

# Mobile Edge Adversarial Detection for Digital Twinning to the Metaverse with Deep Reinforcement Learning

Terence Jie Chua

Graduate College

Nanyang Technological University

terencej001@e.ntu.edu.sg

Wenhan Yu

Graduate College

Nanyang Technological University

wenhan002@e.ntu.edu.sg

Jun Zhao

School of Computer Science & Engineering

Nanyang Technological University

junzhao@ntu.edu.sg

**Abstract**—Real-time Digital Twinning of physical world scenes onto the Metaverse is necessary for a myriad of applications such as augmented-reality (AR) assisted driving. In AR assisted driving, physical environment scenes are first captured by Internet of Vehicles (IoVs) and are uploaded to the Metaverse. A central Metaverse Map Service Provider (MMSP) will aggregate information from all IoVs to develop a central Metaverse Map. Information from the Metaverse Map can then be downloaded into individual IoVs on demand and be delivered as AR scenes to the driver. However, the growing interest in developing AR assisted driving applications which relies on digital twinning invites adversaries. These adversaries may place physical adversarial patches on physical world objects such as cars, signboards, or on roads, seeking to contort the virtual world digital twin. Hence, there is a need to detect these physical world adversarial patches. Nevertheless, as real-time, accurate detection of adversarial patches is compute-intensive, these physical world scenes have to be offloaded to the Metaverse Map Base Stations (MMBS) for computation. Hence in our work, we considered an environment with moving Internet of Vehicles (IoV), uploading real-time physical world scenes to the MMBSs. We formulated a realistic joint variable optimization problem where the MMSPs' objective is to maximize adversarial patch detection mean average precision (mAP), while minimizing the computed AR scene up-link transmission latency and IoVs' up-link transmission idle count, through optimizing the IoV-MMBS allocation and IoV up-link scene resolution selection. We proposed a Heterogeneous Action Proximal Policy Optimization (HAPPO) (discrete-continuous) algorithm to tackle the proposed problem. Extensive experiments shows HAPPO outperforms baseline models when compared against key metrics.

**Index Terms**—Metaverse; resource allocation, reinforcement learning; multi-agent; augmented reality; digital twin; Internet of Vehicles; adversarial.

## I. INTRODUCTION

**Background.** Digital twinning is the keystone of the Metaverse [1], in which real-world objects and events are mapped to and replicated in the virtual world. This opens doors to a myriad of possible applications which require real-time information of the physical environment, such as Augmented Reality (AR)-assisted driving. To facilitate AR-assisted driving capabilities, real-world scenes have to be uploaded to a central Metaverse Map Service provider (MMSP) which functions as a virtual reality host for geographical information. The

physical world scenes have to be collated, aggregated to form a coherent database. Internet of Vehicles (IoVs) can then query information from the MMSP and this information can be displayed as AR scenes on the IoVs' windshield which provide drivers with comprehensive, real-time information such as directions and landmark information to assist their driving.

**Motivation.** The development of the new-age AR-assisted driving technology invites adversaries. Adversaries may physically paste adversarial patches on cars, signboards, traffic lights or on the roads with the intent to corrupt the physical world scene which is to be uploaded to the MMSP for the development of a centralize virtual map. A successful attack as such can have disastrous effects, in which the Metaverse Map scenes may reflect erroneous information which when queried by IoVs can result in misinformation and accidents.

**Compute intensive detection.** These adversarial patches are often inconspicuous [2], and a fairly high resolution image of the patch is required for patch detection. This makes real-time detection of adversarial patches compute intensive, and these adversarial patch detection task have to be offloaded to the Metaverse Map Service Provider Base Stations (MMBSs) for computation. However, the offloading of high-resolution physical world scenes may induce large uplink transmission latency, yet offloading low-resolution images substantially impairs the MMSPs' adversarial patch detection ability. Furthermore, too many IoVs allocated to an MMBS may result in sub-optimal performance and unreliability in the system. Hence, IoVs may be excluded from certain UL transmissions if the occasional exclusion of an IoV results in better system performance. The total number of exclusions (idle count) should be minimized to ensure that the MMSP obtains comprehensive and regular information update from the IoVs.

**Our Approach.** Hence, we proposed a Heterogeneous Action Proximal Policy Optimization (HAPPO) algorithm to be employed within the MMSP orchestrator to tackle the optimization problem of (i) maximizing patch detection mean average precision (mAP) while minimizing the (ii) uplink latency and (iii) IoV idle count. The orchestrator consists of two agents, one to handle (1) discrete IoV-MMBS allocation and the other to handle (2) continuous physical scene resolution

selection. Our HAPPO architecture follows the Centralized Training and Decentralized Execution (CTDE) framework [3].

