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

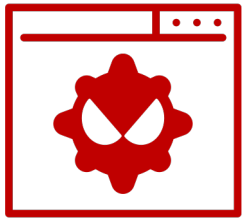
First-**P**erson **S**hooter (**FPS**) game holds 20.9% market share.



Video Source:
[1] Counter-Strike 2

FPS games are vulnerable to cheating

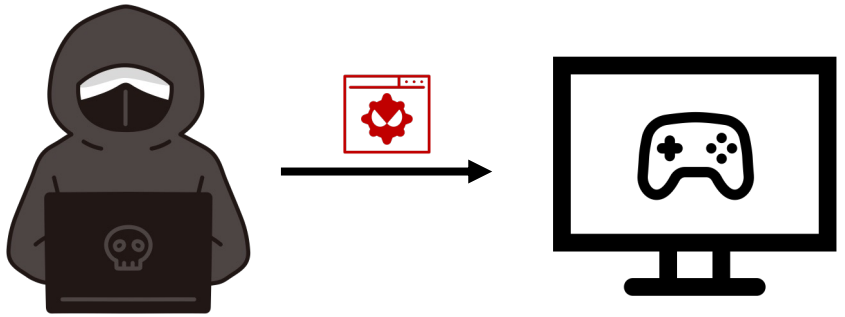
- Malicious software for games is called the **cheat**
  - Cheat developers seek for monetary gain



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



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:

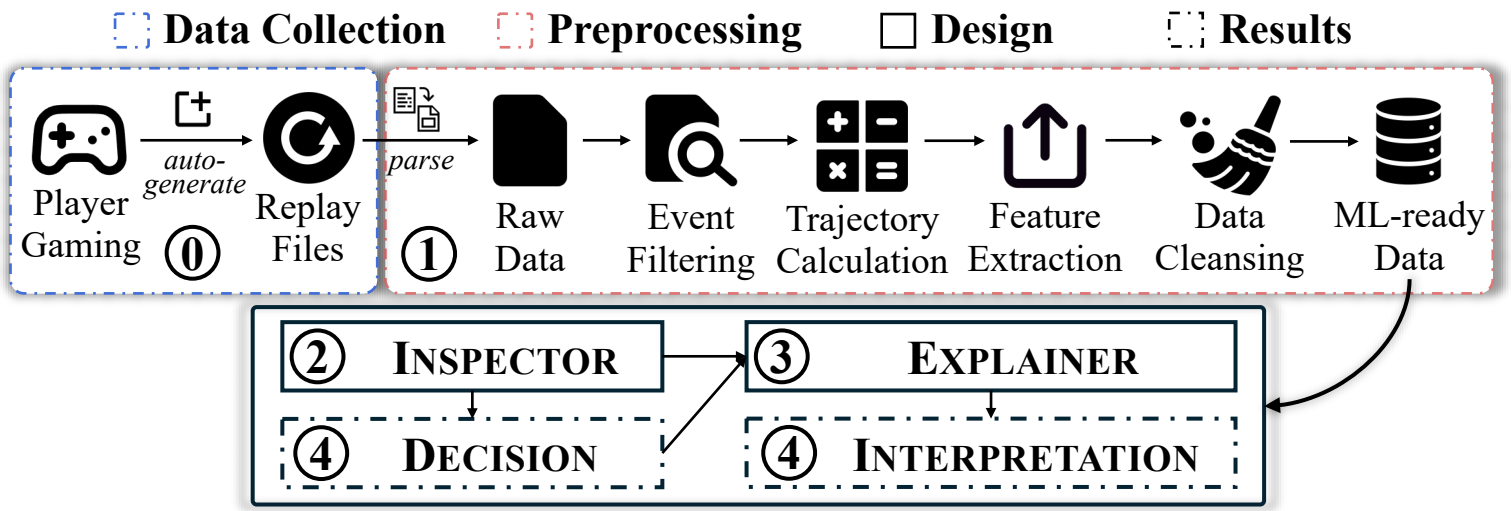
- Functional granularity (e.g., targeting specific body part *vs.* full-body locks)
- Stealth tradeoffs (e.g., blatant play *vs.* behavioral mimicry of legitimate players)

