Development of PPTNet a Neural Network for the Rapid Prototyping of Pulsed Plasma Thrusters

IEPC-2019-169

Presented at the 36th International Electric Propulsion Conference
University of Vienna, Austria
September 15-20, 2019

Vinay Williams*, Vasileios Argyriou†
and Peter Shaw‡
Kingston University, London, SW15 3DW, United Kingdom

Christop Montag§ and Georg Herdrich¶
University of Stuttgart, Keplerstraße 7, 70174 Stuttgart, Germany

Aaron Knoll‖
Imperial College London, London, SW7 2AZ, United Kingdom

and

Maximilian Mörtl**
Technical University of Munich, Garching b. München, 85748, Germany

Abstract: Numerical computations of pulsed plasma thruster performance and behaviour is time and computationally expensive, leaving many thrusters with low efficiencies and high costs for development. This presented work aims to reduce the required resources whilst increasing the performance of PPTs through the use of Deep Learning. Two deep artificial neural network architectures have been proposed for this task, where they were trained to predict the shape of current and voltage discharge profiles of a parallel plate thruster given the electrode separations and the capacitor charge voltage. Both models learned to predict the trend of either current and voltage discharge profiles, however, performed less accurately for seasonality within the data. The more accurate of the two has been put forward for further inspection and has been named PPTNet. Experimentation and testing need to take place using a larger data set with more initial conditions and thruster configurations going forward, however, a preliminary neural network to predict the performance of a pulsed plasma thruster is presented in this study.

*MEng Student, Department of Aerospace and Aircraft Engineering, vinaywilliams00@gmail.com.
†Senior Lecturer, Department of Computer Science, vasileios.argyriou@kingston.ac.uk.
‡Senior Lecturer, Department of Aerospace and Aircraft Engineering, p.shaw@kingston.ac.uk.
§Senior Lecturer, Institute of Space Systems, montag@irs.uni-stuttgart.de.
¶Senior Lecturer, Institute of Space Systems, herdrich@irs.uni-stuttgart.de.
‖Senior Lecturer, Department of Aeronautics, a.knoll@imperial.ac.uk.
**MSc Student, Department of Mechanical Engineering, maximilian.moertl@icloud.com.

Nomenclature

σ	= sigmoid
ANN	= artificial neural network
b	= bias
CNN	= convoluted neural network
DL	= deep learning
e	= Euler's number
EP	= electric propulsion
ES	= electrode separation
f	= activation function
GAN	= generative adversarial network
GRU	= gated recurrent unit
GM	= generative model
$LSTM$	= long short-term memory
mae	= mean absolute error
ML	= machine learning
mse	= mean squared error
$msle$	= mean squared logarithmic error
PIC	= particle in cell
PPT	= pulsed plasma thruster
$ReLu$	= rectified linear unit
RL	= recurrent layers
RNN	= recurrent neural network
sgd	= stochastic gradient decent
$tanh$	= hyperbolic tangent function
V	= voltage
VAE	= variational autoencoder
w	= weight
x	= input
z	= output of summation function

I. Introduction

PULSED plasma thrusters are a subset of EP thrusters which due to their mechanical simplicity and their relatively low material cost have been a constant staple of university researchers over the past half-century. This simplicity hides the true complex nature of the plasma physics involved in these thrusters, as born out from a lack of improvement in the thruster's efficiency over the past decades which has not significantly improved since early experiments. Alternative theories on the underlying physics involved in a PPT discharge have been previously proposed, which suggest that the erosion of the PPT electrodes via a process of Cathode spot erosion, have a more significant role in the discharge process than traditionally thought.[1,2] An attempt was made to model this complex process but it had its limitations, due to available computing resources and that it was not based on a full three dimensional PIC code.[3]

Despite low thruster efficiencies, PPTs have had a recent resurgence due to their scalability and low power requirements which have led to an increased usage of the thruster technology on CubeSats.[4,5] If a breakthrough in thruster efficiency is to be made, providing a better offering compared to other propulsion devices on the market, we need to change our design approach. Currently, because the physics is complex and not well understood, the traditional method of PPT development has started with a prototype that has been altered and then evaluated, in an iterative process. On occasion, three dimensional plasma simulation modelling has complimented the design process but with mixed results, because the governing equations that model the discharge process (including but not limited to the erosion and ablations of propellants or surfaces, late time ablation, high frequency variations in electrode resistance etc.) has been incomplete. Deep learning offers a potential solution that removes the requirement to state what the governing equations of the discharge process actually are.

The field of Deep Learning is part of the broader family of Machine Learning techniques, seeing a significant uprise in the past ten years although having been around since the 1950s.[6] Its use of deep Artificial Neural Networks distinguishes it from the other ML techniques and benefits from ANNs' ability to learn patterns in high dimensionality data. Classification, forecasting, generation, and autonomy are well-known applications, for which models have shown real world success. Classification involves labelling input data based on features identified within, generation is the inverse, models are provided with some labels and it produces its concept of the object described. Forecasting involves the prediction of steps in a sequence given its historical data and finally, autonomy is the autonomous control of a task which previously required human input. Successes include Google's AlphaGo Master model beating the number one ranked player in the game Go, in 2017, Nvidia's CELEB-HQ in 2018 generating images of people with celebrity like features and Telsa's models providing owners of the company's vehicles with self-driving features.[7,8]

In the scope of electric propulsion testing, Deep Learning is a way of training a computer algorithm with large datasets of experimentally gathered data, with varying input conditions, to learn patterns and features within that data and to make, within seconds, intuitive predictions. This ability to circumnavigate the need to define the governing equations of the plasma (and other) processes involved, at compiling speeds unheard of in plasma modelling simulation work, could lead to rapid prototyping of an optimised PPT design. If the process is proven it is hoped to be developed for other EP devices.

II. Background Information

A. Pulsed Plasma Thrusters

The PPT basic components and method of operation has been previously described.[1] In short, a high voltage and high capacitance capacitor is charged, which stores a significant amount of energy. The capacitor is connected to two electrodes located in the PPT discharge chamber by a low inductance and resistance connection. A separate igniter introduces electrons into the discharge gap between the electrodes and subsequently, an electrical connection occurs that closes the circuit and allows the capacitor to dump its energy into the resulting plasma. The material source of the electrons and ions that form the resulting plasma have traditionally been theorised to come from the propellant that is fed into the PPT between the electrodes as either a Teflon propellant bar, or some gas or liquid. It has been shown that this is not always the case and the PPT can also operate by just the erosion of the electrode material, similar to the Vacuum Arc Thruster.[1] Once the plasma is created it is accelerated out of the discharge chamber by interactions with strong magnetic fields created during the discharge process and gas dynamic forces. Although small, the forces produced can be used for satellite propulsion.

During discharge, the flow of current and voltage across the capacitor varies with time and can be used to generate datasets otherwise known as discharge profiles. These profiles are not accurately determined through numerical simulations as many processes occur within the discharge chamber that are still not fully understood. The discharge profiles can be used to make approximations of the thruster performance.[3] This need leads to the interest of the authors in Deep Learning. If an artificial neural network can be trained to 'learn' the relationship between initial values and the discharge profiles, it can then be used to predict the performance and in turn used to optimise efficiency whilst also reducing rapid prototyping times.

In previous research we aimed to inspect the use of PPTs for CubeSats, having run experiments using a parallel plate PPT.[1] Electrode separation and capacitor charging voltage were varied whilst no Teflon propellant was present between the electrodes. The discharge profiles collected in these experiments was used to train a neural network, named PPTNet. The development of PPTNet is described in this work.

