Can't Slow Me Down: Learning Robust and Hardware-Adaptive Object Detectors against Latency Attacks for Edge Devices

Tianyi Wang, Zichen Wang, Cong Wang*, Yuanchao Shu, Ruilong Deng, Peng Cheng, Jiming Chen
Zhejiang University, Hangzhou, China,
{wty1998, withnorman, cwang85, ycshu, dengruilong, lunarheart, cjm}@zju.edu.cn

Abstract

Object detection is a fundamental enabler for many real-time downstream applications such as autonomous driving, augmented reality and supply chain management. However, the algorithmic backbone of neural networks is brittle to imperceptible perturbations in the system inputs, which were generally known as misclassifying attacks. By targeting the real-time processing capability, a new class of latency attacks has been reported recently. They exploit new attack surfaces in object detectors by creating a computational bottleneck in the post-processing module, which leads to cascading failure and puts the real-time downstream tasks at risk. In this work, we take an initial attempt to defend against this attack via background-attentive adversarial training that is also cognizant of the underlying hardware capabilities. We first draw system-level connections between latency attacks and hardware capacity across heterogeneous GPU devices. Based on the particular adversarial behaviors, we utilize objectness loss as a proxy and build background attention into the adversarial training pipeline, and achieve a favorable balance between clean and robust accuracy. The extensive experiments demonstrate the effectiveness of the defense in restoring real-time processing capability from 13 *FPS to* 43 *FPS on Jetson Orin NX, with a better trade-off between the clean and robust accuracy. The source code is available at:* https://github.com/Hill-Wu-1998/underload.

1. Introduction

Real-time object detection lies at the heart of a wide range of downstream applications such as autonomous driving [16], drone navigation [36], and video surveillance [20, 29]. From the early detectors such as (Faster) RCNN [9, 26] to the latest versions of the YOLO family [25, 33, 34], we have seen remarkable improvements in performance and efficiency, *e.g.*, reaching 56-60% mAP on the MS-COCO benchmark with more than 30 FPS on embedded NVIDIA Jetson boards.

(a) Original Image (b) Latency Attack (c) Proposed Defense

Figure 1. Visualization of the *latency attack* [28] that generates an overwhelming number of "phantom objects" and the effectiveness of the proposed defense.

However, in the shadow of these tremendous successes, a new threat called *latency attacks* lurk to impair the real-time processing capability of object detectors [3, 21, 30, 35], which employ hand-crafted Non-Maximum Suppression (NMS) to eliminate duplicate objects in the post-processing stage. By targeting the computational bottleneck inside the NMS module [12], these attacks create more than thousands of "phantom objects" with gradient-based adversarial techniques to congest the NMS processing pipeline as shown in Fig. 1. In addition to the known attacks against object detectors such as disappearing, misclassifying and mislocation [6, 15, 41], they expand the attacker's weapon arsenal with high risks of jeopardizing real-time applications on edge systems. *E.g.*, any latency in detecting an obstacle in autonomous driving could lead to cascading failure in the sensor fusion, decision and steering control subsystems.

Unfortunately, with the arms race on the attack side, we have seen a paucity of research to defend against these

*Corresponding Author

new vulnerabilities. A quick fix is to set hard limits on the number of objects [3], whereas such a limit is hard to select for different applications since a large value would still allow phantom objects to pass and a small one might falsely reject correct instances in case a large number of objects are present, especially in crowd counting and traffic monitoring applications. A comprehensive solution is to completely eliminate the hand-crafted NMS [2, 48]. However, this requires changing all the legacy software dependencies on the edge firmware and NMS-free architectures are still far from achieving real-time performance on edge devices [46]. Admittedly, the computational bottleneck in NMS is not straightforward to circumvent: our system-level analysis unveils that it not only comes from limited computational power (on edge devices), but also extensive data transfer between GPU-CPU that is not discussed in the previous works [3, 28, 35]. Would heterogeneous GPUs have different capacities under these attacks – are high-end GPUs free from such attacks? If the bottleneck persists, how can we thwart latency attacks without removing the NMS module? Given a performance requirement, can we guarantee such requirement by learning robust, adaptive object detectors on different hardware accelerators?

To answer these questions, in this paper, we propose `Underload` against `Overload` [3] and a series of latency attacks [21, 28, 35] via a hardware-adaptive, background-attentive Adversarial Training (AT) mechanism. We first leverage system-level analysis to identify processing bottlenecks and orchestrate these findings to model the GPU capacity with the number of candidate bounding boxes. We discover the usage of objectness loss can be served as a proxy to eliminate phantom objects. By delving into the unique adversarial behaviors of latency attacks, we also find that the background region typically consists of non-robust features [14], that are more vulnerable. To this end, we design background-attentive AT mechanism, which weighs more on the background semantics to achieve a better balance between the robust and clean accuracy. The main contributions are summarized below.

① **Motivation.** Based on extensive programming analysis and system profiling, we discover interesting phenomenons on the migration of processing bottlenecks from compute-bound to memory-bound operations between edge and desktop GPUs. These findings allow us to establish a bridge between attack strength and heterogeneous hardware capacity with a deeper understanding of the system impact on various types of GPUs.

② **Methodology.** We perform in-depth analysis of the objectness loss and draw connection with the latency attacks. Then we find unique footprint regarding the discriminative boundary margins between the background and object regions. Our defense successfully designs these factors into the AT pipeline and achieves a favorable balance between clean and robust accuracy by exploiting the robust/non-robust features.

③ **Evaluation**. We perform extensive experiments across the widely-adopted YOLOv3, YOLOv5 and the latest YOLOv8 (anchor-free) on embedded GPUs (Jetson Xavier/Orin NX), desktop GPUs (4070Ti Super) and multi-tenant cloud GPUs (A100). The results show that our approach achieves up to 8-10% gain on robust accuracy compared to the previous existing defense of MTD [43] and OOD [13] with less clean accuracy loss. Under various latency attacks, our defense is also able to restore the real-time processing capability from 13 FPS to 43 FPS on Jetson Orin NX as well as different types of GPUs.

2. Background and Related Works

Object Detection includes both classification and precise localization of objects within a digital image, addressing the question of "what" and "where" the objects are [49]. The types of object detectors mainly include CNN-based one-stage detectors [18, 24, 25, 33, 34], two-stage detectors [9, 26, 45], transformer-based detectors such as DETR [2, 7, 48], and diffusion detectors [5]. Two-stage detectors first generate region proposal with the objects of interest, then the second stage classifies and refines the bounding boxes for more precise localization [9, 26, 45]. On the other hand, one-stage detectors transform object detection into a regression problem that directly obtain classification and localization with a single pass, without the extensive RoI extraction [18, 24, 25, 33]. Hence, the research community has embraced one-stage detectors due to their simplicity, fast response, and hardware affinity across a wide variety of embedded devices. In response to the latency attacks against the YOLO family, this paper focuses on the vulnerabilities in one-stage detectors.

Non-Maximum Suppression. NMS serves as an essential post-processing backend and becomes an indispensable step for different object detectors [18, 26, 33]. The main purpose is to consolidate and remove redundant objects in each object cluster and return a final bounding box identified as the local maxima, with peaks exceeding their neighboring values by excluding the maxima themselves [23]. A different variety of NMS methods have been proposed such as GIoU [27], CIoU [47], α-IoU [11] and Wise-IoU [31]. However, as per both prior and our analysis, NMS poses inherent security vulnerabilities and opens up a unique attack surface to latency attacks due to its programming structure and system implementation described in the next section.

2.1. Latency Attacks

The rationale behind latency attacks originates from the classic *Denial of Service* attacks that originally cause congestion in networking services. Sponge example is the first

attack in AI systems that aims to increase energy consumption and inference time by firing up more activations in NLP models that lead to more multiply-add operations [30]. Surprisingly, its impact is less significant in vision tasks when the default activation values are already non-zero for most images. This makes the follow-up works to turn their targets to the hand-crafted NMS module [3, 21, 28, 35].

Daedalus is the pioneering work that exploits the vulnerability of NMS [35]. Three adversarial losses are developed to make the final outputs contain an extremely high density of false positives. Phantom Sponge enhances Daedalus's methodology by training a universal adversarial perturbation, and analyzes the NMS execution time to formulate the latency attacks against object detection models. Recent efforts extend these attacks towards resource-constrained edge devices [3] and autonomous driving backends [21]. Overload analyzes the NMS complexity, which analytically confirms latency extension by increasing candidate bounding boxes entering the NMS [3]. Furthermore, it augments the attack efficacy by incorporating spatial attention. SlowTrack adopts latency attacks in the autonomous driving scenario and raises the vehicle crash rate to 95% by executing latency attacks on camera-based autonomous driving systems [21]. There are also another types of latency attacks that does not target object detectors [4, 10]. All relevant works above remain on the offensive side to exploit NMS as a new attack surface. To our best knowledge, this is the first attempt that builds specialized defense against latency attacks for object detectors running on millions of edge devices.

