Bit-Flip Induced Latency Attacks in Object Detection

Manojna Sistla[1,*], Yu Wen[1,*], Aamir Bader Shah[1], Chenpei Huang[1], Lening Wang[2], Xuqing Wu[1], Jiefu Chen[1], Miao Pan[1], Xin Fu[1,†]
[1]University of Houston [2]Prairie View A&M University

Abstract

Deep learning and computer vision have experienced significant advancements, particularly in critical applications such as autonomous driving and real-time surveillance, where object detection (OD) plays a pivotal role. Ensuring the accuracy and speed of these systems is paramount to prevent accidents or failures. Recently, latency-based attacks have emerged as a new threat, driven by the essential need for real-time performance in various applications. These attacks target model responsiveness to disrupt system performance without necessarily compromising accuracy. Our preliminary experiments show that introducing just a few bit flips to key parameters in OD models can significantly increase latency, degrading performance. Meanwhile, recent advancements in memory-based attacks, such as Row Hammer [18], demonstrate the ability to conveniently introduce bit flips at desired locations without physical hardware interaction. Based on the observations, we propose a novel attack on OD models that leverages row-hammer to introduce bit-flips via side channels, targeting the non-maximum suppression (NMS) filter and significantly increasing latency. Unlike previous methods that modify input data, our technique ensures efficiency by minimizing bit-flips through critical path exploitation and achieves practical applicability with only a subset of validation data. Experiments across various datasets and models validate our approach, demonstrating latency increases up to 71.6 *ms* (20.4×) *with just* 31 *bit-flips.*

1. Introduction

The fields of deep learning and computer vision have garnered significant attention for their pivotal roles in advancing intelligent systems, particularly in domains like autonomous driving and real-time surveillance, where OD stands as an indispensable component tasked with crucial responsibilities [41]. Ensuring the accuracy and safety of these systems demands algorithms capable of achieving high precision within constrained timeframes. Failure to meet these standards can lead to catastrophic consequences, such as traffic accidents or the failure of surveillance systems [22]. Therefore, research into the performance and stability of object detection algorithms is necessary to ensure the reliability and safety of intelligent systems.

OD algorithms such as Faster-RCNN [9], Mask-RCNN [12], and YOLO [31] have gained significant popularity recently due to their wide range of applications. However, alongside their widespread adoption, there has been considerable research into the vulnerabilities inherent to these models. A significant portion of these studies focused on attacking the correctness of the models, employing various tactics such as object generation, misclassification, and omission [2, 37]. This type of attack is known as the integrity-based attack, wherein the targeted model can be subjected to malicious behavior by introducing adversarial noise or specific patterns of adversarial patches [7, 38] into the inputs. Alternatively, backdoors can be implanted into the models, triggered by certain stimuli such as stickers or random objects in the input [23].

Recently, latency-based attacks have gained popularity due to their simplicity, widespread impact, and concealment. Unlike integrity-based attacks, which target the correctness of model outputs, these attacks focus on the timing or responsiveness of models. Adversarial attacks, which involve subtly modifying input data to deceive models, have been the primary focus of studies aiming to increase latency. However, a new type of attack has recently gained traction among deep learning models, where adversaries exploit hardware vulnerabilities such as row-hammer [18] to manipulate model functionality by attacking their parameters. By inducing bit-flips into the model parameters during inference via row-hammer, these attacks do not require access to training data and can be launched from side channels without directly accessing the model parameters, thus enhancing their stealth and potency.

Motivated by this concept, we have developed a novel adversarial attack based on bit-flips in OD models via row-hammer to induce bit-flips in the model parameters, thereby increasing the number of candidates passed to the NMS fil-

[*]Co-first authors
[†]Corresponding author: xfu8@central.uh.edu

ter, unlike previous attacks, where we explore the model architecture to identify parameters that maximize latency without affecting actual object detection. Additionally, we enhance our attack by exploring the critical path using Dynamic Programming (DP) techniques, globally optimizing the attack model to achieve powerful results with fewer bit-flips. Moreover, we thoroughly analyze the detection layer to identify vulnerabilities, enabling our attacks to transfer effectively across different models. Compared to previous methods, our approach is more difficult to detect as it leverages row-hammer via side channels without modifying input data. Moreover, it requires only a subset of validation data to determine updated parameter values, enhancing practicality for potential users. The key contributions of this paper are outlined below:

- We investigate the behavior of the OD models and observe that modifying a few model parameter values can significantly increase the confidence score of several candidate predictions.
- We develop a novel loss function to identify parameters with high gradient values and implement back-propagation by applying a mask to generate updated parameter values and determine the required bit-flips.
- We integrate a global optimization to identify the most susceptible parameters and maximize latency along the critical path with minimal bit-flips.
- We conduct a comprehensive analysis of the detection layer to uncover vulnerabilities, enabling effective attack transferability between different models.
- We perform a comprehensive evaluation on various models using different datasets and demonstrate the effectiveness of the technique by increasing the number of candidates passed to NMS up to 38000, resulting in a latency of 71.6 ms (20.4$\times$) with only 31 bit-flips.