B. Deep Learning

The field takes inspiration from nature and biology as within brains lay a complex set of interconnected neurons which process information in the same way, ANNs are comprised of nodes that are interconnected and process data.[9,10] The neurons are usually arranged into three sets, input, hidden and output layers as seen in Fig. 1. Input and output layers allow for the pass in and then out of data respectively, whilst hidden layers are considered to be the 'weightlifters' being responsible for much of the learning which takes places within a neural network. Models with more than three layers, those with two or more hidden layers, are considered deep ANNs.

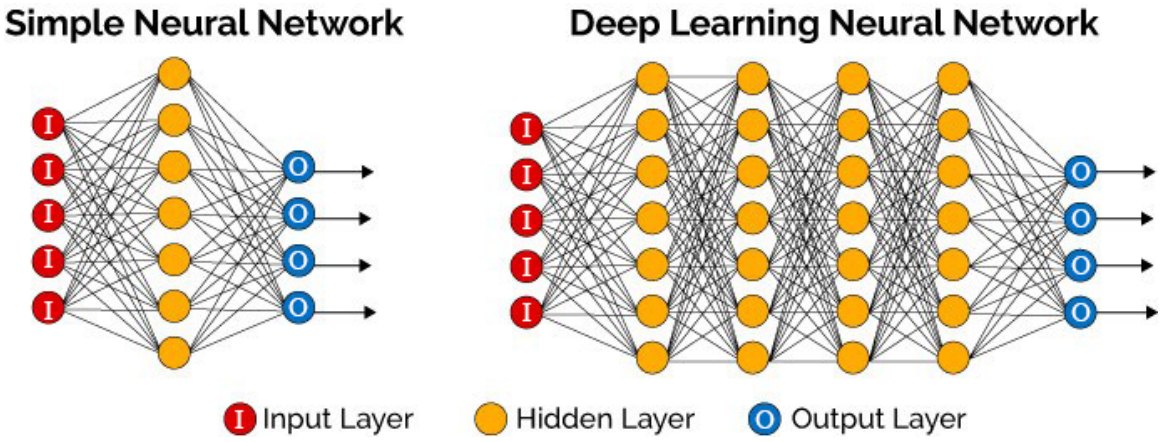


Figure 1: Shows a comparison of a simple and deep neural network, highlighting the different number of hidden layers present in each.[20]

$$z = b + \sum(x_1 * w_1 + x_2 * w_2 ... x_i * w_i) \tag{1}$$

Layers are built of nodes where each takes some data and outputs a transformed version of it. This transformation starts by multiplying the input(s) by weight(s) associated with the neuron and then finding the sum of these multiplications with a bias as seen in Eq. (1). The summation is then inputted into an activation function which performs a constant operation where Fig. 2 allows for a visualisation of the entire operation that occurs at one node. In many cases, the activation function is replaced by another function which is where layer types arise from. Sigmoid, tanh and ReLu activation functions have seen widespread use in Deep Learning and have been utilized heavily in this study and are shown in Eqs. (2-4).[11,12]

$$\sigma(z) = \frac{1}{1 + e^{-}z} \tag{2}$$

$$ReLu(z) = max(0, z) \tag{3}$$

$$tanh(z) = \frac{2}{1 + e^{-2z}} - 1 \tag{4}$$

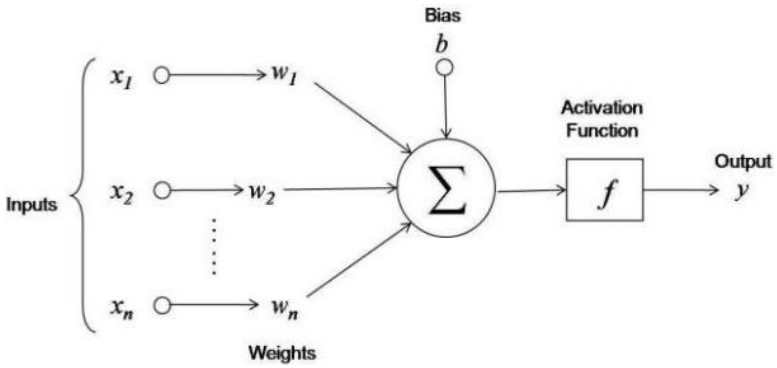


Figure 2: A visualisation of the operations carried out at each node.[25]

Models are given generalised names such as fully connected, recurrent or convolutional based on the layer types which are predominantly used. Dense or fully connected layers are used a lot, showing up in models consisting of only them or in other models using a combination of layer types. Their importance is derived from all the outputs being linked to every input of the following layer as seen in Fig. 2. This high number of connections is where its name arises from. In situations, where the entire input is relevant this layer shows its strength due to all inputs being present at each node in the following layer. The following neurons are then able to 'learn' different features in relation to the entire input which is useful in tasks where the entire input is related to specific features. This layer has possible uses to PPTNet as voltage and current discharge profiles of the PPT are both related and depend on each other, therefore both profiles should be processed at every node. Furthermore, the profiles' shapes are related to each other and every point created is dependant upon the previous points.

Recurrent models make use of recurrent layers where their distinguishing factor is the presence of an internal state in nodes as seen in Fig. 3. These layers perform the same task for every element of a sequence where the feedback received influences the output. Recurrent connections have branched into gated and non-gated connections also seen in Fig. 3. Gated connections are those which there is/are some condition(s) that need(s) to be met for a return state to be processed. These connections have seen more use than their counterparts where the two most popular types are GRU and LSTM.[13] Recurrent layers are important as they allow for 'memory' within ANNs allowing them to reference concepts learnt during the training process. Recurrent layers have possible uses in PPTNet as referencing the previous inputs (i.e. previous processed discharge profiles during the training process) assist in making predictions. The recurrent layer will learn features from previous discharge profiles at different initial conditions and store them in its layers, acting like memory, which can assist in the processing and prediction of future data.

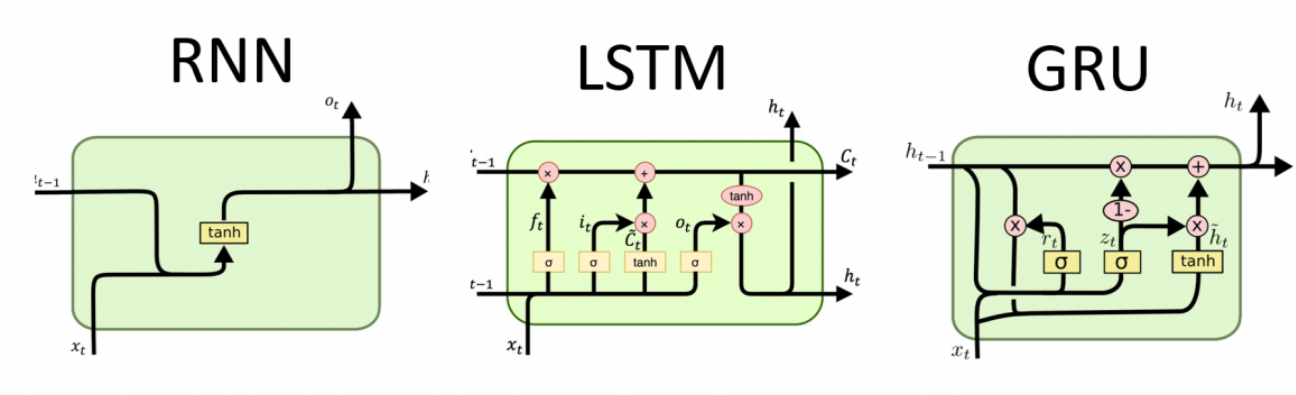


Figure 3: Shows recurrent nodes, both gated and non-gated. RNN is an example of a non-gated recurrent node while LSTM and GRU are examples of gated recurrent nodes.[24]

