Using Artificial Neural Networks to Compensate Negative Effects of Latency in Commercial Real-Time Strategy Games

David Halbhuber
david.halbhuber@ur.de
University of Regensburg
Regensburg, Germany

Maximilian Seewald
maximilian.seewald@stud.uni-regensburg.de
University of Regensburg
Regensburg, Germany

Fabian Schiller
fabian.schiller@stud.uni-regensburg.de
University of Regensburg
Regensburg, Germany

Matthias Götz
mathias.goetz@stud.uni-regensburg.de
University of Regensburg
Regensburg, Germany

Jakob Fehle
jakob.fehle@ur.de
University of Regensburg
Regensburg, Germany

Niels Henze
niels.henze@ur.de
University of Regensburg
Regensburg, Germany

ABSTRACT

Cloud-based game streaming allows gamers to play *Triple-A* games on any device, anywhere, almost instantly. However, they entail one major disadvantage - latency. Latency, the time between input and output, worsens the players' experience and performances. Reducing the latency of game streaming is crucial to provide gamers the same game experience as in local gaming. Previous work demonstrates that deep learning-based techniques can compensate for a game's latency if the artificial neural network has access to the game's internal state during inference. However, it is unclear if deep learning can be used to compensate for the latency of unmodified commercial video games. Hence, this work investigates the use of deep learning-based latency compensation in commercial video games. In a first study, we collected data from 21 participants playing real-time strategy games. We used the data to train two artificial neural networks. In a second study with 12 participants, we compared three different scenarios: (1) playing without latency, (2) playing with 50 ms of controlled latency, and (3) playing with 50 ms latency fully compensated by our system. Our results show that players associated the gaming session with less negative feelings and were less tired when supported by our system. We conclude that deep learning-based latency compensation can compensate the latency of commercial video games without accessing the internal state of the game. Ultimately, our work enables cloud-based game streaming providers to offer gamers a better and more responsive gaming experience.

CCS CONCEPTS

• **Human-centered computing** → *User studies*; • **Applied computing** → **Computer games**.

MuC '22, September 4–7, 2022, Darmstadt, Germany

ACM ISBN 978-1-4503-9690-5/22/09.
https://doi.org/10.1145/3543758.3543767

KEYWORDS

video games, latency, latency compensation, real-time strategy games, deep learning

ACM Reference Format:
David Halbhuber, Maximilian Seewald, Fabian Schiller, Matthias Götz, Jakob Fehle, and Niels Henze. 2022. Using Artificial Neural Networks to Compensate Negative Effects of Latency in Commercial Real-Time Strategy Games . In *Mensch und Computer 2022 (MuC '22), September 4–7, 2022, Darmstadt, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543758.3543767

1 INTRODUCTION

Cloud-based game streaming (CGS) using services, such as Amazon's *Luna* [3] or Nvidia's *Geforce Now* [34], provide gamers multiple advantages compared to conventional local gaming. In CGS, all complex computations, such as physics or AI pathfinding, are realized by a cloud server. The server renders the game and provides it to the gamers via a video stream. The local device, such as a computer or a mobile phone, merely has to display the received video stream and send the gamer's input to the remote server. This streaming technique significantly reduces the hardware requirements for the local device. Thus, gamers do not need to constantly upgrade their system to play the latest *Triple-A* [46] games [41]. Current CGS systems even allow gamers to benefit from novel and sophisticated graphic techniques such as *Deep Learning Super Sampling* or *Raytracing* [35]. Furthermore, gamers do not have to download and install games on their devices as all CGS games are pre-installed on the cloud server and playable within seconds.

While CGS provides multiple advantages to gamers, due to streaming games via the Internet, CGS services are affected by a higher latency than conventional local gaming systems. Latency is the time between a user-generated input and the corresponding output of an interactive system [30, 31]. During streaming, the gamer's input to the local device is sent to the cloud server. The server receives the input, calculates the reaction to the input, renders the game, and sends the output back to the gamer as a video stream. This additional communication increases the overall latency of the game. Latency, however, is known to negatively influence game experience and player performance [11, 15, 33]. Due to the higher latency, gamers using CGS play with a systematic and

constant disadvantage. Thus, gamers, game developers, and CGS providers aim to minimize or compensate for latency. Despite all efforts, CGS systems still have a significantly higher latency of up to 180 ms more than local gaming systems [19].

Previous work highlights multiple methods to compensate for latency in video games. Liu et al. [28] distinguish four latency compensation techniques: (1) *Feedback*, (2) *Prediction*, (3) *Time Manipulation*, and (4) *World Adjustment*. While *Feedback*, *Time Manipulation*, and *World Adjustment* techniques manipulate the game world to compensate for latency [8, 18, 22, 25], *Prediction* techniques use internal game state, such as the players' position in the game world, to extrapolate a future game state. Predictive methods, thus, enable the game to start calculating an output before user input arrives. By predicting the user's actions, the overall perceived latency is reduced.

One sub-category of in-game latency compensation techniques are methods based on deep learning. While deep learning methods conceptually fit the *Prediction* category, they use Artificial Neural Networks (ANNs) instead of classical machine learning to predict users' actions. In recent work [19], ANNs have shown to be capable of reducing the adverse effects of latency on player performance and game experience in a high latency gaming setting. Halbhuber et al. [19] used an ANN to predict the position and orientation of the player's avatar in a custom game, significantly reducing the adverse effects of 180 ms latency. The authors used a custom *First-Person Shooter* (FPS) game to utilize the game's internal state during inference and to integrate the ANN's prediction into the game. This approach, however, is not possible for commercial video games as they are black boxes and usually do not allow access by third-party applications. Thus, it is unclear if ANNs can compensate for the latency of unmodified commercial video games.

