Garrison: A High-Performance GPU-Accelerated Inference System for Adversarial Ensemble Defense

Yan Wang[1,2], Xingbin Wang*[1], Zechao Lin[1,2], Yulan Su[1,2], Sisi Zhang[1], Rui Hou[1] and Dan Meng[1]
[1]Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS
[2]School of Cyber Security, University of Chinese Academy of Sciences

ABSTRACT

In the face of huge threats from adversarial attacks, developing an efficient defense mechanism is crucial for deep learning systems. Adversarial ensemble defense method is one of the most effective techniques for defending against adversarial attacks, which constructs ensembles of multiple DNNs to improve model's robustness. However, deploying ensemble defense methods on existing DNN inference systems is inefficient and impractical due to their dynamics and randomness. To this end, we propose an inference system for adversarial ensemble defense called *Garrison*, which can deliver robust and low-latency predictions using Multi-Instance GPUs. Garrison employs a multi-granularity GPU partitioning strategy, optimizing hardware utilization by capitalizing on the intrinsic heterogeneity of GPUs. It also integrates a reinforcement learning-based scheduling mechanism, enabling random ensemble of diverse defense models to enhance robustness while maintaining bounded latency. Our evaluations show that Garrison can improve adversarial robustness by up to 24.5%, while accelerating ensemble inference by up to 6.6× compared to the state-of-the-art inference framework.

ACM Reference Format:
Yan Wang[1,2], Xingbin Wang*[1], Zechao Lin[1,2], Yulan Su[1,2], Sisi Zhang[1], Rui Hou[1] and Dan Meng[1] . 2024. Garrison: A High-Performance GPU-Accelerated Inference System for Adversarial Ensemble Defense. In *Proceedings of 61st ACM/IEEE Design Automation Conference (DAC'24).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3654810

1 INTRODUCTION

Deep neural networks (DNNs) are vulnerable to adversarial examples, which pose significant security risks. Various adversarial defense methods have been proposed to improve individual models' robustness [1], but single models often struggle to defend against adversarial attacks effectively, resulting in limited robustness. Adversarial ensemble defense (AED) [2, 3] methods outperform single-model methods by randomly constructing diverse ensembles during runtime. The ensemble applies a decision rule (e.g., weighted vote) to combine multiple models' inference results to produce a robust output, making it more challenging for adversarial examples to deceive multiple DNNs than a single DNN. By leveraging dynamism, AED also raises the bar for attackers attempting to steal model information through queries [4], leading to enhanced security against adversarial attacks.

*Xingbin Wang is the corresponding author (wangxingbin@iie.ac.cn)

DAC'24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Association for Computing Machinery.
ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3654810

However, deploying AED methods on current GPU-based inference systems is inefficient and impractical for the following reasons. First, with user experience in mind, queries in the inference system must be processed within a bounded time to meet service-level objectives (SLOs) [6]. However, adversarial ensemble methods increase the resource footprint due to the large number of models required for each query, which exacerbates the computational costs of inference and leads to high variation in latencies. Second, the inherent randomness of AED requires each query to employ different models for inference [8–10]. Existing scheduling mechanisms in the inference systems lack the ability to adapt to this randomness and can only execute ensemble models statically [14], failing to fully exploit the advantages of multi-model defense. Finally, the dynamic nature of AED demands frequent renewal of defense models in the system [4]. However, switching models on previous GPU-based inference systems can degrade performance [15] for co-located models due to interference effects between tasks sharing hardware using the prior spatial multiplexing technology like MPS.

Currently, the emergence of Multi-Instance GPUs (MIGs) provides an opportunity to implement a high-performance inference system that meets the requirements of low latency, high adversarial robustness, and dynamism. As a state-of-the-art (SOTA) spatial multiplexing technology, MIG allows multiple models to share the same GPU to improve ensemble inference throughput. With hardware resource isolation, there is little interference between multiple parallel DNN workloads running on MIGs. This feature aligns well with the dynamic nature of AED. An inference system built upon MIG technology can seamlessly update its model pool without impacting other co-located models, thus preventing service degradation. However, developing an inference system for AED using MIGs presents different challenges from existing ML inference servers.

First, as an advanced hardware-level resource management mechanism, MIG enables users to partition hardware resources heterogeneously and allocate them to different DNN models based on storage and computation requirements. The challenge lies in how to automatically allocate hardware resources for diverse DNNs in the ensemble defense model pool to improve system performance.

