Energy Prediction on the Omega EP Laser System Using Neural Networks

David Villani

The Harley School

Advisor: Mark Guardalben

June 2023

Laboratory for Laser Energetics

University of Rochester

1. Abstract

OMEGA EP is a kilojoule-class laser system with four independently configured beams. The amplification and wavelength conversion processes are highly nonlinear, and laser performance is currently predicted using a physics-based model, *PSOPS*. An alternative neural network model was created and trained using simulated data from *PSOPS* to predict the required laser input energies for a wide range of output energies and pulse shapes. The network model predictions were within 0.05% of *PSOPS* predictions with 1000x faster processing speed. Although initial training time was 3 hours on a Graphics Processing Unit node, by implementing transfer learning the network was retrained in only 24 minutes to accurately predict a new category of pulse shapes. The network model provides a proof-of-concept for laser performance prediction using neural networks and can be implemented as an efficient, accurate replacement for certain *PSOPS* functions within OMEGA EP shot operations.

2. Introduction

2.1. The OMEGA EP System

The OMEGA EP laser at the University of Rochester Laboratory for Laser Energetics is a four-beam, kilojoule-class, frequency-tripled laser system with independent beam configuration and the ability to produce either nanosecond-scale ultraviolet (UV) pulses or picosecond-scale infrared (IR) pulses via optical parametric chirped-pulse amplification [1]. For the purposes of this paper, we are considering only the former functionality and ignoring the latter. Each beam on OMEGA EP starts with a temporally shaped pulse created by an integrated front-end system from a continuous laser with a waveform generator. This pulse is injected into front-end amplifiers before injection into the main beamline, where it is amplified by flashlamp-pumped Nd:glass in several passes through the primary beamline amplifiers, which can be configured to use a different number of amplifier discs [2]. Finally, the infrared pulse is converted via frequency tripling into an ultraviolet wavelength pulse before reaching the target. This structure is depicted in Fig. 1. Each amplification step and frequency conversion are highly nonlinear processes.

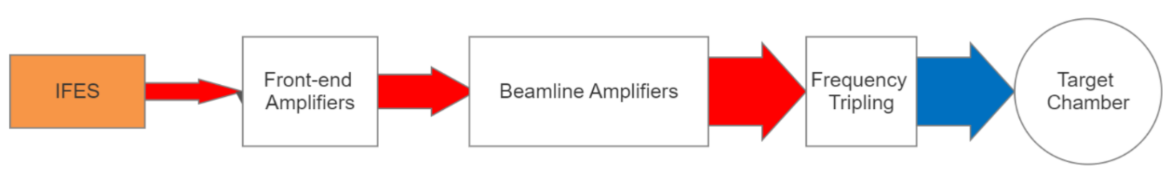

Fig. 1. A simplified diagram of the OMEGA EP laser configuration.
An integrated front-end system (IFES) provides shaped 1053-nm seed pulses from a continuous-wave fiber laser, before amplification and frequency conversion.

2.2. PSOPS and Prediction for OMEGA EP

The ability to produce complex laser pulses is necessary for various fields of high-energy research, such as laser-driven inertial confinement fusion [3]. In such laser facilities, the ability to model, in real time, the energy of a pulse as it travels through a laser system is critical to provide flexibility and optimize operational effectiveness. OMEGA EP uses the *MATLAB*-based PSOPS model for this purpose [4,5]. This model provides accurate, flexible, real-time predictions for pulse shapes, stage energies, and beam profiles in both the forward and backward directions, and allows for real-time optimization of the model between shots using measurements of laser energy, pulse shape, and beam profile at different locations within the laser system. Figure 2 shows the process by which PSOPS predicts required IR laser energy, including the backward prediction step that is discussed in this paper. The machine-learning model discussed herein could replace the physics-based model in this step for faster backward predictions.

