PREDICTING ELECTRON ENERGY SPECTRA IN LASER-PLASMA SIMULATIONS

Michael Pokornik, Erda Wen, and Tofig Mamedov

Group 6

ABSTRACT

This paper aims to utilize machine learning methods to predict electron energy spectra found in Laser-Plasma simulations. The particular focus is to predict the emergence of high energy electrons by predicting Lorentz factor γ. A data set is generated through a PIC simulation and used as a training set for a CNN model. Results show that predictions are highly accurate for timestamps close to the end time. This is an indication of a chaotic system for electron trajectories. Nevertheless, the model can be utilized in tandem with a traditional PIC simulator to save many hours of computing time, as the model can predict the final timestamps much faster.

***Index Terms*—** Laser Plasma, Convolutional Neural Network

1. INTRODUCTION

The production of very energetic charged particles has several applications in areas like imaging, fusion, accelerator physics, and others. In laser-plasma experiments, ion beams and x-rays are often indirectly driven from the production of high energy electrons. It is of interest to the laser-plasma community to accurately resolve the electron energy distribution, which is scaled as $E_e = \gamma mc^2$ where γ is the Lorentz factor. Depending on the physical experiment, high energy electrons could be detrimental (for example, pre-heating a target capsule in inertial confinement fusion applications raises the capsule entropy and makes the target difficult to compress) or crucial (generation of gamma radiation for active interrogation) to its success. Electron heating mechanisms can become very complicated and are usually either omitted in certain laser-plasma codes or are explored through Particle-In-Cell (PIC) simulations. These simulations can be computationally expensive and coupling them to other codes becomes challenging. [1, 2, 3] The goal of this project is to develop a model to replicate physical electron heating and be included in other models that lack this effect. [4] The input to our algorithm is a time series of electron position x and Lorentz factor (hereafter referred to as gamma or γ). We then use a Convolutional Neural Network (CNN) to predict the final gamma value for an electron.

2. RELATED WORK

Machine learning has only recently been introduced into the field of laser-plasma interactions. To our knowledge there has not been any results or attempts to use machine learning for our problem. However, machine learning has had success in the field. In particular, neural networks were trained on PIC simulation data of a plasma harmonic spectra to predict unknown experimental parameters. [5] One of our group members will be working this summer on using machine learning to predict shock wave timing and implosion symmetry for inertial confinement fusion (ICF) applications.

3. DATASET AND FEATURE ANALYSIS

The data set was generated by one of our group members using the EPOCH 1 dimensional PIC code. The physical setup consists of a laser irradiating a target and creating a plasma. The electric fields from the laser and from the created plasma will impart forces on the electrons and cause them to accelerate/decelerate governed by the Lorentz force. Electrons can be pulled out from the target before re accelerating back in. In some cases electrons can stream back and forth through the target several times. The data contains a total of 11,145 electron samples. There were several types of data available at each time step correlating to both electron characteristics and system characteristics throughout its trajectory, but ultimately only electron position and gamma value were used as features. The time series ranged from zero to 3 picoseconds with intervals of .01 picoseconds. Below is a figure that displays example trajectories of randomly selected electrons and their position and gamma values in time. Some of the predictions that the group could have pursued included predicting electron position, the amount of work done on an electron, and the phase of the laser the electron will sample. The gamma value for prediction was chosen because it is an indicator of how relativistic the electron motion is and scales the electron energy. A higher gamma value can correspond to a more energetic electron and can help build an energy distribution.

The distribution of the final gamma value is plotted as Figure 2 as a reference.

