Pareto-guided Pipeline for Distilling Featherweight AI Agents in Mobile MOBA Games

Xionghui Yang
Peking University
Beijing, China
yangxionghui@stu.pku.edu.cn

Bozhou Chen
Peking University
Beijing, China
2301111899@stu.pku.edu.cn

Yunlong Lu
Peking University
Beijing, China
luyunlong@pku.edu.cn

Yongyi Wang
Peking University
Beijing, China
wangyongyi@pku.edu.cn

Lingfeng Li
Peking University
Beijing, China
lingfengli@stu.pku.edu.cn

Lanxiao Huang
Tencent
Chengdu, China
jackiehuang@tencent.com

Lin Liu
Tencent
Chengdu, China
lincliu@tencent.com

Wenjun Wang
Tencent
Chengdu, China
jamesonwang@tencent.com

Meng Meng
Tencent
Chengdu, China
promengmeng@tencent.com

Xia Lin
Tencent
Chengdu, China
hugolin@tencent.com

Wenxin Li
Peking University
Beijing, China
lwx@pku.edu.cn

ABSTRACT

Recent advances in game AI have demonstrated the feasibility of training agents that surpass top-tier human professionals in complex environments such as Honor of Kings (HoK), a leading mobile multiplayer online battle arena (MOBA) game. However, deploying such powerful agents on mobile devices remains a major challenge. On one hand, the intricate multi-modal state representation and hierarchical action space of HoK demand large, sophisticated policy networks that are inherently difficult to compress into lightweight forms. On the other hand, production deployment requires high-frequency inference under strict energy and latency constraints on mobile platform. To the best of our knowledge, bridging large-scale game AI and practical on-device deployment has not been systematically studied. In this work, we propose a Pareto optimality guided pipeline and design a high-efficiency student architecture search space tailored for mobile execution, enabling systematic exploration of the trade-off between performance and efficiency. Experimental results demonstrate that the distilled model achieves remarkable efficiency, including an 12.4× faster inference speed (under 0.5ms per frame) and a 15.6× improvement in energy efficiency (under 0.5mAh per game), while retaining a 40.32% win rate against the original teacher model.

Full Version: The full version of this paper, including the appendix, is available on arXiv.[1]

[1]https://arxiv.org/abs/2602.07521

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/HUOT2523

KEYWORDS

Pareto Optimality; Knowledge Distillation; Game AI; MOBA Games; Mobile Deployment

ACM Reference Format:
Xionghui Yang, Bozhou Chen, Yunlong Lu, Yongyi Wang, Lingfeng Li, Lanxiao Huang, Lin Liu, Wenjun Wang, Meng Meng, Xia Lin, and Wenxin Li. 2026. Pareto-guided Pipeline for Distilling Featherweight AI Agents in Mobile MOBA Games . In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/HUOT2523

1 INTRODUCTION

The development of powerful AI agents for complex environments such as mobile multiplayer online battle arena (MOBA) games marks a major milestone in artificial intelligence. In particular, agents developed for Honor of Kings (HoK) have demonstrated the ability to defeat top-tier professional human players [41–43]. These state-of-the-art systems are typically implemented as large, deep neural networks with intricate, multi-branch architectures designed to process multi-modal state inputs and hierarchical action spaces [25]. Consequently, they demand substantial computational resources, often requiring hundreds of millions of floating-point operations (FLOPs) per inference.

A fundamental tension exists between the architectural complexity required for high performance and the stringent constraints of mobile platform—low latency, limited energy budget, and small memory footprint. Directly deploying large models on-device typically leads to unacceptable inference latency and rapid battery drain, rendering these agents impractical for real-time mobile gaming.

Conventional model compression techniques, such as knowledge distillation (KD) [14], low-rank decomposition (LRD) [33], quantization [16], and pruning [12], offer only partial solution. They

struggle to cope with the unique characteristics of complex MOBA policy networks, which combine heterogeneous components (e.g., CNNs [22, 23], LSTMs [8, 15], and Transformers [37]) under strong inter-module coupling through skip connections and feature fusion. Applying generic compression uniformly across such architectures often results in severe performance degradation, as perturbations in one component propagate throughout the system. More importantly, existing works lack a principled, reproducible pipeline for practical on-device deployment, leaving a significant methodological gap between research prototypes and deployable systems.

To address this gap, we reframe the on-device deployment problem as a multi-objective optimization task. We propose a structured, Pareto-oriented engineering pipeline to approximate the Pareto-optimal frontier between agent performance and on-device efficiency. Our approach integrates a distillation-based architecture search pipeline specifically engineered for the MOBA domain. This end-to-end methodology transforms a complex teacher agent into a deployable mobile counterpart, embedding efficiency considerations at every design stage rather than treating them as an afterthought.

Our main contributions are as follows:

- **Pareto-Guided Systematic Pipeline.** We introduce and formalize the problem of on-device deployment for large-scale multi-agent systems from a Pareto optimality [29] perspective, proposing an end-to-end engineering pipeline that systematically explores the trade-off between performance and efficiency.
- **Lightweight Student Architecture Design.** We design a high-efficiency student architecture optimized for policy distillation in mobile MOBA environments, achieving substantial computational and energy savings with only minor performance degradation.
- **Comprehensive Empirical Validation.** Through extensive experiments, we demonstrate that our method achieves a 12.4× inference speedup and a 15.6× improvement in energy efficiency while maintaining a competitive win rate against the teacher model. We further provide detailed ablations and design analyses, offering practical insights for future research into on-device game AI.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 formulates the problem and describes the HoK environment. Section 4 details our proposed methodology. Section 5 presents experimental results. Finally, Section 6 concludes the paper and discusses future research directions.

