Understanding responses to environments for the Prisoner's Dilemma: A machine learning approach

Nikoleta E. Glynatsi

Month 2020

Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy.

School of Mathematics
Ysgol Mathemateg

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Mae-

cenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Dissemination of Work

Publications (4 Published & 5 In preparation)

1. 2018: **Reinforcement learning produces dominant strategies for the Iterated Prisoner's Dilemma.** Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Nikoleta E. Glynatsi, Owen Campbell - PLOS One - Preprint arXiv:1707.06307

2. 2018: **An evolutionary game theoretic model of rhino horn devaluation.** Nikoleta E. Glynatsi, Vincent Knight, Tamsin Lee. Ecological Modelling - Preprint arXiv:1712.07640

3. 2017: **Evolution reinforces cooperation with the emergence of self-recognition mechanisms: an empirical study of the Moran process for the Iterated Prisoner's dilemma.** Vincent Knight, Marc Harper, Nikoleta E. Glynatsi, Owen Campbell - PLOS ONE - Preprint arXiv:1707.06920

4. 2016: **An open framework for the reproducible study of the Iterated prisoner's dilemma.** Vincent Knight, Owen Campbell, Marc Harper et al - Journal of Open Research Software

In preparation

1. 2019: **A meta analysis of tournaments and an evaluation of performance in the Iterated Prisoner's Dilemma.** Nikoleta E. Glynatsi and Vincent A. Knight - In preparation to be submitted - Preprint arXiv:2001.05911

2. 2019: **A bibliometric study of research topics, collaboration and influence in the field of the Iterated Prisoner's Dilemma.** Nikoleta E. Glynatsi and Vincent A. Knight - In preparation to be submitted - Preprint arXiv:1911.06128

3. 2019: **Game Theory and Python: An educational tutorial to game theory and repeated games using Python.** Nikoleta E. Glynatsi and Vincent A. Knight - Submitted to the Journal of Open Source Education - Available on GitHub Nikoleta-v3/Game-Theory-and-Python

4. 2019: **A theory of mind: Best responses to memory-one strategies. The limitations of extortion and restricted memory.** Nikoleta E. Glynatsi and Vincent A. Knight - Submitted to Scientific Reports Nature - Preprint arXiv:1911.12112

5. 2019: **Recognising and evaluating the effectiveness of extortion in the Iterated Prisoner's Dilemma.** Vincent Knight, Marc Harper, Nikoleta E. Glynatsi, Jonathan Gillard - Submitted to Nature Communications - Preprint arXiv:1904.00973

Talks & Posters

Invited Talks (Keynotes)

- How does a smile make a difference?, PyCon UK, Cardiff, 2018.
- The Fallacy of Meritocracy, PyCon Balkan, Belgrade, 2019.

Others

- Accessing open research literature with Python - PyCon Namibia, Namibia 2017.
- Writing tests for research software - PyCon Namibia, Namibia 2017.
- Optimisation of short memory strategies in the Iterated Prisoners Dilemma - Wales Mathematics Colloquium, Gregynog Hall 2017.
- PIP INSTALL AXELROD (**poster**) - Euroscipy, Erlangen Germany 2017.
- Arcas: Using Python to access open research literature - Euroscipy, Erlangen Germany 2017.
- A trip to earth science with python as a companion - PyConUK, Cardiff 2017.
- The power of memory (**poster**) - SIAM UKIE Annual Meeting, Southampton 2018.
- Rhinos with a bit of Python - PyConNA, Namibia 2018.
- Memory size in the Prisoners Dilemma - Wales Mathematics Colloquium, Gregynog Hall 2018.
- Memory size in the Prisoners Dilemma - SIAM UKIE National Student Chapter, Bath University 2018.
- Stability of defection, optimisation of strategies and testing for extortion in the Prisoner's Dilemma (**poster**) - STEM for Britain, London 2019.
- Stability of defection, optimisation of strategies and testing for extortion in the Prisoner's Dilemma - 18th International Conference on Social Behaviour, Sedona, Arizona 2019.
- An introduction to Time Series - Joint workshop between CUBRIC & Mathematics Departments, Cardiff 2019.

Software Development

- Arcas, an open source package designed to help users collect academic articles' metadata from various prominent journals and pre print serves. **Contribution:** Main developer
- Axelrod-Python library, an open source framework decided to the study of the Iterated Prisoner's Dilemma. **Contributions:** Implementation of spatial tournaments functionality, implementation/addition of strategies (from the literature) to the library, reviewing of code contributed by other contributors
- SymPy, a Python library for symbolic mathematics. **Contributions:** Implementation of Dixon's and Macaulay's resultants which were developed for my 2019 publication "Sta-

bility of defection, optimisation of strategies and the limits of memory in the Prisoner's Dilemma"

- Pandas, an open source library providing high-performance, easy-to-use data structures and data analysis tools. **Contribution:** Fix bug which converted NaN values to strings

Contents

List of Figures

List of Tables

Chapter 1

Training long short-term memory networks to play the Iterated Prisoner's Dilemma

The research reported in this Chapter has been carried out with:

Axerod-Python library version: 4.2.0
Associated data set:

1.1 Introduction

In Chapter **??** it was mentioned that consepsualating and introducing new strategies has been an important aspect of research to the field. The aim of this Chapter is to introduce new IPD strategies based on an archetype that has not received much attention in the literature.

In Chapter **??** it was concluded that one of the properties successful strategies in a IPD competition need to have is cleverness/complexity. Complexity can confer to adaptability, and adaptability is important in performing well in diverse set of environments. This was established not only in Chapter **??** but also from the results of Chapter **??**. The set of complex strategies that ranked highly across distinct tournaments in Chapter **??** included strategies based on archetypes such as finite state automata, hidden Markov models and *artificial neural networks* (ANNs).

ANNs have successfully been trained to play games other than the IPD such as checkers [6], chess [10] and Go [24]. *Feed forward networks* were firstly used to represent IPD strategies in 1996 [13]. Feed forward networks have been used in the literature ever since [1, 2, 9, 11], and possibly the most successful ANN strategies in the game are the ones introduced in [12]. The three ANN based strategies in [12] ranked 7^{th}, 9^{th}, and 11^{th} in a tournament of 223 strategies.

A type of ANNs that have not received much attention in the literature are the *recurrent neural networks* (RNNs). RNNs are a type of neural networks that include a feedback connection and are designed to work with inputs in the form of sequences. RNNs were firstly considered as an archetype in 1996. In [23] a RNN which considered a single previous step as an input was trained via a Q learning algorithm to play against a single opponent. The opponent was either the strategy Tit For Tat or another Q learning strategy. The results of [23] although were somewhat promising, the RNN player learned to optimally play against Tit For Tat, were limited.

The limitations of [23] could potentially have been due to the limitations of the RNNs themselves. As it will be discussed later in section 1.2, RNNs quickly became unable to learn due to the vanishing gradient problem. To improve on the standard recurrent networks a new model called the *long short-term memory network* (LSTM) was introduced in [14]. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

LSTMs are a set of networks that have been proven to be successfully trained and today are being used in a number of innovative applications such as time series analysis [17], speech recognition [22] and prediction in medical care pathways [26]. However, they have not received attention in the IPD literature. The aim of this Chapter is to trained and introduce a number of strategies based on LSTMs. The training has been possible due to the collection of best response sequences generated in Chapter **??**.

A total of 24 LSTMs based strategies are introduced in this Chapter, and their performance is evaluated and compared in a meta tournament analysis of 300 standard tournaments. The results demonstrate that LSTM networks can be trained to successfully compete in IPD tournaments. The rest of the Chapter is structured as follows:

- section 1.2, presents an introduction to artificial, recurrent and long short-term memory networks.
- section 1.3, covers the specifics of the LSTMs trained in this Chapter. It presents their architectural details as well as the different training used to train them.
- section 1.4, evaluates and compares the performance of the 24 LSTMs based strategies in 300 standard IPD computer tournaments.

1.2 Artificial, recurrent neural and long short-term memory networks

ANNs are computing systems vaguely inspired by the biological neural networks that constitute animal brains. As stated in [15] the research on ANNs has experienced three periods of extensive activity. The first peak was in the 1940s. The work of [19] opened the subject by creating a computational model for neural networks. The second peak occurred in 1960s with the introduction of the perceptron [21]. The perceptron is a linear binary classifier and in the 1960s in was shown that it could be used to learn to classify simple shapes with 20×20 pixels input. However, it was impossible for the perceptron to be extended to classification tasks with many categories. The limitations of the model were presented in [20] which halted the

enthusiasm of most researchers in ANNs, resulting in no activity in field for almost 20 years. The third peak occurred in the early 1980s with the introduction of the back propagation algorithm for multilayered networks [27]. Though the algorithm was initially proposed by [27] it has been reinvented several times and it was popularizes by [18]. Following the introduction of the algorithm and the ability to now train more complex models ANNs have received considerable interest and are being used in a number of innovative applications [8, 16].

ANNs based on the connection pattern can be grouped into two categories:

- Feed forward networks.
- Recurrent or feedback networks.

ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges, with weights, are connections between the neuron outputs and the neuron inputs [15]. In feed forward networks the graphs have no loops whereas in recurrent loops occur because of feedback connections. A graphical representation of a feed forward network with a single hidden layer is given by Figure 1.1.

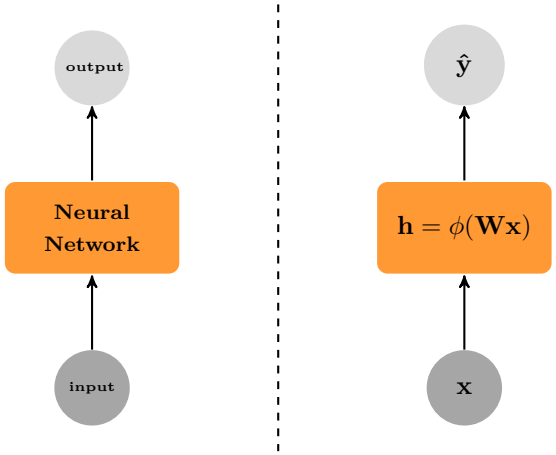


Figure 1.1: Graphical representation of an ANN.

A feed forward network is composed by an input layer, hidden layers, and an output layer. The networks' input is a vector x. The dimensionality, or number of nodes of the input layer is depended on the dimensionality of x. Each element of the input vector is connected to the hidden layer via a set of learned weights. The hidden layer also consists of nodes. The number of nodes of the hidden layer is an architectural decision. The j^{th} hidden node outputs,

$$h_j = \phi(\sum_i w_{ij} x_i), \text{where } \phi \text{ is an activation function.}$$

Activation functions ϕ include the:

- Sigmoid function $\sigma(x)$ which squashes numbers into the range $[0, 1]$.
- Hyperbolic tangent, $tanh(x)$ which squashes numbers into the range $[-1, 1]$.
- Rectified linear unit, $ReLU(x) = max(0, x)$.

In turn, the hidden layer is fully connected to an output layer. The j^{th} output node outputs,

$$y_j = \sum_i v_{ij} h_i.$$

Feed forward networks make predictions using forward propagation which in matrix notation can be described by,

$$h = \phi(Wx) \tag{1.1}$$

$$\hat{y} = \text{softmax}(Vh) \tag{1.2}$$

where W is a weight matrix connecting the input and hidden layers, V is a weight matrix connecting the hidden and output layers. The output layer can transform the raw scores to a probability via a softmax function.

W and V are the learning parameters of the network. Their dimensionality depends on the dimensionality of network's layers. For example if x and $\hat{y}$ were 2 dimensional and the hidden layer had 500 nodes then $W \in \mathbb{R}^{2\times 500}$ and $V \in \mathbb{R}^{2\times 500}$, thus increasing the dimensionality of the hidden layer corresponds to more parameters.

To train a ANN a set of values for the learning parameters need to be found so that the error on the training data is minimised. The function that measures error is called the *loss function*. A loss function commonly used is the *cross entropy* denoted as $L(y, \hat{y})$ where $\hat{y}$ is the prediction and y the true output. For M training examples and K classes the cross entropy error is given by Equation (1.3).

$$L(y, \hat{y}) = -\frac{1}{M} \sum_{m \in M} \sum_{k \in K} y_{m,k} \log(\hat{y_{m,k}}) \tag{1.3}$$

The minimum of the loss function is calculated using the gradients decent algorithm. The gradient decent algorithm relies on the gradients. The gradients are the vector of derivatives of the loss function with respect to the learning parameters, thus $\frac{\partial L}{\partial W}$ and $\frac{\partial L}{\partial V}$. These gradients are calculated with the back propagation algorithm [28] which is a way to efficiently calculate the gradients starting from the output.