Second, AED relies on randomization, which complicates the scheduling of queries. The uneven sampling probabilities among ensemble models presents an opportunity for utilizing temporarily idle models to construct a larger ensemble [11], thereby enhancing adversarial robustness. However, this introduces a notable challenge in scheduler design, i.e., how to build a larger ensemble while maintaining sampling probability. Furthermore, the intrinsic heterogeneity of MIG necessitates the scheduler to address the routing of queries among instances with diverse computational capabilities, adding complexity to the scheduler design.

To this end, we propose Garrison, which to the best of our knowledge, is the first system-level solution for AED. Garrison delivers low-latency and robust DNN predictions by adopting a dual-pronged approach. First, it optimizes hardware resource provisioning for ensemble models based on MIG's intrinsic heterogeneity. Second, it employs a reinforcement learning-based scheduling policy to reduce inference latency while enhancing adversarial robustness. Additionally, leveraging the reconfigurability of MIG, Garrison enables dynamic updates to the model pool, thereby further fortifying the security of the system. The key contributions of the paper are summarized as follows:

- By conducting an in-depth analysis of defense models' inference behavior, we propose a multi-granularity hardware resource partition method to improve the overall utilization of GPUs.
- We design a novel scheduling mechanism based on reinforcement learning (RL), achieving a trade-off between adversarial robustness and inference latency by adapting to the randomness in ensemble model decisions.
- Our experimental results demonstrate that Garrison can improve robustness by up to 24.5% while speeding up ensemble inference by up to 6.6× compared to the SOTA inference framework.

2 BACKGROUND AND MOTIVATION

2.1 Adversarial Ensemble Defense

It is often challenging for potential attackers to directly access a target model deployed on a server. As a result, they resort to training substitute models as proxies for adversarial examples generating [7, 16], which may have different network architectures than the target model. Adversarial transferability between the target model and the substitute model is a crucial factor in carrying out effective adversarial example attacks [1]. Accordingly, the effectiveness of the AED approach stems from the fundamental observation that adversarial transferability is restricted among diverse and random ensembles. In particular, AED exhibits three key characteristics.

First, it employs multiple diverse models for ensemble inference [2, 3, 13, 17, 21]. The diversified architectures of ensemble models weaken the transferability between the target model and the substitute model, simultaneously leading to a varying spectrum of performance bottlenecks in the inference that fundamentally defy a "one-size-fits-all" resource allocation solution.

Second, it incorporates randomness [8–12], as random ensembles always provide better theoretical guarantees in terms of adversarial robustness than deterministic defenses [10]. Each model in the ensemble has a sampling probability, wherein models exhibiting stronger defensive performance are sampled with higher probabilities during inference [11]. If the system can ensemble more models for defense, the robustness of the system will be further improved as it becomes increasingly difficult for an adversary to mislead all DNNs within a larger ensemble[17, 18].

Third, the model pool deployed on servers needs to be dynamically updated to prevent adaptive adversaries from having sufficient time to steal the ensemble models [4, 12, 19]. As pointed out by Goodfellow [5], the static and fixed nature of deployed ML models limits the performance of adversarial defense mechanisms. Models deployed on existing systems are vulnerable to adversarial attacks, as adversaries can repeatedly query the prediction API to extract information and ultimately deceive it.

2.2 Understanding Multi-Instance GPUs

MIG is a hardware feature that allows users to partition a GPU heterogeneously into multiple **G**PU **I**nstances (**GI**s). Each instance has dedicated computing and memory resources, including separate streaming multiprocessors (SMs), L1/L2 cache, and DRAM memory. These hardware resources can be flexibly organized by the user to create GIs of varying granularity. As shown in Figure 1, the Nvidia A100 GPU can be partitioned into instances with 1/7, 2/7, and 4/7 of the total resources, denoted as GI(1), GI(2), and GI(4), respectively. Each GI behaves like a smaller, and full-fledged GPU, enabling concurrent execution of applications without any interference.

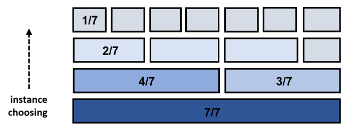

Figure 1: Example configuration of GPU partitions.

2.3 Challenges for Deploying AED Methods

Maximizing MIG throughput. Existing ML frameworks express DNNs as computation graphs with varying parallelism and resource requirements, making it impractical to allocate a uniform granularity of GIs to all models. As shown in Figure 2(a), increasing instance size from GI(1) reduces inference latency sharply for all models. However, the latency decrease saturates after each model reaches its respective optimal point, indicating that the additional hardware resources are not being used efficiently. Therefore, the optimal GI granularity varies across different model architectures. In addition, a large query size (i.e., batch size) can take full advantage of the parallelism and locality between inputs on the computational graphs, thus improving GI utilization. We can observe from Figure 2(b) that large batch sizes of queries can effectively utilize GIs with larger sizes to reduce latency, while small batch sizes cannot. Hence, query size and granularity of GIs must be considered as a joint factor when making utilization-aware allocation decisions.