3. Understanding Latency Attacks

3.1. Algorithmic Vulnerability from NMS

After the feature extractor, the model outputs candidate bounding boxes defined by key attributes of height, width, location, confidence score, and categorical probabilities [28]. Then NMS removes duplicated boxes by consolidating them with closely-matched positions into a single one, i.e., boxes with an *intersection over union* (IoU) value over a specified threshold are preserved, and IoU quantifies the overlap between two boxes relative to their union [35]. The procedures of NMS are detailed in Algorithm 1 and the interactions between the GPU-CPU are visualized in Fig. 2. Specifically, Ln 3-5 perform filtering, radixsort and pairwise IoU calculation of candidate boxes (on GPU); Ln 7-8 remove unqualified boxes by comparing with the threshold (on CPU). For each candidate box, data transfer between the GPU and CPU is required.

Vulnerability (Time Complexity). The execution time of NMS increases quadratically when the number of candidate bounding boxes exceeds a threshold N_t.

Algorithm 1 Non-Maximum Suppression (NMS)

Input: $\mathcal{B}$: candidate boxes, $\mathcal{S}$: scores, Ω_{nms}: NMS threshold.
Output: $\mathcal{R}$: NMS result, $\mathcal{S}$: updated scores after NMS.

1: **while** $B \neq \emptyset$ **do**
2: /* Operations on the GPU */
3: $\mathcal{M} \leftarrow \arg\max \mathcal{S}, \mathcal{R} \leftarrow \mathcal{R} \cup \{M\}, \mathcal{M} \leftarrow \mathcal{M}/\mathcal{B}$.
4: **for** each $b_i \in \mathcal{B}$ **do**
5: $IoU_i = IoU(M, b_i)$.
6: /* Operations on the CPU */
7: **if** $IoU_i > \Omega_{nms}$ **then**
8: $\mathcal{B} \leftarrow \mathcal{B}/b_i, \mathcal{S} \leftarrow \mathcal{S}/S_{b_i}$.
9: **end if**
10: **end for**
11: **end while**
12: **return** $\mathcal{R}, \mathcal{S}$

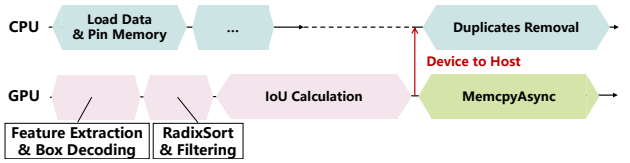


Figure 2. Interactions between GPU-CPU in the NMS.

$$T_{\text{nms}} = \begin{cases} T_{\text{base}}, & |C| \leq N_t \\ a|C|^2, & |C| > N_t \end{cases} \tag{1}$$

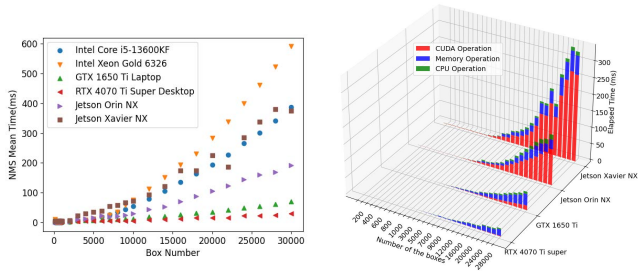
This vulnerability is exploited by [3] as an analytical basis for latency attacks. Shown in Eq. (1), the processing time remains a constant T_{base} if the box count $|C|$ is below the threshold N_t, but increasing quadratically when $|C|$ is larger than N_t with a magnifying factor a. By utilizing adversarial perturbations, the attacker's objective is to maximize the box confidence before NMS, thereby feeding it with more candidate boxes. The adversarial objective can be summarized as,

$$\mathcal{L}_{\text{adv}} = \underbrace{\max \mathcal{L}_{\text{conf}}}_{\texttt{Overload [3]}} + \rho \min(\mathcal{L}_{\text{bbox}} + \mathcal{L}_{\text{max IoU}}), \tag{2}$$
$$\texttt{PhantomSponge[28]}$$

where `Overload` maximizes the box confidence [3] and `Phantom Sponge` utilizes auxiliary losses $\mathcal{L}_{\text{bbox}}$ to reduce the area of bounding boxes for lower overall IoU and $\mathcal{L}_{\text{max IoU}}$ to preserve the detection of original objects [28]. The discussions above only provide algorithmic analysis, to unveil the impact on the computer architecture and system level, we provide a deeper understanding of how latency attacks functionalize across heterogeneous GPUs.

3.2. System-level Impacts of Latency Attacks

Our goal is to differentiate the adversarial impacts on GPUs with heterogeneous processing power as profiled in Fig. 3 with two interesting insights below.



(a) NMS processing time (b) % of different operations

Figure 3. System effects of latency attacks. a) # objects vs. NMS processing time on heterogeneous computing devices; b) percentage of GPU CUDA, memory transfer and CPU logic operations on embedded and desktop GPUs.

Observation 1 (Compute-Bound on Embedded GPUs). On resource-constrained edge devices that lack GPU cores, latency attacks make real-time processing *compute-bound*. It suggests that edge devices need to increase the number of GPU cores to tolerate such attacks.

This is justified by Fig. 3b that both Jetson devices have parabolic increase of CUDA operations with an increasing number of bounding boxes (384 vs. 1024 CUDA cores of Xavier and Orin NX, respectively). It is also cross-validated in Fig. 3a that both Intel CPUs rank the lowest if NMS is executed on the CPUs with limited logical cores.

Observation 2 (Memory-Bound on Desktop GPUs). As the number of GPU cores increases, the bottleneck is mitigated but not vanished, but gradually migrating from computation to data transfer between the GPU and CPU due to logical operations, which suggests that one should increase memory bandwidth and parallelization of CPU cores to eliminate latency attacks. The memory bottleneck becomes more prominent on multi-tenant cloud platforms since the memory bandwidth is shared and interference from users on the same physical machine would magnify the attack impact, which is validated by our experiments using A100 GPUs in Sec. 5.

The observation above is evidenced from Fig. 3b that both 1650Ti and 4070Ti GPUs are dominated by memory operations between the GPU and CPU to retain the boxes after IoU calculation. Based on these results, the number of candidate boxes into the NMS module $\mathbb{E}[f_\theta(x)]_{\text{box}}$ should be bounded by the capacity of the hardware accelerators to satisfy the real-time application requirement. Given the response time T, *e.g.*, $T = 33.3$ ms for standard rate at 30 FPS and $T = 16.6$ ms for high-end tasks at 60 FPS.

$$T_{\text{nms}} = (\alpha \frac{|C|^2}{S_{\text{IoU}}} + \beta \frac{|C|}{B}) < T - T_{\text{backbone}} \tag{3}$$

$$\mathbb{E}[f_\theta(x)]_{\text{box}} < \frac{S_{\text{IoU}}}{2\alpha} \sqrt{\frac{\beta^2}{B^2} - 4\frac{\alpha}{S_{\text{IoU}}}(T - T_{\text{backbone}})} \tag{4}$$

where S_{IoU} is the IoU processing speed of the GPU, B is the PCIe memory bandwidth between CPU-GPU, α and β are scaling factors and T_{backbone} is the processing time of the object detection backbone. Note the values of S_{IoU}, B and T_{backbone} are stable and can be obtained via system profiling. Hence, Eq. (4) provides a connection between the application-level requirements (FPS $\sim 1/T$) with the number of candidate boxes. Next, we build this hardware-adaptive relation into the learning process of robust object detectors.

4. Background-Attentive Adversarial Training

Without changing the NMS module on legacy software, adversarial training (AT) is an effective way to defend against latency attacks by fundamentally screening out the phantom objects into the NMS. It solves a min-max saddle point optimization by launching attacks in the inner maximization and minimizing the AT loss in the outer optimization [22]. Specifically, the goal is to learn θ^* under perturbations within the l_p-norm ball with radius ϵ,

$$\theta^* = \arg\min_\theta \mathbb{E}_{(x,y)\sim\mathbb{D}}\big[\max_{\|\delta\|_p \le \epsilon} \mathcal{L}(f_\theta(x+\delta), y)\big]. \tag{5}$$

However, since the existing latency attacks involve specialized loss functions (Eq. (2)), it still remains to answer which original loss function in object detectors AT should target and what spatial region AT should attend to.