Threat Model

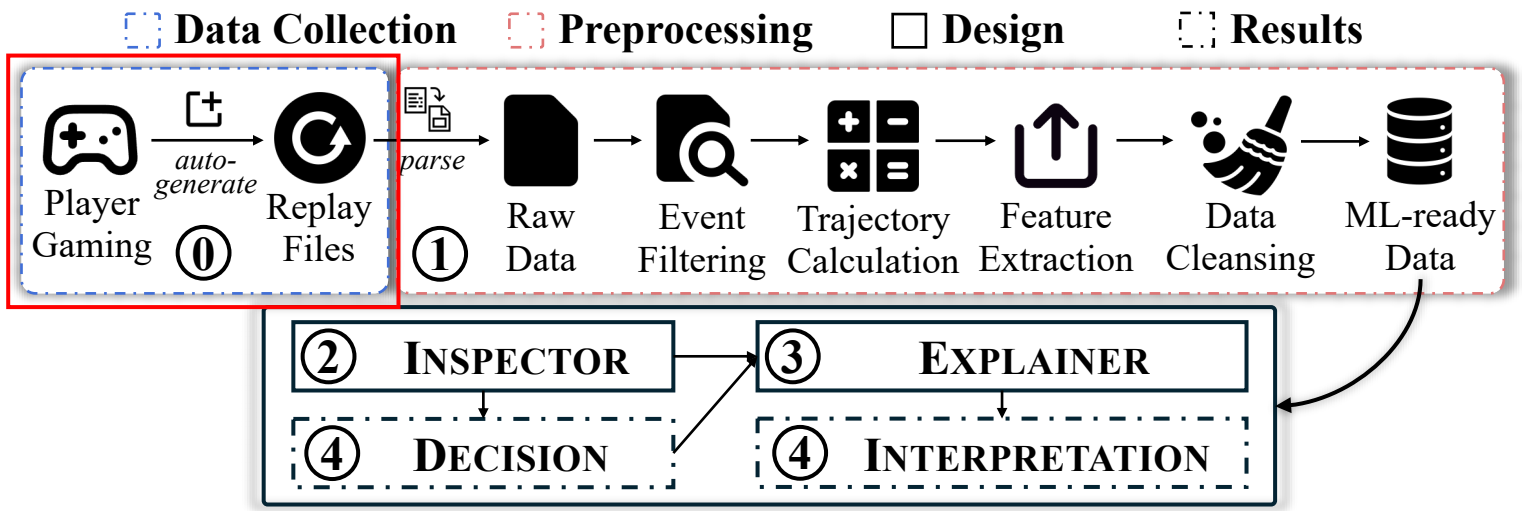
The cheaters

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

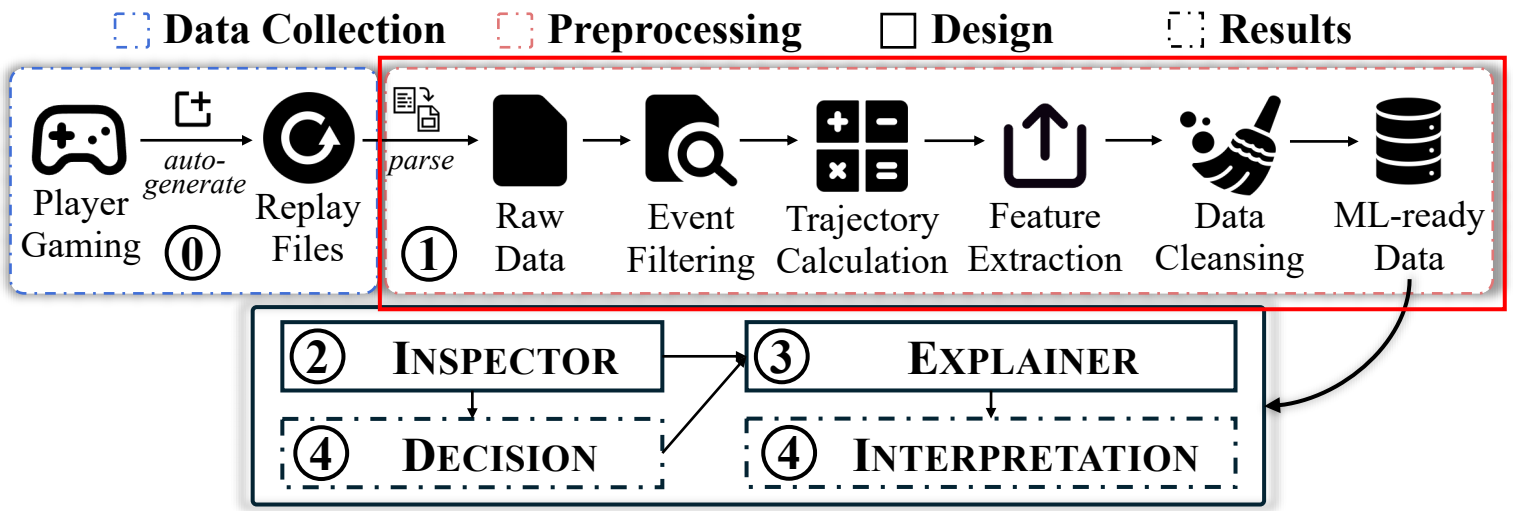
Workflow


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

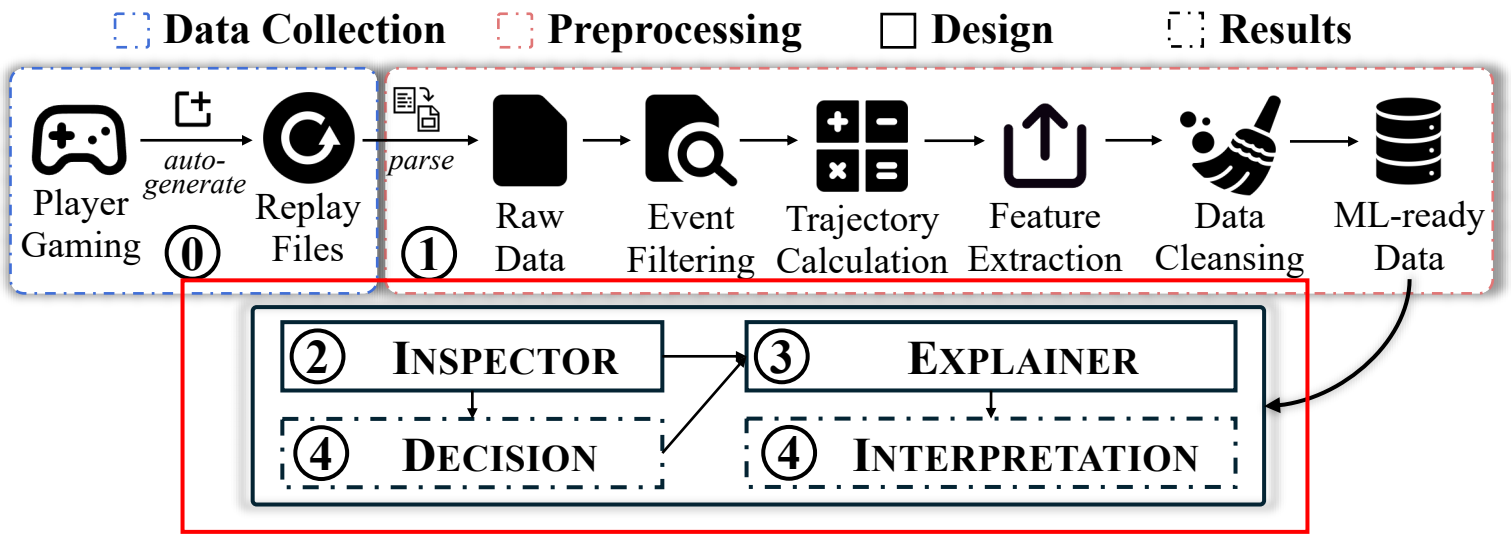
Workflow


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

Dimensions of movement in FPS games

Pitch and Yaw

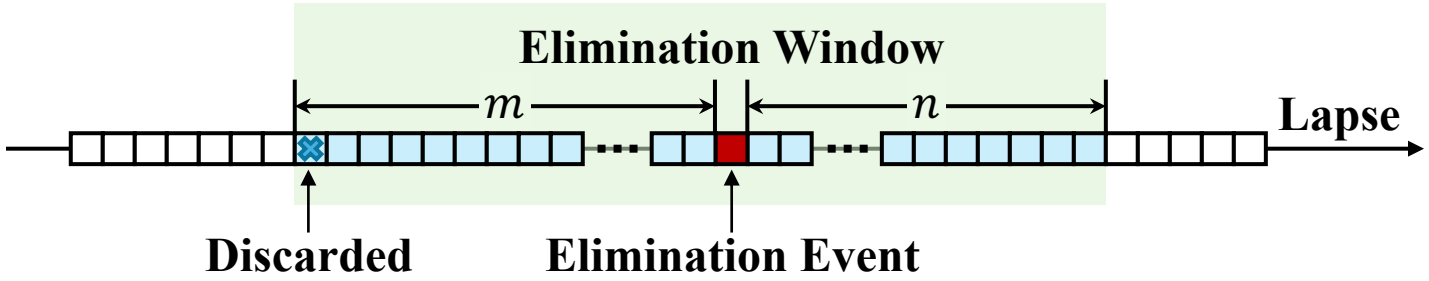
- **Pitch** is the vertical angle of the player's view.
- **Yaw** is the horizontal angle of the player's view.



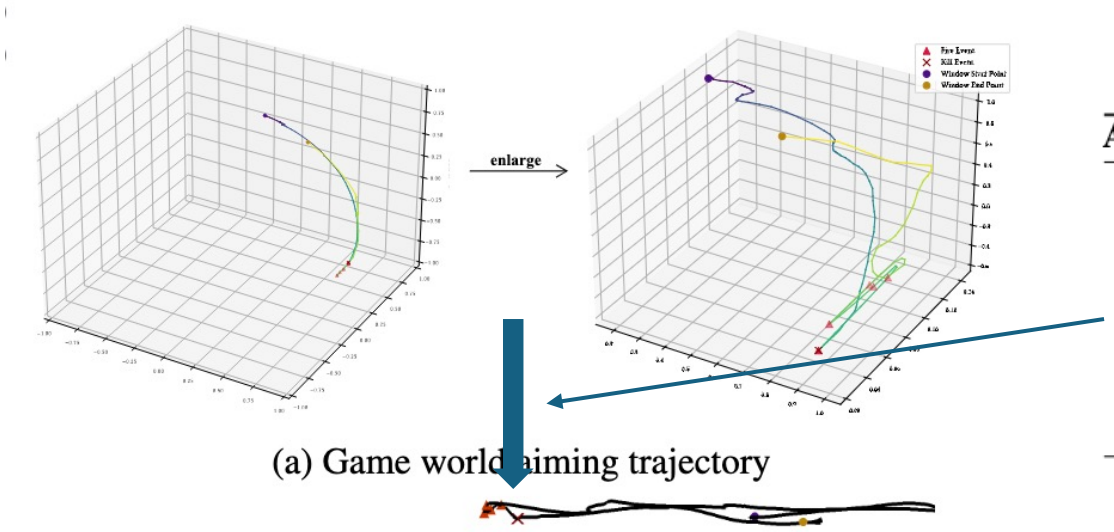
Extraction scope – per elimination

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

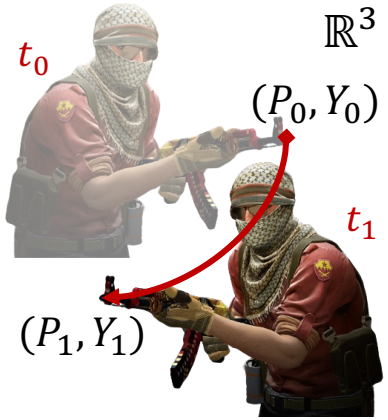
function PITCHYAWTOXY(pitch, yaw, width, height)
  $x \leftarrow \left(\frac{yaw+180}{360}\right) \times width$
  $y \leftarrow \left(\frac{pitch+90}{180}\right) \times height$
  $y \leftarrow height - y$
  return (x, y)
end function

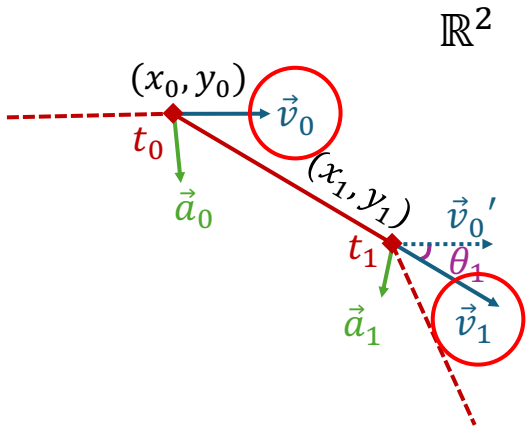
Feature construction

Velocity

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

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



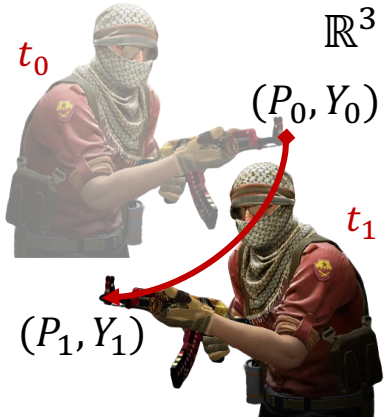


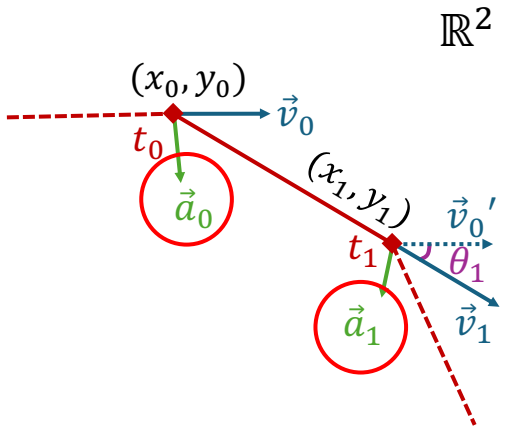
Feature construction

Acceleration

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

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



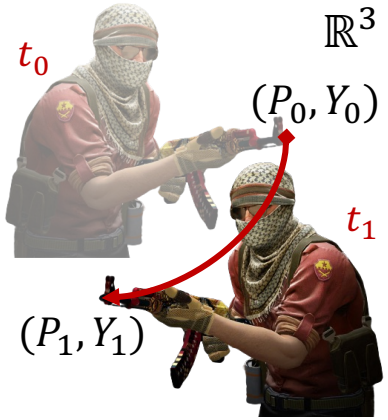


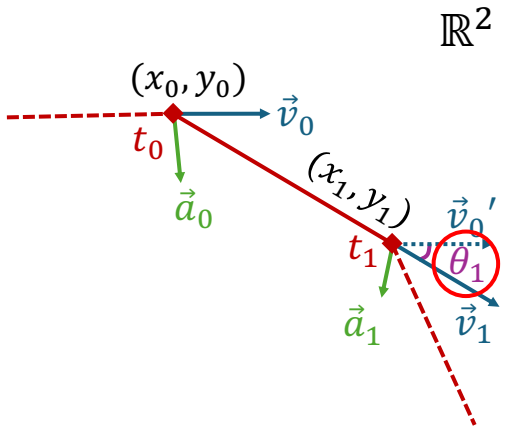
Feature construction

Angular Change

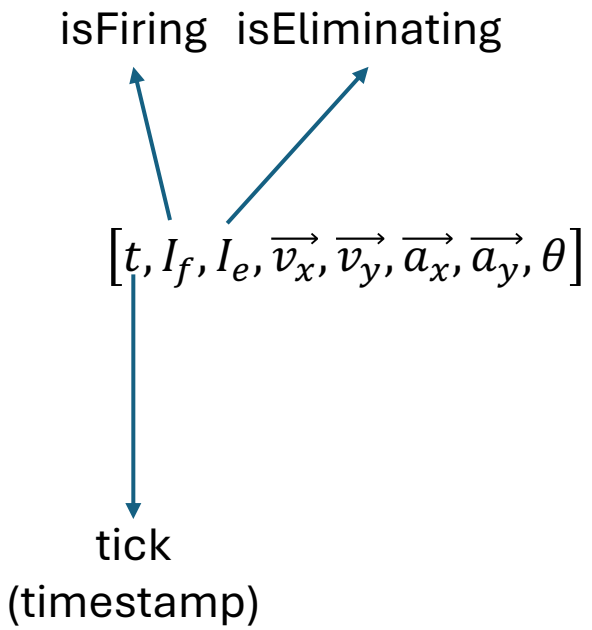
$$\alpha = \arctan 2(\vec{v}_y, \vec{v}_x) \quad (5)$$