#### A. Related work

**Adversarial Patches.** The detection of adversarial perturbations within images has been thoroughly studied [4], [5]. Many works [6]–[8] in the field of adversarial detection have built classifiers to sift out corrupted samples from natural (unperturbed or clean) samples. However, as physical attack’s practicality in real-world gains recognition, there is an increasing number of researches focused on developing better defenses against adversarial patch attacks. Recent works [9], [10] in adversarial defenses have been focused on adversarial detection. These works utilize heuristic-based approaches such as using wavelets [9] and Grad-CAM maps [10] to differentiate between natural and adversarial samples.

**Metaverse applications.** Since the Metaverse is still relatively new, limited studies consider the IoT-Metaverse base station communication and computation framework. Chua *et al.* [11] introduced a AR socialization over 6G wireless networks within the Metaverse scenario and proposed a deep RL approach to tackle it. Han *et al.* [12] addressed resource allocation for the MEC of digital twinning of Internet of Things (IoT). Similarly, Ng *et al.* [13] tackled a resource allocation problem for the MEC of the Metaverse education sector using stochastic optimization.

**Resource Allocation.** Resource allocation for wireless networks have been thoroughly studied [14]–[16]. Works such as those by [17] utilized deep reinforcement learning approaches to allocate power for communications.

**Edge Computing.** Resource allocation and optimization of variables have been a long-standing concern in the field of mobile edge computing (MEC), and there have been several works [18], [19] which presented edge computing problems and developed solutions to tackle their proposed problem. Some works have adopted deep reinforcement learning approaches to tackle optimization problems for mobile edge computing [20]–[22].

**Contributions.** Our contributions are as follows:

- **Adversarial Detection in Defence of the digital twinning:** We present a novel mobile edge computing (MEC)-enabled adversarial patch detection for the defence of digital twinning to the Metaverse scenario, specifically in the context of AR-assisted driving.
- **HAPPO approach to Asymmetric Joint Optimization Problem:** We propose a Heterogeneous Action PPO, dual-agent (discrete-continuous) deep reinforcement learning-based IoV to MMBS orchestrator which aims to maximize adversarial patch detection mAP while minimizing total physical scene up-link transmission delay and IoV idle count.
- **Superiority of HAPPO:** We conducted experiments to compare the performance of our proposed HAPPO against other base-line algorithms and results demonstrate the effectiveness and superiority of our proposed method.

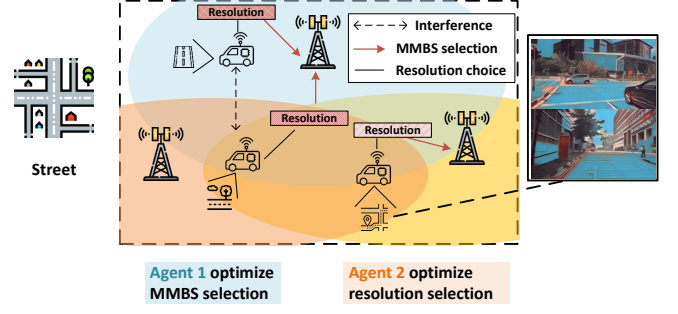


Fig. 1. System Model involving 2 agents to facilitate the adversarial detection offloading.

## II. ADVERSARIAL PATCH

**Adversarial Patch Attack.** For simplicity, we inserted an adversarial patch, digitally, onto images of cars and roads, 2000 from each of the *Stanford-Cars* [23] and *nulImages* [24] datasets, to mimic the placement of physical adversarial patches onto the physical environment. We assigned square-shaped patches of size smaller than 2% of the total image area randomly on our training dataset. We adopted the Projected Gradient Descent (PGD) patch attack [25] as an example attack to be applied on our training set. The mechanism of PGD can be described as such: The algorithm aims to find a perturbation value to be added to the natural example, which maximizes the loss function value, under the constraints in which the norm of the perturbation value falls within a pre-defined threshold (shown in equation 1).

$$\max_{\|\zeta\|_{\infty} \leq \varphi} l(\mathcal{F}(\chi_0 + \zeta; \varpi), \Upsilon_0) \quad (1)$$

where  $\chi_0$  represents the natural example,  $\Upsilon_0$  represents the original label,  $\zeta$  is the perturbation value to be added,  $\varpi$  is the model weights,  $l$  is the loss function, and  $\varphi$  represents the perturbation threshold value.  $\mathcal{F}$  is the predictive function that maps the input to a prediction. Implementing it by iterative gradient descent, we have:

$$\chi_{t+1} = \chi_t + \kappa \cdot \text{sign}(\nabla_{\chi} l(f(\chi_t; \varpi), \Upsilon_0)) \quad (2)$$

where  $\chi_t$  refers to the current image state, and  $\kappa$  is a scalar multiplier.

