Formalization of Tactical Representations in Counter-Strike through Data Preprocessing and Labeling

POVILAS KIRNA, University of Twente, The Netherlands

This paper presents a methodology for preprocessing and labelling raw Counter-Strike: Global Offensive (CS:GO) replay files (demos) for tactical analysis. Such analysis is a key part of all kinds of sports; that includes eSports titles. We developed a tool that enables users to label CS:GO demos and automatically predict strategies executed by the terrorist team. This is done by a series of scripts that convert raw data into structured graphs, which are then processed by a Graph Neural Network (GNN) model trained to classify tactics. It makes the job of analysts easier, understanding opponents' tactics more efficient, and enhances learning crucial insights from replays. Our contributions include the first tactically labelled CS:GO demos dataset, a GNN model optimised for tactic prediction, and thus impact eSports analytics in a way that is completely new.

Additional Key Words and Phrases: Labelling, Preprocessing, CS:GO, Dataset, Tactics, Pipeline.

1 INTRODUCTION

Electronic sports (eSports) is a growing sector projected to grow steadily to USD 5-6 billion by 2029, according to different sources [9, 10]. To this day, it attracts millions of people to participate in the events and live streams [19]. The most successful eSports competitions include Fortnite, Valorant, Dota 2, League of Legends (LoL), and Counter-Strike: Global Offensive (also known as CS:GO and now rebranded as Counter-Strike 2 or CS2).

Most video games that are played in a competitive eSports space are tactical, real-time strategy, or fighting games in which each player controls an individual character and works together with their team to accomplish a common goal like capturing an area, killing all enemy players, or planting/defusing a bomb [22].

Besides players' performance, there are other factors, such as character abilities and purchased equipment, that lead to a multitude of different scenarios, hence the complexity of tactical analysis.

Forecasting is a key piece in all kinds of sports [24]. Today, most pro teams have an analytics department or employ analysts, and these people will observe opponents in order to extract key information to better prepare for the next event, similar to analysts working in physical sports [24].

While in sports such as the NBA or NFL the key factors are players' individual performance, chosen tactic, and team composition [24]. In eSports it can be more complex due to the fact that there are various utilities and abilities, as well as the team composition and chosen tactic.

We chose to analyse CS:GO because it offers a unique combination of tactical depth, large publicly available datasets, and a well-understood game structure that makes it a suitable candidate for studying team-based decision-making and tactics.

A round of CS:GO consists of two teams, Terrorists and Counter-Terrorists (further T and CT), competing to achieve specific objectives such as planting or defusing a bomb or eliminating the opposing team. Each round involves purchasing equipment, planning strategies, executing tactics, and adapting to opponents' moves, all within a limited time.

Our goal is to find a way to efficiently process this data, label the most relevant tactics, apply GNNs [17] that use machine learning, and predict which tactic is used at any time during a round of CS:GO. Without the labelled data, supervised learning models cannot be trained easily. Even though the complexity of the data and gameplay presents a significant challenge, we aimed to successfully address this and build a strong foundation for automated tactical analysis. Therefore, we created the first tactically labelled dataset of CS:GO matches, upon which the same techniques can be used to further research this topic and expand this dataset, as well as expand our understanding of what tactics are effective and why. By achieving this, analysts and researchers could better understand team behaviours, plan accordingly, and better coach teams on tactical matters. Furthermore, our new dataset enables improvements in performance and accuracy in existing and further research, as it will provide higher-quality data about tactical behaviour.

My research was done using Design Science Research Methodology [15]. Which is a way to build and evaluate innovative artefacts to solve a particular problem. The goal is to design a solution that is both technically sound and effective as well as valuable in an academic sense, using iterative design, implementation, and assessment.

2 BACKGROUND

2.1 CS:GO

CS:GO is one of the most popular eSports titles [20], which is a multiplayer tactical first-person shooter. Where two teams Terrorists (T) and Counter-Terrorists (CT), are fighting against each other. Terrorist team has one main goal: plant the bomb and defend it until it explodes. The CT team, on the other hand, tries to stop them [26].

CS:GO is a suitable candidate for tactical analysis because of its emphasis on strategy, spatial positioning, and coordinated play. Each round is different and therefore teams are forced to make real-time decisions based on factors such as money, map control, timing, utility usage (e.g., smoke grenades, molotovs, and flashbangs), etc.

Bayes Esports [23] defined a CS:GO tactic as a coordinated plan executed by a team to achieve a round objective. Common examples include *"rush B"* (a fast attack on B site), *"split A"* (dividing the team to pressure the A site from multiple angles), or *"default"* (a passive setup aimed at gathering information and punishing overaggressive opponents). Each tactic involves predefined players taking on

TScIT 43, July 4, 2025, Enschede, The Netherlands

required roles, using utility, and correct timing [8]. Usually success in a round depends not only on individual skill but also on the behaviour of the team as a whole; in other words, execution of tactics. These types of structured patterns are what make CS:GO a suitable candidate for tactical analysis and why the identification and labelling of these tactics are important for further research. In our research, CS:GO was chosen for structured tactical analysis for a few key reasons:

(1) **High-volume and high-dimensional data**. High-dimensional data, according to [21] is a dataset where the number of features can be even larger than the number of observations.
(2) **Public availability of these demos**. We have a huge sample of publicly available demos in the *Esports Trajectories & Actions* [27] GitHub repository, with over 1500 available games that last on average 34 minutes. Sampled at 2 Hz, it gives us a dataset full of well-documented statistics totalling over 20 GB.

2.2 eSports Analytics and Machine Learning

Teams invest a lot of time into preparation, as it is part of their daily routine, where players spend hours practising, mastering their role, and understanding maps and mechanics, as well as doing tactical analysis on their own gameplay and that of their opponents through video-on-demand (VOD) review [25]. This analytical approach mirrors that of traditional sports, where understanding patterns and strategies plays a critical role in competitive success.

In recent years, machine learning (ML) has emerged as a promising tool in eSports analytics. Studies have been conducted focusing on ML applications in eSports [7, 12, 24] and others. Many of those focus on predicting the outcome of the game/round, ranking player performance, or detecting cheating behaviour.

In theory, ML is a branch of AI that allows machines to learn using algorithms [6]. This research makes use of supervised learning, a type of ML model that needs a lot of well-labelled data. [6] If the model is trained on a small sample size, there could be biases, and further predictions would be unreliable. Then, for every example input, an appropriate output can be computed. This enables the model to generalise and predict the right tactic on unseen data. We chose supervised learning because it allows the training model to recognise in-game tactics if we provide a well-labelled dataset.

