Using a Neural Network to make predictions on data from ultra-relativistic particles

By George Korodimos, Hammersmith Academy

In affiliation with Imperial College London,

Dr Ravindra T. Desai (supervisor)

September 2020

Table of Contents

Abstract

The purpose of this research assignment is to examine the usage of a Neural Network in physics problems. Since I was limited to conducting my research over 4 weeks, I have looked into 2 such problems: predicting the motion of a particle within a single type of magnetic field, and predicting the distribution of low mass negative ion density in Titan's atmosphere. All the code written for this research is done in an iPython notebook in Google-hosted Jupyter notebook service - Google Colaboratory

Introduction

Reasons to use a Neural Network vs mathematical equations.

By using mathematical equations to calculate a result or to model reality we can be assured that the outcome is as accurate as our input data. Additionally, provided we have used our equations correctly, the outcome of our calculations will reflect our reality, and this can be used to progress or disprove a theory. However, the more complex the calculations are, the longer it takes a computer to process, leading to the requirement of increasingly faster computers or even the use of supercomputers.

By using a Neural Network, we can analyse patterns within the data, that may or may not be apparent to us. Based on those patterns, we can make predictions much quicker than using a complex algorithm that tries to understand these patterns using pre-programmed parameters. This results in increased processing speed at the expense of accuracy. Thankfully, this drawback can be reduced by using any number of techniques such as using a larger training data source and a validation dataset that has not been previously seen by the network during training. However, these options are not always available.

It may seem that this comparison, for use in physics, is quite one-sided against the use of a Neural Network however the decision to use a Neural Network or not is heavily dependent on the problem being solved. From conducting preliminary research, I have been able to determine two scenarios in physics where a researcher may choose to use a Neural Network: when they are only able to obtain a limited sample size and for quick and efficient classification of data.

> ***“ The tagger employs a deep neural network (DNN) using an architecture inspired by the CMS DeepJet algorithm.”– "A deep neural network to search for new long-lived ...." 27 Dec. 2019, https://arxiv.org/abs/1912.12238.***

Part I: Understanding Neural Networks

Since I did not know much about using or creating a Neural Network before embarking on this project, the first week and half of the second week of the project was spent researching and learning about how these networks function and how they are structured. Whilst following many tutorials and articles on creating neural networks, I created a Neural Network from scratch (without the use of any libraries that allow you to make one) as a learning exercise.

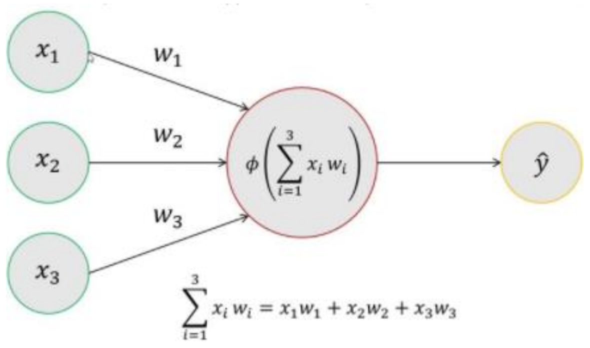

x_1
x_2
x_3
w_1
w_2
w_3
$\phi\left(\sum_{i=1}^{3} x_i w_i\right)$
$\hat{y}$
$\sum_{i=1}^{3} x_i w_i = x_1w_1 + x_2w_2 + x_3w_3$

Figure 1.1[1] – medium.com

Figure 1.1: This figure shows a simple Neural Network that has an input layer with 3 nodes, a hidden layer with 1 node, and an output layer with 1 node. When data travels into a new layer, excluding the input layer, it is typically passed into the formula seen above, x_n is the data from the previous layer (input or hidden), w_n is a weighted value given to each connection between 2 nodes, and Φ is an activation function.

