Enabling Real-Time Prediction of In-game Deaths through Telemetry in Counter-Strike: Global Offensive

Stefan Marshall
Tilburg University
Tilburg, The Netherlands
stefan@cdmgroup.nl

Paris Mavromoustakos Blom
Tilburg University
Tilburg, The Netherlands
p.mavromoustakosblom@uvt.nl

Pieter Spronck
Tilburg University
Tilburg, The Netherlands
p.spronck@uvt.nl

ABSTRACT

Esports have evolved into a major form of entertainment, drawing hundreds of millions of viewers to its online competitive broadcasts. Using Esports telemetry data to predict the outcome of a match is a well-researched topic, but *micropredictions* of specific in-game events are explored only sparingly. How accurately can we predict specific in-game events within a limited time window, and how can these predictions be used in a live broadcast? This research aims at predicting in-game deaths using telemetry data in Counter-Strike: Global offensive (CS:GO). We establish a data processing pipeline to acquire and re-structure raw in-game data and propose a set 36 features which will ultimately be used to predict in-game deaths within a three second window. Three neural network models are compared, namely convolutional (CNN), recurrent (RNN) and long short-term memory (LSTM). Our results show that the LSTM network has the best predictive accuracy (F_1 0.38) when prompted, for all 10 players of a competitive game of CS:GO. The predictions are most influenced by features related to a player's average in-game death count, health points, enemies in range and equipment value. Our model enables real-time micropredictions of deaths in CS:GO, and may be leveraged by Esports commentators and game observers to direct their focus on critical in-game events during a live competitive broadcast.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; *Machine learning*; • **Human-centered computing** → *User models.*

KEYWORDS

Esports analytics, Result prediction, Microprediction, CS:GO, Deep learning

ACM Reference Format:
Stefan Marshall, Paris Mavromoustakos Blom, and Pieter Spronck. 2022. Enabling Real-Time Prediction of In-game Deaths through Telemetry in Counter-Strike: Global Offensive. In *FDG '22: Proceedings of the 17th International Conference on the Foundations of Digital Games (FDG '22), September 5–8, 2022, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3555858.3555859

FDG '22, September 5–8, 2022, Athens, Greece

ACM ISBN 978-1-4503-9795-7/22/09…$15.00
https://doi.org/10.1145/3555858.3555859

1 INTRODUCTION

Competitive video games (Esports) have become one of the quickest-growing segments of the entertainment industry [9]. The market for Esports reached one billion US Dollar revenues in 2021 and an audience of over 720 million viewers, and it is forecasted to grow by 14% year-on-year in 2022 [25]. Among the most popular genres of games played in Esports competitions is the *First Person Shooter* (FPS), which can be defined as games where "a player navigates through a virtual world from a first person perspective and interacts in single- or multiplayer combat sequences with multiple enemies by using a range of weaponry in order to complete an objective" [26]. One such FPS is CS:GO, a game played by two teams of five players in seven different competitive maps. Although released over eight years ago, CS:GO still has over one million active players at present and its Esports competitions regularly attract over 200 thousand concurrent viewers [45].

Esports matches generate a high volume of telemetry data, which has enabled research on the topic of Esports analytics, the practice of mining match telemetry data in order to understand patterns of play and aid decision making during gameplay. Much of the research done in this field focuses on predicting the winner of a match, referred to as result prediction. Although a wide range of Machine Learning (ML) models are used in these works, state of the art seems to be using Deep Learning (DL) models [1, 42].

Despite being relatively young, the field of Esports analytics is attracting considerable research attention. The majority of scientific work in this domain revolves around result prediction in the Multiplayer Online Battle Arena (MOBA) genre [11, 35, 36], while despite their popularity, genres such as FPS and CS:GO in particular are lacking in research volume. In addition, the majority of work in Esports predictive modelling tends to focus on match-level or even multi-match longitudinal studies, while the real-time prediction of near-future game events (micropredictions) is scarcely researched. It is therefore uncertain whether the state of the art in Esports analytics is applicable to the FPS genre (particularly CS:GO) and capable of performing accurate in-game micropredictions. This paper contributes towards addressing this knowledge gap by using a large dataset[1] of competitive CS:GO matches and employing DL methods to enable real-time in-game micropredictions. Specifically, we propose a set of 36 features through which we implement a model that aims to predict player deaths within a three second time window.

Despite the low volume of works exploring micropredictions, we consider them to be of interest to Esports commentators (casters), in-game observers and their audience. During a live Esports

[1]Dataset publicly available for download at: www.kaggle.com/dataset/7360087bbc2bf244422c1fb9346e6233de32c5be99f7aaf83d812c423fcf8997

broadcast, casters must keep track of many different players at once (10 in the case of CS:GO) and provide commentary on important in-game events such as deaths, as they happen. CS:GO in particular is considered a fast-paced game where the maximum length of a round is less than two minutes and combat engagements last a few seconds. If a caster misses the opportunity to direct their focus towards combat in real-time, the viewing audience's experience may be negatively impacted. Through a model that can accurately predict in-game deaths within a short time window, casters may be able to allocate their focus more efficiently and briefly prepare their storytelling, which should in turn improve the audience's overall experience.

2 RELATED WORK

This chapter discusses work related to the current research. Prior work on the topics of Esports analytics and result prediction are reviewed in section 2.1. Section 2.2 explores research on the topic of micropredictions. The chapter is concluded with a review of state of the art deep learning interpretability techniques in section 2.3.

2.1 Esports analytics and result prediction

Esports matches generate a high volume of telemetry data, which is publicly accessible thanks to open APIs provided by game publishers. The broad availability of the data has enabled research on the topic of *Esports analytics*. This is defined as "the process of using Esports related data, [...], to find meaningful patterns and trends in said data, and the communication of these patterns using visualization techniques to assist with decision making processes" [20]. Such analysis is commonplace in professional Esports teams as well as tournaments, where commentators who provide analysis during and between matches are referred to as "statsmen". The work of Mahlmann et al. [20] establishes a framework for encounter-based analysis in DotA 2. As DotA 2 matches can exceed one hour in duration, extracting encounters from raw data enables them to use only the most relevant parts of a match to predict match outcomes. Using four features generated at the start of encounters, their best Logistic Regression (LR) model achieved 78% accuracy in predicting the outcome (winning team) of an encounter.

The majority of research done in Esports analytics focuses on result prediction. Semenov et al. [35] compared the performance of four ML algorithms for DotA 2 win predictions, namely 1) Naive Bayes, 2) LR, 3) Gradient Boosting and 4) Factorization Machines (FM), a general purpose predictor combining support vector machines with factorization models [31]. The comparison by [35] showed their FM model performing best with an AUC value of 0.706, closely followed by the boosting model at 0.701.