An extension of feed forward networks are the recurrent networks. RNNs are capable of processing variable length sequences of inputs. Sequences are data samples over a number of time steps. RNNs can receive sequence inputs and output sequences of the same length, using their internal state/knowledge to a make a prediction on the input at time t based on the decisions made at the previous time steps. A graphical representation of a RNN is given by Figure 1.2 where the loop represents that information can be passed down from one step of the network to the next. In fact they can be thought of as multiple copies of the same network each passing a message to a successor, as demonstrated by Figure 1.2.

Information is passed down by considering that,

$$h_t = \phi(Wx_t + Uh_{t-1})$$

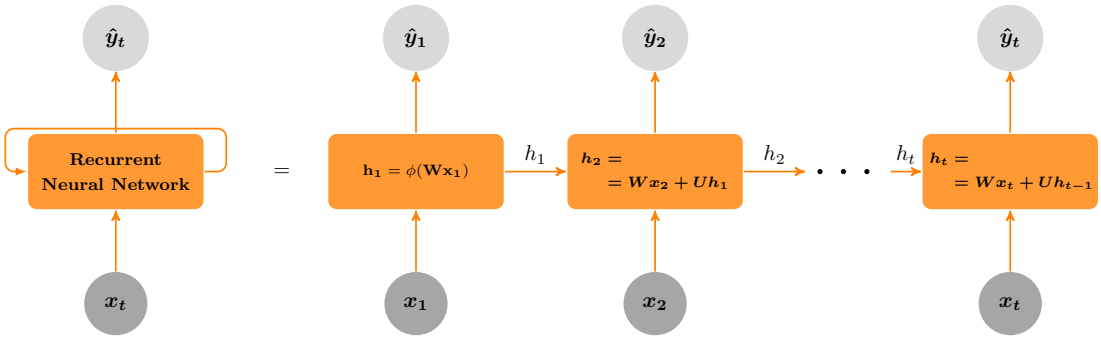


Figure 1.2: Graphical representation of a RNN.

where h_{t-1} is the hidden state computed at time $t-1$ multiplied by some weight vector U. In matrix notation RNNs are described by,

$$h_t = \phi(Wx_t + Uh_{t-1}) \tag{1.4}$$

$$\hat{y}_t = \text{softmax}(Vh). \tag{1.5}$$

Unfortunately, in practice RNNs quickly become unable to learn to connect the information. The more time steps the higher the chance that the back propagation gradient either accumulate and explode or vanish down to zero. The fundamental difficulties of RNNs were explored in depth by [5]. A network specifically designed to avoid the long-term dependency problem. was introduced by [14] called the long short-term memory network.

The core idea to LSTMs is the *cell state* also refereed to as the long term memory, denote as C_t. The cell state is designed to pass down information with only a few carefully regulated changes being applied to it by structures called *gates*. Gates are composed out of a sigmoid neural net layer and a point wise multiplication operator. In order to explain the cell gate and subsequently how LSTMs make predictions consider a LSTM's hidden layer at time step t given by Figure 1.3.

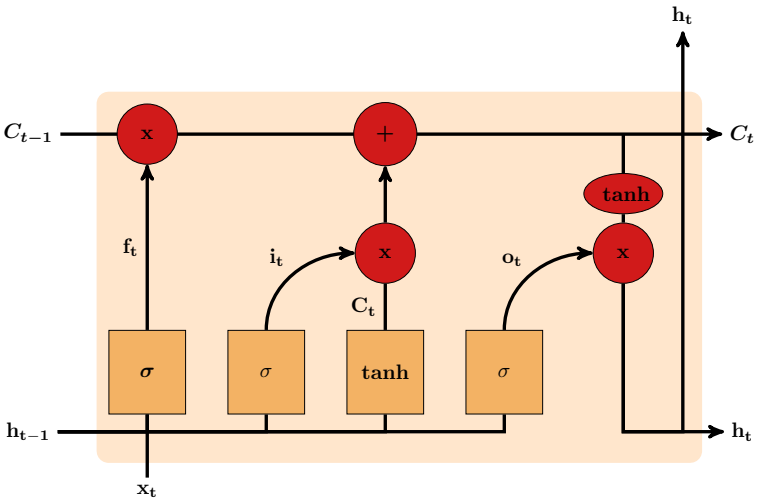


Figure 1.3: An LSTM hidden layer at time step t.

The cell state and hidden state from from the previous time step are feed back into the network. Initially, the network decides what information from the previous cell state is to be discarded. This decision is made by the *forget state*. The forget state considers the hidden state at time $t-1$ and the input at time t,

$$f_t = \sigma(W_f x_t + U_f h_{t-1}). \tag{1.6}$$

Secondly, the network decides what information is going to be stored at the cell gate. There are two parts to this. Firstly, the input gate decides which values are going to be updated,

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i). \tag{1.7}$$

and secondly, a tanh layer creates a vector of candidate values denoted as $\tilde{C}_t$ that could be added to the cell state. $\tilde{C}_t$ is given by Equation (1.8).

$$\tilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c). \tag{1.8}$$

The cell state C_{t-1} is multiplied by f_t, forgetting the values which have been decided to discard. The new candidate values are scaled by how much information has been decided to keep from the input. Thus,

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{1.9}$$

The LSTM outputs a hidden state at each time step which is based on the current cell state. Initially, the cell state goes through a tanh function and then it is multiplied by a sigmoid gate that decides which parts to output from the cell state,

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{1.10}$$

$$h_t = o_t * tanh(C_t). \tag{1.11}$$

This process is being carried out for each time step of the input sequence. At each time step both the cell state and hidden state are feed back into the network. The hidden state can also be used to make a prediction at each time step as demonstrated by Figure 1.4.

LSTM unique architecture allow them to be trained with the back propagation algorithm and they have managed to achieve remarkable results. The training of an LSTM model in this Chapter is carried with the open source project Keras. Details of the training process are presented in section 1.3.

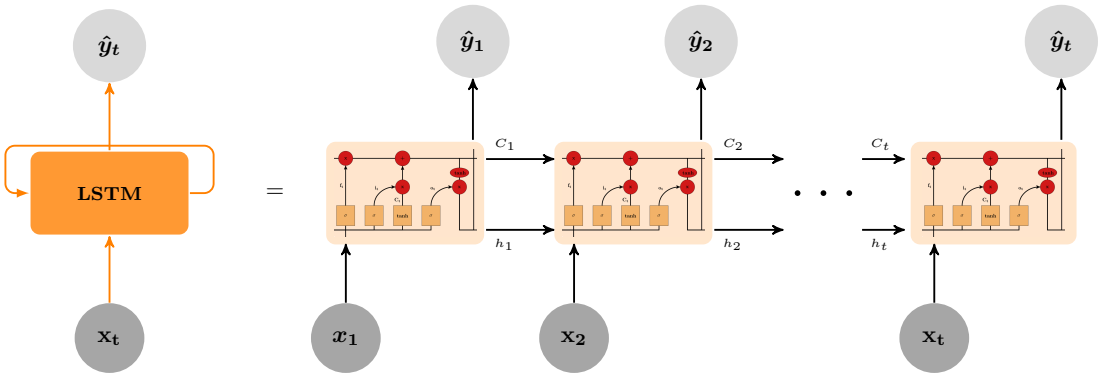


Figure 1.4: Graphical representation of an LSTM network.

1.3 Training LSTM networks to play the Iterated Prisoner's Dilemma

LSTM are trained in a supervised fashion on a set of training sequences. The purpose of training a network in this Chapter is so it can learn to play optimally against IPD strategies. For that reason the networks are going to be trained on the collection of best response sequences generated in Chapter **??**.

The training inputs are the actions of a given strategy for N turns. The expected outputs are the responses to those N action by the opponent's best response sequence. Each pair of opponent's - best response sequence's actions, from the the collection of best response sequences, are transformed into 204 unique training samples. The highest dimensionality of those samples is 204.

Consider the actions of the strategy Adaptive and its best response, as presented in section **??**, given by Table 1.1.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	...	**204**	**205**
Adaptive	1	1	1	1	1	1	1	0	0	0	0	1	...	1	1
S^*	1	1	0	0	0	0	0	0	0	0	0	0	...	0	0

Table 1.1: The actions of the strategy Adaptive against one of the best response sequences to the strategy. Note that $0 \to D$ and $1 \to C$.

Initially the highest dimensionality a training sample based on Table 1.1 can have is 204. This is because the expected output to Adaptive's action in turn 1 is the action of the best response sequence at turn 2. Moreover, the expected output to the Adaptive's 204^{th} action is the best response's 205^{th} action. This is demonstrated by Figure 1.5.

Secondly, in order to train the networks on different input lengths the training sample of Figure 1.5 is transformed to 204 samples. This is done by considering all the possible IPD matches between the pairs where the number of turns $N \in [1, 204]$. For example the training

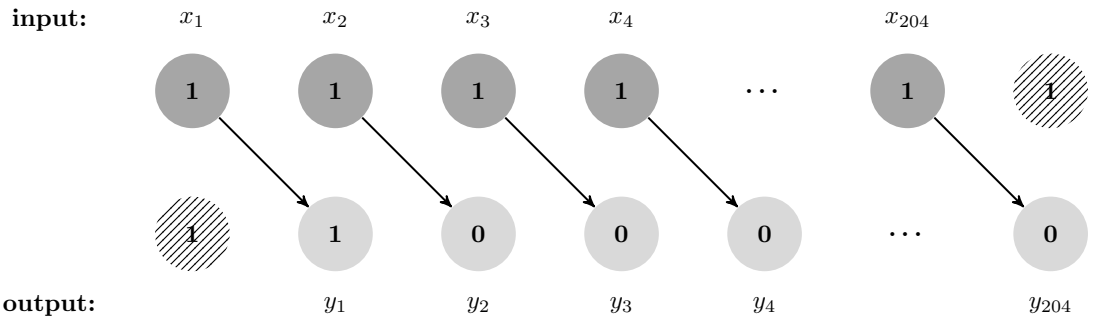


Figure 1.5: An example of a networks input and output of $t = 204$. The last action of Adaptive as well as the first action of the best response sequence are discarded.

samples for Table 1.1 are given by,

$$\text{inputs} = \begin{bmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ 1 & 1 & 1 & & & & \\ 1 & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad \text{expected outputs} = \begin{bmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ 1 & 1 & 0 & & & & \\ 1 & 1 & 0 & 0 & & & \\ 1 & 1 & 0 & 0 & 0 & & \\ 1 & 1 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{1.12}$$

Subsequently, the training data set retrieved from the collection of best responses has a total of $5258 \times 204 = 1122612$ training samples.

Two types of LSTMs have been trained in this Chapter. These are refereed to:

- Sequence to sequence (StoS).
- Sequence to probability (StoP).

Both models take as input a sequence of actions. The StoS network outputs a response to each time step of the input sequence. The StoP network only outputs a response to the sequence at the last time step. Both networks output a probability which is the probability of cooperating given an opponent's history. A graphical representation of the two networks are given by Figures 1.6 and 1.7.

The training samples for the StoS network are in the form of Equation (1.12) whereas the

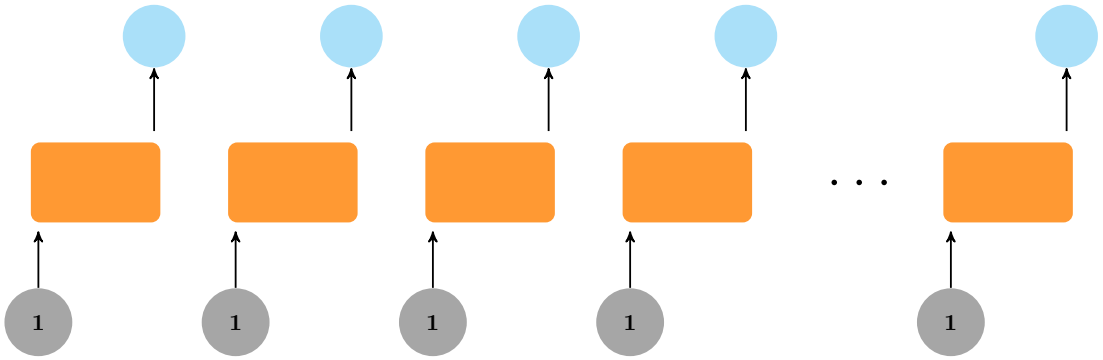


Figure 1.6: A graphical representation of the StoS LSTM model.