Full Version: The full version of this paper, including the appendix, is available on arXiv.[2]

2 RELATED WORK

Model compression has been a central topic in efficient deep learning, aiming to accelerate neural networks and reduce their size while maintaining competitive accuracy. Extensive research in both academia and industry has produced four major categories of methods: low-rank decomposition, pruning, quantization, and knowledge distillation. LRD methods compress networks by approximating the weight tensors of linear and convolutional layers through tensor decomposition, thereby reducing computational redundancy while preserving a degree of structural interpretability [1, 19, 21, 28]. However, their rigid structural constraints often lead to accuracy degradation in complex, highly nonlinear models [17, 19, 21]. Pruning methods, in contrast, selectively remove redundant neurons or connections to achieve sparsity. Structured pruning and automated strategies have enabled high compression rates with minimal accuracy loss [6, 7, 12, 13, 24], though excessive pruning can destabilize training and impair generalization [6, 26]. Quantization reduces the numerical precision of weights and activations, making models more hardware-friendly and improving inference efficiency [5, 11, 16, 18, 27]. Compared to LRD and pruning, quantization offers greater runtime acceleration but introduces numerical instability that must be mitigated through robust quantization-aware training [18, 27, 38]. Finally, KD bridges the performance gap left by structural compression methods by transferring knowledge from a large teacher model to a compact student model [3, 14, 31, 45]. KD preserves accuracy without imposing architectural constraints, yet its effectiveness critically depends on the alignment between teacher and student network designs.

KD has proven effective across domains such as computer vision and natural language processing [9, 10, 20, 34, 40, 46]. Its reinforcement learning counterpart, policy distillation (PD), extends this concept to compress deep reinforcement learning agents [4, 30, 32, 35, 36, 39, 44]. However, most PD studies to date have been confined to relatively simple benchmarks such as Atari games [2], which lack the continuous dynamics, partial observability, and multi-agent coordination present in MOBA environments. In contrast, the HoK environment poses a substantially greater challenge. Unlike Atari, which uses low-dimensional pixel inputs and a small discrete action space (18 actions), HoK requires real-time reasoning over multimodal state representations, a hierarchical action space with over 22 million possible actions, and partial observability (e.g., fog of war) across multiple agents [25]. These properties make HoK an ideal testbed for studying large-scale policy compression under realistic and demanding conditions.

Despite progress in model and policy compression, most existing approaches overlook the stringent requirements of mobile environments, where policies must support high-frequency, low-latency, and energy-efficient inference under severe hardware constraints. These challenges are particularly acute in mobile MOBA games, which require consistent responsiveness and prolonged device operation. Critically, current compression approaches provide only partial solutions and lack a principled, reproducible pipeline for on-device deployment. This gap leaves a disconnect between research prototypes and deployable, resource-efficient systems. Consequently, distilling large-scale policy models from complex environments like HoK into computationally and energy-efficient student agents for on-device deployment remains a challenging and underexplored problem.

3 PRELIMINARIES

This section formalizes the core on-device deployment challenge and introduces the experimental environment. We first frame the problem using multi-objective optimization and then describe the HoK environment that serves as our testbed.

[2] https://arxiv.org/abs/2602.07521

3.1 Problem Formulation

The deployment of complex game AI agents onto mobile devices presents a fundamental conflict: the *computational demand* for high-performance agents versus the stringent *resource constraints* of mobile platforms. We formalize this challenge as a multi-objective optimization problem, where the goal is to find an optimal trade-off between competing objectives.

Consider a design point d within a vast design space D, encompassing all possible model architectures and their configurations. We evaluate d along two primary axes:

- **Performance ($P(d)$)**: The capability of the agent, quantified as its win rate against a benchmark.
- **Efficiency ($E(d)$)**: The on-device efficiency, quantified by minimizing a set of resource consumption metrics:
  - Inference Latency ($L(d)$): Time per decision.
  - Energy Consumption ($B(d)$): Energy used per decision.
  - Peak Memory Usage ($M(d)$): Maximum RAM during inference.
  - Model Size ($S(d)$): Storage footprint.

The ideal solution would simultaneously maximize $P(d)$ while minimizing all efficiency metrics. However, these objectives are inherently competing. This tension leads us to adopt the concept of **Pareto optimality** to define the best possible compromises.

Definition 3.1 (Pareto Dominance). A point d_i *Pareto dominates* another point d_j (denoted $d_i \prec d_j$) if and only if d_i is strictly better in at least one objective and no worse in all others. Formally, for the primary objectives (P, L, B, M, S):

$$\begin{aligned}[P(d_i) \geq P(d_j)] \wedge [L(d_i) \leq L(d_j)] \wedge [B(d_i) \leq B(d_j)] \\ \wedge [M(d_i) \leq M(d_j)] \wedge [S(d_i) \leq S(d_j)]\end{aligned} \tag{1}$$

with at least one inequality being strict.

Definition 3.2 (Pareto Optimality and Pareto Frontier). A point d^* is *Pareto-optimal* (non-dominated) if no other point in D dominates it. The set of all Pareto-optimal points forms the *Pareto frontier*, representing the best achievable trade-offs.

Given the intractability of finding the global Pareto frontier for D, our practical goal is to procedurally generate a restricted design subspace $D' \subset D$ and identify one or more design points $\hat{d} \in D'$ that are non-dominated and lie on or near the *empirical Pareto frontier* of D'.

3.2 HoK Environment

We employ the HoK 3v3 game mode as our experimental testbed, a leading mobile MOBA game that presents a challenging benchmark for multi-agent decision-making under real-time constraints. In this environment, two teams of three agents (heroes) compete to destroy the opponent's base, requiring sophisticated coordination and decision-making at both the tactical and strategic levels.

The environment is characterized by two elements that define the complexity:

High-Dimensional, Multi-Modal Observation Space. The observation space is represented by a 13,758-dimensional vector that comprehensively encodes the game state at each time step. This representation is decomposed into three hero-specific observations, each of 4,586 dimensions, capturing:

- Individual hero attributes (e.g., health, skill cooldowns, position),
- Environmental obstacles and terrain features,
- The dynamic status of all game units (e.g., creeps, turrets, monsters).

