New Tools for Quantifying Turbulent Behaviour in Black Hole Accretion Discs

Alexander Agedah, Selwyn College

May 2023

Abstract

The net vertical magnetic flux in accretion discs surrounding black holes can significantly effect the process of accretion. Regions where there is little net vertical flux undergo standard and normal evolution (SANE) whereas regions with significant net vertical flux can become magnetically arrested. Traditional methods have been used to attempt to identify the two different states in 3D global accretion disc simulations, however have failed to do so on a local scale. This project introduces a novel deep learning tool with the goal of ultimately identifying these two states in global simulations. The model would work by predicting the net vertical flux that would be required in a local simulation to produce a given region in a global simulation. As a first step, this project tackled an analogous problem of predicting the net horizontal magnetic flux associated with 4×4 observations from 2D magnetic Rayleigh-Taylor instability (RTI) simulations. This was done using multilayer perceptron (MLPs). Results from testing various hyperparameters suggested that for this problem, wide MLPs are preferred over deep MLPs, the optimal MLP likely has $\sim$100,000 parameters and that the rectified linear unit (ReLU) is the preferred activation function. The best model was able to assign observations to 3 magnetic RTI simulations with different net horizontal fluxes with over 98% accuracy. In some cases, it was also able to accurately predict the net horizontal flux for observations from simulations with a net horizontal flux that had not been seen during training. The findings suggest that a similar model could be used on a 3D global accretion disc simulation to predict the net vertical flux required in a local simulation to produce a given region. This would help with distinguishing SANE and magnetically arrested regions.

1 Introduction

The magnetorotational instability (MRI) (Balbus and Hawley, 1991; Balbus and Hawley, 1998) is a well-studied process capable of generating turbulence in black hole accretion discs. The turbulence generated by the MRI transports angular momentum radially outwards, which causes material to fall inwards. Accretion discs where the MRI governs angular momentum transport are called standard and normal evolution (SANE). As material falls inwards during accretion, the vertical component of the magnetic field can be transported inwards, causing it to accumulate at the inner edge of the disc (Avara et al., 2016). In environments where the large-scale magnetic field is significant, the disc can eventually become magnetically arrested out to a radius set by the large-scale field. In these regions, the magnetic field in the disc becomes large enough to suppress the MRI. The suppression of the MRI, along with the mechanism for angular momentum transport in environments where the MRI is suppressed, are currently not well understood. McKinney et al. (2012) suggested that the magnetic Rayleigh-Taylor instability (RTI) plays a significant role in driving angular momentum transport in magnetically arrested regions whereas Chatterjee and Narayan (2022) suggested angular momentum transport is mostly driven by a wind.

Observations provide evidence that discs around neutron stars and supermassive black holes can be become magnetically arrested (Comparat et al., 2019; Gold et al., 2020). Additionally, 3D magnetohydrodynamic (MHD) simulations have suggested black hole accretion discs have regions that undergo MRI-driven turbulence, as well as regions are magnetically arrested (Begelman et al., 2022). Traditionally, quantities such as the ratio of gas pressure to magnetic pressure have been used in global simulations to distinguish the two different states. The time average of these quantities are typically taken over large volumes of the disc and regions where the magnetic field is large are labelled as magnetically arrested.

Traditional methods have not been effective in fully showing how SANE and magnetically arrested regions might interact. It is possible that regions that are currently identified as magnetically arrested are much more complex in that there could be sub-regions that are SANE. In order to understand the full complexity of hybrid discs, a new tool is needed which can perform the local identification of the states of a disc. This would be a tool which could take a small region of the disc as an input and produce an output indicating whether the region was SANE or magnetically arrested. The key difference between this tool and previous attempts at the problem (Begelman et al., 2022), is that this tool would work on small, local regions of the disc. Since it would work locally, it would be able to identify the behaviour of the disc at a resolution that has not yet been achieved.

This project introduces artificial neural networks (ANNs) as a new tool to identify SANE and magnetically arrested regions of accretion discs in global simulations. The ANN would be trained by giving it small regions of discs from local simulations. The net vertical magnetic flux in these simulations would range from the zero net flux limit to the net flux where the MRI is suppressed (Salvesen et al., 2016). During training, the model would learn to predict the net flux of the local simulation that small regions of discs came from. The model would then be taken to a global simulation where there were both magnetically arrested and SANE regions. It would take a small region of the global simulation as an input and predict the net flux for that region of disc. This value could be interpreted as the net vertical magnetic flux that would be required in a local simulation to produce the given region. If the predicted value was below the threshold at which the MRI is suppressed, then the region could be identified as SANE. If the predicted value was above the threshold, it could be identified as magnetically arrested.

Rather than tackling the final problem of predicting the net vertical magnetic flux in 3D simulations, this project looked at developing an ANN which could predict the net horizontal magnetic flux in 2D magnetic RTI simulations. 2D data was chosen over 3D data since it would have been unfeasible to find an optimal model using 3D data due to the large computational cost associated with training ANNs on 3D data. It was important to establish how different model design choices effected performance since the No Free Lunch theorem for supervised learning (Wolpert, 1996) implies that no machine learning model should be preferred over another a priori. Instead, the optimal model is purely dependent on the specific problem. This means one of the main goals of this project was to learn about the relationship between model design choices and performance for models which predict the net flux associated with small regions of MHD simulations. This would help with establishing some principles which could then be used when extending the model to 3D. The magnetic RTI (Kruskal and Schwarzschild, 1954) was chosen over the MRI because has a well-defined 2D evolution, whereas the MRI fails to drive continuously drive turbulence in 2D. Additionally, the magnetic RTI serves as a good 2D analogue to the MRI. This is because it suppressed by a sufficiently strong magnetic field, which is analogous to the way a disc can become magnetically arrested when the vertical component of the magnetic field is sufficiently large.

The outline of the report is as follows. In §2, the simulation data that was used is described. In §3, the method for converting the data into a format suitable for supervised learning is described.

In §4, the multilayer perceptron (MLP) is introduced as the class of ANN that was used. §5 details the first supervised learning problem. In short, this problem involved training an MLP to predict the time snapshot a small region of a simulation came from. §6 details the second supervised learning problem. This problem involved training an MLP to predict the net horizontal magnetic flux of the simulation that a small region came from. The report is concluded §7.

2 Simulation Data

The data sets used in the project came from 2D simulations of the magnetic RTI. The RTI (Rayleigh, 1900; Taylor, 1950) arises when there is an interface between a dense fluid that is supported above a light fluid against gravity. The interface is unstable to perturbations and so as time passes, the instability develops and drives turbulence. During this process, gravitational potential energy is converted to kinetic energy. When a magnetic field parallel to the interface is included, (Kruskal and Schwarzschild, 1954) directionality is introduced to the system. Specifically, the magnetic tension is able to suppress high wavenumber perturbations that are parallel to the magnetic field.

The simulations were generated by PLUTO (Mignone et al., 2007). In this project, it was used as a finite-volume code that integrated a system of conservation equations. The conservation of mass is given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0, \tag{1}$$

where ρ is the mass density and $\vec{u}$ is the fluid velocity. The conservation of momentum is given by

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot [\rho \vec{u}\vec{u} - T] = 0, \tag{2}$$

where T is the stress tensor. In ideal MHD, the stress tensor is

$$T = -pI + \frac{1}{4\pi G}\left(\vec{g}\vec{g} - \frac{1}{2}g^2 I\right) - \frac{1}{\mu_0}\left(\vec{B}\vec{B} - \frac{1}{2}B^2 I\right) \tag{3}$$

where I is the identity matrix, p is the gas pressure, $g = -\vec{\nabla}\phi$ is the gravitational field and $\vec{B}$ is the magnetic field. The induction equation is given by

$$\frac{\partial \vec{B}}{\partial t} = \nabla \times (\vec{u} \times \vec{B}). \tag{4}$$

The conservation of energy is given by

$$\frac{\partial}{\partial t}\left[\rho(\frac{1}{2}u^2 + \phi + e) + \frac{B^2}{2\mu_0}\right] + \nabla \cdot \left[\rho\vec{u}(\frac{1}{2}u^2 + \phi + h) + \frac{\vec{E} \times \vec{B}}{\mu_0}\right] = 0, \tag{5}$$

where e is the internal energy per unit mass, $h = e + p/\rho$ is the specific enthalpy and $\vec{E}$ is the electric field. Once an equation of state is provided, the system of equations becomes closed. For all simulations in this project the ideal gas equation of state was used

$$\rho e = \frac{p}{\gamma - 1}, \tag{6}$$

where γ is the ratio of heat capacities. The ratio of heat capacities was set to $\gamma = \frac{5}{3}$. A brief overview of the specific numerical set-up that was used is given below. Readers can see PLUTO User's Guide for more a detailed explanation of the settings that were used.

Basic Options The Roe Riemann solver was used to solve the ideal MHD equations in 2D Cartesian coordinates. Linear reconstruction was used for the spatial order of integration and for time-stepping, Hancock was used. The size of the physical domain was 1×2. This was resolved by 100 cells in the x-direction and 200 cells in the y-direction. This gave 20,000 cells in total for each simulation. The x boundary conditions were periodic and the y boundary conditions were reflective.