2.3 Graph Neural Networks and Tactical Representation

Real-world data can be represented as a graph in many cases; objects are often defined in terms of their connection to other things [17]. The Counter-Strike game state at any point in time t, can be represented as a graph G, defined by its vertices/nodes V and edges E. Vertices are players, bombsites, bomb, and edges represent the positional distances between the vertices. All of this adds up to a *graph*. Researchers [18] have developed a neural network model called *Graph neural network* (GNN). This model can process multiple kinds of data using supervised learning techniques.

GNNs are especially useful in analysing various eSports, but they require labelled data [4]. Therefore, applying GNN models could work for CS:GO tactical representations, as you load more information into the graph nodes, global representation performance improves [17]. We have extensive data about players, their positions, equipment, in-game events, and other interactions, which can all be meticulously organised and arranged into detailed graphs that help predict an outcome — in our case *team's chosen tactic*.

2.4 The Gap: Lack of Labeled Tactical Data

There is a lot of raw data from HLTV formerly Half-Life Television [13] which is a CS:GO news portal storing recordings of Counter-Strike matches since 2002. HLTV now includes news, statistics, rankings, and analysis for the professional eSports scene of CS:GO.

More recently, a well-curated dataset ESTA [27] has been created containing a lot of useful data for the purpose of our analysis. However, still to this day, there is no good dataset of well-labelled CS:GO tactics that can be used to train an ML model. Therefore, it is an essential gap that we wanted to bridge to enable supervised learning on tactical behaviour in CS:GO.

Tactics matter because they capture the strategic intent of teams, which is often the decisive factor in professional-level matches. Understanding not just "what happened" but "why it happened" requires insight into the tactic being executed — something that performance-based or outcome-driven models (e.g., [7, 8, 12]) overlook. These models operate with aggregated features and cannot distinguish between a failed execution of a good tactic and a poor tactical choice. Our work addresses this critical gap by supplying ground-truth tactical annotations aligned to gameplay, enabling more accurate and interpretable models.

2.5 Comparison with Related Work

Most prior work in esports analytics for *CS:GO* has centred on numerical performance metrics or outcome prediction, with relatively little attention to high-level team tactics. For example, Bednarek et al. [5] addressed the fundamental challenge of parsing and integrating raw CS:GO game records (which are complex and undocumented) to enable subsequent player performance analysis. Their focus was on data preprocessing and cleaning, including matching in-game player data with external sources like HLTV, as a prerequisite to any performance evaluation. In a follow-up study, Bednarek et al. [4] quantified player performance by decoding game recordings and linking multiple data sources. These studies proposed pipelines for extracting structured data from demos and computing performance metrics, but they remained purely quantitative—focusing on kills, deaths, economy, and ratings—without modelling team tactics or coordinated actions.

Other researchers focused on predicting match outcomes using machine learning. Björklund et al. [8] showed that match results could be predicted by clustering player play styles and analysing team compositions using neural networks. Makarov et al. [12] compared ML-based predictors with ranking systems like TrueSkill for forecasting team victories. Birant and Birant [7] proposed a multi-instance learning framework to predict wins based on team-level player combinations. While these works achieved high predictive accuracy, they reduced the game to a classification problem and did not engage with the strategic intent of the teams.

The ESTA dataset, introduced by Xenopoulos and Silva [28], is among the largest CS:GO spatiotemporal datasets. It includes detailed player trajectories and actions from professional matches, extracted using the open-source awpy parser. While ESTA supports benchmarks for round-level predictions, it lacks tactical annotations and does not support direct analysis of strategic behaviour.

In contrast, Anzer et al. [1] and Raabe et al. [16] explored tactical pattern detection using graph-based methods, particularly in traditional sports. These studies demonstrate the potential of Graph Neural Networks (GNNs) to identify coordinated actions (tactics) such as overlapping runs or pressing formations. However, no comparable research currently exists in CS:GO that applies GNNs to identify tactical behaviour or uses datasets with labelled tactical categories. See table 1 for existing research comparison.

Our work fills this gap by proposing a method to preprocess CS:GO demos into graph-structured data and label rounds with predefined tactics (e.g., *A Long Rush*, *Mid to B Execute*). Using these labelled graphs, we trained a supervised GNN model to recognise these tactics, contributing both a novel dataset and a new model architecture for tactic identification.

Unlike previous works that used match-level or player-level statistics as model input, we use tactics labels that structure the round's context. For example, while Björklund et al. [8] cluster player behaviours and predict win probability, they do so without accounting for the strategic plan executed in each round. Integrating tactical labels could help in predicting whether round outcomes were due to player performance or team coordination. In this way, our work enables future studies to enrich existing predictive pipelines with strategy-level variables.

In summary, while prior research focuses on win prediction and player-level statistics, our work introduces a high-level conceptual layer that captures the actual *strategic intent* of CS:GO teams. We bridge the gap between raw positional data and interpretable tactical behaviour, enabling supervised tactic recognition in esports for the first time using a GNN framework.

3 PROBLEM STATEMENT

In traditional sports, analysts have access to rich, structured, and labelled datasets that help ML models to predict performance, analyse tactics, and plan strategies. In contrast, while eSports such as CS:GO generate a huge amount of match data and public demos containing raw data are available, there is no tactically labelled CS:GO dataset. This limitation is a key barrier for training a new supervised learning ML model, especially a GNN-based model, which requires labelled data to correctly predict and identify the tactic used by the T team in-game.

My goal was to create a method to preprocess raw CS:GO gameplay data and find a systematic approach to labelling rounds of CS:GO with relevant tactics. Thus, I created a tactically labelled dataset. This dataset formed a foundation for training a GNN model that is capable of identifying tactics during matches. By creating this well-curated dataset, I bridged the gap between raw eSports data and the increasing demands of supervised learning ML models.

3.1 Research Questions

Main Research Question:

- How can we preprocess and label CS:GO match data to enable supervised learning of team tactics using Graph Neural Networks?

Sub-Questions:

(1) How can tactics in CS:GO rounds be formally defined and categorised for labelling purposes?
(2) What preprocessing steps are necessary to convert raw CS:GO demo data into a graph-based format suitable for GNN input?
(3) How can labelling consistency and quality be ensured across a large dataset of rounds?

4 METHODOLOGY

I worked on preprocessing existing raw data to fit our GNN model input and making a prototype for accurately labelling CS:GO tactics from a snapshot of a game. This research was conducted according to Design Science Research methodology [15] as shown in figure 1. This methodology guides the process of identifying the problem, defining the objectives, designing and developing a solution, and lastly presenting and evaluating it. This methodology was applied for my thesis work and how I answered my research questions.

Problem Identification & Motivation

The problem of missing tactically labelled datasets in the CS:GO domain has been outlined in section 2.4. It hinders the development of novel supervised learning ML models that require structured and labelled data. The main objective was to enable supervised learning by creating a method to preprocess raw data from CS:GO matches into graph-structured data and label team tactics.

