CHAMP: Cross-domain Hybrid Architecture for Matchmaking and Prediction in Online Multi-Player Games

Kai Wang
Independent Researcher
Santa Clara, United States
wangjinjie722@gmail.com

Ge Fan*
Independent Researcher
Hangzhou, China
ge.fan@outlook.com

Chaoyun Zhang
Independent Researcher
Beijing, China
vyokky@163.com

Yuyang Jiang
The Hong Kong University of Science and Technology
Hong Kong SAR, China
yjiang257@connect.ust.hk

Yuze Liu*
Swinburne University of Technology
Melbourne, Australia
yuzeliu@swin.edu.au

Abstract

Multiplayer Online Battle Arena (MOBA) games rely on matchmaking to maintain competitive balance. Our prior work, CUPID, framed matchmaking as an assignment re-optimization problem and showed that a single-mode win-rate predictor can meaningfully rebalance teams. However, deploying such a system across diverse player populations exposes three practical bottlenecks: most queueing players lack sufficient in-mode match history (cold start), skill distributions shift drastically across rank tiers (distribution inconsistency), and extreme skill segments are severely data-starved.

We present CHAMP, a cross-domain matchmaking framework that resolves these deployment bottlenecks. To address data sparsity and cold starts, CHAMP replaces the target-mode-only player profile with a hybrid domain feature collection: a timestamp-ordered cross-mode short-term sequence whose slices are annotated with target-domain features, plus per-mode breakdowns of long-term, real-time and team statistics. We further propose the Domain-Aware Win-rate Network (DAWN): a Domain-aware Knowledge Extractor (DAKE) compiles target-mode attributes into learnable representations that feed Domain-Aware Temporal/Spatial/Permutation OmniNet Encoders (DATOE/DASOE/DAPOE), so that mode-conditioned representations and per-mode debiasing are learned jointly inside a single shared network. Online, one trained DAWN serves every supported mode, with per-mode position-satisfaction thresholds as the only mode-specific knob.

Offline, DAWN achieves 67.73% win-rate prediction accuracy, outperforming all evaluated attention and sequence baselines. Online A/B tests across the entire League ladder of a large-scale MOBA game, from novice players up to the top-expert players served by Elite Mode, demonstrate consistent drops in imbalanced matches. For lower-tier players, CHAMP reduces the 5-minute kill crushing rate by up to 20.73%.

Keywords

Matchmaking Systems, Cross-Domain Learning, Deep Learning

1 Introduction

Multiplayer Online Battle Arena (MOBA) games have achieved remarkable popularity and economic success, attracting significant research interest in the gaming and AI communities [31, 33]. Games such as *League of Legends* (LoL) and *Dota 2* boast hundreds of millions of players worldwide [1, 42]. A critical component in these games is the matchmaking system, which assembles teams of comparable skill levels to ensure fair, competitive, and enjoyable matches [5, 7, 39].

The dominant industrial practice still relies on a single Matchmaking Rating (MMR) score, including ELO [3], TrueSkill [19], TrueSkill2 [30], or their learned variants [43], to group ten queueing players into two opposing teams. While computationally cheap, this scalar-skill view exposes several structural defects in a real MOBA: match fairness is approximated by the closeness of two team-MMR sums, which ignores intra-team position assignments and high-order player–player synergies that strongly influence match outcomes [16]; MMR is updated purely from win/loss outcomes, so it converges slowly for new or low-volume players and is noisy at the skill extremes; and once the ten players are grouped, MMR offers no mechanism to revisit team or role assignments, leaving many objectively imbalanced lobbies that the system has no further chance to repair.

To overcome these limitations, prior work Cupid [14] introduced a *re-matchmaking* stage that intervenes after the initial MMR-based grouping. Cupid uses a deep learning model to score candidate team-and-position permutations and pick the assignment that is most satisfying to the players and most fair as a match-up; deployed in the League mode of a popular MOBA game, it yielded significant gains in both position satisfaction and game fairness over MMR-only matchmaking. The win-rate predictor that drives this ranking, however, is trained *within a single game mode*, on features collected exclusively from that mode. In a live MOBA, players rarely live inside one mode: a typical player warms up a new champion in casual or quick-match queues, tries variant lineups in 3v3 or ARAM-style brawls, and then brings the same champion into the League mode, so behavior signals from neighboring modes carry strong information about how that player will perform in the target mode. At the same time, training a dedicated re-matchmaking framework for every mode, each requiring its own data pipeline, training schedule,

*Corresponding Author.

and serving footprint, would be operationally heavy and wasteful, especially because most modes are too data-thin to support a well-calibrated predictor on their own. Generalizing Cupid to this realistic, multi-mode setting therefore exposes three practical challenges that motivate the present work:

(1) **Player Cold Start.** A large fraction of queueing players have very limited match history in the target mode, e.g., hundreds of casual games but only a handful of League matches this season, or a recently switched main role. In-target-mode features alone yield high-variance win-rate estimates and unstable assignment decisions.
(2) **Distribution Inconsistency.** Players' behavior differs substantially across modes: 5v5 League rewards macro rotations, 3v3 brawls emphasize mechanical dueling, and ARAM compresses gold curves. A predictor trained on pooled multi-mode data without explicit mode conditioning is systematically miscalibrated on any single mode, directly biasing the fairness ranking.
(3) **Data Sparsity.** Some modes simply lack enough matches for accurate single-mode training. The most acute case is *Elite Mode* (top-expert players), where the player pool is small and a full season yields orders of magnitude fewer samples than mid-tier modes. Training a dedicated predictor overfits, while a mode-agnostic predictor is dragged toward the data-rich bulk and loses accuracy where matchmaking is hardest.