Convolutional neural networks make use of the convolutional layer type where each node performs the convolutional operation. These layers have seen widespread use due to their ability to 'learn' patterns. Filters are applied to the data in the form of windows. For every application of a filter, the dot product between itself and the corresponding window of data is found and then placed into a new matrix. This is repeated at an adjustable interval named a 'slide', till the entire input is processed with all the filters. The new matrix is traditionally smaller than the inputted one due to the operation, however, padding can take place as seen in Fig. 4 to preserve the input size. The dimensionality of the output data is given by the number of filters for that layer, where filter size is constant for all filters in a layer. CNN models usually follow a structure which start with a convolution layer that has a large filter followed by subsequent convolution layers with ever decreasing filter sizes. In each convolution layer as the filter size decreases usually the number of filters within that layer increases. So the CNN model as a whole will have only a relatively small amount of filters that learn large features or patterns, whilst it will have numerous smaller filters that learn small features or patterns. Because the smaller filters are more numerous only the patterns that are most numerous will be learnt. The classification of images containing dogs and cats can be used to demonstrate this concept. Large features such as number of limbs, the presence of a tail and shape of the head will first be identified, however, as the convolutions reduce in size, smaller features such as textures can be learnt. This layer type has possible uses in PPTNet due to its pattern recognition skills. During development, the number of filters within the individual layers can be modified according to need in such a way that when the model is applied, a combination of the various filters established could allow for accurate generations when combined.

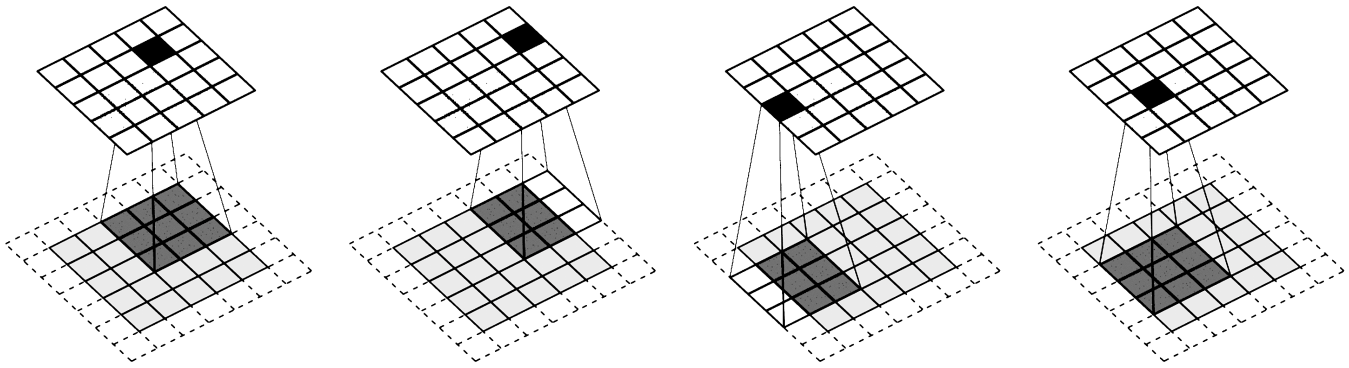


Figure 4: Shows the convolutional operation with padding.[21]

Three commonly used layer types were inspected, however, there are a variety of other types with which the aforementioned ones need to be combined to be useful. In PPTNet, Repeat Vector, Input, Normalization and Dropout layers were either used or referenced and so will be elaborated on. Neural networks operate best when the data they are working on is normalised between the with data values of -1 and 1 therefore when operations such as convolutions are performed, values outside this range are generated, which then need to be reduced, this operation is performed by a normalization layer. Dropout layers are essential when dense layers are used. During training, some connections in fully connected networks may become stronger than others meaning there is a higher flow of data through them. This occurrence then biases a model inhibiting it from learning features within the data which the preferenced connections do not map and overall degrades the performance of the model. Dropout layers counteract this by randomly selecting connections between layers and discontinuing them, forcing other connections to become more weighted and the network to become more evenly trained.[14] Repeat Vector is used to add dimensionality to the data. It duplicates the existing data along a new axis and is useful when the input has a lower dimensionality than the output. Lastly, are input layers; these are nodes that take some data and passes it into the following layer for processing. They only pass along data.

After a model is defined it must be trained and tested before its application to real world problems. Training is important as it determines the final performance and is highly comparable to drawing a best fit line as can be visualized in Fig. 5. Under-fit models are considered those which are not making accurate predictions usually due to not grasping the problem, whilst over-fit models are those which are highly specific. The aim is to train the model to grasp the problem in order to make accurate predictions. Training consists of inputting data, allowing the network to make a prediction, finding the difference of the prediction from the true value and then adjusting the weights within the model for a more accurate prediction. This is

done repetitively until the error is reduced. Loss functions determine the difference between the true and predicted data, where the objective is to minimize this value. Optimization functions steer the direction of change for the weights, deciding whether to increase or decrease them. The process loosely described here is the feedforward and backpropagation method which was developed in the early 1970's.[15]

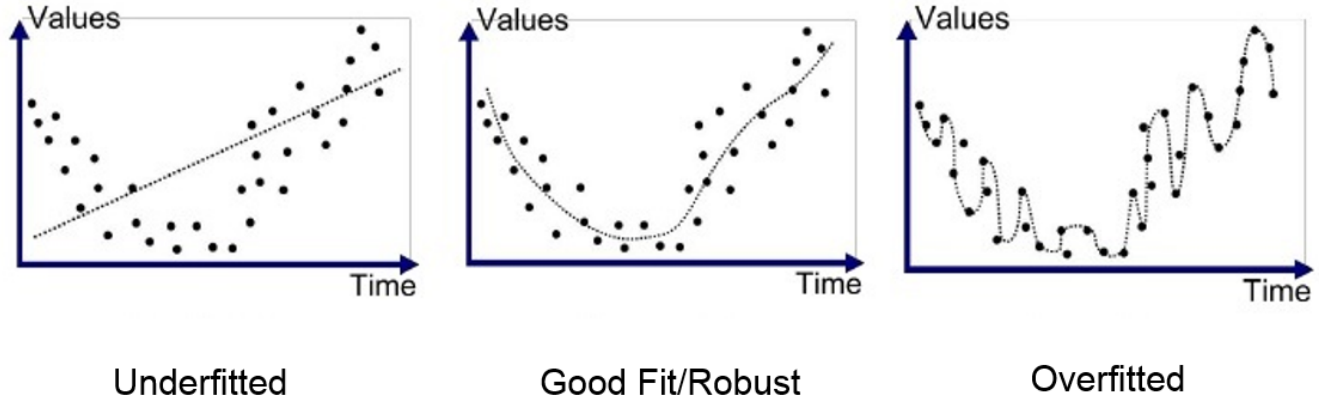


Figure 5: Shows a visualisation of fitness of an ANN through the use of drawing a best fit line.[22]

Neural networks are highly sensitive to the format and quantity of data which they are presented with. The aim during training is to generalize the model which requires large quantities of data. This data is usually ordered in some way, either having labels or the network being told what is the intended output for an input. This leads to two forms of training, supervised and unsupervised where the first was used PPTNet. After a model has been trained its performance must be tested, however, if testing is performed on training data inaccurate results will be seen due to a NN's abilities to memorize data. Therefore, data splitting is performed which involves breaking the data into three portions, training, validation and testing. This is generally done through a ratio of 70: 15: 15 where if 100 samples of data were presented together, 70 would go toward training, 15 toward validation and another 15 toward testing. Validation data is necessary because during training it is important to ensure the model is generalising, therefore it needs to be tested throughout the process. However, although weights are not updated after a prediction is made when testing with validation data, it still becomes incorporated into the model and could skew results. This is why training data is not used.