These features are systematically organized into seven distinct categories, providing agents with a holistic perception necessary for strategic decision-making.

Hierarchical Action Space with Validity Constraints. The action space adopts a two-level hierarchical structure to enable fine-grained control:

- **Level 1 (Action Selection):** Chooses a behavior type from 13 discrete actions, including 2 No-operations, Move, Normal Attack, and 9 hero-specific skills.
- **Level 2 (Parameter Specification):** Determines the parameters for the chosen action through three sub-mechanisms:
  - Directional Control: 25 discrete movement directions.
  - Positional Control: 42 possible skill offsets along the x-axis and z-axis.
  - Target Selection: 7 target types with up to 39 possible targets.

To ensure that actions are contextually valid, the environment incorporates *legal action masks* and *sub-action masks* [41, 43], which dynamically restrict the available choices to those permissible in the current game state. These mechanisms are critical for stabilizing training and ensuring the practicality of the learned policies.

The complexity of the state and action spaces, coupled with intrinsic challenges such as continuous dynamics, partial observability, and multi-agent coordination, makes HoK 3v3 not only a demanding benchmark but also an ideal platform for studying the performance-efficiency trade-offs that underpin practical on-device deployment.

4 METHODOLOGY

4.1 Overview

Following the problem formulation in Section 3, which frames on-device deployment challenge as a multi-objective optimization targeting Pareto optimality, we introduce a systematic multi-stage pipeline. Our overarching goal is to transform a computationally intensive teacher agent into a featherweight student agent suitable for on-device deployment while optimally navigating the performance-efficiency trade-off.

The proposed pipeline, depicted in Figure 1, comprises five stages:

(1) **Architecture Design**: Profiling the teacher network to identify computational bottlenecks and design a mobile-efficient student blueprint with tunable hyperparameters.
(2) **Architecture Search**: Conducting a systematic search over the designed space, distilling each candidate, and evaluating its performance-efficiency profile to progressively construct the Pareto frontier.
(3) **Distillation Training**: Transferring policy knowledge from the teacher to the candidate students via knowledge distillation.

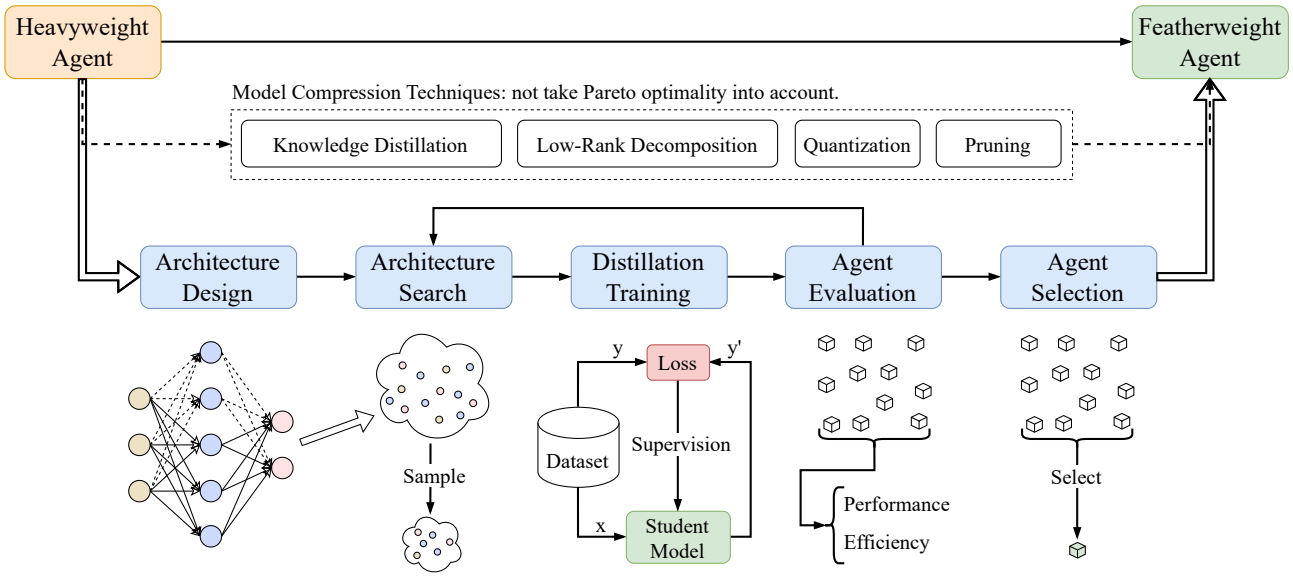

Figure 1: Overview of the proposed Pareto optimality driven distillation pipeline. The process integrates architecture design, automated search, distillation training, evaluation, and final selection, forming an end-to-end pipeline that jointly optimizes model performance and efficiency for on-device deployment.

(4) **Agent Evaluation**: Assessing candidates using key performance (e.g., win rate) and efficiency (e.g., latency, energy) metrics.
(5) **Agent Selection**: Selecting the optimal agent from the empirical Pareto frontier based on predefined constraints.

4.2 Architecture Design

We commence with a rigorous profiling of the teacher model to identify computational bottlenecks. The profiling results, summarized in Table 1, provide a quantitative breakdown of the computational cost distribution across different modules.

Table 1: Computational profile of the teacher model. The Encoder and LSTM modules are identified as the primary bottlenecks.

Module	FLOPs (M)	Params (M)	FLOPs(%)	Params(%)
All	681.84	16.43	100.00	100.00
Encoder	**560.92**	**4.83**	**82.27**	**29.39**
LSTM	**8.40**	**8.40**	**1.23**	**51.11**
CNN	5.06	0.004	0.74	0.02
Others	107.45	3.20	15.76	19.47

Our analysis reveals that the Encoder and LSTM modules collectively account for 83.49% of the computational cost and 80.50% of the model parameters, despite the LSTM contributing negligible FLOPs. This stark imbalance establishes these two modules as the primary bottlenecks and the most promising targets for architectural optimization.