2. Background and Related Work

2.1. Latency-based Attacks

Latency-based attacks aim to increase execution latency, thereby degrading system performance and potentially making models less responsive or unavailable. These attacks are harder to detect and may progressively worsen over time, significantly disrupting user experience and operational efficiency. An early example is Sponge Examples [34], which used an evolutionary search algorithm to find adversarial examples that increase the latency of NLP and image classification models. Subsequent studies expanded on this concept, incorporating adversarial attacks on models such as Multi-exit networks [16], Neural caption generation [6], and Vision Transformers [26]. In [3, 33], adversarial patches were designed to overload the NMS filter in object detection models. Similarly, [20] proposed perturbation attacks on LiDAR-based 3D object detection models. Building on [33], [22] introduced a scheduling algorithm to maximize latency in object-tracking algorithms, significantly impacting the safety of autonomous driving systems. Most studies focus on designing predictable and short-lived adversarial patches. In contrast, our approach uses bit-flip attacks to target internal model parameters, enhancing object detection recall rates by leveraging the model's functionality. By addressing vulnerabilities beyond basic input manipulation, our findings open new avenues for exploring and mitigating deeper model vulnerabilities.

2.2. Bit-Flip Attacks

Bit-flip attacks have emerged as a popular attack method in the machine learning community due to their ability to subtly and efficiently manipulate model parameters, thereby altering model behavior. Unlike traditional attacks such as backdoors, bit-flip attacks often operate during model inference and require minimal training or validation data. For instance, in [15], researchers demonstrated how flipping a single bit can drastically impact model effectiveness. Additionally, bit-flips have been exploited to implant backdoors in various models, leading to the misclassification of inputs containing specific triggers. Studies like [4, 28, 29] have leveraged bit-flip attacks to introduce backdoors in CNNs, while the TrojViT approach [40] successfully inserted backdoors into Vision Transformers using bit-flips. Recent advancements, such as [8], have even integrated bit-flip attacks into model training, allowing adversaries to alter model performance by modifying a single bit. Despite these advances, the potential of bit-flip attacks to disrupt model availability remains underexplored. Our research explores bit-flip attacks on model weights to induce latency in object detection models, aiming to advance our understanding and application of techniques that compromise model availability, which lays the foundation for enhancing model resilience against sophisticated adversarial threats.

2.3. Row-Hammer

DRAMs are susceptible to bit-flips caused by electromagnetic interference between memory cells. Exploited by Row-Hammer attacks [18], it induces bit-flips in vulnerable memory locations (victim rows) by repeatedly accessing adjacent rows (aggressor rows). [30] further revealed that attackers can strategically profile vulnerable memory locations and manipulate access patterns through memory massaging techniques to induce targeted bit-flips. The severity of Row-Hammer depends on various factors including DRAM technology, memory access patterns, and environmental conditions [25]. Row-Hammer vulnerabilities per-

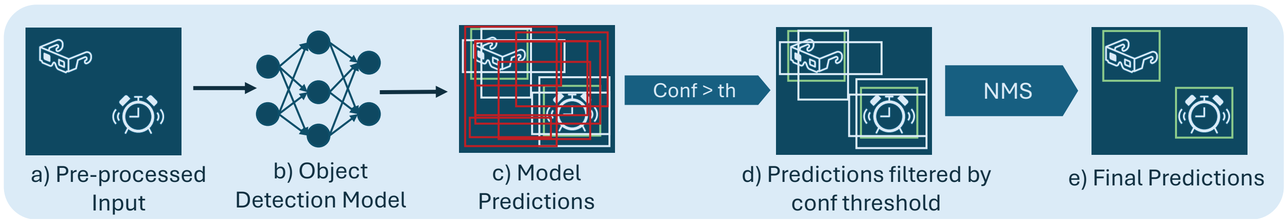


Figure 1. Object detection pipeline.

sist across several state-of-the-art DRAM technologies such as DDR3, DDR4, ECC DRAM, LPDDR2, and LPDDR3 [25]. For example, experiments detailed in [35] show that row-hammer attacks can induce thousands of bit-flips in DDR3 and DDR4 DRAMs within a short period. Recent advancements in the field have accelerated progress significantly. For example, studies indicate that over the past decade, the number of row activations required to induce a bit-flip has decreased by a factor of 10, while the number of bit-flips resulting from the same number of activations has increased by a factor of 500 [24]. These studies underscore the increasing severity of the row-hammer vulnerability, highlighting its growing significance. For further details related to row-hammer deployment, please refer to Appendix A.6.

3. Methodology

This section illustrates the concept of Bit-Flip Induced Latency Attacks in object detection (OD). Fig. 1, presents the pipeline of the object detection model, the transformed image is fed to the model, which extracts features to generate candidate predictions. Each prediction includes bounding box parameters (coordinates, width, and height), an objectness score (likelihood of containing an object), and class probabilities (likelihood of belonging to a specific class). Confidence scores are calculated by multiplying the objectness score by the highest class probability. Predictions with confidence scores above a threshold (th) are retained. Non-maximum Suppression (NMS) then refines these predictions by evaluating overlapping areas using the Intersection-over-Union (IoU) metric, retaining only the highest-confidence candidate per object.