C. Implementation

Google Collaboratory was used to provide the computational resources to train PPTNet. Google Collaboratory is a free to use high performance cloud computing resource provided by Google. Users are given access to two Intel Xeons Processors, 12GB of RAM and an Nvidia K80 GPU in the GPU runtime which was used to train and process the results from PPTNet. TensorFlow and Keras were used in Python 3 for the implementation of models. Tensorflow is an open-source data Machine Learning tookit developed by Google while Keras is a high level neural network API which operates above TensorFlow.

III. PPTNet Development

Generative neural networks arose after the creation of classification models and have seen successes in a variety of areas.[7,12,16,17,18] Considering this, the application of GM to engineering, specifically EP became of interest to the authors of PPTNet. A generative model can be trained to predict the discharge profiles of a PPT given its specifications after which it can be used to predict the profiles of new PPT designs for rapid prototyping.

Classification is the task of labelling an object based on its features as mentioned previously. ANNs have been created to perform this task where they have shown high success rates. These models have seen high usage of convolutional layers which is arguably due to their pattern recognition skills. ANNs designed for

this reduce the size of the input, slowly compressing the varying features, after which the data is flattened where the final layer corresponds to the number of possible labels. The convolutional layers act as feature extractors learning varying features of the input. Convolution sizes start large and slowly reduce which allows for larger features to be extracted first and smaller ones latter on. The number of filters on the layers also vary where large convolutions require fewer filters versus smaller ones. Once a prediction has been made, the value from each output neuron is compared, where the highest value corresponds to the most probable label.

Conversely is generation, the model type which is of focus in PPTNet and involves the creation of the interpretation of an object, given its labels. Initial attempts to perform this task using ANNs involved reversing a trained classification model, however, it was unsuccessful due to values between the nodes going to the extremes. Therefore it was concluded a model would have to be trained solely for generation. Varying architectures were developed, ranging from fully connected, to LSTM, to convolutional models. Different architectures and layer types showed success for different data types as varying data required different abilities to be produced. Three prominent sub-classifications have risen to the forefront in the field which are GANs, VAEs and Autoregressive models.

GANs,[16] comprises of two neural networks, the generator and the discriminator, which work in tandem during training whilst only one is used for prediction. The generator as suggested by its name creates some data given an input whilst the discriminator distinguishes between true or fake data. The trained discriminator is used to direct the learning process of the generator during its training phase. After training, the generator is then 'extracted' and applied. Fig. 6 shows the structure of a GAN during training.

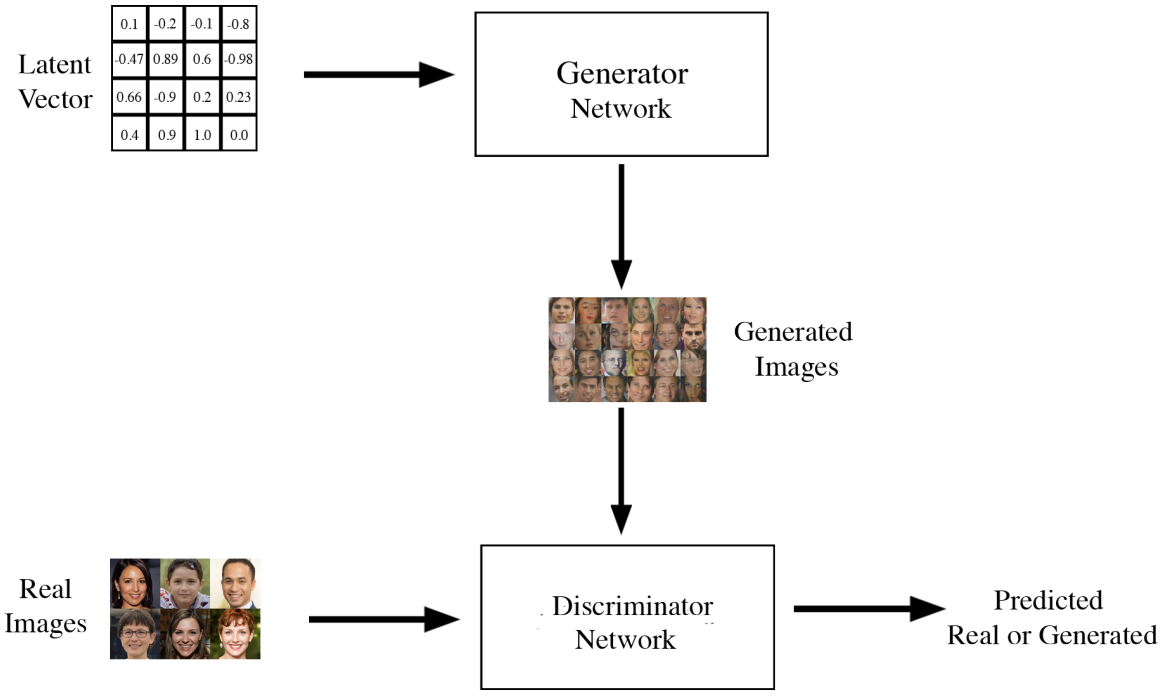


Figure 6: Shows the structure of Generative Adversarial Network during training.[26,27]

Variational autoencoders' importance is based on their shape which is seen in Fig. 7. Input data is compressed into code in the latent space after which the decoder attempts to rebuild the input from the latent space information. However, these models have a very similar shape as autoencoders, their distinguishing factor is how the latent space is handled. Instead of encoding one vector of size n, it encodes two vectors of size n, one comprising of mean values and the other of standard deviation values. The two vectors are then combined into a sample of size n which is decoded. VAEs have seen use to create data when there isn't a large data pool or there is a data pool which one would like to explore variations in.

Autoregressive generators are the simplest of the three generative classifications. These models follow more traditional architectures, making them easier and more stable to train. ARGs have also seen successful

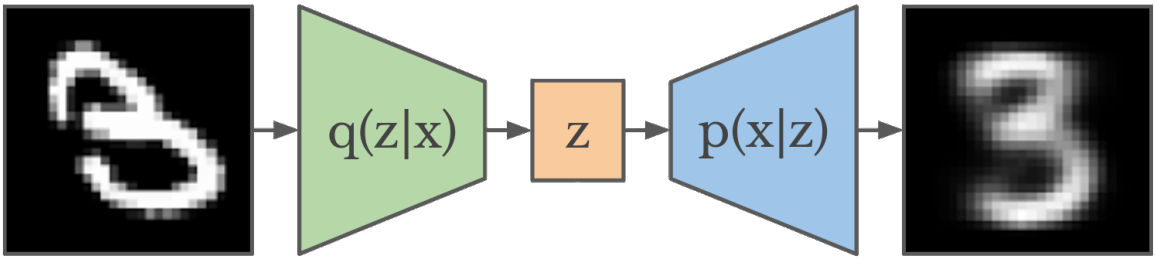


Figure 7: Shows the structure of Variational Autoencoder.[23]

use in audio and image generation such as with Googles PixelRNN, WaveNet. This model type is also able to learn multiple time scales,[18] which is useful to this project's aim as different features in a discharge profile appear on different time scales. These models also take into account the historical input when making predictions, allowing for higher accuracy in problems which need this feature.

A. Data

The experimental data collected in previous research was used to create datasets for the initial training of PPTNet.[1] The dataset accounted for electrode separation and the intended charge voltage of the capacitor at six different separations at voltages ranging from 900-3300V with multiples discharges at each pair. This resulted in 646 discharges each having electrode separation and charge voltage, therefore, resulting in an array of size (2,646). For every discharge 5 microseconds are recorded, where a data point is recorded 500 times per microsecond for voltage and current resulting in an array of size (2,2500,646), 2 for voltage and current, 2500 for all the points in 1 plot and 646 for the total number of discharges.

