REPUBLIC OF TÜRKİYE
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
COMPUTER ENGINEERING DEPARTMENT

INVESTIGATION OF THE PERFORMANCE OF DEEP LEARNING-BASED OBJECT DETECTION SYSTEMS

BARIŞ DİNÇ

Ph.D.

ADANA 2025

REPUBLIC OF TÜRKİYE
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
COMPUTER ENGINEERING DEPARTMENT

INVESTIGATION OF THE PERFORMANCE OF DEEP LEARNING-BASED OBJECT DETECTION SYSTEMS

BARIŞ DİNÇ

Ph.D.

THESIS ADVISOR
ASSOC. PROF. DR. YASİN KAYA

ADANA, 2025

DECLARATION OF CONFORMITY

In this thesis study, which was prepared following the thesis writing rules of Adana Alparslan Türkeş Science and Technology University Institute of Graduate School, I declare that I provide all the information, documents, evaluations and results in accordance with scientific ethics and moral codes without resorting to any means or assistance that would be contrary to scientific ethics and traditions. I also declare that I refer to all of the articles I used in this study with appropriate references and accept all moral and legal consequences if a situation is found contrary to my statement regarding my work.

..../..../20..

[Signature]

Barış DİNÇ

ÖZET

DERİN ÖĞRENME TABANLI NESNE TESPİT SİSTEMLERİNİN PERFORMANSININ ARAŞTIRILMASI

Barış DİNÇ

Doktora, Bilgisayar MühendisliğiAnabilim Dalı

Danışman: Doç. Dr. Yasin KAYA

Temmuz 2025, 91 sayfa

Son yıllarda, derin öğrenme (DÖ) metodolojilerindeki hızlı gelişmeler nedeniyle bilgisayarla görme ile nesne algılama (NA) konusunda önemli ilerlemeler kaydedildi. Bununla birlikte, bölge tabanlı evrişimli sinir ağları (R-CNN'ler) çoğu gereksiz veya alakasız olan çok sayıda aday bölgenin üretilmesi nedeniyle genellikle hesaplama yetersizliklerinden musdariptir. Tek aşamalı algoritmalar NA'yı hızlandırmış olsa da, yüksek frekanslı gürültünün ve alakasız ayrıntıların varlığı bu modelleri arka plan düzensizliklerine karşı hassas hale getirir. Bu sınırlamaları gidermek için, bu çalışma nesne algılama işlemi için giriş görüntüsündeki yüksek frekanslı gürültüyü azaltan yeni bir yaklaşım sunmaktadır. Önerilen model özellikle İlgi Alanı (IA) üretimi ve nesne algılama arasında bir denge sağlayan yeni bir Uyarlanabilir Dolgu (UD) mekanizması ve İlgi Alanı dedektöründen oluşmaktadır. R-CNN kullanılarak beş kuş veriseti üzerinde yürütülen deneysel değerlendirmeler, yöntemimizin Kesişim Birleşim Oranı (IoU) değeri 0,5'ten büyük olan bölge önerilerinin oranını %20,93'ten %65,17'e çıkardığını göstermektedir. Ayrıca, pozitif aday bölgelerin sayısı eğitim sırasında %330 ve test sırasında %726 artarken, gereksiz öneriler %55,48 oranında azaltılmıştır. Önerilen model YOLOv8 için de kullanılmış ve 6 kuş türüne ait 351 görüntü ile test edilmiştir. Model mAp50'yi 0,954'ten 0,984'e ve mAP50-95'i 0,633'ten 0,781'e çıkarmıştır.

Anahtar Kelimeler: Nesne Tespiti, Görüntü İşleme, Viola ve Jones Algoritması, Uyarlanabilir Dolgu, Gauss Bulanıklaştırması

ABSTRACT

INVESTIGATION OF THE PERFORMANCE OF DEEP LEARNING-BASED OBJECT DETECTION SYSTEMS

Barış DİNÇ
Ph.D., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Yasin KAYA
July 2025, 91 pages

In recent years, significant advancements have been achieved in object detection (OD) within computer vision due to rapid developments in deep learning (DL) methodologies. Nevertheless, region-based convolutional neural networks (R-CNNs) often suffer from computational inefficiencies due to the generation of an excessive number of candidate regions, many of which are redundant or irrelevant. Although single-stage algorithms have accelerated OD, the presence of high-frequency noise and irrelevant details makes these models sensitive to background disturbances. To adress these limitations, this study introduces a novel approach reducing the high-frequency noise in the input image for the object detection task. Specifically, the proposed model comprises a novel Adaptive Padding (AP) mechanism and Region of Interest (RoI) detector, which provides a balance between RoI generation and object detection. Experimental evaluations conducted on five bird datasets using R-CNN demonstrate that our method increased the proportion of region proposals with an Intersection over Union (IoU) greater than 0,5 from 20,93% to 65,17%. Furthermore, the number of positive proposals increased by 330% during training and 726% during testing, while redundant proposals were reduced by 55,48%. The proposed model was also used for YOLOv8 and tested with 351 images of 6 bird species. The model increased the mAp50 from 0,954 to 0,984 and the mAP50-95 from 0,633 to 0,781.

Keywords: Object Detection, Image Processing, Viola and Jones Algorithm, Adaptive Padding, Gaussian Blurring.

Dedication

To my family,

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

OD	: Object Detection
CNN	: Convolutional Neural Network
R-CNN	: Region-based Convolutional Neural Network
AP	: Adaptive Padding
RoI	: Region of Interest
IoU	: Intersection over Union
ResNet	: Residual Network
YOLO	: You Only Look Once
SIFT	: Scale-Invariant Feature Transform
HOG	: Histogram of Oriented Gradients
DPM	: Deformable Part Model
SPPNet	: Spatial Pyramid Pooling Network
RPN	: Region Proposal Network
FPN	: Feature Pyramid Networks
VJ	: Viola and Jones
SPPF	: Spatial Pyramid Pooling Faster
HDC	: Hybrid Dilated CNN
EAPT	: Efficient Attention Pyramid Transformer
TBLS	: Temporally Broad Learning System
KOA	: Knee Osteoarthritis
ONNX	: Open-Exchange Neural Network
mAP	: Mean Average Precision
FP	: False Positive
Dfl_loss	: Distribution Focal Loss
CFAR	: Constant False Alarm Rate
PAFPN	: Path Augmentation Feature Pyramid Network

LIST OF SYMBOLS

$ii(a, b)$	: Integral Image
$s(a, b)$	: Cumulative Row Sum
$f(x)$	: Output of Strong Classifier
t_i	: Label Set
$w_{x,i}$	: Weight Parameter
L_i	: Error
$G(x, y)$	: Gaussian Kernel
σ	: Standart Deviation
I_c	: Image Set belonging to class C
A_c^i	: Boundary Boxes of i^{th} Image in class C
A_c	: Set of Proposals in class C
I_c'	: Blurred Image
s	: Single Frame
k	: Constant
w_{obj}	: Object Width
h_{obj}	: Object Height
p_w	: Horizontal Padding
p_h	: Vertical Padding
α	: Alpha
β	: Beta
p_l	: Left Padding
p_r	: Right Padding
p_b	: Bottom Padding
p_t	: Top Padding
w_i	: Image Width
h_i	: Image Height
d_{left}	: Distance of the Object from the Left
d_{right}	: Distance of the Object from the Right

d_x : Horizontal Distance

d_{top} : Distance of the Object from the Top

d_{bottom} : Distance of the Object from the Bottom

d_y : Vertical Distance

1. INTRODUCTION

Automated object detection systems have become a standard in computer vision, especially with the emergence of Convolutional Neural Networks (CNNs) (Zhao, Zheng, Xu, and Wu, 2019). Unlike conventional neural network architectures, CNNs have a deeper and more hierarchical structure. In this way, it enables them to learn complex features by transferring information extracted from lower-level layers to upper layers. Thanks to the development of large-scale training data generation, high-performance processors such as GPUs, network architectures, and training strategies can be successfully used in computer vision systems.

Object detection serves as a crucial element in a wide field of computer vision and plays a significant role in various applications that range from image segmentation (Feng et al., 2020; Hoeser and Kuenzer, 2020; Li et al., 2023) and object tracking (Ciaparrone et al., 2020; Urdiales, Martín, and Armingol, 2023) to activity recognition (Chen et al., 2021; Ferrari, Micucci, Mobilio, and Napoletano, 2023; Helmi, Al-qaness, Dahou, and Abd Elaziz, 2023; Vrskova, Kamencay, Hudec, and Sykora, 2023; Wang, Chen, Hao, Peng, Hu, 2019) and face detection (Liu, Chen, Wang, Li, and Xu, 2022; Masud et al., 2020; Serengil and Ozpinar, 2020). This technology enables computers to effectively interpret visual data by imitating human perception. Thus, objects in images or video frames are identified and located. In recent years, object detection is actively utilized in numerous real-world scenarios. For example, it is used to count vehicles and to optimize traffic flow in traffic management. In digital media, object detection improves image labeling capacity and facilitates better organization and retrieval of visual content. In agriculture and manufacturing, it allows automatic quality control and inventory tracking, which is useful for food inspection. In the field of autonomous driving, object detection has a key role for ensuring reliable navigation and making informed-decisions in changing environments.

Recent advances in network architectures such as ResNet (Residual Network), Faster R-CNN, and YOLO have increased the speed and accuracy of object detection tasks. Advanced training methods such as transfer learning and data augmentation enable CNN's to perform more effectively, particularly when training data is limited for real-world usage. Overall, thanks to

deep network architectures, powerful processing hardware, and learning models, automatic object detection methods have become a reliable technology in computer vision.

The first significant work in object detection was in 2001 by Paul Viola and Michael Jones (Viola & Jones, 2001), who introduced a binary classification method for face recognition based on the study of Papageorgiou et al. (1998). In 2005, the Histogram of Oriented Gradients (HOG)(Dalal & Triggs, 2005) feature descriptor was presented, which significantly surpassed existing pedestrian recognition methods. HOG represents a crucial advancement of the scale-invariant feature transform (SIFT) (Lowe, 1999) and shape contexts (Belongie, Malik, and Puzicha, 2002). To actually balance the feature invariance with the nonlinearity, the HOG is organized to be calculated on a dense grid of uniformly spaced cells while overlapping local contrast normalization within blocks.

In 2009, Felzenszwalb et al. developed a feature-based detection model, the Deformable Part Model (DPM)(Felzenszwalb, McAllester, and Ramanan, 2008). DPM consists of decomposing the object into different parts and aggregating the detections of each object component based on the divide-and-conquer approach. Although today's object detection strategies outperform conventional methods in terms of detection accuracy, they are inspired by localization and feature extraction principles.

In order to better understand the interaction between objects and visual scenes and to analyze complex groups of objects, it is crucial to operate sequential and increasingly sophisticated learning models. In recent years, significant developments have occurred in the field of object detection and image processing with the emergence of CNNs. In 2014, Girshick et al. presented Region-Based Convolutional Neural Networks (R-CNN) inspired by the high computational capability of CNNs (Girshick, Donahue, Darrell, and Malik, 2014). The R-CNN architecture integrates the selective search algorithm(Uijlings, Sande, Gevers, and Smeulders,2013) which generates the candidate object proposals with CNN. The generated object proposals are rescaled to a fixed size and subsequently fed into deep CNN, which ultimately classifies the object proposals using a classifier. Although R-CNN is an important milestone in the field of object

detection, processing each proposal via CNN separately causes a considerable increase in computational cost.

In 2014, He et al.(He, Zhang, Ren, and Sun, 2015) introduced Spatial Pyramid Pooling Networks (SPPNet). Previous CNN models require fixed-size input images for training. SPPNet offers a Spatial Pyramid Pooling (SPP) layer that allows a fixed-size region proposal regardless of the length of the input image. Unlike R-CNN, object proposals are generated by selective search using the feature map instead of extracting directly from the input image. This speeds up the detection task by 20 times without decreasing the detection accuracy. Despite the improvements in the speed of object detection, SPP-Net has certain limitations including only the fully connected layer needs to be fine-tuned and multi-stage training.

In 2015, Girshick introduced Fast R-CNN (Girshick, 2015). Unlike SPPNet, Fast R-CNN allows simultaneous training of all the network layers. For each object proposal, a RoI pooling layer generates a fixed-length feature vector. Each feature vector is fed through a series of fully connected layers and then into two sibling output layers: one that generates a softmax probability value for each class; the other extracts 4 values, the boundary box coordinates for each of the classes. Although Fast R-CNN benefits the advantages of SPP-Net and R-CNN, its in proposal generation and object detection speed is still limited.

In 2015, Ren et al. (Ren, He, Girshick, and Sun, 2015) introduced the Faster R-CNN object detector, which generates cost-free region proposals using a Region Proposal Network (RPN) instead of selective search. The Faster R-CNN consists of two modules: fully connected CNN to extract region proposals and Fast R-CNN object detector. The RPN module tells the Fast R-CNN whether the proposal has the object or not and Fast R-CNN makes the classification and boundary box regression for each object proposal. While Faster R-CNN overcomes the bottleneck of Fast R-CNN, it still has certain computational redundancies.

Lin et al.(Lin et al., 2017) introduced Feature Pyramid Networks (FPN) in 2017. While most of deep learning-based object detectors use feature maps of high-level layers in classification, it has some disadvantages in detecting objects coordinates. Therefore, FPN, a top-down

architecture with lateral connections, was developed to create high-level semantics at various scales. While FPN is successful in detecting objects which has different scales, it has some disadvantages due to its high computational cost and complexity of integration into existing methods.

Region proposal-based object detectors consist of two stages: First, generation of object candidate regions. This stage includes extraction of potential object areas and negative subsamples from the image. Various techniques can be used to generate region propoals, but generation of negative and positive subsamples requires significant computational resources. This causes computational cost in the object detection process. Second, refinement and classification. After the object candidate regions are generated, it is necessary to increase the accuracy of the proposals and assign the detected objects to the relevant categories. This step also involves rescoring the candidate regions according to the highest class probability. The framework for two-stage object detection meothods is illustrated in Figure 1.1.

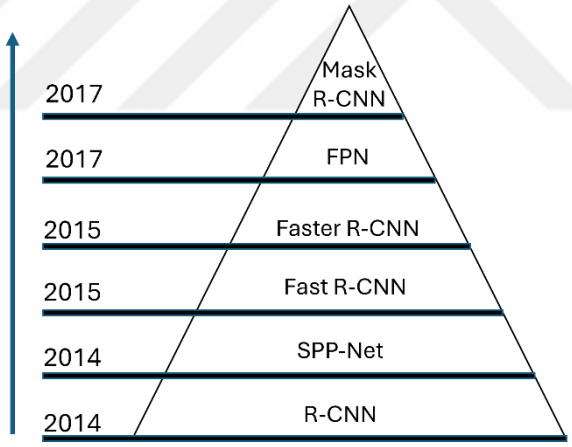

Figure 1.1. The Framework of Two-stage Object Detection Models

The strengths and drawbacks of the most frequently used two-stage object detectors are summarized below:

R-CNN is a widely used architecture to detect objects in images. It employes object boundary boxes and identifies both object classes and their positions in an input image. The architecture includes three key modules. The first module generates category-independent region proposals.

This is achieved by a proposal generator named Selective Search, which systematically segments the image into regions based on color, texture, size, and shape. Selective Search produces typically around 2000 candidate regions for each image. The second module uses CNN to analyze the region proposals generated by the selective search. The CNN encapsulates the most discriminative information about the visual content of each region. It converts the segmented areas into fixed-length feature vectors. The third module is the classifier, where the extracted feature vectors are fed into a fully connected neural network. This network is trained regarding the probabilities to various classes of objects to each proposal.

Training of R-CNN has a multi-stage pipeline that is costly in both time and memory management. Each training phase requires significant computational power and memory because the Selective Search algorithm generates an extensive number of region proposals for every single image. This high demand for resources can lead to considerable slowdowns in the training process, making the implementation of R-CNN a challenging task in scenarios with large datasets. Selective Search operates sub-segmentation and greedy search and recursively merge regions into a conclusive array of thousands potential regions (Wang, Ahsan, Li, and Hagen, 2022). The separate generation of region proposals causes increased computational cost, and bottlenecks that restrict classification speed. These limitations may adversely effect the performance of R-CNN in real-time applications, particularly in scenarios including crowded visual backgrounds and small objects(Lamichhane, Srijuntongsiri, and Horanont, 2025). Selective Search also generates regions from various areas in the scene, including background, corners, and shadows. Most of these regions are assumed to be negative sub-images, which may cause an imbalance between positive and negative samples during the training of object detector. The presence of excessive number of negative regions may cause the learning model to miss distinctive components between the background noise and objects. An excessive number of proposals may results in unreliable decision boundaries for learning model, especially in image data affected by class imbalance.

SPPNet significantly speeds up the training process compared to traditional CNNs. Instead of generating feature maps for each region proposal, SPPNet extracts feature maps only once for an entire image. Multi-scale spatial pooling layer enables SPPNet to detect various size of

objects. The model learns spatially consistent expressions, improving robustness. Since the spatial pyramid pooling layer combines information at various scales, causing more useful feature extraction. Since the feature extraction is applied only once to the entire image, irrelevant calculations observed in R-CNN are eliminated. Conversely, SPPNet is unable for end-to-end training due to its distinct feature extraction and classification steps. Although SPPNet is faster than R-CNN, it still depends on selective search to produce candidate regions. While SPPNet enhances multi-scale detection, it still deals with the detection of small objects, especially when using deep feature maps which may cause a loss of some details. Since the spatial pyramid pooling layer needs numerous pooling processes at various scales, this may cause increasing memory usage.

