Defending Object Detection Models against Image Distortions

Mark Ofori-Oduro

A Thesis
In The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

September 2024

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mark Ofori-Oduro**

Entitled: **Defending Object Detection Models against Image Distortions**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Luis Amador _____ Chair
Dr. Carlos Vázquez _____ External Examiner
Dr. Eugene Belilovsky _____ External to the Department
Dr. Wei-Ping Zhu _____ Examiner
Dr. William E. Lynch _____ Examiner
Dr. Maria Amer _____ Thesis Supervisor

Approved by: _____
Dr. Jun Cai, Graduate Program Director

_____ 20 _____ _____
Dr. Mourad Debbabi
Dean
Gina Cody School of Engineering and Computer Science

Abstract

Defending Object Detection Models against Image Distortions

Mark Ofori-Oduro, Ph.D.
Concordia University, 2024

Object detection has significantly advanced with deep learning but faces challenges under image distortions like noise, compression, blur, fog, and snow. This issue is critical in applications such as self-driving cars and healthcare. While defence methods aim to enhance robustness under distortions, maintaining performance on clean images remains a challenge. To address this challenge, we propose a novel defence approach that generates copies of the original training images and adds distortion-like content to these copies at the pixel level. We balance the number of distorted pixels to prevent bias during learning. Our approach includes two augmentation methods to generate augmented samples: AISbod, which uses artificial immune systems (AIS), and GSES, which employs kernel density estimation (KDE).

Our AISbod uses AIS to distort the original sample (antigen) through cycles of "select, clone, mutate, select" until the augmented data (antibody) reaches a specified similarity to the antigen. However, AIS is limited in diversifying generated antibodies and is computationally expensive. Therefore, in GSES, we create samples by selecting pixels from the original samples, estimating the pixel distribution from multiple distorted versions of the original samples via KDE, and then replacing the selected pixels with new values sampled from the estimated distribution. GSES generates more diverse data than AISbod.

We evaluate our methods on 15 image distortions using state-of-the-art object detection models like DINO and YOLOv7. Our methods improve accuracy under distorted and clean images and remain consistent across datasets and detection models. For instance, DINO on the COCO dataset shows a 4.50% improvement under clean samples, 8.40% on average across all distortions, 2.50% under snow, and 29.30% under impulse noise. The observed improvement is due to the weight regularization effect of our methods, which is evident in the smoother convergence of training and validation loss curves, indicating reduced learning fluctuations and a more stable optimization path. Additionally, the narrower gap between the curves suggests reduced over-fitting, leading to better generalization to unseen data. Simulations indicate that our approach outperforms related defence methods against distortions and extends beyond object detection, improving accuracy in image classification and object tracking models.

Acknowledgments

I extend my heartfelt thanks to all who have significantly contributed to completing this arduous thesis journey. The steadfast support, guidance, and encouragement from my supervisor, colleagues, family, and friends have been crucial.

First and foremost, I express my most profound appreciation to my supervisor, Dr. Maria Amer. Her supervision style compelled me to pursue excellence in my research and writing, leading to the completion of the thesis and publications. I am profoundly grateful for the countless hours she devoted to my work and the additional bursary provided, which was beyond what was initially mentioned in my admission letter.

I am deeply grateful to Dr. Carlos Vázquez from École de technologie supérieure for graciously agreeing to serve as the External Examiner. I also thank the doctoral examination committee members, Dr. Wei-Ping Zhu, Dr. William E. Lynch, and Dr. Thomas Fevens, for their invaluable feedback throughout my PhD program. Furthermore, I am obliged to Dr. Eugene Belilovsky for serving as the Arms-length Examiner and Dr. Luis Amador for chairing my defence.

I appreciate the former and current members of the VidPro group, including Goutam, Doreen, Julien, Amin, Saeid, Prateek, Milad, and Mahdi. Each interaction with you has provided fresh perspectives on both life and research. I extend a special thank you to my roommate and friend, Mathew Birgen, for his unwavering support. Your advice and encouragement have been instrumental in my research and personal growth.

My sincere gratitude goes to the Fraud Analytics team at TD Insurance and the Supply Chain Analytics team at Loblaw Companies Limited for the internship. The invaluable experiences allowed me to apply machine learning skills to real-world problems, effectively bridging the gap between academic research and industry requirements and significantly contributing to my professional growth.

I am eternally grateful to my family for their unending encouragement, understanding, and patience throughout my PhD journey. I sincerely appreciate my parents, Godfred and Felicia, and my uncle, Philip, for their boundless love, unwavering support, and prayers. A heartfelt thank you to my fiancée, Maame, for her emotional support through the highs and

lows of my academic pursuit. Her understanding and care have been a constant source of strength.

Lastly, I thank my friends Dr. Sarfo Kojo Gyamfi and Sylvester Ankamah-Kusi. You have been like brothers to me, and your generosity and constant support have been invaluable. For that, I am forever in your debt.

In conclusion, I acknowledge all those who have contributed to my academic and personal growth, even if they are not mentioned by name. A special note of gratitude goes to my Lord and personal saviour, Jesus Christ. His word, particularly in *1 Corinthians 1:27-31: But God chose the foolish things of the world to shame the wise; God chose the weak things of the world to shame the strong. God chose the lowly things of this world and the despised things—and the things that are not—to nullify the things that are so that no one may boast before him. It is because of him that you are in Christ Jesus, who has become for us wisdom from God—that is, our righteousness, holiness and redemption. Therefore, as it is written: "Let the one who boasts boast in the Lord"*. This word has given me strength and comfort through all difficulties.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network
mAP	mean Average Precision
AIS	Artificial Immune Systems
KDE	kernel density estimation
ViT	Vision Transformer
NLP	Natural Language Processing
RPN	Region Proposal Network
MAE	Masked AutoEncoder
CLONALG	Clonal Selection Algorithm
AIN	Artificial Immune Network
NSA	Negative Selection Algorithm
GAN	Generative Adversarial Networks
AISbod	AIS as a data augmentation
GSES	Generation of data augmentation using KDE
IoU	Intersection-over-Union
mPC	Mean Performance under Corruption
rPC	Relative Performance under Corruption
AO	Area Overlap

SR	Success Rate
GPU	Graphics Processing Unit
CPU	Central Processing Unit
fps	Frames-per-second

List of AISbod variables

Ag^*	Antigens (original samples)
p	Augmentation ratio: hyper-parameter of our method. See Section 5.1 for details.
τ	Affinity threshold: hyper-parameter of our method. See Section 5.1 for details.
Ab_m	Memory antibodies
Ab_r	Response antibodies
K^*	Number of antigens
K	Selected of antigens
L	Resolution of largest Ag^*
Ab	Antibodies (distorted samples)
$Ab(k)$	k^{th} Antibody of Total K Antibodies
$Ag(k)$	k^{th} Antigen of Total K Antigens
D_{kk}	The total pixel matches between $Ab(k)$ and $Ag(k)$
n	Selected antibodies before mutation: value selected based on work in [1].
N	Cloning constant: value was chosen based on work in [1].
α	Multiplying factor: value picked based on work in [1].
Ab_Q	Cloned antibodies: value determined based on work in [1].
Ab_{Q^*}	Matured antibodies

s	Number of randomly selected pixels for mutation; value chosen based on work in [1].
l	Pixel in $Ab(k)$ or $Ag(k)$
q^*	Selected antibodies after mutation; value selected based on work in [1]

List of GSES variables

$\Psi(\cdot)$	Approximate probability density function
S	Number of kernels
$\Phi(\cdot)$	Function of kernels
w	Width of kernels
x	Data point
τ	Affinity threshold: hyper-parameter of our method. See Section 5.1 for details.
D	Distortions types
Ag	Original training images
M	Number of training images
Ab_d	Distortion group samples
T	Number of images in each Ab_d
p	Augmentation ratio: hyper-parameter of our method. See Section 5.1 for details.
Z	Number of distorted samples
$\{\tilde{x}_i\}^Z$	Distorted samples
K	Number of selected pixels in an image for distortion
$\tilde{X}$	Distorted vector
$E_{\tilde{X}}$	Estimation of $\tilde{X}$
Ab	Generated new (distorted) samples

Chapter 1

Introduction

Object detection models are the backbone of many artificial intelligence applications, enabling machines to interpret and comprehend the visual world. These models are pivotal in identifying and locating objects within an image or video frame, classifying them into predefined categories, and determining their precise positions using bounding boxes and confidence scores, as shown in Figure 1.1. The transition from simple classification of single objects to complex detection of multiple objects has significantly bolstered their capabilities. Object detection applications span industries such as autonomous vehicles, surveillance, healthcare, and retail. The current research landscape focuses on enhancing accuracy, efficiency, and robustness, employing supervised and unsupervised learning techniques. While convolutional neural networks (CNNs) have long been the frontrunners, transformer models are now setting new benchmarks, pushing the boundaries of what machines can comprehend and achieve in real-world scenarios.

1.1 Problem statement

Despite significant advancements due to deep learning, the fundamental computer vision task object detection [2–5] remains vulnerable to image distortions [6–9] such as shown in Figure 1.2. For instance, the accuracy (mAP) of state-of-the-art object detection models DINO [2] (transformer-based) and YOLOv7 [3] (CNN-based) dropped significantly when validated under snow (medium severity 3) by 24.50% and 15.90%, and under impulse noise by 27.10% and 24.90%, respectively, for the COCO2017 dataset. Figure 1.3 demonstrates how object detection struggles with Gaussian and impulse noise, often misclassifying or missing objects entirely.

Image distortions are prevalent in real-world applications such as self-driving cars and healthcare monitoring, where the accuracy of object detection models is critical. Defence

Figure 1.1: Two images with colour-coded bounding boxes labelled with the object category and confidence (conf.) score, illustrating the DINO model's performance. In the left image, detected objects include Person 1 (conf. 0.64), Person 2 (conf. 0.89), and a Chair (conf. 0.67). In the right image, detected objects include Person (conf. 0.97), Baseball bat (conf. 0.93), and Baseball ball (conf. 0.98).

methods against image distortions address this vulnerability through noise filtering before feeding images into models [10], modifying the baseline model's architecture [11], or augmenting training data with examples of distorted images [12, 13]. The main challenge in these defence methods is improving performance under distortions while maintaining (or improving) performance on clean samples. Additionally, methods must generalize well to distortions beyond those used to develop the defence strategy. For example, the approach of simply adding known distortions to training samples leads to poor generalisation [13–16].

This thesis aims to fill this critical gap by proposing innovative methodologies that enable object detection models to handle and adapt to a wide range of distortions and clean samples. Robust object detection-based systems can significantly improve efficiency, accuracy, and reliability in healthcare, security, transportation, and beyond.

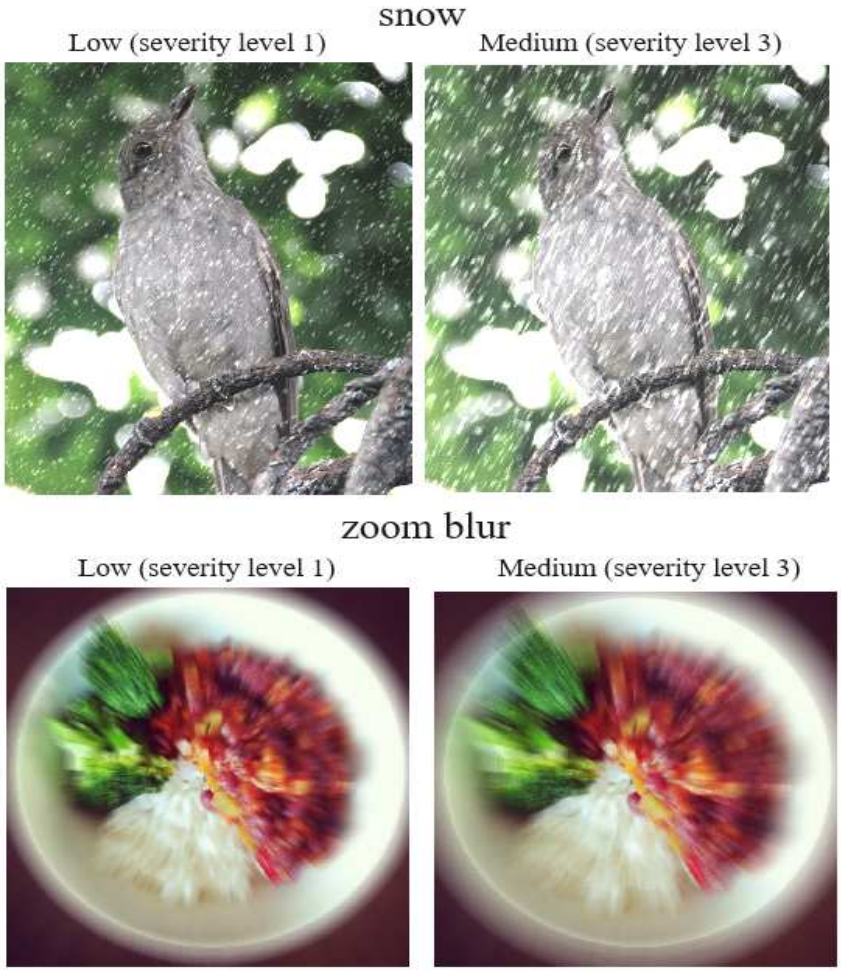

Figure 1.2: Examples of image distortions at varying severity levels. Top row: snow distortion applied to an image of a bird, with low (severity level 1) on the left and medium (severity level 3) on the right. Bottom row: zoom blur distortion applied to an image of a dish, with low (severity level 1) on the left and medium (severity level 3) on the right.

1.2 Objectives

Our main objective is to improve object detection accuracy under distortions while maintaining (or improving) performance (accuracy and speed) under clean (original) samples. To achieve this objective, we will follow these directions:

1. **Interpretability:** It is crucial to investigate the underlying mechanisms to understand how the proposed methods enhance the accuracy of object detection models under various distortions. One practical approach is to visualize the training and validation curves or weights of the trained models with and without the proposed augmentation. By comparing the curves or weights, we can gain valuable insights into the changes

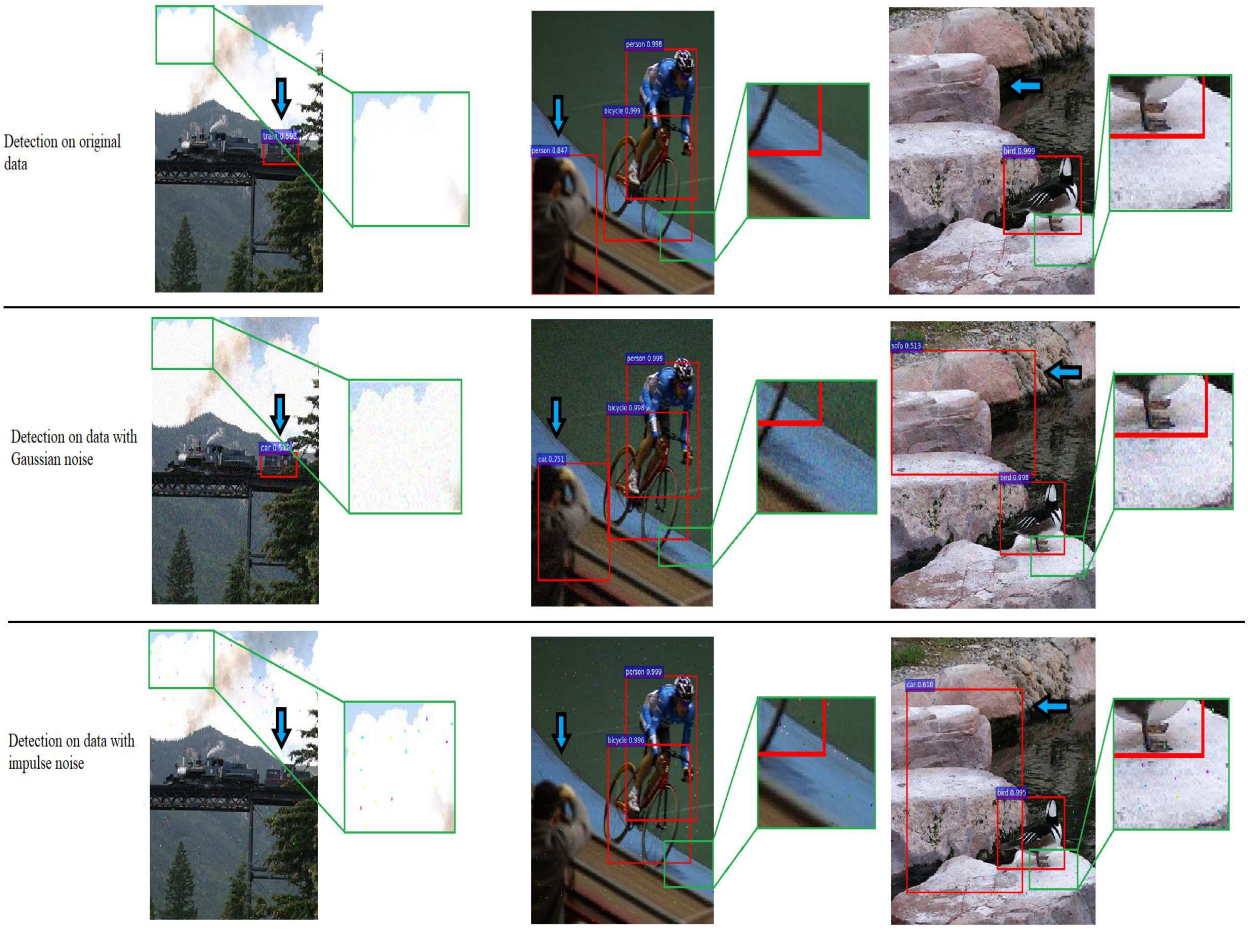

Figure 1.3: Effect of 27 dB PSNR Gaussian and 0.25% impulse noise (added to the RGB input image) on Faster RCNN object detector: For instance, a train is falsely detected as a car (row 2, column 1), a person is either misclassified (row 2, column 2) or missed (row 3, column 2), or rock is falsely detected as a car (row 3, column 3). Blue arrows indicate points of interest.

within the model, contributing to its improved accuracy. For example, a narrower gap between the curves will indicate regularization of the model.

2. **Generalization:** It is essential to demonstrate if the proposed approach has a strong performance across a range of distortions and model architectures. This demonstration will validate the approach's ability to generalize and adapt effectively to diverse distortion types and detection models. This investigation will provide valuable insights into the robustness and versatility of our approach.

3. **Cross-domain generalization:** It will be interesting to see if features learnt by a model using our data augmentation approach are transferable to unseen datasets and different computer vision applications such as image classification [17, 18] or object

tracking [19, 20].

1.3 Thesis statement

Our investigation focuses on enhancing the accuracy of object detection models when faced with a comprehensive range of image distortions. These distortions can be broadly categorized into four main types: noise (Gaussian noise, impulse noise, shot noise), artefacts (jpeg compression, pixelation, elastic transform), blur (motion blur, glass blur, zoom blur, defocus blur), and weather-related conditions (contrast, snow, fog, frost, brightness). Our research strives to improve the accuracy of these models under distortion by operating at the fundamental pixel level. This approach involves introducing diverse distortion-like content to the image and balancing the amount of distorted pixels without compromising the accuracy of clean samples.

We explore two main concepts: Artificial Immune Systems (AIS) and kernel density estimation (KDE). The AIS-based method (AISbod) re-models AIS to enhance model accuracy. This approach is distortion-agnostic and highly adaptable across various model architectures and computer vision tasks (including object detection, image classification, and object tracking). Our KDE-based technique (GSES) expands upon randomization and diversification by incorporating known distortion distributions, further increasing accuracy. Like the AIS-based method, the KDE-based innovation is applicable across multiple model architectures and tasks.

This research contributes to developing more adaptable models and fostering their integration into diverse real-world settings. The ultimate goal is to bridge the gap between theoretical advancements and reliable implementation despite challenging distortions.

1.4 Organization of the Thesis

The rest of the dissertation is organized as follows:

- **Review of related works**: Chapter 2 provides an extensive survey of the literature, encompassing a range of works from computer vision models and data augmentation methods to image defence methods. This Section benchmarks our work against existing studies and highlights the gaps and opportunities our research aims to address.

- **Details of the proposed methods**: In Chapter 3, we present our proposed methods: AISbod, our AIS-based augmentation and GSES, our KDE-based augmentation.

- **Experimental results**: Chapter 4 is dedicated to presenting the empirical outcomes of our methods. We present detailed experimental findings and thoroughly analyze them to show the effectiveness of our proposed methods.

- **Analysis of proposed methods:** In Chapter 5, we present an analysis of how the hyper-parameters (augmentation ratio p and affinity threshold τ) of our data augmentation methods AISbod and GSES are selected. We also demonstrate the effectiveness of the proposed methods by analyzing the impact of various processes and components. Additionally, we examine how model complexity influences these methods and their effect on the stability of the models.

- **Conclusion and future work proposals**: In Chapter 6, we synthesize our key findings and reflect on their contributions to the computer vision field. We also chart out potential avenues for future research inspired by the insights and limitations unearthed through our investigations.

- **Appendix:** The appendices provide supplementary material to support the main text. In Appendix A, we provide a detailed breakdown of the performance of AISbod and GSES for each distortion. Appendix B reviews adversarial attack defences and their impact on image distortions. In Appendix C, we present visual results, and Appendix D shows the details of the loss function of the DINO and YOLOv7 models.

1.5 Publications

This thesis has led to the following publications.

- *Mark Ofori-Oduro and Maria Amer*, **Defending Object Detection Models against Image Distortions** at *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, Waikoloa, Hawaii. (*WACV* is a leading computer vision conference). This publication covers the GSES method from Section 3.2.

- *Mark Ofori-Oduro and Maria Amer*, **Artificial Immune Systems for data augmentation** *Image and Vision Computing (IMAVIS)*, 2024. (*IMAVIS* is a leading Computer vision Journal). This manuscript covers the AISbod method from Section 3.1

- *Mark Ofori-Oduro and Maria Amer*, **Data Augmentation Using Artificial Immune Systems For Noise-Robust CNN Models** *IEEE* 27^{th} *International Conference on Image Processing (ICIP)*, October 2020, Abu Dhabi, UAE. (*ICIP* is a leading

signal processing conference). This publication covers an initial work of the AISbod method from Section 3.1.

The code of our AISbod method is available at `https://github.com/moforio/AISbod` and that of GSES at `https://github.com/moforio/GSES/`.

Chapter 2

Literature review

The main objective of our research is defence techniques to enhance the accuracy of object detection models under image distortions. However, we also aim to verify if our defence techniques are generalized to other computer vision tasks, including image classification and object tracking. This Chapter reviews the first state of the arts in image classification, object detection, and object tracking (see Section 2.1). Additionally, it examines defence techniques in Section 2.2, focusing on methods designed to protect computer vision models from image distortions, such as adversarial training and robust architectural modifications. Finally, the review highlights the differences between existing defences and the novel methods proposed in this thesis in Section 2.3.

2.1 Computer vision baseline models

2.1.1 Image classification

Image classification is the task of categorizing images into one of several predefined classes. It is one of the core tasks in computer vision and serves as a foundation for more complex tasks. Image classification models take an image as input and output a class label or a probability distribution over a set of classes.

2.1.1.1 CNN-based classifiers

LeNet-5 [21], designed by LeCun *et al.*, is recognized as one of the pioneering CNNs specifically developed for digit recognition. Its architecture and principles set the stage for future advancements in the field. Following LeNet-5, a breakthrough in CNN architectures was achieved with the introduction of AlexNet [22], as described by Krizhevsky *et al.* This model gained prominence through its remarkable performance in the ImageNet challenge,

significantly outperforming other models. The deep structure of AlexNet and its innovative use of ReLU non-linearity were central to its success.

Building upon these developments, Simonyan and Zisserman [23] from the Visual Geometry Group of Oxford University introduced the VGG models, particularly VGG-16 and VGG-19. These models demonstrated the critical role of depth in CNNs for enhancing classification accuracy. However, a more substantial milestone was achieved with residual learning, as proposed by He *et al.* [24]. This concept enabled the construction of even deeper networks like ResNet-50 and ResNet-101 while addressing the vanishing gradient problem commonly encountered in deep CNN models.

Recently, EfficientNet [25] and SpinalNet [18] have emerged as significant innovations. EfficientNet, developed by Tan and Le, introduced a systematic method for scaling CNNs, achieving unparalleled accuracy and efficiency with smaller model sizes. On the other hand, SpinalNet represents a novel architectural approach in CNN design aimed at augmenting traditional deep learning models. Inspired by the human spinal cord, its architecture processes information through multiple levels before it reaches the brain, offering state-of-the-art classification results. This innovative design alters the conventional flow of information in the network, enhancing performance and efficiency.

2.1.1.2 Transformer-based classifiers

Dosovitskiy *et al.* [26] was the first to adapt transformers for images with the introduction of the ViT. In ViT, image patches are treated as sequences of tokens, akin to words in NLP, representing a paradigm shift in how images are processed for classification. ViT demonstrated that transformers could adapt to the realm of image classification and achieve state-of-the-art performance, challenging the supremacy of CNN-based approaches; Touvron *et al.* [27] built upon the foundation of ViT by proposing DeiT (Data-efficient Image Transformers). DeiT focused on making the transformer model more data-efficient; The Swin Transformer, introduced by Liu *et al.* [28], marked another significant leap in this domain. It presented a hierarchical structure incorporating shifted windows, a novel approach that allowed for more efficient handling of varying image sizes and resolutions. The Swin Transformer showcased promising performance on various image classification datasets, further cementing the role of transformer-based models in the field.

2.1.2 Object detection

The object detection task goes beyond image classification by identifying multiple objects within an image and determining their locations. This task is achieved by providing a

bounding box around each object of interest.

2.1.2.1 CNN-based detectors

The field of CNN-based image object detection has witnessed significant evolution, transitioning from the complexity of two-stage detectors [29–31] to the efficiency of one-stage detectors [3,4,32–37]. This evolution marks a pivotal shift in the object detection landscape, reflecting advancements in algorithmic efficiency and accuracy.

The Faster R-CNN, introduced by Ren *et al.* [29], represents a significant leap forward from its predecessor, Fast RCNN. Its major innovation lies in replacing the selective search algorithm with a Region Proposal Network (RPN), enhancing speed and recall. The RPN, a lightweight network, comprises an intermediate layer with 256 or 512 3×3 filters (depending on the ConvNet's output channels) and a dual-output layer for classification and regression using 1×1 filters. Following these footsteps, R-FCN [30] mirrors Faster R-CNN's architecture but differs from a fully convolutional RPN, achieving comparable accuracy at higher inference speeds.

