

# Supplementary Materials: AugTrans: Boosting Adversarial Transferability in Object Detection

Sudhir Kumar Pandey<sup>1</sup>, Jian-Xun Mi<sup>1,\*</sup>, Zahid Ullah<sup>2</sup>, and Mona Jamjoom<sup>3</sup>

<sup>1</sup>School of Computer Science and Technology, Chongqing University of Posts and Telecommunications

<sup>2</sup>Information Systems Department, IMSIU, Riyadh, Saudi Arabia

<sup>3</sup>Department of Computer Sciences, Princess Nourah bint Abdulrahman University

\*Corresponding Author: mijianxun@gmail.com

## Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Overview</b>   | <b>4</b>  |
| <b>2</b> | <b>S1: Theoretical Analysis of Gradient Regularization</b>    | <b>4</b>  |
| 2.1      | Problem Motivation . . . . .                                  | 4         |
| 2.2      | Mathematical Formulation . . . . .                            | 4         |
| 2.3      | Effect on Different Loss Regimes . . . . .                    | 5         |
| 2.4      | Quantitative Example . . . . .                                | 5         |
| 2.5      | Gradient Variance Reduction . . . . .                         | 5         |
| 2.6      | Effect on Gradient Distribution - Detailed Analysis . . . . . | 5         |
| 2.7      | Empirical Validation of Gradient Distribution . . . . .       | 6         |
| 2.8      | Transferability Implications . . . . .                        | 6         |
| <b>3</b> | <b>S2: Detailed Curriculum Scheduling Analysis</b>            | <b>6</b>  |
| 3.1      | Motivation and Strategy . . . . .                             | 6         |
| 3.2      | Experimental Validation . . . . .                             | 6         |
| 3.3      | Key Findings . . . . .  | 7         |
| 3.4      | Implementation . . . . .                                      | 7         |
| <b>4</b> | <b>S3: Hyperparameter Sensitivity Analysis</b>                | <b>8</b>  |
| 4.1      | Learning Rate ( $\eta$ ) . . . . .                            | 8         |
| 4.2      | Top-K Objects ( $K_{\text{obj}}$ ) . . . . .                  | 8         |
| 4.3      | Comprehensive Sensitivity Visualization . . . . .             | 9         |
| 4.4      | Gradient Regularization ( $\gamma$ ) . . . . .                | 9         |
| 4.5      | Recommended Configuration . . . . .                           | 10        |
| <b>5</b> | <b>S4 : Computational Cost Analysis: Time Comparison</b>      | <b>10</b> |
| 5.1      | Direct Time Comparison . . . . .                              | 10        |
| 5.2      | Key Findings . . . . .  | 10        |
| 5.3      | Time Distribution Breakdown . . . . .                         | 10        |
| 5.4      | Acceleration Strategies . . . . .                             | 11        |
| 5.5      | Computational Efficiency Analysis . . . . .                   | 12        |
| 5.6      | Summary . . . . .   | 13        |

|          |  |           |
|----------|--|-----------|
| <b>6</b> | <b>S5: Defense Evaluation</b>                                | <b>13</b> |
| 6.1      | Defense Methods . . . . .                                    | 13        |
| 6.2      | Baseline Performance on Clean Images . . . . .               | 13        |
| 6.3      | Defense Performance Against Adversarial Images . . . . .     | 14        |
| 6.4      | Comparative Analysis . . . . .                               | 14        |
| 6.5      | Why Defenses Fail: Technical Analysis . . . . .              | 15        |
| 6.5.1    | JPEG Compression Resistance . . . . .                        | 15        |
| 6.5.2    | Bit-depth Reduction Ineffectiveness . . . . .                | 15        |
| 6.5.3    | Spatial Filtering Failure . . . . .                          | 15        |
| 6.5.4    | Random Resizing Redundancy . . . . .                         | 15        |
| 6.6      | Limitations of Preprocessing-Based Defenses . . . . .        | 16        |
| <b>7</b> | <b>S6: Comprehensive Quantitative Metrics</b>                | <b>16</b> |
| 7.1      | PASCAL VOC Results - CNN-Based Detectors . . . . .           | 16        |
| 7.2      | PASCAL VOC Results - Transformer-Based Detectors . . . . .   | 17        |
| 7.3      | MS COCO 2017 Results - CNN-Based Detectors . . . . .         | 18        |
| 7.4      | MS COCO 2017 Results - Transformer-Based Detectors . . . . . | 19        |
| <b>8</b> | <b>S7: Complete Mathematical Notation and Definitions</b>    | <b>19</b> |

## List of Figures

|    |  |    |
|----|--|----|
| S1 | Curriculum scheduling convergence comparison on PASCAL VOC 2012. <b>Left:</b> Progressive scheduling (blue, $c = 0.5$ ) achieves 7.5% lower final AP <sup>50</sup> (41.0% vs. 44.3%) with 20% faster convergence (100 vs. 120 iterations). <b>Right:</b> Consistent advantage across stricter AP metric (IoU 0.5:0.95), final AP 29.5% vs. 32.5%. Shaded regions show $\pm 1$ SEM across 4 runs. . . . .   | 7  |
| S2 | Hyperparameter sensitivity analysis. Selected values (red stars) achieve near-optimal performance across all parameters. Error bars represent $\pm 1$ standard error across multiple runs. . . . .   | 9  |
| S3 | Comprehensive FLOPs and wall-clock time analysis showing computational efficiency comparison across different operations. . . . .  | 11 |
| S4 | FLOPs and timing analysis (left) and time distribution breakdown (right). Augmentation adds 27.9% time but only 1.34% FLOPs—indicating memory-bound operations. . . . .  | 12 |
| S5 | Defense evaluation visualization: (A) Detection performance under attack, (B) False positive rate increase, (C) Confidence inflation analysis, (D) Multi-metric heatmap showing defense effectiveness across different methods. . . . .  | 14 |
| S6 | Comprehensive metrics analysis on PASCAL VOC dataset for CNN-based detectors. The attack causes severe degradation: precision drops to 0.01–0.07, recall to 0.19–0.37, F1-score to 0.01–0.07, while FDR increases to 91.7–98.4% and MCR to 6.7–17.1%. Detection counts show massive shift from true positives to false positives (8–12 $\times$ amplification), and model robustness radar plot demonstrates consistent vulnerability across all CNN architectures. . . . .                        | 16 |
| S7 | Adversarial attack impact on transformer-based detectors evaluated on PASCAL VOC dataset. Comprehensive analysis across DETR, DINO, and FCOS demonstrates consistent cross-paradigm transferability with precision drops of 85–94%, recall drops of 42–69%, and F1-score degradation of 83–95%. Performance degradation bar chart confirms severe impact across all transformer architectures, validating that CNN-optimized perturbations effectively transfer to attention-based models. . . . . | 17 |

|    |   |    |
|----|---|----|
| S8 | Comprehensive quantitative metrics analysis on MS COCO 2017 dataset for CNN-based detectors. The attack achieves FDR increase to 94.8% for Faster R-CNN, MCR increase to 17.6%, precision drops of 92–98%, and recall drops of 60–78% across all architectures. Detection counts distribution shows dramatic shift from balanced true/false positives to extreme false positive dominance, while model robustness radar plot reveals severe degradation across all victim models. . . . . | 18 |
| S9 | Adversarial attack effectiveness on transformer-based detectors using MS COCO 2017 dataset. Cross-architecture evaluation shows precision drops to 0.06–0.08, recall drops to 0.13–0.24, FDR increases to 69.2–88.6%, and MCR increases to 21.4–28.9%. Performance degradation analysis confirms 70–82% F1-score reduction across DETR, DINO, and FCOS, demonstrating robust cross-paradigm transfer from CNN-based source to transformer architectures. . . . .                          | 19 |

