Improved Action Prediction through Multiple Model Processing of Player Trajectories

Branden Ingram, Benjamin Rosman, Clint van Alten, Richard Klein
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
{branden.ingram, benjamin.rosman1, clint.vanalten, richard.klein}@wits.ac.za

***Abstract*—Action prediction in video games is the process of extracting useful information in order to predict the future actions of a player. Long-range dependencies and the dynamic nature of video games make it difficult for most algorithms to accurately predict the future actions of players. We propose a novel machine learning approach to improving future action prediction from video game trajectories. This method requires having first clustered player trajectories based on behaviour similarities. Our model consists of a set of LSTM based prediction modules each trained on a subset of data based upon a respective cluster. The effectiveness of our model is analysed on both a synthetic and natural dataset. We find that our future action prediction approach of leveraging multiple models trained on individual data subsets results in greater accuracy over a single model on a complete dataset.**

***Index Terms*—action prediction, player modelling**

I. Introduction

We look to solve the problem of action prediction given a dataset of video game trajectories. Video games are dynamic and temporal domains containing long term dependencies which all affect decision making. These characteristics make it difficult to train accurate prediction models. Our goal is to train a model which can forecast the behaviours represented across many trajectories rather than an optimal policy. In a competitive setting, this can help players outperform others or from a developer's perspective predictions can be utilised to modify future gameplay experiences.

Techniques involving analysis and prediction of time-series data have always been key for practical problems [1], [2]. For these problems, the goal is to extract useful information from historical data to determine future values. In particular, we propose a new method to improve the accuracy of future action predictions given an individual's current gameplay trajectory. This is a form of player modelling where abstracted descriptions of players are generated [3]. Another form of player modelling utilised unsupervised clustering techniques to reveal the characteristics representative of play-styles [4]. These learned characteristics serve to inform developers if players are playing as intended. Similar to how clustered data can aid a developer's understanding of their player base, our method utilises clusters to aid in improving prediction accuracy. Thakar and Mehta [5] also showed that separating data based upon identified traits benefited their overall approach.

Our main contribution comes in the form of a new neural-network-based Cluster Assisted Prediction (CAP) model for future action prediction which leverages trajectory clusters to improve prediction accuracy. By utilising the cluster information each sub-component of our model only needs to learn from semantically similar data rather than the entirety of a dataset. This allows the sub-components to specialise on their respective datasets increasing the overall performance. This idea of "cluster-then-predict" has been utilised for sentiment analysis [6]. However, where they used the identified cluster as a feature for a single Support Vector Machine classifier, we train multiple neural networks in a supervised fashion on data separated with respect to a cluster.

II. Related Work

A. *Time-Series Forecasting*

The standard neural network methods for performing time series prediction include MLP, RBF or Cascade Correlation Models, which use a sliding window of N-tuple inputs to predict a single output target value [7]. Since these methods are dependent on the size of the rolling window, important long-range dependencies can be forgotten over the course of the full training set. Long Short-Term Memory (LSTM) [8] solutions are a special form of Recurrent Neural Network's (RNN) [9] which makes use of a component called a "memory block" [8] which allows it to learn long term dependencies. LSTM prediction models have been utilised in StarCraft2 for learning action selection [10]. We, however, employ an LSTM prediction model to video game based trajectory data to predict player actions.

B. *Modelling player actions*

A common implementation of player modelling is that of modelling a player's choices (actions) given different scenarios (states) [3]. In its basic form, this consists of a model which indicates the likelihood of any available player action given any state. Traditionally action models were implemented for board game research to improve game tree searches by predicting opponent moves. In this case, the player model was expressed as an evaluation function [11]. Similar approaches have been applied to more complex games, however, the increasing sizes of state and action spaces make training accurate models difficult. This has resulted in solutions involving large scale computing [10] or utilising abstracted actions spaces called “options” [12]. Our approach differs from these by using supervised learning instead of reinforcement learning to predict low-level accurate action similar to Muñoz, Gutierrez and Sanchis [13].

C. Trajectory Clustering

Although the specific method for clustering trajectories is independent of our methodology we applied a similar model to Xie, Girshick and Farhadi [14] who implemented an LSTM-autoencoder which is capable of handling time-series data effectively. This model jointly optimises for both reconstruction loss as well as a clustering loss in an unsupervised fashion. In a separate study we were able to utilise a similar model to separate gameplay trajectories with respect to play-style [15].

III. Methodology

The proposed model depicted in Figure 1 takes a set X of trajectories as input. This input is fed through a gating mechanism to a corresponding Predictor Node (M_i). This means each $x \in X$ is only input into a single M_i which is selected based upon an identified cluster index (i). This index is the result of a clustering pre-processing step [15] which results in a set of data subsets H_i where $H_i \subset X$ and i is the index of a particular data subset. Here $i \in \{0, \dots, k\}$ where k is the number of identified clusters. The value of k dictates the number of Predictor Nodes found in our proposed CAP model for a given dataset.

