NEURAL NETWORKS FOR SELF LOCALISATION AND AGENT POSITIONING

AN APPLICATION TO ROBOCUP DATA

Western Sydney University: 200045 Quantitative Project

James Monks

25. 10. 2019

Supervisor: Oliver Obst

Abstract

This report examines RoboCup soccer simulation data in the context of common simulated agent problems, with the view to use machine learning methods to address them. Specifically, the localisation of simulated agents is addressed utilising the landmark data that falls within an agents field of view to predict for their coordinates on the field at any given point in time. A number of different neural network models are trained to generate these predictions and are compared to a baseline model that uses geometry to make predictions. It is concluded that the neural networks are much more accurate than the geometric model in every case, with the best performing neural network achieving an accuracy that would likely be suitable for implementation in a production environment.

List of Figures

Contents

1 Introduction

1.1 RoboCup

RoboCup is an international competition in which teams prepare artificial football teams to compete against each other and was originally held in (1997) as a competition to drive forward research in robotics and artificial intelligence systems. Different leagues have since been established that are devoted to improving specific aspects of these fields ("A Brief History of Robocup," n.d.). The large scale robotics league is designed to drive competition in macro movements in large robots. The small scale robots drive work into improving finer movements in smaller scale robots. The standard platform is a league in which all contestants use the same robots, and is designed to showcase the ability of a team to program for robot movement. The final major league and the one that this report focuses on is the simulation league. This league is not linked to any physical robot and as a result is much more accessible to teams. It allows teams to focus solely on the artificial intelligence aspect of the problem, and allows for a prototyping platform for teams to trial different AI approaches before integrating them with physical robots.

1.2 RoboCup Simulation Data

The simulation league of RoboCup takes place on a field of play that is 105m wide and 68m units high The field of play is enclosed within a slightly larger region that represents the range of the possible simulated positions (120m x 80m). This can be viewed as an exterior boundary on a regular field. It should also be noted that the simulated players will be referenced throughout the report as agents in order to follow convention. (Michael et al. 2017)

The agents are placed on the field with a limited field of view in which they perceive other agents, the ball and landmarks with a known location. These landmarks allow the agent to have some idea of where they are on the field without having absolute internal knowledge of their coordinates and will be key to the project goals. The use of these landmarks will be discussed more in section ()

The main source of data - considered here as "perceived data" - that is generated by each agent in the simulation is split up into 2 categories (Michael et al. 2017). The first is the landmarks category, which contains the relative distances and angles from the agent that fall within the

field of view. These landmarks include the lines of the field, goalposts, the corners of the lines and simulated points around the exterior boundary. The second category of data contains information about all entities on the field that are moving and within the field of view. These entities consist of agents from both teams and the ball; as this perception data is generated by one agent about other agents, the agent that generates the data will be known as the observer agent or simply the observer. Agents that are close to the observer and within the field of view have both their team and number known, however, agents that are far away from the observer may not have their number or even their team know (then denoted as an "unknown agent").

It should be noted that for all forms of perception data, there is a perception error factor that is proportional in some way to the distance of the entity from the observer. The exact process by which this occurs is unknown, however it is included to imitate the uncertainty that regular football players have when estimating distances.(Chen et al. 2001)

In addition to the perception data, there are also 2 higher level data sets. The ground-truth data set contains the exact location for every player and the ball at each time point in a game. There is only one of these files for each game and they are treated as a reference for conditions that are known to be true. There is also the landmarks data that is a universal file that describes the exact coordinates of the observable landmarks in a game and is the same for every game. The ground-truth data is only accessible to teams after the game has been played, whereas the landmarks data is public and known to the teams throughout the play.

The data perception data contains a high number of NaN records, as it is structured as one observation for each iteration in time and one variable for each metric (e.g. distance) on each entity (e.g. flag). This results in these NaN records as the majority of the entities on the field are not in the field of view of any given agent, and as such there is no reasonable value to assign except for NaN. This means that the data needs to be extensively cleaned before the modelling process can begin. This data cleaning is done differently for different methods, but generally consists of obtaining only the closest couple of flags and dropping the other variables for any given point in time. An excerpt from a perception landmarks data can be seen below:

# time	f b 0 dist	f b 0 angle	f b 0 Ddist	f b 0 Dangle	f b l 10 dist
1	64.1	16	NaN	NaN	72.2
3	64.1	31	NaN	NaN	72.2
5	63.4	-39	NaN	NaN	71.5
7	62.2	-47	NaN	NaN	70.1
9	60.3	-51	NaN	NaN	68.7
11	59.1	-54	NaN	NaN	66.7
13	57.4	-56	NaN	NaN	65.4
15	55.7	-58	NaN	NaN	63.4
17	55.7	-59	NaN	NaN	62.8
19	NaN	NaN	NaN	NaN	62.8
21	55.1	-58	NaN	NaN	62.8

1.3 Quantitative Project Goals

1.3.1 Self Localisation

This project aims to determine if it is possible to use neural networks to create an accurate prediction of an agent's own place on the field based on the information available to them. In cases in which there are 2 or more landmarks in the field of view of an agent, it is possible to use geometry to determine the theoretically exact coordinates of the agent. The issue with this as mentioned in section 1.2 is that there is perception error associated with these observations. This project specifically aims to determine if it is possible to use neural networks to improve on the predictions of on field locations made through the geometric method.

1.4 Background Theory

1.4.1 Neural Networks

Typical feed forward neural networks are made out of a series of layers of that are designed to capture key predictors of the outcome variables. Each layer consists of a number of nodes, which perform some transformation on input data, before passing the data on to the next layer. In subsequent layers, the value passed to each node is a weighted sum of all nodes in the previous layer, with each connection between nodes being assigned a weight. The input layer corresponds

to the input variables, then there are a number of hidden layers that are connected, and finally an output layer with the same number of nodes as desired predictions. In the case of predicting the coordinates of an agent on the field, this would be 2 nodes.

This layering process allows for important interactions between the predictor variables to be identified. Weights are trained across layers to optimise for the loss function, which in turn creates smaller groups of nodes that operate together which implies an interaction between input variables. This can also be seen as an identification of latent or confounding variables that were not directly observed, but have a strong influence on the outcome. Unfortunately, though neural networks excel at identifying interactions and latent variables, the exact nature of these interactions and variables is opaque, due to the complex interwoven node structure. This is what is meant when neural networks are referred to as "black box algorithms". (Tu 1996).

