Advancing Esports Analytics: A Deep Learning Approach to Valorant Match Prediction Using CNN and LSTM

Sanchit Kedia
UMD, College Park
sanchit@umd.edu

Shantanu Parab
UMD, College Park
sparab@umd.edu

Tanmay Haldankar
UMD, College Park
tanmayh@umd.edu

Vineet Singh
UMD, College Park
vsingh03@umd.edu

Abstract

In this project, we explore the integration of advanced neural network architectures—Convolutional Neural Networks (CNNs) and Long-Short-Term Memory networks (LSTMs) for Esports outcome prediction in Valorant, a popular first-person shooter game. Our approach involves using CNNs to analyze key in-game visual elements while employing LSTMs to understand the temporal patterns to predict the winner of a game round with greater accuracy than previously possible. This report outlines our current progress and delineates the prospective trajectory of the research, which holds the potential to pioneer a state-of-the-art predictive model in the Esports domain.

1. Introduction

Esports, which stands for electronic sports, is a segment of the video game industry where professional players compete in various video games to attain global rankings and win significant prize money. This industry has experienced remarkable growth, with its global viewership and revenue soaring. In 2021, Esports generated approximately 243 million USD in revenue in the United States, while in China, the revenue was around 360 million USD. In 2022, Esports achieved a significant milestone with a massive audience of 532 million, including 249.5 million occasional viewers and 240 million devoted Esports enthusiasts.

Sports prediction algorithms, characterized by their utilization of historical and real-time data, play a pivotal role in traditional sports. These algorithms confer numerous advantages, including heightened fan engagement, strategic decision support, player performance analysis, application in sports betting platforms, and improvement of sports broadcast and media coverage. However such technologies have not been extensively explored or implemented within the realm of Esports. This gap in technological application underscores the potential for research and development in this domain, and holds the promise of bringing similar benefits to Esports.

1.1. Valorant as a Testbed for Esports Predictive Analysis

In our quest to identify a suitable Esports context for our research, we found that Valorant is the best choice. It's one of the most popular and competitive first-person shooter (FPS) games in the Esports world. In Valorant, each round lasts approximately 100 seconds and involves two teams taking turns as attackers and defenders. The game is played on various maps, each of which has 2 to 3 critical spots known as "sites". These sites are labeled A, B, or C, depending on the number of sites on the map. The attackers aim to infiltrate one of these sites, plant a spike, and defend it until it detonates. Conversely, the defenders' goal is to protect the sites by preventing the attackers from planting the spike or defusing it after it has been planted. The round ends when the attackers win by exploding the spike, eliminating all defenders, or the defenders win by preventing a spike plant, eliminating all attackers, or defusing the spike. The first team to win 13 rounds wins the game, and the roles switch after the first 12 rounds. If the score is tied at 12, the game enters overtime, where the first team to win an attacking and a defending round consecutively wins.

1.2. Predictive Modeling and Broadcast Analysis for Valorant

Our project aims to create a deep network that can accurately predict the winning team while watching a professionally streamed Valorant game. Riot Games is responsible for streaming official Valorant tournaments, and their broadcast UI provides valuable data for our predictions. We will segment the Valorant broadcast screen into specific areas to extract this information.

The Mini-Map : The mini-map is located in the top left corner of the screen and provides a quick overview of the active map. It displays the real-time positions of all alive players during the round. Player icons are color-coded, with green representing the defenders and red representing the attackers. Along with player positions, the mini-map also

shows the locations of dead players, marked with an 'x' and color-coded blue or red. Additionally, it displays the deployable abilities of specific agents (represented by icons) and the location of the planted spike (if applicable).

Player information: This segment presents player information as icons and text. The information includes the player's name, health, weapon, credits, and chosen agent with their respective abilities. The display separates this information based on the teams, showing details for all ten players in the game.

Round time and map score: The section on the screen that displays the round time and map score shows the current score and the time that has passed since the round started. Moreover, it indicates the presence of a planted spike and updates the score dynamically based on the winning team after every round.