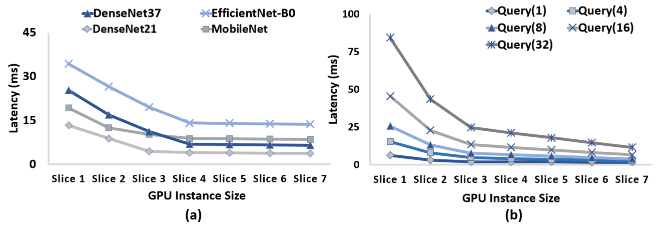

Figure 2: Inference latency analysis for (a) diverse DNNs with a fixed batch size of 8, and (b) varied query sizes using the identical VGG11 model.

Achieving efficient query scheduling. Our inference system exhibits dual concurrency at both the model and instance levels, necessitating the implementation of an efficient scheduling strategy. At the model level, as illustrated in Figure 3(a), simply scheduling queries to all available models for higher adversarial robustness would drastically increase inference latency due to the presence of straggler M_a. The execution time for query Q_1 with a large batch size on model M_a significantly exceeds that of other queries, preventing several subsequent queries from aggregating within the specified latency bounds. Therefore, it is imperative to judiciously select ensemble models to mitigate the straggler effect. At the instance level, as illustrated in Figure 3(b), scheduling queries on MIG in a first-come, first-served manner leads to suboptimal decisions. Failing to consider the matching between queries with different batch sizes and heterogeneous instances with varying computational capabilities leads to violations of SLO. Hence, an awareness of the heterogeneity of MIG constitutes another crucial aspect of implementing an effective scheduling mechanism.

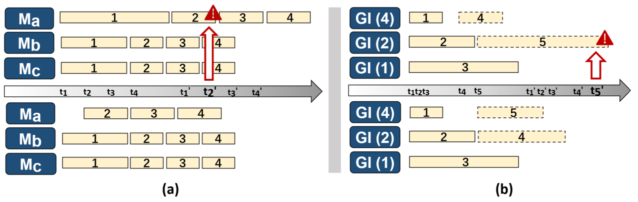

Figure 3: Scheduling complexities encountered at (a) the model level, assuming a higher computational complexity for model A compared to models B and C, and a larger size for Q_1 compared to others; and (b) the instance level. Note: t_i represents the arrival time of query Q_i, while t_i' signifies the time at which Q_i completes inference without breaching the SLO.

3 OVERVIEW OF GARRISON

Figure 4 depicts the high-level design of Garrison, which is implemented as a multi-GPU inference system. To handle user queries effectively, even those from malicious sources, ① the agent in the RL scheduler selects models randomly based on sampling probabilities. This selection incorporates as many models as possible, thereby enhancing the system's adversarial robustness. ② The ensemble of randomly selected models is used as input for the heterogeneity-aware routing optimizer, which selects the appropriate GPU instances. Queries are placed into the ready queue of the chosen instances, waiting to perform model inference. Once the inference of multiple models is completed, ③ an aggregator combines the results according to the different models' weights to obtain a robust prediction against adversarial examples. Throughout this process, ④ the system continually monitors incoming queries to spawn the probability density function of batch sizes. ⑤After each update of the defense models, ⑥ the heterogeneous resource allocator reallocates system resources based on the batch size distribution and the updated models. ⑦ Since MIG supports local reconfiguration, a subset of a GPU's instances can be repartitioned on-the-fly for model replacement without affecting other working instances on the same GPU. The continuous updating of the adversarial ensemble models makes it difficult for adversaries to construct high-quality substitute models through queries.

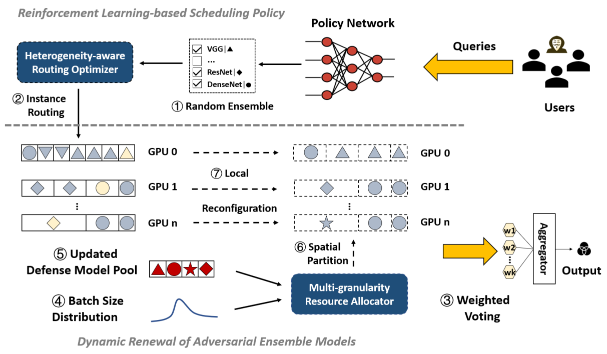
Reinforcement Learning-based Scheduling Policy
Policy Network
Queries
Heterogeneity-aware Routing Optimizer
VGG
ResNet
DenseNet
① Random Ensemble
② Instance Routing
Users
GPU 0
GPU 1
GPU n
⑦ Local Reconfiguration
⑥ Spatial Partition
⑤ Updated Defense Model Pool
④ Batch Size Distribution
Multi-granularity Resource Allocator
③ Weighted Voting
Aggregator
Output
Dynamic Renewal of Adversarial Ensemble Models

Figure 4: High-level overview of Garrison.