To address these challenges, we propose CHAMP, an enhanced re-matchmaking framework that targets cross-mode matchmaking from three complementary angles: feature design, model architecture, and deployment. At the *feature* level, we replace the single-mode behavior sequence used in Cupid with a cross-mode sequence and further enrich each player's profile with per-mode long-term statistics and real-time signals, so that the input describes a player's state in a more complete and mode-aware fashion. At the *model* level, we propose the Domain-Aware Win-rate Network (Dawn), which extends the OwO backbone with a Domain-aware Knowledge Extractor (DAKE) and Domain-Aware Temporal/Spatial/Permutation OmniNet Encoders (DATOE, DASOE, DAPOE): DAKE compiles target-mode attributes into learnable representations that are jointly consumed by the encoders, so that mode-conditioned representations and per-mode debiasing are produced inside a single shared network rather than via separate per-mode heads. At the *deployment* level, instead of training a separate re-matchmaking system per mode, we serve all supported modes from a single Dawn model and reflect mode-specific position-preference characteristics through per-mode satisfaction thresholds in the assignment filter, so that one online service consistently handles diverse game modes. In summary, the contributions of this paper are as follows:

(1) **Cross-Mode Re-Matchmaking Framework.** We extend re-matchmaking from a single-mode pipeline to a unified cross-mode framework that treats a player's behavior across every available game mode as a first-class input. The framework consolidates cross-mode behavior sequences together with per-mode long-term statistics and real-time signals into a single, mode-aware player representation, so that one re-matchmaking system can consistently serve players whose history spans multiple modes and directly mitigate player cold-start and data sparsity at the system level.
(2) **Domain-Aware Win-rate Network (Dawn).** We propose Dawn, a prediction model that extends the OwO backbone with DAKE and three Domain-Aware encoders (DATOE, DASOE, DAPOE). DAKE compiles target-mode attributes into learnable representations that are jointly consumed by the encoders, so that mode-conditioned representations and per-mode debiasing are produced inside a single shared network, yielding 67.73% prediction accuracy that surpasses every evaluated baseline including the OwO model from Cupid.
(3) **Unified Multi-Mode Deployment.** We deploy CHAMP as a single online service that drives all supported game modes from one Dawn model, with mode-specific position preferences expressed through per-mode satisfaction thresholds in the assignment filter. Online A/B tests covering every League tier from novice up to the top-expert players served by Elite Mode show consistent drops in economy and kill crushing rates, with up to 20.73% in kill crushing rate at 5 min for the lowest tier, and CHAMP has been deployed at scale.

2 Related Work

2.1 Matchmaking Systems in Online Games

Online MOBA matchmaking traditionally relies on a scalar MMR, estimated by ELO [3], TrueSkill [19], or TrueSkill2 [30]. Subsequent work goes beyond scalar skill: OptMatch [16] models high-order player–champion interactions, GloMatch [8] optimizes global match quality with deep reinforcement learning, and QuickSkill [43] addresses cold-start skill estimation.

Orthogonal to skill estimation, win prediction models forecast match outcomes either pre-match from roster and history [9, 17, 40] or in-game from live state [18, 20, 22, 29, 35, 41]. Bridging both threads, the Cupid framework [14] introduces *re-matchmaking* (re-optimizing team and position assignments after the initial MMR lobby is formed) and proposes the OwO encoder atop OmniNet [37], with temporal, spatial, and permutation channels, as the underlying pre-match win-rate predictor.

2.2 Domain-Aware Modeling in Data Mining

Domain-aware modeling has emerged from two complementary lines of data mining research. The first is *cross-domain transfer learning* [13, 15, 25–27, 32, 46], in which a data-rich source domain is used to lift performance on a cold or sparse target domain. The second is *multi-domain architectures* such as STAR [36], MMOE [28], and CFM [11, 12], which share most parameters across domains while reserving lightweight domain-specific structure for residual heterogeneity. Adjacent to both, target-relevance attention models such as DIN [45] and DIEN [44] condition representations on a target *item* rather than a target *domain*, and are therefore complementary rather than substitutable.

CHAMP inherits the multi-domain conditioning paradigm from this lineage and applies it to MOBA win prediction by treating each game mode as a domain. The Domain-Aware encoders inject a learnable representation of the target mode directly into the OwO backbone via dedicated domain context tokens, so mode-conditioned

representations and per-mode debiasing are learned inside a single shared network rather than as a post-hoc reweighting layer.

3 Preliminary

3.1 Background and Re-matchmaking

A comprehensive treatment of MOBA matchmaking can be found in Cupid [14]; here we summarize only the components on which CHAMP builds. Modern production matchmaking systems decompose the team-formation task into two sequential stages that operate at distinct granularities: a coarse-grained *pre-matching* stage that assembles candidate lobbies from the live queue, followed by a fine-grained *re-matchmaking* stage that determines the final team-and-position assignment within each lobby.

Pre-matching operates over the entire live player queue. It periodically picks 10 players whose MMR values fall within a narrow window and groups them into a single lobby, with the dual objectives of approximate skill parity at the player level and short queue times. This stage must scale to millions of concurrent queueing players and therefore relies on cheap, scalar MMR-window heuristics rather than per-lobby modeling.

Re-matchmaking then operates inside each grouped 10-player lobby. The role-and-team layout *within* a lobby is a much smaller combinatorial space (a fixed set of role/team assignments over 10 players) but has a disproportionate impact on match outcome: a balanced MMR pool can still produce blowouts if positions are misassigned or if one team systematically concentrates win-rate-correlated traits. Re-matchmaking exploits this small candidate space to optimize both position satisfaction and predicted match fairness using machine-learning models that would be far too expensive to run over the full queue.

The two stages are complementary: pre-matching keeps queue times tractable at production scale, while re-matchmaking spends modeling budget exactly where it matters, namely the small set of intra-lobby permutations that actually drive blowouts. The rest of this paper focuses on the re-matchmaking stage, which proceeds in two further steps as detailed below.

Step 1: Candidate generation by position preference. A position-preference program enumerates a set of candidate team-and-position assignments $\mathcal{A} = \{A_{t_1}^{i_{1,1}}, \ldots, A_{t_N}^{i_N,N}\}$ over the 10 grouped players, and scores each candidate by its overall position satisfaction

$$\mathcal{P}_{\mathcal{A}} = \prod_{n=1}^{N} p_n^{i_n}, \tag{1}$$

where $p_n^{i_n}$ is player n's satisfaction score for position i_n. Candidates whose satisfaction falls below a threshold $\mathcal{P}_\tau$ are pruned, leaving only assignments whose role mapping is acceptable to all 10 players.