Initially, electrode separation and voltage were individually tried as splitting axis, however, after discussions, it was concluded if either was used the model would be cut off from portions of data in which the physics of a thruster varied, therefore, inhibiting it to make accurate predictions if such specifications were inputted. It was decided that validation and test data would have to be taken throughout the various ES and voltage combinations and so every fourth combination and all its discharges were placed into an array whilst the remaining data into another. The latter went on to be the training data whilst half of the first went to test data and the other to validation data. This method did not allow for a 70: 15: 15 data split as mentioned in background information, but for approximately 80: 10: 10. This ratio was opted for as it allowed for the largest quantity of training data whilst retaining enough validation and testing data.

After splitting, six arrays of three pairs resulted, one for testing, one for training and one for validation. All pairs had to be scaled using the same mechanism but different values, therefore, all members in the pair arrays were divided by the highest value in that pair as shown in Eqs. (5-7) for test data.

$$testscaler = max(inputtestdata, outputtestdata) \tag{5}$$

$$scaledinputtestdata = \frac{inputtestdata}{testscaler} \tag{6}$$

$$scaledoutputtestdata = \frac{outputtestdata}{testscaler} \tag{7}$$

B. Model Architecture

The autoregressive generative subclass was the architecture chosen to work with. This is due to the structure allowing for the generation of data on varying timescales which is noted to be important when viewing true profiles. Furthermore, VAEs would be unsuitable for this application as they are meant to replicate the inputted data which is not the objective of this problems whilst GANs were not used due to unavailable resources at the time of development.

The input given the test data was two values in an array of size (1,2), representing ES and voltage. This had to be augmented to an array of size (1,2,2500) for 2 discharge profiles, each consisting of 2500 points. Therefore, another dimension had to be added to the input and was accomplished using a repeat vector as seen in Fig. 8 and was done for all prototypes and final models.

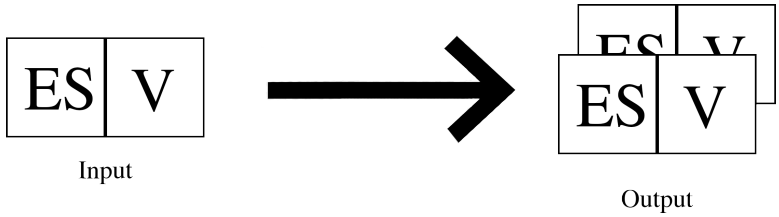
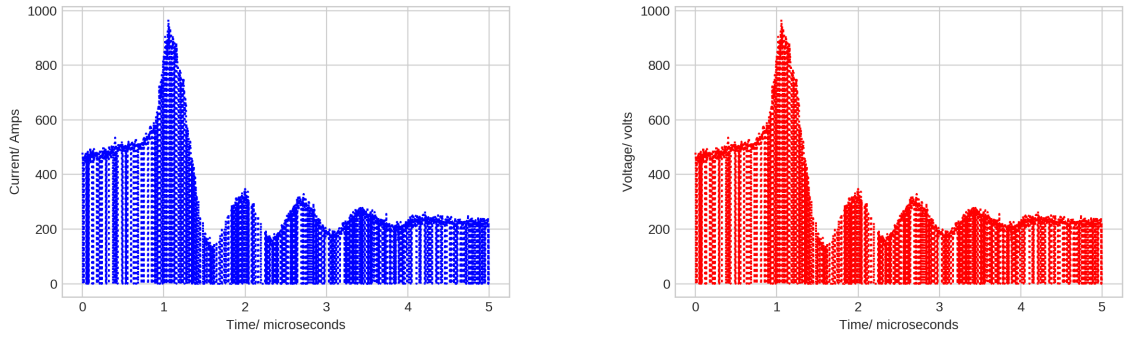


Figure 8: Shows the action of the repeat vector layer used in models tested and used

The first model was fully connected using an adam,[19] optimizer and mse loss function. It comprised of four layers, each of size 2500, all using relu activation functions. However, this resulted in a constant output for both voltage and current, appearing to be a merge between voltage and current profiles as seen in Fig. 10. Sigmoid and tanh activation functions were then tested on the output layer, where tanh saw an improvement in performance, however, a constant profile for both values was still outputted as seen in Fig. 11. Addition of dropout layers at different amounts was tested to ensure overfitting was not occurring due to strong node connections, however, this did not yield any improvements. The optimizer and the loss functions were also varied but there were no desirable results. It was theorized that the model was unable to distinguish between the two profiles which leads to next prototype.



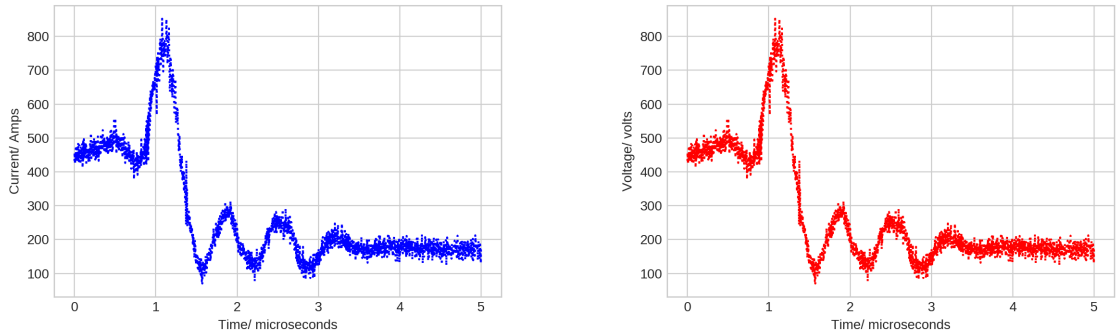
(a) Predicted Current

(b) Predicted Voltage

Figure 9: Shows the predicted profiles when relu activation was used on all layers in a fully connected model along with a mse loss function and an adam optimizer

Recurrent layers were looked at, creating a simple model consisting of one dense and one LSTM layer in the hidden portion of the network and a dense layer for output. Different graphs for the individual values were seen, therefore attempts to modify the architecture was tried. However, the benefits gained from the addition of more layers and variance of layer size did not have significant changes in the output, therefore, the initial model was kept and its structure is seen in Fig. 10a. From experience with the fully connected model, all hidden layers of the recurrent models used the relu activation functions, whilst the output layer used tanh. mse and mae loss functions were both tested where mse yielded better results. RMSprop and adam were tested for optimizers, but the differences were negligible.

Although good results were seen with the addition of recurrent layers, convolutional's ability to recognise patterns could not be ignored, therefore a CNN model was tried. The first attempt used one, a 1D convolutional layer which had a filter size of 20 and 128 filters. This was preceded and proceeded by two dense layers each of size 2500. The hidden layers all used relu activation and the output layer tanh. The result compared to that of the fully connected models initially attempted, where the output contained features of either profile and was constant for both. More convolutions were added but did not affect performance. As seen before, the addition of LSTM layers assisted, therefore, one was added to the model. Varying profiles for either value resulted but, lacked detail, missing smaller variations. Therefore a smaller convolution of size