Physics-Dependent Options The use of numerical methods means that the magnetic field does not naturally remain divergence-free over time. To ensure the $\nabla \cdot \vec{B} = 0$ condition was satisfied, constrained transport was used. For the equation of state, the ideal gas law was used (Equation 6). The physical parameters that were used for the simulations were $\eta = 2.0$ and $g = -0.1$. η is the ratio of the density of the two parts of the fluid. This means that the initial density of the fluid was 2 in the upper half of the fluid and 1 in the lower half (Figure 1). g is the acceleration

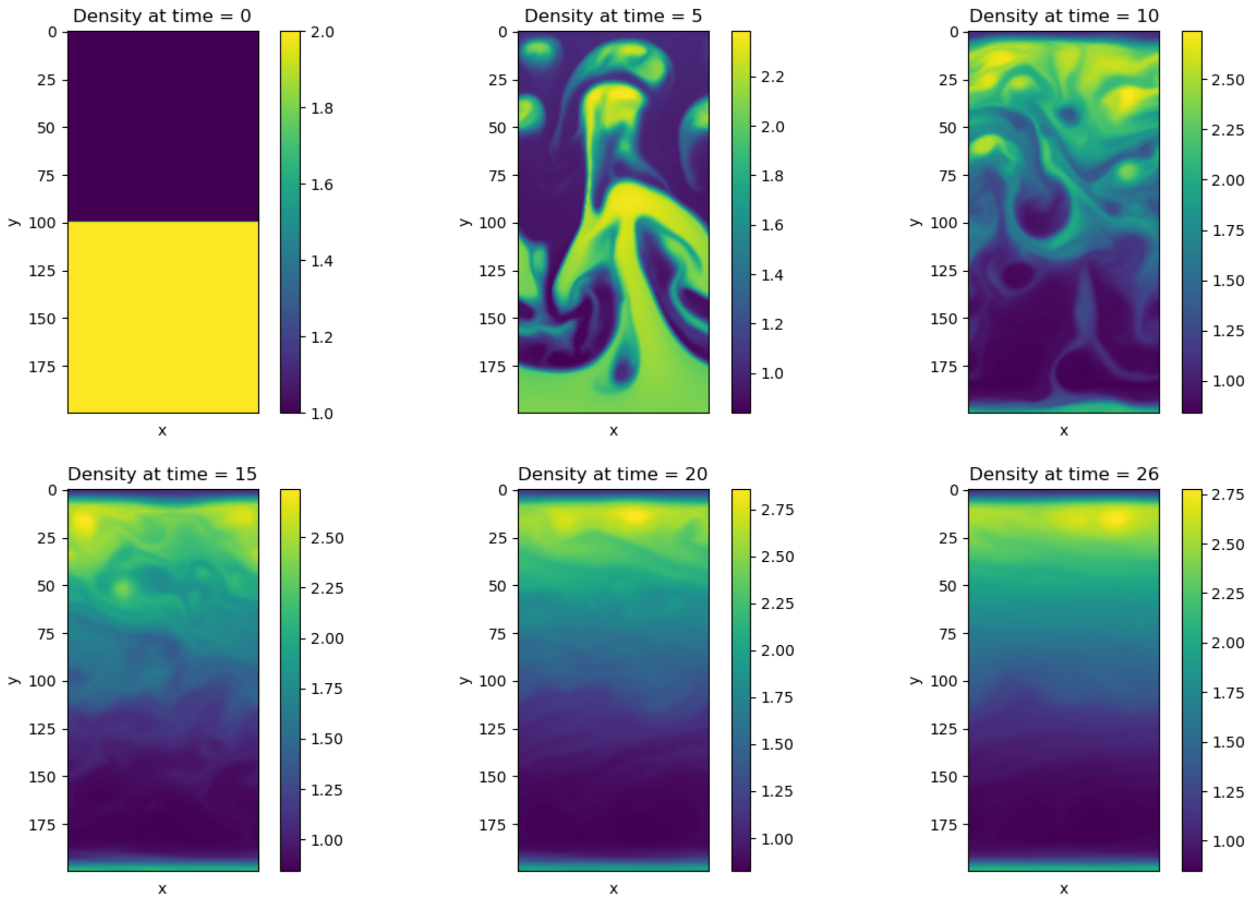


Figure 1: Images showing the density evolution for one of the with $\chi = 0.1$. Initially, the density of the fluid is 2 and 1 in the upper and lower halves respectively. The initial perturbation to the vertical velocity causes the instability to develop and drive turbulence.

due to gravity. A random perturbation to the vertical velocity was used to get the Rayleigh-Taylor instability started. χ is defined such that $B_x(x, y, t = 0) = \chi B_c$, where B_c is the critical value above which perturbations parallel to the magnetic field are stabilised (Boyd and Sanderson, 2003, p. 99). The χ parameter is proportional to the net horizontal magnetic flux, since the net horizontal flux is conserved. This means χ is the 2D analogue of the net vertical magnetic flux in 3D MRI simulations. In total, there were 11 simulations with χ parameters of 0.03, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.13, 0.15, 0.20 and 0.80. The data for each simulation came as 6 variables: ρ, p, B_x, B_y, u_x and u_y. Snapshots of these 6 variables were taken at different moments in time for each simulation.

3 Creating Observations

3.1 Supervised Learning

The branch of machine learning that this project deals with is supervised learning. In supervised learning, the data set consists of observations $(X_1, Y_1), ..., (X_n, Y_n)$, where $Y_i \in \mathbb{R}$ is a random variable called the response and $X_i \in \mathbb{R}^p$ is a random vector whose components are called predictors or features. In supervised learning, the goal is to find a relationship between the features and the response $f(X_i) \approx Y_i$. This can be done to better understand how the features effect the response and/or to predict future responses for a new observation X_{n+1}.

To transform a simulation into observations, at each time snapshot the simulation was split into overlapping 4×4 grids. Each simulation consisted of 20,000 cells therefore this gave 18,816 4×4 observations for each time snapshot. Notice that the number of observations for each time snapshot is less than 20,000 because observations were not taken if the grid would violate the vertical or horizontal boundaries. The vertical boundary was reflective therefore it would not have made sense, from a physical standpoint, to allow observations to cross the vertical boundaries. The observations could have been taken in a manner which allowed them to cross the horizontal boundaries since this boundary was periodic. Since observations crossing the horizontal boundaries would have only made up 3% of all observations, these were ignored. Each simulation had $\sim$30 time snapshots which gave them $18,816 \times 30 \sim 500,000$ observations each.

4×4 observations were chosen as the size of observations based on two opposing factors. On one hand, small observations were thought to be good as they would allow a similar model used on a 3D global simulation to show the full complexity of hybrid discs. On the other hand, small observations were thought to be bad as they would limit the amount of information a model could use to learn about relationships in the data. For example, 2×2 observations would prevent the model from being able to use information involving second order derivatives of fluid variables. The consequence of having insufficient information in observations would be that even the most optimal model would be unable to make accurate predictions. 4×4 observations were considered a size which balanced these two considerations

3.2 Features

Each observation consisted of $4 \times 4 = 16$ cells and each cell had 6 fluid variables. This meant each observation had $6 \times 4 \times 4 = 96$ features in total. Each of these features represented a single fluid variable at one of the cells in the observation. For example, there were 16 features corresponding to the value of p in each cell of the observation: p in the top right cell, p in the bottom right cell, p in the bottom left cell etc...

3.3 Response Variables

3.3.1 χ Response

A model which could be used in a 3D global simulation to identify SANE and magnetically arrested regions would need to be trained on observations from local MRI simulations with net vertical magnetic fluxes ranging from the zero net flux limit to the net flux where the MRI is suppressed. In this setting, the net vertical flux for the simulation that an observation came from would be used as the response variable. This means when the model was taken to 3D, it would be able predict the net vertical flux that would be required in a local simulation to produce the given region. This value could then be used to identify the observed region as magnetically arrested or SANE based on whether it was above or below the critical value where the MRI becomes suppressed.

The χ parameter is proportional to the net horizontal magnetic flux in the 2D magnetic RTI simulations that were used in this project. This means training the model on simulations with different χ parameters is the 2D equivalent to training it on 3D MRI simulations with different net vertical fluxes. Additionally, using χ as the response variable is the 2D analogous to using the net vertical flux as the response in 3D MRI simulations. If a model could successfully learn to use the features to predict the χ parameter, then it would have succeeded at the 2D analogue to the final problem. This would suggest a similar model could be trained on 3D data to predict the net vertical magnetic flux.

3.3.2 Time Response

Even though using χ as the response is the 2D analogue to the response that would be used in the final 3D setting, it was not the response variable that was used chosen in the first part of the project. Instead, the time snapshot that an observation came from was the response that was used. For example, an observation coming from the 10^{th} time snapshot would have had a response of 10. Time was used as the response for the first part of the project while different models were being evaluated. Once the best model had been found, it was trained with χ as the response.

The reason why time was used during hyperparameter optimisation is because it was computationally cheaper to perform hyperparameter optimisation while using time as the response. This is because when time was the response, models could be trained using observations from a single simulation whereas when χ was the response, models had to be trained using observations from multiple simulations. This increased the cost of training models massively. Specifically, on a 2.4 GHz 8-Core Intel Core i9 processor, the final model took $\sim$30 minutes to train when time was the response whereas it took $\sim$300 minutes to train when χ was the response and observations were taken from 7 different simulations. It therefore would have been unfeasible to test many models while using χ as the response.

The approach of using time during hyperparameter optimisation was considered appropriate because it was hypothesised that the model which was optimal when time was the response would also be close to optimal when χ was the response. The reason this was thought to be true is because ultimately, the best model in a machine learning problem is the one whose complexity matches the complexity of the data the model is trying to describe. In both cases, the data is generated by the equations of MHD and so the optimal model in either case is one which has sufficient complexity to learn about relationships in the data that arise as a result of these equations. Since the same equations generate the data in both cases, the relationships are of similar complexity and so the optimal model is likely very similar in both cases.

4 Deep Learning

4.1 Supervised Learning

Approaches to finding the relationship $f(X_i) \approx Y_i$ can be classified as parametric approaches, where an assumption is made about the functional form of the relationship between the features and the response, and non-parametric approaches, where no assumption is made about the functional form. The linear regression model is a parametric approach that assumes the relationship between the features and response takes the form

$$f(X_i) = \beta_0 + \sum_{j=1}^{p} \beta_j X_{ij}, \tag{7}$$

where p is the number of predictor variables and $\beta_0, \beta_1, ..., \beta_K$ are model parameters. The data set is then used to estimate the model parameters $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_K$. The linear regression model can then predict the response for an observation (X_i, Y_i) using $\hat{Y}_i = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j X_{ij}$.

4.2 Single-layer Perceptron

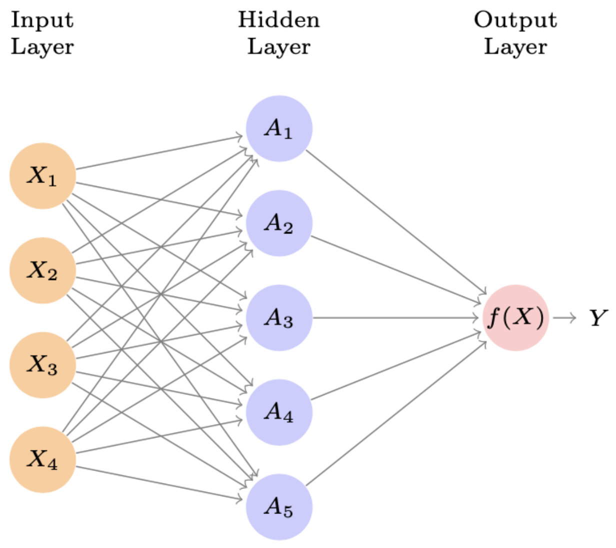


Figure 2: A single layer perceptron that takes 4 features as an input, has 5 units in the hidden layer and produces a single output (James et al., 2013, p.405)

.