4.1. Objectness Loss

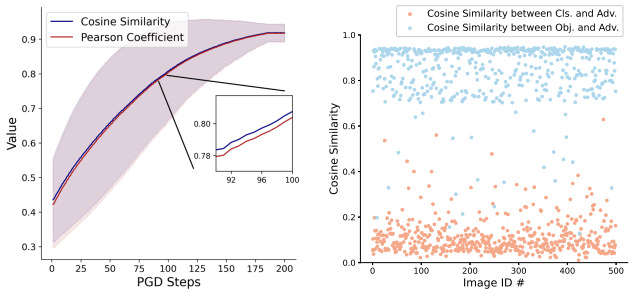
Generally, object detectors learn a function $f_\theta(x) \rightarrow \{\mathbf{p}_k, \mathbf{b}_k\}$ to predict the probability and bounding box of K objects for image $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ with class label $\mathbf{y}$. In YOLOv5, the objective loss function can be decomposed into the classification, localization and objectness losses [33],

$$\begin{aligned} \theta &= \arg\min_\theta \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y},\mathbf{b}} \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y}, \mathbf{b}) \\ &= \arg\min_\theta \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{CIoU}} + \mathcal{L}_{\text{obj}}. \end{aligned} \tag{6}$$

$\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{CIoU}}$ emphasize on different aspects of classification and localization losses, which are exploited in [40] for mis-classification and mis-location attacks. The objectness loss $\mathcal{L}_{\text{obj}}$ indicates whether a specific region contains an object in the candidate bounding box. Given the objectness confidence score $\hat{C}_i \in [0, 1]$, it can be represented by the binary cross-entropy loss ℓ_{BCE},

$$\mathcal{L}_{obj} = \sum_{i=1}^{K} \big[\mathbb{1}_i \ell_{BCE}(1, \hat{C}_i) + (1 - \mathbb{1}_i)\ell_{BCE}(0, \hat{C}_i)\big]. \tag{7}$$

Recall that the adversarial loss in Eq. (2) attempts to maximize the box confidence with auxiliary box minimization effects. Hence, we conjecture that the adversarial loss has a close relation to the objectness loss as described next.



(a) PGD iterations (b) Distr. of cosine similarity

Figure 4. Adversarial relations between $\mathcal{L}_{\text{adv}}$ in [3] and $\mathcal{L}_{\text{obj}}$. a) Pearson coefficient and cosine similarity between the two losses over the PGD iterations; b) Distribution of the cosine similarity for different images in PASCAL-VOC between $(\mathcal{L}_{\text{adv}}, \mathcal{L}_{\text{obj}})$ vs. $(\mathcal{L}_{\text{adv}}, \mathcal{L}_{\text{cls}})$.The former has stronger correlation.

Property 1 (Objectness Loss). The adversarial perturbation δ' generated from $\mathcal{L}_{\text{adv}}$ and the perturbation δ generated from $\mathcal{L}_{\text{obj}}$ are consistent, measured by the cosine similarity $\frac{f_\theta(x+\delta')\cdot f_\theta(x+\delta)}{\|f_\theta(x+\delta')\|\cdot\|f_\theta(x+\delta)\|}$, or Pearson correlation coefficient.

Fig. 4 validates this empirically by tracing the evolution of averaged cosine similarity and Pearson coefficient between $f_\theta(x+\delta')$ and $f_\theta(x+\delta)$ and the distribution of the cosine similarity for different images sampled from the VOC dataset, in which $\mathcal{L}_{\text{cls}}$ is provided as contrastive examples. It is observed that as the PGD attack progresses, the two losses converge to a highly correlated adversarial subspace and the cosine similarity concentrates within a narrow band of $[0.875, 0.95]$ for different images, in contrast to $\mathcal{L}_{\text{cls}}$ with weak correlation between $[0, 0.2]$. This suggests that $\mathcal{L}_{obj}$ can be used as an effective *proxy* in AT. Note that even if the detector does not have a separate objectness head, it could be still evidenced through the semantics to locate where objectness has been displaced, *e.g.*, such as YOLOv8 embeds objectness inside classification [34], which is also evaluated by this work.

An essential function of objectness is to differentiate objects from the background. Denote the objects as $x_{\text{obj}} = \bigcup_{k=1}^{K}(b_i)^{\mathbf{y}}$, $x'_{\text{obj}} = x_{\text{obj}} + \delta_{\text{obj}}$, and the background as $x_{\text{bg}} = x - x_{\text{obj}}, x'_{\text{bg}} = x_{\text{bg}} + \delta_{\text{bg}}$. We define *background boundary margin* as the distance for a background region to the decision boundary of becoming a "phantom object" in the pixel space $\mathcal{X}$ [39],

$$B_\theta(x'_{\text{bg}}) = \min_{x'_{\text{bg}}} \left\|x'_{\text{bg}} - x\right\|_p,$$
$$\text{s.t.} \quad \hat{C}_{\text{th}} - \hat{C}_i(x'_{\text{bg}}, b_i, y_i) = 0, \forall i \in \{1, \cdots, K\}, \quad (8)$$

where $\hat{C}_{\text{th}}$ is the score threshold between the background and object. Similarly, $B_\theta(x'_{\text{obj}})$ can be derived for generating a phantom object on a natural object.

Property 2 (Background vs. Object Boundary Margin). The background and object boundary margins have the following relation:

$$\mathbb{E}_{x\sim\mathbb{D}}[B_\theta(x'_{\text{bg}})] \leq \mathbb{E}_{x\sim\mathbb{D}}[B_\theta(x'_{\text{obj}})], \quad (9)$$

i.e., the perturbation strengths to generate phantoms on existing objects are larger than the background, $\|\delta_{\text{obj}}\|_p \geq \|\delta_{\text{bg}}\|_p$, which means that the background contains more non-robust features compared to the object regions [14].

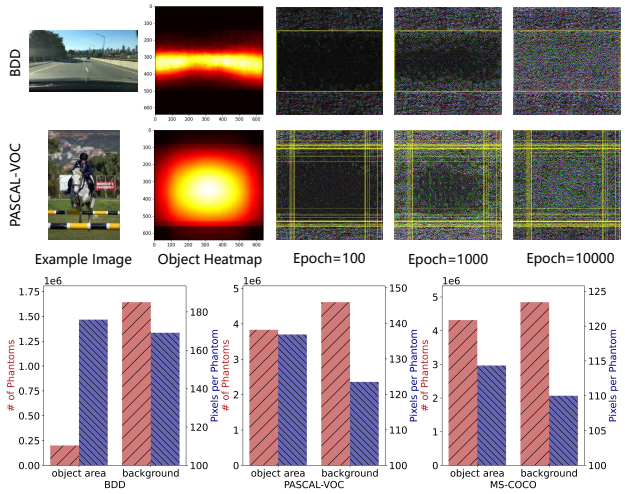


Figure 5. Top: Visualizing the attack processes on the object and background regions; Bottom: statistics of phantoms on different datasets. Fewer phantoms are generated on the objects compared to the background regions (red bars).

Fig. 5 validates this property empirically by visualizing the additive K-PGD process. It is observed that latency attacks initially generate perturbations in the background before the natural objects. *E.g.*, BDD of autonomous driving is often characterized by objects on the road so latency attacks generate objects elsewhere first. More statistics of phantoms are shown across different datasets and the background induces more phantoms with less pixel to perturb per phantom, compared to the actual objects. Since BDD usually involves small objects of vehicles and pedestrians, it is interesting to capture a sharp contrast between # of phantoms on object vs. background.

4.2. Proposed AT Method

Based on the unique adversarial behaviors, we develop a new AT method to improve generalization, which is orthogonal to the prior efforts in regularizing the adversarial loss surface [38, 44]. The key insight is to make the inner optimization attend to background regions to avoid potential overfitting on object areas that necessitate much larger perturbation strengths, which also facilitates to learn more on the *non-robust* regions [14, 37].

