

Towards Scalable O-RAN Resource Management: Graph-Augmented Proximal Policy Optimization

Duc-Thinh Ngo^{*†§}, Kandaraj Piamrat^{†§}, Ons Aouedi[‡], Thomas Hassan^{*§}, Philippe Raipin^{*§}

^{*} Orange Innovation, Cesson-Sévigné, France

[†] Nantes University, École Centrale Nantes, IMT Atlantique, CNRS, INRIA, LS2N, UMR 6004, Nantes, France

[‡] SnT, SIGCOM, University of Luxembourg, Luxembourg

Abstract—Open Radio Access Network (O-RAN) architectures enable flexible, scalable, and cost-efficient mobile networks by disaggregating and virtualizing baseband functions. However, this flexibility introduces significant challenges for resource management, requiring joint optimization of functional split selection and virtualized unit placement under dynamic demands and complex topologies. Existing solutions often address these aspects separately or lack scalability in large and real-world scenarios. In this work, we propose a novel Graph-Augmented Proximal Policy Optimization (GPPO) framework that leverages Graph Neural Networks (GNNs) for topology-aware feature extraction and integrates action masking to efficiently navigate the combinatorial decision space. Our approach jointly optimizes functional split and placement decisions, capturing the full complexity of O-RAN resource allocation. Extensive experiments on both small- and large-scale O-RAN scenarios demonstrate that GPPO consistently outperforms state-of-the-art baselines, achieving up to 18% lower deployment cost and 25% higher reward in generalization tests, while maintaining perfect reliability. These results highlight the effectiveness and scalability of GPPO for practical O-RAN deployments.

I. INTRODUCTION

The rapid evolution of mobile networks and the increasing diversity of service requirements have driven the adoption of Open Radio Access Network (O-RAN), which promises greater flexibility, scalability, and cost efficiency through the disaggregation and virtualization of baseband functions. In O-RAN, operators can dynamically select functional splits and flexibly place virtualized baseband units across distributed computing resources, enabling more efficient resource utilization and improved support for heterogeneous Service Level Agreements (SLAs). However, this flexibility introduces significant challenges for resource management, as orchestrators must jointly optimize functional split selection and the placement of virtualized units under dynamic traffic demands, stringent latency constraints, and complex network topologies.

A variety of approaches have been proposed to address these challenges. Early works typically formulate the resource allocation problem as an Integer Linear Programming (ILPs) task, focusing on either the optimal placement of baseband units (BU) or the selection of functional splits, and sometimes both [1], [2]. More recent studies leverage Deep Reinforcement Learnings (DRLs) to handle the NP-hardness and adapt to dynamic request patterns [3]–[5], while others incorporate Graph Neural Networks (GNNs) to better capture the topological relationships in large-scale O-RAN deployments [6]. Despite

these advances, most existing solutions either treat functional split and placement as separate problems, assume static or simplified network topologies, or neglect the intricate coupling between routing, resource constraints, and SLA requirements. As a result, their scalability and effectiveness in real-world, large-scale, and dynamic O-RAN environments remain limited.

In this work, we propose a novel, scalable framework for O-RAN resource management that jointly optimizes functional split selection and virtualized baseband unit placement. Our approach leverages a Graph-Augmented Proximal Policy Optimization (GPPO) agent, which integrates GNN-based feature extraction with advanced DRL techniques and action masking to efficiently navigate the combinatorial decision space. By modeling the O-RAN substrate as a graph and encoding both node and edge attributes, our method captures the full complexity of network topology, resource availability, and service demands. Extensive experiments on both small- and large-scale O-RAN scenarios demonstrate that our GPPO approach consistently outperforms state-of-the-art baselines in terms of deployment cost, reward, and reliability, achieving superior scalability and solution quality even in highly dynamic and complex environments. Comparing to the closest works [5], [6], our method highlights the joint optimization of functional split and placement, while incorporating GNN to model the complex relationships in the O-RAN substrate, leading to more scalable and efficient resource allocation decisions.

The rest of the paper is organized as follows: Section II reviews related works on O-RAN resource allocation, highlighting the limitations of existing approaches. Section III defines the system model, and Section IV formulates the O-RAN resource allocation problem and optimization objectives. Section V presents our GPPO framework, detailing the GNN-based feature extraction, PPO training process, and action masking techniques. Section VI describes the experimental setup and results, comparing our method against state-of-the-art baselines. Finally, Section VII concludes the paper and discusses future work.

II. RELATED WORKS

We investigate the related works on O-RAN resource allocation, focusing on the following aspects: functional split selection, baseband unit placement, dynamic requests, routing, and deep learning approaches. A summary of the related works is provided in Table I.

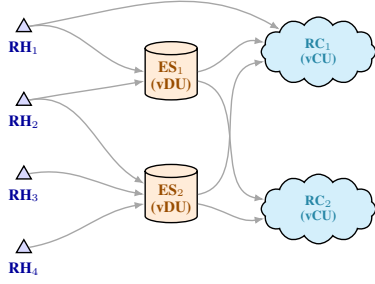


Fig. 1. O-RAN flexible functional split and placement architecture.

A. Joint optimization of functional split and BU placement

We can decompose the resource allocation problem in Radio Access Network (RAN) into two lines of subproblems: baseband unit placement and optimization of flexible functional split. The former addresses the efficient deployment of baseband processing units across computing nodes in centralized or virtualized RAN architectures. For instance, [1], [6], [7] focus uniquely on optimizing the dynamic placement of virtual baseband units on physical DUs and CUs in order to minimize the energy and consumption rate. In contrast, the latter leverages the flexibility of O-RAN base stations to dynamically select the most suitable functional split for each station, based on their specific service demands, often driven by the requirements of different network slices. However, these works often accompany the optimization of virtualized baseband unit placement, as the two problems are closely related. For example, [3], [8] focus on optimizing the functional split selection and vDU placement, while more recent works [4], [5], [9] consider vCU placement additionally. Overall, these problems can always be formulated as an ILP problem, which is NP-hard.