Deep learning, is a specific branch of machine learning which deals with a class of models called ANNs. A single-layer perceptron (SLP) is simple type of ANN. The SLP assumes the relationship between the features and response is given by

$$f(X_i) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k, \tag{8}$$

where A_k are activations and K is the number of units in the hidden layer. Figure 2 shows a neural network architecture diagram of an SLP. The SLP is equivalent to a linear regression model

(Equation 7) that uses the activations as features. The activations are computed by taking a linear combination of the original features and applying a non-linear function called the activation function

$$A_k = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{ij}\right), \tag{9}$$

where g is the activation function that is specified in advance and w_{k0}, w_{k1}, ..., w_{kp} are model parameters. The activations in the hidden layer can therefore be considered transformations of the original features. The use of a non-linear function to transform the original features means that an SLP can learn much more complex relationships than any linear model.

4.3 Multilayer Perceptron

A multilayer perceptron (MLP) is an extension of the SLP perceptron. Rather than the model having a single hidden layer, an MLP can have any number of hidden layers. This means after the first hidden layer, activations are computed by taking linear combinations of activations in the previous hidden layer and applying the activation function. In this investigation, MLPs were the class of ANNs that were used and they were implemented using *Keras* and *TensorFlow* (Chollet et al., 2015; Abadi et al., 2015).

4.3.1 Model Hyperparameters

In this investigation, the model hyperparameters that were varied were the number of hidden units in each hidden layer, the number of hidden layers, the activation function and the initialisation of the model parameters. Details of the values for these hyperparameters and how they were varied are specified throughout §5 and §6. Model hyperparameters that were kept constant were the loss function, optimiser and the learning rate. Details about the values for these hyperparameters and why they were chosen are given below.

Regression vs Classification In regression problems, the response variable is treated as a continuous random variable whereas in classification problems, the response is treated as a discrete random variable that can only take on certain values called classes or labels. For the problems in this project, either approach could have been used. For example, when the model was being trained to predict the time for an observation, it could have been trained to predict the time as a continuous variable or it could have been trained to assign the observation to one of the times that it had seen during training. Similarly, when the model was being trained to predict χ, it could have been trained to predict χ as a continuous variable or it could have been trained to assign the observation to one of the simulations it had seen during training. There are two reasons why the problems were approached as regression problems rather than as classification problems.

The first reason why a regression approach was taken is because of how the data is structured. If a classification approach was taken, then the model parameters for predicting each category would not have been related. This means the model would be trying to learn to predict categories as different problems. For example, a model trying to predict the time snapshot for observations from the $\chi = 0.10$ simulation would be learning to predict the category as 27 different problems, without realising there is structure to how the classes are related. It would not realise that that an observation from the 10th time snapshot is somewhere between an observation from the 5th time snapshot and the 15th time snapshot. Since the model would not know how the classes are related, it would likely not perform as well as one that realised the classes had an order. The same argument can be used for when χ was used as the response.

The second reason why a regression approach was taken is related to the final application of the model. Ultimately, a model which could identify SANE and magnetically arrested regions would need to be able to extrapolate beyond values of the response that it has seen during training. This is because the magnetically arrested state of a disc only occurs in global simulations, and so would not be available to be trained on in local simulations. If the model was trained as a classifier, then it would only assign observations in a global simulation to the classes it has seen during training. This means when the model came across a region in a global simulation that was magnetically arrested, it would still attempt to assign it to an MRI simulation seen during training. By training the model as a regressor, it would be able to predict any net vertical magnetic flux associated with an observation. This would mean when it came across magnetically arrested regions, it can predict a value outside the range of fluxes seen during training, indicating it is magnetically arrested.

Loss Function In machine learning, a loss function is the function used to measure the difference between a model's prediction of the response and the actual value of response. In this project, mean squared error (MSE) was used since this is the default choice of loss function for regression problems. This means that during training, the model parameters were chosen to minimise the MSE on the training data

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{10}$$

where $\hat{Y}_i$ is the model's prediction of the response for the observation (X_i, Y_i). Mean absolute error (MAE) is another loss function that is used for regression problems

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|. \tag{11}$$

Similarly to MSE, it penalises overprediction and underprediction equally. For the application of this model, it seemed reasonable to penalise overprediction and underprediction equally and so MAE could have also been used. MSE was preferred over MAE since MSE is differentiable everywhere. This makes it easier to use optimisation techniques to minimise MSE.

Optimiser and Learning Rate The Adam (adaptive moment estimation) optimiser (Kingma and Ba, 2017) was used to train the all MLPs with a learning rate of 0.001. Adam is an optimisation algorithm that combines the properties of two other popular optimisation algorithms, momentum optimisation (Polyak, 1964) and RMSProp (Hinton and Tieleman, 2012). Adam was chosen because it is an adaptive learning algorithm. Specifically, it computes adaptive learning rates for each model parameter. This means it requires less tuning of the learning rate hyperparameter. This is advantageous because choosing a proper learning rate for an optimiser can be difficult. A learning rate that is too small means that the model parameters converge to an optimal solution very slowly, while a learning rate that is too large can prevent convergence all together. The Adam optimiser has four main hyperparameters: the learning rate, momentum decay, scaling decay and smoothing term. The values used for these were 0.001, 0.1, 0.999, and 10^{-8} respectively as these are the default values proposed in Kingma and Ba (2017).

Early Stopping Early stopping was the main technique used to prevent overfitting. This means that training was stopped once the model trained for 5 epochs without a reduction in the validation

MSE. This is the MSE calculated on a data set that is not used by the model during training. The model parameters were then reset back to the the model parameters with the lowest validation MSE. This method was chosen as it significantly reduced the computational cost of training. L_1 and L_2 regularisation are other techniques that are commonly used to prevent overfitting, however these were not employed as they cannot be used with the Adam optimiser (Loshchilov and Hutter, 2018).

4.4 Performance Metrics

4.4.1 Accuracy and Mean Squared Error

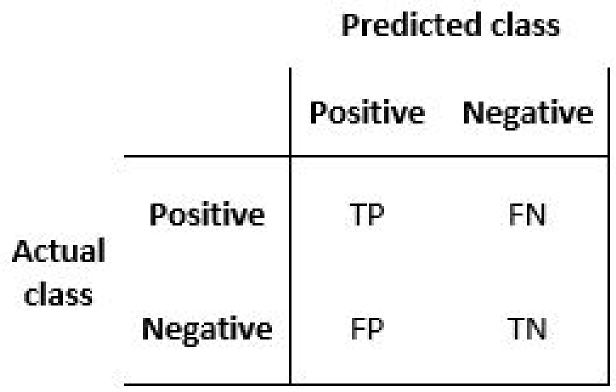
MSE (Equation 10) and accuracy were the two main numerical metrics used to compare the performance of the models. Good models were those with a low MSE and a high accuracy. Accuracy is given by

$$\text{Accuracy} = 1 - \frac{1}{n}\sum_{i=1}^{n} I(Y_i \neq [\hat{Y}_i]) \tag{12}$$

where I is an indicator function which takes on the value 1 for incorrect classifications. Accuracy is a metric that is typically used in classification problems rather than regression problems. In order to use this metric in a regression setting, the model's predictions of the response had to be rounded to the nearest class. This is denoted by $[\hat{Y}_i]$ in Equation 12. The main benefit of using accuracy as metric in this project was that it was easier to interpret than MSE.

4.4.2 Confusion Matrix

The confusion matrix is a metric that was used to assess the performance of the models when predicting the time for an observation. These are tables showing a model's prediction for the response and the true value of a response. An example of a binary confusion matrix is given in Figure 3. Confusion matrices are another metric that are typically used in classification problems



		Predicted class	
		Positive	Negative
Actual class	Positive	TP	FN
	Negative	FP	TN

Figure 3: A binary confusion matrix. Correct classifications are shown on the diagonals with true positives (TP) in the top left and true negatives (TN) in the bottom right. The off-diagonal elements represent incorrect classifications with false negatives (FN) in the top right and false positives (FP) in the bottom left.

rather than regression problems. In order to use this metric in a regression setting, the model's predictions of the response had to again be rounded to the nearest class.

4.4.3 Bias

Bias is a metric that was used assess the performance of the final model which predicted χ. It measures the difference between the model's average prediction of χ and the actual value of χ. It was calculated as

$$\text{Bias}_\chi = \bar{\hat{Y}}_\chi - \chi \tag{13}$$

where $\bar{\hat{Y}}_\chi$ is the sample mean of the model's prediction for observations coming from a simulation with a given χ. Bias was an important metric for understanding how the model should be used in a 3D global simulation. In the 3D global setting, if the model took a $4 \times 4 \times 4$ observations of the simulation as an input, then for each cell in the simulation, it would be able to make a maximum $4 \times 4 \times 4 = 64$ predictions of the net vertical magnetic flux. It would be able to make a prediction of the net flux each time the cell appeared in a different place in the input observation. One way to handle this would be use the mean of the 64 predictions as the final prediction for the net flux for each individual cell in the simulation. If this method was adopted, then it would be desirable to have a model whose predictions where unbiased. This would mean even if the model had a large MSE, after averaging over 64 predictions, the model's average prediction would likely be close to the true value of the net flux.

4.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of ANN that have performed very well on image classification tasks. For the problem at hand, we could consider each observation as a 4×4 image with 6 channels, where each channel of the image would correspond to a certain fluid variable. With this view taken, a CNN may seem like a good choice of model to use. Additionally, they have already been used to quantify MHD turbulence in Peek and Burkhart (2019). Despite this, MLPs seemed like the better ANN to use.