Mask R-CNN [31] builds upon the architecture of Faster R-CNN by introducing an additional branch for pixel-level object classification alongside the existing classification and regression branches. This enhancement enables more precise object detection, especially in complex visual environments.

Redmon *et al.*'s introduction of YOLO [32] in 2016 marked a watershed moment in real-time object detection, paving the way for one-stage detectors. YOLO's novelty lies in its holistic approach to detection. It divides the feature map into 7×7 grids, using two bounding boxes per grid to predict if an object's centre falls within a specific grid. This simultaneous prediction across all grids equips YOLO with a global image perspective, hence its acronym: **Y**ou **O**nly **L**ook **O**nce. However, its spatial constraints and limited bounding boxes per grid can hinder performance, particularly with small, clustered objects.

Building on YOLO's unified detection approach, Liu *et al.*'s Single Shot Detector (SSD) [33] stands out for its balance of speed and accuracy. SSD enhances accuracy by employing six bounding boxes at multiple feature map stages, leading to more precise detections.

RetinaNet, another one-stage model, leverages the ResNet architecture to match the accuracy of leading two-stage models like Mask and Faster R-CNN while offering superior speed. Lin *et al.* [34] identified the class imbalance between foreground and background as a critical challenge in one-stage detectors. To address this, they introduced the focal loss function, modifying the standard cross entropy loss to prioritize more challenging examples during training, thereby improving model performance.

Emerging as a noteworthy competitor in this field is the EfficientDet model [35], which

has made its mark by integrating a compound scaling method. This unique approach scales the network width, depth, and resolution in a balanced manner, leading to a model that is not only more efficient but also highly accurate. EfficientDet's architecture, based on the EfficientNet backbone and a novel bi-directional feature pyramid network, allows for efficient and effective feature integration at various scales.

In recent developments, the YOLO model has undergone a series of transformative iterations, each marked by substantial capabilities enhancements. These continual refinements and groundbreaking innovations have firmly established the YOLO series as a frontrunner for real-time object detection technology. Notably, versions such as YOLOv3 [36] and YOLOv4 [4] have improved upon the original YOLO's limitations. YOLOv3, for instance, uses three different scales to detect small to large objects more effectively, and YOLOv4 further optimizes speed and accuracy, making it well-suited for real-time applications. YOLOv7 [3] represents another significant advancement, with refined architecture for better detection of small and challenging objects and improved efficiency for deployment on diverse hardware. The most recent iteration, YOLOv8 [37], builds upon these advancements, further pushing the boundaries of detection accuracy.

2.1.2.2 Transformer-based detectors

Carion *et al.*'s DETR [38] laid the groundwork for using transformers in object detection. As a pioneering model, DETR capitalized on the transformer encoder-decoder framework to revolutionize the detection process. This model's most notable contribution was its streamlined approach, eliminating the need for traditional components like non-maximum suppression. By doing so, DETR showcased the innate capability of transformers to directly predict object locations and classes, setting a new standard in object detection.

Progressing from this foundation, the Deformable DETR [39] represented a significant advancement, directly addressing some of DETR's limitations. This model introduced deformable attention modules, enhancing the efficiency and accuracy of the detection process. The key innovation here was the model's ability to focus on specific sampling points, thereby improving convergence speed. This refinement indicated a growing sophistication in transformer-based object detection as the models became more adept at handling intricate detection tasks.

The evolution reached a new height with the introduction of DINO [2]. This model marked a substantial leap in the field, particularly in high-resolution object detection. DINO's dual-branch transformer architecture, which integrated both global and local features, was a testament to the versatility and adaptability of transformers. This model excelled in detecting objects of varying sizes, with particular strength in identifying smaller

objects. Such capabilities underscored the potential of transformers to manage complex detection scenarios effectively.

2.1.3 Object tracking

The object tracking task refers to locating and following a specific object or multiple objects within a sequence of frames in a video or a sequence of images. This task has seen significant advancements in recent years with the introduction of several novel approaches.

2.1.3.1 CNN-based trackers

The shift toward end-to-end trainable computer vision systems presented a challenge in visual tracking, which requires real-time learning of a target-specific appearance model during inference. This challenge led to the development of Siamese network trackers [40, 41]. However, these trackers predict a target feature template while neglecting background appearance details, which leads to limited target-background distinction.

Bhat *et al.* [42] introduced an innovative end-to-end tracking architecture that utilizes target and background appearance information to enhance model prediction accuracy. Their approach, based on a discriminative learning loss, enables the rapid generation of a robust model in just a few iterations, and it even learns crucial aspects of the discriminative loss function.

Anchor-based Siamese trackers are hindered by lagging tracking robustness. The limitation arises from the regression network in these methods, which are trained solely on positive anchor boxes, making it challenging to refine anchors with minimal overlap with target objects. In response, Zhang *et al.* [43] introduce an innovative network that predicts target object positions and scales directly without relying on reference anchor boxes. By thoroughly training each pixel within ground truth boxes, their tracker corrects imprecise predictions during inference. Additionally, they integrate a feature alignment module to derive object-aware features from predicted bounding boxes, enhancing the classification of both target objects and background.

2.1.3.2 Transformer-based trackers

Ye *et al.* introduced OSTrack [44], which made a significant stride by integrating the Masked Autoencoder (MAE) pre-trained ViT backbone into object tracking. OSTrack strategically employed early candidate elimination to discard extraneous background information, thereby amplifying tracking speed and safeguarding the attention module against contamination from background token interactions. However, Wu *et al.* highlighted a residual challenge

in utilizing MAE pre-training, revealing its limitations in ensuring robustness for frame-by-frame matching tasks between images. In response, they proposed DropTrack [45], a variant of ViT that dynamically applies spatial-attention dropout during MAE pre-training. This adaptation aimed to disrupt within-frame spatial cues' co-adaptation and bolster between-frame attention, effectively capturing temporal cues.

Chen *et al.* introduced SeqTrack [46], treating tracking as a sequence learning problem. SeqTrack innovatively utilized Start and End tokens reminiscent of language models, seamlessly integrating dynamic template update and window penalty algorithms into its framework without imposing additional hyperparameters on the model. In a parallel effort, Wei *et al.* presented ARTrack [47], which reimagined object tracking as an auto-regressive coordinate sequence interpretation. Using ViT as the pre-trained backbone, ARTrack adopted a unified encoder-decoder structure to directly forecast the target object's bounding box. This unique architecture omitted a prediction head, resulting in a simplified loss function.

2.2 Defense techniques against distortions

As shown in Figure 2.1, we group related defence works into AIS methods [48–50], data augmentation methods [51–64] and enhancer-based defences [11,65–75]. The main drawbacks of these related methods are their poor performance on clean samples and generalization to unknown distortions [13–16].

2.2.1 Artificial immune systems

One of our proposed defence methods uses Artificial Immune Systems (AIS) to generate new samples. In this Section, we present the background of AIS and its use in image processing and computer vision.

2.2.1.1 Background

AIS are computational systems inspired by the human immune system to solve engineering and scientific tasks. The immune system protects the body from pathogens by recognizing and responding to foreign antigens. The concept of AIS was first introduced in the late 1980s. Since then, researchers have applied several developed models of AIS to various fields, such as pattern recognition, optimization, and machine learning. Of all the models developed, the Clonal Selection Algorithm [1], Artificial Immune Network [76], and Negative Selection Algorithm [77] are the most researched and applied models. We present a brief overview of these models below.

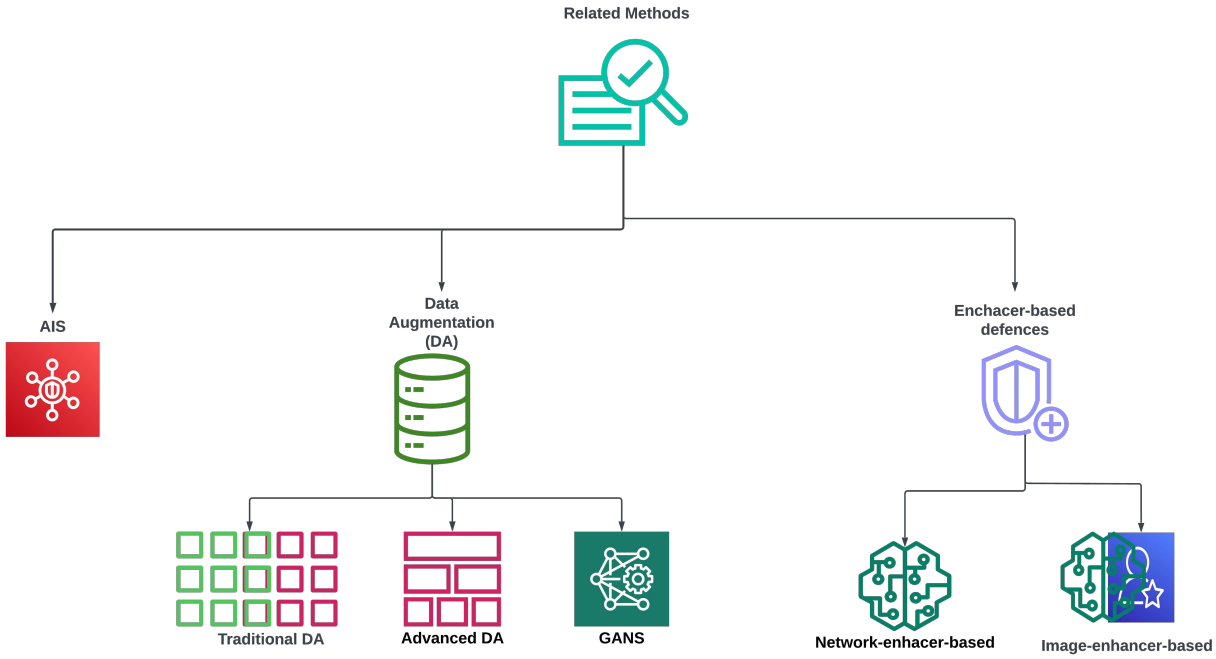

Figure 2.1: Overview of defence methods against distortions.

Clonal Selection Algorithm (CLONALG): The Clonal Selection Algorithm (CLONALG) is a well-known artificial immune system (AIS) algorithm developed by de Castro and Von Zuben in 2000. CLONALG is inspired by the natural immune system, particularly the generation and activities of B-cells, which are part of a class of cells called antigen-presenting cells. These B-cells patrol the human body for pathogens when danger signals are triggered by white blood cells.

As shown in Figure 2.2, upon encountering a pathogen, B-cells learn the pattern of the pathogen to ascertain if it is similar to past encounters. If the binding region of a B-cell fits the pathogen, it binds to the pathogen and presents it through a major histocompatibility complex to be nullified by the killer T-cells (as indicated by I). If the fit is not good enough, the B-cells mature to increase their binding affinity.

The B-cell binding process in CLONALG consists of the following steps:

1. **Selection (II)**: A population of candidate solutions (antibodies) is generated randomly. The fittest solutions (those with the highest binding affinity) are selected for cloning.
2. **Clone (III)**: The selected B-cells are cloned to generate a new population. The number of clones produced depends on the fitness of the original B-cell; the more fit the B-cell, the more it is cloned. This ensures that the system can respond robustly

to the pathogen.

3. **Mutation (IV)**: The cloned B-cells undergo mutation to introduce diversity into the population. These mutations help create variations in the binding regions, some of which may have a better fit to the pathogen. This process mimics the natural immune response, where slight changes can improve the overall effectiveness of the B-cells.
4. **Selection (V)**: The mutated clones are re-evaluated to determine their fitness. The clones with the best binding affinity are selected to form the new population for the next iteration. This step ensures that the most effective B-cells are continuously refined and improved, leading to a stronger immune response.

Researchers have successfully applied CLONALG to prediction, optimization, and pattern recognition problems in various fields. For instance, researchers have applied CLONALG to solve pattern recognition problems in image processing in engineering [1]. In bioinformatics, they have used it to solve problems in DNA sequence analysis, gene expression analysis, and protein structure prediction [78]. Additionally, researchers have applied CLONALG in financial optimization, such as portfolio optimization and option pricing [79, 80]. This versatility makes the algorithm applicable to different tasks by simply modifying and adapting it to the application of choice.

Artificial Immune Networks (AINs): Similar to CLONALG, are modelled off B-cells but as a unit, [76]. These modelled cells solve many problems, including image recognition, classification, clustering, and optimization [81, 82]. Moreover, the cells act like nodes that interact with each other and the environment to learn and adapt to changes in the problem domain.

AINs have two main components [76], clonal Selection and immune network components. The clonal selection component generates a population of candidate solutions (antibodies) and selects the fittest solutions to create a new population in the next iteration. The immune network component represents the antibody interactions, modelled as network nodes. The network topology and connectivity capture the diversity of the antibody population and promote the exploration of the search space. However, one major drawback is selecting the suitable network topology to align with the CLONALG component [76].

Negative Selection Algorithm (NSA): This is a computational algorithm that imitates the process of maturing T-cells in the thymus of vertebrates [77]. First, the varying diversity of the cells undergoes a self-non-self test before the T-cells are dispatched to different locations in the body. Then, T-cells that recognize self-cells are eliminated, and the

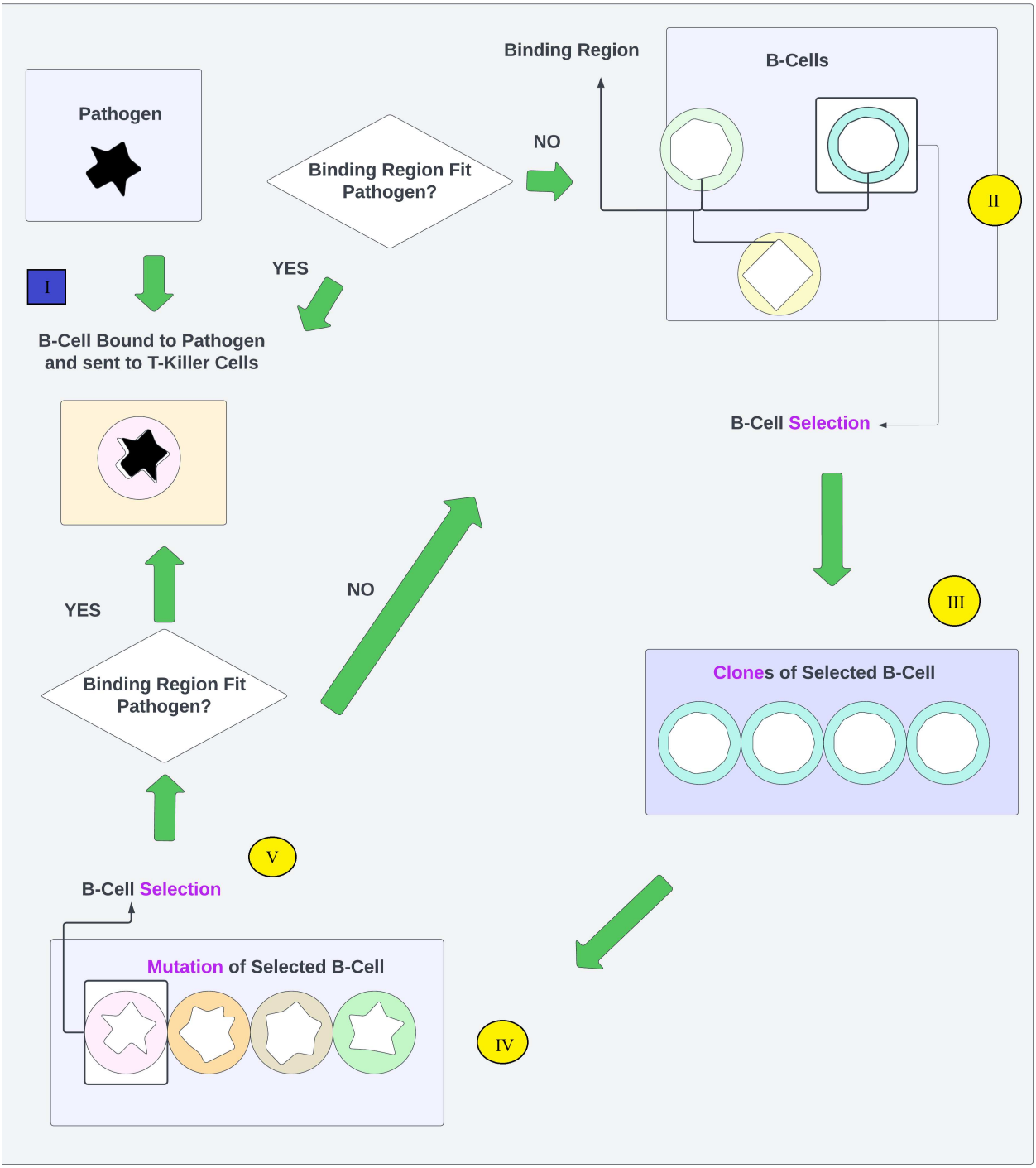

Figure 2.2: A simplified overview of CLONALG.

remaining T-cells are deployed outside the thymus. NSA implementation consists of two primary stages: detector generation and detection. Variant versions of NSA exist for different applications. However, they all employ a negative representation of information (i.e., output the complementary concept of the actual target concept), use some form of detector set as

a detection mechanism, and implement only one class classification (i.e., self or non-self).

NSA's popularity in computer security applications is due to its ability to adapt to changing environments and its robustness against noise and false positives. In addition, it effectively detects a wide range of anomalies, including network attacks, system intrusions, and software exploits [83, 84]. However, one of the key challenges in applying the NSA is the Selection of appropriate parameters, such as the size of the self and non-self sets and the threshold for anomaly detection [77].

2.2.1.2 AIS in computer vision

There is limited research utilizing AIS in computer vision tasks. Gabriele *et al.* [48] use adaptive AIS for the early detection of breast cancer in screening-digital mammography. Achmad *et al.* [49] present a way to improve the classification accuracy of Artificial Immune Recognition System 2 (AIRS2) on the Wisconsin breast cancer dataset; the 'best' features are first selected using a fast correlation-based filter, then used to train AIRS2. Finally, in [50], Bojarczuk *et al.* present a deep AIS to detect weld defects in petroleum pipes; their method is a hybrid of both AIS (for classification) and deep neural networks (for feature extraction).

2.2.2 Data augmentation

Data augmentation involves creating new training data from existing datasets by applying transformations and modifications such as rotations, translations, flips, scaling, and colour adjustments. This technique is essential for enhancing the diversity of the training data, reducing the risk of overfitting, improving the model's ability to generalize, and regularizing model weights to improve performance [51]. The generation of new samples can occur either during training (online) or before training (offline). Comparatively, online methods result in longer training times and must be adjusted to be compatible with different models when data augmentation is applied.

Various techniques have been developed to effectively utilize data augmentation, catering to different tasks and challenges in computer vision. These techniques can be broadly classified into classical methods, advanced methods and those involving Generative Adversarial Networks (GANs).

2.2.2.1 Classical data augmentation methods

In the image classification domain, well-known data augmentation methods include CutMix [51], Random Erasing [52], and ISDA [53]. CutMix, proposed by Yun *et al.* [51], blends

parts of training images and their corresponding labels online, causing the model to learn local features and improve generalization. Random Erasing, introduced by Zhong *et al.* [52], randomly occludes parts of the input images online, improving a model's resilience to background clutter and partial occlusions. ISDA, proposed by Wang *et al.* [53], operates in the feature space of deep networks, adaptively learning data augmentation policies during training, thereby enhancing feature representations and improving generalization.

For object detection, state-of-the-art data augmentation methods include Mosaic [54] and GridMask [55]. Mosaic combines four images into a mosaic, providing diverse training samples to improve the model's performance. GridMask, introduced by Chen *et al.* [55], overlays a grid-like mask on input images online, promoting spatial diversity and encouraging the model to extract features effectively from various regions.

In the context of object tracking, Deng *et al.* [56] introduce a block-erasing data augmentation strategy designed to enhance the robustness of object trackers under object occlusion. In [57], Xu *et al.* propose continuous copy-paste, which fully exploits the pixel-wise annotations provided by multi-object tracking and segmentation datasets to actively increase both the number of instances and the unique instance IDs in training samples without requiring modifications to tracking model frameworks. Furthering this line of research, Cheng *et al.* [58] present DeepMix, which utilizes embeddings from historical samples to generate augmented embeddings online.

2.2.2.2 Advanced data augmentation methods

The SmoothMix method [59] blends two images to create a new sample online for augmentation; the authors show that it makes image classification models robust to 15 image distortions. Rusak *et al.* [60] propose augmenting samples distorted with Gaussian or speckle noise offline helps make classification models robust. In [61], Von *et al.* propose a method to make object detection models used in autonomous vehicles robust to heavy rain conditions; the Det-AdvProp method [62] makes the object detection model EfficientDet [35] robust under 15 image distortions by generating augmenting samples online by adding a weighted signed gradient of the classification and location loss function to the original training samples. Michaelis *et al.* [63] provide a benchmark to measure the robustness of object detection models (such as Mask R-CNN [31] and RetinaNet [34]) to 15 image distortions; they also show that applying the style transfer using stylize model [85] (superpose a style of an image on another image) offline as a data augmentation technique can improve the robustness of the detection models above on the provided benchmark. Beghdadi *et al.* [64] introduce another benchmark for detection models with ten image distortions, 7 of which overlap with the set of 15 distortions in Michaelis*et al.*'s benchmark.

2.2.2.3 Generative adversarial networks

Generative Adversarial Networks (GANs), introduced by Goodfellow *et al.* [86], consist of two neural networks—the generator and the discriminator—trained together. The generator tries to produce data from noise, while the discriminator aims to differentiate between accurate and generated data. GANs have significantly advanced in learning complete data generation models from unlabeled data, which can be used to augment labelled training sets [87].

GANs have been applied in various domains for data augmentation. In medical imaging, where data is scarce due to privacy concerns and acquisition costs, GANs have been used to improve model performance. Han *et al.* [88] demonstrated improved CNN performance for lesion detection in medical images using GAN-generated data. Kossen *et al.* [89] used GANs for cerebrovascular segmentation, showing their effectiveness in this domain.

Class imbalance in datasets poses challenges in training vision models, and GANs have been instrumental in generating synthetic data for minority classes. Techniques like the Synthetic Minority Over-sampling Technique (SMOTE), introduced by Chawla *et al.* [90], laid the groundwork for GAN-based augmentation methods. Bekkar *et al.* [91] found that achieving class balance through synthetic data can optimize classifier performance.

2.2.3 Enhancer-based defence methods

Defence methods in this group can be categorized into network-enhancer-based [11,67–69] or image-enhancer-based [70–75]. Network-enhancer-based techniques typically require modifying the architecture of the base CNN model or adding complementary blocks to it. Image-enhancer-based methods improve or denoise corrupted images before being fed into models.

2.2.3.1 Network-enhancer-based

Models [67] and [68] rectify the batch normalization of distorted samples used to train classification models to improve their robustness on distorted samples without a significant drop in accuracy on clean samples. The work by Borkar and Karam [11] tested the susceptibility of image classification models (such as AlexNet) under Gaussian noise and image blur. They rectify the susceptibility issue by finding, modifying, and retraining the most vulnerable convolutional filters in the models. The rest of the pre-trained filters deemed robust to distortion are left unchanged. In [69], the authors improve the accuracy of clean and noisy samples of compressed models by utilizing a compression-aware loss function during training that encourages the network to learn accurate and robust features of the compression process.

2.2.3.2 Image-enhancer-based

In [70], Dapello *et al.* propose VoneNets, a hybrid vision model for image classification. This vision model class comprises VoneBlock, a more distortion-robust model inspired by the primate primary visual cortex on the front end and a conventional neural network on the back end. The image-enhancer-based defences RetinexFormer [71], UFormer [72], URIE [73], OWAN [74], and DD [75] propose front-end models to enhance or denoise corrupted images before being fed to classification or detection models. Hence, the accuracy of the vision models on distorted images is improved.

2.3 Differences of proposed approach to related works

Our approach to defending against distortions is a data augmentation approach applied before training (in offline mode). It does not modify models' architecture as in network-enhancer-based methods such as [11]. Unlike the image-enhancer-based defense methods RetinexFormer [71], UFormer [72], URIE [73], OWAN [74], and DD [75], we do not enhance or denoise distorted inputs.

Often, data augmentation methods struggle with a bias to augmented samples [92], which leads to a reduction in generalization (e.g., in Random Erasing [52], Mosaic [54], and Stylize [63]). Some methods cannot use the original labels, nor do they balance class distributions (as in SmoothMix [59], CutMix [51], and Mosaic [54]), which leads to a reduction in overall performance. Our approach mitigates these challenges, as explained next: Our methods use the original image label and generate a corresponding distorted image, avoiding the need for new labels. Moreover, our methods generate one new sample from one original sample, avoiding the creation of new samples vulnerable to a class imbalance that occurs when merging multiple original samples randomly.

Our methods preserve the quality of the generated sample by distorting a portion of the original pixels instead of completely changing the style of the original image, as in Stylize [63] and or distorting all pixels, as in Det-AdvProp [62]. Unlike in Stylize [63], our distortion techniques seem to reduce over-fitting, which leads to consistent performance on small and large datasets. Unlike in Stylize [63], we require no training of our methods or a stylize dataset. We have tested our methods on four distinct object detection architectures: DINO, which uses 5scale-R50; YOLOv7, which employs E-ELAN; YOLOv4, which utilizes CSPDarknet; and Faster RCNN, which employs ResNet. This contrasts with Det-AdvProp [62] that only tests on EfficientDet with EfficientNet.

2.4 Summary

This Chapter reviewed image classification, object detection, and object tracking models. We observed that state-of-the-art models primarily employ CNNs, though transformer-based models show promise as viable alternatives. Additionally, we explored techniques that enhance model resilience against image distortions, which are relevant to the scope of this thesis. To conclude the Chapter, we differentiated between our methods presented in the thesis and their related techniques

Chapter 3

Proposed methods

Our goal is to enhance the accuracy of object detection models in the presence of image distortions. We aim to achieve this by augmenting the training data with distorted samples while keeping the models' architecture and inference speed unchanged. We do not simply add noise (or distortion) to the training sample. As shown in [13, 15, 16, 93], adding noise has two main drawbacks: poor performance on clean samples and lack of generalization to unknown distortions. Our approach mitigates these drawbacks by how we distort, the number of pixels we distort, and the number of distorted samples we use for augmentation. Based on this approach, we propose two methods.