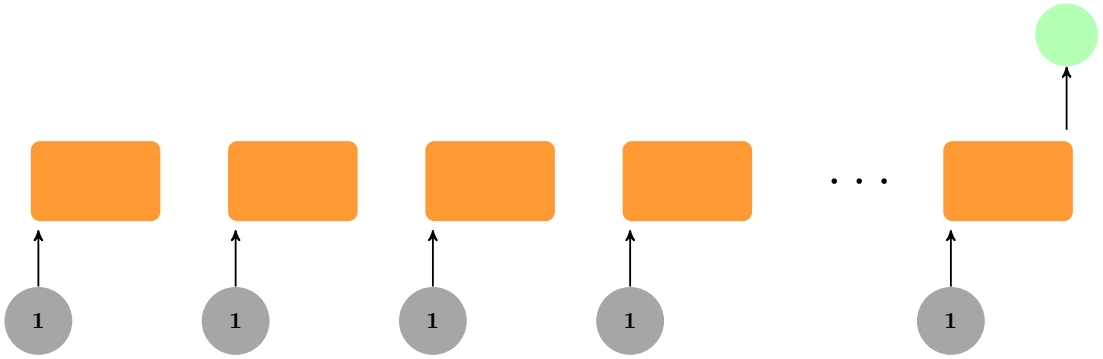


Figure 1.7: A graphical representation of the StoP LSTM model.

training training samples for the StoP network are in the form of Equation (1.13).

$$\text{inputs} = \begin{bmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ 1 & 1 & 1 & & & & \\ 1 & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad \text{expected output} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{1.13}$$

1.3.1 Building the networks with Keras

The open source neural-network package Keras [7] is used to construct and train the networks. Keras is a library written in Python designed to enable fast experimentation with neural networks. Neural layers, cost functions, optimizers and activation functions are all standalone modules that can be combined to create new network models.

The Python code for implementing the StoS model is given by Figure 1.8. In line 12 the model is defined to be of the `Sequential` class. This mean that the model will constructed layer by

layer. The StoS network has a single LSTM layer with 100 nodes. The input to the LSTM network is not a fixed length and there a single time step between the elements of each input sequence. This is defined by the argument `input_shape=(None, 1)`. The LSTM layer outputs a hidden state at each time step. Initially, the hidden states go through a dropout layer. The dropout layer is a simple and yet efficient method to reduce overfitting by randomly dropping out nodes of the hidden states [3]. Finally, the hidden states are transformed into probabilities via a sigmoid layer. There are a total of 41301 learning parameters to the StoS model.

```
>>> from keras.models import Sequential

>>> from keras.layers import (
...     Dense,
...     Dropout,
...     CuDNNLSTM,
... )

>>> num_hidden_cells = 100
>>> drop_out_rate = 0.2

>>> model = Sequential()

>>> model.add(
...    CuDNNLSTM(
...        num_hidden_cells, return_sequences=True, input_shape=(None, 1))
...)

>>> model.add(Dropout(rate=drop_out_rate))

>>> model.add(Dense(1, activation="sigmoid"))
>>> model.summary()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_1 (CuDNNLSTM)     (None, None, 100)         41200
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 100)         0
_________________________________________________________________
dense_1 (Dense)              (None, None, 1)           101
=================================================================
Total params: 41,301
Trainable params: 41,301
Non-trainable params: 0
_________________________________________________________________
```

Figure 1.8: Python code for implementing the StoS LSTM with Keras.

Regarding the dimensionality of the hidden layer there is no direct answer as to what is the optimal number of nodes. Several methods for determining the dimensionality include experimentation, intuition and building on the work of other. A common practice is that the dimensionality of the hidden layer is smaller than the dimensionality of the input layer. The dimensionality of the input layer changes from 1 to 204, and thus a number of 100 nodes was chosen so that the dimensionality of the hidden layer is smaller 50% of the times.

The Python code for implementing the StoP network with Keras is given by Figure 1.9. The implementations of the networks are similar, however, the StoP model contains 2 LSTM layers. The first LSTM layer outputs the hidden states at each time step. In turn these are connected to the second layer which only output the hidden state at final time step. The StoP network has a higher number of learning parameters due to the two LSTM layers. More specifically,

there are 122101 learning parameters to the network.

```
    >>> from keras.models import Sequential
>>> from keras.layers import (
...     Dense,
...     Dropout,
...     CuDNNLSTM,
... )

>>> num_hidden_cells = 100
>>> drop_out_rate = 0.2

>>> model = Sequential()

>>> model.add(
...     CuDNNLSTM(num_hidden_cells, return_sequences=True, input_shape=(None, 1))
... )

>>> model.add(CuDNNLSTM(num_hidden_cells))
>>> model.add(Dropout(rate=drop_out_rate))

>>> model.add((Dense(1, activation="sigmoid")))
>>> model.summary()
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_2 (CuDNNLSTM)     (None, None, 100)         41200
_________________________________________________________________
cu_dnnlstm_3 (CuDNNLSTM)     (None, 100)               80800
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 122,101
Trainable params: 122,101
Non-trainable params: 0
_________________________________________________________________
```

Figure 1.9: Python code for implementing the StoP LSTM with Keras.

Keras includes two implementations for an LSTM layer. The class `LSTM` and the class `CCNLSTM` which is the class used here. `CCNLSTM` provides a faster implementation of `LSTM` with the NVIDIA CUDA Deep Neural Network library. The use of the `CCNLSTM` class means that the networks can be trained on a graphics processing unit (GPU).

1.3.2 Training on GPU

Conventionally the execution of computer code happens on the central processing unit (CPU). The CPU, also called main processor, is essentially the brain of any computing device. Architecturally the CPU is composed of just a few cores designed to support an extremely broad variety of tasks.

A graphical processing unit (GPU), on the other hand, is composed of hundred of cores designed to process a set of simpler and more identical computations in parallel. GPUs were initially designed as dedicated graphical rendering workhorses of computer games but were later enhanced to accelerate other geometric calculations. NVIDIA created a parallel computing architecture and platform for its GPUs called CUDA, which gave developers access and the ability to express simple processing operations in parallel through code.

A CPU core is more powerful than a GPU core, however, the vast majority of this power goes unused by machine learning applications. A GPU core is optimised exclusively for data computations and because of this singular focus a GPU core is simpler and has a smaller die area than a CPU, allowing many more GPU cores to be crammed onto a single chip. Consequently, machine learning applications which perform large numbers of computations on a vast amount of data can see huge performance improvements when running on a GPU versus a CPU.

Due to the time advantage the training process of the networks was carried out on a GPU. The training was performed on the High-Performance Computing (HPC) cluster of Cardiff University.

1.3.3 Training data sets

Section 1.3 covered the training data used to train the LSTM networks. There are a total 1122612 training inputs and expected outputs to the training data set. In order to understand the effect of the training data on the the LSTM strategies' performance the networks are trained on the entire data set and on three unique subsets.

The subsets are based on three collection of opponents. These are a collection of best performing strategies, of strategies with ranks across a standard tournament and a collection of basic strategies. The details of the subsets are given by Table 1.2. A list of the strategies' names using in each subset is given in the Appendix.

Data set	# of opponents	Explanation	# of best response sequences
all strategies	192	The data set as presented in section 1.3.	5258
top performing strategies	18	A data set constructed in the same way as the training data set but only with the best response sequences to 18 strategies. These are the top performing strategies in a standard tournaments of 218 opponents.	714
strategies across ranks	15	A data set generated only with the best response sequences to 15 strategies whose ranks are across the 218 ranks of the standard tournament.	212
basic strategies	11	A data set generated with the best response sequences to 11 strategies which are classified as a set of basic strategies in the APL.	84

Table 1.2: Training data sets used to training the LSTM networks. The IPD standard tournament with the 218 opponent has been carried out using APL version 4.5.0. The results are available online: `https://github.com/Axelrod-Python/tournament`.

1.3.4 Training and validation

Two different LSTM networks were trained on four different training data sets. Thus a total of $4 \times 2 = 8$ networks have been trained in this Chapter.

Due to the different size of the training data the networks have been trained for a different number of epochs. The number of epochs is the number of times the training algorithm has work through an entire training data set. The number of epochs for each of the 8 network is given by Table 1.3.

The StoP networks have a larger number of learning parameters. Subsequently, their training corresponds to more computer time. It can been seen from Table 1.3 that the StoP networks have been trained for less epochs compared to the StoS networks.

At each training process the training data set is split into 80% training samples and 20% test

	all strategies	top strategies	strategies across ranks	basic strategies
sequence to sequence	19999	83999	39999	129999
sequence to probability	7999	74999	39999	84999

Table 1.3: Number of epochs for each of the trained networks.

samples. At each epoch the loss function is calculated for both the training and test samples. The loss function used to trained the networks is the *binary cross entropy*. In essence, the task of predicting IPD actions is a binary classification problem as there can only be two classes.

As discussed in section 1.2, the loss function is used to optimise the learning algorithm. The networks that will be used as IPD strategies in the next section are the networks that achieved the lowest loss value over the epochs. The weights of the best performing networks, based on the loss function, have been archive and are available at.

Another measure which is being reported at each epoch is the accuracy. The accuracy is calculated after the learning parameters have been determined and it is in the form of a percentage. Accuracy is a measure of how accurate the predictions are based on the expected output. Both the measures over the number of epochs are given by Figures 1.10 and 1.11.

In Figure 1.10 the loss and accuracy are given for the StoS networks. Overall the StoS networks, regarding the training data set, have maintained a high value of accuracy over the epochs. Whilst training on the basic strategies there was a minor decrease, however, the accuracy still remained over 0.85 (85%). The loss function values have also remained low over the epochs for all the networks with only a few spikes occurring.

For the StoP networks there appears to be more variation in the test loss and accuracy, Figure 1.11. This is more evident in for the networks that were trained on the top strategies and the strategies across ranks. This could indicate that the networks are overfitting. The StoP networks that was trained on the entire training data set has managed to maintained a low loss value (at 0.5) and a training accuracy of 0.75. The most successfully trained StoP networks appears to be the network trained on the basic strategies.

This section has presented the 8 trained LSTM networks of this thesis. These networks are based on two different architectures and have been trained on four different training data sets. The networks' validation based on the training data sets was presented in this section. In the next section the networks are evaluated as IPD strategies.

1.4 Validation of LSTM base strategies using a meta tournament analysis

This section evaluates the trained LSTM networks as IPD strategies.

A strategy class called the `LSTMPlayer` was implemented in order for the networks, which are Keras models, to interact in an IPD match simulated with APL. The source code for the `LSTMPlayer` is given by Figure 1.12. The class has three input arguments:

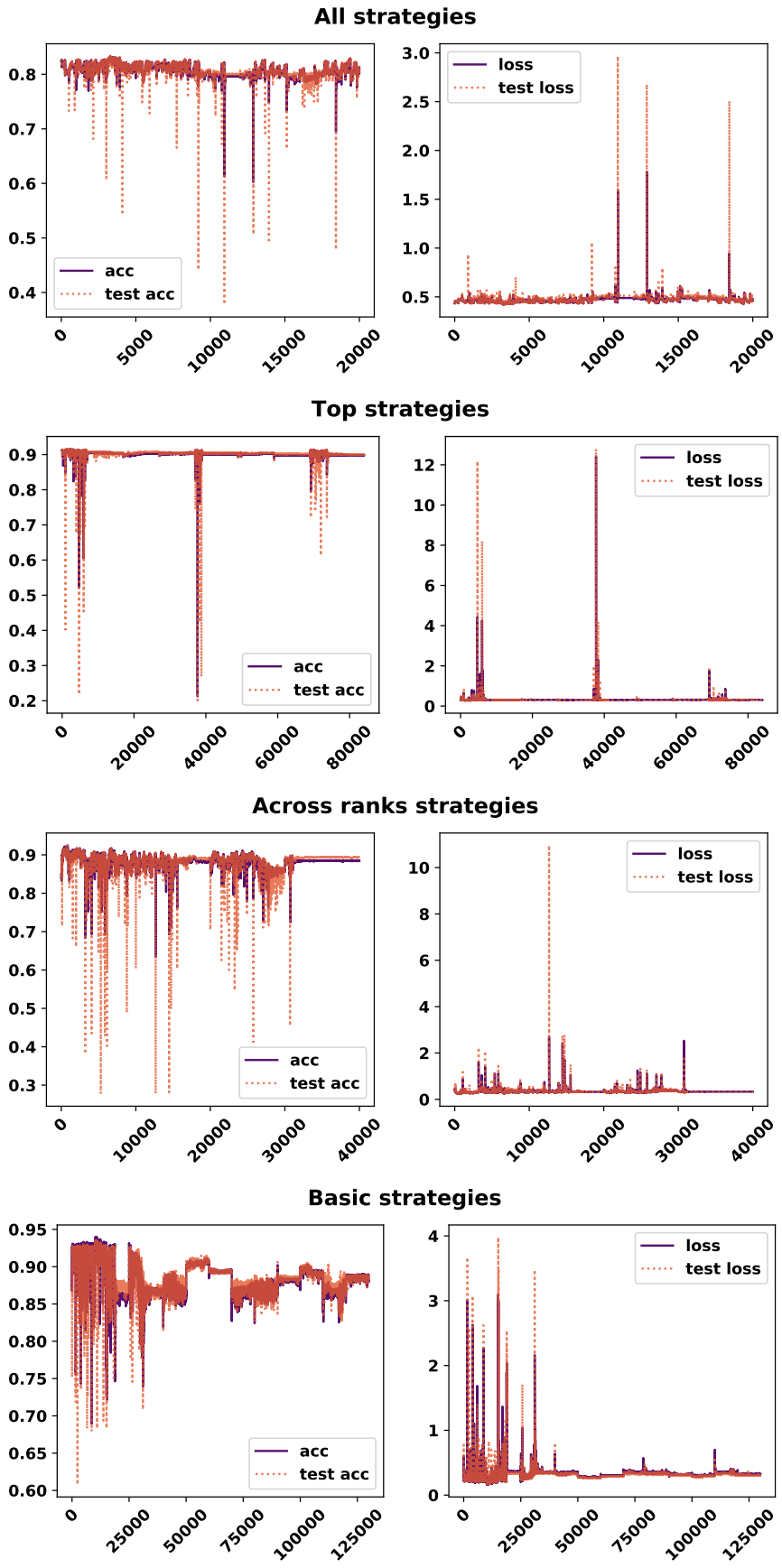


Figure 1.10: Loss function and accuracy of the networks based on the StoS model, over the number of epochs.

