Analysis of CubeSat Langmuir probe data with a neural network[a)]

Matthew Lazo[b)]
Department of Physics, West Virginia University, Morgantown, WV, 26506, USA

(Dated: 9 December 2020)

Abstract; Plasmas across a range of applications are often analyzed with Langmuir probes, which sweep a bias voltage across the probe tip at several kHz and produce a large volume of data to be processed for each current-voltage characteristic. Traditionally, each probe trace is manually truncated and fitted to derive relevant plasma parameters such as temperature and density. Standard Langmuir probe theory is well-understood for Maxwellian plasmas, however, and probe characteristics can be simulated as a piecewise function with noise. Therefore, an opportunity arises to build an automated, neural-network based workflow for extracting plasma parameters. The NN is trained on simulated space plasma data, then trained further on real, human-labelled space plasma data. A discussion of the network's performance is given, noting further steps which must be taken to fully implement the automated workflow.

I. INTRODUCTION

A *plasma* is defined broadly as a hot, ionized gas, and can exist in a wide range of natural and artificial settings. It is often referred to as the fourth state of matter. The Sun generates energy by fusing extremely hot, dense plasma particles in its core; manufacturers use atmospheric-pressure plasmas to etch circuit boards; and stars are born in vast, lightyear-spanning clouds of charged dust[1]. We turn our attention to the Earth's upper atmosphere, where NASA's Simulation-to-Flight (STF-1) CubeSat orbits about $500km$ above the surface, monitoring *space weather*. The plasmas found orbiting the Earth along its magnetic field lines at this altitude are fairly cool and diffuse compared to many other plasma settings[2], and cause the spectacular aurora at the Earth's magnetic poles. They can interact strongly with electromagnetic radiation or electronics, interfering with solar wind or communications between the Earth and high-altitude satellites, even causing electrical issues with equipment submerged in a particularly turbulent local plasma.

Plasma physics has become extremely important in manufacturing, astrophysics, and is required for nuclear fusion and solar physics. Basic diagnostic techniques using a Langmuir probe are employed across the spectrum of plasmas, allowing one to determine key parameters such as the density and temperature of the plasma, as well as particle distribution functions[3]. These often produce high volumes of data, which, for well-explained plasmas, enables the use of deep learning to save many hours of human data analysis. Maxwellian plasmas in particular are well-understood, and we often find much use for Langmuir probe in observing Maxwellian plasmas in a variety of applications. In this work, we develop a neural network which learns to analyze space plasma data from STF-1, opening the door to implementation of a fully automated data analysis workflow, and hence monitoring of the Earth's magnetosphere. This tool is built as an extension to the project by Lazo[4] in which Langmuir probe data from the National Spherical Tokamak Experiment (NSTX) is used to train a neural network to analyze very high temperature, high density fusion plasmas.

We discuss the implementation of a theoretical model to describe the presumed Maxwellian plasmas surrounding STF-1, then give a brief introduction to neural networks and deep

---

[a)]Internal report on final project for the undergraduate computational physics course at West Virginia University

[b)]Electronic mail: mjl0018@mix.wvu.edu

learning. The technical specifications of the analysis program are discussed, and results from training the neural network and evaluating its performance on both synthetic and real space plasma data are presented. A discussion is given on the shortcomings of the theoretical model as of the time of writing, with a look forward to the improvements to come and further plans of full program implementation.

A. Langmuir probes

The most basic plasma diagnostic techniques were devised by Irving Langmuir with his invention of a simple instrument, the Langmuir probe[3]. Consisting of a thin wire or metal cylinder and attached to a voltage supply, the probe is partially inserted into the plasma of interest. A bias voltage V_b is placed across the probe tip, and swept in time from a negative to a positive value relative to the electrostatic potential, V_p of the surrounding plasma. As V_b is swept in time, corresponding values of ion and electron current are collected as particle flux per unit time to the probe surface[3]. When the probe collects zero total current, it is at the floating potential, V_f. The time series data produced from this sweep is referred to as a *current-voltage characteristic*, or I-V trace. Figure 1 shows an example trace from Hutchinson, and works as a reference, despite being an inadequate example for ionospheric plasmas.

Three regions of the trace should be noted from Fig. 1. When V_b is below the floating potential, the probe collects only ion current, and this is the *ion saturation region*, characterized by the ion saturation current, I_is. This will be one of our important plasma parameters. For $V_f \leq V_b \leq V_p$, we pass zero total current and begin collecting exponentially higher electron current over ion current. This is the *transition* region, and from which we derive our second important parameter, the electron temperature, T_e, from a linear fit to the curve in this region. Passing V_p, we have entered the *electron saturation* region, where ion current drops rapidly to zero, and we are left only with electron flux to the probe.