The activation function in a node is used to define the output of the given node, for example: in a classification-type problem, you would want your output values to be between 1 and 0 allowing the network to select an output node that holds the value closest to 1 as its guess (e.g. classifying an image). This value can also tell you the network's confidence in the selected output. Since this calculation is made for each node in a network it is much quicker and resource-efficient to store each layer's data and weights in matrices. Matrices allow the user to pass into the network a batch of inputs rather than one input set at a time. The two most common activation functions used are the Sigmoid function (Figure 1.11) and the Rectified Linear Unit (ReLU) function (Figure 1.12):

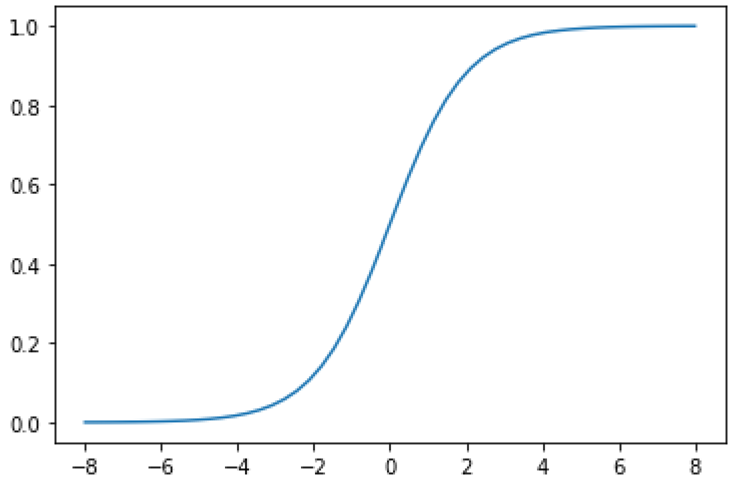


Figure 1.11

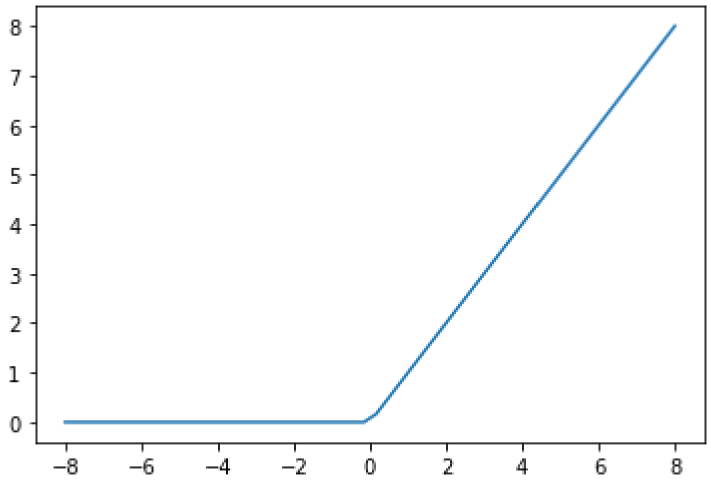


Figure 1.12

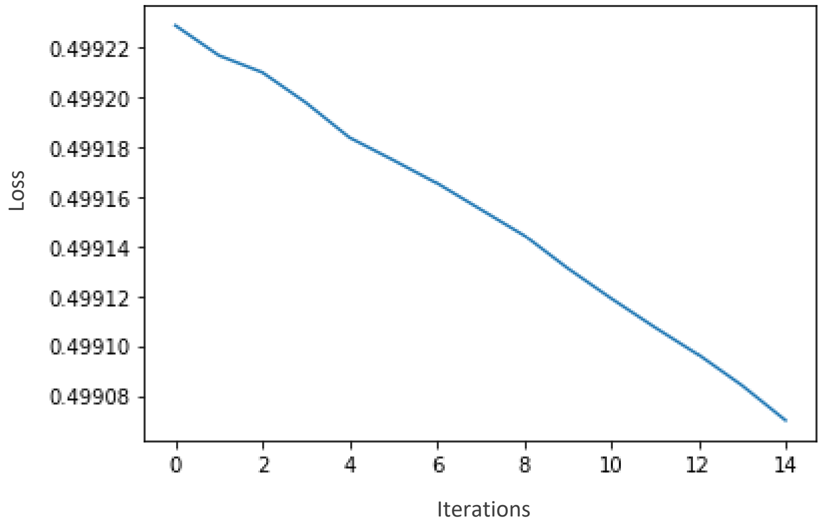


Figure 1.2

Figure 1.2 shows the loss of the network's output after each iteration (lower is better). The loss is calculated by comparing the predicted output with the expected output. Training a network on the patterns of a given data set is simply a process of minimizing this loss value through the use of an optimizer function. This optimizer is a function that calculates the loss of a network and applies a change to the network's weights that reflects how much a given weight should be altered to reach a lower loss (dome weights should be decreased and some increased). A typical loss function for a regression-type problem is Mean Squared Error (MSE):