1.3. Deep Learning Methodologies in Esports Predictive Analysis

Our research explores the use of Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) networks in predictive analytics in Esports. Specifically, we focus on how these networks can predict outcomes in Valorant matches. CNNs are particularly powerful for processing visual data because they capture spatial hierarchies and patterns. They analyze visual cues from frames, breaking down images into tiles and extracting features at various levels of abstraction. This characteristic makes CNNs adept at interpreting the complex visual elements of a Valorant broadcast, such as player positions and movements, which are crucial for predicting round outcomes.

Meanwhile, LSTMs and GRUs are classes of recurrent neural networks which excel in understanding temporal sequences, making them suitable for analyzing the sequential aspect of Esports matches. These networks are good at recognizing and learning from the continuity and progression intrinsic to Valorant gameplay. By retaining information over extended sequences, LSTMs, and GRUs can capture the evolving tactics and player movements that ultimately decide the outcome of each round. Their architecture allows them to remember information for long periods, which is critical for considering the progression of a game when predicting results.

Our project unfolded in two distinct phases. Initially, we employed a Convolutional Neural Network (CNN) during phase 1 to forecast the winner by evaluating the outcomes of individual frames. Subsequently, in phase 2, we advanced our approach by training two models—a CNN+LSTM and a CNN+GRU. These models were designed to predict outcomes based on sequences of frames within a round. Through these phases, our research delved into predictive analytics in Esports, with Valorant as our testbed. The integration of CNNs, adept at processing visual data, and LSTMs and GRUs, excelling in understanding temporal sequences, showcased a comprehensive strategy to capture the game's nuances and predict round outcomes effectively.

2. Related Work

Esport game outcome prediction has emerged as a critical facet of gaming analytics, prompting pioneering investigations into the application of machine learning methodologies. In seminal work [5], the authors demonstrated the viability of predicting Esport game outcomes using traditional techniques such as Linear Regression and Random Forest. Initial endeavors relied on server logs to construct essential datasets for training models. Subsequent research [3] by Tiffany D. Do delve into advanced predictive models, including Deep Neural Networks, Support Vector Classification, K-nearest neighbors, Random Forest Trees, and Gradient Boosting, specifically within the context of the Esports game League of Legends. Notably, this model focused on character selection at the game's outset, utilizing character stats but overlooking player proficiency.

Recognizing the limitations of prior works, our research aims to transcend these boundaries by incorporating real-time game information. Our approach considers dynamic player strategies and their impact on game outcomes, contrasting with character-centric models.

In a related study, [8], the authors investigate combat sequences in the Multiplayer Online Battle Arena game DotA 2. Through game log analyses, combat sequences are modeled as sequences of graphs. Feature selection and decision trees unveil predictive combat patterns. Additionally, [6] explores dynamic changes in team statuses, such as goal difference and kill score difference, for outcome prediction.

Inspired by these methodologies, we introduce our first approach. Diverging from traditional server logs, we harness video feeds from the game screen, extracting relevant features at each time step using Convolutional Neural Networks (CNNs). CNNs, renowned for capturing spatial patterns, are employed to discern spatial features during gameplay for outcome prediction.

However, our methodology acknowledges the criticality of temporal data for comprehensive understanding. Temporal modeling, introduced by [1] through Long Short-Term Memory (LSTM) networks, accentuates their prowess in capturing dependencies over extended periods. Building upon the idea of using LSTM for sequence modeling, we further consider Gated Recurrence Neural Networks (GRU), as showcased in [2], as an alternative for LSTM. This comprehensive evaluation demonstrates the efficacy of GRU for sequence modeling in Esport analytics.

In summary, our literature review establishes the foundation for our research in Esports game outcome predic-

tion. While previous studies have explored diverse dimensions, our innovation lies in integrating both CNN-LSTM and CNN-GRU architectures. This pioneering approach aims to capture the intricate interplay of spatial and temporal dependencies inherent in Esport game data, advancing the state-of-the-art in the field.

3. Dataset Creation

3.1. Data Sourcing and Preliminary Processing

To create our dataset, we began by sourcing professional Valorant broadcast videos from YouTube. We focused on the most recent major international Valorant events, Valorant Champions 2023 and Valorant Masters Tokyo 2023. On average, these videos were around 6.5 hours long and contained two or three games. Since professional broadcasts include much extra content like live analysis, downtime between games, map selection, timeouts, interviews, and highlights, we edited the videos to extract only the round-by-round action. This made the videos more manageable, with an average length of 30 minutes, focusing on the rounds played within a single game (and map).