In this paper, we show how to compensate for the latency of commercial video games without modifying the game or accessing the internal game state. We use ANNs to compensate for adverse effects of latency by predicting the future movement of the mouse in the *Real-Time Strategy* (RTS) game *Age of Empire 2* (AoE2) [16]. To achieve this, we conducted a data collection study with 21 participants playing AoE2 and *Empire Apart* [14]. We used the gathered data to develop, test, and benchmark two different ANNs. The first ANN predicts the future mouse position in 50 ms based on past movements. The second ANN predicts future mouse position in 50 ms based on past movement and game images. We evaluated both ANNs and found that visual data does not lead to a better prediction. Thus, our second study with 12 participants investigated the effects of the ANN predicting the mouse movement while playing AoE2 with controlled latency. Our evaluation shows that the developed ANN significantly enhanced the game experience compared to playing with high latency and no prediction. We conclude that ANNs can be used in commercial RTS games to reduce the adverse effects of latency on the game experience. CGS providers can use our approach to alleviate their services to the same level of game experience as conventional local games.

We provide all resources to enable other researchers to replicate and build upon our work via GitHub. The repository includes all developed models, all source code, and all gathered, anonymous user data[1].

2 RELATED WORK

A growing body of work investigates latency, how it arises, its effects on gamers, and how to compensate for it in video games. This section first provides an overview of how latency in interactive systems arises. Then, we discuss the effects of latency in video games. Next, we continue showcasing multiple examples to compensate for or reduce latency in video games. Finally, we conclude this section with a summary showing that ANNs can potentially be used to reduce latency in CGS.

2.1 Latency in Interactive Systems

Previous research [29] mainly separates latency into two categories: (1) local latency - latency generated by used hardware, and (2) network latency - latency generated by communicating over a network such as the Internet. Both types of latency affect user experience and user performance in different ways. Local latency inherently affects all interactive systems, while network latency is primarily found in multiplayer online video games [29].

Recent work investigating latency found that local latency leads to diminished user performance. For example, Jota et al. [21] and Annett et al. [4] found that local latency above 25 ms decreased the user performance in different tasks. While local latency below 25 ms might not decrease user performance, it is still perceivable. Ng et al. [32] found that latency starting at 2 ms can be perceived by users of a touch device. In later work, Annett et al. [5] found that the perception of latency is task-dependent. For example, in some tasks, such as inking on a tablet, users notice latency at 50 ms.

2.2 Latency in Video Games

Since they are interactive systems, local latency also negatively affects video games. In addition, online video games are negatively affected by network latency as well.

Regardless of its origin, latency negatively affects gamer performance and game experience. The adverse effects of latency on gamer performance in video games manifest in multiple ways: Gamers achieve fewer points, need additional time to complete a task, are less precise, or can not finish game objectives at all [7, 12, 15]. However, not all games and game genres are affected by latency in the same way [10]. Fast-paced games, such as FPS games, are stronger affected by latency than other games. For example, Armitage et al. [6] revealed that starting at 150 ms to 180 ms of latency, gamer performance worsens. Other work investigating game latency found that performance degradation starts at 100 ms of latency. Recent work showed that video games are negatively affected by latency starting at 25 ms [27]. Besides its effect on gamer performance, latency also decreases the game experience. For example, Liu et al. [26] found that latency of 150 ms reduces the overall quality of experience by 25 %.

[1] https://github.com/david-halbhuber/Using-ANNs-to-Compensate-Negative-Effects-of-Latency-in-RTS-Games

2.3 Latency Compensation and Reduction in Video Games

Previous work developed multiple strategies to counteract latency in video games, which can be classified differently. One classification by Liu et al. [28] structures latency compensation techniques in four classes: (1) *Feedback*, (2) *Prediction*, (3) *Time Manipulation*, and (4) *World Adjustment*. Latency compensation methods from the *Feedback* (1) class provide auditory or visual feedback to the gamer based on the current latency. These techniques do not alter the actual latency in the game. Gutwin et al. [18], for example, proposed a method in which specialized game objects signalize the presence, magnitude, and effects of latency in the game. The second category *Prediction* contains methods using the available game state data, such as the gamer's position in the game world or other objects in the game world, to interpolate or extrapolate a future game state. The proactive calculation of a future game state potentially decreases the overall perceived latency. Methods from the categories *Time Manipulation* (3) and *World Adjustment* (4) both directly alter the game world. *Time Manipulation* methods slow down, speed up or stop game time entirely to synchronize game events. One example of such a method is *Time Warp* [8, 22]. *Time Warp* is widely used in online multiplayer video games but often considered unfair by gamers since it always favors the actor [22]. *World Adjustment* methods, such as *Geometrical Manipulation* manipulate game objects depending on the amount of latency, for example, making targets bigger and thus easier to hit in a high latency setting [25].

2.4 Summary

Previous work shows that latency in interactive systems negatively influences user experience and performance starting at 25 ms [21, 32]. Video games and gamers are negatively affected by latency as well [7, 10, 12, 15]. Latency in video games leads to gamers scoring fewer points or needing more time to complete a given task. Ultimately, high latency can even prevent gamers from finishing a given task at all. Considering the higher latency in CGS, gamers using these services are at a constant disadvantage compared to gamers playing on conventional local gaming setups. Compensating high latency in custom video games using ANNs has proven feasible [19]. However, commercial video games are fundamentally different as they usually not allowed to be modified by third-party applications. Hence, while previous work showed that ANNs are, in principle, capable of reducing the negative effects of latency, it is unknown if the same approach can be used in CGS. The higher latency in CGS is one reason for its moderate adaptation rate. Thus, in this work, we investigate if ANNs can be utilized as a compensator for high latency in commercial video games in a CGS scenario.

3 DATA COLLECTION

In line with previous work [19, 39], we used a data-driven approach to develop and evaluate the presented ANNs. First, we developed custom tools to remotely record mouse movement and video output while playing any video game. In a data collection study, we then used the developed tools to collect data from 21 gamers playing the RTS games *AoE2* and *Empire Apart*. We used two different games to increase the ANNs' potential to generalize over different games.

Figure 1: Shows screenshots of the *Real-Time Strategy* games *Empires Apart* [14] (left) and *Age of Empire 2* [16] (right).Both screenshots show the game interface, some player-controlled units and buildings from a vertical bird's eye view. We used the games in a remote data collection study to obtain data suitable for training our latency compensation system. The screenshots were recorded using our data gathering tool.

