Hybrid Prediction for Games' Rollback Netcode

Lior Diler
ESGI
Paris, France
diler.lior@gmail.com

Sami Dalil
ESGI
Paris, France
sdalil@free.fr

Marion Mota
ESGI
Paris, France
marion.mota@outlook.fr

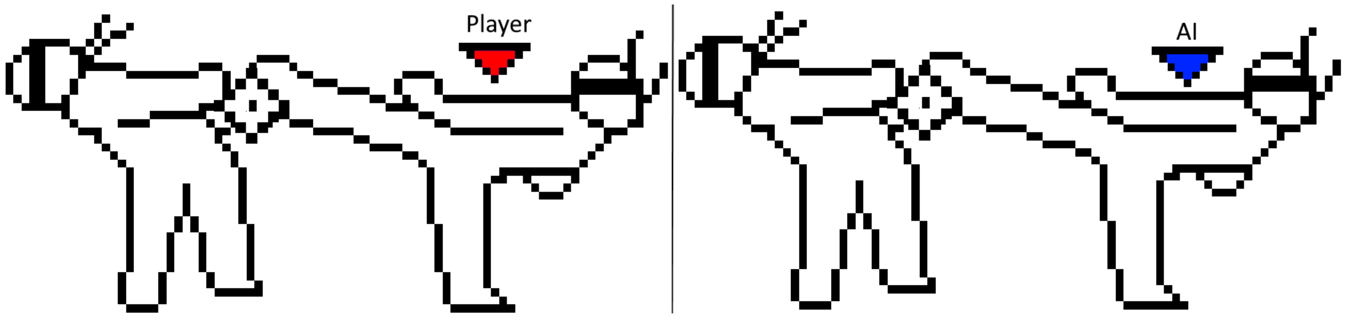

Figure 1: AI imitating player behavior

ACM Reference Format:
Lior Diler, Sami Dalil, and Marion Mota. 2022. Hybrid Prediction for Games' Rollback Netcode. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '22 Posters), August 07-11, 2022.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3532719.3543199

1 INTRODUCTION

The online implementation for fighting games has evolved greatly recently. The fighting game genre needs efficient netcode implementation, because inputs are very precise and even a latency of a few frames will affect gameplay significantly. Of the two major netcode implementations on the market, one is of particular interest. Rollback Netcode, a recent invention and currently the most popular implementation, can be seen as the future for online fighting games. This technique attempts to emulate a more pleasant experience by not delaying the inputs of the players. Instead, when the connectivity is lost for one of the combatants, the netcode tries to naively predict the inputs of the momentarily disconnected player by assuming that the missing inputs are the same as the last received actual player input, as shown in the figure below.

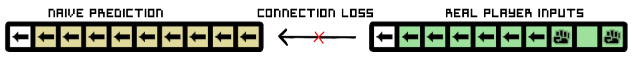

Figure 2: Naive rollback prediction of a player's inputs

Once the link is reestablished, the netcode will compare the real inputs to the ones predicted. If a difference between them affects the state of the game, it rollbacks, meaning it reverts to a previous game state and applies the now received inputs of the player. This procedure is barely noticeable in most cases allowing for a much smoother experience for the players. However when the connection loss is for more than a few frames, the forecast is less likely to be correct and the rollbacks become much more noticeable.

We have come up with a solution to improve the efficacy of this prediction aspect: Hybrid Rollback Netcode.

2 PREREQUISITES

To implement Rollback Netcode, a certain amount of prerequisites must be met. First, the game must run on both machines and be synchronised with a lockstep system. This system ensures that both games are constantly in the same game state. Since player devices must run in continuous synchronisation and the data must be sent in real time, only the inputs of the players are sent. Then, we execute the chosen commands on the other player's machine. Therefore, the game must be deterministic. This means that each time a predetermined input is applied at a chosen game state, the outcome will be the same. It also means that the inputs cannot be floating values, which are estimates. Second, the game must have complete control of the game state. This means the game state must be fully encapsulated and serializable.

3 RELATIVE WORKS

Upon doing research about Rollback Netcode, we have found interesting articles on the topic.

Martin Huynh and Fernando Valarino in their work "An Analysis of Continuous Consistency Models in Real Time Peer-to-Peer Fighting Games" [Huynh and Valarino 2019], proves that rollback netcode is overall the best netcode for user experience. Since the current naive prediction has a decent likelihood of failing and resulting in a rollback, attemping to make the prediction better is very relevant. This is why Anton Ehlert in his article "Improving Input Prediction in Online Fighting Games" [Ehlert 2021], tries to improve the prediction of Rollback Netcode with machine learning using the sliding window method [Dietterich 2021]. They trained the model used with player inputs coming from replays. They then emulated connection loss and saw if the forecasts are more accurate than those of naive input prediction. They concluded that compared to naive input prediction, the estimations of the input via intelligent prediction are more efficient but create false positives. These are much more noticeable for the players than wrong predictions from the naive technique. While the success rate on the predictions is better, it was not enough of an improvement to compensate for false positives.