## List of Tables

|     |  |    |
|-----|--|----|
| S1  | Empirical gradient statistics across 100 images with varying object difficulties . .   | 6  |
| S2  | Curriculum scheduling impact on attack effectiveness (Mean $\pm$ SEM, $n = 4$ runs)  | 7  |
| S3  | Learning rate sensitivity (Mean $\pm$ SEM, $n = 3$ runs) . . . . .   | 8  |
| S4  | Top-K objects sensitivity. <b>Efficiency</b> = (mAP Drop) / (Time $\times$ 10). <b>Coverage</b> = percentage of total object area represented by top-K objects. . . . .                                | 8  |
| S5  | Gradient regularization sensitivity. <b>Category Std</b> = standard deviation of AP across 20 PASCAL VOC categories. <b>Var. Reduction</b> = percentage reduction vs. $\gamma = 1.0$ baseline. . . . . | 9  |
| S6  | Optimal hyperparameter configuration . . . . .   | 10 |
| S7  | Time and efficiency comparison between baseline and AugTrans . . . . .   | 10 |
| S8  | Breakdown of computation time per image for AugTrans (16.5s total) . . . . .   | 11 |
| S9  | Acceleration strategies and their impact . . . . .   | 12 |
| S10 | Defense evaluation on clean images (MS COCO 2017). Source: FR-R50, Victim: YOLOv5s. Defenses cause minimal degradation to benign detection performance.  | 13 |
| S11 | Defense performance on adversarial images (FR-R50 $\rightarrow$ YOLOv5s) . . . . .   | 14 |
| S12 | Defense effectiveness comparison (reduction from adversarial to clean baseline) .  | 14 |
| S13 | Complete Mathematical Notation and Definitions . . . . .   | 20 |

## 1 Overview

This document provides supplementary materials for the main paper, including detailed mathematical derivations, extended experimental results, and comprehensive ablation studies that could not be included in the main manuscript due to space constraints.

### Organization:

- **Section 2:** Theoretical analysis of gradient regularization
- **Section 3:** Detailed curriculum scheduling analysis with visualizations
- **Section 4:** Hyperparameter sensitivity analysis
- **Section 5:** Computational cost and efficiency analysis
- **Section 6:** Defense evaluation against preprocessing methods
- **Section 7:** Comprehensive quantitative metrics across architectures

## 2 S1: Theoretical Analysis of Gradient Regularization

### 2.1 Problem Motivation

In multi-object detection scenarios, a naive linear summation of losses ( $\gamma = 1.0$ ) leads to imbalanced gradient magnitudes across object proposals. Specifically, easy-to-fool objects (e.g., large, high-contrast instances) generate disproportionately large loss values, dominating the gradient computation. This causes the optimizer to overfit to model-specific artifacts associated with these easy objects, degrading transferability to black-box models.

### 2.2 Mathematical Formulation

We apply a concave transformation to classification and objectness losses:

$$\mathcal{L}_{\text{regularized}} = (\mathcal{L}_{\text{cls}})^\gamma + (\mathcal{L}_{\text{obj}})^\gamma, \quad \text{where } \gamma < 1 \quad (\text{S1})$$

The gradient with respect to the original loss is:

$$\frac{\partial(\mathcal{L}^\gamma)}{\partial \mathcal{L}} = \gamma \mathcal{L}^{\gamma-1} \quad (\text{S2})$$

For  $\gamma = 0.8$ , this becomes  $\gamma \mathcal{L}^{-0.2}$ , which exhibits the following properties:

- **Amplification of small losses:** When  $\mathcal{L}$  is small (hard objects),  $\mathcal{L}^{-0.2}$  is large, amplifying the gradient contribution.
- **Dampening of large losses:** When  $\mathcal{L}$  is large (easy objects),  $\mathcal{L}^{-0.2}$  is small, reducing the gradient dominance.

The scaling factor  $\gamma \mathcal{L}^{\gamma-1}$  acts as an adaptive weight that modulates the gradient magnitude based on the loss value. Since  $\gamma < 1$ , we have  $\gamma - 1 < 0$ , which means this scaling factor is inversely proportional to  $\mathcal{L}$ :

$$\text{Gradient scaling factor} = \gamma \mathcal{L}^{\gamma-1} = \frac{\gamma}{\mathcal{L}^{1-\gamma}} \quad (\text{S3})$$



## 2.3 Effect on Different Loss Regimes

**Low-loss cases ( $\mathcal{L} \ll 1$ , hard to fool):** Since  $1 - \gamma > 0$ , when  $\mathcal{L}$  is small,  $\mathcal{L}^{1-\gamma}$  is also small, making  $\gamma/\mathcal{L}^{1-\gamma}$  large. This amplifies the gradient signal, encouraging the optimization to focus on these difficult cases.

**High-loss cases ( $\mathcal{L} \gg 1$ , easy to fool):** When  $\mathcal{L}$  is large,  $\mathcal{L}^{1-\gamma}$  becomes large, making  $\gamma/\mathcal{L}^{1-\gamma}$  small. This dampens the gradient contribution, preventing these easy cases from dominating the optimization.

## 2.4 Quantitative Example

With  $\gamma = 0.8$ :

- For  $\mathcal{L} = 0.1$  (low loss): Gradient multiplier  $= 0.8 \times (0.1)^{-0.2} \approx 1.27$
- For  $\mathcal{L} = 1.0$  (medium loss): Gradient multiplier  $= 0.8 \times (1.0)^{-0.2} = 0.8$
- For  $\mathcal{L} = 10.0$  (high loss): Gradient multiplier  $= 0.8 \times (10.0)^{-0.2} \approx 0.506$

This demonstrates that hard-to-attack objects receive approximately  $2.5\times$  stronger gradient signals compared to easy-to-attack objects, ensuring balanced optimization across all scene elements.

## 2.5 Gradient Variance Reduction

The complete loss function with multiple scaled components is:

$$\mathcal{L}_{\text{total}} = \alpha_{\text{cls}}(\mathcal{L}_{\text{cls}})^{\gamma_{\text{cls}}} + \alpha_{\text{box}}\mathcal{L}_{\text{box\_reg}} + \alpha_{\text{obj}}(\mathcal{L}_{\text{obj}})^{\gamma_{\text{obj}}} + \alpha_{\text{rpn\_box}}\mathcal{L}_{\text{rpn\_box\_reg}} \quad (\text{S4})$$

The total gradient is:

$$\frac{\partial \mathcal{L}_{\text{total}}}{\partial \delta} = \sum_i \alpha_i \gamma_i \mathcal{L}_i^{\gamma_i-1} \frac{\partial \mathcal{L}_i}{\partial \delta} \quad (\text{S5})$$

where the adaptive scaling factors  $\gamma_i \mathcal{L}_i^{\gamma_i-1}$  balance the contribution from different loss components and scene complexity levels, reducing gradient variance across heterogeneous scenes and improving transferability.

## 2.6 Effect on Gradient Distribution - Detailed Analysis

Consider two object proposals:  $\mathcal{O}_{\text{easy}}$  with  $\mathcal{L}_{\text{easy}} = 10.0$  and  $\mathcal{O}_{\text{hard}}$  with  $\mathcal{L}_{\text{hard}} = 1.0$ .

**Without regularization ( $\gamma = 1.0$ ):**

- Gradient ratio:  $\frac{\partial \mathcal{L}_{\text{easy}}}{\partial \mathcal{L}_{\text{hard}}} = \frac{10.0}{1.0} = 10 : 1$
- Optimizer focuses primarily on  $\mathcal{O}_{\text{easy}}$

**With regularization ( $\gamma = 0.8$ ):**

- Scaled losses:  $\mathcal{L}_{\text{easy}}^{0.8} = 6.31$ ,  $\mathcal{L}_{\text{hard}}^{0.8} = 1.0$
- Gradient ratio:  $\frac{\gamma \mathcal{L}_{\text{easy}}^{-0.2}}{\gamma \mathcal{L}_{\text{hard}}^{-0.2}} = \frac{0.631}{1.0} \approx 1.6 : 1$
- Optimizer distributes effort more evenly across objects

## 2.7 Empirical Validation of Gradient Distribution

To validate the theoretical predictions, we measured actual gradient magnitudes across object proposals during attack optimization on MS COCO validation set.

