XGuardian: Towards Generalized, Explainable and More Effective Server-side Anti-cheat in First-Person Shooter Games

Jiayi Zhang, Chenxin Sun, Chenxiong Qian

HKU



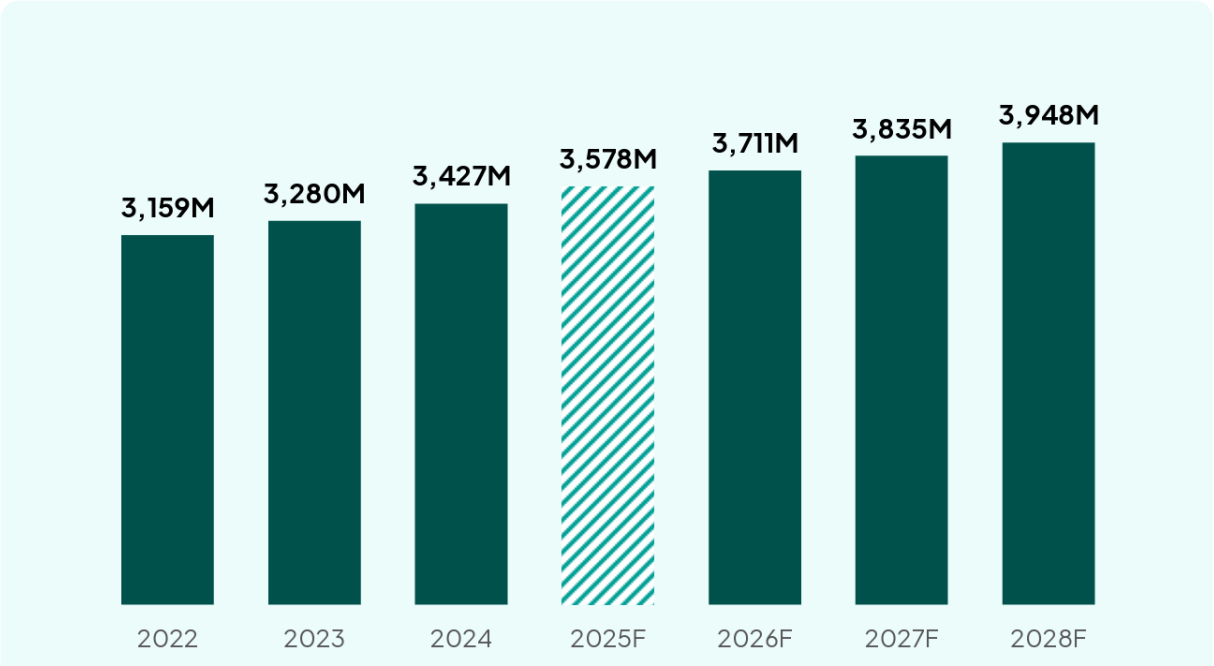
香 港 大 學

THE UNIVERSITY OF HONG KONG

Continued growth of the gaming industry

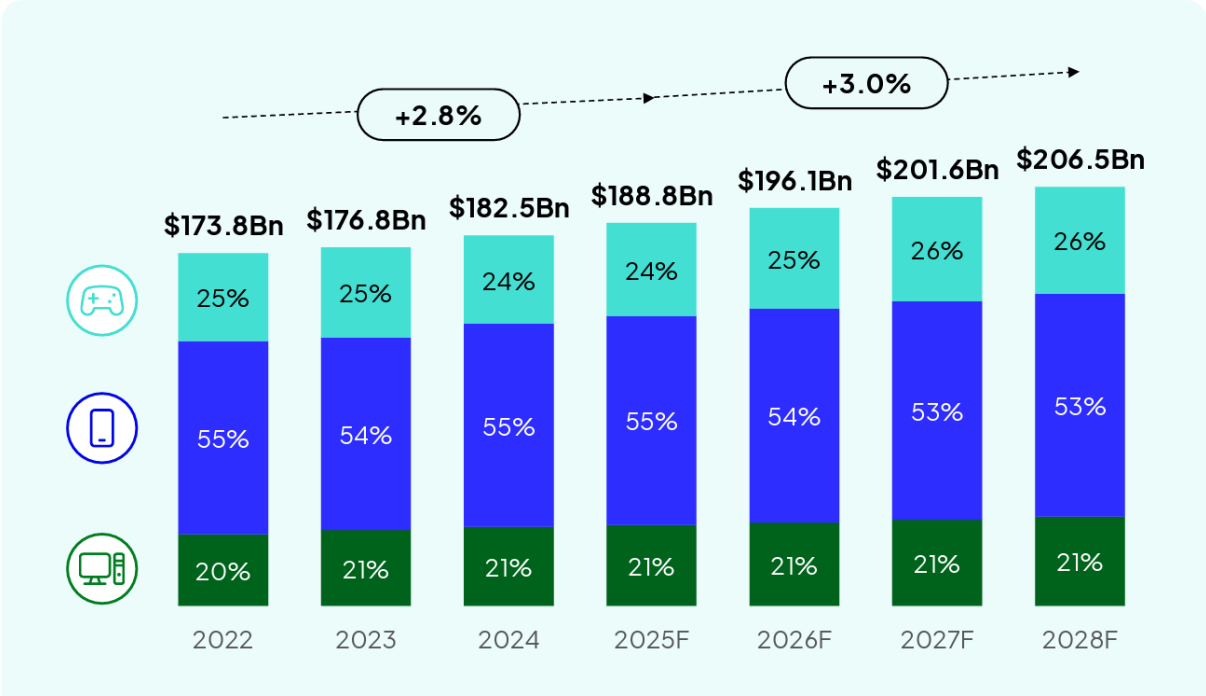
Global player forecast

2022 – 2028F



Global revenue by platform

2022 – 2028F



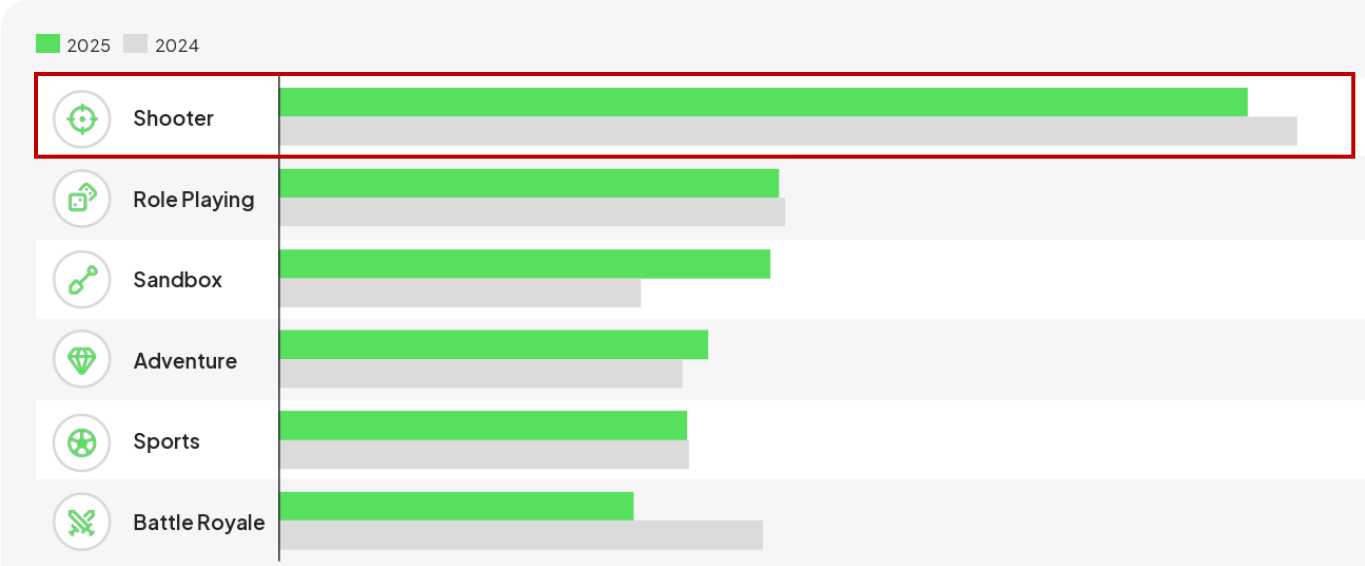
(F indicates forecast, actual statistics for 2025 are still being processed)

Newzoo. (2025). *Global games market report 2025*. https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2025.

FPS games dominate the market

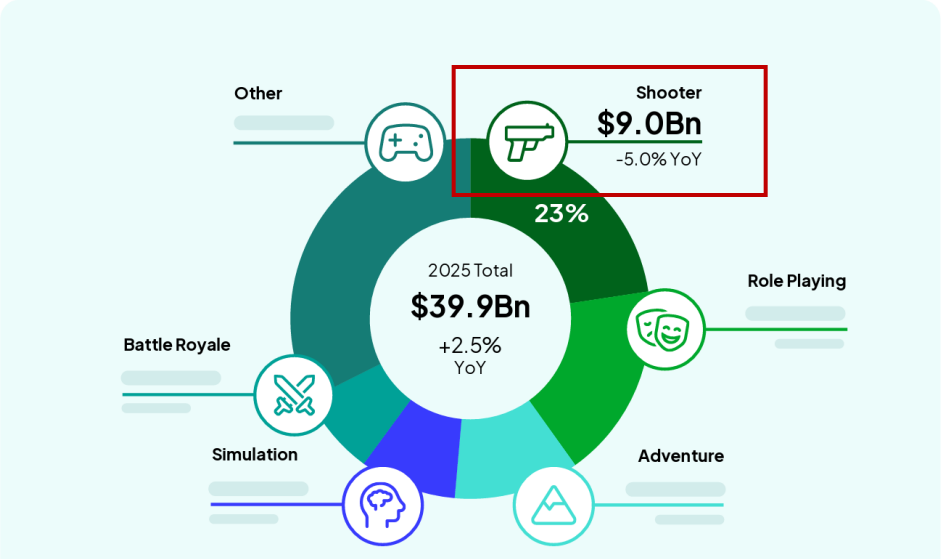
Top genres by playtime

PC, PlayStation, Xbox | 2025 vs. 2024



Global PC revenues by genre

2025F

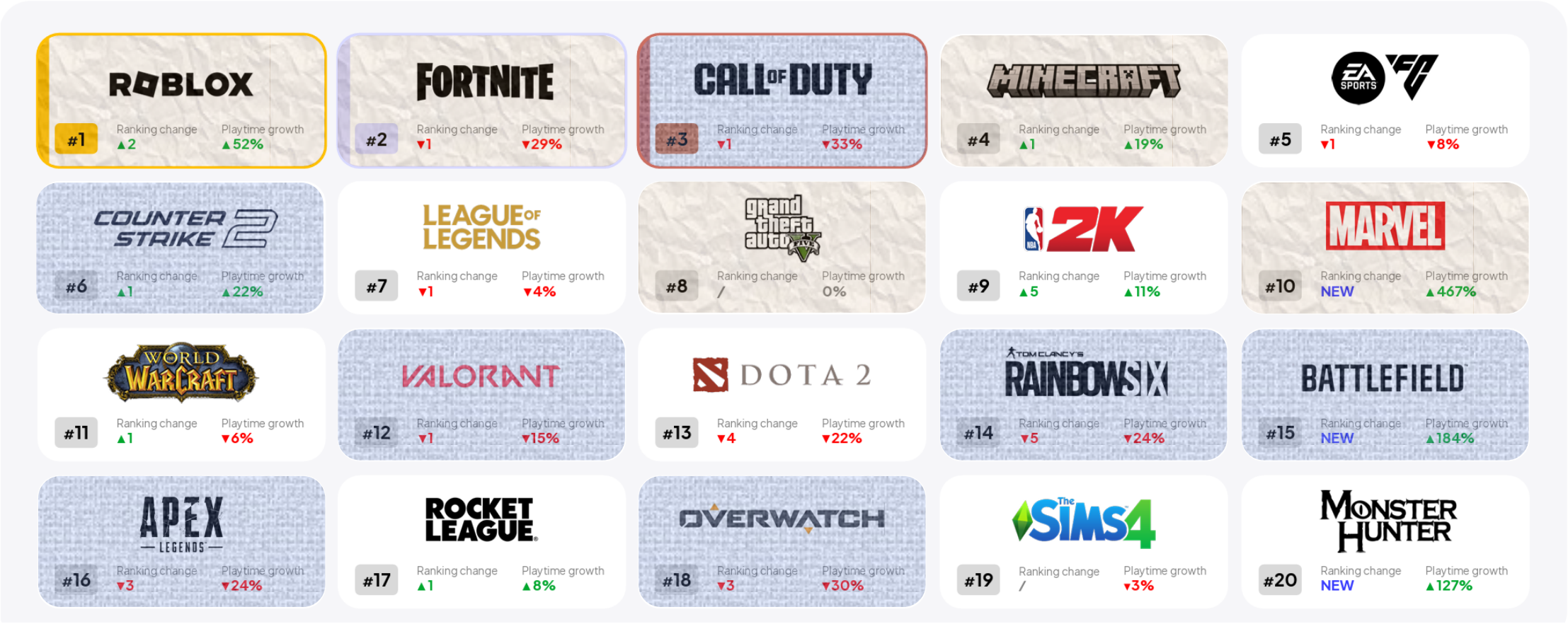


Newzoo. (2026). *The PC & console gaming report 2026*. https://newzoo.com/resources/trend-reports/the-pc-console-gaming-report-2026. (left)
Newzoo. (2025). *Global games market report 2025*. https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2025. (right)

FPS games remain a dominant genre

Top PC/console franchises by overall playtime

PC, PlayStation, Xbox | 2025



Newzoo. (2026). *The PC & console gaming report 2026*. https://newzoo.com/resources/trend-reports/the-pc-console-gaming-report-2026.

What is cheat and cheater

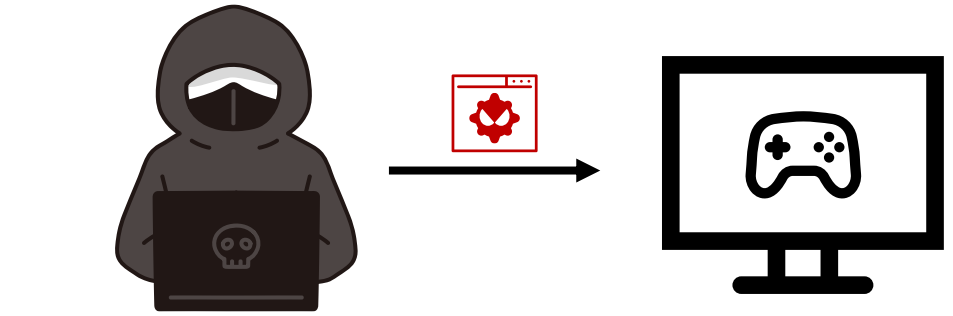
- Exploit measures to gain unfair advantages is called the **cheat**
  - usually through malicious software



Cheat

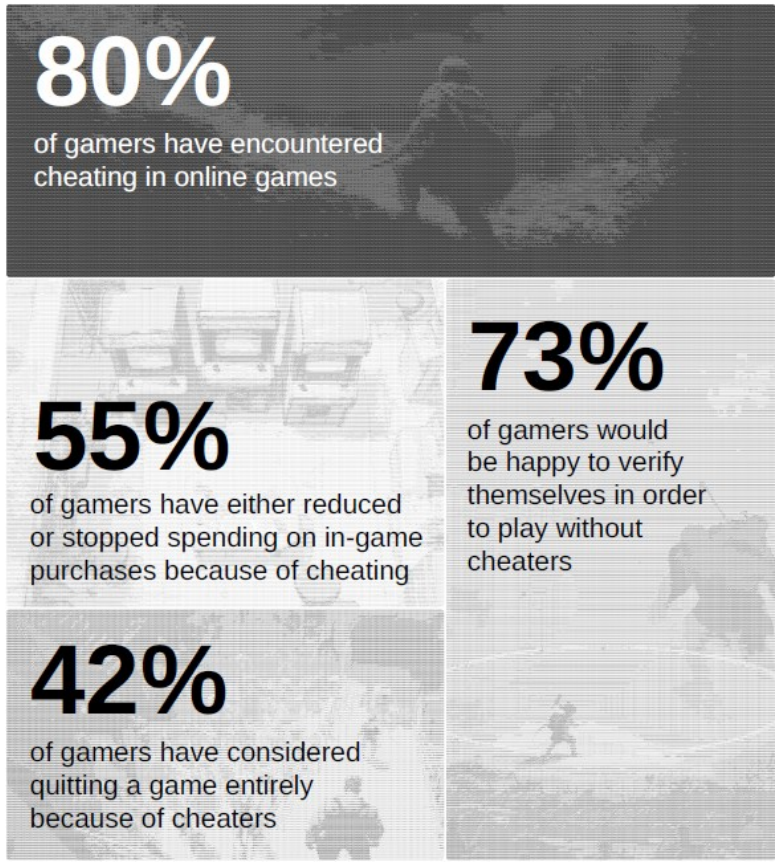
What is cheat and cheater

- Exploit measures to gain unfair advantages is called the **cheat**
  - usually use malicious software
- The person who uses the cheats is called the **cheater**



Cheater

Cheating industry is also “thriving”



PlaySafe ID. (2025). *Gaming's cheating crisis report*. https://playsafeid.com/gamings-cheating-crisis-report.

Cheating industry is also “thriving”



China
(2025)

Cheat Market Size [1]
$1.4B – $1.5B

NA & EU
(2025 Incomplete Statistics)

Annual Revenue Per Seller [2]
$12.8M – $73.2M

[1] Lee Woo Min. (2026, June). *Chinese Game Cheat Market Approaches 2 Trillion Won, Exploding with AI Integration*. https://www.gamemeca.com/en/view.php?gid=1776690.
[2] Sam Collins, Alex Poulopoulos, Marius Muench, and Tom Chothia. 2024. *Anti-Cheat: Attacks and the Effectiveness of Client-Side Defences*. In Proceedings of the 2024 Workshop on Research on offensive and defensive techniques in the context of Man At The End (MATE) attacks (CheckMATE '24). Association for Computing Machinery, New York, NY, USA, 30–43. https://doi.org/10.1145/3689934.3690816.

Cheating industry is also “thriving”

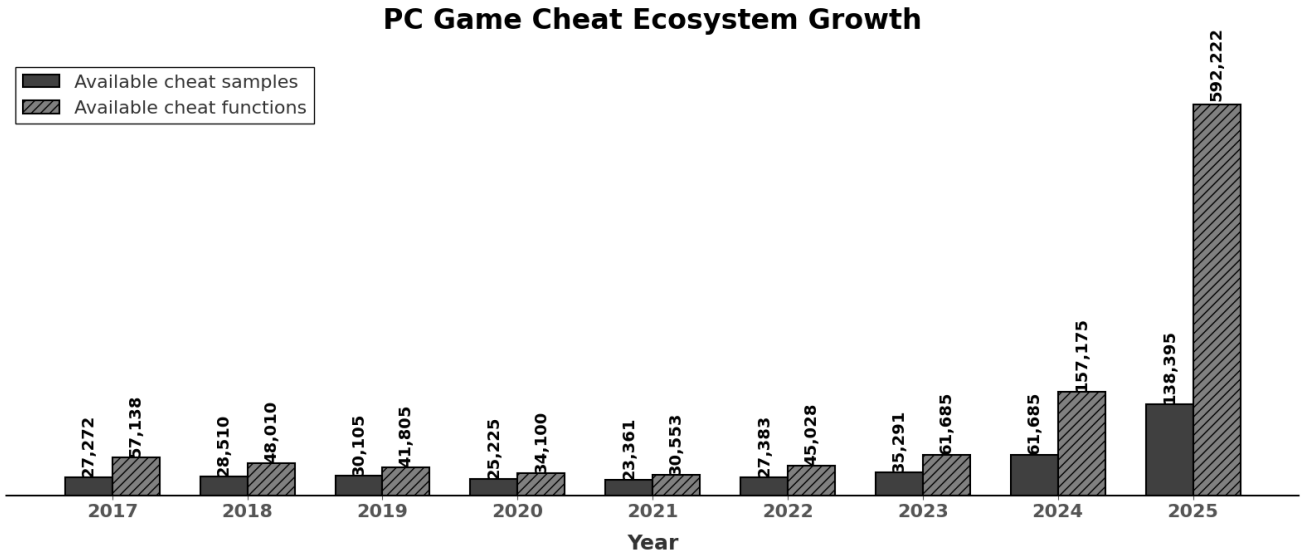


China
(2025)

NA & EU
(2025 Incomplete Statistics)

Cheat Market Size [1]
$1.4B – $1.5B

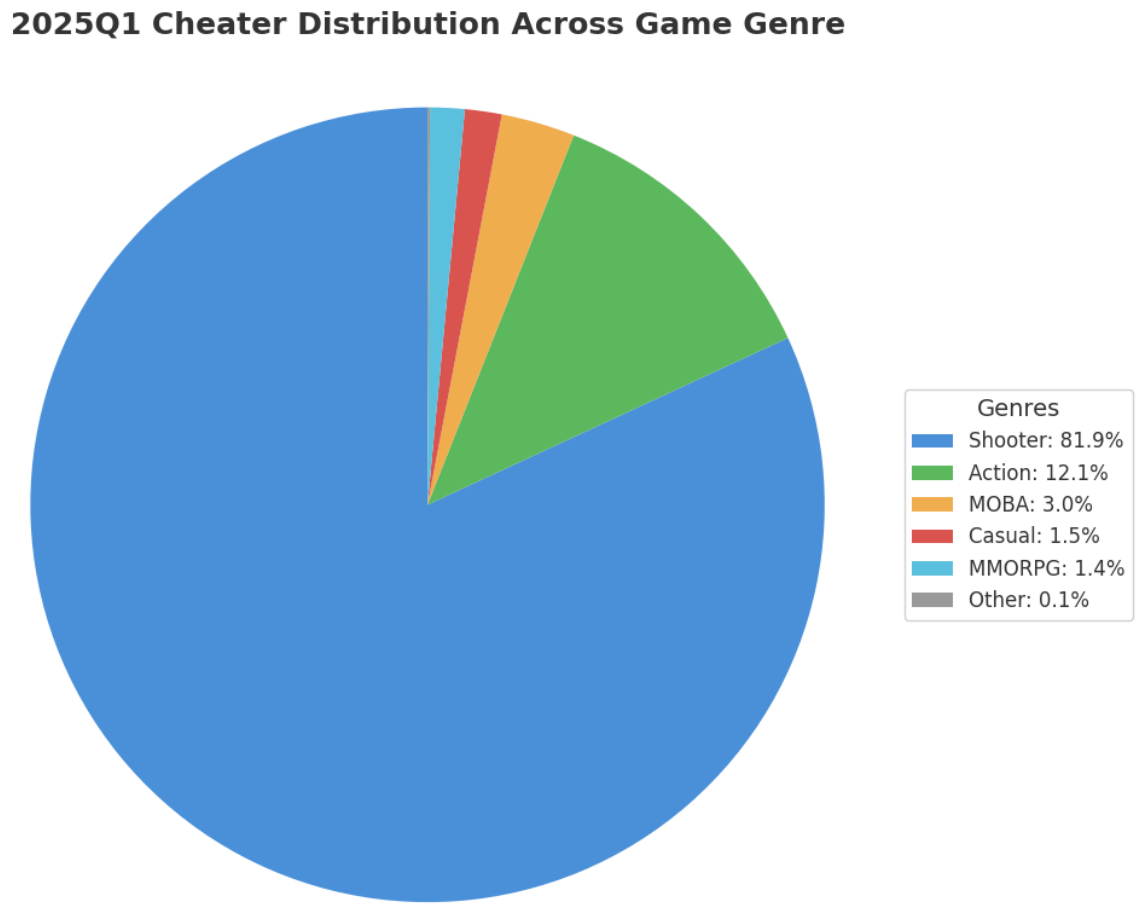
Annual Revenue Per Seller [2]
$12.8M – $73.2M



Sina.cn. (2026, June). *2025 Game security white paper released*. https://www.sina.cn/news/detail/5310825191440690.html.