Step 2: Final selection by win-rate prediction. The surviving candidates are then re-ranked by a win-rate prediction model. For each candidate $\mathcal{A}$ the model produces a predicted win rate $\hat{\mathbf{y}}_{\mathcal{A}}$, from which we derive a fairness score

$$s_{\mathcal{A}} = 1 - 2|\hat{\mathbf{y}}_{\mathcal{A}} - 0.5|, \tag{2}$$

which peaks at 1 when the predicted win rate is exactly 0.5 (a perfectly balanced matchup) and decays toward 0 as the predicted outcome becomes more lopsided. The system returns the candidate with the highest $s_{\mathcal{A}}$, i.e., the position assignment whose predicted win rate is closest to 50%.

The accuracy of $\hat{\mathbf{y}}_{\mathcal{A}}$ is therefore the binding constraint on online fairness: a miscalibrated win-rate predictor will surface lopsided lobbies even when truly fair candidates exist in $\mathcal{A}$. This motivates our focus on cross-domain win-rate prediction in the rest of the paper.

3.2 Cross-Domain Setup

Modern MOBA games host heterogeneous player sub-populations that differ in skill distribution, behavior, and match volume. In this work we treat each such sub-population, whether a distinct game mode or a distinct rank tier within a mode, as a separate *domain* for matchmaking. Offline experiments aggregate across game modes; online evaluation stratifies by rank tier (Table 3), reflecting both senses of domain in our empirical setup.

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ denotes the set of M domains available during training. For a given matchmaking request in domain d_m, the system predict match outcomes using potentially limited in-domain data, while leveraging cross-domain knowledge from the remaining domains $\mathcal{D} \setminus \{d_m\}$.

4 The Design of CHAMP

4.1 CHAMP in a Nutshell

The CHAMP architecture keeps the original Cupid assignment filter and re-matchmaking loop, but redesigns three pieces of the pipeline so that a single re-matchmaking system can serve every game mode of a live MOBA:

(1) **Hybrid Domain Feature Collection.** Each player is represented by a hybrid input that mixes behavior across modes: a timestamp-ordered cross-mode short-term sequence in which every slice is annotated with target-domain-specific features, plus long-term, real-time, and team statistics that are additionally broken down per mode.
(2) **DAWN with Domain-Aware Encoders.** The win-rate predictor is Dawn. DAKE compiles the target mode's match-related attributes into a learnable domain representation, which is fed into the upgraded DATOE, DASOE, and DAPOE encoders so that DATOE and DASOE learn mode-conditioned per-player representations and DAPOE supplies additional match context together with an in-network debiasing channel.
(3) **Unified Online Deployment.** A single trained Dawn instance drives every supported mode online; per-mode position-satisfaction thresholds in the assignment filter are the only mode-specific knobs, so one service consistently handles all supported game modes.

4.2 Hybrid Domain Feature Collection

A standard matchmaking pipeline characterizes each player from a single bag of in-target-mode features. CHAMP instead constructs a *hybrid* input that interleaves cross-mode behavior with target-domain-specific signals on two complementary time scales: a timestamp-ordered short-term sequence and a per-mode statistical profile. Throughout this section we treat a matchmaking request in

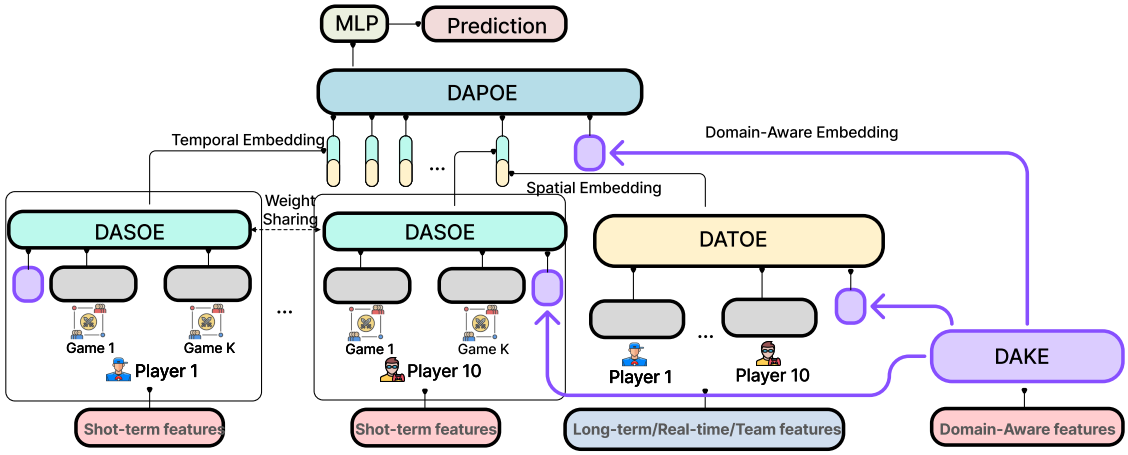

Figure 1: Overall architecture of CHAMP, extending Cupid with the Hybrid Domain Feature Collection, Dawn (DAKE + DATOE/DASOE/DAPOE)

target mode $d_m \in \mathcal{D}$ and parameterize this context with a learnable mode embedding $\mathbf{e}_m \in \mathbb{R}^{d_e}$.

(i) Cross-mode short-term sequence with per-slice domain features. For each player n we fetch the last K matches across *all* modes and order them by timestamp into a single sequence

$$\mathbf{X}_n^{ST} = \left[(\mathbf{f}_n^{(1)}, \mathbf{f}_{n,1}^{d}, m_1), \ldots, (\mathbf{f}_n^{(K)}, \mathbf{f}_{n,K}^{d}, m_K) \right], \tag{3}$$

where $\mathbf{f}_n^{(k)}$ is the per-match action profile (KDA, gold, objectives, etc.), $m_k \in \mathcal{D}$ is the mode of the k-th match, and $\mathbf{f}_{n,k}^{d}$ is a per-slice supplement describing how that match relates to the *target* domain d_m (e.g. position/role match with the requested slot, mode-affinity statistics, recency under d_m). The mixed ordering preserves the natural temporal interleaving of a player's behavior, while the per-slice domain features tell the model how to compare a casual game with a League one when both are queried by the same target mode.