- In Section 3.1, our first method, **AISbod**, we generate new samples (antibodies) by following the AIS cycle.
- In Section 3.2, our second method, **GSES**, generates new samples by distorting a set of pixels through selection, estimation, and sampling.

Results show that GSES has higher accuracy improvement than AISbod. This superior performance of GSES is due to its more diverse samples, which benefits the baseline detection models.

3.1 AISbod: artificial immune systems for data augmentation

3.1.1 Overview of proposed AISbod

In AIS, antigens are original (clean) samples; antibodies are AIS-distorted versions of the antigens; affinity measures the similarity between an antibody and an antigen. An AIS cycle consists of four stages: *select*, which selects some antigens from all the input antigens;

clone, which makes copies of the antibodies following some conditions; *mutate*, which changes (distorts) the antibodies at random locations; *select*, which selects antibodies from the so far generated antibodies. In our approach, antibodies are images mutated at the pixel level; the augmentation ratio p is the fraction of antigens selected in the first *select* stage, and the affinity threshold τ determines the maximum number of pixel matches allowed between a selected antigen and an antibody. The steps to generate the augmented (distorted) samples are outlined in Fig. 3.1:

1. Inputs: antigens Ag^*, augmentation ratio p, and affinity threshold τ.
2. Select some antigens using the augmentation ratio p.
3. Initialize a set of antibodies Ab from the selected antigens.
4. Generate antibodies from the initialized antibodies using the AIS cycle (*select*, *clone*, *mutate*, and *select*) for each antigen in the selected antigens.
5. Check if the antibody meets each cycle's affinity threshold τ. If not, continue the cycle.

Our objective is to improve the accuracy of detection and classification under image distortions by augmenting their training data with new samples (antibodies) having some similarities to the original samples (antigens) without altering the models' architecture, inference speed, or performance under clear (original) samples. Our approach in Algorithm 1 can be divided into the following steps: data pre-processing, initialization, AIS-sample generation, and data post-processing.

3.1.2 Data pre-processing

We pre-process the antigens Ag for augmentation using three steps: padding, selection, and quantization. Assuming there are K^* input antigens Ag^* and L total number of pixels in the antigen with the highest resolution in all 3 RGB channels. (e.g., in the training set of PASCAL 2007 and 2012 [94], $K^* = 16,551$ RGB image samples and $L = 75000 = 500 \times 500 \times 3$). To form Ag^*, as shown in Fig. 3.2, we first unroll each input antigen and vertically stack them together. Since each antigen may have a different resolution, we equalize the resolution of all antigens by padding the lower-resolution antigens with zeros. This zero-padding facilitates the implementation of parallel antibody generation. We generate antibodies for only a fraction p of the total (input) K^* antigens, where $0 < p \leq 1$; augmenting with too many antibodies (e.g., $p = 0.75$) tends to significantly reduce the accuracy of the object detection models under distortion-free samples. We examined two ways to select the $K = p \cdot K^*$ antigens Ag: randomly from Ag^* or by the balanced representation of all classes from the original Ag^*. As shown in Section 5.2, the two approaches are similar regarding accuracy since the class distribution of the datasets we use (for instance, COCO) appears

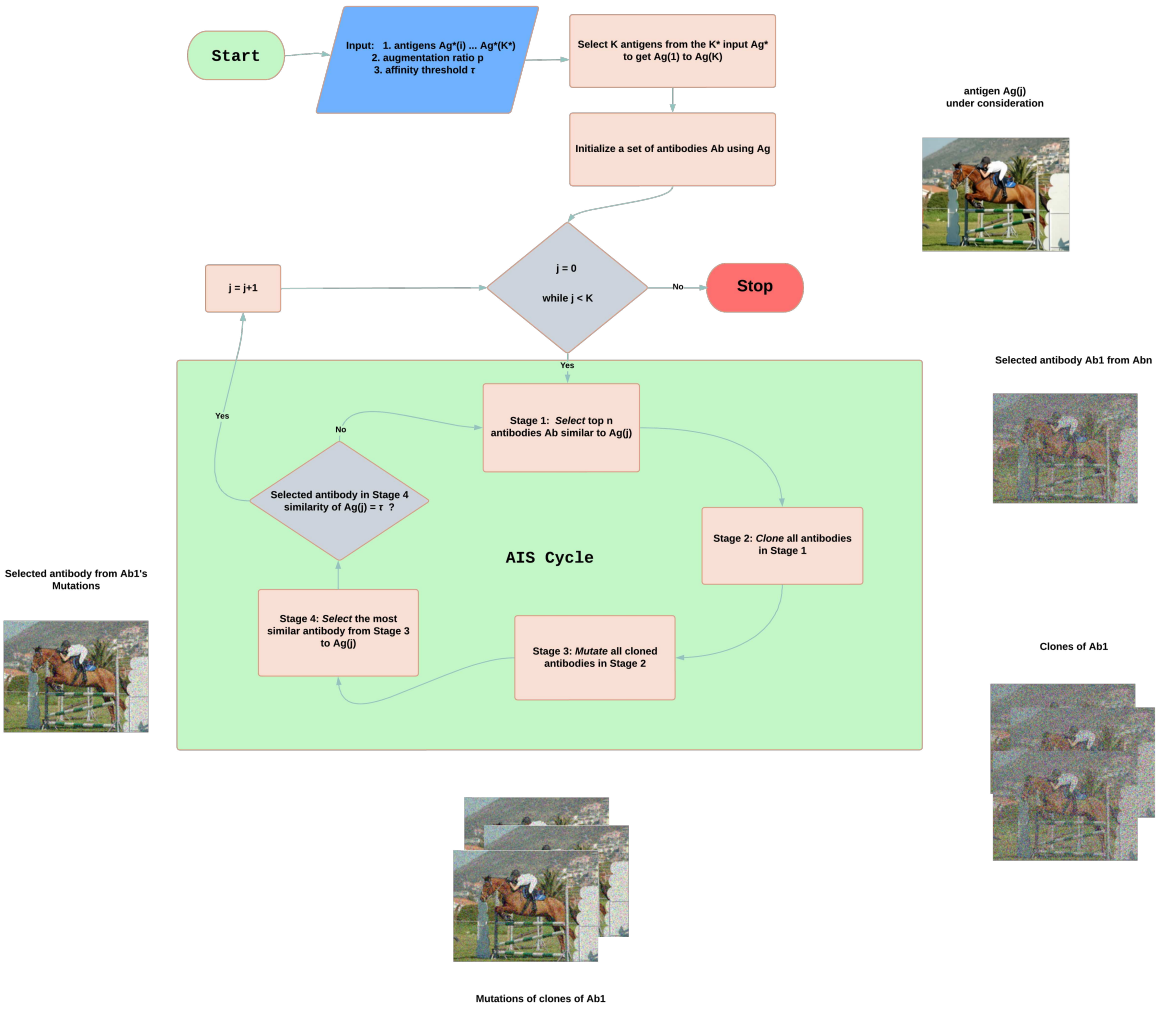

Figure 3.1: A simplified overview AISbod.

uniform, as shown in Figure 3.3. We quantize the 256 image intensities to 16 levels in each K antigens Ag to speed up antibody generation. We show in Section 5.2 that this quantization has a minor effect on the accuracy of our method, but it reduced the computations by 35%.

3.1.3 Initialization

We first uniformly initialized the pixels of the memory antibodies Ab_m and response antibodies Ab_r from the 16 quantization values $0, 16, 32, \cdots, 240$ and antigens Ag. We then set antibodies Ab as a vertical stack of Ab_m and Ab_r. The Ab_m is a size $J_m \times L$ matrix, with $J_m = q^* \cdot K$ and q^* integer. There is a q^* antibody for a corresponding antigen in Ag. The Ab_r is a $J_r \times L$ matrix, where J_r is application dependent, we set $J_r = J_m$. The Ab_m stores

Algorithm 1: Proposed data augmentation.

Input: Ag^* antigens (original samples), p augmentation ratio, τ affinity threshold
Output: Ab_m antibodies and antigens Ag^*
Data pre-processing:
L = Resolution of largest Ag^*;
$Ag^* \longleftarrow$ Unroll and zero-pad{all K^* input Ag^*};
$Ag \longleftarrow$ Select{$K = p \cdot K^*$ from Ag^*};
$Ag \longleftarrow$ Quantize{each antigen $Ag(k)$};
Initialization:
Set:
$n = 5, N = 50, \alpha = 1, s = 10000, q^* = 1$;
$\{Ab_m, Ab_r\} \longleftarrow$ Initialize{$U(0, 240), Ag$};
$Ab \longleftarrow$ Stack{Ab_m, Ab_r};
AIS-sample generation:
for *each antigen* $Ag(k)$ **do**
  $D_{kk} = 0$;
  while $round(\frac{D_{kk}}{L}, 2) \neq round(\tau, 2)$ **do**
    $Ab_n \longleftarrow$ Select{n antibodies from $Ab(j)$, $j = 1 \cdots J$, with highest affinity D_{jk}};
    $Ab_Q \longleftarrow$ Clone{each antibody in Ab_n, Q times, as in Eq. (2)};
    $Ab_{Q^*} \longleftarrow$ Mutate{each antibody in Ab_Q};
    $Ab_{q^*} \longleftarrow$ Select{q^* antibodies from Ab_{Q^*} with highest affinity D_{ck}, $c = 1 \cdots C$, as in Eq. (3)};
    Update{$Ab_m : Ab_m(k) \longleftarrow Ab_{q^*}$};
    Update{$Ab_r : Ab_r \longleftarrow U(0, 240)$};
    $D_{kk} =$ Affinity{$Ab_m(k), Ag(k)$} ;
  end
end
Data post-processing:
$Ab_m \longleftarrow$ Un-pad zeros and reshape{Ab_m};
Output $\longleftarrow$ Shuffle{original antigens Ag^* and antibodies Ab_m};

the best (highest affinity) antibody for an antigen in Ag, and the Ab_r serves the purpose of initialization to help our method search a more expansive space for a better antibody (i.e., an antibody with higher affinity than the one saved in Ab_m). The Ab can be considered as a matrix of size $J \times L$, where each row j is an antibody (AIS image of size L), and the total number of antibodies, J (i.e., $J = J_m + J_r$) is greater than the total number of antigens K.

Since $J_m = q^* \cdot K$ and we set $q^* = 1$, then with $J_r = 4 \cdot J_m$, we get $J = 5 \cdot K$; thus, Ab_m is a K-length array of antibodies. Note that we (uniform randomly) reinitialize Ab_r after every iteration (line 14 Algorithm 1), but values in Ab_m only change when there is an antibody with a higher affinity.

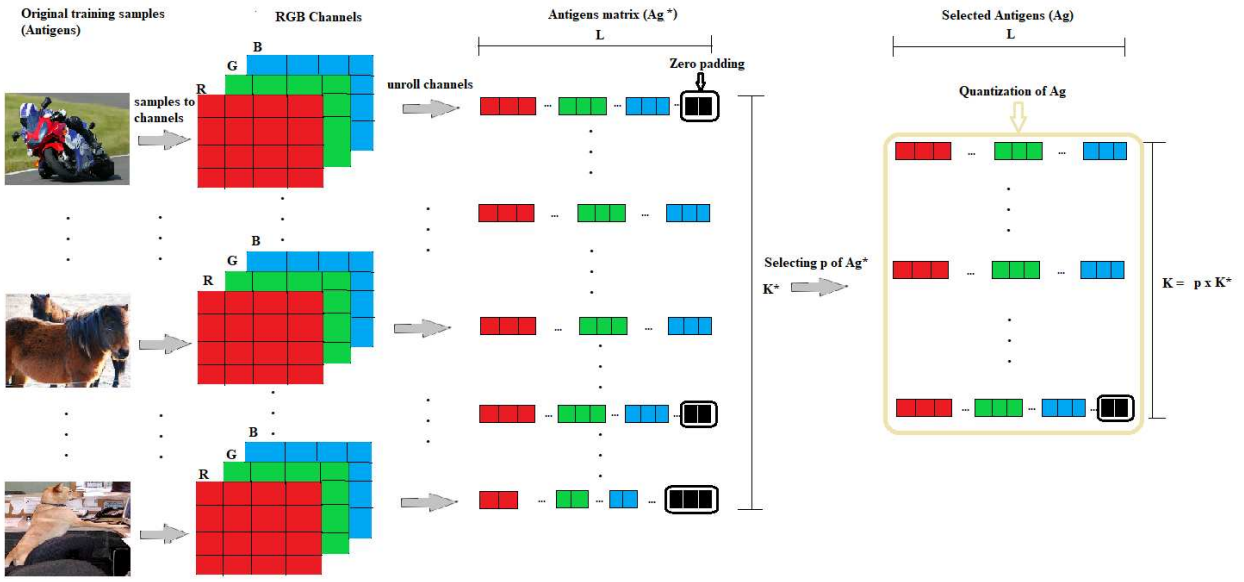

Figure 3.2: Antigen matrix: We unroll the RGB channels of each original training image into one vector and vertically stack the vectors. Red, Green, and Blue indicate the pixels of RGB channels, respectively; Black are the zero padding.

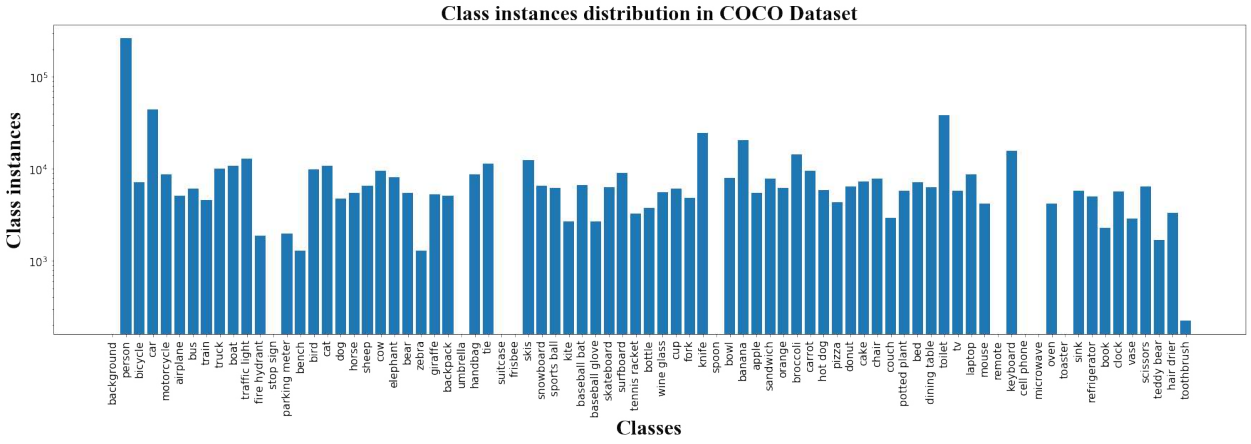

Figure 3.3: Distribution of the 80 object classes in the COCO 2017 dataset.

3.1.4 AIS-sample generation

In the third step of our method (lines 7 to 15 of Algorithm 1), we generate an antibody $Ab(k)$ for each antigen $Ag(k)$ if it has a particular affinity to $Ag(k)$. We define such affinity D_{kk} using a variant of the Hamming distance, as in

$$D_{kk} = \sum_{l=1}^{L} \phi_l; \quad \begin{cases} \phi_l = 1, & \text{if } Ab(k,l) = Ag(k,l) \\ \phi_l = 0, & \text{otherwise.} \end{cases} \tag{1}$$

Our aim is to control the number of pixels that change (mutate) during the antibody generation. We are not interested in how much these pixel values change. As a result, Eqn. 1 is sufficient in determining the affinity between an antigen and the corresponding antibody. To find D_{kk}, we compare each pixel l in $Ab(k)$ to pixel l in $Ag(k)$ at the exact spatial location to check if $Ab(k)$ matches well (high affinity) with $Ag(k)$. D_{kk} is the total pixel matches between $Ab(k)$ and $Ag(k)$. We aim to produce antibodies $Ab_m(k)$ from input antigens $Ag(k)$, $k = 1 \cdots K$, which have affinity ratio τ (rounded to two decimals) equal to affinity threshold $\tau \in [0, 1]$ (rounded to two decimals) for data augmentation purposes; $\tau = 1$ means an antibody $Ab_m(k)$ is (visually) identical to its corresponding antigen $Ag(k)$, and $\tau = 0$ means $Ab_m(k)$ is different at each pixel different and visually indistinguishable from $Ag(k)$. τ should be selected in a way such that the antibodies $Ab_m(k)$ can capture the essential visual features of their corresponding antigens $Ag(k)$ while still representing a distorted version of those antigens $Ag(k)$.

To produce the antibodies $Ab_m(k)$, we iteratively apply the AIS cycle (*select, clone, mutate, select*) on a set of initialized antibodies $Ab(j), j = 1 \cdots J$, until the stopping criteria $round(D_{kk}/L, 2) = round(\tau, 2)$ is reached. Recall that $J = 5 \cdot K$ and Ab consists of memory Ab_m and response Ab_r antibodies. Our AIS cycle, as shown in Fig. 3.4, is based on the clonal selection algorithm (CLONALG) [1]. We explain our cycle in the following and indicate differences and similarities to [1].

For *select*, we, similar to [1], choose the top n antibodies Ab_n from $Ab(j)$, $j = 1 \cdots J$, with the highest affinity D_{kk} for $Ag(k)$; thus, the number of antibodies is reduced from J to n; we set $n = 5$.

We then *clone*, like [1], each of these Ab_n antibodies by ranking them based on their affinity D_{nk} for the antigen $Ag(k)$. Next, we clone the ranked antibodies based on their affinity rank i: an Ab_i is cloned Q times, where

$$Q = \text{round}(\alpha \cdot N/i), \tag{2}$$

N is the cloning constant, and α is the multiplying factor. For example, with $N = 50$, $\alpha = 1$, Ab_1 is cloned 50 times, Ab_2 is cloned 25 times, and so forth; the results are Ab_Q cloned antibodies from the Ab_n. That is, the number of antibodies for $Ag(k)$ increase from n to

$$C = \sum_{i=1}^{n} \text{round}(\alpha \cdot N/i). \tag{3}$$

The next stage is *mutate*; different to [1], we allow all cloned antibodies Ab_Q to mature to become Ab_{Q^*} by randomly selecting s pixels in each cloned antibody in Ab_Q and replacing the pixel values with values randomly selected uniformly from the 16 quantization values.

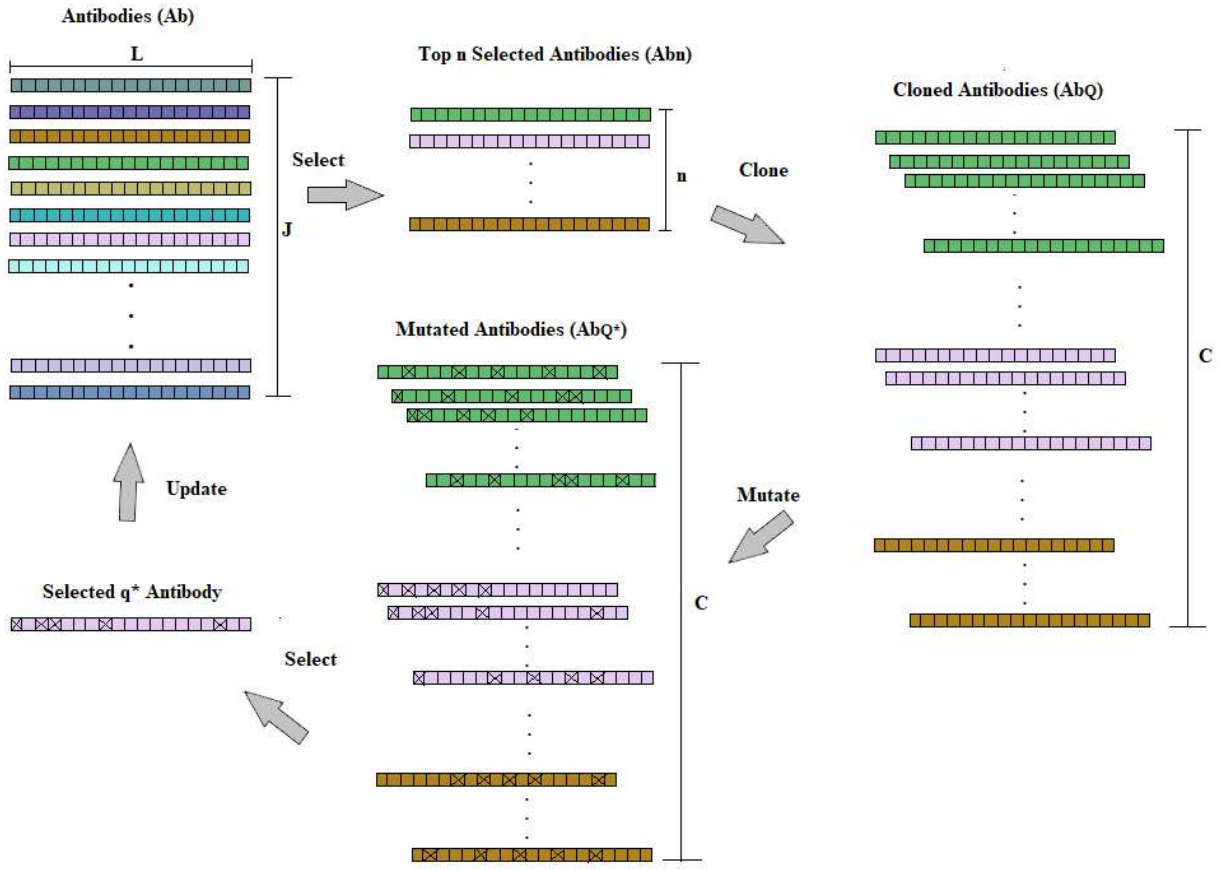

Figure 3.4: AIS cycle: Each unique colour represents a unique antibody, and we use the same colour to depict *clone*s (copies) of the antibody. If a pixel has been *mutate*d (changed), we represent it with an '$\times$' in the antibody.

Unlike in [1], which uses different s for the cloned antibodies, we apply the same s to all the cloned antibodies (since they are all visually similar); we set $s = 10000$. (Note that the number of antibodies in Ab_{Q^*} for $Ag(k)$ remains equal to C.)

We finally *select*, similar to [1], q^* antibody from Ab_{Q^*} with the highest affinity D_{ck} and save in $Ab_m(k)$ for the antigen $Ag(k)$; we set $q^* = 1$; thus, each antigen $Ag(k)$ receive one antibody from $Ab_{Q}*$, that is then saved as an element k in $Ab_m(k)$, which means that Ab_m has one antibody only. Different from [1], we randomly reinitialize Ab_r from the 16 quantization levels. We define the affinity between $Ab_m(k)$ and $Ag(k)$ as D_{kk} of Eq.1 to check for stopping criteria, that is, if the affinity ratio (D_{kk}/L) is equal to τ. Our method converges because of the number of mutation points s, which is set to less than 2% of L, and we always round off the affinity ratio to two decimal places, as shown in line 8 in Algorithm 1; this ensures the affinity of the antibody does not pass the affinity threshold τ selected after each cycle. The number of cycles to generate an antibody depends on the selected τ

and the randomness of our method. It takes an average of 314 cycles to generate antibodies at $\tau = 0.75$ on the PASCAL and COCO datasets.

3.1.5 Data post-processing

Once the cycle (*select*, *clone*, *mutate*, *select*) ends for all $Ag(k)$, we copy the K memory antibodies Ab_m from Ab, but the additional length due to padding the antigens Ag^* with zeros are excluded. We finally add the Ab_m to the original antigens Ag^* and shuffle them (randomly mix) to provide a well-mixed (i.e., antigens and antibodies) training set.

Fig. 3.5 shows samples of antibodies Ab_m generated by our method, where one (row 1, column 1) is displayed at different τ in the first two rows.

At multiple stages, our approach ensures that an object detection model (or image classification) is implicitly trained with diverse features, including distortions, even if not specific, such as noise, blur, artefacts, and weather conditions. First, the select, clone and mutate stages (lines 9 to 11 of Algorithm 1) ensure that distortions of pixels are done randomly. Second, the specified affinity τ ensures maintaining a balance between antigens ("clean pixels") and antibodies ("distorted pixels"); for example, if $\tau = 0.75$, then 75% of the pixels in an antibody are clean (unchanged) pixels; τ enhances the model's capacity to learn robust features. Lastly, regulated by p, we ensure a model is not overloaded with the antibodies. This precaution helps prevent the model from over-fitting the antibodies.

3.2 GSES: defending object detection models against image distortions

3.2.1 Background of KDE

Kernel density estimation (KDE) is a prominent method for estimating the probability density function of data points [95–97]. Illustrated in Fig. 3.6, density estimation involves the reconstruction of an approximate probability density function $\Psi_S(x)$ that underlies a given dataset $x_1, x_2, ..., x_S$ by employing S kernels, denoted as $\Phi_1(\cdot), \Phi_2(\cdot), ..., \Phi_S(\cdot)$. The probability density function is defined as

$$\Psi_S(x) = \frac{1}{S.w} \sum_{s=1}^{S} \Phi_s(\frac{x - x_s}{w}), \tag{4}$$

with the kernels centred at each data point and the width of the kernels controlled with w.

Figure 3.5: Examples of generated antibodies, Ab_m at different τ. "Distorted" pixels result mainly from random mutations in our approach.

The density estimation accuracy hinges on the number of data points and the selection kernel shown in Fig. 3.7. Generally, a more significant number of data points leads to a more precise approximation, while the impact of the specific kernel type is relatively minor. For the kernels $\Phi(\cdot)$, irrespective of the type (e.g., Gaussian, triangular, cosine, or uniform), $\Phi(\cdot)$ must be symmetric, $\int \Phi(x)dx = 1$, and $\lim_{x\to\pm\infty} \Phi(x) = 0$ [95, 96].