Hodge et al. [12] explored this topic further by using LR and Random Forests (RF) to conduct DotA 2 win predictions. Using only pre-match data, LR performed best with a 58.75% accuracy. Using in-game data from the first 20 minutes of gameplay and only professional match data, RF performed best with 74.59% accuracy, while with a mixed dataset of professional and casual matches, RF performed best with 77.51% accuracy. Overall, the models performed better than a random guess in predicting the result of a DotA 2 match. Seeing room for improvement, Hodge et al. [11] approached the task again with an expanded dataset and newly engineered features, and achieved peak prediction accuracy of 77.51% on the mixed dataset and 74.59% on the professional dataset after only five minutes of gameplay data, using an RF model. On both datasets, RF was followed closely by LR and gradient boosting in terms of performance.

Aside from traditional ML methods, some works explore result prediction through deep learning techniques as well. Akhmedov et al. [1] compared LR to both a RNN and a LTSM neural network. The LSTM model achieved 93% average accuracy in predicting the match outcome, versus 88% of the RNN and 82% of LR. Predicting wins in a different game (League of Legends) Silva et al. [36] found that a traditional RNN outperformed LSTM with a peak accuracy of 83.54% with gameplay data from minutes 20 through 25.

Although the majority of work on Esports analytics focuses on the MOBA genre, some do explore FPS and CS:GO in particular. Makarov et al. [21] utilized TrueSkill, a player skill ranking system to predict match outcomes, and achieved a peak accuracy of 0.62, although it is unclear whether the prediction was done post-game. Xenopoulos et al. [42] compared the performance of LR, gradient boosting and DL in CS:GO win prediction. Their results show similar performance in terms of log-loss between gradient boosting and NN with LR performing notably worse. They then employed their model to investigate optimal team spending decisions and define a performance metric, Optimal Spending Error (OSE) to rank how closely team spending decisions line up with predicted optimal spending decisions, and found that win probability is closely linked to OSE.

In conclusion, a wide variety of ML techniques have been applied to Esports analytics and result prediction, and in works that compared neural networks with traditional ML, the neural networks typically performed as well as the best traditional models or better. For that reason, in this paper we compare three types of neural networks, expecting them to yield high predictive accuracy in the context of our experiment.

2.2 Micropredictions

An intuitive next step after Esports match result prediction is to make more granular predictions about specific events taking place within short time windows during a match. This is referred to as micropredictions, as coined by Katona et al. [13]. The authors present a DL model that can predict whether individual players will die within a five second window during DotA 2 games. Telemetry data was collected from 10,000 (semi-)professional matches, from which a feature set of 287 features per player was created. These features include the player state (i.e strength, health points) and position, and player abilities (i.e cooldown times, ability usage costs). Additionally, approximately half of the feature space was used to represent the hero ID in a one-hot encoded form. The specific hero used by the player has unique benefits and drawbacks, and is therefore a valuable (but computationally costly) feature. The model presented is a feed-forward neural network, which achieved 0.377 precision and 0.725 recall on unseen data. This indicates that deep learning could be suitable for making micropredictions in Esports. The authors do note that the model is limited to post-game analyses or professional environments where a live feed of the telemetry data is available. A second limitation is that balance changes made

to the game over time may harm generalizability on newer game data.

The research by Katona et al. [13] expanded on the work of Cleghern et al. [4] who, using similar DotA 2 telemetry data, predicted changes in player health during a match. This was done by splitting health data into small and large changes, and predicting these separately with different models. For large changes, a combination of statistical methods was used to predict both the direction and the magnitude of the health change, while small changes were predicted using an auto-regressive moving average model. Their work is limited by the smaller dataset (542 matches) and limited featureset (only health data). Despite a peak accuracy of 77.2% in predicting small health changes five seconds into the future, it performed poorly predicting the timing for larger health changes 10 seconds into the future.

2.3 Deep learning model interpretability

As the performance of machine learning models approaches or even exceeds human accuracy in classification in prediction tasks, fields like healthcare, finance and criminal justice are warming up to the use of ML in their decision making. Advanced ML models often work in a *black-box* fashion, and the adoption of ML in these high-stakes fields may be hampered by the inability of end users to understand and value the output of the models. A prerequisite to building the necessary trust is the *interpretability* of a model. There is no generally accepted definition of this concept, but Lipton et al. [17] make an attempt to define the properties of an interpretable ML model. Interpretability, or transparency, can be considered at three levels, namely **1)** *simulatability*, or understanding of the model as a whole, **2)** *decomposability*, or understanding of the individual features, and **3)** *algorithmic transparency*, an understanding of the training algorithm. They note that the degree of simulatability is typically low for complex machine learning techniques like deep learning, and that "modern deep learning methods lack algorithmic transparency". Overall, their work indicates that there are substantial challenges in creating interpretable models and even in defining what interpretability means for a specific use case. Considering the current research focuses on using deep learning techniques for micropredictions in CS:GO, interpretability will be approached at the levels of simulatabality and decomposability. Algorithmic transparency will not be considered further.

How does one make a machine learning model interpretable? Ribeiro et al. [32] attempt to answer this question with LIME, an explanation technique can be used to explain the predictions of any machine learning classifier by outputting a list of explanations which reflect the contribution of individual features to a prediction. Their work showed that the use of LIME allowed non-experts to pick classifiers which would generalize reasonably well on real world data, and helped end users of neural networks understand when and why to trust a model. Since its inception, use of LIME has become commonplace, finding applications in various ML tasks such as early detection of Parkinson's disease [19] and predicting determinants of foreign direct investment inflow [37]. In assessing the effectiveness of the LIME framework on four state-of-the-art classification algorithms, Dieber et al. [6] interviewed end users who were not familiar with LIME and found that LIME did increase the interpretability of ML models, but that more work was needed in terms of usability.

Building off LIME and similar methods, Lundberg et al. [18] introduced their unified prediction interpretation framework: *SHapley Additive exPlanations*, or SHAP. This assigns an importance value to each feature, in order to interpret what features contribute to a specific prediction. On the assumption that a "good model explanation should be consistent with explanations from humans who understand that model", they compared the explanations from LIME, SHAP, and other methods to human explanations and found that SHAP values showed stronger agreement to human explanations than other methods. Since then, several works have demonstrated the reliability and utility of SHAP values in ML tasks such as RNA sequence tissue classifiers [46] and predictions from medical sensor data [3].

Several works attempt to compare the strengths of LIME and SHAP with mixed results [8, 22]. None of the works reviewed in section 2.1 utilized either LIME nor SHAP, and so it is not clear that either technique is a preferred choice for the application of result prediction or micropredictions.

3 EXPERIMENT

This chapter presents the game, dataset, feature extraction and evaluation methods that were used for in-game death prediction.