As the probe collects current, a potential difference arises between the probe surface and the surrounding plasma far from the probe. This sets up a small electric field, encouraging nearby electrons and ions to move toward the probe, electrically shielding it in a thin layer of particles. Past this layer, the plasma cannot "see" the probe at a potential different from V_p, and the probe is said to be surrounded by a plasma *sheath*. The sheath is measured on the scale of the *Debye length*, λ_D:

$$\lambda_D = \left(\frac{\epsilon_0 k_B / e^2}{n_e/T_e + n_i/T_i} \right)^{1/2}, \tag{1}$$

where n_e and n_i are electron and ion densities, respectively, in m^{-3}, T_e and T_i are in J, k_B is Boltzmann's constant in J/K, and ϵ_0 is the vacuum permittivity. For typical laboratory plasmas, this value is on the scale of mm to nm, shrinking with increasing density and temperatures. Ionospheric plasmas tend to have very low temperatures and densities[2], and so λ_D is on the order of cm or m in the plasma surrounding STF-1.

As charge carriers form the sheath, it will rapidly expand – at any length scale – to a stable size for the local plasma. Because of the scale of λ_D in low-Earth orbit ($500km$), the spacecraft itself is constantly *within* the plasma sheath, and so the probe will always notice the expansion of the sheath[1,3,5]. It is for this reason that Fig. 1 is not an accurate model for ionospheric plasmas, because the sheath contains mostly ions, and so as it expands, the ion current will increase (usually linearly[5]) with the sheath. This is addressed again later, and is an important distinction between ionospheric and other plasmas[2].

B. Neural networks

The recent rise of machine learning, deep learning, and artificial intelligence across just about every research and engineering discipline has huge implications as we continue to

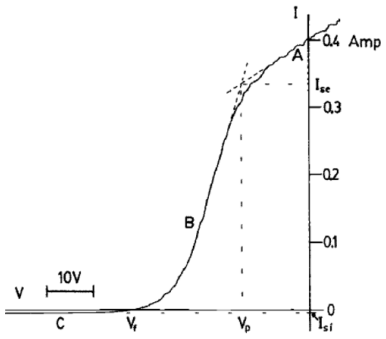

FIG. 1. Example "textbook" current-voltage characteristic by Hutchinson. The trace regions can be deduced from the marked locations of V_f and V_p. The regions are electron saturation, transition, and ion saturation as labelled by A, B, and C from right to left. While a trace of this shape is no reminiscent typical space plasma data, it is a good reference.

advance our knowledge. Coupled with the coming rise of quantum computing, it would not be unlikely to say that these tools will truly revolutionize the world – more than they already have.

Neural networks are a very common tool in machine and deep learning, and are used to solve to broad types of problems: classification and regression. A smartphone camera recognizing a cat has solved a classification problem, whereas a voice assistant interpreting one's request for the daily weather addresses a regression problem. The image of a cat, or thousands of different images of millions of different cats, share common patterns which allow the network to recognize the pixel patterns of a cat when it sees one. For our purposes, time-indexed voltage and current values constitute a complex structure from which we can determine a few key parameters. There are not just three or four possible temperatures for a plasma, and so a neural network tackling a regression problem must learn to perform linear regression on the data to make a determination of these parameters.

A neural network is a large function, often consisting of hundreds of thousands of parameters, which passes a series of input data through several layers, or steps. Each value is stored in a neuron – a node with one value. Each neuron in each layer is connected to each neuron in the following layer by a unique statistical weighting, w_i. In passing a value from a neuron in one layer to a subsequent layer, the value is considered by every neuron in the subsequent layer according to the weight assigned to the "path" between two neurons. The passing of input values through several layers and many neurons eventually enables the network to discern patterns among data, and make a decision about how to classify or calculate an output value. Because I-V traces consist of two time series whose values are correlated by index, we give two dimensions to the network to consider. In this work we choose to output values for electron temperature and ion saturation current, and so have two dimensions of output from the network. Including more plasma parameters increases the dimensionality, and hence the complexity, of the network.

Furthermore, because voltage and current are time series – that is, described by a function which evolves in time – one value of the data influences the next or is influenced by the previous. There is some determination to be made about the points surrounding a given point based on the context of our current location in the I-V trace. Therefore, we implement a recurrent neural network, depicted in Fig. 3 as the repetition of a standard network over many steps to build an understanding of the entire time series. We employ a Long Short-Term Memory (LSTM)[6] recurrent neural network for the purpose of teaching the program to diagnose I-V traces because of the additional layer of time evolution. That is, we want the program to remember the context of one I-V trace versus another, enabling it to understand the time evolution of plasma data over several seconds, given the correct model to train on.