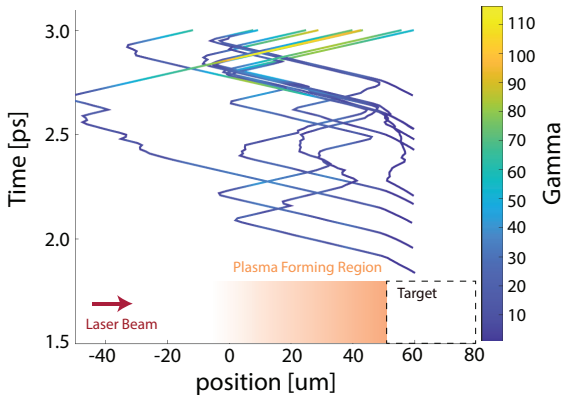

Fig. 1. Gamma and Position over Time for Given Electron Trajectories

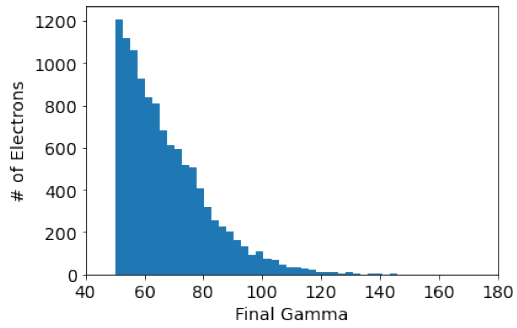

Fig. 2. Final State γ Distribution of the 11,145 Electrons

3.1. Dataset Preprocessing

Comparing to experimental results where noise and abnormal data are common and not ignorable, our the dataset generated by the PIC simulation is very well-organized, thus the preprocessing is relatively simple, which includes 1) zero padding the NaN data before each electron emerges, and 2) normalizing each physical quantity series to [0,1] respectively.

3.2. Feature Importance Analysis

The simulation generate excessive amount of variables for each electron trajectories, and a preliminary study on the each of them is decisive on the feature selection for the prediction model.

To do so, we here adopt a random forest regressor. The regressor takes the whole time series (291 time steps) of physical quantities generated by simulation (position, momentum, work, phase, gammma value) as the input, and take final gamma as the label. The model does not necessarily need to be optimized here, since we only care about the feature importance instead of making any prediction.

Figure 3 shows the feature importance of the two dominant features: position and gamma value at different timestamp, where timestamp 0 refers to the final state at the completion of the simulation. The result agrees well with our physical intuition that most of the information comes from close to the end of the trajectory. This also reveal the trade-off nature of the problem: to save more computational resources by using earlier time steps will lead to less accurate prediction.

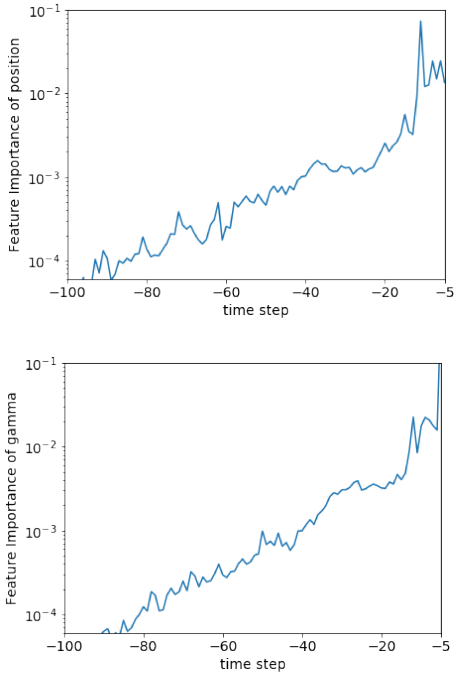

Fig. 3. Feature Importance of Position x and Gamma value γ at Different Timestamps

4. CNN PREDICTION MODEL

Since the data is structured in time series, intuitively one may claim that information lay in the correlation between the neighbouring quantities along time dimension, and thus we choose to use the convolutional neural network (CNN).

In an analytical perspective it is also easy to show that convolutional layers are capable of resembling various operators: considering a size-2 filter with value $-1/\Delta t$ and $1/\Delta t$ acting on a position vector, it can extracting the velocity information since it emulates the discrete form of the derivative $v = dx/dt \approx (x(t + \Delta t) - x(t))/\Delta t$. More examples can be found in Figure 5. This gives an evidence that convolutional layers are potentially a great candidate to emulate the complex underlying physics in the system.

Another advantage of CNN comparing to fully connected neural network is that its training process is much less time-consuming because of a small parameter space the parameter

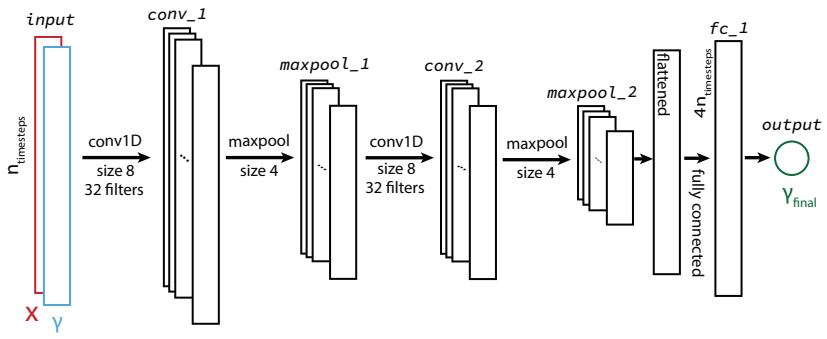

Fig. 4. Convolutional Neural Network Structure

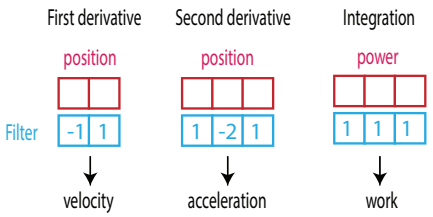

Fig. 5. Examples of Convolutional Filters Resembling Different Operators

sharing. small parameter space also make it easier to handle with the overfitting issue.

