pulse energies between 1.41 and 14.14 mJ. At peak focus, these pulses correspond to intensities of $10^{18}$ and $10^{19}$ W cm$^{-2}$ respectively. The focal distance was varied between −10 and +10 µm. Consequently, the intensity on target for the generated points ranges from ~$10^{17}$ to $10^{19}$ W cm$^{-2}$.

We generated 25,000 data points by randomly sampling this parameter space. The highest proton energy in the data set was near 2.3 MeV while the lowest proton energy was 1.6 keV. The random sampling of the intensity was exponentiated uniformly, so as not to focus too much on high intensity or low intensity, while the random sampling of other parameters was uniform across the range.

We use up to 20,000 of the generated data points for training the ML models. We chose to train on a data set of this size because it is one or two orders of magnitude more "shots" than some of the more data rich experimental papers on proton acceleration that have been published in recent years (e.g., [10, 31]), but it is not such a large amount of data that it is wildly beyond the state of the art. For example, there is a 50 J class laser system that is coming online with a 10 Hz repetition rate [32] that will be able to produce 20,000 shots in about a half hour. Practical limitations, such as data acquisition and the time needed to slew the target and vary the laser pulse energy will reduce the number of truly independent shots. But even with these caveats, experimental papers with tens of thousands of independent proton acceleration shots will become possible in the near future as technology improves and new facilities come online.

As mentioned earlier, we are also interested in kHz repetition rate systems (e.g., Morrison et al. [12]) which produce data even more quickly but are constrained by the limitations mentioned. Focusing on up to 20,000 points is therefore relevant to both repetition rate regimes and it is a stepping stone toward larger data sets.

## 2.3 | Noise Model

To better represent experimental data, we added noise to all three output quantities of the model—max proton energy, total proton energy, and average proton energy. Our noise model involved making Fuchs model predictions for our parameter space and sampling from a log-normal distribution using each prediction as the mean. We generated data sets with different noise levels by assuming that the standard deviation is between 5% and 30% of the mean value. When, in later sections, we refer to a 10% Gaussian noise level, for example, this means that we generated data by sampling a log-normal distribution assuming that the standard deviation was 10% of the mean. The "Gaussian" in this context refers to the Gaussian function that underlies the log-normal distribution. Because the standard deviation is assumed to be some fraction of the mean, as the Fuchs model predictions become larger, the noise level also becomes proportionally larger.

We include with this publication the Python code that was used to generate the synthetic data and the code that was used to train the models [23].

## 3 | ML Methods

We use the synthetic data to train three different ML models. These are NN, support vector regression (SVR), and Gaussian process regression (GPR). In contrast to a NN, SVR and GPR apply a "kernel" function to map data to a higher dimensional space. In this study, we aim to make reasonable efforts to compare the methods and their performance on a single GPU. Specifically, we use an NVIDIA Volta V100 GPU on the Pitzer cluster at the Ohio Supercomputer Center with 32 GB of GPU memory.

As mentioned earlier, there are three independent variables in the data set—target thickness, intensity and focal distance. First, we applied a logarithm to both the intensity and the three outputs of the model: maximum proton energy, total proton energy, average proton energy. This was necessary due to the exponentiated uniform sampling of the intensity. Figure 1 shows that the distribution of the energies looks approximately Gaussian after applying log-scaling. Then, we used standard $z$-score normalization [33], which involves transforming the data point $x^i_j$ of element $i$ and feature $j$ by $\frac{x^i_j-\mu_j}{\sigma_j} \to \overline{x}^i_j$. Here, $\mu_j$ and $\sigma_j$ are the mean and standard deviation of given feature $j$ that are empircally determined from the dataset. This normalization was determined only from the inputs and outputs of the training set. We note that the log-scaling of the outputs introduces a bias

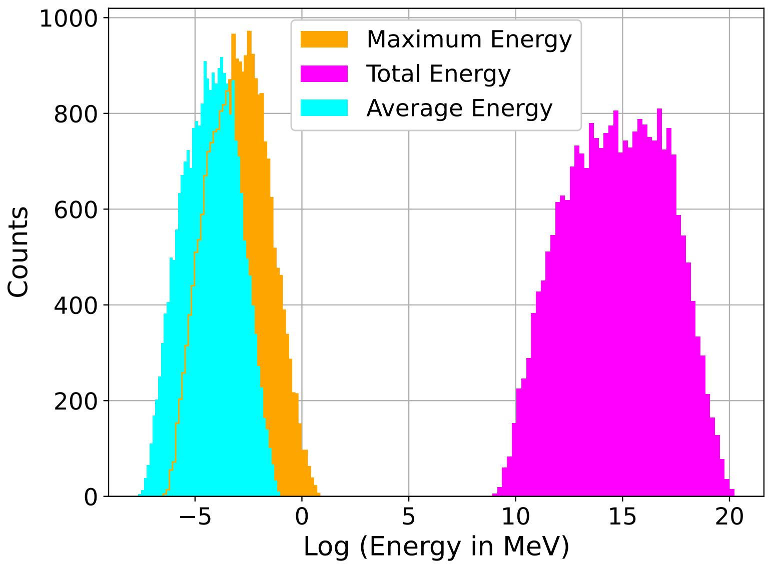

**FIGURE 1** | Histogram of the (natural) logarithm of maximum, total, and average proton energies of all points in the 20,000 point training dataset.

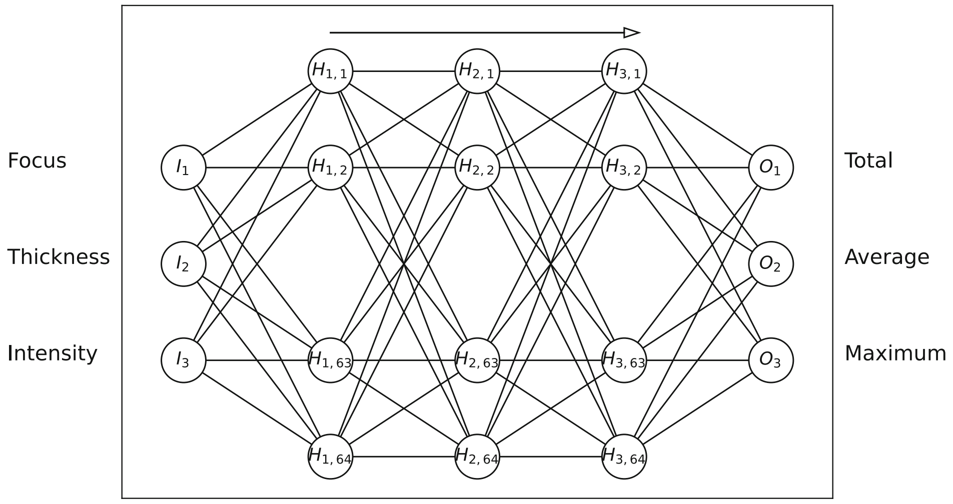

**FIGURE 2** | Architecture of the neural network model. The three inputs (intensity, target thickness, and target focal position) and three outputs (maximum, total, and average proton energy) are fully connected by three hidden layers of 64 neurons each. The middle 60 neurons in each layer are not shown so that the connections between the layers are seen clearly.

that causes under-prediction [34]. In this paper, we multiply the detransformed outputs (i.e., the unscaled outputs obtained by exponentiating the log-scaled predictions) by a correction factor equal to the mean of the training data outputs divided by the detransformed outputs.