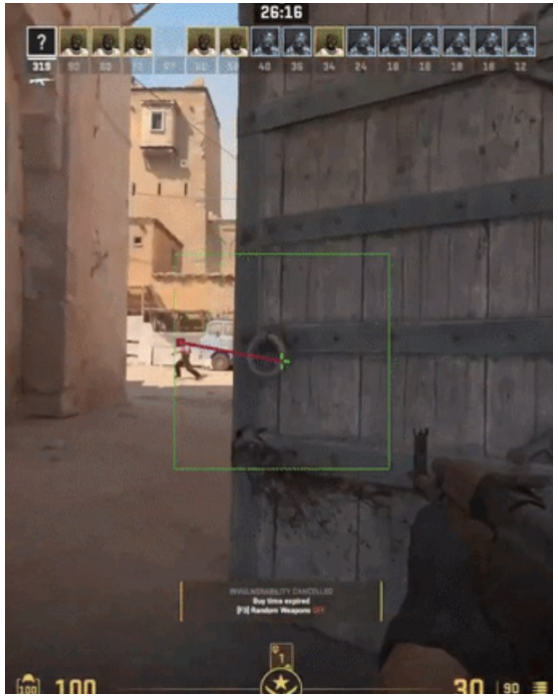
Cheating is prevalent and harmful in FPS games





Data is collected by ChatGPT from multiple sources on June 22, 2026. Picture is generated according to the data by ChatGPT. Data and picture is verified with the sources by the authors. (left) iResearch Inc. (2026). *White paper on insights into plug-in cheating groups in shooting games in 2026*. https://pdf.dfcfw.com/pdf/H3_AP202604081821073777_1.pdf. (right)

How do cheaters cheat in FPS games?





Auto-Aiming Techniques



Starting point



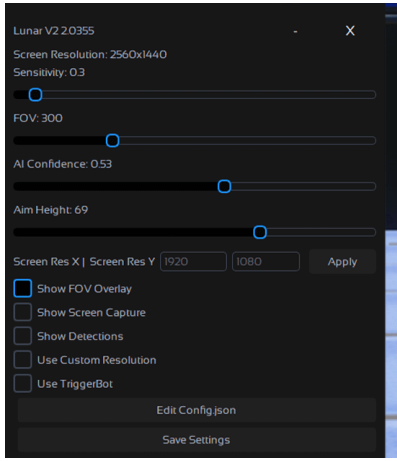
Find the players's health bar



Calculate health bar width. Find the red border around the player -> calculate body coordinates!

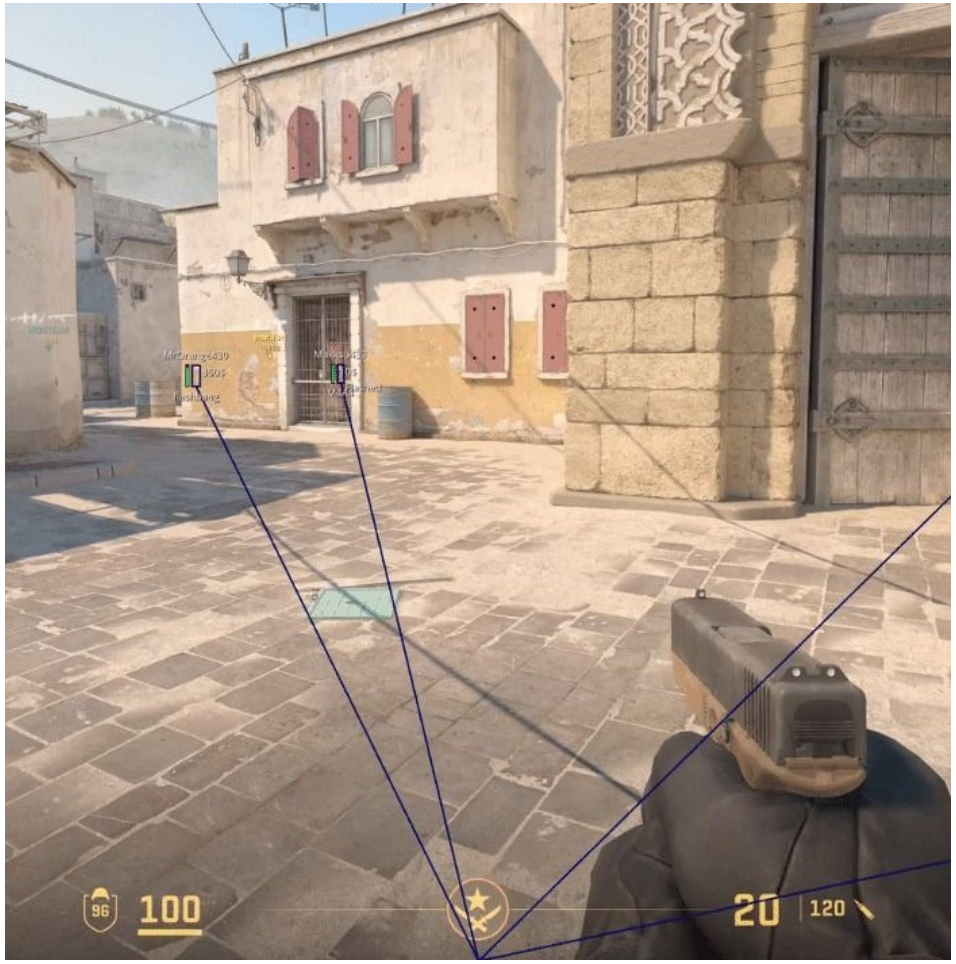


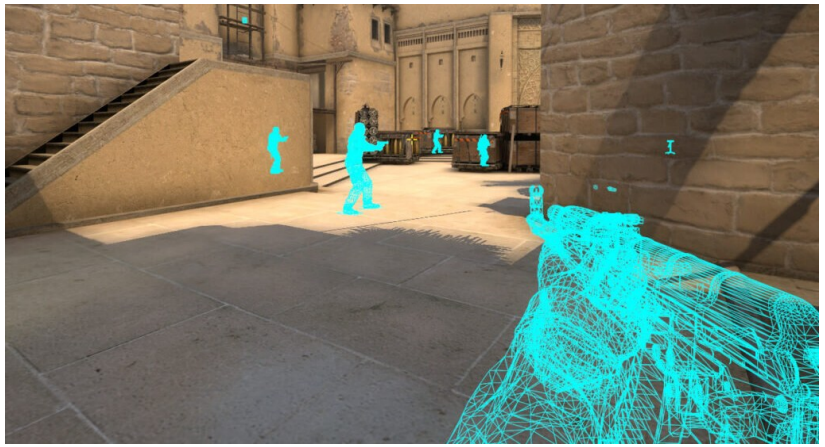
Calculate health bar width. Find the red border around the player -> calculate head coordinates!

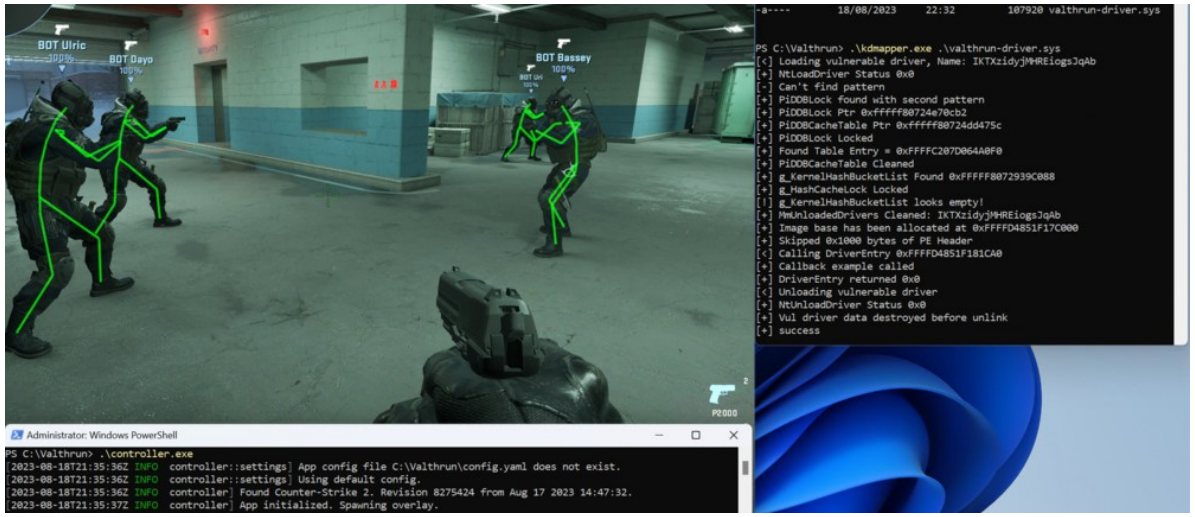




How do cheaters cheat in FPS games?







Extra-Sensory Perception (ESP) Techniques

How do cheaters cheat in FPS games?

Aim-assist cheats account for the overwhelming majority of FPS cheats

- ***TL;DR:*** today's cheats are countless, highly integrated and ever-changing
  - ❑Auto-Aiming techniques
    - *e.g., aimbot (memory, DMA, CV-based), triggerbot, aim trajectory smoothness adjustment…*
  - ❑ESP techniques
    - *e.g., ESP (skeleton, health, object…), wallhack (bounding box, x-ray, skin highlight…), radar…*
  - ❑Aim Compensation techniques
    - *e.g.,* recoil control (RCS), no recoil/spread, movement prediction…
  - ❑Activation techniques
    - *e.g.,* hold key, always on, fire only, toggle…
  - ❑…

How do cheaters cheat in FPS games?

Aim-assist cheats account for the overwhelming majority of FPS cheats

- Highly integrated with hundreds functions including
  - ❑Auto-Aiming techniques
    - *e.g., aimbot (memory, DMA, CV-based), triggerbot, aim trajectory smoothness adjustment…*
  - ❑Extra-S
    - *e.g.,* …*), radar…*
  - ❑Aim Co
    - *e.g.,*
  - ❑Activation techniques
    - *e.g.,* hold key, always on, fire only, toggle…
  - ❑…

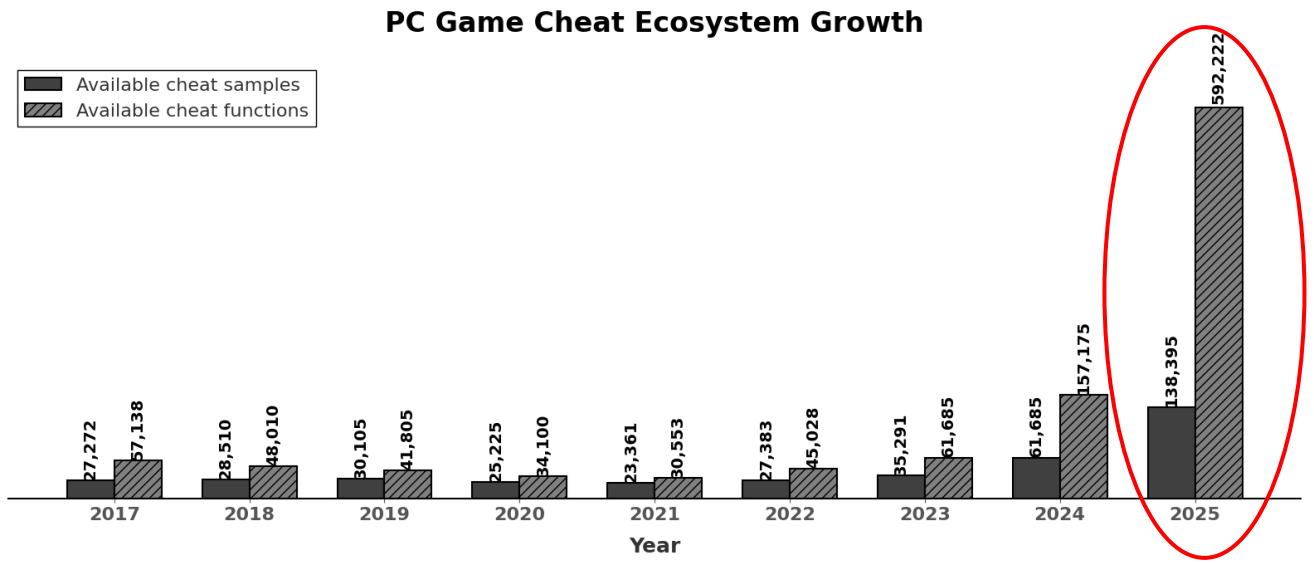
Different functions combinations
→ Countless possibilities!

Today’s real-world FPS anti-cheat up against

EVERYTHING is black-box:

- cheating techniques
- configurations
- functions
- level of cheating sophistication
- cheat developers/vendors

Are all ***DIFFERENT***.

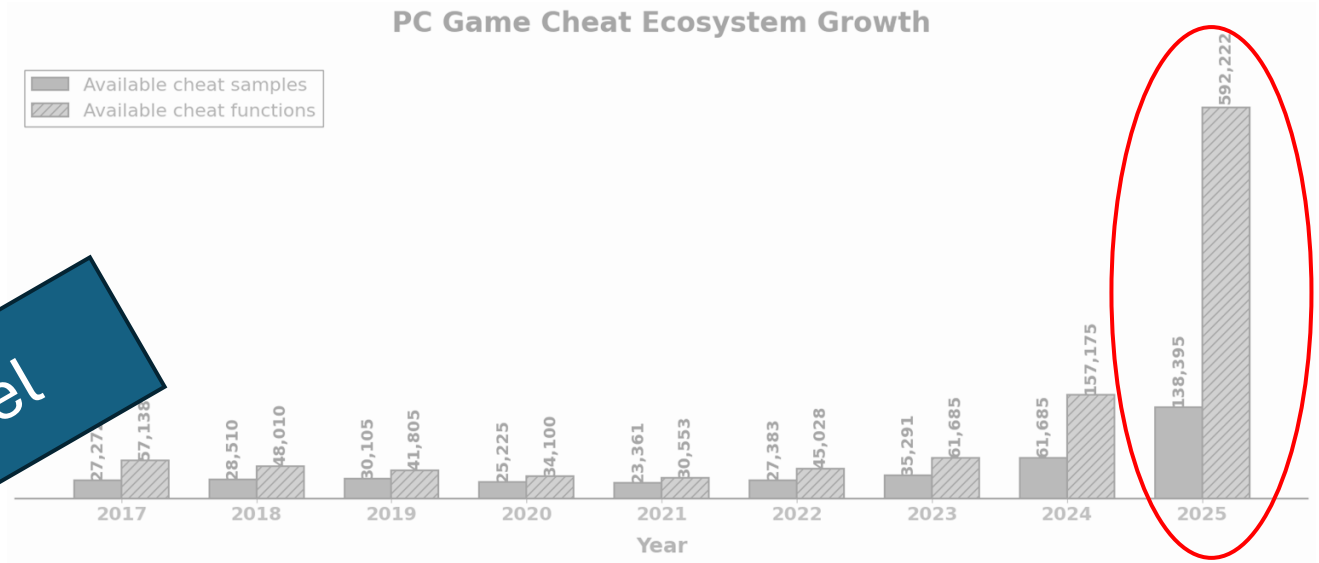


Today’s real-world FPS anti-cheat up against

EVERYTHING is black-box:

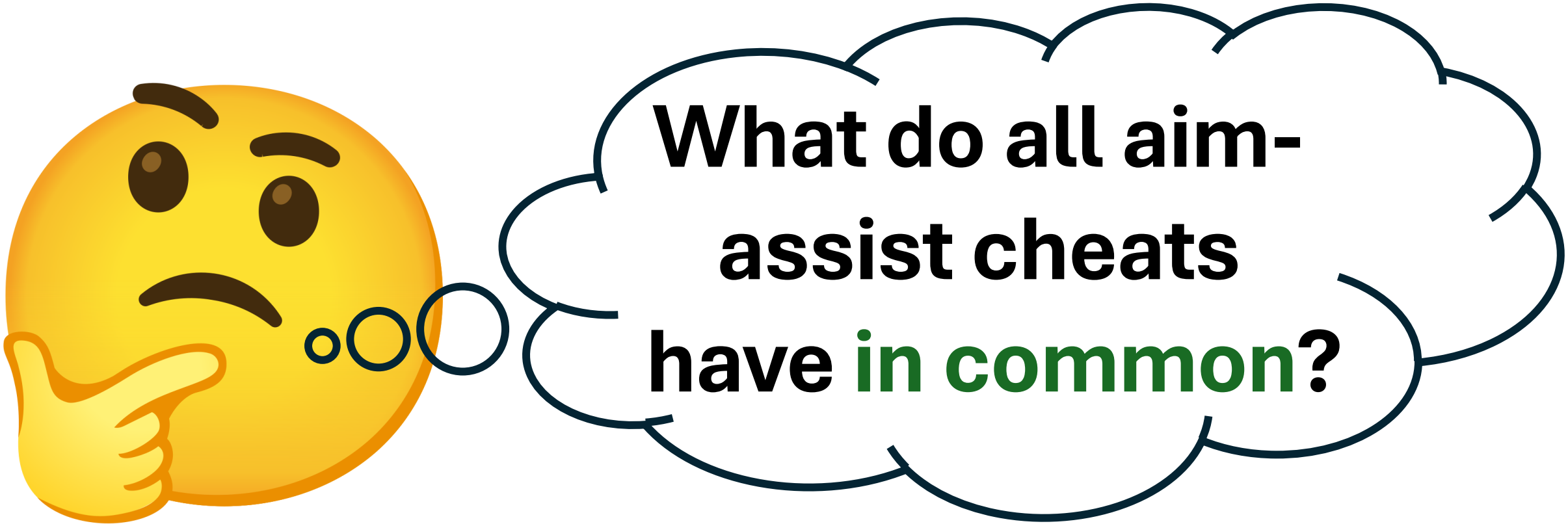
- cheating techniques
- configurations
- functions
- level of cheating sophistication
- cheat developers/vendors

Are all ***DIFFERENT***.

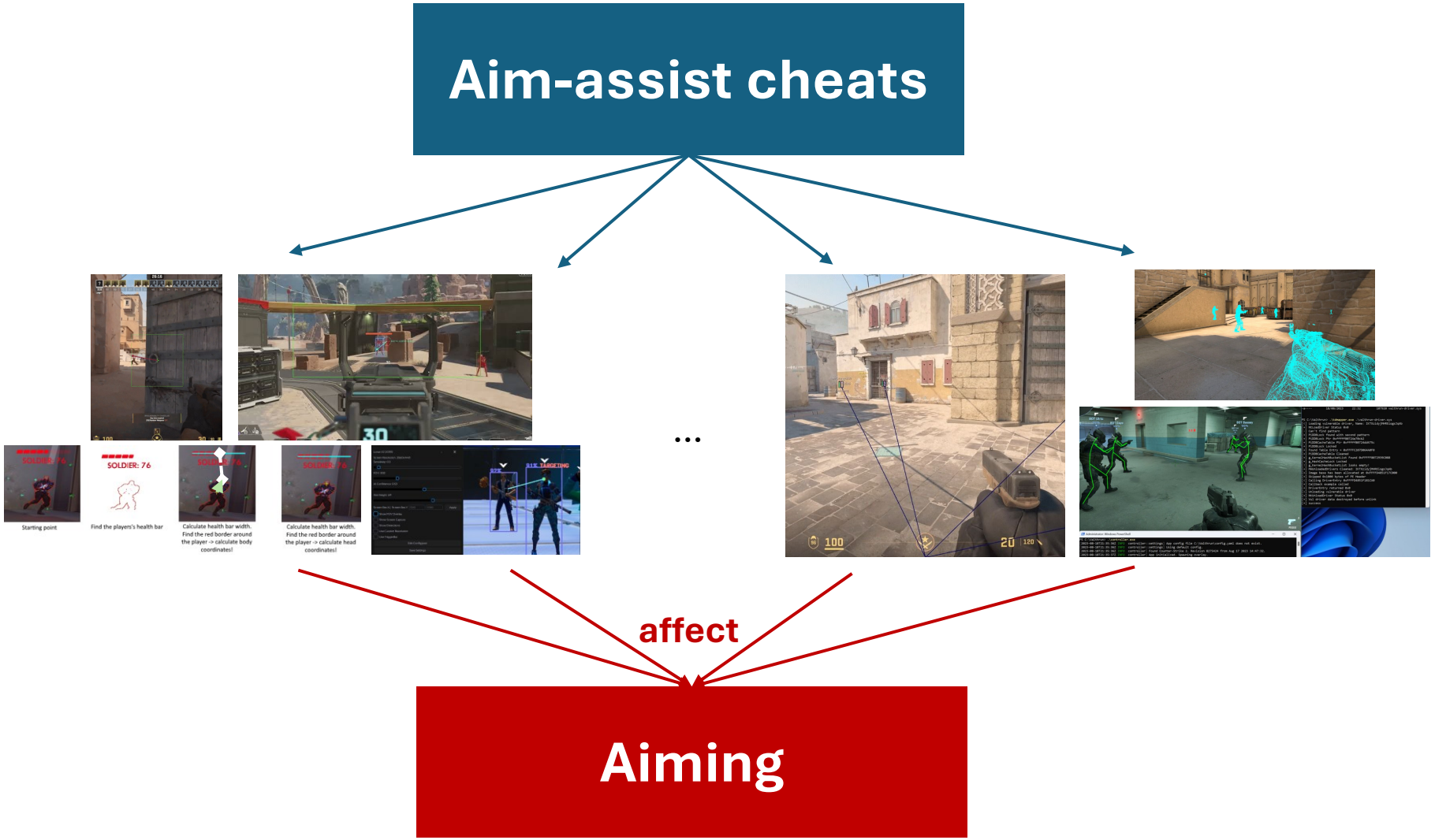


Our Threat Model

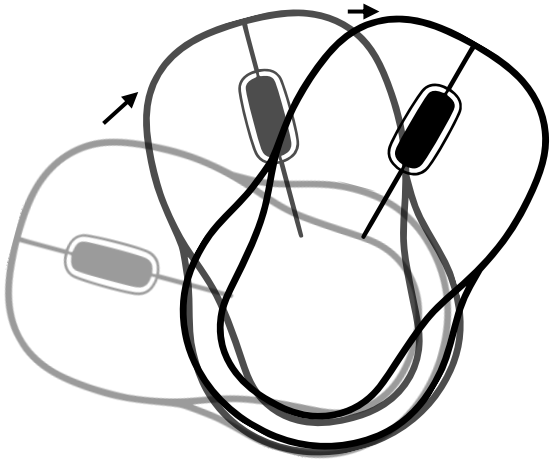
Is there a silver bullet for aim-assist cheats?



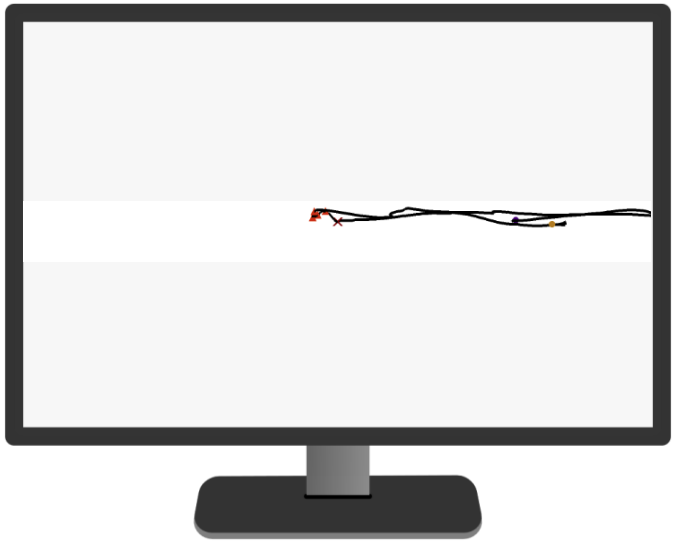
No aiming, no cheat.



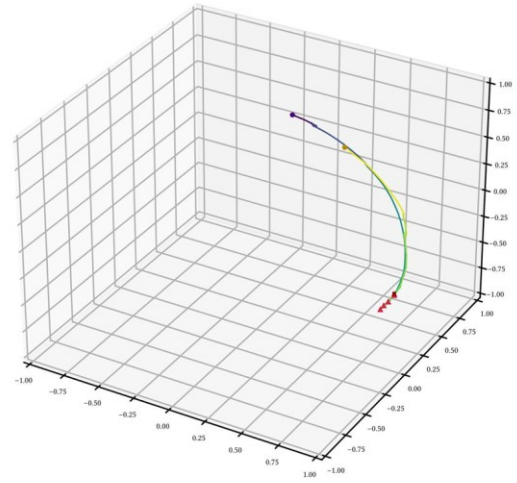
How does the aiming represent in different systems?



peripheral moves



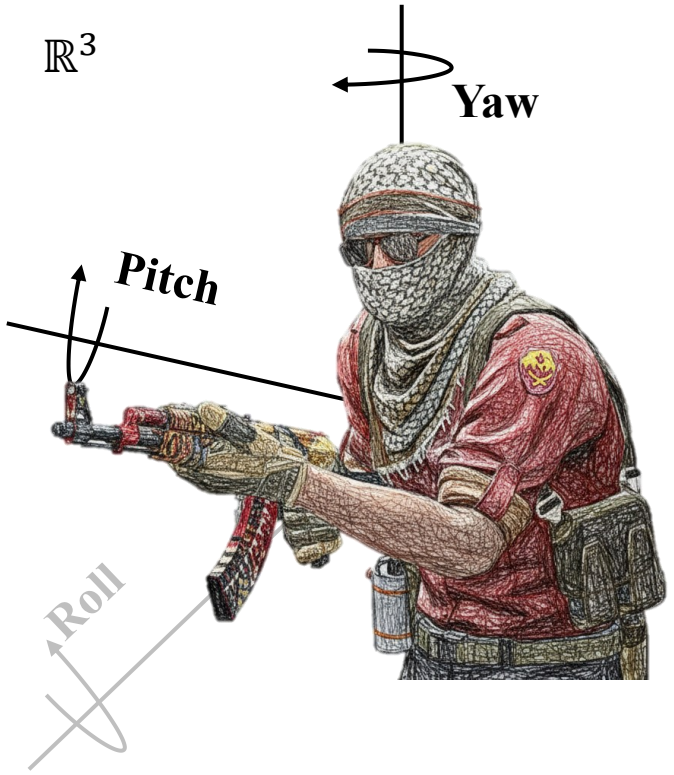
control crosshair aim on the monitor
(2D aim trajectory on the screen)



control avatar view movement
(3D aim trajectory in game)

How do avatars aim in the game world?