4 MULTI-GRANULARITY GI ALLOCATION

4.1 Profiling the Sweet Intervals

As MIGs have hardware resource isolation capabilities, the execution behavior of DNNs is highly predictable [20]. This provides an opportunity to profile the optimal GI granularities for each model across various query size ranges. The key insight behind profiling is that the computational capabilities of the provided GI should align with the parallel potential of the computational graph, determined by the DNN architecture and query size. We observe that, after GIs reach a specific utilization range with the increasing batch size, expanding the batch size hardly brings any further growth in throughput, while the latency increases nearly linearly, signifying inefficient utilization of the increased parallelism in the computational graph. This range is referred to as the "sweet interval." Figure 5 shows an example of such profiling, the sweet interval (16, 32) implies that the scheduler should distribute queries in the range to a GI(4) to achieve a trade-off between latency and GPU utilization.

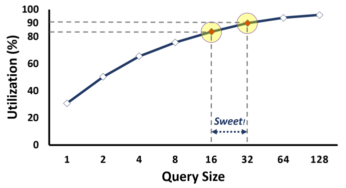

Figure 5: Utilization of different queries on a GI(4) deployed with VGG11.

4.2 Allocating GPU Instances

With offline profiling, we can determine the optimal GI size (GS) and the corresponding inference latency (L) for different query sizes. The multi-granularity resource allocation algorithm is described in Algorithm 1. It takes optimal GI size, inference latency, query size distribution, and model sampling probability as inputs and outputs an absolute number of allocated GIs for each defense model.

Algorithm 1 Multi-Granularity Resource Allocation

Input: Optimal GI size $GS[m, b]$ for model m under the query_size_range b, inference latency $L[m, b]$ of model m under the query_size_range b, cumulative probability density $D[b]$ of query_size_range b, sampling probability $SP[m]$ of model m, total number of GPU slices M.

Output: Absolute Number of GIs $N[m, b]$ allocated for model m under the query_size_range b with optimal GI size.

```
1: for each model $m$ in updated_model_pool do
2:     for each query_size_range $b$ do
3:         $IR[m, b] = GS[m, b] * L[m, b] * SP[m]$
4:     end for
5: end for
6: for each model $m$ in updated_model_pool do
7:     for each query_size_range $b$ do
8:         $RR[m, b] = IR[m, b]/\mathrm{SUM}(IR[:, b])$
9:         $AR[m, b] = RR[m, b] * D[b]$
10:        $P[m, b] = AR[m, b] * M$
11:        $N[m, b] = P[m, b]/GS[m, b]$
12:    end for
13: end for
```

To estimate the Relative Ratio (RR) between the resource requirements of different models under uneven random sampling probabilities, we weight the inference latency and optimal sample probability on the optimal GI size to get the Intermediate Ratio (IR) and then normalize the results. The observation behind this approach is that models with higher inference latency and sampling probability may require more hardware resources, which can create an opportunity for the scheduling mechanism to reduce the load inequality within the ensemble. The distribution probability of each query size range is further weighted by RR to obtain the Absolute Ratio (AR), which provides an opportunity for queries with different batch sizes to be routed to the most appropriate GI to further improve GI utilization. Finally, we multiply the total number of available GPU slices by the resource demand ratio to get the resource supply (P), and the result is divided by the optimal GI size to get the absolute number (N) of GI allocated, which reflects the models' unique computational requirements and the impact of query batch sizes. With one traversal of N, we can collect the demand of each model for different granularities of GIs, which are used to reconfigure the hardware resources of the system.

5 INTELLIGENT SCHEDULING MECHANISM

5.1 Designing the RL-Scheduler

In adversarial ensemble defense, the chance of each model being randomly selected is represented by a sampling probability vector. The selection probability of each model is not uniform; models with superior defense capabilities are assigned higher probabilities, leading to an imbalanced load across models. Leveraging idle models to form larger ensembles can enhance adversarial robustness, yet a crucial challenge arises: directing queries to idle models disrupts the intended sampling probabilities. To tackle this issue, we introduce a negative-feedback-based randomness bias adjustment to achieve the goal of integrating more models while maintaining an approximately optimal sampling probability, grounded in reinforcement learning, which realizes a balance between robustness and latency.