### 3.1 | Support Vector Regression

SVR is based on support vector machines that were developed for classification problems. In SVR, one specifies a tolerance value that is used to determine which points follow the relation given by a particular curve and which points are considered outliers. This makes the task of finding the best-fit curve like a classification problem. For more information about SVR, see Smola and Schölkopf [35].

To accelerate the calculations with GPU, we used the cuML library [36], which is part of the RAPIDS package. The version for our cuML library was 22.10.01. Our investigation used an epsilon of 0.01, tolerance of 0.001, and regularization parameter $C$ of 2.5. Appendix A describes results that justify these choices of hyper-parameters. Additionally, we used the radial basis function (RBF) kernel. The $\gamma$ value is set by cuML's "scale" option, which automatically sets the $\gamma$ value equal to 1/(number of features * input variance). For more details of our specific implementation please see the attached Python file [23].

### 3.2 | Neural Network

We used PyTorch [37] version 1.12.1 to implement a NN model using a three hidden layer architecture described in Figure 2 with 64 neurons per layer. Appendix A describes results that justify our choice of using three hidden layers. We used a "leaky" ReLU activation function between fully connected nodes and trained in a batch size of 256.

We train the NN using the "Adam" scheme for back-propagation as described in Kingma and Ba [38]. We use an initial learning rate of 0.001, which the Adam scheme can increase or decrease as the model improves. 20% of the training set is used as a validation set so that training can automatically stop when the validation loss stops decreasing.

## 3.3 | Gaussian Process Regression

GPR is a Bayesian method that differs substantially from SVR and NN. The approach works by selecting a set of analytic functions that go through a series of data points. The GPR model can make data-informed predictions by taking the weighted average of these functions, with the weights being the likelihood that any one of these functions is the "true" function that correctly captures the model. A potential advantage over SVR and NN is that the GPR model uncertainty can be simply determined by the variance of these functions without any additional work. Schulz et al. [39] provide an excellent introduction to GPR.

To accelerate the computations with GPU, we used the GPyTorch library [40] version 1.9.1. As discussed in Gardner et al. [40], numerical instabilities can arise in cases with low or zero noise which can affect the accuracy of the GPR model. To avoid this we configured GPyTorch to use double-precision floating-point data and the other models were trained with double-precision floating-point data to be consistent. We used the RBF kernel for our GPR model with 30 training iterations and a learning rate of 0.2.

## 4 | Accuracy of Trained Models

Figure 3 highlights our results for the accuracy of the three different ML models in terms of the mean absolute percentage error (MAPE) for predicting the max proton energy (top left), total proton energy (top right), and average proton energy (bottom left). Each panel shows results for each ML method from three different levels of Gaussian noise.

This plot was made by comparing trained ML models to up to 5000 testing points (i.e., an 80–20 split between training and testing data) that were not included in the training data. Black lines with different line types show the MAPE between noisy and

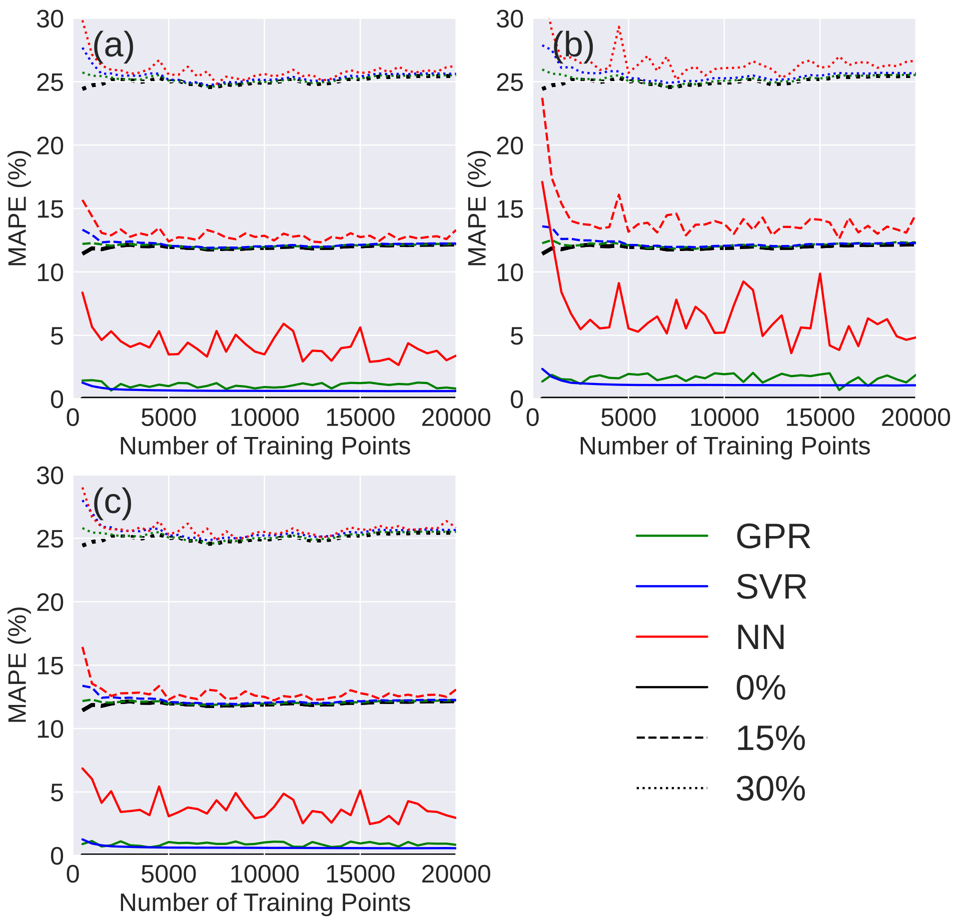

**FIGURE 3** | MAPE versus number of training points from ML model predictions for (a) max proton energy, (b) total proton energy, (c) average proton energy and noisy testing data. Each panel shows results from (solid) 0%, (dashed) 15% and (dotted) 30% added noise in the data. Black lines with different line types indicate the MAPE between the noisy and noiseless data. Because we only compare ML models to noisy data in this figure, these black lines indicate the best that any ML model could conceivably do.

noiseless data. We include this to indicate the best that any ML model could conceivably do when compared to noisy data. Each line type corresponds to a different noise level.