A common strategy in latency-based attacks on OD models involves designing an adversarial patch to elevate confidence scores of candidate predictions above a threshold T_{conf} [3, 22, 33]. The confidence score is the product of the Objectness Score (t_{obj}) and the Maximum Class Probability (c_{obj}). This increases the NMS step's latency, as it must compare each prediction with all others, resulting in an algorithmic complexity of $O(n^2) + O(n)$, where n is the number of candidates [22].

To explore bit-flip attacks on model weights to increase latency, we first examine the training process. Training OD models involves optimizing a loss function with three components: objectness loss (detecting objects within grid cells), IoU regression loss (measuring the accuracy of bounding boxes), and classification loss (assigning object classes). During back-propagation, t_{obj} values are forced to 0 in non-interest regions, ensuring detection only in relevant areas. Since the detection layer is convolutional, altering even one kernel can impact predictions across the image. We exploit this by introducing bit-level perturbations to a small subset of weights, creating phantom objects. Preliminary experiments suggest generating up to 8,000 objects with fewer than 10 parameter changes.

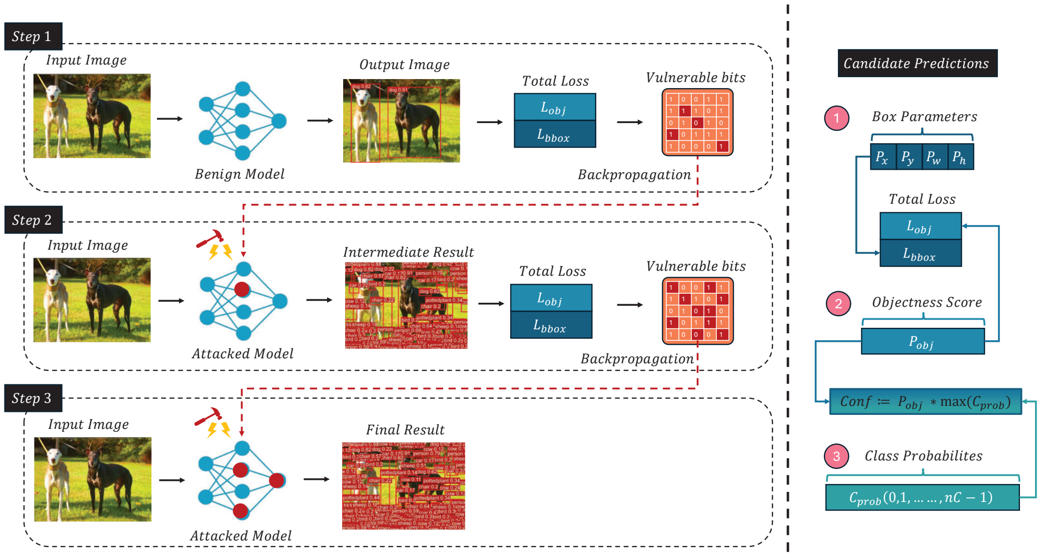
With these insights, we propose a potent bit-flip attack that significantly increases model latency. This attack uses a subset of validation data and can be executed as a side-channel attack during runtime using techniques like row-hammer [18, 21]. Since it does not introduce noise to the input, it operates covertly, evading detection. The following section provides a detailed explanation of the design.

4. Bit-Flip induced latency attack

Building upon our earlier discussion about object detection model training, we recognize that during training, predictions from the forward pass are filtered to isolate image regions of interest, reducing objectness scores for the remaining regions to zero. To make the model detect additional objects, we may raise the objectness scores of these background regions. With this insight, we introduce a novel attack on pre-trained object detection models. Our approach maximizes detection latency by prompting the model to identify numerous phantom objects.

4.1. Attack Overview

Attack Objective: Fig. 2 illustrates an overview of our attack. During the forward pass, candidate predictions, denoted as C, are obtained from the detection layers. Each candidate comprises three components: box parameters (P_x, P_y, P_w, and P_h), Objectness Score (P_{obj}), and class probabilities (C_{prob}). To compute the loss function, the predicted candidates are compared side-by-side with their corresponding targets to delineate regions of interest containing objects. This subset of candidates termed C_{RoI}, is assigned target objectness scores equivalent to the IoU area of the associated bounding box relative to the target bounding box. The remaining candidates, designated as C_m, are



(a) Overview of our Attack model

(b) Candidate prediction details

Figure 2. Attack overview.

considered masked, with their target objectness score set to 0. The objective of our attack is to leverage the masked candidates, C_m, to devise a loss function that increases the number of candidates forwarded to the NMS filter. This manipulation aims to augment inference latency, aligning with our attack objectives. In the following parts of the section, we will delve into the specific details of our strategies, including attack model formulation, detection layer vulnerability analysis, and optimization based on the critical path.

Attack Assumptions: To design an optimized latency attack method, we assume a white-box scenario where the attacker has complete access to the model's architecture and parameters. This approach aligns with numerous studies on adversarial and backdoor attacks [3–6, 29, 33, 40], which also operate under similar assumptions. In practice, the availability of open-source models further supports the feasibility of such attacks [11, 14].