**Adversarial Patch Detection.** After the physical scenes with adversarial patches are offloaded from the IoVs to the MMBSs, the trained adversarial patch detectors on the MMBS will detect for adversarial patches on the uploaded physical world scenes. We adopted a cutting-edge object detector, pre-trained YOLOv4 with a CSPDarknet53 backbone [26] to detect the adversarial patches (shown in Fig. 2). In order to reduce latency, lower resolution images may be transmitted to the MMBS, which consequently result in poorer patch detection mean average precision score (mAP). Vice versa, transmission of higher resolution images results in higher latency but better patch detection mean average precision score (mAP). mAP is a common performance metric used in object detection tasks, which takes the mean of average



Fig. 2. Detection of Adversarial Patches from *nuiImages* (left and middle) [24] and *Stanford-Cars* (right) [23] datasets.

precision (AP) scores across different intersection over union (IoU) bounding box thresholds. In our work, we adopt the IoU threshold values from 0.5 to 0.95 in incremental steps of 0.05.

### III. SYSTEM MODEL

In our system,  $N$  AR vehicles from a set of  $\mathcal{N} = \{1, 2, \dots, N\}$  AR vehicles are capturing and uploading physical world scenes in real-time on the go, to Metaverse Map Service Provider Base Station (MMBSs)  $\mathcal{M} = \{1, 2, \dots, M\}$ . Each AR vehicle  $i \in \mathcal{N}$  moves around a defined geographical space at random and uploads physical environment scenes to an MMBS (shown in Fig. 1). As high-resolution, large data-size physical world scenes are required for patch detection, there may be a hand-over of the physical scenes uploaded, from one MMBS to another, AR vehicles move within a defined space. Several MMBSs and AR vehicles are distributed geographically. These AR vehicles transmit the physical world scenes to the MMBS and may generate interferences that disrupts the effective signal between other AR vehicles and their assigned MMBS. In our paper, we consider intra-cell interference. Intra-cell interference in this context, refers to the signal interference caused by the transmissions of other AR vehicles on the same bandwidth, and are assigned to the same MMBS, as the AR vehicle of interest.

**Uplink Communication model.** Each AR vehicle from a set of  $\mathcal{N} = \{1, 2, \dots, N\}$  will be assigned an MMBS's downlink channel from a set of  $\mathcal{M} = \{1, 2, \dots, M\}$  MMBS. The physical world scenes to be uploaded from the AR vehicles to the MMBS are of size  $d^t = \{d_1^t, d_2^t, \dots, d_N^t\}$ .  $d_i^t$  denotes the size of data to be uploaded by AR vehicle  $i \in \mathcal{N}$  at transmission iteration  $t$ . We denote the AR vehicle-MMBS assignment to be  $\mathbf{c}^t = (c_1^t, \dots, c_N^t)$ , where  $c_i^t = v (i \in \mathcal{N}, v \in \mathcal{M})$  denotes that AR vehicle  $i$  is allocated to MMBS  $v$  at iteration  $t$ . Considering the intra-MMBS interference, the *signal to interference plus noise ratio* of AR vehicle  $i$  at iteration  $t$  is defined as:

$$\Gamma_i^t(\mathbf{c}^t, \mathbf{h}^t) = \frac{g_{i,c_i^t}^t h_i^t}{\sum_{n \in \mathcal{N} \setminus \{i\}: c_n^t = c_i^t} (g_{n,c_i^t}^t h_n^t) + B\sigma^2},$$

where  $h_i^t$  is the power of AR vehicle  $i$  used for the transmission of physical world scenes to MMBS  $c_i^t$  at iteration step  $t$ ,  $g_{c_i^t, i}^t$  is the channel gain between MMBS  $c_i^t$  and AR vehicle  $i$  at iteration step  $t$ ,  $h_n^t$  is the power of AR vehicle  $n$  for communicating with MMBS  $c_i^t$  at iteration  $t$ ,  $B$  is the bandwidth of the communication, and  $\sigma^2$  denotes the additive white Gaussian background noise.

Principally,  $g_{i,c_i^t}^t h_i^t$  is the received signal at MMBS  $c_i^t$  from AR vehicle  $i$  in iteration  $t$ ,  $\sum_{n \in \mathcal{N} \setminus \{i\}: c_n^t = c_i^t} (g_{n,c_i^t}^t h_n^t)$  is the intra-cell interference caused by other AR vehicle  $n \neq i$  assigned to the same MMBS  $c_i^t$ , to AR vehicle  $i$  at iteration  $t$ . In each iteration step  $t$ , the uplink data transfer rate  $r_i^t$  from the AR vehicle  $i$  to its assigned MMBS is influenced by the SINR as such:

$$r_i^t(\mathbf{c}^t, \mathbf{h}^t) = B \cdot \log_2 (1 + \Gamma_i^t(\mathbf{c}^t, \mathbf{h}^t)), \quad (3)$$

From Equation (3), it is evident that the assignment of many AR vehicles to a single MMBS causes large intra-cell interference. A larger intra-cell interference would result in lower effective signals between AR vehicles and MMBS, and this causes the overall data transmission rate to decline. For a fixed data size to be transmitted, a higher data transfer rate results in a shorter uplink transmission delay at iteration step  $t$  as shown:  $\ell_i^t = \frac{d_i^t}{r_i^t}$ , where  $d_i^t$  is the size of the physical world scene to be uploaded from AR vehicle  $i$  to a MMBS at iteration step  $t$ . We consider the transmitted physical world scenes to be square-frames, where data size  $d_i^t$  and resolution  $p_i^t$  captured by AR vehicles  $i$  at iteration  $t$  are related by:  $d_i^t = \xi \cdot (p_i^t)^2$ .  $\xi$  represents the number of bits of information embedded within each pixel. Intuitively, as AR vehicle  $i$  uplink latency at iteration step  $t$  increases, the lower the consistency of update to the virtual world. Furthermore, a more efficient AR vehicle to MMBS allocation would increase each AR vehicles' SINR and consequently result in lower latency. Finally, the transmission of physical environment scenes of lower resolution reduces uplink transmission latency.