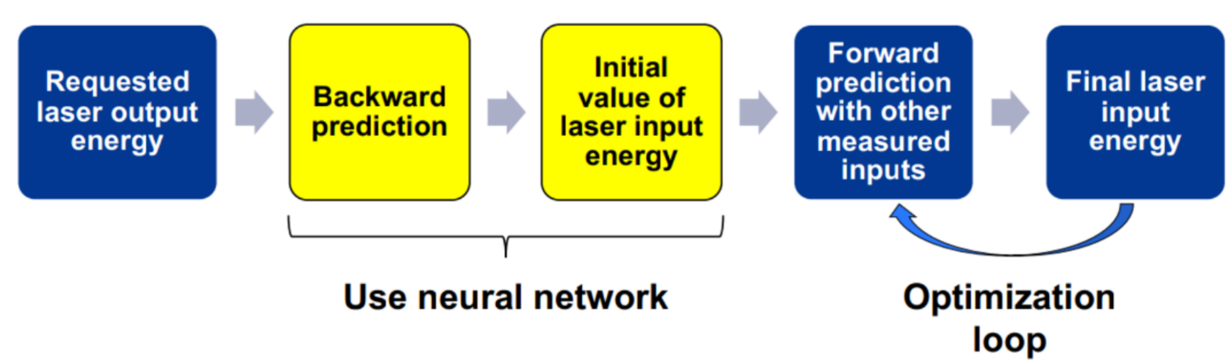

Fig. 2. A diagram of the PSOPS pulse prediction process
To configure the laser for a shot, backward prediction is used to provide an initial value of injected beamline energy for subsequent forward prediction optimization. The neural network discussed in this paper focuses on the backward prediction step, highlighted in yellow.

2.3 Artificial Neural Networks

Artificial Neural Networks, or ANNs, are connectionist optimization models capable of performing a variety of tasks efficiently. An ANN processes input data and passes the processing data through connected neurons.

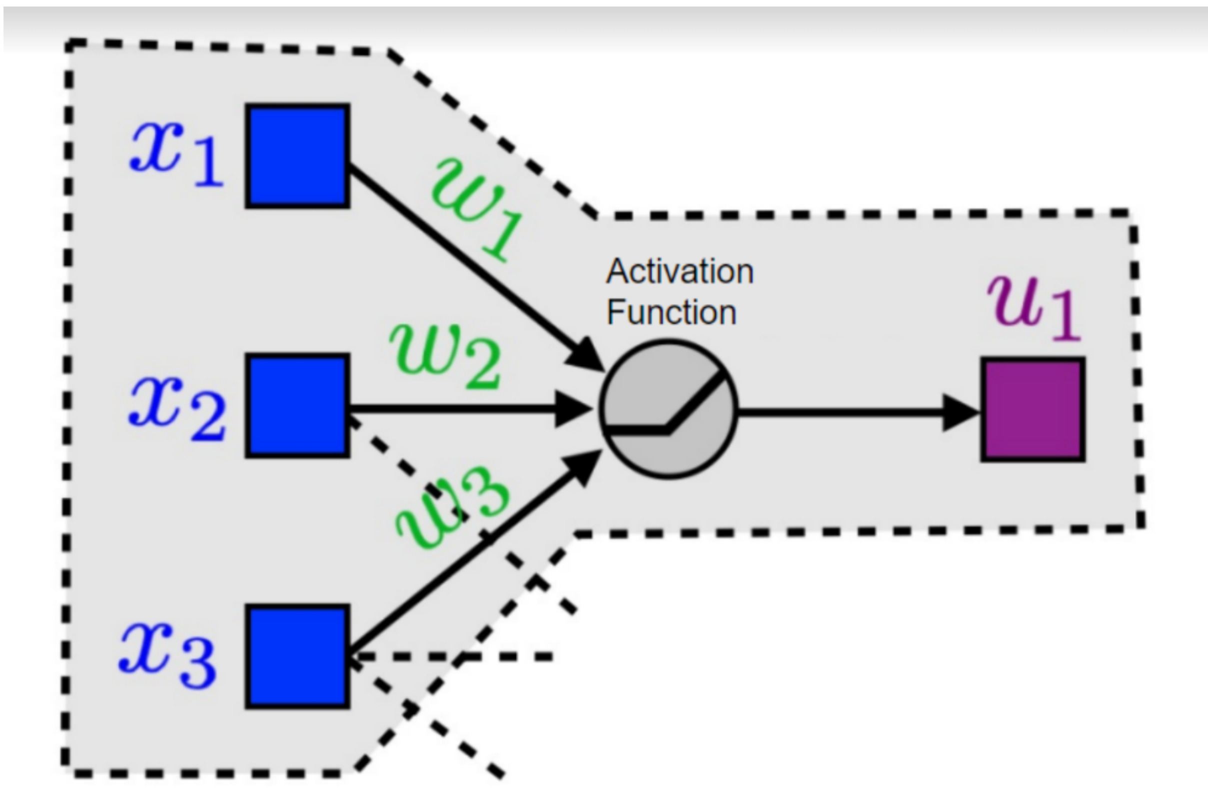

Fig. 3. An example of a neuron in an artificial neural network (based on Ref [6]).
This neuron has three inputs, x_1, x_2, and x_3, which are transformed by the linear functions w_1, w_2, and w_3, respectively. The sum of those transformed values is input into an activation function, in this case a Rectified Linear Unit (ReLU). The output, U1, is then input into the next layer of the neural network. The dotted lines represent how the same data points can also be input into other neurons, with different weights and biases, before ReLU activation.