3.1 Counter-Strike: Global Offensive

Counter-Strike: Global Offensive (CS:GO) is the fourth game in the Counter-Strike series developed by Valve Software, released in 2012. It is a FPS where two teams of five players, namely Terrorists (T) and Counter-Terrorists (CT), face each other in armed combat. In Esports play, the game mode that is used is *DE*, where teams must either detonate (T) or defuse (CT) a bomb. Alternatively, a round can be won by completely eliminating the opposing team [41]. Players start with an $800 budget and choice of 34 guns divided into five categories, each with their own price and characteristics such as recoil and clip size. Depending on metrics like team and individual kills and deaths, players will start the next round with an increased or decreased cash budget (with a maximum of $16,000). It follows that good play leads to a larger budget, which allows for the purchase of more expensive and powerful weaponry.

In competitive play, a match consists of 30 rounds, with each round lasting a maximum of one minute and 55 seconds. The first team to win 16 rounds will win the match. In the case of a draw after 30 rounds, a maximum of 6 rounds of overtime will be played [33]. Professional championships are held throughout the year, with the biggest one known as the "Majors", sponsored by Valve with a prize pool up to $2 million [29]. Most CS:GO tournaments are streamed and freely accessible online, and match replay data is published after the matches on websites such as HLTV.org [10].

3.2 Dataset collection and preprocessing

A dataset was compiled from publicly available CS:GO match replay files (demofiles) published on HLTV.org. These files contain a serialization of the data transferred between clients (player PCs) and the game server during a match of CS:GO. As this includes all client inputs (such as attacks and movements) as well as server-side

Table 1: Overview of the processed dataset.

Metric	Amount
Matches	833
Rounds	21,300
Total datapoints	17,687,784
Positive class datapoints	425,107
Negative class datapoints	17,262,677
Deaths	141,702
Gameplay hours	491
Unique teams	227

events, the demofiles enable replaying a match with high accuracy. No discernment was made in the ranking of teams while collecting the match data, as all matches were played as part of tournaments with prize pools and therefore qualify as professional matches with, we assume, highly skilled teams. Although a CS:GO microprediction model would find most use in a real-time environment, a live telemetry feed of CS:GO matches is typically only available to the hosts or organizers of these matches, necessitating the use of postgame demofiles in the present research. The raw dataset contains nearly all competitive CS:GO matches broadcast on HLTV between September 6 and October 3 2021, save for an incidentally missed match due to connection errors. Table 1 shows on overview of the processed dataset.

The demofiles are an unstructured and low-level data stream that is not suitable for analysis and death prediction. To convert the demofiles into a usable format, the free *awpy* package by Xenopoulos et al. [43] is used, which converts the demofiles into a set of dataframes. Some demofiles failed to parse, either due to errors in the files or bugs in the parser. As this concerned less than 3% of the files, we excluded these files from our dataset. Once parsed, the resulting dataframes are saved in parquet files. The parquet file format is a part of the Hadoop ecosystem that allows for compressed, efficient columnar data representation [2], which is used to store the structured data in 2% of the storage space required for the raw data.

After parsing and restructuring, erroneously sampled rounds are removed from the dataframes. Some rounds may thus have been incorrectly recorded or parsed. Loss of connection of a player during the round can also introduce errors. All rounds that are shorter than 10 seconds, have different frame lengths between players, or are missing ticks (approximately 5% of all rounds) were removed. A manual review of failed rounds did not reveal a pattern of data loss, so it is assumed that the data cleaning did not introduce noteworthy bias to the dataset. All preprocessing steps mentioned in this chapter are conducted with the *pandas* package in Python [30].

3.3 Feature extraction and engineering

The parsed data is stored in several dataframes and contains hundreds of columns with many of them being duplicate or presented in a format not directly suitable for the purpose of classification through neural networks. We use our intuition and experience of how CS:GO is played to determine what features should be extracted or engineered. This is a non-exhaustive approach, but necessary as the large dimensionality of the raw data prevents timely training of a neural network, considering the hardware limitations (detailed in section 3.4). The features generated include a player's equipment and stats, enemies and teammates within certain range, and enemies' equipment. The full list of the 36 features generated can be seen in table 2.

Table 2: List of all features generated.

Feature	Explanation
hp	Remaining health points of the player
armor	Remaining armor points of the player
isBlinded	Whether the player is blinded
isAirborne	Whether the player is airborne
isDucking	Whether the player is ducking
isStanding	Whether the player is standing
isScoped	Whether the player is aiming down the scope
isWalking	Whether the player is walking (not running)
equipmentValue	Total player equipment value
cash	Remaining player cash
hasHelmet	Whether the player has a helmet
kills_from_avg	Distance of players kill count from average
deaths_from_avg	Distance of players death count from average
total_hp_enemy	Total health points of enemies
total_hp_team	Total health points of teammates
num_enemy_alive	The number of enemies alive
num_team_alive	The number of teammates alive
enemy_in_range_200	Number of enemies within 200 units range
enemy_in_range_500	Number of enemies within 500 units range
enemy_in_range_1000	Number of enemies within 1000 units range
enemy_in_range_2000	Number of enemies within 2000 units range
enemy_hp_in _range_500	Total health points of enemies within 500 units range
enemy_hp_in _range_1000	Total health points of enemies within 1000 units range
enemy_hp_in _range_2000	Total health points of enemies within 2000 units range
enemy_equipment _in_range_500	Total equipment value of enemies within 500 units range
enemy_equipment _in_range_1000	Total equipment value of enemies within 1000 units range
enemy_equipment _in_range_2000	Total equipment value of enemies within 2000 units range
team_in_range_200	Number of teammates within 200 units range
team_in_range_500	Number of teammates within 500 units range
team_in_range_1000	Number of teammates within 1000 units range
equipment_value_team	Total equipment value of teammates
equipment_value_enemy	Total equipment value of enemies
distance_closest_enemy	Distance to the closest enemy
hp_closest_enemy	Health points of the closest enemy
active_weapon	Active weapon category of the player (one-hot encoded)
weapon_closest_enemy	Active weapon category of the closest enemy (one-hot encoded)

The engineered features are exported into one parquet file per player per round. Categorical data (such as a player's active weapon) is one-hot encoded. We apply a sliding window with a lag of five steps, meaning that the dataset is restructured to include the current and the previous four steps, creating t-0 through t-4 instances of every feature. After one-hot encoding and windowing, the number of feature columns is 250.

The forecast variable is created as a binary variable where the negative class represents the player will not die within three seconds and the positive class represents the player will die within

three seconds. Three seconds is empirically chosen for the forecast window as a shorter window may limit the usefulness while a longer window sacrifices considerable predictive power as CS:GO combat encounters typically only last a few seconds. The matches are divided into a 70/15/15 train/validation/test split.