Figure 6 delineates the workflow of the query scheduling mechanism in Garrison. For each input query, the RL agent identifies the current execution state, encompassing factors such as workload pattern, instance status, and inference property. Given this observed state, Garrison selects an action representing the models used for ensembling, with the aim of maximizing adversarial robustness while satisfying latency objectives. The output of the policy network is harnessed as input for the heterogeneity-aware routing optimizer to execute instance selection, followed by Garrison conducting the DNN inference on the selected GPU instances.

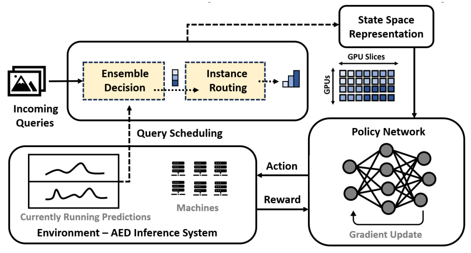

Figure 6: Workflow of the query scheduling mechanism.

During this process, Garrison addresses the conflict between incorporating more models into the ensemble and maintaining sampling probabilities through a negative feedback-based randomness bias adjustment. In particular, it records the models used for constructing the ensemble and calculates the actual sampling frequencies over a past time period to form a frequency table. After each action, the table is updated, and the difference between the sampling frequencies and the sampling probabilities, called randomness bias is calculated. This bias is used as part of the reward function, which reflects the satisfaction of optimal sampling probabilities. The policy network is then updated with the calculated reward, enabling the agent network to learn continuously and select models in a manner that approximates optimal sampling probabilities.

5.2 Ensemble Decision for Higher Robustness

To achieve efficient ensemble decisions, we need to focus on the three core components in reinforcement learning: state, action, and reward. By carefully designing these elements, we can achieve enhanced adversarial robustness by incorporating additional defense models while maintaining optimal sampling probabilities.

State: The state-space is designed in a manner that captures the temporal variations of system status. We compose the state space S as a Cartesian product of subspaces: $S = S_t \times S_p \times S_q$, with S_t, S_p and S_q representing the instance status, inference property, and workload pattern. First, the real-time status of instances is crucial, considering that selecting models on instances with multiple queries awaiting in their ready queue may lead to query timeouts. Thus, the instance status subspace $S_t = \{t_1, t_2, ..., t_n\}$ is established, representing the remaining execution time calculated by the estimator for each instance. Second, the straggler effect significantly influences model-level parallelism. To avoid the compounded effects of model complexity and query size on latency, we profile the inference latency of different-sized queries on various instances, constructing the inference property state subspace $S_p \in \mathbb{R}^{a \times b}$. Here, $S_p = \{\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_a\}$, with $\mathbf{p}_k$ as a vector of length b, its elements indicating the inference time of differently-batched queries on instance k. The RL agent uses this information to assess whether an on-instance model should be selected for a query of a specific batch size. Lastly, query size directly impacts the inference latency on different models. Query loads exhibit specific patterns, with several prior works noting that the query size follows a log-normal distribution [22]. Therefore, we define the workload pattern subspace S_q as $S_q = \{q_1, q_2, ..., q_T\}$, representing the sizes of historical queries arriving at the server within the past time interval of length T.

Action: In RL, actions represent the adjustable control knobs of the system. When formulating ensemble decisions, the action space is delineated as a binary vector $\mathbf{e} \in E \triangleq \{0, 1\}^a$, where $e_i = 1$ signifying the selection of model M_i.

Reward: A reward function models the system's optimization objective. In Garrison, the agent's goal is to learn an optimal policy within a time duration T, resulting in as few SLO violations as possible, maintaining model sampling frequencies as close to optimal sampling probabilities as possible, and ensembling more models to keep the adversarial robustness as high as possible. For this, we define the reward function as $r = w_1 \times \frac{E}{\sqrt{\sum_{i=1}^{n}(f_i - p_i)^2/n}} \times (B - w_2 \times N)$, where w_1 and w_2 are hyperparameters that can be tuned. E represents the number of models used for ensembling, B represents the query batch size, N represents the number of timeout queries in the system, n denotes the number of models, p_i denotes the optimal sampling probability of the i-th model and f_i denotes the actual sampling frequency within a past time interval of length T.

With the state, action, and reward well defined, we apply the actor-critic algorithm to optimize the overall reward by learning a good policy for ensemble decisions. In this work, we employ proximal policy optimization (PPO) because it typically outperforms other methods while being considerably easier to tune. The decision overhead of the agent in the inference stage is about 0.01s, which is within our acceptable range.