SRQ1 (Tactic definition and categorisation: We developed a formal definition and classification of CS:GO tactics based on game analysis, expert interviews, and prior research. Tactics file is a *.json* format file containing a list of objects with key-value pairs. Every tactic object consists of an ID, name, and description key-value pairs. Name is self-explanatory; it is used as a label in our labelling GUI. Description is for more information about the tactic (e.g. All players rush through the long doors), and lastly, ID is the most important part because it is used in various places to correctly identify the tactic, such as round frame labels and graph features such as "*tactic_used*" and GNN input. All of the tactic labels can be found in Appendix D.

Design & Development

Preprocessing the unprocessed CS:GO demo data was my main duty. For the design phase I prepared a set of functional and non-functional requirements that were derived from project goals.

Functional requirements:

- **F1:** Support parsing and processing of raw CS:GO demo files into structured, machine-learning-ready graph data.
- **F2:** Provide a graphical user interface (GUI) for efficient and intuitive per-frame tactical labelling.
- **F3:** Enable tactic labelling at frame-level precision across entire matches.

Study	Focus Area	Data Source	Model/Approach
Bednarek et al. (2017) [5]	Data preprocessing	CS:GO demos + HLTV	Parsing + matching
Bednarek et al. (2018) [4]	Player evaluation	CS:GO demos + stats	Custom metrics
Björklund et al. (2018) [8]	Win prediction	FACEIT match demos	k-means + DNN
Makarov et al. (2018) [12]	Win prediction	Game stats	ML classifier + TrueSkill
Birant & Birant (2023) [7]	Team outcome prediction	Public esports data	Multi-instance learning
Xenopoulos & Silva (2022) [28]	Dataset construction	Pro match demos (ESTA)	Parsing + benchmarks
Anzer et al. (2022) [1]	Tactical pattern detection	Soccer tracking data	VAE + GNN
Raabe et al. (2023) [16]	Tactical graph modeling	Sports trajectory data	Tactical Graph Networks
This Work	**Tactic identification**	**Pro match demos (ESTA)**	**Supervised GNN**

Table 1. Comparison of related studies in esports/sports analytics.

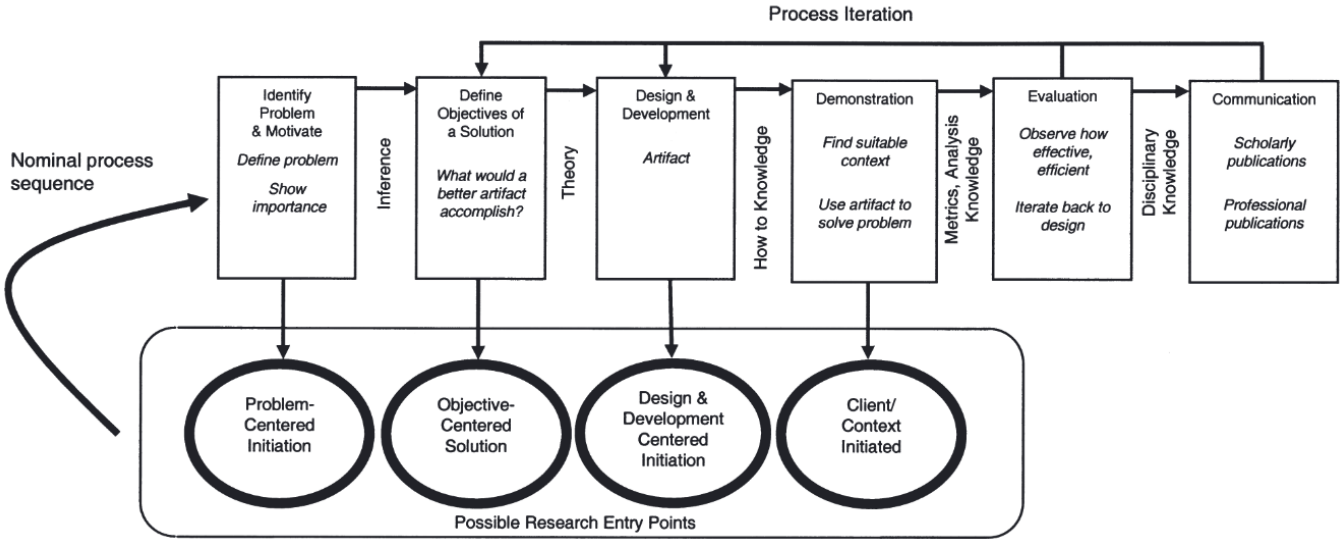

Fig. 1. DSRM Process Model

- **F4:** Output standardised file formats compatible with GNN training pipelines.

Non-functional requirements:

- **NF1 (Modularity):** The architecture should be modular to support extension to other maps or games with minimal code changes. Modularity promotes maintainability and scalability [14], which is critical in research prototypes that evolve iteratively.
- **NF2 (Performance):** The pipeline should handle hundreds of rounds per batch efficiently through parallel processing. Performance is essential to ensure that large-scale datasets (e.g., 5,000+ rounds) can be processed in realistic timeframes.
- **NF3 (Robustness):** The system should handle incomplete or inconsistent data gracefully, providing logs and recovery strategies. Robust systems are critical in real-world data preprocessing, where raw inputs are often noisy or partially corrupted [3].
- **NF4 (Usability):** The labelling tool should be easy to learn and efficient to use by human annotators with minimal training.

Throughout the development of the preprocessing pipeline and labelling tool, I applied software engineering principles to ensure quality, scalability, and maintainability of the solution. The pipeline was designed to be modular, following the principle of separation of concerns [14]. Separating into modules such as:

(1) **Demo handling module**: Automates downloading, naming, and preparing demo files.
(2) **Preprocessing engine**: Parses demos, round-level data (such as player locations, actions, and economy) is extracted and converted into per-frame graphs with graph/node/edge features.
(3) **Labelling GUI**: Allows human annotators to view replay data frame by frame on the minimap and assign tactic labels on a round timeline.

(4) **Graph writer**: Embeds tactic labels into per-frame graph files and ensures consistency in format.
(5) **Debugging + CLI tools**: Support traceable logging, error recovery, visualisation, and Discord webhooks for real-time updates on long processes, as well as various utility scripts to extract and unpack the data for debugging purposes.

All of these were written and documented to be independently testable components. This modularity not only improved code clarity but also enabled easier debugging and future extension. This fulfils NF1 and simplifies debugging. For instance, changing from *de_dust_2* to *de_mirage* would require only map-specific config updates and radar image replacement, with the rest of the pipeline unchanged.