Fast R-CNN first creates a feature map from the entire input image. This feature map is a compact representation of the image obtained from the essential details in the raw-pixel values. After the feature map is created, the algorithm employs Selective Search to identify and extract potential region proposals within the feature map. Applying Selective Search to the feature map instead of the input image speeds up the proposal generation process. This approach reduces the amount of redundant computation since the feature map condenses information about the entire image and allows for faster processing. However, despite this speedup, there is still a significant computational cost due to the extraordinary number of redundant subimages that Selective Search produces. As a result, Fast R-CNN continues to struggle with the challenges of balancing efficiency and computational demand while improving speed compared to its predecessors.

Faster R-CNN significantly enhances region proposals generation stage using a Region Proposal Network (RPN). This method is more efficient than the traditional Selective Search algorithm. The generation of candidate regions initiates after the input image is fed into CNN.. Instead of relying on computationally expensive techniques like Selective Search, the RPN intelligently proposes potential object regions by sliding a small network over the feature map. For each region proposal generated, the RPN calculates an objectness score. The objectness score represents the likelihood about proposed region contains an object or not. Once the region proposals are generated, they undergo resizing to fit a fixed input size for subsequent

processing. The fixed-size proposals are then forwarded to two fully connected layers that operate in parallel.One of these layers determines which object of the class the proposal belongs to while the other layer refines the coordinates of the proposed regions to improve localization accuracy. Despite its enhanced capabilities, Faster R-CNN still has a notable computational bottleneck. Region proposal extraction by RPN causes the processing time and resource usage to increase. However, the precision and effectiveness of Faster R-CNN in detecting objects within images justify the higher computational costs involved.

While traditional CNNs struggle with detecting objects of different scales, FPN can improve detection accuracy at numerous scales. It employs a top-down and lateral connection structure to produce a feature pyramid. FPN maintains high-resolution spatial information in deeper layers and makes it more practical to detect small objects than traditional CNNs. Moreover, traditional architectures depend on high-level features, while FPN uses both low-level and high-level feature maps. This approach enables more meaningful and reliable feature extraction and improves classification and regression success. On the other hand, FPNs require more memory and computational resources because they use multiple feature maps across various scales. FPN needs longer training times due to the multiple feature map processing steps. While FPN improves the detection accuracy, it is slower than single-stage and real-time detectors like YOLO and SSD. Furthermore, FPN combines features from various layers; however some features may be redundant, and this may cause increased computational demands without significant performance improvement.

We have developed a novel preprocessing method with a two-step approach to solve this issue and achieve high accuracy and efficiency. The main primary objective of the proposed model is to replace significant region proposals with a smaller set of candidate regions and mitigate the effects of high-frequency noise in non-object regions. These smaller candidate regions are then used to address the problem of small object detection in R-CNN. Ultimately, this approach aims to increase the number of positive proposals by narrowing the search space, allowing the region proposal algorithm to focus on the positive candidate regions.

As an initial stage in generating the region proposal, we use a Viola and Jones (VJ) algorithm as a lightweight noise suppressor rather than a primary detector. The cascade structure effectively rejects background areas with low computational overhead by employing multiple weak classifiers trained on Haar-like features. As a result, fewer unnecessary regions are fed into the subsequent stages. This preprocessing step eliminates low-probability regions before Selective Search, the computationally intensive technique, is applied. Thus, it enables the model to focus on more salient areas. The lightweight nature of the cascade classifier also contributes to the reduced computational cost during inference. Despite its traditional association with face detection, the VJ can be trained to detect other types of objects as well. It is used in our study to filter potential bird regions, which are then enriched with a Selective Search proposal generator and fused using Gaussian filtering. This hybrid approach combines the speed and robustness of region proposal generation.

The major contributions of the proposed model for two-stage object detectors are:

- Removal of irrelevant background components which may cause class imbalances by applying a low-pass filter mechanism while keeping edge preservation.
- Utilizing an object proposal detector to combine all weak candidate regions for each object in a single frame and implementing a filter on non-object regions.
- Ensuring that the region proposal algorithm searches only noise-filtered candidate regions, reducing the number of proposals associated with the selective search.
- Distinction Refinement for RoI. To enhance the distinction between objects and visual scenes and assist R-CNN in identifying object boundaries, a filtering approach is utilized. The proposed technique aims to enhance the robustness of model in various lighting conditions.

The region proposal-based object detection models detect objects within candidate regions. However, these approaches are less suitable for real-time object detection applications since their high computational cost and slow proposal generator speeds. To adress this limitations, YOLO was introduced by Redmon et al. in 2016 (Redmon, Divvala, Girshick, and Farhadi, 2016). YOLO represents a significant enhancement in the field of object detection by

introducing a novel one-stage paradigm in contrast to two-stage approaches: apply the network to entire image. This network divides the input image into regions and predicts each region simultaneously. Subsequently, the emergence of YOLO models (Redmon and Farhadi, 2017; Redmon and Farhadi, 2018; Bochkovskiy, Wang, and Liao, 2020; Solawetz, 2020; Li et al., 2022; Wang, Bochkovskiy, and Liao, 2023; Jocher, 2020) has caused their widespread application across diverse fields for object detection. The framework for two-stage object detection methods is illustrated in Figure 1.2.

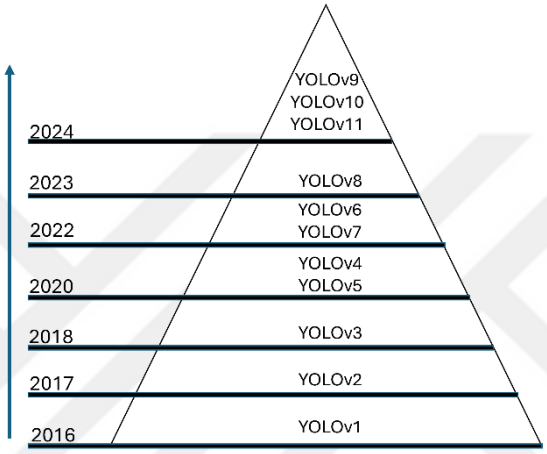

Figure 1.2. The Framework of YOLO Models

These models have shown remarkable efficacy in contrast to their two-stage counterparts, significantly improving performance outcomes in diverse contexts. Neverlethess, the latest YOLO models encounter several technical challenges and difficulties in object detection task, which are summarized below:

- Detection capability of different-scaled objects. Due to variable scene conditions such as light and distance, models may have difficulty detecting objects of various sizes (Zou, Chen, Shi, Guo, and Ye, 2023).
- Sensitivity to high-frequency noise. The presence of high-frequency noise within visual scenes may decrease sharp details between the background and objects.
- Detecting small objects. Small objects may be obscured in feature maps or may lack sufficient distinguishing characteristics using current detectors.
- Performance degradation due to high detection speed. YOLO suffers from drops in localization accuracy because it has a speed-focused architecture.

To overcome these issues, in this study, we propose a novel three-stage preprocessing approach that reduces the high-frequency noise in the input images, diminishing redundant details to an acceptable level with an adaptive padding mechanism. Therefore, the grid-based approach of the YOLO model can focus on sub-images containing more positive samples by simplifying the background details. The increase in the contrast between the background and the (RoI) also provides an advantage to YOLO in detecting the object's edges. It enables the development of a training model with high generalization ability.

The major contributions of this study for single-stage object detectors can be summarized as follows:

- Reduction of irrelevant background details that cause false positives. To smooth background noise, a low-pass filter is applied, while preserving edges.
- Applying Adaptive Padding Mechanism. To ensure that the YOLO grid focuses more on positive regions while avoiding irrelevant empty regions we apply AP mechanism. This prevents the model from assigning unnecessary anchor boxes to empty or redundant regions.
- Contrast Enhancement for RoI. To improve the distinction between objects and background and help YOLO to identify object edges, we apply a blurring mechanism that makes the detector more robust to variations in lighting and background conditions.
- To improve the distinction between objects and background and support YOLO to identify object edges, we apply a blurring mechanism that makes the detector more robust to variations in lighting and background conditions.

The study is organized as follows: The Literature Review section provides an overview of recent studies on region proposal-based and single-stage object detection. The Material and Method section summarizes the dataset used in this experiment and explains the proposed model in depth. The Experimental Test and Results section presents the experimental results, analysis and discussion. Conclusion section provide the strengths and drawbacks of the proposed approach regarding the experimental results.

2. LITERATURE REVIEW

2.1. Region Proposal-based Object Detection

Well-known deep learning approaches for object detection based on region proposals that belong to the R-CNN family are R-CNN, Fast-RCNN and Faster-RCNN. A study (Xu et al., 2022), employed Faster R-CNN and Mask R-CNN to detect road cracks with over 130 images. In terms of identifying both light and dark cracks, the proposed model performed better performance than YOLOv3. VGG-16-based Faster R-CNN was used in another study (Li et al., 2023) to detect bridge cracks in the province of Hunan. The precision, recall, and F1-score were 92.03%, 96.26%, and 94.10%, respectively.

One study (Xu, Ren, Cai, and Zhang, 2023) presented an enhanced version of Faster R-CNN to identify pulmonary nodules. The model was tested on the LUNA16 dataset and surpassed the benchmark models. The presented model increased the detection precision from 76.4% to 90.7%. The paper (Sahin, Ulutas, Yuce, and Erkoc, 2023) presents a study that Faster R-CNN and Mask R-CNN to diagnose COVID-19 on CT images. According to the experimental results, it can be stated that the presented model achieved an accuracy rate of 93.86% employing the VGG-16 backbone. A study (Sunardi, Yudhana, and Putri, 2023) presented a strategy to optimize the classification of breast cancer using Faster R-CNN. The researchers (Xin, Zhang, and Pan,2023) came up with a new hybrid dilated multilayer Faster RCNN model. in order to tackle the issue of a large number of learning parameters in the original Faster RCNN model. They replaced the original VGG16 network with a hybrid dilated CNN (HDC) model to extract features. Additionally, they incorporated a LeakyReLU activation function to enhance the model's ability to detect objects rapidly and accurately by improving the mapping of negative input information. To convert single-scale features into multi-scale features, they utilized a multilayer feature spatial pyramid, and they improved the accuracy of object detection by obtaining higher-resolution information through a deconvolutional network.

Song, Li, Dai, Wang, and Chen (2023) proposed a two-stage underwater detection method named Boosting R-CNN. Their model includes three main parts: a region proposal network called RetinaRPN, a probabilistic inference pipeline, and a boosting-based reweighting method.

According to their results, this model achieved better performance than other advanced object detectors on two underwater datasets.

In another study, Tian, Bi, Han, and Yu (2024) introduced EasyRP-RCNN. The main purpose of their study is to increase detection efficiency while maintaining high accuracy. They used satellite cloud images in the visible light spectrum. They also developed a new RoI selection method named Easy Region Proposal. This proposed model was utilized with a scale-based RoI grouping module which allows to avoid misclassification of undersized RoI.

Liu, Lu, Gao, Bu, and Naohiko (2024) focused on improving instance segmentation. They introduced a new loss function called parameter link loss. The proposed model was designed to reduce the computational and resource-intensive cost for parameter tuning. To enhance accuracy in mask segmentation and localization, they integrated a visual attention network into the backbone. They also used the U-Net structure in the segmentation task, which led to the creation of a model named Simple-Mask.

Wu, Gu, Xie, Wang, Cheng, and Wei (2022) presented a model called PV-RCNN++. The proposed approach uses semantical point-voxel feature interaction to improve detection performance. Unlike traditional approaches, PV-RCNN++ benefits from semantic features for more accurate results. Extensive experiments conducted on the KITTI dataset showed that their model a achieved rate of 81.60%, which is equal to or even better than existing advanced models.

Another study (Zhou and Yu,2022) introduced a novel framework called Point RCNN for detecting rotated objects. The model includes two stages angle of the free detection strategy: PointRPN and PointReg. The authors found that the presented model achieved excellent detection results on various large aerial image dataset. A study (Lin et al., 2021) introduced a novel vision transformer architecture known as the Efficient Attention Pyramid Transformer (EAPT). Extensive evaluation shows that EAPT performs better than state-of-the-art methods across multiple tasks.

A paper (Jiang, Sheng, Li, and Lee, 2022) proposed a comprehensive portrait photography assistant system to enhance the photography skills of beginners. The system includes two main stages: retrieving reference photos and generation of photography GUI. Sheng, Li, Ali, and Chen (2022) proposed a method called Temporally Broad Learning System (TBLS). Their model takes input from videos that are not temporally consistent. It uses both the original frames and their corresponding future frames, as well as the last original frame's output. First, the model applies andomly chosen weights and converts the input data into feature nodes. Then, these nodes are extended into enhancement nodes using a second set of random weights. Finally, both the original feature nodes and the enhancement nodes are connected to the output layer using a specific weight vector. In another study, Mittal, Chawla, and Tiwari (2022) introduced a hybrid method for vehicle detection and traffic density estimation. Their approach combines Faster R-CNN and YOLO models, and it achieved a detection accuracy of up to 98% in estimating traffic density.

Another study(Xia, Huang, Gu, and Wang, 2025) introduced ResDCA-Net, a defect-detection architecture that couples deformable convolutions with a lightweight multiscale attention mechanism to handle irregular defect shapes and high background similarity. An attention-based decoupled head further separates classification and regression cues, boosting both tasks. On the GC10-DET and NEU-DET benchmarks, the network achieves mAP scores of 83.7% and 83.4%, respectively, outperforming prior methods. Ablation studies confirm the contribution of each module. These results highlight the importance of adaptive receptive fields and multiscale attention in industrial surface defect detection. The authors (Luo, Wang, and Wang, 2025) proposed a separable convolution-based neural network for pneumonia detection from chest X-ray images, addressing class imbalance through a novel data augmentation strategy. The model achieved high performance with 97% accuracy, 98% sensitivity, and an AUC of 0.99. Class Activation Mapping (CAM) was used for interpretability. The approach showed superior results compared to state-of-the-art methods, offering a practical solution for regions with limited medical resources.

R-CNN-based architectures described in this section primarily emphasize the accuracy of the model. However, they have not addressed the issue of generate approximately 2000 object

proposals employing the fast mode of selective search. Although generating fewer proposals reduces computational cost, it may also significantly increase the risk of missing proposals. Inaccurate generation of candidate boxes can significantly reduce the localization performance of the object detector.

In this study, we introduce a novel preprocessing model designed to produce more comprehensive and positive proposals focusing on RoI by filtering out background noise. Thus, we enable the generation of various object proposals of different sizes. Similar to the fast mode of selective search, we generate a maximum of 2000 candidate regions per image, comprising up to 1,000 positive and 1,000 negative sub-images using the quality mode of selective search. Thus, we increase the number of positive proposals to maintain the balance between classes.

2.2. Single-Stage Object Detection

YOLO is a well-known single-stage object detection model with significantly advanced real-time object detection capabilities. This single-stage framework employs a single neural network structure to predict the bounding boxes and class labels for numerous objects in an image simultaneously. Unlike traditional two-stage object detection methods, YOLO proposes a detection system that splits the input image into grids and predicts candidate boxes. In recent years, researchers have developed various versions of YOLO and each superior to the previous. A paper (Boudjit and Ramzan, 2021) presented a YOLOv2-based real-time object detection model using the camera of drone. The proposed model achieved higher accuracy and respond time than traditional object detection models in tracking detected person. Another study (Sridhar, Jagadeeswari, Harini, Akshaya, and Haritha, 2022) established an advanced method using deep (CNNs) to detect motorcycle riders who do not follow traffic rules. The method consists of detecting motorcycles and helmets. The proposed YOLOv2 structure produces better results compared with traditional models. One study (Yunus et al., 2022) presented a detection strategy for Knee osteoarthritis (KOA) on radiographic images. The radiograph images were converted into feature maps by PCA, Alex-Net, and Dark-Net. The selected hyperparameters were then trained with YOLOv2 and an Open Neural Network Exchange (ONNX) framework. The proposed model achieved 98% mAP.

A study (Gai, Chen, and Yuan, 2021) proposed an improved YOLOv4 to detect cherry fruits. The model comprises a network based on the YOLOv4 backbone CSPDarknet53 and combined with DenseNet. The density between layers was changed to a circular shape that matched the form of the cherry fruit. The proposed model was compared to YOLOv3, YOLOv3-dense, and YOLOv4 object detectors. The improved YOLOv8 surpassed the YOLOv8, which is a 0.15 higher mAP result. The paper (Ji, Ling, and Han, 2023) introduced MCS-YOLOv4, adding a scale recognition to the existing scales to increase the richness of location details. The model was evaluated on the publicly available small object datasets and surpassed other detection algorithms in detecting small objects.

One effort (Tang, Zhang, and Fang, 2024) presented an improved version of YOLOv5, HIC-YOLOv5. An extra prediction mechanism was included providing a better feature map, and an involution block was added between the backbone and neck to enhance channel details belonging to the feature map. Bao (2024) integrated the ParNet and the RepVGG module and introduced an enhanced version of YOLOv8. The paper (Sapkota, Ahmed, and Karkee, 2024) conducted a comparative study between YOLOv8 and Mask R-CNN in complex orchard environments. The authors revealed that YOLOv8 surpassed Mask R-CNN in terms of both speed and accuracy.