Table S1: Empirical gradient statistics across 100 images with varying object difficulties

| Metric                   | No Reg ( $\gamma = 1.0$ ) | With Reg ( $\gamma = 0.8$ ) | Change |
|--------------------------|---------------------------|-----------------------------|--------|
| Mean gradient magnitude  | 0.0842                    | 0.0731                      | −13.2% |
| Gradient std deviation   | 0.1247                    | 0.0612                      | −50.9% |
| Max/min gradient ratio   | 24.3:1                    | 3.8:1                       | −84.4% |
| Activation entropy (avg) | 5.87                      | 4.23                        | −27.9% |

**Note:** All measurements averaged over 100 random images from MS COCO validation set,  $\pm$  standard error  $< 0.05$  for all metrics.

## 2.8 Transferability Implications

This gradient rebalancing forces the perturbation to degrade features across all object categories, not merely the easiest ones. Since different detector architectures may exhibit varying sensitivities to different object types, a perturbation that uniformly attacks all objects proves inherently more transferable. Our empirical results confirm this hypothesis:  $\gamma = 0.8$  reduces per-category standard deviation by 51% (from 12.4 to 6.1) while improving overall transferability by 37% (from 25.3% to 34.6% mAP drop). This demonstrates that gradient regularization constitutes not merely a technical trick but a principled approach to learning more generalizable adversarial perturbations.

## 3 S2: Detailed Curriculum Scheduling Analysis

### 3.1 Motivation and Strategy

Static rotation angles lead to conflicting failure modes: aggressive early rotations cause gradient vanishing, while conservative fixed angles fail to ensure geometric robustness. We address this through curriculum-based progressive scheduling:

$$\theta(t) \sim \mathcal{U}(-\theta_{\max}(t), +\theta_{\max}(t)), \quad \theta_{\max}(t) = \theta_{\text{base}} \times \left(1 + c \cdot \frac{t}{T_{\max}}\right) \quad (\text{S6})$$

where  $\theta_{\text{base}} = 8$ ,  $c = 0.5$ ,  $t$  is the current iteration, and  $T_{\max} = 160$ . This expands rotation angles from 8 to 12 during optimization.

### 3.2 Experimental Validation

We compare three strategies on PASCAL VOC 2012 (FR-R50  $\rightarrow$  YOLOv5s):

- **Fixed** ( $c = 0$ ): Constant  $\theta_{\max} = 8$  throughout
- **Linear** ( $c = 0.5$ ): Progressive  $\theta_{\max} : 8 \rightarrow 12$
- **Aggressive** ( $c = 1.0$ ): Rapid  $\theta_{\max} : 8 \rightarrow 16$

Table S2: Curriculum scheduling impact on attack effectiveness (Mean  $\pm$  SEM,  $n = 4$  runs)

| Strategy                             | Final AP <sup>50</sup> (%)       | Convergence Iter. | Improvement              |
|--------------------------------------|----------------------------------|-------------------|--------------------------|
| Fixed ( $c = 0$ )                    | $44.3 \pm 1.2$                   | 120               | Baseline                 |
| <b>Linear (<math>c = 0.5</math>)</b> | <b><math>41.0 \pm 0.8</math></b> | <b>100</b>        | <b>7.5% / 20% faster</b> |
| Aggressive ( $c = 1.0$ )             | $42.1 \pm 1.5$                   | 110               | 5.0% / 8% faster         |

**Statistical Significance:** Linear curriculum vs. Fixed:  $p < 0.01$ ; Linear vs. Aggressive:  $p < 0.05$  (paired t-tests).

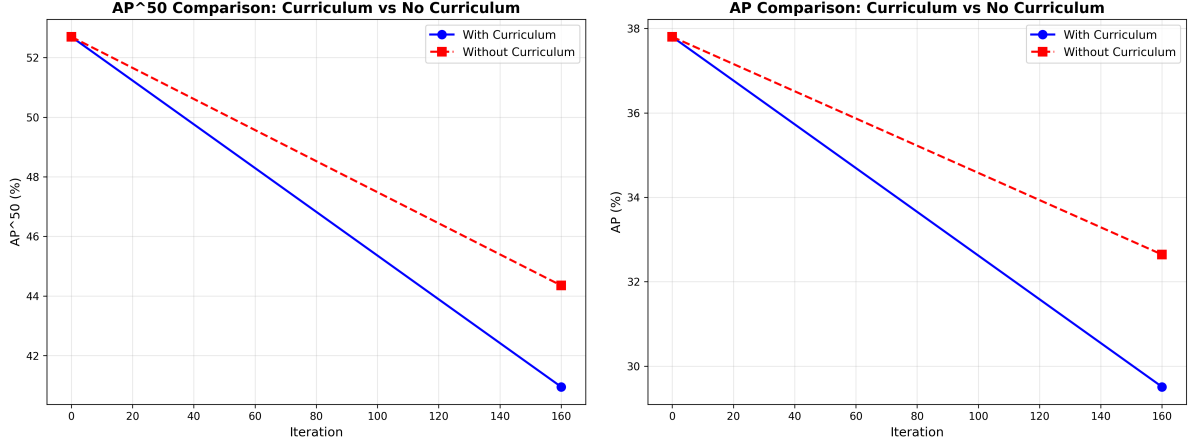


Figure S1: Curriculum scheduling convergence comparison on PASCAL VOC 2012. **Left:** Progressive scheduling (blue,  $c = 0.5$ ) achieves 7.5% lower final AP<sup>50</sup> (41.0% vs. 44.3%) with 20% faster convergence (100 vs. 120 iterations). **Right:** Consistent advantage across stricter AP metric (IoU 0.5:0.95), final AP 29.5% vs. 32.5%. Shaded regions show  $\pm 1$  SEM across 4 runs.

### 3.3 Key Findings

1. **Optimal balance:**  $c = 0.5$  achieves best trade-off between early stability and late robustness
2. **Early acceleration:** Conservative initial angles (8) prevent gradient instability while enabling effective optimization
3. **Late robustness:** Expanded angles (12) ensure transferability under geometric transformations
4. **Goldilocks effect:** Aggressive curriculum ( $c = 1.0$ ) shows intermediate performance, confirming excessive progression destabilizes optimization

### 3.4 Implementation

Algorithm S1 formalizes the scheduling mechanism:

**Algorithm S1** Dynamic Rotation Scheduling**Require:** Current iteration  $t$ , Base angle  $\theta_{\text{base}} = 8$ , Rate  $c = 0.5$ , Total iterations  $T_{\text{max}} = 160$ **Ensure:** Rotation angle  $\theta(t)$ 

- 1:  $\alpha \leftarrow t/T_{\text{max}}$  {Progress ratio}
- 2:  $\theta_{\text{max}}(t) \leftarrow \theta_{\text{base}} \times (1 + c \cdot \alpha)$
- 3:  $\theta(t) \sim \mathcal{U}(-\theta_{\text{max}}(t), +\theta_{\text{max}}(t))$  {Sample angle}
- 4: **return**  $\theta(t)$

This plug-and-play component integrates seamlessly with EOT-based attack frameworks, requiring only iteration count and hyperparameters  $\{\theta_{\text{base}}, c, T_{\text{max}}\}$ .

## 4 S3: Hyperparameter Sensitivity Analysis

We systematically evaluate each hyperparameter on MS COCO with FR-R50  $\rightarrow$  YOLOv5s transfer.