(ii) Per-mode long-term, real-time, and team statistics. Beyond the recent sequence, CHAMP also enriches the long-term, real-time, and team-level views with per-mode breakdowns. Concretely, we concatenate one statistics vector per mode together with a global summary,

$$\mathbf{X}_n^{LT} = \left[\mathbf{x}_{n,d_1}^{LT}; \mathbf{x}_{n,d_2}^{LT}; \ldots; \mathbf{x}_{n,d_M}^{LT}; \mathbf{x}_{n,*}^{LT} \right], \tag{4}$$

where $\mathbf{x}_{n,d_j}^{LT}$ aggregates the player's long-term performance, real-time form indicators and team-context statistics restricted to mode d_j, and $\mathbf{x}_{n,*}^{LT}$ is the corresponding mode-agnostic summary. Together, equations (3)–(4) provide a richer, multi-resolution view of the player than a target-mode-only profile, and serve as the raw inputs consumed by Dawn in Section 4.3.

4.3 DAWN Architecture

The core win-rate predictor in CHAMP is Dawn, illustrated in Figure 1. Dawn keeps the three-encoder OwO backbone of Cupid [14] but upgrades each component into a domain-aware variant. DAKE compiles attributes of the target mode into learnable representations, which the upgraded DATOE, DASOE, and DAPOE consume jointly with their original inputs.

4.3.1 Domain-aware Knowledge Extractor (DAKE).. For a target mode d_m, DAKE takes as input the learnable mode embedding $\mathbf{e}_m$ together with a vector $\boldsymbol{\phi}_m$ of mode-related attributes (map type, party size, role/position layout, mode-level statistics, etc.) and produces a sequence of T domain context tokens

$$\mathbf{C}_m = \mathrm{DAKE}(\mathbf{e}_m, \boldsymbol{\phi}_m) \in \mathbb{R}^{T \times d}. \tag{5}$$

DAKE thus turns mode-side knowledge into a learnable, fixed-shape representation that downstream encoders can attend to and concatenate with their own inputs.

4.3.2 Domain-Aware Encoders (DATOE, DASOE, DAPOE).. Each encoder is an OmniNet [37], a Transformer augmented with omni-directional residual attention,

$$\mathrm{OmniNet}(\mathbf{X}) = \mathrm{Transformer}(\mathbf{X})_L + \mathrm{MLP}\big(O_{\mathrm{att}}(\mathbf{X})\big), \tag{6}$$

where O_{att} aggregates hidden states across all layers. The DA-variants accept the original encoder input and additionally take the DAKE tokens $\mathbf{C}_m$ as an extra input slot, so that mode-conditioned representations can be learned *inside* the encoder rather than only as a late-stage adjustment. Concretely, for the n-th player in a candidate assignment A,

$$\tilde{\mathbf{h}}_n^T = \mathrm{DATOE}\big([\mathbf{X}_n^{ST}; \mathbf{C}_m]\big), \qquad n = 1, \ldots, 10, \tag{7}$$

$$\tilde{\mathbf{h}}_n^S = \mathrm{DASOE}\big([\mathbf{X}_n^{LT}; \mathbf{C}_m]\big), \qquad n = 1, \ldots, 10, \tag{8}$$

where $\mathbf{X}_n^{ST}$ and $\mathbf{X}_n^{LT}$ are the hybrid features defined in equations (3)–(4). Conditioning on $\mathbf{C}_m$ lets DATOE and DASOE adapt *what* to look at on a per-mode basis: a strong KDA, a meaningful gold-per-minute, or a useful objective rotation contribute to win prediction differently across Casual , League, Elite modes.

DAPOE then aggregates the per-player encodings together with a pooled domain summary $\mathbf{c}_m = \text{Pool}(\mathbf{C}_m)$:

$$\mathbf{z} = \text{DAPOE}\big([\,\tilde{\mathbf{h}}_1^T; \tilde{\mathbf{h}}_1^S; \ldots; \tilde{\mathbf{h}}_{10}^T; \tilde{\mathbf{h}}_{10}^S;\ \mathbf{c}_m\,]\big), \tag{9}$$

where the player order is fixed by the team-and-position layout of A, so that any reassignment in re-matchmaking translates directly into a change of input order [14]. The dedicated $\mathbf{c}_m$ slot supplies DAPOE with explicit match-level context and also acts as an *in-network debiasing channel*: gradients prefer to flow through this dedicated mode pathway rather than baking mode-specific shifts into the shared per-player representations, leaving the player encodings cleaner and more transferable across modes.

4.3.3 Prediction Head and Training Objective. A single shared head $g(\cdot)$ maps the pooled representation $\mathbf{z}$ to the win probability,

$$\hat{y} = \sigma\big(g(\mathbf{z})\big), \tag{10}$$

and Dawn is trained with the Normalized Economy Difference (NED) loss inherited from Cupid,

$$\mathcal{L}_{NED} = -\big(\alpha\, y\, \log \hat{y} + (1-\alpha)\,(1-y)\,\log(1-\hat{y})\big), \tag{11}$$

where $\alpha = (TE_1 - TE_2)/\max(TE_1, TE_2)$ rescales the penalty by the normalized team-economy gap so that close, hard-to-predict games dominate the gradient signal. Because mode information is already injected via DAKE and DAPOE, no auxiliary per-mode head or post-hoc calibration term is required.