**Detection mAP-resolution model.** As the actual implementation of continuous real-time detection of adversarial patches is infeasible for the scale of our work, we established

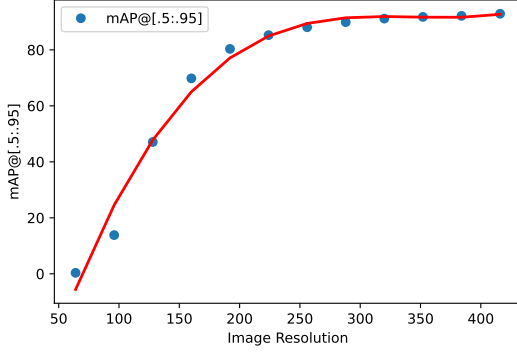


Fig. 3. mAP vs Resolution (pixel per inch, ppi).

the relationship between physical environment image resolution and adversarial patch detection mean average precision (mAP) score empirically. We collected multiple resolution-mAP pairs from the YOLOv4 prediction output and fitted a polynomial best-fit curve to the data points (as shown in Fig. 3). We note that the image resolution  $p$  and mAP are related by a polynomial relationship of  $\text{mAP} = 4.5 \times 10^{-6} \cdot p^3 - 4.7 \times 10^{-3} \cdot p^2 + 1.6 \cdot p - 90$ , for  $p \in [64, 416]$  pixel per inch (ppi).

**Idle Count.** To ensure that we have consistent physical scene transmission for patch detection from IoVs to the MMSP, we aim to reduce the total IoVs' idle count  $\sum_{t=1}^T \sum_{i \in \mathcal{N}} I_i^t$ , which refers to minimizing the total counts in which IoVs are not uploading physical world scenes to the MMBS.

#### A. Problem formulation

To sum up, the goal of the MMSP is to find the optimal IoV-MMBS allocation arrangement  $c^t$  and transmitted physical environment image resolutions  $p^t$  which minimizes the total up-link latency  $\ell^t$  and IoV idle count  $I^t$  while maximizing the IoV patch detection mAP  $\text{mAP}(p^t)$ . We formulated our up-link utility function as:

$$\min_{c^t, p^t} \sum_{t=1}^T \sum_{i \in \mathcal{N}} q \cdot \ell_i^t - b \cdot \text{mAP}(p_i^t) + f \cdot I_i^t, \quad (4)$$

$$\text{s.t. } c_i^t \in \mathcal{M}, \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, \quad (5)$$

$$h_i^t \leq h_{\max}, \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, \quad (6)$$

$$p_{\min} \leq p_i^t \leq p_{\max}, \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \quad (7)$$

where  $T$  is the total number of uplink transmissions of physical environment scenes from the IoVs to the MMSPs. The constraint (5) restricts each IoV to be allocated to at most one MMBS in each iteration step. Constraint (6) ensures that AR vehicle power output lies below  $h_{\max}$ . Constraint (7) ensures the image resolution lies between  $p_{\min}$  and  $p_{\max}$  ppi.  $b, f, q$  are scaling factors which seeks to balance the order and unit difference between  $\ell_i^t$ ,  $I_i^t$  and  $\text{mAP}(p_i^t)$  for joint-variable optimization.

#### IV. REINFORCEMENT LEARNING SETTINGS

For our work, we assign two reinforcement learning agents, *Agent1* and *Agent2*, with *Agent1* performing the discrete action of IoV-MMBS allocation, and *Agent2* performing the continuous action IoV uplink image resolution selection. Both agents are incorporated within the MMSP and represent the MMSP's interests. The rationale for adopting two agents is that we are optimizing two variables in which one has continuous and the other has discrete action spaces.

**State.** For both agents' observation state  $s^t$ , we have chosen to include 1) **channel gain between each IoV and all MMSPs**:  $g_{v,i}^t$ , 2) **image data size to be transmitted by each IoV** at each transmission iteration  $t$ :  $d_i^t$ , as these two variables influences data up-link transmission rate and latency.

**Action.** For *Agent1*, the agent's action is to decide the MMBS to IoV allocation, in which the action space for each IoV  $i$  can be written as such:  $a^{alloc,t} = c^t = (c_1^t, \dots, c_N^t)$ , ( $t \in T$ ). The number of the discrete actions is  $N^{M+1}$ , where  $N$  denotes the number of IoVs and  $M$  is the total number of MMBSs. This signifies that an IoV may or may not be allocated to a MMBS.