3.2. Dataset Organization and Mask Application

We have gathered data from 12 videos, covering 63 games in total. To narrow our focus on specific regions within each image, we have created masks Figure 2 that isolate the screen, as mentioned in the above sections Figure 3. It is worth noting that these masks vary between maps due to each map's differing shapes and layouts. Figure 1 shows the distribution of maps present in our dataset. The footage of the 63 games gathered was saved in separate folders, each named according to the map the game was played on. The videos were purposely edited to present consecutive rounds with a black screen in between to facilitate automated round separation and labeling. Game outcomes were obtained from available online data, with each round's result recorded in a corresponding CSV file bearing the video's name.

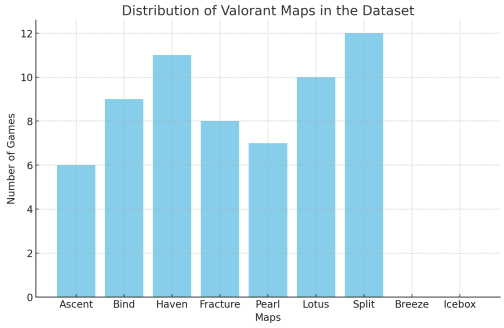

Figure 1. Distribution of Valorant Maps present in our dataset

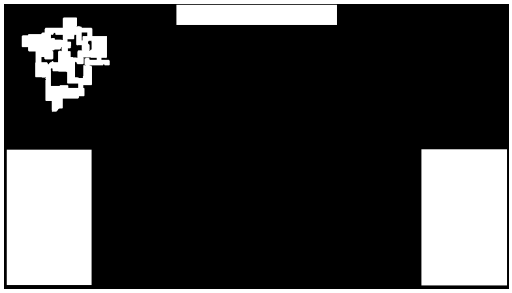

Figure 2. Mask of Valorant map Ascent

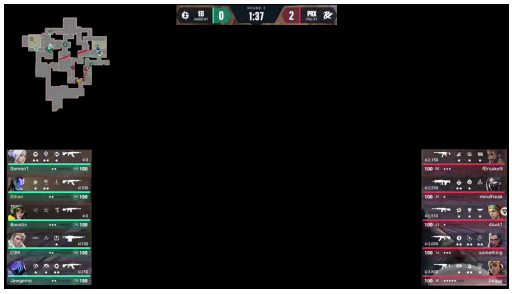

Figure 3. Sample captured frame of Valorant Map Ascent

Figure 4. Sample cropped frame of Valorant map Ascent

3.3. Frame Extraction and Labeling

A Python script was created to aid in data generation by extracting frames from edited videos and applying map-specific masks. For phase 1, the data was generated by capturing every 30th frame (the video data is 30fps) in each video while ignoring black frames. To label the frames, two sub-folders named 'win' and 'loss' were set up inside the phase-1 dataset folder, and frames from each round were placed in one of these folders based on the outcome of the

round the frames belonged to, indicated in the CSV (win for red team victory and loss for their defeat). The resulting stage 1 dataset consists of 99,308 labeled images. To make our Python script compatible with both LSTM and MoViNets, we made some improvements, and a similar approach as before was employed to generate data for phase 2, where the masked footage for every round was saved as a video in sub-folders inside the phase 2 dataset folder, based on the outcome of that round indicated in the CSV. This amounted to 1259 labelled round-wise labelled videos.

3.4. Phase-1 Dataset Compilation and ResNet Model Implementation

The model used in phase 1 was a ResNet-50. While PyTorch offers pre-trained models for classification, these are predominantly trained on the ImageNet dataset. Given the significant divergence in data distribution between ImageNet and our project's specific requirements—centering on game footage not closely related to the ImageNet database—it was imperative to train our ResNet models from the ground up.