The first reason why MLPs were chosen is because the observations were very small. To see why this is important, suppose we had a data set consisting of 128×128 images. This is the same size as the observations in Peek and Burkhart (2019). This would mean that each observation has $128 \times 128 = 16384$ cells. Since there are 6 fluid variables, this would give each observations $16384 \times 6 = 98304$ features. An MLP with only 1,000 units in the first hidden layer (which would not be enough for such a large input) would have 98,305,000 elements in the matrix of weights feeding from the input layer to the first hidden layer. The large number of model parameters would mean that the risk of overfitting would be high and that training would be computationally very expensive. A CNN on the other hand is structured in a way such that the number of model parameters does not depend on the size of the input. This means it is better suited for taking large images as inputs. For this project, the observations only had $4 \times 4 \times 6 = 96$ features. This means an MLP with 100 units in the first hidden layer would have $97 \times 100 = 9700$ elements in the matrix of weights feeding from the input layer to the first hidden layer. This is a reasonable number of model parameters for a neural network.

The second reason an MLP was chosen is because the observations are slightly different from the images that CNNs typically have had success with identifying. For example, if a CNN was trained to identify animals, it would need to be able to recognise an animal whether it was translated, scaled, or rotated. The structure of CNNs means they can recognise the animal regardless of how it is transformed. This is a property that MLPs do not have. In this project, there is no notion of an object that can be translated, rotated or scaled in relation to some background. This means the CNN's translational, rotational and scaling invariance does not give it an advantage over the MLP.

5 Predicting the Time for Observations

Train-Valid-Test Split Train-valid-test split was the technique used to evaluate the performance of the models throughout this section. Firstly, observations were taken from a single simulation in the manner described in §3. The response variable for each observation was the time snapshot that the observation came from. The observations were then split into training, validation and test data sets. The training, validation and test data sets contained 80%, 10% and 10% of the observations respectively. Stratified sampling was used to split the observations so that each data set contained an equal number of observations from each time snapshot. This was implemented using the *preprocessing* module from the *scikit-learn* library (Pedregosa et al., 2011). In this project, the training data set refers to the data set that was used to train the model. The validation data set refers to the data set that the model was evaluated on during training epochs. Unless otherwise specified, this will be the data set that the performance metrics are calculated on. The test data set refers to the data set that was used to evaluate the final model.

5.1 Initial Model Hyperparameters

The No Free Lunch theorem for supervised learning states that no model should be preferred over another a priori. This means a number of simplifying assumptions had to be made to choose the initial hyperparameters for the MLP. The decision was made to start with 2 hidden layers because an MLP with 2 hidden layers is able to approximate any function to arbitrary accuracy (Heaton, 2005, p. 128).

A common rule of thumb that is used in the deep learning community is to use a number of hidden units that is less than the number features to avoid the curse of dimensionality (Bengio, 2012). Since each observation had 96 features, this rule would suggest using less than 96 units in each hidden layer. If additional features were added, then the number of features would grow and the optimal number of units to use in each hidden layer might increase. For this reason, 128 units was chosen for each hidden layer as this would accommodate models that were trained using additional features, and would not be too large causing the model to overfit to the training data set.

The vanishing gradients problem Hochreiter (1998) is a common issue that occurs in deep neural networks. It causes optimisation algorithms to fail to converge to a good solution. Glorot and Bengio (2010) showed that one of the causes of the problem was the use of the sigmoid activation function

$$\sigma(z) = \frac{e^z}{1 + e^z}. \tag{14}$$

Since then, the rectified linear unit (ReLU) activation function has become the most widely used activation function in deep learning as it helps prevent the vanishing gradients problem and is computationally efficient. It is given by

$$\text{ReLU}(z) = \max(z, 0). \tag{15}$$

This was the activation function that was initially chosen.

Another hyperparameter that was found to contribute to the vanishing gradients problem in Glorot and Bengio (2010) was the initialisation method for the model parameters. *Keras*' default choice, Glorot initialisation with a uniform distribution, was initially chosen as the authors found it could significantly improve convergence. For ease, the model with these initial hyperparameters will be referred to as Model 1.

Table 1 shows the performance of Model 1 when predicting the time snapshots of observations from the $\chi = 0.05$, 0.1 and 0.15 simulations.

χ	MSE	Accuracy/%
0.05	5.5	20.0
0.10	1.8	35.1
0.15	8.1	19.0

Table 1: The performance of the Model 1 when predicting the time for observations from the $\chi =$ 0.05, 0.10 and 0.15 simulations.

5.2 Data Preprocessing

5.2.1 Methods

In the previous section, the observations that were given to the Model 1 had not be preprocessed. In other words, none of the features had been scaled or transformed in any way. Including preprocessing steps before training a model ensures the scale of the all features and the response are similar. This can make it easier for the optimiser to find a minimum in the loss function (Nawi et al., 2013). To investigate the effect of preprocessing, Model 1 was trained again, except preprocessing steps were used prior to training. Min-max scaling was applied to the responses using

$$Y' = -1 + 2\frac{(Y - \min(Y))}{\max(Y) - \min(Y)}, \tag{16}$$

where Y is the response for a general observation (X, Y) and Y' is the min-max scaled response. This meant the transformed responses took on values between 1 and -1. Standard scaling was applied to the features using

$$X' = \frac{X - \bar{X}}{S} \tag{17}$$

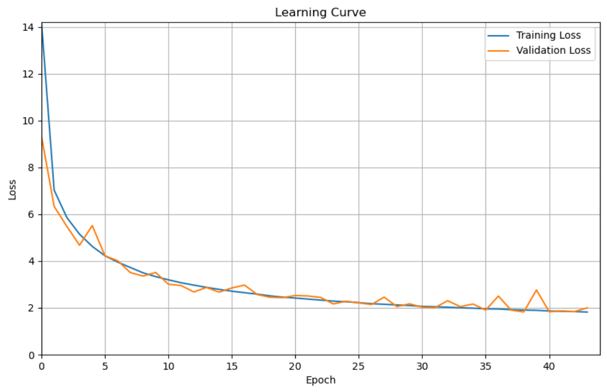
where $\bar{X}$ is the sample mean for the vector of features and S is the sample standard deviation for the vector of features. This meant the transformed features all had a mean 0 and a standard deviation of 1. Both the sample mean and the sample standard deviation were computed using the training data set. The results from training the MLP with the preprocessed data are given in Table 2.

χ	MSE	Accuracy/%
0.05	0.013648	28.8
0.10	0.005807	47.5
0.15	0.015421	27.1

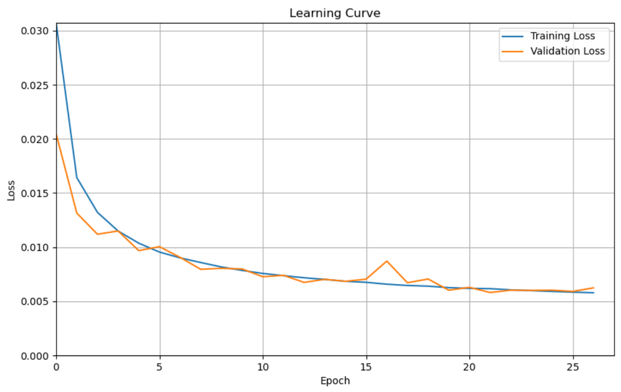
Table 2: The performance of Model 1 when it was trained on preprocessed data to predict the time for observations from the $\chi = 0.05$, 0.10 and 0.15 simulations.

5.2.2 Discussion

In §5.1 Model 1 was trained on unscaled responses whereas in this section, it was trained on scaled responses. Since the scale of the response was different in the two cases, MSE could not be used to compare the performance of the models. Accuracy however, was an appropriate metric to compare the models as it is independent of the scale of the responses. Preprocessing the data before feeding it to the model increased the accuracy of the model by 8.8%, 12.4% and 8.1% for the $\chi = 0.05$, $\chi = 0.10$ and $\chi = 0.15$ simulations respectively. Note that these increases refer to the absolute amount the accuracy increased by. Future references to changes in accuracy should also be taken to mean the absolute change in accuracy. The second benefit of using preprocessing was that it decreased the time to train the model. This is because the model required fewer epochs to train when preprocessing steps were included (Figure 4), while each epoch took approximately the same amount of time in both cases.



(a) Learning curve for Model 1 when trained on un-preprocessed data.



(b) Learning curve for Model 1 when trained on pre-processed data.

Figure 4: Learning curves for Model 1 when trained it was trained to predict the time for observations from the $\chi = 0.10$ simulation. Training the model required 17 fewer epochs when it was trained on preprocessed data.

The better performance and faster training both support the findings of Nawi et al. (2013), in that the optimiser was able to converge to a minimum in the loss function more easily. For this reason, the decision was made to continue to use preprocessing before training all subsequent models.

5.3 Permutation Feature Importance

5.3.1 Method and Results

ANNs are often called black boxes since it it difficult to know how the model is making predictions. Permutation feature importance is one metric that can help give insight into how the ANN is working. The permutation feature importance of a feature measures the decrease in the model's performance when that feature is randomly shuffled (Breiman, 2001). A high permutation feature importance indicates that the model is very dependent on the feature. It is important to note that the value has no statistical meaning. This means it cannot be used to perform hypothesis tests or interval estimation. It simply provides an indication of how dependent a model is on a feature relative to other features. Since each observation was a 4×4 grid, each observation had 16 features corresponding to a each fluid variable. Rather than considering the permutation feature importance of each feature, for example the permutation feature importance of the density in the top left cell, the permutation feature importance of each fluid variable was considered. This means all 16 features corresponding to a fluid variable were randomly shuffled when computing the permutation feature importance of that fluid variable. Specifically, the permutation feature importance of a fluid variable was calculated as

$$\mathrm{PFI}_j = \frac{\mathrm{MSE}_j - \mathrm{MSE}}{\mathrm{MSE}}, \tag{18}$$

where MSE_j is the MSE of the model when the 16 features corresponding the j$^{\text{th}}$ fluid variable were randomly shuffled. The permutation feature importance of the six fluid variables were computed for the $\chi = 0.05$, 0.1 and 0.15 simulations and the results are given in Table 3.

χ	p	ρ	u_x	u_y	B_x	B_y
0.05	32.5	32.8	12.3	7.3	5.9	4.3
0.10	94.2	85.2	40.8	21.1	16.8	14.6
0.15	34.5	35.9	13.6	10.1	11.0	7.3

Table 3: The permutation feature importance of the fluid variables when Model 1 was trained to predict the time of observations from the $\chi = 0.05$, 0.1 and 0.15 simulations.