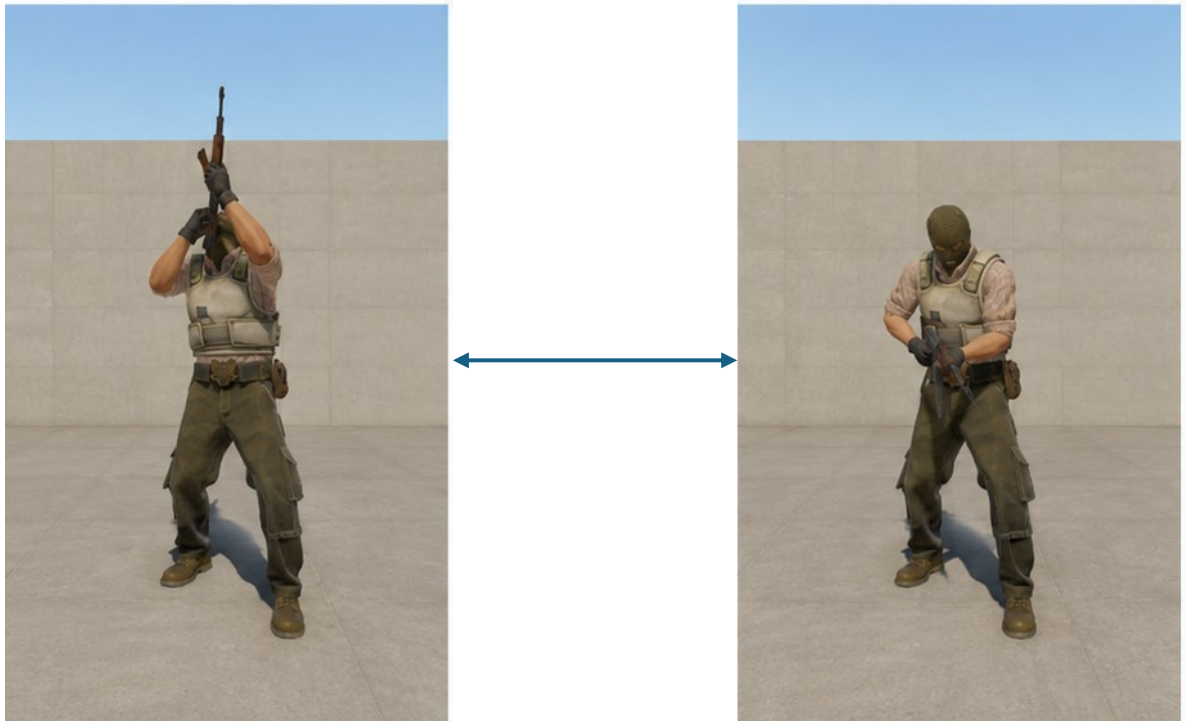
Pitch and ***yaw*** are **must-haves** in FPS games to **control the view**.



How do avatars aim in the game world?

Pitch is

- rotating on the **horizontal axis**
- the tilting of the view **up and down**
- range $[-90°, 90°]$



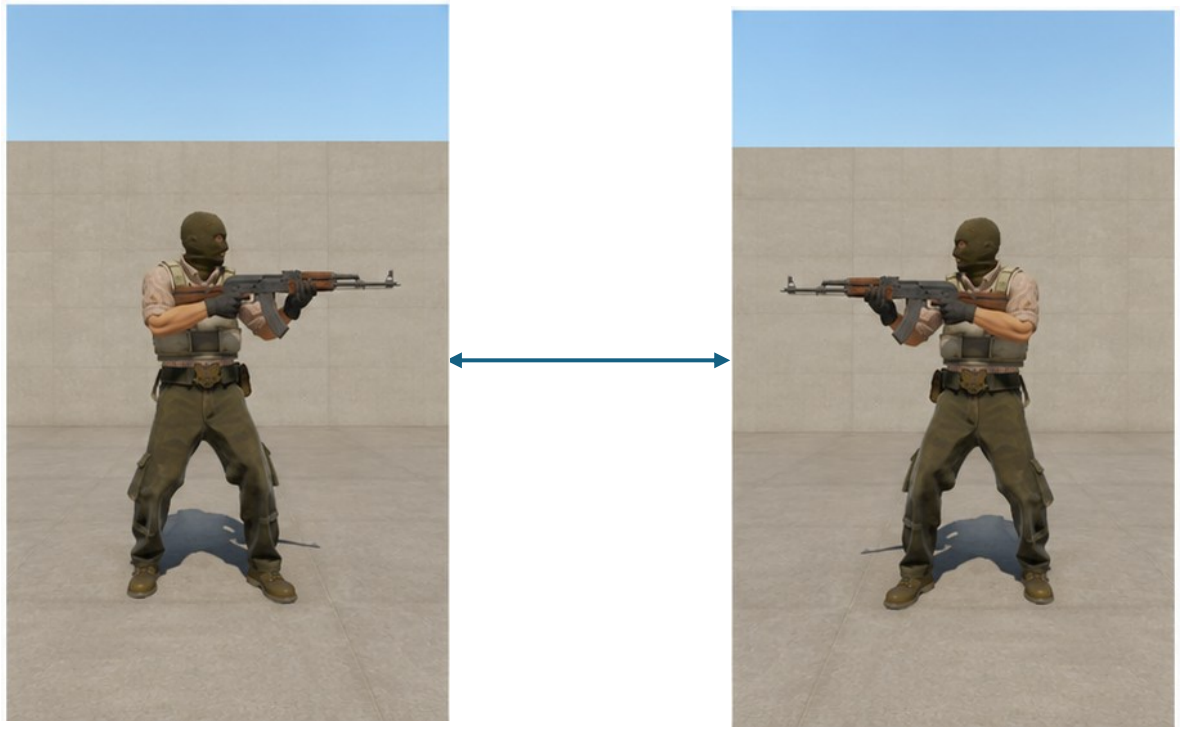
$-90°$
(looking straight up)

$90°$
(looking straight down)

How do avatars aim in the game world?

Yaw is

- rotating on the **vertical axis**
- the tilting of the view **left to right**
- range $[-180°, 180°]$



$\mathbf{0°}$
(to the east)

$\mathbf{-180°}$
(to the west)

How can we get the pitch and yaw?

Old fashion



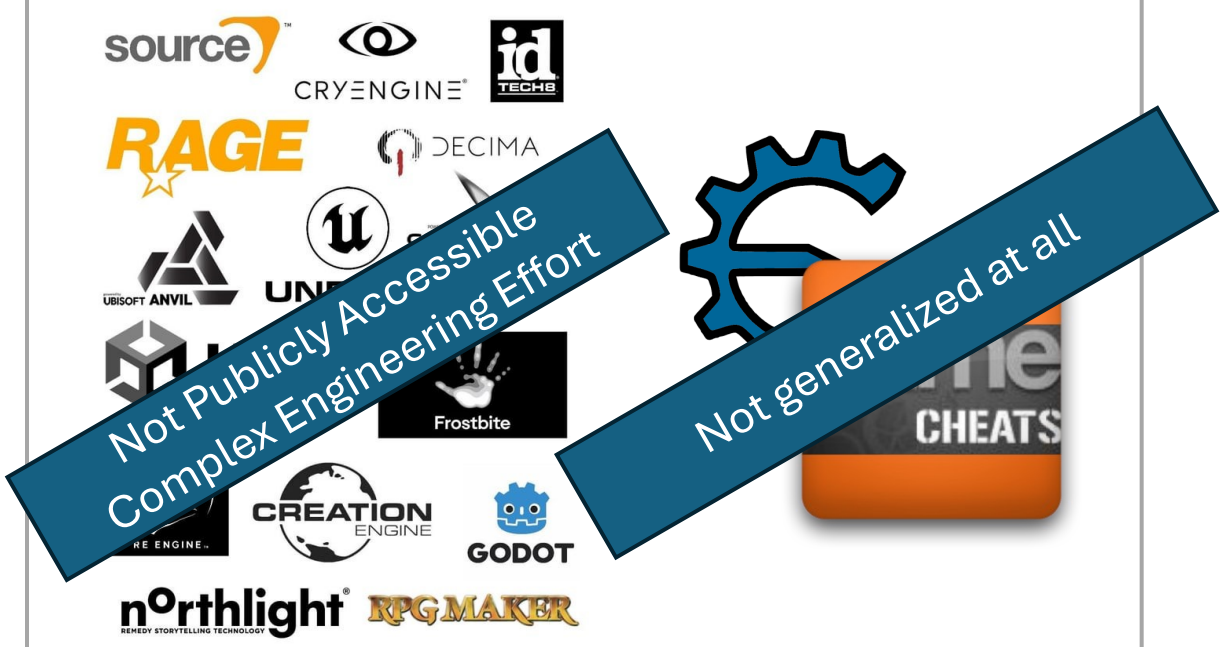
Modify engine



Use cheat's hook

How can we get the pitch and yaw?

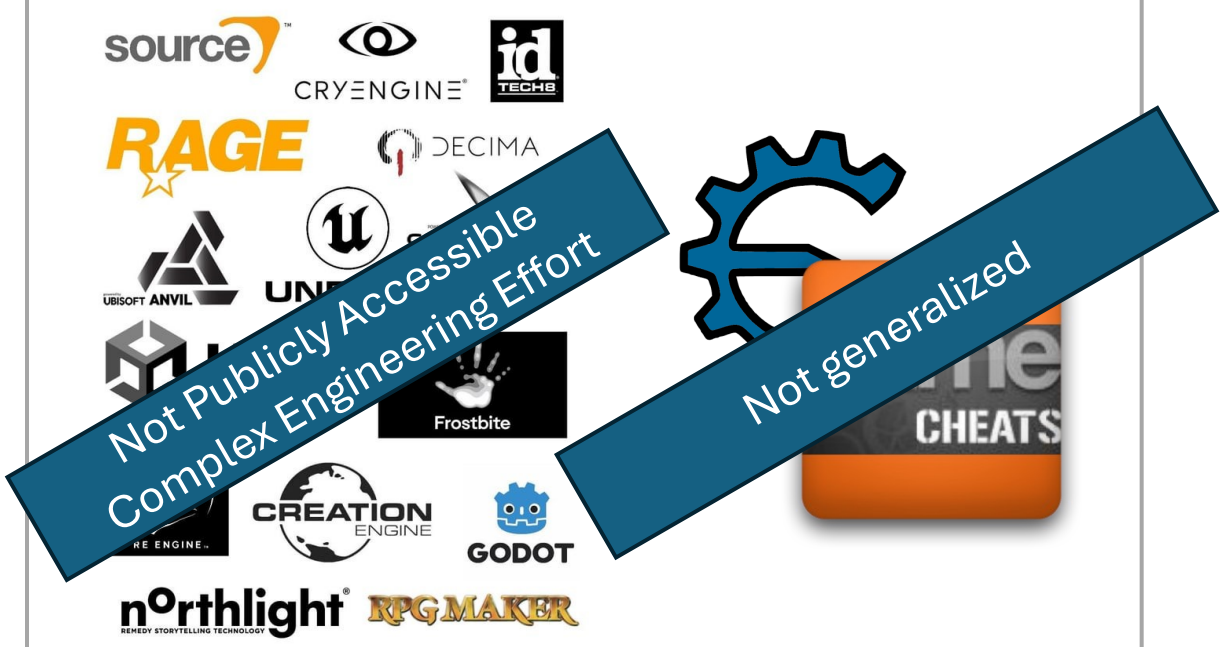
Old fashion



Modify engine **Use cheat's hook**

How can we get the pitch and yaw?

Old fashioned



Modify engine **Use cheat's hook**

Our method

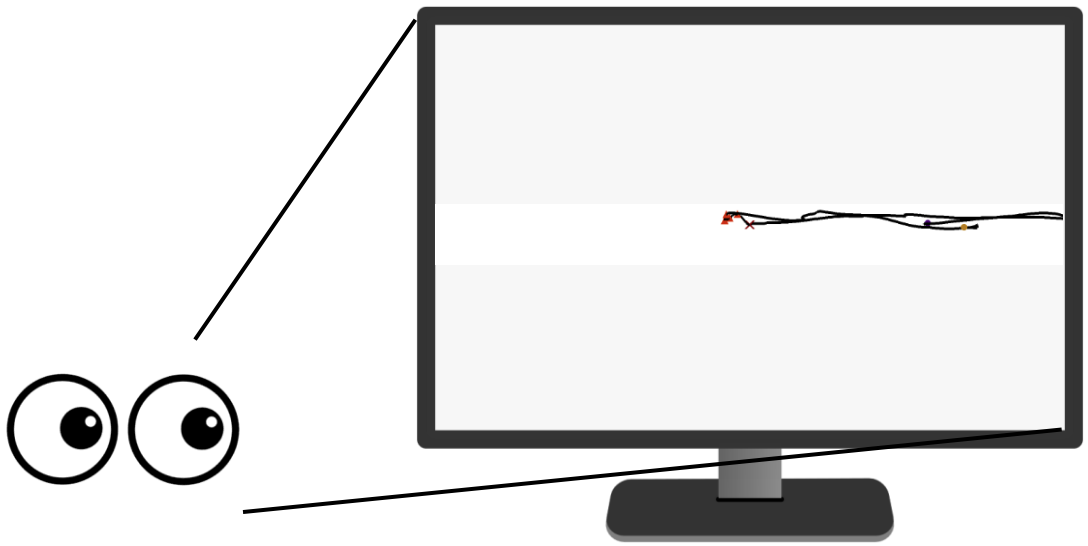
Replay

Replay file

Generalized **Lossless** **Non-Intrusive**

Publicly Accessible **Reconstructible**

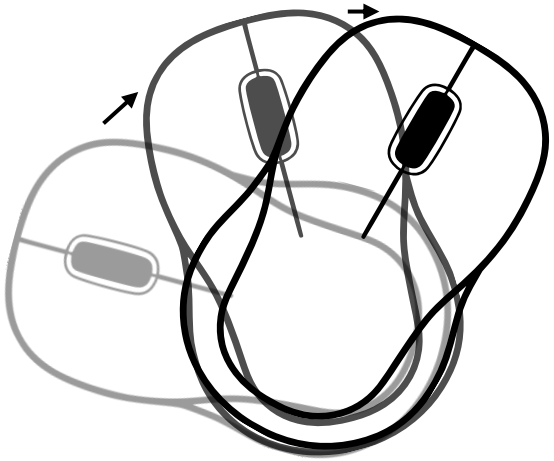
Which perspective does AC review expert check?



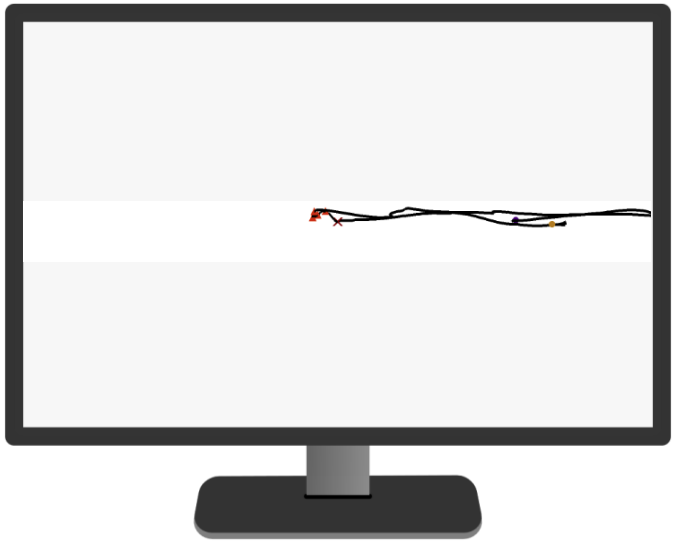
Watch the suspect's POV
2D trajectory!

How to convert pitch and yaw to 2D trajectory?

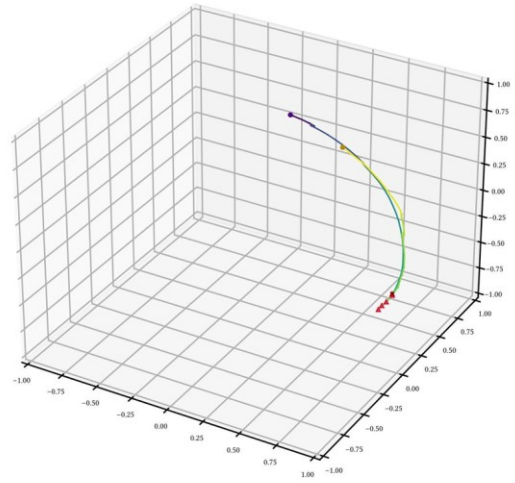
We want this for further analysis ← ? ↓ We have this now



peripheral moves



control crosshair aim on the monitor
(2D aim trajectory on the screen)

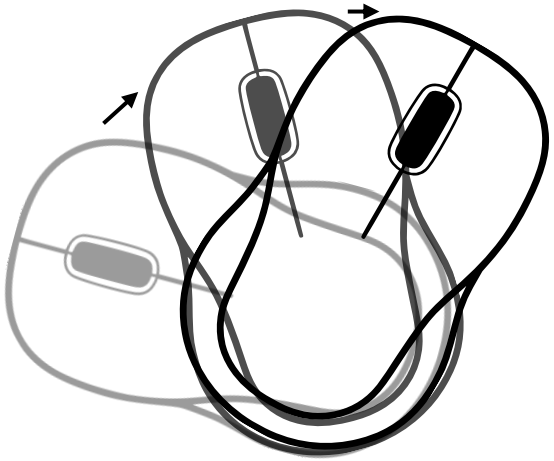


control avatar view movement
(3D aim trajectory in game)

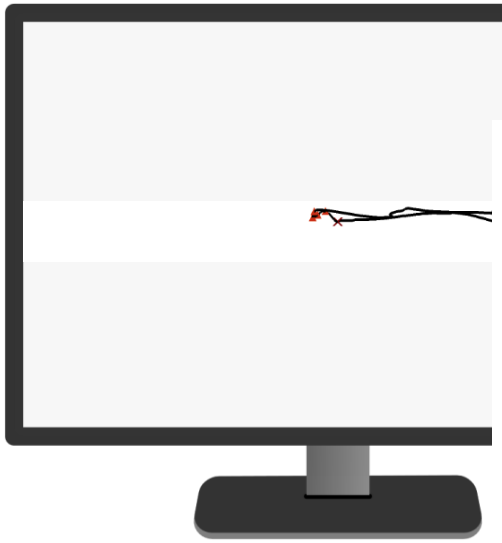
How to convert pitch and yaw to 2D trajectory?

We want this for further analysis

We have this now



peripheral moves



control crosshair aim on the monitor
(2D aim trajectory on the screen)

Algorithm 1 Convert Pitch and Yaw to 2D Coordinates

function PITCHYAWTOXY(pitch, yaw, width, height)

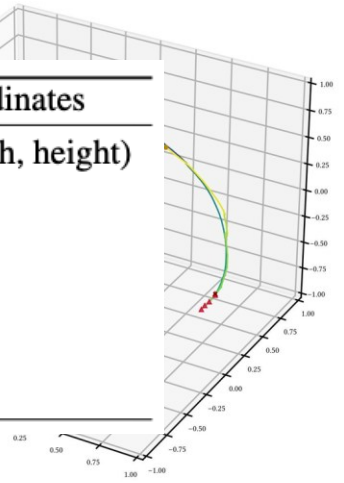
$x \leftarrow \left(\frac{yaw+180}{360}\right) \times width$

$y \leftarrow \left(\frac{pitch+90}{180}\right) \times height$

$y \leftarrow height - y$

return (x, y)

end function



control avatar view movement
(3D aim trajectory in game)

How does AC review expert identify cheaters?

Check suspect's **POV around kill** and see if there is ***unnatural***

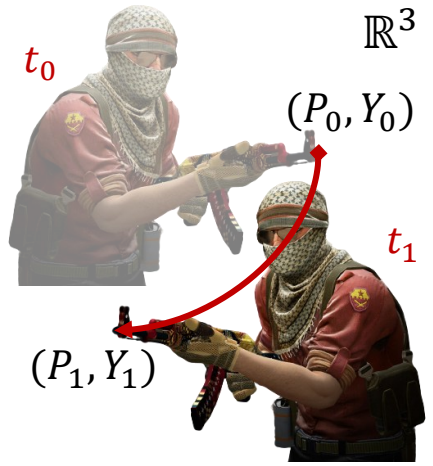
- **aim jitter (sudden bounce)**
- **aim speed and accuracy**
- **pre-aim stableness**

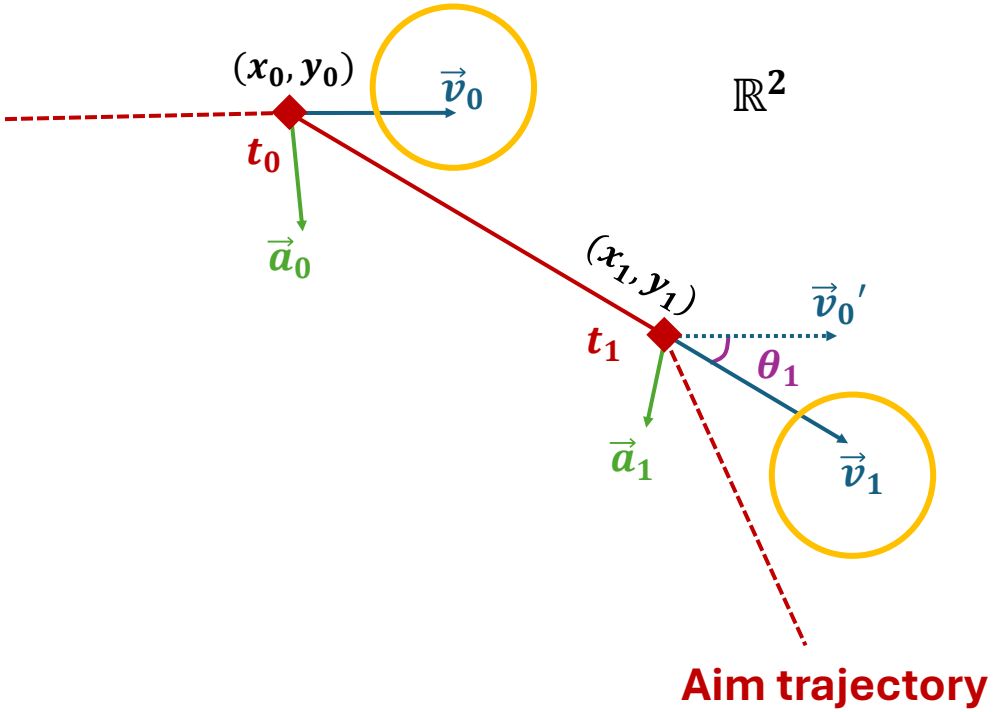
How to represent these aspects?

