Increasing Player Performance and Gaming Experience in High Latency Setups

DAVID HALBHUBER, University of Regensburg, Germany
NIELS HENZE, University of Regensburg, Germany
VALENTIN SCHWIND, Frankfurt University of Applied Sciences, Germany

Cloud gaming services and remote play offer a wide range of advantages but can inherent a considerable delay between input and action also known as latency. Previous work indicates that deep learning algorithms such as artificial neural networks (ANN) are able to compensate for latency. As high latency in video games significantly reduces player performance and game experience, this work investigates if latency can be compensated using ANNs within a live first-person action game. We developed a 3D video game and coupled it with the prediction of an ANN. We trained our network on data of 24 participants who played the game in a first study. We evaluated our system in a second user study with 96 participants. To simulate latency in cloud game streaming services, we added 180 ms latency to the game by buffering user inputs. In the study we predicted latency values of 60 ms, 120 ms and 180 ms. Our results show that players achieve significantly higher scores, substantially more hits per shot and associate the game significantly stronger with a positive affect when supported by our ANN. This work illustrates that high latency systems, such as game streaming services, benefit from utilizing a predictive system.

CCS Concepts: • **Human-centered computing** → **User studies**.

Additional Key Words and Phrases: latency, games, artificial neural networks, user performance

ACM Reference Format:
David Halbhuber, Niels Henze, and Valentin Schwind. 2021. Increasing Player Performance and Gaming Experience in High Latency Setups. *Proc. ACM Hum.-Comput. Interact.* 5, CHI PLAY, Article 283 (September 2021), 20 pages. https://doi.org/10.1145/3474710

1 INTRODUCTION

Cloud game streaming services, such as Google's *Stadia* [20] or Blade's *Shadow* [6], offer gamers a variety of advantages compared to conventional gaming platforms. The entire computing load is borne by the provider's server. This server provides remote play and renders the game via video stream to the players. The local computer only has to display the received video stream, which significantly reduces the hardware requirements. Players do not have to constantly worry about upgrading their gaming PC to meet the latest game requirements [47]. Additionally, gamers do not have to install games on their own devices – games are pre-installed on the server and playable almost instantly.

While cloud streaming has advantages over conventional gaming systems it simultaneously entails some drawbacks. Due to their architecture and mechanism streaming services for games

Authors' addresses: David Halbhuber, david.halbhuber@ur.de, University of Regensburg, Regensburg, Germany; Niels Henze, niels.henze@ur.de, University of Regensburg, Regensburg, Germany; Valentin Schwind, valentin.schwind@acm.org, Frankfurt University of Applied Sciences, Frankfurt, Germany.

2573-0142/2021/9-ART283 $15.00
https://doi.org/10.1145/3474710

cause higher latency than conventional gaming platforms, such as a gaming PC or a video console. Popular game magazines such as *EuroGamer* [16] and *PCGamer* [37] report an average latency of 183 ms for Google's game streaming service *Stadia*. Latency is defined as the time between the user's input and a response to this input by the system [51]. While streaming a game, the user's input needs to be sent to the streaming server via the Internet. The cloud gaming server receives the input, calculates reactions to the input, renders the game, and sends the screen back to the player as a video stream. However, the need for remote control of games via the Internet not only cause latency but also reduces the players' performance and their game experience [11, 15, 36]. This is particularly the case for fast-paced live-action games, which depend on timely user inputs. Split-second decision-making and interaction is essential to perform well in such games [7]. Moreover, as game streaming services have a higher latency than conventional gaming platforms, they can lead to a systematic disadvantage for players using such services in competitive settings [12].

Previous work offers two paths to account for latency in game-based streaming: (1) system-based reduction and (2) game-based compensation. System-based latency reduction aims to reduce latency by improving the communication between client and server. This is achieved by enhancing and adapting the underlying network protocols [38, 48] or by converting and compressing the exchanged data [26]. Game-based latency compensation methods, on the other hand, integrate latency directly into their game mechanics. *Time Warp* [5] and *Geometrical Manipulation* [30] are two examples of game-based latency compensation approach. Both methods compensate latency by directly integrating it into the game loop, thus when calculating game events latency is factored in. Despite efforts to compensate for latency, either by system-based or game-based approaches, current commercial game streaming services still are afflicted by latency, which negatively influences player performance and game experience. Poor performance and dissatisfied users ultimately lead to users abandoning the service. This has economic ramifications for the providers, up to financial bankruptcy [50].

Cloud gaming has the potential to trigger a paradigm shift in the gaming industry. The ability to play cutting-edge AAA productions from anywhere and from any device is within reach. However, providers and game developers need to find a way to eliminate the inherent disadvantages of game streaming in order to compete with traditional gaming systems.

Previous research, not only in the field of human-computer interaction (HCI), emphasizes the capabilities of predictions based on machine learning models [21, 29, 42] and shows that video games benefit from even small reductions in latency [31]. This work closes the gap between cloud game streaming services and conventional gaming, by compensating latency using models based on artificial neural networks (ANNs) that predict the player's avatar movement. In this work, we developed a real-time 3D video game to record player input and derive ANNs that are able to predict a player's movement in the game. We, then, simulate latencies corresponding to the delays of current streaming services and test these ANNs in the same real-time gaming environment to predict the recent avatar position and orientation within the current game state. In a conclusive evaluation, we found evidence that prediction-based latency compensation using ANNs significantly enhances player performance and gaming experience in high latency setups. Our study shows that latency-based delays in real-time games can be reduced and the gaming experience improved using only a few parameters given by the player's avatar. Due to expected advances in the field of deep learning and to enable future work to benchmark and further improve predictive models, we provide all data to replicate the herein presented work. This includes the game, the developed model, all source codes, as well as all the gathered, and anonymized user data[1].

[1] https://github.com/david-halbhuber/gameprediction

2 RELATED WORK

A growing body of work investigates latency, its effects on users, and means of preventing or compensating delays. In this section, we first provide an overview of how latency in HCI arises and which problems users experience through latency. Then, we discuss approaches to compensate latency in HCI based on ANNs. We also discuss and conclude why they are promising approaches to compensate latency specifically while using game streaming services.

2.1 Latency in Human-Computer-Interaction

The analysis of latency and latency-induced effects is deeply anchored in HCI research. Card [7] describes how latency negatively influences the interaction between a user and a system. In his work, the author shows that users interact with interactive systems in a continuous feedback loop. The users initiate the interaction by entering data into the target system, for example by clicking on buttons in user interfaces (UIs). The system receives and processes the input and reacts to it with an output. The user, in turn, can now react to this loop and potentially starts another loop cycle. Latency in interaction increases the loop throughput and, thus, increases the time required to complete a task.

Basically, latency in HCI is considered as the delay between a user's input and a noticeable output of a system [33]. Based on this, Wimmer et al. [51] state that *end-to-end* latency is mainly formed by three different partial latencies: (1) *input latency*, (2) *processing latency*, and (3) *output latency*. *Input latency* is the delay between a user's input and its receiving at a target system. *Processing latency* is the delay between the system receiving an input, processing it, and passing it on for display. *Processing latency* includes several sub-latencies, such as network latency or disk latency. *Output latency* consists of the passed time between the finished processing of an input and its actual display to users. These different latencies have different causes. *Input* and *output latency* is primarily caused by the used external equipment such as keyboards, mice, and displays. *Processing latency* is made up of communication between in- and output of the target system, as well as of the target system's processing performance.

Current work demonstrates that latency in HCI leads to diminished user performance in interactive tasks. Jota et al. [24] and Annett et al. [2] demonstrate that latency above 25 ms leads to a decreased user performance. They did not find an improvement in user performance for latency below 25 ms. The authors conclude that users perform best at 25 ms latency. Although performance does not increase below 25 ms latency, Ng et al. [35] found a just noticeable difference (JND) - users are able to perceive latency starting at a value of 2 ms. Based on this, Ng et al. [34] show in a later work that users are even able to perceive differences between 1 ms latency and 2 ms latency in certain tasks.