$$J = Y^2 - \hat{y}^2$$

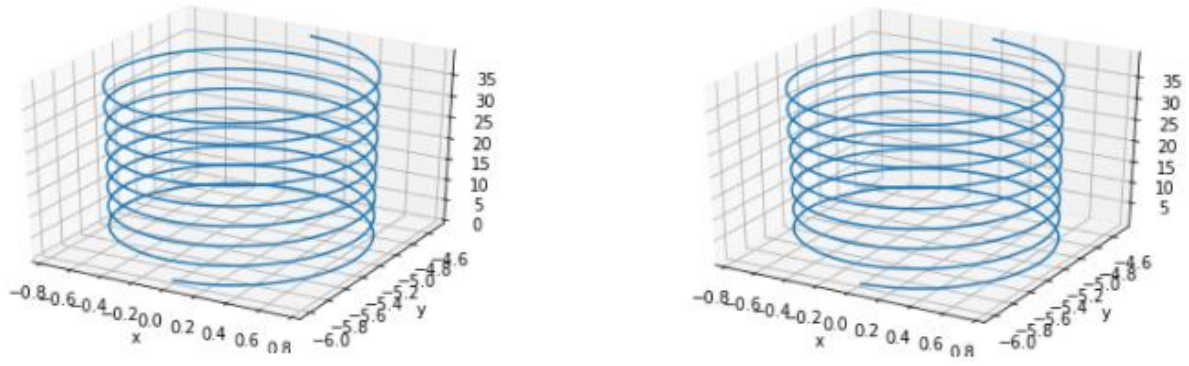
where J is the loss, Y is the expected output and ŷ is the predicted output. This is then used in the following equation to apply a small change to each weight towards a local or global minimum in the function of J. This equation is dependent on the network's optimizer and the following equation is used in an optimizer function called Stochastic Gradient Descent.

$$w_n = w_n + s\frac{dJ}{dw_n}$$

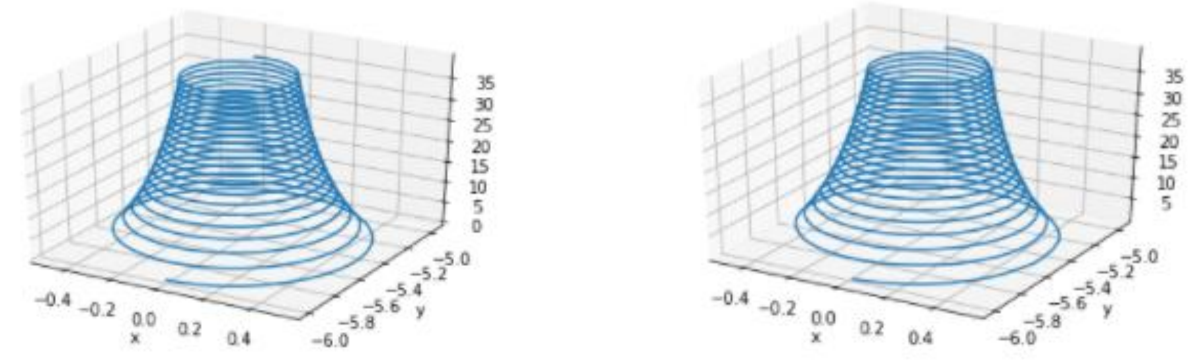
Where w_n is the matrix of weights of a given layer, s is a scalar value used to limit how "aggressive" the change made to the weights is and $\frac{dJ}{dw_n}$ is the change in loss with respect to the change in the layer's weights.

Part II: Modelling Particle Motion

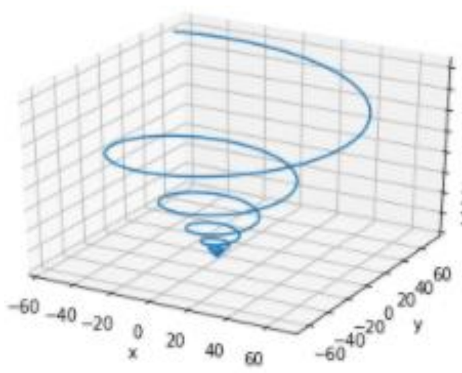
Having acquired a better understanding of how neural networks function, I begun to conduct my research on a neural network's performance in physics problems. For this section, I modified a python script by Vedant Varshney which simulates the motion of a particle within a magnetic field, given the particle's original motion vector. I conducted three "experiments", using a different type of magnetic field for each. First, I looked at predicting the motion of a particle in a constant field (figure 2.1), in a spatially varying field (figure 2.2), and in a time varying field (figure 2.3).