5.3.2 Discussion

For pressure and density, the decrease in the MSE of the model was over $2\times$ higher than for any of the other fluid variables across all three simulations. This indicates that they were the most important fluid variables being used by the model. While it it difficult to know exactly how the variables were being used, plots of the pressure evolution (Figure 5) indicate that quantities related to the change in pressure across the 4×4 grid, such as the pressure gradient, may have some of the derived features that the model was using for prediction.

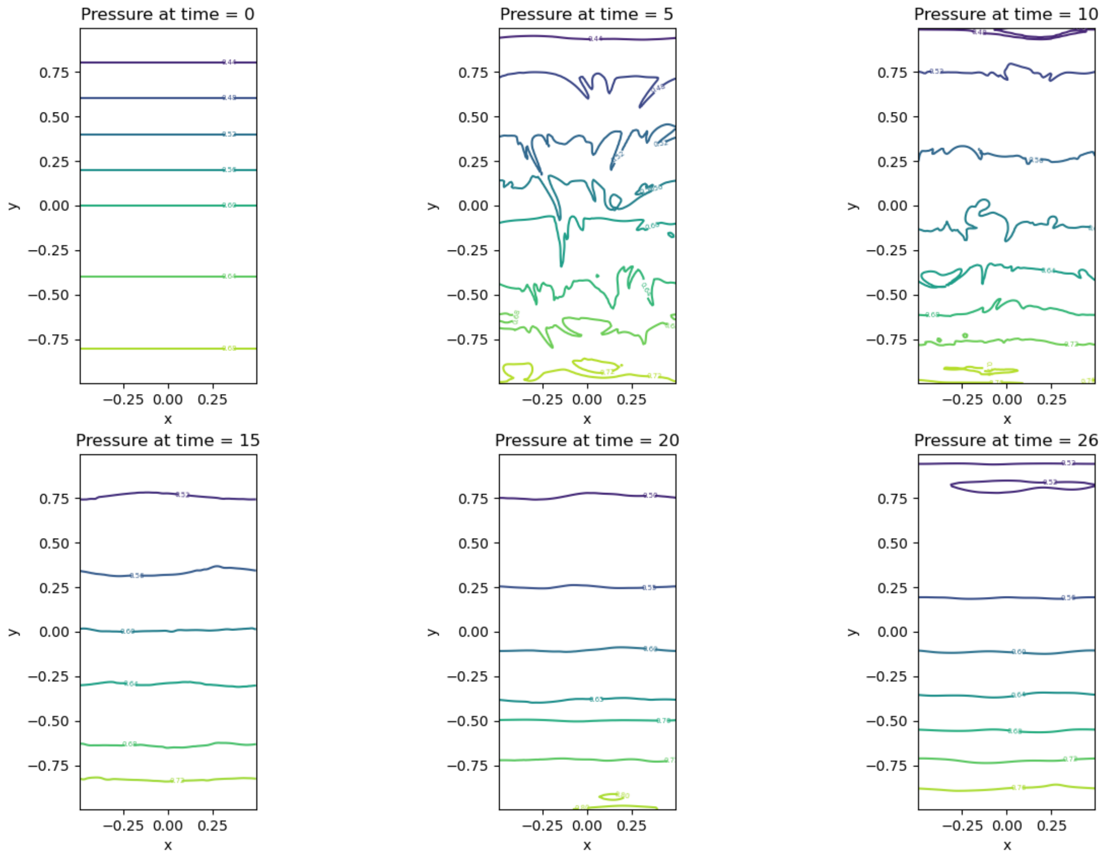


Figure 5: Images of the pressure contours in the $\chi = 0.10$ simulation at different times.

At times 0, 15, 20 and 26, the contours are mostly horizontal. This means the pressure gradient was relatively constant and in the y-direction throughout the simulation. At times 5 and 10, the contours are jagged, meaning the pressure gradient would have been far less uniform within observations. This means it could be a good feature for distinguishing observations that came at early-intermediate times, and may have been one of the ones that the model learnt during training. These plots also help illustrate the importance of using observations that are large enough to allow the model to learn features that involve derivatives of the basic fluid variables. Observations which are smaller than 3×3 would completely prevent the model from being able these features, and so the model would likely be unable to quantify the behaviour of the simulation.

5.4 Feature Engineering

Since each activation in the hidden units of an MLP (Equation 9) can be thought of as a transformation of the original features, an MLP can theoretically learn any feature which might be physically important for quantifying the behaviour of an accretion disc. Despite this, feeding an MLP the most relevant features will often achieve a better performance than purely relying on the model to learn the relevant features (Gibert et al., 2022).

5.4.1 Additional Fluid Variables

Along with the 6 fluid variables that came with the raw data, the following fluid variables were also considered:

- Magnitude of the magnetic field: $B = \sqrt{B_x^2 + B_y^2}$
- Magnetic energy density: $E_m = \frac{B^2}{2\mu_0}$
- Speed: $u = \sqrt{u_x^2 + u_y^2}$
- Speed squared: $u^2 = u_x^2 + u_y^2$
- Kinetic energy density: $E_K = \rho \frac{u^2}{2}$
- One over plasma beta: $\frac{1}{\beta} = \frac{B^2}{2p\mu_0}$
- Alfvén wave speed: $v_A = \frac{B}{\sqrt{\mu_0 \rho}}$
- Speed of sound: $v_s = \sqrt{\frac{\gamma p}{\rho}}$
- Cross-helicity density: $H_m = B_x u_x + B_y u_y$

This is a non-exhaustive list of some physically important scalar quantities in MHD. Vector quantities have two components in 2D, therefore adding a vector fluid variable increases the number of features for each observation by $2 \times 4 \times 4 = 32$. Additionally if the model was taken to 3D, each additional a vector fluid variable would increase the number of features by $3 \times 4 \times 4 = 48$. This is a problem since machine learning models begin to perform badly once the dimensions of the feature space becomes large relative to the number of observations (Venkat, 2018). This problem is often referred to as the curse of dimensionality. To avoid the curse of dimensionality, only scalar quantities were considered since these only added 16 features to each observation.

To determine whether any additional scalar fluid variables should be added, Model 1 was trained using all six original fluid variables, along with all nine fluid variables listed above. The permutation feature performance of each fluid variable was then calculated using the same method as in §5.3.1. This was done for the $\chi = 0.05, 0.1$ and 0.15 simulations. The permutation feature importance of the all the fluid variables that were used in the model were then computed (Table 4).

χ	p	ρ	u_x	u_y	B_x	B_y
0.05	20.6	14.9	7.7	4.4	3.6	2.1
0.10	64.4	29.3	31.6	19.6	9.1	9.3
0.15	42.1	183.9	14.8	10.5	33.4	9.0

(a) The permutation feature importance of the original six fluid variables.

χ	v_s	u	u^2	E_K	v_A	H_m	$\frac{1}{\beta}$	B	E_m
0.05	15.9	8.0	7.14	6.4	5.0	1.6	2.5	4.0	2.0
0.10	74.7	37.3	24.6	13.9	13.4	6.5	4.1	7.1	8.1
0.15	58.6	10.4	5.4	4.8	11.0	9.7	9.9	9.2	6.8

(b) The permutation feature importance of the nine additional fluid variables.

Table 4: The permutation feature importance of the fluid variables. These were computed using Model 1 trained to predict the time of observations from the $\chi = 0.05, 0.1$ and 0.15 simulations.

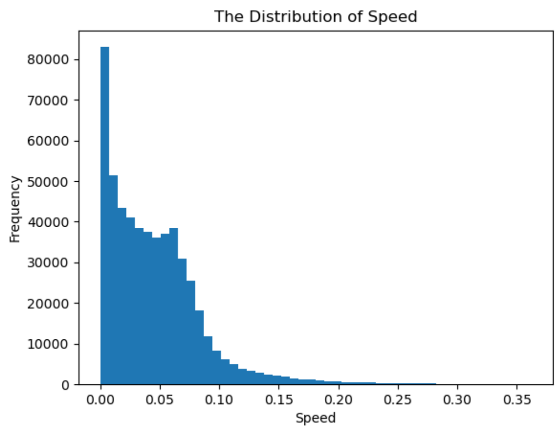
5.4.2 Discussion

Since permutation feature importance is only meant to indicate the relative importance of features, comparisons of values were only made within simulations. For example, the fact the feature importance of u on the $\chi = 0.10$ simulation was higher than ρ on the $\chi = 0.05$ simulation was not considered indicative of anything about the importance of those fluid variables.

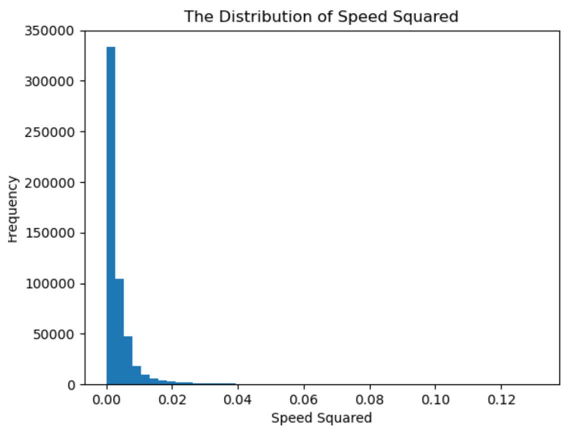
Across the three simulations, the fluid variables with the highest permutation feature importance were pressure, density and the speed of sound. Speed of sound appearing as an important variable seems intuitive since it is simply a combination of pressure and density, which were previously seen to be the most important fluid variables. Other than the speed of sound, none of the other additional variables were consistently more important than the original six variables.

Despite speed of sound appearing as an important variable, the decision was made to not include it as an additional variable in future models. The reason this decision was made is because the cost associated with increasing the dimensionality of the feature space did not seemed justified considering the speed of sound is a relatively easy feature for a model to derive. For example, a model which computed the differences between the pressure and density in the cells would be able to use this derived feature as a proxy for the speed of sound. Since this is feature is linear in pressure and density, it can be derived in the first hidden layer of the model.