Weights on the connections between nodes are the parameters of the neural network that are optimised. They can be viewed in a similar way to the coefficients of terms in a linear regression. These weights are updated through a process known as back propagation. Back propagation works by computing the gradient of the cost produced with the current set and updates the weights accordingly as described by (Plaut and Hinton 1987). This is not as simple as in linear regression however, as the weights and values produced by them are functions of each previous layer. Back propagation works by applying the chain rule to each of the steps in the network progressively to work out how each individual weight needs to be updated. This is formalised below:

$$\delta^L = \nabla_a C \odot g^{L'}(z^L)$$
$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot g^{l'}(z^l)$$
$$\frac{\delta C}{\delta b^l_j} = \delta^l_j$$
$$\frac{\delta C}{\delta w^l_{jk}} = a^{l-1}_k \delta^l_j$$

Where $\nabla_a C$ a vector of variation of the cost C relative to the output of the network a^L_i, which is $\frac{\delta C}{\delta a^L_i}$. Also $\odot$ represents element-wise matrix multiplication. This algorithm was described and explained in this form by (Salloum 2019).

A key aspect of back propagation that should be noted is that the way in which the weights are updated uses a process known as gradient descent. At a high level, this process uses the

gradient of the loss function obtained to update the weights in the direction of the gradient. This process is performed iteratively in order to converge on a local minima on the loss function. There are a number of iterations on this method as the obtained local minima is not necessarily the minimum point of the loss function such as stochastic gradient descent (Bottou 2010).

1.4.2 Training, Validation and Testing Data

The data that will be used for training neural networks and assessing the results is sourced from the HELIOS_2017 team's games in the 2017 RoboCup. The group behind the HELIOS_2017 team are known to produce high quality simulation teams, as they consistent win the world championships or if not, come close. The data consists of 5 games repeated a number of times, with the first 2 rounds from each game being taken (resulting in 10 total matches). This data will be broken down into training and testing data sets in order to avoid over-fitting models to data and to obtain a fair assessment of model accuracy. The first 4 games (8 matches) will be used to train the models and then the last game (2 matches) will be predicted for. The data has been broken into games in this way as it is important for the player positioning portion of this project to not only prevent over-fitting on the training data in general, but also to avoid over-fitting for opposing teams, hence, a brand new team is needed in order to obtain a fair assessment.

1.5 Related Work

Robot localisation is an essential field for any form of automated robotic systems. This is particularly important in the case of self driving cars for example, as extremely accurate locations are needed to prevent accidents. This field has hence undergone research into a number of different methods approaching this problem. One such method is the use of kalman filters which take measurements that have uncertainty in them and updates them with prediction information generated from environmental generation. This process combines uncertainties in order to create a more accurate prediction subject to a lower degree of error. This is particularly useful in instances in which their is a gps system connected to the robot that is subject to an error. This has been demonstrated effectively by (Ashokaraj et al. 2004), however, it is not applicable to the project in question as there is not this inherent measurement available.

1.6 Geometric Applications

The main problem when trying to solve the self-localisation problem geometrically, is that all of the data that an agent has access to is relative to their position and orientation. The only absolute information that is made available is the coordinates of the landmarks. In order to solve this problem, the geometric method establishes a line between the two closest flags to a subject agent, uses trigonometry to obtain the resultant dimensions of the triangle formed by the flags and the player and finally rotating the established line in a way such that it is positioned in the direction of the player. Then it is simple to obtain the unit vector in this direction and multiply it by the observed distance of the player from the flag.

More Formally:

$$let : \alpha = \angle PF_1F_2$$

$$and\ let : \vec{v} = \vec{F_1F_2} \cdot \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(alpha) & \cos(\alpha) \end{pmatrix}$$

$$then : \vec{F_1P} = \frac{\vec{v}}{\|\vec{v}\|} \cdot \|\vec{F_1P}\|$$

The implementation of this process has been tested to ensure that the results are accurate when predicting for known coordinates.

For known coordinates:

$A = (0, 0)$

$B = (0, 1)$

$C = (1, 0)$

The$\triangle ABC$ formed by these coordinates has the following properties:

$\angle BAC = 1.571\ rad$

$\angle ABC = 0.785\ rad$

$\angle BCA = 0.785\ rad$

$\vec{BC} = 1.414$

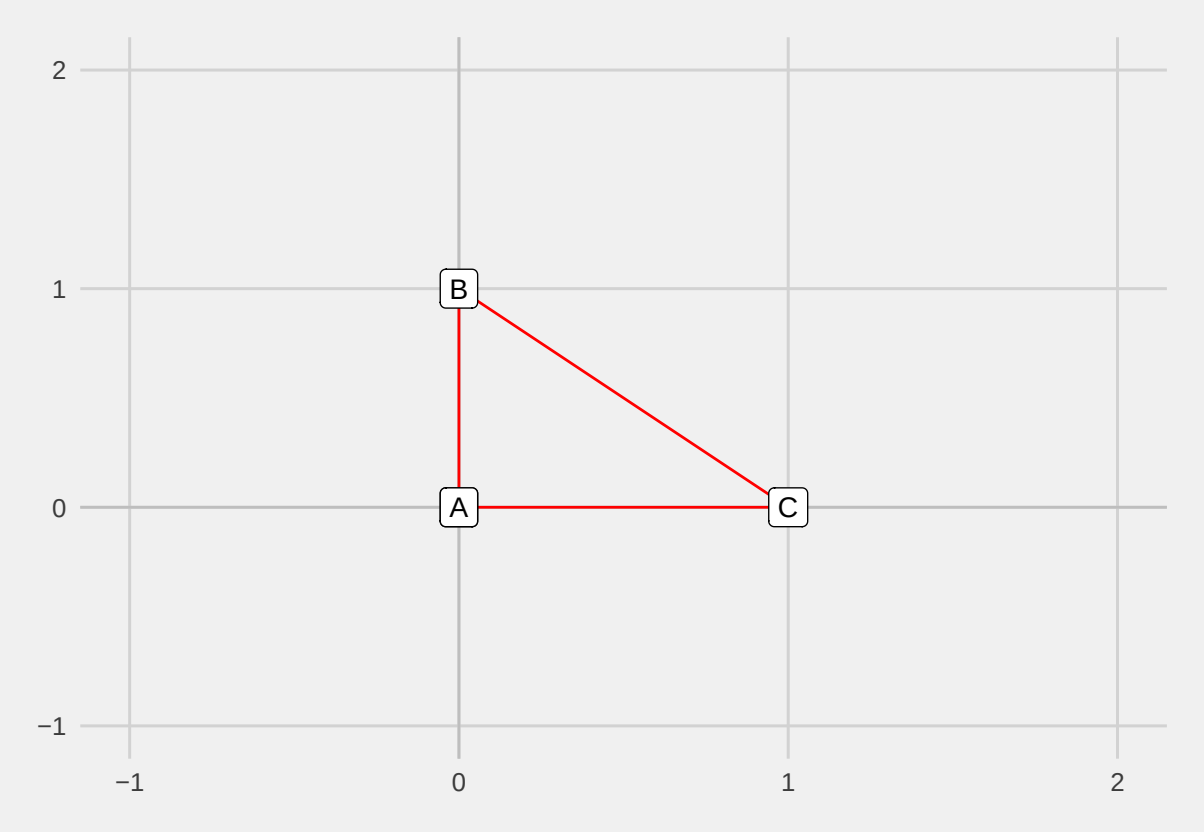

Figure 1: Demonstration of Geometric Application

$\vec{AC} = 1$

$\vec{AB} = 1$

First using figure 1 to demonstrate the intuition behind the geometric process. Trigonometry is used to obtain the line BC and the angle ABC. The vector is then rotated clockwise about B by this angle and subsequently scaled to the size of BA.

Taking A as the agent in this case and applying the geometric process defined above gives a coordinate result of:

x	y
1e-07	-0.0003982

This result is extremely close to the stated point. It is not exactly the same likely due to some internal rounding inconsistencies, but is close enough to be acceptable as a true predictor.

This process represents a re-framing of the relative coordinate system centred on the player to an absolute coordinate system centred on one of the two flags, which makes it a much easier problem to solve. The remaining problem however, is that the perception error in both distance measurements and angle measurements accumulate throughout the estimation.

Utilising the geometric prediction algorithm outlined above, predictions have been made for the testing data set (predicting for the training data set is not needed as only the testing data set is used for model comparison). The results of this prediction can be seen below.

X MSE	Y MSE	Total MSE
7530	2539	15060

This error is very high, despite the fact that this algorithm has been both mathematically and empirically shown to identify the agent's coordinates. This error then must be the result of the perception error that is proportional to the distance as discussed in 1.2. This is a problem as the geometric solution to coordinate prediction results in multiplying these terms that have some error component together a number of times, which creates an error term raised to the power of the number of multiplications. When noting that this is exacerbated by the proportionality to the distance, it is reasonable to obtain results such as above.

2 Methods

2.1 Applying a Neural Network to the same data as the Geometric Solution

The geometric solution to the problem of self-localisation has been obtained and is theoretically a perfect solution. This is not the case however, due to the perception error that is introduced by Robocup's simulation engine and as such the solution is as close as possible to the correct location. This means that this is the theoretical limit of accuracy for a neural network given the exact same input data. That is to say that given enough data, the function that the neural network begins to approximate, should be a good estimation of the geometric solution described above. If this is not the case and the neural network predicts the location of the agent better than the geometric solution, then there must be some form of systematic error being introduced by the perception error (as opposed to random error).

It is simple to apply the neural network to the same data as the geometric application, as inputs and outputs are of the same shape. There are some cases in which players can see less than two flags, and these cases are filtered as to not introduce any errors.

The network is structured with 8 nodes in the input layer (4 for each of the flags). The information used about the flags is the distance, angle, real x coordinate and real y coordinate. Including the real x and y coordinates of the flags allows the neural network to form relative associations between variables, rather than encoding every flag as a variable with 0 for information if it is not being included. There are 2 hidden layers 2 with 6 and 4 nodes respectively. The output layer consists of 2 nodes for an x coordinate and a y coordinate.

2.2 Applying a Neural Network to data with more inputs

The benefit of applying neural networks to this problem is that multiple inputs can be used to estimate the location of a player. This is helpful, as this can allow the perception error to be minimised, along with the possibility of unknown relationships in the data being identified and allowed for. It is for this reason that additional inputs will be added to predict for the same locations. To demonstrate this while remaining inside the scope of the the report and to observe the possible improvements, there will be 3 flags used instead of 2. This means that there will be 12 nodes in the input layer, then 8 and 4 nodes in each of the hidden layers.

An important note here is that in general, neural networks require a uniform input and shape to

work effectively. This means that the NaN values in the data can cause errors that break down the predictions provided by the neural network. This is an issue, as depending on the agent's position and orientation on the field, there are instances where very few flags are observed. It is for this reason that only the observations in the data in which the observed number of flags is greater than or equal to the required inputs for the neural network. In this case this only observations in which an agent sees at least 3 flags will be included.

2.3 Neural Networks with Other Types of Inputs

Neural networks are not limited to including only one kind of predictor variable in the modelling process. This is beneficial as different types of data may provide different kinds of information in making predictions. The point flags that have been used as predictors so far are not the only kind of landmark available, the exterior lines of the field that are in view are included in the landmark data set as well. This has the potential to be informative of the position and orientation of a player more broadly.

In order to include the lines in the neural network, an additional 6 variables need to be included. These variables include: the distance from the agent to the line; the angle made between the agent's direct orientation and the line; and four variables encoding which line it was (i.e. top, right, bottom, left). These line variables are one-hot-encoded meaning that only one of the variables is assigned a 1 indicating which line is in sight for each observation while the others have 0. This increase to 18 nodes in the input layer leads to the nodes in the hidden layers to increase to 12 and 6 respectively.