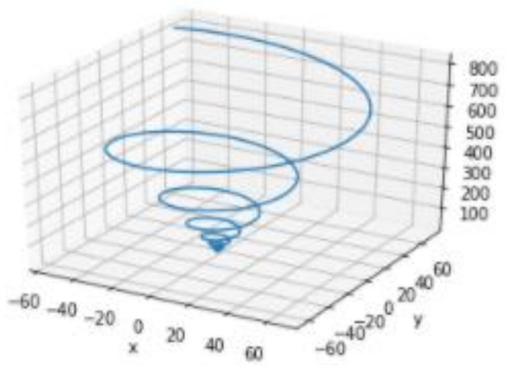


Original Figure 2.1 Predicted



Original Figure 2.2 Predicted





Original Figure 2.3 Predicted

The second half of week two and all of the third week was spent modifying the provided code, which simulated the motion of a particle around a planet's magnetosphere, to make it simulate my three simple magnetic fields.

From this point onwards I used libraries to offload the creation of the model and the creation of and implementation of optimizers and activation functions – specifically, I am using Keras from the TensorFlow library. Once I got my three graphs and datasets, I began to create a Neural Network that would be well suited to analyse the patterns in the particle's motion.

There is no correct way of selecting the network's size and shape, it is down to trial and error. My method for choosing a shape is simple: start with one hidden layer of 50 nodes, then keep adding layers of 50 nodes until the resulting graph of loss over iterations (for a small number of iterations e.g. 500) stops improving and begins to become worse. I then take note of the number of layers and nodes for the best graph produced and start over with a larger number of nodes (increments of 20), if the best shape of the network for the next increment of nodes performs worse than the previous shape, I then focus at node numbers between the two "limits" (similar to how binary search works). I eventually settled on using a network with 3 layers of 110 nodes each for the first graph and slight variations for the others.

In order to achieve the same shape as the original data, the network required between one thousand and ten thousand data points for each field-type and also required training for 1000 iterations. The highest accuracy I could get the network to was on average to 2 decimal places. Since each graph shows a particle's trajectory in different magnetic fields, the network must be retrained when a new field-type is introduced.

Additionally, due to the high level of inaccuracy, attempting to predict the particle's motion from a single position vector (x1, y1, z1) results in a grossly inaccurate predicted trajectory due to the accumulating error. Therefore, the (prediction) graphs presented are a collection of outputs from the network using the actual position of the particle rather than the previous prediction as input for the network's prediction.

Part III: Cassini

For week 4 my supervisor gave me data gathered by the Cassini probe over its many flybys through Titan's atmosphere. This data contains the probe's position (altitude, latitude, and local time) relative to Titan when measurements were taken, as well as the measurements themselves: density of low mass negative ions in the atmosphere, electron temperature, and electron density. However, the last two were not used for training due to the limited sample size showing a weak to no correlation with the rest of the data.

This data has been previously analysed in "[2008.07591] Spatial variations of low mass negative ions ...." 17 Aug. 2020, https://arxiv.org/abs/2008.07591. The paper began to explore the patterns found in the data, suggesting that low mass negative ions exist in higher densities at lower altitudes, around 0°N latitude, and around noon Titan Local Time (the centre of the plot, see figure 3.1).

My task for week 4 was to train a neural network on this data and then use it to predict the distribution of density for the entire graph (see figure 3.2). However, before I could do that, I had to ensure that the network was effectively trained. To accomplish this after each training cycle, I used the network to make predictions on the density for the same positions we had data points recorded by Cassini until the difference between the densities for the two sets of graphs were as low as possible whilst also not overfitting the data (see figure 3.1 for the comparison).