A. Predictor Nodes

Our proposed CAP model makes use of a set of Predictor Nodes M_i, where each node is only trained on data from its corresponding data subset H_i. The individual Predictor Node architecture is depicted in Figure 2. The goal for each of these nodes is, for any given any partial trajectory $(x(0), \dots, x(t))$ within the subset of trajectories (H_i), to predict the next time step in that trajectory ($x(t+1)$), where t represents timestep. Our Predictor Nodes are comprised of two components. The first component is a standard LSTM layer which is required since our trajectories are variable in length and temporal in nature. This layer accepts as input any partial trajectory and outputs a one-dimensional vector which can be now processed by traditional non-recurrent networks. This encoding is then appended to the current time step ($x(t)$) of the partial trajectory and serves as the input to the second component, our fully connected neural network. Finally, this node outputs the next state as its prediction ($x(t+1)$).

IV. Experiments

To analyse the effectiveness of our approach, we compared the overall prediction accuracy of our CAP model with the prediction of a single model trained on the complete dataset (X).

A. Datasets

To validate our method in both synthetic and natural domains, we test our model on different datasets. Synthetic datasets are derived from a grid-world game where a player seeks out a goal with the opportunity of completing two additional optional objectives. The natural dataset is an unlabelled set of trajectories from the game Super Mario Bros [16] collected from human participants and used to demonstrate the model’s effectiveness on real-world data.

1) Grid World: We generate 5 individual datasets T_n from 5 different environments $(E_1, \dots, E_5)$ made up of trajectories generated to exhibit one of four set behaviours. These behaviours represent play-styles which are listed in Table I. This is achieved by modelling play-styles as different reward functions in a reinforcement learning paradigm. This idea of reward shaping has been used to train a set of human-like bots with differing styles [17].

Each environment is a 10×10 grid world, as depicted in Fig. 3. These environments each have a goal state (G, in green) and a start state (S, in blue). Walls (black tiles) cannot be traversed and trap states (red tiles) result in failure. The variety in play-styles is introduced through the addition of two bonus states (B_1, in gold and B_2, in cyan). These are the optional objectives that a player with certain preferences might wish to complete. The set of actions is the movement in any of the 4 primary cardinal directions. The set of reward functions R used to emulate these behaviours is defined in Table I. The respective bonus rewards were only given the first time an agent reached either B_1 or B_2.

TABLE I
Observable play-styles and reward structure

R	Behaviour	G reward	B_1 reward	B_2 reward
1	Moves directly to G	100	0	0
2	Visits B_1 before G	100	50	0
3	Visits B_2 before G	100	0	50
4	Visits B_1 and B_2 before G	100	50	50

In order to generate the trajectories we trained a RL agent for each of the combinations of R and E for 20000 episodes with discount factor $\gamma = 0.99$ and linear ϵ decay in order to ensure our agent converges to the global optimum. The state is given by the tuple (x, y, b_1, b_2) where x and y are the Cartesian grid coordinates and b_1 and b_2 indicate whether an agent has visited B_1 or B_2, respectively. Our dataset consists of 8000 trajectories, randomly selected from the training episodes, per R and E. Therefore a trajectory is a sequence of time steps in the form of (x, y, b_1, b_2).

2) Mario: This dataset consists of 74 playthroughs across 11 different levels of Super Mario Bros. These playthroughs were each captured by logging the actions of a unique human participant [16]. We then refactored this data into a trajectory, where each time step represents the current state of the playthrough at that point. We defined a state as a tuple given by (j, k, r, c, d, e) where j is the number of jumps, k is the number of enemies killed, r number of times the player has started running, c the number of coins collected, d the number

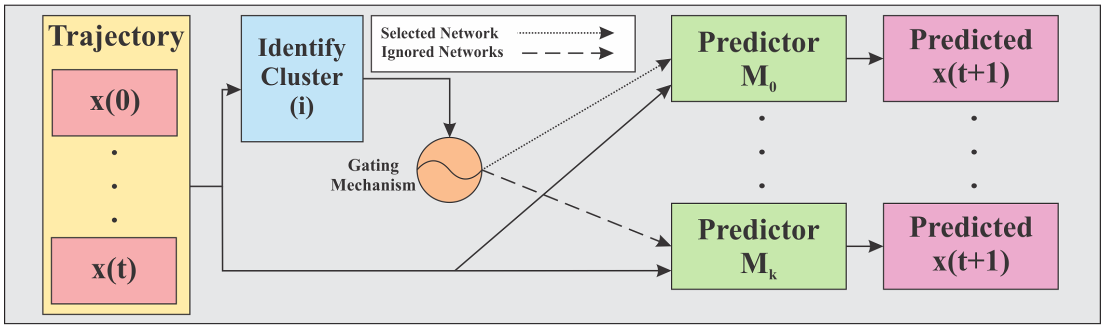

Fig. 1. Cluster Assisted Prediction Model (CAP)