3.1 Development of the Data Gathering Tool

We gathered all data to train our ANNs in the wild to maximize ecological validity. While this approach introduces some randomness, for example, when considering local latency, it simultaneously potentially increases the ANNs' generalization capability. Since the data is not collected in a laboratory setting, it contains more variation. Gathering data in the real world allows the ANNs to potentially learn a more nuanced version of gamers playing video games which is not constrained by the setting in a laboratory study.

We exclusively used *Python 3* to develop our data gathering tool. The tool consists of different methods for starting games, capturing and converting gameplay images, recording mouse inputs, multi-threaded data processing, and archiving and uploading the gathered data. For automatically installing and starting games we used *PyWinHook* [43], *PyNput* [37] and *Steam* [44]. For image capturing, resizing, and converting, we used *OpenCV2* [42]. For mouse movement recording, we utilized *PyWinHook*, and for uploading the gathered data to a remote server, we made use of *PySFTP* [36]. We created an executable from the Python script using *PyToExe* [45].

After starting the program, it automatically logs in to the game library application *Steam* with hard-coded login credentials. If our application can not find a current *Steam* installation, it tries to download and install it. After logging into *Steam*, the tool waits for user input to start the data collection. Data is only collected if an appropriate game is running and maximized. Upon ending the game session, i.e., if the game gets closed, the tool automatically finishes uploading the gathered data before closing itself.

Our tool records the current mouse position in the game every 5 ms. Mouse positions, however, are only recorded if the current mouse position changed compared to the mouse position 5 ms ago to minimize the amount of logged data and to prevent identical mouse position entries. Furthermore, the software captures, resizes (to 848 pixel by 480 pixel), and starts to upload one gameplay image every 41.67 ms (recording in 24 fps). We limited the resolution of the gameplay images to reduce the time needed for uploading the data to our server while obtaining as much visual information as possible. Recording 1 hour of gameplay still produced about 10 GB of data. Figure 1 shows two game screenshots our data gathering tool recorded.

3.2 Data Collection Study

Using the developed tool, we conducted a data collection study to acquire the necessary gameplay data to develop ANNs capable of predicting user inputs to reduce latency.

3.2.1 Apparatus. For the study, we sent our data-gathering tool to our participants. Participants played the games on their devices and thus did not have to install any software manually. The study ran automatically and did not require any additional input from the experimenter.

3.2.2 Procedure and Task. After giving informed consent to the data collection, participants received an e-mail with our tool and detailed instructions. Participants started the tool by double-clicking the received executable. After confirmation by the participants and maximization of the game, the data logging (mouse movement + gameplay images) started. Next, participants were asked to play one hour of *free play*, which is a sandbox mode in both games, allowing them to start playing right away. After playing for one hour, participants closed the game. Our tool automatically compressed and uploaded the gathered data to our remote server. The study, and thus the data collection, received ethical clearance by our institution's ethics policy.

3.2.3 Participants. Twenty-one participants (4 f, 17 m) were recruited via our institute's mailing list. Participants were selected independent of age, gender, and experience playing RTS games. All participants were students and compensated with one credit point for their course of study. The study took about 70 minutes per participant. The participants' average age was 25.3 years (SD = 3.6 years), ranging from 20 years to 36 years.

4 DEVELOPMENT OF THE ARTIFICIAL NEURAL NETWORKS

Overall we collected 1 412 802 gameplay images (due to rare transmission errors, the number of collected images does not reflect the theoretical maximum) and 3 479 488 mouse positions in our data collection study. We used the gathered data to develop two deep learning-based ANNs: (1) a *Deep Neural Network* (DNN) [24] and, (2) a mixed-model *Convolutional Neural Network* (CNN) [24] combined with a DNN. We optimized the ANNs' parameter to minimize the loss on a train-test set combination. The accuracy of both ANNs was evaluated using a separate validation set (80/10/10 split). Both ANNs' goal was to predict the mouse position in 50 ms. This prediction could be sent to the streaming server in a CGS scenario. The server receives this prediction after a certain time, for example, after 50 ms. However, since the server received a predicted position in the future, the subsequent rendering will not be affected by latency. This predictive method effectively reduces the gamer's experienced latency. We defined the prediction value for both ANNs per previous work, which shows that latency negatively affects gamers starting at 25 ms. Furthermore, Halbhuber et al. [19] showed that a prediction of 60 ms already significantly increased gamer performance and game experience. Thus, to increase the likelihood of the amount of latency and latency prediction influencing gamers, we set the prediction value to 50 ms.

4.1 General Description

Both ANNs were trained using a supervised deep learning approach. Hence, we defined the amount of data used for inference (the input) and the corresponding frame or mouse position as output in training. We used *TensorFlow* [1] to train all models and *Optuna* [2] for hyperparameter optimization. For training, we used a local server with an Intel i9-990k CPU, 16 GB Ram and two GPUs, one Nvidia RTX 2060 with 6 GB VRAM, and one Nvidia 1080Ti with 8 GB VRAM. Crucial for both ANNs was the time needed to predict the next output. Since the output of the ANNs has to be applied to the game in real-time, the duration for inference had to be minimized. The prediction must not slow down the interaction with the game. Since typical frame rates in games range from 30 to 60 frames per second (FPS), generating the next output and merging it back into the game had to be finished within 16 ms to 33 ms. We optimized the *Mean Absolute Error* (MAE) between the actual delta value of the mouse position in 50 ms and the predicted value for all models.