(a) Predicted Current

(b) Predicted Voltage

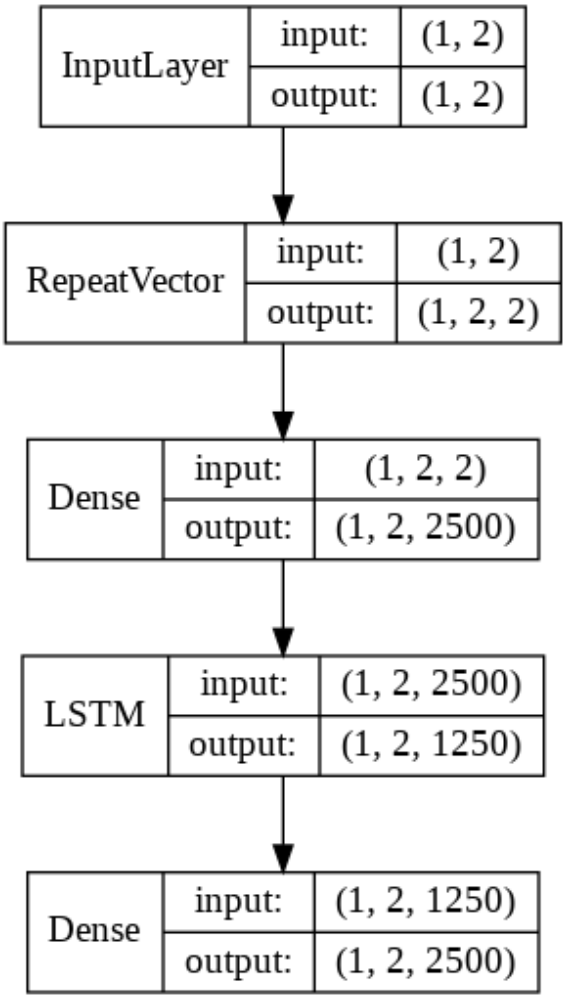
Figure 10: Shows the predicted profiles when tanh activation was used on the output layer and relu activation on the hidden layers in a fully connected model along with a mse loss function and an adam optimizer

5 was added, which allowed these missing features to be generated. Further experimentation with the depth and size of the layers were attempted allowing for improvements where the addition of batch normalization layers between the two main convolutions significantly affected performance. The final architecture is seen in Fig. 10b where layer choices and sizes resulted from inspecting true profiles to determine the size of features and through trial and error. mse and mae were tested for loss functions and RMSprop and adam for optimizers which found that the convolutional model operated better on a mae loss function whilst the same results were seen for the optimizer.

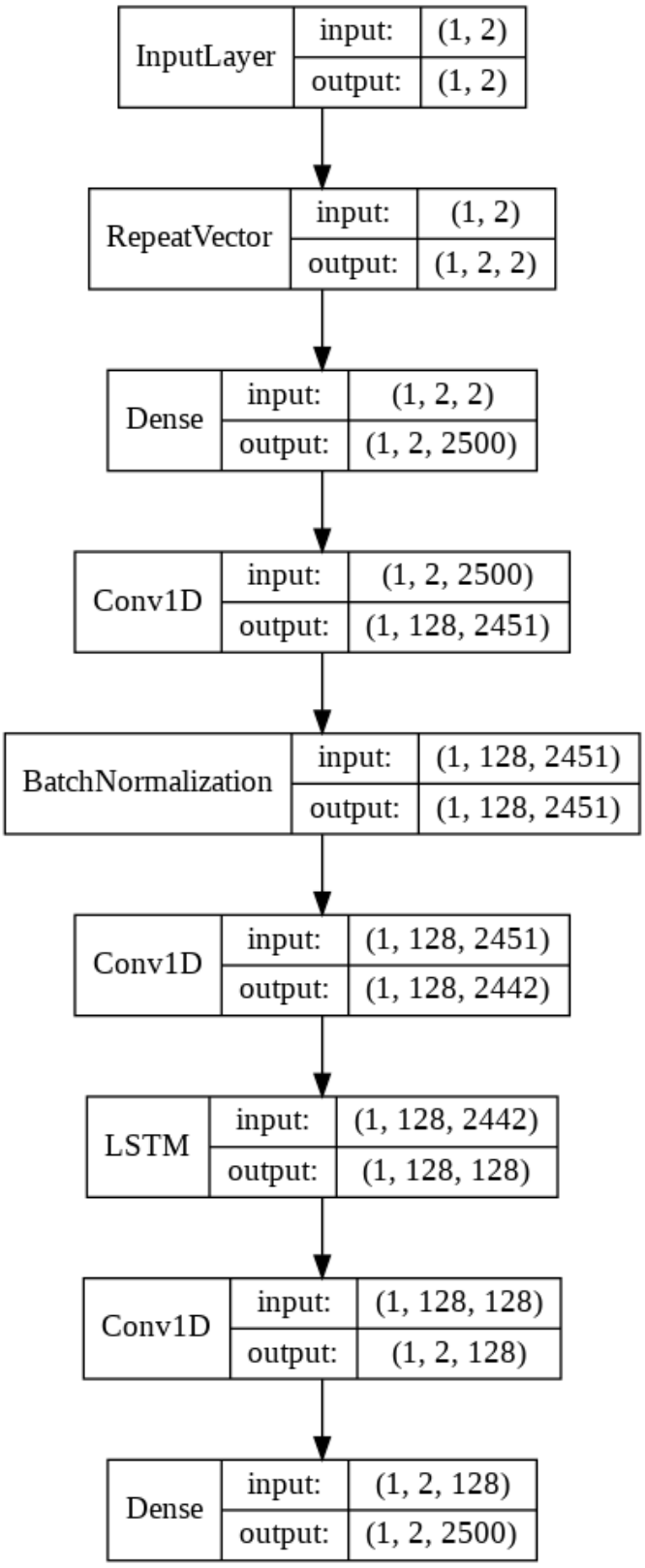
Additionally, it was found during model development that LSTM layers did not inhibit the production of constant graphs. These layers instead allowed for smaller layer sizes in the other layer types. This was seen with both models at different instances when the layer sizes were being refined. In the case of the LSTM model, when the size of the fully connected layers which preceded the LSTM layer was reduced from 2500 units to 500 units where graphs as seen in Fig. 10 were outputted. For the convolutional model, when there was the addition of an LSTM layer between the two main convolutions this occurred again, and when the number of units in the Dense layer which precedes the first convolution was reduced to less than the number of filters on that convolutional layer this again resulted. The exact reasoning behind this occurrence was not ascertained during these experiments and remains to be uncovered in future work.

C. Training

The PPTNet model was initially trained for 1 epoch at 500 steps per epoch and 100 validation steps to verify they operated as expected. Once this was done, the number of epochs to be trained for was set to 1000 whilst 'early stopping' and 'save best' was turned on. Save best allows for the best configuration of models to be saved during the training process which prevents them from over-fitting or under-fitting. Early stopping, inspects the performance of the model and stops training once improvements haven't occurred after a user specified period.



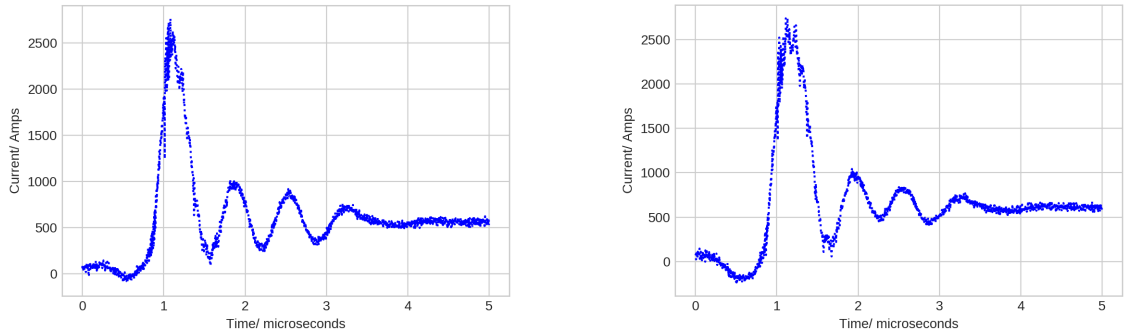
(a) LSTM PPTNet



(b) Convoluted PPTNet

Figure 11: Images showing the architectures of the two models which produced the best results