2.4 Neural Networks with Lagged Inputs

The position at which the agent was previously in would be an extremely good predictor for their current location theoretically, however there are a number of problems with including this. The initially obvious problem is that the actual location at the previous time increment is not available, only a prediction that is made using this same method. This would mean relying on predictions that have been made that recursively rely on other predictions made since the beginning of the game. If the actual previous location is presumed to be known perfectly, there is still an issue with including such a predictor, as if the neural network makes a prediction that is the same as the previous location, the loss function will be very low, however, this then

assumes that there is no movement which is not true. There is also the issue that no movement or velocity information is captured in this information, only the previous state of the agent.

A better way to include known prior (or lagged) information is to include the prior observations of all of the inputs. This means that there is no assumed perfect knowledge, as the observed distances and angles are always known to the agent. It also means that there is no predictor that would overpower the others in terms of weighting of relevant information. There is also the subtler encoding of both the direction and speed of movement through these lagged observations, through observing how much the distance and angle to each flag changes in each time step. This results in 30 nodes in the input layer, with 20 and 10 nodes in each of the hidden layers.

2.5 Ablation Study

Ablation studies in machine learning are commonly performed by removing some component of a constructed model in order to determine the resultant effect. They often involve removing key aspects of the model such as an entire class of inputs. In the case of the self localisation model, knowledge of the problem allows the recognition that localisation is not possible utilising only line data. This is because the known flags are essential for anchoring the agent to an absolute coordinate location on the field. This knowledge leads to the question of how the lagged neural network will perform if the line data is removed. This may have a negative effect as an entire class of information is being removed, however, it may also have a positive effect as some of the noisy input data is being removed. This has been implemented resulting in 24 input nodes, 12 nodes in the first hidden layer and 6 nodes in the second hidden layer.

2.5.1 Applying to More Inputs and Lagged Inputs

2.6 Hyper-parameter tuning

The parameters in a feed forward neural network such as the type that are being used in this project are the weights that are applied to the connections between nodes in each layer. These weights are trained through back-propagation in order to optimise a loss function. There are however a number of higher level parameters that cannot be trained directly in this way that are referred to as hyper parameters. Examples of hyper-parameters can include the learning rate, batch size, weight initialisation, batch size and number of epochs. Some of the most important hyper-parameters however, are the number of hidden layers and the number of nodes in each of

the hidden layers of the neural network. A lot of time can be spent tuning the hyper parameters and as this is not the purpose of this project, hyper-parameter tuning will only be performed on these two.

A traditional method of tuning hyper-parameters is the method of grid search, in which a list of groups of parameters are trialed and the group with the lowest loss function result are deemed to be the best. This is very computationally expensive and is flawed in that it gives equal weight to all parameters. A more efficient way described by (Bergstra and Bengio 2012) is to sample all hyper parameters from a range of possible values, which leads to double (in the 2 parameter case) the number of distinct values being assessed for each parameter. This means that if one parameter is extremely influential and another is not, computational resources are not wasted looking at the same value of the influential parameter multiple times. This is the method that will be used in this project.

This project tunes the hyper parameters of the number of nodes in each layer and the number of layers. For the purposes of sampling, it is equally valid to view these as three parameters, i.e. the total number of nodes, the number of layers and the proportion of nodes in each layer. This has the advantage of being able to specifically set a maximum and minimum number of total nodes. These nodes are then distributed based on a sampled proportion over a number of layers. It should be noted that in some cases this proportion is extremely close to zero and when working with integers such as number of nodes, could result in a layer with 0 nodes in it. This is a structural impossibility in a neural network and will throw an error if this is attempted to be put into code, as such in these situations, one node is placed inside these layers.

3 Results

3.1 Accuracy Tables

Model	X MSE	Y MSE	Total MSE
Geometric	7530	2539	10069
2 Flag NN	102	170	272
3 Flag NN	78	109	187
3 Flag + Line NN	114	140	254
Lagged Line NN	75	205	280
Lagged Input NN	53	74	127
Hyper Parameter Tuned NN	2	3	5

3.2 Visualisations

In the following visualisations, numbers will be used for brevity and visual simplicity. The numbers correspond to the models via the key:

Number	Model
1	Geometric
2	2 Input NN
3	3 Input NN
4	3 Input NN + Line
5	Lagged 3 Input + Line
6	Lagged 3 Input
7	Lagged 3 Input with Hyper-Parameter Tuning

Figures 2-5 show the RMSE of the predictions made divided up into regions and distributed over a representation of the soccer field. Figures 2-3 show the actual location with colour representing error, while figures 4-5 show the error in each of the predicted locations (note these locations may be well outside the bounds of the field). It should also be noted that the geometric model has been been separated from the neural networks in the following visualisations. If all of the models were grouped in the one visualisation, the extreme values in the geometric model (both in terms of error and predicted location) would distort the scale, leaving other models to appear