4.2 Training the Dense Neural Network

The first ANN uses the last ten mouse positions to predict a delta value for the mouse position in X and Y coordinates in 50 ms. Since we recorded one mouse position every 5 ms, the prediction is based on the data of the last 50 ms. Hence, the ANN's input consists of 10 pairs of X and Y coordinates. The network consists of five fully connected layers. The first layer L_1 - the input layer, has 20 neurons and passes the input to the first of three fully connected hidden layers (L_2 = 64 neurons, L_3 = 32 neurons, L_4 = 16 neurons). The last layer L_5 - the output layer - has two neurons and serves the predicted value of the mouse position in 50 ms in X and Y coordinates. We used *Adaptive Movement Estimation* (ADAM) [23] with a batch size of 128 samples and a learning rate of 0.001 for backpropagation. As activation function we used *TensorFlow*'s built-in *Rectified Linear Unit* (ReLU) implementation [17]. We used *Early Stopping* [38] to prevent overfitting and custom callback functions to deal with local optimization minima. The model was trained for 100 epochs, with one epoch taking approximately 36 minutes. In evaluation, the model achieves an MAE of 12.4 pixels per coordinate on the unknown validation set. Considering a full HD monitor with a resolution of 1920 pixels by 1080 pixel the ANN creates a prediction with an error of less than 1 % on the X-axis and about 1.2 % on the Y-axis. Generating one inference takes about 1.6 ms, which is fast enough considering our requirement of a prediction time of less than 16 ms.

4.3 Training the mixed-model Convolutional Neural Network

The second ANN uses five past game images and the corresponding mouse positions to predict the mouse position in 50 ms. Thus, the prediction is based on gameplay from the past 208 ms. The dense part of the network is identical to the first presented ANN. The convolutional part of the model consists of seven layers. L_1, a fully connected dense layer, receives the screenshot in the recorded resolution (848 px X 480 px) and scales it down by factor 10 to 84 px X 48 px. Additionally, L_1 grey-scales the input images. Layer L_2 the first convolution layer (neurons = 64, stride = $(3, 3)$, padding = 3) is followed by a *Max Pooling* layer (pool = $(2, 2)$). L_4 (convolution,

neurons = 32, stride = (3, 3), padding = 3) and L_5 (*Max Pooling*, pool = (2, 2)) follow the same structure as L_2 and L_3. The next layer, L_6, is a *Flatten* layer, which receives the multi-dimensional output from the previous layer and transforms it into a one-dimensional array. In the next step, this array is passed to a dense layer (L_7, neurons = 16), which generates X, Y coordinates predicting the mouse curser's position. The output of the convolutional and the network's dense part are concatenated in a final dense layer. We, again, used ADAM with a learning rate of 0.1 and a batch size of 4 samples as well as ReLU to optimize. We utilize *Drop Out* (drop out rate = 0.2) [40] to prevent overfitting of the model. The second ANN was trained for ten epochs, with one epoch taking 36 hours to train. After training, the ANN achieved an MAE error of 18.8 px in predicting the mouse position in 50 ms. Again, considering a full HD monitor the MAE corresponds to an error of less than 1 % on the X-axis and about 1 8% error on the Y-axis. The model finishes one inference in 5.4 ms. Figure 2 shows an overview of the developed model (left) and an architecture plot of the convolution branch of the model (right).

We did not find that using gameplay images increases the prediction accuracy. Thus, we used the DNN for further investigation.

4.4 Integration of ANN to the Game

To predict the mouse position in 50 ms using the developed model, the model needs the means to communicate with the game. Since AoE2 is a commercial game, it is impossible to manipulate the game or the current game state directly. Hence, we used the Python library *PyNPut* to override the mouse position on the operating system (OS) level. Using this approach, the game never receives the actual mouse hardware value but only the predicted values of our model. Upon starting the inference pipeline, the ANN waits for ten mouse positions to be received before starting to pipe the inference to the game. Via rolling cache procedure, the ANN removes the oldest mouse positions upon receiving a new position. Using this method allows the ANN to predict a future mouse position continuously.

5 EVALUATION

We conducted a second study to determine the effects of our system. In the study, we created 50 ms artificial, controlled latency by buffering the mouse input on the OS level using a custom Python script and the Python library *PyNPut*.

5.1 Study Design

We investigated the effects of our prediction using a within-subject design. Hence, we used Delay as an independent within-subject variable. The variable had three levels: (1) *No Latency* - which is used as a control condition and corresponds to playing with no latency and no applied prediction -, (2) *Latency* - which corresponds to playing with 50 ms of controlled latency and no prediction being applied in the gaming session -, and lastly, (3) *Prediction* - categorizing gaming sessions with a controlled latency of 50 ms and an enabled prediction compensating 50 ms of latency. To measure the game experience, we used the in-game modules of the *Game Experience Questionnaire* (GEQ) [20]. We used the six sub-scales *Competence*, *Flow*, *Tension*, *Challenge*, *Positive Affect*, and *Negative Affect* of the in-game module and expanded it by the four sub-scales *Positive Experience*, *Negative Experience*, *Tiredness* and *Return to Reality* by the post-game module to quantify the subjective effects of our system on the gamers. We measured the effect of Delay on the gamer performance using the dependant variable *Score*. *Score* is generated using AoE2's built-in scoring system. All participants played with all levels of Delay resulting in three different conditions for our study. The condition order in the study was randomized to prevent a bias induced by sequence effects.

5.2 Apparatus

We installed AoE2 on a stationary work station (Intel i9-9900k, 16 GB RAM, Nvidia RTX 2060 6 GB VRAM) in our laboratory. The workstation was attached to a monitor (24" FullHD @60Hz), a computer mouse (Logitech M10), and a wired headset. The laboratory was quiet and free of external disturbance.

5.2.1 Procedure and Task. Participants were greeted at our institution's laboratory by the experimenter. After giving informed consent and agreeing to the data collection, participants were seated at the workstation running AoE2. Participants were not informed about the exact purpose of the study (investigating the effects of the developed prediction system). However, they were briefed about the general procedure of the study. Next, all participants played three rounds of AoE2's free play mode for 15 minutes. After each round, they filled out the GEQ. Upon finishing the third round and the third time filling out the GEQ, we collected demographics and information about past gaming experiences in RTS games and AoE2. Lastly, participants were debriefed. Participation in the study took about 1 hour.