We are interested in determining not only whether the ML models are reasonably accurate compared to the known level of noise in our synthetic data set, but also how well the ML models (which are trained on noisy data) compare to noiseless data when restricted to a limited amount of points. Figure 4 compares the accuracy of the ML models trained on 2000 data points with between 0% and 30% Gaussian noise compared to 500 test data points that both do and do not include noise. For comparison, a diagonal solid black line shows the MAPE between noisy and noiseless data as a function of the noise level. This is helpful to interpret the comparison between the ML models and the noisy data as a kind of best-case scenario, and in the comparison to noiseless data, to the extent that the ML models fall below this diagonal solid black line, it indicates that the ML models are successfully averaging out the noise.

The figure shows that when the ML models are compared to the noisy data, except for the low noise NN results, the MAPE tends to be dominated by the noise as expected. Compared to noiseless data, the ML models (which were trained on noisy data) tend to have a MAPE that is smaller than the data sets they were trained on. Again, the only exception is the NN results for $\leq 5\%$ Gaussian noise.

## 5 | Performance and Memory Consumption

Figure 5 shows our primary results for the execution time of the different ML models using between 500 and 20,000 synthetic training data points averaged between noise levels of 0%–30%.

Of the three models, the clear winner with respect to execution time is SVR, followed by NN and, last, GPR. Although we expected the execution time for GPR to approach $O(N^3)$ [41], the increase in the execution time with increasing numbers of

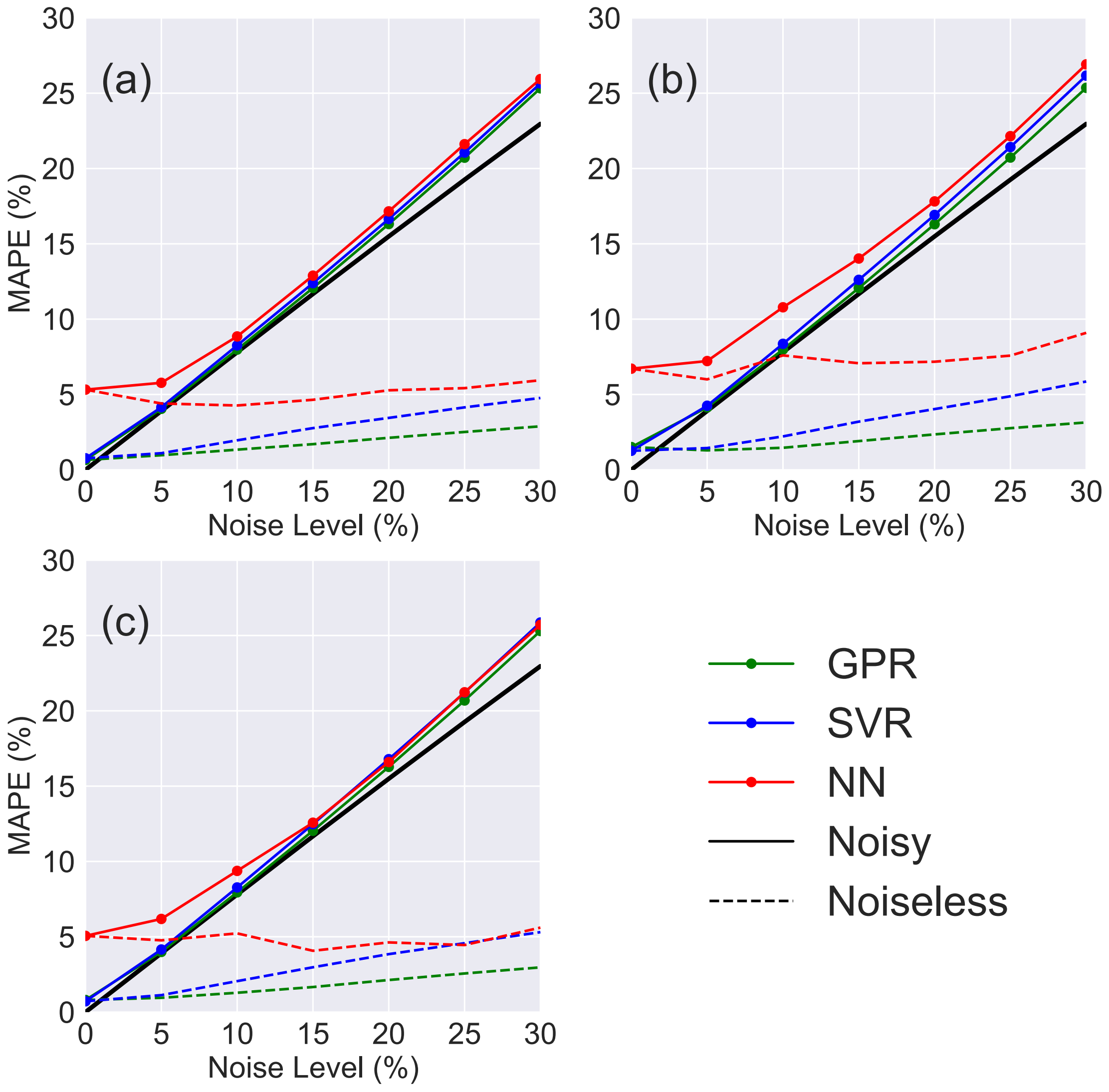

**FIGURE 4** | Solid lines show the typical MAPE in (a) maximum proton energy, (b) total proton energy, and (c) average proton energy when the ML models (which were trained on 2000 synthetic data points with noise) are evaluated on data with different levels of noise. Dashed lines show the typical error when those same ML models are evaluated on noiseless test data. Black solid lines indicate the MAPE between the noisy and noiseless data.