3.4 Neural network implementation

The neural networks used are implemented in TensorFlow (TF), a commonly used Python package for expressing machine learning algorithms and executing such algorithms [38]. We use Google Colab Pro+ to train the models as it provides affordable top-tier Graphics Processing Units (GPUs) that would otherwise not be accessible for this research. The use of high end GPUs is necessary to reduce training time as TF is optimized for GPU acceleration. This does introduce several practical restrictions for the research, namely **1)** relatively small storage available in a Colab VM, **2)** no guarantee of GPU availability (fair use policy) and **3)** a maximum session time of 24 hours. These restrictions cause limitations on the current research in terms of processed dataset size and hyperparameter tuning.

Three neural network architectures were implemented and compared against each other: A convolutional neural network (CNN), which is a type of neural network in which the layers utilize convolution, a mathematical operation that expresses the amount of overlap of one function as it is shifted over another function [39]. CNN are used for processing data with a known grid-like topology such as image data (a two dimensional grid of pixels) and time series data (one dimensional grids) [7]. CNNs are most often used for two dimensional data tasks like image classification and object detection [27]. One dimensional CNN find frequent application in classifying medical sensor data such as ECG signals [15] but can be used for any time series data such as inventory forecasting [44] and stock price prediction [24]. Although none of the reviewed literature on Esports result prediction or microprediction used CNN, the time series nature of the data used in the current research make CNN a relevant candidate.

Recurrent neural networks (RNN) are a broad class of neural networks using feedback loops that enable them to remember and learn from previous inputs. It is typically used for handling sequential or temporal data. It finds frequent use in natural language processing tasks, like spelling correction [14] and hate speech detection [5], as its memorization allows it to remember the context of a word or phrase. RNNs have found successful use in Esports result prediction in the works of Silva et al. [36] who achieved peak performance through an RNN model compared to an LSTM and of Akhmedov et al. [1] whose RNN model was outperformed by an LSTM network. These mixed results encourage further comparison of the two architectures.

Long short-term memory (LSTM) networks are a subclass of RNNs. They are intended to resolve the vanishing gradient problem encountered in traditional RNN. LSTM, like RNN, have found applications in natural language processing [40], but also in gain prediction [16] where LSTM showed higher accuracy but lower F_1 score than RNN. Applying LSTM, RNN and CNN to stock price prediction, [34] found that CNN performed best.

Overall, literature shows that all three DL model architectures are valid choices for time series data, and that the best architecture is highly application specific.

3.5 Hyperparameter tuning

A common way to adjust hyperparameters for a TF model is to use the *GridSearchCV* function from the scikit-learn [28] package in combination with the *fit* function in TF. This approach was not used because the rounds-based nature of the processed data necessitated the use of the *train_on_batch* function in TF which does not work in combination with GridSearchCV. In addition, the Colab session time limits prevent certain hyperparameters (like a larger number of epochs) to be tested succesfully. For these reasons we opted to keep hyperparameter tuning minimal, primarily using default settings and code samples from TF documentation [38]. The only hyperparameter we tuned was the number of epochs, where we experimented using values [*1, 2, 3, 4, 5, 8, 10*]. We expect that increasing the number of epochs will increase prediction accuracy, up to a hypothetical point of diminishing returns. Due to the aforementioned computational restrictions, the highest number of epochs that all models could efficiently be trained for was 5, so this number was used. The Adam optimizer is used for all models, as it is a common choice for time series data, utilizing the default values [*learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False*]. Furthermore, the *binary_crossentropy* loss function is used to compile the models. The batch size is the amount of rounds present in a match, due to the connected nature of rounds within a match, versus the relative disconnect between matches themselves. This allows the LSTM model to "learn" from connections between rounds as well, rather than purely the data from the current round.

To establish appropriate class weights to train the model, we manually sampled 20 randomly selected rounds and computed their class weights using *sklearn.compute_class_weight* function. This showed that the class weights ranged from {0:0.5, 1:2} to {0:0.5, 1:40}, a consistent weight for the negative class with a strongly varying weight of the positive class. We trained the models with class weights [*1:2, 1:10, 1:20, 1:40*] and found the best performance using class weights {0:0.5, 1:2}, so these weights were used.

3.6 Network architecture

The architecture, or layers of the neural networks are largely derived from TF documentation and default values. The detailed architectures for the models can be seen in Figure 1.

Most layers use *ReLU* activation as it is the standard in literature due to its all-round efficiency and suitability. The LSTM layers use *tanh* activation as they require this to enable GPU acceleration. The sigmoid output layer is chosen as it grants an output between values 0 of 1, suitable for the binary classification problem of CS:GO death prediction. Other choices (like padding, # of filters) were left to default or chosen as recommended in TF documentation.

3.7 Evaluation methods

A number of the works discussed in section 2.1 use accuracy as a metric for evaluating performance. This metric is inadequate for imbalanced datasets where the positive classification event is rare, such as deaths in CS:GO. Two commonly used metrics for

```
model_Conv1D =
tf.keras.Sequential([
tf.keras.layers.Conv1D(filters=128,
kernel_size=WINDOW_LENGTH,
activation='relu',
padding='causal',
input_shape=(WINDOW_LENGTH,
num_features)),
tf.keras.layers.Conv1D(filters=128,
kernel_size=WINDOW_LENGTH,
activation='relu',
padding='causal'),
tf.keras.layers.Flatten(),
```

```
model_RNN =
tf.keras.Sequential([
tf.keras.layers.SimpleRNN(128,
activation='relu',
input_shape=(WINDOW_LENGTH,
num_features),
return_sequences=True),
tf.keras.layers.SimpleRNN(128,
activation='relu'),
```

```
model_LSTM =
tf.keras.Sequential([
tf.keras.layers.LSTM(128,
activation='tanh',
input_shape=(WINDOW_LENGTH,
num_features),
return_sequences=True),
tf.keras.layers.LSTM(128,
activation='tanh'),
```

```
tf.keras.layers.Dense(64, activation = 'relu'),
tf.keras.layers.Dense(32, activation = 'relu'),
tf.keras.layers.Dense(16, activation = 'relu'),
tf.keras.layers.Dense(1, activation = 'sigmoid')])
```

Figure 1: Model showing the specific layers utilized in the three models. The same dense/output layers are used for all models.

imbalanced classification are precision (the fraction of positive classifications that were correct) and recall (the fraction of true events that were detected) [7]. One prior work on micropredictions [13] also evaluated model performance using precision and recall.