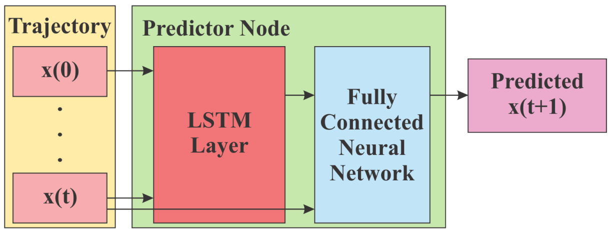

Fig. 2. Individual LSTM Predictor Node

of time the player died and e the unique encoding for each of the 37 actions.

B. Training

We trained our CAP model on a set of trajectories (X) where the cluster for each $x \in X$ has been identified and is represented by i. This identification is important as it dictates which Predictor Node our input trajectory should be passed to. The fully connected neural network component found in each Predictor Node of our CAP model consists of 3 hidden layers of size 5 with ReLu activations. Additionally, these nodes utilised a non-stacked LSTM which outputs an encoded state of size 8. By appending this encoded state by the last time step of the partial trajectory we get the input of the fully connected neural network. The input dimension of both the fully connected component and the LSTM layer is, therefore, dependent on the dataset's timestep dimension. For comparison, we trained a model (Single) with the same architecture as depicted in Figure 2 on the complete dataset (X). This model has the same structure as a single Predictor Node found in our CAP model. The difference is the size of the hidden layers found in the fully connected component. To ensure fairness between our CAP and Single models, we keep the total capacities of the models roughly equal. This is achieved by increasing the size of the hidden layers in the Single model by a factor dependent on the number of clusters. For the Single model, we used a hidden layer size of $5 \times k$ where k was the number of clusters. The resultant accuracy from this model would serve as the base of comparison with our other CAP model.

Both our CAP and Single models were each trained for 10000 episodes using the Adam optimiser with a learning rate 0.001 using Mean Square Error as our loss function. During each episode progressively longer partial trajectories are fed forward through the complete network until the entire trajectory has been processed. For each of these partial trajectories $(x(0), \ldots, x(t))$ the output (y') is compared to $x(t+1)$ with the loss being backpropagated through the entire network.

Our approach is dependent on utilising clustered data and, therefore, its overall effectiveness is dependent on the quality of the clustering. To demonstrate a benchmark benefit of our approach we utilise the fact that the grid world data was generated to exhibit particular styles. This means that a ground-truth clustering is known. We therefore also trained our CAP model on the data subsets separated using these ground truth labels. This process was repeated for each of our 6 environments $(E_1, \ldots, E_5, \texttt{Mario})$.

V. Results and Discussion

For testing purposes, we use unseen trajectories from each environment. For each trajectory (x) we process every partial trajectory $(x(0), \ldots, x(t))$ through our CAP and Single models to generate predictions y'. The overall performance is the percentage of correct predictions across the given testing set. The results of this process are depicted in Table II with bold values representing the best model for a given environment. Here we can see for each of the environments, whether structurally similar in the case of $E_1, \ldots, E_5$ or more complex (MARIO), our CAP model outperforms the Single model. Notably, this performance level was achieved on both synthetic and natural domains. This indicates our approach of utilising clustered subsets of data can be beneficial when using real-world data. Since our approach is partially dependent on the clustering accuracy, we analysed the effect of using the ground truth clusters available from the grid world environments. The results of using ground truth clustering serve as a benchmark comparison to minimize the impact of clustering performance. We observed that our CAP model performed at an even higher level when using the data separated using the ground truths. This indicates that with better clustering our CAP model is able to achieve greater accuracy.

Lastly, we analysed the ability of the individual sub-models $(M_0 \ldots M_k)$ to specialise in their respective data subsets $(H_0 \ldots H_k)$ averaged across all environments. This is achieved