Video games and players can be affected by latency as well. While Claypool and Claypool [10] show that gaming performance in certain types of game is susceptible to latency, Long and Gutwin [32] use latency to predict performance. Negative effects of latency manifest in different forms. Players score fewer points, need more time to complete tasks, or cannot solve tasks at all [4, 15]. Those negative effects of latency in video games become increasingly apparent when players use game streaming services, such as Google's *Stadia* [20] or Blade's *Shadow* [6]. Recently, popular game magazines such as *EuroGamer* [16] or *PCGamer* [37] report an average latency of 183 ms for Google's game streaming service *Stadia*. These values are higher than the optimal latency threshold of 25 ms proposed by Jota et al [24]. This leads to a systematic disadvantage for players using game streaming services [12] compared to playing locally, which can have negative impacts e.g. in multiplayer scenarios.

2.2 Compensating Latency Through ANNs

ANNs have been used to compensate for latency in various settings. In principal, they reduce emergent latency by predicting user inputs. By generating this prediction the supported system does not have to wait for the actual input and can consequently start computing an anticipated output or action earlier. Compared to a system without latency compensation, this creates a temporal advantage. Conversely, this reduces the latency perceived by the user. Thus, for instance, Henze et al. [21] show an approach for latency compensation for touch devices by predicting the users' touch input. In a user study, the authors show that they can increase the users' performance in a *Fitt's Law* task [17]. Furthermore, the authors show in three different tasks, that prediction using neural networks is more precise than linear and polynomial extrapolation. Le et al. [29] refine this approach, by coupling the ANN with supporting information based on inertial measurement units (IMUs) worn by the user. Conclusively, their network is able to predict users' touch input as well as the acceleration values of the users' hands. Using this additional data the authors are able to reduce latency-based effects and also improve the results by Henze et al. [21]. In recent work, Schwind et al. [42] even use neural networks to compensate latency in motion tracking based virtual reality (VR). The authors show that users of high latency systems, such as a motion tracking based VR systems, benefit from input prediction based on ANNs. Although Schwind et al. were not able to improve user performance, they improved the perceived accuracy of the body location in VR, which can ultimately lead to an increased user experience, particularly in game settings. To the best of our knowledge, latency compensation based on ANNs has not been researched in real-time streaming of games.

2.3 Compensating Latency in (Cloud) Gaming

Related work highlights two paths to account for latency in cloud game streaming services: (1) system-based latency reduction and (2) game-based latency compensation. System-based latency reduction aims to reduce the occurring latency by improving the communication between client and server. This is achieved by enhancing and adapting the underlying network protocols [38, 48] or by manipulating and compressing the exchanged data [26]. Game-based latency compensation methods on the other hand compensate latency by manipulating game mechanics. *Time Warp* [5] and *Geometrical Manipulation* [30] are two examples of game-based compensation methods. *Time Warp* compensates latency in multiplayer games by taking it into account when calculating game events. To determine, for instance, if a player hit a selected target under a certain refresh rate, the game estimates the actual time of the user's input rather than the time the input was received by the game. *Time Warp* is widely used in online first-person view (FPV) games but often considered to be unfair by gamers, since it always favors the actor [25]. Through *Geometrical Manipulation* game objects' geometrical dimensions are manipulated in dependencies of latency. This aims to lower the game's difficulty to account for an increased error rate through a heightened latency. Despite efforts to reduce or compensate latency, either by system-based or game-based approaches, current commercial game streaming services still are afflicted by latency which negatively influences player performance and game experience.

2.4 Summary

Recent work shows that latency down to 1 ms is (negatively) perceived by users [34, 35] and can affect user performance in interaction above 25 ms [2, 24]. Latency in video games leads to players scoring fewer points, needing more time to complete takes, or not being able to solve tasks at all [4, 10, 15]. Considering the generally higher latency in game streaming service [8], players using them are disadvantaged compared to using conventional gaming systems [12]. Compensating

latency using ANNs is researched in various settings and has proven its feasibility to increase user performance and user experience [21, 29, 42]. ANNs have not been used to compensate for latency in high latency video game scenarios or game streaming, yet. Currently, it is unknown how users of high latency game streaming services can be enabled to achieve low-level-latency performance and game experience. In this work, we investigate if ANNs in a data-driven approach can be utilized to compensate latency in a cloud gaming scenario.

3 REDUCING LATENCY THROUGH ARTIFICIAL NEURAL NETWORKS

Similar to previous work [29, 42], we followed a data-driven approach to compensate latency using ANNs in a simulated high latency system setup. To test if ANNs are able to compensate for latency, we designed and developed a FPV game to ensure full control about our research environment. We implemented various functions to gather game data, such as avatar position, avatar orientation, and a number of in-game metrics. Secondly, we gathered training data for the ANN in a preliminary data acquisition study. The participants were instructed to play the game. In-game data was recorded using self-developed logging functions. Next, we trained deep learning based algorithms using the data set and tested the capability of in-game avatar prediction for latency compensation. We optimized the model's parameter to minimize the loss on a separate test set. Finally, we evaluated our prediction model in a user study with 96 participants. Again, the participants were instructed to play the game, while we recorded in-game data and user experience.

3.1 Game Development

One requirement for the game and the user study was barrier-free access for potential participants during the pandemic of COVID-19. To meet this requirement, we conducted a data acquisition study and a user study in this work exclusively online instead of in a laboratory setting. Although, this decreases the internal validity and control of local phenomena, such as input latency, it emphasizes the ecological validity using an in-the-wild approach. To be able to conduct our data acquisition and user studies online, we used the browser as the game's target platform. As the game is fully hardware-accelerated, all processing and rendering is performed locally on the players' computers.

Since players of FPV games react particularly sensitive to latency [10] we selected them as the target group of our research. Next, we designed and developed the game world and the avatar in *Unity3D* (Version 2019.f16.2), additionally, we implemented a *First-Person-Controller* for the player to be able to control the avatar within the game world. The game world is divided into three parts: The first two parts comprise an introductory tutorial (basic controls and shooting), which allows the player to familiarize themselves with all avatar controls and the UI. The third part of the game world is the actual game, in which the player is situated in a jungle environment with enemies spawning at five different portals. Figure 1 shows an aerial view of the implemented game world – the different game parts are highlighted. Additionally, Figure 2 shows the player's view while playing in the game arena.

The player's objective during the game is to shoot a fixed number of hostile monsters before they reach the player's avatar. If a monster reaches the player, the player loses points from a pool of limited life points. In case the player has been hit four times by the enemies, the game was over and restarted in the third game section. If the player manages to shoot all enemies, the player reaches the next game level while earning points equivalent to the number of enemies shot. To keep the player motivated while playing, we gradually increase the game's difficulty with each level by increasing the number of enemies and the enemy's speed. We used a database backend to log all game events, such as avatar position and orientation, points, hit counts, and positional information.

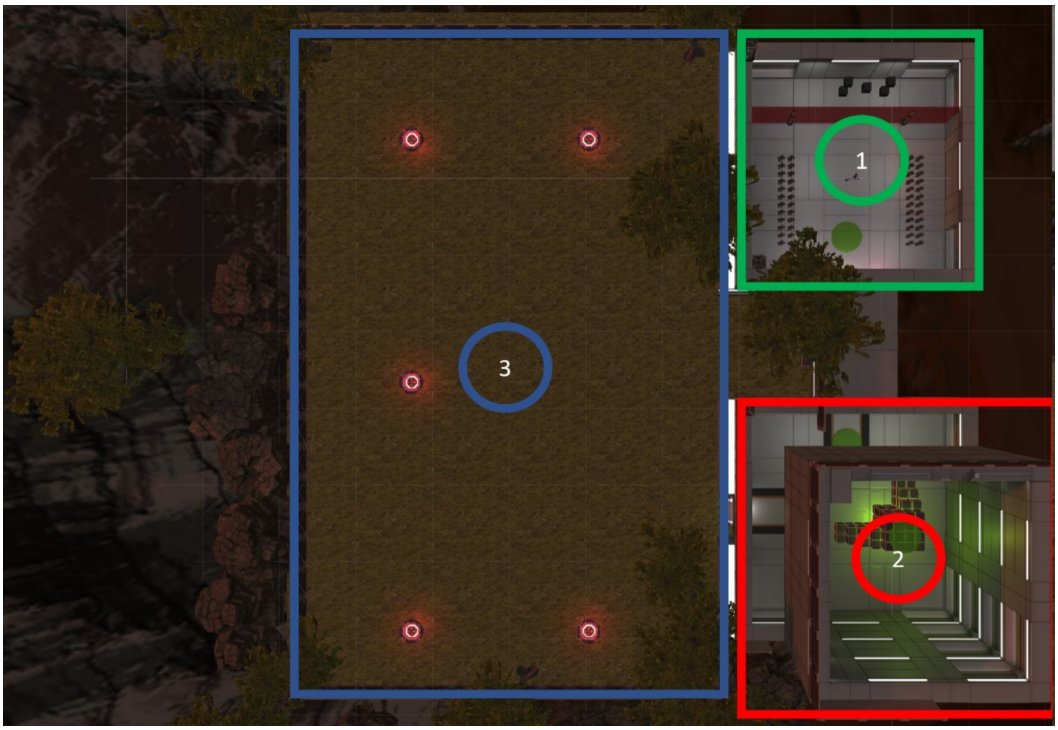

Fig. 1. Shows a top view of the designed game world. The different game areas are color-coded. The green and red game area were parts of the tutorials (cf. Tutorial A and Tutorial B). In the blue game area, the players have to compete against the AI opponents.