For *Agent2*, the action space is continuous and the action dimension is  $N$ , in which there is one image resolution value selected for each IoV to transmit the physical environment scenes to its assigned MMBS. The uplink action space for all the IoVs at each transmission iteration is written as such:  $a^{resol,t} = p^t = \{p_1^t, \dots, p_N^t\}$ , ( $t \in T$ ), where  $p_{i=N}^t$  is the uplink image resolution selected at iteration  $t$  for IoV  $N$ .

**Reward.** Although we have a single objective function, in practice, we break down the overarching objectives into smaller rewards to be assigned to each of our agents. We assign only components of the objective function which is influenced by an agent's decision to that agent.

For the *Agent1* the reward is given at transmission iteration  $t$  as such:

$$\mathcal{R}^{alloc,t} = -\frac{\sum_{i \in \mathcal{N}} (q \cdot \ell_i^t + f \cdot I_i^t)}{N} \quad (8)$$

while for *Agent2*, the reward given to the agent at transmission iteration  $t$  is given as such:

$$\mathcal{R}^{resol,t} = -\frac{\sum_{i \in \mathcal{N}} q \cdot \ell_i^t - b \cdot \text{mAP}(p_i^t)}{N} \quad (9)$$

We divide the reward functions by  $N$  IoVs to find an average reward, as the average reward received per IoV is more intuitive than the reward sum.

#### A. Heterogeneous Actions PPO

Inspired by the well-known Centralized Training Decentralized Execution (CTDE) framework [3], we developed the Heterogeneous Actions Proximal Policy Optimization (HAPPO) algorithm for our dual-agent RL model. This model features both discrete and continuous action spaces and utilizes PPO as the backbone, as PPO is considered a state-of-the-art algorithm with performance stability. We do not directly use traditional CTDE algorithms like Multi-Agent PPO (MAPPO)



because the actions in our scenario contain both discrete and continuous actions, and it is not feasible to directly concatenate them to form a unified action. This is because the discrete action space PPO and the continuous action space PPO use different networks and distributions for sampling actions.

Similar to PPO [27], HAPPO uses separate policies  $\pi_\theta$  and  $\pi_{\theta'}$  for sampling trajectories (during training) and evaluation, respectively. Here,  $\pi_{\theta_1}$  and  $\pi_{\theta_2}$  are two separate distributions instead of a shared distribution in policy optimization. KL divergence constraints are applied to both Actors' policies to prevent major policy changes in each update. As the Actor-network is based on policy gradient [28], according to PPO [27], we formulate the update function of Actors as:

$$\mathbb{E}_{(s^t, a^{alloc,t}) \sim \pi_{\theta'_1}} [f^{alloc,t}(\theta_1)(A^{alloc,t} + A^{resol,t})] \quad (10)$$

$$\mathbb{E}_{(s^t, a^{resol,t}) \sim \pi_{\theta'_2}} [f^{resol,t}(\theta_2)(A^{alloc,t} + A^{resol,t})] \quad (11)$$

where

$$f^{alloc,t}(\theta_1) = \min\{\mathcal{R}^{alloc,t}(\theta_1), \text{clip}(\mathcal{R}^{alloc,t}(\theta_1), 1 - \epsilon, 1 + \epsilon)\} \quad (12)$$

$$\text{and } \mathcal{R}^{alloc,t}(\theta_1) = \frac{\pi_{\theta_1}(a^{alloc,t}|s^t)}{\pi_{\theta'_1}(a^{alloc,t}|s^t)}. \quad (13)$$

$f^{resol,t}(\theta_2)$  and  $\mathcal{R}^{resol,t}(\theta_2)$  is also defined in the same manner as equation (13), (14), respectively, with *resol* replacing *alloc* in the superscripts.  $\epsilon$  refers to the policy clipping parameter.

Here,  $A^{alloc,t}$  and  $A^{resol,t}$  are the advantages of actions selected by *Agent1* and *Agent2*, respectively. The advantages are computed by the truncated version of TD( $\lambda$ ) [29].

$$A^{alloc,t} = \delta^{alloc,t} + \dots + (\gamma\lambda)^{\bar{T}-1} \delta^{alloc,t+\bar{T}-1}, \quad (14)$$

$$\text{where } \delta^{alloc,t} = \mathcal{R}^{alloc,t} + \gamma V_{\phi'}(s^{t+1}) - V_{\phi'}(s^t). \quad (15)$$

$A^{resol,t}$  and  $\delta^{resol,t}$  is defined in the same manner as equation (15), (16), respectively, with *resol* replacing *alloc* in the superscripts.  $\bar{T}$  is the trajectory segment,  $\lambda$  is the trace decay parameter and  $\gamma$  is the discount rate.