The sorting of video frames into distinct folders based on their corresponding labels allowed us to streamline data loading using PyTorch's "torchvision.datasets.ImageFolder" function. This function generates a Dataset object and assigns labels based on the folder hierarchy. The dataset.class_to_idx attribute reveals the correct classification schema: {'loss': 0, 'win': 1}. The frames loaded were transformed in line with the input specifications recommended for Res-Net50 on PyTorch listed in Table 1.

Our dataset was partitioned into training and testing subsets, following an 80-20 split. The training subset was further divided similarly (80-20) to create distinct training and validation sets. For phase 1, we utilized a batch size of 32 with shuffle enabled.

Parameter	Value
Original frame size	3x300x300
Resize	256
Center Crop	224
Normalize	mean=[0.485, 0.456, 0.406] std=[0.229, 0.224, 0.225]
Transformed frame size	3x224x224

Table 1. Image Transformation parameters: ResNet-50

3.5. Dataset Preparation for Advanced Modeling in Phase-2

Unlike image data, PyTorch does not natively support loading video data from separate folders. This necessitated the creation of a custom dataset class for phase 2, requiring us to override the default methods from PyTorch's Dataset class.

To facilitate the loading of video data, we compiled a CSV file containing the paths to each round-specific video along with their corresponding labels and modified the getitem() method of the Dataset class in Pytorch. It was overridden to read individual videos, extract frames at a rate of 30 frames per second (FPS), and compile them into a tensor for processing. We ensured that each frame within this tensor underwent the same transformations as in phase 1, considering that the underlying model architecture was based on the same ResNet model previously trained.

The dataset was again divided into training, validation, and testing sets, employing the same method as in Phase 1. However, given the considerably larger size of the video data, the batch size for the dataloaders was reduced to 5.

Our dataset contains videos of various lengths. This variation in length posed a challenge for training LSTM or GRU models as these models require sequences of the same length within each batch. To solve this problem, we used a collate function that identifies the longest video in each batch and pads the shorter videos with black frames. This ensures that the sequence lengths are standardized across the batch. Moreover, we used the torch.nn.utils.rnn.pad_sequence function converts these sequences into 'padded sequences' suitable for LSTM and GRU layers in PyTorch. It's important to note that the added black frames do not contribute to the learning process of the model. This ensures that only the original video frames are used for model training and analysis.

Parameter	Stage 1	Stage 2
Model	ResNet-50	LSTM/GRU + ResNet-50
Dataset	99,308(images)	1,259(videos)
In. Size	3*224*224	3*224*224
In. Dim(Train)	32*3*224*224	5*N*3*224*224
In. Dim(Eval)	3*224*224	N*3*224*224
Total Params	58,147,906	17,361,794
Out. Dim(Train)	32*2	5*2
Out. Dim(Eval)	1*2	1*2
Optimizer	Adam	Adam (α = 1e-5)

Table 2. Comparison of Stage 1 and Stage 2 Models

3.6. Model Optimization and Performance Evaluation

In addressing the classification challenge in our study, we have adopted the cross-entropy loss function as the pri-

mary criterion for model optimization. To evaluate the performance of our models, accuracy has been chosen as the critical metric. These metrics offer a balanced approach: while cross-entropy loss ensures that the model's predictions are as close as possible to the actual labels, accuracy provides a clear and understandable measure of the model's overall classification performance.

To further elucidate our methodology, we have included a comprehensive table 2 that details the output of the model and its dimensions, the loss function and evaluation metric utilized, and other critical aspects of the model's size and architecture.

4. Methodology

In our study, we addressed the challenge of predicting the outcome of a game round in Valorant based on the visual information presented on the screen. Our overarching aim is to develop a system capable of analyzing gameplay in real time and predicting the round outcome. We approached this problem in two distinct stages, utilizing the data from the mini-map, which provides crucial information like player positions, player count, and the location of the spike.

4.1. Phase 1

In phase 1, we employed a classification model to categorize each frame of a Valorant round as either a win or loss for the red team. For this purpose, we trained a ResNet-50 on our dataset comprising 99,308 labeled images. While conscious of the training time required for the model to learn from scratch, we sought a model deep enough to capture the complexities of Valorant gameplay, hence our choice of ResNet50. The fully connected (FC) layer of the ResNet was modified to output predictions for two classes, and the architecture can be seen in Figure 5.