The methods introduced in this section mainly emphasize the accuracy of the model. However, they have not addressed the high-frequency noise caused by the irrelevant background details. What sets the proposed approach apart from existing methodologies is that it reduces the background noise to an acceptable level, allowing the single-stage object detection algorithm to focus on more positive regions, thus reducing the classification accuracy and the computational cost by narrowing the search space. An adaptive padding mechanism is added to the RoI to provide more flexible object detection.

3. MATERIALS AND METHODS

This section introduces the proposed model in detail and the bird dataset used in this experiment. It contains the following subsections: the input images fed into R-CNN and YOLOv8, VJ algorithm, a blurring model, and an adaptive padding mechanism to reduce high-frequency noise in non-object regions in image data and make precise and focused predictions.

3.1. Materials

The model was evaluated using five datasets obtained from the TRAKUS (trakus,n.d.), which contains images of 497 bird species, 458 of which live and are certified in Turkey for the training of R-CNN. Table 3.1 shows the descriptions of the dataset used for the R-CNN.

Table 3.1. Dataset Descriptions for R-CNN.

Dataset	**Number of Images**
White-backed Woodpecker	32
Rustic Bunting	16
Short-toed Treecreeper	26
Little Crake	61
White-eared Bulbul	32

The model was also evaluated using a collection of 351 bird images obtained from the TRAKUS bird repository for YOLOv8. Table 3.2 gives a comprehensive overview of the dataset. Large images have a maximum size of 1024 pixels, whereas Very Large images have a maximum pixel width of 7680. Wide, Very Wide and Tall represent aspect ratios of 2:1, 3:1 and 0.5:1, respectively.

Table 3.2. Dataset Descriptions for YOLOv8

# Images	# Annotations	# Classes	Image Size		Aspect Ratio		
			Large	Overlarge	Wide	Very Wide	Tall
351	355	6	45,9%	54,1%	31,1%	66,1	2,6%

Annotating ground truth data is challenging because detecting small objects requires considerable effort during evaluation. Manual labeling of ground truth data still requires much attention, mainly when the input image contains many or small object proposals. In both cases, labeling and detecting the objects are time-consuming. In this study, in addition to a novel preprocessing step for object detection, we also propose novel bird datasets that are manually annotated for this experiment. Manual annotation was performed using a custom-built Python-based image annotation tool named LabelImg. Each image was meticulously examined and labeled according to a consistent annotation standard: bounding boxes were precisely drawn to tightly encapsulate the target objects with minimal background, and class labels were assigned following a predefined taxonomy. The annotations were saved as XML files in PASCAL VOC style and double-checked for consistency and completeness. As illustrated in Figure 3.1., we give some examples of image annotations for R-CNN.

Figure 3.1. Visualization of Ground Truth Annotations

For training YOLOv8, additional labeling was conducted resulting in the creation of a second dataset. Figure 3.2. shows some examples of annotated data for evaluation of proposed model using YOLOv8. Figure 3.3. demonstrates the dimensions and aspect ratios of the image data used for YOLOv8. Figure 3.4. also illustrates the locations of the majority of image annotations. Color gradients have been utilized to specify the frequency of the annotations within per grid cell.

Figure 3.2 Visualization of Ground Truth Annotations for YOLOv8. From Top to Bottom: Original Images and Blurred Images by the Proposed Model

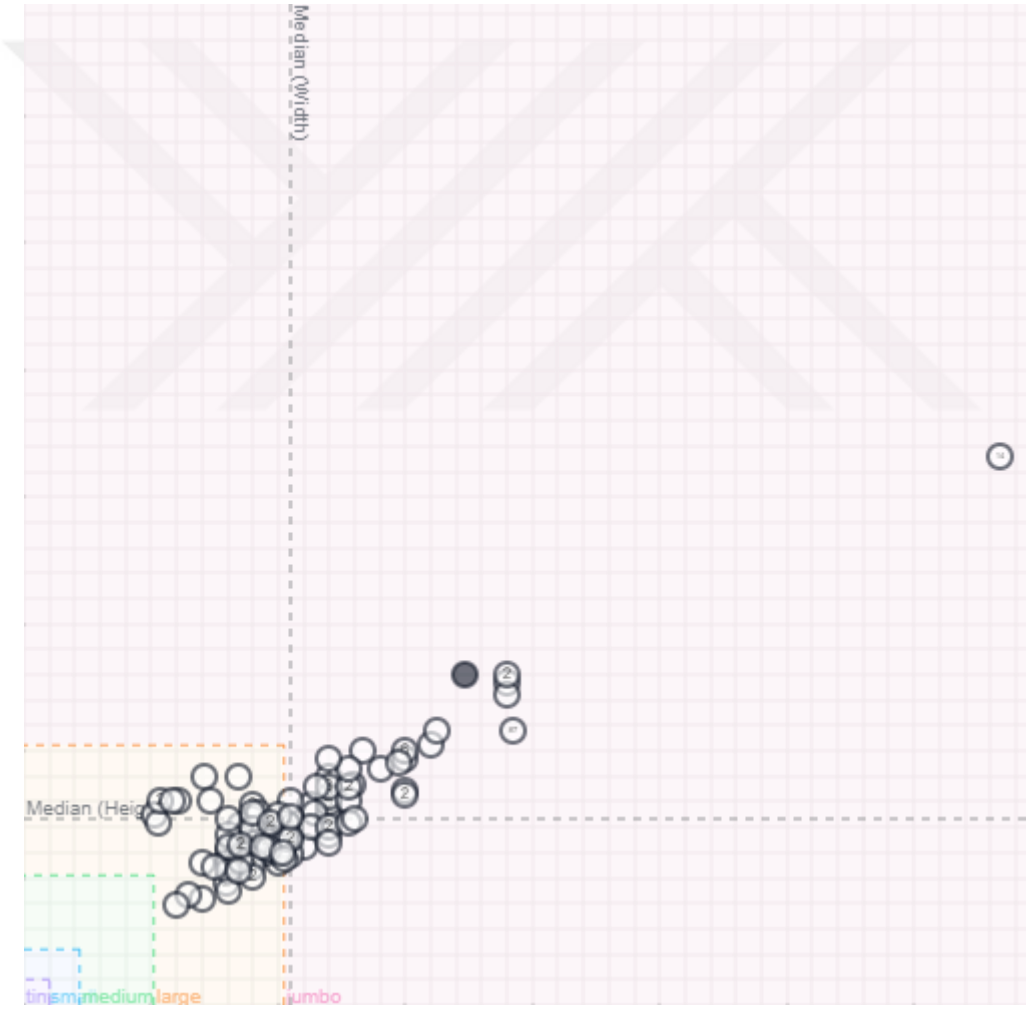

Figure 3.3. Outline of the Dimensions and Aspect Ratios of the Image Data Used for YOLOv8

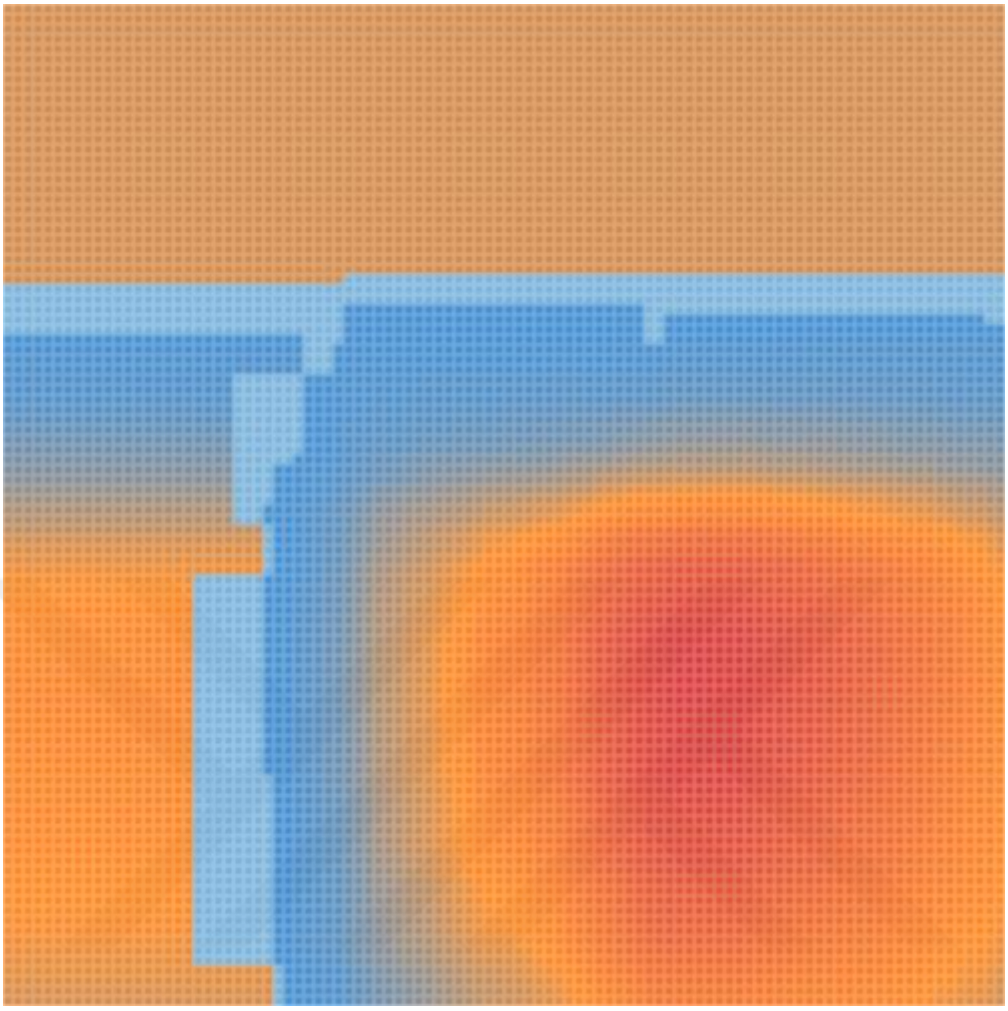

Figure 3.4. The Locations of the Majority of Image Annotations based on Color Gradients

3.2. Methods

The primary concept of the proposed model is to minimize the high-frequency noise in the background image so that the object detection algorithm can focus only on the positive regions and achieve more effective detection tasks by taking advantage of the sharp details in the non-blurred sub-image. Complex and crowded background images may cause the object detection algorithm to produce false positive results. A more meaningful feature map can be extracted in a denoised image by removing irrelevant features. For this purpose, a three-stage preprocessing model is applied in this study: First, the Viola and Jones (VJ) algorithm determines the regions that can be objects in the image. Then, these regions are merged in a single frame to create the RoI, and an adaptive padding mechanism is applied to make the detected RoI region more

flexible. Finally, the Gaussian filter is employed for the rest of the image. This filter limits the search area and focuses on more positive images. While the proposed method was evaluated using R-CNN framework, VJ and Gaussian Filter were employed. In the evaluations involving YOLOv8 Adaptive Padding mechanism was also implemented. The block diagram of the proposed model including Cascade Classifier and low-pass filter for R-CNN is shown in Figure 3.5.

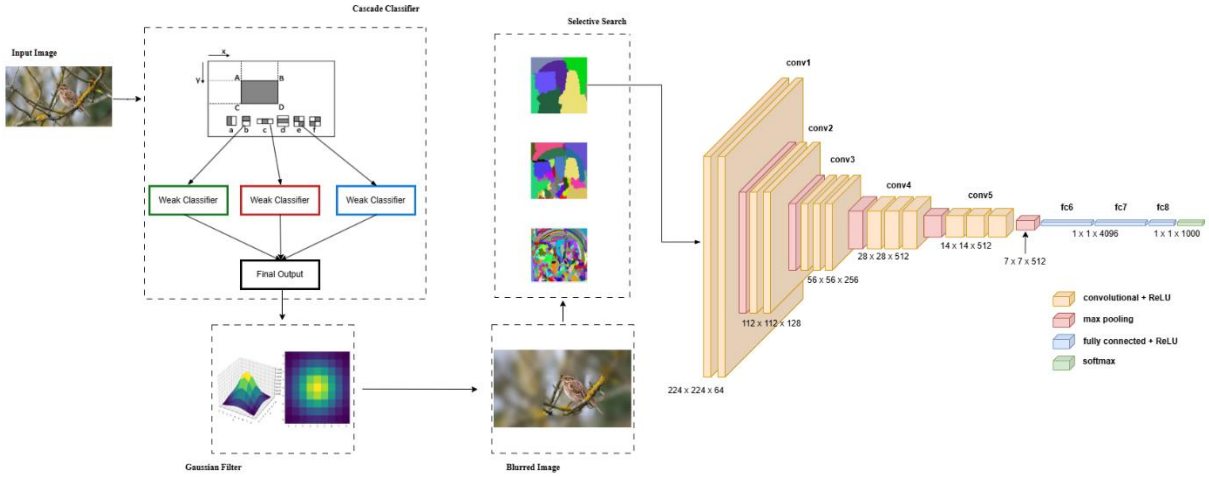

Figure 3.5. The Architectural Diagram of the Proposed Method for R-CNN

The individual steps of presented framework are also shown in Figure 3.6. for R-CNN.

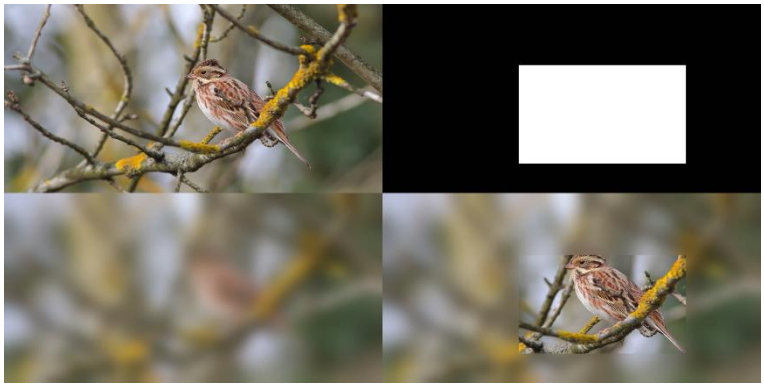

Figure 3.6. Sequential Stages of the Proposed Detection Framework

Adaptive Padding mechanism has been intagrated into the proposed model to provide flexibility in the RoI. The newly introduced module has been evaluated in YOLOv8 by combining it with

VJ algorithm and low-pass filter. The model diagram of the proposed method for YOLOv8 is presented in Figure 3.7. The CBS module in Figure 3.7. consists of convolution, batch normalization (BN), and SiLu activation functions.

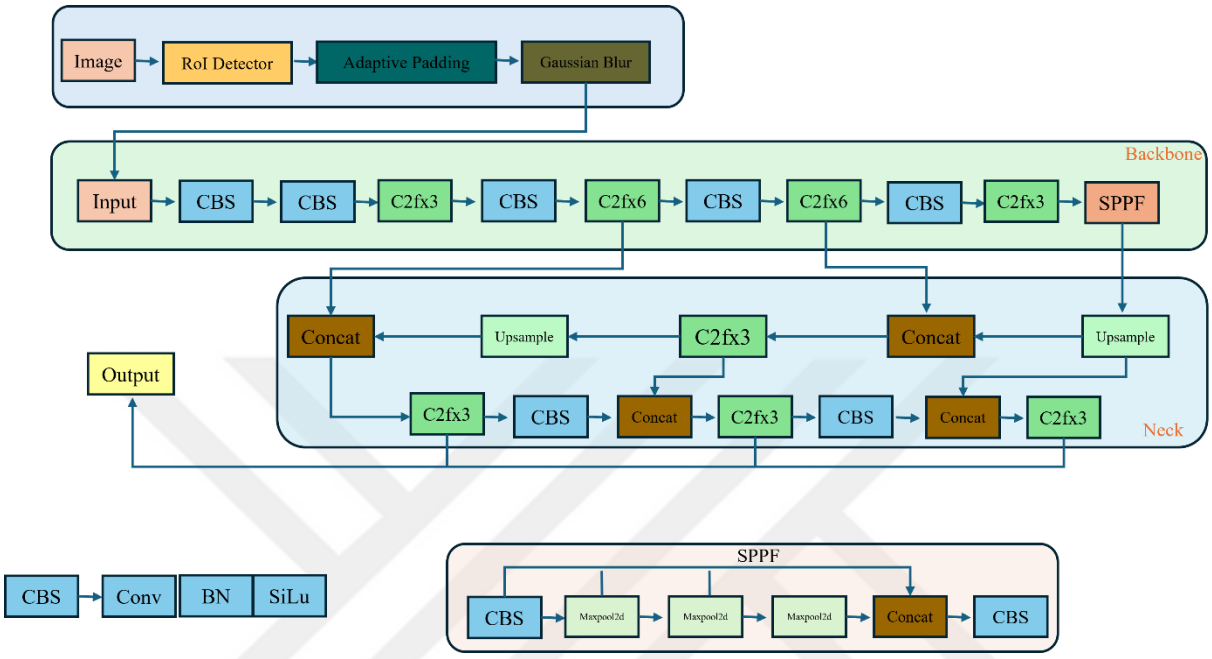

Figure 3.7. The Architectural Diagram of the Proposed Method for YOLOv8.

3.2.1. Framework of the Viola and Jones Algorithm

The VJ algorithm is a classification model used particularly in facial region detection tasks. It was first proposed by Viola and Jones (2001), based on the study by Papageorgiou et al. (1998). The algorithm consists of generation of rectangular features called Haar-like features, calculating the integral image, selecting top features using the AdaBoost algorithm, and classifying regions using the cascade structure steps.