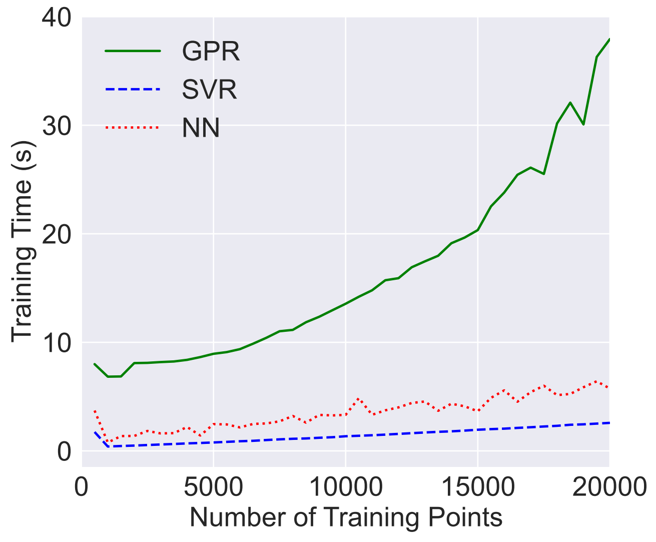

**FIGURE 5** | Comparing the execution time of the different ML models averaged across noise levels in computing the maximum, total, and average proton energies.

**TABLE 1** | Average GPU memory consumption results when training on data with 20,000 points.

| | SVR | NN | GPR |
|---|---|---|---|
| GPU memory utilization (GiB) | 1.6 | 1.1 | 14.0 |

training data points was not as steep as this, at least for the range of $N$ that we probed and the data set we used.

Table 1 summarizes the average GPU memory consumption of the three different ML models on 20,000 synthetic data points averaged across noise levels. The NN had the lowest memory consumption while GPR consumed the most memory.

## 6 | Optimization Task

In this section, we use trained models to perform a simple optimization task which is to determine a set of input parameters that would produce a proton spectrum up to a particular cutoff energy—and that would provide the highest possible numbers of protons in that energy range despite the limited amount of laser energy available. This is a simple but interesting task because one can produce a proton energy spectrum in a particular energy range by placing the target at peak focus and decreasing the laser energy and intensity, or one can keep the laser energy fixed and move the target out of focus until the intensity produces the desired proton energy range, or one can keep the target at peak focus and increase the thickness until the proton energy range decreases to the desired range. Therefore, although our modified Fuchs et al. [16] function is relatively simple this optimization task should produce a non-trivial result.

We perform this optimization by re-casting it as a minimization problem. Producing a proton energy spectrum with a max kinetic energy of $KE_{\text{cutoff,goal}}$ and with significant laser energy to proton energy conversion efficiency is like minimizing this function

$$f\left(KE_{\text{cutoff}}, \eta_{\text{proton}}\right) = \frac{|KE_{\text{cutoff}} - KE_{\text{cutoff,goal}}|}{KE_{\text{cutoff,goal}}} + \frac{C}{\eta_{\text{proton}}} + g\left(KE_{\text{cutoff}}, KE_{\text{cutoff,goal}}\right) \quad (3)$$

where $C$ is a parameter that is introduced to ensure that both the first and second terms have similar importance. $g\left(KE_{\text{cutoff}}, KE_{\text{cutoff,goal}}\right)$ is a penalty term that is set to a large positive number when $KE_{\text{cutoff}}$ is more than $\pm 15\%$ away from $KE_{\text{cutoff,goal}}$.

The ML models provide estimates of $KE_{\text{cutoff}}$ and $\eta_{\text{proton}}$ given the input parameters of laser energy, distance from peak focus to the target and target thickness. We therefore vary these three parameters and use the ML models to determine where in this 3D parameter space the function defined by Equation (3) is a minimum. We then compare performing this same task for the modified Fuchs et al. model without any noise to obtain the true values of the parameters that minimize the function. The goal of this exercise is to determine if the ML methods, which demonstrate different levels of accuracy as discussed in §4, are more or less robust to noise in performing this task.

To the extent that the ML models perform this task well their conclusions regarding the ideal conditions for three different maximum kinetic energy cutoffs should resemble the results shown in Figure 6, which is not from any trained ML model but is derived from the analytic model itself without any noise. The three different maximum kinetic energy cutoffs are 0.25, 0.5, and 1.0 MeV.

There are many different combinations of target thickness and focal distance that produce proton spectra with these cutoffs and the thin red lines in Figure 6 indicate those possibilities for each cutoff energy. In some applications, having a kinetic energy cutoff that is 15% higher or lower than the target cutoff may be acceptable, so we also indicate on the plot with gray lines the thicknesses and focal distances that produce energies that are 15% above or below the three desired cutoff energies.

Although there are many combinations of target thickness and focal distance that can produce a proton energy spectrum with the right cutoff, not all of these combinations also optimize the laser-to-proton energy conversion efficiency ($\eta_{\text{proton}}$). Because $\eta_{\text{proton}}$ tends to increase for increasing maximum kinetic energy cutoffs, in Equation (3) we chose $C = 0.8 \cdot 10^{-3}, 1.8 \cdot 10^{-3}$ and $4.5 \cdot 10^{-3}$ respectively for cutoff energies 0.25, 0.5, and 1.0 MeV to ensure that the optimization prioritizes both matching the maximum kinetic energy cutoff and increasing $\eta_{\text{proton}}$. The green areas in Figure 6 show the part of the parameter space that would achieve a proton energy cutoff within $\pm 15\%$ of the cutoff goal *and* have a significant $\eta_{\text{proton}}$ value. Generally, in our analytic model, the need to optimize $\eta_{\text{proton}}$ favors focal distances near zero and thinner targets. A green star shows the ideal target thickness and focal distance given these constraints. This is equivalent to using Equation (3) as the optimization function but without the first term. The results from using all the terms in Equation (3) is shown with a blue shaded area. Specifically, this area includes a minimum of Equation (3) as well as conditions that cause the optimization function to be within 5% of the global minimum. Although similar to the green area, now the optimization will favor conditions that have both a significant $\eta_{\text{proton}}$ and which produce cutoffs closer to the desired energy. The ideal thickness and focal distance including all the terms in Equation (3) is shown with a blue star. Generally, the blue star is close to the green star except that the blue star is always closer to the desired cutoff energy (red line).