In terms of the value network (Critic), HAPPO uses identical Critics as per other Actor-Critic algorithms; and the loss function can be formulated as:

$$L(\phi) = [V_\phi(s^t) - ((A^{alloc,t} + A^{resol,t}) + \gamma V_{\phi'}(s^{t+1}))]^2 \quad (16)$$

where  $V(s)$  is the widely used state-value function [30], which is estimated by a learned critic network with parameter  $\phi$ . We update  $\phi$  by minimizing the  $L(\phi)$ , and the parameter  $\phi'$  of target state-value function periodically with  $\phi$ .

## V. EXPERIMENT

In this section, we will describe our experimental configurations and provide in-depth analyses on the results.

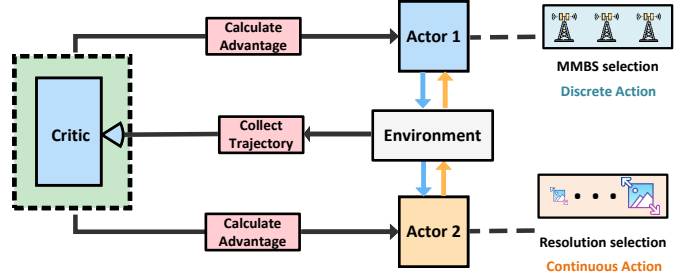


Fig. 4. Heterogeneous Action PPO (HAPPO) structure.

### Algorithm 1 Heterogenous Action PPO

**Initiate:** critic parameter  $\phi$  and target network  $\phi'$ , *Agent1* actor parameter  $\theta_1$ , *Agent2* actor parameter  $\theta_2$ , initialize state  $s^t = s^1$

- 1: **for** iteration = 1, 2, ... **do**
- 2: *Agent1* and *Agent2* execute action according to  $\pi_{\theta'_1}(a^{alloc,t}|s^t)$  and  $\pi_{\theta'_2}(a^{resol,t}|s^t)$ , respectively
- 3: Get  $\mathcal{R}^{alloc,t}$  and  $\mathcal{R}^{resol,t}$  and next state  $s^{t+1}$
- 4: sample trajectories:  
 $\tau = \{s^t, a^{alloc,t}, a^{resol,t}, s^{t+1}, \mathcal{R}^{alloc,t}, \mathcal{R}^{resol,t}\}$  iteratively
- 5: Compute advantages  $\{A^{alloc,t}, A^{resol,t}\}$
- 6: Compute target values  $\{V_{targ}^{alloc,t}, V_{targ}^{resol,t}\}$
- 7: **for**  $k = 1, 2, \dots, K$  **do**
- 8: Shuffle the data's order, set batch size  $bs$
- 9: **for**  $j=0, 1, \dots, \frac{T}{bs} - 1$  **do**
- 10: Compute gradient for downlink and uplink actors:  
 $\nabla \theta_1, \nabla \theta_2$
- 11: Apply gradient ascent on  $\theta_1$  using  $\nabla \theta_1$
- 12: Apply gradient ascent on  $\theta_2$  using  $\nabla \theta_2$
- 13: Update critic with loss using eq. (16)
- 14: **end for**
- 15: Assign target network parameters  $\phi' \leftarrow \phi$  after  $C$  iterations
- 16: **end for**
- 17: **end for**

### A. Configuration

We use five congestion settings: 3 MMBS and with 3 to 7 IoVs (denoted as "3x" for 3 MMBS and  $x$  number of IoVs) to test our proposed HAPPO orchestrator. We compared (i) our proposed HAPPO against baseline models (ii) Independent Dual Agent PPO-PPO, (iii) Heterogeneous A2C (HAA2C) which utilizes similar structure to HAPPO, and (iv) a random IoV-MMBS allocation and image resolution selection agent. The bandwidth and noise are simulated to be  $B = 10$  MHz and  $\sigma^2 = -100$  dBm. We initialize and constrain IoV power output, image resolution, and IoV locations for different IoVs to (1.5, 2.0) Watt, (64, 416) ppi,  $x, y \in (0, 1000)$  m, respectively.  $x$  and  $y$  represents the relative longitudinal and latitudinal directions in our 1000m by 1000m map, and IoVs randomly move a maximum of 100m in  $x$  and  $y$  directions in each transmission iteration. We set  $b, q$  and