Considering the use case of an Esports caster using micropredictions as a supplement during CS:GO matches, it is not clear whether it would be preferable to prioritize for high recall or precision. Both false positive and false negative classifications could cause a caster to lose focus of an important gameplay event, harming the viewing experience. However, it is possible that false positives may indicate near-death events in the game which would still be of interest to the caster. If this is the case, a model skewed towards recall may be preferable, but this assumption must be tested by reviewing individual predictions of a model. We evaluate the performance of the three models by comparing the F_1 score (a function of precision and recall) on validation data. The best performing model is used on test data, and the ROC curve as well as AUC is included in the results. The ROC curve gives insight into the tradeoff between true positives and false positives, and offers a baseline comparison (No Skill Classifier), while the ROC AUC indicates the overall performance, enabling the comparison of the models to future works exploring CS:GO death prediction.

3.8 Interpretation frameworks

The review of interpretability frameworks in section 2.3 revealed two frameworks, LIME and SHAP, neither of which are used in the known Esports result prediction or microprediction literature. As the current research is dealing with Esports time series data, the SHAP package is not suitable as it cannot handle variable input lengths while CS:GO rounds do vary in length. For this reason, the LIME package [32] is used to manually review randomly selected rounds and inspect individual predictions to understand the features driving those predictions.

To make the data suitable to train the LIME explainer, the one-hot encoding must first be reversed, as LIME cannot use one-hot encoded features. Then, the prediction classes are downsampled using the pandas resample function with 500 samples. Finally, the explainer is trained on 50 random matches and the one-hot encoding is reapplied.

4 RESULTS

In section 4.1, the performance of the three neural networks is compared. Section 4.2 discusses the interpretation of the best model in terms of CS:GO features.

4.1 Neural network performances

Table 3 shows the performance metrics of the three tested models. We observe that the performance for the negative class on precision, recall and F_1 score are identical for the three models. These high scores on the negative class (no death) naturally occur due to the imbalance of the dataset. In classifying the positive class, the three models show comparable performance, with precision and recall numbers varying at the default threshold value of 0.5, but the F_1 scores show LSTM performing best at F_1 0.38, followed by CNN (0.37) and RNN (0.36).

The best performing model was shown to be the LSTM model. On test data, this model achieved a precision of 0.38, recall of 0.37 (at default threshold) and an F_1 score of 0.38. The LSTM model showed identical F_1 scores on training, validation and test data, indicating no sign of over- or underfitting.

Figure 2 shows the ROC curve for the LSTM model, illustrating the tradeoff between false positives and true positives. The AUC for this model is 0.92. Compared to the no skill classifier (0.5) serving as a baseline, the LSTM demonstrates considerably better performance.

Figure 3 illustrates the predictions made for all ten players during a round of a randomly selected CS:GO match. These show that

Table 3: Performance comparison of the three models on precision, recall and F_1 score at threshold value 0.5 on validation data.

Class	Model	Precision	Recall	F_1 score
0	CNN RNN LSTM	0.98	0.99	0.99
1	CNN	0.42	0.33	0.37
	RNN	0.54	0.27	0.36
	LSTM	0.39	0.36	**0.38**

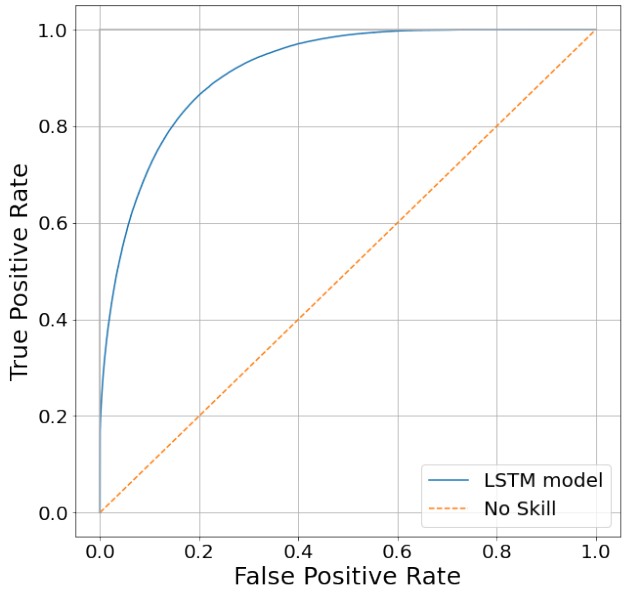

Figure 2: ROC curve for the LSTM model on test data compared to a No Skill Classifier

in the observed round, at the default threshold of 0.5, a positive classification only occurs one to two seconds before a death if at all, but at a lower threshold of 0.25, positive classifications could already occur around 5-10 seconds before the actual death. It can also be observed that death probabilities are greater than 0 for players that did not die during a round, likely because they were the victors in a combat encounter. For the round shown, the model predicted at least some (*>0.25*) probability of death 1-3 seconds before it occurred. From a manual review of numerous rounds, we conclude that the model is more likely to predict a death prematurely than to not predict it at all. The graph shown in figure 3 is typical of the matches we observed, and additional graphs can be found in [23].

4.2 Model interpretation

The LIME explainer offers an explanation of the driving features for a specific prediction. We randomly reviewed 20 predictions from different matches to gain an understanding of what features were

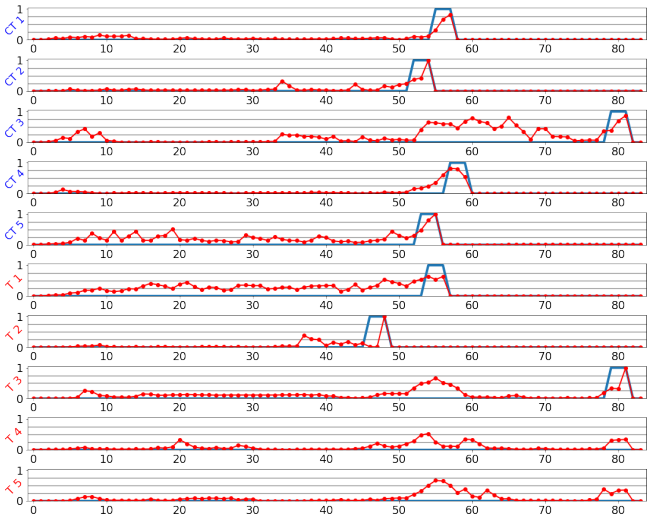

Figure 3: Death predictions for 10 players during the 3rd round of the match *young-ninjas-vs-saw-youngsters-m1-nuke* using the LSTM model. The red dotted line represents prediction probability of player death within 3 seconds. The blue line represents actual player death within 3 seconds. The sharp drop of the blue line indicates death (end of 3 second window). The grey lines indicate different prediction thresholds [*0.25, 0.5, 0.75*].