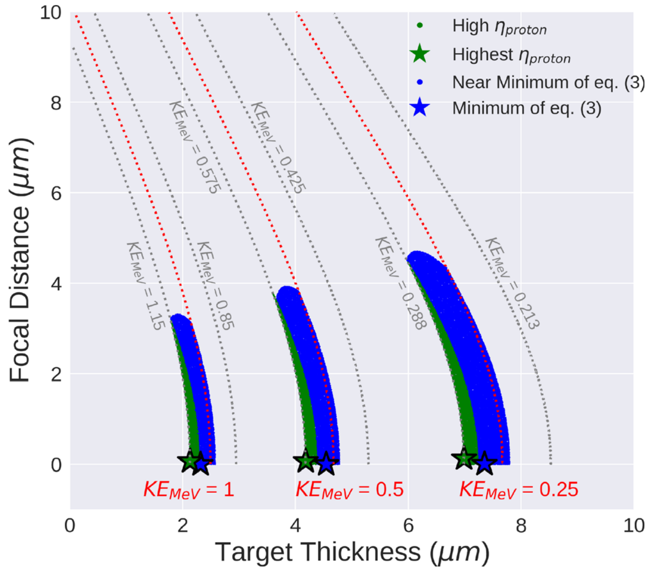

**FIGURE 6** | Parameters that produce maximum proton energy cutoffs in three different desired ranges: 1.0, 0.5, and 0.25 MeV. Combinations of thickness and focal distance that produce these energy cutoffs (irrespective of the laser-to-proton conversion efficiency) are shown with dotted red lines. With each red line we also show with dotted gray lines the thicknesses and focal distances that produce proton energy cutoffs that are +15% or −15% of the cutoff goal. Green shaded areas show regions where the laser-to-proton conversion efficiency is high (i.e., within 5% of the optimal value). A green star shows the ideal conditions for maximizing the proton conversion efficiency. The blue region corresponds to using all the terms in Equation (3) and the blue star indicates the ideal conditions according to that minimization scheme.

Because the analytic function we are using yields the same results for positive and negative focal distances, we only consider positive focal distances. In a real experiment, one would need to explore both positive and negative focal distances, and the contours that optimize proton energy generation will likely have more structure because of prepulse effects that are especially important at peak focus (zero focal distance).

Whereas Figure 6 shows results for optimum conditions from our analytic model without any noise, Figure 7 shows results from the same exercise using the three ML models. To help with the comparison the three sets of lines in Figure 6 also appear in Figure 7. Specifically, we show results from training the three models on synthetic data with 30% added noise and with 2000 points in the training set. Analogously to Figure 6, we can repeatedly query these trained ML models to obtain their predictions. We then use these predictions with Equation (3) and obtain both the optimum values and the regions that fall within 5% of that optimum value.

Figure 7 shows that both SVR and NN tend to find optima that occur at thinner target configurations than the gray lines would indicate. This means that these ML models incorrectly favor conditions that would produce protons with more than 15% of the intended cutoff energy. More generally, Figure 7 shows that there are practical consequences for using ML models that perform better or worse on the tests described in §4.

## 7 | Discussion

Of the three ML models, the NN tended to be less accurate than the other models (Figures 3, 4, and 7). As a comment on this result, it is important to note that the approach of the NN method is very different from either SVR or GPR in that it is essentially trying to fit 8771 free parameters using 20,000 data points or less. Even with sophisticated methods to determine those parameters like "Adam" back-propagation [38], this is not enough to reach high accuracy without more data. A reasonable question is whether our choice of using three hidden layers (Figure 2) could have affected this result. In Appendix A, we answer this question

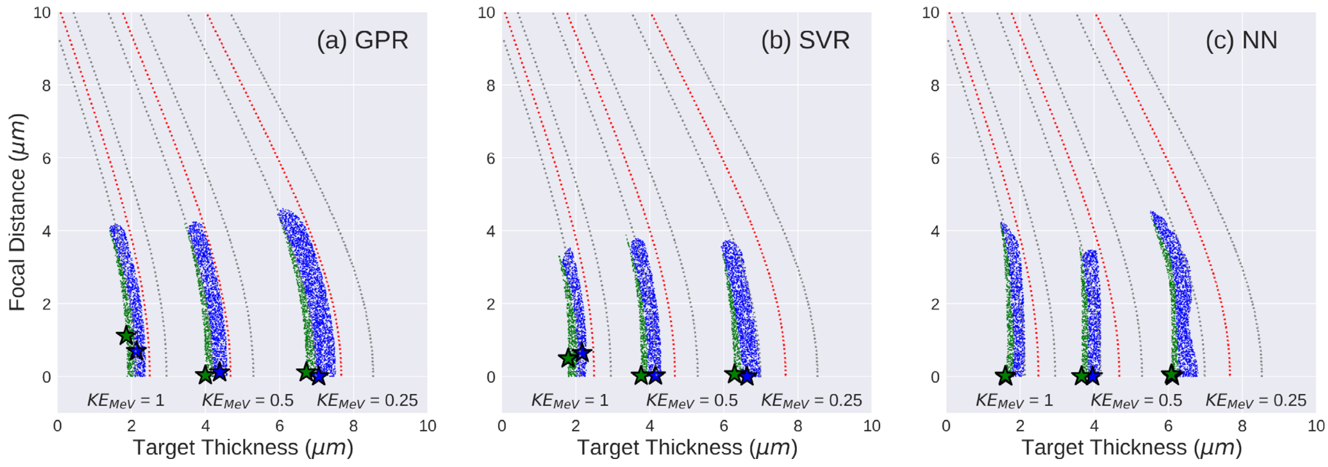

**FIGURE 7** | Parameters that produce maximum proton energy cutoffs according to three trained ML models on 2000 data points with 30% added noise: (a) GPR, (b) SVR, (c) NN compared against the red and gray lines plotted in Figure 6. The green and blue shaded regions are scatter plots of a subset of evaluated points that fall within 5% of the model's predicted optimum according to the same criteria in Figure 6.

by conducting a grid search through different hidden layers and neurons to optimize the network architecture.

We examined the performance of the ML models running on one GPU. We found that the SVR training time was much less than GPR and NN, which is perhaps not surprising since SVR has a reputation for being a fast method (e.g., Gandhi [42], Xu and Suzuki [43]). The performance results made us optimistic that at least some ML models can be accurately trained using one GPU in quasi-real time in kHz ultra-intense laser experiments.

We found that the GPR method consumed the most GPU RAM of the three models. Efforts to use GPR in ultra-intense laser experiments that produce significantly more than 20,000 data points on a short timescale may need to worry about GPU RAM constraints, or instead use an approximate GPR method (e.g., Hensman et al. [44]).