We adopt a binary mask $\mathbf{M} \in \{\mathbf{0}, \mathbf{1}\}^{r_x^i \times r_y^i}$ to control the amount of perturbation being injected into the inner maximization. r_x^i, r_y^i are the width and height of an object i. Our

Model	Attack	Standard			MTD			ODD			Underload*		
		VOC	COCO	BDD	VOC	COCO	BDD	VOC	COCO	BDD	VOC	COCO	BDD
YOLOv3t	Clean	**55.8**	**36.5**	**30.4**	38.3	24.9	16.1	35.8	23.9	16.9	48.5	25.8	18.3
	Daedalus	2.4	0.2	0.1	32.3	17.7	13.2	32.4	18.2	14.7	**42.8**	**19.3**	**16.1**
	Phantom	5.4	9.7	0.2	31.7	16.9	12.1	31.5	14.9	15.3	**41.3**	**18.8**	**15.6**
	Overload	5.2	4.1	0.4	29.9	16.5	12.0	29.4	16.4	16.3	**33.3**	**18.7**	**16.8**
YOLOv5s	Clean	**73.3**	**51.3**	**50.7**	57.7	40.5	34.0	55.9	38.1	33.8	68.9	43.6	34.6
	Daedalus	12.8	15.1	6.4	50.1	33.0	28.1	48.4	32.8	27.6	**50.3**	**33.4**	**28.8**
	Phantom	7.5	8.8	7.6	54.7	36.8	24.7	53.3	36.5	24.5	**61.3**	**36.9**	**25.1**
	Overload	4.5	7.9	4.7	49.4	36.5	24.5	49.2	35.8	25.8	**53.3**	**36.7**	**26.5**
YOLOv8s	Clean	**82.5**	**57.7**	**53.4**	68.8	43.6	36.1	68.7	42.2	37.1	72.6	45.7	39.3
	Daedalus	18.8	16.1	2.6	65.1	41.8	30.1	64.3	40.7	26.9	**69.7**	**43.6**	**33.2**
	Phantom	6.8	10.9	3.5	61.3	41.9	34.5	64.6	40.3	34.4	**66.1**	**42.3**	**36.5**
	Overload	7.7	20.1	5.7	53.0	40.6	34.3	62.4	39.1	34.9	**66.5**	**42.0**	**36.0**

Table 1. Defense performance (mAP50 %) under the latency attacks of Daedalus [35], Phantom Sponge [28] and Overload [3] while comparing with existing defense of MTD and OOD. The **Best** and Second Best values in each row are marked in **Red** and Blue. The first rows of "Clean" compare the clean accuracy drop with different AT methods.

Algorithm 2 Background-Attentive Adversarial Training

Input: Pre-trained model parameter θ_0, dataset $\mathcal{D}$, epochs E, batch size s, attack budget ϵ, learning rate γ, system metrics on R.H.S. of Eq. (4), bounding box step size $\Delta x, \Delta y$, maximum PGD step K.

Initialize $|C_{\max}| \leftarrow \frac{S_{\text{IoU}}}{2\alpha}\sqrt{\frac{\beta^2}{B^2} - 4\frac{\alpha}{S_{\text{IoU}}}(T - T_{\text{basenet}})}$, $r_x, r_y \leftarrow W_0, H_0$
while $\mathbb{E}[f_\theta(\mathbf{x}+\delta^M)]_{\text{box}} < |C_{\max}|$ **do**
  $\theta \leftarrow \theta_0$. ▷ re-initialize model parameter
  for epoch $\in \{1, \cdots, E\}$ **do**
    for i $\in \{1, \cdots, K\}$ **do**
      $\delta \leftarrow \epsilon \cdot \text{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}_{obj}(f_\theta(\mathbf{x}), \mathbf{y}, \mathbf{b})\right)$
      $\delta^M \leftarrow \Pi_{\|\delta\|_p \le \epsilon}(\mathbf{M} \odot \delta)$, $\mathbf{M} = \{\mathbf{0}, \mathbf{1}\}^{r_x \times r_y}$
      $\theta \leftarrow \theta - \gamma \cdot \nabla_\theta \mathcal{L}(f_\theta(\mathbf{x}+\delta^M), \mathbf{y}, \mathbf{b})$
    end for
  end for
  $r_x, r_y \leftarrow r_x, r_y + \Delta x, \Delta y$ ▷ reduce mask size
end while

Output: Obtain robust model $\theta^* \leftarrow \theta$.

goal is to learn a robust model θ^* by minimizing the overall loss function $\mathcal{L}$ defined by Eq. (6) such that the latency attacks are suppressed under the hardware capacity. The optimization is formalized as,

$$\theta^* = \arg\min_{\theta, \delta^M \in \mathcal{S}} \mathcal{L}\big(f_\theta(\mathbf{x}+\delta^M), \mathbf{b}, \mathbf{y}\big) \tag{10}$$

where

$$\mathcal{S} = \big\{\delta^M \leftarrow \mathbf{M} \odot \arg\max_{\|\delta\|_p \le \epsilon, i \in \mathcal{K}} \mathcal{L}_{obj}(f_\theta(\mathbf{x}+\delta), b_i, y_i)\big\} \tag{11}$$

$$\mathbb{E}[f_\theta(x+\delta^M)]_{\text{box}} < \frac{S_{\text{IoU}}}{2\alpha}\sqrt{\frac{\beta^2}{B^2} - 4\frac{\alpha}{S_{\text{IoU}}}(T - T_{\text{backbone}})} \tag{12}$$

$$\mathbf{M} = \{\mathbf{0}, \mathbf{1}\}^{r_x^i \times r_y^i}, r_x^i \in [0, W_b^i], r_y^i \in [0, H_b^i], i \in \mathcal{K}. \tag{13}$$

Eq. (11) finds the set of background attentive adversarial perturbations $\mathcal{S}$ targeting the objectness loss, in which $\odot$ is the Hadamard product. Eq. (12) states that the number of candidate boxes in the object detector generated by masked perturbation δ^M should be less than the hardware capacity obtained from Eq. (4). This successfully connects the optimization process with the hardware capacity on edge systems with a stopping criteria. Eq. (13) bounds the 0-1 mask generation with the width W_b^i and height H_b^i of an object.

Property 3 (Monotonicity). The box count $\mathbb{E}[f_\theta(x+\delta^M)]_{\text{box}}$ is monotonously increasing regarding the mask size $r_x^i \times r_y^i$.

Based on this property, we develop an efficient algorithm to learn a robust model θ^* as long as the box count is below hardware capacity. The details are shown in Algorithm 2. Starting with a pre-trained model, we initialize the mask sizes to the maximum size of $W_b \times H_b$. Then we perform E epochs of AT before we reduce the mask with step sizes of $(\Delta x, \Delta y)$. The iteration stops until we reach the hardware capacity. More experiments of the robust vs. clean accuracy are available in the next section.

5. Experiments

5.1. Experimental Settings

Datasets & Models. We conduct experiments on three standard datasets including PASCAL-VOC [8], MS-COCO [17], and Berkeley DeepDrive (BDD) [42]. PASCAL-VOC and MS-COCO are common benchmarks in object detection, while BDD is frequently used for autonomous driving. We follow the standard "07+12" protocol on VOC. For MS-COCO, we train on the `train+valminusminival` 2017 set and test on the `minival 2017 test`. For BDD, we use BDD100K with 70K training and 10K testing samples.

We evaluate three models in our evaluation including YOLOv3 [24], YOLOv5 [33] and YOLOv8 [34]. The ex-

isting attacks primarily target anchor-based models such as YOLOv3 and YOLOv5 [3, 28]. Our work extends this to cover the latest anchor-free YOLOv8 models [34], which provides a holistic coverage of mainstream object detectors for edge devices across different YOLO generations.

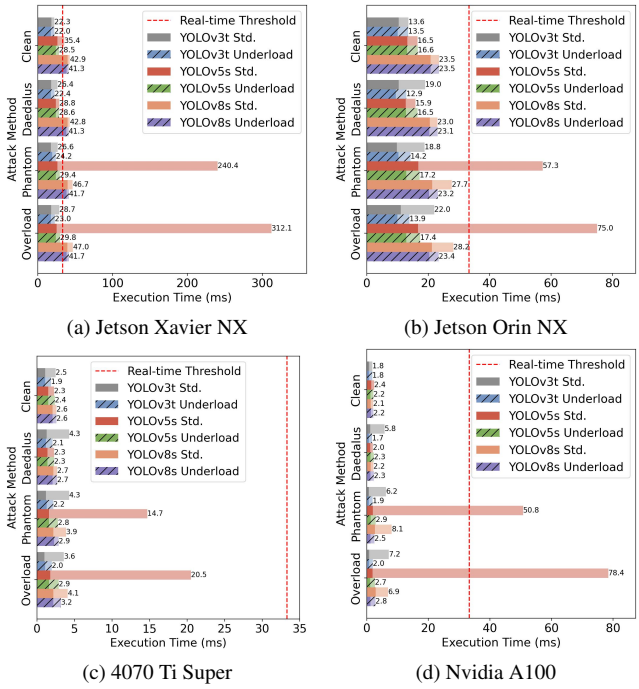
Attack and Defense Settings. We evaluate our defense against three existing attacks: `Daedalus` [35], `Phantom Sponge` [28] and `Overload` [3]. We set the hyperparameters following [28], $\lambda_1 = 1, \lambda_3 = 0$ with the l_2 norm and utilize K-step PGD ($K = 4$) with perturbation magnitude of $\epsilon = 1/255$ for AT [22]. Starting from the pre-trained model, we employ AdamW [19] as the optimizer with scheduled learning rate and follow [32] to set other hyperparameters for reproducibility.