This model was trained over 8 epochs with a learning rate of 0.0001, using the Adam optimizer and cross-entropy loss. The training, conducted on a system equipped with a 3070Ti laptop GPU, required approximately 20 minutes per epoch. The trained ResNet50 achieved a training accuracy of 94.35%, a validation accuracy of 92.32%, and a test accuracy of 92.55%. These results are depicted in Figures 6, 7, with the latter illustrating the prediction versus ground truth for a test image batch.

To further analyze the classification model outcomes, we utilized Gradient-weighted Class Activation Mapping (Grad-CAM) [7] [4]. The implementation of Grad-CAM required a custom forward pass for the ResNet, capturing features after layer 4 and gradients relative to the higher score prediction using PyTorch's forward and backward hooks. The heatmap was generated by pooling the gradients, multiplying the features by these pooled gradients, averaging them, and then resizing the activations to match the original input size. We also explored gradient-free methods

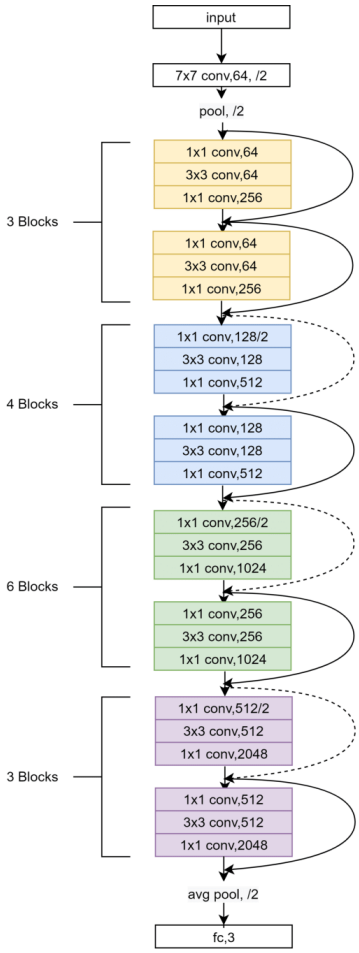

Figure 5. Modified ResNet-50 architecture

like AblationCAM and Eigen CAM, finding similar attention maps, as shown in Figure 8.