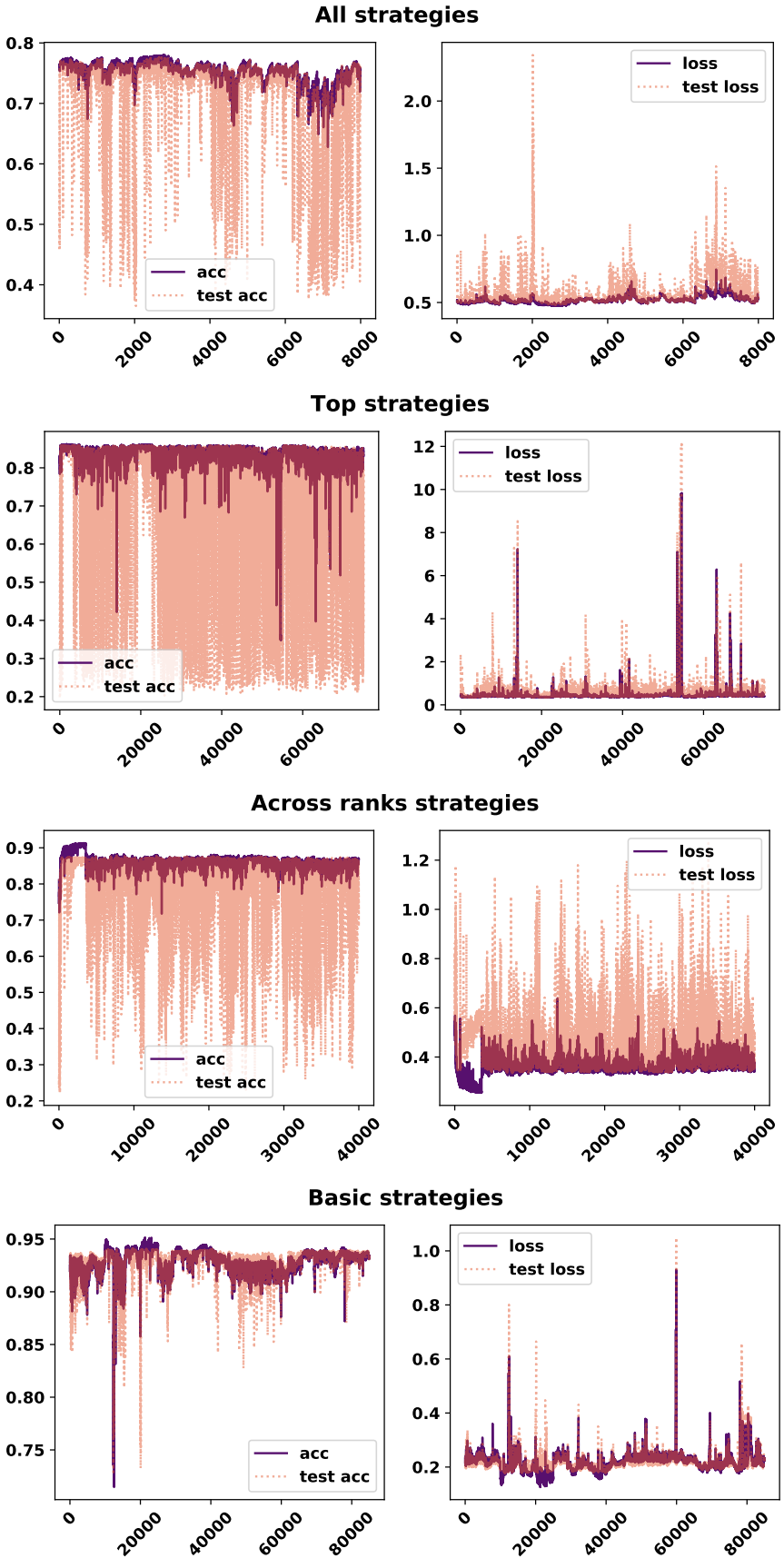


Figure 1.11: Loss function and accuracy of the networks based on the StoP model, over the number of epochs.

- A Keras model. The input models are the 8 trained LSTM networks presented in section 1.3.
- A reshape history function. A function that reshapes the opponent's history to an LSTM input.
- The probability that the strategy cooperates on the first turn denoted as p_o. The LSTM networks can be used to predict the strategy's next action following the opponent's opening turn. Thus, the strategy's opening action must be manually defined.

Following the opening turn the LSTM strategy makes a prediction based on the opponent's history. The strategy has an infinite memory because it needs to remember all the actions made by the opponent. The prediction of the networks correspond to the probability of cooperating. The LSTM strategy makes a deterministic decision based on the predicted probability. It cooperates if the prediction on the last time step is greater than 0.5, otherwise it defects.

```python
import numpy as np

import axelrod as axl
from axelrod.random_ import random_choice
from keras.layers import LSTM, Dense, Dropout
from keras.models import Sequential

C, D = axl.Action.C, axl.Action.D


class LSTMPlayer(axl.Player):
    name = "The LSTM player"
    classifier = {
        "memory_depth": float("inf"),
        "stochastic": True,
        "inspects_source": False,
        "manipulates_source": False,
        "manipulates_state": False,
    }

    def __init__(self, model, reshape_history_funct, opening_probability=0.78):
        self.model = model
        self.opening_probability = opening_probability
        self.reshape_history_function = reshape_history_funct
        super().__init__()
        if opening_probability in [0, 1]:
            self.classifier["stochastic"] = False

    def strategy(self, opponent):
        if len(self.history) == 0:
            return random_choice(self.opening_probability)

        history = [action.value for action in opponent.history]
        prediction = float(
            self.model.predict(self.reshape_history_function(history))[0][-1]
        )

        return axl.Action(round(prediction))

    def __repr__(self):
        return self.name
```

Figure 1.12: Implementation of the `LSTMPlayer` class.

The 8 trained LSTM networks are used to introduce 24 new IPD strategies. Each network corresponds to three distinct player with a different opening move. More specifically, three

different values of p_o are used here. These are $p_o = 0$, $p_o = 1$ and $p_o = 0.78$. The probability 0.78 is the probability that the best response sequences of Chapter **??** open with a cooperation. Thus, a total of $8 \times 3 = 24$ IPD strategies are evaluated in this section.

The performance of the LSTM strategies are evaluated and compared in 300 standard tournaments. The process of collecting the tournament results for each strategy is given by Algorithm 1.

Algorithm 1: Data collection Algorithm

foreach *seed* $\in [0, 300]$ **do**

$N \leftarrow$ randomly select integer $\in [s_{min}, s_{max}]$;
players $\leftarrow$ randomly select s players;
players $\leftarrow$ players + LSTM strategy;
$N \leftarrow s + 1$;
$k \leftarrow 50$;
$n \leftarrow 200$;

result standard $\leftarrow$ Axelrod.tournament(players, n, k);

return *result standard*;

For each trial a random size $s \in [5, 10]$ is selected, and from the 192 strategies of Appendix, a random list of s to the list of players, increasing the size to $s+1$. For the given list of strategies a standard tournament of 200 turns is performed and repeated 50 times. The number of turns is fixed at 200. In Chapter **??** the sequences were fixed to 205 turns which resulted in many best response sequences to defect on the last turn as the match was coming to an end. To avoid a series of unconditional defections by the LSTM strategies their performance is evaluated in 200 turns, which are a common number of turns used in the IPD literature.

A total of 300 trials of Algorithm 1 have been run. For each trial a result summary (in the format of Table **??**) is exported. The performance of the strategies are evaluated on the normalised rank r, and more specifically on the median normalised rank $\bar{r}$. As a reminder r is calculated as a strategy's rank divided by $N - 1$.

The $\bar{r}$ of each of the 24 strategies over the 300 standard tournaments is given by Table 1.4.

	sequence to sequence			**sequence to probability**		
	$p_o = 0$	$p_o = 1$	$p_o = 0.78$	$p_o = 0$	$p_o = 1$	$p_o = 0.78$
All strategies	0.667	0.222	0.333	0.778	0.333	0.500
Top strategies	0.714	0.444	0.500	0.500	0.429	0.429
Across ranks strategies	0.750	0.667	0.683	0.500	0.250	0.300
Basic strategies	0.800	0.600	0.625	0.800	0.300	0.429

Table 1.4: The median normalised ranks of the 24 LSTM strategies over the standard tournaments. A $\bar{r}$ closer to 0 indicates a more successful performance.

The strategy with the lowest $\bar{r}$ over the 300 tournaments is the LSTM strategy based on the

StoS network trained over entire training set with $p_o = 1$. The strategy achieved a $\bar{r}$ of 0.222. The second most successful performance is by the StoP based strategy trained against across the rank strategies with $p_o = 1$. In section 1.3 it was indicated that the specific network was overfitting, however, the strategy based on the network out performs any other StoP strategy. There are a few strategies that have achieved a $\bar{r}$ close to 0.3. These include the StoS strategy trained on the entire data set with $p_o = 0.78$, and the StoP strategies trained against all strategies, across the ranks strategies, and against the basic strategies with $p_o = 1$, $p_o = 0.78$ and $p_o = 1$ equivalently.

The LSTM strategies that open with cooperation outperform any other strategy based on the same network and training data set that does not. From Table 1.4 it is indicated the strategies trained on the subsets perform better when they are based on the StoP model.

The r distributions for each strategy are given by Figures 1.13- 1.14.

Figure 1.13 gives the r distributions for the strategies based on the StoS network. All the distributions for $p_o = 0$ have a median higher than 0.66, indicating that those strategies on average perform in the bottom half of a tournament. The most successful strategy is the strategy trained against all strategies with $p_o = 1$. The strategies distribution shows that the strategy ranked highly in most of the tournament it participated with only a few exceptions. Even if when the p_o is lowered to 0.78 the strategy still performs adequately. The rest of the distributions appear to have peaks either in the middle or closer to 1. A statistical summary of the distributions are given by Table 1.5.

		count	mean	std	min	10%	25%	50%	75%	95%	max	skew	kurt
All strategies	$p_o = 0$	300.0	0.620	0.278	0.0	0.200	0.429	0.667	0.839	1.000	1.0	-0.523	-0.587
	$p_o = 1$	300.0	0.295	0.252	0.0	0.000	0.111	0.222	0.458	0.779	1.0	0.702	-0.251
	$p_o = 0.78$	300.0	0.368	0.265	0.0	0.000	0.143	0.333	0.560	0.833	1.0	0.433	-0.647
Top strategies	$p_o = 0$	300.0	0.655	0.273	0.0	0.250	0.500	0.714	0.875	1.000	1.0	-0.629	-0.436
	$p_o = 1$	300.0	0.461	0.274	0.0	0.090	0.222	0.444	0.667	0.875	1.0	-0.029	-0.940
	$p_o = 0.78$	300.0	0.509	0.255	0.0	0.167	0.333	0.500	0.704	0.876	1.0	-0.158	-0.710
Across ranks strategies	$p_o = 0$	300.0	0.643	0.306	0.0	0.141	0.486	0.750	0.875	1.000	1.0	-0.768	-0.523
	$p_o = 1$	300.0	0.636	0.262	0.0	0.250	0.500	0.667	0.833	1.000	1.0	-0.680	-0.198
	$p_o = 0.78$	300.0	0.645	0.255	0.0	0.250	0.500	0.683	0.833	1.000	1.0	-0.754	-0.034
Basic strategies	$p_o = 0$	300.0	0.728	0.263	0.0	0.333	0.600	0.800	1.000	1.000	1.0	-0.949	0.229
	$p_o = 1$	300.0	0.556	0.283	0.0	0.143	0.375	0.600	0.778	1.000	1.0	-0.365	-0.803
	$p_o = 0.78$	300.0	0.579	0.280	0.0	0.167	0.375	0.625	0.800	1.000	1.0	-0.435	-0.772

Table 1.5: Statistics summary of the r distributions for the strategies based on the StoS network.