As shown in Figure 3, each neuron applies a linear transformation (the slope of which is known as the weight and the y-intercept as the bias) followed by a nonlinear activation function to each of its inputs.

Therefore, each neuron of a neural network is a system of equations

$$T_n = \sum_1^k (W_k x_k + B_k) \qquad (1)$$

$$u_n = f(T_n) \qquad (2)$$

where W_k is the weight, x_k is one input, B_k is a bias, T_n is the temporary sum of inputs to the activation function, and u_n is the output of a neuron.

As Fig. 4 shows, the output of each neuron in a layer can then either be collated into output data or input into another layer of neurons, creating a deep neural network. ANNs essentially manipulate and sum nonlinear components to approximate continuous functions. They have a strong ability to learn and apply nonlinear mapping and function modeling, allowing for the modeling of complex behaviors while maintaining low operational complexity due to only summing simple linear or nearly linear functions. The activation function (such as ReLU) provides the nonlinearity required to model complex functions, as otherwise the sum of a series of linear functions will always be linear.

An ANN is initialized with random weights and biases, and then optimized via backpropagation during a training process. Backpropagation is the recursive process by which a neural network is gradually optimized to ‘learn’ a data set, by calculating the error compared to a training data set, finding how to change each weight and bias to best reduce the error, and then iteratively repeating this process.

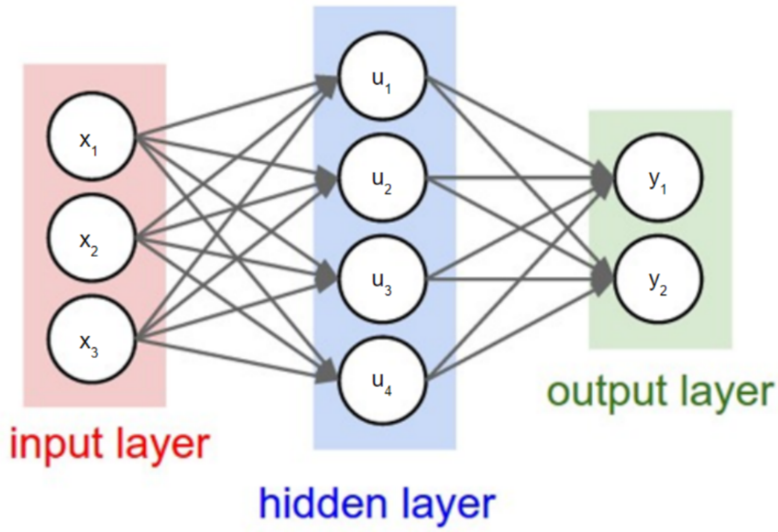

Fig. 4. A diagram of a neural network with three inputs, four hidden nodes, and two outputs.

This is a feed-forward, fully connected neural network, as each neuron in one layer is connected to every neuron in the next. More fully connected hidden layers can be added to increase computational complexity and ability.

Neural networks almost exclusively use backpropagation for optimization. During backpropagation, the Jacobian matrix (a matrix of the first-order partial derivatives of a multivariable neural network) of the ANN is calculated that contains the first derivatives of the errors in the network with respect to each weight and bias. Traditionally, for a feed-forward neural network, error is calculated as the mean squared error (MSE) of each output across the entire training data set. Algorithms such as the Levenberg-Marquardt algorithm [7] (a modified version of Newton's method) are then used to generate approximations of the Hessian matrix H (the square matrix of second-order derivatives) and gradient g of each variable of the neural network,

$$H = J^T J \qquad (3)$$

$$g = J^T e \qquad (4)$$

where J is the Jacobian matrix of the network and e is the set of vectors of network output errors relative to each weight and bias [8]. Following this, the weights and biases of the neural network, w_i, are then incremented with Newton-like updates

$$w_{k+1} = w_k - [J^TJ + \mu I]^{-1} J^Te \qquad (5)$$

where μ is the learning rate (a scalar that is decreased after each successive step of the function by a ratio determined by the gradient descent function) and I is the identity matrix. The number of vectors in e equals the total amount of training data used. This process is repeated until the performance of the neural network ceases to significantly improve, as measured by the mean squared difference between the network output and the known training (or ‘ground truth’) data, or until another termination limit, such as a certain number of epochs of training or a minimum gradient, is reached. Other methods, such as Newton’s Method gradient descent, are possible but typically approach local minima less efficiently, so Levenberg-Marquardt training was used for the model described in this paper.