4.4 Online Deployment

A single trained Dawn instance drives every supported game mode online. For a matchmaking request in target mode d_m over a 10-player lobby, the live system proceeds in the following four steps:

(1) **Feature retrieval and processing.** For each of the 10 players, the online feature store is queried to retrieve and process the hybrid domain features: the cross-mode short-term sequence $\mathbf{X}_n^{ST}$ (annotated with per-slice target-domain features) and the per-mode statistics $\mathbf{X}_n^{LT}$ (long-term, real-time, and team breakdowns).
(2) **Candidate generation by position satisfaction.** A position-preference program enumerates candidate team-and-position assignments. Each candidate A is scored by its overall position-satisfaction product $\mathcal{P}_A = \prod_{n=1}^{N} p_n^{in}$, and only assignments whose satisfaction exceeds the per-mode threshold τ_m are retained as the candidate set $\mathcal{A}$.
(3) **Win-rate scoring.** The surviving candidates are forwarded to Dawn. For each $A \in \mathcal{A}$, DAKE builds the domain context $\mathbf{C}_m$ from the target-mode attributes, and the DATOE/DASOE/DAPOE encoders produce the match-level representation $\mathbf{z}$, from which the shared head yields the predicted win probability $\hat{y}(A)$ (Eqs. 5–10).
(4) **Fairness-optimal assignment selection.** The candidate whose predicted win probability is closest to 50% is returned as the final assignment, i.e., $A^* = \arg\min_{A\in\mathcal{A}} |\hat{y}(A) - 0.5|$.

Because Dawn is a single shared model, deploying a new mode reduces to registering its attribute vector $\boldsymbol{\phi}_m$ and per-mode threshold τ_m; no per-mode model is trained or hosted.

5 Experiments

In this section, we design experiments to answer the following three research questions:

- **RQ1.** How does Dawn compare with state-of-the-art methods on pre-match win prediction?
- **RQ2.** How do the proposed components (Hybrid Domain Feature Collection, DAKE, and the three Domain-Aware encoders) actually contribute to the overall model performance?
- **RQ3.** How does deploying Dawn in a live production matchmaking pipeline affect downstream business metrics?

RQ1 and RQ2 are answered offline in this section; RQ3 is answered through large-scale online A/B testing in Section 5.4.

5.1 Experiment Settings

5.1.1 Datasets. We collect industrial match logs from a popular online MOBA title and construct three datasets corresponding to the three game modes that the production matchmaking system serves: **Casual**, **League**, and **Elite** (the dedicated mode for top-expert players). The three datasets contain approximately **40M**, **30M**, and **0.1M** samples, respectively, exhibiting a clear long-tail in data scale: Elite, which serves the smallest top-expert player pool, is also the sparsest mode and is therefore the primary stress-test for cold-start and data-sparsity behavior. To prevent any temporal leakage, we partition each dataset *chronologically* rather than by random sampling: the earliest matches form the training set, the most recent form the test set, and 5% of the training set (still earlier than the test horizon) is held out as a validation split for hyperparameter tuning and early stopping.

5.1.2 Baselines. Following the protocol of Cupid [14], we benchmark Dawn against: Logistic Regression (LR) [34] and a three-layer MLP [23] as shallow baselines; LSTM [21] as a sequence baseline; and Transformer [38] and OwO [37] (the omnidirectional attention backbone adopted by Cupid) as attention baselines. We additionally include **Dawn-single**, an instance of Dawn trained on the data of a single target mode (i.e., the same single-domain regime as every baseline), to isolate the effect of multi-mode joint training from the architecture itself.

5.1.3 Evaluation Metrics. Our pre-match win prediction task is naturally class-balanced (each match has one winning team and one losing team), so we report the two most widely used metrics for balanced binary prediction: **Accuracy (ACC)**, which captures the discrete classification quality, and **Root Mean Square Error (RMSE)**, which captures the calibration of the predicted win probabilities. Higher ACC and lower RMSE both indicate better performance [4, 6, 10].

5.1.4 Implementation Details. Dawn is optimized via Adam [24] in TensorFlow [2]. To ensure a rigorously controlled comparison, the core OwO hyperparameter footprint (layer depth, hidden dimensions, attention heads) within Dawn mirrors the original OwO configuration.

Table 1: Model comparison across three datasets (bold = best).

Model	Casual		League		Elite	
	ACC	RMSE	ACC	RMSE	ACC	RMSE
LR	0.5657	0.4976	0.5040	0.5069	0.5912	0.4938
MLP	0.5753	0.4771	0.5153	0.4866	0.6041	0.4725
LSTM	0.5931	0.4582	0.5231	0.4701	0.6336	0.4514
Transformer	0.6012	0.4412	0.5429	0.4518	0.6528	0.4316
OwO	0.6321	0.4203	0.5565	0.4349	0.6559	0.4160
DAWN-single	0.6534	0.4109	0.5622	0.4291	0.6580	0.4098
DAWN	**0.6612**	**0.4022**	**0.5685**	**0.4216**	**0.6773**	**0.3987**

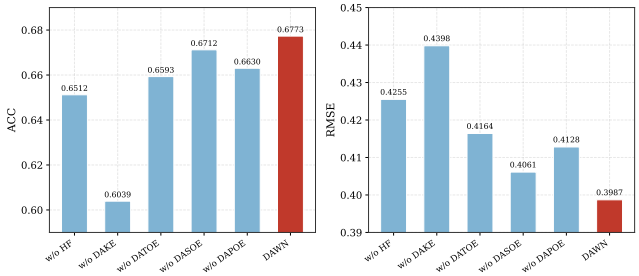

Figure 2: Ablation results on the Elite dataset.

5.2 Results Compared with Baselines (RQ1)

Table 1 reports pre-match win prediction performance. We include Dawn-single (trained on one target mode, same regime as all baselines) and Dawn (jointly consuming all three modes via the hybrid domain feature collection).

On every dataset, both variants dominate all baselines on ACC and RMSE. The performance ranking is consistent across all three modes, with Dawn at the top followed by Dawn-single, confirming that the gains stem from cross-mode joint training and the DAKE-conditioned encoders rather than dataset-specific tuning.

Crucially, Dawn outperforms Dawn-single on every cell, and the gap widens under data scarcity. On Elite, the sparsest dataset, Dawn-single barely edges out OwO (0.6580 vs. 0.6559 ACC, +0.32%), whereas Dawn jumps to 0.6773 (+2.93% over Dawn-single, the largest mode-wise gain). This confirms that cross-mode joint training effectively transfers behavioral knowledge from data-rich to data-sparse modes, directly mitigating the cold-start bottleneck that motivates CHAMP.