Guided by this profiling, we design a featherweight and effective student architecture. As shown in Figure 2, the architecture is tailored to address the challenges of HoK's multi-modal inputs and hierarchical action space while preserving the core decision-making logic of the teacher model. Its core components are structured as follows:

(1) **Feature Extraction Module**: Processes each hero's multi-modal observations in parallel using a combination of CNNs and MLPs.
(2) **Feature Fusion Module**: In the teacher model, the Encoder and LSTM handle cross-modal and cross-temporal fusion, respectively. In our student model, both functions are implemented as MLPs (see the red and blue dashed boxes in Figure 2).
(3) **Triplet Max-Fusion Gate**: Facilitates cooperation among heroes by sharing and max-pooling public embeddings across the team (indicated by the cyan dotted line in Figure 2).
(4) **Prediction Heads**: Comprise specialized MLPs that output distributions for the hierarchical action space.

To further refine the student model, we parameterize the architecture using a set of tunable hyperparameters, which serves as the foundation for architecture search in the subsequent stage.

4.3 Architecture Search

Building upon the macro-architecture defined in Section 4.2, we conduct a systematic search within our architectural space to identify high-performing candidate agents. To manage the search complexity and ensure a balanced distribution of candidates, we employ a constrained sampling strategy. The space is decomposed into seven core building blocks, each parameterized by its layer count

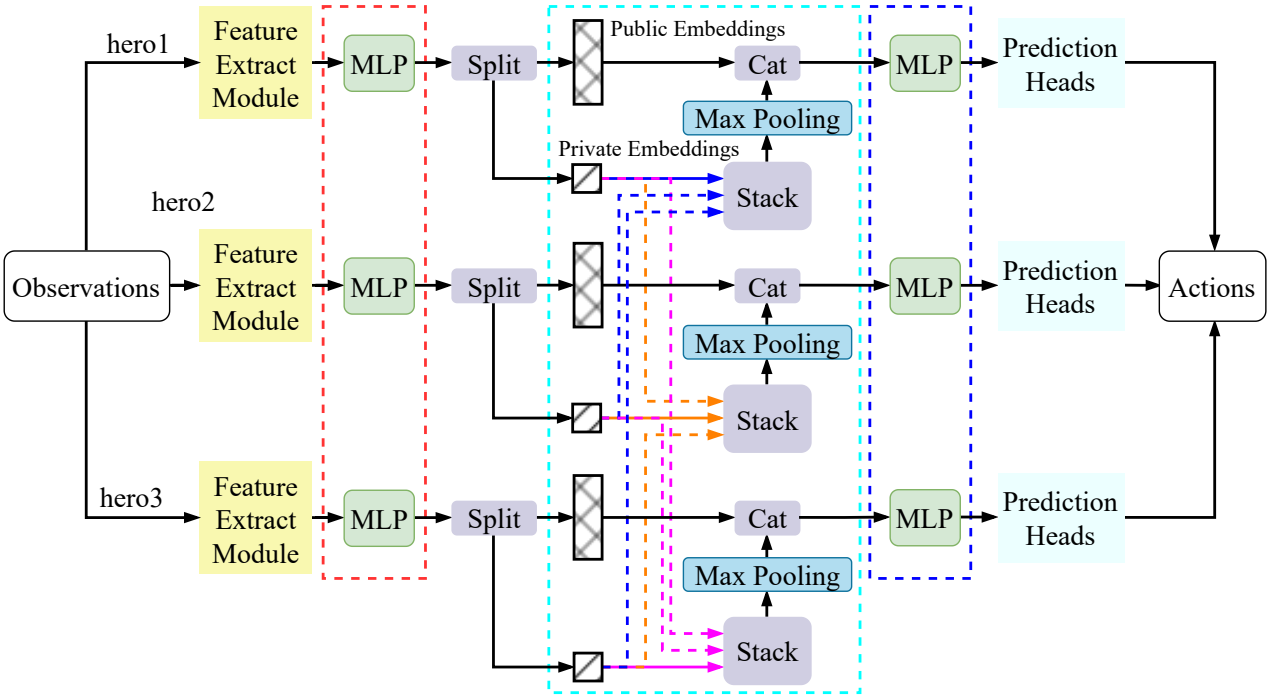

Figure 2: Overview of the featherweight student architecture. The design streamlines the teacher model by removing the attention-based components and the LSTM module, and adopting lightweight MLP structures. The red and blue dashed boxes indicates the simplified feature fusion module, while the cyan dashed boxes highlights the triplet max-fusion gate that enables team-level cooperation. This architecture efficiently processes multi-modal inputs and produces hierarchical action distributions.

and per-layer feature dimension n_i. Candidate architectures are sampled from these parameter ranges, subject to a global compute budget of $[1\%, 20\%]$ of the teacher's FLOPs.

We discretize the FLOPs range into 20 uniform intervals and sample candidate models within each interval. This structured approach ensures efficient and representative coverage of the performance-efficiency design space, providing a diverse population for subsequent Pareto-optimal selection.

Our search procedure is outlined in Algorithm 1. For each architecture $q \in \mathcal{S}$, we obtain its distilled weights θ_q and an evaluation vector $\mathbf{m}_q \in \mathbb{R}^d$ (e.g., win rate, latency, energy). The accumulated results after round t form the solution set $\mathcal{D}_t = \{(q, \mathbf{m}_q)\}$, from which we derive the Pareto frontier $\mathcal{P}_t$.