5.3 Instance Routing for Bounded Latency

After the RL agent intelligently distributes the server's input queries to different models, the system must select the optimal GIs to execute these models. For the individual model, the critical objective of routing is to distribute queries of different sizes to appropriate GIs for maximizing the throughput of each model. To solve this problem, we transform it into an equivalent minimization problem.

Suppose that for model j, there are m queries in the waiting queue, denoted as $Q^j_1, Q^j_2 \dots Q^j_m$, and n GIs allocated by the system, denoted as $GI^j_1, GI^j_2 \dots GI^j_n$. If query Q^j_i is routed to GPU instance GI^j_k, its serving time $D^j_{i,k}$ includes the inference latency and the remaining time for the query which is being executed on GI^j_k. Note that in our heterogeneous configuration, the same wall-clock usage time on different GIs does not have the same value. For example, a query of batch size (8) can be routed to MobileNet on GI(2) or GI(4), which both have an inference time of 5 ms, but with different hardware utilization. That is, one millisecond on GI(2) is not equivalent to one millisecond on GI(4) as the former is more cost-effective. Thus we define normalized usage time U instead of wall-clock usage time D to reflect the resource utilization of a heterogeneous system as $U^j_{i,k} = L^j_k D^j_{i,k}$, where L^j_k indicates the intrinsic parallelism level of the k-th instance of model j, which is set as the granularity value of GI^j_k.

For optimizer, optimization variables of model j can be defined as an $m \times n$ matrix P^j, where $P^j_{i,k} = 1$ indicates that query Q^j_i is served by GPU Instance GI^j_k and 0 otherwise. Since the higher a model's throughput, the smaller its overall normalized usage time, we can formulate the instance-routing problem as follows:

$$\min_{P} \sum_{i=1}^{m} \sum_{k=1}^{n} U^j_{i,k} P^j_{i,k} \tag{1}$$

$$s.t.\ \forall i,k,\ (D^j_{i,k} + W^j_i) P^j_{i,k} \leqslant SLO_{target} \tag{2}$$

$$\forall i,k,\ \sum_{i=1}^{m} P^j_{i,k} \leqslant 1,\ \sum_{k=1}^{n} P^j_{i,k} \leqslant 1 \tag{3}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{n} P^j_{i,k} \geqslant \min\{m,n\} \tag{4}$$

where $i \in \{1,2,\dots,m\}$ and $k \in \{1,2,\dots,n\}$. Eq.2 ensures that queries are completed within the SLO constraints. Eq.3 assures one-to-one mapping between queries and GIs, and Eq.4 ensures that each query is routed to a GI if there are more instances than queries, and each instance receives a query if there are more queries than instances. This formulation reduces the instance routing problem to a min-cost bipartite matching problem that can be handled directly by the Jonker-Volgenant algorithm.

6 EVALUATION

6.1 Methodology

Workloads. To effectively evaluate the performance of Garrison, we construct defensive ensembles with various DNNs (ResNet20, GoogLeNet, MoblieNet, DenseNet40, EfficientNet, VGG11) under four representative application scenarios in Table 1. We evaluate the robustness of ensembles using the method from DVERGE [21] under a black-box adversary. To evaluate Garrison's response and

Workload	Ensemble Models	SLO
Workload Scenario-A	MoblieNet(2), GoogLeNet(2), DenseNet40(6)	100 ms
Workload Scenario-B	MoblieNet(2), GoogLeNet(2), VGG11(2), EfficientNet-b0(2), EfficientNet-b3(2)	100 ms
Workload Scenario-C	VGG11(2), GoogLeNet(2), MoblieNet(2), DenseNet40(2), ResNet20(6), EfficientNet-b0(2)	150 ms
Workload Scenario-D	VGG11(2), MoblieNet(2), GoogLeNet(2), DenseNet40(6), EfficientNet-b0(2), EfficientNet-b3(2)	150 ms

Table 1: DNN workload scenarios and their SLOs. Note: the value in parentheses denoted by N signifies N homogeneous models trained on distinct datasets.

sensitivity to workload varies, we use log-normal distributed batch sizes as in former works [22]. In experiments, batch sizes are set to range from 1 to 64 and SLO is included as 99^{th} tail latency target.

Baselines. We select the current SOTA multi-GPU inference server, Triton, as a baseline to construct an adversarial defense system, called TritonX. To meet the strict requirements for tail latency in real serving scenarios, defense models in ensembles are formed in a greedy manner. The GI granularity assigned to each model is the optimal size that can meet the latency requirements of queries, which are formed through offline profiling. We also further evaluate Garrison's scheduling mechanism against the SOTA scheduling policies (Clockwork [20], Cocktail [6], and DeepRecSys [22]).