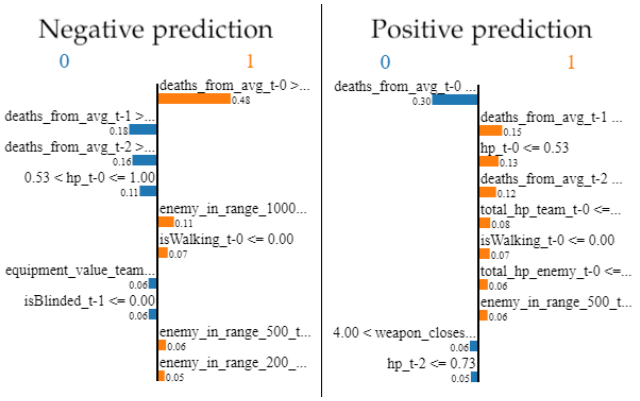

Figure 4: LIME Explainer scores for a randomly selected negative and positive prediction during the 22nd round of the match *young-ninjas-vs-saw-youngsters-m1-nuke* using the LSTM model. The features with orange bars weigh in favor of the positive class, while the blue bars contribute towards the negative class.

most influential. Figure 4 shows an example of the LIME explainer output for two predictions, one negative (close to 0) and one positive (near 1). Unfortunately, due to formatting limitations of the LIME package, the full feature list, names, and values could not always be displayed. Note that the LIME values (next to the bars) do not sum to 1 due to the output data of the model, but this does not affect its usability in interpreting the predictions. Some features seem

to appear multiple times as they are windowed 5 seconds into the past, creating t-0 to t-4 instances of every feature.

For the negative prediction, it can be noted that a value for deaths_from _avg_t-0 greater than 0.62 would indicate player death is likely. This suggests that if a player has died considerably more than average, their skill level is low enough that they may die even when other features weigh in their favor. It also shows that having over 53 health points, teammates having more valuable equipment, and not being blinded, are good indicators that a player will live. On the other hand, the number of enemies within ranges 1000, 500 and 200, as well as whether the player is running, all suggest the player may die.

The positive prediction utilizes a similar set of features with different values. Having fewer deaths than average weighs in favor of survival, while having more deaths, less than 53 health points, walking, low team health points and enemies within range 500 all contribute towards predicting player death. Interestingly, a weapon_closest_enemy value of 5 (the encoded value for Assault Rifles) weighs in favor of the player living. This suggests that this weapon type is ill suited for close-range encounters.

Overall, the above figures combined with several manually reviewed predictions indicate that the importance levels of the features that drive the predictions are logical, and align with our intuition and knowledge of how CS:GO is played.

5 DISCUSSION

The purpose of this research was to utilize deep learning to predict in-game deaths in CS:GO. Section 5.1 summarizes the key findings. In section 5.2, the limitations of this research are discussed.

5.1 Key findings

Out of the three models we compared, LSTM was found to be the best performing model (based on F_1 score) for predicting deaths in CS:GO. This is largely in line with expectation, as literature showed LSTM is considered to be an improvement on RNN. However, it should be noted that the differences in performance were small, at most 0.02 F_1 points. This is possibly due to same structure of dense layers used in all three models. Further hyperparameter tuning could affect the performance ranking of these models. This belief is strengthened by the fact that Silva et al.[36] and Akhmedov et al.[1] both used RNN and LSTM for Esports result prediction and showed conflicting results as to the best predicting model. Observing predictions from the LSTM model showed that at the default threshold value it will predict most deaths one second prior to occurring, whereas lowering the threshold will allow the model to anticipate deaths sooner, at the cost of increased false positives. This may be preferable for the Esports broadcasting use case, where a caster may benefit from false positives when they indicate a near-death experience. The runtime is approximately 5ms per 10 datapoints (one prediction for all 10 players in a match), indicating sufficient speed to perform these predictions in real-time, if a real-time data feed was available. Overall, we can conclude that deep learning can be used to predict deaths in CS:GO matches.

In terms of model interpretability, the LIME explainer showed that the typical features used to drive predictions were the amount of deaths from average, the amount of health points, enemies in a range, equipment value of the team and of enemies, whether the player was blinded and whether they were walking or running. These features all align with our intuition of CS:GO. Having fewer health points means a player is closer to death, more enemies nearby represents impending danger and things like sight or movement speed are easily understood as impacting a players survivability. There are several noteworthy finds in the features that are omitted from the LIME graphs as they provide less predictive power. Contrary to expectation, the number of kills from average, the number of teammates in a range and the active weapon of player and enemy all did not appear in any of the graphs produced by the LIME explainer, meaning they provided little predictive power. Despite this expectation, overall our intuition suggests that deaths from average, health points, enemies in a range, and equipment value can be reasonable indicators of upcoming player death.

5.2 Limitations

The current research succeeded in using deep learning for CS:GO death predictions. Despite this, we identify four potential limitations of this research, which may warrant further research.

Firstly, the evaluation metrics used in this research (precision, recall, F_1 score and ROC/AUC) don't give a complete picture of model performance for CS:GO death prediction, as different types of mispredictions may have a different impact on the Esports caster relying on these predictions. In their work, Katona et al. [13] classify mispredictions in four categories:

(1) False negatives where a player will die, but it was not predicted
(2) False positives where a player will die, but not within the three second window
(3) False positives where a player will not die, but the situation may be considered as dangerous
(4) False positives where a player will not die, and the situation is not considered as dangerous

A model producing a high number of category 1 errors can still be useful, as it will provide less frequent predictions with a higher certainty. A model frequently producing category 2 and 3 errors may arguably prove useful as it will still provide the caster with a relevant indication of tense gameplay moments. Finally, category 4 errors will harm the usefulness of the model in the Esports casting use case. To distinguish between category 3 and 4 errors, CS:GO domain expertise is needed to (visually) analyse these positive predictions and determine the danger of the combat encounters.

Secondly, the data processing pipeline and model rely on match demo files that are only published after a match is over. For the purpose of an Esports caster needing to conduct real-time death prediction during a match, these demo files would not be available. Live telemetry data is typically only available to the hosting institution of a match and its broadcasters and so was not used in this research. To fulfil this use case, both the pipeline and model need to be adjusted to utilize live data. Despite this limitation, our research has shown that our best performing model could in fact be used to conduct micropredictions in real time if such a datafeed were available, as 1) it does not rely on complete match data to predict deaths, and 2) it can deliver a prediction for all ten players in a match in 5ms.

Third, the LIME classifier only outputs feature importance values for specific predictions, and so does not provide a global overview of what features are most important. Unlike with SHAP, where there is precedent to simply sum and average values, it is not clear what the correct practice is for assessing LIME values in aggregate. This creates a reliance on manual review of a small subset of predictions and drawing conclusions from this. While usable, this is potentially less illustrative than an automated aggregation approach with a larger sample size.