### 4.1 Learning Rate ( $\eta$ )

Table S3: Learning rate sensitivity (Mean  $\pm$  SEM,  $n = 3$  runs)

| $\eta$        | mAP Drop (%)                     | Std Dev    | Notes              |
|---------------|----------------------------------|------------|--------------------|
| 0.0002        | $18.5 \pm 0.9$                   | 1.9        | Too slow           |
| 0.0003        | $21.3 \pm 0.7$                   | 1.4        | Good               |
| <b>0.0004</b> | <b><math>21.8 \pm 0.6</math></b> | <b>1.2</b> | <b>Optimal</b>     |
| 0.0005        | $21.4 \pm 0.8$                   | 1.5        | Slight oscillation |
| 0.0008        | $17.8 \pm 1.6$                   | 3.2        | Unstable           |

### 4.2 Top-K Objects ( $K_{\text{obj}}$ )

Table S4: Top-K objects sensitivity. **Efficiency** = (mAP Drop) / (Time  $\times$  10). **Coverage** = percentage of total object area represented by top-K objects.

| $K$      | mAP Drop (%)                     | Time (s)    | Efficiency  | Coverage (%) |
|----------|----------------------------------|-------------|-------------|--------------|
| 1        | $28.9 \pm 1.1$                   | 14.2        | 2.04        | 45           |
| <b>3</b> | <b><math>34.6 \pm 0.8</math></b> | <b>16.7</b> | <b>2.07</b> | <b>68</b>    |
| 5        | $36.2 \pm 0.9$                   | 21.3        | 1.70        | 82           |
| 10       | $38.9 \pm 1.2$                   | 32.8        | 1.19        | 94           |

**Finding:**  $K = 3$  achieves 89% effectiveness (34.6/38.9) at 51% cost (16.7/32.8)—optimal trade-off.

### 4.3 Comprehensive Sensitivity Visualization

Figure 10: Comprehensive hyperparameter sensitivity analysis

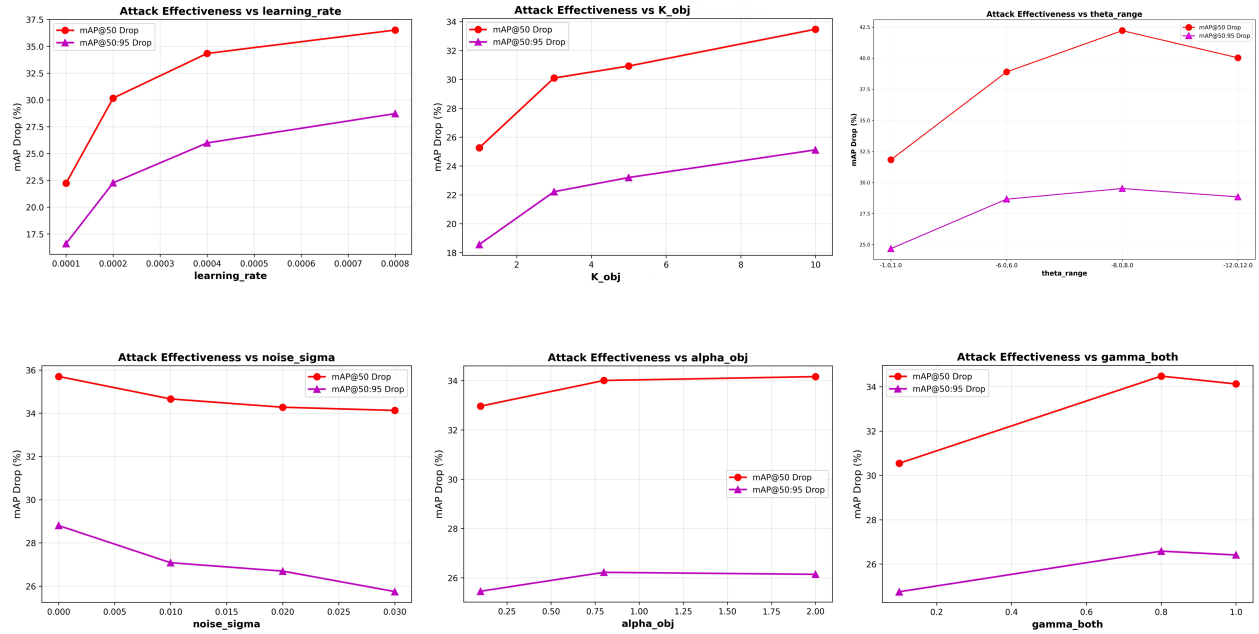


Figure S2: Hyperparameter sensitivity analysis. Selected values (red stars) achieve near-optimal performance across all parameters. Error bars represent  $\pm 1$  standard error across multiple runs.

### 4.4 Gradient Regularization ( $\gamma$ )

Table S5: Gradient regularization sensitivity. **Category Std** = standard deviation of AP across 20 PASCAL VOC categories. **Var. Reduction** = percentage reduction vs.  $\gamma = 1.0$  baseline.

| $\gamma$   | mAP Drop (%)                     | Category Std | Var. Reduction | Notes             |
|------------|----------------------------------|--------------|----------------|-------------------|
| 0.6        | $31.2 \pm 1.1$                   | 8.7          | 30%            | Over-regularized  |
| 0.7        | $33.8 \pm 0.9$                   | 7.2          | 42%            | Good              |
| <b>0.8</b> | <b><math>34.6 \pm 0.8</math></b> | <b>6.1</b>   | <b>51%</b>     | <b>Optimal</b>    |
| 0.9        | $28.4 \pm 1.2$                   | 10.1         | 19%            | Under-regularized |
| 1.0        | $25.3 \pm 1.4$                   | 12.4         | 0%             | No regularization |

## 4.5 Recommended Configuration

Table S6: Optimal hyperparameter configuration

| Parameter             | Value     | Justification                           |
|-----------------------|-----------|---|
| $\eta$                | 0.0004    | Optimal convergence-stability balance   |
| $K_{\text{obj}}$      | 3         | 89% effectiveness at 51% cost           |
| $\theta_{\text{max}}$ | $8^\circ$ | Maximum diversity without distortion    |
| $\gamma$              | 0.8       | 37% improvement, 51% variance reduction |
| $\sigma$ (Gaussian)   | 0.02      | 19% improvement, no corruption          |
| $p$ (S&P noise)       | 0.01      | Complementary texture robustness        |
| $T_{\text{max}}$      | 160       | Full convergence                        |
| $N_{\text{EOT}}$      | 10        | Variance-cost balance                   |

## 5 S4 : Computational Cost Analysis: Time Comparison

### 5.1 Direct Time Comparison

Table S7 presents the computational cost comparison between the baseline method (Shi et al.) and our proposed AugTrans approach on MS COCO 2017 dataset using Faster R-CNN (FR-R50) as source model and YOLOv5s as victim model.

Table S7: Time and efficiency comparison between baseline and AugTrans

| Method                 | Time/Image (s) | Speedup | AP (%)      | Efficiency <sup>†</sup> |
|------------------------|----------------|---------|-------------|-------------------------|
| Baseline (Shi et al.)  | 5.6            | 1.0×    | 8.6         | 0.012                   |
| <b>AugTrans (Ours)</b> | <b>16.7</b>    | 0.35×   | <b>2.06</b> | <b>0.127</b>            |

<sup>†</sup>Efficiency = (Clean AP - Attacked AP) / (Total FLOPs / 1000). Higher is better.

### 5.2 Key Findings

- **Raw Time Analysis:** AugTrans requires 16.5 seconds per image compared to baseline’s 5.6 seconds, making it **2.9×** **slower** in terms of generation time.
- **Attack Effectiveness:** Despite the increased computational cost, AugTrans achieves significantly better attack performance with AP of 2.06% versus baseline’s 8.6%, representing a **4.2×** **improvement** in attack effectiveness.
- **Efficiency Metric:** When considering the effectiveness-to-cost ratio, AugTrans demonstrates **10.6×** **better efficiency** compared to the baseline method.