Velocities – *"aim speed"*

$$\vec{v}_x = \frac{dx}{dt}$$

$$\vec{v}_y = \frac{dy}{dt}$$



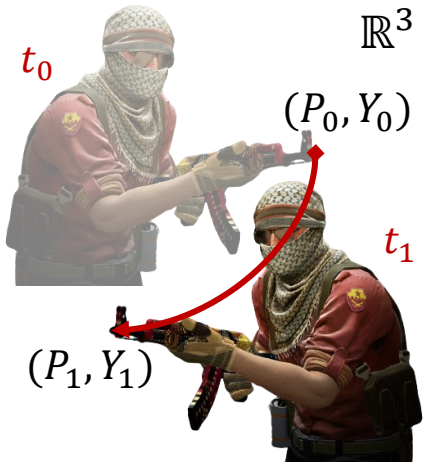


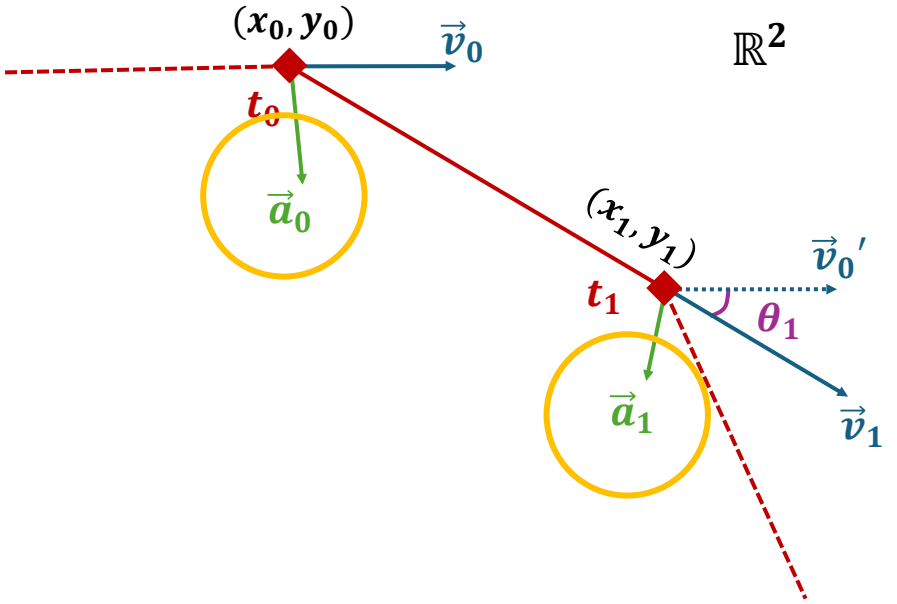
How to represent these aspects?

Accelerations – *“aim speed & jitter”*

$$\vec{a}_x = \frac{d\vec{v}_x}{dt} = \frac{d^2x}{dt^2}$$

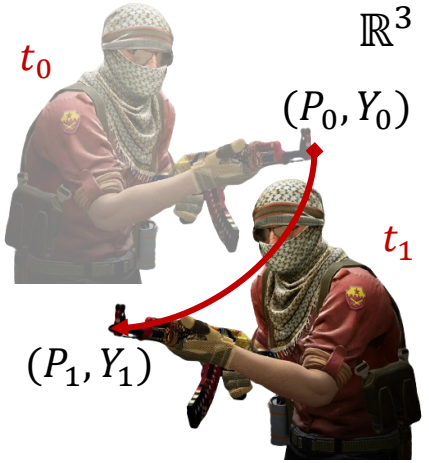
$$\vec{a}_y = \frac{d\vec{v}_y}{dt} = \frac{d^2y}{dt^2}$$





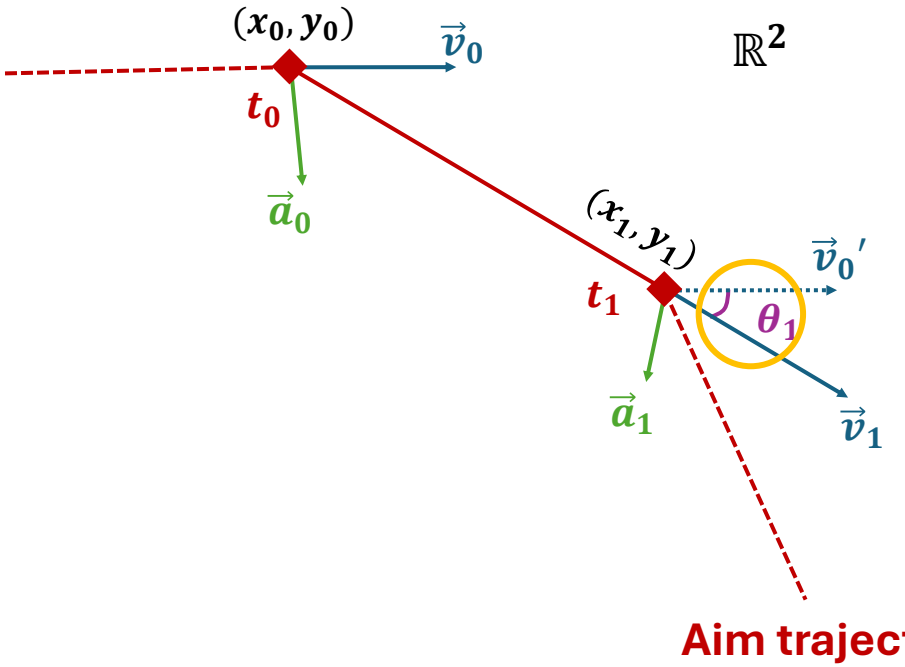
How to represent these aspects?

Angular change – *“aim jitter & pre-aim stableness”*



$$\alpha = \arctan 2(\vec{v}_y, \vec{v}_x)$$

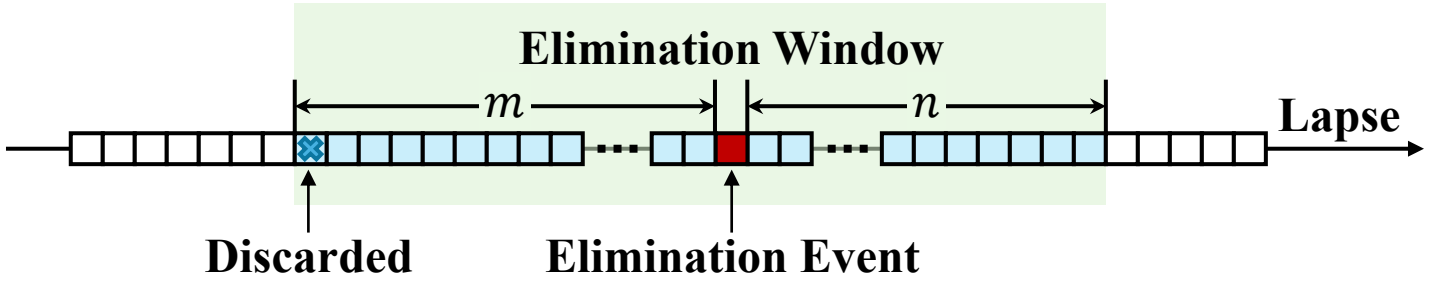
$$\theta = \frac{d\alpha}{dt}$$



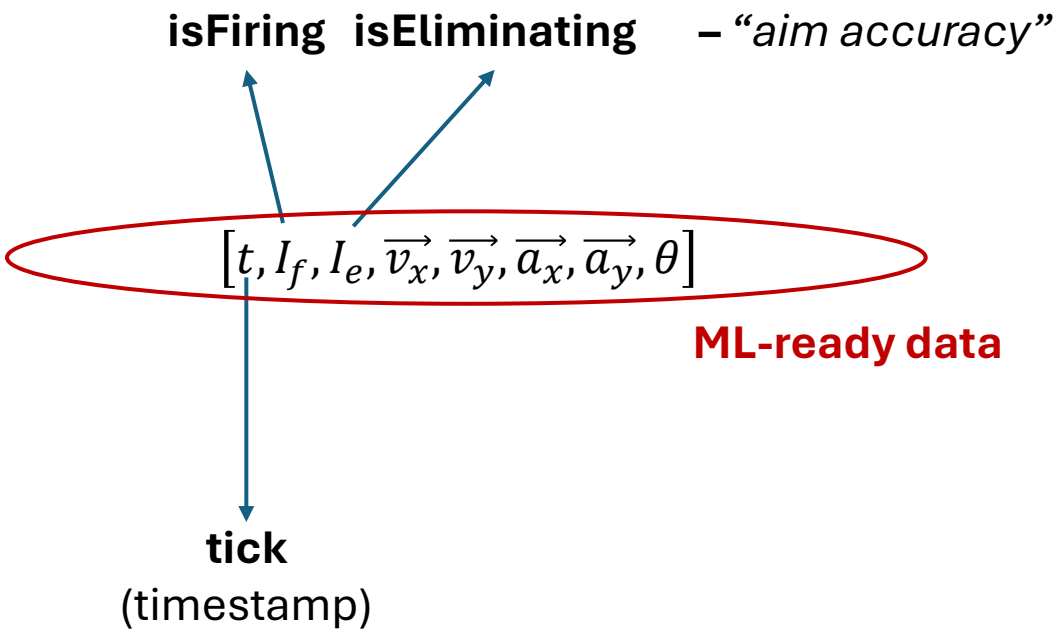
How to represent these aspects?

Elimination window – *“around kill”*

- most of the times players are idle – data may introduce **noise**!
- created when a player triggers an elimination event



How to represent these aspects?



Implementation

- GRU-CNN + FC aggregation + RF for elimination-wise and match-wise detection
- Modified GradientSHAP + TreeSHAP to explain the predictions

XGuardian contributions

- Effective, efficient and generalized solution
- Explainable visualizations for each kill and each match
- Real-world datasets

Datasets

Primary evaluations

- *Counter Strike 2*
- Real-world data and matches from *5E-Play* [1]

Table 2: Dataset description.

Data Split	A	B	C
#Tick	992,064	1,014,624	1,062,528
#Total Elimination Window	10,334	10,569	11,068
#Cheating Elimination Window	2,385	2,760	2,900
#Normal Elimination Window	7,949	7,809	8,168
#Match	971	944	988
#Cheater	108	115	127
#Normal Player	1,711	1,671	1,754
#Total Player	1,819	1,786	1,881

[1] 5EPlay. https://arena.5eplay.com/.

Datasets

Generalizability evaluation

- *Farlight84 & CS:GO* (from *HAWK* [1])
- Real-world data and matches from *5E-Play* and *Lilith Games* [2]

Table 18: FARLIGHT84 dataset descriptions.

Scale	Train		Validation		Test	
	#Normal	#Cheater	#Normal	#Cheater	#Normal	#Cheater
Sample	11,402	2,601	8,552	1,951	8,553	1,952
Match	7,554	1,493	6,147	1,232	6,149	1,256
Player	5,513	64	4,678	64	4,691	66

Table 19: HAWK dataset descriptions.

Scale	Train		Validation		Test	
	#Normal	#Cheater	#Normal	#Cheater	#Normal	#Cheater
Sample	152,113	12,273	146,543	11,970	146,887	12,313
Match	1,007	539	975	528	993	541
Player	8,971	357	8,660	336	8,800	336

[1] J. Zhang et al., "Identify as a Human Does: A Pathfinder of Next-Generation Anti-Cheat Framework for First-Person Shooter Games," in IEEE Transactions on Information Forensics and Security, vol. 21, pp. 240-255, 2026, doi: 10.1109/TIFS.2025.3635024.
[2] Lilith Games. https://ancient.lilith.com/.

What experiments have we done?

- Detection performance
- Feature ablation study
- Inspector's model and parameter selection
- Comparative study with prior works
- Generalizability study
- Explainability case study
- Robustness study under adversarial attack
- Overhead study
- User study on Explainer’s potential bias

What experiments have we done?

- Detection performance
- Feature ablation study
- Inspector's model and parameter selection
- Comparative study with prior works
- **Generalizability study**
- **Explainability case study**
- Robustness study under adversarial attack
- Overhead study
- User study on Explainer’s potential bias

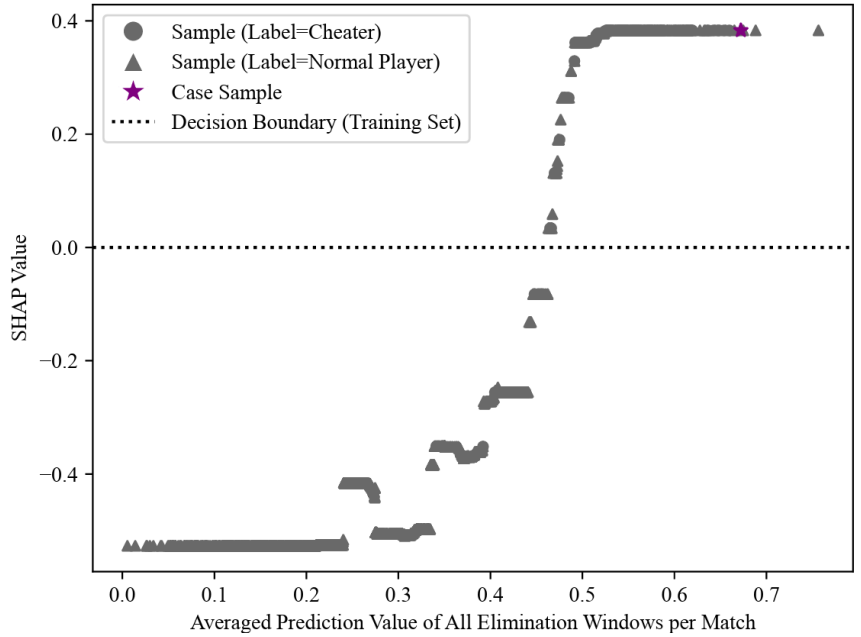
Generalizability study

- Different titles, sub-genres, platforms and game engines
- Stable performance
- Results are weighted averaged due to imbalanced dataset

Dataset	HAWK [63]	FARLIGHT84	XGUARDIAN
Game (Release Year)	CS:GO (2012)	Farlight84 (2023)	CS2 (2023)
Sub-genre	Tactical Shooter	Battle Royal	Tactical Shooter
Engine	Source	Unreal Engine 4	Source 2
Platform	PC	Mobile	PC
Accuracy	0.841	0.816	0.889
FPR	0.056	0.144	0.062
Recall	0.841	0.816	0.889
Precision	0.944	0.856	0.938
F1-Score	0.878	0.829	0.907

Explainability case study

- **Per-frame** visualization analysis for different features
- **Per-match** analysis for demonstrating the distribution among all matches





XGuardian: Towards Generalized, Explainable and More Effective Server-side Anti-cheat in First-Person Shooter Games

Jiayi Zhang, Chenxin Sun, Chenxiong Qian

HKU

Links

Artifact

Paper & Tech Rec.

Demo

XGuardian: Towards Generalized, Explainable and More Effective Server-side Anti-cheat in First-Person Shooter Games

Jiayi Zhang, Chenxin Sun, Chenxiong Qian

HKU



香 港 大 學

THE UNIVERSITY OF HONG KONG

Gaming industry is thriving

Online gaming market is projected to grow to ***$386 billion*** by 2026.



Picture Sources:
[1] Liquipedia Wiki
[2] OWCS EWC 25 Champions

FPS game genre is market leader

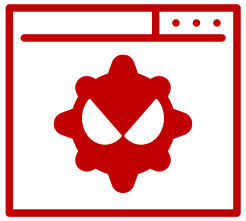
First-**P**erson **S**hooter (**FPS**) game holds 20.9% market share.



Video Source:
[1] Counter-Strike 2

FPS games are vulnerable to cheating

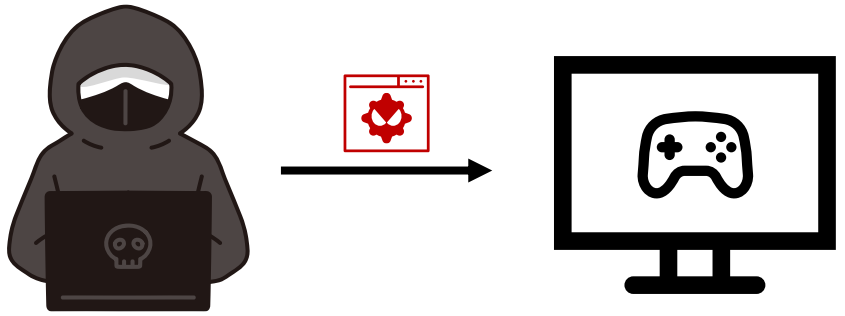
- Malicious software for games is called the **cheat**
  - Cheat developers seek for monetary gain



Cheat

FPS games are vulnerable to cheating

- Malicious software for games is called the **cheat**
  - Cheat developers seek for monetary gain
- The person who uses the cheats is called the **cheater**
  - cheaters seek in-game illegal advantages against their opponents



Cheater

FPS games are vulnerable to cheating

- Malicious software for games is called the **cheat**
  - Cheat developers seek for monetary gain
- The person who uses the cheats is called the **cheater**
  - cheaters seek in-game illegal advantages against their opponents
- **Cheating** poses user churn and financial loss

FPS games are vulnerable to cheating

- Malicious software for games is called the **cheat**
  - Cheat developers seek for monetary gain
- The person who uses the cheats is called the **cheater**
  - cheaters seek in-game illegal advantages against their opponents
- **Cheating** poses user churn and financial loss
- **Anti-cheat** aims to detect or prevent cheating

Aim-assist cheats in FPS Games

Two types of aim-assist cheats in modern FPS games:

- **Aimbot** – auto-aim and auto-shoot



Video Source:
[1] Clean Aimbot

Aim-assist cheats in FPS Games

Two types of aim-assist cheats in modern FPS games:

- **Aimbot** – auto-aim and auto-shoot
  - Mutations: triggerbot (left), ai-aimbot (right)

Video Source:
[1] aptitude.pub

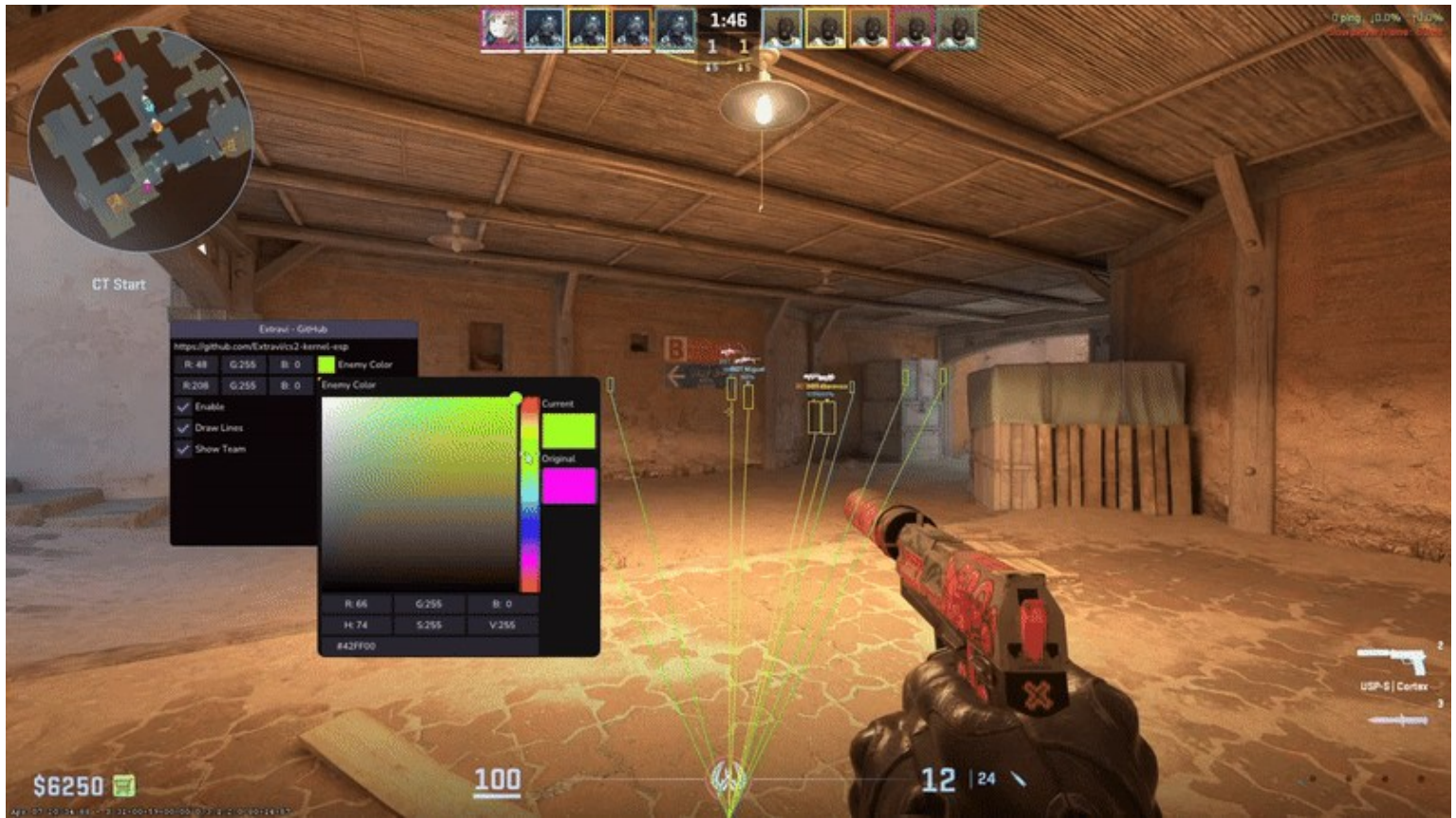




Aim-assist cheats in FPS Games

Two types of cheats in modern FPS games:

- **Wallhack** – see through obstacles



Video Source:
[1] Github cs2-kernel-esp

Aim-assist cheats in FPS Games

Aim-assist cheats commonly integrate *aimbots* with *wallhacks* and constitute a significant portion of all FPS cheats.

XGuardian addresses aim-assist cheats.

Server-side and client-side anti-cheat

Server-side anti-cheat systems are defined by their architectural design to collect, process, and analyze gameplay data exclusively on secure server infrastructure.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6] BotScreen	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4] HAWK	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1] **Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2] **Feature**: ⚔ denotes part or all of the features are not generalized; [3] **Evaluated**: If the method's generalizability is evaluated through experiment(s); [4] **SSL**: Simulated, small-scale, low complexity; [5] **Unknown**: Have not been evaluated.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6]	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4]	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1] **Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2] **Feature**: ⚔ denotes part or all of the features are not generalized; [3] **Evaluated**: If the method's generalizability is evaluated through experiment(s); [4] **SSL**: Simulated, small-scale, low complexity; [5] **Unknown**: Have not been evaluated.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6]	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4]	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1]**Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2]**Feature**: ⚔ denotes part or all of the features are not generalized; [3]**Evaluated**: If the method's generalizability is evaluated through experiment(s); [4]**SSL**: Simulated, small-scale, low complexity; [5]**Unknown**: Have not been evaluated.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6]	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4]	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1]**Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2]**Feature**: ⚔ denotes part or all of the features are not generalized; [3]**Evaluated**: If the method's generalizability is evaluated through experiment(s); [4]**SSL**: Simulated, small-scale, low complexity; [5]**Unknown**: Have not been evaluated.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6]	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4]	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1] **Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2] **Feature**: ⚔ denotes part or all of the features are not generalized; [3] **Evaluated**: If the method's generalizability is evaluated through experiment(s); [4] **SSL**: Simulated, small-scale, low complexity; [5] **Unknown**: Have not been evaluated.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6]	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4]	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1] **Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2] **Feature**: ⚔ denotes part or all of the features are not generalized; [3] **Evaluated**: If the method's generalizability is evaluated through experiment(s); [4] **SSL**: Simulated, small-scale, low complexity; [5] **Unknown**: Have not been evaluated.

Existing solutions' limitations

[L1] Client-side dependency and resource constraints

[L2] Limited real-world performance

[L3] Limited generalizability

[L4] Lack of explainability

Table 1: ML-based anti-cheat detection works comparison.

Side	Method	Work(s)	Availability[1]	Generalizability			Interpretability	Real-world Functionality		
				Data Collection	Feature[2]	Evaluated[3]		Dataset	System Constraint	Performance
Client	Unsupervised Angle Anomaly Detection	[6]	T, C, D	✗	⚔	✗	✗	SSL[4]	TEE (Intel SGX)	Unknown[5]
	Statistical Classification	[8, 11]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
Server	Statistical Classification	[7, 9, 10]	T, C, D	✗	✓	✗	✗	SSL	Concurrent overhead	Unknown
	Multi-view Ensemble Learning	[4]	T, C, D	✓	⚔	✗	✗	Real-world	None	Low
	Temporal Aim Trajectory Classification	XGUARDIAN	T, C, D	✓	✓	✓	✓	Real-world	None	High

[1] **Availability**: Public availability of technical details (T), source code (C), and datasets (D), red and green indicate unavailable and available; [2] **Feature**: ⚔ denotes part or all of the features are not generalized; [3] **Evaluated**: If the method's generalizability is evaluated through experiment(s); [4] **SSL**: Simulated, small-scale, low complexity; [5] **Unknown**: Have not been evaluated.

Propose XGuardian

Explainable **Guardian** (XGuardian)

- Avoids the inherent limitations of client-side solutions and leverages replay files as the data source to perform detection asynchronously during server idle periods, minimizing overhead **[L1]**.