Error handling mechanisms were included to detect and log inconsistencies in demo data while maintaining the integrity of the overall processing batch. I developed a *strict* mode in the preprocessing engine, which stopped the process and logged the cause of the error in detail; any inconsistencies were caught, and partial data was patched. The labelling interface was iteratively improved using usability feedback from team members, with emphasis on clarity, speed of annotation, and inter-annotator consistency.

SRQ 2 (Preprocessing): We developed a pipeline to extract spatial-temporal data from raw demos and convert them into graph-based structures, meeting both data science and software engineering quality standards.

In summary, the applied software engineering decisions reflect best practices grounded in modularity, fault tolerance, and human-cen-tred interface design, ensuring the tools developed are usable, reliable, and maintainable in both research and applied contexts.

Presentation & Evaluation

The labelling and preprocessing components were integrated into the larger team project, which includes training a GNN model for tactic recognition. The quality of learning outcomes and the preprocessing pipeline's compatibility with the GNN input requirements were used to evaluate its performance. Inter-annotator [2] consistency checks were used to assess labelling accuracy.

SRQ 3 (Labelling quality): We created a prototype labelling interface and defined labelling criteria to ensure labelling consistency. Labelling was done by 3 people on our team, ensuring the labels' correctness, and any confusion was peer-reviewed by each other. Other parts of the demo that were not clear were automatically marked as "*tactic uncertain*", which let us ignore these frames during GNN training.

Communication

All developed components — including the preprocessing pipeline, the annotation GUI, and the labelled dataset — were integrated into a single system designed for use in both research and practical applications. These were documented and shared via a publicly accessible GitHub repository [11], and demonstrated through a formal project presentation. Detailed results and performance evaluations of these components are discussed in Section 5.

5 RESULTS

This section presents the outcomes of the system described in the Methodology section. Provided here are performance results and usage insights for both the preprocessing pipeline and the annotation interface, as well as details of the final tactical dataset.

The result of this thesis is a fully implemented preprocessing pipeline and a functional labelling interface. We created them in order to enable users to quickly preprocess and label multiple rounds of a game to then later use it as an input for supervised learning of CS:GO team tactics through our fine-tuned GNN model. The preprocessing pipeline successfully parsed and transformed raw CS:GO demo files into structured graph representations, extracting spatial-temporal features such as player positions, in-game events, and economy data. These features were extracted on a frame accuracy level and formatted into graph structures suitable for Graph Neural Network (GNN) input. During this time we developed a labelling interface and iteratively refined it to support the annotation of tactical behaviour within selected rounds. This allowed the creation of a labelled dataset containing representative tactical patterns, forming the foundation for training and evaluating machine learning models.

5.1 Preprocessing Pipeline Outcomes

During our research we focused on one map specifically—*de_dust_2*. We chose to do our research only on this map, mainly because of these reasons: simple map design, a long history of the map being used in professional and casual settings, and simple, well-defined tactics. I analysed our progress and extracted key statistics that are important for our research. It includes information about games, rounds, frames, and graphs. See table 2.

Number of demo files processed	195
Total rounds extracted	5133
Frames skipped due to issues	0 per game
Average number of frames per round	≈ 186
Processing time per frame	≈ 1 to 4 seconds
# games that could be proccessed parallely	64
Number of node features extracted	29 per graph

Table 2. Preprocessing statistics

Each graph that is stored in a .pkl file represents a single frame of a CS:GO round and is composed of three main components: `graph_data`, `nodes_data`, and `edges_data`. The `nodes_data` structure maps each entity in the game—such as players, the bomb, or bombsites—to a 29-dimensional feature dictionary. All of the features extracted can be found in Appendix C. These features include spatial coordinates, velocity vectors, viewing direction, player status flags (e.g., *isAlive*, *isPlanting*), economy-related values (e.g., cash, equipment value), and contextual indicators like whether the entity is inside the bombsite or buy zone. The `edges_data` defines pairwise relationships between entities based on proximity and team membership, allowing downstream Graph Neural Network models to reason about interactions between teammates and opponents. Graph files that are stored as *.pkl* files can be easily converted into standard formats compatible with PyTorch Geometric for further

training and evaluation. For example, I developed a debugging tool that can extract this *.pkl* file into a *.json* file, which has a set of key-value pairs enabling us to see what went wrong in the graph creation process, adding another layer of security.

The preprocessing pipeline proved to be working efficiently and is scalable since we developed a multithreaded solution allowing users to process up to 64 files simultaneously. During preprocessing, we encountered occasional issues where positional distances between nodes could not be computed, typically due to missing player data or desynchronised game state frames. To resolve this, we implemented a *strict mode* which found all the issues and let us patch it, approximating the missing values by interpolating between the closest valid frames before and after the gap. This ensured that all graph instances maintained structural consistency and prevented downstream errors during graph construction or model training. These fixes improved the overall robustness of the dataset without compromising its integrity.

By focusing on a single map, *de_dust_2*, we were able to maintain consistency in spatial layout while extracting over 5,000 high-resolution rounds with rich tactical information. The resulting graph structures capture both individual player states and inter-player relationships in a format that is well-suited for later machine learning models.

5.2 Labeling Interface and Dataset

To support the creation of a high-quality labelled dataset, we developed a custom labelling graphical user interface (GUI). This interface enables users to efficiently review CS:GO demo replays and annotate the tactics used by the Terrorist team on a per-frame basis, as illustrated in figure 4 found in the appendix. The GUI includes a number of specialised features: a *round timeline* at the bottom of the screen for frame selection, a *players' inventory panel* next to the minimap, an array of *tactic labelling buttons* on the far right, a *predict* button for model-assisted labelling, and a dynamic *minimap* that shows current player positions, death markers, utility usage, and bomb location.

Labelling a sequence is simple: a user can drag across the timeline using the middle mouse button to select a sequence of frames, observe the tactical movement on the minimap, and click a tactic button to label the frames. The labelling backend was implemented into the GUI. Once a user selects and labels a sequence of frames in the GUI, this information is saved to a structured *.json* file. Each file is named in this format (`game_uuid_round_id.json`) and contains a list of objects containing:

- `frame`: the labeled frame,
- `tactic_id`: a reference to the predefined tactic taxonomy (see Table 4),

These files can later be reloaded or fed into our preprocessing pipeline (`create_graphs.py`) and embedded into each graph as part of the `graph_data` structure (e.g., under the `strategy_used` key). This makes the label data directly available during GNN model training, aligning frame-accurate tactical annotations with spatial-temporal graph structures. A list of relevant scripts is provided in Appendix E.