Finally, the hyperparameter tuning done in the current research was minimal, due to the restrictions on computational resources. The downside of this is that any performance increase attainable with further hyperparameter tuning goes unexplored. A continuation of this research may benefit from more extensive hyperparameter tuning.

6 CONCLUSION

This paper presents three different neural networks aimed at predicting player in-game deaths in CS:GO in real-time. We establish a data processing pipeline to turn raw CS:GO match data into an ML ready dataset suitable for conducting micropredictions. Then, we compare the performance of the three models on this dataset, where LSTM shows the best performance in terms of F_1 score. The ROC curve shows that the LSTM model outperforms a "no skill" classifier and this, combined with further review of individual predictions, shows that the LSTM model may indeed be valuable for the use case of a storytelling aid in CS:GO Esports live casting. Through the LIME explainer, we discovered the most important features for predictions (*deaths from avg, health points, enemies in range, equipment value, is blinded and is walking*) and established that the model operates in an interpretable way that is consistent with our intuition of CS:GO gameplay. Overall, we conclude that deep learning can be used to predict in-game deaths in CS:GO, and that the extent to which it succeeds in this is dependent on hyperparameter tuning. We furthermore conclude that the predictions of the neural network can successfully be interpreted in terms of the relative importance of CS:GO features through use of the LIME explainer. Further research may explore additional hyperparameter tuning to improve predictive performance. It may also explore generating additional features or parsing the demo data at a higher resolution, or combining LIME data from many predictions in order to create a global rather than local ranking of feature importance.

REFERENCES

[1] Kodirjon Akhmedov and Anh Huy Phan. 2021. Machine learning models for DOTA 2 outcomes prediction. *arXiv:2106.01782 [cs]* (June 2021). http://arxiv.org/abs/2106.01782 arXiv: 2106.01782.

[2] Apache. 2021. Apache Parquet. https://parquet.apache.org/documentation/latest/

[3] Michal Bugaj, Krzysztof Wrobel, and Joanna Iwaniec. 2021. Model Explainability using SHAP Values for LightGBM Predictions. In *2021 IEEE XVIIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. IEEE, Polyana (Zakarpattya), Ukraine, 102–106. https://doi.org/10.1109/MEMSTECH53091.2021.9468078

[4] Zach Cleghern, Soumendra Lahiri, Osman Özaltin, and David L. Roberts. 2017. Predicting future states in DotA 2 using value-split models of time series attribute data. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. ACM, Hyannis Massachusetts, 1–10. https://doi.org/10.1145/3102071.3102095

[5] Amit Kumar Das, Abdullah Al Asif, Anik Paul, and Md. Nur Hossain. 2021. Bangla hate speech detection on social media using attention-based recurrent neural network. *Journal of Intelligent Systems* 30, 1 (April 2021), 578–591. https://doi.org/10.1515/jisys-2020-0060

[6] Jürgen Dieber and Sabrina Kirrane. 2020. Why model why? Assessing the strengths and limitations of LIME. *arXiv:2012.00093 [cs]* (Nov. 2020). http://arxiv.org/abs/2012.00093 arXiv: 2012.00093.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org

[8] Alex Gramegna and Paolo Giudici. 2021. SHAP and LIME: An Evaluation of Discriminative Power in Credit Risk. *Frontiers in Artificial Intelligence* 4 (Sept. 2021), 752558. https://doi.org/10.3389/frai.2021.752558

[9] Kirstin Hallmann and Thomas Giel. 2018. eSports – Competitive sports or recreational activity? *Sport Management Review* 21, 1 (Jan. 2018), 14–20. https://doi.org/10.1016/j.smr.2017.07.011

[10] HLTV. 2021. HLTV. https://www.hltv.org/

[11] Victoria Hodge, Sam Devlin, Nick Sephton, Florian Block, Peter Cowling, and Anders Drachen. 2019. Win Prediction in Multi-Player Esports: Live Professional Match Prediction. *IEEE Transactions on Games* (2019), 1–1. https://doi.org/10.1109/TG.2019.2948469

[12] Victoria Hodge, Sam Devlin, Nick Sephton, Florian Block, Anders Drachen, and Peter Cowling. 2017. Win Prediction in Esports: Mixed-Rank Match Prediction in Multi-player Online Battle Arena Games. *arXiv:1711.06498 [cs]* (Nov. 2017). http://arxiv.org/abs/1711.06498 arXiv: 1711.06498.

[13] Adam Katona, Ryan Spick, Victoria J. Hodge, Simon Demediuk, Florian Block, Anders Drachen, and James Alfred Walker. 2019. Time to Die: Death Prediction in Dota 2 using Deep Learning. In *2019 IEEE Conference on Games (CoG)*. IEEE, London, United Kingdom, 1–8. https://doi.org/10.1109/CIG.2019.8847997

[14] A. Cumhur Kinaci. 2018. Spelling Correction Using Recurrent Neural Networks and Character Level N-gram. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, Malatya, Turkey, 1–4. https://doi.org/10.1109/IDAP.2018.8620899

[15] Dan Li, Jianxin Zhang, Qiang Zhang, and Xiaopeng Wei. 2017. Classification of ECG signals based on 1D convolution neural network. In *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, Dalian, 1–6. https://doi.org/10.1109/HealthCom.2017.8210784

[16] Chen Lin and Min Chi. 2017. A Comparisons of BKT, RNN and LSTM for Learning Gain Prediction. In *Artificial Intelligence in Education*, Elisabeth André, Ryan Baker, Xiangen Hu, Ma. Mercedes T. Rodrigo, and Benedict du Boulay (Eds.). Vol. 10331. Springer International Publishing, Cham, 536–539. https://doi.org/10.1007/978-3-319-61425-0_58 Series Title: Lecture Notes in Computer Science.

[17] Zachary C. Lipton. 2017. The Mythos of Model Interpretability. *arXiv:1606.03490 [cs, stat]* (March 2017). http://arxiv.org/abs/1606.03490 arXiv: 1606.03490.

[18] Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. *arXiv:1705.07874 [cs, stat]* (Nov. 2017). http://arxiv.org/abs/1705.07874 arXiv: 1705.07874.

[19] Pavan Rajkumar Magesh, Richard Delwin Myloth, and Rijo Jackson Tom. 2020. An Explainable Machine Learning Model for Early Detection of Parkinson's Disease using LIME on DaTSCAN Imagery. *Computers in Biology and Medicine* 126 (Nov. 2020), 104041. https://doi.org/10.1016/j.compbiomed.2020.104041

[20] Tobias Mahlmann, Matthias Schubert, and Anders Drachen. 2016. Esports Analytics Through Encounter Detection. Boston.