This paper inspired us to come up with our own answer to solve this issue while also improving the success rate of predictions.

4 HYBRID ROLLBACK

The goal of Hybrid Rollback is to mitigate the problem of false positives from intelligent input prediction while improving the success rate of the calculations. This will naturally reduce the likelihood of false positives from occurring. To do that, Hybrid Rollback Netcode has a unique model for each player on a specific character. Compared to the model presented by Anton Ehlert [Ehlert 2021], this machine learning model is more specific. It is able to pick out more patterns depending on the player and how they play each character. The figure below represents a successful prediction of the player's inputs.

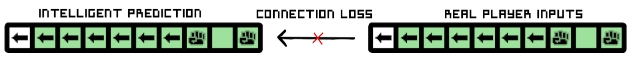

Figure 3: Intelligent prediction preventing a rollback

However, since the model will need to assemble player-specific data rather than draw from prepared generic data, it needs a substantial amount of games played by the user to perform well. Until the AI model has enough data to be efficient, the Hybrid Rollback Netcode employs the naive input prediction system instead. While it is using the naive input prediction, it emulates rollbacks and compares the intelligent prediction with the actual inputs of the players. Once the success rate is high enough, it transitions to the intelligent input prediction. This is the innovating point of the Hybrid Rollback Netcode: the ability to switch between prediction methods depending on which one is optimal.

4.1 Edge cases

There are scenarios that Hybrid Rollback Netcode needs to consider. For example, if a user is playing on another player's account. Since the data is specific to the player, the predictions are likely to be wrong. The Hybrid Rollback method needs to detect this by emulating rollbacks and noticing that the predictions are suddenly wrong. Similarly, the player might improve and change the way they play. In these cases, despite a high average success rate on the calculations, the Hybrid Rollback Netcode will transition back to a naive input prediction system and start training the model with the new data.

5 IMPLEMENTATION

Our goal is that Hybrid Rollback Netcode can be implemented in any game that uses rollback. This is why we plugged our method into the most popular, open source library GGPO [Cannon 2006] for rollback implementation. It handles the connection and packet loss between the users and the naive input prediction. We decided to start our implementation on an open source fighting game called Footsies[HiFight 2018]. The first step was to implement an AI focusing solely on predicting what a player would do in any situation.

5.1 Retrieving data and model training

For the model to learn, we collected all the data that constitutes the game state which a player can perceive. For the AI to be more realistic, we retrace the game state across multiple frames because a player reacts to past events in order to choose their action. We recorded data of one specific player against various other players in a dataset. With it, we trained a model with classification algorithms. In order to get the best results, we cleaned its content. This process involves making the data structures as simple as possible. It also implies detecting correlations between various data.

5.2 Analysis on the model's results

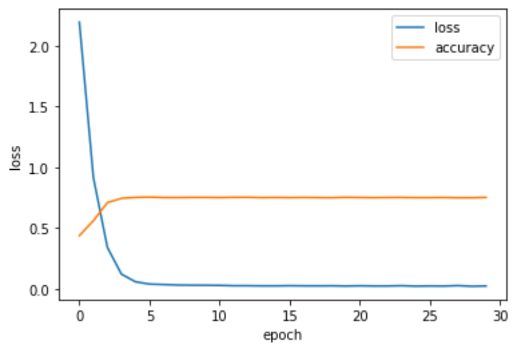

Figure 4: Evolution of the model's accuracy and loss

Our model for Footsies[HiFight 2018] has reached an average accuracy of 75% while diminishing loss values across generations. The results can improve with more preprocessing on the dataset with the previously mentioned methods and more data collected.

6 FUTURE WORKS

Currently we have only implemented our solution one game. We would like to implement hybrid prediction on other more complex games to further test the viability of our method. Another major part of the future works revolves around generically and automatically testing the quality of the data to train the models.

REFERENCES

Tony "Ponder" Cannon. 2006. GGPO. https://github.com/pond3r/ggpo
Dietterich. 2021. Learning for Sequential Data: A Review. (2021).
Anton Ehlert. 2021. Improving Input Prediction in Online Fighting Games. (2021).
HiFight. 2018. Footsies. https://github.com/hifight/Footsies
Martin Huynh and Fernando Valarino. 2019. An Analysis of Continuous Consistency Models in Real Time Peer-to-Peer Fighting Games. (2019).