To solve the linear regression problem of diagnosing space plasma, the task of the neural network is to find the optimal parameters to describe the linear fit for the data at hand.

This is handled by one of several different optimization algorithms; here, we use the Adam optimization algorithm[7]. The training performance of the network is interpreted with the mean-squared error (MSE) metric,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - Y_i^p\right)^2, \tag{2}$$

where n is the number of points, Y_i is the actual value of each data point, and Y_i^p is the value of the predictions. The overall network performance is evaluated by the coefficient of determination[8] or "R-squared", R^2, value associated with each fit. A value of $R^2 = 1$ implies that the network parameters perfectly describe the data, and $R^2 \leq 0$ implies the opposite.

We will utilize an LSTM implementation written by Google, and therefore do not hash out the detailed mathematical definition of the function here. However, we should take note of the *activation functions* which serve a key role in neural networks. These functions will act on data as it is passed from our LSTM layers to the remaining standard layers in the network, and between each standard layer. An activation function will map the data onto a certain range, and this data must be normalized for use with the deep learning library we employ. Specifically, we utilize the *sigmoid* activation,

$$S(x) = \frac{1}{1+e^{-x}}, \tag{3}$$

which maps points to the range $(0, 1)$, and the *hyperbolic tangent* activation,

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{4}$$

which maps points to the range $(-1, 1)$.

We employ supervised deep learning techniques to train a neural network to determine values for two characteristic plasma parameters, T_e and I_{is}. The network must learn to solve a regression problem for predicting these labels, and so we "supervise" the learning by providing the network with the correct labels for each training data set, allowing the network to immediately adjust the internal weight w_i to improve the regression model.

II. PROGRAM STRUCTURE

The program is implemented in the Python programming language and utilizes Tensorflow 2.3.0[9], an open-source machine learning library developed by Google. A public GitHub repository containing the program source code used in this project can be accessed at *https://github.com/Lazeau/PHYS301_final_project_lazo*. The overarching structure of the final program follows the flowchart in Fig. 4, and will consist of two training stages followed by an implementation stage. Random values for selected plasma parameters are generated, and synthetic time series are created from these, based on a theoretical model[5]. The network learns to make predictions on this data, and is then given real, labelled data to make *predictions* on in the second stage. As of the date of writing, the author must still undergo implementation of the *training* for the second stage, and the implementation of the fully automated workflow. The current limitation of the second stage to only predictions is intended to allow for direct comparison between network performance on fusion plasma data[4] and the space plasma data discussed here. Further plans include transfer learning between these implementations.

The neural network is implemented with four bidirectional LSTM layers, allowing it to analyze data forwards and backwards to deduce patterns. Following these are three standard Tensorflow Dense layers, with the number of neurons per layer decreasing from 128 to 2 over the seven total layers. The first standard layer utilizes a sigmoid activation

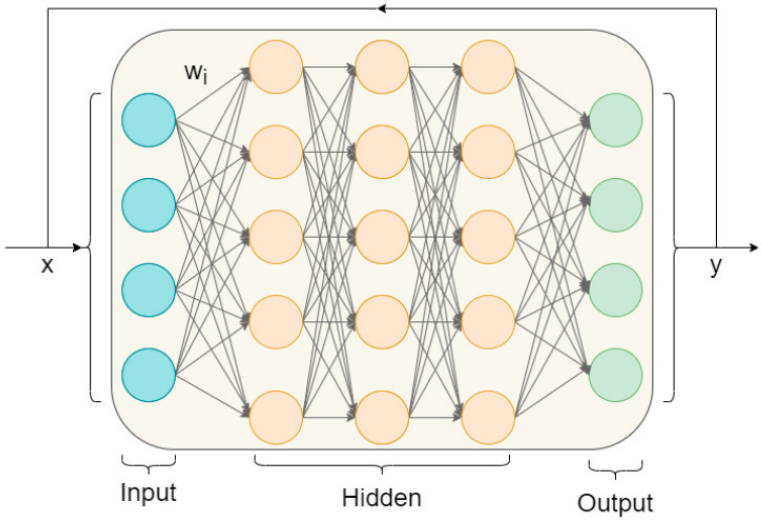

FIG. 2. A basic neural network. Input values, x, are fed into the algorithm and each has an associated weight, w_i. As the network learns to address the problem at hand, the weights are adjusted based on the inputs to produce the desired output values, y.