[21] Ilya Makarov, Dmitry Savostyanov, Boris Litvyakov, and Dmitry I. Ignatov. 2018. Predicting Winning Team and Probabilistic Ratings in "Dota 2" and "Counter-Strike: Global Offensive" Video Games. In *Analysis of Images, Social Networks and Texts*, Wil M.P. van der Aalst, Dmitry I. Ignatov, Michael Khachay, Sergei O. Kuznetsov, Victor Lempitsky, Irina A. Lomazova, Natalia Loukachevitch, Amedeo Napoli, Alexander Panchenko, Panos M. Pardalos, Andrey V. Savchenko, and Stanley Wasserman (Eds.). Vol. 10716. Springer International Publishing, Cham, 183–196. https://doi.org/10.1007/978-3-319-73013-4_17 Series Title: Lecture Notes in Computer Science.

[22] Xin Man and Ernest P. Chan. 2021. The Best Way to Select Features? Comparing MDA, LIME, and SHAP. *The Journal of Financial Data Science* 3, 1 (Jan. 2021), 127–139. https://doi.org/10.3905/jfds.2020.1.047

[23] Stefan Marshall. 2022. *Not so sudden death: Death prediction in CS:GO*. Master's thesis. Tilburg University, Tilburg.

[24] Sidra Mehtab, Jaydip Sen, and Subhasis Dasgupta. 2020. Robust Analysis of Stock Price Time Series Using CNN and LSTM-Based Deep Learning Models. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, Coimbatore, India, 1481–1486. https://doi.org/10.1109/ICECA49313.2020.9297652

[25] Newzoo. 2021. *Global Esports & Live Streaming Market Report*. Technical Report. https://newzoo.com/insights/trend-reports/newzoos-global-esports-live-streaming-market-report-2021-free-version

[26] David B. Nieborg. 2006. First Person Paradoxes: The Logic of War in Computer Games. In *Game Set and Match II. On Computer Games, Advanced Geometries and Digital Technologies*. 107–115.

[27] Keiron O'Shea and Ryan Nash. 2015. An Introduction to Convolutional Neural Networks. *arXiv:1511.08458 [cs]* (Dec. 2015). http://arxiv.org/abs/1511.08458

arXiv: 1511.08458.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[29] Adam Rathbun. 2021. 7 Largest CS:GO Competitions In The World. https://geekinsider.com/7-largest-cs_go-competitions-in-the-world/

[30] Jeff Reback, Jbrockmendel, Wes McKinney, Joris Van Den Bossche, Tom Augspurger, Phillip Cloud, Simon Hawkins, Gfyoung, Matthew Roeschke, Sinhrks, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Patrick Hoefler, Shahar Naveh, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, JHM Darbyshire, Richard Shadrach, Marco Edward Gorelli, Fangchen Li, Matthew Zeitlin, Vytautas Jancauskas, Ali McMaster, Pietro Battiston, and Skipper Seabold. 2021. pandas-dev/pandas: Pandas 1.3.4. https://doi.org/10.5281/ZENODO.3509134

[31] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*. IEEE, Sydney, Australia, 995–1000. https://doi.org/10.1109/ICDM.2010.127

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938 [cs, stat]* (Aug. 2016). http://arxiv.org/abs/1602.04938 arXiv: 1602.04938.

[33] M. Nazhif Rizani and Hiroyuki Iida. 2018. Analysis of Counter-Strike: Global Offensive. In *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*. IEEE, Pangkal Pinang, 373–378. https://doi.org/10.1109/ICECOS.2018.8605213

[34] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. 2017. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Udupi, 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078

[35] Aleksandr Semenov, Peter Romov, Sergey Korolev, Daniil Yashkov, and Kirill Neklyudov. 2017. Performance of Machine Learning Algorithms in Predicting Game Outcome from Drafts in Dota 2. In *Analysis of Images, Social Networks and Texts*, Dmitry I. Ignatov, Mikhail Yu. Khachay, Valeri G. Labunets, Natalia Loukachevitch, Sergey I. Nikolenko, Alexander Panchenko, Andrey V. Savchenko, and Konstantin Vorontsov (Eds.). Vol. 661. Springer International Publishing, Cham, 26–37. https://doi.org/10.1007/978-3-319-52920-2_3 Series Title: Communications in Computer and Information Science.

[36] Antonio Luis Cardoso Silva, Gisele Lobo Pappa, and Luiz Chaimowicz. 2018. Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks.

[37] Devesh Singh. 2021. Interpretable Machine-Learning Approach in Estimating FDI Inflow: Visualization of ML Models with LIME and H2O. *TalTech Journal of European Studies* 11, 1 (May 2021), 133–152. https://doi.org/10.2478/bjes-2021-0009

[38] TensorFlow. 2021. TensorFlow. https://doi.org/10.5281/ZENODO.4724125

[39] Eric W. Weisstein. 2021. Convolution. https://mathworld.wolfram.com/Convolution.html

[40] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. *arXiv:1508.01745 [cs]* (Aug. 2015). http://arxiv.org/abs/1508.01745 arXiv: 1508.01745.

[41] Wikipedia. 2021. Counter-Strike: Global Offensive. https://en.wikipedia.org/wiki/Counter-Strike:_Global_Offensive

[42] Peter Xenopoulos, Bruno Coelho, and Claudio Silva. 2021. Optimal Team Economic Decisions in Counter-Strike. *arXiv:2109.12990 [cs]* (Sept. 2021). http://arxiv.org/abs/2109.12990 arXiv: 2109.12990.

[43] Peter Xenopoulos, Harish Doraiswamy, and Claudio Silva. 2020. Valuing Player Actions in Counter-Strike: Global Offensive. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, Atlanta, GA, USA, 1283–1292. https://doi.org/10.1109/BigData50022.2020.9378154

[44] Ning Xue, Isaac Triguero, Grazziela P. Figueredo, and Dario Landa-Silva. 2019. Evolving Deep CNN-LSTMs for Inventory Time Series Prediction. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Wellington, New Zealand, 1517–1524. https://doi.org/10.1109/CEC.2019.8789957

[45] Sergey Yakimenko. 2021. The growth in popularity of CS:GO since 2019. https://escharts.com/blog/growth-popularity-csgo-one-and-half-years

[46] Melvyn Yap, Rebecca L. Johnston, Helena Foley, Samual MacDonald, Olga Kondrashova, Khoa A. Tran, Katia Nones, Lambros T. Koufariotis, Cameron Bean, John V. Pearson, Maciej Trzaskowski, and Nicola Waddell. 2021. Verifying explainability of a deep learning tissue classifier trained on RNA-seq data. *Scientific Reports* 11, 1 (Dec. 2021), 2641. https://doi.org/10.1038/s41598-021-81773-9