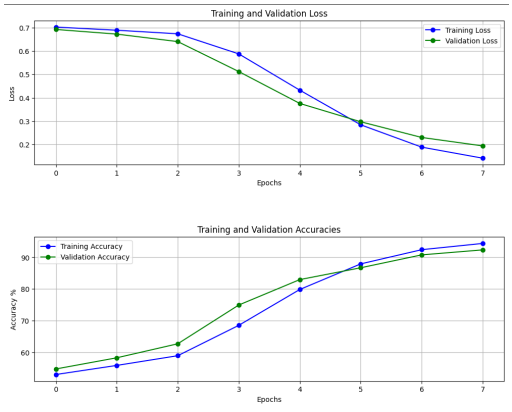

Figure 6. Phase 1 CNN training graphs

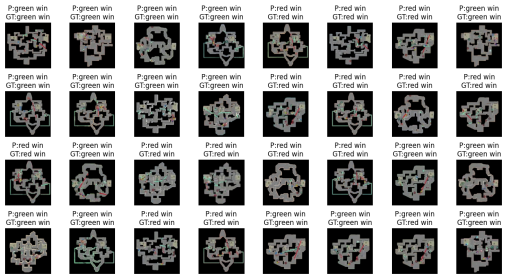

Figure 7. Phase 1 prediction results

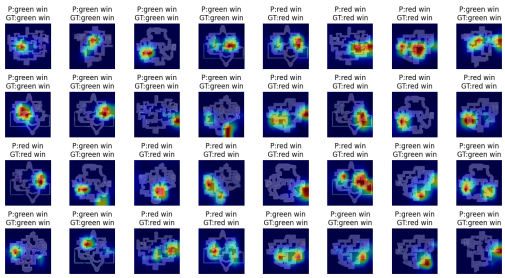

Figure 8. Grad-CAM Visualization for trained ResNet-50

4.2. Phase 2

The second phase evolved from the first, focusing on LSTM+CNN and GRU+CNN models to understand and predict temporal sequences. We utilized torch.nn.LSTM and torch.nn.GRU models in PyTorch, setting the input size to 512, hidden size to 256, and number of layers to 3. The output from the LSTM and GRU layers was then passed through two additional fc layers. The intent was to use the final output of the LSTM and GRU to make predictions based on the accumulated and altered hidden states throughout the round. A challenge encountered was the network's initial inability to learn when training only the LSTM and GRU layers. To overcome this, we retrained the layers from layer 4 onwards of the ResNet model from phase 1. The trained layers and the number of parameters can be seen in Figure 9. To ensure the preservation of the most proficient model, the weights were saved during training after each epoch where the model attained the highest validation accuracy.

Our best LSTM model took 794 minutes (13.23 hours) to train, achieving training, validation, and testing accuracies of 91.18%, 84.65%, and 80%, respectively. The GRU model, while faster at 205 minutes (3.42 hours) for 10 epochs, reached training, validation, and testing accuracies of 87.6%, 69.5%, and 80%, respectively. The training results and plots for both LSTM and GRU models are presented in Figure 10 and Figure 11. The hyperparameters for both models were a learning rate of 0.001 and a batch size of 5.

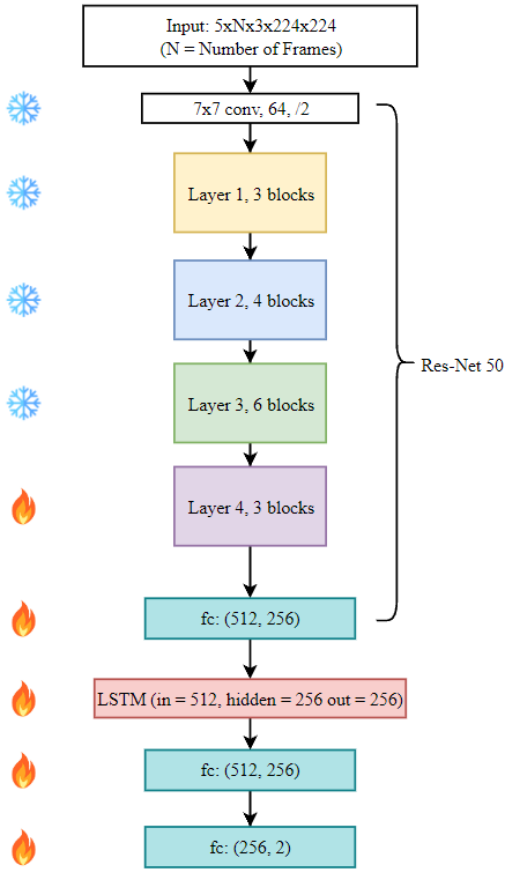

Figure 9. Combined CNN-LSTM architecture

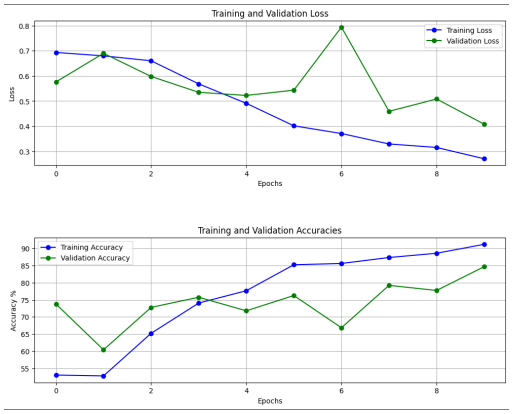

Figure 10. Phase 2 LSTM training graphs

5. Experiments

In our research, we have pioneered the use of CNN, CNN+LSTM, and CNN+GRU models on a dataset of this scale and type, focusing on the strategic game of Valorant. To validate the effectiveness and generalizability of our models, we conducted two separate experiments using

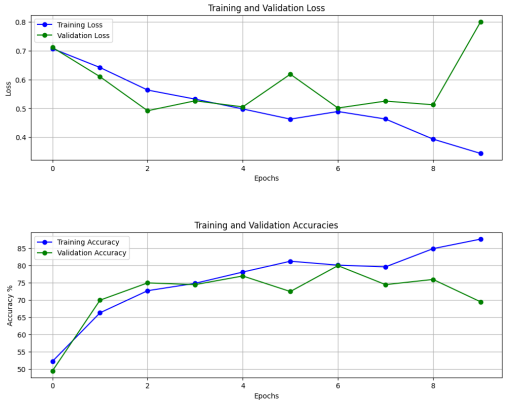

Figure 11. Phase 2 GRU training graphs

data from the Valorant Champions Tour 2023 Grand Finals, a dataset previously unseen by our models. In the graphs depicted, the green line shows the predicted score for red loss/green win and the red line shows red win.