3.2 Game Data Collection

We conducted a data acquisition study to collect the in-game data necessary for training an ANN capable of predicting avatar movement in real-time.

3.2.1 Apparatus. For the study, we hosted the game on a publicly reachable web server. Participants played the game on their own devices in a browser of their choice. It was not necessary to download game files or manually install the game on the player's devices. Content delivery is fully handled by the web server and does not require any additional user input.

3.2.2 Procedure and Task. After entering the website, participants were presented with a consent form including the purpose of the study. After giving informed consent to data collection, the participants could move on to the game start screen. Participants were aware of the data collection, but not of the precise purpose of the collection or the exact type of data collected. The study received ethical clearance via the ethics policy of our institution. By clicking on the start button the participants started playing the game. All further instructions for the study were handled directly in the game via a user interface containing a dialog box. The players were guided by the game through Tutorial A (basic controls) and Tutorial B (shooting). Upon reaching the third game area the participant's goal was to fight and survive the enemies waves, while simultaneously obtaining as many points as possible. After 800 seconds of playtime, the game automatically closed and ended the session.

Fig. 2. Shows the players' perspective while playing in the third game area.

3.2.3 Participants. We recruited 24 participants (9 female, 14 male, 1 preferred not to say) evenly from two sources: (1) mailing list of our institute and (2) crowd-sourcing via *prolific.co* (N = 12). Both groups were compensated either with credit points for their study course (1) or £3 as monetary compensation (2). A total duration of 20 minutes has been estimated for participation. Their average age was 25.8 years (SD = 6.0), with the age ranging from 18 years to 41 years.

3.3 Model Development

The gathered data has been processed in three steps to gain a model for a prediction of the player's input and movement: (1) pre-processing of the data, (2) using the data for training deep learning models, and (3) integrating the developed model into the game.

3.3.1 Data Pre-Processing. We logged a total of 602 943 unique samples from the participants in the data collection study. The data has been divided into three categories: (1) continuous frame data, (2) event-based data, and (3) system specifications. The majority of the data consists of continuous frame data. One frame consists of the following data points: (1) frames passed since game start, (2) seconds passed since game start, (3) X-, (4) Y-coordinates of the current raw mouse position in pixel, (5) X-, (6) Y-, (7) Z-coordinates of the current avatar position as well as (8) X-, (9) Y- and (10) Z-coordinates of the current avatar orientation. We recorded a total of 522 165 frame samples. On average, participants generated 21 756 frames (SD = 8.625) in each gaming session. 80 788 samples (80 764 events, 24 system specifications) were generated from event-based data and system specification logs. Table 1 shows all logged events in our study. Participants fired their weapons a total of 20 060 times (M = 835.6, SD = 800.5) while fighting against 3 228 enemies. In total, participants played for 19 200 seconds (5.3 hours) with an average frame rate of 30.2 FPS (SD = 4.9).

For this study, we trained solely on continuous frame data of each player. In detail, each row of the final training data set maps one frame consisting of the following eight data points: (1-2) raw

Event ID	Event Description
1	Player fired.
2	Player hit enemy with shot.
3	Player collected item.
4	Player hit by enemy.
5	Player lost all life points.
6	Game was restarted.
7	Player completed tutorial.
8	Player finished current wave.

Table 1. Shows the recorded event ID and the corresponding event. Data is being recorded when a player triggers an event.

mouse coordinates in X, Y, (3-5) avatar coordinates in X, Y, Z, (6-8) avatar orientation in X, Y, Z. To increase prediction accuracy we enlarged the prediction baseline. Instead of inferring based on one frame our ANN uses five successive frames to predict future avatar position and orientation. Finally, we performed the training of the ANN based on 40 unique input values using this approach.

3.3.2 Latency Determination. To determine the predictive values of the model we determined typical latency values in game streaming applications. We based our prediction values on current latency measurements of popular game magazines such as *EuroGamers* [16] and *PcGamer* [37]. The elaborated latency of Google's *Stadia* based on six different reports averages at 183 ms (SD = 103 ms). Although the standard deviation of the reported measures is high, we choose to use the mean as the upper bound of the artificially added latency, based on a trifold reasoning. (1) Technical advancements in cloud gaming are leading to ever-reducing latencies. An optimistic choice with low latencies corresponds better to a real-world scenario in ongoing improvement rather than the choice of conservative high latencies. (2) Ng et al. [34] already showed that user's perceive latency down to 1 ms, following the authors findings it is clear that lower latency values play an important role in HCI. Choosing a higher upper artificial latency limit, would have lead to neglecting and blurring of the lower values. (3) Since previous work already has proven that high latency values highly influence game experience and performance, we choose to investigate lower latency values. Which rendered us able to perform a finer graded and more detailed compensation via the ANN prediction. This, in turn, allows for more detailed analyse of the added latency, the ANN prediction and the effects on participants playing without compensation technique. Consecutively, we defined the maximum prediction value of our system to be at 180 ms (rounded down). Thus the trained model is able to compensate for 180 ms latency in a high latency environment such as a game streaming service.

3.3.3 ANN Training and Model Development. The ANN's goal is to compute the avatar position and orientation in 180 ms based on the given training data set. Crucial for choosing the ANN architecture was the time needed for predicting the next output. Since the output had to be generated and merged with the game in real time, the duration for inference had to be minimized. The prediction must not interfere with or slow down the execution of the gameloop. Considering typical game frame rates from 30 to 60 frames per second (FPS), inference and merge had to be done within 33 ms

to 16 ms. Thus, we choose a lightweight ANN implementation based on a *Multistep-Dense-Network* architecture [49].

The network's first layer (L_1 = 40 neurons) is used to transform multi-dimensional input data into a one-dimensional array of scalars. This layer is followed by four fully connected hidden layers(L_2=1024 neurons, L_3 = 128 neurons, L_4 = 128 neurons, L_5 = 1024 neurons). The number of hidden layers and the number of neurons implemented in each layer were determined using *Grid Search*. The last hidden layer passes the processed data to the last layer – the output layer (L_6 = 8 neurons). To account for overfitting, we added a drop out function between the last hidden layer and the output layer [45]. The output layer in turn outputs the avatars orientation and position in the chosen prediction value, e.g. 180 ms. To account for non-linearity we solely used *Rectified Linear Unit* (ReLU) [19]. We optimized the data using *Adaptive Movement Estimation* (ADAM) [27] with a batch size of 64 samples, a learning rate of 0.1 and a loss implementation based on the mean square error (MSE). Through *Keras* callback function, we dynamically changed the learning rate in the training to enable the underlying back-propagation method in our ANN to deal with local optimization minima. Figure 3 shows the structure of our ANN.