3.2.2 Overview of proposed GSES

Fig. 3.8 overviews our proposed method to generate new samples resembling the original training images. We randomly select a pixel location for a given image x and change that

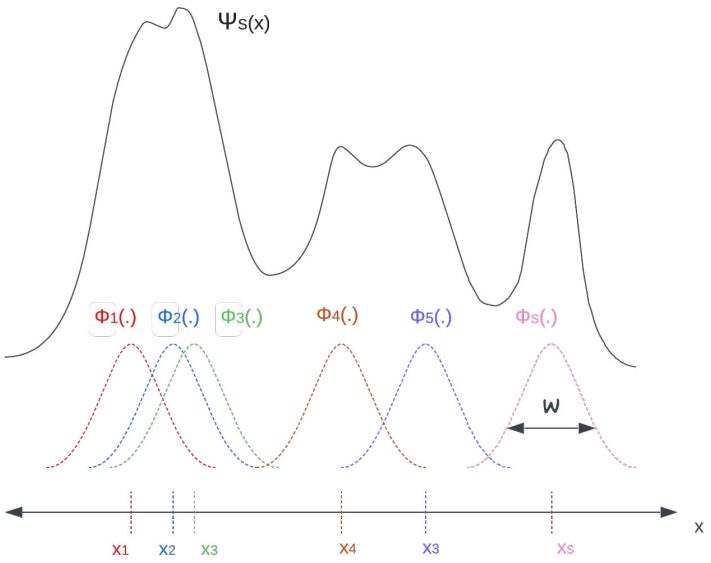

Figure 3.6: Approximation of the probability density function of $x_1, x_2, ..., x_S$ using KDE.

pixel based on a pixel distribution obtained from multiple pixel distortions using kernel density estimation. We repeat this not for all input image pixels but for a selected set. Employing this approach, the generated new sample possesses distorted pixels while maintaining a certain degree of similarity to the clean image x. This degree of similarity is essential to maintaining a balance between performance under distorted and clean images.

To ensure that our method maintains a balance between object detection performance under clean and distorted images, we generate new samples for a subset of the original training images and maintain a certain level of similarity between these new and their corresponding original images. Overall, our data augmentation approach can be divided into three main steps as detailed in the following three subsections and Algorithm 2:

- Data pre-processing, where we randomly select a portion of the original images for generating new samples.
- New sample generation, where we generate the new samples by creating distorted versions of the selected original images while maintaining the desired level of similarity.
- Data augmentation, where we use these generated new samples to augment the original images for training.

The effectiveness of our method comes from two main aspects:

1. randomly, at the pixel level, balancing the similarity of the new and the original samples (i.e., using τ in Algorithm 2).
2. diversity (that is, we use $D = 15$ distortions for pixel distribution estimation and pixel sampling in Algorithm 2).

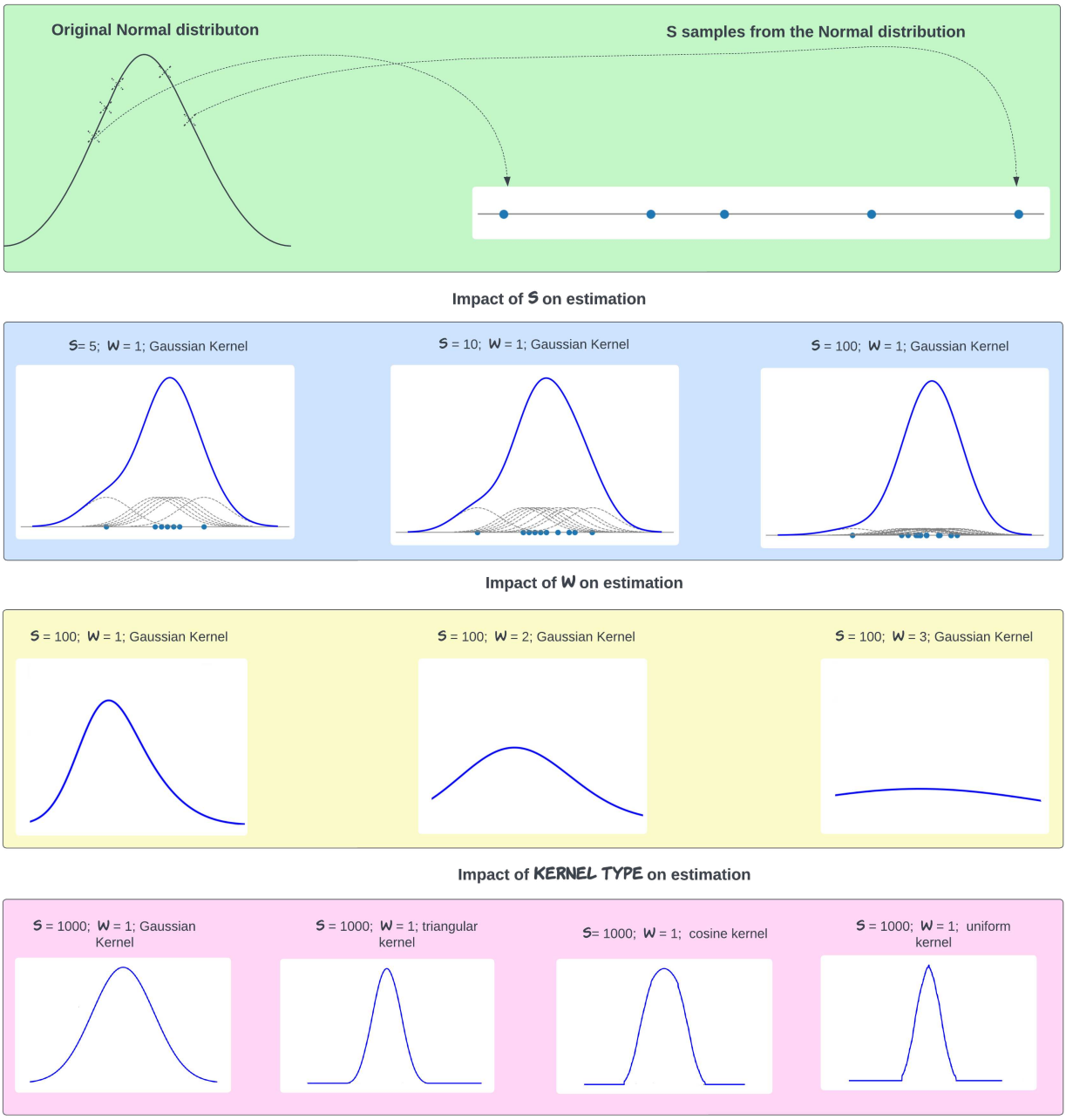

Figure 3.7: Visualizing KDE Estimation (Ψ_s) of a Normal Distribution: Row 1 depicts S samples drawn from a Normal distribution; Row 2 demonstrates Ψ_s using various sample sizes (S) employing a Gaussian kernel; Row 3 showcases Ψ_s across different bandwidths (w); Row 4 exhibits Ψ_s computed with 4 kernel functions.

A detection model trained with our data augmentation learns more robust features from clean and diversely distorted pixels. We show in Section 4 how decreasing similarity (larger τ) causes accuracy loss under clean samples and does not significantly improve the accuracy under distortion. We also show that pixel diversity helps the detection model generalize

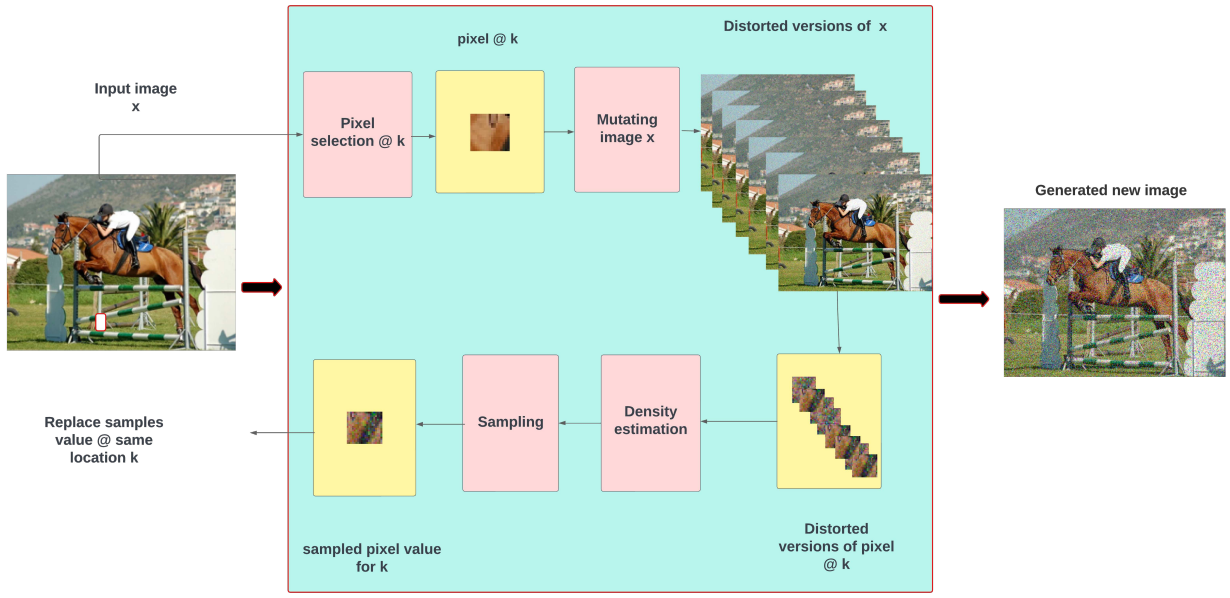

Figure 3.8: A simplified overview of GSES.

to unknown distortions and that decreasing diversity decreases detection accuracy under distortions.

3.2.3 Data pre-processing

Let the training dataset $\{x^i, y^i\}_{i=1}^{M}$ consist of M training images, where each input image x^i is associated with a corresponding label y^i. We collectively denote these images as Ag. We aim to create distorted versions of these images x^i, collectively denoted as Ab, to train object detection models and enhance their accuracy under various image distortions.

In the data pre-processing step, as outlined in Algorithm 2 lines 1 to 6, we randomly select N from the input M images $\{x^i, y^i\}_{i=1}^{M}$ to get $\{x^i, y^i\}_{i=1}^{N}$ Ab, where $N{=}p \cdot M$ and $0 < p \leq 1$ is an augmentation ratio p, a hyper-parameter of our method (see Algorithm 2 input). Before this random selection, it is always important to check the class distribution of the training images Ag. When dealing with a skewed class distribution, one should select images class-awarely to prevent class imbalance in the final training set. We select d different distortion types to consider various distortions and divide these Ab into D groups Ab_d. Each Ab_d for a distortion type d contains $T = N/D$ images (see Algorithm 2 line 6). Given $d = 1$ to D, when $d = 1$, i is from 1 to T, when $d = 2$, i is from $T + 1$ to $2T$ and so on. And since the datasets we use (PASCAL [94], and COCO [98]) have uniformly distributed classes, we can divide randomly.

Algorithm 2: Proposed new sample generation by distribution estimation and sampling.

Input: Original images $Ag = \{x^i, y^i\}_{i=1}^{M}$, D distortions types, augmentation ratio p, affinity threshold τ

Output: new samples Ab

Data pre-processing:

Number of Ab: $N = p \cdot M$;

Number of Ab_d: $T = N/D$;

New sample generation:

$Ab \longleftarrow$ *Select* N images randomly from Ag ;

for *distortion* $d = 1$ *to* D **do**

  $Ab \longleftarrow$ Randomly *shuffle* Ab;

  Select T images from Ab to get $Ab_d = \{x^i, y^i\}_{i=((d-1)\cdot T)+1}^{d \cdot T}$;

  for *each image* $x^i \in Ab_d$ **do**

    $\{\tilde{x}_i\}^Z \longleftarrow$ *Mutate*: generate Z distorted versions of x^i;

    $K \longleftarrow$ *Select* a fraction $1 - \tau$ of pixels from x^i, uniformly at random ;

    for *each pixel* k *in* K **do**

      $\tilde{X} \longleftarrow$ *Copy* the distorted pixels at position k from all images in $\{\tilde{x}_i\}^Z$;

      $E_{\tilde{X}} \longleftarrow$ *Estimate* the distribution of $\tilde{X}$ using KDE ;

      $\tilde{x}_{new} \longleftarrow$ *Sample* one pixel uniformly at random from pixel distribution $E_{\tilde{X}}$;

      $x^i \longleftarrow$ *Mutate* x^i by replacing the value at k with $\tilde{x}_{new}$;

    end

  end

end

Data augmentation:

Output $\longleftarrow$ Shuffle{original images Ag and new samples Ab};

3.2.4 New sample generation

This Section describes how new samples are generated in Algorithm 2, which *selects* an image, *estimates* its distribution, and *mutates* an image. We aim to mutate (change) selected pixels of an image x^i with the desired distortion. We perform the image mutation using two steps:

1. distortion of the original images using image processing tasks taking distortion D into account.
2. mutation (changing) of selected pixels by sampling from the distorted pixel distribution.

In the first mutation step, given an image x^i and a distortion type D (e.g., Gaussian noise, jpeg compression, and snow), we generate S distorted images $\{\tilde{x}_i\}^S$ using image processing tasks filtering, clipping, scaling, and shuffling taking D into account. The image processing tasks cause all pixels to be mutated in the distorted image $\tilde{x}_i$, resulting in not only distortion

D but also other changes (see [63] for details). Note that for each of the input T images selected for a distorted type (see Algorithm 2 line 6), we generate S distorted images, where $S = 100$ (see Algorithm 2 line 8); higher S unnecessarily increases computations.

In the second mutation step, we focus on how many pixels are changed and not how much the pixels are changed. In each image x^i, we mutate only a fraction τ of its pixels to keep a level of similarity with its clean version. We uniformly select K pixels (see Algorithm 2 line 9), where $K = \tau \cdot \text{pixels}\{x_i\}$ of the pixels we want to mutate from x^i uniformly.

Our method requires the hyper-parameter τ, while the distortion types D are fixed to 15. For example, if $\tau = 0.75$, 25% of the pixels of x^i are selected for mutation by estimating and sampling from the distortion type under consideration. In other words, 75% of the pixels remain unchanged. We consider $D = 15$ distortion types, i.e., Gaussian noise, impulse noise, shot noise, jpeg compression, pixelated, elastic transform, motion blur, glass blur, zoom blur, defocus blur, contrast, snow, fog, frost, and brightness. τ controls how many pixels are left unchanged (clean) and how many are mutated. It compromises clean and distorted pixels, improving object detection under distorted images without deterioration on clean photos. During training, the object detection model learns a mixture of "distorted" and "clean" features.

We need to estimate distortion distributions to mutate pixels in x^i effectively. We can achieve this in two main ways, namely parametric and non-parametric approaches. We opt for the standard non-parametric approach of kernel density estimation (KDE) because it makes no assumptions about the distributions of the distortion, allowing us to apply it to different distortion types. KDE has been adapted for various applications in computer vision, such as for spatio-temporal background modelling [99].

For KDE, for a selected pixel location k in x^i, we form the distorted vector $\tilde{X}$. Recall $\tilde{x}$ is a distorted version x^i and there are S such images for each x^i; we form $\tilde{X} = [\tilde{x_i}^1, \tilde{x_i}^2, ..., \tilde{x_i}^S]^\top$ for pixel location k. We use Gaussian kernels in KDE to estimate the distribution $\tilde{X}$. More specifically, for each pixel location, $\tilde{X}$ is expressed as a linear combination of equal-width Gaussians centred around each point $\tilde{X}$, where the width (or standard deviation) of the Gaussians is a hyper-parameter. As KDE is a general non-parametric method, we may utilize different kernels, including uniform or triangular kernels. We limit ourselves to the Gaussian kernel because this fits a mixture of Gaussians (a universal density approximator [100]) to the pixel distribution. Once the estimation of $\tilde{X}$ is obtained and denoted as $E_{\tilde{X}}$, we sample one pixel value from $E_{\tilde{X}}$ to get $\tilde{x}_{new}$ and use $\tilde{x}_{new}$ to mutate x^i by replacing pixel value of x^i at index k. For example, in Figure 3.9, we show new samples generated by mutating pixels with new pixel values sampled from an estimated distribution of Gaussian noise (row 1), motion blur (row 2), jpeg compression (row 3), and snow (row 4).

Figure 3.9: Examples of generated new samples by GSES. The method estimates and samples from the distribution of Gaussian noise (row 1), motion blur (row 2), jpeg compression (row 3), and snow (row 4).

3.2.5 Data augmentation

In the data augmentation step, we augment the new samples Ab to the original training images Ag to train object detection models. Before the training, we shuffle the Ab and Ag to prevent bias to either of them.

3.3 Summary

This thesis Chapter explored two novel data augmentation methods, AISbod and GSES, specifically designed to enhance the robustness of object detection models against image distortions. The methods aimed to fortify these models against distortions without deteriorating their accuracy under clean images. Central to the efficacy of both AISbod and GSES is their operation at the pixel level, which allowed for meticulous manipulation and enhancement of the original samples. AISbod distorted the original training samples by following the AIS cycle involving sequential selection, cloning, mutation, and selection steps. On the other hand, GSES distorted pixels within the original samples by replacing their values with sampled values obtained from an estimated distribution. Crucially, both AISbod and GSES continued their distortion until a predefined similarity threshold with the original sample was achieved.

In Chapter 4, we present simulation results, and in Chapter 5, we analyze why our methods are effective by examining the inner workings of the methods.

Chapter 4

Experimental Results

In the result tables, the best results are in red, and the second best in blue. The +() indicates gain, the difference between the baseline model without and with a defence method.

4.1 Datasets and evaluation metrics

- **COCO** [98]: The COCO dataset is renowned for its emphasis on realistic instances of objects within their natural settings, advancing object detection research. It encompasses 80 object classes annotated across approximately 2.5 million instances within 328k images.
- **PASCAL** [94]: Initially organized as an object detection competition from 2005 to 2012, VOC provided updated datasets annually. It consists of over $20,000$ images across 20 object classes, each annotated with object bounding boxes. While the competition concluded in 2012, researchers continue to rely on PASCAL's comprehensive annotations, making it a pivotal benchmark for evaluating object detection algorithms.

Most researchers use Mean Average Precision (mAP) to capture the accuracy of object detection models in classifying and localizing detected objects. It is defined as

$$mAP = \frac{1}{N_c} \sum_{c=1}^{N_c} AP_c, \tag{5}$$

where N_c is the total number of object classes, and AP_c is the average precision per object class obtained from the area under the $Precision - Recall$ curve of the detection model for a given intersection over union (IOU). $Precision$ of the model is calculated as

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{6}$$

Recall is defined as

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{7}$$

IOU computed as

$$IOU = \frac{B^{ground} \cap B^{predicted}}{B^{ground} \cup B^{predicted}}. \tag{8}$$

Under mAP, a True Positive is the number of correct detections with $IOU \geq$ a specified threshold; False Positive is the number of wrong detections with $IOU <$ a specified threshold, and False Negative is the number of instances when a ground truth is not detected. B^{ground} and $B^{predicted}$ are the ground truth and predicted bounding boxes respectively. We follow standard practice in evaluating the PASCAL dataset [98] and use $IOU = 0.5$ for all reported mAPs. For COCO, unless otherwise stated, the mAP is the average for 10 IOU thresholds ($IOU = 0.50 : 0.05 : 0.95$).

We measure accuracy under corruption using the "mean performance under corruption" (mPC) defined [63] as

$$mPC = \frac{1}{N_a} \sum_{a=1}^{N_a} \frac{1}{N_b} \sum_{b=1}^{N_b} mAP_{a,b}, \tag{9}$$

where N_a is the total number of corruptions (a), N_b is the severity (b) level for each corruption type, and $mAP_{a,b}$ is the mAP for a corruption at a specific severity. We also use "relative performance under corruption" (rPC) which is computed [63] as

$$rPC = \frac{mPC}{mAP_{org}}, \tag{10}$$

where mAP_{org} is the mAP under original samples.

4.2 Experimental Setup

The hyper-parameters of our approach are p, the augmentation ratio, and τ, the affinity threshold. For all object detection models, $\tau = 0.75$ for both GSES and AISbod. Only p was different for both methods: $p = 0.50$ for GSES and $p = 0.25$ for AISbod as detailed in Section 5.1.

To evaluate our approach, we use 15 image distortions: Gaussian noise, impulse noise, shot noise, jpeg compression, pixelated, elastic transform, motion blur, glass blur, zoom blur, defocus blur, contrast, snow, fog, frost, and brightness, each having five levels of severity (level 5 is highest); the details on the distortions levels are available at `https://github.com/bethgelab/imagecorruptions/blob/master/imagecorruptions/corruptions.py`. These

15 distortions can be categorized into four types: noise, blur, artefacts, and weather conditions.

To validate our proposed methods, we must evaluate them on both small and large datasets. We use the PASCAL 2007 and 2012 datasets for the small dataset, which comprises 16,551 training images, 4,952 validation images, and 20 class categories. We use the COCO 2017 dataset for the large dataset, with approximately 118,000 training images, 5,000 validation images, and 80 class categories. Our simulations use the same ground truth label of an antigen (original sample) for the corresponding generated sample.

We select the object detection models DINO (4Scale, one-stage detector), YOLOv7 (E-ELAN, one-stage detector), YOLOv4 (CSPDarknet-53, one-stage detector), and Faster RCNN (ResNet-101, two-stage detector). These selected object detection models are the fastest in their respective series. DINO inherently incorporate random crop and scale augmentation, YOLOv7 applies Mosaic [54] for data augmentation, YOLOv4 uses CutMix [51] augmentation, and Faster RCNN uses left-right flipping augmentation. We use the codes of these baseline models as provided by their authors and their data augmentation methods at training. We consider two modes of training: with and without our augmentation. As in the detection models' original works, the models are pre-trained on ImageNet. The trained models (for each instance) are then validated on the original validation samples and their distorted versions.

4.3 Results

When we started our investigation, we used the then-top models YOLOv4 and Faster RCNN on the small dataset PASCAL. We present their preliminary results in Section 4.3.1. In Section 4.3.2, we present results for the more recent top models DINO and YOLOv7 on the large dataset COCO. The remaining results Sections discuss how our approach generalizes to unknown distortions, image classification, object tracking, and cross-domains.

4.3.1 Preliminary results with PASCAL dataset

For the comparison of distortion defence methods, we compare our methods, AISbod and GSES, with the data augmentation methods, Stylize [63] and SmoothMix (SMix) [59]. With image enhancement methods RetinexFormer [71], UFormer [72], URIE [73], OWAN [74], and DeepN [75]. (We do not compare to Det-AdvProb [62] and [65] because the code for training is not publicly available). Table 4.1 for the PASCAL dataset shows that our methods balance performance under clean and distorted images among all related works. Among the data

augmentation methods, GSES outperforms (+9.54% and +5.77%) the second-best method, Stylize (+7.96% and 5.57%) in both YOLOv4 and Faster RCNN under distorted images, respectively. Under clean images, our methods do not result in any deterioration, which is significant in data augmentation methods [13]. For the image enhancers, as demonstrated in Table 4.1, only UFormer and URIE showed improvement under distortions, while OWAN failed to provide any improvement. The OWAN model did not undergo training on the 15 distortions evaluated, unlike the URIE model. DeepN only improved (4.96%) under noise (Gaussian, impulse, and shot) but resulted in significant deterioration in other distortion, hence overall negative (-10.47%) impact on accuracy.

Table 4.1: PASCAL validation set for YOLOv4 and Faster RCNN: Comparison of our methods with image distortion defence and image enhancer methods.

Model	mAP_{org}	mPC	rPC
YOLOv4 (as-is)	83.31	52.48	0.6299
+ GSES (Our)	83.32 (+0.01)	62.02 (+9.54)	0.7444
+ AISbod (Our)	83.50 (+0.19)	58.87 (+6.40)	0.7067
+ Stylize	83.90 (+0.59)	60.44 (+7.96)	0.7254
+ SMix	83.65 (+0.34)	53.04 (+0.56)	0.6367
+ RetinexFormer	-	37.84 (-14.64)	0.4542
+ UFormer	-	54.24 (+1.76)	0.6511
+ URIE	-	58.05 (+5.57)	0.6968
+ DeepN	-	41.45 (-11.02)	0.4976
+ OWAN	-	40.10 (-12.38)	0.4813
Faster RCNN (as-is)	79.25	54.63	0.6893
+ GSES (Our)	79.30 (+0.05)	60.40 (+5.77)	0.7621
+ AISbod (Our)	79.84 (+0.59)	58.26 (+3.63)	0.7351
+ Stylize	78.52 (-0.73)	60.20 (+5.57)	0.7596
+ SMix	79.65 (+0.40)	54.82 (+0.20)	0.6918
+ RetinexFormer	-	52.10 (-2.53)	0.6574
+ UFormer	-	55.35 (+0.72)	0.6984
+ URIE	-	60.08 (+5.45)	0.7581
+ DeepN	-	44.72 (-9.91)	0.5643
+ OWAN	-	41.72 (-12.91)	0.5264

We evaluate the performance of our method against conventional data augmentation methods: MixUp [101], CutMix [51], AugMix [102], RandE [52], and Mosaic [54] using YOLOv4 under the 15 distortions on the PASCAL validation dataset. As shown in Table 4.2, while all methods improve accuracy (mAP_{org}) under clean images, our methods significantly outperform these conventional augmentation methods under the 15 distortions.

4.3.2 Main results with COCO dataset

Under the COCO validation set, as shown in Table 4.3, we use recent top baseline detectors DINO and YOLOv7. The first observation from the Table is that our defence methods, GSES and AISbod, are very beneficial to the state-of-the-art DINO: Under clean (original) samples, when we add our GSES and AISbod to DINO, it mAP_{org} jumps by a remarkable

Table 4.2: YOLOv4 and PASCAL validation set: Comparison of our method with conventional data augmentation methods. YOLOv4* is YOLOv4 without its baseline augmentation CutMix.

Model	mAP_{org}	mPC	rPC
YOLOv4*	82.77	50.45	0.6095
+ GSES (Our)	83.22 (+0.45)	60.82 (+10.37)	0.7348
+ AISbod (Our)	83.16 (+0.39)	58.07 (+7.62)	0.7016
+ MixUp	82.90 (+0.13)	50.76 (+0.31)	0.6133
+ CutMix	83.31 (+0.54)	52.48 (+2.03)	0.6340
+ AugMix	81.27 (-1.50)	51.55 (+1.10)	0.6228
+ RandE	83.43 (+0.66)	49.91 (-0.54)	0.6030
+ Mosaic	83.61 (+0.84)	51.05 (+0.60)	0.6168

4.50% and 4.00%, respectively. Under distortions (mPC), with GSES and AISbod, DINO improves by a significant 8.40% and 6.50%, respectively. YOLOv7 also improves when GSES AISbod is used to defend against distortions by 4.43% and 3.68%, respectively. The second observation is that compared to related defence methods, our GSES and AISbod overall better improve the accuracy of the baselines. Specifically, our methods perform better than Stylize for clean and distorted samples. UFormer and URIE also improve the models under distorted images but to a lesser extent than our methods.