5.3 Ablation Studies (RQ2)

Figure 2 reports the ablation study on the sparse Elite dataset; the Casual and League datasets exhibit the same monotone ordering across all variants, and are omitted for brevity. We construct each variant by removing one component of Dawn while keeping the rest intact: *w/o HF* replaces the hybrid domain feature collection with the single-mode sequence and statistics used by every baseline; *w/o DAKE* drops DAKE (so the encoders are the original OwO encoders), which is equivalent to training a vanilla OwO on the union of all modes; and *w/o DATOE / DASOE / DAPOE* substitutes the corresponding Domain-Aware encoder with its plain OwO counterpart.

Table 2: Online A/B test results. Negative is better.

Mode	Economy Crushing Rate			Kill Crushing Rate	
	@5min	@10min	@15min	@5min	@15min
League Mode	−2.14%	−3.05%	−3.45%	−9.00%	−8.05%
Elite Mode	−3.32%	−3.75%	−3.84%	−9.62%	−8.19%

Across all five variants the deletion of any single component produces a measurable degradation relative to the full Dawn (ACC drops of 0.61–7.34 points on Elite), and the same monotone ordering holds for RMSE. This confirms that every component of the architecture, including the hybrid feature collection, the DAKE conditioning module, and each of the three Domain-Aware encoders, contributes complementary, non-redundant capacity, and that the design as a whole is internally well-balanced rather than dominated by a single block.

Most strikingly, removing DAKE causes by far the largest collapse: ACC plummets to 0.6039, which is even *below* the 0.6559 achieved by the single-mode OwO baseline on the same Elite dataset (Table 1). In other words, naively pooling all modes into a vanilla OwO does not just forfeit Dawn's gains; it actively underperforms training that vanilla OwO on Elite data alone. This indicates that without an explicit mode-conditioning signal, the heterogeneous distributions and behavioral conventions of different modes appear as noise to a shared encoder, and cross-mode supervision becomes harmful rather than beneficial. DAKE's learnable target-mode tokens are therefore the load-bearing component that makes multi-mode training viable: only once the encoders are conditioned on the target mode does aggregating cross-mode data turn from a liability into the source of Dawn's headline gains.

5.4 Online Experiments (RQ3)

While offline accuracy validates the representational power of Dawn, the ultimate test of a matchmaking engine is whether it suppresses blowout matches in a live, non-stationary environment. We therefore deployed CHAMP in two production game modes of a popular online MOBA title, specifically **League Mode** and **Elite Mode** (the dedicated mode serving top-expert players), and ran strict A/B testing against the production Cupid system as the control. The experiment spanned over 30 days, covered tens of millions of players, and generated billions of player-game participations. All improvements reported in Tables 2–3 are statistically significant (two-proportion z-test, $p < 0.01$).

Following Cupid [14], we track two primary indicators of match imbalance: **Economy Crushing Rate**, the proportion of matches whose team gold differential breaches a critical threshold at 5, 10, or 15 minutes; and **Kill Crushing Rate**, the proportion exhibiting a critical kill differential at 5 and 15 minutes. Negative shifts indicate fewer crushing matches, i.e., a tighter and more competitive experience.

Table 2 reports the aggregate impact for the two deployed modes. CHAMP uniformly reduces both crushing-rate families on *every* time horizon and in *both* modes: 5-minute kill blowouts drop by 9.00% in League and 9.62% in Elite, while 15-minute economy blowouts fall by 3.45% and 3.84%, respectively. The signs are consistent across all ten (mode × metric) cells, indicating that the gain

Table 3: League Mode A/B results by skill tier.

Rank Tier	Economy Crushing Rate			Kill Crushing Rate	
	@5min	@10min	@15min	@5min	@15min
Novice	−4.98%	−7.32%	−8.15%	−20.73%	−16.45%
Junior	−2.74%	−3.70%	−4.35%	−12.74%	−10.61%
Senior	−1.99%	−3.00%	−3.59%	−8.73%	−8.76%
Expert	−1.71%	−2.32%	−2.45%	−6.30%	−5.62%

is not a horizon-specific or metric-specific artifact but a structural improvement in lobby balance.

Comparing the two modes, Elite consistently outperforms League on every metric, with the largest relative gap on early-game economy crushing (−3.32% vs. −2.14%, a 55% larger reduction at 5 minutes). We attribute this to two reinforcing mechanisms. *First*, Elite is by far the sparsest of the modes we serve (~0.1M samples vs. tens of millions in League, see Section 5); single-mode supervision on Elite alone barely converges, and our offline results (Table 1) already showed that the marginal lift from cross-mode joint training is largest on Elite. Online, this larger predictive headroom is directly cashed in through tighter assignment filtering. *Second*, the top-expert players served by Elite sit in an extremely narrow skill band where micro-advantages compound rapidly into decisive outcomes: the same incremental gain in win-probability calibration (lower RMSE) translates into a disproportionately larger reduction in critical-imbalance events, because the crushing-rate threshold is reached by a smaller residual mismatch. DAPOE's in-network debiasing channel is most useful precisely in this regime, where the residual win-probability mass is already concentrated near 0.5 and small calibration errors are the binding constraint on online fairness.

6 Lessons Learned

Building and deploying CHAMP in a live multi-mode MOBA surfaced several non-obvious insights that we believe generalize to other applied cross-domain serving systems.

Cross-Mode Data Is Not Free Without Conditioning. The most counter-intuitive finding from our ablation study (Figure 2) is that naively pooling all modes into a vanilla OwO (the *w/o DAKE* variant) collapses Elite accuracy to 0.6039, which is markedly *worse* than the 0.6559 achieved by the same OwO trained on Elite data alone (Table 1). In other words, simply adding cross-mode supervision is not a free improvement; without an explicit conditioning signal, the heterogeneous distributions and behavioral conventions of different modes appear as noise to a shared encoder and actively degrade target-mode performance. The practical implication is that, in a multi-domain serving system, the conditioning channel should be designed *before* the corpus is scaled up: otherwise every additional domain bolted onto a shared backbone can become a liability rather than an asset.