3.2.1.1. Generating Haar-like Features

Haar-like features are sets of features that detect components such as edges, corners, and lines by calculating the intensity difference between two or more neighboring rectangular regions in an image. More precisely, the haar-like features consists of three types of features: A two-rectangle feature represents the dissimilarity between the summation of the pixels within two

rectangular areas. These regions have the identical dimensions and shape and are positioned either horizontally or vertically neighboring. A three-rectangle feature calculates the summation of the pixels comprised in two outer rectangles and subtracts this total from the summation in a center rectangle. Ultimately, a four-rectangle feature calculates the difference between pixel sums in two diagonal couples of rectangles. These extracted features are then used to determine whether an object is in the RoI. Given that the detector's intrinsic resolution is 24x24, the dimensions of the rectangular features are relatively extensive, exceeding 180,000. The calculation of rectangular features can be efficiently performed by employing the integral image, which serves as an intermediary model of the input image. Figure 3.8. shows the some examples of Haar-like features.

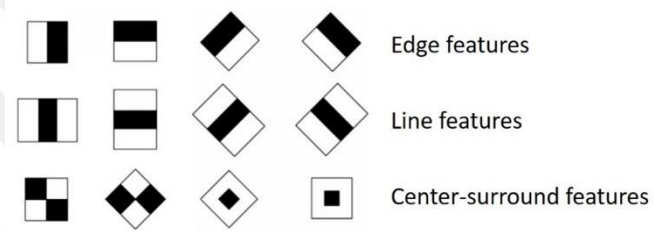

Figure 3.8. Haar-like Features

3.2.1.2. Calculating Integral Image

The integral image contains the sum of all pixel values above and to the left of each pixel in an image. Thus, Haar-like features are calculated much more efficiently by performing four additions and one subtraction operation, which calculates pixel intensity differences much more efficiently than those calculated with the original image. Considering (a,b) is a coordinate on the image, the integral image is:

$$ii(a,b) = \sum_{a' \le a, b' \le b} i((a',b') \tag{3.1}$$

where i(a,b) is the original image and ii(a,b) is the integral image. The cumulative sum of the ii(a,b) is computed using the following formulas:

$$s(a,b) = s(a,b-1) + i(a,b) \tag{3.2}$$

$$ii(a,b) = ii(a-1,b) + s(a,b) \quad (3.3)$$

where s(a,b) is the cumulative row sum, s(a,-1) = 0 and ii(-1,b)=0. The computation procedure of integral image is shown in Figure 3.9.

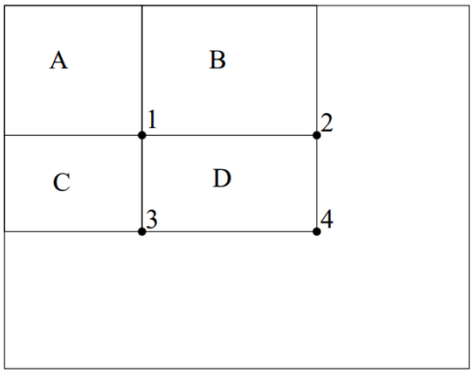

Figure 3.9. Calculation of Haar-like Features using the Integral Image.

The summation of the pixels in area D uses four array references: 4+1-(2+3).

3.2.1.3. Adaboost Learning Algorithm

The AdaBoost is a learning model that aims to construct a robust classifier by combining a series of weak classifiers. For each Haar-like feature, the AdaBoost creates a weak classifier. Thus, it is determined whether the image has this Haar-like feature. It is essential to emphasize that each subwindow of the input image produces over 180,000 rectangular Haar-like features,surpassing the total number of pixels. During the training of AdaBoost, the weight of the incorrectly classified samples is increased while the weight of the correctly classified samples is decreased. Thus, the aim is to select the most discriminative features to create a strong classifier. The output of the strong classifier, object or not, is determined by using a threshold value determined to be half of the total weights of the weak classifiers. Suppose the weak classifier c_j contains a Haar-like feature a_j, a threshold k_j and a parity s_j:

$$f(x) = \begin{cases} 1, & if\ s_j a_j(x) < s_j k_j \\ 0, & otherwise \end{cases} \quad (3.4)$$

The steps of the boosting algorithm are shown in Algorithm 1.

Algorithm 1: Adaboost Algorithm

input: Given example images $(d_1, t_1)...(d_n, t_n)$, where t_i = 1, 0 for positive and negative samples, respectively.
output: The final strong classifier c_x
Initialize weights $w_{1,i} = \frac{1}{2n}, \frac{1}{2p}$ for t_i = 1, 0, respectively, where p and n are the number of positive and negative samples, respectively.
for x := 1 to X **do**
Normalize the weights $w_{x,i} \leftarrow \frac{w_{x,i}}{\sum_{j=1}^{n} w_{x,j}}$ where w_x is a probability distribution.
For a feature j, train the classifier c_j limited to a single feature.
The error concerning w_x, $L_j = \sum_i w_i |c_j(d_i, t_i)|$.
Select the classifier c_x with the lowest error L_x.
Update the weights: $w_{x+1,i} = w_{x,i}\beta_x^{1-L_i}$
L_i = 0 if d_i is classified correctly
L_i = 1 otherwise and $\beta_x = \frac{L_x}{1-L_x}$
end
The final strong classifier $f(x) = \begin{cases} 1, & if \ \sum_{x=1}^{X} b_x c_x \geq \frac{1}{2}\sum_{x=1}^{X} b_x \\ 0, & otherwise \end{cases}$
where $b_x = log\ (\frac{1}{\beta_x})$

3.2.1.4. Cascading Classifiers

The cascade classifier comprises several steps, each expressing several weak learning models. The weak learners are trained by boosting, which enables highly accurate classification based on the mean prediction of multiple models. The classifier uses this prediction to decide whether an object has been detected (positive) or should proceed to the next step (negative). Figure 3.10. illustrates the cascading classifiers with n steps as a decision tree. At each step, a list of filters is applied to the region within the sliding sub-window. Each time the sliding sub-window moves, the new area within the sliding sub-window is processed step by step by the cascading classifier. The rectangular feature is evaluated at each step, and a weak classifier is calculated. A threshold value is then used to check whether the region is rejected as a face candidate or must be processed further in the next stage (Irgens, Bader, Le, Saxena, and Ababei, 2017).

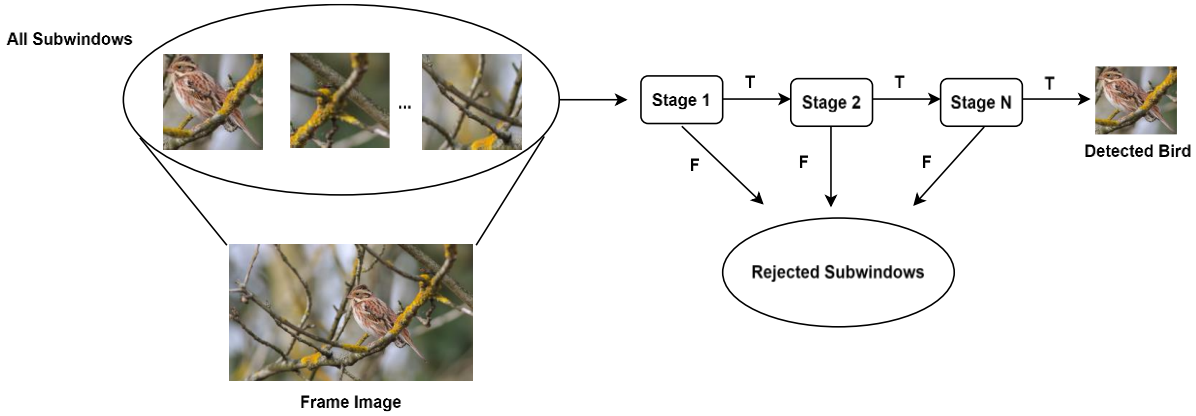

Figure 3.10. Cascade Architecture for Sequential Object Filtering

3.2.2. Gaussian Blurring

Gaussian blurring: Gaussian blurring is a low-pass filter used in computer vision to smooth the image and reduce high-frequency noise. Each pixel in the image is recalculated using a weighted average of neighboring pixels. These weights are determined by a Gaussian distribution, and neighbors closer to the center pixel have a higher impact. Therefore, this reduces high-frequency noise while providing a more homogeneous and balanced distribution between pixels, allowing for a natural blurring effect in the image (Dinç and Kaya, 2023). The Equation 3.5 represents the computation of Gaussian Kernel:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.5}$$

where (x,y) are the horizontal and vertical coordinates. The symbol σ represents the standard deviation. For a kernel matrix, suppose the dimensions of the kernel are x and y. The kernel size is characterized by odd numbers. Consequently, the row values range from -x/2 to +x/2, while the column values extend from -y/2 to +y/2. In order to generate the kernel matrix, the related row and column values are used to the relevant position in Equation 3.5.

Suppose the image set $I_c = \{I_{c,1}, I_{c,2}, \dots, I_{c,n}\}$ is the collection of n images belonging to C^{th} class, $A_c^i = \{A_c^{i,1}, A_c^{i,2}, \dots, A_c^{i,m}\}$ are the boundary boxes of i^{th} image in class C where m is the number of proposals for each image and $A_c = \{A_c^1, A_c^2, \dots, A_c^n\}$ is the set of proposals.

The steps of the proposed model are shown in Algorithm 2 for region proposal-based object detectors.

Algorithm 2: The proposed model for region proposal-based object detectors

input: $I_c = \{I_{c,1}, I_{c,2}, \dots, I_{c,n}\}$
output: Blurred Image Set I_c'
Single frame $s = \{(x_1, y_1), (x_1, y_1)\}, n, m, k$
for e := 1 to n **do**
    Initialize a single frame $s_e = \{(inf, inf), (-inf, -inf)\}$, for $I_{c,e}$
    Get $x_1^e, y_1^e, x_2^e, y_2^e \epsilon A_c^e$
    for j := 1 to m **do**
        $if\ x_1 > x_1^{e,c}$
            Update x_1
        $if\ y_1 > y_1^{e,c}$
            Update y_1
        $if\ x_2 < x_2^{e,c}$
            Update x_2
        $if\ y_2 < y_2^{e,c}$
            Update y_2
    end
    Extract RoI using s_e
    Use Gaussian Blurring to reduce image noise. The kernel size is k.
end

The choice of the kernel size in image processing depends on the amount of noise in the image. By adjusting the kernel size correctly, it is possible to increase the overall image quality. This improvement plays an important role in increasing object detection performance (Sahu, Pradhan, Wilczynski, Anis, and Pal, 2023). We dynamically adjust the Gaussian kernel size according to the image's width and height to produce consistent blurring across different image sizes. We applied Gaussian blurring with a kernel size of 5% of the image width and height, and a standard deviation of 0. By using a proportional approach, we were able to preserve object

boundaries and achieve adequate noise reduction while maintaining a consistent smoothing effect across resolutions. In particular, the adaptive kernel size enhanced the region proposals generated by the Haar Cascade, yielding more precise and reliable inputs for the subsequent R-CNN and Selective Search stages.

3.2.3. Adaptive Padding Mechanism

To enhance the selection accuracy for the region where the Gauss filter will be applied, two adaptive padding mechanisms are introduced around the RoI.

3.2.3.1. Adaptive Padding 1

The equations of the Adaptive Padding 1 algorithm used in conjunction with the Gaussian filter before object detection employing the YOLOv8 are shown below::

$$w_{obj} = x_2 - x_1 \tag{3.6}$$

$$h_{obj} = y_2 - y_1 \tag{3.7}$$

In general terms, the padding equations to be applied horizontally and vertically are defined as follows:

$$p_w = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} * w_{obj} \tag{3.8}$$

$$p_h = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} * h_{obj} \tag{3.9}$$

where α and β are scaling factors:

$$0 < \alpha < \beta < 1 \tag{3.10}$$

Specifically, the padding for left, right, top, and bottom of the RoI are calculated as follows:

$$p_l = \begin{cases} |w_{obj}.\beta|, & if\ x_1 \geq w_i - x_2 \\ |w_{obj}.\alpha|, & otherwise \end{cases} \quad (3.11)$$

$$p_r = \begin{cases} |w_{obj}.\alpha|, & if\ x_1 \geq w_i - x_2 \\ |w_{obj}.\beta|, & otherwise \end{cases} \quad (3.12)$$

$$p_t = \begin{cases} |h_{obj}.\beta|, & if\ y_1 \geq h_i - y_2 \\ |h_{obj}.\alpha|, & otherwise \end{cases} \quad (3.13)$$

$$p_b = \begin{cases} |h_{obj}.\alpha|, & if\ y_1 \geq h_i - y_2 \\ |h_{obj}\ .\beta|, & otherwise \end{cases} \quad (3.14)$$

$$x_1 = max(0, x_1 - p_l) \quad (3.15)$$

$$y_1 = max(0, y_1 - p_t) \quad (3.16)$$

$$x_2 = min(w_i, x_2 + p_r) \quad (3.17)$$

$$y_2 = min(h_i, y_2 + p_b) \quad (3.18)$$

where p_l, p_r, p_l, and p_b are the AP to be applied to the left, right, top, and bottom of the RoI, respectively. $w_i$and h_i denotes the width and height of input image. Equations [3.15-3.18] ensure that the boundary constraints keep the region within the image dimensions.

The steps of the proposed model are shown in Algorithm 3 for single-stage object detectors including AP mechanism.

Algorithm 3: The proposed model for single-stage object detectors

---

input: $I_c = \{I_{c,1}, I_{c,2}, \dots, I_{c,n}\}$
output: Blurred Image Set I'_c
Single frame $s = \{(x_1, y_1), (x_1, y_1)\}, n, m, k$
for e := 1 to n **do**
    Initialize a single frame $s_e = \{(inf, inf), (-inf, -inf)\}$, for $I_{c,e}$
    Get $x_1^e, y_1^e, x_2^e, y_2^e \epsilon A_c^e$
    for j := 1 to m **do**
        $if\ x_1 > x_1^{e,c}$
            Update x_1
        $if\ y_1 > y_1^{e,c}$
            Update y_1
        $if\ x_2 < x_2^{e,c}$
            Update x_2
        $if\ y_2 < y_2^{e,c}$
            Update y_2
    end
    Extract RoI using s_e
    Apply Adaptive Padding Mechanism around RoI
    Use Gaussian Blurring to reduce image noise regarding the kernel size k
end

3.2.3.2. Adaptive Padding 2

In addition to the Adaptive Padding 1 mechanism that works integrated with YOLO, a novel AP mechanism named Adaptive Padding 2 has been designed and implemented using R-CNN. This novel approach has been used in a publicly available rabbit dataset. The equations of the Adaptive Padding 2 mechanism are shown below:

$$\alpha = 1 - \frac{w_{obj}}{w_i} \tag{3.19}$$

$$\beta = 1 - \frac{h_{obj}}{h_i} \tag{3.20}$$

$$p_x = w_{obj} * \alpha \tag{3.21}$$

$$p_y = h_{obj} * \beta \tag{3.22}$$

$$d_{left} = x_{min} \tag{3.23}$$

$$d_{right} = w_i - x_{max} \tag{3.24}$$

$$d_x = d_{left} + d_{right} \tag{3.25}$$

$$p_r = p_x \frac{d_{right}}{d_x} \tag{3.26}$$

$$p_l = p_x - p_r \tag{3.27}$$

$$d_{top} = y_{min} \tag{3.28}$$

$$d_{bottom} = h_i - y_{max} \tag{3.29}$$

$$d_y = d_{top} + d_{bottom} \tag{3.30}$$

$$p_b = p_y \frac{d_{bottom}}{d_y} \tag{3.31}$$

$$p_t = p_y - p_b \tag{3.32}$$

where p_x and p_y are the total padding horizontally and vertically respectively. The distance of the object from the left is d_{left}, the distance from the right is d_{right}, and the total horizontal distance is d_x. Similarly, the distance of the object from bottom, top and total are d_{bottom}, d_{top}, and d_y, respectively.

3.2.4. Object Detection With R-CNN

The R-CNN algorithm is a region-based ConvNet widely used for deep learning-based object detection tasks. The model consists of three modules: The first extracts category-independent proposals from the input image using a selective search. These proposals most likely contain

the target objects and represent the candidate regions for the CNN. Proposals with fewer redundant regions have a more significant impact on the success of the R-CNN (Chen, Liu, Tuzel, and Xiao, 2017). The second module is a Deep CNN that accepts region proposals and extracts a fixed-length feature vector. The third part is a set of classifiers that recognize which class of region proposals the feature vector represents. In this section, the individual modules are explained. The overview of the R-CNN is shown in Figure 3.11.

3.2.4.1. Region Proposals

Many recent works have proposed approaches for generating category-independent proposals, such as selective search (Uijlings et al., 2013), multiscale combinatorial grouping (Arbalaez, Pont-Tuset, Barron, Marques, and Malik, 2014), objectness (Alexe, Deselaers, Ferrari, 2012), constrained parametric min-cuts (CPMC) (Carreira and Sminchisescu, 2010). In R-CNN, selective search is used to identify and extract both negative and positive region proposals.

3.2.4.2. Feature Extraction

The R-CNN extracts 4096 dimensional feature vectors from each region proposal, which are resized to 227*227 RGB images before being fed to the CNN. The features are computed by passing through five CNN blocks and two fully connected layers (Jia, Shelhamer, Donahue, Karayev, Long, Girshick, Guadarrama, and Darrell, 2014; Krizhevsky, Sutskever. Hinton, 2012).