5.3 Participants

Using our institution's mailing list, we invited 12 participants (2 f, 10 m). The participants' mean age was 23.16 years (SD: 2.31 years). Participants were chosen blind to age and gender. To prevent a bias induced by individual gaming skills, participants had little to no experience playing AoE2. Participant pre-screening was crucial since experienced AoE2 players deploy sophisticated gameplay strategies such as using shortcuts, queuing of unit construction, and simultaneously controlling independent units, which would have distorted the data of our study. Furthermore, all participants were students at our institution and were compensated for participation with one credit point for their study course.

5.4 Results

Based on the authors' recommendation [20] and in line with previous work [19], we independently evaluated each sub-scale of the GEQ. Table 1 shows the mean score and standard deviation for all sub-scales. All measures showed no violation of normality using Shapiro-Wilk's test (all $p > 0.05$). Hence, we used an analysis of variance (ANOVA) for further statistical testing. All post-hoc tests are Alpha-corrected Tukey tests.

A one-way ANOVA (Delay: *Latency* vs. *No Latency* vs. *Prediction*) showed no significant effect of Delay on the in-game sub-scales *Competence*, *Flow*, *Challenge*, *Positive Affect* and no significant effect on the post-game sub-scales *Positive Experience*, *Negative Experience*, and *Returning to Reality* (all $p > 0.2$, all $\eta_p^2 < 0.1$). However, ANOVA

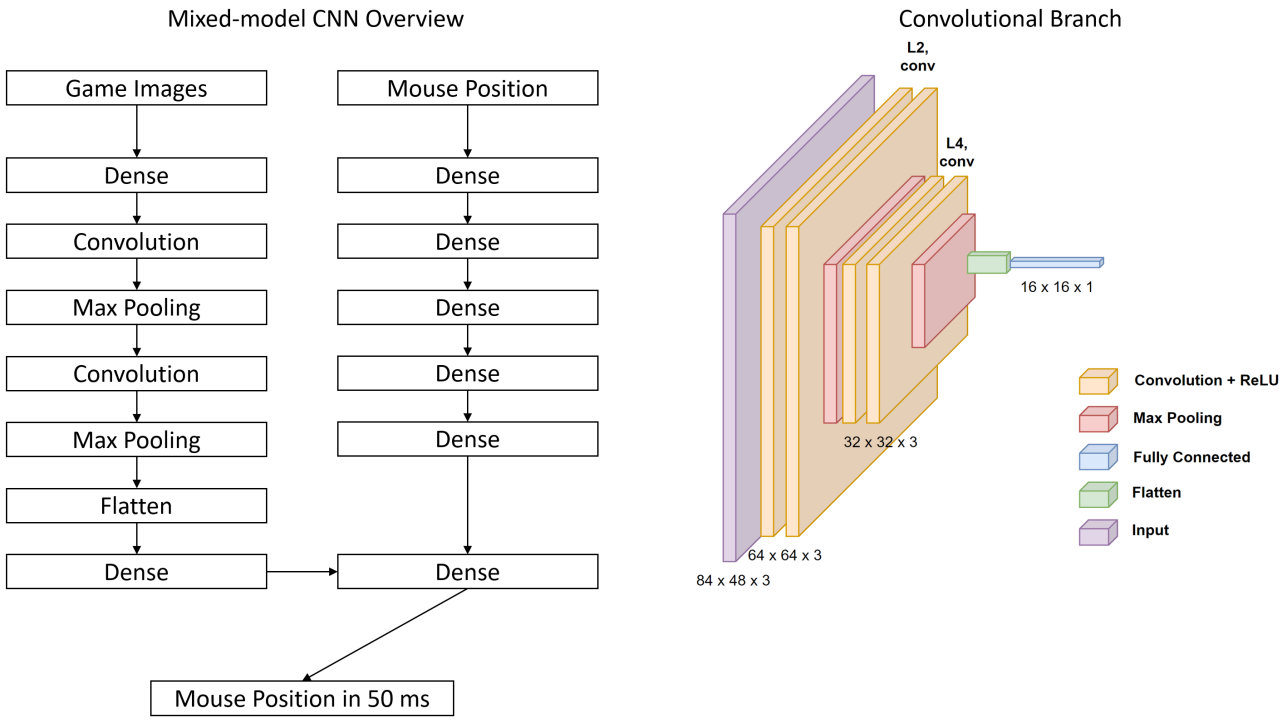

Figure 2: Shows an overview of the mixed-model *Convolutional Neural Network* (CNN) (left), and the architectural structure of its convolutional branch used for in-game image processing (right). In the model, mouse positions are processed by series of dense layer while the in-game images are processed by the convolutional branch. To reduce computational load the original images are downsized to 84 pixels by 48 pixels. This downsized image is passed through two convolutional layer coupled to *Max Pooling*. The multi-dimensional image is then transformed to an one-dimensional array using a *Flatten* layer. The output of the convolutional branch is fed to a dense layer which also receives the processed mouse positions. Finally, last dense layer outputs a X,Y coordinate for the mouse position in 50 ms.

Score Game Experience Questionnaire										
Latency	Com.	Flo.	Ten.	Cha.	Pos.	Neg.	Pos. E.	Neg. E.	Tired.	Real
No Delay	2.75/0.83	1.75/0.81	0.54/0.23	1.95/1.07	2.58/0.88	0.66/0.57	2.09/0.89	0.68/0.39	0.63/0.57	0.94/0.69
Delay	1.95/1.07	1.54/0.87	1.75/1.21	2.21/1.03	1.38/0.99	1.71/1.16	1.39/0.99	1.26/0.86	1.66/0.91	1.02/0.78
Prediction	2.25/1.37	1.71/1.01	0.95/1.08	1.52/1.02	1.84/1.05	0.71/0.54	1.85/1.05	0.81/0.58	0.71/0.72	1.05/1.01

Table 1: Shows the mean scores with standard deviation (mean/SD) for all sub-scales of the *Game Experience Questionnaire*. The data is grouped by the three levels of Latency.