Table 4.3: COCO validation set: Comparison of our methods with Stylize, UFormer, and URIE.

Model	mAP_{org}	mPC	rPC
DINO (as-is)	43.40	22.40	0.5161
+ GSES (our)	47.90 (+4.50)	30.80 (+8.40)	0.7097
+ AISbod (our)	47.40 (+4.00)	28.90 (+6.50)	0.6659
+ Stylize	41.80 (-1.60)	26.30 (+3.90)	0.6060
+ UFormer	-	22.60 (+0.20)	0.5207
+ URIE	-	23.50 (+1.10)	0.5392
YOLOv7 (as-is)	43.10	22.32	0.5179
+ GSES (Our)	45.20 (+2.10)	26.75 (+4.43)	0.6206
+ AISbod (Our)	44.64 (+1.54)	26.00 (+3.68)	0.6032
+ Stylize	41.45 (-1.65)	25.40 (+3.06)	0.5893
+ UFormer	-	22.40 (+0.08)	0.5197
+ URIE	-	24.35 (+2.03)	0.5650

For the COCO test set, we evaluate the performance of our method, Stylize and URIE, using DINO and YOLOv7. Because of the limitations placed on submitting results to the COCO server, we evaluate under four distortions with severity level 3, one from each type of distortion, i.e., noise, blur, artefacts, and weather conditions. We thus use impulse noise, motion blur, jpeg compression, and snow because of their significant impact on DINO and YOLOv7 for each distortion type, respectively. As shown in Table 4.4, GSES (+8.35%, +5.50) and AISbod (+6.23%, +4.35%) are superior to URIE (+4.16%, +3.10%) and Stylize (+3.20%, +2.50%). Combining our methods with URIE further improved DINO (+10.35%, +7.50%) and YOLOv7 (+8.15%, +6.40%) under distortion.

Table 4.4: COCO test set: Comparison of our methods with Stylize and URIE. Here, we used impulse noise, zoom blur, jpeg compression, and snow, each severity level 3.

Model	mAP_{org}	mPC	rPC
DINO (as-is)	43.50	20.35	0.4678
+ GSES (our)	47.80 (+4.30)	28.70 (+8.35)	0.6598
+ AISbod (our)	47.30 (+3.80)	26.58 (+6.23)	0.611
+ Stylize	38.20 (-5.30)	23.55 (+3.20)	0.5414
+ URIE	-	24.51 (+4.16)	0.5634
+ GSES (our) + URIE	47.80 (+4.30)	30.70 (+10.35)	0.7057
+ AISbod (our) + URIE	47.30 (+3.80)	28.88 (+8.53)	0.6639
YOLOv7 (as-is)	43.00	20.10	0.4674
+ GSES (Our)	44.25 (+1.25)	25.60 (+5.50)	0.5953
+ AISbod (our)	44.10 (+1.10)	24.45 (+4.35)	0.5686
+ Stylize	40.50 (-2.50)	22.60 (+2.50)	0.5256
+ URIE	-	23.20 (+3.10)	0.5395
+ GSES (our) + URIE	44.25 (+1.25)	27.60 (+7.50)	0.6419
+ AISbod (our) + URIE	44.10 (+1.10)	26.50 (+6.40)	0.6163

Our results in Table 4.3 and 4.4 show that GSES improves the accuracy of both detection models better than AISbod, but DINO benefits more. For example, under the COCO test set, GSES is better by +2.12% and +1.15% under distortions for DINO and YOLOv7, respectively. This outcome indicates that GSES is more effective in handling various distortions, making it a more robust solution than AISbod. This robustness could be attributed to more diversification in the augmented samples generated by GSES. Recall that in AISbod, we distorted the augmented samples by randomly replacing pixels from a sampled pool of values. In contrast, in GSES, we distorted the samples by selecting pixels from the original samples, estimating the pixel distribution from multiple distorted versions of the original samples via kernel density estimation and then replacing the selected pixels with new values sampled from the estimated distribution.

An interesting question arises from our above results: how do classical data augmentation methods affect recent baseline models DINO and YOLOv7, and how do they compare to our data augmentation approach? First, recall that we use the lightweight versions of these models (DINO 4scale, 12 epochs and YOLOv7-L, 300 epochs). DINO's code applies random crop and scale augmentation for data augmentation. YOLOv7 uses the well-known data augmentation method Mosaic [54]. To effectively compare our approach with Mosaic, we present results for the baseline of YOLOv7 with and without Mosaic alongside our AISbod augmentation. As shown in Table 4.5, compared to Mosaic, our GSES and AISbod significantly improve DINO both under clean sample (by 4.50% and 4.00%) and under distortion (by 8.40% and 6.50%). For YOLOv7 without its augmentation (Mosaic), our GSES and AISbod improve it under clean samples (by 4.40% and 4.33%) and distortion (by 5.10% and 4.22%) well outperforming Mosaic. We see that at the baseline, DINO and YOLOv7 (each with its data augmentation) have comparable accuracy under clean (43.40% versus 43.10%)

and distortions (22.40% versus 22.32%). Additionally, combining Mosaic with our methods enhances results. For DINO, Mosaic with GSES increases performance by 5.20% under clean conditions and 8.80% under distortion, while Mosaic with AISbod shows a 4.90% and 6.55% improvement. For YOLOv7, Mosaic with GSES improves by 5.10% under clean and 5.45% under distortion, and Mosaic with AISbod by 4.54% and 4.70%, respectively. Overall, the results in Table 4.5 indicate that our data augmentation techniques have significantly improved object detection models' accuracy under distorted and clean samples.

Table 4.5: Effect of data augmentation on recent models DINO and YOLOv7 applied on the COCO validation set.

Model	mAP_{org}	mPC	fps (GPU)
DINO (as-is)	43.40	22.40	20
+ GSES (our)	47.90 (+4.50)	30.80 (8.40)	
+ AISbod (our)	47.40 (+4.00)	28.90 (+6.50)	
+ Mosaic	45.60 (+2.20)	23.20 (+0.80)	
+ Mosaic + GSES (our)	48.60 (+5.20)	31.20 (+8.80)	
+ Mosaic + AISbod (our)	48.30 (+4.90)	28.95 (+6.55)	
YOLOv7 (without its Mosaic)	40.10	21.30	42
+ GSES (our)	44.50 (4.40)	26.40 (+5.10)	
+ AISbod (our)	44.43 (+4.23)	25.52(+4.22)	
+ Mosaic (YOLOv7 as-is)	43.10 (+3.00)	22.32 (+1.02)	
+ Mosaic + GSES (our)	45.20 (+5.10)	26.75 (+5.45)	
+ Mosaic + AISbod (our)	44.64 (+4.54)	26.00 (+4.70)	

At the baseline, DINO and YOLOv7 (each with their default data augmentation) exhibit comparable accuracy under clean (43.40% vs. 43.10%) and distortion conditions (22.40% vs. 22.32%). In terms of complexity, baseline DINO maintains a speed of 20 fps, while YOLOv7 operates at 42 fps on an RTX 8000 48GB machine with 260GB RAM and 36 CPU cores (Intel Xeon @ 2.30 GHz). Notably, the baseline models equipped with our AISbod at training improve accuracy at inference without compromising the inference speeds.

To study how our augmentation approach affects training and validation, we examine the training and validation error curves shown in Figure 4.1. The loss function of DINO is

$$L_{DINO} = L_{cls} + \lambda_{giou} \cdot L_{giou} + \lambda_{box} \cdot L_{box}, \tag{11}$$

and that of YOLOv7 is

$$L_{YOLOv7} = L_{cls} + \lambda_{obj} \cdot L_{obj} + \lambda_{box} \cdot L_{box}. \tag{12}$$

In these equations, L_{cls} is the classification loss, L_{box} is the bounding box regression loss, L_{giou} is the generalized IoU ($GIoU$) loss, and L_{obj} is the objectness loss. The terms λ_{box}, λ_{giou}, and λ_{obj} are hyper-parameters that balance the contributions of their respective losses. (See Appendix D for details).

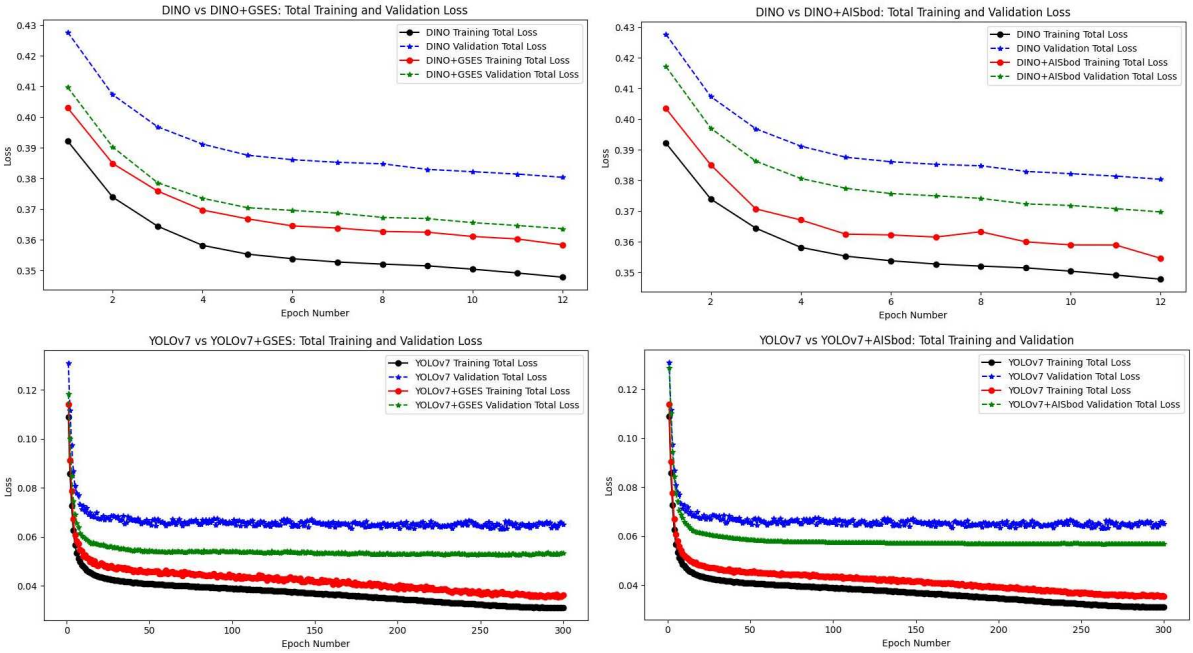

Figure 4.1: Graphical representation of the total training and validation loss of DINO and YOLOv7 using the COCO dataset as given by Eqn. 11 and 12 respectively, for GSES and AISbod.

The loss curves in Figure 4.1 confirm that the validation loss of the baseline DINO and YOLOv7 equipped with our GSES and AISbod (green) is lower than that of the baseline (blue). Compared to the base model (black and blue), the curves of the base model with our method (red and green) exhibit steeper convergence trajectories. This smoother convergence indicates reduced fluctuations in the model's learning process, reflecting a more stable optimization path. Furthermore, the narrower gap between the training and validation curves (red versus green) suggests mitigated over-fitting on the training data, as the models with our methods learn to generalize better to unseen validation data. The Figure also shows that the training loss (black versus red) with our methods is comparatively higher than the baseline; this is an expected outcome since our methods increase training samples (i.e., the original training plus the augmented samples). Recall that the total training loss is the sum of loss per input training sample.

4.3.3 Generalization to unknown distortions

Our GSES method and URIE have prior knowledge of the 15 image distortions used in our experiments. This Section checks how both methods perform under unknown distortions, such as Gaussian blur, speckle noise, spatter, and saturate. Like the 15 known distortions,

each unknown is added to the clean image with five severity levels. As shown in Table 4.6, for the PASCAL validation set and YOLOv4, GSES is effective and can be generalized for unknown distortions. Also, we see GSES significantly outperforms URIE for the unknown distortions. We provide the results for AISbod and Stylize, which do not know the distortions, to show how they compare to the distortion-dependent methods, GSES and URIE.

Table 4.6: PASCAL validation set and YOLOv4: comparison of our methods with URIE under unknown distortions. (Distortion-agnostic methods AISbod and Stylize are added for comparison.)

Model	mAP_{org}	mPC	rPC
YOLOv4	83.31	63.13	0.7578
+ GSES (Our)	83.32 (+0.01)	68.75 (+5.62)	0.8252
+ URIE	-	53.89 (-9.24)	0.6468
+ AISbod (Our)	83.50 (+0.19)	68.00 (+4.87)	0.8162
+ Stylize	83.90 (+0.59)	67.90 (+4.77)	0.8151

4.3.4 Generalization to image classification

4.3.4.1 Image classification datasets and metrics

- **CIFAR-10**: This widely-used benchmark dataset focuses on image classification tasks, comprising 60,000 32x32 colour images distributed among ten classes, with 6,000 images per class. The dataset covers everyday objects such as aeroplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.
- **CIFAR-100**: Similar to CIFAR-10, CIFAR-100 contains 100 classes with 600 images per class, all 32x32 pixels in size. This dataset offers more diverse categories, including superclasses (20) and subclasses (100), encompassing a wider range of objects and concepts than CIFAR-10. It is a more challenging dataset, often used for fine-grained classification tasks and evaluating model robustness.
- **Caltech-101**: This image dataset comprises 101 object classes, totalling approximately 9,000 images in varying sizes and resolutions. It covers a broad spectrum of objects, including faces, animals, vehicles, and household items, with about 50 to 800 images per category.

The main metric for evaluating classification models is accuracy. This metric represents correctly classified samples' proportions. For a given model, its *Accuracy* is computed as

$$Accuracy = \frac{\text{Number of correctly classified objects}}{\text{Total number of objects}}. \tag{13}$$

We highlight the versatility of our methods by extending them to image classification

tasks. For this purpose, we employ SpinalNet [18], a recent classification model, and evaluate its performance on CIFAR-10, CIFAR-100, and Caltech-101. Similar to the evaluation conducted on the COCO dataset, we utilize distorted versions of the respective classification datasets, incorporating impulse noise, motion blur, jpeg compression, and snow with severity level 3. The results presented in Table 4.7 demonstrate the efficacy of our approach in enhancing the accuracy of SpinalNet across tested distortions.

Table 4.7: CIFAR-10, CIFAR-100, and Caltech-101 test set: *Accuracy* comparison of our methods on SpinalNet. Here, we used severity level 3.

	CIFAR-10			CIFAR-100			Caltech-101		
	SpinalNet	+ GSES (Our)	+ AISbod (Our)	SpinalNet	+ GSES (Our)	+ AISbod (Our)	SpinalNet	+ GSES (Our)	+ AISbod (Our)
clean	97.50	97.71	97.65	86.79	87.24	87.04	97.07	97.35	97.43
impulse noise	33.55	94.47	95.21	11.83	79.44	81.84	94.34	95.88	96.81
motion blur	42.44	42.44	63.08	23.75	67.09	25.88	92.39	95.51	93.53
jpeg compression	54.80	54.80	68.84	26.09	59.67	32.07	95.28	96.58	96.64
snow	79.85	78.85	81.15	48.8	74.49	48.37	89.26	92.74	89.38
Average Accuracy	52.66	67.64	77.07	27.62	70.17	47.04	92.82	95.18	94.09

4.3.5 Generalization to object tracking

4.3.5.1 Object tracking datasets and metrics

- **Generic Object Tracking 10K (GOT-10k) [103]:** GOT-10k is a large-scale dataset that contains over 10,000 video sequences collected from various real-world scenes and situations. The dataset features a diverse range of object classes, motion patterns, and challenges, making it perfect for evaluating the performance and robustness of object-tracking algorithms under different conditions. Each video sequence in GOT-10k includes a single annotated target object throughout the video, allowing for accurate evaluation and benchmarking of tracking algorithms. Its test videos mainly promote the generalization of tracking algorithms towards unseen object categories.
- **TrackingNet [104]:** It is a comprehensive object-tracking dataset that provides a wide range of object appearances, scales, and motions, with over 30 object classes and fully annotated videos. This dataset includes occlusions, scale variations, background clutter, and illumination changes, representing real-world challenges encountered in object-tracking scenarios. TrackingNet aims to facilitate the development and evaluation of robust tracking algorithms through its diverse set of annotated video sequences.
- **Large-Scale Single Object Tracking (LaSOT) [105]:** LaSOT is designed explicitly for single object tracking tasks and comprises over 1,400 sequences with diverse object classes, background settings, and challenges. Its long video sequences characterize the dataset significantly longer than many other tracking datasets. This makes

LaSOT particularly suitable for evaluating the long-term tracking performance of algorithms. It covers various scenarios, including crowded scenes, occlusions, scale variations, and appearance changes, providing a comprehensive benchmark for assessing the robustness and accuracy of tracking methods.

Two of the commonly used metrics in tracking are:

- **Area Overlap (AO):** Area Overlap measures the accuracy of object tracking by assessing the overlap between predicted bounding boxes and ground truth bounding boxes. It quantifies the similarity between the predicted and ground truth bounding boxes. The resulting AO value ranges from 0 to 1, with higher values indicating better alignment and vice versa.
- **Success Rate (SR):** The success rate in object tracking is the percentage of frames or instances in a tracking sequence where the predicted bounding box of the tracked object meets specific thresholds. Success rate metrics can vary depending on the threshold used, such as $SR_{0.5}$ and $SR_{0.75}$, which measure the success rate at thresholds of 0.5 and 0.75, respectively. Higher success rate values indicate better tracking performance, representing a higher proportion of frames where the tracked bounding box aligns well with the ground truth bounding box.

We use the transformer-based tracker OSTrack [44] with the GOT10k [103] dataset for object tracking. We test our method for 15 distortions using the tracker metrics AO, $SR_{0.5}$, and $SR_{0.75}$ [103]. As shown in Table 4.8, our methods improve OSTrack under distortion for all recorded metrics.

Table 4.8: GOT10k test set: *AO, $SR_{0.5}$, and $SR_{0.75}$* comparison when using our method for object tracking on 15 distortions with severity level 3.

	AO	$SR_{0.5}$	$SR_{0.75}$
OSTrack	63.42	72.56	55.04
+GSES (Our)	64.26	73.41	56.55
+AISbod (Our)	64.16	73.35	56.22

4.3.6 Cross-domain generalization

To show if features learnt by a model using our best method, GSES, are transferable to unseen object detection datasets, we investigate the effect of training YOLOv4 on COCO but validating it on PASCAL without fine-tuning. As shown in Table 4.9, our method added to YOLOv4 still improves the accuracy under the clean samples (on average by 0.60%) and the 15 distortions (on average by 7.94% under severity level 3). Comparing the cross-dataset (COCO to PASCAL) and in-dataset (PASCAL to PASCAL) in Table 4.9, we observe two

things. First, the cross-dataset approach improves YOLOv4 under distortions by +3.69 % in terms of mPC (56.25 versus 52.56). This improvement could be because the model uses features (learned from COCO) that are fundamental to detecting objects in PASCAL. These COCO features appear to be more robust to distortions. Second, YOLOv4 performs lower by -2.61% in terms of mAP_{org} (80.70) under cross-datasets compared to in-dataset (83.31). This observation could be attributed to the difference in the resolution and object distribution of images in both datasets.

Table 4.9: Cross-dataset: Gain introduced by YOLOv4+GSES on training on COCO and validating on PASCAL. Here, we used severity level 3. (Results for PASCAL to PASCAL are provided for comparison.) See Appendix A for more discussion.

	YOLOv4	+ GSES (Our)	YOLOV4	+ GSES (Our)
	COCO to PASCAL	COCO to PASCAL	PASCAL to PASCAL	PASCAL to PASCAL
clean	80.70	81.30 (+0.60)	83.31	83.50 (+0.19)
Gaussian noise	56.07	74.17 (+18.11)	52.83	72.49 (+19.66)
shot noise	58.54	75.99 (+17.45)	56.36	73.98 (+17.62)
impulse noise	50.96	80.41 (+29.45)	48.48	73.56 (+25.08)
motion blur	48.47	54.09 (+5.62)	37.72	42.62 (+4.90)
zoom blur	42.12	45.90 (+3.78)	37.72	38.23 (+0.51)
glass blur	27.24	38.37 (+11.13)	20.12	25.14 (+5.02)
defocus blur	57.69	63.86 (+6.17)	47.84	51.12 (+3.28)
contrast	77.31	78.47 (+1.16)	73.69	74.08 (+0.39)
jpeg compression	59.17	64.85 (+5.68)	55.76	69.38 (+13.62)
pixelate	32.69	37.88 (+5.20)	31.76	43.94 (+12.18)
elastic transform	50.64	56.49 (+3.22)	53.11	53.96 (0.85)
frost	64.72	67.94 (+1.19)	60.64	59.99 (-0.65)
fog	78.08	79.26 (+3.88)	75.76	76.02 (+0.26)
snow	60.40	64.28 (+1.19)	57.37	56.88 (-0.49)
brightness	79.58	80.77 (+7.94)	79.27	79.70 (+0.43)
mPC	56.25	64.18 (+7.94)	52.56	59.41 (+6.84)
rPC	0.6970	0.7953	0.6309	0.7131 (+0.0822)

We also evaluated the performance of our method trained on COCO and validated on the autonomous driving datasets KITTI [106] and BDD100K [107] without fine-tuning. They include images with real-world distortions. Hence, we do not add any distortions to them during validation. In Table 4.10, we report results for the class of objects that overlap with the COCO dataset. Our method improves mAP_{org} of YOLOv4 by 1.69% and 1.15%, respectively.

4.4 Computational cost

The training and validation of DINO, YOLOv7, and YOLOv4 were done on an RTX 8000 48GB machine with 260GB RAM and 36 CPU cores (Intel Xeon @ 2.30 GHz). Faster RCNN was carried out on a Tesla P100-PCIE 12GB GPU machine with 32GB RAM and 8 CPU cores (Intel Silver E5-2650 v4 Broadwell @ 2.2GHz). We use the same hyper-parameters

Table 4.10: Cross-domain: Gain introduced by YOLOv4+GSES on training on COCO and validating on KITTI and BDD100K.

	YOLOv4 COCO to KITTI	+ GSES (Our) COCO to KITTI	YOLOv4 COCO to BDD100K	+ GSES (Our) COCO to BDD100K
Person	47.62	47.90 (+0.28)	45.60	46.24 (+0.64)
bicycle	4.12	9.30 (+5.18)	34.27	34.35 (+0.08)
car	74.70	74.80 +(+0.10)	57.79	58.09 (+0.30)
bus	5.00	5.74 (+0.74)	39.62	41.63 (+2.10)
truck	18.79	20.97 (+2.18)	33.36	34.66 (+1.30)
motorbike	-	-	32.03	32.65 (+0.62)
traffic light	-	-	21.63	24.70 (+3.07)
mAP_{org}	30.05	31.74 (+1.69)	37.76	38.90 (+1.15)

specified by the authors of the models in all our simulations. The training times for the models have been summarised in Table 4.11: as expected, all methods increase the training time for the base model. However, our methods introduce the least training time. It is about 1.5 times faster than the most computationally intensive method (Stylize).

Table 4.11: Training time (in days) of models on PASCAL and COCO. (Time for COCO is placed in ().)

Model	DINO (as-is)	YOLOv7 (as-is)	YOLOv4 (as-is)	Faster RCNN (as-is)
-	- (5)	(6)	4 (13)	1
+ GSES (Our)	- (7)	(7)	5 (16)	1
+ AISbod (Our)	- (7)	(7)	5 (14)	1
+ Stylize	- (9)	-(11)	6 (20)	2

The antibodies were generated from AISbod using the same PC used for Faster RCNN. Table 4.12 shows the average number and time for developing an antibody on the PASCAL and COCO datasets for a given affinity threshold offline; i.e., the antibody is generated before the training of the models. As depicted, the average cycles and time increase with higher τ.

Table 4.12: Cost of generating an antibody offline.

τ	Avg. number of cycles	Avg. time (s)
0.75	314	10
0.65	290	8
0.50	210	6

The new samples from GSES were generated on the same PC used for YOLOv4. In Table 4.13, we present the average time required to create a new sample for the PASCAL and COCO datasets with an affinity threshold of $\tau = 0.75$, factoring in the estimation and sampling of a distortion type. Notably, 70% of the time allocated for the new sample generation is consumed in creating S distorted images (see Algorithm 2 line 8) during the estimation and sampling phases (see Algorithm 2 lines 10 to 15). Regarding time consumption, blur

distortions (defocus, glass, motion, and zoom) demand the most time, whereas artefacts (contrast, elastic transform, pixelation, and jpeg compression) require the least.

Table 4.13: Cost of generating a new sample.

Distortion	Average time (s)	Distortion	Average time (s)
Gaussian noise	2.96	snow	5.16
shot noise	5.16	frost	2.90
impulse noise	2.69	fog	2.82
-	-	brightness	7.54
defocus blur	3.10	contrast	2.56
glass blur	8.57	elastic transform	7.61
motion blur	4.61	pixelate	1.90
zoom blur	28.21	jpeg compression	1.91

4.5 Summary

This Chapter compared proposed GSES and AISbod methods against existing techniques. These encompassed two image defence methods, three image enhancers, and five conventional data augmentation methods. Across 15 distinct distortions, our methods, on average, showed superior performance, establishing their efficacy in making models accurate against diverse distortions. Moreover, the Chapter unveiled the versatility of our methods, demonstrating their ability to generalize effectively to unknown distortions while exhibiting encouraging adaptability across varied tasks such as image classification and object tracking. The Chapter also showed cross-domain examination, probing the adaptability and efficacy of our methods across diverse domains. This investigation shed light on the robustness of GSES and AISbod beyond specific datasets. Furthermore, the Chapter delved into the impact of our methodologies on object sizes, unravelling insights into how these methods interacted with different object scales. This scrutiny contributed to understanding the nuanced effects of GSES and AISbod on objects of varying sizes in image processing. The Chapter ended with examining the computational costs incurred by our methods, offering insights into their efficiency in practical implementation. These observations highlighted an intriguing finding: while both GSES and AISbod provided performance enhancements for all models evaluated, GSES notably outperformed AISbod in terms of accuracy improvement and computational efficiency.

In Chapter 5, we give more insight and analysis on why our methods are effective.