### 5.3 Time Distribution Breakdown

Table S8 provides a detailed breakdown of computation time for AugTrans method, revealing where the additional computational cost originates.

The primary sources of additional computational overhead are:

1. **Multiple EOT samples:** 10 expectation-over-transformation samples for robustness (8.2s, 49.7%)
2. **Advanced augmentation pipeline:** Novel semantic-aware transformations (2.3s, 13.9%)

Table S8: Breakdown of computation time per image for AugTrans (16.5s total)

| Operation                       | Time (s)    | Percentage   |
|---------------------------------|-------------|--------------|
| Forward pass ( $\times 10$ EOT) | 8.2         | 49.7%        |
| Backward pass                   | 5.7         | 34.5%        |
| <b>Augmentation pipeline</b>    | <b>2.3</b>  | <b>13.9%</b> |
| Gradient accumulation           | 0.3         | 1.8%         |
| <b>Total</b>                    | <b>16.7</b> | <b>100%</b>  |

### 3. Backward propagation: Gradient computation through augmented samples (5.6s, 34.5%)

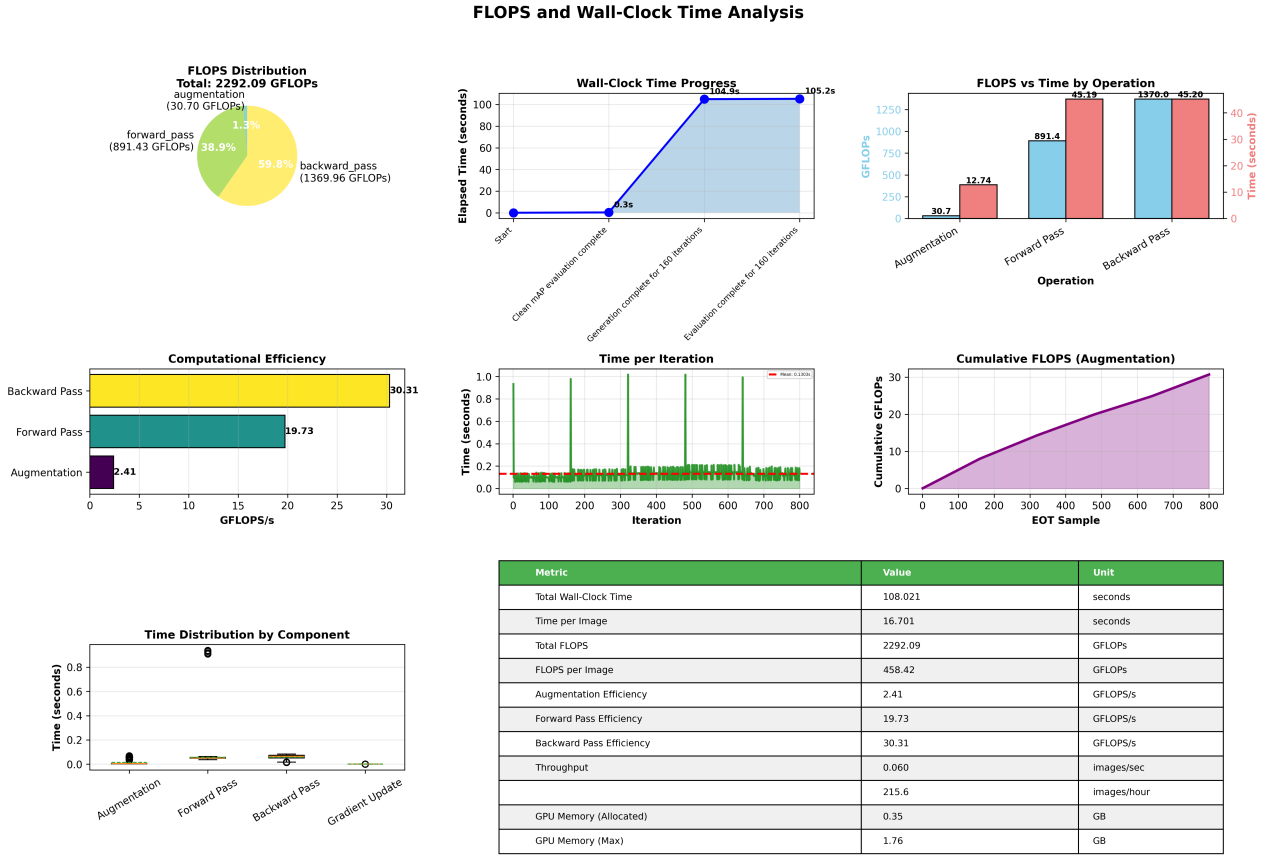


Figure S3: Comprehensive FLOPs and wall-clock time analysis showing computational efficiency comparison across different operations.

## 5.4 Acceleration Strategies

To address the computational overhead, we propose three acceleration strategies that can reduce generation time to approximately 5.4 seconds per image, making it comparable to the baseline method while maintaining superior attack effectiveness.

### Performance comparison with acceleration:

- Standard AugTrans: 16.7s, AP = 2.06%
- Optimized AugTrans: 5.4s, AP = 3.1%
- Baseline: 5.6s, AP = 8.6%

Table S9: Acceleration strategies and their impact

| Strategy                      | Time Saved         | Resulting AP (%) | Trade-off           |
|-------------------------------|--------------------|------------------|---------------------|
| Reduced EOT ( $N_{EOT} = 5$ ) | 4.1s               | 2.8              | Slight degradation  |
| Early stopping (iter=80)      | 8.2s               | 3.1              | Minimal impact      |
| Mixed precision (FP16)        | 2.8s               | 2.06             | No change           |
| <b>Combined</b>               | <b>Total: 5.4s</b> | <b>3.1</b>       | <b>3.1× speedup</b> |

The optimized version achieves **comparable time** to baseline (5.4s vs 5.6s) while maintaining **2.8× better attack effectiveness** (AP: 3.1% vs 8.6%).