The proposed CNN structure is shown as Figure 4. Depending on the computational resources intended to be saved, the network take the simulated results of position and gamma value before time step T as the input, and the final gamma as the output label. Two sets of convolutional layers are used and a fully connected layer is adopted before the output. The activation function are ReLU for all layers and the loss function is set to be the mean square errors, $MSE = \frac{1}{n}\sum(\gamma_i - \hat{\gamma}_i)^2$. For training, the data was split the whole dataset into 8916 training samples and 2229 validation samples.

4.1. Hyperparameter Tuning and Regularization

Convolutional filter size and filter number are two dominant factor that impact the accuracy of this model. In practice, we found that the accuracy begin to converge for filter size larger than 6 and channel number larger than 16. However, large parameter space also aggravate the overfitting issue. Therefore, we choose the filter size to be 8 and channel number to be 32. To deal with the overfitting, two strategies are introduced:

1) A dropout with a probability of 0.3 is added before the fully-connected dense layer *fc_1*, and

2) an L2 regularization term $0.4\sum w_i^2$ is applied to *fc_1*.

Figure 6 shows a comparison between the network with regularization aforementioned and with no regularization at all.

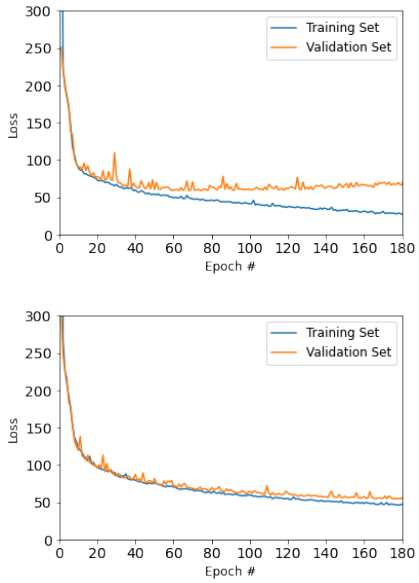

Fig. 6. Loss vs Epochs for Training and Validation Set. Upper: No Regularization; Lower: with Dropout and L2 Regularization

5. RESULTS DISCUSSION

We examine both MSE and the coefficient of determination $R^2 = 1 - \frac{\sum(\gamma_i - \hat{\gamma}_i)^2}{\sum(\gamma_i - \bar{\gamma}_i)^2}$ of the validation set to evaluate the performance of the prediction using position and gamma series before timestamp T. To some extent R^2 score is the same with MSE, but it takes into account the variance of the original data and is thus more informative.

As is visible in Figure 7, the model struggles to make predictions for timestamps that aren't a few steps away from the end time, however the model rapidly converges close to the end. A MSE of near 200 was the maximum error that the model produced, quite consistently for all timestamps until around -25, when significant decreases were observed.

Good model performance near the end time also agree with Figure 4, where the feature importance of position and

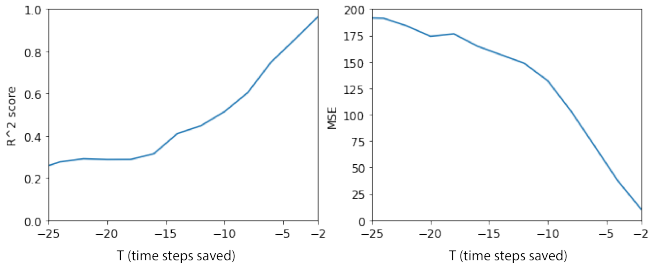

Fig. 7. R^2 and MSE Scores for Timestamps Counted from End Time.

gamma increases significantly near the end time. For timestamps in the middle or closer to the beginning, those features are quite irrelevant. Keep in mind that the final gamma values are observed at timestamp 291 in the simulation. As a better visualization of how earlier end time impact the prediction, Figure 8 shows how predicted final γ distribution changes with different end timestamps T.