Hardware implementation. We conduct our experiments on a testbed that is equipped with 8 NVIDIA A100 GPUs, 1 AMD EPYC Rome CPU, and 750GB DRAM memory. As each A100 contains 7 slices, a max total of (7 × 8) = 56 slices can be utilized by Garrison to serve up to 56 models at any given time.

6.2 Overall System Performance

We compare Garrison with TritonX(N) (N denotes the number of defense models used for full ensemble in TritonX) under given hardware resource conditions for overall system performance evaluations. As shown in Figure 7, the latency of queries in TritonX(5) far exceeds that of SLO, making it unacceptable for users. In comparison, Garrison provides 6.6 ×, 4.87 ×, 3.64 ×, and 4.33 × improvement in latency for the four workload scenarios, avoiding SLO violations while achieving more than 20% robustness gain in all cases. In particular, for the four workload scenarios, the robustness of Garrison outperforms TritonX(4) by 24.5%, 21.9%, 24.8%, and 25.7% respectively. As the number of ensemble defense models in TritonX decreases to four, the query latency is significantly decreased, but the robustness is considerably lower than our solution.

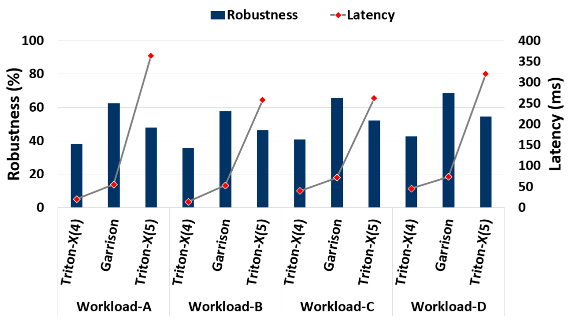

Figure 7: Garrison's robustness and latency improvements over TritonX.

To further observe the variation of adversarial robustness over time in the inference system, we show latency-bounded robustness as a function of sampling time, defining window robustness as the system's accuracy for adversarial examples within the sampling interval, and cumulative robustness as the system's accuracy for adversarial examples since the beginning of sampling. We observe from Figure 8(a)-(d) that Garrison can effectively scale up and down the defense model while maintaining high and stable cumulative robustness across all workload scenarios, achieving better security performance compared to TritonX(4)'s adversarial robustness of 38.1%, 35.9%, 40.8%, and 42.82% in the four scenarios, respectively.

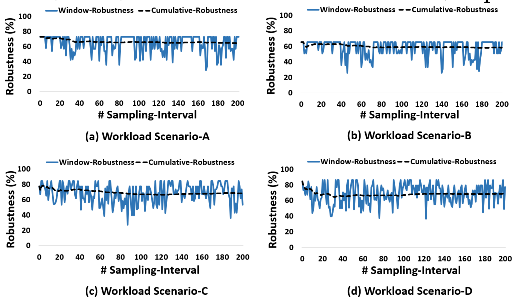

Figure 8: Garrison's robustness in the context of inference serving.

6.3 The Key Performance Improvements

Benefits from multi-granularity resource allocation. To demonstrate the benefits of the multi-granularity resource allocation algorithm, we allocate GIs of the same granularity to the ensemble models and transplant the RL scheduler from Garrison to TritonX for model and instance selection. As shown in Figure 9 (a), our multi-granularity resource allocation scheme achieves a robustness improvement of 10.1%, 15.3%, 7.2% and 6.8% over the homogeneous resource allocation scheme (TritonX) for the four scenarios, respectively.

The homogeneous solution ensures that queries are completed with a latency difference of less than 5s, which is comparable to the heterogeneous scheme, without violating SLO. However, such a choice results in fewer models being used for ensembling than the multi-granularity solution and thus reducing adversarial robustness. This is because the same-granularity resource allocation scheme does not consider whether the hardware parallelism can be fully utilized for different-batched queries, resulting in a waste of resources as the computational capabilities of the GI exceed the parallel potential of the computational graph.

Benefits from RL-based scheduling. To demonstrate the benefits of the RL-based scheduler, we transplant the multi-granularity GPU resource allocation from Garrison to TritonX and replace its First-In-First-Served (FIFS) scheduling mechanism with SOTA scheduling strategies for a fair comparison. Figure 9 (b) compares the performance difference between different SOTA schedulers and the intelligent scheduling mechanism of Garrison.