Additionally, our method only requires a subset of the validation data, a realistic assumption supported by prior work [29], which demonstrates that such information can be obtained through side-channel attacks. If the attacker has domain knowledge of the specific application, they could further enhance the attack's effectiveness by generating synthetic data using techniques such as Generative Adversarial Networks (GANs) or Diffusion Models (DDPMs).

In practice, executing the attack requires the victim to run a specific malicious program to perform row-hammer and induce bit-flips. This can be achieved through malware or social engineering. The malicious program can also help the attacker determine the victim model's memory address and layout at runtime, eliminating the need for prior knowledge. For instance, [10] demonstrates how row-hammer attacks can be executed through JavaScript on web browsers, highlighting the practical feasibility of such methods. More details on the attack and its practical feasibility are included in Appendix A.3 and A.4.

4.2. Fundamental Loss-Driven Attacks

In this part, we will systematically introduce our proposed attack steps to provide a thorough comprehension of our attack model. First, to facilitate the enhancement of confidence scores for predictions from background regions, we devised a loss function with two primary components: objectness loss and bounding-box area loss. Our approach aims to increase the model's confidence in detecting objects within these regions.

Objectness Loss: The objectness score (t_{obj}) is trained to reflect the presence of an object within each candidate or grid cell. Specifically, if an object exists within a particular grid cell, the corresponding t_{obj} is set to the IoU area between the candidate bounding box and the target bounding box provided in the labels. Conversely, for candidates where no object is present, t_{obj} is enforced to approach zero. To facilitate the detection of phantom objects, we adjust the t_{obj} values for all candidates without objects to a predefined threshold value, Th_{obj}. This adjustment is achieved through the design of our loss function, outlined below:

$$L_{obj} = Th_{obj} - t_{obj} | C \in C_m \; and \; t_{obj} < Th_{obj} \tag{1}$$

The value of Th_{obj} is carefully selected to ensure that it does not impact the t_{obj} values of candidates with actual predictions.

Bounding box Area Loss: During the application of the NMS filter, only the candidate with the highest confidence score is retained for each object, among candidates with

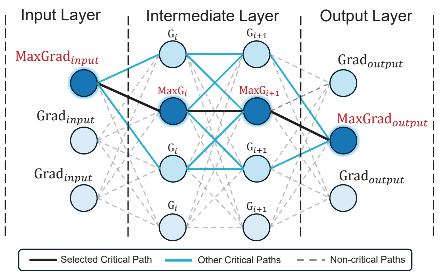


Figure 3. Representation of critical path in neural networks.

overlapping bounding boxes. To maximize the retention of objects during this step, we aim to minimize the overlap between bounding boxes by reducing their areas. This objective is pursued through the following loss function:

$$L_{bbox} = \frac{1}{N_{cm}} \sum_{c \in C_m} \frac{b_w^c * b_h^c}{W * H} \tag{2}$$

Here, N_{cm} represents the number of masked candidates, and b_w^c and b_h^c denote the width and height of the candidate bounding box, and with W and H denoting the width and height of the input image, respectively. Thus, the final loss function can be derived as follows:

$$L_{Total} = \lambda_1 L_{obj} + \lambda_2 L_{bbox} \tag{3}$$

where λ_1 and λ_2 are hyperparameters.

After calculating the model's loss using the function (3) with a batch of validation data, gradients are computed by back-propagating through the detection layer. These gradients are then used to generate a gradient mask, which identifies the top-n parameters requiring adjustment. Through several epochs of optimization using gradient descent, these selected parameters are updated to minimize the loss. The Hamming distance between the original and updated parameter values is calculated to determine the number of necessary bit-flips. For efficiency, our approach focuses on modifying only the top 10 to 13 most significant bits (MSBs) of each parameter. This strategy not only reduces the number of bit-flips required but also enhances latency in certain scenarios.

4.3. Optimization via Critical Paths

To further enhance the effectiveness of our attack and comprehensively assess its impact on the model, we extended the back-propagation of loss beyond the detection layer and calculated the gradients. This analysis revealed non-uniform propagation of loss across parameters of the preceding layers, with certain parameters exhibiting higher gradient values relative to others. These parameters, identified as critical path parameters, are pivotal in shaping the predictions of the OD model for any given input.

Building on this insight, we introduce a technique leveraging critical path analysis to optimize attack strategies. Fig. 3 illustrates an example of a critical path in a network with input, intermediate, and output layers. Parameters with large-magnitude gradients, highlighted in black, form critical paths for specific inputs. As model activations vary with inputs, multiple critical paths may exist. In this study, we analyze critical paths to determine the optimal number of updating parameters to launch the attack efficiently and successfully. By tracking the gradient propagation path of the loss function given in Eq. (3), we identify parameters crucial for increasing the confidence score of the model predictions and determining the optimal parameters for updating. Specifically, we compute the gradient value G_i for each parameter, further decomposed into weights (w) and biases (b), using gradient descent algorithms typically employed in OD models. We compute the maximum gradient in each layer and derive the critical path using the equation:

$$\text{MaxGrad}_{\text{layer}} = \max_i \left(\frac{\partial L}{\partial p_i} \right) = \max_i \left(\frac{\partial L}{\partial w_i}, \frac{\partial L}{\partial b_i} \right) \tag{4}$$