IV. Results and Discussion

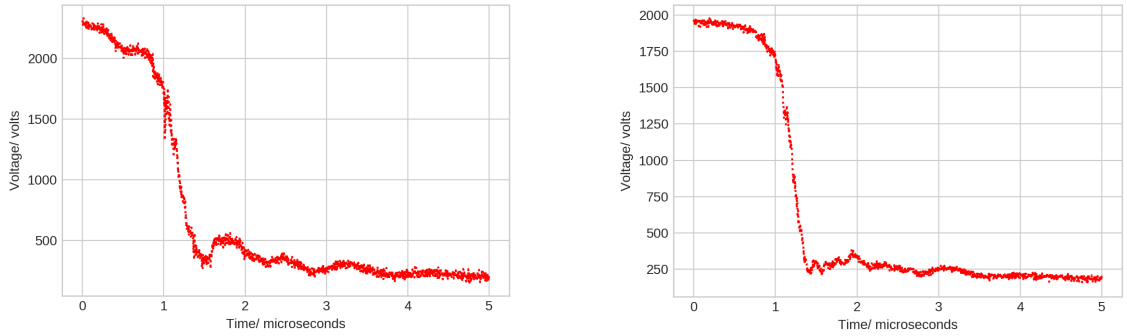


(a) Adam optimizer

(b) RMSprop optimizer

Figure 12: A comparison of adam and RMSprop optimizers

Although defining the architectures seen in Fig. 10, a choice of optimizer and loss function was needed before a model could be trained. From preliminary work with the fully connected models, it was determined that the adam optimizer showed good results and therefore was initially used for both models. However, when it came to refining the networks, other functions were tested where it was found that RMSprop produced almost indistinguishable results from that of adam. Both, optimization functions were then tested keeping the loss function constant to find the better of the two. Fig. 13 shows a comparison of current profiles generated for either with the same input conditions, both with the LSTM model.



(a) Mean Squared Error

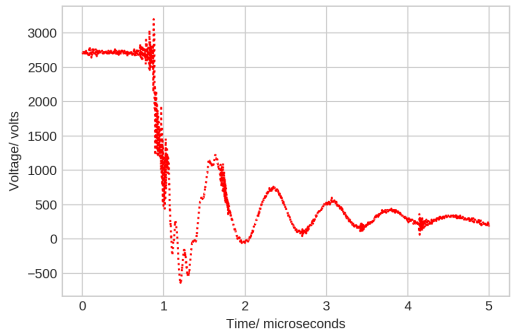
(b) Mean Absolute Error

Figure 13: A comparison of mae and mse loss functions

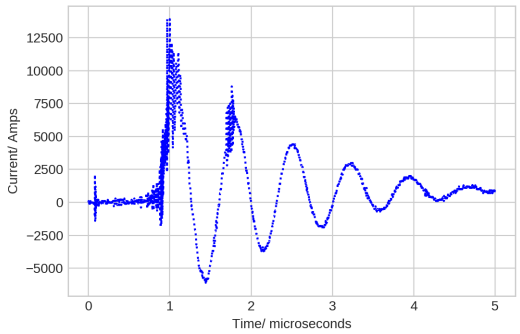
msle, sgd, mae, mse and binary cross-entropy were all tested for loss functions but mae and mse stood out. These two were tested keeping the optimizer constant and showed the results as seen in Fig. 12. mse produced graphs with smoother peaks than mae which is due to the square operation in mse. It allows for the development of oscillations due to not finding the exact difference which causes harsher alterations to the model. After testing mse on both models, it was found that it outperformed mae on the LSTM model but resulted in the same profile being generated with the convolutional model. To counteract this, mae was tested on the convolutional model which solved the issue. This is likely due to the filters in convoluted models needing to be accurate, not suffering from the easy going nature of mse, in order for accurate predictions to be made. Therefore, mse was used for LSTM PPTNet and mae for Conv PPTNet.

Inspecting Fig. 14 it can be seen that both models gained an understanding of discharge profile generation. Although the generation of trends is seen, either prediction suffers from varying absences in seasonality.

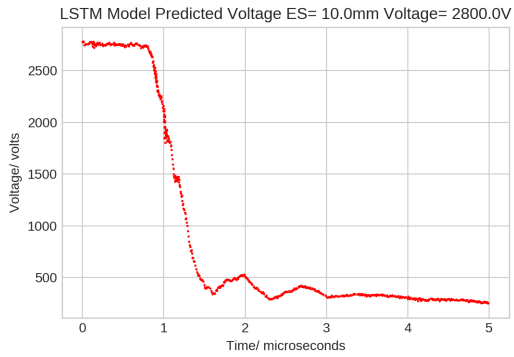
For the LSTM model, the occurrences in either profile is likely due to the model outputting an average shape of the training data. The training graphs all resemble each other in trend, however, data for larger electrode separation was also used. In these, there is a reduced presence of ringing in the profiles after discharge has taken place, which likely causes the model to output the shapes seen in Fig 14c and 14d.



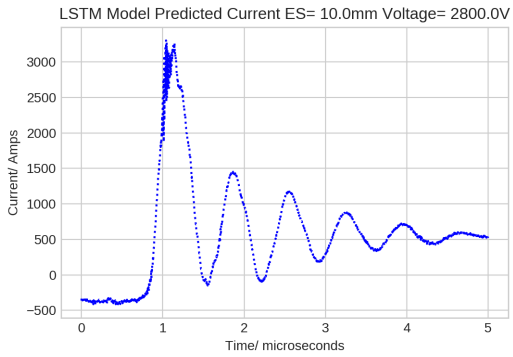
(a) True Voltage Profile



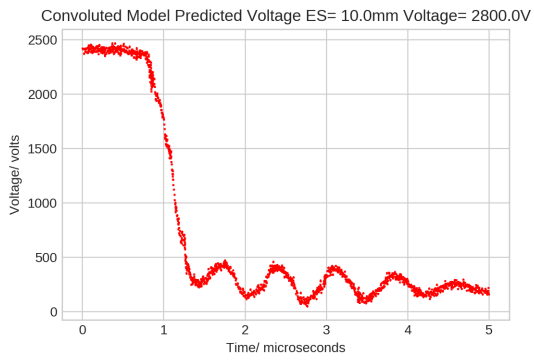
(b) True Current Profile



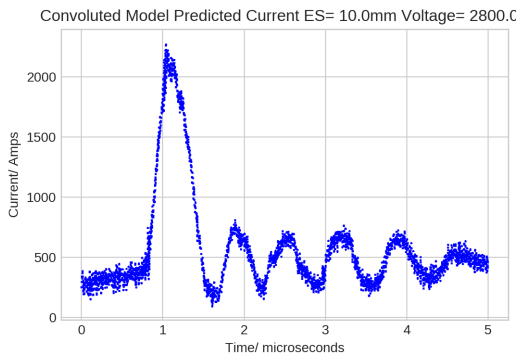
(c) LSTM Predicted Voltage Profile



(d) LSTM Predicted Current Profile



(e) Convoluted Predicted Voltage Profile



(f) Convoluted Predicted Current Profile

Figure 14: A comparison of the generated voltage and current profiles for the final PPTNet models at an electrode separation of 10mm and capacitor charge voltage of 2800V

This reduction in oscillations meant during training the network would be adjusted to output profiles which presented with fewer oscillations when larger separations were inputted. This combined with the ability of recurrent layers to remember training data, causes an average of the range of profiles to be outputted for smaller input separations.

Figure 14e and 14f show the convolutional model has begun to understand seasonality within the data. For voltage, the ringing area see a less developed oscillation initially which develops around the 2-3 microsecond region and some smoothing is observed after 4 microseconds. Although not as pronounced, this shows the model's grasp of ringing once a discharge has occurred. When comparing the CNN's profiles to LSTM's and the ground truths, there is a higher quantity of noise. This is likely attributed to under training and or the need for convolutions of different sizes. The smallest convolution was of size 10, however, if a larger size were used this can possibly be avoided as trends are more likely to be seen through a larger window. On the other

hand, if the convolution size is not the influencing factor, the amount of training is. In this case, filters for the convoluted layers are likely not refined enough, possessing too much noise which causes this occurrence.