Quantities which combined 3 or more of the original quantities via non-linear transformations, such as the Alfvén wave speed and kinetic energy density, are examples of variables which would have been considered if their permutation feature importance was higher than some of the original six fluid variables. Since they did not appear to be more important than ρ, p, B_x, B_y, u_x or u_y, the decision was made to not use feature engineering before training subsequent models.



(a) The distribution of u in the cells of the $\chi = 0.1$ simulation.



(b) The distribution of u^2 in the cells of the $\chi = 0.1$ simulation.

Figure 6: The distributions of u and u^2 in the cells of the $\chi = 0.1$ simulation. The u^2 distribution had a higher skewness than the u distribution.

5.5 Dimension Reduction

5.5.1 Methods and Results

§5.4.1 referred to the curse of dimensionality as a problem that all machine learning models face. To combat the issue and improve the performance of models, the dimensions of the feature space is often reduced before a model is trained. Principal component analysis is one of the most popular techniques for reducing the dimensions of the feature space. It involves creating $M < p$ linear combinations of the original features, and then fitting a model to this reduced set of features. Other techniques include non-negative matrix factorisation and linear discriminant analysis. Rather than using one of these methods that learn from the data itself, knowledge of the physical quantities was used. Since the vector quantities, velocity and magnetic field, each had $4 \times 4 \times 2 = 32$ features associated with them, the method used to reduce the dimensions of the feature space was simply to take the magnitude of the vector quantities. This method was preferred since using a method that respected the underlying physics seemed like it might give the best results. The magnitudes were preferred over the magnitudes squared since the squared quantities had distributions with a large amount of skewness (Figure 6) and machine learning models typically prefer features which have little skewness.

The MSEs of Model 1 on the 3 data sets when it was trained using p, ρ, u and B are given in Table 5).

χ	MSE	Accuracy/%
0.05	0.0200	24.4
0.10	0.0129	36.7
0.15	0.0279	22.9

Table 5: The performance of the Model 1 when to predict the time of observations from the $\chi = 0.05, 0.1$ and 0.15 simulations using a reduced set of features.

5.5.2 Discussion

The accuracy of Model 1 decreased by 4.4%, 10.8% and 4.2% on the $\chi = 0.05, 0.1$ and 0.15 simulations respectively. These results suggest that the direction of the magnetic field and velocity contain information which is important for the model to make accurate predictions. This means dimension reduction techniques which lose this information should be avoided. Based on these findings, the decision was made to continue with the six original fluid variables for subsequent models and not to use any dimension reduction techniques.

5.6 Hyperparameter Optimisation

This subsection deals with finding the optimal hyperparameters for the MLP. The hyperparameters that were adjusted were: the number of hidden layers, the number of units in each hidden layer, the activation function and the initialisation method. Hyperparameter optimisation is computationally expensive therefore it could not be performed on all three simulations. Instead, the $\chi = 0.10$ data set was simulation was selected as it had the fewest time snapshots and therefore the fewest observations. This meant training could be done for the least computational cost.

Interpreting the performance of the models was also not important for this process. The most important thing regarding performance was simply the relative performance between models. For this reason, MSE was the only metric that was used when comparing models as it could be computed more easily than accuracy.

5.6.1 Hidden Layers and Units

The two hyperparameters that were first to be tuned were the number of hidden layers and units per hidden layer. For simplicity, the number of units in each hidden layer was kept the same for all hidden layers. This meant there were only two hyperparameters to tune. The number of hidden layers considered were 2, 4, 8 and 16. The number of units in each hidden layer considered were 32, 64, 128 and 256. This gave a total of $4 \times 4 = 16$ models. For models with 32 units in each hidden layer, only models with 8 and 16 hidden layers were tested since a model with small hidden layers would require more hidden layers for good performance. Additionally, for models with 256 units in the hidden layer, only models with 2 and 4 hidden layers were tested since a model with large hidden layers would require less hidden layers for good performance. The results from the hyperparameter optimisation are given in Table 6.

Discussion The results from the hyperparameter optimisation seem to go against some common rules of thumb in the deep learning community about the width and depth of neural networks. Firstly, it is often suggested that the number of units in the hidden layers should be less than the number of features (Bengio, 2012). Despite this, the models with 128 and 256 units in each hidden layer achieved the best 4 performances. Secondly, deep ANNs have generally been found to perform better than wide ANNs (Geron, 2017, p. 349). Despite these findings by the deep learning community, the deepest MLPs all performed poorly in this setting. Specifically, for MLPs with a given number of units in the hidden layer, the MLP with 16 hidden layers performed the worst than the shallower MLPs.

A possible reason why wide MLPs performed better than deep MLPs is because deep MLPs are not required to compute many of the important physical quantities in MHD. For example, a quantity related to the changes in pressure, such as $\vec{\nabla} p$, appears as if it could be important for the model (§5.3.2). This type of feature could be derived in the first hidden layer by taking the difference between the pressure in different cells. A similar argument can be made to explain how

Units	Hidden Layers	Model Parameters	MSE
128	8	128129	0.0023
256	2	90881	0.0030
128	4	62081	0.0032
256	4	222465	0.0037
64	4	18753	0.0050
64	8	35393	0.0052
128	2	29057	0.0056
128	16	260225	0.0057
32	8	10529	0.0072
32	16	18977	0.0076
64	2	10433	0.0080
64	16	68673	0.0097

Table 6: The MSE for MLPs with different units in each hidden layer and different numbers of hidden layers.

a model could derive features related to $\vec{\nabla} \times \vec{u}$ and $\vec{\nabla} \times \vec{B}$ in the first hidden layer. Furthermore, proxies for quantities which combine multiple fluid variables, such as v_s and B, can be derived in the first hidden layer by summing or subtracting fluid variables in the same cell. Since these quantities can be derived in the lower hidden layers of an MLP, it is possible that deep MLPs have a type of complexity that is not suitable for MHD data.

The final thing to note from the results is that the models with ~100,000 parameters generally outperformed the models with fewer parameters. This helps give a guide to the number of model parameters that MLPs likely require for optimal performance.

Based on the results in this section, the decision was made to use 8 hidden layers and 128 units in each hidden layers for all subsequent models.

5.6.2 Activation Functions

Glorot and Bengio (2010) showed the importance of a good activation function in helping with the vanishing gradients problem. Additionally the activation function is the means by which non-linearity is introduced to an ANN, allowing it to learn complex patterns in the data. Since 2010, a number of different variants of the ReLU have been developed and each variant introduces a different amount of complexity to the model. This means a variant of the ReLU may achieve superior performance if it allows the model to discover more complex patterns in the data. Alternatively, variants achieve inferior performance if they result in a model with too much complexity relative to the data. In this project, four variants of the ReLU were considered to better understand whether adding additional complexity to the existing model would improve performance. These were the exponential linear unit (ELU), the scaled ELU (SELU), the Gaussian error linear units (GELU) and Swish. A brief description of the activation functions are given below.

ELU Clevert et al. (2016) proposed the exponential linear unit (ELU)

$$\mathrm{ELU}_\alpha(z) = \begin{cases} \alpha(e^z - 1) & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases} \tag{19}$$

where α is a hyperparameter that is typically set to 1. This activation was considered in this project as it outperformed the ReLU, along with some of its variants, in the authors' experiments. The α hyperparameter was set to the standard value of 1.

SELU The scaled ELU (SELU) was later introduced (Klambauer et al., 2017). It is given by

$$\mathrm{SELU}(z) = \begin{cases} \lambda\alpha(e^z - 1) & \text{if } z < 0 \\ \lambda z & \text{if } z \geq 0 \end{cases} \tag{20}$$

where $\alpha \approx 1.05$ and $\lambda \approx 1.67$. This is a scaled variant of the ELU. An interesting property of the SELU is that for MLPs, the network will self-normalise. This solves the vanish gradients problem and means the SELU can outperform other activations, particularly for deep MLPs.

GELU Hendrycks and Gimpel (2016) proposed the Gaussian error linear units (GELU) which is given by

$$\mathrm{GELU}(z) = z\Phi(z), \tag{21}$$

where $\Phi(z)$ is the standard normal cumulative distribution function. This activation has been found to outperform the ReLU and GELU in some tasks since the complex shape of the GELU means that model can find it easier to learn complex patterns.

Swish Hendrycks and Gimpel (2016) also introduced the sigmoid linear unit (SiLU)

$$\mathrm{SiLU}(z) = z\sigma(z), \tag{22}$$

where $\sigma(z)$ is the sigmoid function (Equation 14). The GELU outperformed the SiLU in their paper however later, Ramachandran et al. (2017) used automatic search techniques to find that the best activation function was the SiLU. Their findings showed that it even outperformed the GELU. In their paper, they renamed the SiLU to Swish.

The MLP used to compare the activation functions was the MLP with 8 hidden layers and 128 units in each hidden layer. This was again trained on the $\chi = 0.10$ simulation. The activation function was varied and the MSE was used to compare the models. For the ReLU, ELU, GELU and Swish, the model parameters were initialised using He initialisation (He et al., 2015), since this is the initialisation that is most commonly used for variants of the ReLU. For the SELU, the model parameters were initialised using LeCun normal initialisation since this is one of the requirements for self-normalisation. The results from the hyperparameter optimisation are given in Table 7.

Activation	MSE
ReLU	0.0042
Swish	0.0067
GELU	0.0070
SELU	0.0114
ELU	0.0121

Table 7: The MSE for the MLPs with different activation functions.

Discussion The best performing model from the 5 that were considered was one which used the ReLU. This means the ReLU may be the optimal activation function to use when applying MLPs to small observations of MHD turbulence. This may because physically important quantities in MHD can be computed using linear relationships. As discussed in §5.4.2 and §5.6.1, features which involve the products or quotients of two fluid variables, such as the speed of sound, can be derived by taking the sum or difference between fluid variables in a cell. §5.6.1 also mentioned that features which involve derivatives can be derived by taking the difference between the same fluid variable in different cells. Since the ReLU is linear for $z \neq 0$, it would be the activation which would best allow the model to learn about important features related to products, quotients and derivatives. The other activations have greater non-linearity which likely introduces a type of complexity that is not desirable in this setting.