$$\theta = \frac{d\alpha}{dt} \quad (6)$$





ML-ready data



Framework

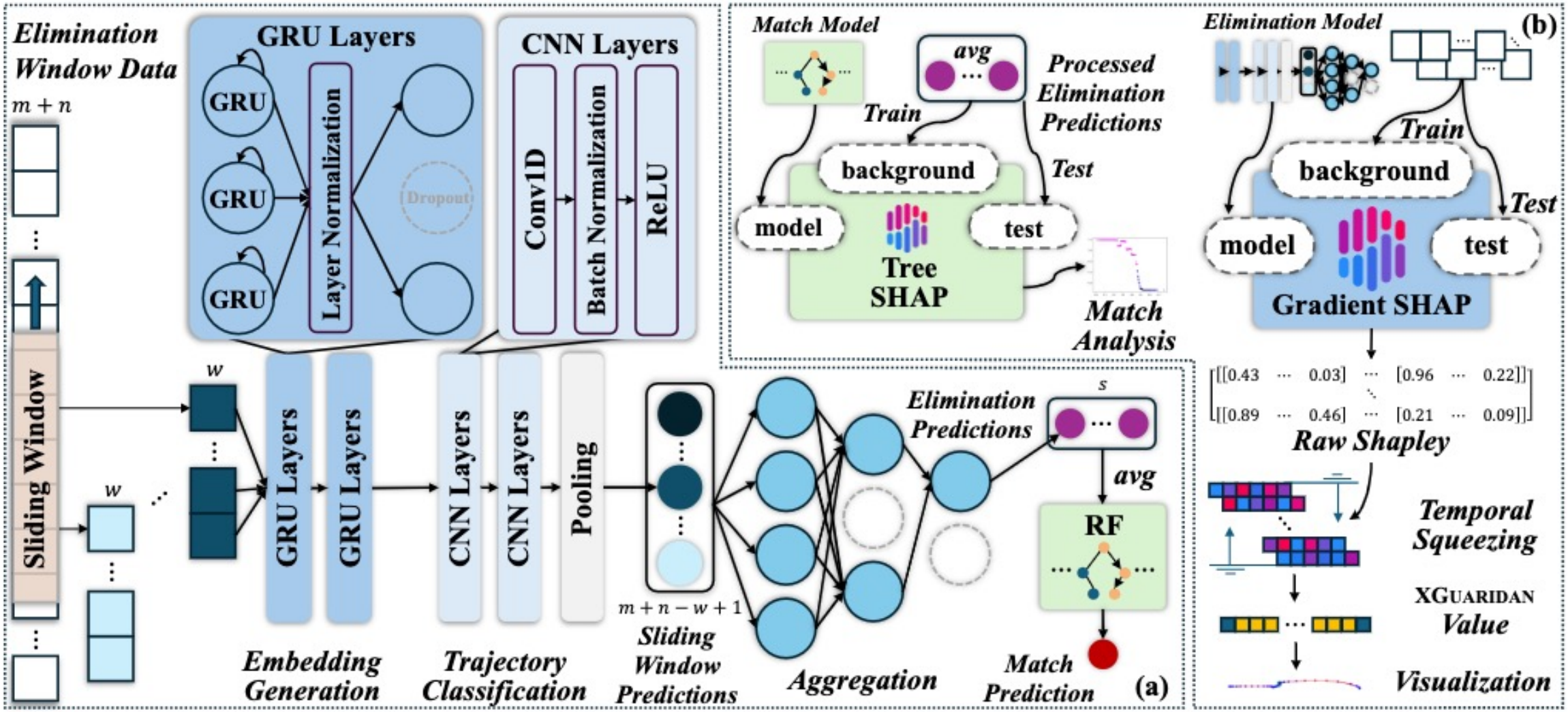


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

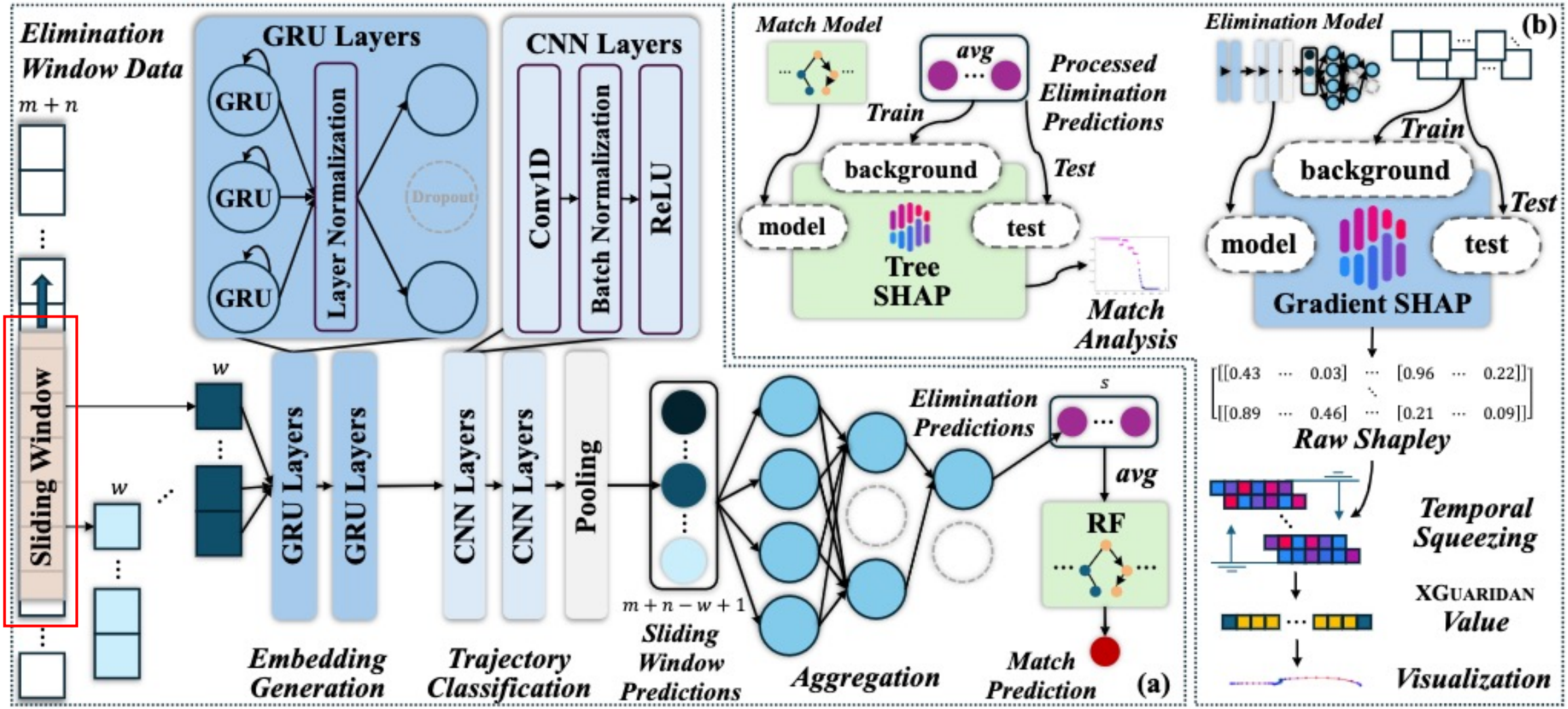


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

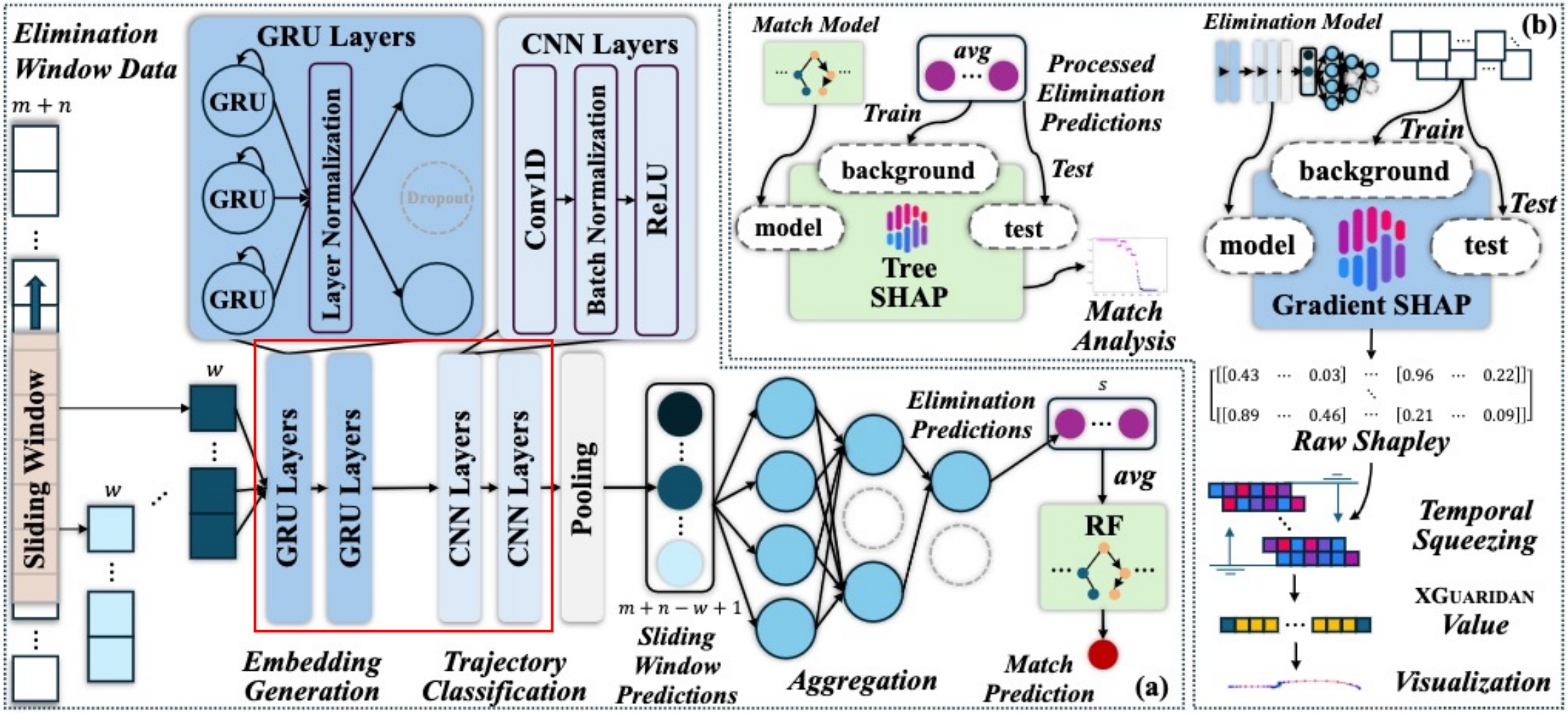