Chapter 5

Analysis

In this Chapter, we present an analysis of the selection process for the hyper-parameters (augmentation ratio p and affinity threshold τ) of our data augmentation methods, AISbod and GSES in Section 5.1. Additionally, detailed insights and analyses on the effectiveness of AISbod and GSES are provided in Sections 5.2 and 5.3, respectively.

5.1 Hyper-parameter analysis

The hyper-parameters common in our data augmentation methods AISbod and GSES are p (augmentation ratio) and τ (the affinity threshold). They affect the accuracy of the object detection models. The other parameters of AISbod are n (the selected antibodies Ab before mutation), N (the total number of Ab to be cloned for a particular antigen), α (the multiplying factor), s (number of points of mutations) and q^* (the selected Ab after mutation). These parameters mainly affect the speed (computational time) of the CLONALG cycle adopted in AISbod and have been extensively covered in [1]. Their values are given in Algorithm 1. Therefore, the analysis will focus on τ and p while keeping n, N, α, s, and q^* as suggested in [1] for AISbod. Although we presented results for different datasets, models, and distortions, to minimize bias in our findings, we performed hyper-parameter tuning using only the PASCAL dataset under Gaussian noise only. We exclusively trained and validated on the PASCAL 2012/2007 train and validation sets, using only YOLOv4 and Faster R-CNN models.

The chosen approach to hyper-parameter tuning for AISbod and GSES data augmentation methods, focusing on p and τ, offers several advantages over conventional methods such as grid search [108, 109] and random search [110–112]. By leveraging domain-specific knowledge, it targets the most impactful parameters, reducing complexity and computational burden. Empirical observations guide optimal value selection, enhancing practical insights

and robustness. This approach ensures reproducibility and consistency across different object detection models while maintaining computational efficiency by fixing other parameters based on established research.

To check the effect of p, we fix $\tau = 0.75$ while changing p. We experimented with three different p (0.30, 0.25, and 0.20 for AISbod, and 0.25, 0.50 and 1.00 for GSES) and observed that our methods were not sensitive to variation of p under distortions. However, under AISbod, for $p > 0.30$, the mAP of the object detection model worsens on distortion-free samples, and $p < 0.20$ reduces the robustness of the model significantly. We observed that selecting a probability of $p = 0.25$ for AISbod and $p = 0.50$ for GSES is an optimal compromise for all the object detection models under evaluation. The GSES model benefits from a higher p because the augmented samples it generates are more diverse compared to those from AISbod.

We also study the effect of different values of τ for the chosen value of p (0.25 and 0.50 for AISbod and GSES, respectively). The robustness of the object detection model decreases with decreasing τ (i.e., $\tau = 0.85$ to 0.50). This means the difference between the generated (antibodies) and clean samples (antigens) should not be too high. Values outside the given range resulted in either lowering the mAP on both distortion-free or distorted samples. We, thus, select $\tau = 0.75$ for both methods for all object detection models under evaluation (i.e., DINO, YOLOv7, YOLOv4, Faster RCNN, and SSD).

5.2 Analysis of AISbod

In this Section, we provide the inner workings of the AISbod method. We show how each component in the technique impacts the robustness of models and why AISbod helps make models robust to distortions.

5.2.1 Class balance

Recall that ASIbod generates antibodies using only a $p = 0.25$ fraction of all original training samples. Using YOLOv4 and PASCAL, we conducted two sets (A and B) of simulations every five times. In set A, we select K antigens with attention to class balance (using the annotation files of PASCAL), while in set B, we select K antigens at random without attention to class balance. Table 5.1 shows that Set A (attention to class balance) performs slightly better (higher mean and lower STD) than Set B. We then performed a hypothesis test using the Student's t-test and the Mann-Whitney U test [113]. The null hypothesis for the test is that there is no statistical difference in means (for the t-test) or distributions (for

the U test) between Set A and Set B. Using the Student's t-test, the p-values (to validate the hypothesis against observed data) for mAP_{org} and mPC were 0.0620 and 0.6045 respectively. The Mann-Whitney U test yielded p-values of 0.0712 and 0.2654 for mAP_{org} and mPC, respectively. With the p-values greater than 0.05 in both the Student's t-test and Mann-Whitney U test, there is not enough evidence at the 95% confidence level to reject the null hypothesis that there is no statistical difference between sets A and B. This outcome means that sets A and B are not statistically different. However, Set A likely performs better on an original dataset where the class distribution is significantly skewed towards a particular class.

Table 5.1: Mean and STD of five experiments: YOLOv4+AISbod on PASCAL with and without attention to class balance in the generated antibodies Ab_m.

	Set A (with attention)	Set B (without attention)
mAP_{org}	83.48 (±0.1254)	83.28(± 0.1626)
mPC	58.76 (± 0.1628)	58.60(± 0.2122)

5.2.2 Quantization

In AISbod, the objective of the quantization (reducing the number of pixel intensities in the generated antibodies from 256 to 16, line 4 of Algorithm 1) is to decrease computational time; indeed, it reduces this time per antibody by 35%. We tested quantization to 8 levels, but this produced antibodies very different from the antigens and have visually significant artefacts compared to 16 levels, at the same affinity τ. To quantify the effect of the quantization, we trained YOLOv4 on PASCAL using our method AISbod, but without quantization; the mAP_{org} is 83.46 (compared to 83.50 with quantization), and the mPC is 58.38 (compared to 58.87 with quantization); we conclude that quantization has little effect on the accuracy of YOLOv4. Interestingly, when we augmented the original training samples of PASCAL with 16-level quantized samples (without any AIS samples) and then trained YOLOv4 itself on these augmented samples (again $p = 0.25$), we found that this reduced accuracy mAP_{org} under clean samples by only 0.33% (that is, from 83.31 to 82.98). Accuracy mPC under distortions decreased by only 0.47% (from 52.48 to 52.01). This observation confirms the earlier conclusion that the quantization does not significantly impact the accuracy of YOLOv4.

5.2.3 Why does AISbod make object detection more robust?

Different models' architectures make it difficult to have a generalized approach to regularizing their weights to make them robust to image distortions. Our method helps the models regularize those weights. We show this by studying the weights of YOLOv4 and

YOLOv4+AISbod. We extract all the network weights from the Tensorflow versions of both models. The layers comprise the convolutional layers and some batch normalization layers. Since YOLOv4 and YOLOv4+AISbod have the same architecture, each layer has the same number of weights. For each layer, we query the absolute value of the weights of YOLOv4 (Org) and YOLOv4+AISbod (AISbod) and perform the following steps. We calculate the mean and standard deviation of the absolute values of the weights in the YOLOv4 model and compare them with those of the YOLOv4+AISbod model. To show that the difference in mean between the weights from YOLOv4 and YOLOv4+AISbod is statistically significant, we perform a hypothesis test between the two sets of weights in that layer. We then plot a histogram of each layer's absolute weights for the YOLOv4 and YOLOv4+AISbod models to observe visual differences. (See Fig. 5.1 for one of such plots). We use the Mann-Whitney U test since it makes no assumptions about the distribution of the absolute weights. The null hypothesis here is that there is no statistical difference in the absolute value of weights in the given layer between the YOLOv4 and YOLOv4+AISbod models. Using a 95% confidence level, we list the layers where we reject the null hypothesis, i.e., where the absolute value of weights between the YOLOv4 and YOLOv4+AISbod models is statistically different. We find that there were nine layers where the null hypothesis was rejected. We noticed that in all nine layers where the absolute weights are statistically different (i.e., null hypothesis rejected), the YOLOv4+AISbod model has stochastically smaller absolute weights than the YOLOv4 model. This outcome is achieved from another Mann-Whitney U test, where the alternative hypothesis is that the absolute weights from the YOLOv4+AISbod are stochastically smaller than from the YOLOv4 model. These smaller absolute weights suggest that training with the augmented data helps regularise the network by making the weights smaller (at least in the sense of the L_1 since we are comparing the absolute values). The smaller weights are a result of implicit regularization of the form: $\min_w \mathcal{L}(X, w) + \alpha||w||_1$, which results in the network weights becoming smaller than the base model where $\mathcal{L}$ is the loss function, X is the training samples, and w is the network weights. Our algorithm seems to control the degree of regularization α through the affinity threshold τ. The proposed augmentation essentially provides an extra constraint on the model to force it to adjust its weight to learn not only the original images but also the augmented distorted versions. This approach consequently regularizes the network by reducing the magnitude of the weights.

5.3 Analysis of GSES

In this Section, we investigate the underlying reasons behind the effectiveness of GSES. The KDE estimator plays an essential role in GSES. Therefore, we first provide details on how

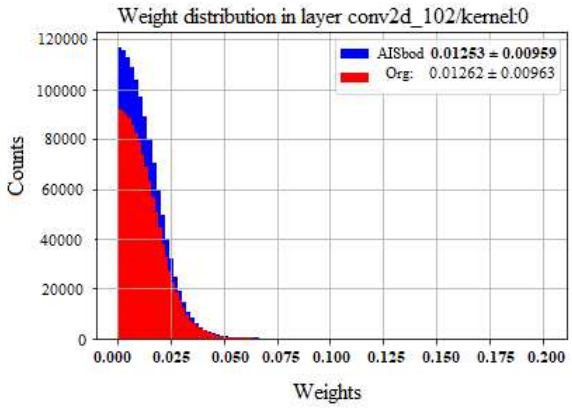

Figure 5.1: Regularization of weights in a layer using our AISbod. The legend shows the mean and STD of the weight distribution. The lowest mean and STD are in bold.

it works and then show how it is utilized in GSES.

5.3.1 Impact of KDE

We delve into the influence of KDE parameters within the context of GSES, particularly examining the effect of two key factors: the number of samples S utilized in KDE and the choice of kernel employed, using the PASCAL dataset with YOLOv4. Our analysis presents mean and standard deviation data from 5 distinct experiments for each observed outcome to capture the expected result.

In Table 5.2, the trend indicates that the resilience of YOLOv4 amplifies with an augmented S. However, this augmentation inversely impacts the model's accuracy when handling clean samples. When $S = 100$, a compromise emerges between robustness and accuracy under clean sample conditions.

Table 5.2: Effect of the number of samples S with a Gaussian kernel and $w = 1$ used in new sample generation for YOLOv4 and PASCAL.

	mAP_{org}	mPC
$S = 10$	83.58±0.2025	60.11±0.5594
$S = 100$	83.29±0.1696	62.28±0.4469
$S = 200$	83.08±0.1117	62.94±0.3357

For the kernel type's impact, Table 5.3 showcases the superiority of the Gaussian kernel over other options across both clean and distorted samples. This result reaffirms the Gaussian kernel's status as a universal density approximator [100].

Table 5.3: Effect of the type of kernel with $S = 100$ and $w = 1$ used in new sample generation for YOLOv4 and PASCAL.

Kernel	mAP_{org}	mPC
Gaussian	83.29±0.1696	62.28±0.4469
Triangular	81.47±0.2236	61.94±0.3045
Cosine	82.35±0.3365	62.04±0.4475
Uniform	81.650±.3354	62.16±0.4236

5.3.2 Impact of distortion types

The study is about the effect of the distortion diversity, that is, the number of distortions D in Algorithm 2. We aim to maintain an equilibrium among different distortions within the total number used to maintain a balanced diversity. For instance, when we tested for four distortions, we used one distortion from noise, blur, artefacts, and weather condition distortion. Table 5.4 suggests that increasing the number of distortions enhances performance in the presence of distortions. However, this increase may cause the performance under clean to reduce, although not worse than the base model.

Table 5.4: Effect of the number of distortions used in new sample generation for YOLOv4 and PASCAL.

No. of distortions	mAP_{org}	mPC
4	83.71(+0.40)	59.41(+6.93))
8	83.87(+0.56)	60.32(+7.84)
12	83.32(+0.01)	61.31 (+8.83)
15	83.32(+0.01)	62.02(+9.54)

5.4 Effect of object size on GSES and AISbod

We evaluated the effectiveness of our methods, Stylize and URIE, in detecting objects of varying sizes: small, medium, and large, as defined by [98]. Under distortions, YOLOv4 particularly faces challenges in detecting small objects compared to large ones, as seen in Table 5.5. Adding our methods GSES and AISbod to YOLOv4 improves its accuracy by 5.93%, 10.30%, and 10.87% and 3.35%, 8.10%, and 12.25% for small, medium, and large objects, respectively. Also, our methods consistently outperform Stylize and URIE across all size categories. The solid performance of our method seems to stem from its pixel-level operation. While Stylize and URIE exhibit improvements, their most notable enhancements are in medium and large object sizes.

Table 5.5: COCO test set: Impact of our method and related works on object size averaged under the distortions impulse noise, motion blur, jpeg compression, and snow with severity level 3.

size	YOLOv4	GSES (our)	+ AISbod (our)	+Stylize	+ URIE
small	8.53	14.45 (+5.93)	11.88 (+3.35)	9.33 (+0.80)	9.30 (+0.77)
medium	23.90	34.20 (10.30)	32.00 (+8.10)	26.43 (+2.53)	27.90 (+4.00)
large	32.78	43.65 (10.87)	45.03 (+12.25)	38.13 (+5.35)	40.30 (+7.52)

5.5 Impact of model complexity on GSES and AISbod

An object detection model's complexity is critical to its accuracy; this could be attributed to the bias-variance trade-off to the total number of model parameters [114] as shown in Fig 5.2. Indeed, observing Table 5.6, the best accuracy of Faster RCNN is achieved with ResNet-101 for both GSES and AISbod, which has approximately 44M compared to VGG-16's 138M parameters.

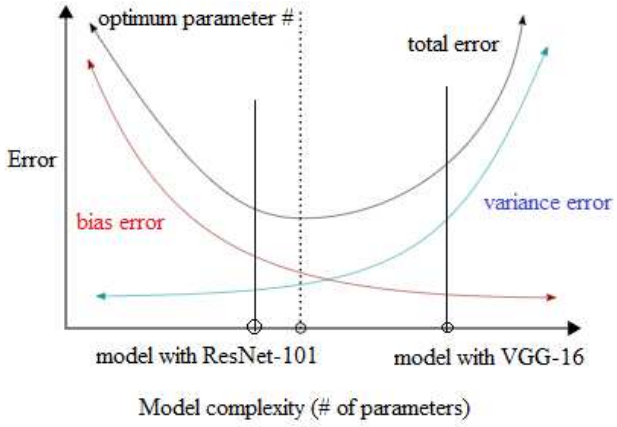

Figure 5.2: Bias-variance trade-off and model complexity.

Table 5.6: Effect of the complexity of Faster RCNN and our methods.

Model	mAP_{org}			mPC		
Faster RCNN	-	+ GSES (our)	+ AISbod (our)	-	+ GSES (our)	+ AISbod (our)
w/ ResNet-101	79.25	79.30	79.84	54.63	60.40	58.26
w/ VGG-16	75.51	75.55	74.68	45.89	49.00	50.09

5.6 Stability of GSES and AISbod

For a stability study, we conducted five experiments (i.e., we trained and validated each model five times). We used our methods (GSES and AISbod) and Stylize (the second-best method) to calculate their mean and standard deviation. As shown in Table 5.7, YOLOv4 appears

to be the most stable under clean images. However, under distortion, YOLOv4+AISbod seems to be the most stable, followed by YOLOv4+GSES. This observation is consistently confirmed under mPC and rPC. This observation aligns with that GSES employ more diverse distortion types to generate new samples than AISbod. Overall. It seems our methods provide the best stability under clean and distorted images.

Table 5.7: Stability analysis using mean and standard deviation.

Model	YOLOv4	+ GSES (Our)	AISbod (Our)	+ Stylize
mAP_{org}	83.39(±0.1559)	83.29(±0.1696)	83.45(±0.1838)	83.80(±0.1753)
mPC	52.20(±0.4739)	62.28(±0.4469)	58.57(±0.2637)	61.11(±0.5129)
rPC	0.6259(±0.0058)	0.7476(±0.0054)	0.7024(±0.0030)	0.7328(±0.0065)

5.7 Summary

This thesis Chapter explored our proposed methodologies, AISbod and GSES, through an extensive analysis encompassing various dimensions of their performance and underlying mechanisms. The Chapter also showed the proposed methods' stability analysis and the effect of model complexity on them.

The main reasons for the accuracy improvement in our methods were regularization and diversification. Our methods improved robustness by regularizing the models' weights during training. We confirmed this in the narrow gap between the train versus validation loss curves shown in Figure 4.1. This regularization effect was underscored in Section 5.2.3, where we related the affinity threshold τ of our AISbod to the regularization form $\min_w \mathcal{L}(X, w) + \alpha||w||_1$. However, we saw that GSES had higher accuracy improvement than AISbod even though it has the same τ as AISbod. This superior performance of GSES is because it has more diverse samples, which we show in Table 5.4 to benefit the detection models.

Chapter 6

Conclusion and future work

6.1 Conclusion

Object detection is crucial in vision systems like self-driving cars and healthcare monitoring. While substantial progress has been made, image distortions still affect the accuracy of detection models. In this thesis, we first evaluated the deterioration in the accuracy of object detection models in the presence of image distortions (e.g., the accuracy of the DINO baseline drops by 24.50% under snow and 27.10% under impulse noise). We then proposed two data augmentation methods to generate distorted samples from the original training samples by changing them at the pixel level. The first method, AISbod, iteratively updates an antibody through "select, clone, mutate, select" cycles until a particular affinity to the original is achieved. The second method, GSES, creates new samples by randomly replacing pixels with new values sampled from an estimated distribution while maintaining a level of similarity to the antigens.

Our augmentation approach is novel in creating new training samples that are "similar" to the original samples but still "different enough" to improve object detection performance under clean and distorted images. It is effective because of 1) diversification, i.e., how we add different distortions to the images, 2) how we fix the number of pixels to distort, and 3) how we decide the number of distorted samples for augmentation. We do not simply add noise (or distortion) to the training sample. As shown in [13, 15, 16, 93], adding noise has two main drawbacks: poor performance on clean samples and lack of generalization to unknown distortions. Our approach mitigates these drawbacks by how we distort (Algorithms 1 and 2), the number of pixels we distort, and the number of distorted samples we use for augmentation.

Through extensive experiments, we have demonstrated that the overall accuracy of the proposed methods (AISbod and GSES) surpasses that of 15 related works (under COCO

by 6.23% and 8.35% for DINO, respectively). These results are consistent across different architectures (DINO, YOLOv7, YOLOv4 and Faster RCNN) and datasets (PASCAL and COCO). Our methods stand out as they improve models under distortion and clean samples, a significant achievement considering that data augmentation methods that enhance accuracy under distortions typically suffer under clean samples. Comparing our AISbod and GSES methods, we see that the GSES method is more effective overall. This is because 1) GSES produces more varied distortions, which increases the diversification in the training samples for all evaluated models, 2) GSES reduces over-fitting by regularising the models consistently more than in AISbod, and 3) GSES is computationally more efficient.

In conclusion, this thesis showed that we can use the complex mathematical concept AIS and the well-established KDE to alter images at the pixel level for data augmentation. Our research has thoroughly explored the proposed methods and dissected their components through extensive analysis and ablation studies, shedding light on their contributions. Our efforts have extended to optimizing the KDE technique within GSES, and we have explored diverse density estimators to enhance its performance. Furthermore, our investigation has ventured beyond the boundaries of object detection, extending the applicability of AISbod and GSES to image classification and object tracking. By achieving these milestones, we aim to inspire the development of more resilient computer vision systems for unseen data and applications.

6.2 Future work

We categorize the possible extensions of our methods as follows.

- **Multi-pixel selection:** Our proposed approach is pixel-based, which may be more suited to handling localized, random noise (such as impulse noise) rather than structured distortions (such as fog). Exploring a more global approach, such as block-based, to select pixels for augmentation could be beneficial under different types of distortions.

- **Test time augmentation adaptation:** Our AISbod and GSES augment data to train the deep learning model during the training phase. This approach can be termed training time augmentation. However, another augmentation approach is called test time augmentation (TTA). TTA is a technique used during a machine learning model's inference or testing phase, where multiple augmented versions of the input data are fed into the model to make predictions [115–117]. These predictions are then aggregated in some way, such as averaging or taking the majority vote, to produce the final output.

TTA aims to improve the robustness and generalization of the model by presenting it with variations of the test data, similar to what it experienced during training.

The two main issues with TTA are: 1) implementing TTA during inference requires computing predictions for each augmented version of the test image, which increases the computational overhead and might become impractical in real-time models. This issue is particularly relevant when considering deploying computer vision systems in resource-constrained environments. 2) Deciding on what augmentation methods to use and how to aggregate final predictions is not trivial. This decision may vary for different datasets and models, and finding the optimal strategy can be complex. It will be interesting to see how best AISbod and GSES could be integrated in a TTA manner in future works while addressing these main drawbacks of TTA.

- **Extension to adversarial attacks:** Our research has demonstrated promising results (see Appendix B) in employing our best defence method, GSES, against adversarial attacks [13]. However, as observed in line with other existing adversarial defence methods [14–16], the performance of clean samples remains substantially affected. Addressing this limitation becomes a focal point for future endeavours.

  A proposed approach for mitigating the toll on clean samples involves adopting a hybrid approach combining image and network-domain strategies to fortify models against adversarial attacks. Adversarial attacks are typically devised with a deep understanding of a model's cost function and the distribution of its trained weights. As a result, a holistic defence strategy is needed. In the image domain, future work should focus on generating novel samples using the GSES approach. In the network domain, the focus should be pinpointing vulnerable weights within a model susceptible to adversarial attacks. Understanding the specific weaknesses in a model allows for a targeted and efficient defence mechanism. The proposed hybrid defence will retrain only the vulnerable weights using new samples generated through the GSES approach. By isolating and reinforcing the weak points in the model, the aim will be to create a more robust defence against adversarial attacks without compromising the performance on clean samples.

- **Application to catastrophic forgetting in large language models:** Catastrophic forgetting [118], a phenomenon where neural networks, including large language models (LLMs) [119, 120], lose previously learned information upon learning new data, is not just a challenge, but a significant hurdle in sequential learning scenarios. This issue is particularly critical in LLMs, designed to maintain a vast and diverse set of knowledge. The continual updating of model parameters to optimize for new tasks often leads

to degradation in performance on earlier tasks, making the problem of catastrophic forgetting in LLMs an urgent and crucial concern in machine learning and natural language processing.

Addressing catastrophic forgetting efficiently, without excessive computational costs, is essential for effectively deploying LLMs. To this end, several parameter-efficient strategies called Parameter-Efficient Fine-Tuning (PEFT) [121] have been developed, and they include approaches like soft prompts [122].

PEFT techniques address the issue of catastrophic forgetting by fine-tuning only a small subset of model parameters, providing a practical and efficient solution. This approach reduces computational costs and minimizes the risk of overwriting previously learned information. By freezing most of the pre-trained model's parameters and only fine-tuning additional small parameters or adapter layers, PEFT ensures that the model's core knowledge remains intact. This approach allows for significant memory and computational savings while achieving performance comparable to full fine-tuning.

In soft prompts, continuous vectors are added to the input tokens to guide the LLM's behaviour for specific tasks. These prompts are trained during fine-tuning and can be used without altering the core model parameters. Soft prompts are lightweight, computationally efficient, and effective in preventing catastrophic forgetting by maintaining the integrity of the pre-trained model while adapting to new tasks. It will be interesting to see the impact that using AISbod to increase and diversify these vectors will have on the performance of LLMs.

Bibliography

[1] L. N. de Castro and F. J. V. Zuben, "Learning and Optimization using the Clonal Selection Principle," *IEEE Trans. on Evolutionary Computation*, vol. 6, pp. 239–251, 2002.

[2] H. Zhang *et al.*, "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection," in *Proc. Int. Conf. on Learning Representations*, 2023.

[3] C.-Y. Wang *et al.*, "YOLOv7: Trainable Bag-of-freebies sets New state-of-the-art for real-time Object Detectors," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 7464–7475, 2023.

[4] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 13029–13038, 2021.

[5] Z. Zou *et al.*, "Object Detection in 20 years: A Survey," *Proc. of the IEEE*, 2023.

[6] Z. Peng *et al.*, "LGGD+: Image Retargeting Quality Assessment by Measuring Local and Global Geometric Distortions," *IEEE Trans. Circuits Syst. Video Techn.*, 2021.

[7] Y. Zhou *et al.*, "Omnidirectional Image Quality Assessment by Distortion Discrimination Assisted Multi-Stream Network," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 32, pp. 1767–1777, 2021.

[8] Z. Pan *et al.*, "DACNN: Blind Image Quality Assessment via a Distortion-Aware Convolutional Neural Network," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 32, pp. 7518–7531, 2022.

[9] S. Dodge and L. Karam, "Understanding how Image Quality affects Deep Neural Networks," in *Proc. Int. Conf. on Quality of Multimedia Experience*, pp. 1–6, 2016.

[10] M. Koziarski and B. Cyganek, "Image Recognition with Deep Neural Networks in Presence of Noise-Dealing with and Taking Advantage of Distortions," *Integrated Computer-Aided Engineering*, vol. 24, pp. 337–349, 2017.

[11] T. S. Borkar and L. J. Karam, "DeepCorrect: Correcting DNN Models against Image Distortions," *IEEE Trans. Image Process.*, vol. 28, pp. 6022–6034, 2019.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *Proc. Int. Conf. on Learning Representations*, 2015.

[13] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Machine Learning at Scale," in *Proc. Int. Conf. on Learning Representations*, 2017.

[14] Z. Dong, P. Wei, and L. Lin, "Adversarially-Aware Robust Object Detector," in *Proc. European Conf. Computer Vision*, pp. 297–313, 2022.

[15] P.-C. Chen, B.-H. Kung, and J.-C. Chen, "Class-Aware Robust Adversarial Training for Object Detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 10420–10429, 2021.

[16] H. Zhang and J. Wang, "Towards Adversarially Robust Object Detection," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 421–430, 2019.

[17] Z. He, "Deep Learning in Image Classification: A Survey Report," in *Proc. Int. Conf. on Information Technology and Computer Application*, pp. 174–177, 2020.

[18] H. D. Kabir *et al.*, "SpinalNet: Deep Neural Network with gradual input," *IEEE Trans. on Artificial Intelligence*, 2022.

[19] B. Yan *et al.*, "Alpha-Refine: Boosting Tracking Performance by Precise Bounding Box Estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 5289–5298, 2021.

[20] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic Regression for Visual Tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 7181–7190, 2020.

[21] Y. LeCun *et al.*, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Proc. Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[23] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 770–778, 2016.

[25] M. Tan and Q. Le, "Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. Int. Conf. on Machine Learning*, pp. 6105–6114, 2019.