Cold-Start Cohorts Drive the Production Lift. While Table 2 reports the aggregate gain in each deployed mode, the most important applied finding only becomes visible once we stratify the League population by skill tier. Table 3 expands the League row of Table 2 into per-tier reductions, exposing a pronounced asymmetry in who actually benefits from CHAMP in production.

The lift is sharply skill-dependent: the novice segment absorbs a 20.73% reduction in 5-minute kill crushing, roughly *twice* the all-League average (−9.00%) and more than 3× the Expert figure (−6.30%). Reading down the table, the magnitude of the reduction shrinks monotonically as the tier rises (Novice → Junior → Senior → Expert), confirming that the production payoff is concentrated precisely on the players with the thinnest in-tier behavioral histories. This inverts the common deployment intuition that the most populous, data-rich tiers should yield the largest and most stable lift, and indicates that cross-domain transfer is doing real work where single-mode models cannot fit well: the gain at Novice is essentially CHAMP imputing the missing target-mode signal from cross-mode behavior. The applied lesson is that cross-domain matchmaking methods must be evaluated cohort by cohort; aggregate lift can systematically understate the gain on the cold-start slice where the production payoff actually lives. Note further that *Elite Mode*, the mode dedicated to top-expert players, sits at the opposite end of the data spectrum (the sparsest of all modes, Section 5) and is lifted by a different mechanism (DAPOE's in-network calibration), so the two extremes of the user distribution are rescued by different parts of the same shared architecture.

Collapse the Mode-Specific Operational Surface. CHAMP deliberately collapses every mode-specific knob into one interpretable scalar, the position-satisfaction threshold τ_m in the assignment filter, and pushes all other mode dependence into shared learnable parameters (DAKE-conditioned encoders and DAPOE's debiasing slot $\mathbf{c}_m$). This was the single largest contributor to maintainability in our deployment, replacing N per-mode deployment artifacts with one.

The same collapse also reshapes online serving cost. Aggregate re-matchmaking QPS is approximately conservative across modes: when a new mode launches, players *migrate* into it rather than appear net new. After Elite Mode opened, Elite's per-mode QPS spiked while League decreased by a comparable amount, leaving global QPS essentially unchanged. A per-mode service would still have to be capacity-planned for its own independent peak, so deployed capacity would scale with the number of modes even though the shared workload does not. Routing all modes through one Dawn service makes capacity follow global QPS, avoiding this multiplicative overprovisioning cost. Finally, because Dawn adds only DAKE and the domain context slots atop OwO, load tests confirmed no significant difference in mean, P50, or P99 serving latency between CHAMP and Cupid at the same QPS.

7 Conclusion

This paper presented CHAMP, a unified cross-domain re-matchmaking framework powered by Dawn to serve multiple MOBA game modes via a single shared model. CHAMP effectively resolves cold-start and data-sparsity bottlenecks by leveraging a Domain-aware Knowledge Extractor (DAKE) for joint representation learning. Empirically, CHAMP achieves 67.73% offline accuracy and reduces 5-minute kill crushing by up to 20.73% for novice players online. Key operational guidelines include: (i) cross-mode data requires domain-aware conditioning; (ii) production gains concentrate on cold-start cohorts; (iii) collapsing mode-specific configurations to a single scalar ensures scalable maintainability.

GenAI Usage Disclosure

The authors used large language models (e.g., ChatGPT) for grammar checking and polishing of the manuscript. All technical content, experiments, and results are solely the work of the authors.

References

[1] 2023. League of Legends Live Player Count and Statistics. https://activeplayer.io/league-of-legends/. [Online].

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: a system for Large-Scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.

[3] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

[4] Junhua Chen, Wei Zeng, Junming Shao, and Ge Fan. 2019. Preference modeling by exploiting latent components of ratings. *Knowledge and Information Systems* 60 (2019), 495–521.

[5] Mingliu Chen, Adam N Elmachtoub, and Xiao Lei. 2022. Matchmaking Strategies for Maximizing Player Engagement in Video Games. In *Proceedings of the 23rd ACM Conference on Economics and Computation*. 1040–1040.

[6] Yuyan Chen, Qiang Fu, Ge Fan, Lun Du, Jian-Guang Lou, Shi Han, Dongmei Zhang, Zhixu Li, and Yanghua Xiao. 2023. Hadamard Adapter: An Extreme Parameter-Efficient Adapter Tuning Method for Pre-trained Language Models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 276–285.

[7] Mark Claypool, Jonathan Decelle, Gabriel Hall, and Lindsay O'Donnell. 2015. Surrender at 20? Matchmaking in league of legends. In *2015 IEEE Games Entertainment Media Conference (GEM)*. IEEE, 1–4.

[8] Qilin Deng, Hao Li, Kai Wang, Zhipeng Hu, Runze Wu, Linxia Gong, Jianrong Tao, Changjie Fan, and Peng Cui. 2021. Globally optimized matchmaking in online games. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2753–2763.

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

[10] Ge Fan, Biao Geng, Jianrong Tao, Kai Wang, Changjie Fan, and Wei Zeng. 2022. PPPNE: Personalized proximity preserved network embedding. *Neurocomputing* 472 (2022), 103–112.

[11] Ge Fan, Chaoyun Zhang, Junyang Chen, Paul Li, Yingjie Li, and Victor CM Leung. 2023. Improving Rating Prediction in Multi-Criteria Recommender Systems Via a Collective Factor Model. *IEEE Transactions on Network Science and Engineering* (2023).

[12] Ge Fan, Chaoyun Zhang, Junyang Chen, and Kaishun Wu. 2021. Predicting ratings in multi-criteria recommender systems via a collective factor model. In *DeMal@ the web conference*. 1–6.

[13] Ge Fan, Chaoyun Zhang, Kai Wang, and Junyang Chen. 2022. MV-HAN: A Hybrid Attentive Networks based Multi-View Learning Model for Large-scale Contents Recommendation. In *37th IEEE/ACM International Conference on Automated Software Engineering*. 1–5.

[14] Ge Fan, Chaoyun Zhang, Kai Wang, Yingjie Li, Junyang Chen, and Zenglin Xu. 2024. CUPID: Improving Battle Fairness and Position Satisfaction in Online MOBA Games with a Re-matchmaking System. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. ACM.