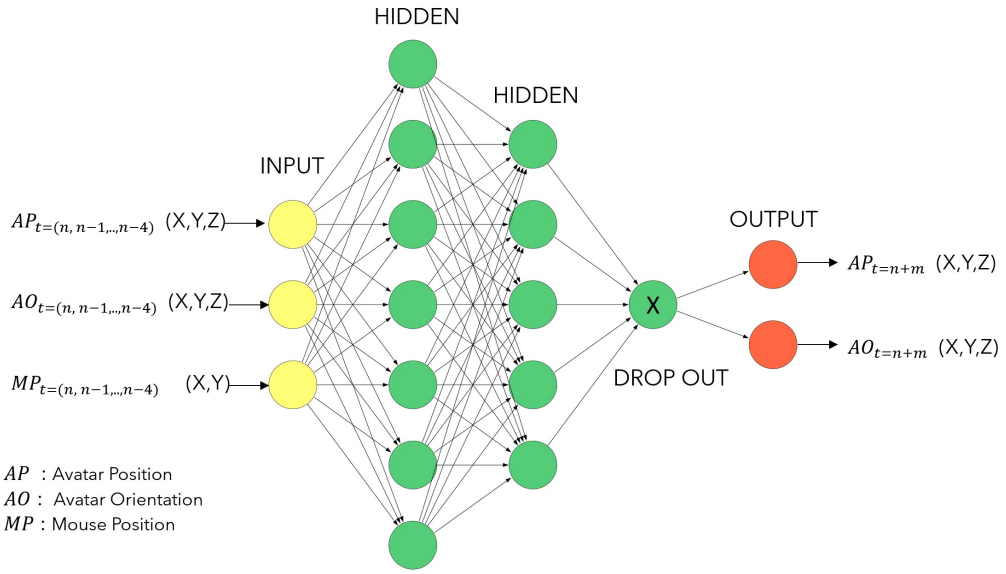

Fig. 3. Depicts the conceptual structure of our artificial neural network (ANN) used for latency compensation. Due to size limitations, not all layers and neurons are depicted. We input the current and the four most recent($t = (n, n-1, .., n-4)$) avatar positions, orientations as well as mouse positions. The input (yellow) reduces the multi-dimensional input to a one-dimensional tensor. Consecutively, the networks' hidden layers (green) follow, which are connected to a drop out before feeding to the output layer (red), which outputs avatar position and orientation for the desired moment of time ($t = n + m$).

We trained additional prediction values aside from the 180 ms forecast. The additional values are factors of 180 ms making their effect comparable to the original 180 ms prediction. Thus, we trained the model to predict 60 ms, 120 ms, and 180 ms in the "future". We trained each prediction mode for 45 epochs until no further improvement in loss optimization was observable. Loss was calculated using Unity's Worldspace Coordinates (UWC) and the MSE. The final losses were 19.24 UWC for the *+60 ms* model, 42.31 UWC for the *+120 ms* model and 48.32 UWC for the *+180 ms* model. The

training was performed on a PC with Windows 10, AMD Ryzen 7 1800X, 64 GB RAM, and two NVIDIA GeForce GTX 1080 TIs with a total of 22 GB VRAM. Training the ANN took about 4.4 hours for each prediction mode, totaling in a overall training time of 13.2 hours.

3.3.4 Model Integration. To predict avatar position and orientation, we integrated the model in the environment of the game. We used Tensorflow.JS [1, 43] to serve the model online. Inferences from the model can be requested directly from the game via *JavaScript*. For inference, the model needs the previously defined 40 input values. We implemented various game functions to cache avatar position, avatar orientation, and mouse position. Once the caching function saved five frames, the game sends a request for inference to the model, waits for the result, and applies the received values directly in the game by updating the avatar position and orientation. Consecutively, the game repeats this process for every frame, discarding the last cached frame and adding the current frame to the rolling cache. All these processing steps are performed within one game loop, i.e. within a single frame, thus the model inference does not increase the game's execution time. Figure 4 shows the prediction pipeline.

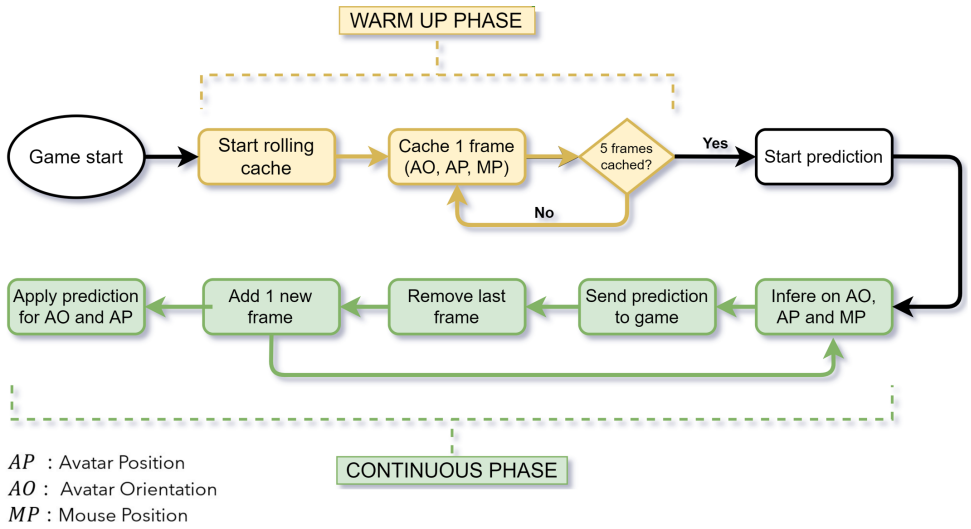

Fig. 4. Depicts a flow diagram of the prediction pipeline. With the start of the game the warm up phase starts (orange) as well. In warm up the pipeline catches five consecutive frames. After five frames the warm up phase ends and the continuous phase (green) starts. In this phase the pipeline inferes on Avatar Position (AP), Avatar Orientation (AO) and Mouse Position (MP). Then the prediction is sent back to the game. Before applying it in the game, the oldest frame in the cache is replaced with the current frame, which triggers a new prediction iteration.

4 EVALUATION OF THE SYSTEM

To determine the effects of our prediction model we conducted a user study. In the study, we aimed to remove latency-based effects in a high latency system by predicting the avatar's position and orientation in our game. We inflicted our game with 180 ms artificial latency by buffering user input, to create a latency condition similar to those in modern game streaming settings.

4.1 Method

We conducted an online user study to test the hypotheses that different Prediction Times impact game experience and player performance. We used Prediction Time as between-subject variable. In addition to a baseline model with (1) *0 ms* prediction, which is achieve by completely disabling the ANN prediction, we tested three different ANN models: (2)*+60 ms*, (3)*+120 ms* and (4)*+180 ms*. The four levels of Prediction Time were evenly distributed and randomly assigned to the participants.

We collected data about player performance and game experience for each participant. Player performance is measured in three dependent variables: (1) *MaxScore* - Maximum number of points achieved, (2) *HitCoef* – hit-shot-ratio and (3) *EnemyHit* – number of enemy hits. We used the maximum number of points achieved instead of the average score, because we wanted to measure peak performance instead of average performance. To measure game experience we used the *Game Experience Questionnaire* (GEQ) [22], with its sub-scales competence (COM), sensory score (SEN), flow score (FLO), tension score (TEN), challenge score (CHA), negative affect (NEG) and positive affect (POS).

4.1.1 Apparatus. The apparatus was similar to the one used for data collection. We, again, hosted the game on a publicly reachable web server. Participants played the game on their own devices in a browser of their choice. In dependence of their assigned conditions players either played the *0 ms* baseline game or were supported by one of the three ANN forecast models (*+60 ms*, *+120 ms* or *+180ms*). Resulting in the following four conditions: (1) *0 ms*, (2) *+60 ms*, (3) *+120 ms* and (4) *+180 ms*.

4.1.2 Procedure and Task. On entering the web page, participants were presented with a consent form. After giving informed consent, participants were able to start the game. Participants were aware of and agreed to data collection but were not aware that they tested different ANN models. Consequently, they did not know that they were being assisted in their gaming session by a neural network (*+60 ms, +120 ms, +180 ms)*. The study received ethical clearance as per the ethics policy of our institute. All further study instructions were handled directly in the game via a user interface. The game guided the participants through the tutorial. Upon reaching the third game area the participants goal was to survive against the enemies, while simultaneously obtaining as many points as possible. After 800 seconds of playtime, the game automatically closed the play session, and opened the final questionnaire within the browser window.