An alternative explanation is that the ReLU simply achieved the best performance due to the way in which hyperparameter optimisation was done. Since the number of hidden layers and number of hidden units were optimised using the ReLU, it is possible that the model with 128 units and 8 hidden layers is the optimal size for a model using the ReLU. For a more complex activation function such as the GELU, the model with 128 units and 8 hidden layers may be too complex. Instead, it may be optimal to use a slightly smaller model. This slightly smaller model may then be better suited for predicting the time of observations. To confirm whether this was the case, hyperparameter optimisation could have been done, varying the number of hidden layers, number of units in each hidden layer and the activations simultaneously. Including the activations in the hyperparameter optimisation in §5.6.1 would have increased the search space by $5\times$ up to 80. Computational restrictions meant that it would have been unfeasible to evaluate this number of models.

Despite the ReLU performing the best, changing the initialisation from Glorot initialisation with a uniform distribution to He initialisation caused the MSE of the model to increase by 86.6%. For this reason all subsequent models continued to use the ReLU with Glorot initialisation with a uniform distribution.

5.7 Final Model

The final model had 8 hidden layers with 128 units in each hidden layers, giving a total of 128,129 model parameters. The ReLU was used as the activation function and Glorot initialisation with a uniform distribution was used to initialise the model parameters. The remaining hyperparameters are the same as those specified in §4.3.1. For ease, this model is referred to as Model 2. Table 8 summarises the performance of Model 2 on the $\chi = 0.05, 0.10$ and 0.15 simulations. Since this model was the final model trained to predict the time of observations, its performance was evaluated using the test data set. Up until this point, the test data set had not been used and so it had no influence on the final model hyperparameters that were chosen.

χ	MSE	Accuracy/%
0.05	0.0057	41.7
0.10	0.0031	62.9
0.15	0.0127	31.6

Table 8: The performance of the Model 2 when predicting the time of observations from the $\chi = 0.05, 0.10$ and 0.15 simulations. Both the MSE and Accuracy were calculated using the test data sets.

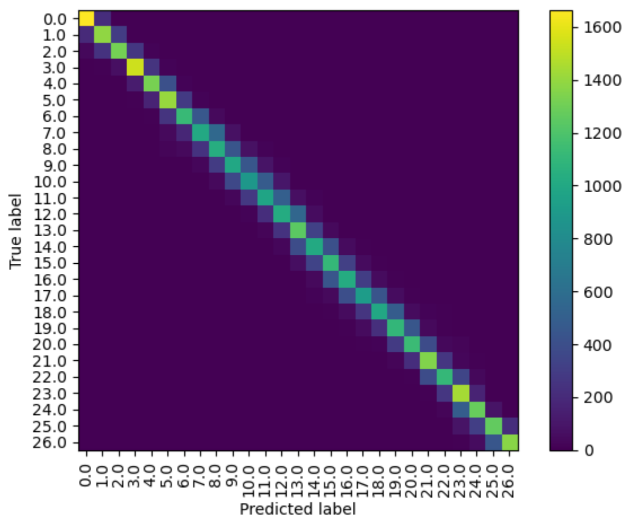


Figure 7: The confusion matrix of Model 2 when predicting the time of observations from the $\chi = 0.10$ simulation.

5.7.1 Discussion

The final model performed better than initial model that was proposed. The accuracy of Model 2 was 12.9%, 15.4 and 4.5% higher than Model 1 on the $\chi = 0.05, 0.10$ and 0.15 simulations respectively. Additionally, the MSE of Model 2 was 58%, 47% and 17% lower than Model 1 across the simulations.

In other machine learning tasks, models have achieved much higher accuracies than those achieved in this project. For example, the Modified National Institute of Standards and Technology (MNIST) database (Deng, 2012a) is a database of handwritten digits that is often used to train machine learning models to classify the digits. On this problem ANNs have achieved accuracies above 99.6% (Deng, 2012b). Compared to this accuracy, the accuracy of the model that was used seems low however there are a few important things to consider. Firstly, humans can achieve an accuracy around 98% classifying digits. In comparison, a human would likely struggle to place a 4×4 observation into the correct time snapshot. That is to say that this problem is arguably more difficult, and so an accuracy of 99.6% would be very hard to achieve. Secondly, the confusion matrix (Figure 7) shows a more complete picture of the model's performance. It shows that the model often predicts the time correctly to within 1 time snapshot. When the accuracy metric was relaxed so that classification was considered correct if the rounded time prediction was within 1 time snapshot of the actual time, the accuracy increased to 73.1, 86.4% and 66.9% respectively on the $\chi = 0.05, 0.1$ and 0.15 simulations. This shows that the model's performance is better than the original accuracy metric might suggest.

There is a clear difference in the performance of the model on the simulations. The model performs best on the $\chi = 0.10$ simulation and worst on the $\chi = 0.15$ simulation. While it is not clear why this was the case, a possible explanation is that overfitting was occurring on the model selection level. The hyperparameters for the model were chosen to achieve the best performance on the $\chi = 0.10$ data set, and it may be that these hyperparameters are not optimal for the $\chi = 0.15$ data set. If hyperparameter optimisation was performed on the $\chi = 0.15$ data set, then the performance on this data set might have risen to 60%, while the performance on the $\chi = 0.10$ data set might have dropped to 40%. To confirm this, hyperparameter optimisation would need be done simultaneously on all three data sets to see to what extent hyperparameter choice depends on χ.

6 Predicting The Net Horizontal Flux

The goal of §5 was to find the best model when time was the response variable. This model could then be applied when χ was the response, under the assumption that the best model for predicting the time of an observation will close to the best model for predicting χ. Predicting χ was important as this parameter was proportional to the net horizontal magnetic flux in the simulation (§2). This meant predicting χ was the 2D analogue to predicting the net vertical magnetic flux in 3D MRI simulations.

6.1 Model 1 vs. Model 2

In order to verify whether the assumption that Model 2 would be better than Model 1 with χ as the response (§3.3.2), the models were trained to predict χ for observations coming from the $\chi = 0.05, 0.10$ and 0.15 simulations. The observations were split into training, validation and test data sets in the same manner as in §5. Since the response variable was χ, the use of stratified sampling meant that ∼33% of observations in each data set came from each simulation. For both models, the data was preprocessed by applying min-max scaling (Equation 16) to the response and standard scaling (Equation 17) to the features. The results for the two models are given in Table 9.

Model	MSE	Accuracy
Model 1	0.0420	95.8%
Model 2	0.0144	98.4%

Table 9: The performance of Model 1 and Model 2 when predicting χ for observations from the $\chi = 0.05, 0.1$ and 0.15 simulations.

6.1.1 Discussion

Both models performed far better in predicting χ for observations compared to predicting time. This is because the difference between observations from separate simulations is likely larger than the difference between observations from the same simulation at different times. In other words, observations from $\chi = 0.05$ and 0.10 are likely quite distinct, making it easy for the model to distinguish them, whereas observations from time snapshot 10 and 11 are likely similar, making it difficult for the model to distinguish them.

The steps that were taken to improve the model for predicting the time seemed to carry over for predicting χ. This is because the MSE for Model 2 was 65.6% lower than for Model 1. This is larger than any of the improvements that were made by Model 2 on the time problem. This validates the assumption that better performance on predicting the time would translate to better performance on predicting χ. More generally, it supports the idea a model which is good at quantifying MHD behaviour based on small observations can be used across many different problems. In the case of this project, it supports the idea that a model which can predict the net horizontal magnetic flux in 2D magnetic RTI simulations can also be trained to predict the net vertical magnetic flux in 3D MRI simulations.

6.2 Predicting χ For Unseen Simulations

The final model which would attempt to identify SANE and magnetically arrested regions in a 3D global accretion disc simulation would do so by predicting the net vertical magnetic flux that would be required in a local simulation to produce the given observation. Since the net vertical flux is a continuous quantity, it is inevitable that the model would encounter regions that would require a net flux that had not been seen during training. This may be because the required net flux is some value between those which were seen during training. It may also be because the region is magnetically arrested, and so the net flux required is above any values seen during training. To understand whether it was possible for the model to predict the net flux in these regions, the model was evaluated on all 11 simulations that were available in the project, however it was only trained on the $\chi = 0.03, 0.05, 0.06, 0.07, 0.09, 0.10$ and 0.13. This meant observations from the $\chi = 0.08, 0.15, 0.20$ and 0.80 simulations were not used in training. To evaluate the model, each simulation, was given to the model and the model's mean prediction, MSE (Equation 10), bias (Equation 13) and standard deviation were computed (Table 10) . Here standard deviation refers to the sample standard deviation of the model's predictions. It therefore measured the variability of the model's predictions.

χ	MSE	Mean	Bias	Standard Deviation
0.03	0.0184	0.030	0.000	0.008
0.05	0.0213	0.054	0.004	0.008
0.06	0.0272	0.063	0.003	0.009
0.07	0.0311	0.072	0.002	0.010
0.08	0.0364	0.081	0.001	0.012
0.09	0.0341	0.090	0.000	0.011
0.10	0.0391	0.099	-0.001	0.012
0.13	0.0452	0.127	-0.003	0.012
0.15	0.0513	0.141	-0.009	0.010
0.20	0.1011	0.121	-0.079	0.021
0.80	0.4532	0.144	-0.656	0.007

Table 10: The performance of Model 2 when predicting χ for observations from various simulations. During training, the model did not see observations from the $\chi = 0.08, 0.15, 0.20$ and 0.80 simulations.

6.2.1 Discussion

For observations from simulations which the model saw during training, the model was able to predict χ to a reasonable accuracy. For these simulations, the model's average prediction was always correct to two decimal places. This shows that the model gives unbiased predictions of χ. The average MSE of the model's predictions for the data sets that were seen during training was 0.0316. This indicates that the degree of variability in the model's predictions was large enough for it to indicate that an observation came from the wrong simulation. As mentioned in §4.4.3, variability would not be problematic in the final application of the model if the model was unbiased. This is because the model's final prediction of the net vertical magnetic flux in the cell of a 3D simulation could be averaged over $4 \times 4 \times 4 = 64$ predictions. Additionally, it is reasonable to expect that there is a range of net magnetic fluxes that can produce a given observation. This means some degree of variability in the model's prediction is inevitable, as it simply reflects the fact that the observation could have come from a simulation with a different, but similar net flux.