4.4 Distillation Training

Knowledge transfer from the teacher to the student is formalized as a policy distillation task. The core objective is to align the student's action distribution with the teacher's policy, enabling the lightweight model to retain expert-level decision-making capability while achieving significant efficiency gains.

Given an input state s sampled from replay dataset $\mathcal{B}$, let $\hat{z}$ and z be the raw logits of the teacher and student, respectively. The temperature-scaled probability distributions are:

$$\hat{p}_i = \frac{\exp(\hat{z}_i/\tau)}{\sum_j \exp(\hat{z}_j/\tau)}, \quad p_i = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}, \tag{2}$$

where τ controls the smoothness of the output distribution and aids knowledge transfer.

The student policy π_θ is optimized to minimize the divergence from the teacher policy π_T over the dataset:

$$\min_\theta \mathbb{E}_{s\sim\mathcal{B}} \left[\mathcal{D} \left(\pi_T(\cdot|s) \parallel \pi_\theta(\cdot|s) \right) \right], \tag{3}$$

For the hierarchical action space of HoK with $\mathcal{A} = 5$ sub-distributions, we define the loss as the average KL divergence across all sub-actions:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{A}|} \sum_{k=1}^{|\mathcal{A}|} \mathbb{E}_{s\sim\mathcal{B}} \left[\mathrm{KL} \left(\pi_T^{(k)}(\cdot|s) \,\middle\|\, \pi_\theta^{(k)}(\cdot|s) \right) \right], \tag{4}$$

where distributions are softened with τ as in Equation 2.

This distillation process is applied to each candidate architecture sampled during the search stage, training a diverse population of models for the subsequent Pareto-optimal selection.

4.5 Agent Evaluation and Selection

In the evaluation phase, Each candidate model is evaluated across the dual objectives of **performance** and **efficiency**, using the

Algorithm 1: Pareto-Guided Architecture Search with Distillation

Input: Search space $\mathcal{S}$; teacher model π_T; architecture sampler $Q(\cdot \mid \mathcal{S})$; batch size k; max iterations R; threshold $\varepsilon > 0$.

Output: Pareto-optimal architecture set $\mathcal{A}^\star$.

$\mathcal{D}_0 \leftarrow \varnothing, \mathcal{P}_0 \leftarrow \varnothing$.

for $t \leftarrow 1$ **to** R **do**

`// (a) sample architectures`

$\mathcal{A}_t \leftarrow \{\, q \mid q \sim Q(\cdot \mid \mathcal{S}) \,\}$

foreach $q \in \mathcal{A}_t$ **do**

`// (b) distillation training (teacher` π_T`)`

Train student $\pi_q(\cdot|s;\theta)$ by minimizing $\mathcal{L}(\theta) = \mathrm{KL}\left(\pi_T(\cdot|s) \,\middle\|\, \pi_q(\cdot|s;\theta)\right)$, obtain θ_q.

`// (c) evaluation`

Measure $\mathbf{m}_q$ on validation set and system metrics (FLOPs/accuracy/win rate/latency/energy, etc.).

`// (d) update the solution set`

$\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(q, \mathbf{m}_q)\}$

`// (e) compute new Pareto frontier`

$\mathcal{P}_t \leftarrow \{\, \mathbf{u} \in \{\mathbf{m}_q\} \mid \nexists \mathbf{v} \text{ s.t. } \mathbf{v} \preceq \mathbf{u} \;\&\; \mathbf{v} \neq \mathbf{u} \,\}$

`// (f) stopping by frontier discrepancy`

(Example metric: symmetric Chamfer distance in normalized objective space)

$$\Delta_t \leftarrow \frac{1}{|\mathcal{P}_t|} \sum_{\mathbf{u} \in \mathcal{P}_t} \min_{\mathbf{v} \in \mathcal{P}_{t-1}} \|\mathbf{u} - \mathbf{v}\|_2 \;+\; \frac{1}{|\mathcal{P}_{t-1}|} \sum_{\mathbf{v} \in \mathcal{P}_{t-1}} \min_{\mathbf{u} \in \mathcal{P}_t} \|\mathbf{v} - \mathbf{u}\|_2$$

if $\Delta_t < \varepsilon$ **then**

$t^\star \leftarrow t$; **break**

$\mathcal{A}^\star \leftarrow \{\, q \mid (q, \mathbf{m}_q) \in \mathcal{D}_{t^\star},\ \mathbf{m}_q \in \mathcal{P}_{t^\star} \,\}$.

metrics described in Section 5.1, encompassing win rate, inference latency, energy consumption, peak memory usage, and model size.

The final agent is selected from the empirical Pareto frontier according to the principle of Pareto optimality: we choose the non-dominated candidate that satisfies predefined on-device deployment constraints (e.g., win rate > 40% , latency < 0.5 ms). This constraint-driven approach ensures the selected agent optimally balances performance and efficiency for practical on-device deployment.

5 EXPERIMENTS

5.1 Experimental Setup

Environment. Our experiments are conducted within the HoK 3v3 game mode on the "Changping Attack and Defense" map. To ensure a robust and comprehensive evaluation, we employ a fixed set of eight distinct hero compositions. These compositions are systematically generated from a pool of six popular heroes (e.g., Zhao Yun, Li Yuanfang, Zhuge Liang), covering a variety of strategic roles and team synergies. This approach mitigates potential evaluation bias from testing on a single, favorable composition, thereby providing a more reliable assessment of agent performance across diverse in-game scenarios.

Dataset. The dataset for knowledge distillation is constructed through self-play between two identical instances of the teacher model, generating a corpus of 2.02 million state-action transitions. This dataset is partitioned into 2 million samples for training and 20,000 for validation.

Hyper-parameters Configuration. The hyper-parameters used for distillation training are set as follows. We adopt the Adam optimizer with a learning rate of 2×10^{-4}, where the first- and second-order momentum coefficients are set to $\phi_1 = 0.9$ and $\phi_2 = 0.999$, respectively, and the numerical stability constant is fixed to $\epsilon = 10^{-8}$. Training is performed with a batch size of 512 for a total of 1×10^6 optimization steps. For knowledge distillation, the temperature parameter is set to $\tau = 4$.