B. Dynamic requests

The dynamic nature of requests in O-RAN systems introduces additional complexity to the resource allocation problem. It can incur significant overhead in terms of reconfiguration and resource management, as the system must adapt to changing traffic patterns and service requirements. Most of the existing works focus on proposing a single configuration given static requests. For instance, [3], [8] consider solving an optimization problem for each timeslot independently, which does not account for the dynamic nature of requests. In contrast, [5], [9] propose solutions to the problem of reconfiguration that can handle dynamic requests, allowing for more efficient resource allocation and management.

C. Routing

In a large-scale O-RAN architecture with complex topology, routing, which is a coupling problem with the resource placement problem, is crucial and challenging to address. Moreover, when routing is considered in a large-scale substrate network, it can increase the complexity of the problem combinatorially. Most of the existing works focus on the placement of virtualized baseband units and functional split selection, while routing is often assumed to be static or predefined. For example,

TABLE I
Summary of related works on RAN resource allocation.

Ref.	Flexible split	VNF placement	Routing	Scale*	Dynamic requests	DRL	GCN
[8]	✓			128			
[3]	✓			201		✓	
[9]	✓	✓		21	✓	✓	
[6]		✓	✓	6	✓	✓	✓
[5]	✓	✓	✓	20	✓	✓	
Ours	✓	✓	✓	70	✓	✓	✓

*: Scale of the problem, defined as the number of nodes in the network.

[8] considers a single-CU scenario, where the routing is straightforward as all Radio Heads (RHs) are connected to a single CU. [3] considered that each RH is physically associated to a distinct DU, neglecting the need to optimize the routing policy. In [9], CU placement is decided by the optimizer but the routing (choosing DU placement in this case) is decided by the shortest path algorithm, which only partially aligns with the cost minimization objective. Recent works accounting for routing, e.g., [5], [6] suffer from seriously limited scalability, leading them to only consider small-scale networks with a limited number of nodes. In our approach, we propose a novel method that integrates routing optimization into the resource allocation process but still maintains scalability, allowing for efficient routing decisions even in large-scale O-RAN networks.

D. Deep learning approaches

To address the NP-hard and dynamic nature of the O-RAN resource allocation problem, many existing works leverage deep learning techniques, particularly DRL [3]–[6]. These approaches can learn by automatically exploring and exploiting the simulated environment to make optimal decisions based on the current state of the system. Moreover, to resolve the challenging routing problem, especially when the topology is complex, GNNs can be used to model the relationships between nodes and edges in the network, allowing for more efficient allocation decisions [6]. In this work, we propose a novel approach that combines DRL with GNNs to address the O-RAN resource allocation problem, leveraging the strengths of both techniques to achieve both scalability and efficiency.

III. SYSTEM MODEL

We consider an O-RAN architecture comprising three types of nodes: *RHs*, *Edge Servers (ESs)*, and *Regional Clouds (RCs)*. The baseband processing functions are disaggregated and distributed across vDUs hosted on ESs and vCUs hosted on RCs, connected through the cross-haul network. Figure 1 illustrates an example of a small hierarchical O-RAN. At each time slot, under a uniform temporal granularity, a request arrives consisting of a traffic load and an associated end-to-end latency requirement. These requests are categorized into three primary slice types, each associated with distinct SLAs: *Enhanced Mobile Broadband (eMBB)*, *Ultra-Reliable Low Latency Communication (URLLC)*, and *Massive Machine Type Communication (mMTC)*. For every RH, the O-RAN orchestrator must simultaneously make two key decisions: (i)

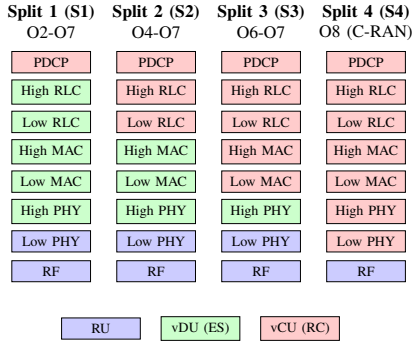


Fig. 2. Functional split configurations showing protocol layer distribution across RU, vDU (hosted on ES), and vCU (hosted on RC).

TABLE II
Functional split options and their associated traffic load and delay requirements.

Split option	Traffic load (Gbps)	Delay (ms)
O2	λ	10
O4	λ	1
O6	$1.02\lambda + 0.5$	0.25
O7	10.01	0.25
O8	157.3	0.25

Note: λ is the traffic load of the RH in Gbps, which is determined by the service slice type.

select an appropriate functional split and (ii) allocate virtualized baseband processing units to computing nodes.

Following standardized O-RAN recommendations [5], we consider four primary functional split configurations that combine high-layer splits (HLS) between vCU and vDU with low-layer splits (LLS) between vDU and radio units as shown in Figure 2. In practical deployments, these constraints are multifaceted and include both feasibility conditions and SLA-driven requirements. From a feasibility standpoint, the placement of vDUs and vCUs must ensure connectivity between the selected ES, RC, and their associated RH, while also satisfying the technical requirements imposed by the selected functional split —particularly regarding cross-haul latency and bandwidth. Each split imposes distinct cross-haul bandwidth and latency requirements, with more centralized splits demanding higher bandwidth capacity and stricter latency bounds. To avoid overloading compute resources and to meet SLA targets, the orchestrator must also ensure that server capacity constraints are not violated and that the end-to-end latency remains within the required bounds.

Beyond maximizing the success ratio, deployment cost is a critical optimization objective. These costs encompass both processing and routing fees. In O-RAN systems, processing on ESs typically incurs higher costs than on RCs, which incentivizes centralizing baseband functions at regional clouds. However, such centralization may increase routing costs or, in some cases, prevent SLA compliance due to tighter constraints on bandwidth and latency.