[26] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. on Learning Representations*, 2021.

[27] H. Touvron *et al.*, "Training data-efficient image transformers; distillation through attention," in *Proc. Int. Conf. on Machine Learning*, pp. 10347–10357, 2021.

[28] Z. Liu *et al.*, "Swin transformer: Hierarchical Vision Transformer using Shifted Windows," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 10012–10022, 2021.

[29] S. Ren *et al.*, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Proc. Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[30] J. Dai *et al.*, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Advances in Neural Information Processing Systems*, pp. 379–387, 2016.

[31] K. He *et al.*, "Mask R-CNN," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 2961–2969, 2017.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 779–788, 2016.

[33] W. Liu *et al.*, "SSD: Single Shot Multibox Detector," in *Proc. European Conf. Computer Vision*, pp. 21–37, 2016.

[34] T.-Y. Lin *et al.*, "Focal Loss for Dense Object Detection," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 2980–2988, 2017.

[35] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 10781–10790, 2020.

[36] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

[37] D. Reis and otherss, "Real-time flying object detection with yolov8," *arXiv preprint arXiv:2305.09972*, 2023.

[38] N. Carion *et al.*, "End-to-end object detection with transformers," in *Proc. European Conf. Computer Vision*, pp. 213–229, 2020.

[39] X. Zhu *et al.*, "Deformable {detr}: Deformable transformers for end-to-end object detection," in *Proc. Int. Conf. on Learning Representations*, 2021.

[40] K. Yang *et al.*, "SiamCorners: Siamese corner networks for visual tracking," *IEEE Transactions on Multimedia*, vol. 24, pp. 1956–1967, 2021.

[41] Z. Chen *et al.*, "Siamese Box Adaptive Network for Visual Tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6668–6677, 2020.

[42] G. Bhat *et al.*, "Learning Discriminative Model Prediction for Tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6182–6191, 2019.

[43] Z. Zhang *et al.*, "Ocean: Object-Aware Anchor-Free Tracking," in *Proc. European Conf. Computer Vision*, pp. 771–787, Springer, 2020.

[44] B. Ye *et al.*, "Joint Feature Learning and Relation modeling for Tracking: A One-Stream Framework," in *Proc. European Conf. Computer Vision*, pp. 341–357, Springer, 2022.

[45] Q. Wu *et al.*, "DropMAE: Masked Autoencoders with Spatial-Attention Dropout for Tracking Tasks," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 14561–14571, 2023.

[46] X. Chen and otherss, "SeqTrack: Sequence to Sequence Learning for Visual Object Tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 14572–14581, 2023.

[47] X. Wei *et al.*, "Autoregressive Visual Tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 9697–9706, 2023.

[48] G. Magna *et al.*, "Adaptive Classification Model Based on Artificial Immune System for Breast Cancer Detection," in *Proc. IEEE AISEM Annual Conf.*, pp. 1–4, 2015.

[49] A. Ridok, W. F. Mahmudy, and M. Rifai, "An Improved Artificial Immune Recognition System with Fast Correlation Based Filter (FCBF) for Feature Selection," in *Int. Conf. on Image Information Processing*, pp. 1–6, 2017.

[50] C. C. B. Fioravanti *et al.*, "A Deep Artificial Immune System to Detect Weld Defects in DWDI Radiographic Images of Petroleum Pipes," *IEEE Access*, vol. 7, pp. 180947–180964, 2019.

[51] S. Yun *et al.*, "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6023–6032, 2019.

[52] Z. Zhong *et al.*, "Random Erasing Data Augmentation," in *Proc. AAAI Conf. on Artificial Intelligence*, pp. 13001–13008, 2020.

[53] Y. Wang, X. Pan, S. Song, H. Zhang, G. Huang, and C. Wu, "Implicit Semantic Data Augmentation for Deep Networks," in *Proc. Advances in Neural Information Processing Systems*, 2019.

[54] F. Dadboud, Patel, *et al.*, "Single-stage UAV detection and classification with YOLOv5: Mosaic Data Augmentation and Panet," in *Proc. IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pp. 1–8, 2021.

[55] P. Chen, S. Liu, H. Zhao, and J. Jia, "Gridmask Data Augmentation," *arXiv preprint arXiv:2001.04086*, 2020.

[56] K. Deng *et al.*, "Jointing Recurrent Across-Channel and Spatial Attention for Multi-Object Tracking with Block-Erasing Data Augmentation," *IEEE Trans. Circuits Syst. Video Technol.*, 2023.

[57] Z. Xu, A. Meng, *et al.*, "Continuous copy-paste for one-stage multi-object tracking and segmentation," in *Proc. IEEE Int. Conf. Computer Vision*, 2021.

[58] Z. Cheng *et al.*, "DeepMix: Online auto data augmentation for robust visual object tracking," in *International Conf. on Multimedia and Expo ICME*, pp. 1–6, IEEE, 2021.

[59] J.-H. Lee *et al.*, "SmoothMix: A Simple Yet Effective Data Augmentation to Train Robust Classifiers," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 756–757, 2020.

[60] E. Rusak *et al.*, "A Simple Way to make Neural Networks Robust against Diverse Image Corruptions," in *Proc. European Conf. Computer Vision*, pp. 53–69, 2020.

[61] A. Von Bernuth, G. Volk, and O. Bringmann, "Augmenting Image Data Sets With Water Spray Caused by Vehicles on Wet Roads," in *Proc. IEEE Int. Intelligent Transportation Systems Conf.*, pp. 3055–3060, 2021.

[62] X. Chen *et al.*, "Robust and Accurate Object Detection via Adversarial Learning," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 16622–16631, 2021.

[63] C. Michaelis *et al.*, "Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming," *arXiv preprint arXiv:1907.07484*, 2019.

[64] A. Beghdadi, M. Mallem, and L. Beji, "Benchmarking Performance of Object Detection under Image Distortions in an Uncontrolled Environment," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pp. 2071–2075, 2022.

[65] Z. Sun *et al.*, "Feature Quantization for Defending against Distortion of Images," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 7957–7966, 2018.

[66] H. Li, G. Li, and Y. Yu, "ROSA: Robust Salient Object Detection against Adversarial Attacks," *IEEE Trans. on Cybernetics*, vol. 50, pp. 4835–4847, 2020.

[67] P. Benz *et al.*, "Revisiting Batch Normalization for Improving Corruption Robustness," in *Proc. of the IEEE/CVF Winter Conf. on Applications of Computer Vision*, pp. 494–503, 2021.

[68] S. Schneider *et al.*, "Improving Robustness against Common Corruptions by Covariate Shift Adaptation," in *Proc. Advances in Neural Information Processing Systems*, pp. 11539–11551, 2020.

[69] S. Cygert and A. Czyżewski, "Robustness in Compressed Neural Networks for Object Detection," in *Int. Joint Conf. on Neural Networks*, pp. 1–8, 2021.

[70] J. Dapello *et al.*, "Simulating a Primary Visual Cortex at the Front of CNNs Improves Robustness to Image Perturbations," in *Proc. Advances in Neural Information Processing Systems*, pp. 13073–13087, 2020.

[71] Y. Cai *et al.*, "Retinexformer: One-stage Retinex-based Transformer for Low-light Image Enhancement," in *Proc. IEEE Int. Conf. Computer Vision*, 2023.

[72] Z. Wang *et al.*, "Uformer: A general u-shaped transformer for image restoration," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 17683–17693, 2022.

[73] T. Son *et al.*, "URIE: Universal Image Enhancement for Visual Recognition in the Wild," in *Proc. European Conf. Computer Vision*, pp. 749–765, 2020.

[74] M. Suganuma, X. Liu, and T. Okatani, "Attention-Based Adaptive Selection of Operations for Image Restoration in the Presence of Unknown Combined Distortions," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 9039–9048, 2019.

[75] D. Liu *et al.*, "When Image Denoising Meets High-Level Vision Tasks: A Deep Learning Approach," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 2018.

[76] J. C. Galeano, A. Veloza-Suan, and F. A. González, "A Comparative Analysis of Artificial Immune Network Models," in *Proceedings of the 7th Annual Conf. on Genetic and Evolutionary Computation*, pp. 361–368, 2005.

[77] Z. Ji and D. Dasgupta, "Revisiting Negative Selection Algorithms," *Evolutionary computation*, vol. 15, pp. 223–51, 2007.

[78] E. D. Ülker, "Gene Selection in Microarray Data Using an Improved Approach of CLONALG," in *Proc. of the Int. Conf. on Artificial Intelligence and Applied Mathematics in Engineering*, pp. 466–472, 2020.

[79] L. Li, Q. Lin, and Z. Ming, "A Survey of Artificial Immune Algorithms for Multi-Objective Optimization," *Neurocomputing*, vol. 489, pp. 211–229, 2022.

[80] J. F. García-Mejía *et al.*, "A Clonal Selection Algorithm for the design of an Optimal Investment Portfolio," in *IEEE Int. Summer Power Meeting/Int. Meeting on Communications and Computing (RVP-AI/ROC&C)*, pp. 1–6, 2021.

[81] D. Dasgupta, *Artificial Immune Systems and their Applications*. Springer Science & Business Media, 2012.

[82] J. Zheng, Y. Chen, and W. Zhang, "A Survey of Artificial Immune Applications," *Artificial Intelligence Review*, vol. 34, pp. 19–34, 2010.

[83] Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[84] J. Brown, M. Anwar, and G. Dozier, "An Artificial Immunity Approach to Malware Detection in a Mobile Platform," *EURASIP Journal on Information Security*, vol. 2017, pp. 1–10, 2017.

[85] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer using Convolutional Neural Networks," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 2414–2423, 2016.

[86] I. Goodfellow *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information processing Systems*, 2014.

[87] T. Salimans *et al.*, "Improved Techniques for Training GANs," in *Proc. Advances in Neural Information Processing Systems*, 2016.

[88] X. Han, "MR-based synthetic CT Generation using a Deep Convolutional Neural Network Method," *Medical Physics*, vol. 44, no. 4, pp. 1408–1419, 2017.

[89] T. Kossen *et al.*, "Synthesizing Anonymized and Labeled TOF-MRA patches for brain vessel Segmentation using Generative Adversarial Networks," *Computers in Biology and Medicine*, vol. 131, p. 104254, 2021.

[90] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[91] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation Measures for Models assessment over Imbalanced Data Sets," *Journal of Information Engineering and Applications*, vol. 3, no. 10, 2013.

[92] R. Geirhos *et al.*, "ImageNet-trained CNNs are biased Towards Texture; Increasing Shape bias Improves Accuracy and Robustness," in *Proc. Int. Conf. on Learning Representations*, 2019.

[93] W. Xu *et al.*, "Robust and Accurate Object Detection Via Self-Knowledge Distillation," in *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 91–95, 2022.

[94] M. Everingham *et al.*, "The PASCAL Visual Object Classes (VOC) challenge," *Int. J. Computer Vision*, vol. 88, pp. 303–338, 2010.

[95] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, pp. 161–187, 2017.

[96] A. J. Izenman, "Review papers: Recent developments in nonparametric density estimation," *Journal of the american statistical association*, vol. 86, pp. 205–224, 1991.

[97] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, pp. 1065–1076, 1962.

[98] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Proc. European Conf. Computer Vision*, pp. 740–755, 2014.

[99] S. Sahoo and P. K. Nanda, "Adaptive Feature Fusion and Spatio-Temporal Background Modeling in KDE Framework for Object Detection and Shadow Removal," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 32, pp. 1103–1118, 2021.

[100] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[101] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "MixUp: Beyond empirical risk minimization," in *Proc. Int. Conf. on Learning Representations*, 2018.

[102] D. Hendrycks *et al.*, "AugMix: A Simple Method to Improve Robustness and Uncertainty under Data Shift," in *Proc. Int. Conf. on Learning Representations*, 2020.

[103] L. Huang, X. Zhao, and K. Huang, "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 43, no. 5, pp. 1562–1577, 2019.

[104] M. Muller *et al.*, "TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild," in *Proc. European Conf. Computer Vision*, pp. 300–317, 2018.

[105] H. Fan *et al.*, "LaSOT: A High-Quality Large-Scale Single Object Tracking Benchmark," *Int. J. Computer Vision*, vol. 129, pp. 439–461, 2021.

[106] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[107] F. Yu *et al.*, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 2636–2645, 2020.

[108] D. Wu, B. Sun, and M. Shang, "Hyperparameter learning for deep learning-based recommender systems," *Transactions on Services Computing*, 2023.

[109] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization," in *Congress on Evolutionary Computation (CEC)*, pp. 1551–1559, IEEE, 2021.

[110] L. Li *et al.*, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1–52, 2018.

[111] J. Bergstra and Y. Bengio, "Random Search for Hyper-parameter Optimization," *Journal of machine learning research*, vol. 13, 2012.

[112] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[113] T. W. MacFarland, J. M. Yates, *et al.*, *Introduction to nonparametric statistics for the biological sciences using R*. Springer, 2016.

[114] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of Statistical Learning: Data Mining, Inference, and Prediction*, vol. 2. Springer, 2009.

[115] M. Fawakherji *et al.*, "TextAug: Test time Text Augmentation for Multimodal Person Re-identification," in *IEEE Winter Conf. App. Computer Vision*, 2024.

[116] M. S. Ayhan and P. Berens, "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks," in *Medical Imaging with Deep Learning*, 2022.

[117] J. Nalepa, M. Myller, and M. Kawulok, "Training-and test-time data augmentation for hyperspectral image segmentation," *Geoscience and Remote Sensing Letters*, vol. 17, 2019.

[118] I. J. Goodfellow *et al.*, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[119] Y. Chang *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, pp. 1–45, 2024.

[120] H. Zhao *et al.*, "Explainability for large language models: A survey," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, pp. 1–38, 2024.

[121] S. Mangrulkar *et al.*, "PEFT: State-of-the-art parameter-efficient fine-tuning methods." `https://github.com/huggingface/peft`, 2022.

[122] B. Lester, R. Al-Rfou, and N. Constant, "The Power of Scale for Parameter-Efficient Prompt Tuning," in *Empirical Methods in Natural Language Processing*, 2021.

[123] S. Lee, H. Lee, and S. Yoon, "Adversarial Vertex Mixup: Toward better Adversarially Robust Generalization," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 272–281, 2020.

[124] S. Gowal *et al.*, "Achieving Robustness in the Wild via Adversarial Mixing with Disentangled Representations," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 1211–1220, 2020.

[125] H. Zhang and J. Wang, "Defense against Adversarial Attacks using Feature Scattering-Based Adversarial Training," in *Proc. Advances in Neural Information Processing Systems*, vol. 32, 2019.

[126] J. Rony *et al.*, "Decoupling Direction and Norm for Efficient Gradient-Based l2 Adversarial Attacks and Defenses," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4322–4330, 2019.

[127] A. Kumar *et al.*, "Adv-Cut Paste: Semantic Adversarial Class Specific Data Augmentation Technique for Object Detection," in *Proc. IEEE Int. Conf. on Pattern Recognition (ICPR)*, pp. 3632–3638, 2022.

[128] A. Jeddi *et al.*, "Learn2Perturb: An End-to-End Feature Perturbation Learning to Improve Adversarial Robustness," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 1241–1250, 2020.

[129] S. Gopalakrishnan *et al.*, "Combating Adversarial Attacks using Sparse Representations," in *Proc. Int. Conf. on Learning Representations*, 2018.

[130] D. Li, J. Zhang, and K. Huang, "Universal Adversarial Perturbations against Object Detection," *Pattern Recognition*, vol. 110, p. 107584, 2021.

Appendix A

Detailed breakdown of each distortion

We provide a detailed breakdown of the performance of our method and related works here using YOLOv4 for both PASCAL and COCO. We show the accuracy of each of the 15 distortions and their corresponding severity level from 1 to 5, with level 1 being the lowest and five the highest severity. As shown in Table A.1 and A.2, the results affirm our analysis in Section 4. Our methods are more effective for noise-like distortions (e.g., impulse noise) than for weather-related distortions, as exposed mainly under PASCAL. This difference in performance might be due to the pixel-wise distortion approach used in both methods, which is better suited to handling localized, random noise rather than more uniform or structured distortions like those caused by weather conditions. To address this imbalance and make our methods more general under different datasets and distortions, exploring a global distortion approach, such as block-wise distortion, could be beneficial. Implementing this change might help make the improvements offered by the methods more consistent across different types of distortions.

Table A.1: PASCAL validation set: Detailed comparison of our methods (GSES and AISbod) with related works on YOLOv4. The best results are in red and second in blue in each row; the +() indicates gain, that is, the difference between the original and defence method.

		YOLOv4	+ GSES (our)	+ AISbod (our)	+ Stylize	+ SMIX	+ URIE
	clean	83.31	83.32 (+0.01)	83.50 (+0.19)	83.90 (+0.59)	83.65 (+0.34)	
Gaussian noise	severity 1	74.06	79.64 (+5.58)	78.69 (+4.63)	76.41 (+2.35)	73.86 (-0.20)	68.56 (-5.50)
	severity 2	66.34	77.64 (+11.30)	76.55 (+10.21)	70.47 (+4.13)	64.43 (-1.91)	64.88 (-1.46)
	severity 3	52.83	73.10 (+20.27)	72.49 (+19.66)	60.74 (+7.91)	47.08 (-5.75)	59.13 (+6.30)
	severity 4	34.37	68.52 (+34.15)	65.93 (+31.56)	46.90 (+12.53)	27.19 (-7.18)	52.44 (+18.07)
	severity 5	14.62	58.47 (+43.85)	55.82 (+41.20)	27.33 (+12.71)	12.00 (-2.62)	41.04 (+26.42)
shot noise	severity 1	74.89	80.16 (+5.27)	79.07 (+4.18)	76.92 (+2.03)	74.70 (-0.19)	69.24 (-5.65)
	severity 2	66.98	78.12 (+11.14)	77.01 (+10.03)	69.83 (+2.85)	64.27 (-2.71)	65.60 (-1.38)
	severity 3	56.36	74.66 (+18.30)	73.98 (+17.62)	62.12 (+5.76)	50.60 (-5.76)	60.59 (+4.23)
	severity 4	35.69	68.31 (+32.62)	65.87 (+30.18)	46.69 (+11.00)	28.09 (-7.60)	52.15 (+16.46)
	severity 5	22.34	61.60 (+39.26)	58.79 (+36.45)	33.86 (+11.52)	16.86 (-5.48)	45.89 (+23.55)
impulse noise	severity 1	61.09	80.30 (+19.21)	76.38 (+15.29)	70.35 (+9.26)	63.55 (+2.46)	66.64 (+5.55)
	severity 2	54.40	78.73 (+24.33)	75.24 (+20.84)	66.63 (+12.23)	57.27 (+2.87)	62.58 (+8.18)
	severity 3	48.48	75.87 (+27.39)	73.56 (+25.08)	61.73 (+13.25)	48.77 (+0.29)	58.99 (+10.51)
	severity 4	31.68	70.94 (+39.26)	67.76 (+36.08)	46.50 (+14.82)	25.92 (-5.76)	51.12 (+19.44)
	severity 5	15.12	64.58 (+49.46)	59.24 (+44.12)	28.22 (+13.10)	10.93 (-4.19)	41.50 (+26.38)
motion blur	severity 1	70.50	71.45 (+0.95)	72.16 (+1.66)	70.64 (+0.14)	70.42 (-0.08)	65.65 (-4.85)
	severity 2	57.25	59.51 (+2.26)	60.62 (+3.37)	60.36 (+3.11)	57.08 (-0.17)	61.39 (+4.14)
	severity 3	37.72	42.11 (+4.39)	42.62 (+4.90)	45.91 (+8.19)	39.88 (+2.16)	54.03 (+16.31)
	severity 4	22.87	25.94 (+3.07)	26.56 (+3.69)	30.72 (+7.85)	26.69 (+3.82)	45.32 (+22.45)
	severity 5	17.13	18.47 (+1.34)	19.32 (+2.19)	23.72 (+6.59)	19.97 (+2.84)	39.87 (+22.74)
zoom blur	severity 1	53.81	54.78 (+0.97)	55.22 (+1.41)	57.32 (+3.51)	53.75 (-0.06)	55.96 (+2.15)
	severity 2	43.78	44.06 (+0.28)	44.99 (+1.21)	46.77 (+2.99)	43.71 (-0.07)	50.74 (+6.96)
	severity 3	37.72	40.02 (+2.30)	38.23 (+0.51)	43.13 (+5.41)	39.69 (+1.97)	45.21 (+7.49)
	severity 4	30.22	31.72 (+1.50)	30.23 (+0.01)	34.34 (+4.12)	32.76 (+2.54)	40.39 (+10.17)

Table A.1: (continued)

		YOLOv4	+ GSES (our)	+ AISbod (our)	+ Stylize	+ SMIX	+ URIE
	severity 5	24.92	26.33 (+1.41)	23.91 (-1.01)	28.83 (+3.91)	28.62 (+3.70)	35.02 (+10.10)
glass blur	severity 1	69.08	74.20 (+5.12)	71.01 (+1.93)	71.82 (+2.74)	68.98 (-0.10)	63.89 (-5.19)
	severity 2	55.68	67.95 (+12.27)	59.46 (+3.78)	63.44 (+7.76)	55.74 (+0.06)	58.55 (+2.87)
	severity 3	20.12	54.20 (+34.08)	25.14 (+5.02)	36.61 (+16.49)	21.26 (+1.14)	45.68 (+25.56)
	severity 4	15.32	47.25 (+31.93)	18.74 (+3.42)	29.13 (+13.81)	14.58 (-0.74)	39.85 (+24.53)
	severity 5	11.61	32.24 (+20.63)	12.99 (+1.38)	26.16 (+14.55)	11.32 (-0.29)	31.31 (+19.70)
defocus blur	severity 1	70.21	70.42 (+0.21)	70.81 (+0.60)	71.00 (+0.79)	67.18 (-3.03)	62.53 (-7.68)
	severity 2	61.94	63.68 (+1.74)	63.83 (+1.89)	64.16 (+2.22)	56.13 (-5.81)	58.54 (-3.40)
	severity 3	47.84	49.57 (+1.73)	51.12 (+3.28)	52.74 (+4.90)	41.67 (-6.17)	49.12 (+1.28)
	severity 4	36.33	36.59 (+0.26)	39.46 (+3.13)	42.65 (+6.32)	29.88 (-6.45)	40.09 (+3.76)
	severity 5	26.53	26.02 (-0.51)	28.67 (+2.14)	33.49 (+6.96)	21.59 (-4.94)	34.18 (+7.65)
contrast	severity 1	78.89	79.52 (+0.63)	79.11 (+0.22)	81.09 (+2.20)	80.59 (+1.70)	73.75 (-5.14)
	severity 2	77.67	76.40 (-1.27)	77.29 (-0.38)	79.93 (+2.26)	79.70 (+2.03)	73.56 (-4.11)
	severity 3	73.69	71.66 (-2.03)	74.08 (+0.39)	77.55 (+3.86)	78.04 (+4.35)	72.97 (-0.72)
	severity 4	63.15	60.11 (-3.04)	63.69 (+0.54)	72.23 (+9.08)	72.60 (+9.45)	71.08 (+7.93)
	severity 5	41.18	43.02 (+1.84)	42.32 (+1.14)	63.27 (+22.09)	61.66 (+20.48)	66.40 (+25.22)
jpeg compression	severity 1	72.71	77.86 (+5.15)	76.92 (+4.21)	76.56 (+3.85)	76.11 (+3.40)	68.52 (-4.19)
	severity 2	63.79	75.00 (+11.21)	72.86 (+9.07)	72.17 (+8.38)	69.63 (+5.84)	66.88 (+3.09)
	severity 3	55.76	73.12 (+17.36)	69.38 (+13.62)	69.54 (+13.78)	64.03 (+8.27)	65.38 (+9.62)
	severity 4	35.47	66.32 (+30.85)	54.17 (+18.70)	59.04 (+23.57)	46.41 (+10.94)	60.75 (+25.28)
	severity 5	22.08	56.16 (+34.08)	35.12 (+13.04)	44.02 (+21.94)	30.45 (+8.37)	54.92 (+32.84)
pixelate	severity 1	75.02	80.88 (+5.86)	77.95 (+2.93)	79.95 (+4.93)	74.93 (-0.09)	70.47 (-4.55)
	severity 2	69.89	80.15 (+10.26)	74.67 (+4.78)	77.34 (+7.45)	70.03 (+0.14)	69.56 (-0.33)
	severity 3	31.76	74.31 (+42.55)	43.94 (+12.18)	68.42 (+36.66)	33.51 (+1.75)	65.71 (+33.95)
	severity 4	12.02	51.64 (+39.62)	15.95 (+3.93)	50.15 (+38.13)	12.78 (+0.76)	61.09 (+49.07)
	severity 5	5.82	28.67 (+22.85)	7.85 (+2.03)	31.90 (+26.08)	5.49 (-0.33)	55.35 (+49.53)
elastic transform	severity 1	73.23	74.67 (+1.44)	73.62 (+0.39)	78.62 (+5.39)	71.91 (-1.32)	64.04 (-9.19)
	severity 2	66.21	68.10 (+1.89)	66.59 (+0.38)	73.35 (+7.14)	62.93 (-3.28)	59.65 (-6.56)
	severity 3	53.11	56.08 (+2.97)	53.96 (+0.85)	65.29 (+12.18)	47.44 (-5.67)	51.99 (-1.12)
	severity 4	42.41	47.48 (+5.07)	43.78 (+1.37)	57.94 (+15.53)	36.39 (-6.02)	47.30 (+4.89)
	severity 5	30.18	36.34 (+6.16)	30.88 (+0.70)	45.76 (+15.58)	24.17 (-6.01)	40.20 (+10.02)
frost	severity 1	76.27	75.49 (-0.78)	76.03 (-0.24)	76.77 (+0.50)	77.51 (+1.24)	65.38 (-10.89)
	severity 2	66.79	65.24 (-1.55)	66.45 (-0.34)	68.90 (+2.11)	70.09 (+3.30)	59.03 (-7.76)
	severity 3	60.64	58.05 (-2.59)	59.99 (-0.65)	63.25 (+2.61)	64.67 (+4.03)	53.60 (-7.04)
	severity 4	58.76	56.14 (-2.62)	58.36 (-0.40)	61.78 (+3.02)	62.61 (+3.85)	53.10 (-5.66)
	severity 5	53.10	49.75 (-3.35)	52.24 (-0.86)	57.04 (+3.94)	58.69 (+5.59)	49.28 (-3.82)
fog	severity 1	78.53	78.28 (-0.25)	78.51 (-0.02)	81.10 (+2.57)	80.47 (+1.94)	71.58 (-6.95)
	severity 2	77.43	76.33 (-1.10)	77.39 (-0.04)	80.58 (+3.15)	79.99 (+2.56)	70.79 (-6.64)
	severity 3	75.76	74.83 (-0.93)	76.02 (+0.26)	80.00 (+4.24)	79.22 (+3.46)	69.57 (-6.19)
	severity 4	75.00	73.03 (-1.97)	74.57 (-0.43)	78.97 (+3.97)	78.54 (+3.54)	68.34 (-6.66)
	severity 5	70.98	67.96 (-3.02)	69.95 (-1.03)	75.79 (+4.81)	76.04 (+5.06)	64.47 (-6.51)
snow	severity 1	72.13	68.22 (-3.91)	70.55 (-1.58)	73.68 (+1.55)	70.11 (-2.02)	63.34 (-8.79)
	severity 2	59.74	59.33 (-0.41)	61.25 (+1.51)	62.65 (+2.91)	59.34 (-0.40)	59.23 (-0.51)
	severity 3	57.37	54.06 (-3.31)	56.88 (-0.49)	61.69 (+4.32)	57.48 (+0.11)	57.39 (+0.02)
	severity 4	47.90	45.59 (-2.31)	46.70 (-1.20)	54.16 (+6.26)	48.65 (+0.75)	50.73 (+2.83)
	severity 5	46.64	46.61 (-0.03)	48.78 (+2.14)	51.48 (+4.84)	49.98 (+3.34)	53.79 (+7.15)
brightness	severity 1	81.39	81.80 (+0.41)	80.98 (-0.41)	82.24 (+0.85)	81.98 (+0.59)	72.48 (-8.91)
	severity 2	80.01	81.27 (+1.26)	80.42 (+0.41)	81.53 (+1.52)	81.26 (+1.25)	71.61 (-8.40)
	severity 3	79.27	80.43 (+1.16)	79.70 (+0.43)	80.93 (+1.66)	80.54 (+1.27)	70.08 (-9.19)
	severity 4	78.12	78.31 (+0.19)	78.40 (+0.28)	79.92 (+1.80)	79.16 (+1.04)	67.80 (-10.32)
	severity 5	76.11	75.92 (-0.19)	76.83 (+0.72)	78.42 (+2.31)	76.52 (+0.41)	64.99 (-11.12)
Overall summary	mPC	52.48	62.02 (+9.54)	58.87 (+6.40)	60.44 (+7.96)	53.04 (+0.56)	58.05 (+5.57)
	rPC	0.6299	0.7444	0.7067	0.7254	0.6367	0.6968

Table A.2: COCO validation set: Detailed comparison of our methods (GSES and AISbod) with related works on YOLOv4. The best results are in red and second in blue in each row; the +() indicates gain, that is, the difference between the original and defence method.