[15] Ge Fan, Nan Zhao, Kai Meng, Cong Luo, Yang Fu, Huiping Chu, Jialin Liu, Yuning Jiang, and Bo Zheng. 2026. Uniboost: Global Coordination with Value Alignment for Fair and Efficient Traffic Allocation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Australia) *(SIGIR '26)*. Association for Computing Machinery, New York, NY, USA, 4572–4577. https://doi.org/10.1145/3805712.3808411

[16] Linxia Gong, Xiaochuan Feng, Dezhi Ye, Hao Li, Runze Wu, Jianrong Tao, Changjie Fan, and Peng Cui. 2020. Optmatch: Optimized matchmaking via modeling the high-order interactions on the arena. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2300–2310.

[17] Yin Gu, Qi Liu, Kai Zhang, Zhenya Huang, Runze Wu, and Jianrong Tao. 2021. Neuralac: Learning cooperation and competition effects for match outcome prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4072–4080.

[18] Yin Gu, Kai Zhang, Qi Liu, Xin Lin, Zhenya Huang, and Enhong Chen. 2023. MassNE: Exploring Higher-Order Interactions with Marginal Effect for Massive Battle Outcome Prediction. In *Proceedings of the ACM Web Conference 2023*. 2710–2718.

[19] Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkill™: a Bayesian skill rating system. *Advances in neural information processing systems* 19 (2006).

[20] Juan-Agustin Hitar-Garcia, Laura Moran-Fernandez, and Veronica Bolon-Canedo. 2022. Machine learning methods for predicting league of legends game outcome. *IEEE Transactions on Games* (2022).

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[23] Alireza Khotanzad and J-H Lu. 1990. Classification of invariant image representations using a neural network. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38, 6 (1990), 1028–1038.

[24] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[25] Pan Li and Alexander Tuzhilin. 2020. DDTCDR: Deep dual transfer cross domain recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 331–339.

[26] Yuze Liu, Yunhan Wang, Tiehua Zhang, Zhishu Shen, Cheng Peng, Libing Wu, Feng Xia, and Jiong Jin. 2026. A Structure-Agnostic Co-Tuning Framework for LLMs and SLMs in Cloud-Edge Systems. In *Proceedings of the ACM Web Conference 2026*. 5667–5675.

[27] Zeye Liu, Jing Xu, Chengliang Yin, Guojing Han, Yue Che, Ge Fan, Xiaofei Li, Lixin Xie, Lei Bao, Zimin Peng, et al. 2024. Development and external validation of an artificial intelligence-based method for scalable chest radiograph diagnosis: a multi-country cross-sectional study. *Research* 7 (2024), 0426.

[28] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[29] Ilya Makarov, Dmitry Savostyanov, Boris Litvyakov, and Dmitry I Ignatov. 2018. Predicting winning team and probabilistic ratings in "Dota 2" and "Counter-Strike: Global Offensive" video games. In *Analysis of Images, Social Networks and Texts: 6th International Conference, AIST 2017, Moscow, Russia, July 27–29, 2017, Revised Selected Papers 6*. Springer, 183–196.

[30] Tom Minka, Ryan Cleven, and Yordan Zaykov. 2018. Trueskill 2: An improved bayesian skill rating system. *Technical Report* (2018).

[31] Marçal Mora-Cantallops and Miguel-Ángel Sicilia. 2018. MOBA games: A literature review. *Entertainment computing* 26 (2018), 128–138.

[32] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[33] Muhammad Farrel Pramono, Kevin Renalda, and Harco Leslie Hendric Spits Warnars. 2018. Matchmaking problems in MOBA Games. *Indonesian Journal of Electrical Engineering and Computer Science* 11, 3 (2018), 908–917.

[34] Mark Schmidt, Nicolas Le Roux, and Francis Bach. 2017. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming* 162 (2017), 83–112.

[35] Aleksandr Semenov, Peter Romov, Sergey Korolev, Daniil Yashkov, and Kirill Neklyudov. 2017. Performance of machine learning algorithms in predicting game outcome from drafts in dota 2. In *Analysis of Images, Social Networks and Texts: 5th International Conference, AIST 2016, Yekaterinburg, Russia, April 7-9, 2016, Revised Selected Papers 5*. Springer, 26–37.

[36] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One model to serve all: Star topology adaptive recommender for multi-domain CTR prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4104–4113.

[37] Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip M Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. 2021. Omninet: Omnidirectional representations from transformers. In *International Conference on Machine Learning*. PMLR, 10193–10202.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[39] Maxime Véron, Olivier Marin, and Sébastien Monnet. 2014. Matchmaking in multi-player on-line games: studying user traces to improve the user experience. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. 7–12.

[40] Kai Wang, Hao Li, Linxia Gong, Jianrong Tao, Runze Wu, Changjie Fan, Liang Chen, and Peng Cui. 2020. Match tracing: A unified framework for real-time win prediction and quantifiable performance evaluation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2781–2788.

[41] Zelong Yang, Zhufeng Pan, Yan Wang, Deng Cai, Shuming Shi, Shao-Lun Huang, Wei Bi, and Xiaojiang Liu. 2022. Interpretable Real-Time Win Prediction for Honor of Kings—A Popular Mobile MOBA Esport. *IEEE Transactions on Games* 14, 4 (2022), 589–597.

[42] Nigel Zalamea. 2022. Dota 2 The International: All TI winners throughout the years. https://www.oneesports.gg/dota2/the-international-all-ti-winners/. [Online].

[43] Chaoyun Zhang, Kai Wang, Hao Chen, Ge Fan, Yingjie Li, Lifang Wu, and Bingchao Zheng. 2022. QuickSkill: Novice Skill Estimation in Online Multiplayer Games. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3644–3653.

[44] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[45] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[46] Feng Zhu, Yan Wang, Chaochao Chen, Jun Liu, Longfei Huang, and Guanfeng Li. 2021. Cross-domain recommendation: challenges, progress, and prospects. *arXiv preprint arXiv:2103.01696* (2021).