4.1.3 Participants. We recruited 96 participants (23 female, 70 male, 3 preferred not to say) through the crowd sourcing platform *Prolific.co*. Thus, each condition was tested using a total of 24 participants. Participants who participated in our data collection study could not attend. Participants were compensated with £3. A total duration of 20 minutes had been estimated for participation. The average age of the participants was 24.6 years (SD = 5.7), with the age ranging from 18 years to 51 years.

4.2 Results

In total we recorded 2 652 949 unique samples from the participants. The major part is the continuous frame data, which sums up to a total of 2 297 526 unique frames. On average, participants generated 23 933 frames per session (SD = 9 487 frames). Event-based data, as well as the unique system specification data, add up to 355 423 samples (35 5327 events and 96 system information). During the conducted user study, participants fired their gun a total of 51 533 times ($M = 536.80$, $SD = 635$). The participants fought against a total of 13 732 enemies. The game was played by the participants for 86 373 seconds (23.99 hours/25.9 min per participant) in all conditions.

4.2.1 Maximum Score (MaxScore). Before further analysis we checked *MaxScore* for normal distribution using the Shapiro-Wilk test. Neither data from *0 ms* (W = 0.76, p = <.001), *+60 ms* (W = 0.81, p = <.001), *+120 ms* (W = 0.77, p = <.001) nor data from *+180 ms* (W = 0.86, p = <.001) is normally distributed. Concluding, a Kruskal-Wallis test showed a significant effect of Prediction Time ($\chi^2(3) = 17.59$, p = <.001) on the maximum game score of the players. Pairwise comparison using Wilcoxon signed rank test showed, in combination with a Bonferroni α correction, a significant difference between *0 ms* and *+120 ms* (W = 114, p = .005) as well as between *0 ms* and *+180 ms* (W = 107.5, p = <.001). Mean *MaxScore* values, as well as p values are depicted in Figure 5. Increasing the prediction increased player performance. Players performed best in the *+180 ms* condition.

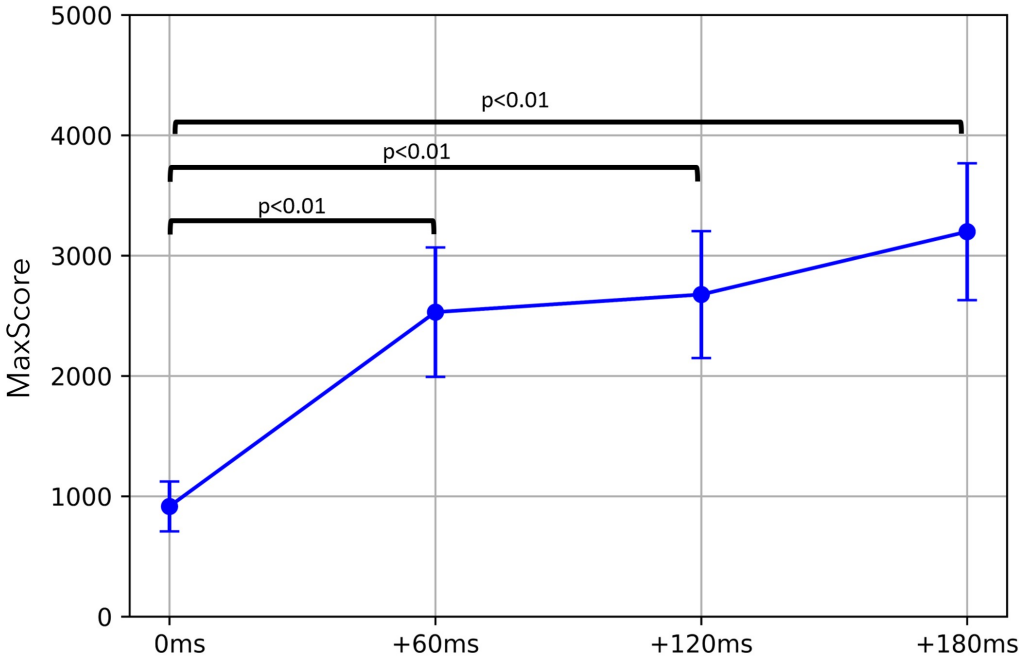

Fig. 5. Shows the evaluation of the MaxScore variable. Illustrates the average maximum score achieved by all players over all conditions. Significant differences are marked at the corresponding places via p-bars. Significant differences between *0 ms* and *+120 ms* and between *0 ms* and *+180 ms* could be determined after Bonferroni correction. The error bars show the standard error.

4.2.2 Shot-to-Hit Ratio (HitCoef). Shapiro-Wilk test showed that the gathered data from *0 ms* (W = 0.94, p = 0.23), *+60 ms* (W = 0.96, p = 0.39) and *+120 ms* (W = 0.98, p = 0.39) follows a normal distribution. Data from the *+180 ms* condition does not fit a Gaussian distribution (W = 0.89, p = 0.01). Since not all *HitCoef* data is parametric, we choose to use Kruskal-Wallis test again. The test showed a significant effect of Prediction Time ($\chi^2(3) = 34.39$, p = <.001) on the shot-to-hit ratio of the players. Pairwise comparison using Wilcoxon signed rank test showed, in combination with a Bonferroni α correction, a significant difference between *+120 ms* and *0 ms* (W = 50, p < .001), *+120 ms* and *+60 ms* (W = 53, p < .001) and *+120 ms* and *+180 ms* (W = 76.0, p < .001). Mean *HitCoef* values, as well as p values are depicted in Figure 6 (left). Increasing the prediction up to 120 ms increased players shot-to-hit ratio. Predicting 180 ms did not increase the *HitCoef* further. Player performed best in the *+120 ms* condition.

Shapiro-Wilk Test GEQ Categories				
	0 ms	+60 ms	+120 ms	+180 ms
COM	W = 0.96, p = 0.44	W = 0.95, p = 0.44	W = 0.96, p = 0.56	W = 0.96, p = 0.66
SEN	W = 0.92, p = 0.06	W = 0.87, **p = <0.01**	W = 0.92, p = 0.06	W = 0.92, p = 0.07
FLO	W = 0.96, p = 0.57	W = 0.93, p = 0.15	W = 0.92, p = 0.09	W = 0.98, p = 0.92
TEN	W = 0.92, p = 0.07	W = 0.91, **p = 0.05**	W = 0.88, **p = <0.01**	W = 0.95, p = 0.24
CHA	W = 0.95, p = 0.25	W = 0.94, p = 0.17	W = 0.96, p = 0.46	W = 0.94, p = 0.15
NEG	W = 0.96, p = 0.58	W = 0.93, p = 0.09	W = 0.91, **p = 0.05**	W = 0.90, **p = 0.02**
POS	W = 0.89, **p = 0.01**	W = 0.95, p = 0.26	W = 0.97, p = 0.68	W = 0.93, p = 0.15

Table 2. Shows the evaluation of the Shapiro-Wilk test for the different questionnaire categories (Competence (COM), Sensory (SEN), Flow (FLO), Tension (TEN), Challenge (CHA), Negative Affect (NEG) and Positive Affect (POS)) and conditions (*0 ms*, *+60 ms*, *+120 ms* and *+180 ms*).

4.2.3 Hit by Enemies (EnemyHit). Using Shaprio-Wilk we determined that data from condition *0 ms* ($W = 0.92$, $p = 0.08$) and condition *+120 ms* ($W = 0.94$, $p = 0.24$) is normally distributed. Data from condition *+60 ms* ($W = 0.82$, $p = <.001$) and *+180 ms* ($W = 0.91$, $p = 0.03$) is not normally distributed. We did not find significant differences in the times a player got hit by enemies ($\chi^2(3) = 3.73$, $p = 0.29$) using the Kruskal-Wallis test. However, looking at Figure 6 (right), a general trend is recognizable, but yet to be evidently proven. Figure 6 (right) shows the mean values of *EnemyHit*. Increasing Prediction Time did not significantly decrease the time players got hit by enemies.