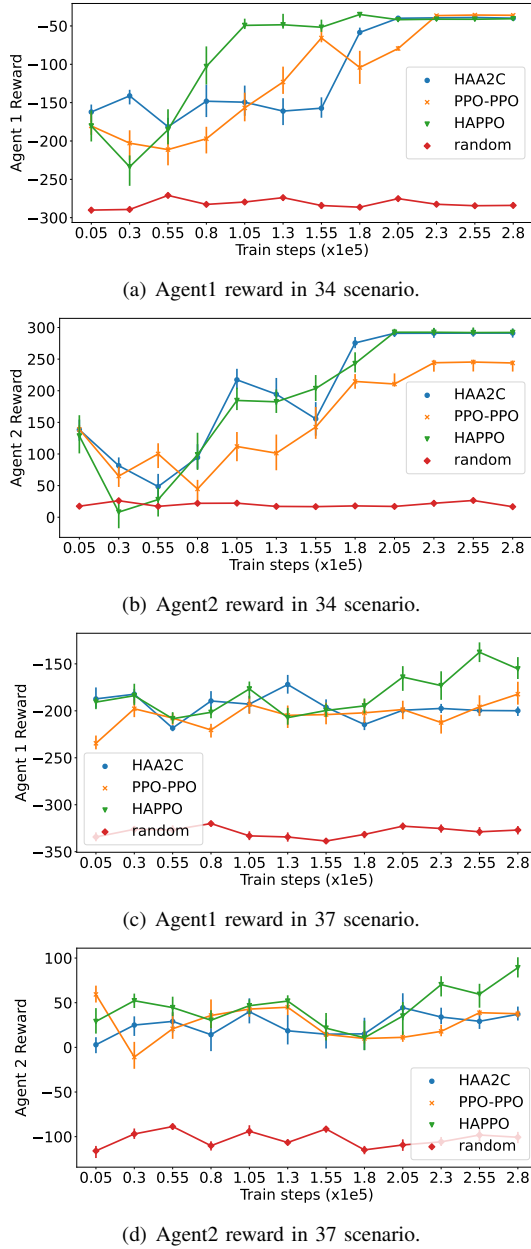


Fig. 5. Reward during training in scenario 34 and 37.

$f$  to be 50, 60 and 75, respectively, and these numbers are empirically derived. We adopt the ADAM optimizer [31] for all our implemented algorithms. To better observe the final performance, we use 280,000 steps for training. We conducted the training and simultaneous evaluation of the models for each of the configurations at different seed settings: seed 0 to seed 9.

### B. Result analyses

We present the final obtained rewards for both *Agent1* and *Agent2* in Table I. In the simpler "33" and "34" settings, most of the RL algorithm pairs we adopted performed fairly well and achieved convergence, with the exception of independent PPO-PPO in the "34" setting (reflected in poorer rewards obtained shown in Fig. 5(b)). Nevertheless, we observed a

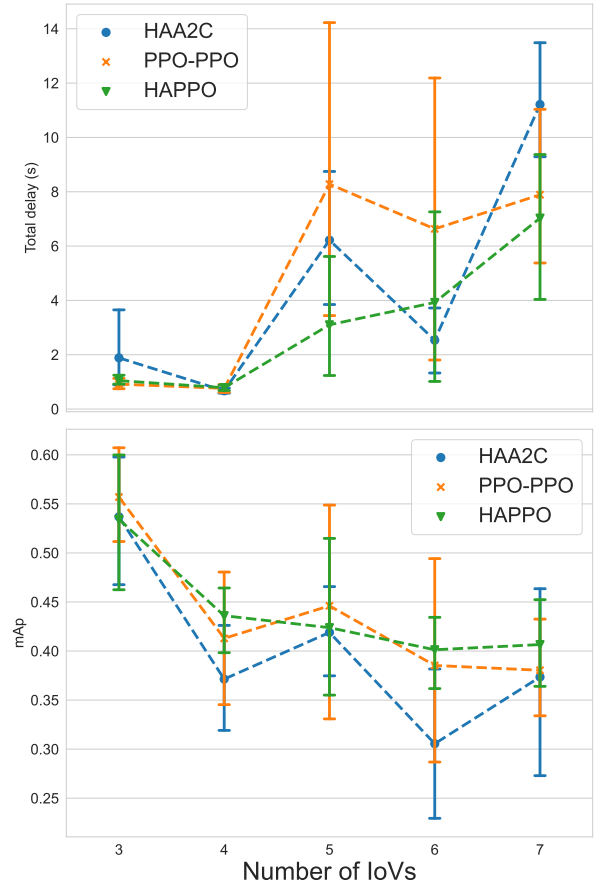


Fig. 6. Metrics with different User numbers.

 TABLE I  
OVERALL REWARDS

Number of IoV	HAPPO	HAA2C	Independent agents (PPO-PPO)
Agent1 Reward			
3	-47.53	<b>-43.82</b>	-48.94
4	-42.27	-41.10	<b>-40.36</b>
5	<b>-132.23</b>	-160.45	-154.83
6	<b>-148.34</b>	-173.42	-165.78
7	<b>-154.47</b>	-197.47	-184.23
Agent2 Reward			
3	<b>363.68</b>	344.59	360.36
4	<b>287.43</b>	286.72	238.58
5	<b>234.72</b>	152.53	163.49
6	<b>169.34</b>	134.29	145.6
7	<b>88.74</b>	38.10	39.34

notably quicker training convergence by the proposed HAPPO algorithm (shown in Fig. 5(a)). In the more complex scenarios such as "35", "36", and "37", we found that our adopted HAPPO achieved significantly better rewards than the other baseline RL algorithms (shown in Table I and Fig. 5(c) and Fig. 5(d)).