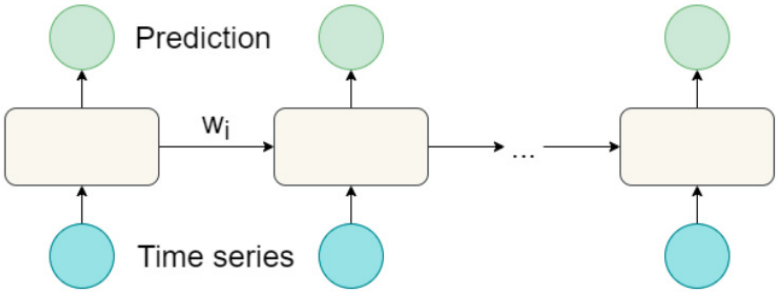

FIG. 3. A recurrent neural network. This structure builds off the neural network structure from Fig. 2, where the output values y are fed back into the network for repeated passes over the same data in order for the network to learn to solve the problem at hand. An LSTM network is a flavor of recurrent neural network, using activation functions and several comparisons between values to determine the output.

to map data output from the last LSTM layer. The two following standard layers use a hyperbolic tangent activation, mapping inputs from the sigmoid layer across a wider range. The Tensorflow model is compiled with the Adam[7] optimizer, using the MSE loss metric through each training epoch. The R^2 value is used as the final metric for measuring network performance. This value determines how well the network has performed in the predictions it is asked to make at the end of each training stage. This network architecture is the same as that used by Lazo in their work analyzing Langmuir probe data from the NSTX fusion reactor, although the number of training epochs and size of synthetic data training set differ. A few samples of synthetic I-V traces and real STF-1 I-V traces are shown in Fig. 5.

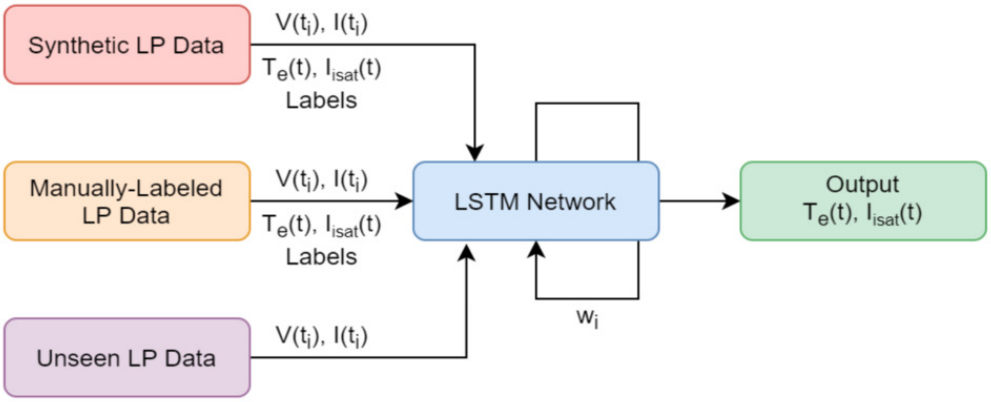

FIG. 4. Flowchart of the data analysis workflow. Synthetic Langmuir probe time series are first created from randomly generated T_e and $I_{i,sat}$ within some specified range, and these labels are used to create Langmuir probe time series. The labels and features are passed to the LSTM network for training. Real, manually-labelled probe data is then used for a second round of training in the same fashion. When both training stages conclude, the program is given unlabelled time series to make predictions on. Internal weights in the network, w_i, are adjusted with each pass over the training time series. Each set of current and voltage time series has one corresponding temperature and saturation current label, hence the input time series are on a finer scale than the labels, which can be accumulated to analyze the evolution of a plasma in time.

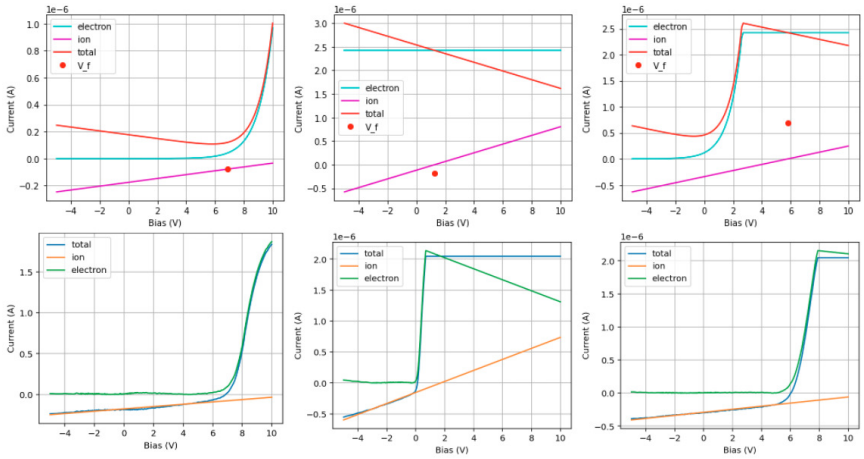

FIG. 5. Samples of synthetic traces (top three) and real traces (bottom three). The ion and electron currents are plotted in addition to the total current to show their relationship. In the synthetic traces, the point of floating potential is plotted to demonstrate that, while the first synthetic trace constructed based on the below STF-1 trace appears accurate, this is not the case for almost any other trace generated by SpaceBox. In the top-middle trace, we believe that this is a trace shifted entirely into the electron saturation region. The top-right trace is usable, but has an inaccurate floating potential as determined by SpaceBox.