We compare the proposed `Underload` with the SOTA defense in object detection: the multi-task domain defense (MTD) [43] and objectness-oriented defense (OOD) [13]. MTD leverages the asymmetric role of task losses for improving robustness with AT. OOD develops an attack against the objectness loss and performs AT in an identical manner to MTD.

5.2. Robustness Evaluation

The primary results are available in Table 1 measured by mAP50, a metric representing the average precision across all classes at the 50% IoU threshold. For each model, the rows represent the model accuracy under "Clean" (no attack) followed by three latency attacks. In each column, we compare the defense performance of Standard (no defense), MTD and OOD with `Underload`. Although MTD is not specifically tailored against latency attacks, it retains certain defensive capabilities due to the intertwined nature of loss functions in multi-tasking object detectors.

By tracing through the values marked in red color, we can see that `Underload` demonstrates superior performance in both clean and robust accuracy compared to existing defense. *E.g.*, on YOLOv5s, `Underload` brings the robust accuracy from 7.5% back to 61.3% under the `Phantom Sponge` attack and from 4.5% back to 53.3% under the `Overload` attack for VOC, with only 4.4% drop of clean accuracy, compared to 15.6% and 17.4% drop using MTD and OOD. This is because `Underload` has taken active measures to only inject useful perturbations into the AT process. MTD could screen out some attacks but its performance is not stable on the VOC-YOLOv3t and VOC-YOLOv8s pairs. OOD does not consider the balance between clean accuracy and robust accuracy, thus suffering from more clean accuracy drops compared to `Underload`. BDD dataset features small objects such as vehicles and pedestrians on the road that leave a large background space open for the attack – `Underload` also achieves 1-3% higher clean/robust accuracy under these hard-to-defense cases than the benchmarks.



(a) Jetson Xavier NX (b) Jetson Orin NX

(c) 4070 Ti Super (d) Nvidia A100

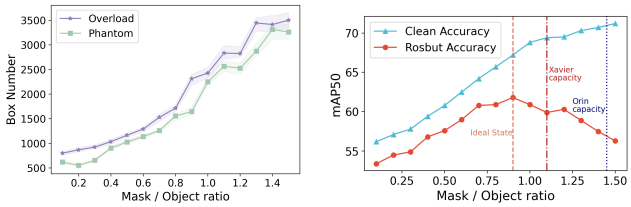
Figure 6. Execution time (ms) across heterogeneous GPUs under different attacks and effectiveness of our defense.

5.3. Evaluation on Heterogeneous GPUs

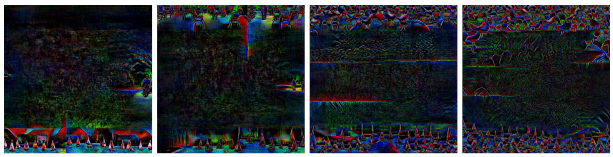
Next, we demonstrate how our defense restores the real-time processing capabilities across heterogeneous devices of edge, desktop and server-grade GPUs. Fig. 6 shows the total execution time of the basenet and NMS on VOC. First, we can see that `Phantom Sponge` and `Overload` are the two strongest attacks that successfully push processing time over the real-time threshold (30 FPS), even on A100 GPUs. This validates that the true bottleneck migrates from computation to memory on high-end GPUs, which is still a serious problem though the computational power is sufficiently high. Further, when memory bandwidth is shared on multi-tenant cloud platforms, latency attacks become more effective on A100 by comparing with the 4070Ti's single-tenant setup. Fortunately, our defense restores the processing time of all the cases close to the original states (clean). *E.g.*, on Jetson Orin NX, we successfully restore the latency back to 13, 17, and 23 ms for different YOLO generations that meet the 30 FPS requirements for most end-user applications.

5.4. Ablation Study of Mask Size

Mask size plays a pivotal role to steer the attention of AT, thus potentially balancing the clean and robust accuracy. Fig.7 illustrates this relation by examining the mask/object ratio, which varies from 10% to 150% on the x-axis. First, it is obvious from Fig.7a that the box number increases with the mask size (unprotected area). In Fig.7b, the clean accu-



(a) Box # vs. Mask Ratio (b) Clean vs. Robust Acc



(c) Perturbation generated by Overload for different ratios.

Figure 7. Ablation study of mask size for clean accuracy and robustness on PASCAL-VOC.

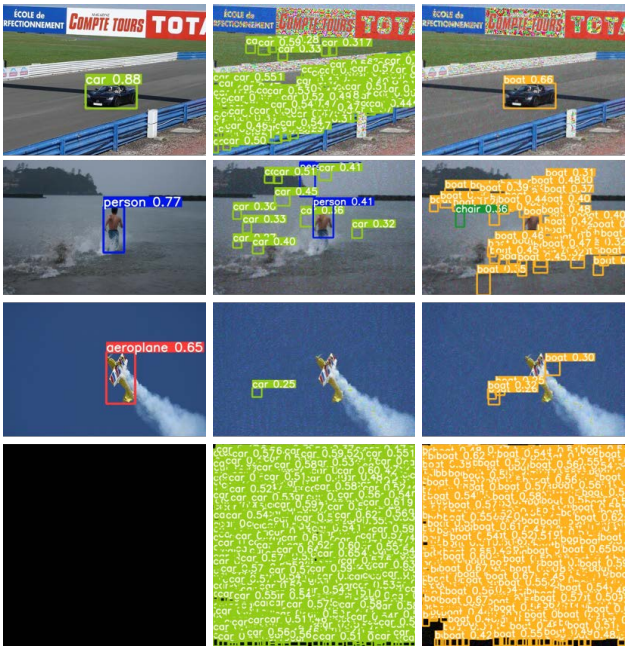
racy climbs continuously because larger mask sizes reduce the amount of perturbations injected into the inner optimization, that helps improve generalization reflected on the clean accuracy. Robust accuracy exhibits a more interesting pattern, which increases with the mask/object ratio and peaks around 0.9 for VOC. This is counter-intuitive as the initial segments of the robust curve should have decreased when the protected area shrinks (increment of mask size).

We conjecture that if a mask is too small compared to natural objects, it is difficult to generate phantom objects in the mask area. Hence, the mask size could be raised close to the contour of natural objects for maximizing the attack capacity but not sacrificing the clean accuracy too much. Fig.7c visualizes the phantom perturbation generated from the mask ratio of $\{0.1, \cdots, 1.3\}$. We can see that phantoms of repetitive patterns are being generated in the letterbox area first. Background regions with a larger mask ratio are filled with more phantoms because the unprotected area is also larger.

5.5. Background and Object Semantics

Finally, we expose intriguing artifacts between the background and object semantics through the lens of *targeted attacks*, that echo with our design of background-attentive AT. The goal is to launch targeted attacks to generate phantom objects of "cars" and "boats" under different background regions to different images in Fig. 8. For image #1, it is much easier to generate more "cars" on the track than "boats" and the opposite is true for #2. For image #3, it is a bit easier to generate more "boats" in the sky than "cars", due to the blue color of the sky similar to the ocean. For image #4, it is equally possible to generate as many "cars" or "boats" since the blank image contains no semantic information.

From the attacker's perspective, the attack success rate can be improved by targeting the closest semantics to the background region, *E.g.*, generating phantom "birds" in



(a) Original image (b) Targeting "car" (c) Targeting "boat"

Figure 8. Validating the background and object semantics by *targeted attacks* of creating phantom "cars" and "boats": "#1: a racing car on the track", "#2: a person on the beach", "#3: an aeroplane in the sky", and "#4: a black image with no information". More "cars" are generated than "boats" on the road and opposite is found on the beach. The blank image is used for comparison as it contains no semantic information.

the "sky", or "persons" on the "road", which not only requires less perturbation (stealthier), but also generates more phantoms with higher latency in the NMS. This is because the semantically close objects/background are more vulnerable to adversarial attacks that slightly push the objects/background region over the manifold to be adversarial. From the defense perspective, with prior knowledge about the application domain (training set), it is also possible to finetune AT against specific semantic classes (vulnerable classes).

6. Conclusion

This work restores the real-time processing capabilities back to object detectors under latency attacks. We leverage background-attentive AT to focus more on the vulnerable background regions, and bring hardware capacity into the robust learning process. The extensive experiments demonstrate effectiveness to defend against latency attacks on various datasets, models and GPUs. We also acknowledge transformer-based detectors (DETR) that are NMS-free [2, 7, 48] and their potential immunity to latency attacks. For more details on latency attacks against DETR, we refer interested readers to the supplement materials.