		YOLOv4	+ GSES (our)	+ AISbod (our)	+ Stylize	+ URIE
	clean	40.67	40.81 (+0.14)	40.87 (+0.20)	38.95 (-1.73)	
Gaussian noise	severity 1	35.02	47.57 (+12.55)	37.85 (+2.83)	34.55 (-0.47)	30.55 (-4.47)
	severity 2	30.20	36.99 (+6.79)	36.85 (+6.65)	31.04 (+0.83)	28.32 (-1.88)
	severity 3	22.72	34.66 (+11.94)	35.51 (+12.79)	26.47 (+3.75)	24.96 (+2.24)
	severity 4	14.35	38.06 (+23.71)	31.29 (+16.94)	20.09 (+5.74)	20.60 (+6.25)
	severity 5	5.56	18.15 (+12.59)	24.51 (+18.95)	11.55 (+5.99)	15.49 (+9.93)
shot noise	severity 1	35.00	46.57 (+11.57)	38.03 (+3.03)	34.49 (-0.51)	30.49 (-4.51)
	severity 2	29.68	37.37 (+7.69)	35.90 (+6.22)	30.73 (+1.05)	28.12 (-1.56)
	severity 3	23.29	35.43 (+12.14)	32.64 (+9.35)	26.68 (+3.39)	25.38 (+2.09)
	severity 4	13.73	35.52 (+21.79)	25.30 (+11.57)	19.21 (+5.48)	20.40 (+6.67)
	severity 5	8.00	22.77 (+14.77)	18.74 (+10.74)	14.01 (+6.01)	16.95 (+8.95)
impulse noise	severity 1	26.03	49.70 (+23.67)	39.34 (+13.31)	31.47 (+5.44)	29.41 (+3.39)
	severity 2	23.49	47.08 (+23.59)	38.44 (+14.95)	29.03 (+5.54)	27.01 (+3.53)
	severity 3	20.98	48.84 (+27.86)	37.68 (+16.70)	26.74 (+5.76)	24.77 (+3.79)
	severity 4	13.08	36.33 (+23.25)	35.26 (+22.19)	19.77 (+6.70)	20.12 (+7.04)
	severity 5	5.79	32.85 (+27.06)	30.98 (+25.19)	12.50 (+6.71)	15.45 (+9.66)
motion blur	severity 1	33.71	42.29 (+8.58)	33.98 (+0.26)	32.43 (-1.29)	29.12 (-4.59)
	severity 2	26.46	28.66 (+2.20)	27.86 (+1.40)	27.48 (+1.02)	26.24 (-0.23)
	severity 3	17.70	23.72 (+6.02)	19.95 (+2.25)	21.17 (+3.47)	22.43 (+4.73)
	severity 4	10.06	15.44 (+5.38)	12.27 (+2.20)	14.93 (+4.87)	17.73 (+7.67)
	severity 5	6.49	15.14 (+8.65)	8.60 (+2.11)	11.51 (+5.02)	14.93 (+8.44)
zoom blur	severity 1	16.13	19.71 (+3.58)	17.45 (+1.32)	18.06 (+1.93)	16.84 (+0.72)
	severity 2	10.60	22.03 (+11.43)	11.66 (+1.06)	12.52 (+1.91)	12.83 (+2.23)
	severity 3	7.81	12.10 (+4.29)	8.61 (+0.80)	9.90 (+2.09)	10.26 (+2.45)
	severity 4	5.37	7.96 (+2.59)	6.11 (+0.74)	7.13 (+1.76)	8.01 (+2.64)
	severity 5	4.12	5.47 (+1.35)	4.86 (+0.74)	5.77 (+1.65)	6.70 (+2.58)
glass blur	severity 1	33.19	41.59 (+8.40)	34.82 (+1.63)	33.38 (+0.19)	28.55 (-4.64)
	severity 2	26.91	40.96 (+14.05)	30.12 (+3.20)	29.49 (+2.57)	25.76 (-1.16)
	severity 3	10.61	35.4 (+24.79)	15.48 (+4.87)	17.29 (+6.68)	19.20 (+8.59)
	severity 4	8.04	28.31 (+20.27)	12.29 (+4.26)	14.41 (+6.37)	16.90 (+8.87)
	severity 5	5.53	16.22 (+10.69)	8.74 (+3.21)	10.77 (+5.24)	12.59 (+7.06)
defocus blur	severity 1	33.98	42.89 (+8.91)	35.04 (+1.06)	33.53 (-0.45)	27.65 (-6.32)
	severity 2	29.79	36.37 (+6.58)	31.34 (+1.55)	30.39 (+0.59)	25.51 (-4.28)
	severity 3	22.56	32.01 (+9.45)	24.87 (+2.32)	24.67 (+2.12)	21.11 (-1.44)
	severity 4	16.50	25.97 (+9.47)	19.63 (+3.12)	19.75 (+3.25)	17.01 (+0.51)
	severity 5	11.27	16.45 (+5.18)	14.78 (+3.51)	15.65 (+4.37)	13.24 (+1.97)
contrast	severity 1	38.71	45.6 (+6.89)	38.87 (+0.16)	37.78 (-0.93)	34.13 (-4.58)
	severity 2	37.54	45.31 (+7.77)	37.81 (+0.26)	37.11 (-0.44)	34.09 (-3.46)
	severity 3	35.14	39.49 (+4.35)	35.46 (+0.33)	35.75 (+0.61)	33.73 (-1.41)
	severity 4	28.65	33.92 (+5.27)	29.11 (+0.46)	32.23 (+3.58)	32.67 (+4.02)
	severity 5	17.80	19.21 (+1.41)	19.12 (+1.32)	26.21 (+8.41)	29.71 (+11.91)
jpeg compression	severity 1	34.02	45.53 (+11.51)	35.61 (+1.59)	33.82 (-0.20)	30.34 (-3.69)
	severity 2	29.07	43.57 (+14.50)	31.76 (+2.69)	30.67 (+1.60)	29.34 (+0.28)
	severity 3	25.47	30.53 (+5.06)	29.07 (+3.60)	28.39 (+2.92)	28.58 (+3.11)
	severity 4	15.90	22.93 (+7.03)	20.31 (+4.42)	21.17 (+5.28)	26.14 (+10.24)
	severity 5	8.60	13.1 (+4.50)	12.28 (+3.69)	14.66 (+6.07)	22.64 (+14.04)
pixelate	severity 1	37.45	48.39 (+10.94)	38.44 (+0.99)	36.90 (-0.55)	31.83 (-5.62)
	severity 2	35.60	48.50 (+12.90)	36.98 (+1.38)	36.12 (+0.52)	31.67 (-3.93)
	severity 3	22.01	29.8 (+7.79)	26.08 (+4.07)	31.14 (+9.13)	30.00 (+7.99)
	severity 4	8.04	18.67 (+10.63)	11.40 (+3.37)	24.41 (+16.37)	28.04 (+20.01)
	severity 5	3.24	6.10 (+2.86)	5.28 (+2.04)	18.25 (+15.01)	25.80 (+22.56)
elastic transform	severity 1	34.25	42.11 (+7.86)	35.02 (+0.76)	35.54 (+1.28)	28.06 (-6.20)
	severity 2	29.90	37.60 (+7.70)	31.27 (+1.36)	32.82 (+2.91)	25.29 (-4.62)
	severity 3	23.55	48.39 (+24.84)	25.36 (+1.81)	28.02 (+4.48)	21.51 (-2.04)
	severity 4	19.48	34.05 (+14.57)	21.57 (+2.09)	24.34 (+4.86)	18.88 (-0.61)
	severity 5	14.19	37.76 (+23.57)	16.50 (+2.30)	19.39 (+5.19)	15.76 (+1.57)
frost	severity 1	35.73	29.56 (-6.17)	36.03 (+0.31)	35.18 (-0.55)	29.46 (-6.27)
	severity 2	30.38	28.04 (-2.34)	31.32 (+0.94)	30.98 (+0.60)	25.13 (-5.25)
	severity 3	26.45	28.67 (+2.22)	27.36 (+0.91)	28.01 (+1.55)	22.11 (-4.35)
	severity 4	25.26	26.40 (+1.14)	26.25 (+1.00)	27.12 (+1.86)	21.31 (-3.94)
	severity 5	22.98	24.50 (+1.52)	24.30 (+1.32)	25.46 (+2.48)	19.37 (-3.61)
fog	severity 1	38.40	50.95 (+12.55)	38.68 (+0.28)	37.61 (-0.79)	33.16 (-5.24)
	severity 2	37.69	51.15 (+13.46)	37.94 (+0.25)	37.14 (-0.55)	32.91 (-4.78)

Table A.2: (continued)

		YOLOv4	+ GSES (our)	+ AISbod (our)	+ Stylize	+ URIE
	severity 3	6.71	44.59 (+7.88)	37.10 (+0.40)	36.62 (-0.09)	32.20 (-4.50)
	severity 4	36.25	42.11 (+5.86)	36.82 (+0.57)	36.48 (+0.23)	31.89 (-4.36)
	severity 5	34.55	37.60 (+3.05)	35.07 (+0.52)	35.37 (+0.82)	30.51 (-4.04)
snow	severity 1	30.48	48.39 (+17.91)	32.93 (+2.46)	32.63 (+2.16)	27.28 (-3.20)
	severity 2	23.57	44.59 (+21.02)	27.21 (+3.64)	27.99 (+4.43)	23.78 (+0.22)
	severity 3	22.35	42.11 (+19.76)	24.37 (+2.02)	27.20 (+4.85)	22.57 (+0.23)
	severity 4	17.54	37.60 (+20.06)	19.60 (+2.06)	23.53 (+5.99)	18.32 (+0.78)
	severity 5	17.85	37.76 (+19.91)	20.30 (+2.46)	23.22 (+5.37)	19.57 (+1.73)
brightness	severity 1	40.38	50.95 (+10.57)	40.45 (+0.06)	38.63 (-1.75)	33.46 (-6.93)
	severity 2	39.64	51.15 (+11.51)	39.71 (+0.07)	38.01 (-1.63)	33.32 (-6.32)
	severity 3	38.77	44.59 (+5.82)	38.95 (+0.19)	37.44 (-1.33)	32.72 (-6.05)
	severity 4	37.54	42.11 (+4.57)	37.90 (+0.36)	36.61 (-0.94)	31.49 (-6.06)
	severity 5	36.14	37.60 (+1.46)	36.83 (+0.69)	35.54 (-0.60)	29.74 (-6.40)
Overall summary	mPC	23.61	32.81 (+9.20)	27.51 (+3.90)	26.59 (+2.97)	24.52 (+0.91)
	rPC	0.5806	0.8067	0.6764	0.6537	0.6029

Appendix B

Adversarial attack defences

Adversarial attacks like FGSM, PGD, and DAG are crafted to fool CNN models. Hence, the deployment of the models in sensitive applications is limited. As a result, a significant amount of work has been done to mitigate this issue. First, we review adversarial attack defence methods and then give results showing their performance under image distortions.

Image-domain: The generalization of methods to several attacks is a significant issue in addressing adversarial attacks. Lee *et al.* [123] tackle this by generating adversarial examples online for classification models whose distribution has been extended in multidimensional space. When these examples are used in training, they increase the models' generalization to several adversarial attacks. Similarly, in [124], Gowal *et al.* use a StyleGAN to generate adversarial examples offline, with perturbations similar to real-world attacks to make classification models more robust. Also, in [125], Zhang *et al.* provides adversarial samples online for augmenting classification models in the latent space in an unsupervised way using feature scattering while in [126] they are produced online via gradient and norm decoupling of adversarial perturbations. Contrary to the methods above applied to image classification models, Zhang *et al.* [16] generate adversarial samples for object detection models online via a weighted combination of attacks in the classification and localization tasks. While in [127], Kumar *et al.* propose Adv-Cut Paste for object detection tasks. It uses a pre-trained semantic segmentation model to generate adversarial perturbations on the regions containing the target objects. It pastes them onto new backgrounds to create new training samples offline.

Network-domain: There are few existing works in this domain. However, in [128], Jeddi *et al.* present an end-to-end framework, Learn2Perturb, to incorporate perturbation modules into both the training and inference stages of an image classification model to make the model robust to adversarial attacks. Furthermore, because these modules are trainable, they can introduce the optimum uncertainty into the model's feature space. Moreover, in [129], Gopalakrishnan *et al.* show that linear models can be used to defend CNN models

against adversarial attacks. Finally, authors in [130] propose a novel form of attack called universal adversarial perturbation (UAP) for object detection models and possible direction for defending against UAP.

Hybrid-domain: Adversarial training is the primary method for improving robustness, but most work sacrifices accuracy under clean samples for such robustness. Xu *et al.* [93] present a novel fine-tuning paradigm that achieves superior performance over existing methods by investigating the combination of self-knowledge distillation and adversarial training for object detection. Dong *et al.* [14] propose a robust detector (RbDet) to disentangle the gradients for clean and adversarial images during training. This leads to adversarial robustness with less deterioration on clean samples. In [15], authors present a class-aware robust adversarial training technique that generates a universal adversarial perturbation in a given image. This method simultaneously attacks all objects occurring in a given training image by jointly maximizing their respective loss. However, the total loss per class is decomposed instead of normalizing the total loss by the number of objects. It normalizes the loss for each class using the number of objects. This equally improves the adversarial robustness of trained models for all object classes.

We compare our methods to adversarial attack defence methods MTD [16], RbDet [14], and CWAT [15] to show that these methods do not extend to 15 image distortions. Hence, we do not claim the effectiveness of our method against adversarial attacks. We use the same model and dataset (i.e., SSD [33] and PASCAL) as in MTD [16], RbDet [14], and CWAT [15]. As shown in Table B.1, we see a significant drop in all adversarial methods except ours.

Table B.1: PASCAL validation set: Comparison of our AISbod with adversarial defence methods against image distortion.

Model	SSD	GSES (our)	+ AISbod (our)	+ MTD	+ RbDet	+ CWAT
mPC	57.59	63.09 (+5.50)	62.31 (+4.72)	23.12 (-34.47)	48.58 (-9.01)	25.27 (-32.32)

Appendix C

Visual results

Subjectively, Fig. C.1 and C.2 show the accuracy of the Faster RCNN model for a Gaussian noise level of 27 dB and impulse density of 0.25% on some samples, respectively. The model is easily tricked by adding noise, as seen in the first row. Objects are wrongly classified, and detection is missed in others. These vulnerabilities are mitigated with our methods (as shown in their respective rows in the Figures).

Figure C.1: Comparison of Faster RCNN, Faster RCNN+GSES, and Faster RCNN+AISbod on 27 dB Gaussian noisy samples. For instance, the *Aeroplane* correctly detected in the original image (row 1, column 6) is missed in its noisy version (row 2, column 6). This error is corrected in our methods (rows 3 and 4, column 6). (Points of interest indicated by blue arrows are better viewed by zooming in).

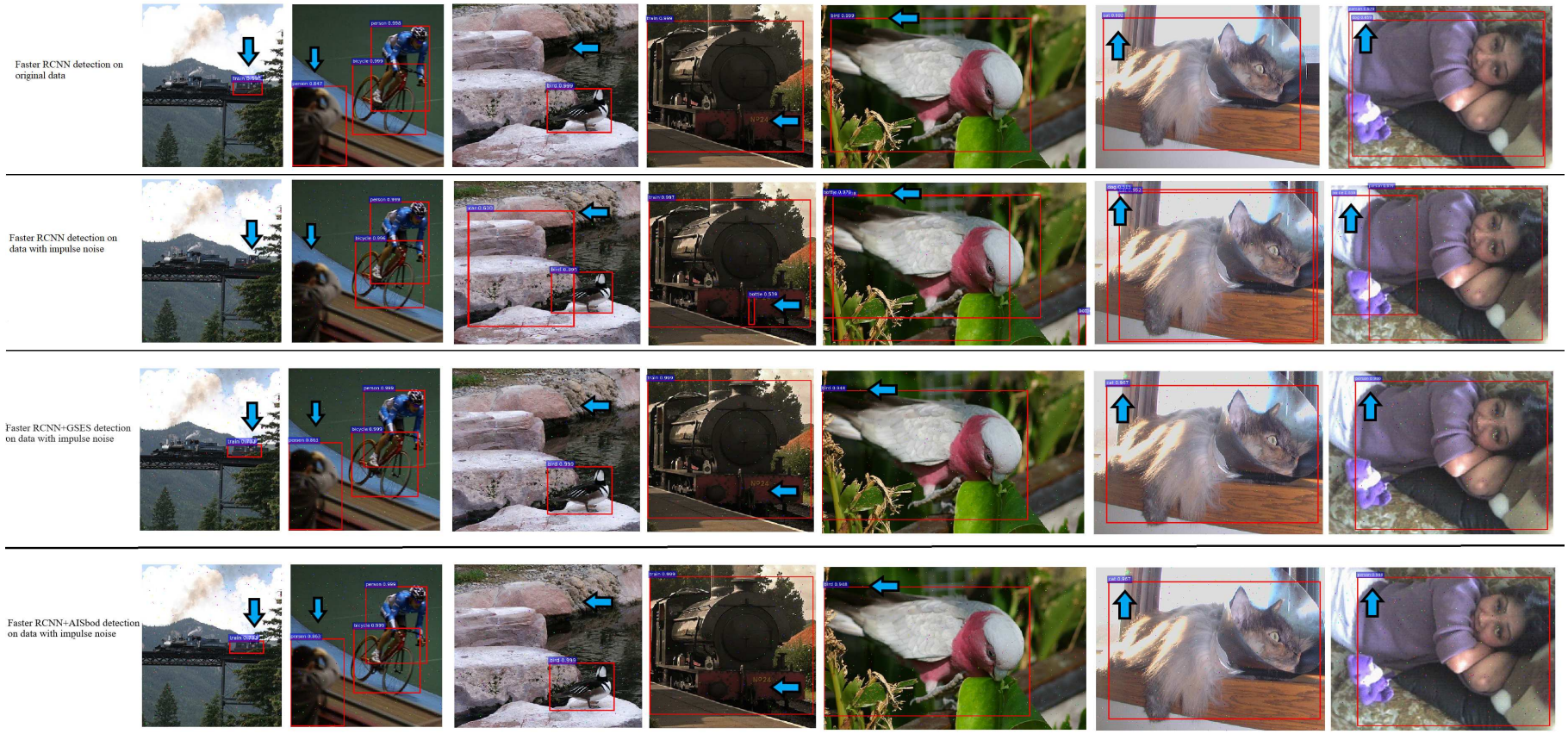

Figure C.2: Comparison of Faster RCNN Faster RCNN+GSES, and Faster RCNN+AISbod on 0.25% impulse noisy samples. For instance, the *Train* correctly detected in the original image (row 1, column 1) is missed in the noisy version (row 2, column 1). This error is corrected in GSES and AIS (rows 4 and 5, column 1). (Points of interest indicated by blue arrows are better viewed by zooming in).

Appendix D

Loss functions of DINO and YOLOv7

D.1 DINO

The overall loss function for DINO is a combination of classification loss, bounding box regression loss, and auxiliary matching loss. The loss function can be expressed as

$$L_{DINO} = L_{\text{cls}} + \lambda_{\text{box}} L_{\text{box}} + \lambda_{\text{giou}} L_{\text{giou}}, \tag{14}$$

where L_{cls} is the classification loss, L_{box} is the bounding box regression loss, L_{giou} is the generalized IoU (GIoU) loss, and λ_{box} and λ_{giou} are hyperparameters that balance the contributions of the different losses.

D.1.1 Detailed Components

- **Classification Loss (L_{cls})**: The classification loss measures how well the model predicts the correct class for each object. In DINO, it is calculated using cross-entropy loss as

$$L_{\text{cls}} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} \log(p_{ic}), \tag{15}$$

  where y_{ic} is the ground truth label for class c for the i-th object and p_{ic} is the predicted probability for class c.

- **Bounding Box Regression Loss (L_{box})**: The bounding box regression loss measures the difference between the predicted and ground truth bounding box coordinates. This

is calculated using a smooth L1 loss as

$$L_{\text{box}} = \sum_{i=1}^{N} \mathbf{1}_i^{\text{obj}} \left(|x_i - \hat{x}_i| + |y_i - \hat{y}_i| + |w_i - \hat{w}_i| + \left| h_i - \hat{h}_i \right| \right), \tag{16}$$

where (x_i, y_i, w_i, h_i) are the predicted bounding box coordinates, $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$ are the ground truth bounding box coordinates, and $\mathbf{1}_i^{\text{obj}}$ is an indicator function that denotes if the i-th prediction corresponds to an object.

- **Generalized IoU Loss (L_{giou})**: The generalized IoU (GIoU) loss is used to improve the quality of the bounding box predictions by considering the overlap between the predicted and ground truth boxes. It is expressed as

$$L_{\text{giou}} = \sum_{i=1}^{N} \mathbf{1}_i^{\text{obj}} \left(1 - \text{GIoU}(B_i, \hat{B}_i) \right), \tag{17}$$

where B_i is the predicted bounding box and $\hat{B}_i$ is the ground truth bounding box. The GIoU function calculates the generalized intersection over union between these boxes.

D.2 YOLOv7

The overall loss function for YOLOv7 combines classification loss, bounding box regression loss, and objectness loss. The loss function can be expressed as:

$$L_{YOLOv7} = L_{\text{cls}} + \lambda_{\text{box}} L_{\text{box}} + \lambda_{\text{obj}} L_{\text{obj}}, \tag{18}$$

where L_{cls} is the classification loss, L_{box} is the bounding box regression loss, L_{obj} is the objectness loss, and λ_{box} and λ_{obj} are hyperparameters that balance the contributions of the different losses.

D.2.1 Detailed Components

- **Classification Loss (L_{cls})**: The classification loss measures how well the model predicts the correct class for each object. In YOLOv7, it is calculated using the binary cross-entropy loss as

$$L_{\text{cls}} = -\sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbf{1}_{ij}^{\text{obj}} \sum_{c=1}^{C} \left[y_{ijc} \log(p_{ijc}) + (1 - y_{ijc}) \log(1 - p_{ijc}) \right], \tag{19}$$

where y_{ijc} is the ground truth label for class c for the j-th bounding box in cell i, and p_{ijc} is the predicted probability for class c.

- **Objectness Loss ($L_{\mathbf{obj}}$)**: The objectness loss measures the confidence score for object presence in each bounding box. YOLOv7 uses binary cross-entropy for this part, which is given by

$$L_{\text{obj}} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \left[\mathbf{1}_{ij}^{\text{obj}} \left(C_{ij} - \hat{C}_{ij} \right)^2 + \mathbf{1}_{ij}^{\text{noobj}} \left(C_{ij} - \hat{C}_{ij} \right)^2 \right]. \tag{20}$$

  C_{ij} is the predicted objectness score and $\hat{C}_{ij}$ is the ground truth objectness score. The indicator functions $\mathbf{1}_{ij}^{\text{obj}}$ and $\mathbf{1}_{ij}^{\text{noobj}}$ indicate if the bounding box contains an object or not, respectively.

- **Bounding Box Regression Loss ($L_{\mathbf{box}}$)**: The bounding box regression loss measures the difference between the predicted and ground truth bounding box coordinates. In YOLOv7, it is calculated using a combination of mean squared error and IoU loss as

$$L_{\text{box}} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbf{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + \left(h_i - \hat{h}_i \right)^2 \right], \tag{21}$$

  where (x_i, y_i, w_i, h_i) are the predicted bounding box coordinates, $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$ are the ground truth bounding box coordinates, and $\mathbf{1}_{ij}^{\text{obj}}$ is an indicator function that denotes if the j-th bounding box in cell i is responsible for the prediction.