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

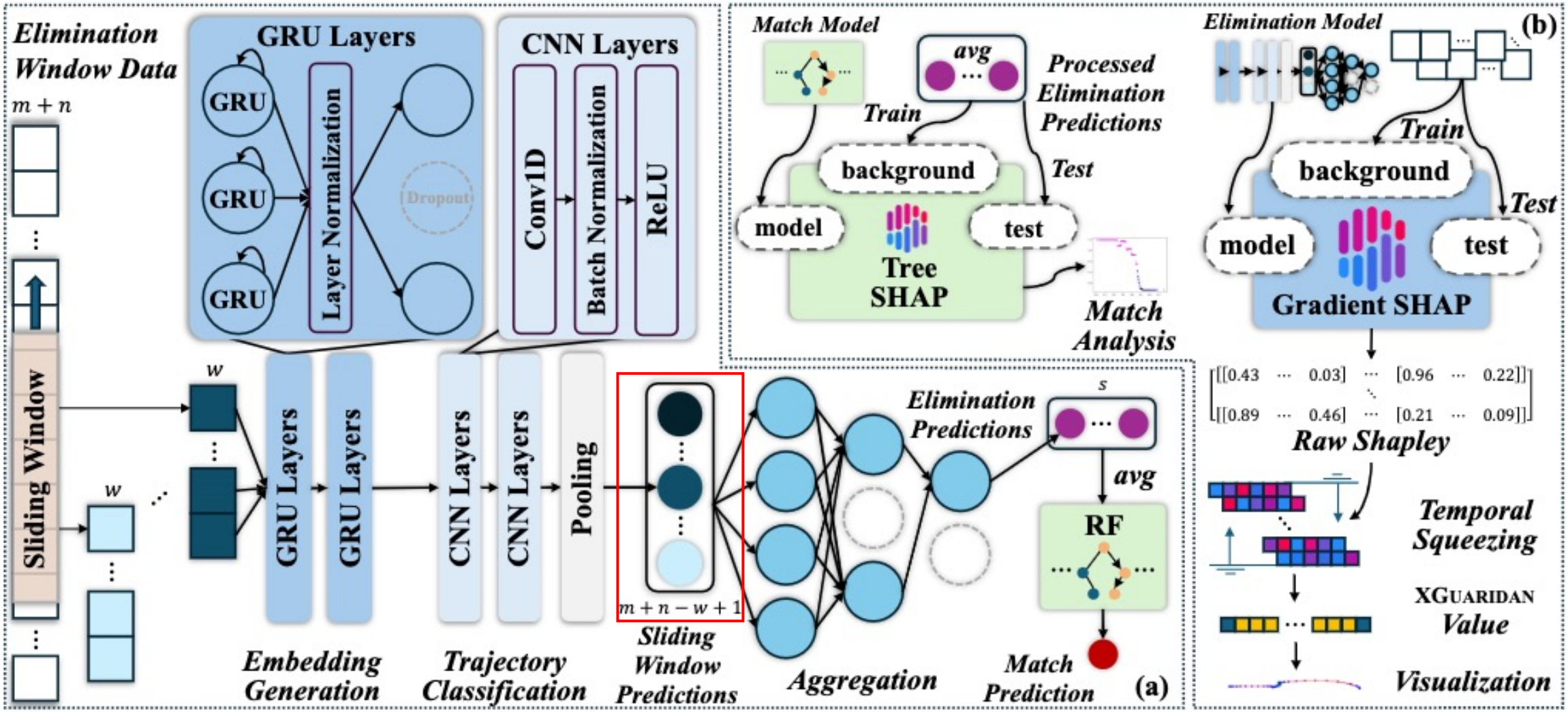


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

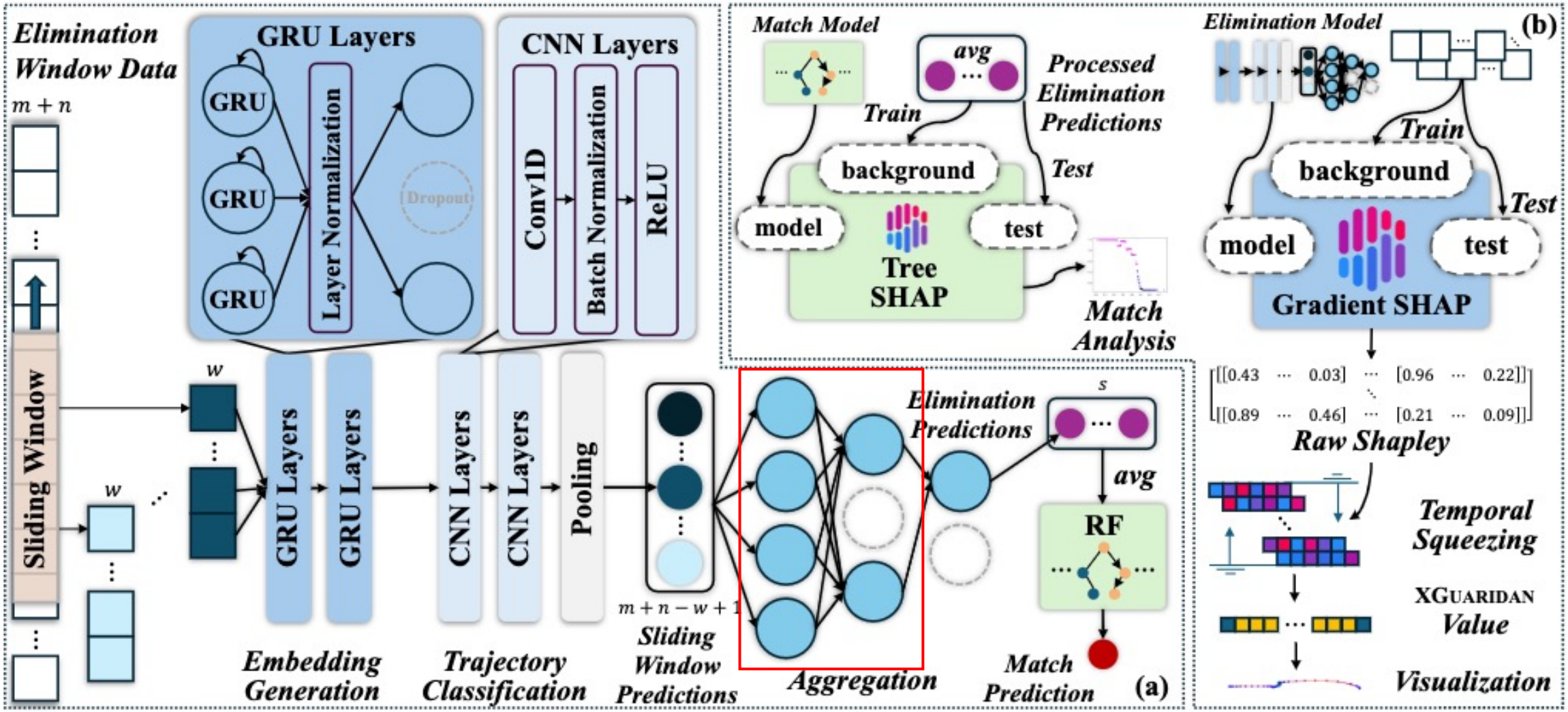


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

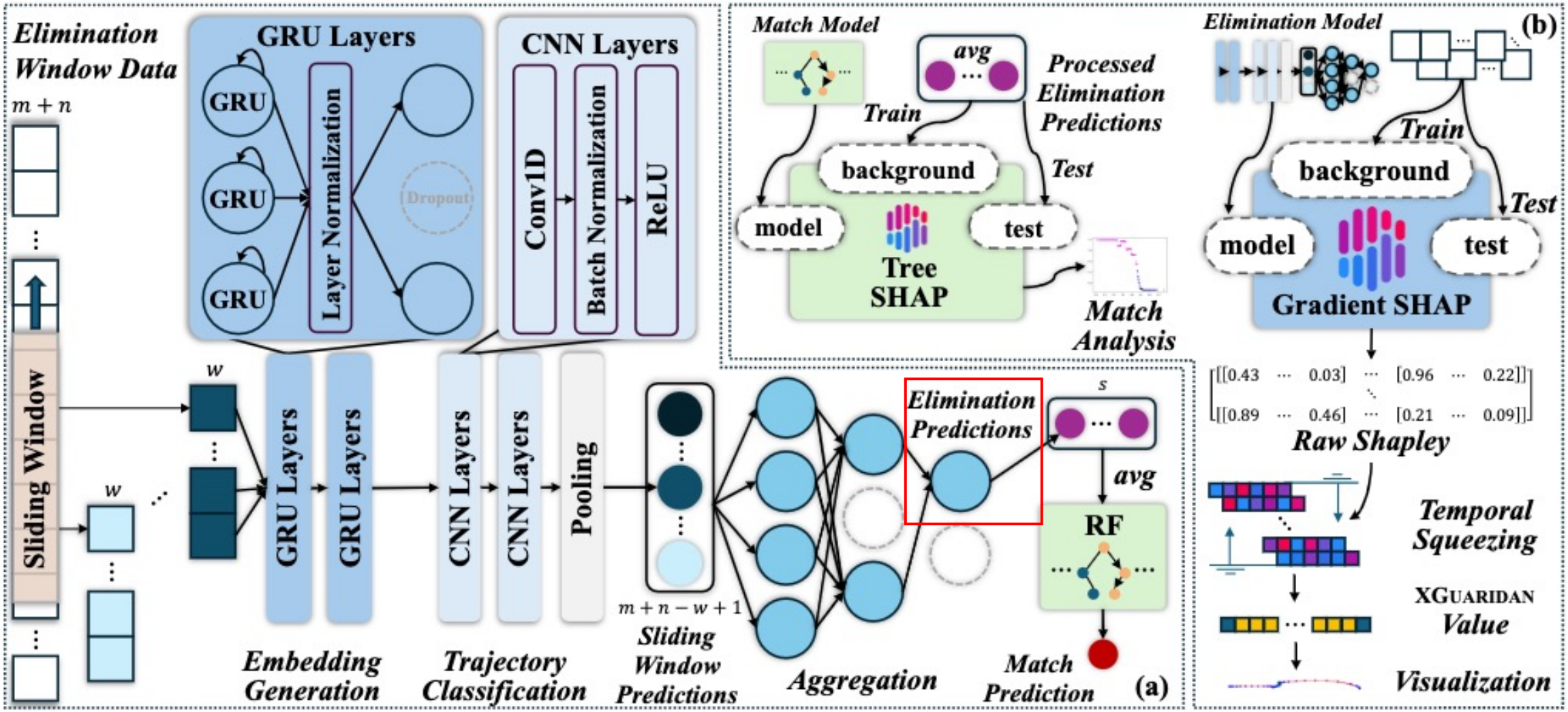