revealed a significant main effect of Delay on *Tension* (F(2,33) = 4.632, p = 0.017, η_p^2 = 0.219). A Tukey post-hoc test showed significant differences between *No Latency* and *Latency* (p_{tukey} = 0.014, d_{cohen} = 0.223). All other differences were not significant (all p_{tukey} > 0.137, all d_{cohen} < 0.091). The one-way ANOVA also revealed a significant main effect of Delay on *Negative Affect* (F(2,33) = 4.632, p = 0.005, η_p^2 = 0.278). Post-hoc testing showed significant differences between *Latency* and *No Latency* (p_{tukey} = 0.009, d_{cohen} = 0.287) and between *Latency* and *Prediction* (p_{tukey} = 0.013, d_{cohen} = 0.235). However, no significant difference between *No Latency* and *Prediction* (p_{tukey} = 0.991, d_{cohen} = 0.051). Furthermore, ANOVA revealed a significant main effect of Delay on Tiredness. Post-hoc tests showed significant differences between *Latency* and *No Latency* (p_{tukey} = 0.005, d_{cohen} = 0.393) and between *Latency* and *Prediction* (p_{tukey} = 0.013, d_{cohen} = 0.281). However, no significant difference between *No Latency* and *Prediction* (p_{tukey} = 0.991, d_{cohen} = 0.096).

Overall, when using our predictive system participants had a lower *Negative Affect* and *Tiredness* compared to playing with 50 ms of latency and no support by the ANN. Table 2 shows all ANOVAs performed on the GEQ data - significant results are highlighted. Figure 3 shows the scores for the sub-scales *Negative Affect* (left) and *Tiredness* (right).

One-way ANOVA found no significant main effect of Delay on *Score* (F(2,33) = 0.914, p = 0.411, η_p^2 = 0.053). Participants on average achieved 10,042.42 points ± 1204,5 points.

ANOVAs of Game Experience Questionaire										
	Com.	Flo.	**Ten.**	Chal.	Pos.	**Neg.**	Pos.E	Neg.E.	**Tired.**	Real.
DF	2	2	**2**	2	2	**2**	2	2	**2**	2
Residual	33	33	**33**	33	33	**33**	33	33	**33**	33
F-value	1.540	0.263	**4.632**	1.441	1.180	**6.369**	1.569	2.708	**7.188**	0.031
p-value	0.229	0.770	**0.017**	0.251	0.320	**0.005**	0.223	0.081	**0.003**	0.970
η_p^2	0.085	0.016	**0.219**	0.080	0.067	**0.279**	0.087	0.141	**0.303**	0.002

Table 2: Results of the one-way ANOVAs investigating the effects of Delay on the different sub-scales of the Game Experience Questionnaire [20]. Significant results are displayed in bold. We found significant differences between *Tension* (Ten.), *Negative Affect* (Neg.) and *Tiredness* (Tired.). Participants playing with our presented latency compensation method associated the game with significantly less negative feelings and were significantly less tired after the gaming session.

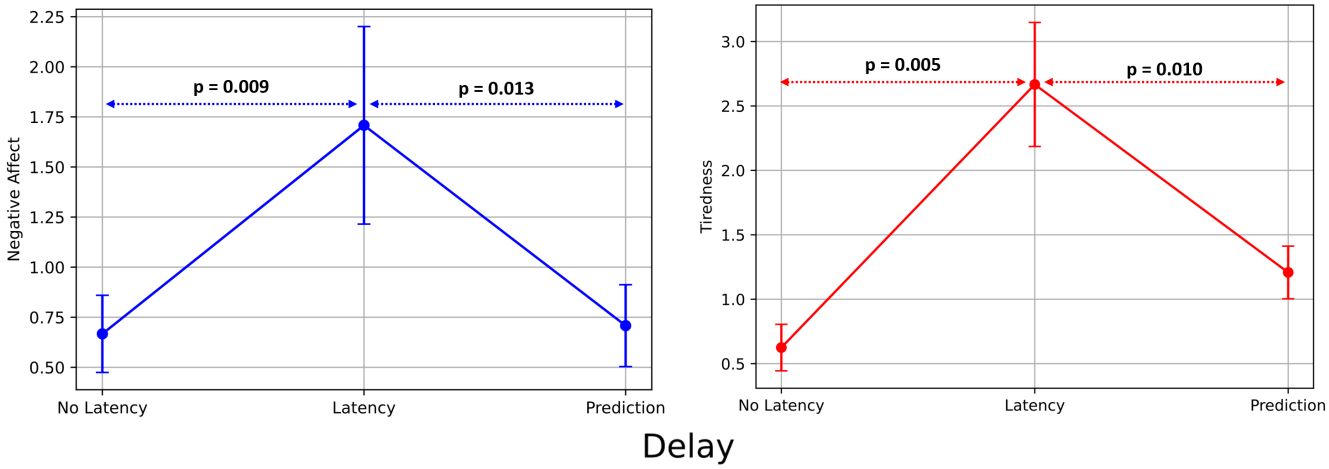

Figure 3: Shows the sub-scales *Negative Affect* (left) and *Tiredness* (right) of the Game Experience Questionnaire [20]. Significant differences are highlighted via p-bars. Error bars depict the standard error (SE). Participants rated the negative affect associated with the game significantly highest when playing with latency and no prediction. In the same manner, participants rated the gaming session with latency and no prediction with most tiring. We found no significant difference between playing with no latency and our prediction, hence our system did not negatively influence the game experience.

6 DISCUSSION

Our results show that using a deep learning-based predictive system to compensate for latency in commercial *Real-Time Strategy* games significantly reduces the negative effects of latency on the players' game experience. Gamers playing *Age of Empire 2* with a latency of 50 ms supported by our system, associated the game with a significantly lower *Negative Affect* and experienced significantly lower *Tiredness* than playing without the ANN. The objective game performance remained constant in all tested conditions. Thus, our analysis shows that our system removes the negative effects of latency without introducing negative secondary effects.

In the following, we first explain and discuss the found effects based on previous work investigating the negative influence of latency on game experience. We then discuss the implication of our findings and our work for game developers, researchers, and cloud-based game streaming providers.