Number of games labeled	20
Number of frames labeled	28,468
Number of *uncertain tactic* frames	18,705
Number of unique tactics annotated	15
Most common tactic	*t_control_mid*

Table 3. Labeling statistics

The "**tactic uncertain**" label was automatically filled in when annotators were unsure about the tactical intent or when mid-round transitions made tactical classification ambiguous and nothing was marked on the timeline. This label plays a critical role in maintaining labelling quality and ensuring that uncertain data can be reviewed later or excluded from training. As shown in figure 2, certain tactics such as *t_control_mid* and *t_execute_a_short* appear more frequently. This distribution reflects the natural bias in professional gameplay on *de_dust_2*, where mid control and short executions are fundamental components of most round strategies. By capturing both common and rare tactics, our dataset enables models to generalise across a wide range of in-game situations and opens the door for more nuanced tactical analysis.

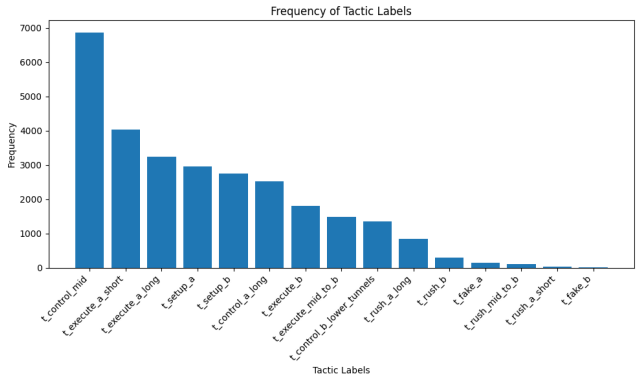

Fig. 2. Tactic frequency distribution across labeled rounds

5.3 Tactical Taxonomy and Labeling Process

To label CS:GO tactics consistently across rounds, we developed a map-specific tactical taxonomy tailored to *de_dust_2*. This taxonomy was derived through a hybrid approach. Initially, we deductively researched possible tactics by reviewing professional match footage, tactical guides from the community such as Zestrat [29], and direct feedback from experienced players. The goal was to identify commonly recurring tactical structures that could be reliably recognised in-game and meaningfully distinguished by human annotators. Later this set of tactics was refined inductively by annotating demos, during which we observed and adjusted the label set due to the patterns that emerged consistently.

We defined a tactic as a sequence of coordinated team behaviour with a clear strategic goal and execution path, typically involving

synchronised movement, utility usage, and site pressure. The final taxonomy includes 15 tactics, such as:

- **A Long Rush** — All players push through Long Doors with flash and smoke support to quickly take A Site.
- **Mid to B Execute** — Mid players smoke CT and join Tunnel players to collapse onto B Site.
- **A setup** — Players spread across the map to gain early information, slowly moving towards the A bombsite with the bomb.
- **A short execute** — Players group up mostly at the end of the "*catwalk*" area and the start of the "*A short*" area and use utility such as flashes, smokes, and molotovs to flush out the remaining CT players out of positions before taking control of the A bombsite.

Each label is applied based on the observable coordination of players on the minimap and their positional evolution over time. Figure 3 shows an example of how the "*A short execute*" tactic corresponds to specific player positions across "*A short*".

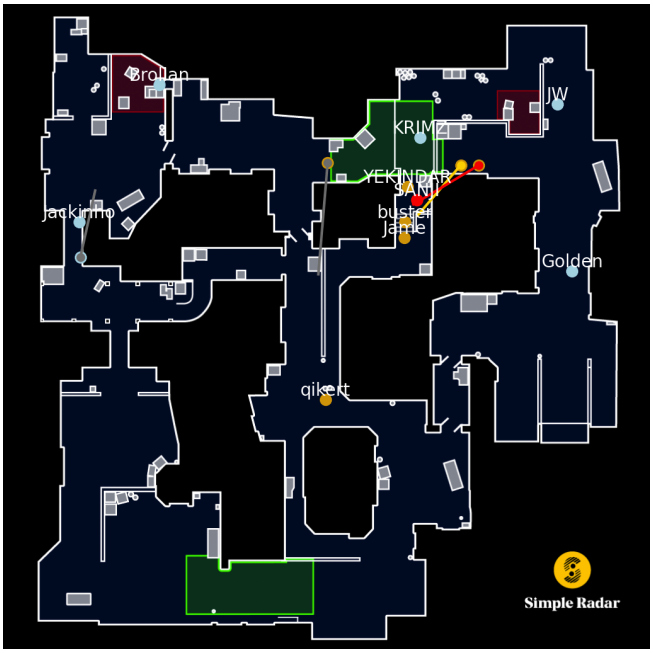

Fig. 3. Example of player positioning during a "A short execute" tactic: 4 players are advancing on short; 1 is catching up from the middle.

The labelling process itself required careful conceptual thinking and iterative refinement. Initially, the boundaries between some tactics (e.g., A setup vs. mid control) were difficult to distinguish. We addressed this by introducing a round timer and encouraging annotators to label tactics that were clearly observable and could not be mistaken during their active execution window, not their preparation phase. This allowed us to increase our labelling accuracy, as now a "*rush*" tactic is only labelled when it happens immediately after the round starts. Additionally, we allowed annotators to leave frames unlabelled if the tactic was uncertain. These would be automatically filled with the "*tactic uncertain*" label during graph creation to maintain continuity. Some of the main struggles included:

- Differentiating similar mid-control setups without relying on team comms, specific equipment, and counter-tactics.
- Determining when a tactic starts and ends based purely on positional data.
- Avoiding annotation bias and uncertainty.

Despite these challenges, the final taxonomy proved both expressive and practical for supervised learning. The resulting label set balances tactical granularity with annotator feasibility and forms the backbone of our dataset's semantic structure.

The final set of 15 tactics — which includes rushes, executions, setups, control plays, and fakes — reflects both the structured understanding from prior literature and the real-world variability observed in professional matches. This taxonomy is included in Appendix D.

6 DISCUSSION

6.1 Comparison to Existing Work

Previous eSports research has focused on outcome prediction, player ranking, or anomaly detection. For example, Bednarek et al. [4, 5] developed data pipelines that processed data from demo files to extract performance statistics such as kills, economy, etc. This work was focused on individual player performance; however, it lacks any kind of high-level tactical modelling.

Studies such as Birant & Birant [7] and Makarov et al. [12] applied ML to predict match outcomes, but in their research, they treated rounds as isolated events without incorporating context such as strategy. Björklund et al. [8] did similar research, clustering player behaviour and using neural networks to predict win outcomes, but the strategy context was not used in this research.

The Esports Trajectories and Actions (ESTA) dataset [28] added a significant contribution by offering spatial and temporal player tracking data from professional matches. However, as noted in Section 2.4, this dataset also lacks tactical labels and does not support supervised learning approaches for understanding strategic behaviour.