3. Predictions on Simple Pulse Shapes

MATLAB was selected as the programming language and workspace for the creation of neural networks. The software already has a suite of premade machine learning algorithms and the ability to handle large-scale matrices. Initially, an ANN was developed and trained using PSOPS ‘ground truth’ data to predict simple ESG (EP Super Gaussian) pulse shapes. Fig. 5 shows an example of an ESG output pulse and the input pulse needed to produce it. This system used a number of inputs describing the laser configuration and the requested UV energy and pulse shape, and the injected IR energy as the output. For the model described in this section, unlike that used in the rest of the paper, the Full Width at Half Maximum (FWHM) was used to describe the shape of the UV pulse. Prior models have used a similar system, representing shaped pulses with the FWHM [9].

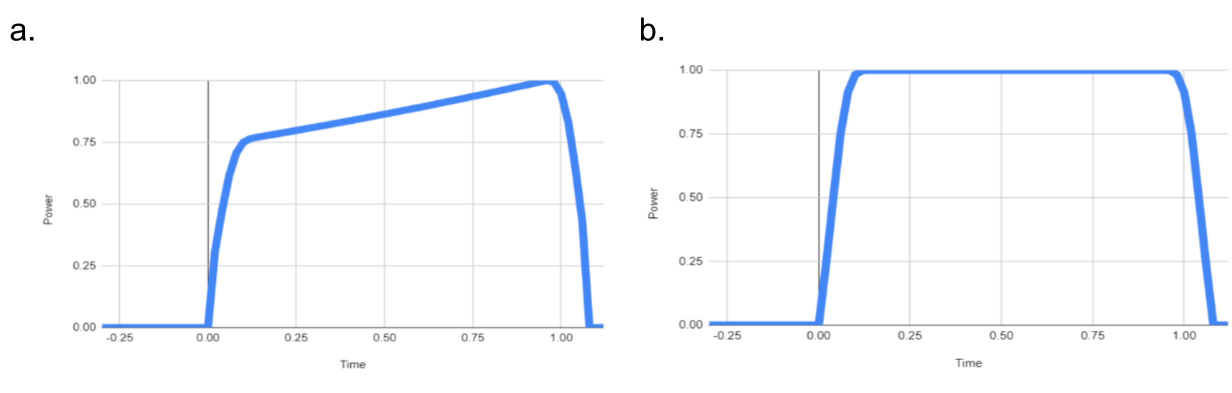

Fig. 5. *Pulse shapes as power (W) versus time (ns), with maximum normalized to 1. (a) The required IR injected pulse prior to beamline amplification to produce the ESG UV pulse shown in (b). (b) The corresponding UV pulse after beamline amplification and frequency conversion.*

This model used a structure with 4 inputs, 2 hidden fully connected layers with 10 neurons each, and a single output node. A Leaky ReLU function, which, unlike a traditional Rectified Linear Unit (ReLU), is capable of negative output, was used as an activation function, and is applied to each layer

$$f(x) = \max(x, x/10) \qquad (6)$$

which provides nonlinearity and combines computational simplicity with the ability to output negative values. The *MATLAB fitnet* function was used to create and train the function-fitting neural network.

The model performed accurately to within ±1% of PSOPS-generated ground truth values across a broad range of ESG pulse lengths and energies. Validation and test data were randomly selected from the initial training data, with 70% of the pulses as training values, 15% as validation, and 15% as test. Validation data are used between model updates during training to prevent an ANN from overfitting by terminating training when error values for the validation data begin to increase rather than decrease. As the neural network is using training data to map to the validation data, test data that are separate from the training process are used after training to determine the quality and accuracy of the neural network. Overfitting occurs when a network adapts too precisely to noise in the training data and its ability to generalize to data it was not trained on decreases.

4. Predictions for Complex Pulse Shapes

For more complicated pulse shapes, such as the ESS (EP Stepped Square) and ERM (EP Ramp) pulses shown in Fig. 6, the FWHM was found to insufficiently sample the pulses. Thus, a new feature set to describe the input pulse shape was developed that consisted of piecewise integrals of the pulse in time. Each pulse was divided into a given number of sections, or bins, and the integral of the pulse section over its width was taken as a pulse shape feature.

Excess zeros were removed from the beginning and end of each pulse, after which the pulse was normalized to a maximum of 1 and divided into 20 concatenated pulse sections.