6.1 Tension, Negative Affect, and Tiredness

Generally, previous work showed that latency negatively affects the game experience of video games [13]. Liu et al. [27], for example, showed that latency starting at 25 ms leads to an experience degradation. The authors showed that this decrease in quality of experience linearly correlates with the amount of latency - the higher the latency, the more pronounced the effects on the game experience. In our work, we successfully compensated for the negative effects of controlled latency without negatively influencing the positive aspects of the gaming experience. We found a significant decrease in negative affect when playing with our ANN compensating latency compared to playing with latency and no prediction. A gamer's negative affect is a manifestation of their experienced negative emotions, such as anger, fear, and disgust while playing the game. Similar effects of latency compensation based on deep learning on the game experience have been reported by Halbhuber et al. [19]. Interestingly, Halbhuber et al. [19], contrary to our findings, found an increase in the positive affect experienced while playing with their compensation method instead of a decrease in

the negative affect. This difference could be due to the different amounts of latency induced. While Halbhuber et al. used a baseline latency of 180 ms artificially added latency, we evaluated our system during gameplay with 50 ms of latency. Comparing both works indicates the evolution of latency-based effects. While compensating for high latency (180 ms) increased positive feelings, compensating for lower latency (50 ms) decreases negative feelings associated with and triggered by the gaming session. Both approaches, reducing low latency and high latency, are beneficial to optimizing the overall game experience. A direct comparison suggests that compensating for different latency levels might improve different aspects of the game experience.

Our analysis also showed that the tiredness induced by the gaming session was significantly lower when playing with our latency compensation ANN compared to playing without it. A gamer's tiredness after a gaming session indicates how exhaustive the session was. The exhaustion in gaming may be due to multiple reasons. One possible reason for a high level of exhaustion is a cognitively demanding gaming session. For example, if gamers have to simultaneously observe, control, and manage various game resources such as units in a *Real-Time Strategy* game [9]. However, since we did not find a significant difference between playing without latency and playing with our system compensating latency, it is unlikely that the game itself induced a high level of tiredness. On the contrary, as we only found a significant increase in gamers playing with latency and no compensation, we conclude that latency in video games directly induces a higher level of exhaustion. The resulting de-synchronization induced by the latency between input and visual confirmation increased the cognitive demand of all in-game tasks. The absence of a responsive confirmation of a performed action led to the fact that performed actions had to be observed and controlled for a more extended period. Thus, ultimately, latency induced higher tiredness.

We found that the tension experienced while playing with latency was significantly higher compared to playing without latency. This shows that we successfully induced latency in the game - hence, the increased tension. We did, however, not find a decrease in experienced tension when playing with our ANN's prediction. However, we also did not find a significant difference between playing with no latency and playing with the support of our ANN. Our ANN could not fully compensate for the negative effects of latency on the perceived tension. It, however, was able to lower the effects to the point that they are no longer statistically distinguishable from playing with no latency.

In summary, as prior work did [10, 12, 27], our work shows that latency negatively influences the game experience. Additionally, our work presents a solution to the latency problem in cloud-based game streaming - deep learning-based latency compensation. Furthermore, a comparison with related work [19] indicates that compensating for different levels of latency might improve different aspects of gamers' experience.

6.2 Implication of our Findings

Game developers should be aware of the different effects of latency and latency compensation techniques on gamers. Especially potential dissimilarities when compensating for high latency compared to a lower latency may be relevant in the game optimization process. Reducing a high latency increases the positive affect. Thus, it increases the fun and the number of positive emotions associated with the game. Reducing a low latency leads to the decrease of negative affect. Game developers can utilize this knowledge to improve games designed for cloud-based game streaming in the early development stages. Developers can focus on decreasing high latency early on, knowing that any negative association in playtesting may be attributed to a remaining latency. Decreasing only high latency early on in the development allows for saving resources that can be utilized in other development areas. In later optimization stages, the remaining low latency may then be reduced to optimize and finalize the game experience. Furthermore, game developers should be aware of the different latency conditions when simultaneously developing a game for conventional local gaming systems and cloud-based game streaming platforms.

Researchers may also benefit from our work. We showed that deep learning-based latency compensation techniques are well equipped to reduce negative effects induced by latency. Furthermore, we showed that the different latency levels, and thus their compensation, affect gamers differently. Researchers can build on our work to further investigate the effects of latency compensation on game experience in greater detail. We showed that deep learning-based latency compensation could reduce the negative effects of latency in *Age of Empire 2*. However, since we used the same game to train and evaluate our ANN, it remains unknown if ANNs can generalize across different games. We assume that by using a large enough data set of gameplay data of multiple games and genres, deep learning-based models could generalize over all games. Consequently, researchers should investigate the generalizability of deep learning-based latency compensation techniques in video games. The findings of our work and the presented method are also relevant to researchers outside of video games. Although we used game-specific parameters to train our system, the presented approach is potentially suitable for any software operated by a mouse and keyboard.For example, by integrating mouse prediction, a software could achieve higher responsiveness and thus enhancing the overall user experience.

Finally, our findings and the presented method to compensate for latency are also relevant to cloud-based game streaming providers. Streaming providers need to continue to improve and upgrade their server infrastructure and their method of compressing and delivering their content. Further optimizing latency conditions in cloud-based game streaming is essential to alleviate it to the latency level of conventional gaming systems. To do so, providers can use the herein presented latency compensation technique. Our presented method may be used for any type of game. In removing the inherent latency of game streaming, providers can offer gamers a gaming platform with the same game experience and performance potential as local gaming setups.

7 CONCLUSION

This work presents a novel approach to compensate for latency in commercial video games by predicting the mouse position in 50 ms in *Age of Empire 2* (AoE2) using ANNs. In contrast to previous work, we do not require the internal game state or to modify the

game. Consequently, our approach can compensate for the latency of unmodified commercial games to reduce the negative effects on the game experience induced by latency.