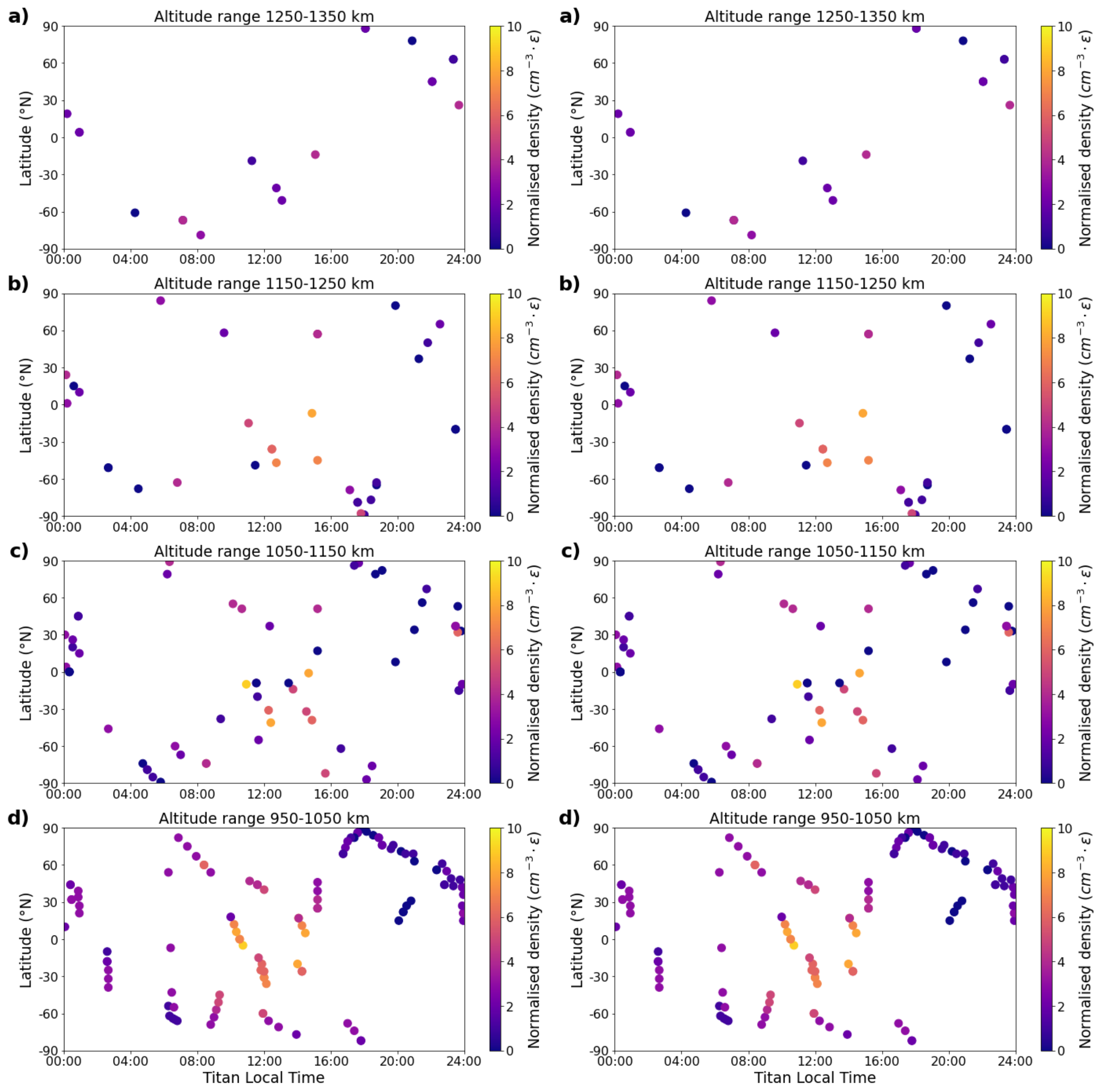


Figure 3.1: the left shows the data gathered by Cassini, and the right shows for the same measurement locations but having the density (colour) of each predicted by the neural network. These results were obtained after training a network made of 4 layers (90, 120, 120, 90 node configuration using the SeLU activation function) for 6400 iterations.