Thanks to the well-designed RL scheduling mechanism, Garrison achieves lower latencies than the full ensemble scheme for all four scenarios. On workload C, Garrison achieved an average of 8.9×, 9.8×, and 15.6× latency reduction compared to Clockwork, DeepRecSys, and Cocktail, respectively. On workload D, Garrison provides an average of 9.1×, 10.0×, and 17.8× improvement in latency compared to the three static defense schemes. This is because the excessive number of models under limited resources leads to fewer available instances allocated to individual models, further exacerbating the congestion effect of large queries on large models due to the existing scheduling mechanism, causing the overall inference time of the ensemble to far exceed the latency target. On workloads A and B, Garrison provides an average of 8.5× and 6.8× improvement in latency compared to Cocktail. Although DeepRecSys attempts to achieve better instance selection by identifying

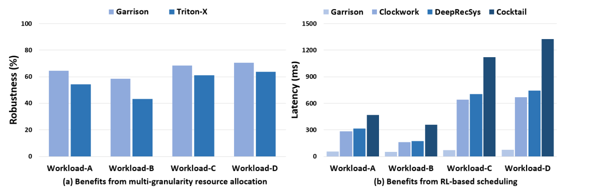

Figure 9: The key performance improvements from resource allocation and query scheduling.

the heterogeneity of GPU instances, it still results in many SLO violations under the static ensemble mechanism.

7 CONCLUSION

There is an imminent need to develop model serving systems that can deliver robust and low-latency predictions under limited computing resources. In this paper, we introduce Garrison, a novel inference system designed for adversarial ensemble defense methods utilizing multi-instance GPUs. Our evaluation with a diverse set of workloads shows that Garrison significantly outperforms SOTA techniques.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No. 62272459 and 62125208. The corresponding author is Xingbin Wang.

REFERENCES

[1] Fan M, et al. Enhance transferability of adversarial examples with model architecture. ICASSP, 2023.
[2] Amada T, et al. Adversarial Robustness for Face Recognition: How to Introduce Ensemble Diversity among Feature Extractors? SafeAI@ AAAI. 2021.
[3] Yang, Zhuolin, et al. Trs: Transferability reduced ensemble via promoting gradient diversity and model smoothness. NeurPS, 2021.
[4] Song Q, et al. DeepMTD: Moving Target Defense for Deep Visual Sensing against Adversarial Examples. TOSN, 2021.
[5] Goodfellow, Ian. A research agenda: Dynamic models to defend against correlated attacks. arXiv preprint arXiv:1903.06293 (2019).
[6] Gunasekaran, et al. Cocktail: A multidimensional optimization for model serving in cloud. NSDI, 2022.
[7] Liu, Yanpei, et al. Delving into transferable adversarial examples and black-box attacks. ICLR, 2017.
[8] Heredia, et al. On the Role of Randomization in Adversarially Robust Classification. NeurIPS, 2023.
[9] Pinot R, et al. On the robustness of randomized classifiers to adversarial examples. ICML, 2023.
[10] R. Pinot, et al. Randomization matters how to defend against strong adversarial attacks. ICML, 2020
[11] Dbouk, et al. On the robustness of randomized ensembles to adversarial perturbations. ICML, 2023.
[12] Peng, Qi, et al. Dynamic Stochastic Ensemble with Adversarial Robust Lottery Ticket Subnetworks. NeurIPS, 2022.
[13] B. Huang, et al. Adversarial defence by diversified simultaneous training of deep ensembles. AAAI, 2021.
[14] Crankshaw, et al. Clipper: A Low-Latency online prediction serving system. NSDI, 2017.
[15] Dhakal, Aditya, et al. Gslice: controlled spatial sharing of gpus for a scalable inference platform. SoCC, 2020.
[16] Ilyas, et al. Black-box adversarial attacks with limited queries and information. PMLR, 2018.
[17] Guo, Yong, et al. Improving robustness by enhancing weak subnets. ECCV, 2022.
[18] Cui, Sen, et al. Synergy-of-Experts: Collaborate to Improve Adversarial Robustness. NeurIPS, 2022.
[19] Amich, et al. Morphence-2.0: Evasion-Resilient Moving Target Defense Powered by Out-of-Distribution Detection. arXiv preprint arXiv:2206.07321 (2022).
[20] Gujarati, Arpan, et al. Serving DNNs like clockwork: Performance predictability from the bottom up. OSDI 20, 2020.
[21] H. Yang, et al. Dverge: diversifying vulnerabilities for enhanced robust generation of ensembles. NeurIPS, 2020.
[22] Gupta, Udit, et al. DeepRecSys: A system for optimizing end-to-end at-scale neural recommendation inference. ISCA, IEEE, 2020.