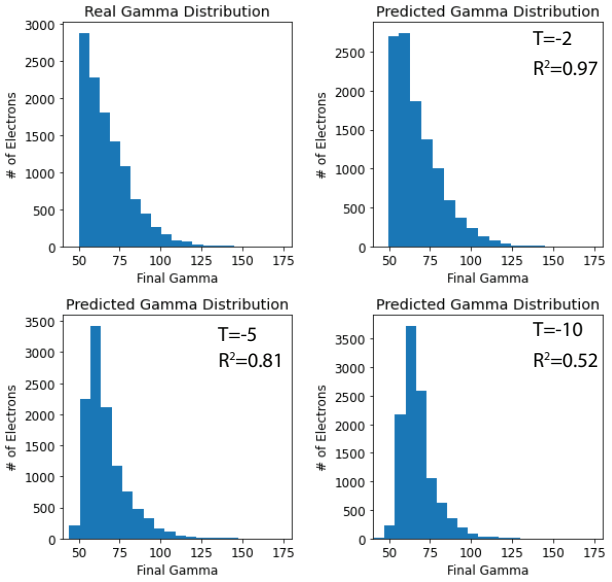

Fig. 8. Comparison between Predicted γ Distributions with Different End Timestamp T.

6. CONCLUSION AND FUTURE WORK

6.1. Chaotic Properties

That fact that the model only makes accurate predictions near the end time indicates that the system is of a chaotic nature. When the electrons travel on their trajectories they are never alone. Other electrons nearby and could influence the trajectory in different ways. Interactions between electrons are random. The influence is therefore also random, electrons might either gain or lose energy in those interactions during their trajectory. Since interactions could happen at any given time, energy gained in the beginning could just as easily be lost later. Therefore, there is some logic behind why the model is at peak accuracy near the end, simply that those interactions near the end time will be the latest ones giving an impact.

To illustrate this reasoning, Figure 1 plots the gamma value throughout the trajectory of one arbitrarily chosen electron. The gamma value varies throughout the trajectory, both gaining and losing. This indicates a chaotic system.

6.2. Run Time

Even though saving a few time steps might seem insignificant, it actually isn't. Keep in mind that it took several hours to generate this dataset of around 11,000 electrons. Other simulations might require more electrons for larger time frames. Consequently, this model will save both time and computing power if applied to large problems.

Depending on what scope the problem has, a compromise between model accuracy and time consumption is crucial. Not all applications will require the highest accuracy, some might settle for less. In that case 10-15 timestamps could be done by the model, thus saving even more computational resources.

6.3. Future Work

One of the problems with our data set is that it is too specific, with certain set initial conditions that could vary depending on the simulation purpose. To make this model useful, generalizing it for different initial conditions would be a priority. One way to do that would be to obtain a larger training set with data generated from different initial conditions and then training a similar machine learning model.

Something else of importance would be to further look into how much time that could be saved with this model. By running the simulations just short of a few timestamps and letting the model finishing the simulation would put a number to it.

Contributions

Michael Pokornik worked on generating the data set and buliding a RFR for prediction. Michael also created a neural network using features in addition to γ and position, but Erda's CNN model ultimately yielded the best performance.

Erda Wen worked on building and optimizing the RFR featrue analysis model and CNN model for prediction as a comparison to the baseline model.

Tofig Mamedov worked with trying out the convolutional neural network that Erda built on different feature sets. A bit of hyperparameter tuning and testing prediction results for different time windows.

7. REFERENCES

[1] Charles K Birdsall and A Bruce Langdon. *Plasma physics via computer simulation*. CRC press, 2004.

[2] Paul Gibbon. *Short pulse laser interactions with matter*. World Scientific Publishing Company Singapore, 2004.

[3] William Kruer. *The physics of laser plasma interactions*. CRC Press, 2018.

[4] K Weichman, APL Robinson, FN Beg, and AV Arefiev. Laser reflection as a catalyst for direct laser acceleration in multipicosecond laser-plasma interaction. *Physics of Plasmas*, 27(1):013106, 2020.

[5] A Gonoskov, Erik Wallin, A Polovinkin, and I Meyerov. Employing machine learning for theory validation and identification of experimental conditions in laser-plasma physics. *Scientific reports*, 9(1):1–15, 2019.

Replies to Critical Reviews

Critique from **group 22**

The group clearly introduces the topic of laser-plasma interactions in a way that is very helpful for someone not familiar with the field. The features of the data, model architecture, and results are explained very well. Overall this seems like a very interesting project.

Thanks!

Some improvements/unclear things:

- How many simulated electron trajectories are included in the dataset? How much variety is included amongst these trajectories? Is this simulated training data representative of all possible conditions someone might want to simulate?