The $\chi = 0.08$ simulation was not seen during training yet the model was still able to predict χ to a similar accuracy as for the simulations that were seen in training. The MSE of the model's predictions was only 0.0048 higher than the average MSE for the simulations seen in training. This means the error of the model's predictions did not increase significantly. Additionally, the model's bias was lower than the bias for the $\chi = 0.03, 0.06, 0.07$ and 0.13 simulations. This suggests that in a global 3D simulation, the model would be able to predict the net vertical magnetic flux for regions which would require a net vertical magnetic flux that had not been seen during training, as long as the net flux was within the range of net fluxes that had been seen during training.

Observations from the $\chi = 0.15$ simulation set had a χ parameter 15% above the maximum value seen during training. Here, the MSE of the model was 0.0197 higher than the average MSE for simulations seen in training. This means there was a noticeable increase in the error of the model's predictions. Additionally, the bias of the model's predictions increased in magnitude to 0.009. Despite this, the model was still able to indicate that the net horizontal magnetic flux was outside the range seen during training, since the mean prediction of χ was 0.141. For a model to correctly identify magnetically arrested regions, its predictions of the net vertical magnetic flux in a 3D global simulation would need to be above the highest net flux seen during training. As long as it could reliably predict that the net flux was above any net flux seen in training, then the exact value of the prediction would not be important for identifying magnetically arrested regions. The fact that this model is able to indicate when an observation has a net flux outside of the range seen during training suggests that in the 3D setting, a similar model would be able to do the same for magnetically arrested regions.

Observations from the $\chi = 0.20$ and 0.80 simulations had a χ parameter 54% and 515% above the maximum χ seen during training. For these observations, the model was unable to make accurate predictions. The MSE and magnitude of the bias was much larger than for the other simulation. This shows the model is limited in the amount it can extrapolate beyond the range of magnetic fluxes seen during training. Despite the model being inaccurate, it was still able to indicate that observations from the $\chi = 0.80$ data set were outside the range of values seen in training. This suggests that in highly magnetically arrested regions, the model could still indicate that the required net vertical flux would be above the threshold where the MRI is suppressed. Despite the predictions from the $\chi = 0.20$ data set having a lower MSE and bias than the predictions from the $\chi = 0.80$ data set, these predictions are worse considering the final application of the model. This is because the model was failing to indicate that the observations are coming from a simulation with a net flux outside the range of fluxes seen during training. In a global 3D simulation, this would mean the model would misclassify some regions as SANE when they were actually magnetically arrested. This suggests further work needs to be done to improve the model and avoid this type of error.

Despite this shortcoming, a model making this type of error may be of practical use in a global simulation. This is because the standard deviation of the model's predictions was $\sim 2\times$ larger for the $\chi = 0.20$ simulation than for any other simulation. This suggests that the standard deviation of the model's predictions for each cell in the simulation could be a useful metric for understanding where the model is failing. For cells where the standard deviation is low, then the model's mean prediction can be interpreted as reliable. For cells where the standard deviation is high, then the model's mean prediction can be interpreted as less reliable and likely coming from a regime that has not been seen during training. This would allow the experimenter to determine which regions being classified as SANE may actually be magnetically arrested.

7 Conclusion

This first goal of this project was to find the best model for predicting the net horizontal magnetic flux in 2D magnetic RTI simulations. The best MLP had 128 units in each hidden layer and 8 hidden layers, giving it a total of 128,129 parameters. The ReLU activation function was found to outperform the ELU, SELU, GELU, and Swish. Additionally, Glorot initialisation with a uniform distribution gave better performance than He initialisation. More generally, the following guidelines are proposed for MLPs which aim to predict the net flux:

- Observations should be at least 3×3 in 2D or $3 \times 3 \times 3$ in 3D.
- Features and responses should always be standardised before training.
- If dimension reduction techniques are used, they should aim to preserve information about the direction of the the magnetic field and velocity.
- MLPs should have $\sim$100,000 parameters to have sufficient complexity to describe the data.
- The ReLU should be used as the activation function.
- Wide MLPs should be used over deep MLPs. As a rule of thumb, one can double the number of features to find the number of units to use in each hidden layer.

The best MLP was found via the problem where the time snapshot was the response variable. When predicting the time of observations, the best MLP's MSE was on average, 41% lower than the initial model that was proposed. Despite not using the net flux as the response while iterating on models, the final model also performed better when predicting the net flux. It's MSE was 66% lower than the initial model that was proposed. This supports the idea that the model which was optimal when time was the response would also be close to optimal when the net flux was the response.

Once the best model was found, the goal of the project was to determine whether an MLP could predict the net horizontal magnetic flux associated with observations from various 2D magnetic RTI simulations. When trained and evaluated on observations from three different simulations, it could do this to 98.4% accuracy. Additionally, the model had some ability to predict the net flux for observations with net fluxes that had not been seen during training. The model only became limited in its ability to make predictions for observations with a net flux 54% and 515% above the largest net flux seen in training. The MSE for observations from these simulations was $2\times$ and $8\times$ higher respectively than for the other simulations that were used. Despite this, the model was still able to consistently indicate that observations from the simulation with the highest net flux had a net flux above any value seen in training. The overall conclusion from these results is that an MLP can predict the net flux in most cases. In cases where the model has to extrapolate far beyond the net fluxes that have been seen in training, the model begins to make large errors, however the predictions can still be useful.

The next stage in this line of work would be to take the model proposed in this project and use it to predict the net horizontal flux associated with observations from various 3D local MRI simulations. If the results achieved on the 3D data were similar to those achieved on here, then the model could be taken to a 3D global simulation. Based on these results, the model would likely be able to identify regions going undergoing MRI-driven turbulence and regions beginning to enter the magnetically arrested regime. For regions that are strongly magnetically arrested, the model may begin to fail. Despite this, an experimenter could use the variability of the model's predictions to infer where the model was failing. These regions where the model's predictions became highly variable could be interpreted as magnetically arrested as they are likely in a regime that the model has not seen in training. Additionally, a visual plot of the model's predictions may be of use to an experimenter. This is because there should be some degree of continuity in the model's predictions over space. Regions where there are sudden changes in the predictions may also be interpreted as regions that are magnetically arrested, as the model is likely failing.

References

Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Mark J. Avara, Jonathan C. McKinney, and Christopher S. Reynolds. Efficiency of thin magnetically arrested discs around black holes. *Monthly Notices of the Royal Astronomical Society*, 462 (1):636–648, 07 2016. ISSN 0035-8711.

Steven Balbus and John Hawley. Instability, turbulence, and enhanced transport in accretion disks. *Reviews of Modern Physics*, 121:90, 01 1998.

Steven A. Balbus and John F. Hawley. A Powerful Local Shear Instability in Weakly Magnetized Disks. I. Linear Analysis. *apj*, 376:214, July 1991.

Mitchell C Begelman, Nicolas Scepi, and Jason Dexter. What really makes an accretion disc MAD. *Monthly Notices of the Royal Astronomical Society*, 511(2):2040–2051, jan 2022.

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures, 2012.

T. J. M. Boyd and J. J. Sanderson. *The Physics of Plasmas.* Cambridge University Press, 2003.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, oct 2001. ISSN 0885-6125.

K. Chatterjee and R. Narayan. Flux eruption events drive angular momentum transport in magnetically arrested accretion flows. *The Astrophysical Journal*, 941(1):30, dec 2022.

Francois Chollet et al. Keras, 2015.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

J Comparat, A Merloni, M Salvato, K Nandra, T Boller, A Georgakakis, A Finoguenov, T Dwelly, J Buchner, A Del Moro, N Clerc, Y Wang, G Zhao, F Prada, G Yepes, M Brusa, M Krumpe, and T Liu. Active galactic nuclei and their large-scale structure: an eROSITA mock catalogue. *Monthly Notices of the Royal Astronomical Society*, 487(2):2005–2029, may 2019. doi: 10.1093/mnras/stz1390.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012a.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012b.

Aurelien Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Sebastopol, CA, 2017. ISBN 978-1491962299.

Daniel Gibert, Jordi Planes, Carles Mateu, and Quan Le. Fusing feature engineering and deep learning: A case study for malware classification. *Expert Systems with Applications*, 207:117957, 2022. ISSN 0957-4174.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Roman Gold et al. Verification of Radiative Transfer Schemes for the EHT. *apj*, 897(2):148, July 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

Jeff T. Heaton. *Introduction to Neural Networks with Java.* Heaton Research, Inc., 2005. ISBN 097732060X.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv e-prints*, art. arXiv:1606.08415, June 2016.

Geoffrey Hinton and Tihmen Tieleman. Neural networks for machine learning, lecture 6a, 2012.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Springer, New York, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks, 2017.

Martin David Kruskal and Martin Schwarzschild. Some instabilities of a completely ionized plasma. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 223 (1152), 1954.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam, 2018.

Jonathan C. McKinney, Alexander Tchekhovskoy, and Roger D. Blandford. General relativistic magnetohydrodynamic simulations of magnetically choked accretion flows around black holes. *Monthly Notices of the Royal Astronomical Society*, 423(4):3083–3117, jun 2012.

A. Mignone, G. Bodo, S. Massaglia, T. Matsakos, O. Tesileanu, C. Zanni, and A. Ferrari. Pluto: A numerical code for computational astrophysics. *The Astrophysical Journal Supplement Series*, 170(1):228, may 2007.

Nazri Mohd Nawi, Walid Hasen Atomi, and M.Z. Rehman. The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technology*, 11:32–39, 2013. ISSN 2212-0173. 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.

Fabian Pedregosa et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

J. E. G. Peek and Blakesley Burkhart. Do androids dream of magnetic fields? using neural networks to interpret the turbulent interstellar medium. *The Astrophysical Journal Letters*, 882(1):L12, sep 2019.

Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 12 1964.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

Lord Rayleigh. *Scientific Papers, Vol. II.* Cambridge University Press, Cambridge, 1900.

Greg Salvesen, Jacob B. Simon, Philip J. Armitage, and Mitchell C. Begelman. Accretion disc dynamo activity in local simulations spanning weak-to-strong net vertical magnetic flux regimes. *Monthly Notices of the Royal Astronomical Society*, 457(1):857–874, 01 2016. ISSN 0035-8711.

Geoffrey Ingram Taylor. The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 201(1065), 1950.

Naveen Venkat. The curse of dimensionality: Inside out, 09 2018.

David H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390, 10 1996. ISSN 0899-7667.