To demonstrate how the quality of the ML model can lead to practical consequences and to provide a simple application of our framework, we used the trained ML models to predict the laser parameters that produce a desired proton energy spectrum while also maximizing the laser-to-proton energy conversion efficiency. We performed such tests in Section 6 using 2000 training points and found (Figure 7) that the NN and SVR models incorrectly favor thinner targets than would be needed to produce protons in a desired energy range and in some cases NN and SVR incorrectly favored a non-zero focal distance even though the analytic model (which does not have any prepulse physics) indicates that zero focal distance was always most favorable for maximizing the conversion efficiency.

All these results are based on a modified Fuchs et al. [16] model that we developed to generate synthetic data. Overall, this model is relatively smooth and well behaved whereas real experiments typically have more features (e.g., Morrison et al. [12], Loughran et al. [10]). Certainly, this limits the generality of our results. Our goal was to create a kind of standard candle for evaluating ML methods for TNSA experiments. In a forthcoming study, we are exploring larger and more realistic data sets with more physics (and features) included in the synthetic data model.

## 8 | Conclusions

We tested three different ML models on a synthetic data set for laser-accelerated protons by training these models on up to 20,000 synthetic data points. The data set was generated using a modified Fuchs et al. [16] model that included Gaussian noise to simulate the kind of noise that could be present in a real experiment. The ML models were GPR, SVR, and a three-hidden-layer NN.

Of the three methods, the NN models were the least accurate and this was especially true when the number of training points was low. This included an exercise where we used the ML models to predict optimum conditions for proton acceleration at different cutoff energies. The poor accuracy of the NN model likely stems from the large number of free parameters that must be precisely determined for the model to become highly accurate.

In terms of performance on one GPU, SVR was by far the fastest model for execution time. GPR was the slowest, taking 30 s to train on 20,000 data points. The NN used the least amount of GPU memory while GPR consumed 5–10 times more GPU memory than the other models. Generally, the performance results suggest that quasi-real-time training of these models on kHz repetition rate laser systems should be feasible, especially with SVR.

We stress that these results should not be regarded as the final word on the usefulness of these ML models in the context of laser acceleration of protons. More complex data sets may yield different results. Moreover, there are certainly ML models that deserve attention that we did not study here.

We provide Jupyter notebooks with our Python code. We also provide the 25,000-point synthetic data sets that we generated [23]. By providing these files, we hope to encourage others to compare other ML models against our results as a benchmark.

### Acknowledgments

Supercomputer allocations for this project included time from the Ohio Supercomputer Center. We also thank Pedro Gaxiola for help with early work with SVR. We acknowledge support provided by the National Science Foundation (NSF) under Grant No. 2109222. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. C.O. and R.O. were supported in summer 2022 by the Air Force Office of Science Research summer faculty program. This work was supported by Air Force Office of Scientific Research (AFOSR) Award number 23AFCOR004 (PM: Dr. Andrew B. Stickrath). This work was also supported by Department of Energy Award number 89243021SSC000084 (PM: Dr. Kramer Akli).

### Conflicts of Interest

The authors declare no conflicts of interest.

### Data Availability Statement

The data that support the findings of this study are openly available in zenodo at https://zenodo.org/records/12752264, reference number 10.5281/zenodo.12752264.

### References

1. R. Anirudh, R. Archibald, M. S. Asif, et al., "2022 Review of Data-Driven Plasma Science," *IEEE Transactions on Plasma Science* 51, no. 7 (2023): 1750–1838, https://doi.org/10.1109/TPS.2023.3268170.

2. A. Döpp, C. Eberle, S. Howard, F. Irshad, J. Lin, and M. Streeter, "Data-Driven Science and Machine Learning Methods in Laser-Plasma Physics," *High Power Laser Science and Engineering* 11 (2023): 1–50, https://doi.org/10.1017/hpl.2023.47.

3. P. V. Heuer, S. Feister, D. B. Schaeffer, and H. G. Rinderknecht, "Preface to Special Topic: The High Repetition Rate Frontier in High-Energy-Density Physics," *Physics of Plasmas* 29, no. 11 (2022): 110401, https://doi.org/10.1063/5.0130801.

4. R. J. Shalloo, S. J. D. Dann, J. N. Gruse, et al., "Automation and Control of Laser Wakefield Accelerators Using Bayesian Optimization," *Nature Communications* 11 (2000): 6355, https://doi.org/10.1038/s41467-020-20245-6.

5. S. Jalas, M. Kirchen, P. Messner, et al., "Bayesian Optimization of a Laser-Plasma Accelerator," *Physical Review Letters* 126 (2021): 104801, https://doi.org/10.1103/PhysRevLett.126.104801.

6. J. Lin, Q. Qian, J. Murphy, et al., "Beyond Optimization–Supervised Learning Applications in Relativistic Laser-Plasma Experiments," *Physics of Plasmas* 28, no. 8 (2021): 083102, https://doi.org/10.1063/5.0047940.

7. H. Ye, G. Yuqiu, X. Zhang, et al., "Fast Optimization for Betatron Radiation From Laser Wakefield Acceleration Based on Bayesian Optimization," *Results in Physics* 43 (2022): 106116, https://doi.org/10.1016/j.rinp.2022.106116.

8. J. Duris, D. Kennedy, A. Hanuka, et al., "Bayesian Optimization of a Free-Electron Laser," *Physical Review Letters* 124 (2020): 124801, https://doi.org/10.1103/PhysRevLett.124.124801.

9. N. Breckwoldt, S.-K. Son, T. Mazza, et al., "Machine-Learning Calibration of Intense X-Ray Free-Electron-Laser Pulses Using Bayesian Optimization," *Physical Review Research* 5 (2023): 023114, https://doi.org/10.1103/PhysRevResearch.5.023114.

10. B. Loughran, M. J. V. Streeter, H. Ahmed, et al., "Automated Control and Optimization of Laser-Driven Ion Acceleration," *High Power Laser Science and Engineering* 11 (2023): e35, https://doi.org/10.1017/hpl.2023.23.

11. T. Ma, D. Mariscal, R. Anirudh, et al., "Accelerating the Rate of Discovery: Toward High-Repetition-Rate HED Science," *Plasma Physics and Controlled Fusion* 63, no. 10 (2021): 104003, https://doi.org/10.1088/1361-6587/ac1f67.