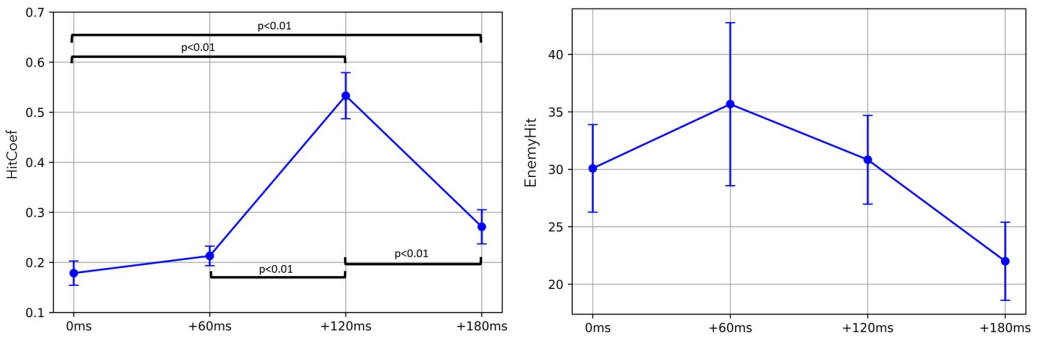

Fig. 6. Shows the evaluation of the shot-to-hit coefficient (HitCoef) (left) and enemy hits (EnemyHit) (right) measure. The left diagram illustrates the average hit quotient achieved by all players across all conditions. Significant differences are shown at the corresponding places via p-bars. Significant differences could thus be determined after Bonferroni correction between *0* ms and *+120 ms*, *+120 ms* and *+60 ms*, and *+120 ms* and *+180 ms*. The error bars show the standard error. The right diagram illustrates the average number of hits that opponents have scored at the end of the game. Despite a noticeable downward trend, no significant differences in the different distributions could be found. The error bars show the standard error.

4.2.4 Game Experience Questionnaire. Based on the authors' recommendations, we evaluated each category of the Game Experience Questionnaire (GEQ) individually [22], starting with an analysis for normal distribution. Table 2 shows the evaluation of the Shapiro-Wilk test on the different questionnaire categories.

Oneway ANOVA for parametric, independent data showed no significant differences in the categories competence ($F(3) = 1.14$, $p = .32$), flow ($F(3) = 0.74$,$p = .52$) and challenge ($F(3) = 0.25$, $p = .85$). Kruskal-Wallis statistical test for non-parametric data showed no significant differences in the categories sensory ($\chi^2(3) = 1.99$, $p = 0.57$), tension ($\chi^2(3) = 5.02$, $p = 0.17$) and negative affect ($\chi^2(3) = 2.79$, $p = 0.42$). The test revealed significant difference in the category positive affect ($\chi^2(3) = 12.16$, $p = <.001$).

Further analysis via Wilcoxon test showed significant differences between the baseline *0 ms* condition and the *+120 ms* condition ($W = 118.5$, $p < 0.01$). Increasing Prediction Time to 120 ms, significantly increased the experienced positive feelings and emotions that player perceived or experienced while playing. Participants assigned the *+120 ms* condition the highest positive affect score.

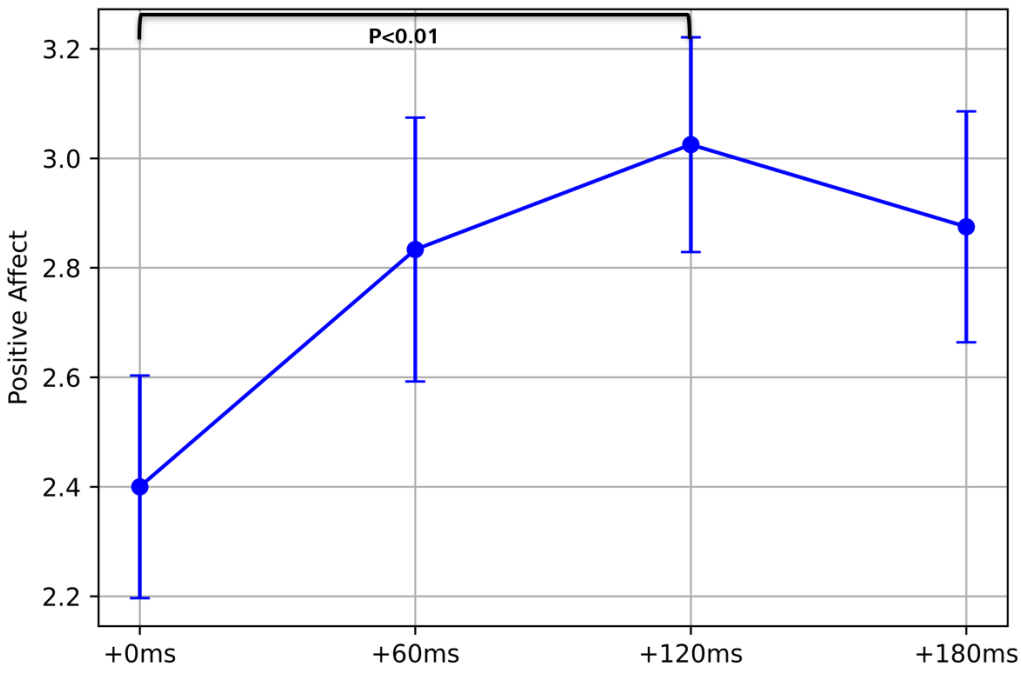

Fig. 7. Shows the evaluation of the positive affect score of the game experience questionnaire. Illustrates the average rating submitted by all players overall conditions. Significant differences are marked at the corresponding places via p-bars. Significant differences between *0 ms* and *+120 ms* could be determined after Bonferroni correction. The error bars show the standard error. Players significantly stronger associated the game with a positive affect in the *+120 ms* condition compared to the *0 ms* condition.

4.3 Discussion

Our results indicate that Prediction Time has the highest impact on *MaxScore* with maximum prediction. Generally, we showed that the participants achieve higher *MaxScore* values in the game with all predictions. The highest game scores were achieved in the *+180 ms* condition. These results are in accordance with the latency/performance limits of Claypool et al. [13]. According to the authors, FPS games are most affected by latency. Claypool et al. state that games of this genre

experience negative effects starting at a latency value of 100 ms. Furthermore, the authors explain that a further reduction of latency does not lead to a significant improvement in user performance. We also did not find a significant increase in performance by further decreasing latency. We found no significant difference in *MaxScore* between *+60 ms*, *+120 ms* and *+180 ms*.

Considering the average values for *HitCoef*, we showed that participants using the *+120 ms* condition achieved the highest shot-to-hit-ratio. Consequently, latency compensation of more than 120 ms did not lead to a further improvement of the shot-to-hit ratio. *HitCoef* ratio can be equated with the accuracy of the participants. Participants are more accurate when a larger number of opponents are hit with fewer shots. Despite the decreasing accuracy in the *+180 ms* condition, we show that the accuracy of the participants is significantly improved by our predictive system. These results are also consistent with the work of Beigbeder et al. [4]. The authors show that latency-induced effects reduce the accuracy of players by up to 50%. Our work shows that these effects can be compensated by predicting avatar position and orientation. The compensation of latency significantly increases the accuracy of participants in systems with latency. This is generally caused by the reduced input/display offset. By predicting, the participant gets a more direct reaction to the input action than in a system without prediction. We conclude that due to the real-time representation in the game, participants have more time to place the crosshairs more consciously and consequently more accurately, which ultimately increases the shot-to-hit-ratio.

The third analyzed variable is *EnemyHit*. The variable describes how often the participants were hit by opponents on average during a game session. We found no significant difference for *EnemyHit*, although a descending trend can be recognized when looking at the mean values of *EnemyHits*. Thus, the variable first rises slightly in the comparison of condition *0 ms* and condition *+60 ms*, but then falls continuously over condition *+120 ms* to its minimum average value in condition *+180 ms*. The missing significant differences in *EnemyHit* can be attributed to the circumstance that the number of enemy hits depends strongly on the ingame position of the avatar. Important is the tactical and anticipatory behavior of the participants. Since the opponents attack from any side, the participants benefit more strongly from anticipatory gameplay than from a latency-free environment. Our ANN cannot support the tactical positioning of the avatar.

We found significant differences in the positive affect category of the GEQ. All other categories showed no significant differences. Positive affect describes the feelings and emotions experienced by the players during the game. These feelings and emotions include, for example, joy, pleasure, and the subjectively perceived fun of playing [22]. Participants in the study rated the condition with 120 ms prediction with the highest positive affect score. Participants in this condition most strongly associated their experience with the game with fun. Positive affect, in particular the enjoyment of an activity, has a systematic and positive influence on the performance during this activity [3]. This is also shown by the in-game metrics discussed above. Participants in the *+120 ms* condition achieved better performance scores and experienced more fun while playing than participants in all other conditions. Therefore we argue, that using a system to compensate for latency significantly increases the game experience by increasing the perceived positive affect.