IV. PROBLEM FORMULATION

In this section, we formulate the dynamic O-RAN resource allocation problem as a multi-objective optimization problem that jointly addresses functional split selection and virtualized

TABLE III
Summary of Notation

Symbol	Description
<i>Sets and Indices</i>	
\mathcal{R}	Set of RHs, indexed by r
\mathcal{D}	Set of ESs, indexed by d
\mathcal{C}	Set of RCs, indexed by c
\mathcal{S}	Set of functional split options, indexed by s
t	Time slot index
<i>Decision Variables</i>	
f_{rs}^t	Binary: RH r uses split s at time t
x_{rd}^t	Binary: vDU of RH r placed on ES d at time t
y_{rc}^t	Binary: vCU of RH r placed on RC c at time t
<i>Network Parameters</i>	
e_{dc}	Binary: connectivity between ES d and RC c
δ_{rd}	Link delay between RH r and ES d
δ_{dc}	Link delay between ES d and RC c
B_{dc}	Bandwidth capacity of link between ES d and RC c
<i>Resource Parameters</i>	
c_s^{DU}	Computing resource requirement for vDU under split s
c_s^{CU}	Computing resource requirement for vCU under split s
R_d	Computing capacity of ES d
R_c	Computing capacity of RC c
β^s	Traffic load generated by split s
<i>QoS Parameters</i>	
Δ_r	End-to-end latency requirement for RH r
$\Delta_s^{\text{DU-CU}}$	Cross-haul latency requirement for split s
<i>Cost Parameters</i>	
ψ_s^{DU}	Unit cost for vDU deployment of split s
ψ_s^{CU}	Unit cost for vCU deployment of split s
ϕ_R	Reconfiguration penalty factor
ϕ_L	Routing cost factor
<i>Objective Functions</i>	
J_C	Computing resource cost
J_R	Reconfiguration cost
J_L	Routing cost

baseband unit placement. The optimization decisions must be made at each time slot t while satisfying multiple constraints related to resource capacity, connectivity, and SLA requirements.

A. Problem Scope and Formulation

We consider a discrete-time system where decisions are made at each time slot $t \in \{1, 2, \dots, T\}$. At each time slot, the O-RAN orchestrator must jointly determine: (i) the functional split assignment for each RH, (ii) the placement of vDUs on ESs, and (iii) the placement of vCUs on RCs. These decisions are represented by the binary decision variables f_{rs}^t , x_{rd}^t , and y_{rc}^t as defined in Table III. The optimization problem aims to minimize the total cost while satisfying resource capacity, connectivity, and SLA constraints. The optimization problem is subject to the following constraints:

1) *Assignment Constraints:* Each RH must be assigned exactly one functional split, one ES for vDU placement, and one RC for vCU placement:

$$\sum_{s \in \mathcal{S}} f_{rs}^t = 1, \quad \forall r \in \mathcal{R}, t \quad (\text{C1})$$

$$\sum_{d \in \mathcal{D}} x_{rd}^t = 1, \quad \forall r \in \mathcal{R}, t \quad (\text{C2})$$

$$\sum_{c \in \mathcal{C}} y_{rc}^t = 1, \quad \forall r \in \mathcal{R}, t \quad (\text{C3})$$

2) *Connectivity Constraint*: The selected ES and RC for each RH must be connected by a physical link:

$$x_{rd}^t y_{rc}^t \leq e_{dc}, \quad \forall r \in \mathcal{R}, d \in \mathcal{D}, c \in \mathcal{C}, t \quad (C4)$$

3) *Resource Capacity Constraints*: The total computing resource consumption on each server cannot exceed its capacity:

$$p_d = \sum_{r \in \mathcal{R}} x_{rd}^t \sum_{s \in \mathcal{S}} c_s^{\text{DU}} f_{rs}^t \leq R_d, \quad \forall d \in \mathcal{D}, t \quad (C5)$$

$$p_c = \sum_{r \in \mathcal{R}} y_{rc}^t \sum_{s \in \mathcal{S}} c_s^{\text{CU}} f_{rs}^t \leq R_c, \quad \forall c \in \mathcal{C}, t \quad (C6)$$

4) *End-to-End Latency Constraint*: The total end-to-end delay from each RH to its assigned RC must satisfy the SLA requirement:

$$l_{e2e}^r = \sum_{d \in \mathcal{D}, c \in \mathcal{C}} (\delta_{rd} + \delta_{dc}) x_{rd}^t y_{rc}^t \leq \Delta_r, \quad \forall r \in \mathcal{R}, t \quad (C7)$$

5) *Cross-Haul Latency Constraint*: The cross-haul delay between vDU and vCU must not exceed the requirement imposed by the selected functional split:

$$l_r^{\text{cross}} = \sum_{d \in \mathcal{D}, c \in \mathcal{C}} \delta_{dc} x_{rd}^t y_{rc}^t \leq \sum_{s \in \mathcal{S}} \Delta_s^{\text{DU-CU}} f_{rs}^t, \quad \forall r \in \mathcal{R}, t \quad (C8)$$

6) *Link Bandwidth Constraint*: The total traffic on each cross-haul link must not exceed its capacity:

$$b_{dc} = \sum_{r \in \mathcal{R}} x_{rd}^t y_{rc}^t \sum_{s \in \mathcal{S}} \beta^s f_{rs}^t \leq B_{dc}, \quad \forall d \in \mathcal{D}, c \in \mathcal{C}, t \quad (C9)$$

B. Objective Function

The optimization problem aims to minimize the total deployment cost, which comprises three components:

1) *Computing Resource Cost*: This component captures the cost of deploying vDUs and vCUs, where processing on ESs is more expensive than on RCs:

$$J_C = \sum_{r \in \mathcal{R}, d \in \mathcal{D}} x_{rd}^t \sum_{s \in \mathcal{S}} c_s^{\text{DU}} f_{rs}^t \psi_s^{\text{DU}} + \sum_{r \in \mathcal{R}, c \in \mathcal{C}} y_{rc}^t \sum_{s \in \mathcal{S}} c_s^{\text{CU}} f_{rs}^t \psi_s^{\text{CU}} \quad (1)$$

where ψ^{DU} and ψ^{CU} are the unit costs for vDU and vCU deployment, respectively.

2) *Reconfiguration Cost*: To account for the operational overhead of changing network configurations, we include a penalty for placement modifications between consecutive time slots:

$$J_R = \phi_R \left(\sum_{r,d} |x_{rd}^{t+1} - x_{rd}^t| + \sum_{r,c} |y_{rc}^{t+1} - y_{rc}^t| \right) \quad (2)$$

where ϕ_R is the reconfiguration penalty factor.