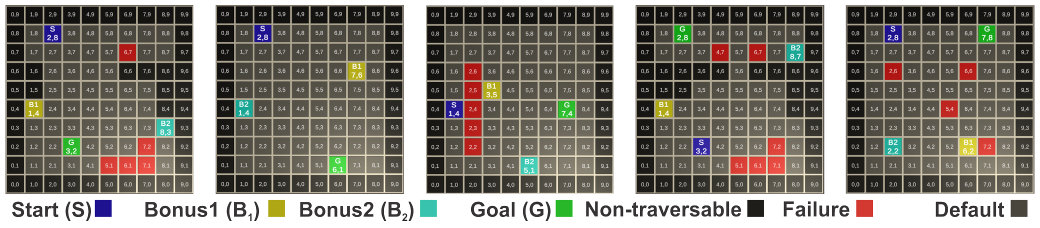

Fig. 3. Randomly generated grid world environments $E_1, \ldots, E_5$

TABLE II
PREDICTION ACCURACY ON CLUSTERED DATA

E	Random	Single	CAP	CAP (ground-truth)
E_1	0.25	0.56	0.6	**0.63**
E_2	0.25	0.51	0.53	**0.6**
E_3	0.25	0.34	0.47	**0.48**
E_4	0.25	0.58	0.62	**0.68**
E_5	0.25	0.52	0.56	**0.57**
Mario	0.025	0.30	**0.36**	N/A

TABLE III
PREDICTION ACCURACY OF SUB-MODELS ON EACH DATA SUBSET

	M_0	M_1	M_2	M_3	CAP
H_0	**0.65**	0.29	0.21	0.2	
H_1	0.25	**0.56**	0.19	0.21	
H_2	0.15	0.15	**0.66**	0.23	
H_3	0.22	0.21	0.2	**0.44**	
Average	0.32	0.3	0.32	0.27	**0.56**

by comparing the prediction accuracy of each sub-model on all data subsets for a given environment. The results of this process were then averaged across all environments and are depicted in Table III. Here we observe that the prediction accuracy of $M_0 \ldots M_3$ is higher when making predictions from trajectories within their corresponding data subset depicted by the bold values. Conversely, the prediction accuracy on trajectories from other subsets is far lower than even the average performance of the overall CAP model depicted in Table II. It is by specialising in these data subsets that when combined into our CAP model we obtain the overall benefit. Lastly, we observe that the average prediction accuracy of our CAP model across all environments is greater than the average performance for each sub-model.

VI. CONCLUSION

This paper proposes a new multi-model approach dubbed CAP which demonstrates the benefit of training multiple models on separated data over a single model on a complete dataset. Our CAP model was trained to perform optimally in a future action prediction task on multi-dimensional variable-length gameplay trajectories which had been separated with respect to a clustering process. We tested this on a synthetic dataset where cluster information was known as well as a natural domain where the ground truth clusters were not known. Through empirical analysis, we demonstrated that our CAP model consistently outperformed the Single model trained on the overall dataset. We also demonstrated that the performance increase came from the ability of the sub-models to specialise on their respective data subsets. We conclude that with the use of an initial data separation process you can obtain superior performance when training future state prediction models in video game environments.

REFERENCES

[1] E. M. Azoff, *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc., 1994.

[2] T. Edwards, D. Tansley, R. Frank, and N. Davey, "Traffic trends analysis using neural networks," in *Procs of the Int Workshop on Applications of Neural Networks to Telecommunications*, 1997.

[3] S. C. Bakkes, P. H. Spronck, and G. van Lankveld, "Player behavioural modelling for video games," *Entertainment Computing*, vol. 3, no. 3, pp. 71–79, 2012.

[4] A. Drachen, A. Canossa, and G. N. Yannakakis, "Player modeling using self-organization in tomb raider: Underworld," in *2009 IEEE symposium on computational intelligence and games*. IEEE, 2009, pp. 1–8.

[5] P. Thakar, A. Mehta *et al.*, "A unified model of clustering and classification to improve students' employability prediction," *(IJISA)*, vol. 9, no. 9, p. 10, 2017.

[6] R. Soni and K. J. Mathai, "Improved twitter sentiment prediction through cluster-then-predict model," *arXiv preprint arXiv:1509.02437*, 2015.

[7] N. A. Gershenfeld and A. S. Weigend, *The future of time series*. Xerox Corporation, Palo Alto Research Center Palo Alto, CA, USA, 1993.

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

[9] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[10] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[11] D. Cannel and S. Markovitch, "Learning models of opponent's strategy game playing," in *Proceedings of the 1993 AAAI Fall Symposium on Games: Learning and Planning*, 1993, pp. 140–147.

[12] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[13] J. Muñoz, G. Gutierrez, and A. Sanchis, "Towards imitation of human driving style in car racing games," in *Believable bots*. Springer, 2013, pp. 289–313.

[14] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[15] B. Ingram, B. Rosman, R. Klein, and C. van Alten, "Play-style identification through deep unsupervised clustering of trajectories," in *2022 IEEE Conference on Games (CoG)*. IEEE, 2022.

[16] M. Guzdial and M. Riedl, "Game level generation from gameplay videos," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[17] C. Arzate Cruz and J. A. Ramirez Uresti, "Hrlb: A reinforcement learning based framework for believable bots," *Applied Sciences*, vol. 8, no. 12, p. 2453, 2018.