7. Acknowledgement

This work is supported in part by NSF of Zhejiang Province LZ25F020007, NSFC 62394341, 92467301, 62293511, the Fundamental Research Funds for the Central Universities 226202400182 and the ZJUCSE-Enflame Cloud and Edge Intelligence Joint Laboratory.

References

[1] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10231–10241, 2021. 1

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2, 8, 1

[3] Erh-Chung Chen, Pin-Yu Chen, I Chung, Che-Rung Lee, et al. Overload: Latency attacks on object detection for edge devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24716–24725, 2024. 1, 2, 3, 5, 6, 7

[4] Simin Chen, Cong Liu, Mirazul Haque, Zihe Song, and Wei Yang. Nmtsloth: understanding and testing efficiency degradation of neural machine translation systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1148–1160, 2022. 3, 1

[5] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19830–19843, 2023. 2

[6] Ka-Ho Chow, Ling Liu, Margaret Loper, Juhyun Bae, Mehmet Emre Gursoy, Stacey Truex, Wenqi Wei, and Yanzhao Wu. Adversarial objectness gradient attacks in real-time object detection systems. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 263–272. IEEE, 2020. 1

[7] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2988–2997, 2021. 2, 8

[8] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111:98–136, 2015. 6

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer vision and Pattern Recognition*, pages 580–587, 2014. 1, 2

[10] Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. Ilfo: Adversarial attack on adaptive neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14264–14273, 2020. 3, 1

[11] Jiabo He, Sarah Erfani, Xingjun Ma, James Bailey, Ying Chi, and Xian-Sheng Hua. α-iou: A family of power intersection over union losses for bounding box regression. *Advances in Neural Information Processing Systems*, 34:20230–20242, 2021. 2

[12] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4507–4515, 2017. 1

[13] Jung Im Choi and Qing Tian. Adversarial attack and defense of yolo detectors in autonomous driving scenarios. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1011–1017. IEEE, 2022. 2, 7, 1

[14] Junho Kim, Byung-Kwan Lee, and Yong Man Ro. Distilling robust and non-robust features in adversarial examples by information bottleneck. *Advances in Neural Information Processing Systems*, 34:17148–17159, 2021. 2, 5

[15] Yuezun Li, Daniel Tian, Ming-Ching Chang, Xiao Bian, and Siwei Lyu. Robust adversarial perturbation on deep proposal-based models. *BMVC*, 2018. 1

[16] Mingfu Liang, Jong-Chyi Su, Samuel Schulter, Sparsh Garg, Shiyu Zhao, Ying Wu, and Manmohan Chandraker. Aide: An automatic data engine for object detection in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14695–14706, 2024. 1

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 6

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. 2

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2018. 7

[20] Yan Lu, Zhun Zhong, and Yuanchao Shu. Multi-View Domain Adaptive Object Detection in Surveillance Cameras. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023. 1

[21] Chen Ma, Ningfei Wang, Qi Alfred Chen, and Chao Shen. Slowtrack: Increasing the latency of camera-based perception in autonomous driving using adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4062–4070, 2024. 1, 2, 3

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018. 4, 7

[23] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on*

Pattern Recognition (ICPR'06), pages 850–855. IEEE, 2006. 2

[24] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2, 6, 1

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1, 2

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016. 1, 2

[27] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019. 2

[28] Avishag Shapira, Alon Zolfi, Luca Demetrio, Battista Biggio, and Asaf Shabtai. Phantom sponges: Exploiting non-maximum suppression to attack deep object detectors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4571–4580, 2023. 1, 2, 3, 6, 7

[29] Jiang Shiqi, Lin Zhiqi, Li Yuanchun, Shu Yuanchao, and Liu Yunxin. Flexible High-resolution Object Detection on Edge Devices with Tunable Latency. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2021. 1

[30] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 212–231. IEEE, 2021. 1, 3

[31] Zanjia Tong, Yuhang Chen, Zewei Xu, and Rong Yu. Wise-iou: bounding box regression loss with dynamic focusing mechanism. *arXiv preprint arXiv:2301.10051*, 2023. 2

[32] Ultralytics. default hyp in yolo. https://github.com/ultralytics/yolov5/tree/master/data/hyps, 2022. Accessed: 2024-7-16. 7

[33] Ultralytics. Yolov5. https://github.com/ultralytics/yolov5, 2022. Accessed: 2024-4-26. 1, 2, 4, 6

[34] Ultralytics. Yolov8. https://github.com/ultralytics/ultralytics, 2024. Accessed: 2024-7-16. 1, 2, 5, 6, 7

[35] Derui Wang, Chaoran Li, Sheng Wen, Qing-Long Han, Surya Nepal, Xiangyu Zhang, and Yang Xiang. Daedalus: Breaking nonmaximum suppression in object detection via adversarial examples. *IEEE Transactions on Cybernetics*, 52 (8):7427–7440, 2021. 1, 2, 3, 6, 7

[36] Kunyu Wang, Xueyang Fu, Yukun Huang, Chengzhi Cao, Gege Shi, and Zheng-Jun Zha. Generalized uav object detection via frequency domain disentanglement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1064–1073, 2023. 1

[37] Yifei Wang, Liangchen Li, Jiansheng Yang, Zhouchen Lin, and Yisen Wang. Balance, imbalance, and rebalance: Understanding robust overfitting from a minimax game perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 5

[38] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems*, pages 2958–2969. Curran Associates, Inc., 2020. 5

[39] Yanru Xiao and Cong Wang. You see what i want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1934–1943, 2021. 5

[40] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017. 4

[41] Mingjun Yin, Shasha Li, Chengyu Song, M Salman Asif, Amit K Roy-Chowdhury, and Srikanth V Krishnamurthy. Adc: Adversarial attacks against object detection that evade context consistency checks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3278–3287, 2022. 1

[42] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020. 6

[43] Haichao Zhang and Jianyu Wang. Towards adversarially robust object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 421–430, 2019. 2, 7, 1

[44] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 7472–7482. PMLR, 2019. 5

[45] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen. Dynamic r-cnn: Towards high quality object detection via dynamic training. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 260–275. Springer, 2020. 2

[46] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16965–16974, 2024. 2, 1

[47] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybernetics*, 52(8):8574–8586, 2021. 2

[48] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable trans-

formers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2, 8

[49] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023. 2

Can't Slow Me Down: Learning Robust and Hardware-Adaptive Object Detectors against Latency Attacks for Edge Devices

Supplementary Material

A. More examples of NMS and Latency Attacks

During the training phase, object detectors such as YOLO [24, 33] and Faster-RCNN [26] usually apply a many-to-one matching strategy, that the prediction results contain multiple detection boxes for the same object with redundancy. The NMS module removes this redundancy by reducing the number of detection boxes to balance the precision and recall. As shown in Fig. 9, the model predicts a number of boxes to detect the object of cat. The box number is related to the hyperparameters such as the number of anchors. The initial confidence filtering removes the most irrelevant background bounding boxes.

The primary goal of latency attacks is to ensure that the confidence scores of most bounding boxes exceed the confidence threshold. Unlike another type of attack that solely targets model efficiency [4, 10], latency attacks on object detectors not only reduces the processing speed, but also the detection accuracy. Therefore, its defense carries greater practical value.

B. When DETR meets Latency Attacks

DETR (Detection Transformer) utilizes the Hungarian algorithm for one-to-one matching between predicted boxes and ground truth boxes, enforcing a strict limit on the number of detection boxes (e.g., 100 boxes) [2]. Intuitively, DETR should be agnostic to the number of objects. In terms of robustness to perturbations, the previous works also suggest that transformers such as ViT exhibit much higher robustness against gradient-based PGD attacks compared to CNN models [1].

These have collectively led to the following question: *whether the DETR families have the similar latency attack surface as the CNN-based object detectors?* To answer this question, we analyze the performance variations with the number of instances ranging from $[0, 100]$. We also investigate the latest advance from RT-DETR (Real-Time Detection Transformer) [46] under the pressure of latency attacks.

We select images with a single instance from three categories as the candidates, placing each image into an $N \times N$ grid to generate images with N^2 instances. We employ DETR and RT-DETR to perform inference on these images 100 times and examine the average execution time on Nvidia 4070Ti Super and Jetson Orin NX in Fig 10.

The preliminary experimental results show that execution time does not vary significantly with different number of instances. From an architectural design perspective, the stability is because DETR predicts all objects from end-to-end without an additional hand-craft NMS module for redundant box elimination. Therefore, it is tempting to conclude that, as long as the matching threshold/number of boxes has been set under the hardware capacity, DETRs do not expose the same vulnerability to latency attacks as their CNN counterparts.