5.1. Experiment 1: ResNet Model Inference

The first experiment involved the application of the trained ResNet model to predict the outcome of each round based on every 30th frame extracted from the video. A graph was plotted to visually represent these probabilities over the course of the game. Additionally, we calculated an overall accuracy metric, representing the proportion of rounds correctly predicted by the model. The ResNet achieved an accuracy of 66% over the entire game. The findings from this experiment are presented in Figure 12.

Figure 12. Experiment 1 CNN model predictions

Due to the relatively low computational demands of running inference with the ResNet model, this process can be executed in real-time. This capability allows for the potential of live broadcast analysis, where a similar graph could be generated by capturing relevant screen segments during a live game

5.2. Experiment 2: CNN+LSTM and CNN+GRU Model Inference

The second experiment aimed to demonstrate the superiority of utilizing temporal information throughout a Valorant round for prediction. Here, we input the rounds extracted from the Grand Finals into our CNN+LSTM and CNN+GRU models. Unlike the first experiment, this approach involved capturing every 30th frame and predicting the round's outcome based on the sequence formed by these frames up to that point. This method is significantly more computationally intensive and, as such, is not currently feasible for real-time analysis on our existing hardware. The CNN+LSTM and CNN+GRU models achieved an inference accuracy of 79% and 71% respectively. The outcomes of this experiment, highlighting the efficacy of temporal data utilization in prediction, are showcased in Figure 13 and Figure 14.

Figure 13. Experiment 2 LSTM+CNN model predictions

Figure 14. Experiment 2 GRU+CNN model predictions

These experiments played a critical role in validating the performance of our models and their potential application in real-world scenarios. While the ResNet model allows for real-time predictions, the CNN+LSTM and CNN+GRU models, despite their high computational intensity, offer a more comprehensive analysis by incorporating the temporal progression of the game.

6. Limitations

6.1. Implementation Limitations:

Incomplete Utilization of Available Screen Information: At present, our models are not utilizing all the relevant on-screen information available. However, we are planning to introduce future enhancements that will incorporate a multi-modal approach, which includes additional data from player information, round time, and map score displayed on the screen. We expect that this expansion will lead to an enriched learning capacity of our models and improve the accuracy of predictions.

Video Quality and Masking Imperfections: The videos we used in our study have a resolution of 720p. How-

ever, the 300x300 area that is focused on the mini-map tends to appear blurry, which makes it difficult to see the visual information clearly. Additionally, the manual creation of masks using tools like Photoshop and Paint has resulted in some imperfections. These masks sometimes allow screen effects to bleed into the map area, which adds noise to our dataset unintentionally.

Data Collection Inaccuracies: Collecting and editing videos manually could lead to human errors. For example, we found some videos had more than one round, causing us to remove them. This analysis assesses the quality of the dataset, considering possible mistakes in labeling and editing, particularly for images and videos from the same game.

6.2. Methodological Limitations:

Real-Time Prediction Challenges: Our ultimate goal of enabling real-time prediction of game outcomes using LSTM+CNN and GRU+CNN methods is currently unfeasible due to the substantial computational resources required for real-time sequence inference.

Frequent Game Updates: Valorant's popularity ensures regular game updates, including the introduction of new agents and maps, along with modifications to existing content. Since our data collection, the game has already added two new maps and one new agent. Keeping pace with such a dynamic game necessitates continuous data collection and frequent model retraining to stay current with the evolving gameplay dynamics.