Figure 1.14 gives the r distributions for the strategies based on the StoP network. For $p_o = 0$ the performance of the strategies remains poorly. The strategies that have been trained against all the strategies, across the ranks and basic strategies with $p_o = 1$ appears to have managed to win several of the tournaments they participated in. The statistics summary of the distributions is given by Table 1.6.

An interesting question that arises is: what is the probability that a strategy rank in the top half of a tournament? This is answered by calculating the cumulative distribution function (CFD) of the r distributions. They are given by Figures 1.15-1.16.

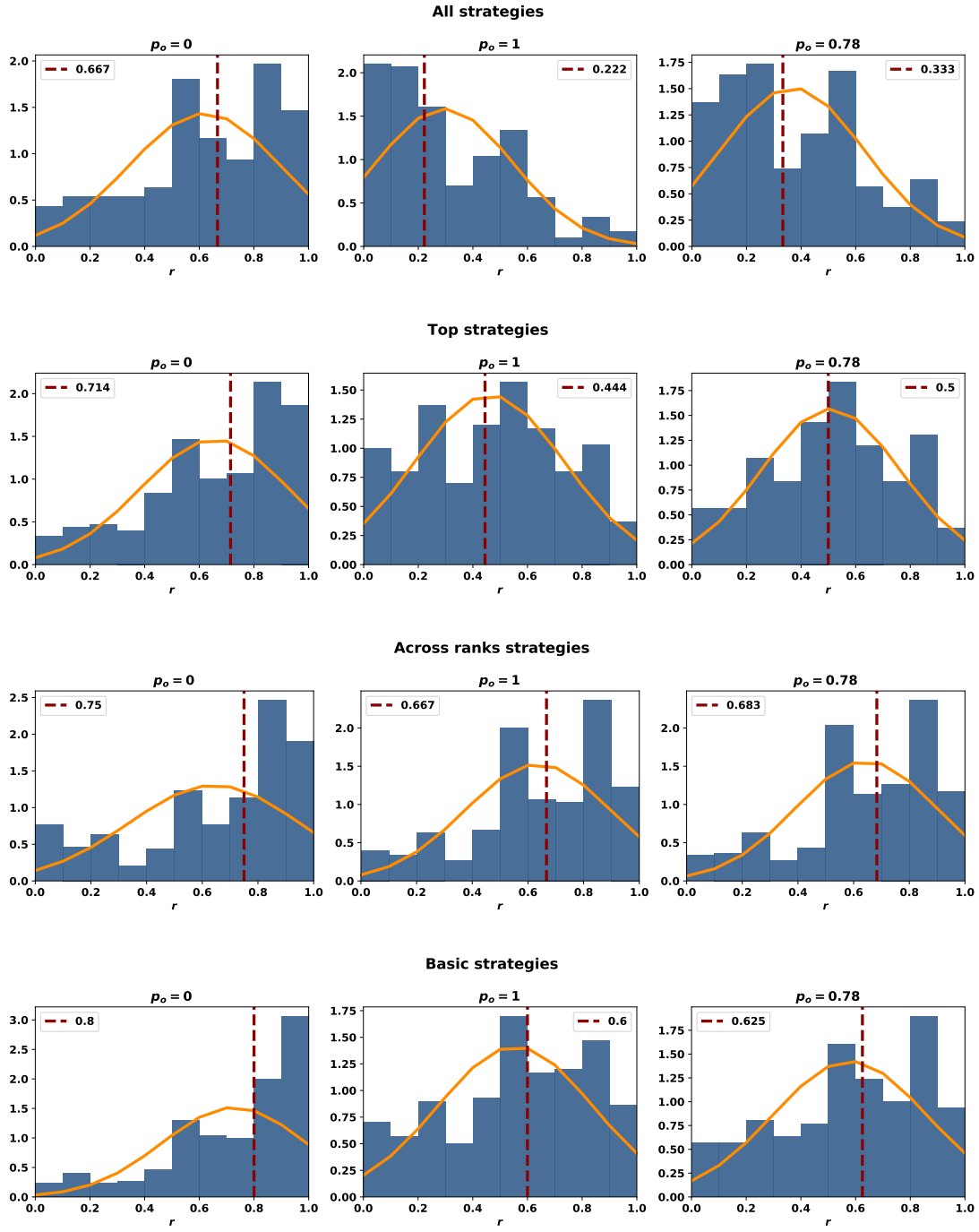


Figure 1.13: Normalised rank distributions for the strategies which are based on the StoS LSTM.

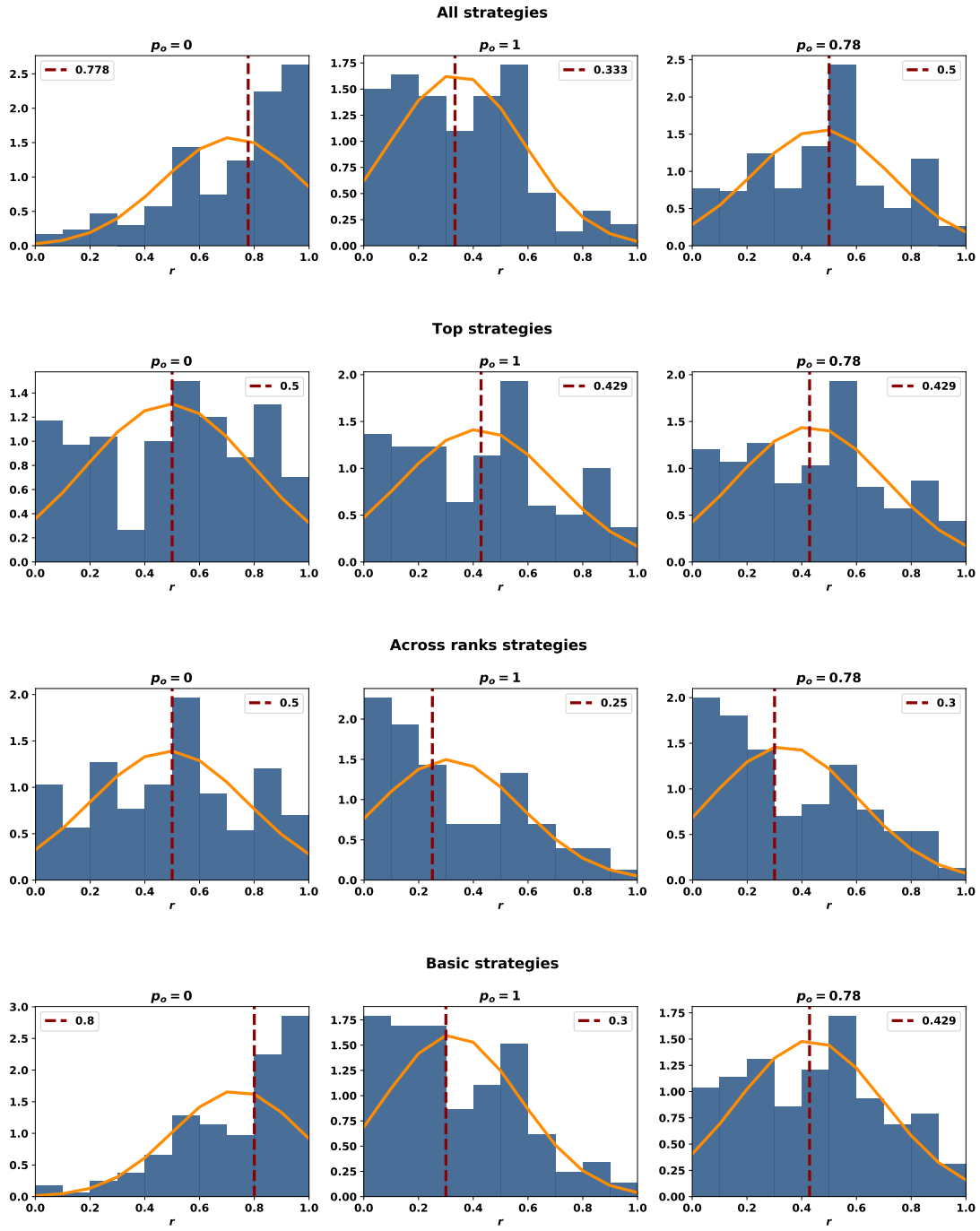


Figure 1.14: Normalised rank distributions for the strategies which are based on the StoP LSTM.

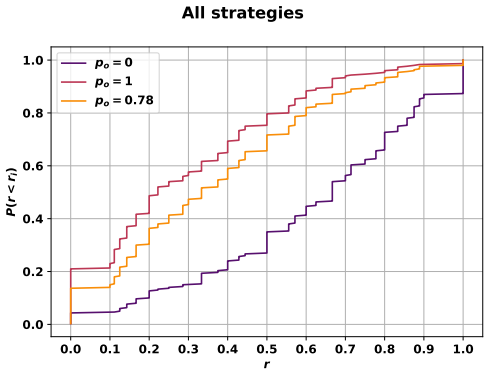
		count	mean	std	min	10%	25%	50%	75%	95%	max	skew	kurt
All strategies	$p_o = 0$	300.0	0.720	0.254	0.0	0.333	0.571	0.778	0.900	1.000	1.0	-0.860	0.056
	$p_o = 1$	300.0	0.339	0.244	0.0	0.000	0.125	0.333	0.500	0.800	1.0	0.458	-0.280
	$p_o = 0.78$	300.0	0.471	0.255	0.0	0.125	0.286	0.500	0.625	0.875	1.0	-0.064	-0.696
Top strategies	$p_o = 0$	300.0	0.491	0.305	0.0	0.000	0.200	0.500	0.714	1.000	1.0	-0.131	-1.140
	$p_o = 1$	300.0	0.417	0.283	0.0	0.000	0.167	0.429	0.600	0.875	1.0	0.157	-0.974
	$p_o = 0.78$	300.0	0.432	0.277	0.0	0.000	0.200	0.429	0.625	0.876	1.0	0.109	-0.891
Across ranks strategies	$p_o = 0$	300.0	0.487	0.287	0.0	0.000	0.286	0.500	0.700	1.000	1.0	-0.037	-0.899
	$p_o = 1$	300.0	0.308	0.267	0.0	0.000	0.111	0.250	0.500	0.800	1.0	0.586	-0.695
	$p_o = 0.78$	300.0	0.335	0.272	0.0	0.000	0.125	0.300	0.556	0.800	1.0	0.465	-0.867
Basic strategies	$p_o = 0$	290.0	0.738	0.239	0.0	0.400	0.600	0.800	0.975	1.000	1.0	-0.881	0.315
	$p_o = 1$	290.0	0.323	0.249	0.0	0.000	0.125	0.300	0.500	0.778	1.0	0.491	-0.535
	$p_o = 0.78$	290.0	0.432	0.269	0.0	0.000	0.200	0.429	0.625	0.875	1.0	0.096	-0.916

Table 1.6: Statistics summary of the r distributions for the strategies based on the StoP network.

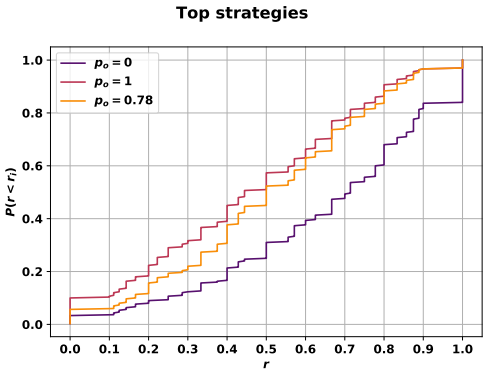
For the StoS strategies trained against all the strategies there is a 0.78 and 0.71 probability that the strategies rank is smaller than 0.5 for $p_o = 1$ and $p_o = 0.78$ equivalently. For the StoP strategies the highest probabilities of ranking in the top half of a tournament are 0.78, 0.76, 0.73 and 0.77 for the strategies trained against all strategies with $p_o = 1$, against across the rank strategies with $p_o = 1$ and $p_o = 0.78$, and against the basic strategies with $p_o = 1$. Overall, the successful strategies covered in this section have a 0.70-0.80 probability of being on the top ranks on a standard IPD tournament.