  The dataset contains 11,145 electron samples and it is pretty safe to say that it includes enough number and variety under this particular setup so as to be representative. The distribution plot of final gamma value also verifies this since there are plenty amount of electron samples all the way through the trail with higher gamma value.

- How did you choose the model architecture? Did you guys try other architectures?

  We also try to build and optimize a random forest regressor. A comparison between them shows that CNN wins over RFR slightly (e.g. for t=-5, r^2 score for CNN:0.83, for RFR:0.78; for t=-10, r^2 score for CNN: 0.45, for RFR: 0.41). Thus, we choose CNN model as the predictor.

- Losses are shown but it might be helpful to see accuracy plots also to get a better idea of how well the model predicts

  R^2 scores are also shown as an evaluation of the accuracy.

- Does this model generalize? i.e. If someone wanted to change up the initial conditions or environment and use this model to predict electron energies, would this model work? Or can it only be used with the conditions you simulated for your training?

  Unfortunately, this model works only for this particular setup. We were planning to come us with a more generalized model for other environments (plasma thickness, laser pulse type, etc), but we don't have enough time to generate enough dataset since these simulations are extremely time-consuming. We put it as future works in our report.

Critique from **group 75**

Group 6 made a fluent and lucid introduction of electron states and laser-plasma applications. They also compared the tradition energy prediction method with their ML method, explained the reason of using ML in electron energy prediction: not more accurate but able to save time and computational resources, use model to replace physics.

They applied their convolutional neural network on a simulated 1D particle-in-cell sequence data, to mimic physics operators. They also used dropout and L2 regularization to decrease overfitting. Their result performance are better when sequence approaches end-time.

Thank you!

However, there are still some improvements we could come up according to their work.

1 In presentation, the video windows of presenters are overlapping with their slides, sometimes blocked important text.

  Sorry for the carelessness. Hopefully nothing is blocked in the report :)

2 We thought they could make more pictures to visualize the evolution of electrons and the 1D particle-in-cell simulator, to show a more vivid explanation of their topic and data.

Yes, that is actually a very good point. We put some exemplary electron trajectory plot in the report to show what is going on in the system.

3 We thought they could make some figures or even examples of the logic and results of traditional energy computing methods, then compare them with ML methods. This will give audience a clearer understanding about pros (save time and computation resources) and cons (lack of accuracy) of ML methods.

We are adding more details on traditional methods to make a better comparison between them.

4 The final code exhibition part is kind of short. We thought they could show more codes such as how the 1D particle-in-cell simulator is working.

We have several parts of the code: an RFR as a preliminary feature selection, CNN work for a single time step T setup and a T sweep. RFR is not the core part of the predictor, and T sweep code is time-consuming as well as tedious. Thus, we choose to show a most simple case where T=-5. Unfortunately, it is not very practical and helpful to show the 1D particle-in-cell simulator, both because it will be hard to read embedded with excessively amount of formula and numerical analysis, and it takes hours or even days to run it.

Critique from **group 89**

The presentation succinctly and sufficiently explained the background of the task to be predicted, as well as the steps they took to achieve their results. The presenters had a good explanation of the results and conclusions, including the connection with the weather problem from the first homework. The code portion of the video was rather brief despite involving a full training run of the model.

Thanks!

Improvements:

- Further explanation and/or connection to the convolutional derivative/integration models could help, especially how it is represented in the model. It currently seems a bit handwavy.
  The derivative/integration are some evidence that shows the filters are capable of resembling various operators, as part of the justification that convolutional layers are potentially a great candidate to emulate the complex underlying physics.

- Brief explanations of the model and metrics chosen:
  - Why 1D convolutions, filter sizes, etc.

  Since we only have a single dimension (time), we choose 1D convolution. While tuning the network, we found that the accuracy converges for large filter sizes and channel numbers. However, large parameter space will also aggravate the overfitting issue. Therefore, we come up with a trade-off where filter sizes and channel numbers are sufficiently large while the overfitting is rather easy to deal with.

  - the loss type wasn't mentioned during presentation

  We use MSE as the loss function.

  - Why was used for result evaluations r2

  To some extent r^2 score is the same with MSE, but it takes into account the variance of the original data and is thus more informative.

- A comparison with a second model, or training with different hyperparameters, could be useful in evaluating the model, since we would have a baseline to compare to. Maybe the random forest results could have been shown since it was already fitted to the dataset.
  Yes, we also tried a random forest regressor predictor which can be considered as a baseline. A comparison shows that CNN wins over optimized RFR slightly (e.g. for t=-5, r^2 score for CNN:0.83, for RFR:0.78; for t=-10, r^2 score for CNN: 0.45, for RFR: 0.41).