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

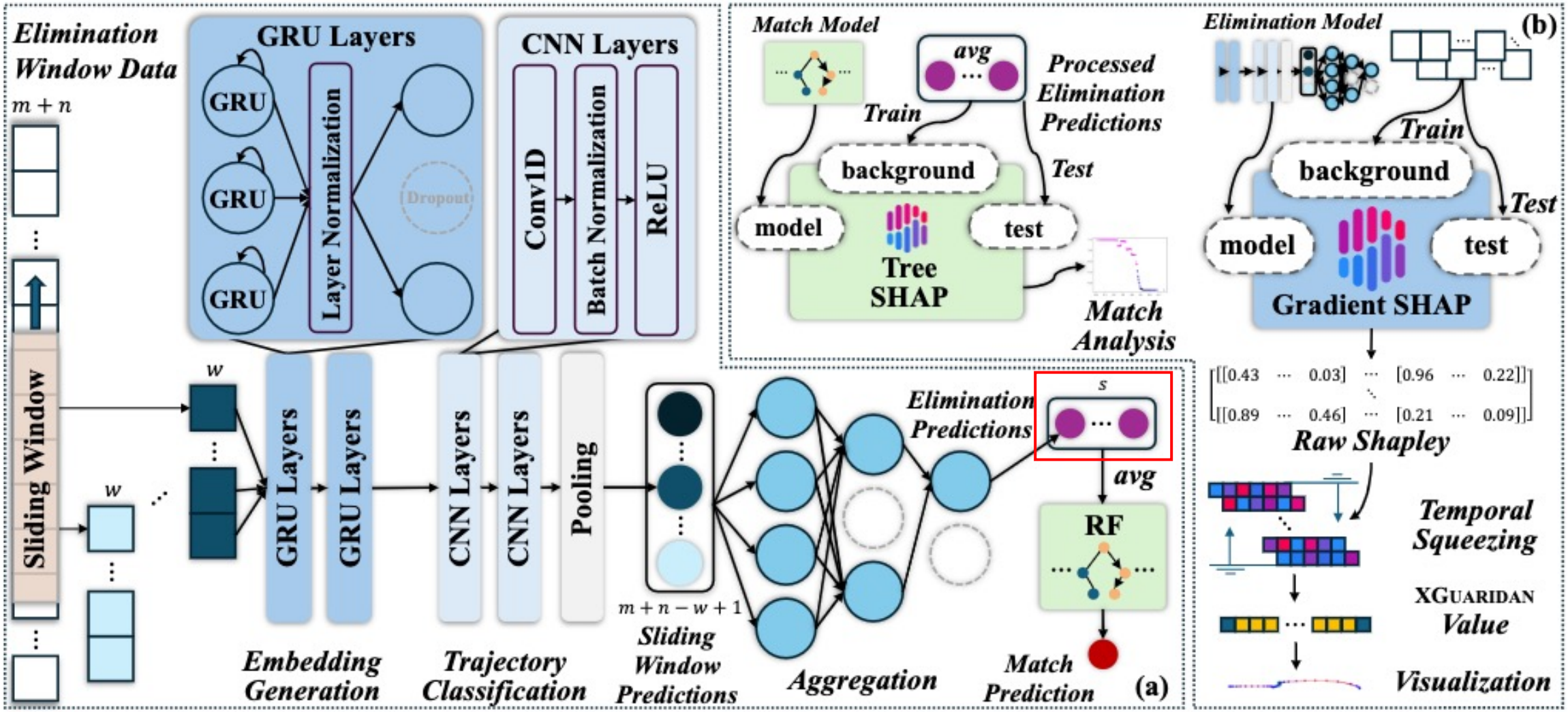


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

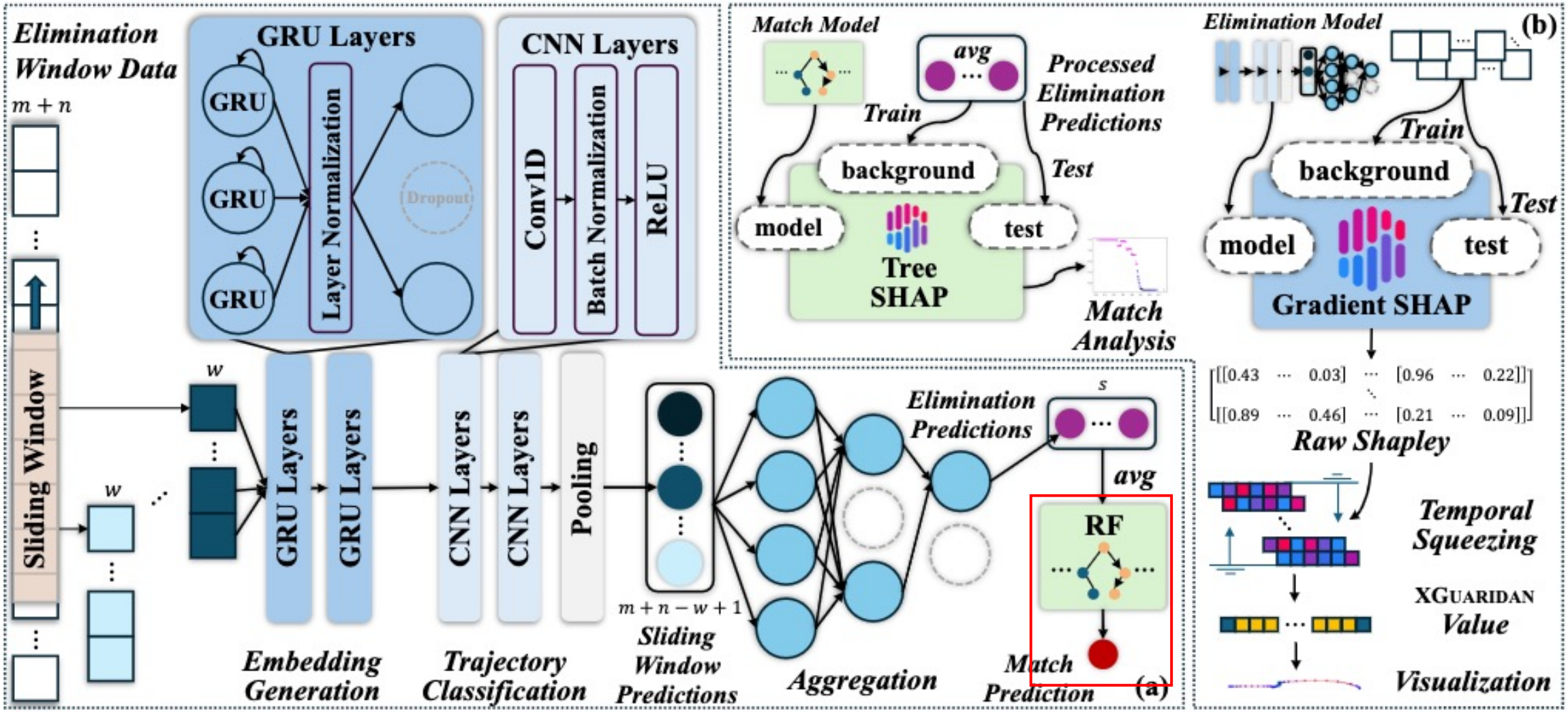


Figure 5: XGuardian framework. The subfigures illustrate the architectures of (a)-Inspector and (b)-Explainer.