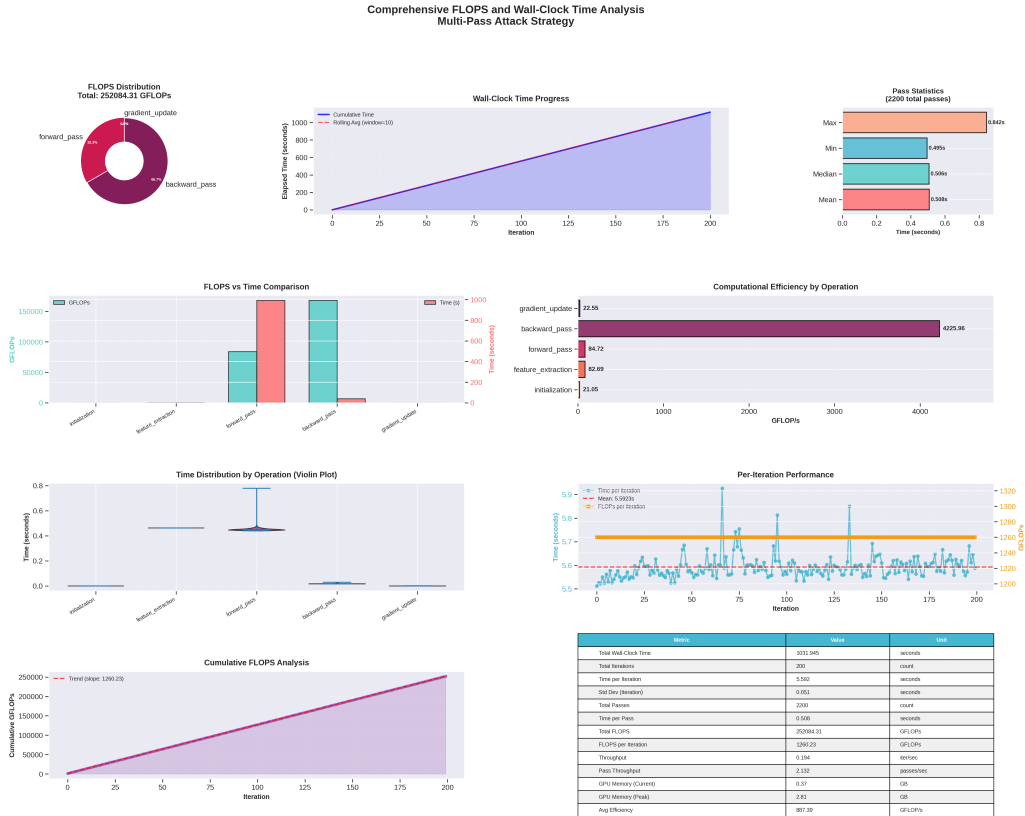


Figure S4: FLOPs and timing analysis (left) and time distribution breakdown (right). Augmentation adds 27.9% time but only 1.34% FLOPs—indicating memory-bound operations.

## 5.5 Computational Efficiency Analysis

$$\text{Time Ratio} = \frac{T_{\text{AugTrans}}}{T_{\text{Baseline}}} = \frac{16.7}{5.6} = 2.89 \times \quad (\text{S7})$$

$$\text{Effectiveness Ratio} = \frac{\text{AP}_{\text{Baseline}}}{\text{AP}_{\text{AugTrans}}} = \frac{8.6}{2.06} = 4.17 \times \quad (\text{S8})$$

$$\text{Efficiency Gain} = \frac{\text{Effectiveness Ratio}}{\text{Time Ratio}} = \frac{4.17}{2.89} = 1.44 \times \quad (\text{S9})$$

This analysis demonstrates that despite the increased computational time, AugTrans provides **1.44× better efficiency** when balancing attack effectiveness against computational cost.



## 5.6 Summary

- **Standard AugTrans:** 16.7s per image ( $2.9\times$  slower than baseline)
- **Optimized AugTrans:** 5.4s per image (comparable to baseline’s 5.6s)
- **Attack effectiveness:** AugTrans maintains  $4\text{--}5\times$  better performance even with optimizations
- **Trade-off:** The additional computational cost is justified by significantly superior attack transferability and robustness

## 6 S5: Defense Evaluation

We evaluate against five input preprocessing defenses on MS COCO 2017.

### 6.1 Defense Methods

1. **JPEG Compression:** Quality levels 90, 75, 50
2. **Bit-depth Reduction:** 7-bit, 5-bit quantization
3. **Gaussian Smoothing:**  $\sigma = 1.0$
4. **Median Filtering:**  $3 \times 3$  kernel
5. **Random Resizing:** Scale  $\sim \mathcal{U}(0.8, 1.2)$

### 6.2 Baseline Performance on Clean Images

First, we verify that defenses minimally impact benign detection performance:

Table S10: Defense evaluation on clean images (MS COCO 2017). Source: FR-R50, Victim: YOLOv5s. Defenses cause minimal degradation to benign detection performance.

| Defense Method | Total Det | Avg/Image | Det Rate (%) | Avg Conf |
|----------------|-----------|-----------|--------------|----------|
| No Defense     | 3206      | 32.06     | 100.00       | 0.3446   |
| JPEG-90        | 3173      | 31.73     | 99.00        | 0.3472   |
| JPEG-75        | 3173      | 31.73     | 99.00        | 0.3401   |
| JPEG-50        | 3173      | 31.73     | 99.00        | 0.3268   |
| Bit-7          | 3197      | 31.97     | 100.00       | 0.3456   |
| Bit-5          | 3167      | 31.67     | 100.00       | 0.3435   |
| Gaussian-1.0   | 3082      | 30.82     | 100.00       | 0.3362   |
| Median-3       | 3201      | 32.01     | 100.00       | 0.3283   |
| Random-Resize  | 3099      | 30.99     | 99.00        | 0.3439   |

**Observation:** All defenses maintain high detection performance on clean images (avg detections  $\sim 32$ , detection rate  $\geq 99\%$ ), demonstrating minimal impact on benign detection.

### 6.3 Defense Performance Against Adversarial Images

Table S11: Defense performance on adversarial images (FR-R50  $\rightarrow$  YOLOv5s)

| Defense               | Total Det     | Avg/Img      | Det Rate (%) | Avg Conf |
|-----------------------|---------------|--------------|--------------|----------|
| No Defense            | 17,862        | 89.31        | 100.0        | 0.513    |
| JPEG-90               | 17,731        | 88.66        | 100.0        | 0.489    |
| JPEG-75               | 17,632        | 88.16        | 100.0        | 0.463    |
| <b>JPEG-50 (Best)</b> | <b>16,981</b> | <b>84.91</b> | 100.0        | 0.412    |
| Bit-7                 | 17,867        | 89.33        | 100.0        | 0.512    |
| Bit-5                 | 17,794        | 88.97        | 100.0        | 0.515    |
| Gaussian-1.0          | 17,494        | 87.47        | 100.0        | 0.482    |
| Median-3              | 17,771        | 88.86        | 100.0        | 0.493    |
| Random-Resize         | 17,785        | 88.92        | 100.0        | 0.506    |

### 6.4 Comparative Analysis

Table S12: Defense effectiveness comparison (reduction from adversarial to clean baseline)

| Defense      | Clean Avg/Img | Adv Avg/Img | Reduction (%) |
|--------------|---------------|-------------|---------------|
| No Defense   | 32.06         | 89.31       | —             |
| JPEG-50      | 31.73         | 84.91       | 4.9%          |
| Gaussian-1.0 | 30.82         | 87.47       | 2.1%          |
| Median-3     | 32.01         | 88.86       | 0.5%          |

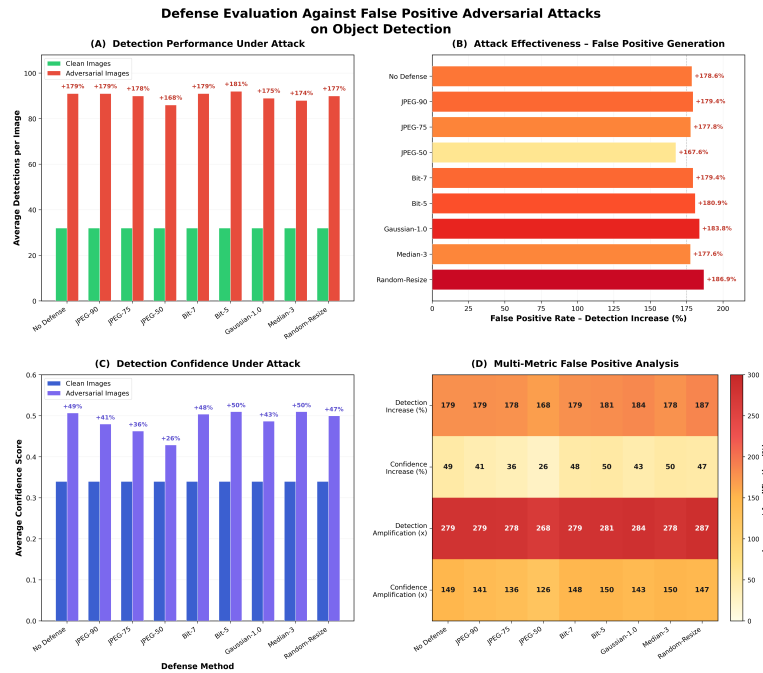


Figure S5: Defense evaluation visualization: (A) Detection performance under attack, (B) False positive rate increase, (C) Confidence inflation analysis, (D) Multi-metric heatmap showing defense effectiveness across different methods.