Propose XGuardian

Explainable **Guardian** (XGuardian)

- Avoids the inherent limitations of client-side solutions and leverages replay files as the data source to perform detection asynchronously during server idle periods, minimizing overhead **[L1]**.
- Achieves state-of-the-art performance on a large, complex real-world dataset **[L2]** and is the first work to demonstrate wide generalizability by validating its effectiveness on large datasets from different sub-genres of games **[L3]**.

Propose XGuardian

Explainable **Guardian** (XGuardian)

- Avoids the inherent limitations of client-side solutions and leverages replay files as the data source to perform detection asynchronously during server idle periods, minimizing overhead **[L1]**.
- Achieves state-of-the-art performance on a large, complex real-world dataset **[L2]** and is the first work to demonstrate wide generalizability by validating its effectiveness on large datasets from different sub-genres of games **[L3]**.
- Addresses the explainability bottleneck by instantly pinpointing the exact suspicious frames and features within a match **[L4]**.

Replay system and replay file

- **Replay system** is a dynamic toolset for player interaction with recorded content post-game and has been extensively adopted in notable FPS games
- **Replay file** is a comprehensive record of a match with unique data format that can be interpreted by the replay system

XGuardian uses replay files to extract raw data.

Cheating sophistication

Cheaters dynamically adjust sophistication levels of using aim-assist cheats across two axes:

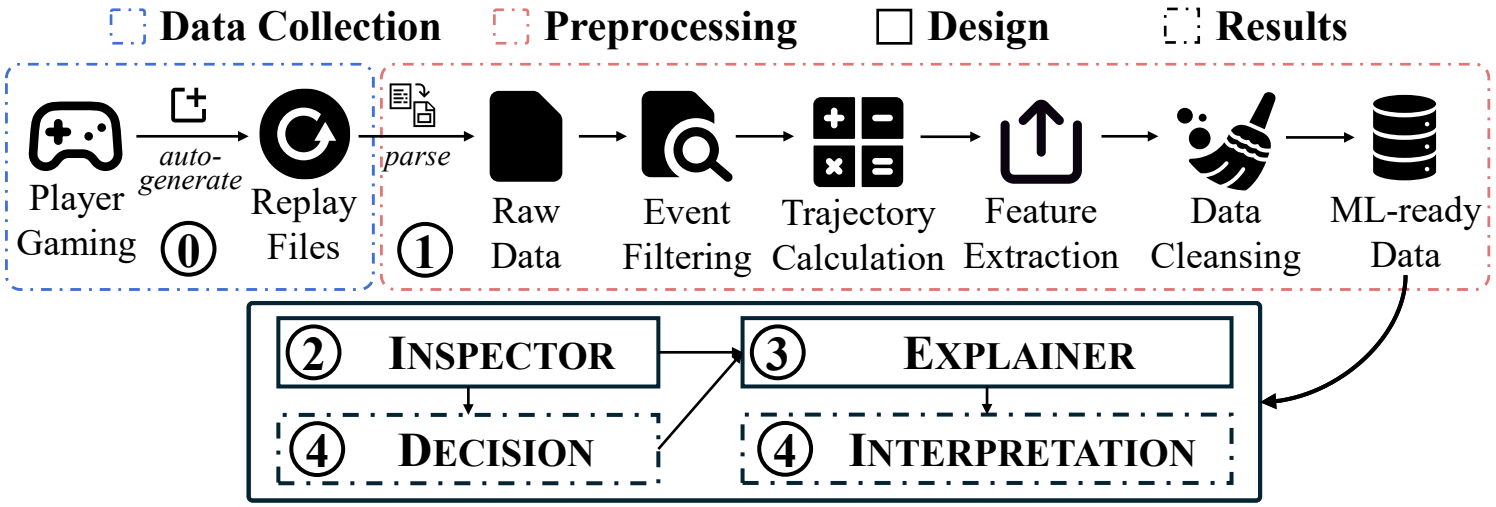
- Functional granularity (e.g., targeting specific body part *vs.* full-body locks)
- Stealth tradeoffs (e.g., blatant play *vs.* behavioral mimicry of legitimate players)

Threat Model

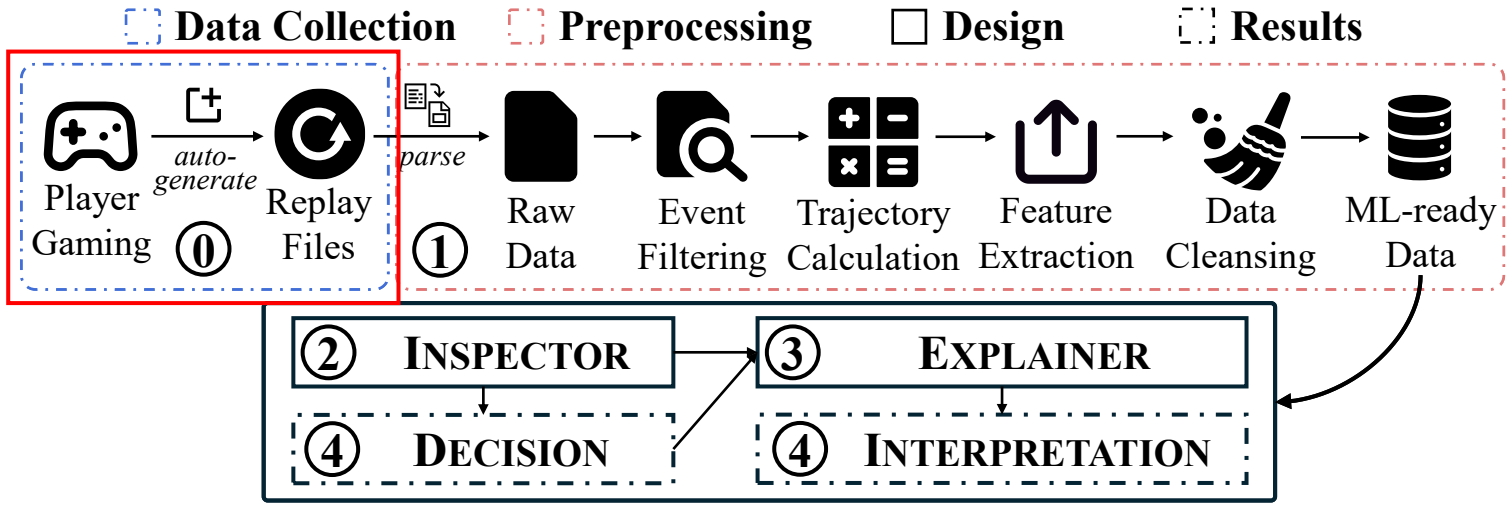
The cheaters

- Utilize aim-assist cheats
- Cheat sophistications are black-box

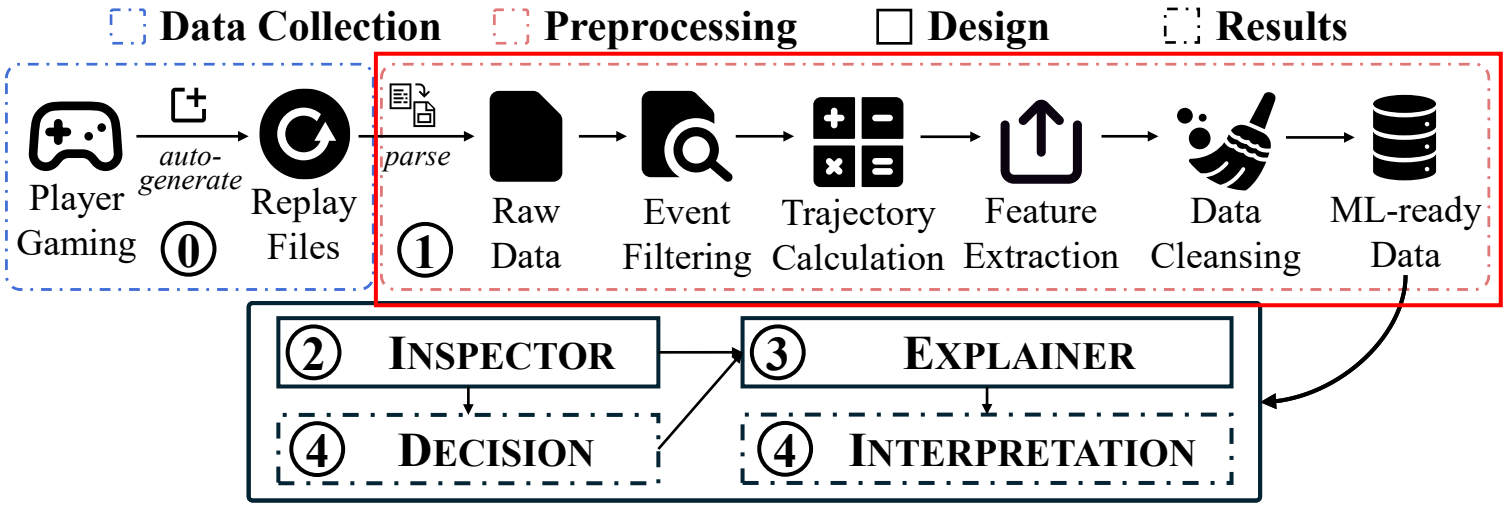
Workflow


Data Collection
Preprocessing
Design
Results
Player Gaming
auto-generate
Replay Files
parse
Raw Data
Event Filtering
Trajectory Calculation
Feature Extraction
Data Cleansing
ML-ready Data
0
1
2 INSPECTOR
3 EXPLAINER
4 DECISION
4 INTERPRETATION

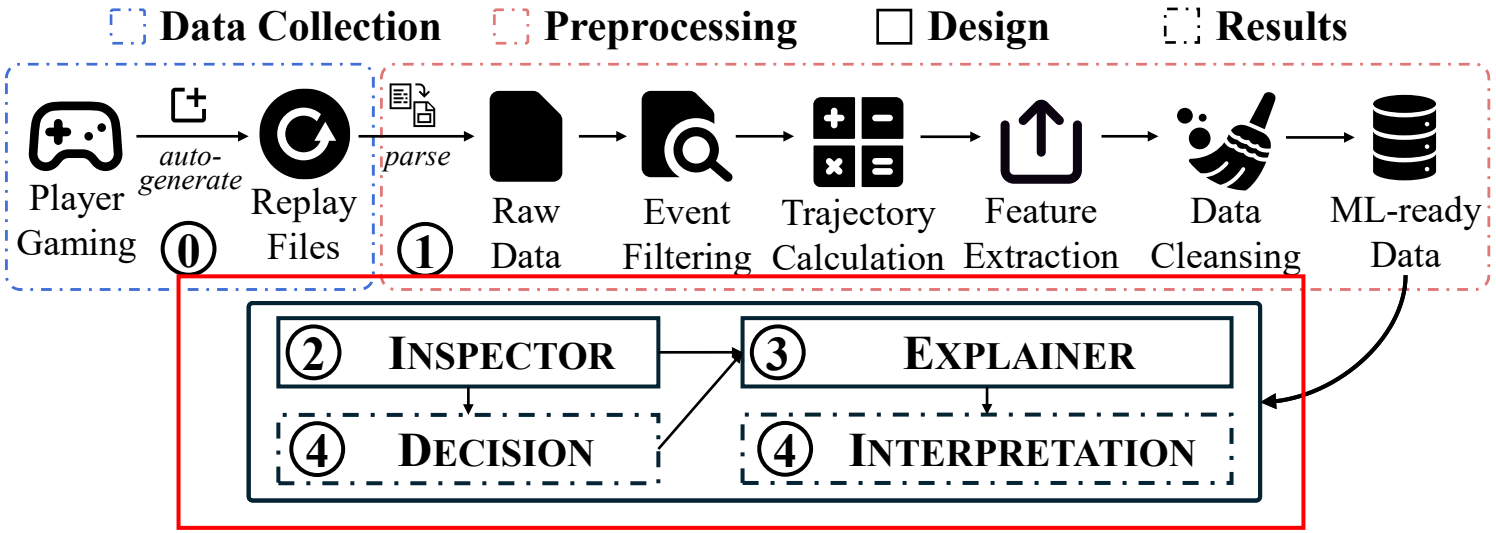
Workflow


Data Collection
Preprocessing
Design
Results
Player Gaming
auto-generate
Replay Files
parse
Raw Data
Event Filtering
Trajectory Calculation
Feature Extraction
Data Cleansing
ML-ready Data
0
1
2 INSPECTOR
3 EXPLAINER
4 DECISION
4 INTERPRETATION

Workflow


Data Collection
Preprocessing
Design
Results
Player Gaming
auto-generate
Replay Files
0
parse
1
Raw Data
Event Filtering
Trajectory Calculation
Feature Extraction
Data Cleansing
ML-ready Data
2 INSPECTOR
3 EXPLAINER
4 DECISION
4 INTERPRETATION

Workflow


Data Collection
Preprocessing
Design
Results
Player Gaming
auto-generate
Replay Files
parse
Raw Data
Event Filtering
Trajectory Calculation
Feature Extraction
Data Cleansing
ML-ready Data
0
1
2 INSPECTOR
3 EXPLAINER
4 DECISION
4 INTERPRETATION

Dimensions of movement in FPS games

Pitch and Yaw

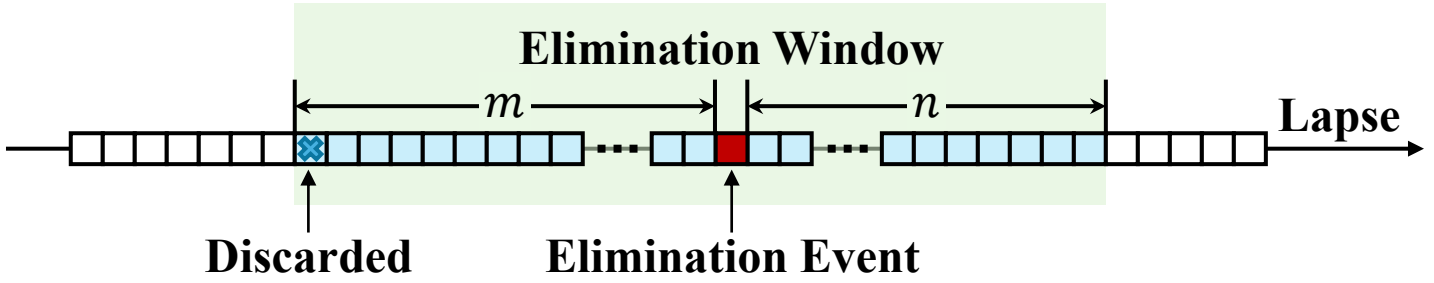
- **Pitch** is the vertical angle of the player's view.
- **Yaw** is the horizontal angle of the player's view.



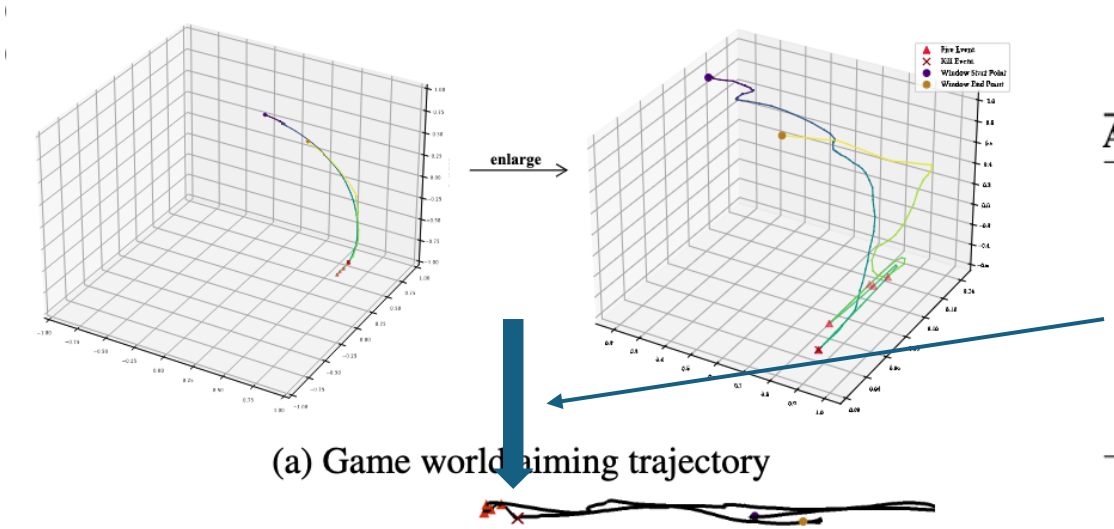
Extraction scope – per elimination

Elimination Window

- Created when a player triggers an elimination event (conduct a kill on an opponent).
- Size is 96.



3D aiming trajectory extraction and 2D mapping



(a) Game world aiming trajectory

(b) Monitor screen aiming trajectory

Figure 4: The example of 3D trajectory (a) and 2D screen trajectory mapping (b). The enlarged 3D trajectory (right) uses different measurement units for each coordinate axis for better illustration. 2D trajectory visualization is mapped onto a 1920×1080 resolution, and the coordinates of the elimination event are relocated to the center of the screen.

Algorithm 1 Convert Pitch and Yaw to 2D Coordinates

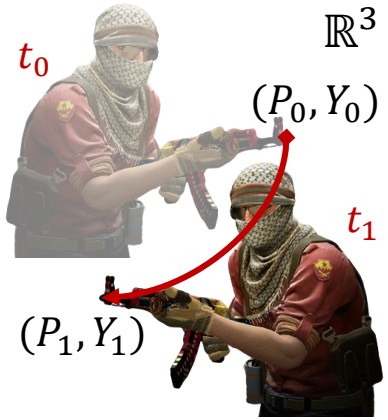
function PITCHYAWTOXY(pitch, yaw, width, height)
  $x \leftarrow \left(\frac{yaw+180}{360}\right) \times width$
  $y \leftarrow \left(\frac{pitch+90}{180}\right) \times height$
  $y \leftarrow height - y$
  return (x, y)
end function

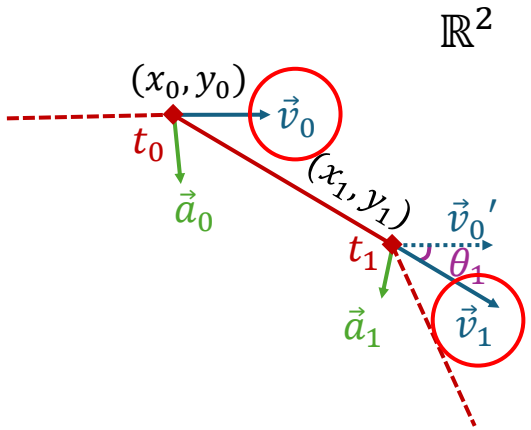
Feature construction

Velocity

$$\vec{v}_x = \frac{dx}{dt} \quad (1)$$

$$\vec{v}_y = \frac{dy}{dt} \quad (2)$$



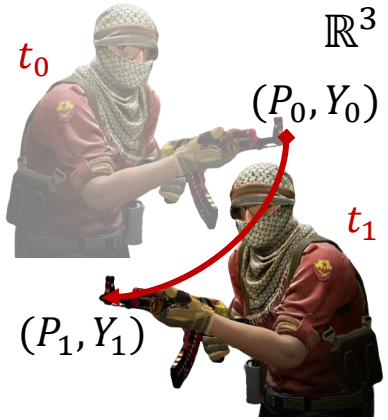


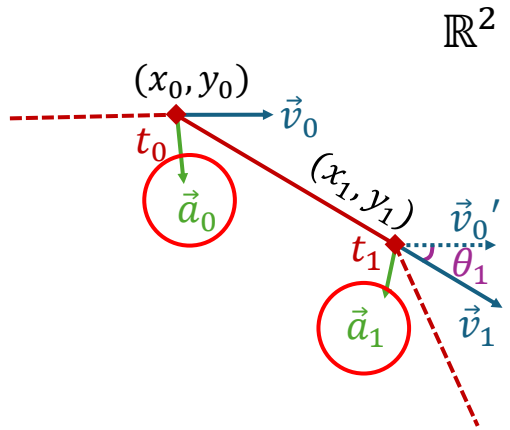
Feature construction

Acceleration

$$\vec{a}_x = \frac{d\vec{v}_x}{dt} = \frac{d^2x}{dt^2} \quad (3)$$

$$\vec{a}_y = \frac{d\vec{v}_y}{dt} = \frac{d^2y}{dt^2} \quad (4)$$



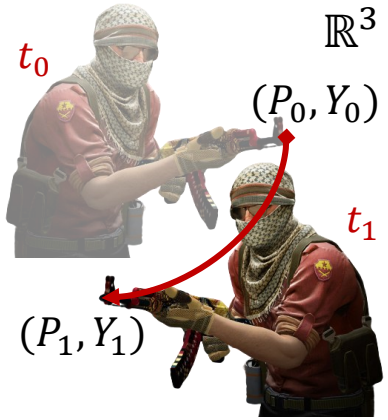


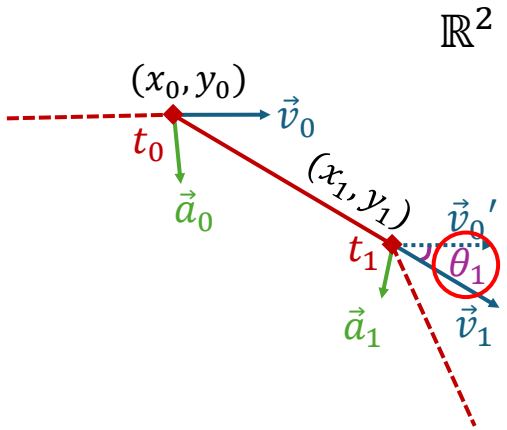
Feature construction

Angular Change

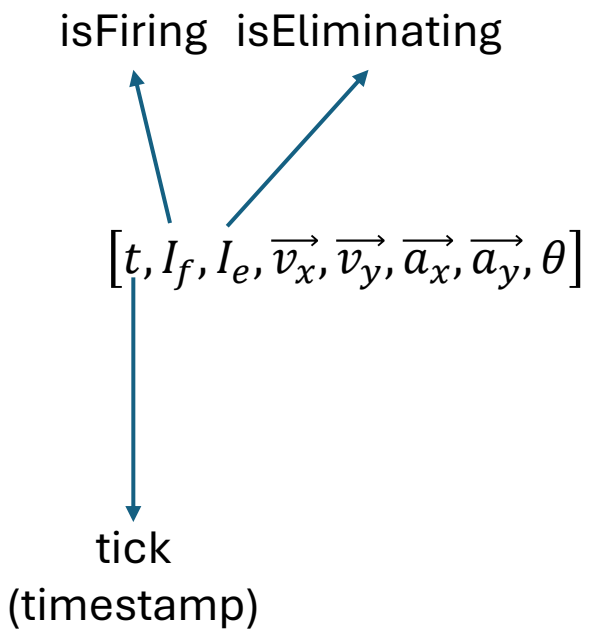
$$\alpha = \arctan 2(\vec{v}_y, \vec{v}_x) \tag{5}$$

$$\theta = \frac{d\alpha}{dt} \tag{6}$$





ML-ready data



Framework

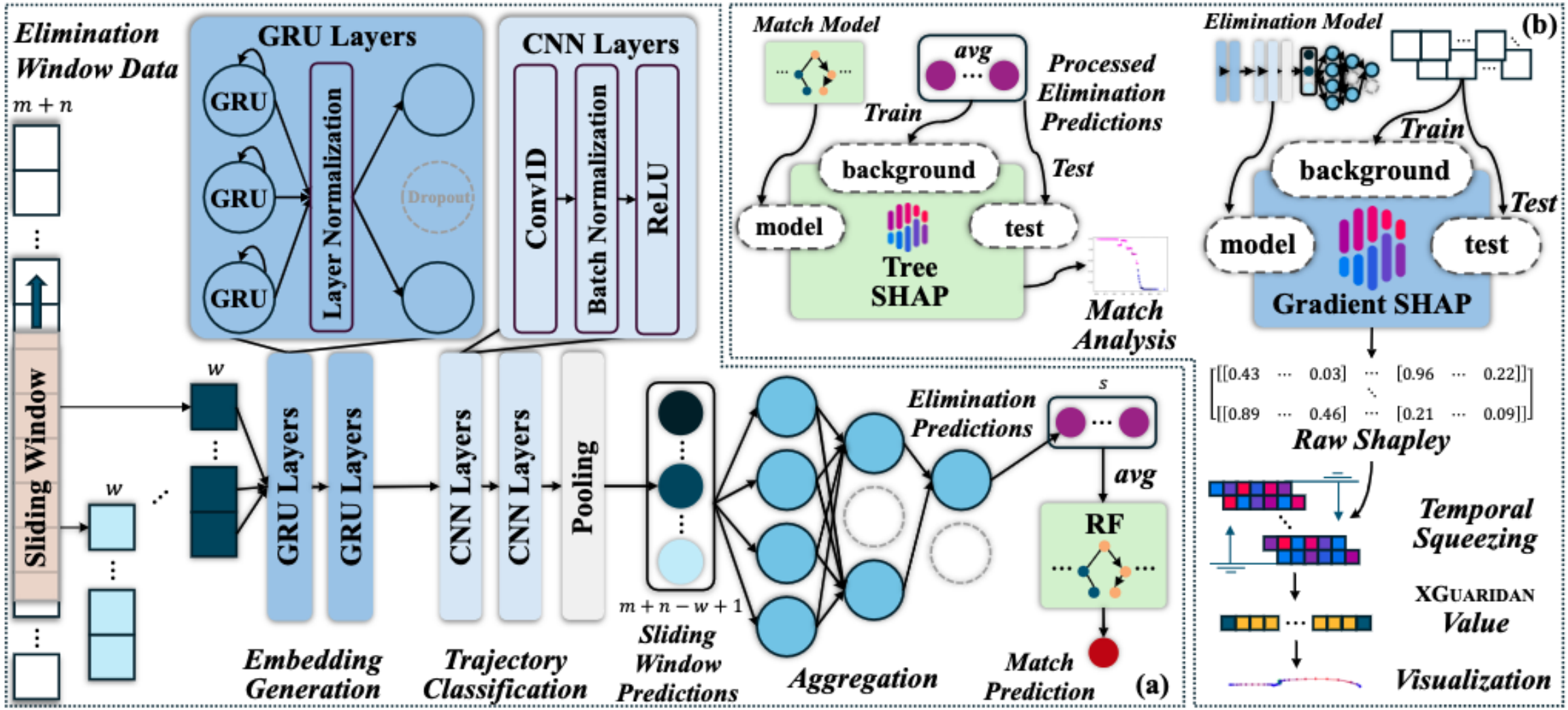


Figure 5: XGuardian framework. The subfigures illustrate the architectures of (a)-Inspector and (b)-Explainer.