Furthermore, the size of the data set used for training needs to be accounted for. Although 646 different discharges occurred in experimentation,[1] only 540 of them were used for training where there were 81 ES and V combinations used due to multiple discharges being present for one combination. Therefore, some of the observations can be attributed to overfitting because of a small training data set. This problem needs to be looked at in future work, where the creation of a larger data set is necessary.

Due to an inability for certainty, the statements made on the performance of the PPTNet model should be further tested for confirmation of influencing factors as well as refinement of the neural networks.

V. Future Work

From the results presented the authors think that there is a future for PPTNet. The approach in this paper is proof of concept and has only been demonstrated for a specific PPT configuration. To increase certainty and accuracy within PPTNet and make it applicable to more PPT designs (i.e. coaxial electrode design, differing capacitor types etc.) there is a need to increase the overall experimental dataset size which the model is trained upon. In future work, this will be done with two approaches. The first approach is to incorporate experimental data from PPTs developed by other institutes and organisations. Efforts have already started in incorporating the Stuttgart University SIMPLEX PPT data into the PPTNet dataset and we would be happy to collaborate with other institutes in incorporating there PPT firing data. The second approach is to create our own experimental PPT testbed which can be manipulated to change the configuration of the PPT easily and gather experimental data semi-autonomously. It would also be advantageous to add in additional data into the PPTNet dataset that includes an increased number of initial input conditions and more output data that could include impulsebit measurements, magnetic probe data and even high speed imagery of the discharge process.

It must be noted that the lessons learnt and models created for PPTNet to predict PPT performance could also be applied to other EP devices, given large enough datasets. The authors would be interested in speaking with other institutes or industry who may be interested in exploring this opportunity further.

Furthermore, PPTNet is only one application of Deep Learning which the authors have been looking into. The use of ANNs in replicating PIC plasma simulation code is another, where thus far a model, nicknamed PlasmaNet, has been created. It has shown preliminary success where short term predictions of electrons in a plasma field have been demonstrated. The model is currently trained using data extracted from UCLA's 3D Serial Electromagnetic Spectral code, however, access has been granted to UCLA's PIC OSIRIS code[28,29] and works for a model to replicate it is underway. Long term predictions currently suffer from a rolling error, but discussions have taken place and proposals to solve this problem have come about where one approach is the creation of a new layer type. Currently, this work is halted waiting on the setup of a new High Performance Cluster at Kingston University and is due to restart mid-September 2019.

VI. Conclusion

The preliminary work performed in this paper has shown a promising future for the marriage of Deep Learning and PPTs. Data from experiments carried out using a parallel plate PPT configuration was used to train two models so that if electrode separation and voltage are inputted a prediction for the voltage and current discharge profiles would be outputted. The models both showed the capability to produce profiles which followed trends within the data, however, they lacked the ability to produced the seasonality of the varying initial condition combinations. This is likely due to a small data set being used in the training process with improvements expected with a larger, more comprehensive one. Furthermore, the influence of layer types and sizes needs to be looked into as well as varying architectures such as a branched model with multiple outputs. The convolutional model shown, is put forward for further inspection and is the proposed predecessor of a future PPTNet version.

PPTNet is a preliminary model which needs large amounts of testing and training to be carried out before implementation can take place, however, the potential of this model is arguable and it is believed Deep Learning techniques such as those demonstrated in this study can be used to assist and or accomplish many tasks in the field of electric propulsion and more work should be carried out on these concepts.

Acknowledgements

The authors would like to acknowledge the OSIRIS Consortium, consisting of UCLA and IST (Lisbon, Portugal) for providing access to the OSIRIS 4.0 framework. Work supported by NSF ACI-1339893.

References

[1] P. V. Shaw, Pulsed plasma thrusters for small satellites. PhD thesis, University of Surrey (United Kingdom), 2011.

[2] P. V. Shaw, "Modeling of a pulsed plasma thruster; simple design, complex matter," Space Propulsion Conference, 2010.

[3] P. V. Shaw, "A high current cathode plasma jet flow model for the pulsed plasma thruster," 32nd International Electric Propulsion Conference, 2011.

[4] R. L. Burton, "Pulsed plasma thrusters," Encyclopedia of Aerospace Engineering, 2010.

[5] S. Kenyon, "Strand-1: Use of a $500 smartphone as the central avionics of a nanosatellite," 62nd International Astronautical Congress, Cape Town, South Africa, 2011.

[6] B. Farley and W. Clark, "Simulation of self-organizing systems by digital computer," Transactions of the IRE Professional Group on Information Theory, vol. 4, no. 4, pp. 76–84, 1954.

[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," arXiv preprint arXiv:1710.10196, 2017.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V.Pan-neershelvam, M. Lanctot, et al., "Mastering the game of go with deep neural networks and tree search," nature, vol. 529, no. 7587, p. 484, 2016.

[9] S. Haykin, Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.

[10] K. Gurney, An Introduction to Neural Networks. Taylor and Francais e-Library, 2005.

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814, 2010.

[12] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT press, 2016.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting ," The journal of machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.

[15] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al., "Learning representations by back-propagating errors," Cognitive modeling, vol. 5, no. 3, p. 1, 1988.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, pp. 2672–2680, 2014.

[17] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," arXiv preprint arXiv:1601.06759, 2016.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," arXiv preprint arXiv:1609.03499, 2016.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[20] F. Vázquez, "Deep Learning made easy with Deep Cognition," Medium, 04-Jan-2018. [Online]. Available: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351. [Accessed: 10Sep-2019].

[21] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way," Medium, 17-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutionalneural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 10-Sep-2019].

[22] W. Koehrsen, "Overfitting vs. Underfitting: A Conceptual Explanation," Medium, 31-Jan-2018. [Online]. Available: https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanationd94ee20ca7f9. [Accessed: 10-Sep-2019].

[23] Building Variational Auto-Encoders in TensorFlow. [Online]. Available: https://danijar.com/buildingvariational-auto-encoders-in-tensorflow/. [Accessed: 10-Sep-2019].

[24] A. dprogrammer, "RNN, LSTM & GRU, Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM) & Gated Recurrent Unit (GRU)" dProgrammer lopez, 06-Apr-2019. [Online]. Available: http://dprogrammer.org/rnn-lstm-gru. [Accessed: 10-Sep-2019].

[25] G. Ognjanovski, "Everything you need to know about Neural Networks and Backpropagation - Machine Learning Easy and Fun," Medium, 07-Feb-2019. [Online]. Available: https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a. [Accessed: 11-Sep-2019].

[26] "This Person Does Not Exist," This Person Does Not Exist. [Online]. Available: https://www.thispersondoesnotexist.co [Accessed: 12-Sep-2019].

[27] K. McDonald, "How to recognize fake AI-generated images," Medium, 14-Dec-2018. [Online]. Available: https://medium.com/@kcimc/how-to-recognize-fake-ai-generated-images-4d1f6f9a2842. [Accessed: 12-Sep-2019].

[28] R. A. Fonseca, L. O. Silva, F. S. Tsung, V. K. Decyk, W. Lu, C. Ren, W. B. Mori, S. Deng, S. Lee, T. Katsouleas, et al., "Osiris: A three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators," in International Conference on Computational Science, pp. 342–351, Springer, 2002.

[29] R. G. Hemker, "Particle-in-cell modeling of plasma-based accelerators in two and three dimensions," arXiv preprint arXiv:1503.00276, 2015.