3) *Routing Cost*: This component represents the cost of utilizing network links for cross-haul traffic:

$$J_L = \phi_L \sum_{d \in \mathcal{D}, c \in \mathcal{C}} \delta_{dc} \sum_{r \in \mathcal{R}} x_{rd}^t y_{rc}^t \sum_{s \in \mathcal{S}} \beta^s f_{rs}^t \quad (3)$$

where ϕ_L is the routing cost factor, and the term $\sum_{r \in \mathcal{R}} x_{rd}^t y_{rc}^t \sum_{s \in \mathcal{S}} \beta^s f_{rs}^t$ represents the total traffic on link (d, c) .

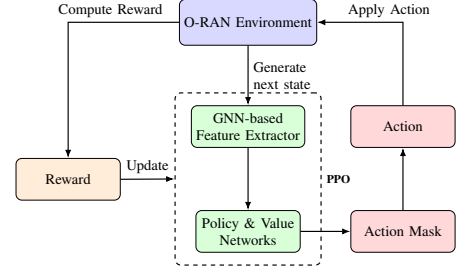


Fig. 3. Workflow of the GNN-enhanced PPO agent for O-RAN placement.

4) *Total Objective*: The complete optimization problem is formulated as:

$$\begin{aligned} \min \quad & J_C + J_R + J_L \\ \text{s.t.} \quad & \text{Constraints (C1-C9)} \end{aligned} \quad (4)$$

This formulation balances efficient resource utilization, system stability, and overall network performance while enforcing all SLA and capacity constraints. Because all decision variables are binary and constraints are linear, the problem is ILP-compliant. However, it remains NP-hard [3], [4], making exact optimization in large networks impractical. Therefore, we employ a reinforcement learning approach to efficiently explore the combinatorial decision space and learn effective placement and split-selection policies.

V. METHODOLOGY

A. O-RAN placement environment

To be able to solve this dynamic placement problem, we first reformulate it into an Markov Decision Process (MDP), represented by a 4-tuple (S, A, P_a, R_a) . S and A are the space of states and actions. $P_a(s', s)$ is the probability of the transition from state s' to the state s , conditional on the taken action a . Similarly, $R_a(s', s)$ is the immediate reward when the environment state transits from s' to s in the effect of the taken action a . While the transition probability is usually unknown and has to be implicitly learned from the exploration of the RL agent, the state-action space and reward function have to be explicitly defined when constructing an RL environment. To better reflect the optimization problem, we construct an MDP as follows:

1) *State*: The state variables will contain information about the current status of the environment, allowing the RL agent to reason and take the proper actions. Therefore, the state space should provide information about the demands of each RH and remaining computing resources of servers, remaining bandwidth of all links, and other static information, i.e., substrate network topology and link delays.

2) *Action*: The action space is the sample space of all actions that an RL agent can take. In this joint optimization problem, the RL agent has to make decisions on functional split assignment and virtual units placement simultaneously. Therefore, the action vector must be a vector of dimension $3N$ where the first N discrete variables are to indicate which functional split should be assigned to the corresponding RH. Besides, the last $2N$ are discrete variables to indicate which

ES and RC should be the hosts of the corresponding RH's vDU and vCU.

Formally, we denote the action vector at time t as:

$$\mathbf{a}_t = [a_{1,t}^{\text{split}}, \dots, a_{N,t}^{\text{split}}, a_{1,t}^{\text{DU}}, \dots, a_{N,t}^{\text{DU}}, a_{1,t}^{\text{CU}}, \dots, a_{N,t}^{\text{CU}}] \quad (5)$$

where $a_{i,t}^{\text{split}}$ indicates the functional split selected for RH i , $a_{i,t}^{\text{DU}}$ indicates the selected ES (vDU placement) for RH i , and $a_{i,t}^{\text{CU}}$ indicates the selected RC (vCU placement) for RH i .

3) *Reward*: While the original problem is to minimize the cost, the MDP's natural objective is to maximize the rewards. Moreover, due to the large action space, which grows exponentially with the number of RH, the exploration process of the RL agent becomes much more difficult since the number of invalid actions also increases exponentially.

One solution to resolve this sparse reward problem is to design the reward system with care, so that even when invalid actions are taken, the returned reward can give the agent hints about the direction of the exploration, thus guiding the exploration more efficiently.

Another solution is to relax constraints. We notice that among the constraints raised in Section IV, only the constraint C4 concerns the feasibility of deployment. Other constraints are mostly SLA-driven. Therefore, even if these constraints are violated, the deployment is still feasible. Thus, we can relax these constraints and impose a cost of SLA-violation instead. This cost will be added to the reward function we are constructing.

$$J_{\text{SLA}} = \sum_{i \in \mathcal{D} \cup \mathcal{C}} \max(R_i - p_i, 0) + \sum_{r \in \mathcal{R}} \max(\Delta_r - l^{\text{e2e}}, 0) \\ + \sum_{r \in \mathcal{R}} \max(\Delta_r - l^{\text{cross}}, 0) + \sum_{i \in \mathcal{D} \cup \mathcal{C}} \max(R_i - p_i, 0) \quad (6)$$

Thus, the total cost in a relaxed optimization problem is:

$$J = J_L + J_R + J_C + J_{\text{SLA}} \quad (7)$$

This allows us to construct our reward function as:

$$R = \begin{cases} -\frac{n_{\text{fail}}}{2N}, & \text{if C4 not satisfied } \forall d, c. \\ -1, & \text{if early terminated.} \\ (1 + \log(1 + J))^{-1}, & \text{otherwise.} \end{cases} \quad (8)$$

where n_{fail} is the number of failed links, which is always smaller than the total of established links $2N$, since each RH requires at most 2 links RU-DU and DU-CU. This reward function is ensured to be bounded in $[-1, 1]$. When the action is infeasible, which does not meet the constraint C4, the reward is negative, indicating a penalty. When the action is feasible, the reward is positive and is a decreasing function of the deployment cost, meaning that maximizing the reward leads to minimizing the cost.