Framework

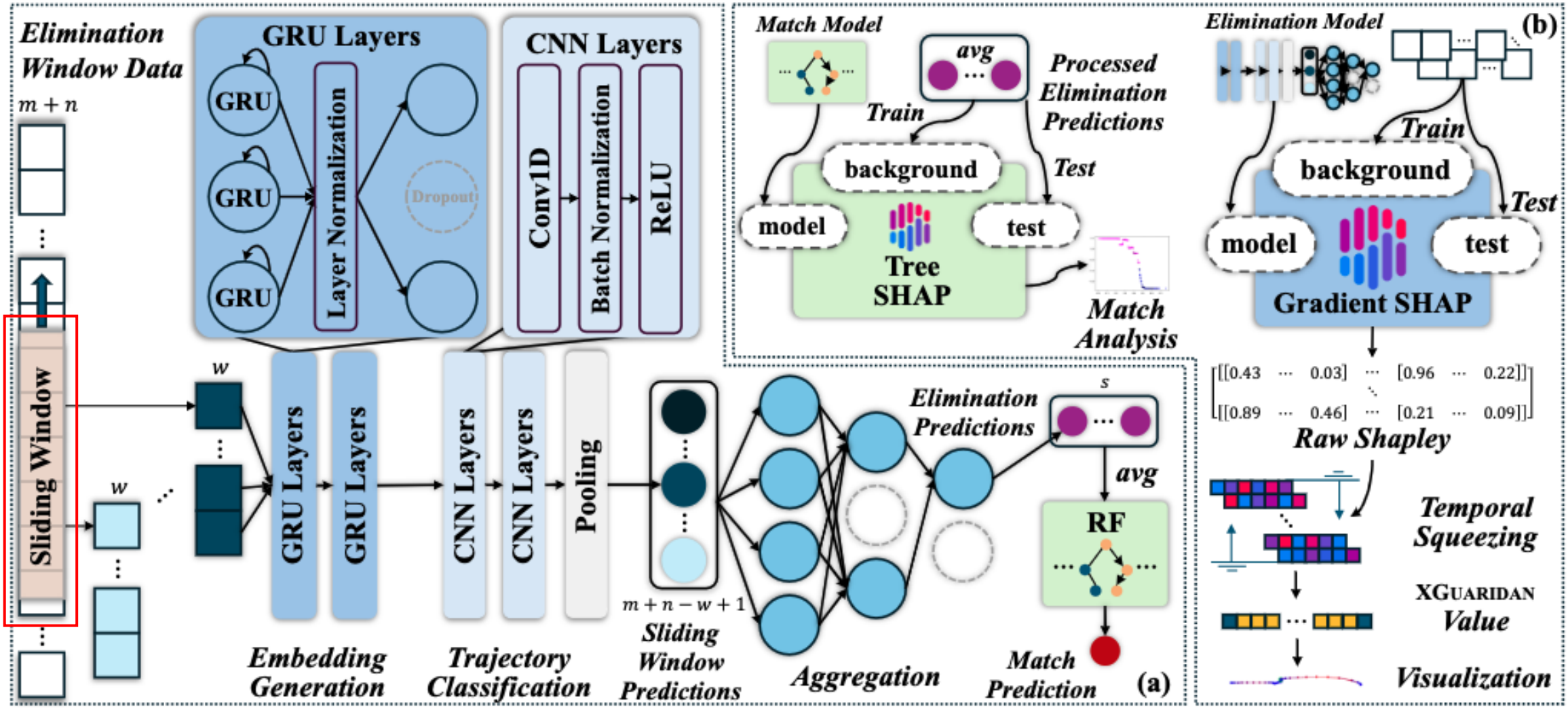


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

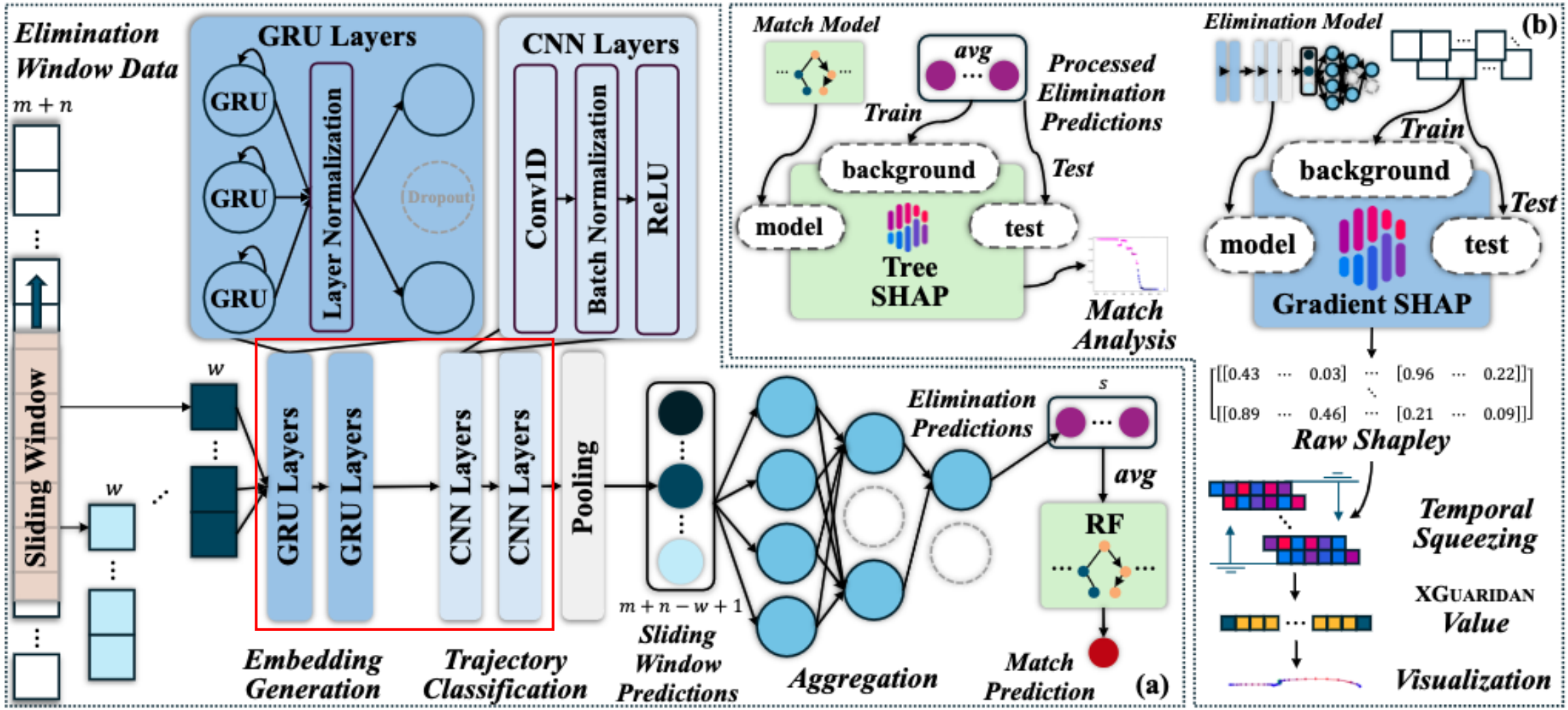


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

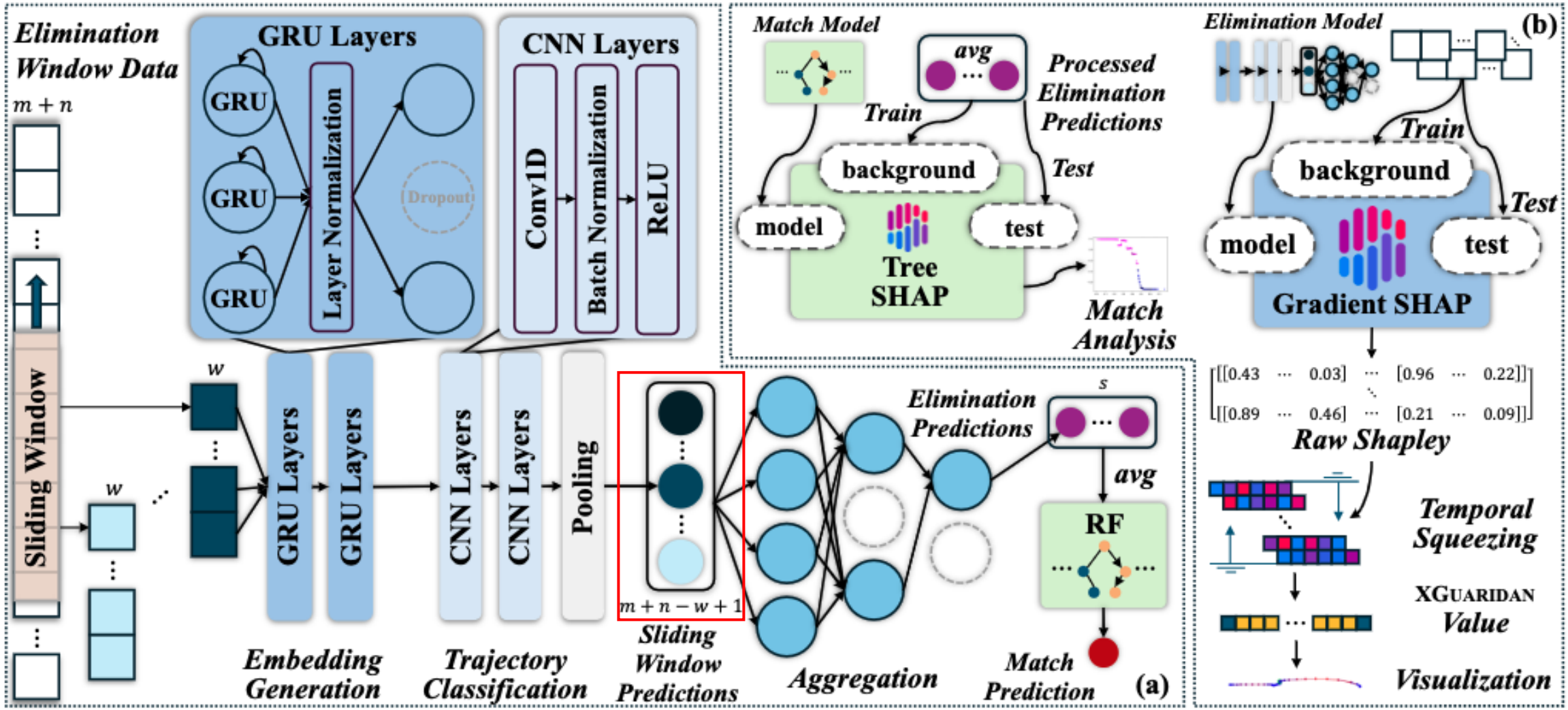


Figure 5: XGuardian framework. The subfigures illustrate the architectures of (a)-Inspector and (b)-Explainer.