A. Training on synthetic data

In the first training stage, 200,000 I-V traces are generated using T_e in eV and I_{is} in A drawn from a Numpy uniform distribution according to the typical bounds for these parameters in the regime of ionospheric plasmas[2]. We use electron temperature because it is a common characteristic plasma parameter. Particle density is typically used with temperature to characterize a plasma, however our usage of ion saturation current allows us

to directly calculate particle density, and avoids floating point errors associated with values on the order of $10^9 - 10^{20} m^{-3}$. Random values for T_e are thus selected between $0.1eV$ and $8.5eV$, and values for I_{is} are selected between $-2.0*10^{-7}A$ and $8.0*10^{-7}A$. Two additional parameters, β_m and β_b, which characterize a linear fit to the probe ion current before the floating potential is reached, are also chosen within a range determined experimentally from STF-1 data.

The `SpaceBox.py` class defines the "black box" object which generates the synthetic data. Invoking the object's `__call__()` function will yield several Numpy arrays: an array of times spanning the window of the trace over the voltage sweep period; a voltage sweep time series for each I-V trace; a current sweep time series for each I-V trace; the T_e labels for each trace; and the I_{is} labels for each trace.

The Langmuir probe on-board the CubeSat has a built-in cutoff value for electron current collection, set at $2.420\mu A$ so as to avoid damage to the instrument. We therefore enforce this cutoff in the synthetic data, as well. The total current as a function of bias voltage is the combination of electron current I_e and ion current I_i, in amps, as $I = I_e - I_i$, where positive current is defined as electron particle flux to the probe in time. The class `lanngmuir.py` contains functions for I_e and I_i. The ion current is generated as a linear function of voltage,

$$I_i = \beta_m V + \beta_b, \tag{5}$$

where β_m and β_b are allowed to vary randomly in an experimentally-determined range due to expansion of the plasma sheath with increasing bias voltage. The electron current utilizes the exponential dependence on voltage discussed previously. The voltage is shifted, however, by the floating potential V_f so that,

$$I_e = I_{es} exp\left(\frac{V - V_f}{T_e}\right), \tag{6}$$

where I_{es} is the electron saturation current in A, and T_e is in eV. Once passed back to the `SpaceBox` object, the electron current is artificially capped at the STF-1 cutoff value.

Once all of the desired synthetic I-V traces are constructed, the labels and time series data are passed to the main program script, `network.py`, where the labels and time series data are normalized and converted to Tensorflow `Tensor` objects, and used to create a `Dataset` object for the training data.

The program will attempt to load a saved Tensorflow model from the local `model` folder, or default to creating a new model if one is not found. In the latter case, a Keras `Sequential` model is instantiated with the specified seven-layer network structure, Adam optimizer, and MSE loss function, and then trained on the training set over 100 epochs with an evaluation interval of 200, additional validation set of 512 traces in validation steps of 50, and with Keras `EpochDots()` callbacks. The training loss is saved in a plot, and the network is asked to make 5,000 predictions on the validation set. The training predictions are passed to the `r2plot.py` script to handle R^2 plotting of normalized labels versus normalized predicted values.

B. Training on STF-1 data

The second training stage picks up where the first left off, using a series of real I-V traces instead of synthetic ones. This stage comes second because real data often has noise or artifacts which should not be interpreted by the network to impact the labels, and also because it is far easier to generate 200,000 "textbook" traces rather than acquire nad process 200,000 real ones (the acquisition of adequate real data is a significant part of the work in machine learning, generally).

The class `stf1.py` defines two functions, `read_data()` and `analyze_data()`, which can be called independently to fully analyze an STF-1 data set, spanning one or several polar orbits about the Earth. The size of each data set varies according to equipment status

TABLE I. Table of pre-processed STF-1 data files available for use with the to-date distribution of the program.

Date (YYYY-MM-DD)	Index	Traces in File
2019-01-16	0	734
2019-02-27	1	244
2019-04-05	2	101
2019-04-19	3	254

and local space weather conditions – for example, the spacecraft may shut down to avoid electrical overload from the local plasma, or may shut down as a result of it, interrupting data collection. The class is equipped to handle certain defined files, characterized by date and referenced by a zero-indexed key. This is because each data set must be individually treated upon receipt from the CubeSat. To date, the available files for use are detailed in Table 1. That list contains all available 2019 data, and 2020 data will be included with following releases.