In our environment, an episode is early terminated if the agent takes 5 consecutive invalid actions (i.e., actions that violate the hard connectivity constraint (C4)). When this occurs, the environment ends the episode immediately and assigns a reward of -1 for that step. This mechanism discourages the

agent from repeatedly exploring infeasible regions of the action space and accelerates learning by providing immediate negative feedback for persistent constraint violations.

B. Proximal Policy Optimization (PPO) with Invalid Action Masking

To solve the formulated MDP, we adopt the PPO [10] algorithm, a widely used on-policy reinforcement learning method that offers a good trade-off between stability and performance. PPO is designed to improve learning stability by preventing the policy from changing too drastically during each update. Given a policy parameterized by θ , the PPO objective is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad (9)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and \hat{A}_t is the estimated advantage at time t .

The key idea behind PPO is to optimize a surrogate objective function that includes a clipping mechanism. This mechanism limits how much the new policy can differ from the previous one during training. Instead of allowing the policy to fully follow the estimated advantage of each action (which can be noisy), PPO clips updates that go too far, ensuring more gradual learning and avoiding instability. Compared to the earlier method TRPO [11], PPO achieves similar stability without requiring complex second-order optimization techniques, making it faster and easier to implement.

In our problem, the joint decision space includes functional split selection and vDU/vCU placement for every RH. As the number of RHs increases, the action space grows combinatorially, resulting in a large number of invalid or suboptimal actions, particularly those that violate hard feasibility constraints such as broken connectivity between ES and RC nodes.

To improve learning efficiency and avoid wasting exploration steps on infeasible actions, we integrate action masking into PPO [12]. The agent is then restricted to sampling actions only from the valid subset defined by the current action mask. This will prevent the agent from exploring unnecessary invalid actions and thus accelerate the training progress. In this solution, we apply a mask to the action logits before applying the SoftMax function so that the probability of the masked action is reduced to 0. It has been proved that this masking still produces a valid policy gradient [12].

In our problem, we mask out all $a_{i,t}^{\text{DU}}$ and $a_{i,t}^{\text{CU}}$ actions that are not connected to the RH i in the substrate network, i.e., all $d \in \mathcal{D}$ and $c \in \mathcal{C}$ such that $e_{rd} = 0$ and $\sum_{d \in \mathcal{D}} e_{rd} e_{dc} = 0$. This ensures that the agent only considers feasible placements that maintain connectivity between the vDU and vCU of each RH. However, this does not guarantee that the proposed placement is feasible, as the selected RC are not guaranteed to be connected to the selected ES. Similarly, we mask out all $a_{i,t}^{\text{split}} = 4$ (Split 4) actions for RHs that do not have a direct link to any RC in the substrate network, i.e., all $c \in \mathcal{C}$ such that $e_{rc} = 0$, as Split 4 requires a direct connection between the RH and RC.

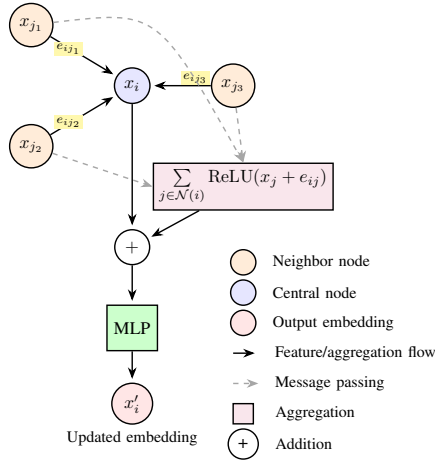


Fig. 4. Message passing and aggregation of the GINEConv block.

C. Graph Neural Network (GNN) as Feature Extractor

1) *Node and Edge Feature Design*: The substrate network is modeled as a graph where nodes represent RHs, ESs, and RCs, and edges denote physical links annotated with delay and bandwidth capacity. Each node is assigned features relevant to its functional role.

For non-RH nodes (i.e., ESs and RCs), features include their remaining resource capacity and a scalar *node order*, which denotes the fixed index of the node in the graph. As GNNs are inherently permutation-invariant, they do not encode positional information about nodes by default. However, to enable the RL agent to take placement actions, the positional information is necessary. Therefore, the node order serves both as a positional inductive bias to help the GNN distinguish nodes and allow the RL agent to indicate the node index when taking placement actions.

For RHs, we do not include resource capacity, assuming that all RHs have sufficient internal capability to host the RU segment of any split. Instead, RH are described by their service demands, including traffic load and end-to-end latency requirement. These demand attributes are also concatenated with their node order and processed separately.

Edge features consist of delay and bandwidth capacity. These are used during message passing to inform the model of topological constraints and link-specific characteristics.

Once encoded, all node and edge features are projected to the same embedding dimension and passed to the GNN layers for message passing. Formally, let the substrate network be represented as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. For each node $i \in V$, the node feature vector is defined as:

$$\mathbf{h}_i = [r_i, o_i, \lambda_i, \Delta_i] \quad (10)$$

where $r_i = R_i - p_i$ is the remaining resource capacity (for ES/RC nodes, $r_i = 0$ for RHs), o_i is the node order (index), λ_i is the traffic demand (for RHs, $\lambda_i = 0$ for ES/RCs), and Δ_i is the latency requirement (for RHs, $\Delta_i = 0$ for ES/RCs). The feature vector is constructed such that only relevant fields are non-zero for each node type.

For each edge $(i, j) \in E$, the edge feature vector is:

$$\mathbf{e}_{ij} = [b_{ij}, \delta_{ij}] \quad (11)$$

where b_{ij} is the bandwidth capacity and δ_{ij} is the link delay between nodes i and j . All node and edge features are projected to a common embedding dimension before being processed by the GNN layers.

2) *GNN Blocks*: Our feature extractor uses a two-layer Graph Isomorphism Network with Edge attributes (GINEConv) [13] for the core blocks. This variant of GIN aggregates features from neighboring nodes with additive edge feature fusion. Particularly, for each node i , the GINEConv updates take the form:

$$x'_i = \text{MLP} \left(x_i + \sum_{j \in \mathcal{N}(i)} \text{ReLU}(x_j + e_{ij}) \right) \quad (12)$$

where x_i is the embedding of node i , e_{ij} is the edge attribute between nodes i and j , and $\mathcal{N}(i)$ is the set of neighbors. Two GINEConv layers are applied sequentially, with learnable MLPs and non-linear activations. This allows the model to propagate both node and edge information across the graph.