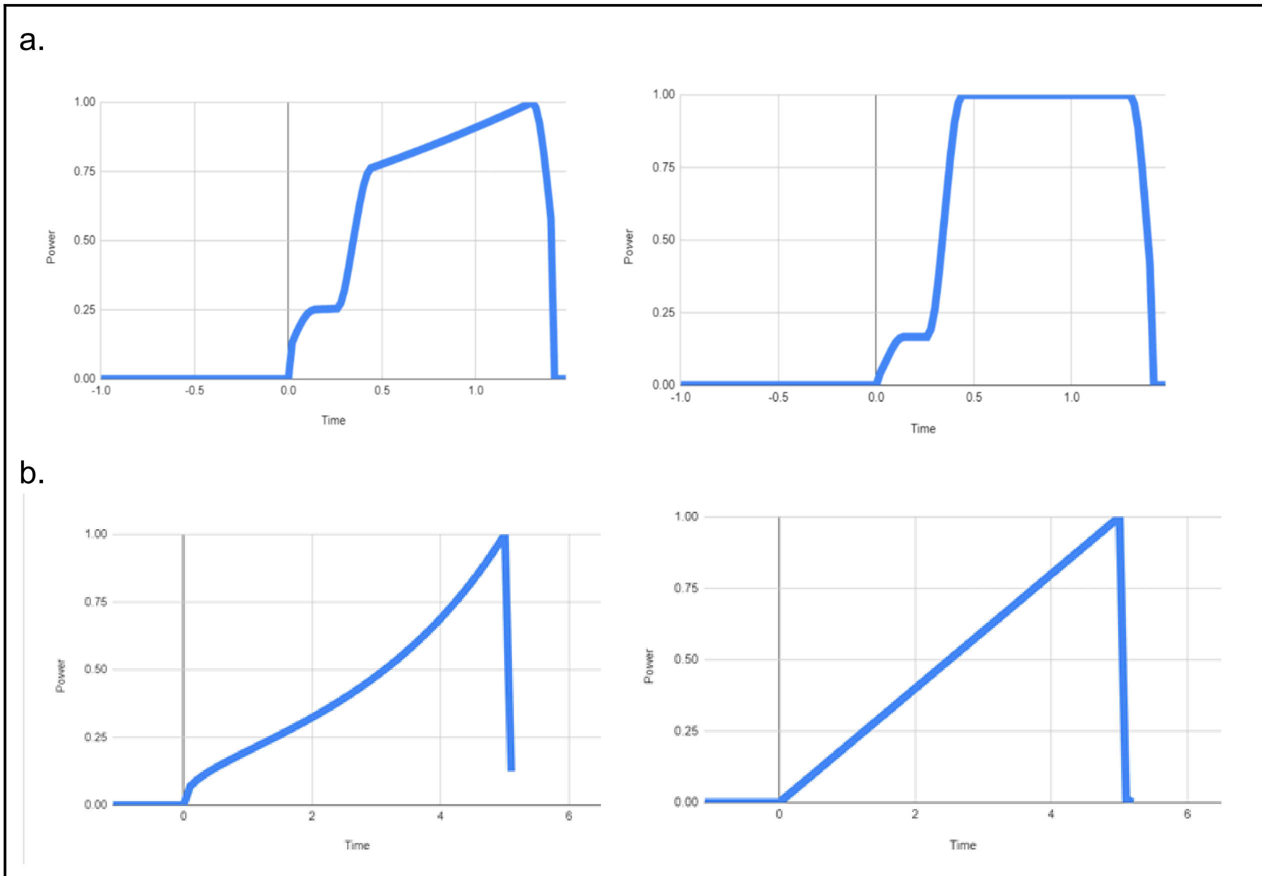

Fig. 6. *As in Fig. 5, but showing (a) ESS (stepped square) and (b) ERM (ramp) pulses before (left) and after (right) amplification.*

This number of pulse shape features was determined by balancing the model's accuracy against training time. Each section was then summed separately, resulting in 20 numerical values to be used as input features to the neural network. Additional input features included a manually input integer modifier (a discussion of which is beyond the scope of this paper), beamline number, number of active amplifier discs, and total UV energy. The output of the network is a single node, the total predicted input energy, which is returned as a scalar value. To account for the increased input complexity, the size of the neural network was increased to two hidden leaky ReLU layers with 30 neurons each. Fig. 7 shows the structure of this model.