While some traditional sports research (e.g., Anzer et al. [1] and Raabe et al. [16]) has begun to apply graph-based models to detect coordinated team actions, such approaches had not yet been transferred to the CS:GO context. Our work extends this line of research into eSports by constructing graph-structured representations of rounds with explicit tactical annotations.

In contrast to prior literature, our method does not focus merely on player statistics or outcome prediction but introduces a conceptual layer of tactical abstraction. This is made possible through a map-specific tactical taxonomy (Section 5.3), a frame-level annotation interface (Section 5.2), and a preprocessing pipeline that outputs GNN-compatible graph data (Section 5.1). These innovations enable the direct modelling of strategic intent — an aspect of gameplay that has been absent from most CS:GO analytics research to date.

6.2 Key Contributions and Implications

Following the Design Science Research Methodology (DSRM) [15], we iteratively developed and evaluated a tactical annotation pipeline tailored to CS:GO. This resulted in several important contributions with meaningful practical implications for eSports analytics and machine learning applications:

Structured preprocessing pipeline: We created a reproducible preprocessing pipeline that converts raw CS:GO demo files into structured graph representations for GNN input. This process includes demo downloading, parsing the files, patching incomplete data, extracting the graph features, and lastly, creating the graphs suitable for our GNN training. As a result, researchers and developers can now automate large-scale conversion of unstructured eSports data into machine-readable formats, reducing manual preprocessing effort in future work.

Interactive labelling interface: We created a custom labelling graphical user interface (GUI) that streamlines the tactical annotation process. The tool supports frame-accurate labelling with intuitive controls, shortcut keybindings, and real-time visual feedback both on the minimap and in the labeling timeline improving annotator efficiency and accuracy throughout the process of labelling many rounds.

Map-specific tactical taxonomy: To enable the user to annotate the tactics in a specific round, we developed a tactical taxonomy tailored to the map *de_dust_2*. This taxonomy was formed by iteratively reviewing match footage, researching online [29], and talking to field professionals. It includes 15 distinct tactics, referencing the execution style and map region. We also introduced an *"uncertain"* label to accommodate ambiguous or transitional phases. It is applied automatically when the frame is not labelled when creating graphs. If uncertain tactics were labelled, it would create confusion in our GNN model training; therefore, it helps preserve dataset reliability without forcing artificial precision. This taxonomy played a critical role in enabling accurate and frame-aligned labelling as well as enabling precise GNN training; it also provides a reusable structure for future research and sets an example for similar taxonomies on other maps and games.

Annotated dataset of tactical behaviour: Using the tools and taxonomy, we created a labelled dataset of over 28,000 frames, capturing 15 distinct tactics observed in professional matches on *de_dust_2*. This dataset is one of the first tactically labelled, structured datasets that has important graph features and captures the team behaviour. Its availability enables supervised learning on tactical behaviour and allows future studies to explore how specific tactics relate to round outcomes, player roles, or counter-strategies.

GNN integration and validation: We integrated our GNN model architecture into the final product, validating the structural compatibility of the dataset for tactical learning [17]. This confirms that our dataset has the potential for machine learning. Our dataset supports further development of interpretable models that can classify in-game tactics in near real time — with potential applications in coaching tools, match analysis, and live broadcast enhancement.

These outcomes mark a shift from black-box outcome prediction to explainable and interpretable team behaviour modelling. By making tactics computationally tractable, this work creates a foundation for new research directions, such as win prediction models that account for tactical execution, automated strategy mining, or role-based performance evaluation. It also opens practical pathways for real-time coaching support, tactical scouting, and deeper competitive insight across the eSports industry.

6.3 Usability and Engineering Observations

From a software engineering perspective, the development process prioritised usability, reusability, and reliability. Feedback from testers indicated that the labelling interface was intuitive and improved significantly with each iteration. After implementing timeline zoom, reduced clutter, keyboard shortcuts (e.g., number keys for quick tactic selection), and enhanced minimap overlays, the average annotation time per round decreased from approximately 3-4 minutes to 1-2 minutes (measured across 10 test rounds by 3 different annotators). Users reported greater confidence and lower cognitive load when labelling due to the improved visual clarity and interface responsiveness. Furthermore, the entire pipeline was designed with modularity in mind, allowing for easy adaptation to other maps with minimal refactoring.

6.4 Limitations and Future Work

Despite its impact, this work has limitations that open many possibilities for future research and improvements.

First, the dataset is limited to a single map (*de_dust_2*) and a sample of 20 matches, meaning tactic diversity may not fully capture the variability across different competitive settings. Expanding the dataset to other maps, such as *de_mirage* and *de_inferno* with varying layouts, chokepoints, tactics, and player heatmaps, will inevitably strengthen the dataset's integrity and allow for further development on the GNN.

Second, although the annotation GUI supports precise, frame-level tactic labelling, it still relies on manual input, which makes the process of labelling "*large-scale*" time-intensive and infeasible without significant human effort and introduces subjective bias. Future work could explore possibilities of a semi-automated labelling workflow, where a trained GNN model offers the labels prematurely and human annotators have to verify and correct those labels, which could increase the speed and accuracy of the labelling process.

Third, the current tactic taxonomy, while effective, represents a relatively coarse categorisation of tactical behaviour. It does not capture nuanced sub-tactics or player-specific roles (e.g., lurker vs. entry fragger), nor does it differentiate between successful and unsuccessful executions of the same tactic. A more granular taxonomy or a hierarchical model of tactics could enable deeper insights into team coordination strategies.

Lastly, evaluation of labelling consistency was conducted via peer review and inter-annotator agreement checks, but not through large-scale quantitative metrics. Future work could formalise annotation reliability using established measures like Cohen's Kappa or Krippendorff's Alpha to more rigorously assess consistency.

Together, these directions highlight the potential for this work to serve as a foundation for more advanced tactical analysis systems in eSports, blending human expertise with machine learning at scale.

REFERENCES

[1] Gabriel Anzer, Pascal Bauer, Ulf Brefeld, and Dennis Fassmeyer. 2022. Detection of Tactical Patterns Using Semi-Supervised Graph Neural Networks. https://www.sloansportsconference.com/research-papers/detection-of-tactical-patterns-using-semi-supervised-graph-neural-networks

[2] Ron Artstein. 2017. Inter-Annotator Agreement. In *Handbook of Linguistic Annotation*. Springer, Dordrecht, 297–313. https://doi.org/10.1007/978-94-024-0881-2_11

[3] Len Bass, Paul Clements, and Rick Kazman. 2012. *Software Architecture in Practice: Software Architect Practice_c3*. Addison-Wesley.