Here, $MaxGrad_{\text{layer}}$ represents the maximum gradient in the current layer, i denotes the parameter index, and $\frac{\partial L}{\partial w_i}$ and $\frac{\partial L}{\partial b_i}$ are the gradients of weights and biases, respectively. Next, we identify critical path parameters by selecting parameters with the highest gradient values in each layer. Thus, we can find critical path parameters in each layer and further compute the critical path of the entire model. Nevertheless, the direct connection of neurons with maximal gradients into critical paths may not always be feasible, as non-MLP architectures may lack such pathways, and even if present, they cannot ensure global optimality. We adopt a dynamic programming (DP) algorithm to obtain critical paths for the attack, gaining insights into the model's structure and determining influential layers for updates. The DP process can be formulated as follows:

$$\text{MaxG}_i = \max(\sum \text{Grad}_{i-1}, \ \text{Grad}_i + \sum \text{Grad}_{i-2}) \tag{5}$$

By computing the critical path, we gain valuable insights into the structure of the model. Specifically, we first identify influential neurons and then determine the most impactful neuron and its corresponding layer based on magnitude, guiding our decisions on where to prioritize updates. Focusing on the most influential neurons maximizes the perturbation's impact on the model's predictions, enhancing the efficiency of our attacks and unveiling previously overlooked vulnerabilities. This enables the attacker to launch a stealthier and more powerful attack with fewer bit-flips, making it more practical in real-time scenarios. In summary, analyzing critical paths is essential for refining attack strategies and uncovering vulnerabilities in unknown models, particularly in OD networks.

Table 1. Quantitative comparison of the attack performance in NMS and Latency

Dataset	Model	Baseline		Attacked model				
		NMS/image	Avg. Latency(ms)	Bit-flips	NMS/image		Avg. Latency (ms)	
COCO	YOLOv3	125	3.5	31	38457	(307.65×)	**71.6**	(20.4 ×)
	YOLOv4	81	1.6	26	39881	(492.35×)	**6.40**	(4.00 ×)
	YOLOv5	551	1.5	31	29865	(54.201×)	**20.4**	(13.6 ×)
	YOLOv7	131	2.1	27	10728	(81.893×)	**69.0**	(32.8 ×)
PascalVOC	YOLOv3	128	1.8	6	33200	(259.37×)	**12.6**	(7.00 ×)
	YOLOv4	183	1.6	34	31568	(172.50×)	**7.80**	(4.90 ×)
	YOLOv5	262	1.5	6	33334	(127.23×)	**10.0**	(6.70 ×)
	YOLOv7	17	1.6	19	20809	(1224.1×)	**11.8**	(7.40 ×)
BDD100K	YOLOv3	127	0.7	18	31640	(249.13×)	**17.7**	(25.3 ×)
	YOLOv4	65	0.7	18	42267	(650.26×)	**6.10**	(8.70 ×)
	YOLOv5	369	0.7	6	33199	(89.970×)	**12.3**	(17.6 ×)
	YOLOv7	435	0.6	31	32256	(74.151×)	**15.4**	(25.6 ×)

4.4. Attack Transfer via Detection Layer Analysis

Our attack model highlights the susceptibility of OD models to latency induced by bit-flips, underscoring the need for robust defenses. Deploying such attacks in real-world scenarios poses challenges, including the extensive time required for training over multiple epochs to update parameters. To address these, we conduct a vulnerability analysis of the detection layer, using back-propagation through the loss function from equation 3. This analysis identifies exploitable weaknesses, enabling the transfer of the attack to similar models in a grey-box scenario, without needing detailed model parameters or extensive optimization.

We begin by examining the critical parameters within the detection layer of OD models and their direct impact on model predictions, particularly their role in object recognition and localization. By analyzing the model outputs and the NMS step, we pinpoint parameters influencing the objectness score computation. In instances where objectness scores are less critical, our attack strategy can pivot towards influencing other aspects, such as class probabilities (e.g., increasing the probability $P(c)$ of specific classes).

Our analysis of these critical parameters reveals identifiable patterns, notably involving sign reversals in parameters responsible for computing objectness scores. This adjustment is crucial as it alters how the model interprets the absence or presence of objects, affecting prediction confidence scores. Moreover, manipulating parameter magnitudes, particularly toggling the most significant bits (MSBs), can significantly impact model predictions by potentially increasing the number of generated predictions. Based on these insights, our attack strategy involves selecting parameters following the pattern in the detection layer and introducing bit-flips strategically to induce sufficient latency. This approach allows us to transfer our attack methodology to models with NMS and similar architecture, minimizing extensive optimization overhead.

5. Evaluation

In this section, we evaluate the efficacy and performance of our Bit-flip Latency Attacks across several leading object detection networks. We start with a brief overview of the implementation in Section 5.1, followed by an in-depth results analysis in Section 5.2. Finally, we present various ablation experiments in Section 5.3 to assess the impact of our proposed design. For detailed experiment settings and additional evaluations, please refer to Appendix A.1 and A.2.