Baseline Methods. To rigorously evaluate our approach, we compare it against several representative baselines spanning different compression paradigms: (1) **Teacher Model**: The original, uncompressed model serves as the performance upper bound, representing the maximum achievable capability before any compression is applied. (2) **LRD**: Applies low-rank matrix factorization to the weight tensors of the teacher model, approximating the original parameters with lower-rank representations. Parameter count and computational complexity are reduced while the fundamental structure of the original network is preserved as much as possible. (3) **Quantization**: Reduces the numerical precision of weights and activations from 32-bit floating point to 8-bit integers (INT8) through post-training quantization. This technique primarily decreases model size and improves inference speed on supported hardware, though potential precision loss may degrade policy quality. (4) **Pruning**: Removes connections with the smallest magnitudes based on a global threshold via structured weight pruning. Sparsity in the weight matrices is increased, leading to reductions in parameter count and computational cost, albeit with possible loss of structural integrity and performance. (5) **Linear Model**: An extremely simple architecture (a MLP) serves as the efficiency upper bound, representing the theoretical maximum efficiency achievable while typically exhibiting negligible performance.

Metrics. Evaluation is conducted from two primary perspectives to holistically assess the agent's quality: (1) **Performance ($P(d)$)**: The key performance metric is the *Win Rate* against the original teacher model, calculated as the average win rate over three independent runs of 1,000 games each to ensure statistical significance. (2) **Efficiency ($E(d)$)**: Metrics are measured on mobile platforms and include: *Inference Latency ($L(d)$)*: Average time required for a single decision. *Energy Consumption ($B(d)$)*: Total energy used over 5,000 inferences, corresponding to a typical game length. *Peak Memory Usage ($M(d)$)*: Maximum RAM allocated during inference. *Model Size ($S(d)$)*: Storage footprint of the deployed model.

To ensure reliable and reproducible measurements of on-device efficiency, all metrics are evaluated on a fixed set of 5,000 frames randomly sampled from the validation set. The evaluation process encompasses two stages: (1) a one-time initialization phase to load all necessary components (model, frame data, libraries), followed by (2) 5,000 consecutive inferences. The *Inference Latency* is reported as the average time per inference across the entire process (stages 1 and 2 combined), while the *Energy Consumption* is measured for the complete sequence. Since the initialization overhead is negligible compared to the total computation, its impact is amortized across

the inferences, ensuring the metrics accurately reflect sustained performance. This controlled approach mitigates fluctuations inherent in live game environments. All measurements were repeated three times, and the mean and standard deviation are reported.

Hardware Configuration. All model training and distillation procedures were performed on a high-performance computing cluster consisting of four servers. Each server was equipped with a 112-core Intel Xeon Gold 6348 CPU (2.60 GHz), 378 GB of RAM, and eight NVIDIA GeForce RTX 3090 GPUs. The on-device efficiency was evaluated on an iQOO 12 smartphone, equipped with a Qualcomm Snapdragon 8 Gen 3 system-on-chip (which integrates an Adreno GPU and Hexagon NPU) and 16 GB of RAM.

Wall-Clock Time. A complete run of the proposed pipeline, including architecture search, distillation, and evaluation, completes within approximately 10-15 days on the hardware setup described above.

5.2 The Performance of Proposed Pipeline

This section presents a macro-level evaluation to answer the fundamental question: does our proposed pipeline achieve a superior trade-off between agent performance and on-device efficiency? We compare our final Featherweight Agent (FA) against the original teacher model and all baseline methods across the comprehensive set of metrics defined in Section 3.1.

Macro-Level Results. The comprehensive benchmarking results are summarized in Table 2. Our FA achieves a competitive win rate of **40.32%**, well preserving the decision-making ability of the teacher model. Crucially, this performance is attained while achieving a dramatic reduction in resource consumption. Compared to the teacher model, FA reduces inference latency by 91.8% (from 5.41ms to 0.44ms) and energy consumption by 93.6% (from 7.62mAh to 0.49mAh). Furthermore, while Low-Rank Decomposition and Quantization maintain reasonable performance (~40%), their efficiency gains are modest, achieving less than 50% latency reduction. In contrast, Pruning achieves high efficiency but suffers a catastrophic performance collapse. Our FA, by fundamentally re-architecting the model to be mobile-native, achieving both high performance and superior efficiency.

Pareto Frontier Analysis. The performance-efficiency trade-off is visually articulated in Figure 3, which plots the empirical Pareto frontier. Each point represents a candidate model's win rate versus its inference latency. The plot clearly shows that our FA resides on the estimated Pareto frontier, demonstrating that it is a non-dominated solution. In contrast, the standard compression baselines lie deep within the interior of the plot, indicating they are strictly inferior in the trade-off space (i.e., other points offer better or equal performance with lower latency). This provides compelling visual evidence that our approach successfully identifies a superior operating point.