[4] David Bednárek, Martin Kruliš, Jakub Yaghob, and Filip Zavoral. 2018. Player Performance Evaluation in Team-Based First-Person Shooter eSport. In *Data Management Technologies and Applications*. Springer, Cham, 154–175. https://doi.org/10.1007/978-3-319-94809-6_8

[5] David Bednárek, Martin Krulis, Jakub Yaghob, and Filip Zavoral. 2017. Data Preprocessing of eSport Game Records - Counter-Strike: Global Offensive:. In *Proceedings of the 6th International Conference on Data Science, Technology and Applications*. SCITEPRESS - Science and Technology Publications, Madrid, Spain, 269–276. https://doi.org/10.5220/0006475002690276

[6] Jason Bell. 2022. What Is Machine Learning? In *Machine Learning and the City*. John Wiley & Sons, Ltd, Chapter 9, 207–216. https://doi.org/10.1002/9781119815075.ch18

[7] Kokten Ulas Birant and Derya Birant. 2023. Multi-Objective Multi-Instance Learning: A New Approach to Machine Learning for eSports. *Entropy* 25, 1 (Jan. 2023), 28. https://doi.org/10.3390/e25010028

[8] Arvid Björklund, William Johansson Visuri, Fredrik Lindevall, and Philip Svensson. 2018. Predicting the Outcome of CS:GO Games Using Machine Learning. (2018). https://hdl.handle.net/20.500.12380/256129

[9] Jessica Clement. 2025. Esports - Worldwide | Statista Market Forecast. https://www.statista.com/topics/3121/esports-market/

[10] Arizton Advisory & Intelligence. 2025. eSports Market Size & Share, Growth & Trends, Revenue Forecast Report. https://www.arizton.com/market-reports/esports-market-report

[11] Povilas Kirna. 2025. PovilasKirna/MOD12-Research-Project. https://github.com/PovilasKirna/MOD12-Research-Project

[12] 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

[13] Olifaye. 2025. HLTV. *Wikipedia* (March 2025). https://en.wikipedia.org/w/index.php?title=HLTV&oldid=1279833604

[14] D. L. Parnas. 1972. On the Criteria to Be Used in Decomposing Systems into Modules. *Commun. ACM* 15, 12 (Dec. 1972), 1053–1058. https://doi.org/10.1145/361598.361623

[15] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2007///Winter2007/2008. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems* 24, 3 (2007///Winter2007/2008), 45–77. https://doi.org/10.2753/MIS0742-1222240302

[16] Dominik Raabe, Reinhard Nabben, and Daniel Memmert. 2023. Graph Representations for the Analysis of Multi-Agent Spatiotemporal Sports Data. *Applied Intelligence* 53, 4 (Feb. 2023), 3783–3803. https://doi.org/10.1007/s10489-022-03631-z

[17] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. 2021. A Gentle Introduction to Graph Neural Networks. *Distill* 6, 9 (Sept. 2021), e33. https://doi.org/10.23915/distill.00033

[18] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (Jan. 2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[19] Ivan Šimić. 2024. Top 10 Most-Viewed Esports Events of 2024. https://esportsinsider.com/2024/12/most-viewed-esports-events-2024

[20] Sportsvenue. 2025. Top 10 Most Popular Esports Games in the World. https://www.sportsvenue-technology.com/articles/top-10-most-popular-esports-games-in-the-world

[21] Sahar Tahvili and Leo Hatvani. 2022. Transformation, Vectorization, and Optimization. In *Artificial Intelligence Methods for Optimization of the Software Testing Process*. Elsevier, 35–84. https://doi.org/10.1016/B978-0-32-391913-5.00014-2

[22] Di Tang, Raymond Kim-wai Sum, Minghui Li, Ruisi Ma, Peichi Chung, and Ray Wai-keung Ho. 2023. What Is Esports? A Systematic Scoping Review and Concept Analysis of Esports. *Heliyon* 9, 12 (Dec. 2023). https://doi.org/10.1016/j.heliyon.2023.e23248

[23] ESI Editorial Team. 2021. CS:GO Strategies, Analysed: A Look at How Bayes Esports Analyses Data Using AI. https://esportsinsider.com/2021/06/csgo-strategies-bayes-esports

[24] Wang Tian. 2018. Predictive Analysis on eSports Games: A Case Study on League of Legends (LoL) eSports Tournaments. https://cdr.lib.unc.edu/concern/masters_papers/8s45qd54c

[25] Fen Regis Trophies. 2023. How Do Esports Players Train? https://fenregistrophies.co.uk/how-do-esports-players-train/

[26] Bye woody. 2025. *Counter-Strike: Global Offensive*. *Wikipedia* (March 2025). https://en.wikipedia.org/w/index.php?title=Counter-Strike:_Global_Offensive&oldid=1280151159

[27] Peter Xenopoulos. 2025. Pnxenopoulos/Esta. https://github.com/pnxenopoulos/esta

[28] Peter Xenopoulos and Claudio Silva. 2022. ESTA: An Esports Trajectory and Action Dataset. https://doi.org/10.48550/arXiv.2209.09861 arXiv:2209.09861 [cs]

[29] Zestrat. 2025. Dust2 - All Tactics and Nades to Learn. https://www.zestrat.gg/csgo/maps/dust2.

A USE OF CHATGPT

During the preparation of this work, I used ChatGPT to help generate ideas and assist in structuring sections of the thesis. All content was thoroughly reviewed and edited by me to ensure correctness and alignment with academic standards.

B SYSTEM ARCHITECTURE OVERVIEW

The system was designed using a modular architecture to enable maintainability and traceability. The overall data flow consists of the following components:

(1) **Demo Ingestion:** Demos are downloaded using `download_demo_from_repo.py` and extracted with `extract_demos.py` into JSON format.
(2) **Preprocessing Engine:** The core script `create_graphs.py` processes raw JSON data into structured per-frame graphs, capturing player, event, and contextual features.
(3) **Tactic Labelling:** Using the GUI (`gui.py`), annotators label frame sequences with predefined tactics, which are saved as JSON files (see Table 4).
(4) **Label Injection:** These JSON label files are reloaded by the graph processor and merged with per-frame graph files by embedding tactic metadata (e.g., `strategy_used`).
(5) **GNN Training Preparation:** The output graph dataset is ready for direct use in Graph Neural Network training pipelines.

The implementation of these modules and supporting utilities is listed in Appendix E.