3) *Graph Embedding and Output Integration*: After message passing, the final node embeddings are aggregated using *global mean pooling* to form a fixed-dimensional representation of the graph. This vector encodes the global state of the O-RAN substrate, including topology, resource availability, and service demands. The resulting graph-level embedding is provided to both the policy and value networks. This enables the RL agent to make globally informed joint decisions, i.e., assigning a functional split and placing the vDU and vCU for every RH in a single forward pass, based on a unified, topology-aware state representation. By incorporating this GNN-based encoder, our architecture enables generalization across varying network sizes and topologies, and supports efficient learning in large, structured action spaces.

VI. EXPERIMENTS

A. Simulation set up

1) *Overview*: We evaluate our approach on two O-RAN substrate scales: a small network (8 RHs, 3 ESs, 2 RCs) and a large network (64 RHs, 4 ESs, 2 RCs). We measure performance in terms of cumulative reward and total deployment cost, and test generalization on 5 different topologies with varied node distributions, connectivity patterns, and latency characteristics. All experiments share identical training conditions.

2) *Substrate Topology*: We generate two substrate topologies: (i) **Small-scale**: 8 RHs, 3 ESs, 2 RCs; (ii) **Large-scale**: 64 RHs, 4 ESs, 2 RCs. To ensure feasibility, each RH is connected to at least one ES and each ES to at least one RC, guaranteeing at least one valid RU-DU-CU path per RH. The generated topologies are shown in Figure 5.

3) *Split-4 Support*: For the legacy Split 4 configuration (omitting the DU), we add direct RH-RC links with a 10% probability per RH. These links are provisioned to satisfy split-4 throughput and latency requirements.

TABLE IV
Simulation request statistics for different network slices.

Network slice	Traffic load (Mbps)	End-to-end latency (ms)
eMBB	U(250, 300)	U(15, 20)
mMTC	U(150, 200)	U(180, 200)
uRLLC	U(20, 40)	U(2, 4)

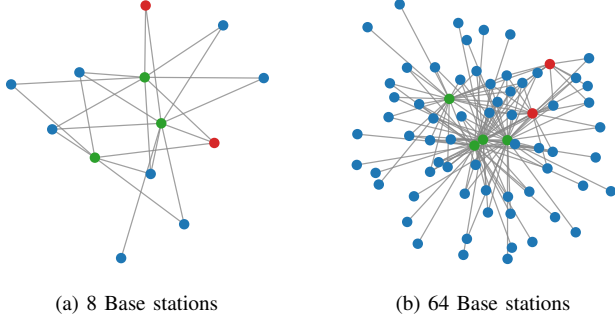


Fig. 5. Substrate network topologies (a) Small-scale network with 8 RHs, 3 ESs, and 2 RCs. (b) Large-scale network with 64 RHs, 4 ESs, and 2 RCs.

4) *Resource Configuration*: Following prior work [5], [6], we assign ES and RC compute capacities of 20 and 100 Computing Cores (CCs), respectively. Link bandwidths and latencies are sampled uniformly from [10, 40] Gbps and [0, 3.6] ms. Direct RH–RC links use 160 Gbps bandwidth and latencies in [0.1, 0.25] ms to meet strict split-4 constraints.

5) *Request Generation*: At each timeslot, every RH issues a request consisting of traffic load and end-to-end latency requirement. We sample slice-specific SLA parameters from uniform distributions defined in [14] and summarized in Table IV.

6) *Costing system*: The costing system is designed to reflect the deployment costs of the O-RAN system, including processing, routing, and reconfiguration costs. Each split s has an associated processing cost ψ_s^{DU} for vDU deployment on ESs and ψ_s^{CU} for vCU deployment on RCs. For split $s \in \{1, 2, 3, 4\}$, each vDU incurs a processing cost of $\psi_s^{\text{DU}} = \{0.05, 0.04, 0.00325, 0\}$ CCs per Mbps, and $\psi_s^{\text{CU}} = \{0, 0.001, 0.00175, 0.05\}$ CCs per Mbps for each vCU, respectively, as in [3], [4]. Routing and reconfiguration costs are defined as in Equations 2 and 3. In experiments, we set $\phi_R = 1$ and $\phi_L = 1$, but in reality, these can be adjusted to reflect different cost structures.

B. Evaluation setup

1) *Baseline Methods*: We conduct comprehensive comparisons against three state-of-the-art baseline methods to evaluate the effectiveness of our proposed GPPO approach:

- **DDPG** [15]: A deep deterministic policy gradient algorithm implemented with an MLP-based actor-critic architecture. The continuous action outputs are rescaled and rounded to transform into discrete actions.
- **PPO** [10]: The standard proximal policy optimization algorithm, using an MLP-based policy network that

directly operates on the discrete action space without action masking.

- **MPPO** [12], [16]: PPO with invalid action masking to improve learning efficiency by preventing selection of infeasible actions.
- **GPPO** (ours): Our proposed method that augments MPPO with a GNN-based feature extractor to enable topology-aware state representation and decision-making.

2) *Implementation Details*: All methods are implemented in Python using the Stable-Baselines3 [17] framework and its contrib package [16]. We employ Optuna [18] for automated hyperparameter optimization across all baseline methods to ensure fair comparison. The training infrastructure utilizes 32 parallel environments executed on 12 CPU cores with SubprocVecEnv vectorization for efficient sample collection.

3) *Training Configuration*: Each agent is trained for 600 000 timesteps across both network scales. Training episodes span 288 time slots each, with a resource releasing ratio of 0.5 to maintain dynamic demand patterns. The environment generates requests following the slice-specific distributions detailed in Table IV.