5.1. Implementation

We implemented our attack method using PyTorch 1.9.1 [27] and evaluated it on various YOLO models [1,17,31,32, 36], known for their speed and real-time applicability. The models were trained on the MS-COCO 2017 [19] dataset and fine-tuned on BDD100k [39] and Pascal VOC [13]. We assessed induced bit-flips, NMS candidates, and detection latency, comparing our method to state-of-the-art latency attacks like PhantomSponges [33] and Overload [3].

5.2. Result Analysis

5.2.1 Optimization of Individual Models

Attack performance: Table. 1 summarizes our attack's performance across different object detection models and datasets. We used a consistent confidence threshold of 0.25 for NMS filtering in all experiments, adjusting the IoU threshold to 0.65 for MS-COCO and Pascal VOC datasets, and to 0.45 for BDD100k. For accurate latency evaluation, experiments were conducted on the test datasets containing between $2,000$ to $5,000$ images. We measured the average latency increase in both baseline and attacked models. Typically, the baseline models exhibit an average latency of 0.6-3.5 ms, with only a few hundred predictions reaching NMS, while our attack significantly amplifies this count with a minimal number of bit-flips. In the best-case sce-

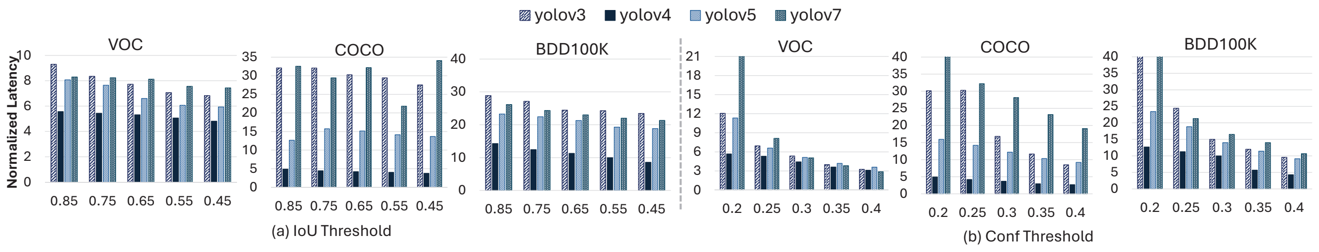


Figure 4. Evaluating NMS filter thresholds on the performance (a) IoU Threshold vs Latency, (b) Confidence Threshold vs Latency.

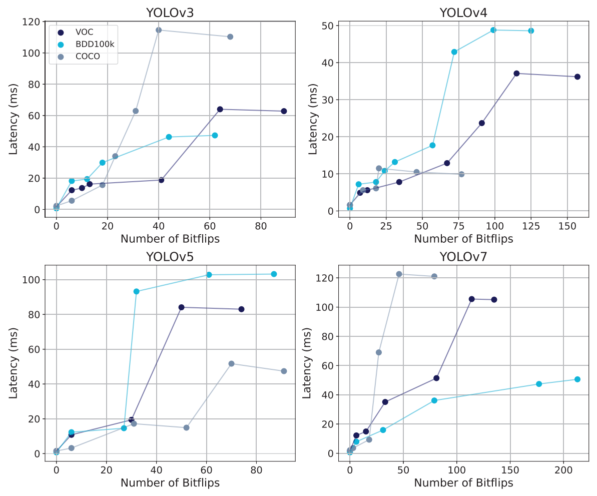


Figure 5. Latency with increasing bit-flip numbers.

nario, our attack increases average latency up to 71.6 ms (20.4×) using only 31 bit-flips, and in the worst case, it still achieves a latency of 6.4 ms (4.0×) with 26 bit-flips. These results demonstrate that our bit-flip attack achieves a consistent increase in latency across various conditions.

Latency vs. number of bit-flips: The latency induced by our attack increases with the number of bit-flips applied to the model. Fig. 5 shows the relationship between latency and the number of bit-flips across different models and datasets. On average, latency rises by 30 to 60 ms with fewer than 100 bit-flips, and in some instances, up to 100 ms. Even in the worst-case scenarios, latency was increased by up to 11 ms. As latency increases, model recall deteriorates significantly, potentially causing major disruptions.

Confidence and IoU thresholds: The number of objects retained after the NMS step is influenced by confidence and IoU thresholds. We investigated their impact on our attack strategy. Results in Fig. 4 show the effect of IoU thresholds from 0.45 to 0.85 and confidence thresholds from 0.2 to 0.4. The results demonstrate the robustness of our attack strategy across various IoU threshold values. While increasing confidence thresholds showed a slight impact on latency, adjusting thresholds alone is insufficient to fully mitigate the attack. Conversely, excessively high thresholds could compromise the model's performance by potentially missing genuine objects.

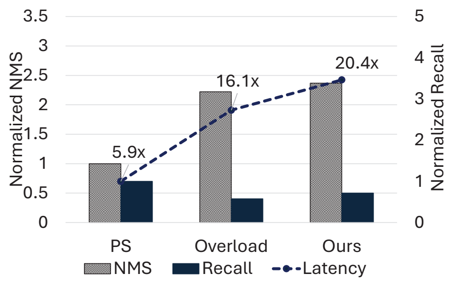


Figure 6. Performance comparison against SOTA latency attacks.