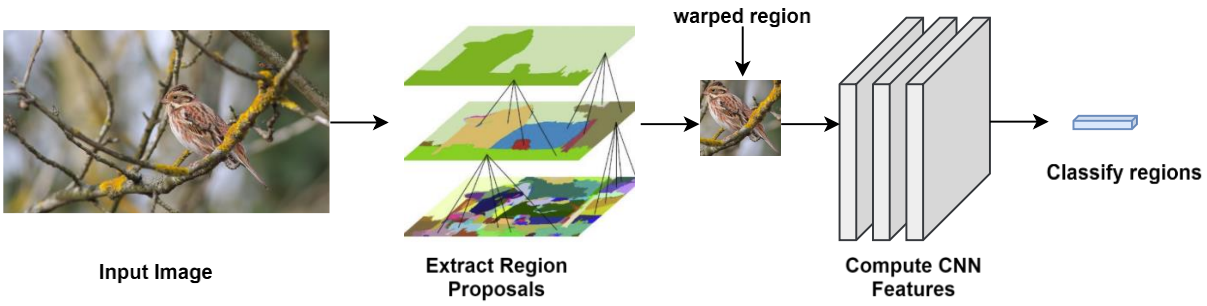

Figure 3.11. Architectural Design of the R-CNN Framework

We used the pre-trained VGG-16 model on the ImageNet dataset and applied the transfer learning approach. The convolutional base of VGG-16 was frozen to preserve the learned feature representations, and only the newly added fully connected 3 layers were trained. Algorithm 4 summarizes the training process of the VGG-16 framework adopted in this study, including its convolutional layers, pooling operations, and fully connected layers leading to the final prediction.

Algorithm 4: The VGG-16 training framework

---

input: RGB Image (X_{RGB})
output: Trained Classification Model
Initialize model parameters
for each sample of X_{RGB} in Dataset **do:**
    F_{C1} ← Apply 3x3 convolution with 64 filters to RGB Image
    F_{C2} ← Apply 3x3 convolution with 64 filters to F_{C1}
    F_{MP1} ← Apply 2x2 Max Pooling with 2x2 Stride to F_{C2}
    F_{C3} ← Apply 3x3 convolution with 128 filters to F_{MP1}
    F_{C4} ← Apply 3x3 convolution with 128 filters to F_{C3}
    F_{MP2} ← Apply 2x2 Max Pooling with 2x2 Stride to F_{C4}
    F_{C5} ← Apply 3x3 convolution with 256 filters to F_{MP2}
    F_{C6} ← Apply 3x3 convolution with 256 filters to F_{C5}
    F_{C7} ← Apply 3x3 convolution with 256 filters to F_{C6}
    F_{MP3} ← Apply 2x2 Max Pooling with 2x2 Stride to F_{C7}
    F_{C8} ← Apply 3x3 convolution with 512 filters to F_{MP3}
    F_{C9} ← Apply 3x3 convolution with 512 filters to F_{C8}
    F_{C10} ← Apply 3x3 convolution with 512 filters to F_{C9}
    F_{MP4} ← Apply 2x2 Max Pooling with 2x2 Stride to F_{C10}
    F_{C11} ← Apply 3x3 convolution with 512 filters to F_{MP4}
    F_{C12} ← Apply 3x3 convolution with 512 filters to F_{C11}
    F_{C13} ← Apply 3x3 convolution with 512 filters to F_{12}
    F_{MP5} ← Apply 2x2 Max Pooling with 2x2 Stride to F_{C13}
    F_{FLAT} ← Flatten the output to F_{MP5}
    F_1 ← Dense layer with 4096 units and ReLU activation applied to F_{FLAT}
    F_2 ← Dense layer with 4096 units and ReLU activation applied to F_1
    $\hat{y}$ ← Dense layer with 1 unit and Sigmoid activation applied to F_2
    Train the model
end for
output ← *TrainedModel*

$X_{RGB} \in R^{224x224x3}$ represents the RGB image input. The convolution operation applied in the block 1 for the X_{RGB} with a 3 × 3 kernel and same padding is given by:

$$F_{C1} = ReLu(W_1 * X_{RGB} + b_1) \tag{3.33}$$

where $W_1 \in R^{3x3x3x64}$ is the weight tensor for the convolution, $*$ specifies the convolution process, and b_1 is the bias. The convolution operation applied on F_{C1} with a 3 × 3 kernel and same padding is given by:

$$F_{C2} = ReLu(W_2 * F_{C1} + b_2) \tag{3.34}$$

where $W_2 \in R^{3x3x64x64}$ is the weight tensor for the convolution, $*$ specifies the convolution process, and b_2 is the bias. The max pooling operation applied on F_{C2} is given by:

$$F_{MP1} = MaxPool_{2*2,s=2}(F_{C2}) \tag{3.35}$$

where the feature map $F_{MP1} \in R^{112x112x64}$. The convolution operations applied in block 2 with 3 x 3 kernel and same padding are given by:

$$F_{C3} = ReLu(W_3 * F_{MP1} + b_3) \tag{3.36}$$

$$F_{C4} = ReLu(W_4 * F_{C3} + b_4) \tag{3.37}$$

where $W_3 \in R^{3x3x64x128}$, $W_4 \in R^{3x3x128x128}$ are the weight tensors for the convolution, $*$ specifies the convolution process, and b_3 and b_4 are the bias parameters. The max pooling operation applied on F_{C4} is given by:

$$F_{MP2} = MaxPool_{2*2,s=2}(F_{C4}) \tag{3.38}$$

where the feature map $F_{MP2} \in R^{56x56*128}$. The convolution operations applied in block 3 with 3 x 3 kernel and same padding are given by:

$$F_{C5} = ReLu(W_5 * F_{MP2} + b_5) \quad (3.39)$$

$$F_{C6} = ReLu(W_6 * F_{C5} + b_6) \quad (3.40)$$

$$F_{C7} = ReLu(W_7 * F_{C6} + b_7) \quad (3.41)$$

where $W_5 \in R^{3x3x128x256}$, $W_6 \in R^{3x3x256x256}$, and $W_7 \in R^{3x3x256x256}$ are the weight tensors for the convolution, $*$ specifies the convolution process, and b_5, b_6 and b_7 are the bias parameters. The max pooling operation applied on F_{C7} is given by:

$$F_{MP3} = MaxPool_{2*2,s=2}(F_{C7}) \quad (3.42)$$

where the feature map $F_{MP3} \in R^{28x28*256}$. The convolution operations applied in block 4 with 3 x 3 kernel and same padding are given by:

$$F_{C8} = ReLu(W_8 * F_{MP3} + b_8) \quad (3.43)$$

$$F_{C9} = ReLu(W_9 * F_{C8} + b_9) \quad (3.44)$$

$$F_{C10} = ReLu(W_{10} * F_{C9} + b_{10}) \quad (3.45)$$

where $W_8 \in R^{3x3x256x512}$, $W_9 \in R^{3x3x512x512}$, and $W_{10} \in R^{3x3x512x512}$ are the weight tensors for the convolution, $*$ specifies the convolution process, and b_8, b_9 and b_{10} are the bias parameters. The max pooling operation applied on F_{C10} is given by:

$$F_{MP4} = MaxPool_{2*2,s=2}(F_{C10}) \quad (3.46)$$

where the feature map $F_{MP4} \in R^{14x14*512}$. The convolution operations applied in block 5 with 3 x 3 kernel and same padding are given by:

$$F_{C11} = ReLu(W_{11} * F_{MP4} + b_{11}) \quad (3.47)$$

$$F_{C12} = ReLu(W_{12} * F_{C11} + b_{12}) \quad (3.48)$$

$$F_{C13} = ReLu(W_{13} * F_{C12} + b_{13}) \quad (3.49)$$

where $W_{11} \in R^{3x3x512x512}$, $W_{12} \in R^{3x3x512x512}$, and $W_{13} \in R^{3x3x512x512}$ are the weight tensors for the convolution, $*$ specifies the convolution process, and b_{11}, b_{12} and b_{13} are the bias parameters. The max pooling operation applied on F_{C13} is given by:

$$F_{MP5} = MaxPool_{2*2,s=2}(F_{C13}) \quad (3.50)$$

where the feature map $F_{MP4} \in R^{7x7*512}$. The flatten operator applied on F_{MP5} is given by:

$$F_{FLAT} = Flatten(F_{MP5}) \quad (3.51)$$

where $F_{FLAT}\, R^{25088}$. Following the feature extraction process, flattened feature maps are fed into the fully connected layers for classification. The first fully connected layer transforms F_{FLAT} into 4096 dimensional feature vector. Then second fully connected layer transforms this representation into another 4096 dimensional feature vector:

$$F_1 = ReLu(W_{14} * F_{FLAT} + b_{14}) \quad (3.52)$$

$$F_2 = ReLu(W_{15} * F_1 + b_{15}) \quad (3.53)$$

where , $W_{14} \in R^{25088x4096}$, $W_{15} \in R^{4096x4096}$, $b_{14} \in R^{4096}, and\ b_{15} \in R^{4096}$. The output layer employes Sigmoid activation function for the binary classification:

$$\hat{y} = Sigmoid(W_{16} * F_2 + b_{16}) \quad (3.54)$$

where $W_{16} \in R^{4096*1}$ and $b_{16} \in R^1$.

Once the region proposals are extracted by the selective search, the input data for The CNN is limited to a maximum of 1000 positive and negative samples for eachi mage, as the training data is too large to fit in memory. All training samples with an overlap of >= 0,5 IoU with the ground truth boxes are treated as positive, while < 0,3 is labeled as negative.

In the test stage, R-CNN accepts region proposals for each test image generated by the selective search. All region proposals are extracted in the selective search quality mode for the training and testing phases. Each proposal is forwarded to the CNN to calculate the feature vector. The object proposal is then evaluated by the classifier for each class.

3.2.5. Object Detection with YOLOv8

The YOLOv8 structure has two main components: the backbone and head, which employ a complete CNN.

Backbone: YOLOv8 has a novel backbone network that is a changed version of the CSPDarknet53 architecture consisting of 53 convolutional layers and uses a strategy named cross-stage partial connections to improve the transmission of knowledge across the different levels of the network (Bochkovskiy, Wang, and Liao, 2020).

The backbone includes several convolutional layers placed one after another. These layers extract meaningful features from the input data. The newly introduced C2f module combines high-level features with contextual information to enhance detection accuracy. In addition, the Spatial Pyramid Pooling Faster (SPPF) module and the layers that follow use features at multiple scales for better representation (He et al., 2015).

Head: This part of the model recieves the feature maps produced by the backbone and utilizes them for generating the final outputs. The predictions comprises boundary boxes and object class. In YOLOv8, the head is built as a detachable module. This design allows it to handle objectness scoring, classification, and regression tasks separately and more efficiently.

This method enables each unit to concentrate its own task enhancing the overall accuracy of the model. The upsample layers raise the resolution of the feature maps. The head operates a series of convolutional layers to investigate the feature maps. Subsequently a linear layer predicts the boundary boxes and class probabilities.

4. RESULTS AND DISCUSSION

This section presents a comprehensive analysis of all performance evaluations performed within this study, separately analyzed under their respective sub-sections.

4.1. Results with R-CNN

In this experiment, 3096 positive bird images among 10346 data belonging to 61 bird species obtained from TRAKUS were randomly selected for the training of the VJ Model. 6183 nature images without bird data were collected from the internet and labeled as negative. Some examples of negative and positive data are given in Figure 4.1 and Figure 4.2. The number of negative and positive images, number of training stages, sample width, sample height, and minimal hit are also shown in Table 4.1.

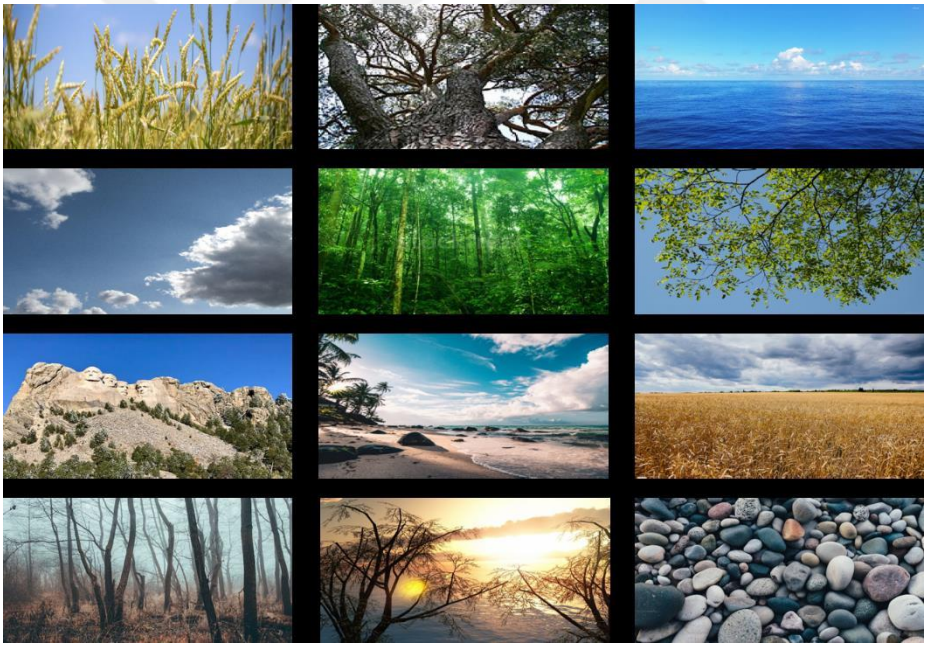

Figure 4.1. Some Examples of Negative Data for VJ Algorithm

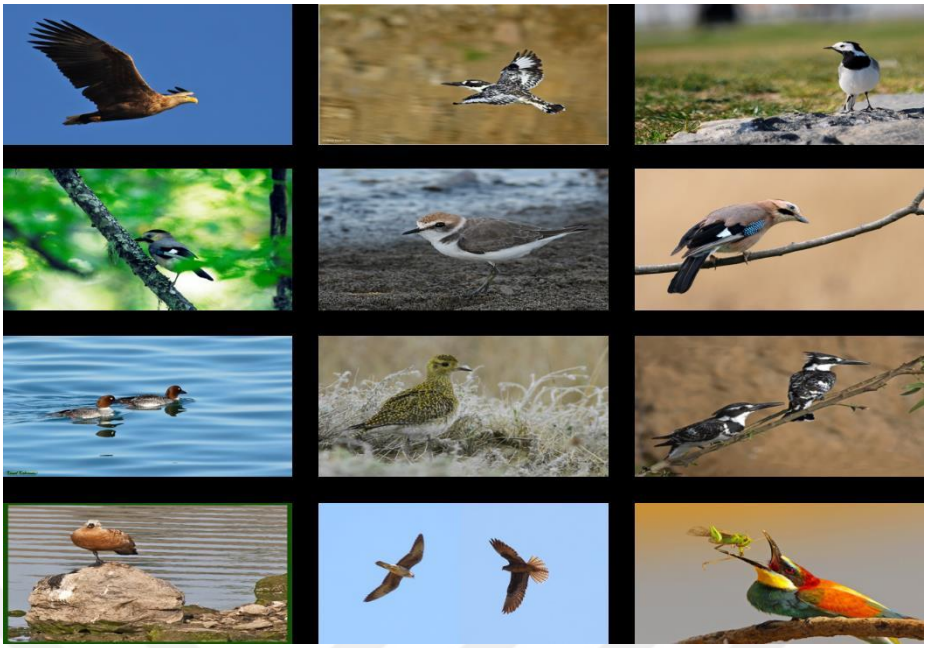

Figure 4.2. Some Examples of Positive Data for VJ Algorithm

Table 4.1. Parameter settings of Cascade Classifier for R-CNN

Number of Positive Images	3096
Number of Negative Images	6183
Number of Training Steps	15
Sample Width	36
Sample Height	24
Minimal Hit Rate	0,99

In this study, we perform experiments utilizing the R-CNN object detector, which uses the VGG-16 model. To initiate the model, we employ a pre-trained ImageNet model. Then, the model is utilized on bird datasets consisting of two classes: Birds and non-birds.

During the training of R-CNN, 10% of the data was defined as the validation set. R-CNN was trained and tested with an image set where the cascaded classifier and the Gaussian filter were applied in addition to the original bird images to compare the object detection results. The parameter settings of R-CNN are listed in Table 4.2.

Table 4.2. Parameter settings of R-CNN

Maximum Iteration	20
Learning Rate	0,0001
Batch Size	32
Patience	5
Loss	Binary Cross Entropy

4.1.1. Evaluation Metrics for R-CNN

This experiment used the Intersection over Union (IoU) as an evaluation metric between the ground truth and the predicted boundary boxes. IoU is an evaluation metric used to calculate the performance of an object detection model for a given sample set. It is mainly evaluated to measure the localization accuracy of the object detector concerning the ground truth and predicted boundary boxes. IoU is computed by dividing the overlap between the ground truth and predicted boxes by the area of their union, as shown in Figure 4.3 and Equation 4.1.