12. J. T. Morrison, S. Feister, K. D. Frische, et al., "MeV Proton Acceleration at kHz Repetition Rate From Ultra-Intense Laser Liquid Interaction," *New Journal of Physics* 20, no. 2 (2018): 022001, https://doi.org/10.1088/1367-2630/aaa8d1.

13. Z. Cao, Z. Peng, Y. Shou, et al., "Vibration and Jitter of Free-Flowing Thin Liquid Sheets as Target for High-Repetition-Rate Laser-Ion Acceleration," *Frontiers in Physics* 11 (2023): 1172075, https://doi.org/10.3389/fphy.2023.1172075.

14. B. M. Knight, C. M. Gautam, C. R. Stoner, et al., "Detailed Characterization of kHz-Rate Laser-Driven Fusion at a Thin Liquid Sheet With a Neutron Detection Suite," *High Power Laser Science and Engineering* 12 (2024): e2, https://doi.org/10.1017/hpl.2023.84.

15. C. Palmer, "Paving the Way for a Revolution in High Repetition Rate Laser-Driven Ion Acceleration," *New Journal of Physics* 20, no. 6 (2018): 061001, https://doi.org/10.1088/1367-2630/aac5ce.

16. J. Fuchs, P. Antici, E. Emmanuel D'Humières, et al., "Laser-Driven Proton Scaling Laws and New Paths Towards Energy Increase," *Nature Physics* 2 (2005): 12–54, https://doi.org/10.1038/nphys199.

17. M. Daniele, B. Schmitz, D. Kreuter, and O. Boine-Frankenheim, "Modeling of a Liquid Leaf Target Tnsa Experiment Using Particle-In-Cell Simulations and Deep Learning," *Laser and Particle Beams* 2023 (2023): 2868112, https://doi.org/10.1155/2023/2868112.

18. B. Z. Djordjević, A. J. Kemp, J. Kim, et al., "Modeling Laser-Driven Ion Acceleration With Deep Learning," *Physics of Plasmas* 28, no. 4 (2021): 043105, https://doi.org/10.1063/5.0045449.

19. E. J. Dolier, M. King, R. Wilson, R. J. Gray, and P. McKenna, "Multi-Parameter Bayesian Optimisation of Laser-Driven Ion Acceleration in Particle-In-Cell Simulations," *New Journal of Physics* 24, no. 7 (2022): 073025, https://doi.org/10.1088/1367-2630/ac7db4.

20. B. Z. Djordjević, J. Kim, S. C. Wilks, et al., "Transfer Learning and Multi-Fidelity Modeling of Laser-Driven Particle Acceleration," *Physics of Plasmas* 30, no. 4 (2023): 043111, https://doi.org/10.1063/5.0139285.

21. P. Mora, "Plasma Expansion Into a Vacuum," *Physical Review Letters* 90 (2003): 185002, https://doi.org/10.1103/PhysRevLett.90.185002.

22. B. Z. Djordjević, A. J. Kemp, J. Kim, et al., "Characterizing the Acceleration Time of Laser-Driven Ion Acceleration With Data-Informed Neural Networks," *Plasma Physics and Controlled Fusion* 63, no. 9 (2021): 094005, https://doi.org/10.1088/1361-6587/ac172a.

23. R. Desai, "Datasets and Code From Applying Machine Learning Methods to Laser Acceleration of Protons: Lessons Learned From Synthetic Data," 2024, https://doi.org/10.5281/zenodo.12752264.

24. J. Schreiber, F. Bell, F. Grüner, et al., "Analytical Model for Ion Acceleration by High-Intensity Laser Pulses," *Physical Review Letters* 97 (2006): 045005, https://doi.org/10.1103/PhysRevLett.97.045005.

25. M. Passoni and M. Lontano, "Theory of Light-Ion Acceleration Driven by a Strong Charge Separation," *Physical Review Letters* 101 (2008): 115001, https://doi.org/10.1103/PhysRevLett.101.115001.

26. M. Passoni, L. Bertagna, and A. Zani, "Target Normal Sheath Acceleration: Theory, Comparison With Experiments and Future Perspectives," *New Journal of Physics* 12, no. 4 (2010): 045012, https://doi.org/10.1088/1367-2630/12/4/045012.

27. M. Zimmer, S. Scheuren, T. Ebert, et al., "Analysis of Laser-Proton Acceleration Experiments for Development of Empirical Scaling Laws," *Physical Review E* 104 (2021): 045210, https://doi.org/10.1103/PhysRevE.104.045210.

28. E. L. Clark, K. Krushelnick, J. R. Davies, et al., "Measurements of Energetic Proton Transport Through Magnetized Plasma From Intense Laser Interactions With Solids," *Physical Review Letters* 84 (2000): 670–673, https://doi.org/10.1103/PhysRevLett.84.670.

29. S. P. Hatchett, C. G. Brown, T. E. Cowan, et al., "Electron, Photon, and Ion Beams From the Relativistic Interaction of Petawatt Laser Pulses With Solid Targets," *Physics of Plasmas* 7, no. 5 (2000): 2076–2082, https://doi.org/10.1063/1.874030.

30. R. A. Snavely, M. H. Key, S. P. Hatchett, et al., "Intense High-Energy Proton Beams From Petawatt-Laser Irradiation of Solids," *Physical Review Letters* 85 (2000): 2945–2948, https://doi.org/10.1103/PhysRevLett.85.2945.

31. M. N. u. Haq, H. Ahmed, T. Sokollik, et al., "Parametric Scalings of Laser Driven Protons Using a High Repetition Rate Tape Drive Target System," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 909 (2018): 164–167, https://doi.org/10.1016/j.nima.2018.02.108.

32. R. S. Nagymihály, F. Falcoz, B. Bussiere, et al., "The Petawatt Laser of Eli Alps: Reaching the 700 Tw Level at 10 Hz Repetition Rate," *Optics Express* 31, no. 26 (2023): 44160–44176, https://doi.org/10.1364/OE.509615.

33. J. Han, M. Kamber, and J. Pei, "Data Mining: Concepts and Techniques," in *The Morgan Kaufmann Series in Data Management Systems* (Waltham, MA, USA: Elsevier Science, 2011), 744.

34. D. M. Miller, "Reducing Transformation Bias in Curve Fitting," *American Statistician* 38, no. 2 (1984): 124–126.

35. A. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression," *Statistics and Computing* 14 (2004): 199–222.

36. RAPIDS Development Team, "RAPIDS: Libraries for End to End GPU Data Science," 2023, https://rapids.ai.

37. A. Paszke, S. Gross, F. Massa, et al., "Pytorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Vancouver, Canada: Curran Associates, Inc, 2019) https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

38. P. K. Diederik and J. Ba, "Adam: A Method For Stochastic Optimization," 2017.

39. E. Schulz, M. Speekenbrink, and A. Krause, "A Tutorial on Gaussian Process Regression: Modelling, Exploring, and Exploiting Functions," *Journal of Mathematical Psychology* 85 (2017): 1–16.