5.3 Comparison with Standard Compression Baselines

This section validates the superiority of our pipeline by contrasting it against standard, architecture-preserving compression techniques: Low-Rank Decomposition, Quantization, and Pruning. As quantified in Table 2 and visually articulated in Figure 3, models

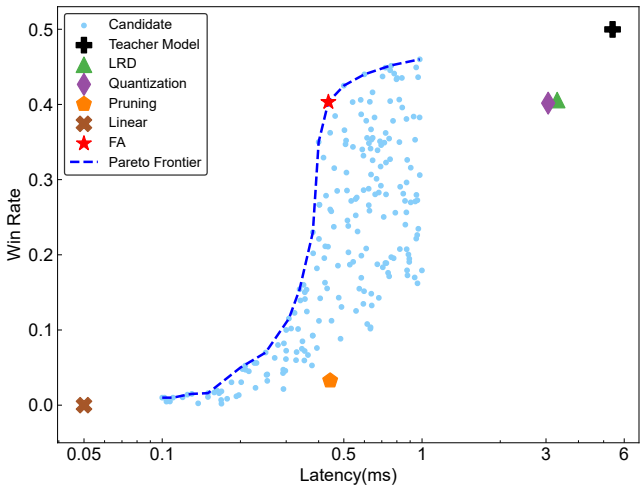

Figure 3: The empirical Pareto frontier characterizing the performance-efficiency trade-off. The frontier is derived from the systematic assessment of all candidate agents, illustrating the set of non-dominated solutions where no improvement can be made in one objective without deteriorating the other.

compressed by these standard techniques reside far from the empirical Pareto frontier. While they reduce model size and latency, they incur a severe performance penalty. For instance, the pruned model's win rate collapses to 3.27%, rendering it practically useless, despite its low latency (0.44ms). LRD and Quantization maintain ~40% win rates but achieve only modest latency reductions (~3ms).

The failure of each method stems from fundamental limitations when applied to complex policy networks: Pruning causes irreversible structural damage by removing critical pathways in the highly non-linear policy network, Quantization leads to information degradation as aggressive precision loss (e.g., INT8) disrupts the delicate policy function, Low-Rank Decomposition suffers from representational inadequacy, as low-rank constraints poorly approximate the full-rank functions needed for sophisticated policies.

In summary, while standard compression techniques are inherently architecture-preserving, merely compressing an existing, inefficient structure, our approach is fundamentally architecture-agnostic. By undertaking a ground-up, mobile-native redesign based on the teacher model, our Featherweight Agent successfully preserves the core decision-making capabilities while achieving superior efficiency. These results underscore a critical insight: when the source architecture is inherently ill-suited for the target platform, post-hoc compression is fundamentally limited in its ability to yield an optimal solution. This work demonstrates that a mobile-native, ground-up architectural redesign is a substantially more effective strategy.

5.4 Ablation Study

We conduct a comprehensive set of ablation studies to rigorously validate the necessity and optimality of each key design choice in our Featherweight Agent. The results, summarized in Table 3,

Table 2: Comprehensive benchmarking results comparing the performance and efficiency of the teacher model, standard compression baselines, and our Featherweight Agent. Results demonstrate that FA achieves a superior trade-off, dominating other methods by simultaneously maintaining competitive performance (40.32%win rate) while achieving dramatic efficiency improvements (12.4× faster inference speed and 15.6× improvement in energy efficiency than the teacher). Metrics are reported as mean values with standard deviations in parentheses.

Method	FLOPs(M)	Parameters(M)	Win Rate(%)	Time(ms)	Energy(mAh)	Memory(MB)	Size(MB)
Teacher Model	681.84	16.43	49.98 (±4.59)	5.411 (±0.188)	7.62 (±0.27)	230.35 (±1.94)	32.7
LRD	412.36	11.56	40.58 (±1.39)	3.312 (±0.104)	4.56 (±0.18)	181.22 (±0.98)	19.05
Quantization	380.18	10.13	40.17 (±1.18)	3.058 (±0.096)	4.21 (±0.13)	176.39 (±0.93)	14.23
Pruning	45.22	6.22	3.27 (±0.14)	0.443 (±0.005)	0.52 (±0.01)	156.62 (±0.17)	12.36
Linear	4.43	2.22	0.00 (±0.00)	0.05 (±0.001)	0.06 (±0.01)	79.63 (±0.06)	3.63
FA (ours)	**45.74**	**5.96**	**40.32 (±1.03)**	**0.436 (±0.006)**	**0.49 (±0.01)**	**152.71 (±0.19)**	**9.89**

are analyzed to demonstrate how each alteration leads to a strictly inferior trade-off.

Table 3: Ablation study results. Key metrics include win rate against the teacher, inference latency, and energy consumption.

Model Variant	Win Rate (%)	Latency (ms)	Energy (mAh)
Teacher Model	49.98 (±4.59)	5.411 (±0.188)	7.62 (±0.27)
FA (ours)	**40.32 (±1.03)**	**0.436 (±0.006)**	**0.49 (±0.01)**
FA (w/ LSTM)	40.67 (±0.91)	0.578 (±0.012)	0.72 (±0.02)
FA (w/ Encoder)	45.21 (±1.64)	4.656 (±0.128)	6.55 (±0.21)
FA (w/ Large)	44.00(±1.27)	0.601 (±0.007)	0.68 (±0.02)
FA (w/ Small)	23.02(±0.98)	0.380 (±0.005)	0.43 (±0.01)

5.4.1 Impact of Different Network Modules. To validate our architecture efficiency, we reintroduced the original complex modules into the FA. Reinstating the LSTM (FA w/ LSTM) increases latency by 32.6% (0.436ms to 0.578ms) and energy consumption by 46.9%, with no meaningful win rate improvement. Reinstating the Encoder (FA w/ Encoder) has a more severe impact, increasing latency by 10.7× and energy consumption by 13.4×, despite a modest win rate increase to 45.21%. These results demonstrate the reasonablility of our architecture design. The original complex modules are indeed computational bottlenecks, and replacing them with simpler alternatives is essential for on-device deployment.

5.4.2 Effect of Model Scale. An ablation study on the overall model scale validates the optimality of our selected architecture (FA). The results, described in Table 3, demonstrate the performance-efficiency trade-off in the vicinity of the Pareto frontier.

Scaling up the model (FA w/ Large) yields a negligible performance gain of only 3.68% in win rate, but incurs a substantial efficiency cost, with a 37.8% increase in latency. Conversely, scaling down the model (FA w/ Small) achieves a minimal latency reduction of 12.8% at the expense of a significant 42.9% performance penalty.