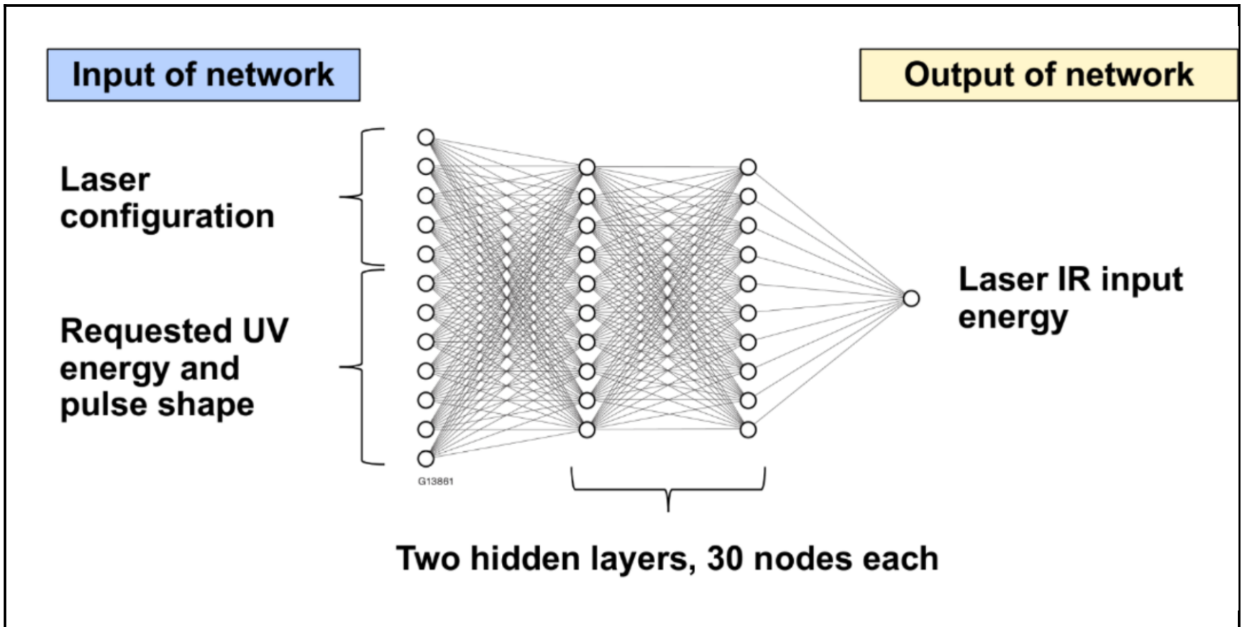

Fig. 7. A graphical depiction of the neural network used for backward prediction
Compared to the earlier, FWHM-based neural network, this model was significantly more complex, with 24 inputs and 2 hidden Leaky ReLU layers with 30 neurons.

This ANN was found to be extremely accurate when tested on ESS, ERM, and ESG pulses similar to those it had trained on, using the same randomly selected 70% training, 15% validation, and 15% test data from 16,000 PSOPS simulations. Fig. 8 shows the error distribution across the test data set, with a mean error of μ = $-2.77*10^{-5}$% (defined as the average of the difference between true and predicted data, and used here as a measure of skew), and a standard deviation $\sigma=1.2*10^{-3}$%.

There is an increased percentage error for low injected energy values. This is due to the ANN training to minimize mean squared error (MSE), which incentivizes reducing percent error in higher energy pulses. This could be avoided by backpropagating in a manner that minimizes the mean squared percentage error rather than MSE.

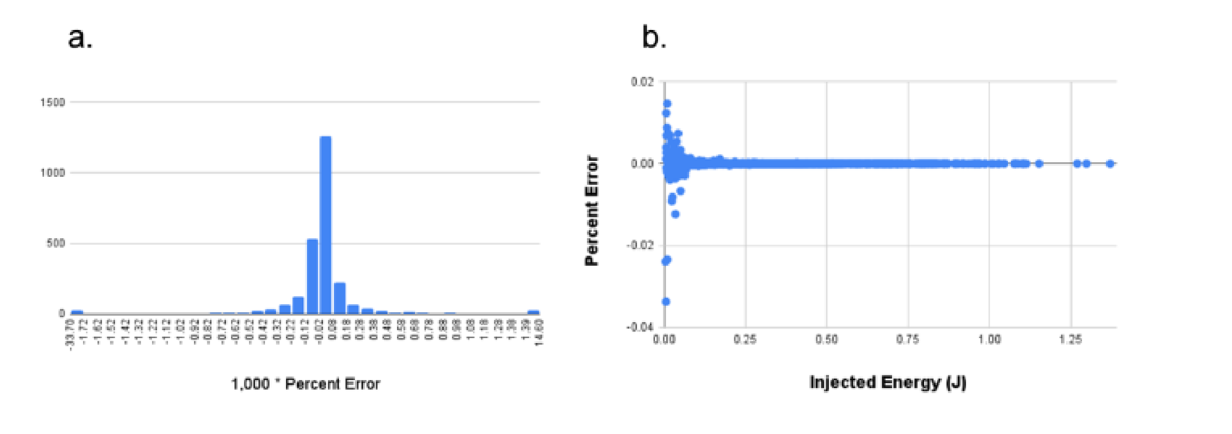

Fig. 8. Visual representations of errors in ANN predictions.
(a) A histogram of prediction percent error for the test data set. There is no significant skew in either the positive or negative direction. (b) A graph showing percent error versus injected energy (in Joules). Notably, low-energy pulses have a significantly greater percent error, as described in the text.