4) *Hyperparameter Settings*: Through Optuna optimization, we identified the following key hyperparameters for the PPO-based methods (PPO, MPPO, and GPPO): learning rate $\alpha = 10^{-4}$, batch size of 128 samples, discount factor $\gamma = 0.98$, GAE lambda $\lambda = 0.97$, clipping range $\epsilon = 0.3$, and entropy coefficient $c_{\text{ent}} = 10^{-6}$. DDPG employs independently optimized hyperparameters following standard actor-critic configurations with experience replay and target networks.

5) *Network Architectures*: The MLP-based policy and value networks in MPPO and GPPO consist of two hidden layers with 256 units each, using ReLU activation functions. Our GPPO method additionally incorporates a GNN feature extractor with 1024 hidden features, implemented using two-layer GINEConv blocks as described in Section 4.3. All networks are initialized using Xavier uniform initialization [19].

6) *Evaluation Protocol*: In the normal setup, we evaluate trained models on the same static topologies used during training to assess learning effectiveness. Each model performs 10 evaluation episodes per network scale, and we report the mean and standard deviation of cumulative reward and total deployment cost. For the large-scale scenarios, we include only episodes achieving feasible deployments (no connectivity constraint violations) in the cost analysis to ensure fair comparison. In the generalization setup, we trained a DRL model on 5 different 8-RH topologies simultaneously and evaluated them on the same 5 topologies for 100 evaluation episodes in total. All reported results represent averages over six independent training runs with different random seeds to ensure statistical reliability and account for training variance.

C. Results

We evaluate performance primarily using two metrics: reward and deployment cost. Mean rewards include all episodes regardless of feasibility, as negative rewards from constraint violations provide important information about method reliability.

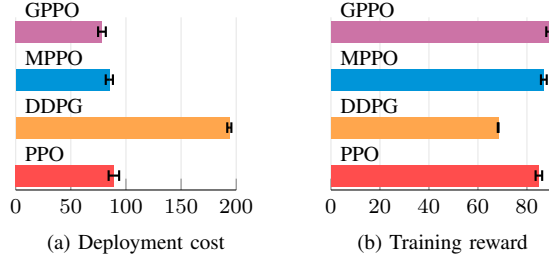


Fig. 6. Performance comparison on the 8-RH topology: (a) deployment cost and (b) training reward.

However, deployment costs are only reported for episodes that achieve feasible solutions, since infeasible deployments cannot be physically implemented and thus have no meaningful cost.

1) *Small scale*: We first evaluate our approach on the small-scale topology with 8 RHs. Figure 6 presents the comparative performance across all baseline methods. The results demonstrate clear performance differences across methods. As detailed in Table V, GPPO consistently outperforms all baseline methods across both performance metrics and network scales. For the small-scale topology, GPPO achieves the best overall performance in both reward and deployment cost metrics. MPPO follows closely in reward performance but incurs slightly higher deployment costs. PPO shows moderate performance across both metrics, while DDPG significantly underperforms with the highest costs and lowest rewards. The performance gap between GPPO and MPPO on this simple topology is modest, with GPPO achieving a **7.9%** cost reduction while maintaining similar reward levels. As shown in Figure 8, GPPO shows slightly faster convergence on the 8-RH topology compared to MPPO, though both methods achieve comparable final performance.

TABLE V
Quantitative performance comparison across network scales and methods.

Topology	Method	Mean Reward	Deploy Cost	Success Rate
Small-scale (8 RHs)	DDPG	68.26 ± 0.15	193.76 ± 1.87	6/6
	PPO	84.88 ± 1.34	89.06 ± 4.76	6/6
	MPPO	86.90 ± 1.18	84.94 ± 3.25	6/6
	GPPO (ours)	88.92 ± 1.02	78.23 ± 3.50	6/6
Large-scale (64 RHs)	DDPG*	-1.57 ± 0.05	—	0/6
	PPO*	-1.13 ± 0.03	—	0/6
	MPPO	24.28 ± 11.34	$7,982.97 \pm 531.18$	3/6
	GPPO (ours)	52.48 ± 0.36	$6,528.84 \pm 166.38$	6/6

*: Deployment costs unavailable for infeasible solutions since no actual deployment occurs.

2) *Large scale*: To assess scalability, we evaluate our method on the large-scale topology with 64 RHs. Figure 7 demonstrates the performance under increased network complexity.

The large-scale results reveal significant scalability challenges for traditional methods. Most critically, PPO and DDPG fail to achieve consistent feasible deployments, with all training runs converging to negative rewards, indicating complete failure to satisfy connectivity constraints. For these failed runs, the negative rewards are included in the reported mean performance,

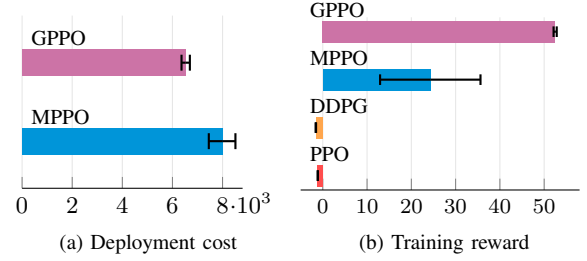


Fig. 7. Performance comparison on the 64-RH topology: (a) deployment cost for successful deployments and (b) training reward.

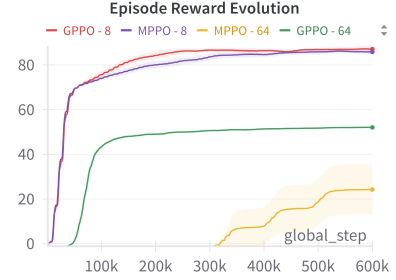


Fig. 8. Reward evolution during training across methods and network scales.

but no deployment costs can be computed since no actual network deployment occurs when constraints are violated. MPPO shows limited scalability, with only 3 out of 6 runs achieving feasible solutions. Moreover, even when MPPO does produce feasible deployment policies, these solutions are substantially suboptimal in cost efficiency.

In stark contrast, our proposed GPPO achieves a perfect success rate across all training runs with substantially higher mean rewards than MPPO. This dramatic performance gap highlights the critical importance of graph-based feature extraction for complex network topologies. Notably, GPPO not only ensures consistent feasibility but also delivers superior solution quality, achieving an **18.2%** cost reduction while maintaining perfect reliability, as detailed in Table V.