This section has evaluated the performance of 24 newly introduced IPD strategies based on LSTM networks. The performance of the strategies was evaluated based on their normalised ranks over 300 standard computer tournaments. On the whole, the analysis of this section has shown that:

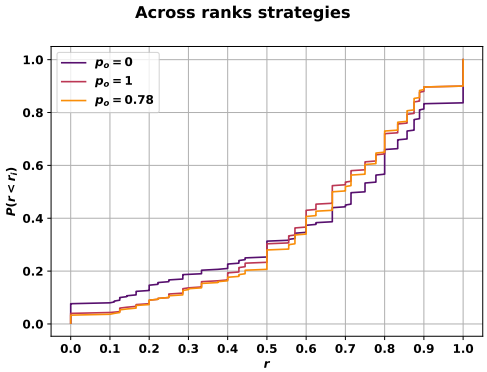
- The LSTM strategies which were trained on the entire data set of best responses were successful strategies regardless of the LSTM architecture. Both the StoS and StoP networks have produced strategies that can win IPD tournaments, and on average rank on the top 30% of any given standard tournament.
- The LSTM strategies trained on subsets of the training data set, perform better when trained with the StoP network than the StoS network. The StoP networks for the subsets have been trained for a longer number of epochs compared to the StoP network over the entire data set. An interesting question is whether the StoP strategy, trained on the entire data set, would perform even better than it's equivalent StoS strategy if it was trained for longer.
- The successful strategies of this analysis, strategies with $\bar{r} < 0.33$, have a 0.70-0.80 probability of ranking in top half of a standard tournament.
- The LSTM strategies that have been trained against the top ranked strategies performed poorly. This could indicate that training only on the top strategies did not provide enough diverse training samples. This could have in turn made the strategies less adaptable to diverse environments.
- Overall the most successful strategies of the analysis have been strategies that open with



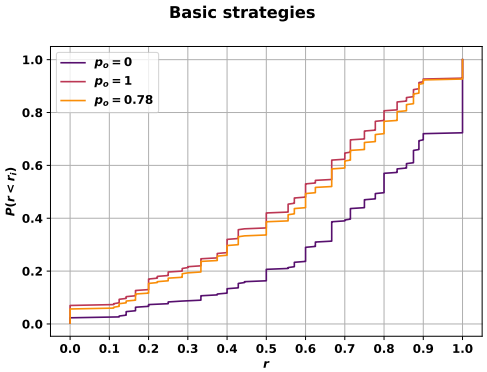
(a) For $p_o = 0$ the probability that $r < 0.5$ is 0.350. For $p_o = 1$ the probability that $r < 0.5$ is 0.797. For $p_o = 0.78$ the probability that $r < 0.5$ is 0.717.



(b) For $p_o = 0$ the probability that $r < 0.5$ is 0.310. For $p_o = 1$ the probability that $r < 0.5$ is 0.573. For $p_o = 0.78$ the probability that $r < 0.5$ is 0.523.

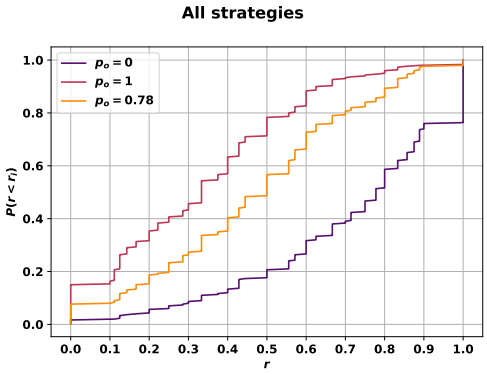


(c) For $p_o = 0$ the probability that $r < 0.5$ is 0.313. For $p_o = 1$ the probability that $r < 0.5$ is 0.303. For $p_o = 0.78$ the probability that $r < 0.5$ is 0.280.

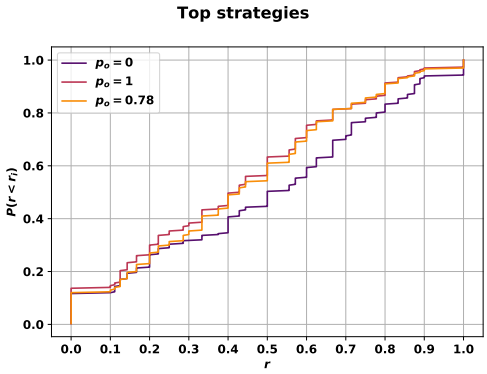


(d) For $p_o = 0$ the probability that $r < 0.5$ is 0.207. For $p_o = 1$ the probability that $r < 0.5$ is 0.420. For $p_o = 0.78$ the probability that $r < 0.5$ is 0.387.

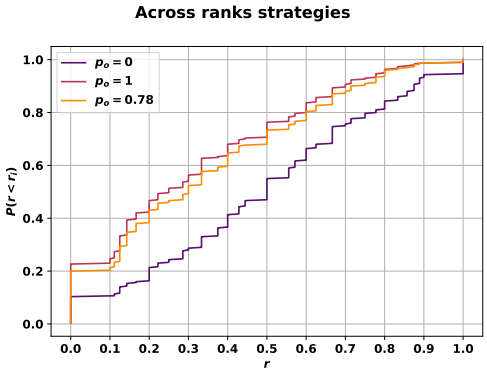
Figure 1.15: The cumulative distribution function (CFD) for the r distributions for the LSTM strategies based on the StoS network.



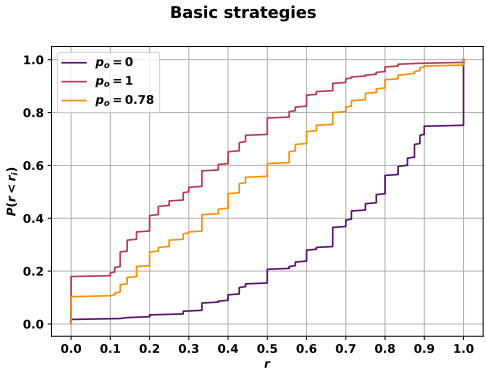
(a) For $p_o = 0$ the probability that $r < 0.5$ 0.207. For $p_o = 1$ the probability that $r < 0.5$ 0.783. For $p_o = 0.78$ the probability that $r < 0.5$ 0.567.



(b) For $p_o = 0$ the probability that $r < 0.5$ 0.503. For $p_o = 1$ the probability that $r < 0.5$ 0.633. For $p_o = 0.78$ the probability that $r < 0.5$ 0.610.



(c) For $p_o = 0$ the probability that $r < 0.5$ 0.550. For $p_o = 1$ the probability that $r < 0.5$ 0.763. For $p_o = 0.78$ the probability that $r < 0.5$ 0.733.



(d) For $p_o = 0$ the probability that $r < 0.5$ 0.207. For $p_o = 1$ the probability that $r < 0.5$ 0.779. For $p_o = 0.78$ the probability that $r < 0.5$ 0.607.

Figure 1.16: The cumulative distribution function (CFD) for the r distributions for the LSTM strategies based on the StoP network.

a cooperation.

1.4.1 Fingerprinting the LSTM based strategies

The 24 strategies that have been introduced in this Chapter are based on an LSTM archetype. These strategies are based on a complex structure and interpreting their behaviour is not trivial. The difference between the strategies is not straightforward either. In Chapter **??** a method that produces a functional signature of a strategy called fingerprinting was presented. More specifically, two types of fingerprints were discussed which were the Ashlock's fingerprints and the transitive fingerprints.

Ashlock's fingerprints compute the score of a strategy against a spectrum of opponents. The basic method is to play the strategy against a probe strategy with varying noise parameters. The fingerprints for the 24 strategies based on Ashlock's approach have been generated for the probe strategies Tit For Tat and Pavlov. These are given by Figures 1.17 - 1.20. Note that the strategies that appear on the same row have had the same training process. This means that they are based on the same network type and have been trained on the same training data set.

For Figures 1.17 and 1.18 Tit For Tat was used as the probe strategy. These demonstrate that the strategies that had the same training process but have a different opening move are quite similar. However, the strategies based on different networks and training data sets appear to behave in a different way. The only strategy that has some similarities regardless the network architecture is the strategy trained against the top strategies.

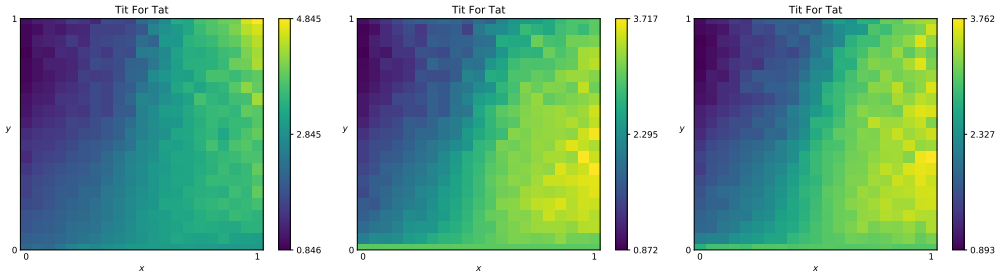
Figures 1.19 and 1.20 give the Ashlock fingerprints whilst Pavlov is used as a probe. The fingerprints of the strategies across the different training methods appear to be more similar when they are matched with Pavlov.

To further explore the similarities of the strategies a set of more interpretable fingerprints, the transitive fingerprints, implemented in APL have also been generated. The transitive fingerprints represent the cooperation rate of a strategy against a set of opponents over a number of turns. There are two set of opponents used here to generate the transitive fingerprints, these are the collection of strategies from [25] and from [4].

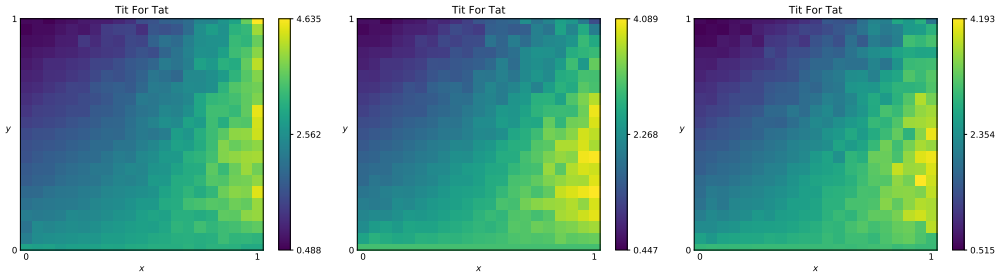
The transitive fingerprints are given by Figures 1.21 -1.24. The differences between the strategies are more distinct using the transitive fingerprinting method. It can be seen that the strategies that have had different training processes behave differently against the same list of opponents. The strategies that have had the same training process but have a different opening moves, have some similarities, but overall behave differently. More specifically, it can been seen that the strategies that open with a cooperation achieve a higher cooperating rate compared to strategies that do not.

1.5 Chapter Summary

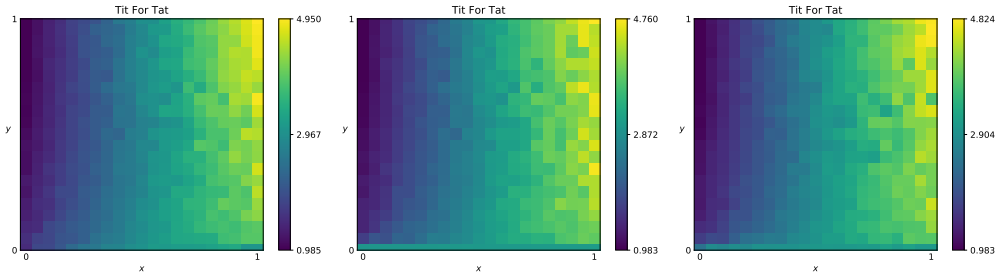
This Chapter has introduced a total of 8 new IPD strategies based on LSTM networks. LSTMs have received a lot of attention in the machine learning literature due to their remarkable results on applications such as speech recognition. However, LSTMs have not received much attention



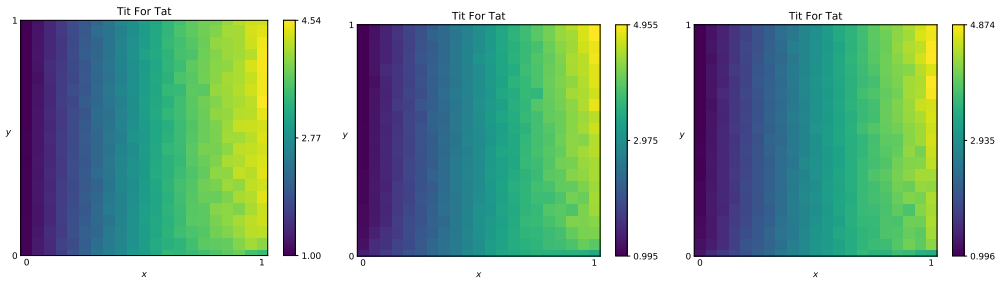
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

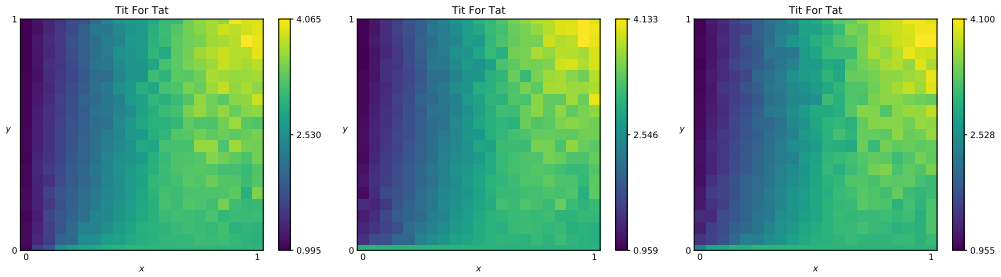


(c) Strategies trained against the across the ranks strategies.