Framework

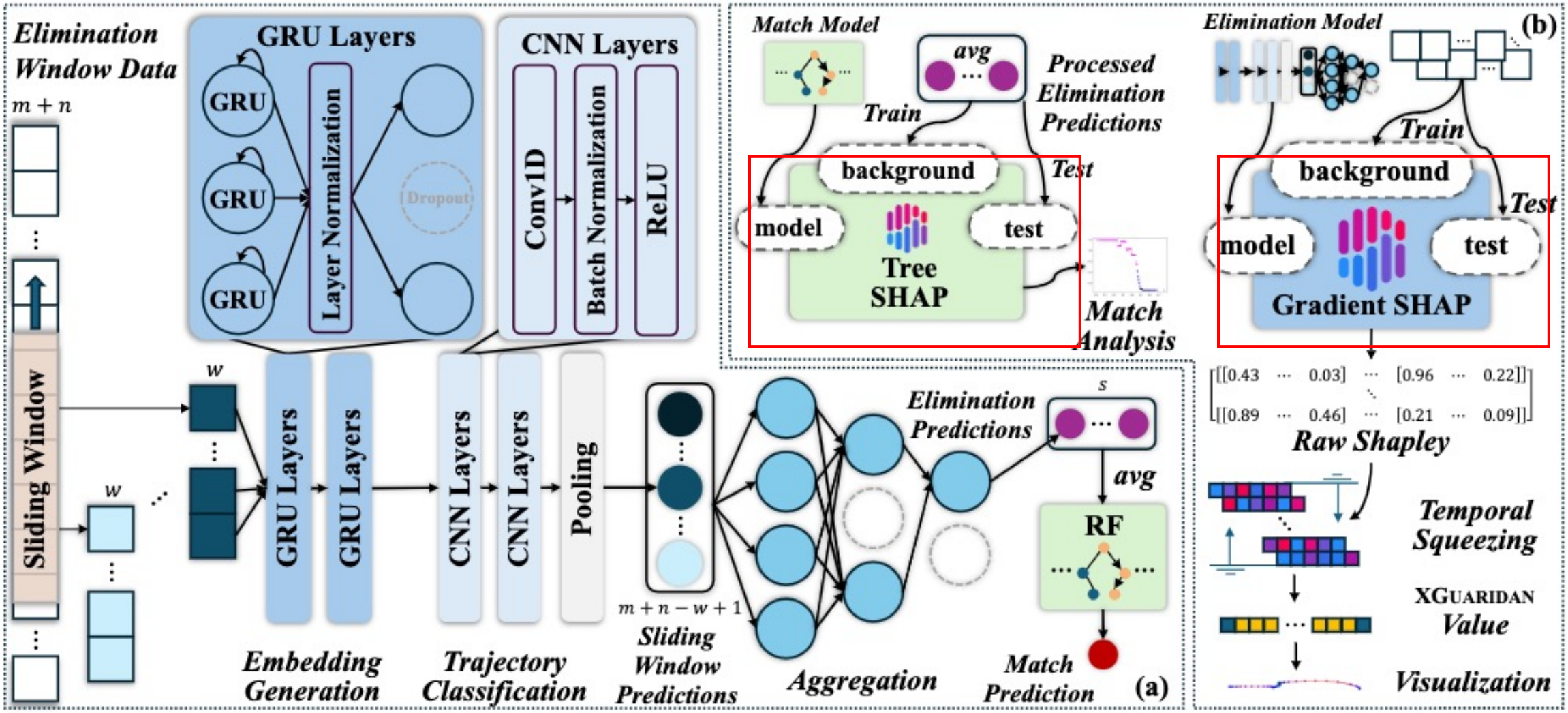


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

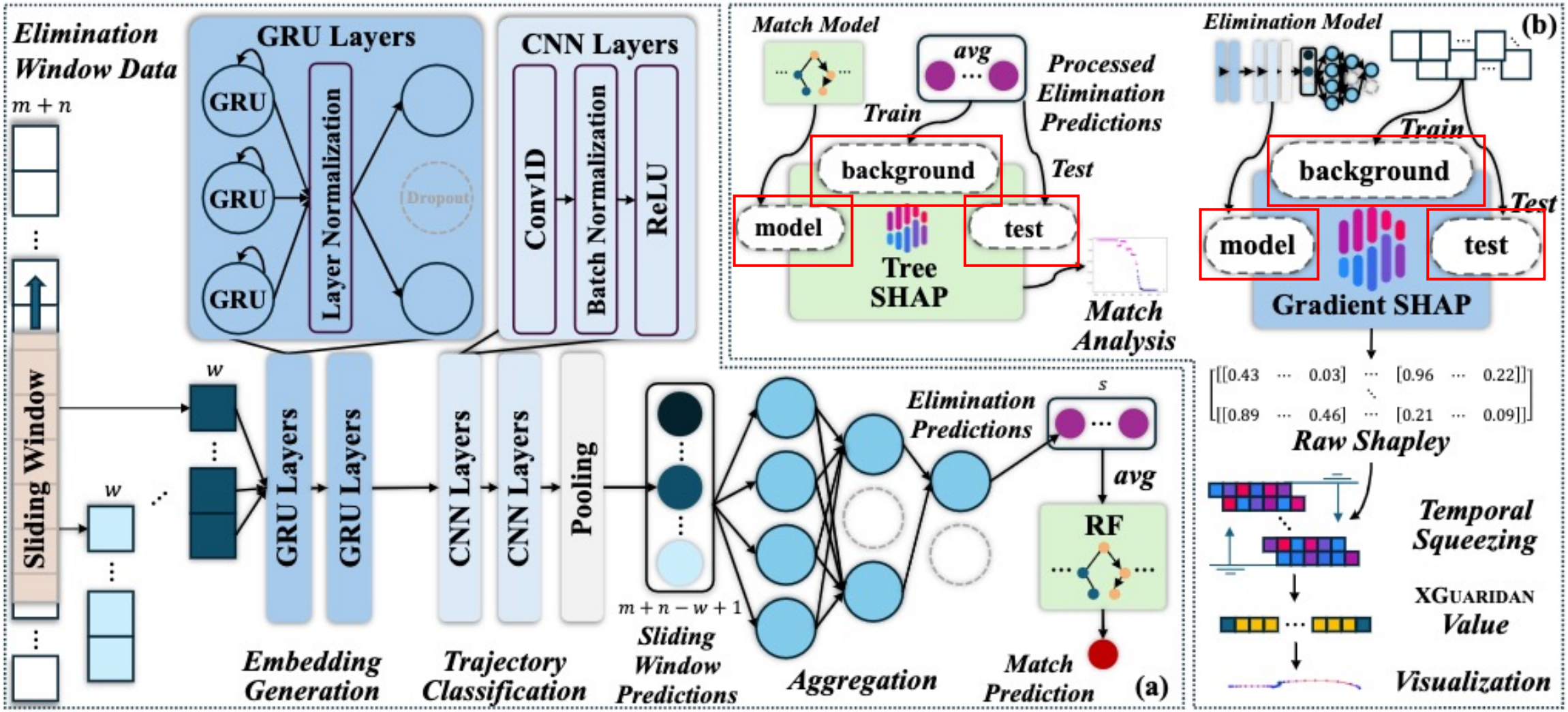


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

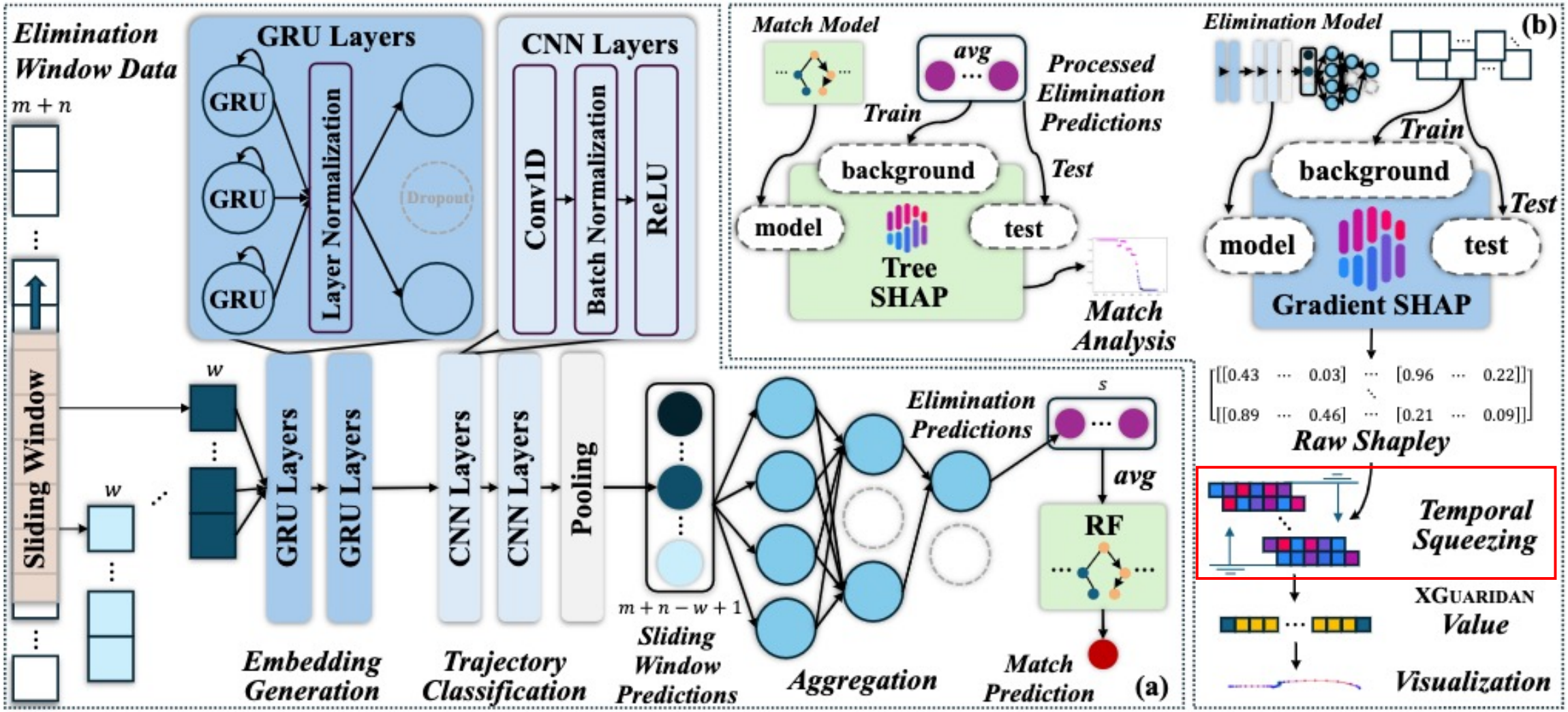


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

Framework

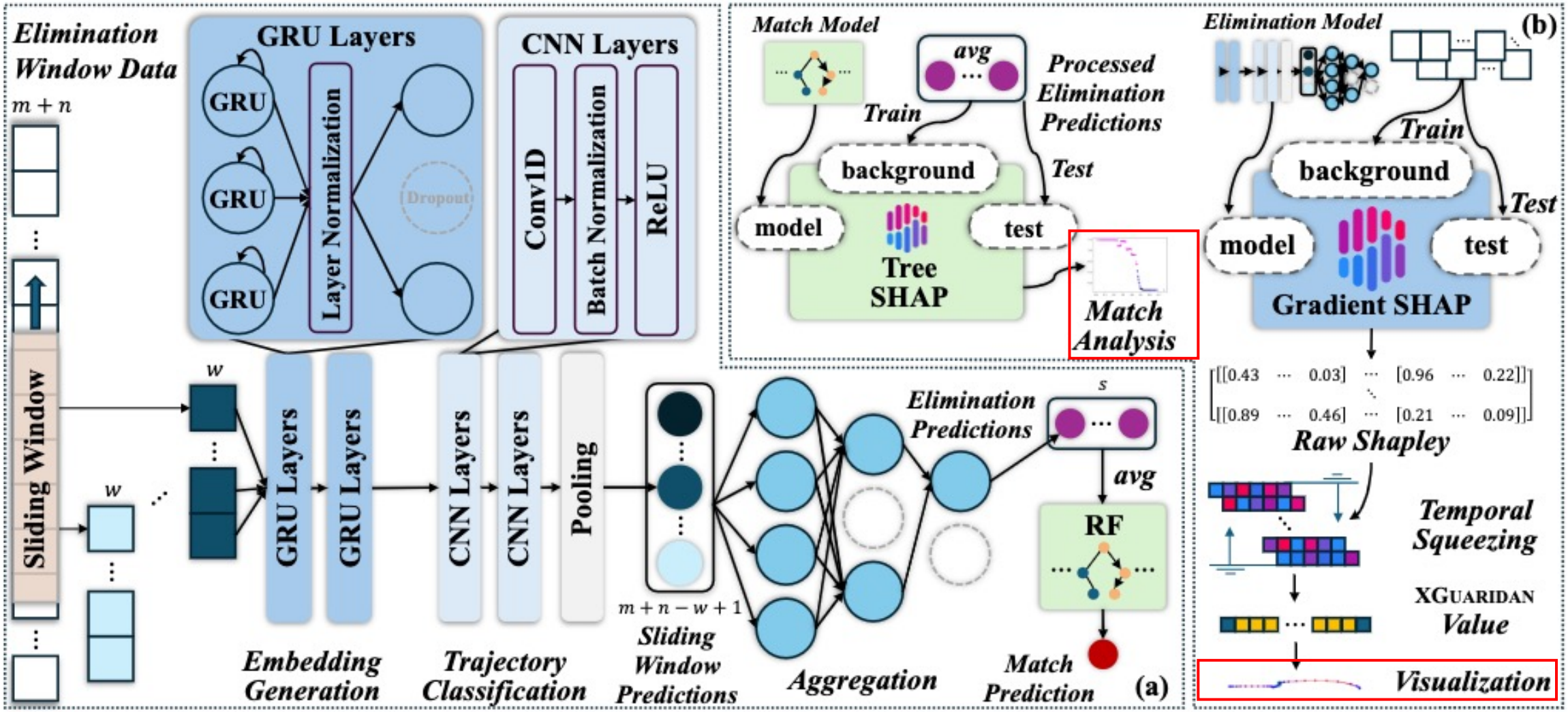


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

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.

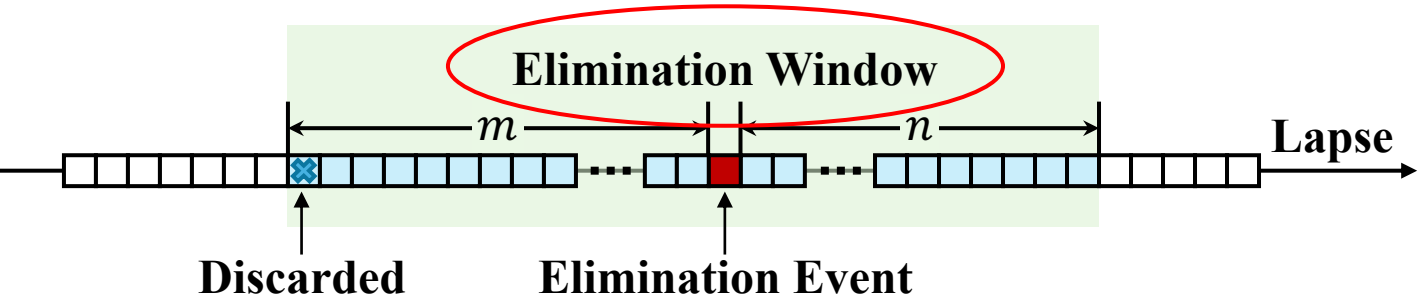
Feature			Metrics				