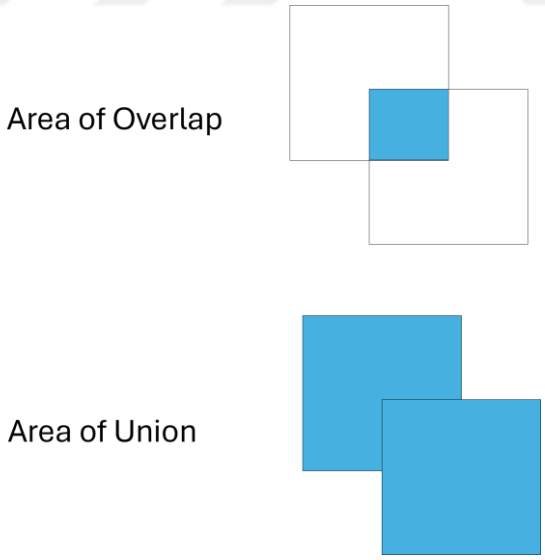

Figure 4.3. Computation of Intersection over Union

$$iou = \frac{Area\ of\ Intersection}{Area\ of\ Union} \quad (4.1)$$

4.1.2. Numerical Results

As a result of generating object proposals, the candidate regions with IoU greater than zero were combined into a single frame, and a Gaussian filter was applied to the non-object regions for each image sample.

In the detection phase of the anatomical landmarks, the threshold for the overlap over the union of the predicted boundary boxes by R-CNN is set to 0.5 (Arbalaez, Tuset, Baron, Marquez, and Malik, 2014; Uijlings, Sande, Gevers, and Smeulders, 2013). Therefore, the IoU results in this experiment are divided into three intervals: 0-0.3, 0.3-0.5, and 0.5-1.

Table 4.3 and 4.4 summarize the region proposals and sample statistics using proposed model and baseline R-CNN, respectively. They also compare the number of region proposals generated by the selective search of R-CNN.

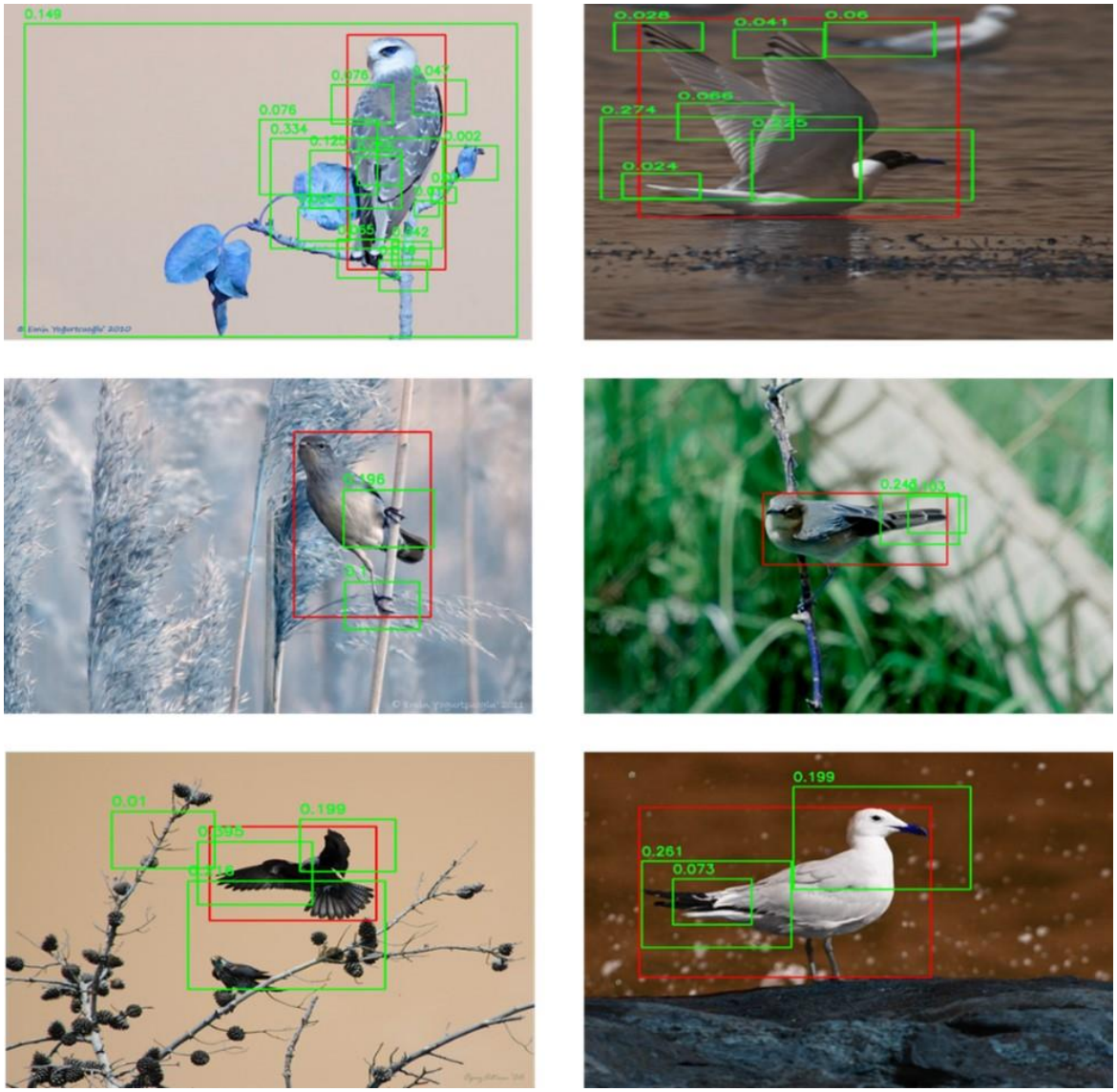

Figure 4.4. Visual Comparison of Haar Cascade Predictions (Blue) and Ground Truth (Red) on Test Images

Figure 4.4 shows the object candidate regions and IoU detected by the VJ Algorithm. Each weak object candidate region will be enclosed in a single frame and evaluated as the RoI in the next stage.

Table 4.3. Region Proposal and Sample Statistics for R-CNN using Proposed Model

Dataset	# ss regions	# positive	# negative
White-backed Woodpecker	**178717**	**426**	22000
Rustic Bunting	**187516**	**185**	11000
Short-toed Treecreeper	**382040**	**189**	**15000**
Little Crake	**535661**	**953**	42751
White-eared Bulbul	**122331**	**801**	21000

Table 4.4. Region Proposal and Sample Statistics for Baseline R-CNN

Dataset	# ss regions	# positive	# negative
White-backed Woodpecker	401468	99	22000
Rustic Bunting	205544	91	11000
Short-toed Treecreeper	415443	184	16000
Little Crake	891631	526	43000
White-eared Bulbul	196897	242	21000

Table 4.5 also demonstrates the reduction rate of region proposals (r) and the increase rate of positive training samples (i) given in Table 4.3 and 4.4. The experiment results demonstrate that the proposed method significantly augmented the number of positive samples in the training sets, increasing it by over 330%.

Table 4.5. Reduction Rate of Boundary Boxes (r) and Increase Rate of Positive Training Samples (i) using the Proposed Model

Dataset	r	i
White-backed Woodpecker	55,484%	330,303%
Rustic Bunting	8,771%	103,297%
Short-toed Treecreeper	8,040%	2,717%
Little Crake	39,923%	81,179%
White-eared Bulbul	37,871%	230,992%

Table 4.6 shows the number of test images and bounding boxes predicted as positive by R-CNN. As in the training samples, the number of positively predicted data in the test was increased by 726% compared to the results obtained with the original images.

Table 4.6. Number of Test Images and Boundary Boxes Predicted by R-CN

Dataset	Number of Images	Number of Boundary Boxes	
		Proposed	Original
White-backed Woodpecker	10	**178**	86
Rustic Bunting	5	**189**	128
Short-toed Treecreeper	8	**392**	152
Little Crake	18	**427**	185
White-eared Bulbul	10	**1215**	147

Table 4.7 and 4.8 also show the IoU distribution of the test results using the proposed model and baseline R-CNN, respectively.

Table 4.7. IoU Distributions of Predicted Boundary Boxes by R-CNN using the Proposed Model on Test Data

Dataset	IoU			
	0	0-0.3	0.3-0.5	>0.5
White-backed Woodpecker	0,00%	8,99%	25,84%	65,17%
Rustic Bunting	0,00%	12,17%	24,87%	62,96%
Short-toed Treecreeper	0,00%	4,84%	27,81%	67,35%
Little Crake	0,47%	11,00%	26,70%	61,83%
White-eared Bulbul	0,00%	19,67%	35,06%	45,26%

Table 4.8. IoU Distributions of Predicted Boundary Boxes by Baseline R-CNN on Test Data

Dataset	IoU			
	0	0-0.3	0.3-0.5	>0.5
White-backed Woodpecker	0,00%	24,42%	54,65%	20,93%
Rustic Bunting	0,00%	11,71%	34,37%	53,90%
Short-toed Treecreeper	1,32%	11,84%	28,29%	58,55%
Little Crake	0,00%	10,27%	34,59%	55,14%
White-eared Bulbul	1,36%	22,45%	32,65%	43,54%

As seen from the test results with the proposed model and baseline R-CNN in Table 4.7 and 4.8, the proposed method increases the percentage of IoU greater than 0,5 in all bird datasets.

Figure 4.5 shows some examples of region proposals reduced by theproposed model and extracted from the original bird images. The proposed method reduces the noise and the redundant sub-images, leading to filtering the search space. This improvement is promising, considering the computational cost of R-CNN. The proposed model also increases the number of positive boundary boxes in all of the training data and the most of test samples. The presented

model exhibits resilient and efficient results in accurately aligning the actual data with the predicted bounding boxes in detecting bird samples by increasing the rate of proposals with an IoU above 0,5 from 20,93% to 65,17%.

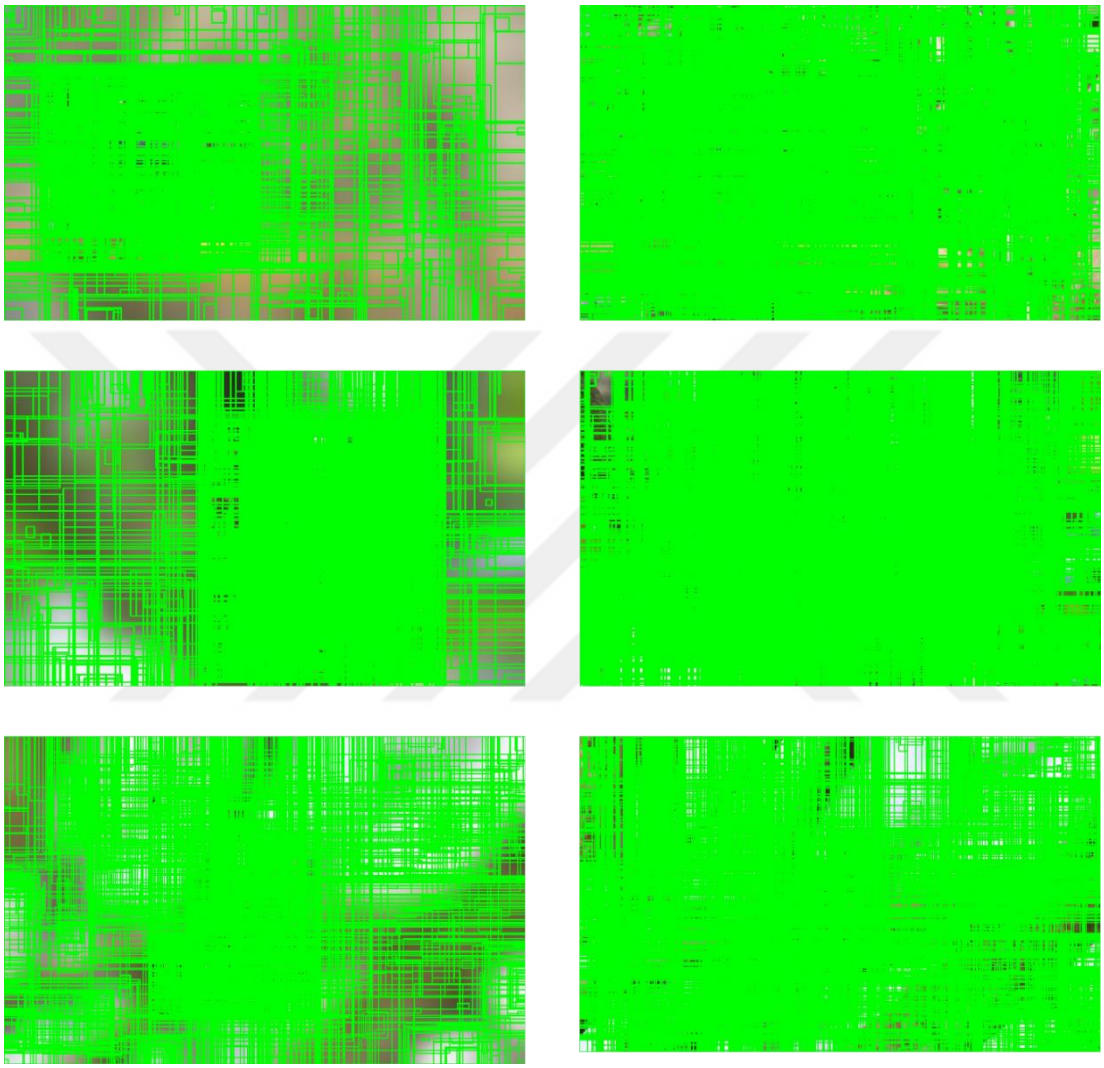

Figure 4.5. Comparison of Region Proposals Generated by Selective Search on Blurred (Proposed) and Original Images

Figure 4.6 to 4.15 show the classification loss of R-CNN for five bird datasets The learning rate was set to 0,0001 when training the R-CNN and kept constant. As shown in the Figures 4.6 to 4.15, the classification loss functions of R-CNN integrated with the proposed preprocessing approach converge faster towards the stopping condition.

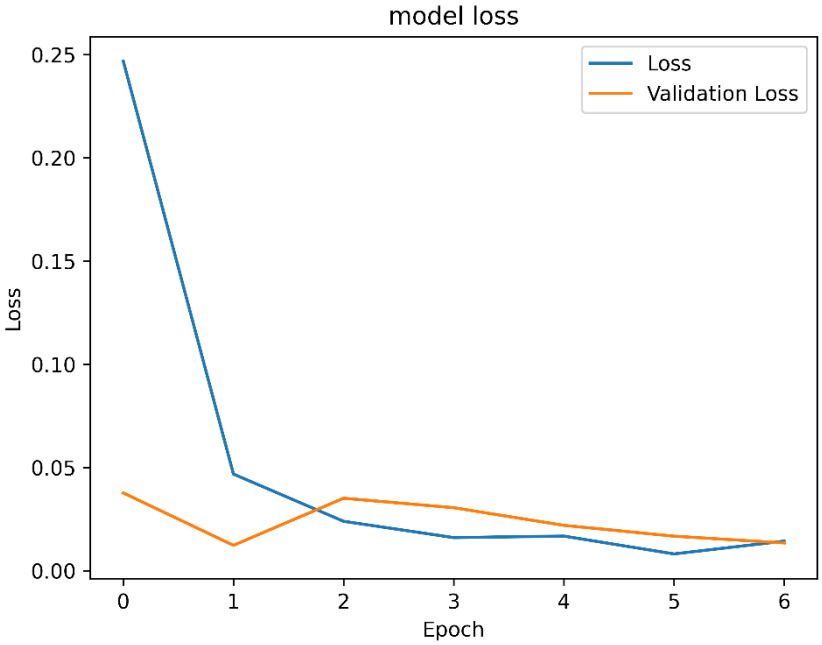

Figure 4.6. Training Loss of R-CNN on White-backed Woodpecker Dataset using Proposed Model