In our work, we did not find an increased accuracy when using visual material for the mouse prediction. However, the lower accuracy might be due to the small number of epochs trained. One can assume that an increased time spent on training increases the accuracy of the CNN and the mixed model. Both models did not converge to an optimization minimum yet in our preliminary evaluation. Future work, thus, should build on our work and further investigate the use of CNN and visual material for latency compensation techniques. Furthermore, the lack of increased prediction accuracy using images may be due to the image resolution used for training. In our work, we scaled the training images down to a resolution of 84 pixels by 48 pixels. Down-scaling was necessary to be able to train the CNN in a reasonable amount of time. Nevertheless, even with down-scaled images training, one epoch took approximately 36 hours. Future work, thus, should investigate if it is a feasible approach to increase computational power for training.

In our work, we simulated latency by buffering user input. In doing so, we created local latency. While our findings are evidently valid for this scenario, results might differ for other types of latency. Previous work showed that local and network latency differently affects gamers [26]. Similarly, the compensation of local latency may lead to other effects than network latency compensation. Future work should further investigate the different effects of compensating local latency and compensating network latency using deep learning-based techniques.

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] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/3292500.3330701

[3] Amazon. 2022. luna - Play your favorite games straight from the cloud. https://www.amazon.com/luna/landing-page. Accessed on 2022-02-22.

[4] 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

[5] Michelle Annett, Albert Ng, Paul Dietz, Walter F Bischof, and Anoop Gupta. 2020. How low should we go? Understanding the perception of latency while inking. In *Graphics Interface 2014*. AK Peters/CRC Press, 167–174. https://doi.org/10.5555/2619648.2619677

[6] Grenville Armitage. 2003. An experimental estimation of latency sensitivity in multiplayer Quake 3. In *The 11th IEEE International Conference on Networks, 2003. ICON2003.* IEEE, 137–141. https://doi.org/10.1109/ICON.2003.1266180

[7] 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

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

[9] Yang Chen, Jian Ou, and David M Whittinghill. 2015. Cognitive Load in Real-Time Strategy Gaming: Human Opponent Versus AI Opponent. *The Computer Games Journal* 4, 1 (2015), 19–30. https://doi.org/10.1007/s40869-015-0002-z

[10] Mark Claypool and Kajal Claypool. 2006. Latency and player actions in online games. *Commun. ACM* 49, 11 (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*. IEEE, 1–6. https://doi.org/10.1109/NetGames.2014.7008964

[13] Robert Dabrowski, Christian Manuel, and Robert Smieja. 2014. The Effects of Latency on Player Performance and Experience in a Cloud Gaming System. *Interactive Qualifying Project MLC-AAEZ* (2014). https://doi.org/10.1109/NetGames.2014.7008964

[14] DESTINYbit. 2022. An exciting free-to-play Real Time Strategy game. https://empiresapart.net/. Accessed on 2022-02-22.

[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] Microsoft Games. 2022. Age of Empire 2 Definitive Edition. https://www.ageofempires.com/games/aoeiide/. Accessed on 2022-02-22.

[17] 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.

[18] Carl Gutwin, Steve Benford, Jeff Dyck, Mike Fraser, Ivan Vaghi, and Chris Greenhalgh. 2004. Revealing delay in collaborative environments. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 503–510. https://doi.org/10.1145/985692.985756

[19] David Halbhuber, Niels Henze, and Valentin Schwind. 2021. Increasing Player Performance and Game Experience in High Latency Systems. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (2021), 1–20. https://doi.org/10.1145/3474710

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

[21] 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

[22] 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

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). https://doi.org/10.48550/arXiv.1412.6980

[24] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

[25] 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

[26] Shengmei Liu, Mark Claypool, Atsuo Kuwahara, James Scovell, and Jamie Sherman. 2021. The Effects of Network Latency on Competitive First-Person Shooter Game Players. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 151–156. https://doi.org/10.1109/QoMEX51781.2021.9465419

[27] Shengmei Liu, Mark Claypool, Atsuo Kuwahara, Jamie Sherman, and James J Scovell. 2021. Lower is Better? The Effects of Local Latencies on Competitive First-Person Shooter Game Players. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[28] Shengmei Liu, Xiaokun Xu, and Mark Claypool. 2022. A Survey and Taxonomy of Latency Compensation Techniques for Network Computer Games. *ACM Computing Surveys (CSUR)* (2022). https://doi.org/10.1145/3519023

[29] 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*. 285–297. https://doi.org/10.1145/3242671.3242678

[30] 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. https://doi.org/10.1145/169059.169431

[31] Scott I. 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* (Amsterdam, The Netherlands) *(CHI '93)*. Association for Computing Machinery, New York, NY, USA, 488–493. https://doi.org/10.1145/169059.169431

[32] 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

[33] 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

[34] Nvidia. 2022. The next generation in cloud gaming. https://www.nvidia.com/en-us/geforce-now/. Accessed on 2022-02-22.

[35] Nvidia. 2022. Nvidia DLSS 2.0: A Big Leap in AI Rendering. https://www.nvidia.com/en-gb/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/. Accessed on 2022-02-22.

[36] Jeff Hinrichs Palmér. 2022. pysftp 1.7.6. https://pypi.org/project/pysftp/. Accessed on 2022-02-22.

[37] Moses Palmér. 2022. pynput 1.7.6. https://pypi.org/project/pynput/. Accessed on 2022-02-22.

[38] Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 55–69. https://doi.org/10.1007/978-3-642-35289-8_5

[39] 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. https://doi.org/10.1145/3385956.3418941

[40] 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. https://doi.org/10.5555/2627435.2670313

[41] 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)*.

[42] OpenCV team. 2022. OpenCV – 4.5.5. https://opencv.org/. Accessed on 2022-02-22.

[43] Tungsteno. 2022. pyWinhook 1.6.2. https://pypi.org/project/pyWinhook/. Accessed on 2022-02-22.

[44] Valve. 2022. Steam - The ultimate destination for playing, discussing, and creating games. https://store.steampowered.com/. Accessed on 2022-02-22.

[45] Brent Vollebreg. 2022. Auto PY to EXE. https://pypi.org/project/auto-py-to-exe/. Accessed on 2022-02-22.

[46] Wikipedia. 2022. AAA (video game industry). https://en.wikipedia.org/w/index.php?title=AAA_(video_game_industry)&oldid=1067619619. Accessed on 2022-02-22.