Framework

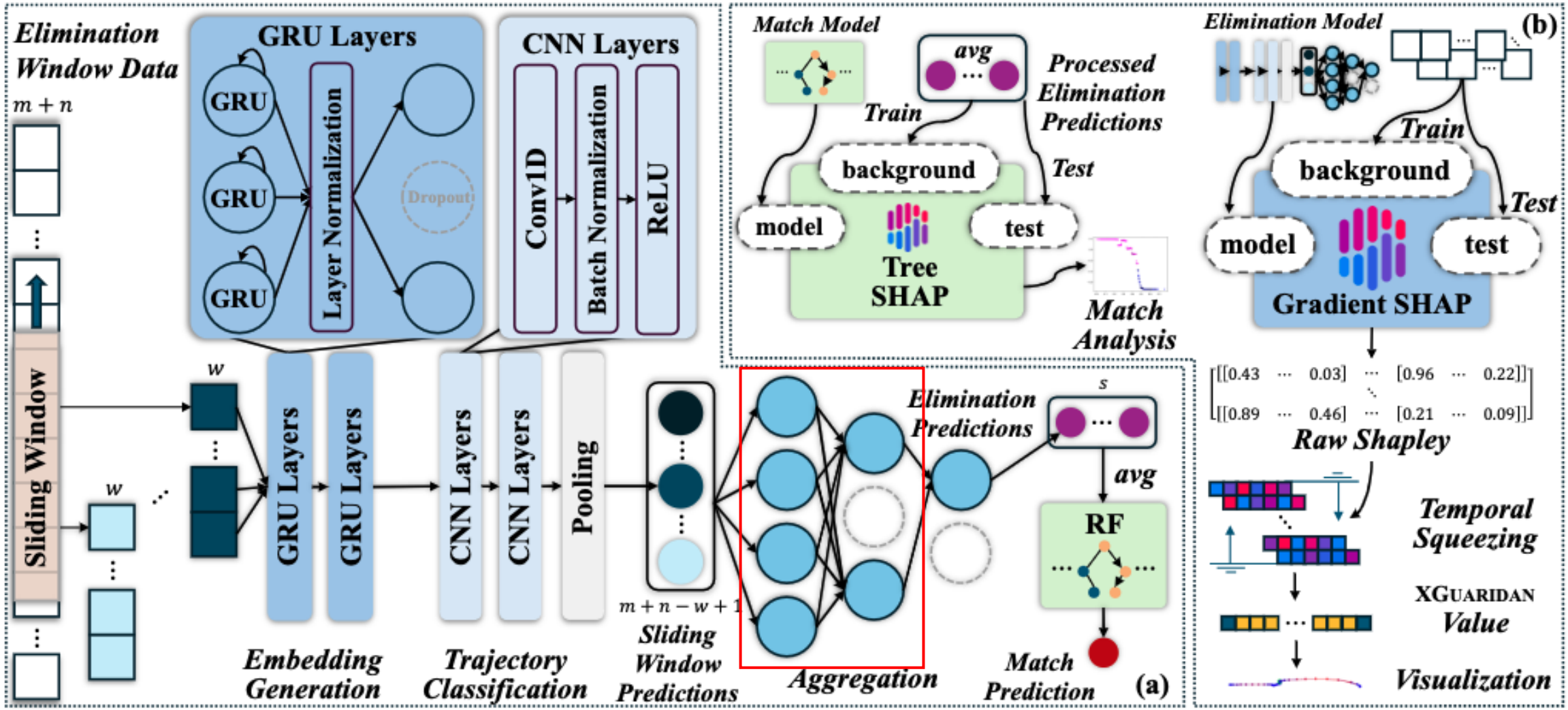


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

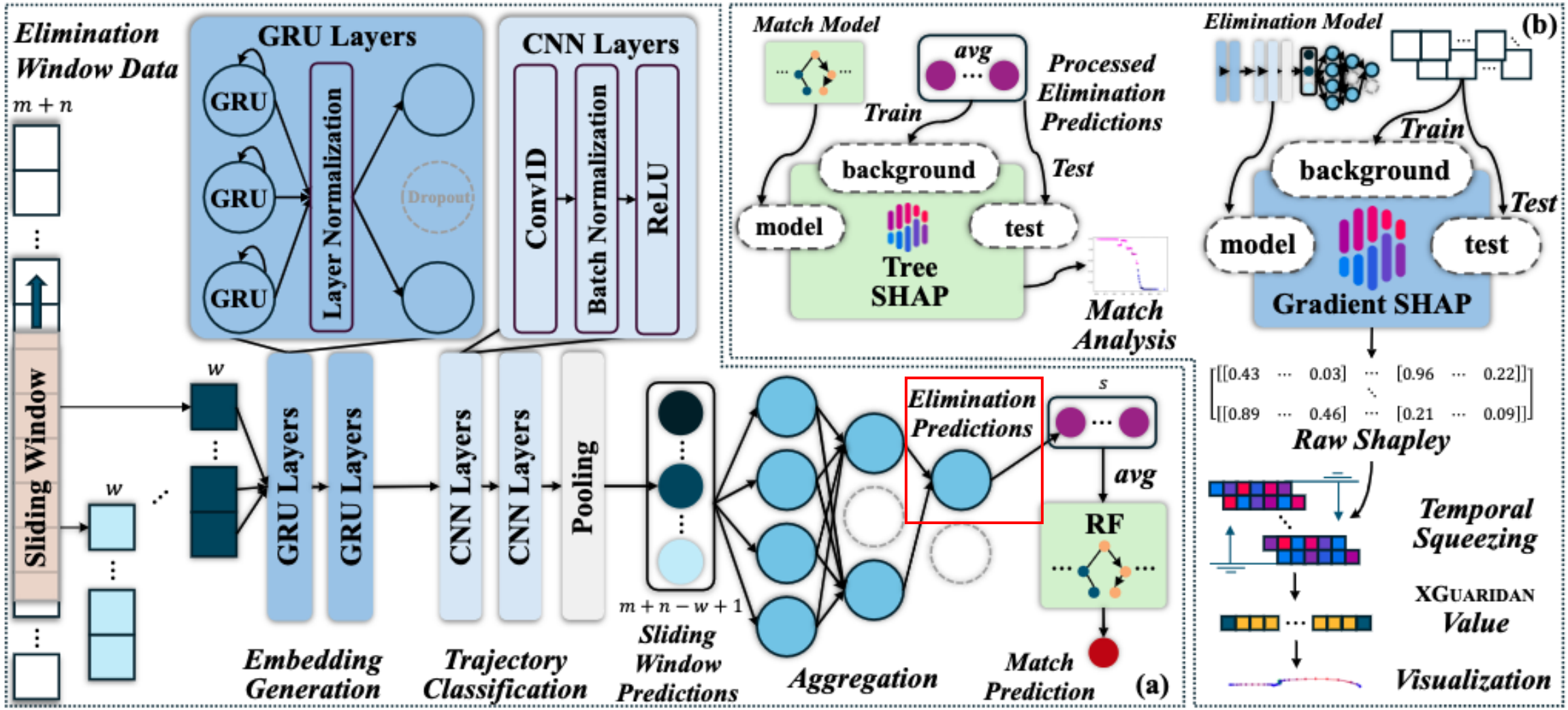


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

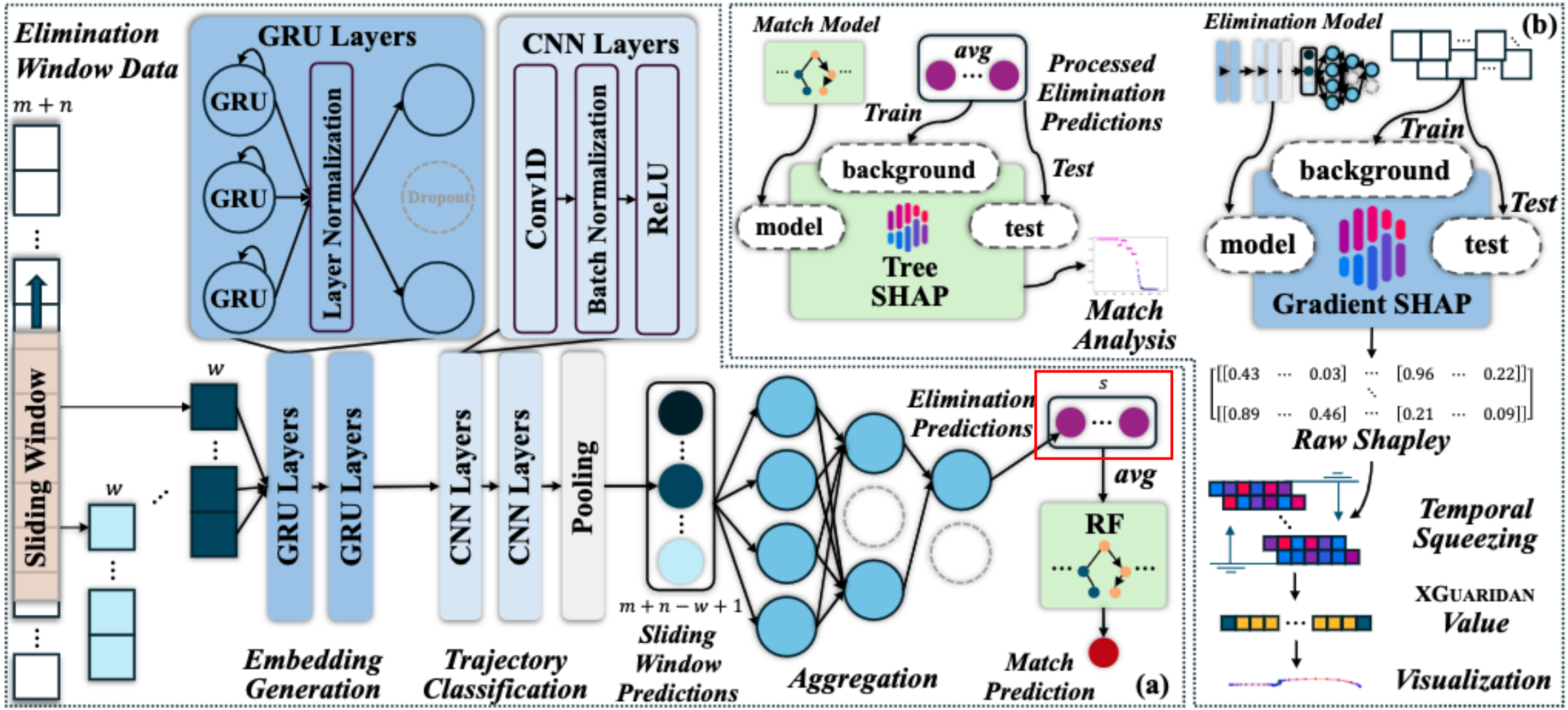


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

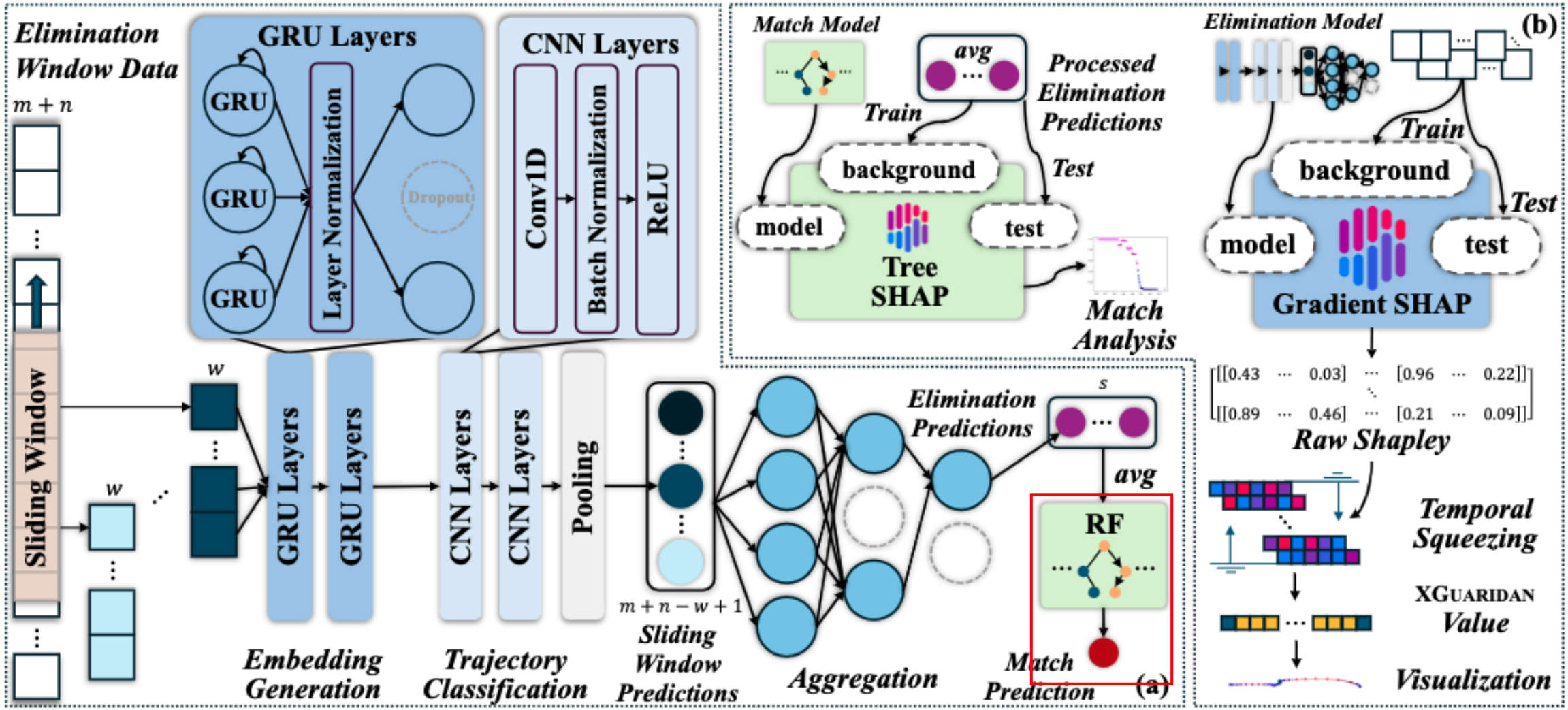


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

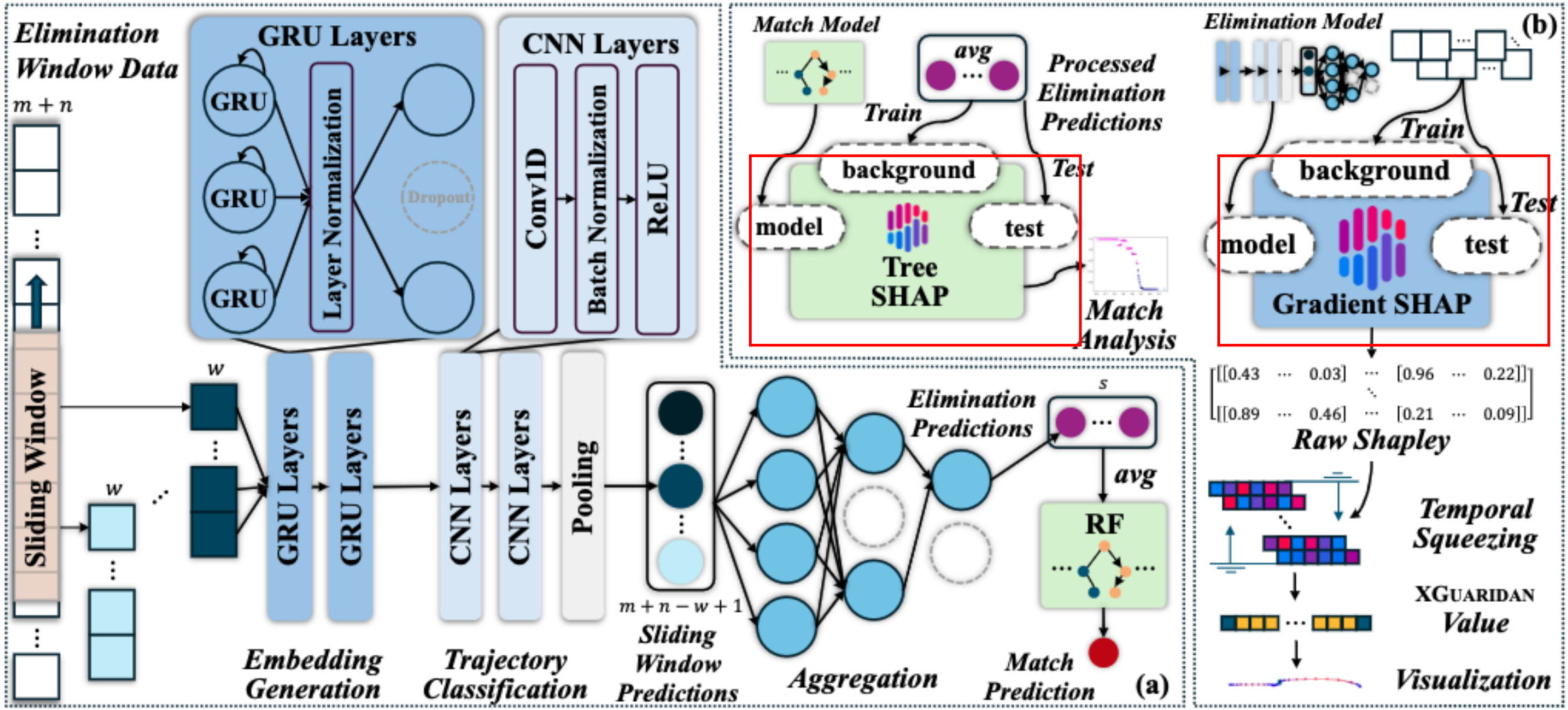


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

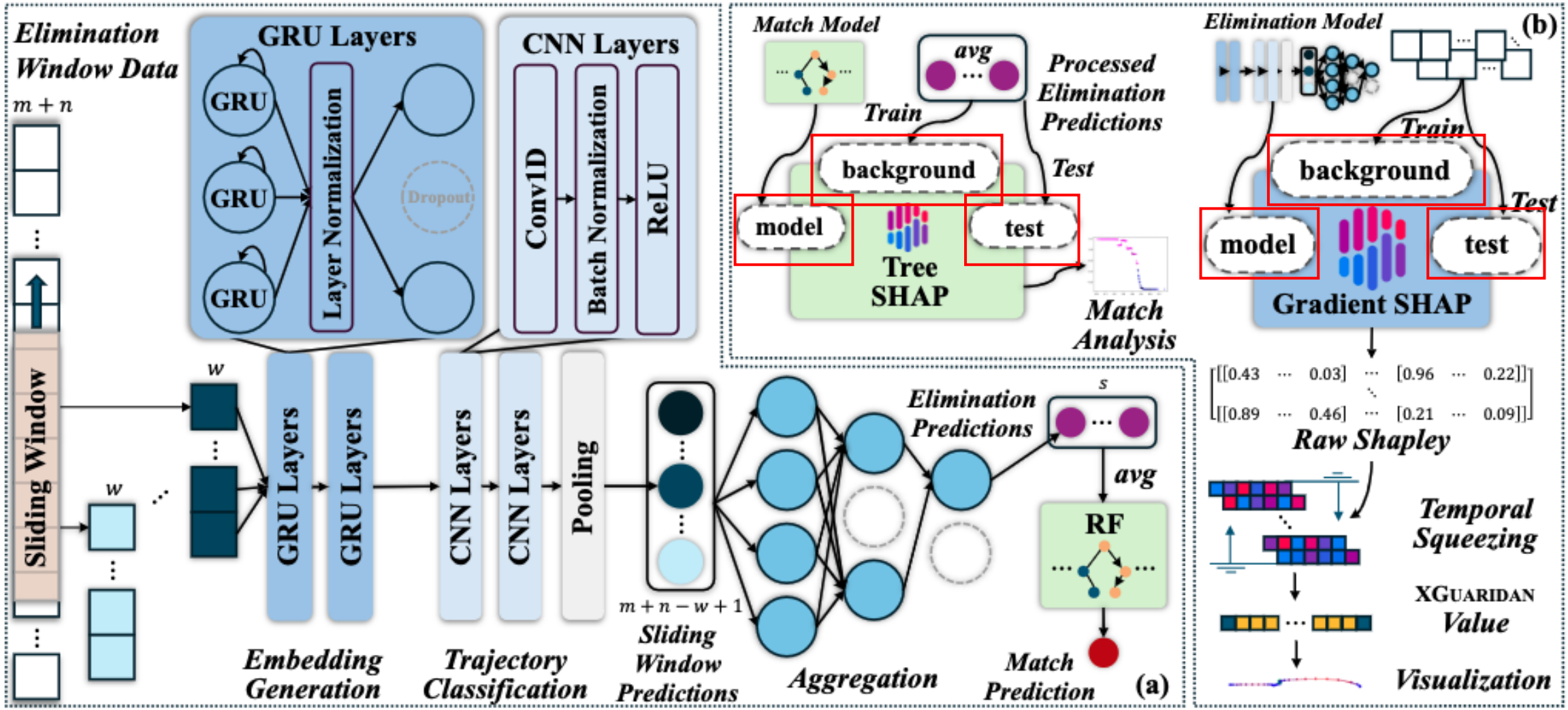


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

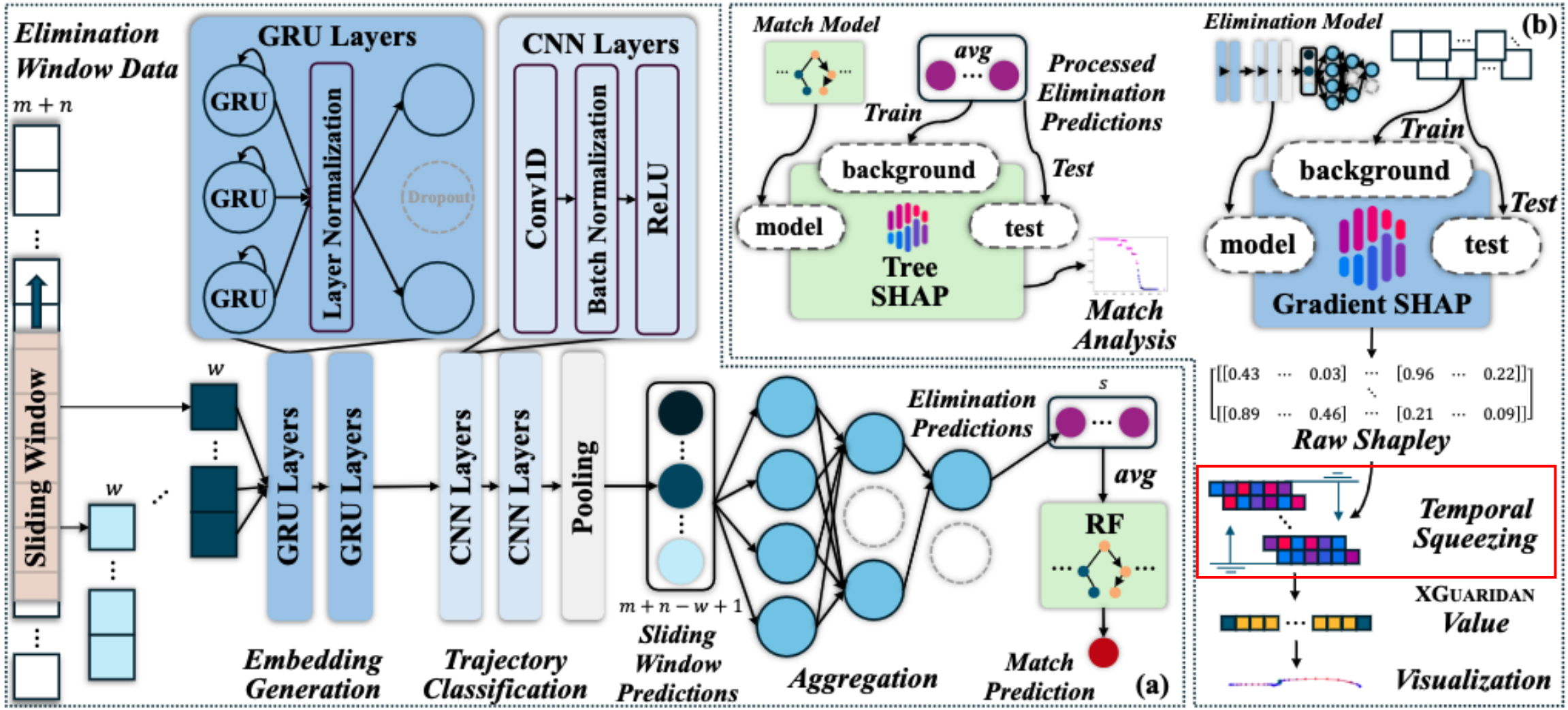


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

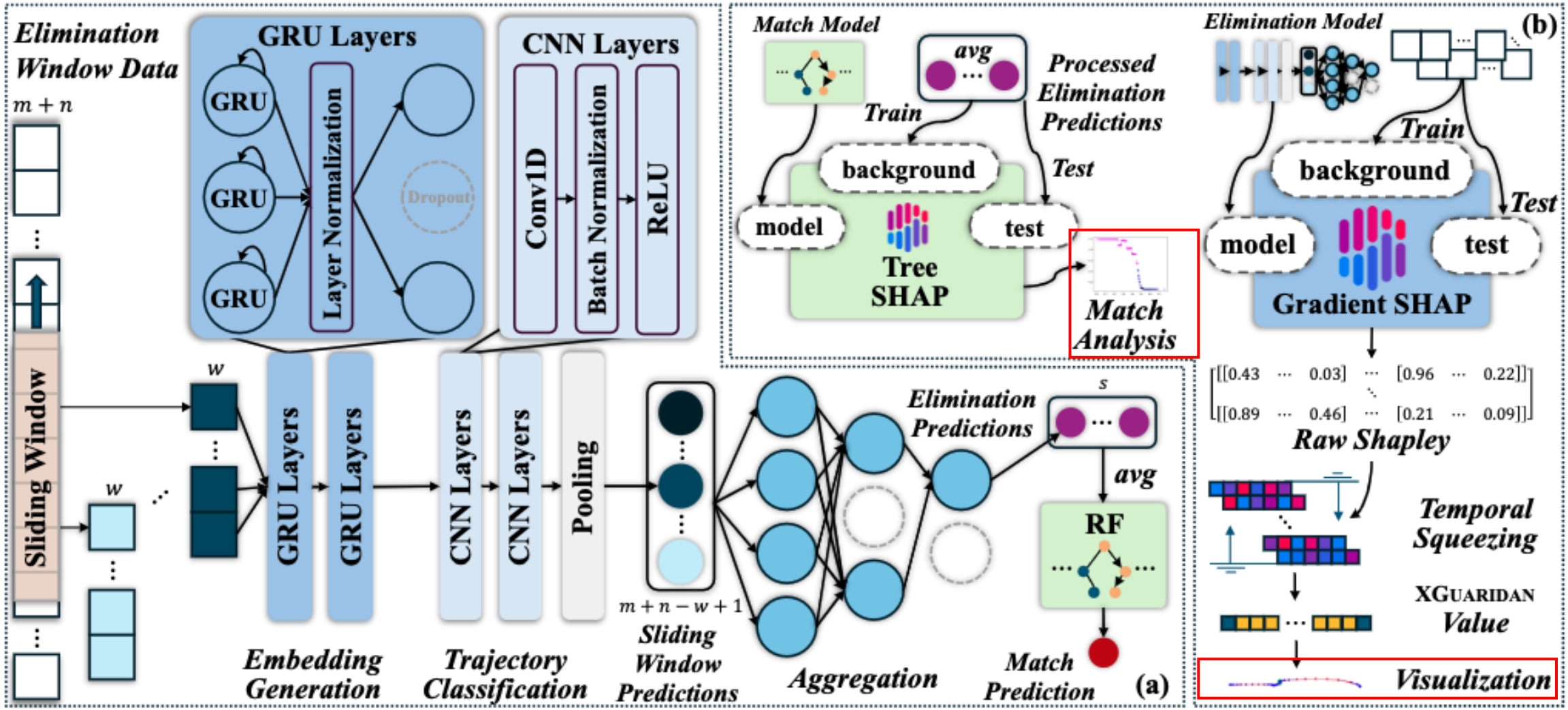


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Dataset

Data is from *5EPlay*, a world-class *CS2* platform.

Table 2: Dataset description.

Data Split	A	B	C
#Tick	992,064	1,014,624	1,062,528
#Total Elimination Window	10,334	10,569	11,068
#Cheating Elimination Window	2,385	2,760	2,900
#Normal Elimination Window	7,949	7,809	8,168
#Match	971	944	988
#Cheater	108	115	127
#Normal Player	1,711	1,671	1,754
#Total Player	1,819	1,786	1,881

Detection Performance

Table 3: XGUARDIAN weighted average overall performance with different set combinations. **Blue**, **green**, **and red** in the combination rows indicate the use of the set as training, validation, and test set, respectively.

		Accuracy(↑)	FPR(↓)	Recall(↑)	Precision(↑)	F1-Score(↑)
Combination	ABC	0.889	0.062	0.889	0.938	0.907
	ACB	0.876	0.049	0.876	0.951	0.901
	BAC	0.882	0.101	0.882	0.899	0.890
	BCA	0.907	0.041	0.907	0.959	0.924
	CAB	0.872	0.050	0.872	0.950	0.898
	CBA	0.879	0.048	0.879	0.952	0.904
mean (μ)		0.884	0.059	0.884	0.942	0.904
var (σ^2)		1.32E-04	4.00E-04	1.32E-04	4.00E-04	1.08E-04
std (σ)		1.15E-02	2.00E-02	1.15E-02	2.00E-02	1.04E-02

Detection Performance

Table 3: XGUARDIAN weighted average overall performance with different set combinations. **Blue**, **green**, **and red** in the combination rows indicate the use of the set as training, validation, and test set, respectively.

		Accuracy(↑)	FPR(↓)	Recall(↑)	Precision(↑)	F1-Score(↑)
Combination	ABC	0.889	0.062	0.889	0.938	0.907
	ACB	0.876	0.049	0.876	0.951	0.901
	BAC	0.882	0.101	0.882	0.899	0.890
	BCA	0.907	0.041	0.907	0.959	0.924
	CAB	0.872	0.050	0.872	0.950	0.898
	CBA	0.879	0.048	0.879	0.952	0.904
mean (μ)		0.884	0.059	0.884	0.942	0.904
var (σ^2)		1.32E-04	4.00E-04	1.32E-04	4.00E-04	1.08E-04
std (σ)		1.15E-02	2.00E-02	1.15E-02	2.00E-02	1.04E-02

Detection Performance

Table 3: XGUARDIAN weighted average overall performance with different set combinations. **Blue**, **green**, **and red** in the combination rows indicate the use of the set as training, validation, and test set, respectively.

		Accuracy(↑)	FPR(↓)	Recall(↑)	Precision(↑)	F1-Score(↑)
Combination	ABC	0.889	0.062	0.889	0.938	0.907
	ACB	0.876	0.049	0.876	0.951	0.901
	BAC	0.882	0.101	0.882	0.899	0.890
	BCA	0.907	0.041	0.907	0.959	0.924
	CAB	0.872	0.050	0.872	0.950	0.898
	CBA	0.879	0.048	0.879	0.952	0.904
mean (μ)		0.884	0.059	0.884	0.942	0.904
var (σ^2)		1.32E-04	4.00E-04	1.32E-04	4.00E-04	1.08E-04
std (σ)		1.15E-02	2.00E-02	1.15E-02	2.00E-02	1.04E-02

Feature ablation study

$$[t, I_f, I_e, \overrightarrow{v_x}, \overrightarrow{v_y}, \overrightarrow{a_x}, \overrightarrow{a_y}, \theta]$$

Table 4: Feature ablation study weighted average results.

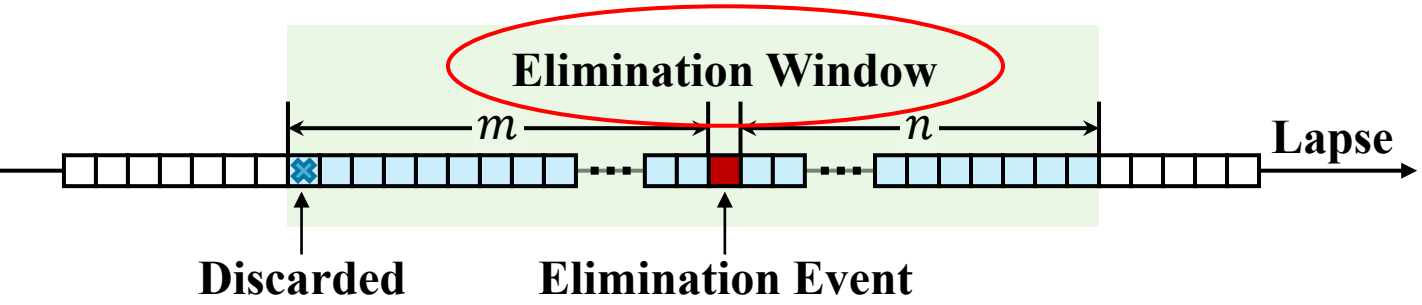
Feature			Metrics				
$\vec{a}_x$,$\vec{a}_y$	$\vec{v}_x$,$\vec{v}_y$	θ	**Accuracy**	**FPR**	**Recall**	**Precision**	**F1-Score**
✓	✗	✗	0.838	0.059	0.838	0.941	0.873
✗	✓	✗	0.884	0.060	0.884	0.940	0.904
✗	✗	✓	0.877	0.057	0.877	0.943	0.900
✓	✓	✗	0.886	0.062	0.886	0.938	0.905
✓	✗	✓	0.880	0.058	0.880	0.942	0.902
✗	✓	✓	0.901	0.063	0.901	0.937	0.915
✓	✓	✓	0.890	0.060	0.890	0.940	0.908

Table 5: XGUARDIAN input feature *tick* indexing comparative analysis weighted average results. 'wo/' and 'w/' denote without and with. 'Reset index' indicates re-allocating each elimination window's tick index from 0 to $m+n-1$.

Tick Indexing	Accuracy	FPR	Recall	Precision	F1-Score
wo/ tick index	0.801	0.078	0.801	0.922	0.846
w/ reset index	0.784	0.073	0.784	0.927	0.834
XGUARDIAN (w/ raw index)	**0.889**	**0.062**	**0.889**	**0.938**	**0.907**

Inspector model and parameters

- Different elimination window sizes
- Different sliding window sizes



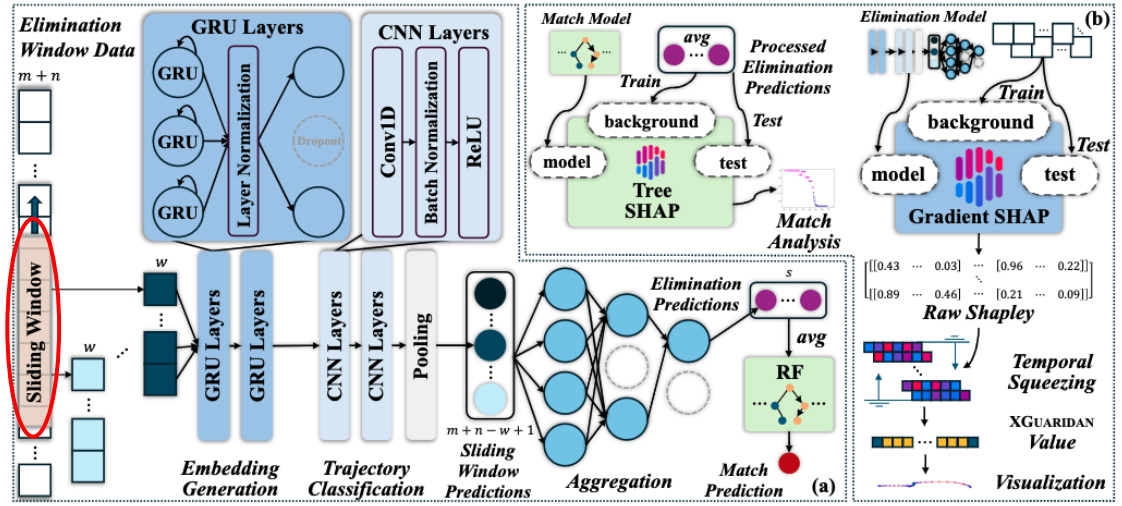
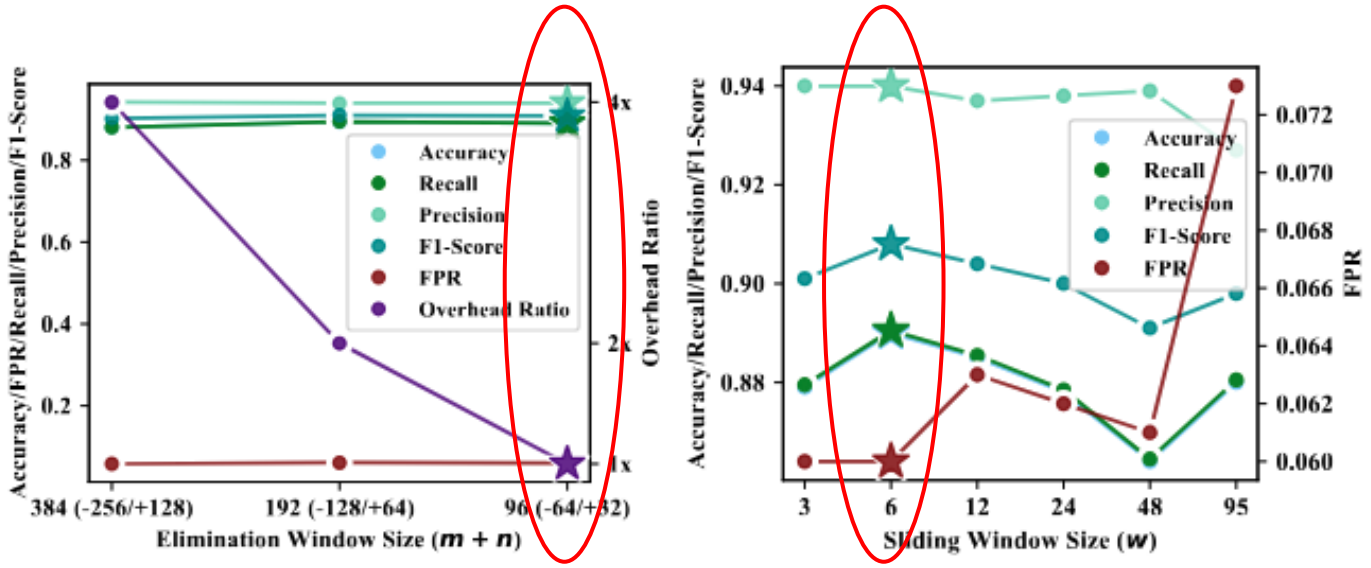


Figure 5: XGuardian framework. The subfigures illustrate the architectures of (a)-Inspector and (b)-Explainer.