The `read_data()` function imports and formats the raw spacecraft data, converting bias and measurement values to SI units of volts and amps, respectively. Timestamps are recorded for each trace, and the data are time series by default. The `analyze_data()` function follows the conventional Langmuir analysis procedure detailed by Thompson. It calculates and returns Numpy arrays with a first-dimension length equal to the number of traces in the data file, and second-dimension length of 301, for the 300 zero-indexed measurements. The returned arrays contain the parameters T_e, I_{is}, ion density n_i, V_f, and the slope and intercept for the linear fit to the ion current before V_f, β_m and β_b; all are in SI units, with T_e in eV.

Once passed to `network.py`, the time series data and labels from the STF-1 file undergo the exact same normalization and conversion process as the synthetic data. Then, the network is asked to make 5,000 predictions on the real traces. The predictions are passed to the `r2plot.py` script to handle R^2 plotting of normalized labels versus normalized predicted values. At this point in development, predictions on the real data are made without the full second-stage training to assess the accuracy of the theoretical model in describing real plasma data.

III. RESULTS

The network was run with the aforementioned training parameters, and the network's performance in making 5,000 predictions on ionospheric plasma data is plotted on a 2D histogram in Fig. 6. The top two plots show performance on synthetic data, and the bottom two plots show performance on real data. These predictions follow just the first training stage shown in Fig. 4, as the second training and final implementation stages must still be completed.

It is interesting to ask the network to make predictions on real data sets after training only on synthetic data to get a qualitative measure for how well our theoretical model truly predicts real data. Figure 6 indicates that our model could use some work – and this is discussed deeper momentarily – but, even with exposure only to the model, the network is able to make some accurate predictions on real data. This can be contrasted with the predictions shown in Fig. 7, taken from Lazo[4]. These plots show this implementation applied to fusion reactor data from NSTX. The fusion plasma and ionospheric plasma models differ, as do the temperature and saturation regimes for these two plasma settings. However, since both seek to diagnose Langmuir probe data, we can notice that our fusion model seems to be very well-defined, as far as our network is concerned. However, when exposed to real data, the network clearly has no idea what it is doing, as real R^2 values cannot be calculated.

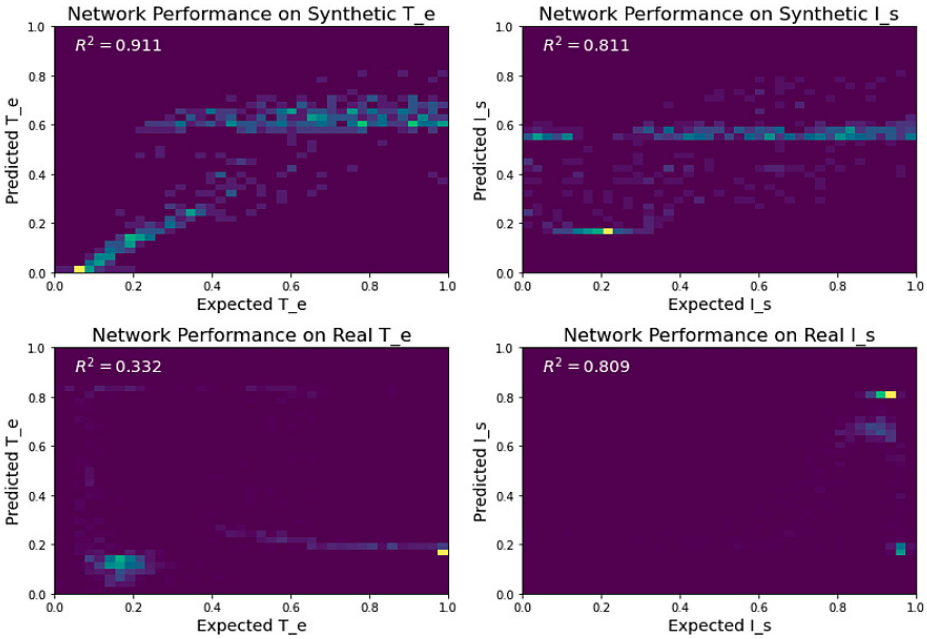

FIG. 6. Results of 5,000 network predictions of electron temperature and ion saturation current for ionospheric data. The top two plots show performance of the network making predictions on synthetic data. The bottom two plots show performance of predictions made on real data. The coefficient of determination is used as the metric to measure performance. These were the result of training on 200,000 synthetic data sets for 30 minutes.