Our analysis reveals that the chosen FA operates near a knee point on the empirical Pareto frontier. At this region, the trade-off curve exhibits a characteristic sharp bend: reducing the model scale beyond this point results in a disproportionate drop in performance for a minimal efficiency gain, while increasing the model scale leads to diminishing returns in performance accompanied by a steep efficiency loss. This confirms that FA represents a balanced operating point where the marginal utility of scaling is nearly equilibrated, providing the most favorable compromise for on-device deployment.

5.4.3 Summary. Collectively, these studies validate our pipeline: architectural design enables efficiency and the chosen model scale optimally balances both objectives. Any deviation yields a strictly inferior solution, demonstrating the necessity of each design choice.

6 CONCLUSION AND FUTURE WORK

This paper addresses the fundamental challenge of deploying computationally intensive MOBA game AI agents on resource-constrained mobile devices. We present a systematic pipeline that reframes on-device deployment as a multi-objective optimization problem, introducing a Pareto-guided distillation pipeline coupled with an efficient, mobile-native student architecture. Our approach enables principled exploration of the performance-efficiency trade-off, producing agents that achieve substantial practical improvements while maintaining competitive gameplay ability.

Experimental results demonstrate the effectiveness of our Pareto-optimality driven approach in balancing the performance and efficiency for real-world on-device deployment scenarios.

While our approach focuses on the complex Honor of Kings environment, the proposed methodology establishes a general pipeline for efficient policy compression that can be adapted to other real-time decision-making domains with stringent resource constraints. This work represents a meaningful advancement in bridging the gap between large-scale game AI research and practical on-device deployment.

In future work, we plan to extend our study toward more general mobile game environments and real-time decision-making tasks, exploring efficient search strategies and hardware-aware co-design to further enhance scalability and robustness.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to TiMi L1 Studio for their valuable support in providing services through the Tencent AI Arena platform (https://tencentarena.com).

REFERENCES

[1] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590* (2019).

[2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research* 47 (2013), 253–279.

[3] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. 2017. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems* 30 (2017).

[4] Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. 2019. Distilling policy distillation. In *The 22nd international conference on artificial intelligence and statistics*. PMLR, 1331–1340.

[5] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems* 35 (2022), 30318–30332.

[6] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[7] Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*. PMLR, 10323–10337.

[8] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[9] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. MiniLLM: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543* (2023).

[10] Gousia Habib, Sheikh Musa Kaleem, Tufail Rouf, Brejesh Lall, et al. 2024. A Comprehensive Review of Knowledge Distillation in Computer Vision. *arXiv preprint arXiv:2404.00936* (2024).

[11] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[12] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*. 784–800.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

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

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2018. Quantized neural networks: Training neural networks with low precision weights and activations. *journal of machine learning research* 18, 187 (2018), 1–30.

[17] Yerlan Idelbayev and Miguel A Carreira-Perpinán. 2020. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8049–8059.

[18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713.

[19] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866* (2014).

[20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* (2019).

[21] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530* (2015).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 2002. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (2002), 2278–2324.

[24] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[25] Lin Liu, Jianzhun Shao, Xinkai Chen, Yun Qu, Boyuan Wang, Zhenbin Ye, Yuexuan Tu, Hongyang Qin, Yang Jun Feng, Lin Lai, Yuanqin Wang, Meng Meng, Wenjun Wang, Xiyang Ji, QIANG FU, Lanxiao Huang, Minwen Deng, Yang Wei, Houqiang Li, Wengang Zhou, Ning Xie, Xiangyang Ji, Lvfang Tao, Lin Yuan, Juchao Zhuo, YANG GUANG, and Deheng Ye. 2023. HoK3v3: an Environment for Generalization in Heterogeneous Multi-agent Reinforcement Learning. https://openreview.net/forum?id=calKOSmBxj

[26] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (2018).

[27] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*. PMLR, 7197–7206.

[28] Alexander Novikov, Pavel Izmailov, Valentin Khrulkov, Michael Figurnov, and Ivan Oseledets. 2020. Tensor train decomposition on tensorflow (t3f). *Journal of Machine Learning Research* 21, 30 (2020), 1–7.

[29] Vilfredo Pareto. 2014. *Manual of political economy: a critical and variorum edition*. OUP Oxford.

[30] Xinghua Qu, Yew Soon Ong, Abhishek Gupta, Pengfei Wei, Zhu Sun, and Zejun Ma. 2022. Importance prioritized policy distillation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1420–1429.

[31] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* (2014).

[32] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295* (2015).

[33] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6655–6659.

[34] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[35] Giacomo Spigler. 2024. Proximal Policy Distillation. *arXiv preprint arXiv:2407.15134* (2024).

[36] Yuxiang Sun and Pooyan Fazli. 2019. Real-time policy distillation in deep reinforcement learning. *arXiv preprint arXiv:1912.12630* (2019).

[37] 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).

[38] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8612–8620.

[39] Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. 2024. Rldg: Robotic generalist policy distillation via reinforcement learning. *arXiv preprint arXiv:2412.09858* (2024).

[40] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116* (2024).

[41] Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. 2020. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 621–632.

[42] Deheng Ye, Guibin Chen, Peilin Zhao, Fuhao Qiu, Bo Yuan, Wen Zhang, Sheng Chen, Mingfei Sun, Xiaoqian Li, Siqin Li, et al. 2020. Supervised learning achieves human-level performance in moba games: A case study of honor of kings. *IEEE transactions on neural networks and learning systems* 33, 3 (2020), 908–918.

[43] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. 2020. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 6672–6679.

[44] Xinqiang Yu, Chuanguang Yang, Chengqing Yu, Libo Huang, Zhulin An, and Yongjun Xu. 2024. Online Policy Distillation with Decision-Attention. In *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[45] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. 2021. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4388–4403.

[46] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461* (2023).