5. Processing Speed and Transfer Learning

Training and predictions were run on a GPU node at the University of Rochester's Laboratory for Laser Energetics Computing Facility with the Storm High Performance Computing system. The average training time across several models was approximately 3 hours, and prediction time on test data was an average of 5.4 ms. The current PSOPS system has an average prediction time of 4-6 seconds, marking a nearly thousandfold increase in prediction speed for the ANN.

Transfer learning is the process by which a pre-trained neural network is retrained with new data in order to decrease the amount of computation required to predict new types of data. The ANN can be retrained easily to incorporate new data with a high degree of accuracy, providing high agility.

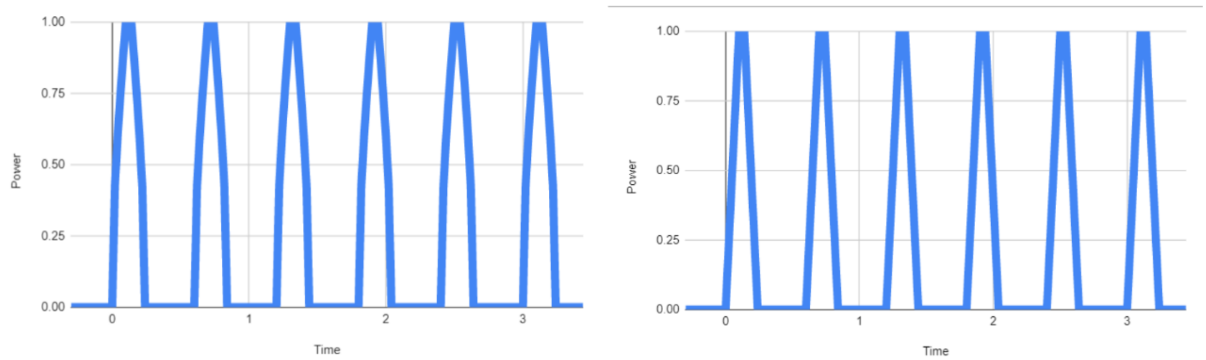

Fig. 9. *As in Fig. 5 but showing the normalized shapes of picket (EPM) pulses before and after amplification and frequency conversion.*

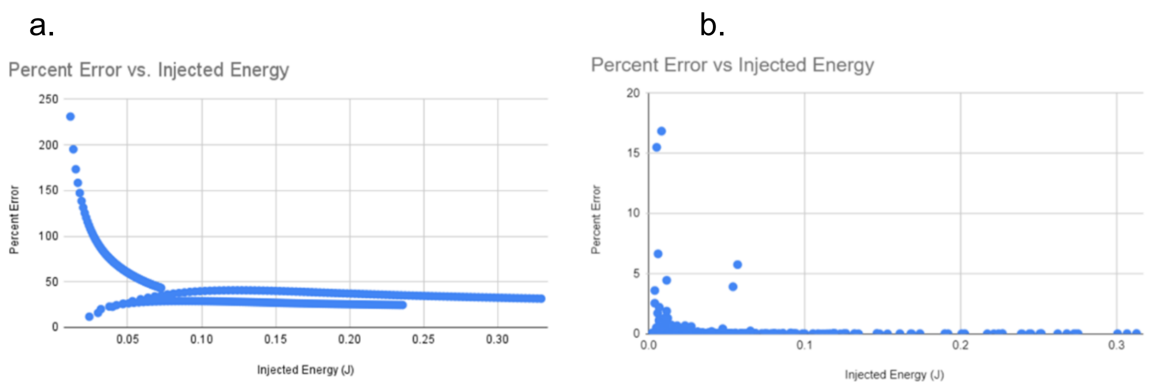

Fig. 10. Percent error versus injected energy for picket pulses before and after retraining.
(a) All provided picket pulses were input into the ANN without retraining, leading to substantial errors. (b) After retraining with 70% of the picket pulses, the randomly selected test data set has significantly less error across the range of injected energies.

A new and dramatically different type of pulse, picket pulses (EPM), as shown in fig. 9, were input into the ANN without retraining. Fig. 10 shows the error before and after retraining the neural network for picket pulses. A comparatively small number of new pulses (300 as opposed to 16,000) were used for retraining, leading to decreased accuracy compared to previous training.