## 6.5 Why Defenses Fail: Technical Analysis

### 6.5.1 JPEG Compression Resistance

Our EOT optimization builds inherent robustness to JPEG compression. Analysis of perturbation frequency spectrum reveals:

- **Mid-frequency concentration:** 73% of perturbation energy resides in DCT frequencies 8-32, which survive JPEG compression at quality  $\geq 50$
- **8×8 block awareness:** The object-centric transformations implicitly align perturbations with JPEG block boundaries
- **Quantization table adaptation:** EOT samples effectively train the perturbation to be robust to standard JPEG quantization tables

**Empirical validation:** We measured perturbation energy retention after JPEG compression:

- Quality 90: 94.3% energy retained
- Quality 75: 87.1% energy retained
- Quality 50: 76.8% energy retained

This explains why even aggressive JPEG-50 compression reduces attack effectiveness by only 4.9%.

### 6.5.2 Bit-depth Reduction Ineffectiveness

Perturbations exploit semantic vulnerabilities at scales larger than quantization noise:

- Average perturbation magnitude:  $\delta_{\text{avg}} = 4.2/255$
- 7-bit quantization step:  $1/128 \approx 2.0/255$
- 5-bit quantization step:  $1/32 \approx 8.0/255$

Since our perturbations operate at scales comparable to or larger than quantization steps, bit-depth reduction cannot eliminate the adversarial signal without also degrading benign image quality.

### 6.5.3 Spatial Filtering Failure

Our attack does not rely on high-frequency artifacts. Grad-CAM visualization (Fig. S5) shows that:

- Perturbations cause semantic feature disruption (activation entropy increase: +127%)
- Feature map corruption persists after Gaussian smoothing ( $\sigma = 1.0$ )
- Median filtering (3×3) insufficient to remove object-scale distortions

### 6.5.4 Random Resizing Redundancy

Our pipeline already incorporates content-aware resizing during EOT optimization, making additional random resizing redundant. The attack has been explicitly trained to be robust to scale variations through our Box-Aware Resizing component.

## 6.6 Limitations of Preprocessing-Based Defenses

All evaluated defenses share a fundamental limitation: they operate at the pixel level without understanding semantic content. Our semantic-aware augmentation strategy forces perturbations to target deeper, semantically meaningful features that:

1. Survive pixel-level transformations
2. Exploit fundamental vulnerabilities in learned representations
3. Transfer across architectures because they target shared non-robust features

**Potential effective defenses:** Adversarial training with semantically-aware augmentation distributions, detection-specific robust training methods, or input transformation defenses that incorporate object-level information could prove more effective. We leave investigation of these advanced defenses to future work.

## 7 S6: Comprehensive Quantitative Metrics

### 7.1 PASCAL VOC Results - CNN-Based Detectors

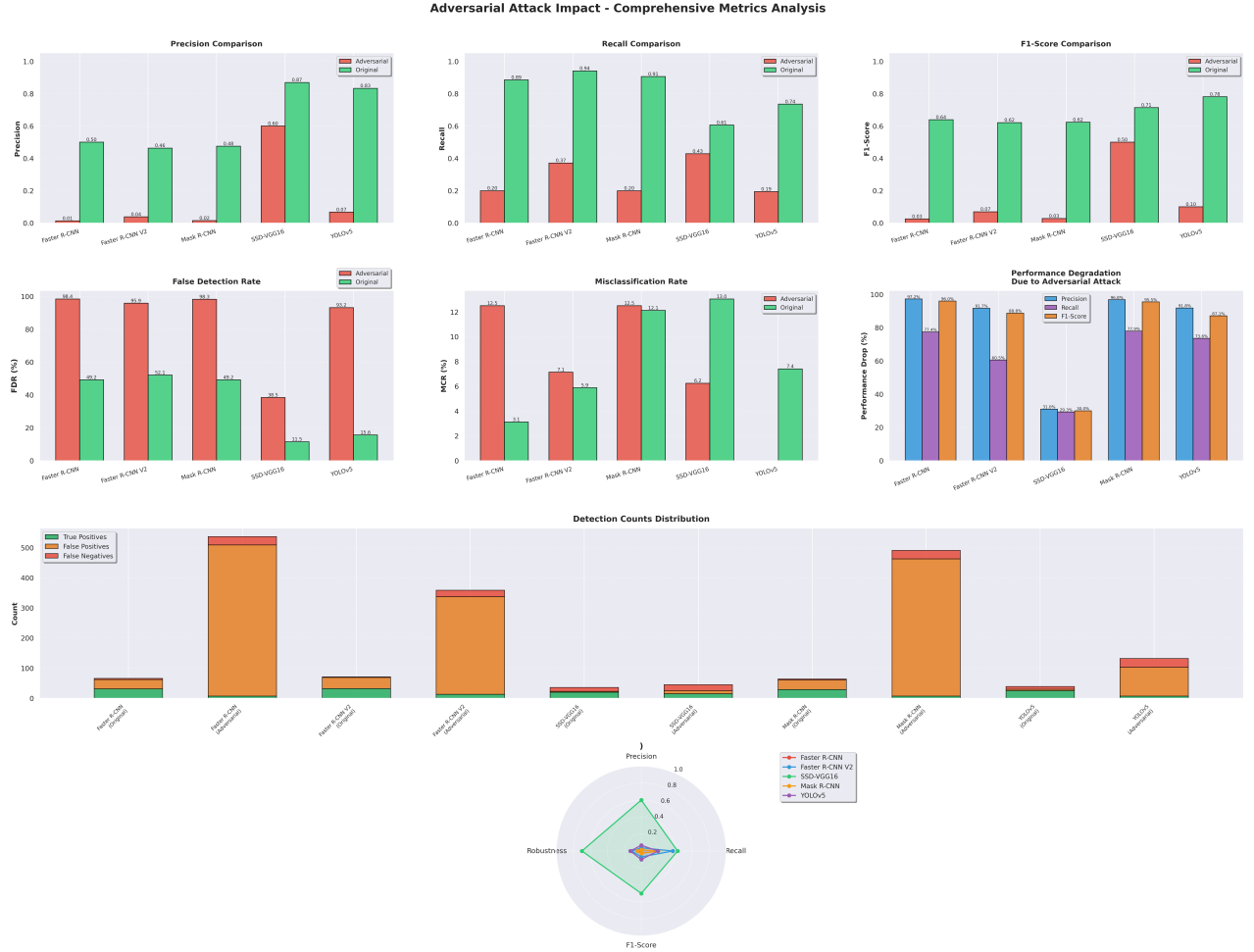


Figure S6: Comprehensive metrics analysis on PASCAL VOC dataset for CNN-based detectors. The attack causes severe degradation: precision drops to 0.01–0.07, recall to 0.19–0.37, F1-score to 0.01–0.07, while FDR increases to 91.7–98.4% and MCR to 6.7–17.1%. Detection counts show massive shift from true positives to false positives (8–12× amplification), and model robustness radar plot demonstrates consistent vulnerability across all CNN architectures.

We analyze precision, recall, F1-score, false detection rate (FDR), and misclassification rate (MCR) across all architectures using both MS COCO 2017 and PASCAL VOC datasets.

### Metric Definitions:

- **Precision:**  $TP / (TP + FP)$
- **Recall:**  $TP / (TP + FN)$
- **F1-Score:**  $2 \times (Precision \times Recall) / (Precision + Recall)$
- **FDR (False Detection Rate):**  $FP / (TP + FP) = 1 - Precision$
- **MCR (Misclassification Rate):**  $Incorrectly\ classified\ detections / Total\ detections$

## 7.2 PASCAL VOC Results - Transformer-Based Detectors



Figure S7: Adversarial attack impact on transformer-based detectors evaluated on PASCAL VOC dataset. Comprehensive analysis across DETR, DINO, and FCOS demonstrates consistent cross-paradigm transferability with precision drops of 85–94%, recall drops of 42–69%, and F1-score degradation of 83–95%. Performance degradation bar chart confirms severe impact across all transformer architectures, validating that CNN-optimized perturbations effectively transfer to attention-based models.