Comparison with SOTA latency attacks: Fig. 6 compares our attack's performance against state-of-the-art latency attacks [3, 33]. The results show that PS introduced an adversarial patch resulting in a latency increase of up to 6×, while Overload achieved a 16.1× latency increase at the expense of recall (only 57% compared to PS). In contrast, our proposed bit-flip attack achieves superior performance, reaching up to 20.4× latency increase, surpassing both previous works while maintaining up to 71.5% recall compared to PS.

5.2.2 Analysis of Attack Transferability

In this section, we evaluated the transferability of our attack across different variants in the YOLO family, as they all incorporate the NMS process. As the NMS is located in the detection layer, we perform the detection layer analysis as discussed in Section 4.4.

Specifically, we first conduct a vulnerability analysis on the detection layer of the YOLOv3 model using the BDD100K dataset. Given that models with similar architectures and identical NMS processes are likely to exhibit comparable vulnerability patterns, we extend our analysis to YOLO variants. After identifying key vulnerability trends in YOLOv3, we apply bit-flips to the same locations in these variants, randomly flipping between 50 and 70 bits within their detection layers, following the identified pattern. This evaluation encompassed various YOLO models optimized for COCO, VOC, and BDD100K datasets to ensure generalizability. The experimental results are shown in Fig.7, where Fig.7(a) illustrates the number of bit-flips required for each evaluation, and Fig. 7(b) displays the induced latency. The results indicate a latency increase of 2× to 13×,

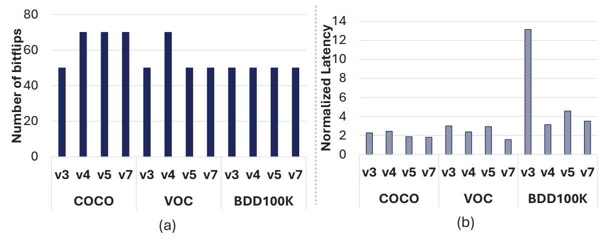


Figure 7. Evaluation of attack transferability

even without optimization. Additionally, we observe that the number of bit-flips needed to induce latency increases with the number of classes in the dataset. This suggests that the transferability of our attacks may improve across models when applied to more uniform application scenarios, revealing the practical feasibility of achieving gray-box attacks through transferability.

5.3. Ablation Study

We validate our design through a structured evaluation process, establishing a baseline by evaluating the vanilla model's performance without attacks, assessing fundamental loss-driven attacks, and evaluating the CP attack. Fig. 8. (a) shows bit-flips required for different attack methods: Vanilla model (attack-free), Loss-driven (LD) attack optimized on the detection layer, and Critical Path-based (CP) attack. Our CP attack reduces maximum bit-flips from 74 to 34, with a minimum of 6. Fig. 8. (b) displays induced latency by each attack compared to the vanilla model. The LD attack induces up to 12 ms latency, effective but less so with complex datasets. The CP attack induces latency from 6.1 ms to 71.6 ms, significantly improving performance with fewer bit-flips. This evaluation highlights the efficiency and gains from optimized attack strategies, crucial for robust defense in object detection systems.

6. Limitation and Future Work

While we have demonstrated successful latency increases with minimal bit-flips, the practical implementation of our method exceeds the scope of this paper. Future research efforts could focus on refining these attacks for real-world applications, including designing specific applications to trigger row-hammer attacks. While effective, our method's reliance on prior knowledge of the loss function may restrict its applicability in certain scenarios. Future research should focus on training models capable of performing attacks without this prerequisite, thereby enhancing method accessibility. Additionally, exploring the transferability of attacks to similar models is essential for assessing broader impacts. Analyzing transferability across different architectures and configurations can offer insights into their generalizability and robustness.

Our method extends beyond YOLO models and can be applied to other NMS-based object detection models.

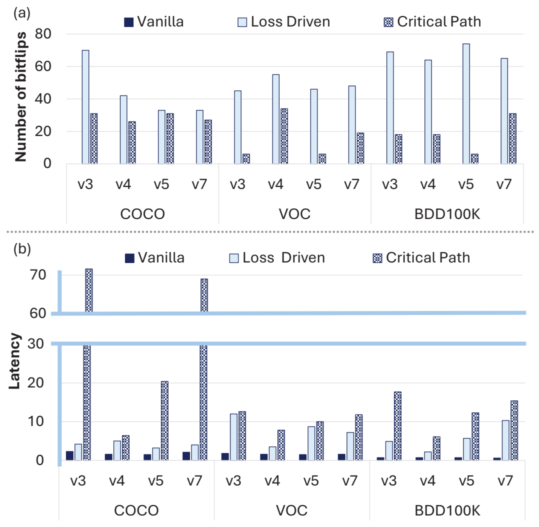


Figure 8. (a) Bit-flips required and (b) Latency analysis for the vanilla model, Loss Driven attack, and Critical Path-based attack.

However, due to significant architectural differences, direct transferability may not be feasible, requiring manual optimizations. For details on extending the attack to other networks, please refer to Appendix A.5.