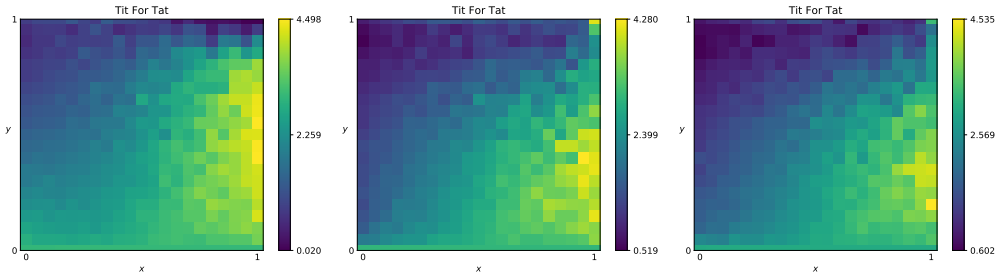


(d) Strategies trained against basic strategies.

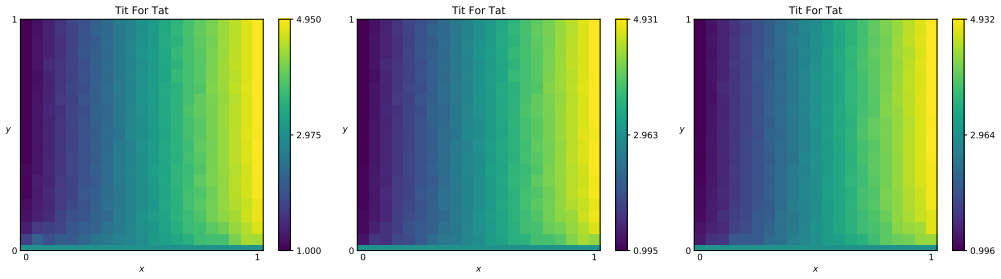
Figure 1.17: Ashlock's fingerprints for the LSTM strategies based on the StoS network when Tit For Tat is the probe strategy.



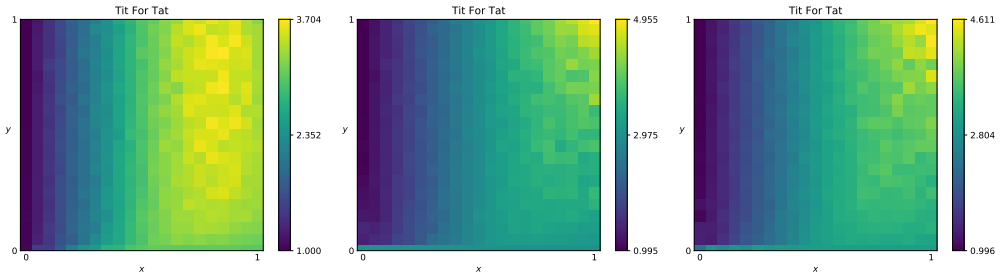
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

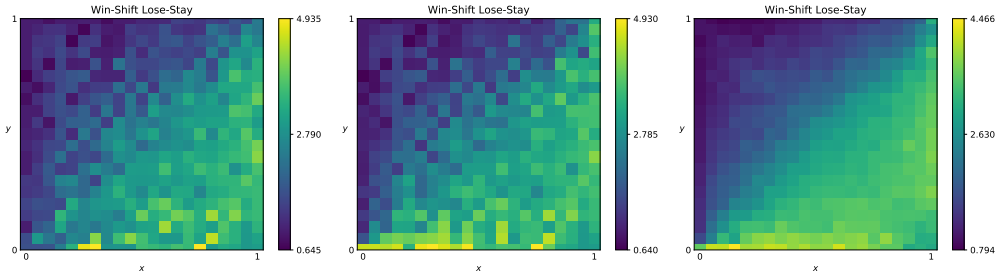


(c) Strategies trained against the across the ranks strategies.

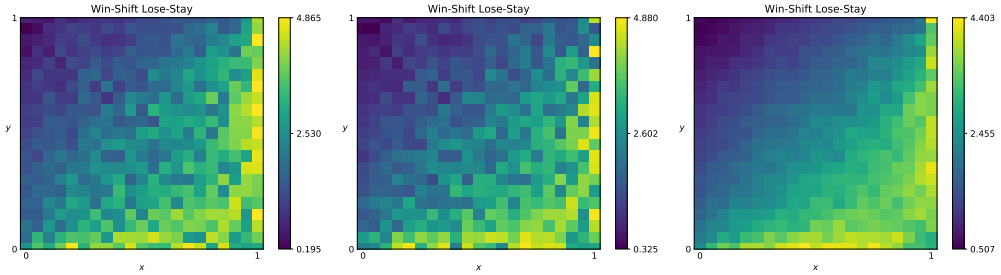


(d) Strategies trained against basic strategies.

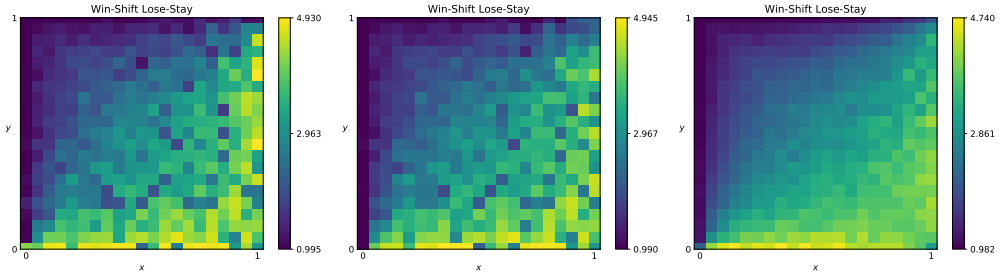
Figure 1.18: Ashlock's fingerprints for the LSTM strategies based on the StoP network when Tit For Tat is the probe strategy.



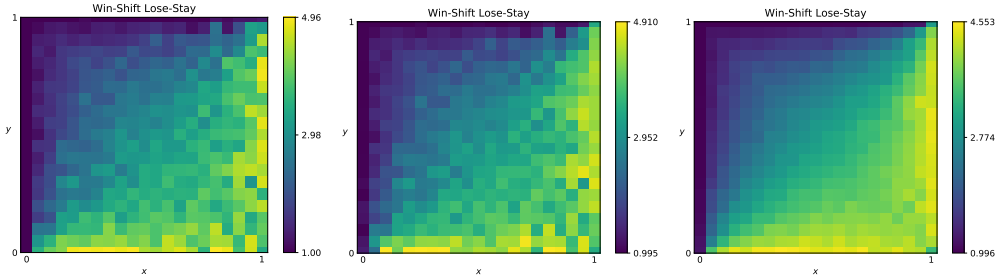
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.



(c) Strategies trained against the across the ranks strategies.

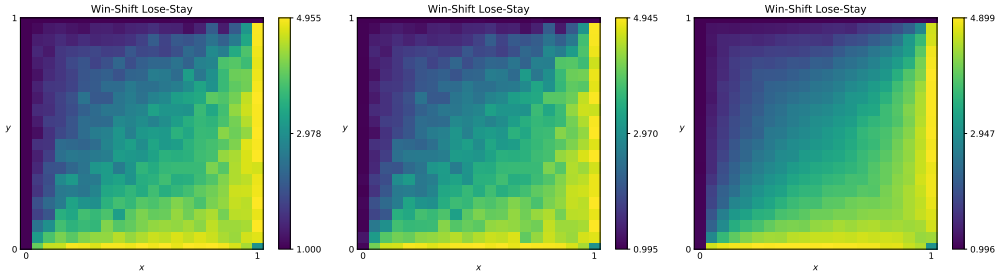


(d) Strategies trained against basic strategies.

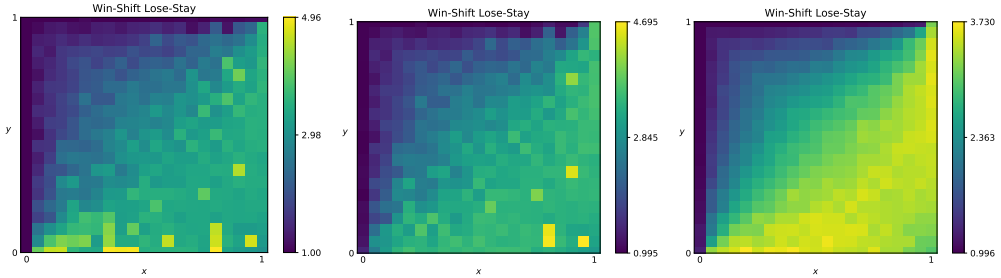
Figure 1.19: Ashlock's fingerprints for the LSTM strategies based on the StoS network when Pavlov is the probe strategy.

(a) Strategies trained on the entire training data set.

(b) Strategies trained against the top performing strategies.



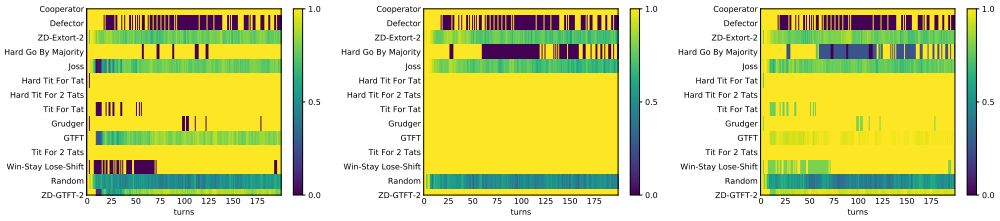
(c) Strategies trained against the across the ranks strategies.



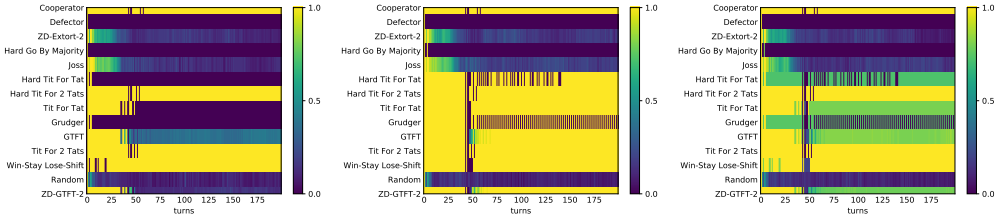
(d) Strategies trained against basic strategies.

Figure 1.20: Ashlock's fingerprints for the LSTM strategies based on the StoP network when Pavlov is the probe strategy.

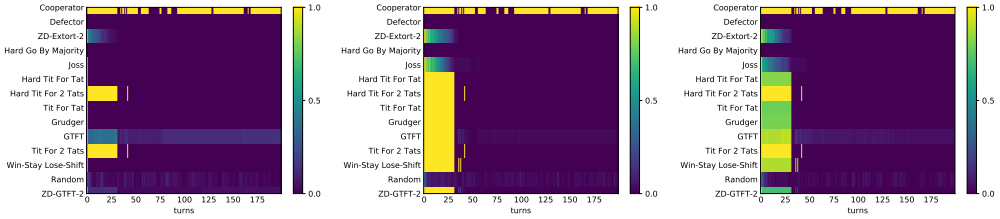
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

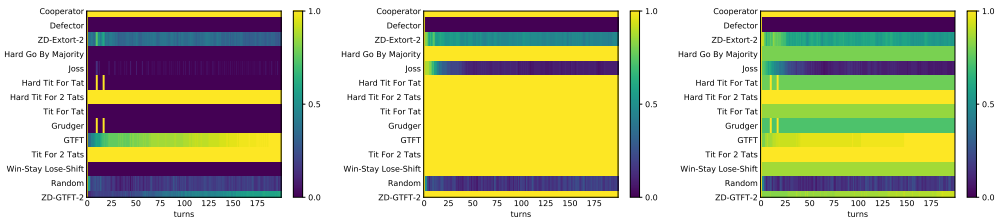


(c) Strategies trained against the across the ranks strategies.