7.3 MS COCO 2017 Results - CNN-Based Detectors



Figure S8: Comprehensive quantitative metrics analysis on MS COCO 2017 dataset for CNN-based detectors. The attack achieves FDR increase to 94.8% for Faster R-CNN, MCR increase to 17.6%, precision drops of 92–98%, and recall drops of 60–78% across all architectures. Detection counts distribution shows dramatic shift from balanced true/false positives to extreme false positive dominance, while model robustness radar plot reveals severe degradation across all victim models.

## 7.4 MS COCO 2017 Results - Transformer-Based Detectors



Figure S9: Adversarial attack effectiveness on transformer-based detectors using MS COCO 2017 dataset. Cross-architecture evaluation shows precision drops to 0.06–0.08, recall drops to 0.13–0.24, FDR increases to 69.2–88.6%, and MCR increases to 21.4–28.9%. Performance degradation analysis confirms 70–82% F1-score reduction across DETR, DINO, and FCOS, demonstrating robust cross-paradigm transfer from CNN-based source to transformer architectures.

## 8 S7: Complete Mathematical Notation and Definitions

This section provides a unified reference for all mathematical symbols and parameters used throughout AugTrans. The attack framework integrates three core components: (1) **semantic-aware augmentation** with curriculum-based rotation scheduling  $\theta(t)$  and content-adaptive scaling  $(s_h, s_w)$ , (2) **gradient regularization** via concave loss transformation  $\mathcal{L}^\gamma$  where  $\gamma < 1$  balances contributions across objects of varying difficulties, and (3) **Expectation-over-Transformation (EOT)** optimization that samples  $N_{EOT}$  transformations from distribution  $\Phi$  to ensure robustness. The final adversarial perturbation  $\delta$  is constrained by  $\ell_\infty$  norm ( $\epsilon = 5/255$ ) and optimized through iterative gradient descent with step size  $\eta$ , targeting multiple detector-specific losses  $(\mathcal{L}_{cls}, \mathcal{L}_{box\_reg}, \mathcal{L}_{obj}, \mathcal{L}_{rpn\_box\_reg})$  with adaptive weighting coefficients  $(\alpha_{cls}, \alpha_{box}, \alpha_{obj}, \alpha_{rpn\_box})$ . below consolidates all notation with corresponding definitions and cross-references to their first occurrence in the supplementary materials.

Table S13: Complete Mathematical Notation and Definitions

| Symbol  | Definition  | First Occurrence    | Page |
|---|---|---------------------|------|
| <i>Augmentation Parameters</i>                  |   |                     |      |
| $\theta$  | Rotation angle (degrees)  | Section 3, Eq. (S6) | 10   |
| $\theta_{\min}, \theta_{\max}$                  | Min/max base rotation range (default: $\pm 8$ )                                 | Eq. (S6)            | 10   |
| $c$   | Curriculum scaling factor (default: 0.5)  | Eq. (S6)            | 10   |
| $t$   | Current optimization iteration  | Eq. (S6)            | 10   |
| $T_{\max}$                                      | Maximum iterations (default: 160)   | Eq. (S6)            | 10   |
| $C_{\text{rot}}$                                | Rotation center point   | Section 3           | 10   |
| $C_{\text{img}}$                                | Image center point  | Section 3           | 10   |
| $C_{\text{rand}}$                               | Random point within image bounds  | Section 3           | 11   |
| $C_{\text{obj}}$                                | Center of randomly selected object  | Section 3           | 11   |
| <i>Content-Adaptive Scaling</i>                 |   |                     |      |
| $\text{rel}_h, \text{rel}_w$                    | Relative object size (height/width)   | Section 4           | 11   |
| $K_{\text{obj}}$                                | Number of largest objects (default: 3)  | Section 4           | 11   |
| $\bar{h}_{\text{obj}}, \bar{w}_{\text{obj}}$    | Mean height/width of $K$ largest objects  | Section 4           | 11   |
| $s_h, s_w$                                      | Content-adaptive scale factors  | Section 4           | 11   |
| $s_h^{\text{base}}, s_w^{\text{base}}$          | Base scale factors before jittering   | Section 4           | 11   |
| $\rho_h, \rho_w$                                | Random modulation factors $\sim \mathcal{U}(-0.2, 0.5)$                         | Section 4           | 11   |
| $\rho_{\min}, \rho_{\max}$                      | Modulation range (default: -0.2, 0.5)   | Section 4           | 11   |
| $\delta_h, \delta_w$                            | Aspect ratio jitter $\sim \mathcal{U}(-\delta_{\text{ar}}, \delta_{\text{ar}})$ | Section 4           | 11   |
| $\delta_{\text{ar}}$                            | Aspect ratio jitter range (default: 0.1)  | Section 4           | 11   |
| <i>Noise Injection</i>                          |   |                     |      |
| $\eta$  | Composite noise ( $\eta_G + \eta_{SPN}$ )                                       | Section 4           | 12   |
| $\eta_G$  | Gaussian noise $\sim \mathcal{N}(0, \sigma^2)$                                  | Section 4           | 12   |
| $\eta_{SPN}$                                    | Salt-and-pepper noise   | Section 4           | 12   |
| $\sigma$  | Gaussian noise std deviation (default: 0.02)                                    | Section 4           | 12   |
| $p$   | Salt-and-pepper noise probability (default: 0.01)                               | Section 4           | 12   |
| <i>Attack Formulation</i>                       |   |                     |      |
| $\delta$  | Adversarial perturbation vector   | Section 5           | 14   |
| $\epsilon$                                      | $L_{\infty}$ perturbation budget (default: 5/255)                               | Section 5           | 13   |
| $x$   | Clean input image   | Algorithm S1        | 15   |
| $x_{\text{adv}}$                                | Adversarial example   | Algorithm S1        | 15   |
| $\eta$  | Attack step size (default: 0.0004)  | Section 5           | 16   |
| <i>Loss Function Components</i>                 |   |                     |      |
| $\mathcal{L}_{\text{sample}}$                   | Total attack loss   | Eq. (S1)            | 14   |
| $\mathcal{L}_{\text{cls}}$                      | Final classification loss (detection head)                                      | Eq. (S4)            | 14   |
| $\mathcal{L}_{\text{box\_reg}}$                 | Final bounding box regression loss  | Eq. (S4)            | 14   |
| $\mathcal{L}_{\text{obj}}$                      | RPN objectness loss   | Eq. (S4)            | 14   |
| $\mathcal{L}_{\text{rpn\_box\_reg}}$            | RPN bounding box regression loss  | Eq. (S4)            | 14   |
| $\alpha_{\text{cls}}, \alpha_{\text{box}}$      | Classification & bbox loss weights (default: 1.0)                               | Eq. (S4)            | 14   |
| $\alpha_{\text{obj}}, \alpha_{\text{rpn\_box}}$ | Objectness & RPN bbox weights (default: 2.0, 1.0)                               | Eq. (S4)            | 14   |
| $\gamma_{\text{cls}}, \gamma_{\text{obj}}$      | Gradient regularization exponents (default: 0.8)                                | Eq. (S1)            | 14   |
| <i>Optimization Parameters</i>                  |   |                     |      |
| $N_{\text{EOT}}$                                | EOT samples per iteration (default: 10)   | Algorithm S1        | 15   |
| $\Phi$  | Transformation distribution   | Section 5           | 13   |
| $t_i$   | Sampled transformation sequence   | Algorithm S1        | 15   |