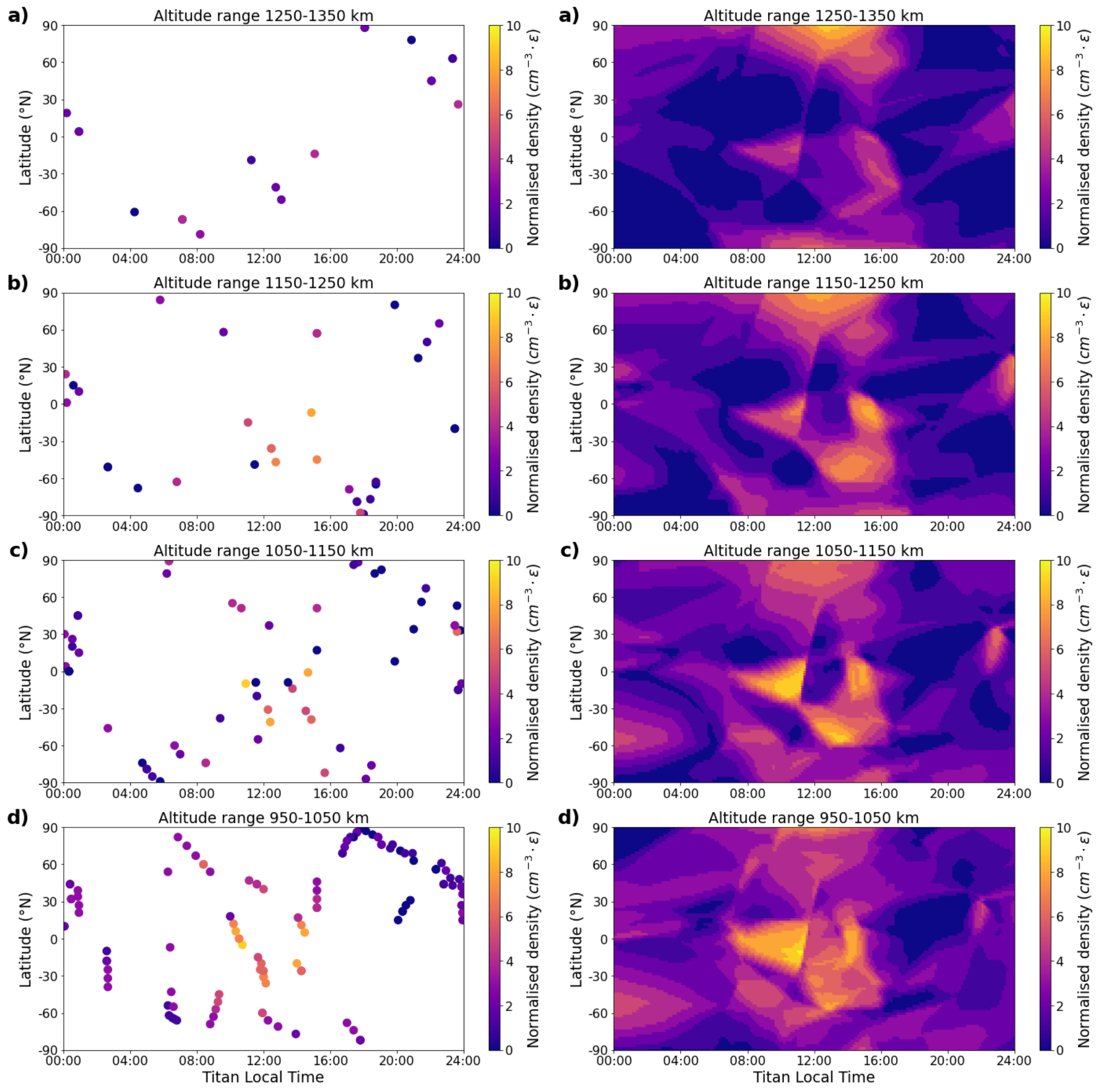


Figure 3.2: the left shows the data gathered by Cassini, and the right shows the network predicting the density for the entire plot. This builds on the hypothesis of higher density existing mostly in the centre of the plot. We can see that there are: regions of high density concentrated around ±30°N latitude and 12:00 to 13:00 Titan local time at low altitudes, and regions of high density near the poles at higher altitudes. Regions that are mainly affected by one or two points would benefit most from additional data, reducing sharp edges in different contours.

Conclusions

During the four weeks of my research placement, I became familiar with the concept of neural networks. I have faced and overcame several challenges such as using trial and error to fine-tune the accuracy of the predicted outcomes.

I have shown that a Neural Network can assist scientists, by predicting the motion of charged particles. Following this, I have created a Neural Network to try and get a better understanding of the physical processes on Titan. The atmosphere of Titan contains various physics and chemistry mysteries, leading to the second hypothesis: can a Neural Network be used to understand what is happening from a limited dataset gathered by space missions? The graphs above show that this is indeed possible by revealing underlying patterns in the data which are not obvious when looking at the original data. However, these results are preliminary and so caution should be exercised when interpreting them.

Acknowledgements

I would like to thank Dr Ravindra T. Desai (my supervisor) for all the help and scientific knowledge he provided, this assignment would not have been possible without him.