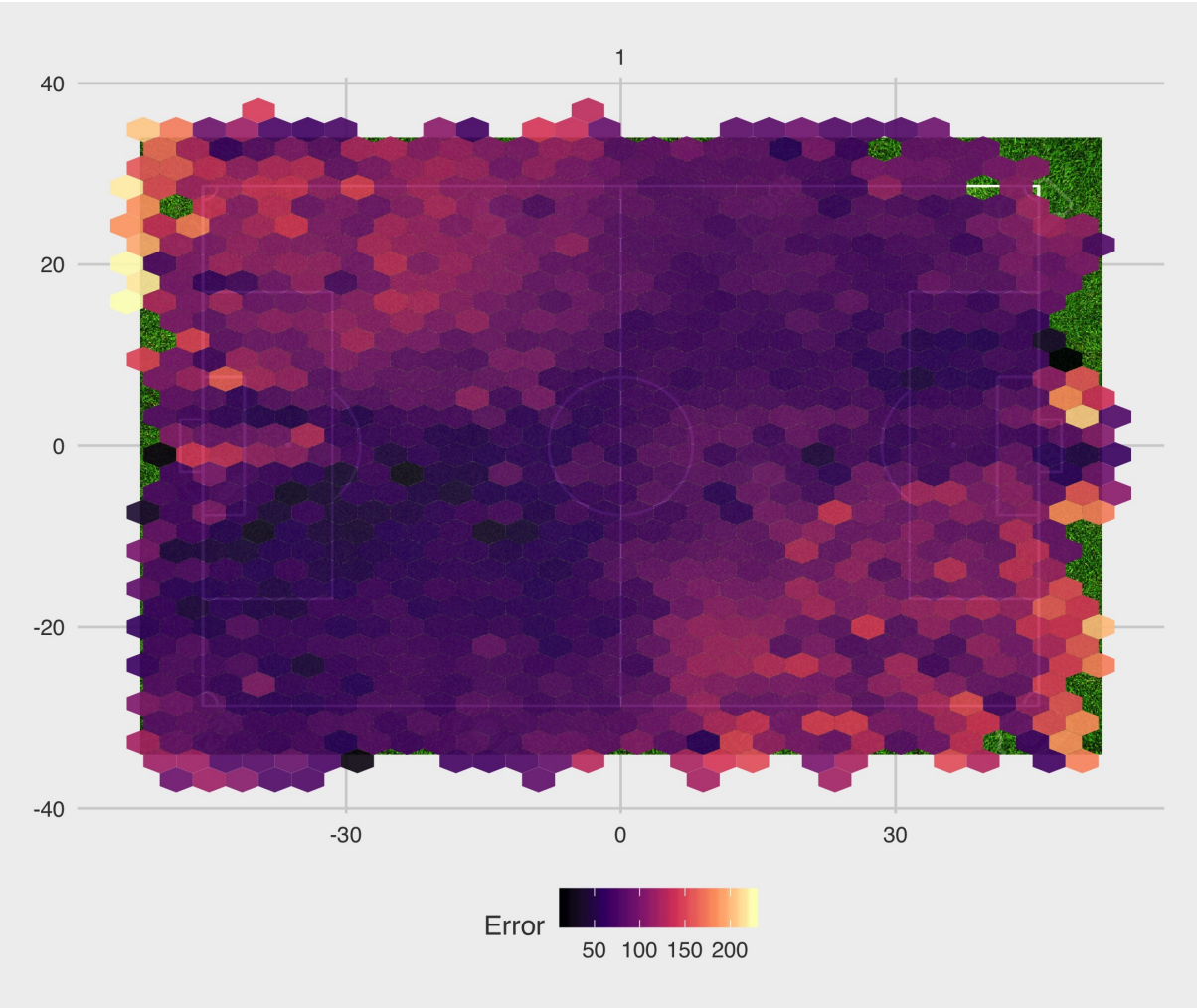

Figure 2: Location of Agents and Prediction Error for Geometric Model

deceivingly good. These visualisations have been constructed to aid the understanding of the distribution of predictions and where the predictions are going wrong.

4 Discussion

4.1 General Results

It can be seen that even with only 2 flags as inputs, the neural network performs much better than the geometric solution. This performance is built on by the neural network with 3 inputs bringing the down the MSE for both the X coordinate and Y coordinate. The addition of line variables increases the MSE, which is unexpected as including these variables includes additional information for prediction and represent a different kind of information. This is discussed in