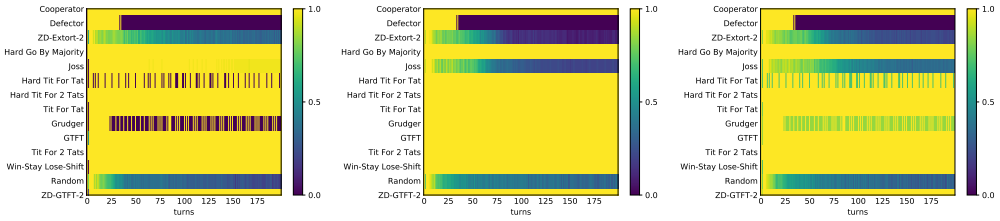


(d) Strategies trained against basic strategies.

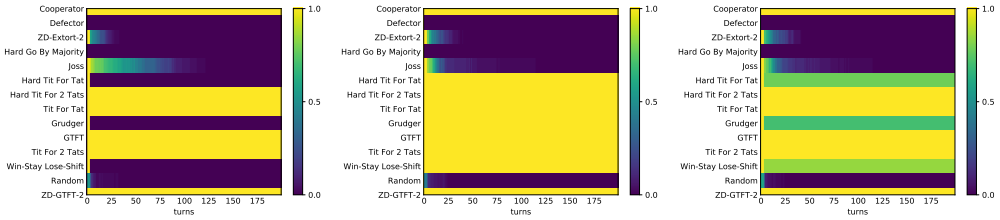
Figure 1.21: Transitive fingerprints for the LSTM strategies based on the StoS network against the list of opponents from [25].



(a) Strategies trained on the entire training data set.



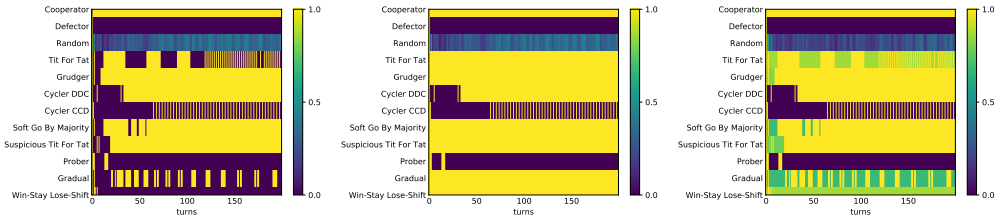
(b) Strategies trained against the top performing strategies.



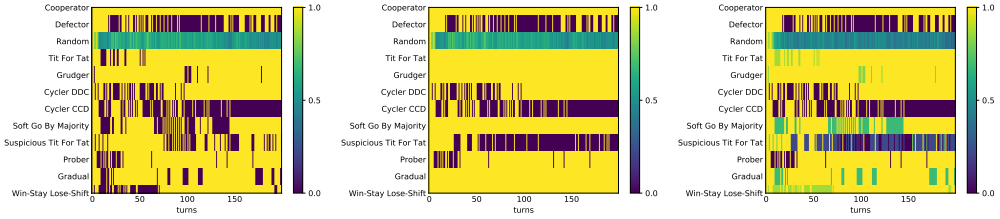
(c) Strategies trained against the across the ranks strategies.

(d) Strategies trained against basic strategies.

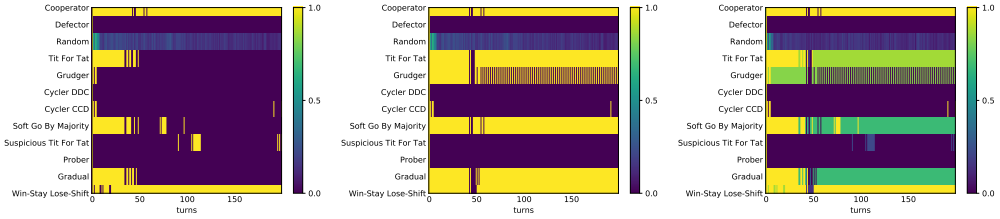
Figure 1.22: Transitive fingerprints for the LSTM strategies based on the StoP network against the list of opponents from [25].



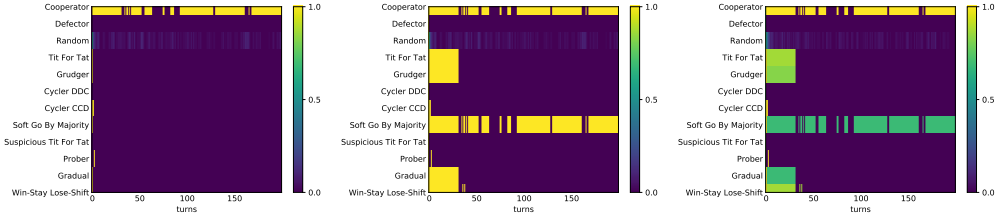
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

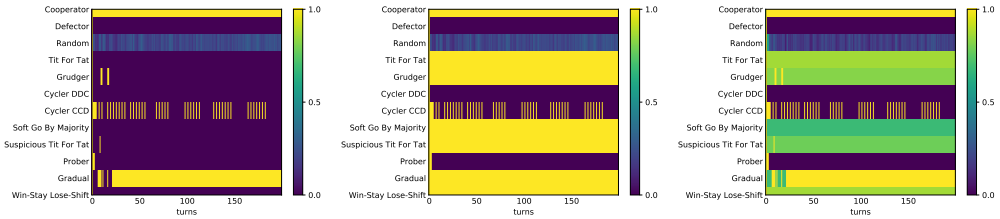


(c) Strategies trained against the across the ranks strategies.

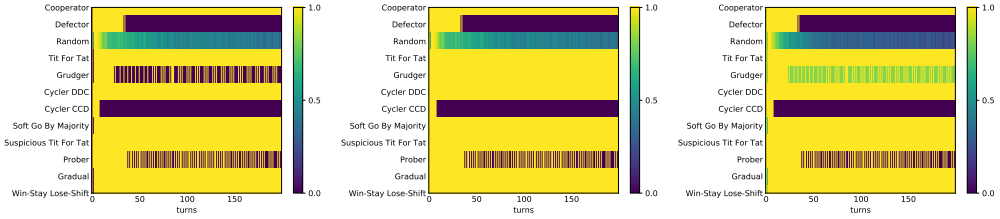


(d) Strategies trained against basic strategies.

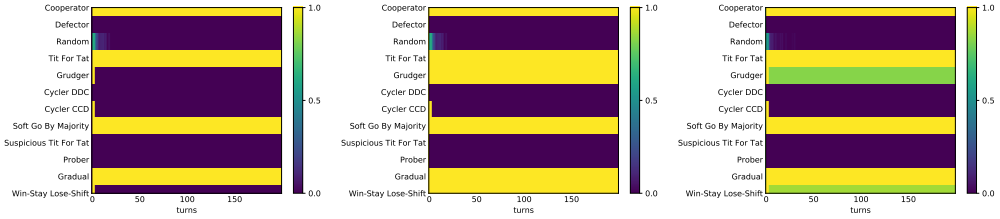
Figure 1.23: Transitive fingerprints for the LSTM strategies based on the StoS network against the list of opponents from [4].



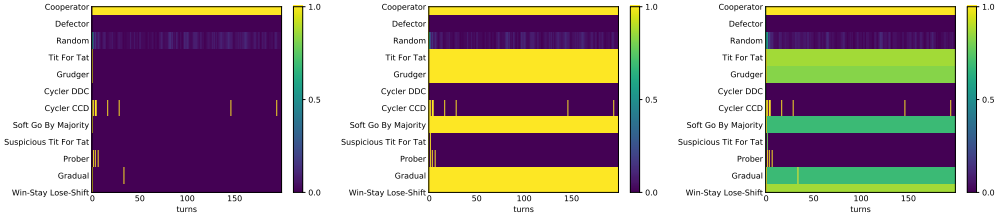
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.



(c) Strategies trained against the across the ranks strategies.



(d) Strategies trained against basic strategies.

Figure 1.24: Transitive fingerprints for the LSTM strategies based on the StoP network against the list of opponents from [4].

in the IPD literature.

The advantage of using LSTMs contrary to feed forward networks is that the LSTM incorporate a mechanism of memory, as presented in section 1.2. LSTM work with inputs in the form of sequences. The collection of best response sequences of Chapter **??** was purposely generated so that LSTMs could be trained to play optimal against a list of known IPD strategies.

Two types of LSTM networks have been trained in this Chapter. These were refereed to as the sequence to sequence network and the sequence to probability network. The differences between the networks were discussed in section 1.3. The networks were trained not only on a single a training data set, but on four different training data sets generated by the collection of best response strategies. Three sub sets of the training data set were consider to understand the effect of the training data on the network's performance in a IPD match. The subsets included the best response strategies to top performing strategies, to strategies with ranks across a standard tournament and the a set of basic strategies. The networks were developed and trained using the open source package Keras, and the training process was carried out on a GPU.

A total of 8 LSTM networks were trained, and those corresponded to 24 strategies when the opening move was taken into account. The opening move of an LSTM strategy had to be manually defined and three different values were chosen to carried of the evaluation analysis. These were opening with a defection, with a cooperating, and with a 0.78 probability of cooperation. The performance of the 24 LSTM strategies were evaluated in 300 standard tournaments. The results of the meta tournament analysis demonstrated that the strategies trained on the entire data set performed well in the 300 standard tournaments regardless the LSTM network type. However, the strategies trained on the subsets performed well only for the StoP network. The strategies that performed well in the analysis of this Chapter have a high probability of ranking in the top half of any standard tournament. This demonstrates that LSTM strategies trained on a collection of sequences can lead to successful behaviours in the IPD.

An interesting result demonstrated by the analysis was the effect of the opening with a cooperation. The most successful strategies of this Chapter have been the strategies that cooperated on the first turn. Moreover, the transitive fingerprints demonstrated that is because these strategies achieve a higher cooperating rate compared to strategies based on the same training process that do not. This result reinforces the discussion started ny Axelrod: opening with a cooperation is a property that successful strategies in a IPD competition need to have.

Bibliography

[1] D. Ashlock and E. Y. Kim. Fingerprinting: Visualization and automatic analysis of prisoner's dilemma strategies. *IEEE Transactions on Evolutionary Computation*, 12(5):647–659, Oct 2008.

[2] D. Ashlock, E. Y. Kim, and N. Leahy. Understanding representational sensitivity in the iterated prisoner's dilemma with fingerprints. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(4):464–475, July 2006.

[3] Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in neural information processing systems*, pages 2814–2822, 2013.

[4] B. Beaufils, J. P. Delahaye, and P. Mathieu. Our meeting with gradual: A good strategy for the iterated prisoner's dilemma. 1997.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[6] Kumar Chellapilla and David B Fogel. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE transactions on neural networks*, 10(6):1382–1391, 1999.

[7] François Chollet et al. Keras. `https://keras.io`, 2015.

[8] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[9] Paul J Darwen and Xin Yao. Why more choices cause less cooperation in iterated prisoner's dilemma. 2:987–994, 2001.

[10] David B Fogel, Timothy J Hays, Sarah L Hahn, and James Quon. A self-learning evolutionary chess program. *Proceedings of the IEEE*, 92(12):1947–1954, 2004.

[11] Nelis Franken and Andries Petrus Engelbrecht. Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. *IEEE Transactions on evolutionary computation*, 9(6):562–579, 2005.

[12] Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Nikoleta E. Glynatsi, and Owen Campbell. Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *PLOS ONE*, 12(12):1–33, 12 2017.

[13] P. G. Harrald and D. B. Fogel. Evolving continuous behaviors in the iterated prisoner's dilemma. *Biosystems*, 37(1):135 – 145, 1996.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[16] Soteris A Kalogirou. Applications of artificial neural-networks for energy systems. *Applied energy*, 67(1-2):17–35, 2000.

[17] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89. Presses universitaires de Louvain, 2015.

[18] James L McClelland, David E Rumelhart, PDP Research Group, et al. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2:216–271, 1986.

[19] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[20] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

[21] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[22] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[23] Tuomas W Sandholm and Robert H Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37(1-2):147–166, 1996.

[24] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[25] A. J. Stewart and J. B. Plotkin. Extortion and cooperation in the prisoner's dilemma. *Proceedings of the National Academy of Sciences*, 109(26):10134–10135, 2012.

[26] Le Wang, Xuhuan Duan, Qilin Zhang, Zhenxing Niu, Gang Hua, and Nanning Zheng. Segment-tube: Spatio-temporal action localization in untrimmed videos with per-frame segmentation. *Sensors*, 18(5):1657, 2018.

[27] PJ Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. ph. d. thesis, harvard university, cambridge, ma, 1974. 1974.

[28] Barry J Wythoff. Backpropagation neural networks: a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 18(2):115–155, 1993.