Figure 8 reveals that PPO and DDPG fail completely on the 64-RH topology, while MPPO shows delayed learning after 350k steps with high variance. GPPO maintains consistent learning from early stages, demonstrating the GNN’s effectiveness in extracting topological patterns for complex scenarios.

TABLE VI
Generalization performance comparison on different O-RAN topologies.

Method	Mean Cost	Mean Reward
MPPO	261.46 ± 92.50	48.47 ± 12.57
GPPO (ours)	211.80 ± 63.09	60.57 ± 10.72

3) *Generalization*: As expected, the generalization performance of both models has significant variance, due to the diverse topological characteristics of the 5 topologies. However, GPPO consistently outperforms MPPO across all metrics, achieving a **19%** cost reduction and **25%** reward improvement on average, as shown in Table VI. This demonstrates GPPO’s superior ability to generalize learned policies to unseen network

structures, validating the effectiveness of graph-based feature extraction for topology-aware decision-making.

4) **Key Findings:** Our experimental evaluation reveals several important insights:

Generalization Capability: GPPO demonstrates a clear advantage in generalizing to unseen O-RAN topologies, consistently outperforming MPPO in both cost and reward metrics. This highlights the effectiveness of graph-based feature extraction for robust, topology-aware decision-making in dynamic network environments.

Scalability Advantage: The most significant finding is GPPO's superior scalability. While multiple methods achieve reasonable performance on the small-scale topology, only graph-enhanced approaches maintain effectiveness as network complexity increases. The complete failure of PPO and DDPG on the large-scale topology, combined with MPPO's poor reliability, demonstrates the fundamental limitations of traditional Reinforcement Learning (RL) methods for complex O-RAN optimization.

Reliability vs Performance Trade-off: Beyond mean performance metrics, success rates reveal critical differences in method reliability. GPPO achieves perfect reliability across all scales, while MPPO's 50% success rate on large topologies makes it unsuitable for production deployment despite reasonable performance when successful.

Cost-Performance Synergy: Notably, GPPO achieves both higher rewards and lower costs across all scenarios, as shown in Table V. On the small-scale topology, GPPO's 2.3% reward improvement over MPPO comes with 7.9% cost reduction. At scale, this advantage reduces to 18.2% lower costs. This demonstrates that graph-enhanced learning not only ensures reliable feasibility but also leads to fundamentally superior optimization solutions, outperforming traditional methods even in their successful deployment cases.

Action Masking Benefits: The consistent improvement of MPPO over vanilla PPO validates the importance of constraint-aware action selection, though this benefit diminishes at scale without topological awareness.

VII. CONCLUSION

This paper presented a scalable framework for O-RAN resource management based on Graph-Augmented Proximal Policy Optimization. By jointly optimizing functional split selection and virtualized baseband unit placement, and leveraging GNN-based feature extraction with action masking, our approach effectively addresses the complexity and dynamics of real-world O-RAN environments. Experimental results on two different network scales demonstrate that GPPO achieves superior performance, reliability, and generalization compared to existing methods, reducing deployment costs and improving reward metrics even when handling different topologies at the same time. These findings highlight the importance of topology-aware learning and joint optimization for future O-RAN systems. Future work will explore further improvements in scalability, adaptation to online network changes, and integration with end-to-end network orchestration frameworks.

REFERENCES

- [1] N. Mharsi, M. Hadji, D. Niyato, W. Diego, and R. Krishnaswamy, "Scalable and cost-efficient algorithms for baseband unit (bbu) function split placement," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.
- [2] D. Harutyunyan and R. Riggio, "Flexible functional split in 5G networks," in *2017 13th International Conference on Network and Service Management (CNSM)*, Oct. 2017, pp. 1–9.
- [3] F. W. Murti, S. Ali, and M. Latva-Aho, "Constrained Deep Reinforcement Based Functional Split Optimization in Virtualized RANs," *IEEE Transactions on Wireless Communications*, vol. 21, no. 11, pp. 9850–9864, Oct. 2022.
- [4] E. Amiri, N. Wang, M. Shojafar, M. Q. Hamdan, C. H. Foh, and R. Tafazolli, "Deep Reinforcement Learning for Robust VNF Reconfigurations in O-RAN," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 1115–1128, Feb. 2024.
- [5] F. W. Murti, S. Ali, G. Iosifidis, and M. Latva-aho, "Deep Reinforcement Learning for Orchestrating Cost-Aware Reconfigurations of vRANs," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 200–216, Feb. 2024.
- [6] H. Li, P. Li, K. D. Assis, A. Aijaz, S. Shen, R. Nejabati, S. Yan, and D. Simeonidou, "NetMind: Adaptive RAN Baseband Function Placement by GCN Encoding and Maze-solving DRL," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2024, pp. 1–6.
- [7] K. Ali and M. Jammal, "Proactive VNF Scaling and Placement in 5G O-RAN Using ML," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 174–186, Feb. 2024.
- [8] F. Z. Morais, G. M. F. De Almeida, L. L. Pinto, K. Cardoso, L. M. Contreras, R. D. R. Righi, and C. B. Both, "PlaceRAN: Optimal placement of virtualized network functions in Beyond 5G radio access networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.
- [9] V. H. L. Lopes, G. M. Almeida, A. Klautau, and K. V. Cardoso, "O-RAN-Oriented Approach for Dynamic VNF Placement Focused on Interference Mitigation," *IEEE International Conference on Communications*, 2024.
- [10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.
- [11] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
- [12] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.
- [13] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations (ICLR)*, 2020.
- [14] Y. Wu, L. Liang, Y. Jia, W. Wen, and Z. Chen, "Slicing Enabled Flexible Functional Split and Multi-Dimensional Resource Provisioning in 5G-and-Beyond RAN," *IEEE Transactions on Wireless Communications*, vol. 23, no. 2, pp. 1213–1227, Feb. 2024.
- [15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [16] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3 contrib," <https://github.com/Stable-Baselines-Team/stable-baselines3-contrib>, 2021.
- [17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
- [18] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2623–2631.
- [19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*, 2010, pp. 249–256.