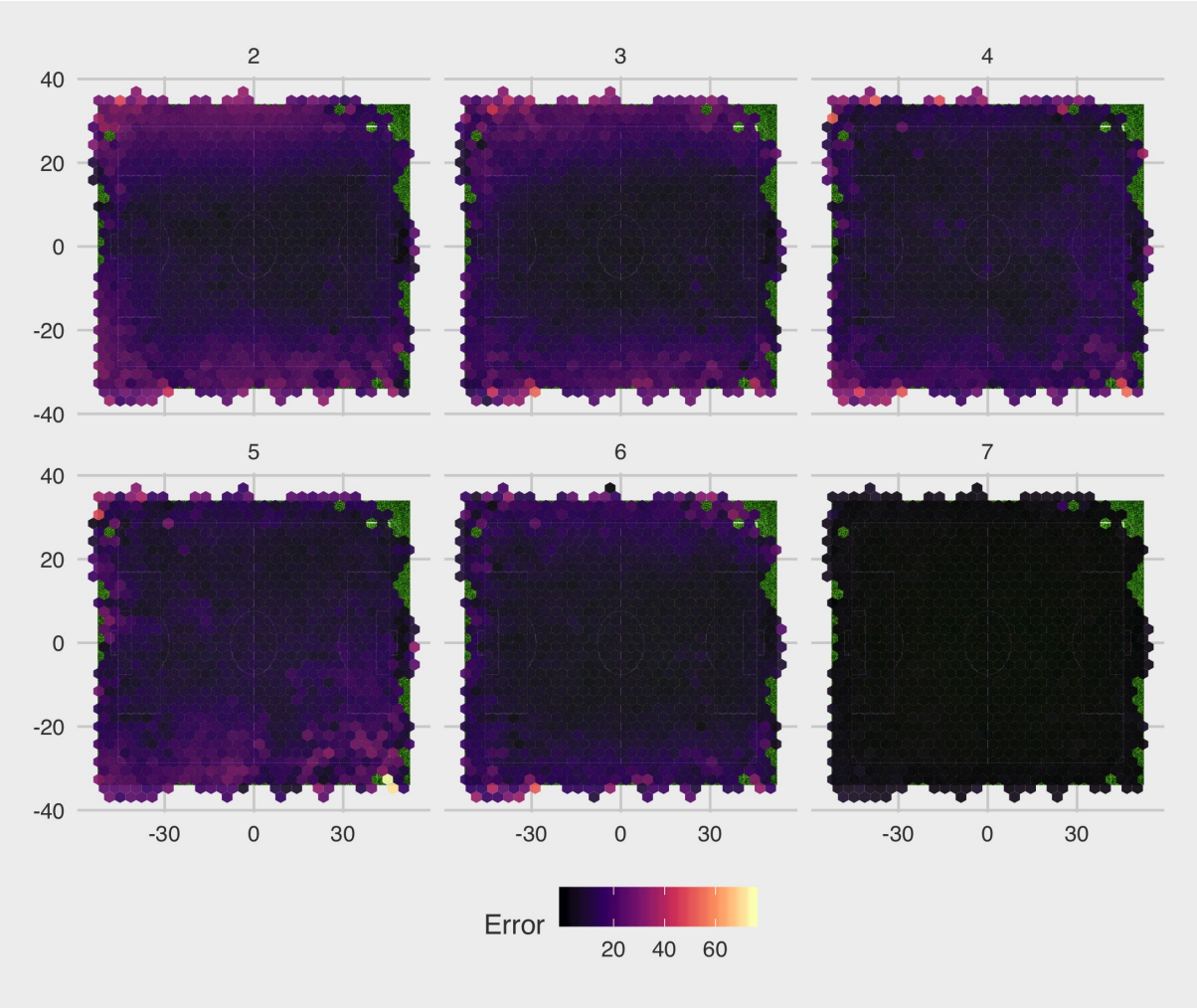

Figure 3: Location of Agents and Prediction Error

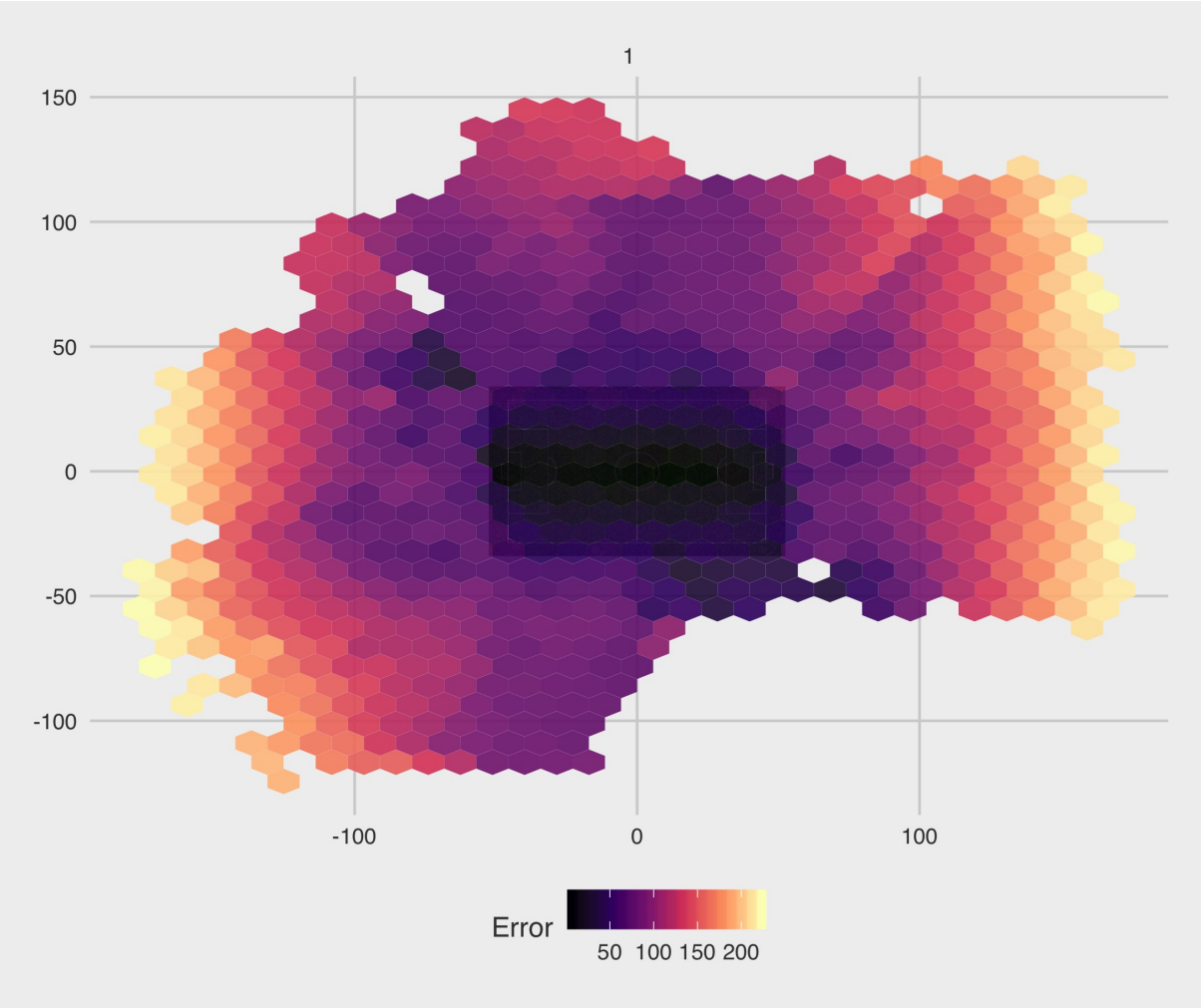

Figure 4: Predicted Location of the Agent and Prediction Error for Geometric Model