Inspector model and parameters



(a) Elimination window (b) Sliding window

Figure 6: INSPECTOR sliding window and elimination window size comparative analysis weighted average results. The star (★) indicates the parameters used by XGUARDIAN.

Inspector model and parameters

- Different models in Embedding Generation
- Different models in Trajectory Classification

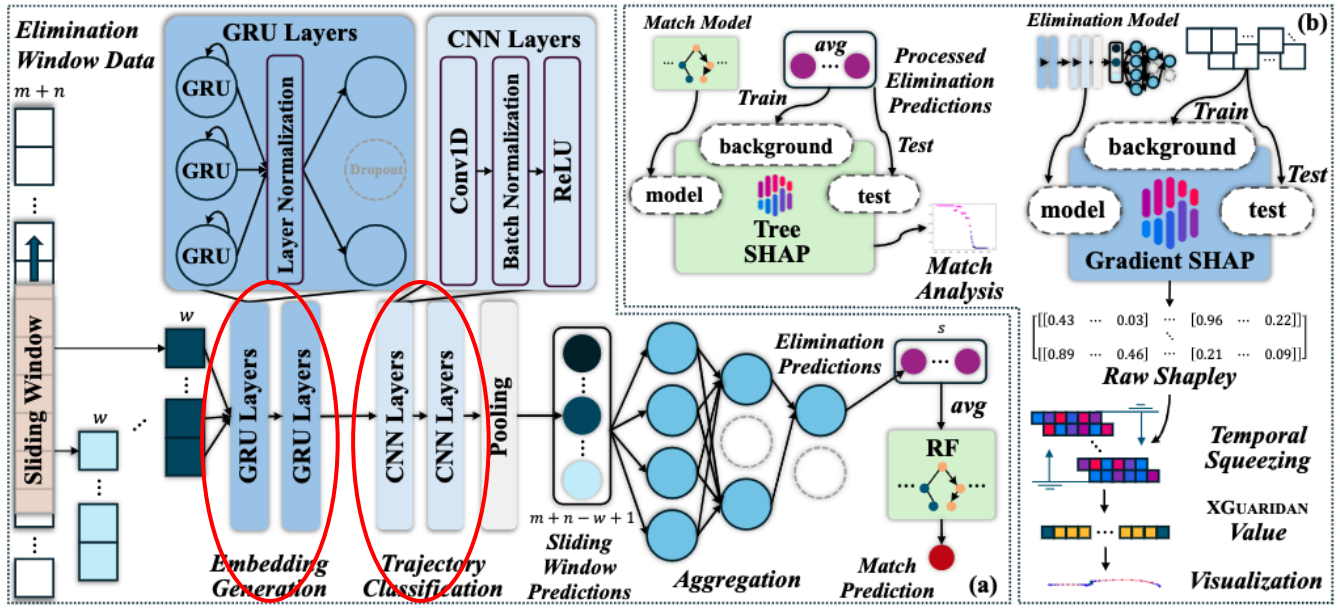
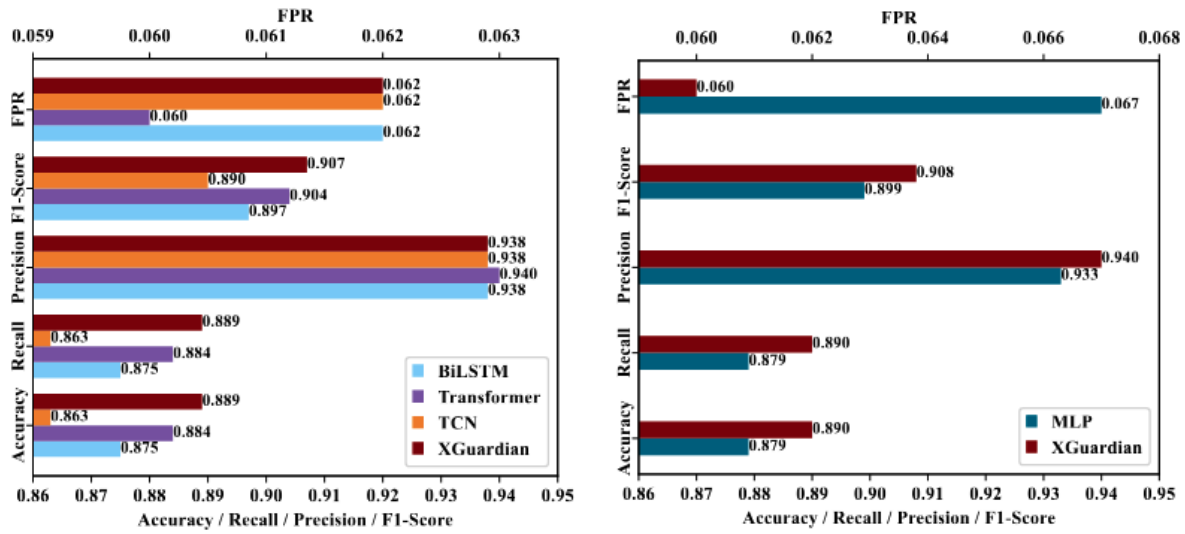


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Inspector model and parameters



(a) Embedding generation stage (b) Trajectory classification stage

Figure 7: INSPECTOR embedding generation and trajectory classification stages model selection weighted average results across SOTA models [30–33].

Inspector model and parameters

- Different input styles in Elimination Predictions

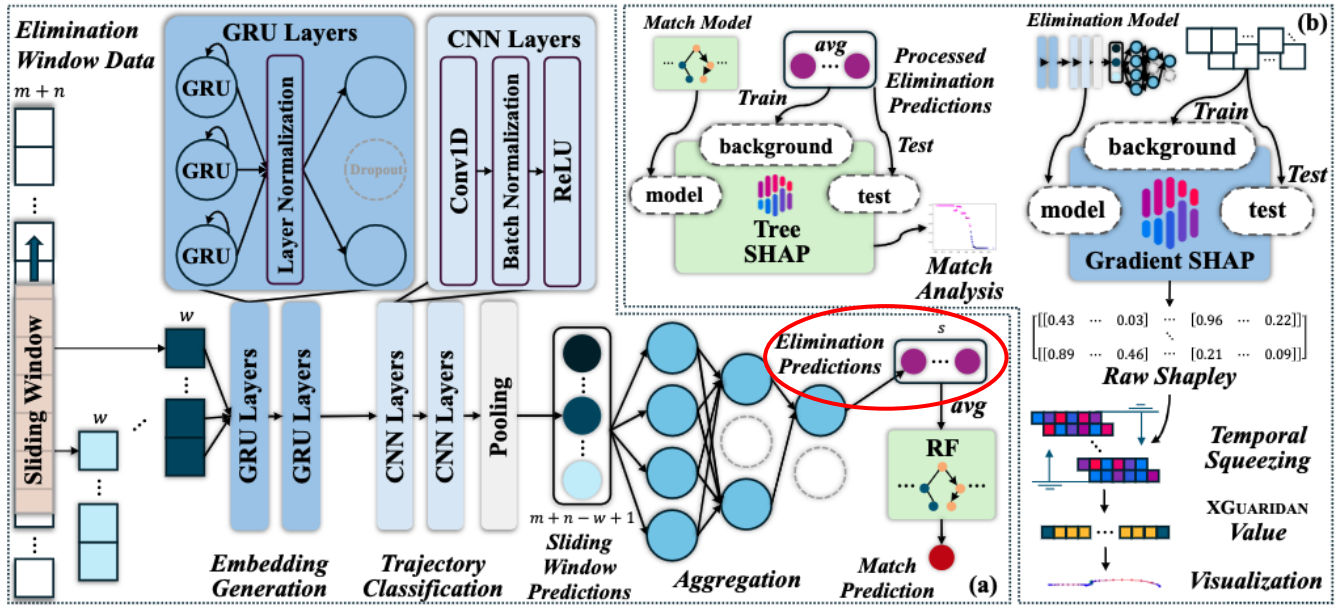


Figure 5: XGuardian framework. The subfigures illustrate the architectures of (a)-Inspector and (b)-Explainer.

Inspector model and parameters

Table 6: INSPECTOR elimination predictions stage embedding style comparison. 'Binary' denotes embedding binary classified predictions as match prediction's input. 'Original' denotes embedding raw predictions as match prediction input. XGUARDIAN's configuration is bolded.

Input Style	Accuracy	FPR	Recall	Precision	F1-Score
Binary	0.881	0.061	0.881	0.939	0.902
Binary+Original	0.881	0.061	0.881	0.939	0.902
XGUARDIAN (Original)	**0.889**	**0.062**	**0.889**	**0.938**	**0.907**

Comparative Study

Table 7: XGUARDIAN comparative study to prior works using different data sources. The optimal results are bolded.

Dataset	Method	Accuracy	FPR	Recall	Precision	F1-Score
XGUARDIAN	TH_ACCA [7]	0.738	0.455	0.738	0.545	0.627
	TH_HITACC [10]	0.644	0.086	0.644	0.914	0.711
	XGUARDIAN	**0.889**	**0.062**	**0.889**	**0.938**	**0.907**
HAWK [4]	REVPOV [4]	0.616	0.151	0.616	0.849	0.694
	REVSTATS [4]	0.814	0.109	0.814	0.891	0.842
	EXSPC [4]	0.793	0.114	0.793	0.886	0.827
	HAWK [4]	0.716	0.112	0.716	0.888	0.772
	XGUARDIAN	**0.841**	**0.056**	**0.841**	**0.944**	**0.878**

Comparative Study

Table 7: XGUARDIAN comparative study to prior works using different data sources. The optimal results are bolded.

Dataset	Method	Accuracy	FPR	Recall	Precision	F1-Score
XGUARDIAN	TH_ACCA [7]	0.738	0.455	0.738	0.545	0.627
	TH_HITACC [10]	0.644	0.086	0.644	0.914	0.711
	XGUARDIAN	**0.889**	**0.062**	**0.889**	**0.938**	**0.907**
HAWK [4]	REVPOV [4]	0.616	0.151	0.616	0.849	0.694
	REVSTATS [4]	0.814	0.109	0.814	0.891	0.842
	EXSPC [4]	0.793	0.114	0.793	0.886	0.827
	HAWK [4]	0.716	0.112	0.716	0.888	0.772
	XGUARDIAN	**0.841**	**0.056**	**0.841**	**0.944**	**0.878**

Generalizability Study

Table 8: XGUARDIAN's generalizability evaluation results on different sub-genre games, platforms, and engines. The descriptions of the other two datasets are shown in Appendix F.

Dataset	HAWK [4]	FARLIGHT84	XGUARDIAN
Game (Release Year)	CS:GO (2012)	Farlight84 (2023)	CS2 (2023)
Sub-genre	Tactical Shooter	Battle Royal	Tactical Shooter
Engine	Source	Unreal Engine 4	Source 2
Platform	PC	Mobile	PC
Accuracy	0.841	0.816	0.889
FPR	0.056	0.144	0.062
Recall	0.841	0.816	0.889
Precision	0.944	0.856	0.938
F1-Score	0.878	0.829	0.907

Explainability Case Study

https://xguardian-anti-cheat.github.io/

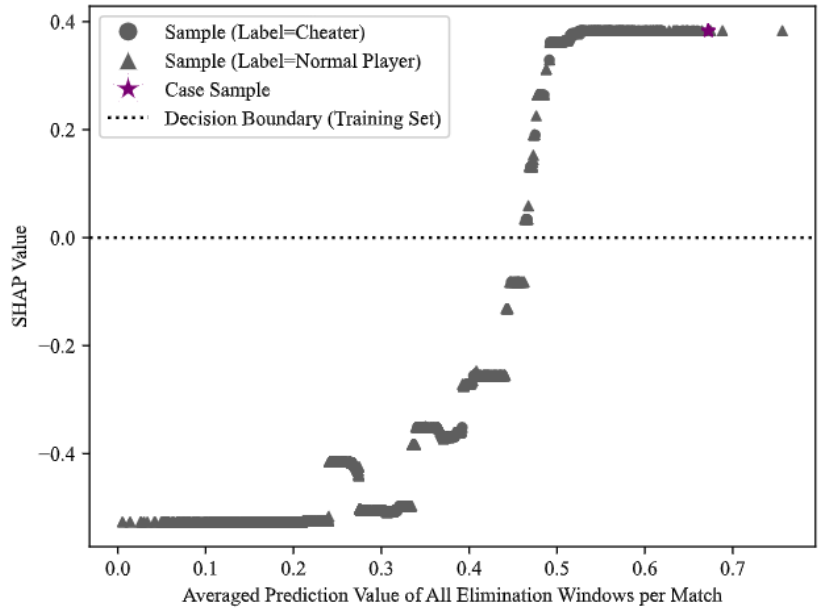


Figure 10: Explainable match-scope visualizations of an identified aim-assist cheater.

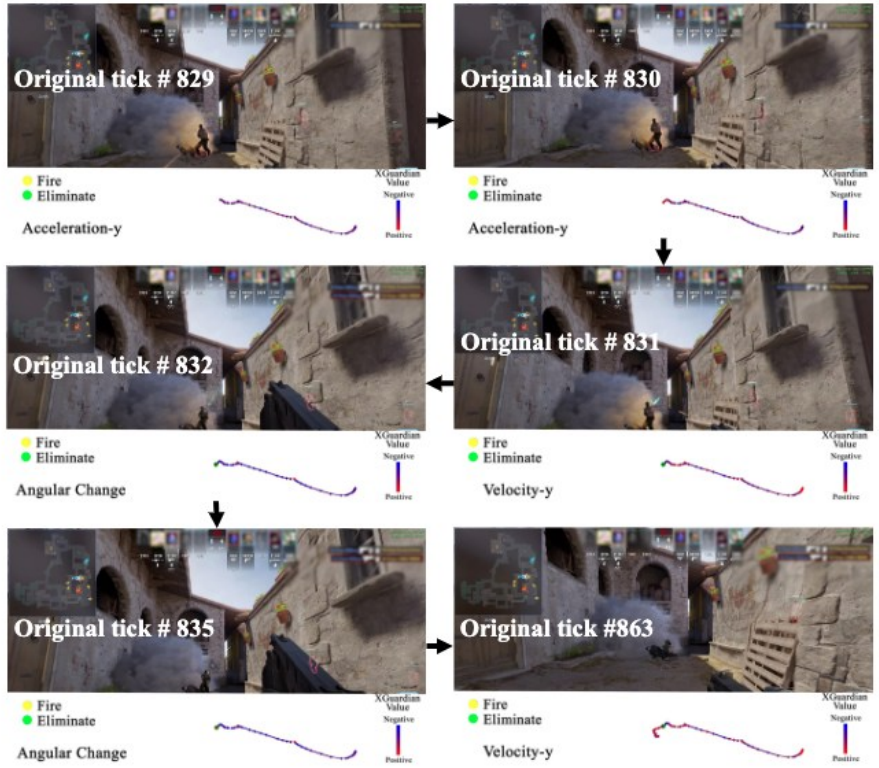


Figure 9: Explainable elimination-scope visualizations of an identified aim-assist cheater.

Robustness Under Adversarial Attack

- ***18/22*** - Detected / All adversarial samples
  - achieves ***0.818*** *recall* and ***0.900*** *F1-Score*
- *Two-proportion Z-test* and *Fisher's exact test* indicate that the observed decrease in recall under adversarial attack is statistically **insignificant** ($p > 0.05$).

Overhead

- Only takes less than ***9.98*** seconds of prediction time per match with CPU-only
- A remarkable improvement of ***4,104.31%*** compared to the SOTA server-side solution HAWK, which requires over *400* seconds

Table 9: XGUARDIAN's overheads tested with the entire dataset. The numbers are retrieved by the Python libraries time [39] and resource [40].

Overheads	Preprocessing	Prediction	Training
User time (sec)	52.4	28966.87	28960.42
System time (sec)	30.21	2905.86	2905.14
Elapsed time (sec)	48.03	3.95	11423.11
Max. resident set size (GB)	3.226	8.596	8.596

EXPLAINER's Potential Bias Investigation

Table 11: User study results of the EXPLAINER's potential bias questionnaire.

Reviewer	Decision[1]						Confidence[2]						Time Consumption					
	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B
#1	1	1	1	1	1	1	5	5	5	4	5	5	3′36″	1′45″	4′14″	1′08″	2′54″	1′03″
#2	1	1	1	1	0	1	5	5	4	5	4	3	1′17″	0′40″	4′59″	2′31″	3′59″	2′43″
#3	1	1	1	1	1	1	5	5	4	5	4	3	0′37″	1′19″	4′27″	2′22″	7′45″	7′04″
#4	1	1	1	1	1	1	5	5	5	5	5	5	2′30″	0′14″	2′49″	0′34″	2′47″	1′49″

[1]**Decision**: 1 and 0 denote the review's decision is correct and incorrect, respectively; [2]**Confidence**: 1, 2, 3, 4, 5 represent reviewer's decision confidence from the least confident to the most confident; [3] *Grey background* denotes the current case is applying XGUARDIAN;

Table 12: User study reviewers' CS2 skill level, where *5EPlay* [1] and *Perfect World* [39] are the platform (introduced in Ethical Considerations) ranks, *Premier* is the official rank. All invited reviewers are the former reviewers of CS:GO Overwatch Investigators [53].

Reviewer	CS2 Skill Level	Playtime (hrs)
#1	*5EPlay* - **A**, *Perfect World* - **S**	3,563
#2	*5EPlay* - **A**, *Perfect World* - **A**, *Premier* - 18,491	2,020
#3	*5EPlay* - **A**, *Perfect World* - **S**, *Premier* - 19,486	2,400
#4	*5EPlay* - **A+**, *Perfect World* - **S**, *Premier* - 20,000	2,917

EXPLAINER's Potential Bias Investigation

- Reviewers with XGuardian have higher decision accuracy.

Table 11: User study results of the EXPLAINER's potential bias questionnaire.

Reviewer	Decision[1]						Confidence[2]						Time Consumption					
	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B
#1	1	1	1	1	1	1	5	5	5	4	5	5	3′36″	1′45″	4′14″	1′08″	2′54″	1′03″
#2	1	1	1	1	0	1	5	5	4	5	4	3	1′17″	0′40″	4′59″	2′31″	3′59″	2′43″
#3	1	1	1	1	1	1	5	5	4	5	4	3	0′37″	1′19″	4′27″	2′22″	7′45″	7′04″
#4	1	1	1	1	1	1	5	5	5	5	5	5	2′30″	0′14″	2′49″	0′34″	2′47″	1′49″

[1]**Decision**: 1 and 0 denote the review's decision is correct and incorrect, respectively; [2]**Confidence**: 1, 2, 3, 4, 5 represent reviewer's decision confidence from the least confident to the most confident; [3] *Grey background* denotes the current case is applying XGUARDIAN;

EXPLAINER's Potential Bias Investigation

- XGuardian's false determination does not affect the correctness of reviewer's final decision.

Table 11: User study results of the EXPLAINER's potential bias questionnaire.

Reviewer	Decision[1]						Confidence[2]						Time Consumption					
	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B
#1	1	1	1	1	1	1	5	5	5	4	5	5	3′36″	1′45″	4′14″	1′08″	2′54″	1′03″
#2	1	1	1	1	0	1	5	5	4	5	4	3	1′17″	0′40″	4′59″	2′31″	3′59″	2′43″
#3	1	1	1	1	1	1	5	5	4	5	4	3	0′37″	1′19″	4′27″	2′22″	7′45″	7′04″
#4	1	1	1	1	1	1	5	5	5	5	5	5	2′30″	0′14″	2′49″	0′34″	2′47″	1′49″

[1]**Decision**: 1 and 0 denote the review's decision is correct and incorrect, respectively; [2]**Confidence**: 1, 2, 3, 4, 5 represent reviewer's decision confidence from the least confident to the most confident; [3] *Grey background* denotes the current case is applying XGUARDIAN;

EXPLAINER's Potential Bias Investigation

- XGuardian's false determination does not affect the correctness of reviewer's final decision.

Table 11: User study results of the EXPLAINER's potential bias questionnaire.

Reviewer	Decision[1]						Confidence[2]						Time Consumption					
	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B
#1	1	1	1	1	1	1	5	5	5	4	5	5	3′36″	1′45″	4′14″	1′08″	2′54″	1′03″
#2	1	1	1	1	0	1	5	5	4	5	4	3	1′17″	0′40″	4′59″	2′31″	3′59″	2′43″
#3	1	1	1	1	1	1	5	5	4	5	4	3	0′37″	1′19″	4′27″	2′22″	7′45″	7′04″
#4	1	1	1	1	1	1	5	5	5	5	5	5	2′30″	0′14″	2′49″	0′34″	2′47″	1′49″

[1]**Decision**: 1 and 0 denote the review's decision is correct and incorrect, respectively; [2]**Confidence**: 1, 2, 3, 4, 5 represent reviewer's decision confidence from the least confident to the most confident; [3] *Grey background* denotes the current case is applying XGUARDIAN;

EXPLAINER's Potential Bias Investigation

- XGuardian's false determination does not affect the correctness of reviewer's final decision.

Table 11: User study results of the EXPLAINER's potential bias questionnaire.

Reviewer	Decision[1]						Confidence[2]						Time Consumption					
	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B	1A	1B	2A	2B	3A	3B
#1	1	1	1	1	1	1	5	5	5	4	5	5	3′36″	1′45″	4′14″	1′08″	2′54″	1′03″
#2	1	1	1	1	0	1	5	5	4	5	4	3	1′17″	0′40″	4′59″	2′31″	3′59″	2′43″
#3	1	1	1	1	1	1	5	5	4	5	4	3	0′37″	1′19″	4′27″	2′22″	7′45″	7′04″
#4	1	1	1	1	1	1	5	5	5	5	5	5	2′30″	0′14″	2′49″	0′34″	2′47″	1′49″

[1]**Decision**: 1 and 0 denote the review's decision is correct and incorrect, respectively; [2]**Confidence**: 1, 2, 3, 4, 5 represent reviewer's decision confidence from the least confident to the most confident; [3] *Grey background* denotes the current case is applying XGUARDIAN;

Data collection generalizability

- Certain niche titles or those developed by smaller studios may lack replay system
- Developers can alternatively collect **pitch** and **yaw** data which are ALL FPS games must-haves

Undetectable cases

- Cannot detect players who do not secure kills
  - e.g., purely support-oriented heroes
- Cannot detect a cheater employs techniques that do not influence the aiming trajectory
  - e.g., illegally sharing information with teammates

XGuardian: Towards Generalized, Explainable and More Effective Server-side Anti-cheat in First-Person Shooter Games

Jiayi Zhang, Chenxin Sun, Chenxiong Qian
HKU

Links

Artifact

Paper

Demo