$\vec{a}_x$,$\vec{a}_y$	$\vec{v}_x$,$\vec{v}_y$	θ	**Accuracy**	**FPR**	**Recall**	**Precision**	**F1-Score**
✓	✗	✗	0.838	0.059	0.838	0.941	0.873
✗	✓	✗	0.884	0.060	0.884	0.940	0.904
✗	✗	✓	0.877	0.057	0.877	0.943	0.900
✓	✓	✗	0.886	0.062	0.886	0.938	0.905
✓	✗	✓	0.880	0.058	0.880	0.942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



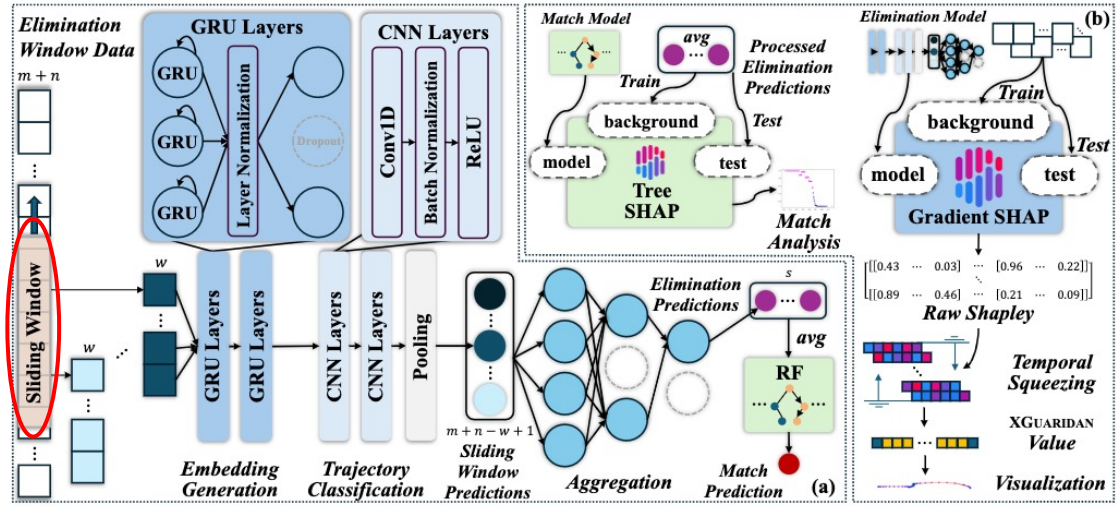
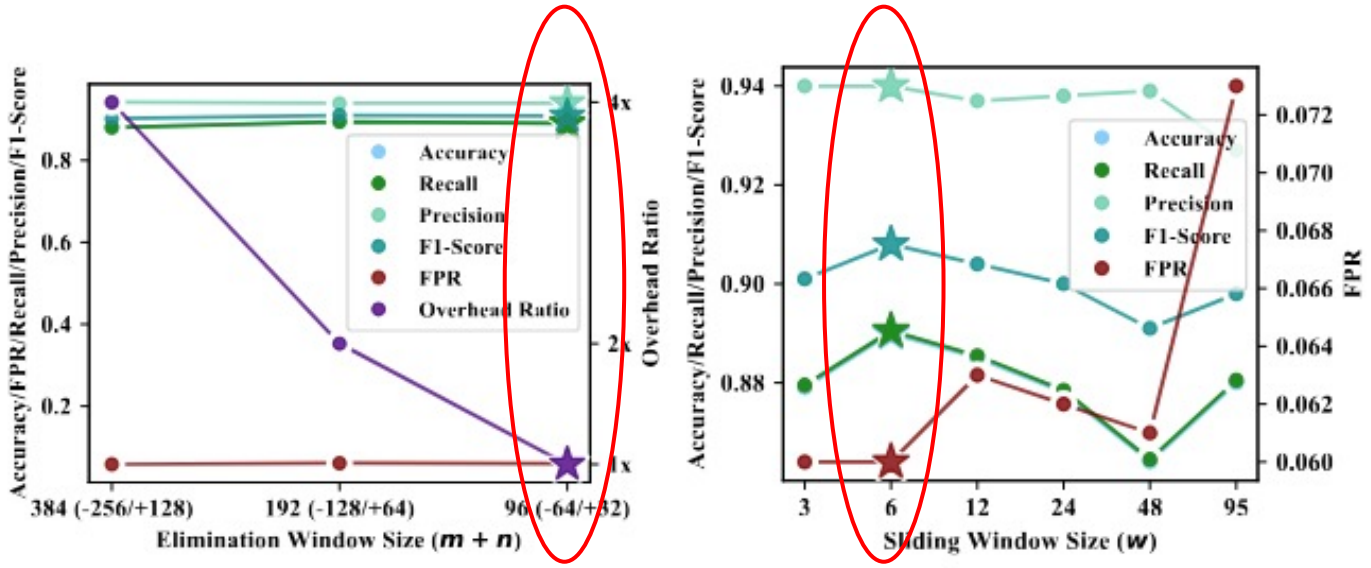


Figure 5: XGuardian framework. The subfigures illustrate the architectures of (a)-Inspector and (b)-Explainer.