7. Challenges and Unique Aspect

In this project, we aimed to innovate in the Esports domain by developing a real-time prediction model based on game footage analysis. This approach is novel in the rapidly evolving Esports industry, impacting areas such as coaching, analysis, and betting. Our project, aspires to provide real-time predictions by utilizing live footage and in doing so distinguishes itself from older methodologies reliant on historical data and statistics. This marks a departure from traditional methods based solely on retrospective data. The unique nature of this project required an understanding of deep learning, Esports, and the game Valorant, and presented several challenges.

A primary challenge we encountered was the extensive volume of footage, totaling over 78 hours, which required meticulous editing and labeling. This task was prone to human error, as evidenced by inaccuracies detected during data analysis. Consequently, we were compelled to regenerate the initial dataset for Phase 1. Additionally, our automated labeling script for reading CSV files initially misinterpreted headers as data entries, leading to mislabeled rounds. Identifying and rectifying this error involved rewriting the script, regenerating the dataset, and retraining the model, a process that was both resource-intensive and time-consuming.

Another significant hurdle was the uncertainty surrounding the need to retrain the ResNet for Phase 2. Initially, training focused solely on the LSTM layers, but the lack of learning progress, indicated by constant training and validation losses and the absence of convergence, prompted us to retrain layer 4 of the ResNet. This adjustment yielded more promising results.

Despite satisfactory outcomes with the LSTM model, the GRU model's performance was subpar, and both Phase 2 models exhibited a tendency to overfit the training data. We experimented with various strategies to mitigate this issue, including L2 regularization, adjusting batch sizes, experimenting with dropout values, applying batch normalization, and selectively freezing components within ResNet's layer 4. However, these efforts have yet to yield the desired improvements. Furthermore, the model training process is time-intensive, typically requiring 8-14 hours, which hampers the efficiency of hyperparameter tuning. Additionally, computational demands frequently led to CUDA out-of-memory errors, causing system crashes.

This project, despite its challenges, has contributed to our understanding of applying deep learning in Esports and highlights the potential for further advancements in this field.

8. Conclusion

This research represents a significant effort into Esports predictive analysis. It uses advanced deep learning methodologies to predict outcomes in Valorant matches.

During phase 1, a CNN was used to extract features from individual frames. In phase 2, hybrid models—CNN+LSTM and CNN+GRU—were introduced to analyze sequences of frames within a round. These models showed an enhanced ability to capture evolving tactics, player movements, and visual cues crucial for predicting round outcomes effectively. The proposed approach was able to predict game outcomes as conceptualized, proving the effectiveness of the strategy used. Through our experiments and the accuracy on the inference data, we can confirm that the sequence based methods show superior performance in predicting the outcome for an unseen dataset.

Although there have been some improvements, there is still room for further development. Future research should focus on integrating a wider range of in-game statistics, improving data quality, and ensuring that the model can cope with frequent game updates. These enhancements are essential for maintaining the model's relevance and accuracy in the rapidly evolving e-sports industry. Moreover, examining the integration of player-specific data and team strategies could provide a more nuanced understanding and prediction ability in this competitive field.

References

[1] Kodirjon Akhmedov and Anh Huy Phan. Machine learning models for dota 2 outcomes prediction. *arXiv preprint arXiv:2106.01782*, 2021.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[3] Tiffany D Do, Seong Ioi Wang, Dylan S Yu, Matthew G McMillian, and Ryan P McMahan. Using machine learning to predict game outcomes based on player-champion experience in league of legends. In *Proceedings of the 16th International Conference on the Foundations of Digital Games*, pages 1–5, 2021.

[4] Jacob Gildenblat and contributors. Pytorch library for cam methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

[5] Victoria J Hodge, Sam Devlin, Nick Sephton, Florian Block, Peter I Cowling, and Anders Drachen. Win prediction in multiplayer esports: Live professional match prediction. *IEEE Transactions on Games*, 13(4):368–379, 2019.

[6] Sang-Kwang Lee, Seung-Jin Hong, and Seong-Il Yang. Predicting game outcome in multiplayer online battle arena games. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1261–1263. IEEE, 2020.

[7] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[8] Pu Yang, Brent E Harrison, and David L Roberts. Identifying patterns in combat that are predictive of success in moba games. In *FDG*, 2014.