C. More Details of Experimental Settings

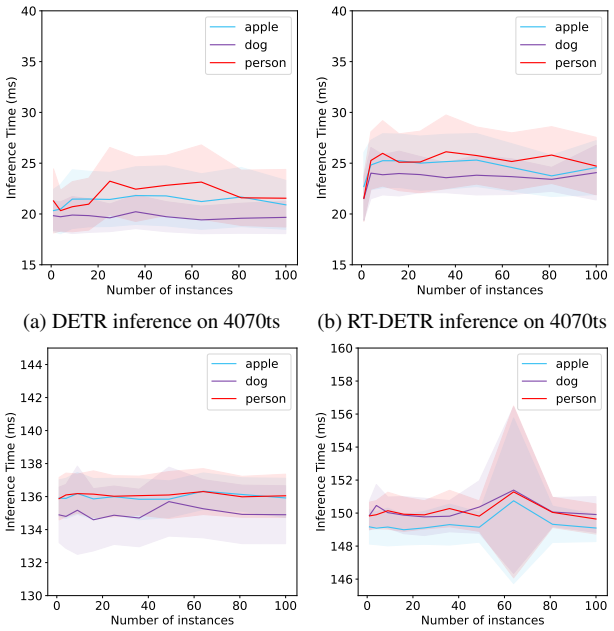
We perform all the AT on a workstation equipped with 8 Nvidia RTX 4090 GPUs, Intel Xeon Gold 6326 CPUs and 480 GB of RAM. The Python environment used for training and validation is configured with Python 3.9, PyTorch 2.0.0, Torchvision 0.15.0, and CUDA version 11.7. The edge device utilized Python 3.8, PyTorch 2.0.0, Torchvision 0.15.0, and CUDA version 11.4. The AT implementation for YOLOv3 and YOLOv5 uses the Ultralytics-YOLOv5 interface [33], while YOLOv8 adopts the Ultralytics interface [34]. Hyperparameter selection are described in the main text of the paper, and the experiments are repeated for 5 times.

D. Analysis of Training Process

As described before, the AT process initiates from the pretrained model available via the Ultralytics Github site[1]. We observe the training and validation losses of $\mathcal{L}_{\text{box}}, \mathcal{L}_{\text{obj}}, \mathcal{L}_{\text{cls}}$ with the mAP50/95 in Fig.11. We can see that the loss of the proposed Underload is less than the other two AT methods of MTD [43] and OOD [13] on $\mathcal{L}_{\text{box}}, \mathcal{L}_{\text{cls}}$ except $\mathcal{L}_{\text{obj}}$ because Underload employs objectness loss as an adversarial proxy. Injecting adversarial perturbations in the inner optimization against the objectness loss makes the outer minimization more difficult to learn with a larger training loss (but it is below the OOD loss). In the last two columns, the mean average precision of Underload is much higher than both of MTD and OOD from the beginning. The Precision/Recall curves in Fig. 12 reveal that the AT methods do not impact any specific category. However, because Underload considers the trade-off between clean and robust accuracy, the accuracy drop is mild and even negligible for some categories (e.g., the accuracy for the bicycle category only drops 0.001, compared to the drop of 0.082 and 0.095 for MTD and OOD).

[1] https://github.com/ultralytics/yolov5

Figure 9. Visualization of the entire NMS process: a) original image; 2) pre-processed results with all the prediction boxes; 3) box filtering with confidence threshold; 4) additional filtering with IoU threshold; 5) Final detection result.



(a) DETR inference on 4070ts (b) RT-DETR inference on 4070ts

(c) DETR inference on Orin NX (d) RT-DETR inference on Orin NX

Figure 10. DETR and RT-DETR inference time evaluation on different devices.

l_2-norm	Attack	Standard	MTD	OOD	**Underload***
10	Daedalus	**73.0**	57.6	55.8	68.7
	Phantom	**72.1**	57.7	55.7	68.7
	Overload	**71.8**	57.7	55.6	68.6
20	Daedalus	**70.9**	57.5	55.6	68.4
	Phantom	**70.2**	57.4	55.6	68.3
	Overload	66.1	57.2	55.5	**68.6**
30	Daedalus	66.6	57.2	55.3	**67.9**
	Phantom	64.2	57.4	55.8	**67.6**
	Overload	51.2	57.0	55.7	**68.4**
50	Daedalus	41.1	55.0	53.5	**62.0**
	Phantom	32.9	56.2	54.7	**65.4**
	Overload	17.1	54.0	54.8	**59.3**
70	Daedalus	12.8	50.1	48.4	**50.3**
	Phantom	7.5	54.7	53.3	**61.3**
	Overload	4.5	49.4	49.1	**53.3**

Table 2. Variation of the attack strengths in terms of l_2-norm for YOLOv5s. The **bold** l_2-norm is the default parameter used in the main text. The **Best** and Second Best values in each row are marked in **Red** and Blue.

E. Hyperparameter Tuning

We provide additional experiments of two key hyperparameters: *attack strength*, measured by the l_2-norm, and *balance between precision/recall*, measured by the IoU threshold Ω_{nms}.

E.1. Attack Strength

To assess attack strength, we select the maximum l_2-norm in latency attack [3, 28, 35] for Underload to ensure that our defense remains effective under challenging conditions. As shown in Table 2, the l_2-norm is set between $[10, 70]$. We find that at lower strengths when l_2-norm equals to 10 and 20, latency attacks have minimal impact on the accuracy of both standard (unprotected) and robust models (Underload AT), since the attack strength is insufficient to push a background region across the boundary margin to generate phantom objects.

However, when l_2-norm reaches 30, it starts to affect the accuracy of the standard model. When the l_2-norm exceeds 50, the accuracy of the unprotected model declines sharply to single digits when l_2-norm increases to 70. On the other hand, we observe that the robust accuracy maintains above the 60% mAP level for most of the attack strength and is still above 50% under the maximum l_2-norm of 70. In comparison to MTD and OOD, our approach achieves an accuracy of 0.2% to 10.7% improvements under different attack strengths.

E.2. Balance between Precision/Recall

The Intersection over Union (IoU) threshold is an important parameter that balances the precision and recall in object detection. A higher IoU threshold during the NMS process retains fewer candidate boxes, which reduces the occurrence of false positives and enhances the model's precision. However, setting the IoU threshold too high may inadvertently remove some true positives, thereby reducing the recall. Conversely, reducing the IoU threshold can enhance recall by keeping more candidate boxes, but it also leads to

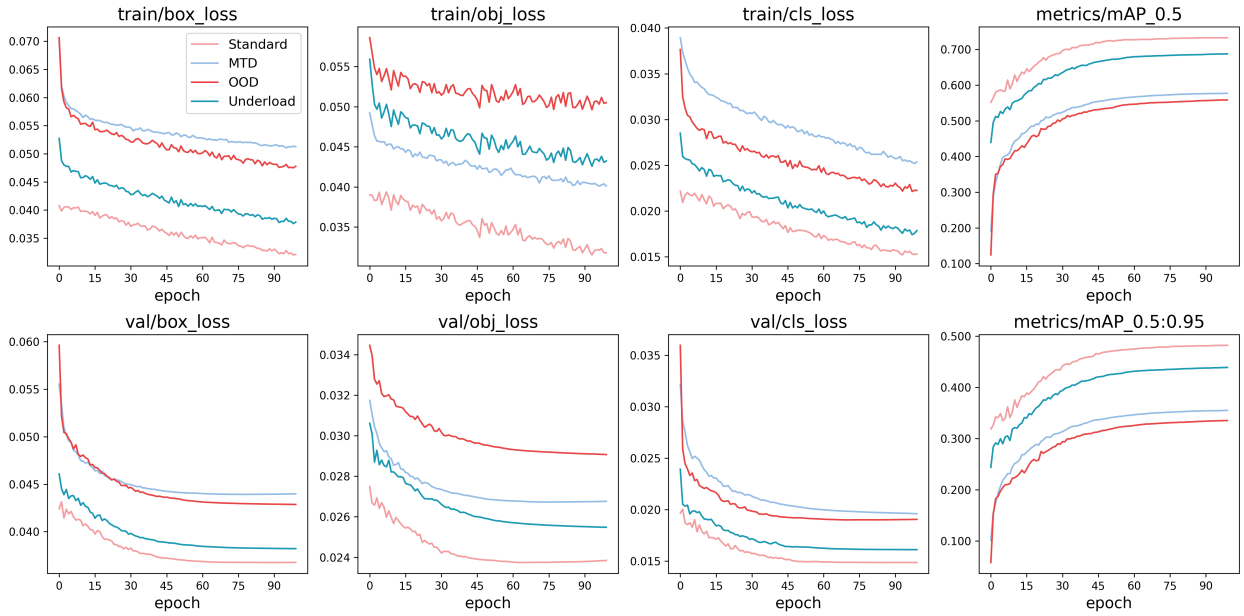


Figure 11. Visualization of the training process of YOLOv5s on the PASCAL-VOC dataset.