7. Conclusion

Object detection is vital in deep learning and computer vision, yet its susceptibility to attacks poses serious safety and security risks. While previous research has mainly addressed integrity-based attacks, the emergence of latency-based attacks has introduced new concerns about object detection model reliability. We proposed a novel bit-flip-based latency attack method for object detection models. Our approach, executed through side channels like memory manipulation, requires no modifications to input data, enhancing its practicality for potential attackers. By inducing bit-flips in model parameters, we significantly increased the number of candidates passed to the NMS filter, resulting in notable latency increases with minimal bit-flips. Additionally, our global optimization identified critical parameters to maximize latency with minimal bit-flips, and our analysis of the detection layer uncovered vulnerabilities, enhancing attack transferability between models. These findings underscore the imperative of addressing latency vulnerabilities in deep neural networks and pave the way for exploring defense mechanisms against such attacks.

8. Acknowledgement

This research is partially supported by NSF grants CCF-2130688, and CNS-2107057.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 6

[2] Shih-Han Chan, Yinpeng Dong, Jun Zhu, Xiaolu Zhang, and Jun Zhou. Baddet: Backdoor attacks on object detection. In *European Conference on Computer Vision*, pages 396–412. Springer, 2022. 1

[3] Erh-Chung Chen, Pin-Yu Chen, I Chung, Che-rung Lee, et al. Overload: Latency attacks on object detection for edge devices. *arXiv preprint arXiv:2304.05370*, 2023. 2, 3, 4, 6, 7

[4] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7718–7727, 2021. 2, 4

[5] Simin Chen, Hanlin Chen, Mirazul Haque, Cong Liu, and Wei Yang. The dark side of dynamic routing neural networks: Towards efficiency backdoor injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24585–24594, 2023. 4

[6] Simin Chen, Zihe Song, Mirazul Haque, Cong Liu, and Wei Yang. Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15365–15374, 2022. 2, 4

[7] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, pages 52–68. Springer, 2019. 1

[8] Jianshuo Dong, Han Qiu, Yiming Li, Tianwei Zhang, Yuanjie Li, Zeqi Lai, Chao Zhang, and Shu-Tao Xia. One-bit flip is all you need: When bit-flip attack meets model training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4688–4698, 2023. 2

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[10] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer. js: A remote software-induced fault attack in javascript. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13*, pages 300–321. Springer, 2016. 4

[11] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 4

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[13] Derek Hoiem, Santosh K Divvala, and James H Hays. Pascal voc 2008 challenge. *World Literature Today*, 24(1):1–4, 2009. 6

[14] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *Advances in Neural Information Processing Systems*, 35:8068–8080, 2022. 4

[15] Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraș. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 497–514, 2019. 2

[16] Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*, 2020. 2

[17] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, Yonghye Kwon, Kalen Michael, Jiacong Fang, Colin Wong, Zeng Yifu, Diego Montes, et al. ultralytics/yolov5: v6. 2-yolov5 classification models, apple m1, reproducibility, clearml and deci. ai integrations. *Zenodo*, 2022. 6

[18] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014. 1, 2, 3

[19] 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

[20] Han Liu, Yuhao Wu, Zhiyuan Yu, Yevgeniy Vorobeychik, and Ning Zhang. Slowlidar: Increasing the latency of lidar-based detection using adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5146–5155, 2023. 2

[21] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. in 2017 ieee. In *ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138. 3

[22] 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*, volume 38, pages 4062–4070, 2024. 1, 2, 3

[23] Hua Ma, Yinshan Li, Yansong Gao, Alsharif Abuadbba, Zhi Zhang, Anmin Fu, Hyoungshick Kim, Said F Al-Sarawi, Nepal Surya, and Derek Abbott. Dangerous cloaking: Natural trigger based backdoor attacks on object detectors in the physical world. *arXiv preprint arXiv:2201.08619*, 2022. 1

[24] Onur Mutlu et al. Fundamentally understanding and solving rowhammer. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 461–468, 2023. 3

[25] Onur Mutlu and Jeremie S Kim. Rowhammer: A retrospective. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(8):1555–1571, 2019. 2, 3

[26] KL Navaneet, Soroush Abbasi Koohpayegani, Essam Sleiman, and Hamed Pirsiavash. Slowformer: Universal adversarial patch for attack on compute and energy efficiency of inference efficient vision transformers. *arXiv preprint arXiv:2310.02544*, 2023. 2

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[28] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1211–1220, 2019. 2

[29] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13198–13207, 2020. 2, 4

[30] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1–18, 2016. 2

[31] 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, 6

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

[33] 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. 2, 3, 4, 6, 7

[34] 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. 2

[35] Youssef Tobah, Andrew Kwong, Ingab Kang, Daniel Genkin, and Kang G Shin. Spechammer: Combining spectre and rowhammer for new speculative attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 681–698. IEEE, 2022. 3

[36] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023. 6

[37] Wenbin Wang and Haixia Long. A study of backdoor attacks against the object detection model yolov5. In *2023 2nd International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM)*, pages 278–284. IEEE, 2023. 1

[38] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 1–17. Springer, 2020. 1

[39] 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

[40] Mengxin Zheng, Qian Lou, and Lei Jiang. Trojvit: Trojan insertion in vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4025–4034, 2023. 2, 4

[41] 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. 1