C GRAPH FEATURE EXTRACTION EXAMPLE

The following JSON structure shows an example output generated by `create_graphs.py`:


```
{
  "graph_data": {
    "tick": 74499,
    "seconds": 0.31496062992125984,
    "bombPlanted": false,
    "tFreezeTimeEndEqVal": 26350,
    "tRoundStartEqVal": 1000,
    "tRoundSpendMoney": 25350,
    "tBuyType": "Full Buy",
    "strategy_used": "t_control_mid"
  },
  "nodes_data": {
    "0": {
      "x": -329.98,
      "y": -713.30,
      "z": 89.50,
      "velocityX": 5.72,
      "velocityY": 199.92,
      "velocityZ": 292.62,
      "viewX": 91.97,
      "viewY": 20.39,
      "hp": 100,
      "armor": 100,
      "activeWeapon": 9,
      "totalUtility": 2,
      "isAlive": true,
      "isDefusing": false,
      "isPlanting": false,
      "isReloading": false,
      "isInBombZone": false,
      "isInBuyZone": true,
      "equipmentValue": 6650,
      "equipmentValueFreezetimeEnd": 6650,
      "equipmentValueRoundStart": 200,
      "cash": 450,
      "cashSpendThisRound": 4400,
      "cashSpendTotal": 13150,
      "hasHelmet": true,
      "hasDefuse": false,
      "hasBomb": false,
      "areaId": 8577,
      "nodeType": 1000
    }
    ...
  },
  "edges_data": [
    [0, 1, {"dist": 360.45}],
    [4, 7, {"dist": 4514.03}],
    [6, 0, {"dist": 165.70}],
    ...
  ]
}
```

D TACTICAL LABELS

Tactic Label	Description
t_setup_a	Slow map control with lean toward A site
t_setup_b	Passive default with eventual B site lean
t_control_a_long	Gaining map control through A Long area
t_control_mid	Controlling the mid-area for flexibility or split
t_control_b_lower_tunnels	Slow approach through lower tunnels for B control
t_execute_a_long	Structured push through A Long with utility
t_execute_a_short	Execution via short (catwalk) with nades
t_execute_mid_to_b	Mid-to-B split with CT smoke and tunnel join
t_execute_b	Full B site execute through tunnels
t_rush_a_long	Fast rush through A Long
t_rush_a_short	Aggressive rush through short (catwalk)
t_rush_mid_to_b	Fast-paced mid-to-B attack
t_rush_b	Direct rush into B site via upper tunnels
t_fake_a	Fake towards A to draw rotations
t_fake_b	Fake towards B to manipulate defenders

Table 4. Final Tactical Label Set for *de_dust_2*

E SCRIPT OVERVIEW

Script Location	Script Name	Language	Purpose
root	create_graphs_filenames.json	JSON	Stores names of files to convert to graphs.
root	dust2_demos_filenames.json	JSON	Stores names of Dust2 demos.
root	.env	-	Environment variables and runtime config.
research_project/src	create_graphs.py	Python	Preprocesses data and generates graphs.
research_project/src	gui.py	Python	Creates the labelling GUI.
research_project/src/utils	directory_files.py	Python	Lists files in a directory and outputs to JSON.
research_project/src/utils	discord_webhook.py	Python	Sends Discord messages for long processes.
research_project/src/utils	download_demo_from_repo.py	Python	Downloads demos from the ESTA repository.
research_project/src/utils	extract_demos.py	Python	Converts demos into readable JSON.
research_project/src/utils	filter_weapons.py	Python	Identifies unknown weapons and filters them.
research_project/src/utils	logging_config.py	Python	Sets up logging configuration.
research_project/src/utils	stats.py	Python	Script for extracting evaluation statistics.
research_project/src/utils	unpack_pkl.py	Python	Unpacks .pkl graph files into JSON format.

Table 5. Overview of Scripts Used in the Project

F LABELLING GUI

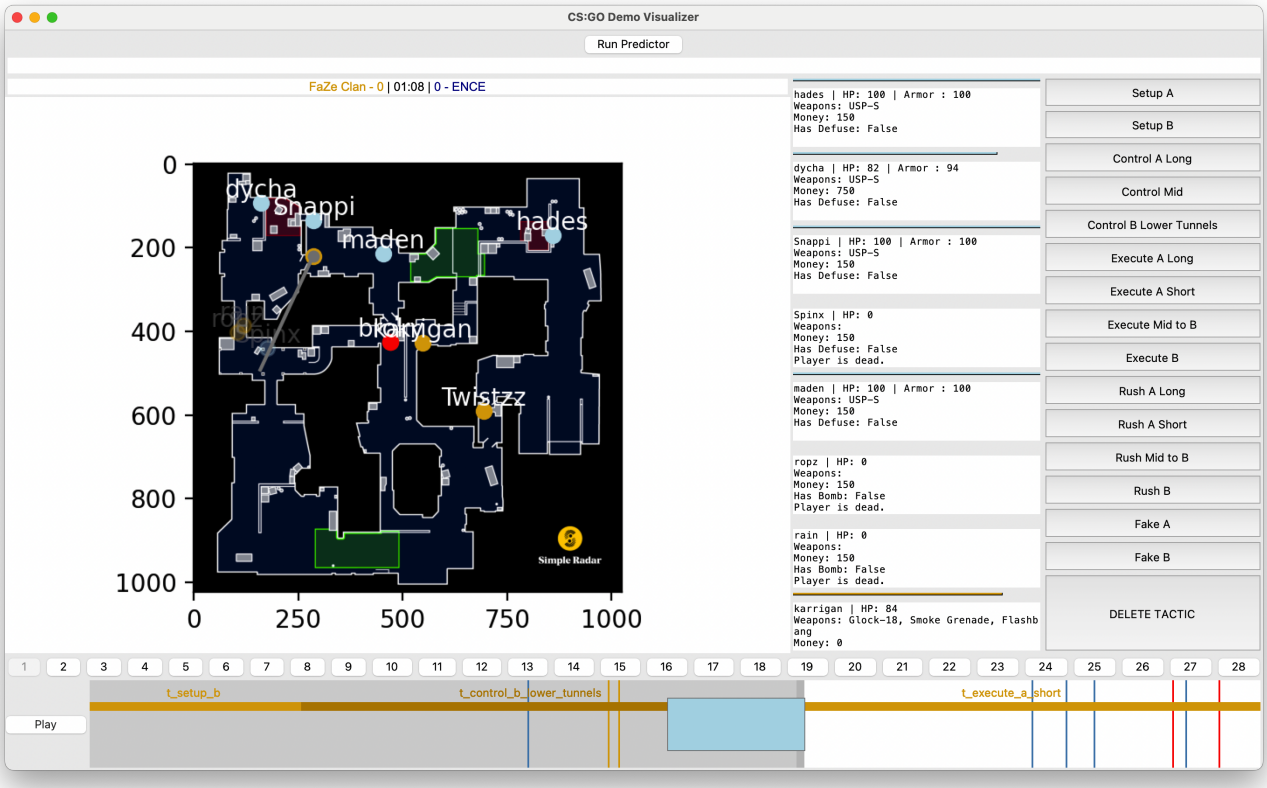

Fig. 4. Labelling GUI interface with a minimap, timeline, and tactical tagging controls.