The retraining of the neural network started with the already-trained network, on both the picket pulses and the previous pulse shapes. The ANN took an average of 00:24:13 to retrain for picket pulses, and maintained high accuracy predictions on ESS, ESG, and ERM pulse profiles. No significant change was observed in single pulse prediction time. This transfer learning would likely improve retraining time compared to the initial 3-hour training even with more picket pulse values, as the majority of the computational intensity is due to maintaining accuracy on the larger amount of previous (non-picket) data.

6. Conclusion

A fully connected, feed-forward neural network model was created and used to predict the required OMEGA EP IR input energy for a given UV output energy, laser pulse shape, and laser configuration. The difference between predicted energies using the ANN and the physics-based PSOPS model was less than 0.05%, and energy prediction using the neural network was approximately 1000 times faster than the physics-based model. Accurate prediction was achieved across a broad range of laser pulse shapes and output energies. Furthermore, transfer learning was found to reduce training time for new pulse shapes by 88%. The prediction accuracy and speed of the ANN model demonstrates the viability and potential advantage of using neural networks for laser performance prediction.

ANNs have several other advantages. The ANN discussed here was trained with noiseless, predicted data, but a neural network can be trained on actual pulses and can account for several variables, such as humidity, that are difficult to predict with high-energy physics models. On the other hand, it is challenging to maintain the calibration of the model, and retraining might be required to account for changes in the laser system over time. Further research is needed to determine the viability of ANNs with greater predictive capability, for example forward energy calculations and pulse shape predictions, as well as neural networks trained on raw data as opposed to simulated noiseless data. With more diagnostic information and further development, such models could replace the current physics-based predictive models with neural networks that could considerably increase the operational efficiency of the laser system.

7. Acknowledgements

I would like to thank those who assisted with the completion of my project. First, I thank Dr. Craxton for giving me and the other interns in the LLE summer internship program the opportunity to work in a research environment. I would also like to thank my fellow student interns in the Annex Conference Room. Their dedication and hard work created an environment that supported each of us in our various projects, as well as making the summer much more enjoyable. Finally, I give special thanks to my advisor, Mark Guardalben, for the amount of time and effort he has devoted to my project, none of which would have been possible without his guidance. I am extremely grateful for his efforts and advice.

References

1. L. Waxer, J. Kelly et al. "The OMEGA EP High-Energy, Short-Pulse Laser System," in Frontiers in Optics 2008/Laser Science XXIV/Plasmonics and Metamaterials/Optical Fabrication and Testing, OSA Technical Digest (CD) (Optica Publishing Group, 2008), paper FWQ2.
2. J. Kelly, R. Jungquist et al., "Optical Engineering of the OMEGA EP Laser System," in Frontiers in Optics 2008/Laser Science XXIV/Plasmonics and Metamaterials/Optical Fabrication and Testing, OSA Technical Digest (CD) (Optica Publishing Group, 2008), paper OFB2.
3. D. Cao, T. R. Boehly, M. C. Gregor, D. N. Polsin, A. K. Davis, P. B. Radha, S. P. Regan, and V. N. Goncharov, “Theoretical quantification of shock-timing sensitivities for direct-drive inertial confinement fusion implosions on OMEGA,” *Physics of Plasmas*, vol 25, 052705 (2018)
4. MATLAB®R2013b, The MathWorks Inc., Natick, MA 01760-2098 (http://www.mathworks.com)
5. M. Guardalben, M. Barczys, B. Kruschwitz, M. Spilatro, L. Waxer, & E. Hill, “Laser-system model for enhanced operational performance and flexibility on OMEGA EP,” *High Power Laser Science and Engineering, 8*, E8. doi:10.1017/hpl.2020.6 (2020)
6. G. Peyré (2020) “Mathematics of Neural Networks,” unpublished manuscript. Retrieved from https://mathematical-tours.github.io/book-basics-sources/neural-networks-en/NeuralNetworksEN.pdf
7. J.J. Moré (1978). “The Levenberg-Marquardt algorithm: Implementation and theory.” In: Watson, G.A. (eds) Numerical Analysis. Lecture Notes in Mathematics, vol 630. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0067700
8. M.T. Hagan, and M. Menhaj, “Training feed-forward networks with the Marquardt algorithm,” *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1999, pp. 989–993, 1994.
9. L. Zou, Y. Geng, G. Liu, L. Liu, F. Chen, B. Liu, D. Hu, W. Zhou, and Z. Peng, "Laser energy prediction with ensemble neural networks for high-power laser facility," Opt. Express 30, 4046-4057 (2022).