Game experience is crucial for the success of any game. The failing of countless high production-value examples, such as *Anthem* [18], *Fallout 76* [28] and *Warcraft 3: Reforged* [9], illustrated the effects of neglecting game experience in the game design process. In this regard, our work showed, as prior work did [4, 15, 41], that perceived latency negatively influences game experience and performance. Additionally our work presents a solution to this quandary – prediction-based latency compensation. Our findings, and the resulting implication, are relevant to game streaming providers, game developers, and researchers as well.

Providers of game streaming services must continue their endeavors to improve existing infrastructures. Server availability and density must be further increased so as to further reduce the

resulting network latency. Optimizing network latency will result in lower end-to-end latency for gamers, which in turn will improve game experience. In addition, carriers should work closer with game development studios to enable them to account for the expected latency ranges in the design phase. Ideally, game streaming services should allow developers to incorporate the method presented in this paper. In doing so, providers could offer an interface through which a prediction for latency compensation is trained and integrated - individually for each deployed game.

Game developers, on the other hand, need to be aware of the difference between traditional gaming platforms and game streaming services. Developing a game for a traditional system should not be equally approached as developing a game for a streaming services. Although developers use different tools to adapt the game to different platforms, latency is not taken into consideration. Adaption is carried out exclusively on a technical level. Future game developments should consider possible latency already in the design phase. Thus, latency-sensitive game sections should be adapted in such manner that they become more robust against latency. Besides adjusted development, game studios should also consider the possibility of a prediction-based latency compensation in their game. In this case, game developers could incorporate a prediction directly into the game, similar to the method presented in this paper. The prediction should react adaptively to the current latency level and thus ultimately enable a latency-independent game experience.

Finally, the findings of our work and the presented method is relevant to other researchers as well. Not only researchers in the field of video games but researcher in HCI generally, possibly profit from the presented method. Although, we used game specific parameters to train our model it is likely that the presented approach is suitable for any kind of software operated by mouse and keyboard. For example, by integrating mouse prediction, a software could achieve higher responsiveness and thus enhance the perceived user experience. Naturally, other researchers in the gaming domain could evaluate the method shown with other commercially available or self-developed games.

5 CONCLUSION

In this paper, we present a novel approach to compensate for high latency in video games using ANNs. We predict the position and orientation of the in-game avatar in a self-developed 3D video game. This prediction effectively reduces the necessary time to process a feedback loop between player and game, since our system anticipated the player's next move and implements it in the game before the actual user input is received.

The prediction of our system is based on data of 24 participants playing a self-developed first-person video game. This data was collected in a data collection study and includes information about avatar position and orientation, the position of enemies as well as some event-based information, such as data about firing behavior and the current score and maximum score of the players. After data collection, we trained an ANN to predict changes in avatar position and orientation based on past frames. We trained three different level of predictions: *+60 ms*, *+120 ms* and *+180 ms* – the highest value is based on the current latency of commercial game streaming services. After training, we conducted a second user study with 96 participants to validate the approach.

In our second study, we showed that players in conditions of *+120 ms* and *+180 ms* prediction achieved significant higher game scores than in the other conditions. Additionally, we showed that by using the *+120 ms* model players achieved higher accuracy values compared to all other conditions (*0 ms, +60 ms, +180 ms*). In addition to the increased scores and accuracy, participants rated the *+120 ms* condition with significantly higher scores in the positive affect category of the GEQ. In summary, we were able to show that negative latency-based effects, such as performance degradation, can be compensated by our system. Ultimately, enabling players to achieve low-latency performance and game experience.

5.1 Limitation and Future Work

Our ANN is trained by data from our game. This allows us to directly implement model inference into the game, but simultaneously limits the usage of the ANN to said game. A more general approach to train and implement an ANN-based prediction could be used to evaluate our method in commercially available video games. This approach, although, would need to be implemented on the operating system level, since most commercial video games do not allow interference from a third party application. An exception to this are games with an open modding culture. These games could easily be enabled to integrate a prediction based latency compensation, by providing the model data to the modding community.

Another shortcoming can be found in the trained prediction values - 60 ms, 120 ms, and 180 ms. We solely trained discrete fixed timings, but latency in real-world applications is not represented by a single discrete value but a dynamic range of values. In the context of our work assuming discrete latency is valid to fully controlled the environment but future work should focus on developing and investigating adaptive systems that adjust the prediction to latency measured in real-time.

The used ANN architecture in this paper is based on commonly used deep learning architectures. A more sophisticated and complex model may be able to generate more precise predictions and further improve our results. We optimized the ANN until we were satisfied with the achieved loss, our results show that this approach can already improve players' performance and experience. The architecture and the model parameters can, however, be further optimized – for example, the number of hidden layers could be increased when more training data is available. We encourage further research and provide all data and the model to the public. This enables future work beyond latency research. For example, a deeper analysis of the collected data sets can provide further insights into the behavior of gamers. In-depth studies could, for example, show how gamers behave in certain situations, e.g., how they react to stress-inducing scenarios such as being surrounded by enemies. This could shed light on whether game-internal metrics, such as shooting behavior, change depending on the situation.

Additionally, future work could provide insight into whether our presented method is valid in different gaming context, such as networked multiplayer games. In such games, different latency levels create an unjust situation. Players with high latency are more disadvantaged by latency than players with lower latency. Especially in competitive esports, players often travel far to compete with other players under identical and controlled conditions. High latency can make the difference between virtual survival and death and consequently can cost the players millions in prize money [14, 46]. An adaptive system, similar to the presented system, could bring the latency of all gamers to a common denominator and thus creating a fair gaming environment for competitive players.

In a manner similar to that presented in this work, future work could verify if ANNs can be used to compensate local input latency. Local input latency is the time that passes before the effect of an input is displayed on an output device. Previous work showed that input latency may reach values of up to 243 ms [23]. As with network latency, differences in input latency lead to imbalanced or unsatisfactory game play. This has created, especially in the last few years, a market for low-latency input and output devices specifically designed for gaming. Gaming hardware often comes at a much higher price tag [39, 40] then their counterparts - creating a situation in which spending money equals better performance or experience in video games. ANNs that specialize in minimizing local input latency could compensate for these socioeconomic differences as they arise, promoting a fair and fun gaming experience for all players alike [44].

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://doi.org/10.5555/3026877.3026899 Software available from tensorflow.org.

[2] Michelle Annett, Fraser Anderson, Walter F. Bischof, and Anoop Gupta. 2014. The Pen is Mightier: Understanding Stylus Behaviour While Inking on Tablets. In *Proceedings of Graphics Interface 2014* (Montreal, Quebec, Canada) *(GI '14)*. Canadian Information Processing Society, CAN, 193–200. https://doi.org/10.5555/2619648.2619680

[3] F Gregory Ashby, Alice M Isen, et al. 1999. A neuropsychological theory of positive affect and its influence on cognition. *Psychological review* 106, 3 (1999), 529. https://doi.org/10.1037/0033-295X.106.3.529

[4] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. 2004. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games* (Portland, Oregon, USA) *(NetGames '04)*. Association for Computing Machinery, New York, NY, USA, 144–151. https://doi.org/10.1145/1016540.1016556

[5] Yahn W Bernier. 2001. Latency compensating methods in client/server in-game protocol design and optimization. In *Game Developers Conference*, Vol. 98033.

[6] BLADE. 2020. Shadow - Verwandle deine Geräte in einen Gaming PC. https://shadow.tech/. Accessed on 2020-08-23.

[7] Stuart Card. 1981. The model human processor: A model for making engineering calculations of human performance. In *Proceedings of the Human Factors Society Annual Meeting*, Vol. 25. SAGE Publications Sage CA: Los Angeles, CA, 301–305.