40. J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox Matrix-Matrix Gaussian Process Inference With

Gpu Acceleration," 2018, In *Advances in Neural Information Processing Systems*.

41. J. Wang, "An Intuitive Tutorial to Gaussian Processes Regression," *Computing in Science and Engineering* 25 (2022): 4–11.

42. R. Gandhi, "Support Vector Machine—Introduction to Machine Learning Algorithms," 2023, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

43. X. Jian-Wu and K. Suzuki, "Massive-Training Support Vector Regression and Gaussian Process for False-Positive Reduction in Computer-Aided Detection of Polyps in CT Colonography," *Medical Physics* 38, no. 4 (2011): 1888–1902, https://doi.org/10.1118/1.3562898.

44. J. Hensman, N. Fusi, and N. D. Lawrence, Gaussian Processes for Big Data 2013, *arXiv e-prints*, Art, arXiv:1309.6835, https://doi.org/10.48550/arXiv.1309.6835.

45. K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks* 2, no. 5 (1989): 359–366, https://doi.org/10.1016/0893-6080.

46. M. Leshno, V. Ya, A. P. Lin, and S. Schocken, "Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate any Function," *Neural Networks* 6, no. 6 (1993): 861–867, https://doi.org/10.1016/S0893-6080.

47. F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-Learn: Machine Learning in Python," *Journal of Machine Learning Research* 12 (2011): 2825–2830.

## Appendix A: on Choosing Optimal Hyperparameters

The number of hidden layers is an important concern when designing a NN. Work done by Hornik et al. [45], Leshno et al. [46] showed that NNs with at least one hidden layer and using a nonpolynomial activation function can serve as an universal approximator for any function given a sufficient number of nodes. Adding more hidden layers can, in some cases, reduce the training time and allow for faster convergence for a NN. However, this depends on the data set with more complex data sets benefiting from additional hidden layers.

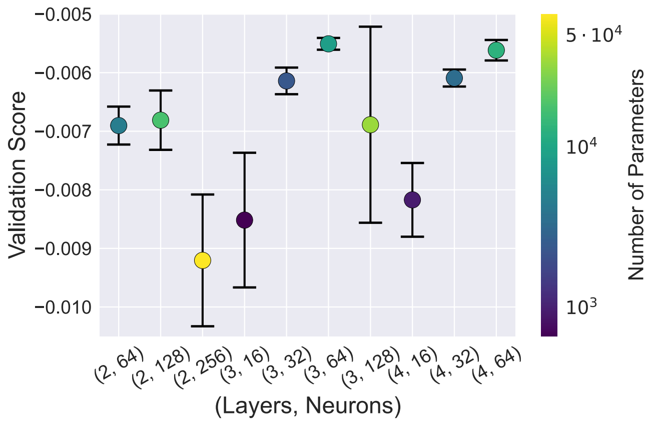

**FIGURE 8** | Validation score (using negative mean square error) with error bars from 5-fold cross-validation plotted against the number of hidden layers and neurons per layer from a grid search of hyperparameters on a 5000 point synthetic dataset with 10% added noise. The following hyperparameters are fixed: batch size (256), learning rate decay $\gamma$ (0.98), activation function (LeakyReLU), optimizer (Adam). The coloring shows the number of learnable parameters (computed from Equation (A1 in the neural network model.

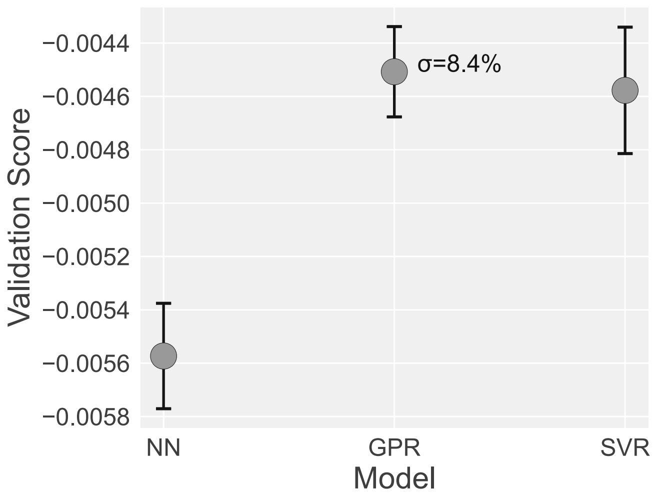

**FIGURE 9** | Validation score comparison with error bars from 5-fold cross-validation for the SVR, NN, and GPR models from a grid search of hyperparameters on a 5000 point synthetic dataset with 10% added noise.

Our NN model architecture was chosen by performing a grid search of model hyperparameters using scikit-learn's [47] `GridSearchCV` module. In Figure 8, various model architectures between 1 and 4 hidden layers and 16–256 neurons per layer were tested. The architectures with highest validation score have 3 or 4 hidden layers. Notably, the 4 hidden layer scores are not higher than the three hidden layer scores, demonstrating that adding more layers does not always improve the network. The results with only one hidden layer all had an order of magnitude lower score than the scores shown and were not included as a result.

Additionally, the number of learnable parameters in the NN model is shown in the color bar of Figure 8. This model has a learnable parameter for every connection between nodes in adjacent layers and additional bias parameters for every neuron in the output and hidden layers. This results in a total of

$$f(n,\ell) = 7n + 3 + n(n+1)(\ell - 1) \tag{A1}$$

parameters where $n$ is the number of neurons per layer and $\ell$ is the total number of hidden layers. In Figure 8, we can see that more learnable parameters do not necessarily equate to a better model.

For the SVR and GPR, we also used `GridSearchCV` to settle on the optimal hyperparameters chosen in Section 3 and the results can be seen in Figure 9.

Here, the GPR narrowly outperforms the SVR which both strongly outperform the NN model. An added benefit of the GPR model is its inherent uncertainty estimation given by $\sigma = 8.4\%$ (can be compared to the noise level 10% of the training data) which was determined by averaging the uncertainty estimate over all points in the training dataset. When data are expensive to obtain, the GPR's uncertainty estimate can guide one's choice of new data points to take (where the uncertainty is greater).