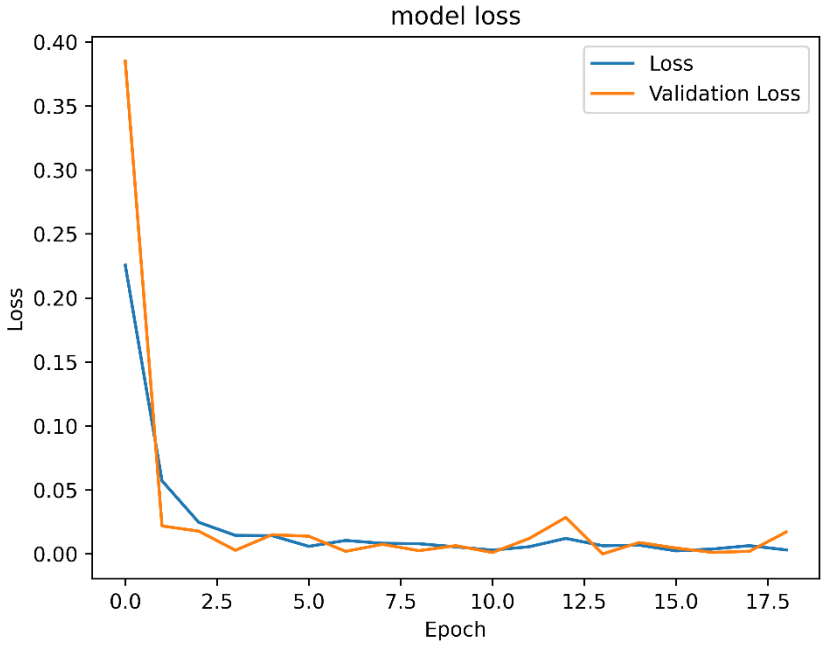

Figure 4.7. Training Loss of Baseline R-CNN on White-backed Woodpecker Dataset

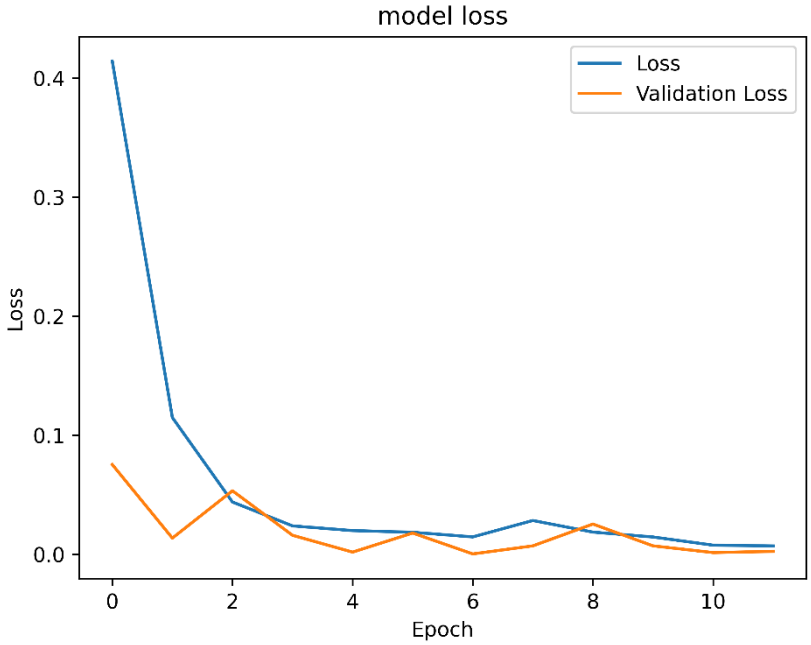

Figure 4.8. Training Loss of R-CNN on Rustic Bunting Dataset using Proposed Model

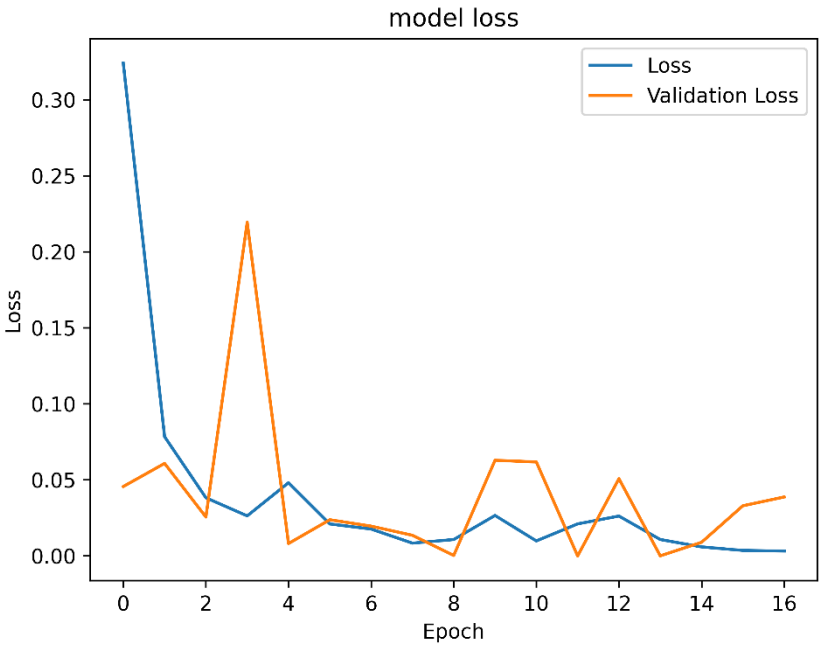

Figure 4.9. Training Loss of Baseline R-CNN on Rustic Bunting Dataset

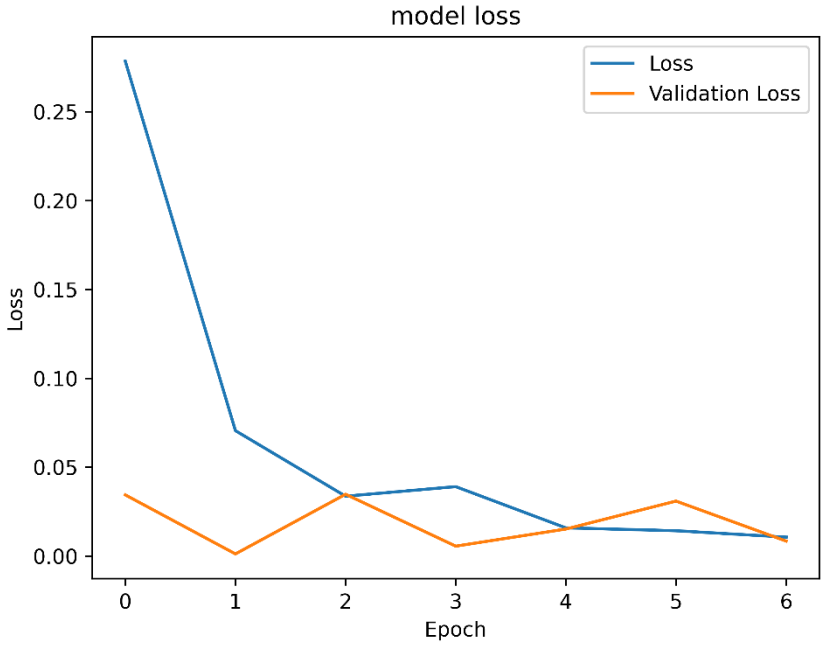

Figure 4.10. Training Loss R-CNN on Short-toed Treecreeper Dataset using Proposed Model

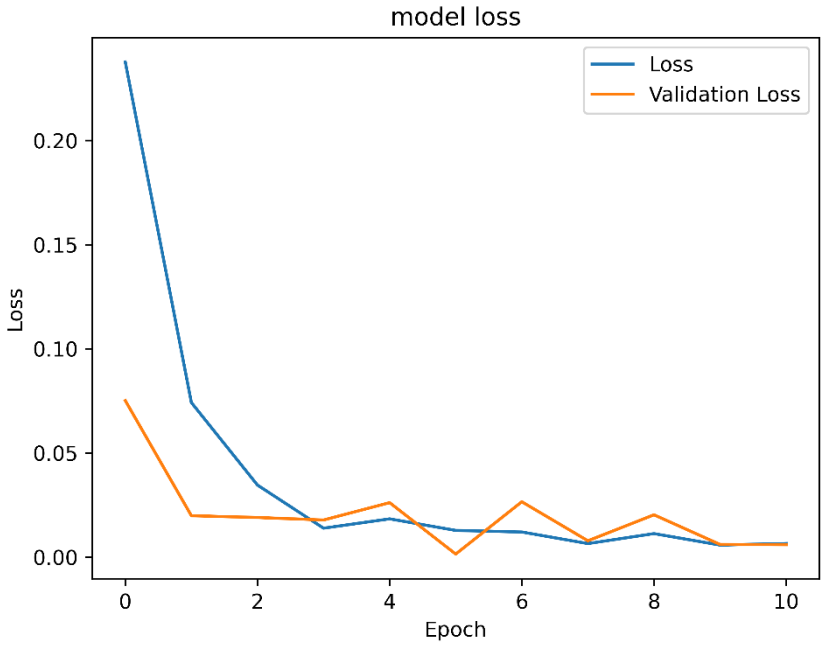

Figure 4.11 Training Loss of Baseline R-CNN on Short-toed Treecreeper Dataset

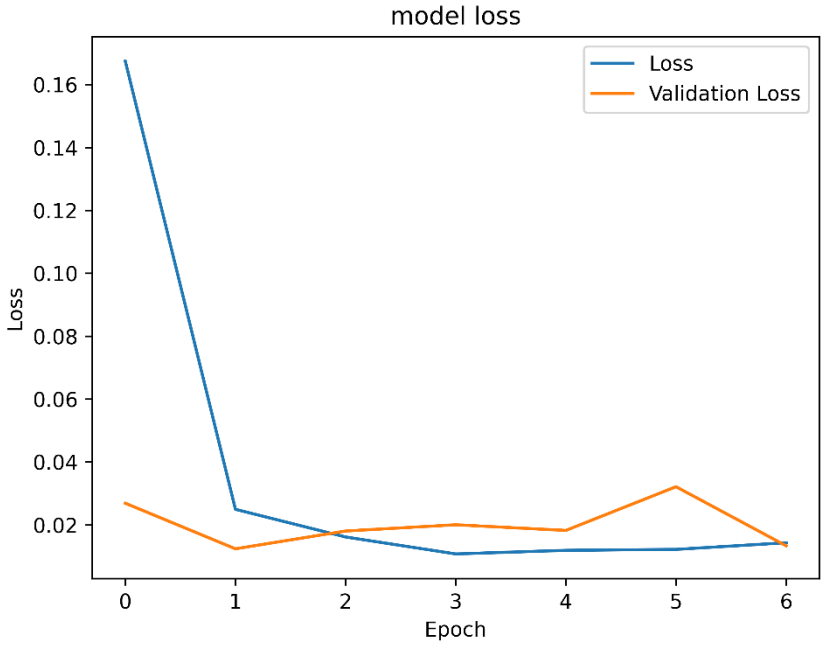

Figure 4.12. Training Loss of R-CNN on Little Crake Dataset using Proposed Model

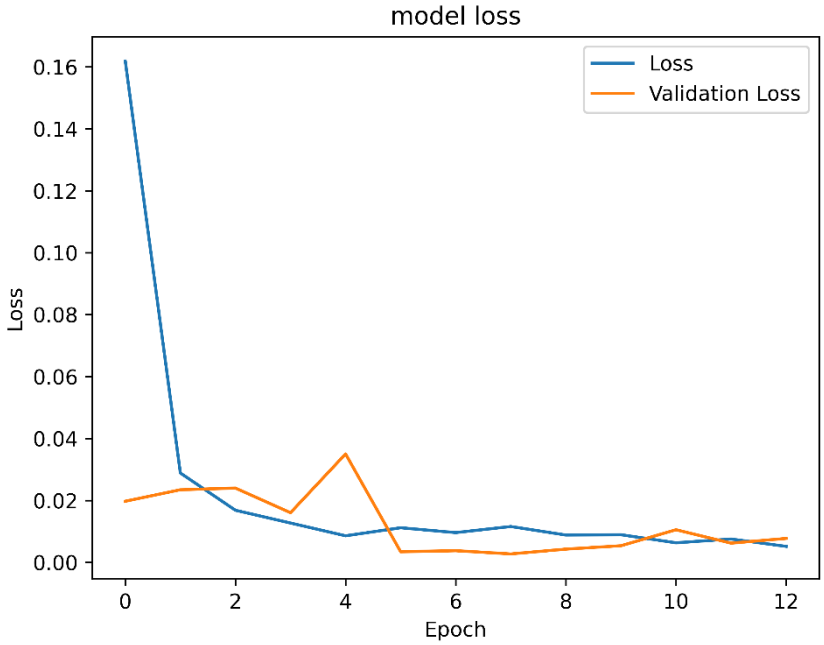

Figure 4.13. Training Loss of Baseline R-CNN on Little Crake Dataset

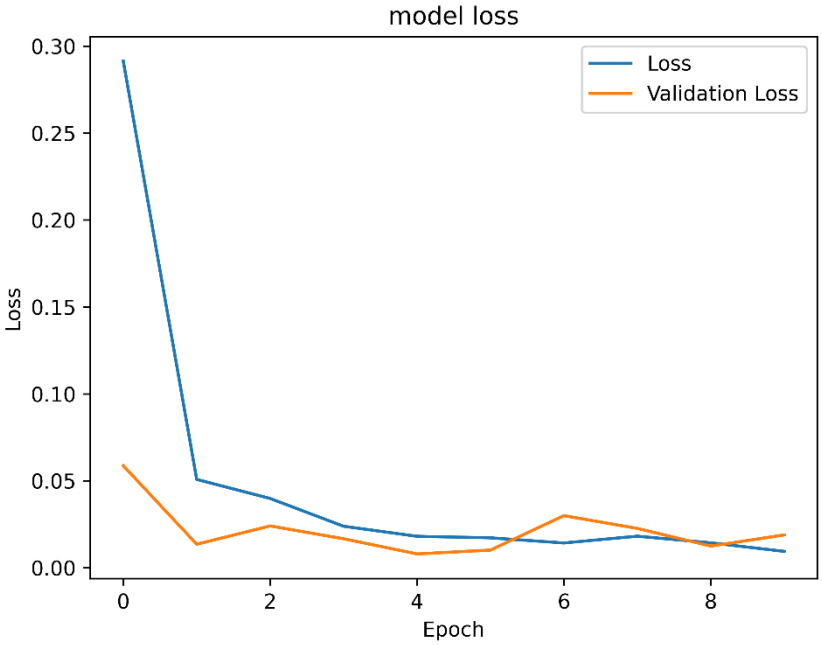

Figure 4.14. Training Loss of R-CNN on White-eared Bulbul Dataset using Proposed Model

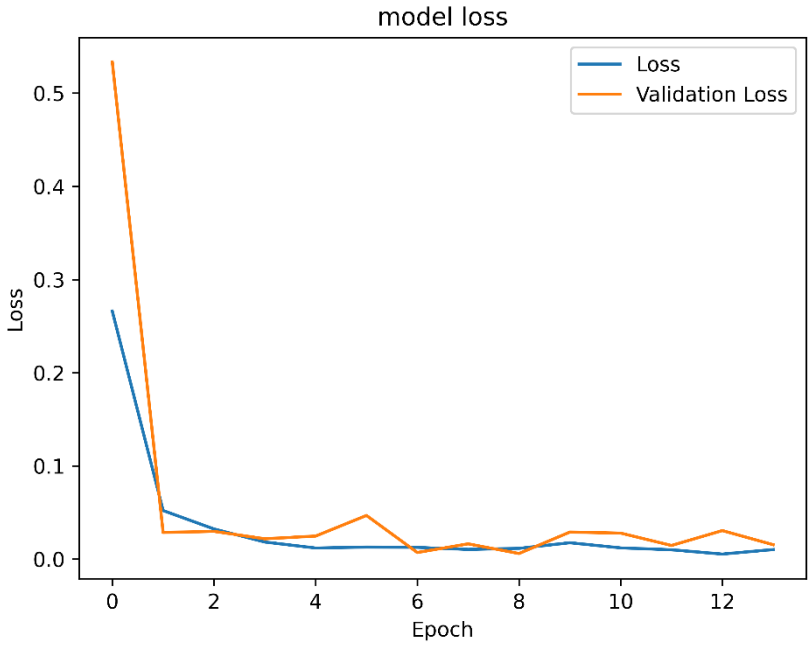

Figure 4.15. Training Loss of Baseline R-CNN on White-eared Bulbul Dataset

The proposed model contributed to R-CNN to convergence faster and improved the proposal generator algorithm's ability to identify more positive sub-images by efficiently filtering the search space. The findings indicate that adopting a two-stage approach significantly improves convergence speed increasing the number of correctly identified positive samples, and reducing the number of region proposals processed by the R-CNN. The proposed model achieves an impressive 55,484% reduction in region proposals and also increases the rate of positive boundary boxes by 330% during training and by 726% during testing.

To further evaluate the generalization capability of the proposed method, we extended our experiments beyond bird species and evaluated the model using a novel adaptive padding mechanism called Adaptive Padding 2 and a challenging publicly available dataset: Eastern Cottontail Rabbits Object Detection Dataset (roboflow, 2021). This dataset presents a diverse range of object categories with varying poses, occlusions, and natural backgrounds. Although the model was initially trained on bird images, it was employed on the rabbit dataset to evaluate its adaptability to other species. Since the proposed model was trained on bird data, the RoI was defined with a slightly broader margin to suppress noisy background regions during detection effectively. The dataset consists of 63 annotated rabbit images. Following the same protocol used for the bird datasets, 70% of the images were used for training and validation, while the remaining 30% were reserved for testing.

Table 4.9 presents the number of candidate regions generated by selective search and positive and negative proposals obtained with the proposed method and average region proposal generation time per image in seconds. The proposed method decreased the number of candidate boxes obtained using the original images over 7%, while the number of positive boundary boxes increased by over 41%. This indicates that the proposed region proposal strategy is more efficient at locating relevant object regions, thereby increasing the ratio of useful training samples. Consequently, an enhancement was achieved in the positive training sub-images fed to R-CNN. The proposed method also reduced the average proposal generation time per image from 4,631 seconds to 4,499 seconds in the training phase and from 4,525 seconds to 4,164 seconds in the testing phase. Although the time difference is slight, it reflects the improved processing efficiency of the proposed region proposal strategy. This efficiency becomes more

significant when scaling to larger datasets or real-time applications, where cumulative time savings can have a substantial impact.

Table 4.9. Region Proposal and Sample Statistics for R-CNN on Eastern Cottontail Rabbits Object Detection Dataset

Model	# ss regions	# positive	#negative	Proposal generation time(second)	
				Train	Test
Proposed	**152858**	**1516**	43149	**4,449**	**4,164**
Original	164584	1071	40630	4,631	4,525

The IoU distributions produced by the proposed and baseline models on the Eastern Cottontail Rabbits dataset are given in Table 4.10. The proposed method yielded a higher proportion of bounding boxes with an IoU greater than 0.5 (52,83%) compared to the original model (48,25%), indicating a better intersection with the ground truth annotations. Additionally, the proposed model exhibited a more balanced distribution across mid-range IoU intervals (0.3–0.5), while maintaining the same false positive rate (0% for both models). Although the original model generated slightly more boxes in the 0.3–0.5 range, the proposed model's improvement in the high-IoU category offers more precise and reliable object localization, which is critical for downstream tasks such as classification and counting.