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

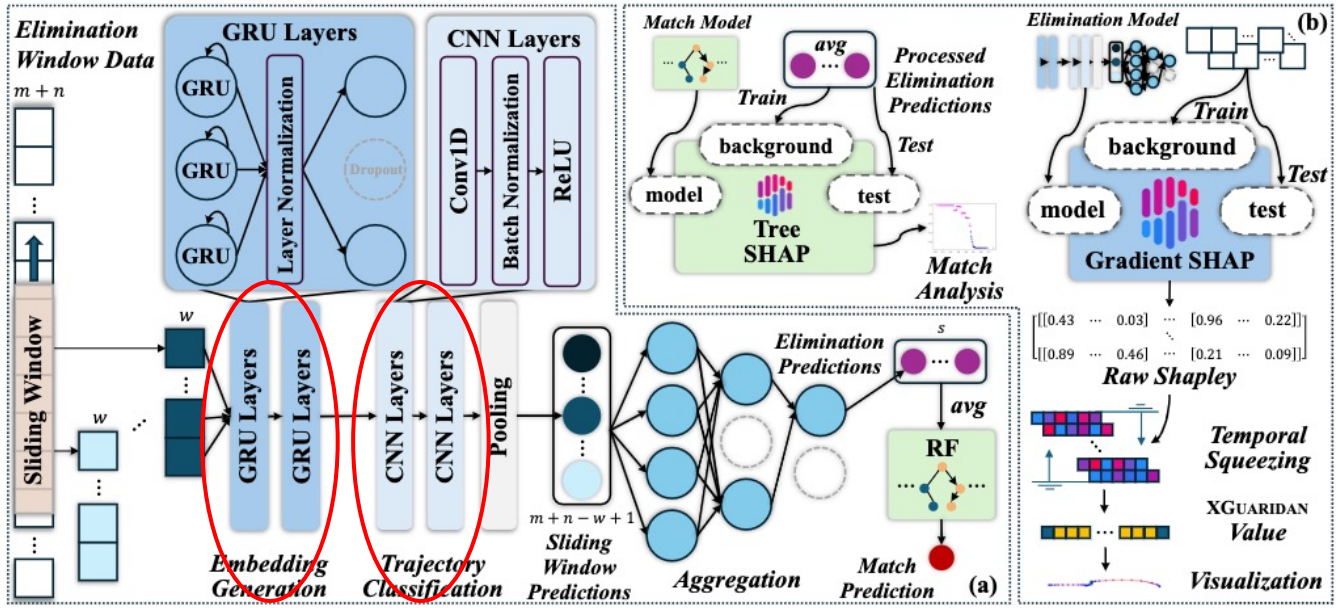
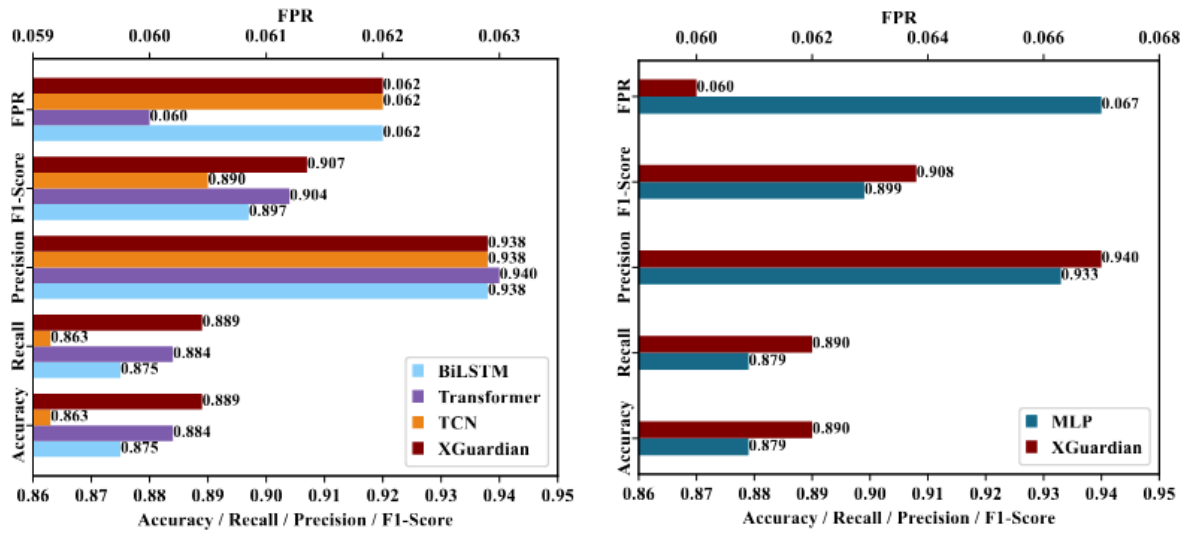


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

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

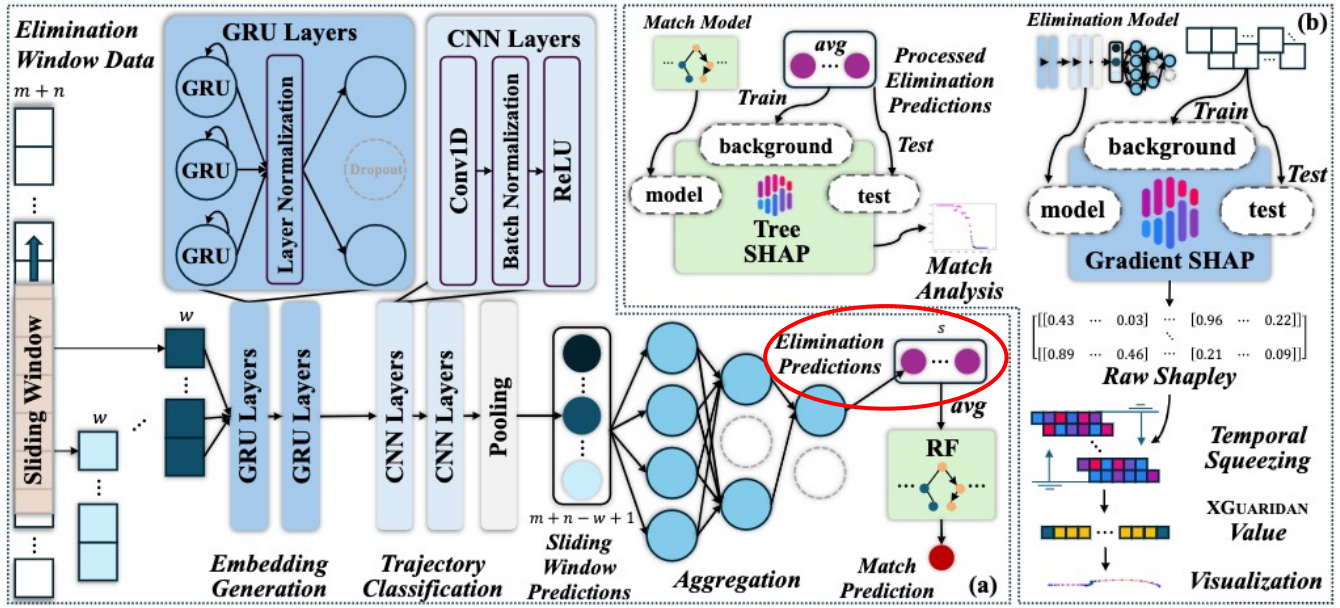


Figure 5: XGUARDIAN framework. The subfigures illustrate the architectures of (a)-INSPECTOR and (b)-EXPLAINER.

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/

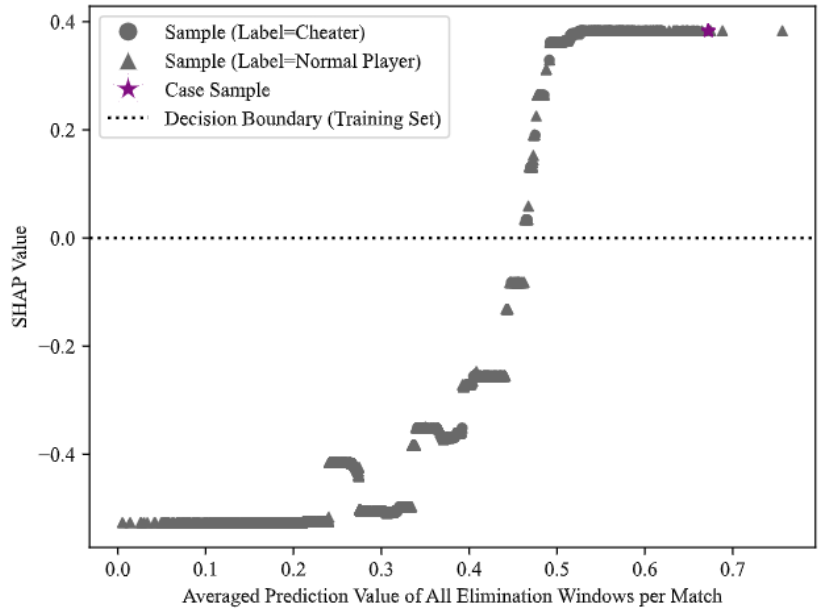


Figure 10: Explainable match-scope visualizations of an identified aim-assist cheater.

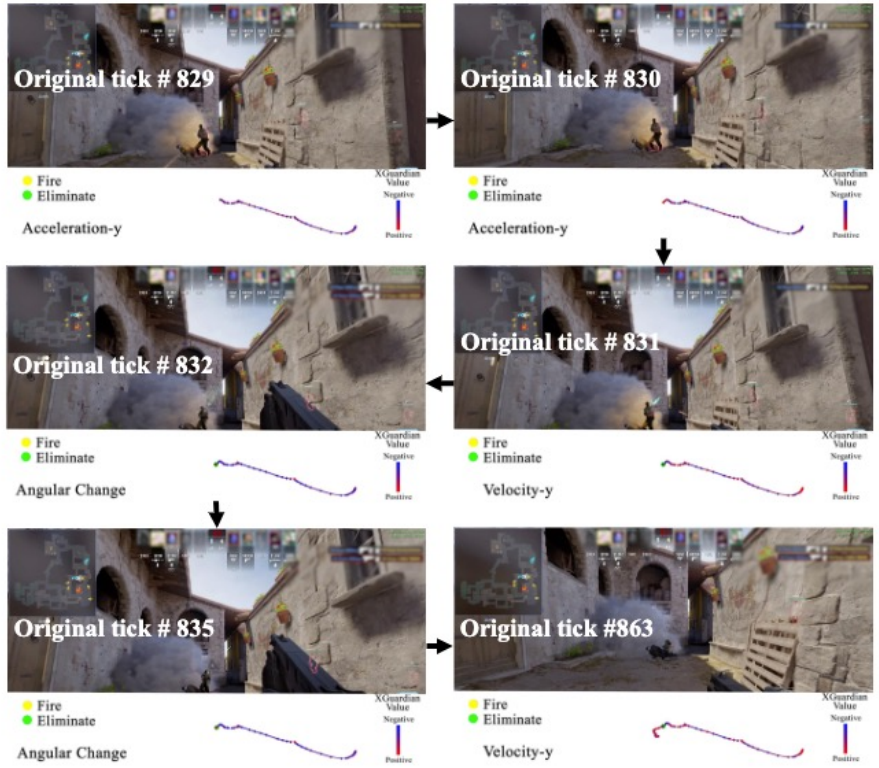


Figure 9: Explainable elimination-scope visualizations of an identified aim-assist cheater.

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