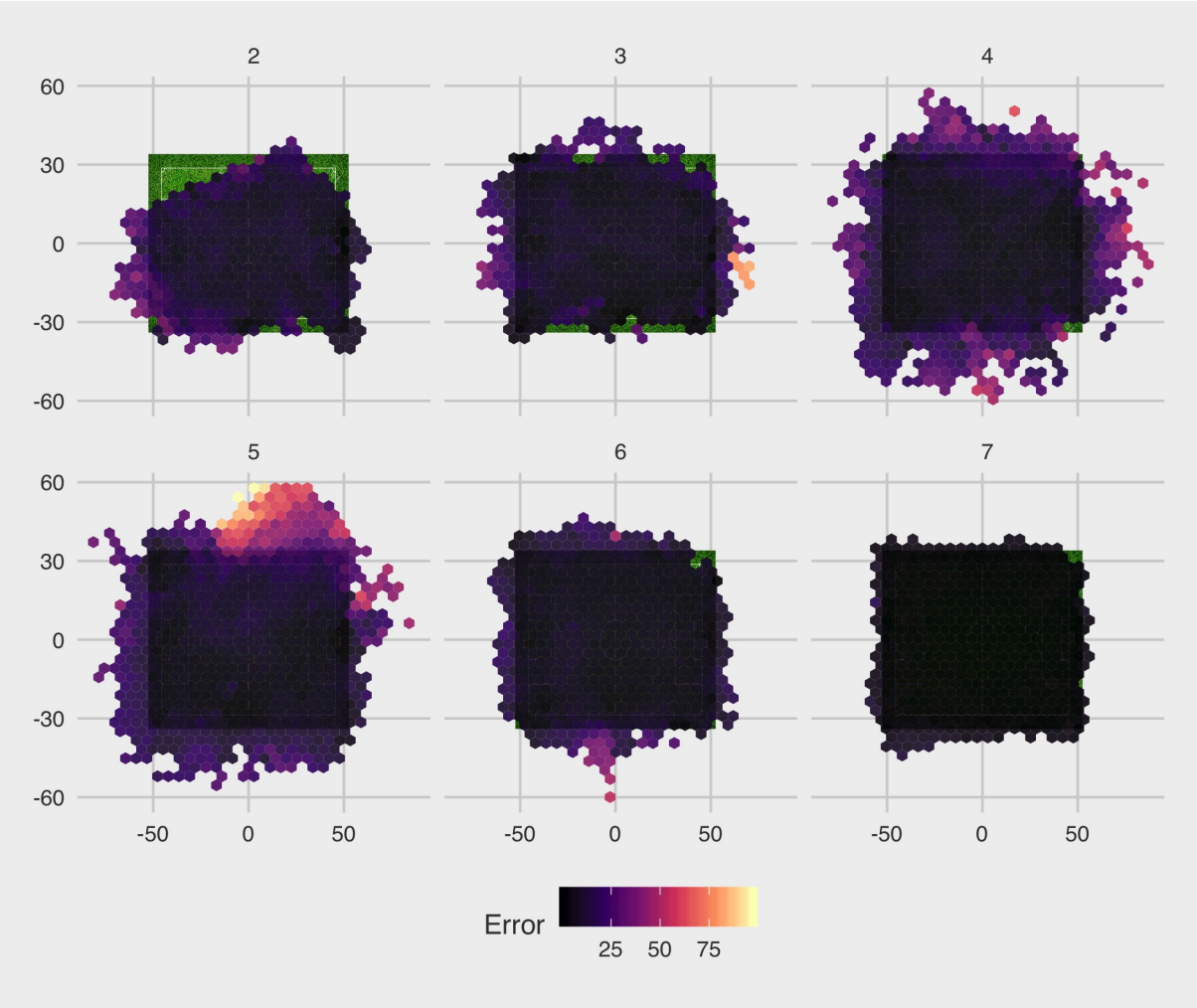

Figure 5: Predicted Location of the Agent and Prediction Error

more detail in section 2.5. The lagged neural network that included a line performs marginally better than the 3 input without any line variables in the X axis, but performs significantly worse in the Y axis. The ablation study model containing 3 flags as inputs along with the time lagged versions of these inputs performed the best.

It is interesting to note that in the geometric case, the predictions have a lower MSE in the Y direction, however, all of the neural network based approaches have a lower MSE in the X direction. This is particularly interesting when considered in the context of there being a much larger range of potential values in the X direction (-60 to 60) compared to the Y direction (-40 to 40). It would normally be expected that the predictions in the Y direction would be more performant than those in the X direction for this reason. One possible explanation for this discrepancy in expectation and the results is that the direction of play tends to happen in the X direction and agents tend to be facing the oppositions goal. This means that there would likely be a much larger sample in the data of agents being oriented in the X direction and these observations would contain within them more information about their X coordinate than their Y coordinate.

4.2 Discussion on Geometric Performance

The geometric solution to this problem is significantly worse than any of the neural network based approaches, even the neural network that shares the same input variables. The geometric approach should theoretically be the best possible model given these inputs assuming that perception error is only dependent on distance. There are two possible reasons for this to not be true. The first is the possibility of some form of systematic behaviour in the error term in perception error that is being learned by the neural network. This should be investigated, because if this is the case, allowances could be made for this in the geometric calculations. The second is that the neural network may have learned a function that involved significantly less multiplication of terms that contain error. As discussed in section 1.6 repeated multiplication leads to an error term that is proportional to the distance with a high degree. It is likely that this is not the method with the fewest instances of multiplication, or that there is a better way to implement this method that has been learned by the model.

4.3 Results by Position on Field

Figure 2 shows that worst predictions were made in the top left and bottom right field sections. This may be due to the orientation of the player in these areas, as if they are facing the opposite corner, then the predictions would be very bad due to extreme perception error. It should be noted that there are some extremely dark sections around each goal, that imply some predictions were made very well when looking at flags that are very close by.

The neural network prediction error observed in figure 3 seems to also be mostly focused around the outside of the field. Light strips can be observed on the top and bottom in models 2 through 4, with this effect being reduced in the lagged models 5 and 6 (this is likely where the improvement in lagged performance is coming from). These trends are not observable in model 7, as it appears at this scale to be highly accurate over the entire field.

Figure 4 shows the full range of predictions made by the geometric model and sheds light onto how it was possible for the accuracy to be so poor. The figure shows the that there is a lot more variation in the positions predicted in the X direction than in the y direction. It is interesting to note the region in which there is no predicted agent locations in the bottom right hand corner. The cause for this should be investigated in further work.

Figure 5 shows a considerably more accurate set of predictions made by the neural networks, although there are still many predicted locations that are out of the range. Model 2 appears to be a skewed version of the field while model 3 shows a corrected version with some added noise. Models 4 and 5 show an extremely spread out, scattered prediction set, that allows the effect of including the line to be seen clearly for the first time. The line creates much more uncertainty around the edges, causing some extreme predictions to be made. Model 6 and 7 are much more accurate, with model 7 even taking on an extremely similar shape to that of the true distribution of locations (seen in figures 2 and 3).

4.4 Ablation Discussion

The ablation study revealed that removing the variables related to line data decreased the accuracy of the resultant neural network. This means that this information is not helpful to the network when making predictions. It is possible that this is because the line distance and are not measured in the same way as with the flags; along with the fact that the lines do not provide any concrete information about coordinates (they convey information about a range of

them). The angles and distances from the agent to each of the lines is measured as the distance to the point of intersection and their line of sight, likewise the angle is the acute angle formed by the line of sight and the line that is being observed. For any given pair of measurements, there is a range of possible locations that an agent could then be located in. This is likely what is causing the accuracy to go down.