Ω_{nms}	Attack	Standard	MTD	OOD	**Underload***
0.30	Clean	**72.7**	58.1	57.4	68.8
	Daedalus	12.7	**50.4**	49.5	49.9
	Phantom	7.2	54.8	54.7	**61.2**
	Overload	4.1	49.1	50.3	**52.6**
0.45	Clean	**73.6**	58.7	57.8	69.5
	Daedalus	13.0	**50.9**	50.0	50.5
	Phantom	7.4	55.6	55.1	**61.9**
	Overload	4.3	49.7	50.6	**53.4**
0.60	Clean	**73.3**	57.7	55.9	68.9
	Daedalus	12.8	50.1	48.4	**50.3**
	Phantom	7.5	54.7	53.3	**61.3**
	Overload	4.5	49.4	49.1	**53.3**
0.75	Clean	**71.4**	53.7	50.7	65.7
	Daedalus	12.1	46.6	43.7	**47.5**
	Phantom	7.4	51.1	48.5	**58.6**
	Overload	4.4	46.6	45.1	**51.3**
0.9	Clean	**62.5**	39.6	35.1	52.7
	Daedalus	12.0	33.9	29.5	**36.9**
	Phantom	6.3	37.6	33.7	**46.5**
	Overload	3.7	35.0	31.6	**41.1**

Table 3. Variations of the IoU threshold Ω_{nms} in YOLOv5s. The **bolded** Ω_{nms} is the default parameter used in the main text. The **Best** and Second Best values in each row are marked in **red** and blue. The first row of "Clean" compares the clean accuracy drop with different AT methods under different Ω_{nms}.

an increase in overlapping detection, which ultimately decreases the precision.

Under latency attacks, the IoU threshold can be adjusted to simulate various attack scenarios. Setting the IoU threshold to 1.0 keeps all the candidate boxes (no box is removed), thereby retaining the artifacts produced by the latency attack that emulates the worst-case scenario. We evaluate five IoU thresholds ranging from 0.3 to 0.9, with an increment of 0.15, covering a wide range of IoU threshold values. We observe that both the clean and robust accuracy of the standard and robust models exhibit an initial increase followed by a decrease as the IoU threshold increases. Among the selected IoU thresholds, the highest accuracy occurs at an IoU threshold of 0.45. At this threshold, the reduction of Underload in clean accuracy is minimal, with a decrease of only 4.1%. In most cases, the robust accuracy of the Underload outperforms the other two AT methods. However, there are a few outliers in the low IoU cases (when the IoU threshold is 0.3 or 0.45), the robust accuracy of the MTD exceeds that of the Underload. We conjecture that it is due to the Daedalus method, which simultaneously optimizes the confidence and size of the phantom objects, may generate some high-confidence and large-area phantoms (compared with other latency attacks). When the IoU threshold is low, these phantoms can interfere with natural objects, leading to a reduction in the robust accuracy.

F. Framework and Hardware Optimization under Latency Attacks

In addition to PyTorch implementation, we also convert YOLOv5s and YOLOv8s to other implementations including ONNX and TensorRT for specialized acceleration. For

Model	Device	Attack	Standard		**Underload***	
			ONNX	TensorRT	ONNX	TensorRT
YOLOv5s	1650Ti Laptop	Clean	19.4	14.1	19.0	14.2
		`Overload`	69.2	64.9	18.7	14.0
	4070Ti Super	Clean	7.4	6.3	7.2	6.2
		`Overload`	31.1	28.6	7.0	6.3
	Jetson Orin NX	Clean	30.3	19.2	30.1	20.0
		`Overload`	100.8	87.7	29.8	19.9
	Jetson Xavier NX	Clean	57.7	30.8	57.6	30.0
		`Overload`	447.2	418.5	56.0	31.0
YOLOv8s	1650Ti Laptop	Clean	23.1	13.8	23.0	14.0
		`Overload`	25.3	16.0	23.1	13.9
	4070Ti Super	Clean	8.3	2.9	8.4	3.0
		`Overload`	9.4	4.0	8.3	2.8
	Jetson Orin NX	Clean	30.1	18.2	30.0	18.0
		`Overload`	33.2	21.7	30.3	18.1
	Jetson Xavier NX	Clean	58.1	42.2	58.0	40.0
		`Overload`	59.8	45.7	57.6	40.3

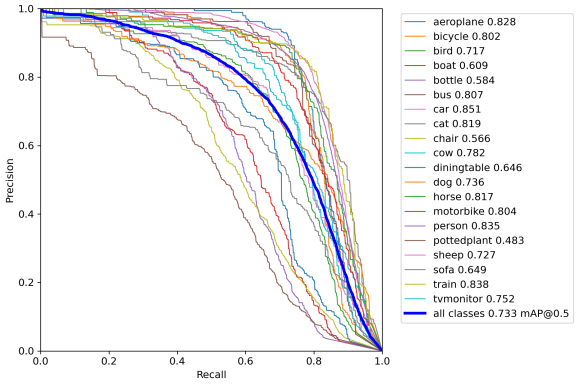
Table 4. Different frameworks of YOLOv5s and YOLOv8s model inference time (ms) in FP32.

Package	Desktop Ver.	Edge Device Ver.
CUDA	11.7	11.4
Ultralytics	8.2.100	8.2.100
ONNX	1.14.0	1.17.0
ONNXRuntime	1.16.0	1.16.0
TensorRT	8.6.0	8.5.2

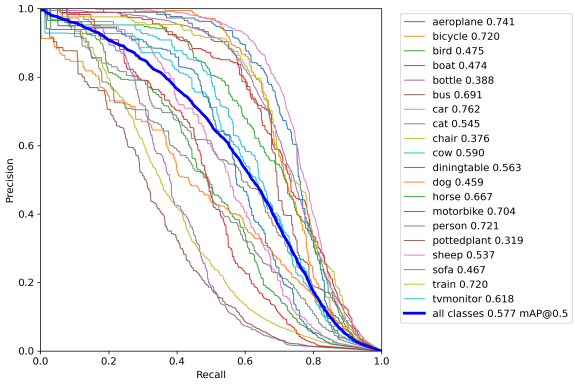
Table 5. ONNX and TensorRT models inference environments.

reproducing purposes, the environment setup is shown in the Table 5. We employ the same method to attack the implementations of ONNX and TensorRT. In details, we export ONNX and TensorRT models using the official implementation, which convert only the backbone without utilizing the `INMSLayer` or `EfficientNMSPlugin`. Other configurations remain the same as Sec. 5.

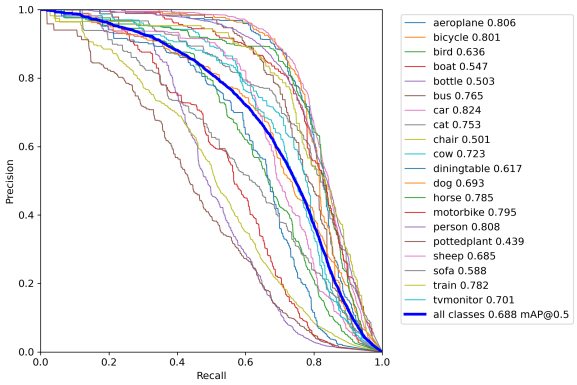
From Table 4, we find that latency attacks affect models across different implementations even for TensorRT. Despite of hardware-specific optimizations, the execution time still increases by $1.1 - 13.5\times$ under the `Overload` attack. In TensorRT, `EfficientNMSPlugin` is also vulnerable because as per to our evaluation, its execution time increases with the number of candidate boxes. Fortunately, the AT models exported from our `Underload` defense is able to defend against the corresponding latency attacks and portable to different frameworks and edge devices.



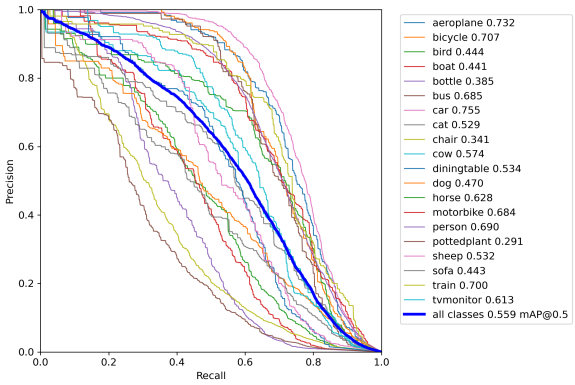
(a) Standard



(b) MTD



(c) Underload



(d) OOD

Figure 12. Precision/Recall curves for different AT methods.