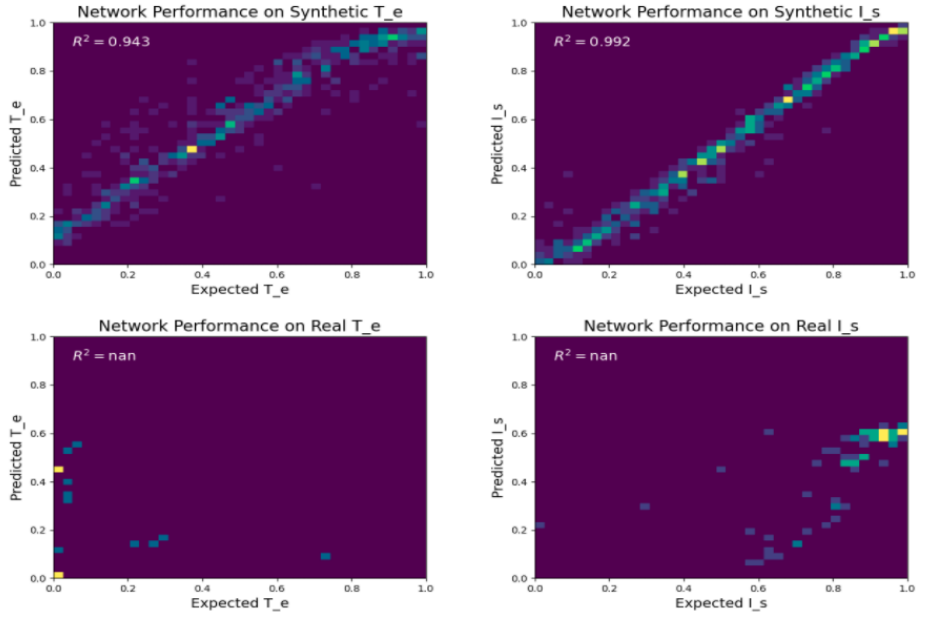

FIG. 7. Results of 5,000 network predictions of electron temperature and ion saturation current for NSTX fusion data. The top two plots show performance of the network making predictions on synthetic data. The bottom two plots show performance of predictions made on real data. The coefficient of determination is used as the metric to measure performance. These were the result of training on 100,000 synthetic data sets for 2 hours.

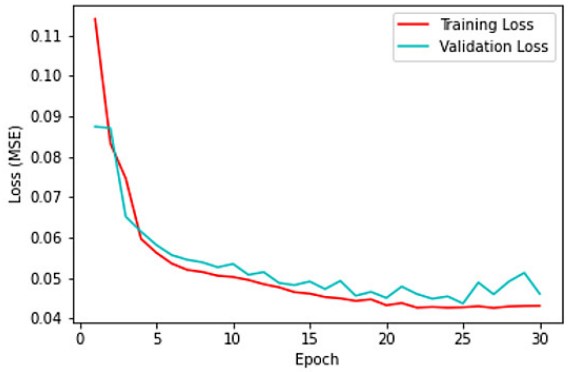

FIG. 8. Mean squared error (MSE) loss versus epochs. Training and validation loss over 30 epochs after constructing the network. A significant relative drop in loss is visible, seeming to level off toward epoch 30.

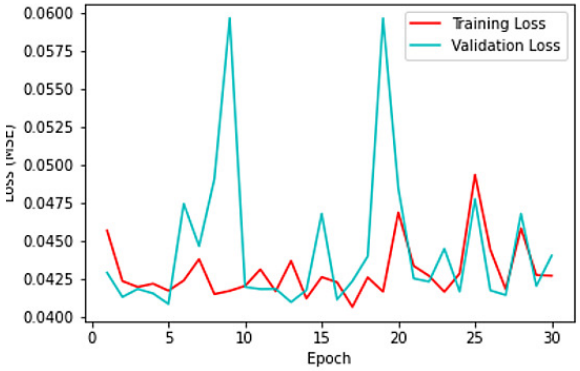

FIG. 9. Mean squared error (MSE) loss versus epochs. Training and validation loss over 30 epochs for a second round of training, following the 30 epochs plotted in Fig. 8. A decline in loss does not appear over this epoch scale, and could be evidence of overfitting.

We can also analyze the evolution of the network over time. As the network trains, we might expect that error between predictions and labels to drop, and indeed, it does, as shown in Figures 8 and 9. These two plots of MSE loss versus epoch analyze two rounds of the network detailed in this work training on synthetic ionospheric data. That is, we trained the network for 30 epochs, noted that the predictions could have been better, and ran that same training for another 30 epochs, resulting in the predictions plotted in Fig. 6.