The model that has been generated through the ablation study is the best model that has been created. It is for this reason that hyper parameter tuning will be conducted on this model.

4.5 Hyper-Parameter Tuning Discussion

4.6 Applications

The neural network models have been trained to an extent at which the agent is able to predict their coordinate location with a high degree of accuracy. This means that there is an argument to put the model into production in an actual game. This would greatly improve on the geometric localisation that has been used previously, and allow for more advanced techniques to be implemented by the agents. As an example, the agents could perform complicated manoeuvres in which it is critical to know their exact location to receive a pass. This knowledge provides another advantage when shooting for goal, as knowledge as to exactly how far they are from the goal to adjust their kicking style or power applied. These advantages over other teams could be result in a higher probability of winning, especially when combined with additional optimisations mentioned in section 5.2. This needs to be integrated tightly with the rest of the programming of each agent in order to take full advantage of the benefits.

One trade off that is worth considering is the accuracy to speed ratio. This should be as high as possible, as it is not beneficial to the agent to have a perfect prediction of their position if they have moved on from that location in the time the prediction takes. Explicit timing has not been conducted however, it is likely that neural networks with a higher number of nodes will take longer to train. This should be investigated further before putting the neural network in to production, as the geometric method is very fast as it only requires a relatively small number of multiplications.

5 Conclusions and Future Work

5.1 Conclusions

The neural network models that have been trained show a much higher level of accuracy than that of the geometric model. The worst performing neural network that only included the 2 landmark inputs is much more accurate than the geometric model, however, would not be suitable in a production environment due to the error being relatively high. The final neural network that was created using hyper-parameter tuning would be very effective in a production environment, with a total MSE of 5 the agent's in question would have a good idea of their location in almost every increment of time.

5.2 Improvements

There are a number of ways that the work done in this project could be improved, for a production context. One such way is to create multiple different neural networks that can be used by agent's based on the number of flags that are within their field of view. In this project, only 3 flags have been in all cases, with situations in which the agent cannot see 3 flags filtered out. Utilising one neural network for each count of visible flags, the cases with fewer than 3 flags could be predicted for and the cases with more than 3 flags could be predicted for better. It is likely that this should be capped by number as there would reach a point of vanishingly small increases for each additional flag included. A cap should also be put on distance as the information returned from very distant flags is not worth including in the model.

It is likely possible to improve on the network that had tuned hyper-parameters through simulating these parameters more times and including additional kinds of hyper-parameters. This has not been conducted in this project as the computation time required is prohibitive, however, when creating a model that is destined for production, the training time is not as important. The improvements seen would likely be only marginal as the results are very good, however only 30 sets of hyper-parameters were tested which means it is very possible to optimise further.

There is the potential to rectify this problem by building into the program the tendency to look in the Y direction frequently to obtain Y coordinate information. This would generate data in which it is easier for the neural networks to learn the conditions for different coordinates in both

directions.

The data that the neural network is trained on has an inherent impact on the quality of predictions. It was noted in the discussion that the predictions in the X direction were more accurate than those in the Y direction, despite assumptions to the contrary. It was also noted that this may be caused by the fact that agents tend to look towards the opposing goal. This is a problem inherent to the programming of the agents and the data produced. A way to potentially improve the Y coordinate prediction is to either use more data to obtain more information about the Y coordinates or to intentionally construct situations under simulation to obtain this information.

5.3 Extensions

Extending on this project it would be interesting to assess neural networks with different architectures. Specifically, recurrent neural networks (RNNs) could be used, as they handle time delayed data well.

Neural networks are often treated as black box algorithms, as there are no direct coefficients that are applied to the input variables. It is therefore hard to assess the importance of each of the input variables in the generation of the final prediction that is being made. An extension on the work conducted on this project could involve assessing this importance information which could then be used to both understand how the neural network reaches its predictions, along with giving some insight into ways in which the model could be improved. It could also be used to illuminate any problematic bias that has been introduced through the training data. A number of approaches to solve this have been developed, however, the lime (Local Interpretable Model-agnostic Explanations) framework developed by (Ribeiro, Singh, and Guestrin 2016) is designed to examine the behaviour of any model, through making small changes to the input variables and observing the changes in predictions that are output. This is ideal as it is not dependent on the architecture of the neural network and can even be used on the geometric model to compare.

References

"A Brief History of Robocup." n.d. *RoboCup Federation Official Website*. https://www.robocup.org/a_brief_history_of_robocup.

Ashokaraj, Immanuel, Antonios Tsourdos, Peter Silson, and Brian White. 2004. "Sensor Based Robot Localisation and Navigation: Using Interval Analysis and Extended Kalman Filter." In *2004 5th Asian Control Conference (Ieee Cat. No. 04EX904)*, 2:1086–93. IEEE.

Bergstra, James, and Yoshua Bengio. 2012. "Random Search for Hyper-Parameter Optimization." *Journal of Machine Learning Research* 13 (Feb): 281–305.

Bottou, Léon. 2010. "Large-Scale Machine Learning with Stochastic Gradient Descent." In *Proceedings of Compstat'2010*, 177–86. Springer.

Chen, Mao, Klaus Dorer, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, et al. 2001. "Robocup Soccer Server." *Manual for Soccer Server Version* 7.

Michael, Olivia, Oliver Obst, Falk Schmidsberger, and Frieder Stolzenburg. 2017. "RoboCup-SimData: A Robocup Soccer Research Dataset," November.

Plaut, David C, and Geoffrey E Hinton. 1987. "Learning Sets of Filters Using Back-Propagation." *Computer Speech & Language* 2 (1). Elsevier: 35–61.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 1135–44. ACM.

Salloum, Ziad. 2019. "Back Propagation, the Easy Way (Part 1)." Edited by Towards Data Science. Medium. https://towardsdatascience.com/back-propagation-the-easy-way-part-1-6a8cde653f65.

Tu, Jack V. 1996. "Advantages and Disadvantages of Using Artificial Neural Networks Versus Logistic Regression for Predicting Medical Outcomes." *Journal of Clinical Epidemiology* 49 (11): 1225–31. https://doi.org/10.1016/s0895-4356(96)00002-9.