[8] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. 2011. Measuring the latency of cloud gaming systems. *MM'11 - Proceedings of the 2011 ACM Multimedia Conference and Co-Located Workshops*, 1269–1272. https://doi.org/10.1145/2072298.2071991

[9] Video Games Chronicle. 2020. Warcraft 3: Reforged is now the worst user scored game ever on Metacritic. https://www.videogameschronicle.com/news/warcraft-3-reforged-is-now-the-worst-user-scored-game-ever-on-metacritic/. Accessed on 2021-07-05.

[10] Mark Claypool and Kajal Claypool. 2006. Latency and Player Actions in Online Games. *Commun. ACM* 49, 11 (Nov. 2006), 40–45. https://doi.org/10.1145/1167838.1167860

[11] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 2016. The Effects of Delay on Game Actions: Moving Target Selection with a Mouse. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts* (Austin, Texas, USA) *(CHI PLAY Companion '16)*. Association for Computing Machinery, New York, NY, USA, 117–123. https://doi.org/10.1145/2968120.2987743

[12] Mark Claypool and David Finkel. 2014. The effects of latency on player performance in cloud-based games. In *2014 13th Annual Workshop on Network and Systems Support for Games*. 1–6. https://doi.org/10.5555/2755535.2755538

[13] Mark Claypool, David Finkel, Alexander Grant, and Michael Solano. 2014. On the performance of OnLive thin client games. *Multimedia systems* 20, 5 (2014), 471–484. https://doi.org/10.1007/s00530-014-0362-4

[14] EarlyGame. 2020. Remote esports – is it sustainable? https://www.earlygame.com/remote-esports-is-it-sustainable/. Accessed on 2021-07-05.

[15] Ragnhild Eg, Kjetil Raaen, and Mark Claypool. 2018. Playing with delay: With poor timing comes poor performance, and experience follows suit. In *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6. https://doi.org/10.1109/QoMEX.2018.8463382

[16] Eurogamer. 2020. Doom Eternal on Stadia looks great - but the lag is just too high. https://www.eurogamer.net/articles/digitalfoundry-2020-doom-eternal-stadia-looks-the-part-but-lag-is-too-high. Accessed on 2020-08-21.

[17] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381. https://doi.org/10.1037/h0055392

[18] Forbes. 2019. Why 'Anthem' Failed And Why It Was Never Destined To Succeed. https://www.forbes.com/sites/erikkain/2019/05/30/why-anthem-failed-and-why-it-was-never-destined-to-succeed/. Accessed on 2021-07-05.

[19] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[20] Google. 2020. Stadia - Game Streaming. https://stadia.google.com/. Accessed on 2020-08-23.

[21] Niels Henze, Markus Funk, and Alireza Sahami Shirazi. 2016. Software-reduced touchscreen latency. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 434–441.

[22] Wijnand A IJsselsteijn, Yvonne AW de Kort, and Karolien Poels. 2013. The game experience questionnaire. *Eindhoven: Technische Universiteit Eindhoven* (2013), 3–9.

[23] Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. 2015. Quantifying and mitigating the negative effects of local latencies on aiming in 3d shooter games. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 135–144.

[24] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How Fast is Fast Enough? A Study of the Effects of Latency in Direct-Touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2291–2300. https://doi.org/10.1145/2470654.2481317

[25] S. W. K. Lee and R. K. C. Chang. 2017. On "shot around a corner" in first-person shooter games. In *2017 15th Annual Workshop on Network and Systems Support for Games (NetGames)*. 1–6. https://doi.org/10.1109/NetGames.2017.7991545

[26] Taekhyun Kim, Swagat Mohapatra, Mukta Gore, and Alok Ahuja. 2014. Latency reduction by sub-frame encoding and transmission. US Patent App. 13/728,296.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Kotaku. 2019. Bug-Riddled Update Shows Why Fallout 76 Needs A Public Test Server. https://kotaku.com/bug-riddled-update-shows-why-fallout-76-needs-a-public-1836457946. Accessed on 2021-07-05.

[29] Huy Viet Le, Valentin Schwind, Philipp Göttlich, and Niels Henze. 2017. PredicTouch: A System to Reduce Touchscreen Latency Using Neural Networks and Inertial Measurement Units. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 230–239.

[30] Injung Lee, Sunjun Kim, and Byungjoo Lee. 2019. Geometrically compensating effect of end-to-end latency in moving-target selection games. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12. https://doi.org/10.1145/3290605.3300790

[31] Shengmei Liu, Mark Claypool, Atsuo Kuwahara, James Scovell, and Jamie. 2021 Sherman. 2021. Lower is Better? The Effects of Local Latencies on Competitive First-Person Shooter Game Players. In *CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, 285–297. https://doi.org/10.1145/3411764.3445245

[32] Michael Long and Carl Gutwin. 2018. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play* (Melbourne, VIC, Australia) *(CHI PLAY '18)*. Association for Computing Machinery, New York, NY, USA, 285–297. https://doi.org/10.1145/3242671.3242678

[33] I Scott MacKenzie and Colin Ware. 1993. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 488–493.

[34] Albert Ng, Michelle Annett, Paul Dietz, Anoop Gupta, and Walter F. Bischof. 2014. In the Blink of an Eye: Investigating Latency Perception during Stylus Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1103–1112. https://doi.org/10.1145/2556288.2557037

[35] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-Latency Direct-Touch Input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 453–464. https://doi.org/10.1145/2380116.2380174

[36] James Nichols and Mark Claypool. 2004. The Effects of Latency on Online Madden NFL Football. In *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (Cork, Ireland) *(NOSSDAV '04)*. Association for Computing Machinery, New York, NY, USA, 146–151. https://doi.org/10.1145/1005847.1005879

[37] PCGamer. 2019. Here's how Stadia's input lag compares to native PC gaming. https://www.pcgamer.com/heres-how-stadias-input-lag-compares-to-native-pc-gaming/. Accessed on 2020-08-21.

[38] Andreas Petlund, Kristian Evensen, Pål Halvorsen, and Carsten Griwodz. 2008. Improving application layer latency for reliable thin-stream game traffic. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*. 91–96.

[39] Razer. 2021. RAZER DEATHADDER V2 PRO. https://www2.razer.com/eu-en/store/razer-deathadder-v2-pro. Accessed on 2021-07-05.

[40] Razer. 2021. RAZER HUNTSMAN V2 ANALOG - US. https://www2.razer.com/eu-en/store/razer-huntsman-v2-analog/. Accessed on 2021-07-05.

[41] Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Babak Naderi, Carsten Griwodz, and Sebastian Möller. 2020. A latency compensation technique based on game characteristics to mitigate the influence of delay on cloud gaming quality of experience. In *Proceedings of the 11th ACM Multimedia Systems Conference*. 15–25.

[42] Valentin Schwind, David Halbhuber, Jakob Fehle, Jonathan Sasse, Andreas Pfaffelhuber, Christoph Tögel, Julian Dietz, and Niels Henze. 2020. The Effects of Full-Body Avatar Movement Predictions in Virtual Reality using Neural Networks. In *26th ACM Symposium on Virtual Reality Software and Technology*. 1–11.

[43] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. 2019. Tensorflow. js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350* (2019).
[44] Josef Spjut. 2021. A Case Study of First Person Aiming at Low Latency for Esports. (2021).
[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
[46] Statista. 2021. DOTA 2 The International championships prize pool from 2011 to 2019. https://www.statista.com/statistics/749033/dota-2-championships-prize-pool/. Accessed on 2021-07-05.
[47] Jiawei Sun and Mark Claypool. 2019. Evaluating Streaming and Latency Compensation in a Cloud-based Game. In *Proceedings of the 15th IARIA Advanced International Conference on Telecommunications (AICT)*.
[48] Wendell Sun. 2019. Adaptive bitrate streaming latency reduction. US Patent 10,432,982.
[49] TensorFlow. 2015. TensorFlow. https://www.tensorflow.org/tutorials/structured_data/time_series. Accessed on 2020-10-21.
[50] PCMag UK. 2015. OnLive Streaming Service Shutting Down at End of April. https://uk.pcmag.com/gaming-systems/40903/onlive-streaming-service-shutting-down-at-end. Accessed on 2021-04-28.
[51] Raphael Wimmer, Andreas Schmid, and Florian Bockes. 2019. On the Latency of USB-Connected Input Devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

Received February 2021 ; revised June 2021 ; accepted July 2021