Figure 8 clearly shows quick improvement in the network's understanding of the model. This is as we might hope, as epoch 0 here begins from a freshly-created network, with no training done. Toward epoch 30, we see relative spikes in the validation loss. This appears to continue when the network is put through another 30 epochs of training, shown in Fig. 9. Validation loss spikes relatively high several times, and potentially sees an uptick toward epoch 30 of the second round. This may be evidence of *overfitting*, which is indicative of our network becoming quite good at making predictions on the model data. Future training runs would benefit from a larger training set, or some tweaks to the theoretical model which introduce more realistic complexities. We can iteratively train the network in this way, noting where overfitting and underfitting occur. Underfitting occurs when the model cannot seem to pick up the patterns prsent in the data, and would benefit from other

parameter adjustments, such as a change in batch size or different tweaks to the model.

IV. MODEL IMPROVEMENT

The caption of Fig. 5 makes note of some issues with the current synthetic data model. For each synthetic trace shown, the total current is plotted in red, and the constituent electron and ion currents are plotted as cyan and magenta lines, respectively. The floating potential is plotted at $(V_f, I_i s)$ in each synthetic plot. Based on our understanding of ionospheric plasma[1,2,3,5], and the definition of the floating potential, we should always expect the red dot to land on the line for the ion current. This is the case only for the top-left plot, because it was this plot that would used to construct and verify the `SpaceBox` class, based on the real trace in the bottom left. It is clear even from this one case that improvements can be made. Further, the top-middle plot looks like nonsense. We believe that, qualitatively, the situation here is that the I-V trace has been inadvertently shifted several volts too high, as the electron current is locked at the cutoff value of $2.420\mu A$. The top-right trace is an example of a randomly generated trace which has some use, but it again theoretically imperfect. The bottom three traces are the constituent and total I-V traces from real STF-1 data. We can note immediately that one improvement to make is to ensure that the electron current remains at zero prior to the floating potential, where the transition region begins.

V. FURTHER PLANS

The first and foremost further plans will be the model improvements mentioned previously. Our network seems capable of making predictions on at least our theoretical model with upwards of 90% accuracy (for T_e, or 80% for I_{is}), and this knowledge can even be applied directly to real data, likely because of a lack of significant noise found in the real data. We therefore hypothesize that improvement of the model will be a top priority, in an effort to achieve prediction results similar to those on the NSTX data of 94-99% accuracy. We might additionally expect better performance on real data as a result of model improvement, as the model currently makes somse good predictions on real data without the second training stage.

Another clear step to take is to fully implement the second and third program stages as depicted in Fig. 4. While useful for assessing the model usefulness, the first training stage is meant to be just that: a first step. We have shown in this and another[4] work that LSTM networks are capable of learning to diagnose Langmuir time series data, such that it would be feasible to build an open-source, modular, and completely automated program for diagnosing Langmuir probe data across the a range of plasma settings.

As a final note on future plans, it is the author's intention to utilize the results shown here in collaboration with NSTX-U scientists at the Princeton Plasma Physics Laboratory to continue to develop this program, potentially releasing it as a tool for the wider plasma physics community.

VI. CONCLUSION

Plasmas ranging from the coldest, most diffuse to the hottest, most dense are regularly diagnosed with Langmuir probes, a relatively simple device which produces significant volumes of data in an attempt to paint a picture of the time evolution of a plasma. Advancements in computational methods have enabled the use of deep learning to attempt to remove the need for manual human analysis, saving much time and enabling new technologies. These may include the use of neural networks to provide live updates on space weather and fusion reactor temperatures. We use Tensorflow 2.3.0 and plasma theory provided by Thompson[5]

to construct synthetic ionospheric plasma time series data and train the neural network, showing that, after less than an hour of training, the network shows promise. Compared with a similar experiment done on NSTX data, the author will pursue this implementation further to complete the program and hopefully contribute a meaningful tool to the plasma community.

VII. REFERENCES

[1]F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*, 3rd ed. (Springer International Publishing, 2016).
[2]Y. Kamide and A. Chian, *Handbook of the Solar-Terrestrial Environment* (Springer Science+Business Media, 2007) p. 257.
[3]I. H. Hutchinson, *Principles of Plasma Diagnostics* (Cambridge University Press, 2002).
[4]M. J. Lazo, "Langmuir probe data interpretation with a neural network," Tech. Rep. (Princeton Plasma Physics Laboratory, 2020) dept. of Energy internal report.
[5]D. S. Thompson, "Three-dimensional multispecies distribution functions in a plasma boundary with an oblique magnetic field," dissertation (2018).
[6]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation **9**, 1735–1780 (1997), english translation.
[7]D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," (2015), iCLR 2015 Conference Paper.
[8]https://en.wikipedia.org/wiki/Coefficient_of_determination, "Coefficient of determination," Wikipedia article (2020).
[9]https://www.tensorflow.org/, website (2020).