The total up-link transmission (of a batch of 1000 scenes) increases substantially as the number of IoVs increase, while the mAp score achieved decreased as the number of IoVs increased (shown in Fig. 6). This is not unexpected as the more complex scenarios involve more IoVs sharing computing resources with an unchanging number of MBBS. HAPPO

showcased its superiority over the other algorithm by obtaining the lowest average total transmission delay and considerably good mAP score, across the different congestion settings. Furthermore, HAPPO exhibits a much narrower range of total uplink transmission time and mAP score (as shown by the error bars in Fig. 6) when compared to other algorithm, indicating greater stability.

Despite the disparities in performance between the different algorithms, all algorithm performed better than an agent which allocates IoV-MBBS and selects uploaded image resolution randomly (shown in Fig. 5(a), 5(b), 5(c), 5(d)). This further substantiates that our proposed orchestrator improves the uplink communication in terms of maximizing accuracy, minimizing transmission delay and IoV idle counts.

## VI. CONCLUSION

In our work, we have proposed a real-time adversarial patch detector, enabled by mobile edge computing, in the defence of digital twinning to the metaverse. We formulated a realistic joint variable optimization problem where the MMSPs' objective is to maximize adversarial patch detection mAP, while minimizing the uplink transmission latency and IoV idle counts, through optimizing the MMBS allocation and IoV uplink image resolution. We proposed a Heterogenous Action PPO (HAPPO) (discrete-continuous) to tackle our proposed problem. We have demonstrated that our proposed HAPPO model outperforms baseline models and achieved superior performance based on key metrics.

## ACKNOWLEDGEMENT

This research is partly supported by the Singapore Ministry of Education Academic Research Fund under Grant Tier 1 RG90/22, RG97/20, Grant Tier 1 RG24/20 and Grant Tier 2 MOE2019-T2-1-176; and partly by the NTU-Wallenberg AI, Autonomous Systems and Software Program (WASP) Joint Project.

## REFERENCES

- [1] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," *arXiv preprint arXiv:2110.05352*, 2021.
- [2] T. Bai, J. Luo, and J. Zhao, "Inconspicuous adversarial patches for fooling image recognition systems on mobile devices," *IEEE Internet of Things Journal*, 2021.
- [3] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, 2017.
- [4] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.
- [5] Z. Zheng and P. Hong, "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [6] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.
- [7] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.
- [8] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [9] M. Arvinte, A. Tewfik, and S. Vishwanath, "Detecting patch adversarial attacks with image residuals," *arXiv preprint arXiv:2002.12504*, 2020.
- [10] Z. Xu, F. Yu, and X. Chen, "Lance: A comprehensive and lightweight cnn defense methodology against physical adversarial attacks on embedded multimedia applications," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 470–475.
- [11] T. J. Chua, W. Yu, and J. Zhao, "Resource allocation for mobile metaverse with the Internet of Vehicles over 6g wireless communications: A deep reinforcement learning approach," in *8th IEEE World Forum on the Internet of Things (WFIoT)*, 2022.
- [12] Y. Han, D. Niyato, C. Leung, C. Miao, and D. I. Kim, "A dynamic resource allocation framework for synchronizing metaverse with IoT service and data," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1196–1201.
- [13] W. C. Ng, W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, and C. Miao, "Unified resource allocation framework for the edge intelligence-enabled metaverse," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 5214–5219.
- [14] X. Liu, Z. Qin, Y. Gao, and J. A. McCann, "Resource allocation in wireless powered iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4935–4945, 2019.
- [15] W. Ahsan, W. Yi, Z. Qin, Y. Liu, and A. Nallanathan, "Resource allocation in uplink NOMA-IoT networks: A reinforcement-learning approach," *IEEE Transactions on Wireless Communications*, 2021.
- [16] N. Q. Hieu, D. T. Hoang, D. Niyato, D. N. Nguyen, D. I. Kim, and A. Jamalipour, "Joint power allocation and rate control for rate splitting multiple access networks with covert communications," *arXiv preprint arXiv:2203.16807*, 2022.
- [17] N. Q. Hieu, D. T. Hoang, D. Niyato, and D. I. Kim, "Optimal power allocation for rate splitting communications with deep reinforcement learning," *IEEE Wireless Communications Letters*, 2021.
- [18] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2018.
- [19] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in iot networks via reinforcement learning," in *ICC 2019-2019 IEEE international conference on communications (ICC)*, 2019.
- [20] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2019.
- [21] L. Huang, X. Feng, L. Qian, and Y. Wu, "Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing," in *Machine Learning and Intelligent Communications: Third International Conference, MLICOM 2018, Hangzhou, China, July 6-8, 2018, Proceedings 3*. Springer, 2018, pp. 33–42.
- [22] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile networks and applications*, pp. 1–8, 2018.
- [23] J. Krause, J. Deng, M. Stark, and L. Fei-Fei, "Collecting a large-scale dataset of fine-grained cars," 2013.
- [24] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.
- [25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
- [26] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [28] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.
- [29] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
- [30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.