Table 4.10. IoU Distributions of Predicted Boundary Boxes by R-CNN on Eastern Cottontail Rabbits Object Detection Dataset

Model	IoU			
	=0(FP)	0-0.3	0.3-0.5	>0.5
Proposed	0.00%	16.09%	31.08%	**52.83%**
Original	0.00%	13.48%	38.27%	48.25%

Figure 4.16 to 4.17 show the classification loss of R-CNN for Eastern Cottontail Rabbits Object Detection dataset. The learning rate was set to 0,0001 when training the R-CNN and constant. As shown in Figure 4.16, the classification loss function of R-CNN integrated with the proposed preprocessing steps converges faster towards the stopping condition. While the original model required 18 epochs to stabilize, the proposed model reached optimal performance in just 12 epochs. This shows that the proposed pipeline not only improves accuracy but also accelerates convergence during training.

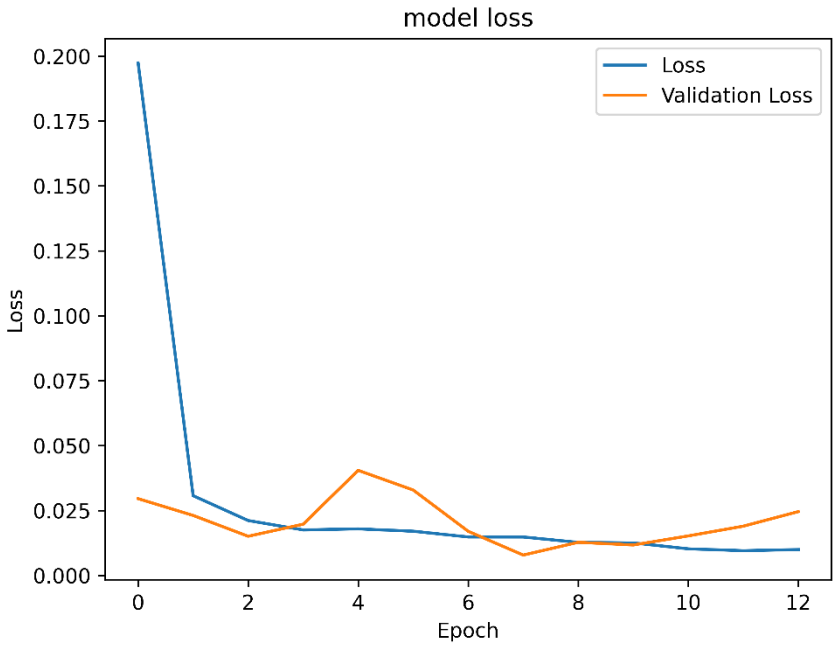

Figure 4.16. Training Loss of R-CNN on Eastern Cottontail Rabbits Object Detection Dataset using Proposed Model

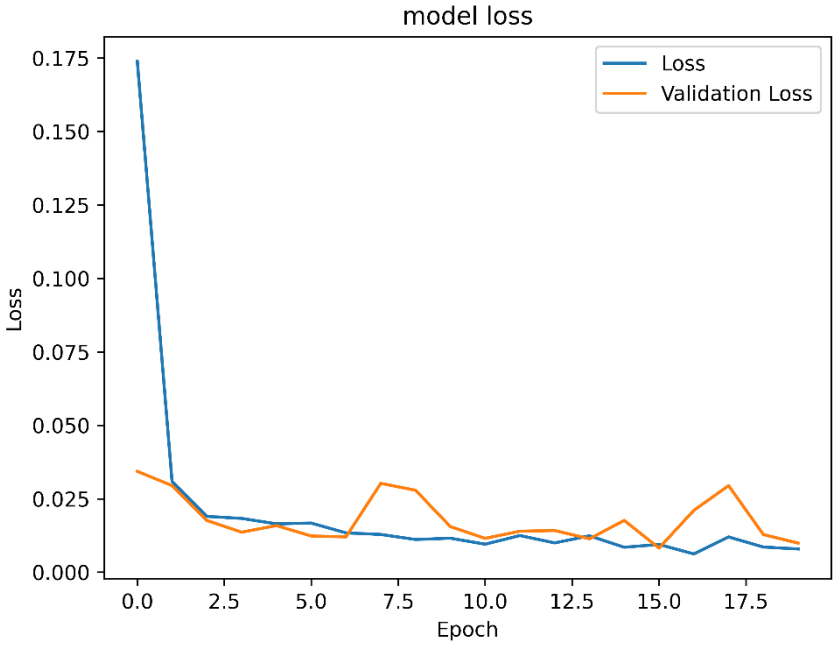

Figure 4.17. Training Loss of R-CNN on Eastern Cottontail Rabbits Object Detection Dataset using Original Data

4.2. Results with YOLOv8

In this study, 3096 bird imags of 61 species obtained from TRAKUS were randomly selected to train the VJ Model. Additionally, a collection of 12400 nature images without bird objects was sourced from the internet and marked as negative. The parameter settings of the VJ algorithm are given in Table 4.11.

Table 4.11. Parameter Settings of Cascade Classifier for YOLOv8

Number of Positive Images	3096
Number of Negative Images	12400
Number of Training Steps	20
Sample Width	36
Sample Height	24
Minimal Hit Rate	0,99

Following the generation of object proposals through the trained VJ model, the candidate regions for each of the 351 images were combined into a single frame within each image. A padding mechanism was subsequently applied in RoI. Furthermore, a Gaussian filter was employed for the non-object regions for each image sample.

Before training the YOLOv8, the images were resized to a fixed dimension of 800*800 pixels. To increase the diversity of the training data, data augmentation techniques were applied. These included 15% grayscale transformation to the images and performing horizontal flips. The dataset was split into three: 70% training, 20% validation, and 10% test. The training parameters of YOLOv8 are shown in Table 4.12.

Table 4.12. Parameter Settings of YOLOv8

Number of Classes	6
Maximum Iteration	300
Number of Images	351
Patience	50
Batch Size	16
Image Size	800*800
Learning Rate	0,01

Table 4.13 gives the training and validation results obtained employing the proposed preprocessing method and the original images with YOLOv8. The proposed model achieved in lower box loss (0,512) and classification loss (0,496) compared to the results obtained with the original data. The proposed model predicted more accurate bounding boxes during training. Further, the proposed model caused a decline in Distribution Focal Loss (dfl_loss), presenting improved bounding box regression during the training stage. Precision improved from 0,867 to 0,930, meaning fewer false positives. Recall increased from 0,916 to 0,980. This indicates that YOLOv8 found more true positives. Additionally, it can be seen that the proposed method provides a balance between precision and recall. In validation, the proposed model significantly decreased box loss (0,868) and classification loss (0,498). This means better generalization to the validation data. The dfl_loss decreased from 2,096 to 1,662, further demonstrating improved accuracy in bounding box regression.

Table 4.13. Comparison of Metrics using Proposed Model and Baseline YOLOv8

Metrics	Proposed	Original
Train/box loss	**0,512**	0,677
Train/classification loss	**0,496**	0,564
Train/dfl_loss	**1,077**	1,170
Precision	**0,930**	0,867
Recall	**0,980**	0,916
mAP50	**0,984**	0,954
mAP50-95	**0,781**	0,633
Validation/box loss	**0,868**	1,247
Validation/classification loss	**0,498**	0,897
Validation/dfl_loss	**1,662**	2,096

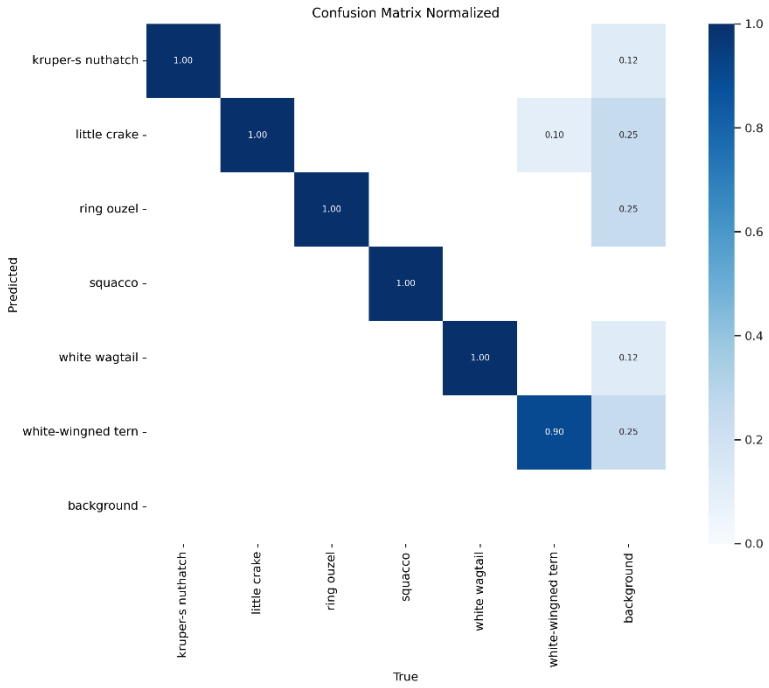

Figure 4.18. Normalized Confusion Matrix of YOLOv8 on Validation Data using Proposed Model

YOLOv8 achieved a mean Average Precision (mAP) of 0,984 at an IoU threshold of 0.50 using the proposed model. This is an improvement over the 0,954 mAP obtained with the original images. Moreover, the mAP50–95 increased from 0,634 to 0,780. This indicates the model performs better in difficult situations, such as detecting overlapping or small objects.

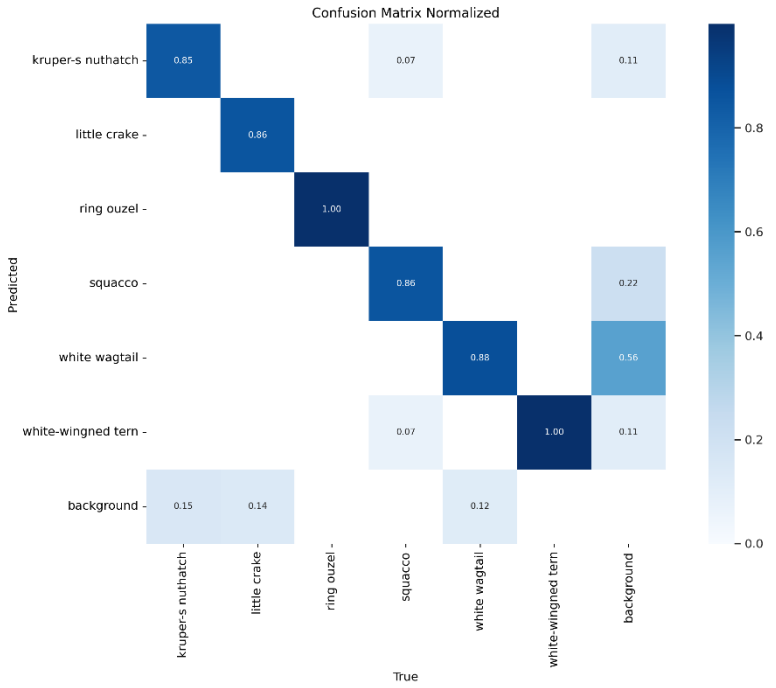

Figure 4.19. Normalized Confusion Matrix of Baseline YOLOv8 on Validation Data

Overall, Figure 4.18 emphasizes a more robust detection model with enhanced generalization, particularly in challenging scenarios like distinguishing objects and backgrounds.

Figure 4.18 and 4.19 show the normalized confusion matrices of YOLOv8 using the proposed method and the original images on the validation sets. The experimental results from the proposed method achieved 100% accuracy in nearly all classes. On the other hand, the results obtained with the original images only achieved 100% accuracy in two classes. The classification results obtained using baseline YOLOv8 show that the detector cannot correctly distinguish between certain classes and background noise. This may be due to insufficient training data or overlapping features.

Figures 4.20 and 4.21 show the Precision-Recall curves obtained using the proposed method and the original images, respectively. As can be seen in Figure 4.20, the proposed method achieved balanced results between 0,962 and 0,995 on a class basis. The results obtained using the original images fell behind on proposed model, especially in the White-wagtail and Kruper's

Nuthatch datasets. This shows that the variance between classes using the original images is less than the proposed model.

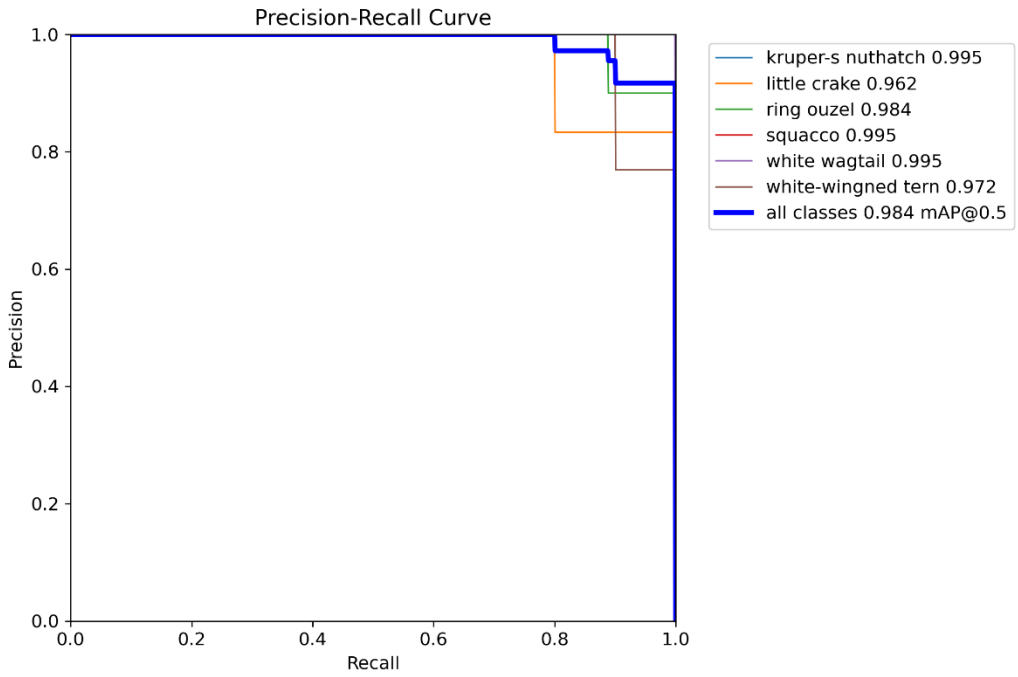

Figure 4.20. Precision-Recall Curve of YOLOv8 using Proposed Model

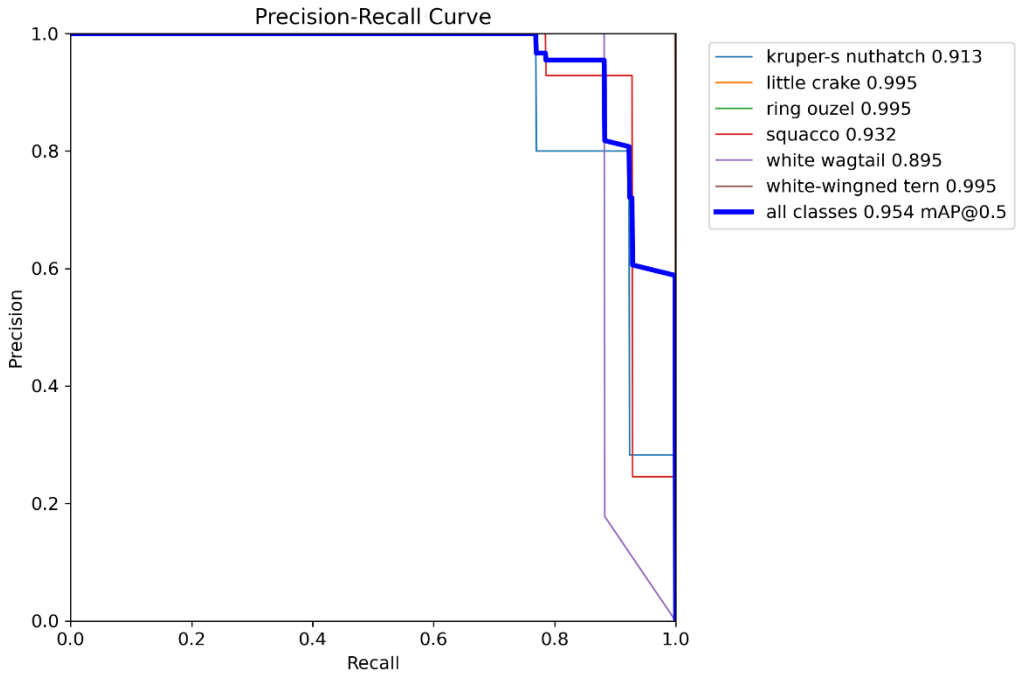

Figure 4.21. Precision-Recall Curve of Baseline YOLOv8

Figures 4.22 and 4.23 show the YOLOv8 F1 curves obtained with the proposed method and the original images. When examining the results obtained with the proposed method, it can be seen that the model achieves an F1 value of 0,95 at a confidence level of 0,618. The fact that the F1 value remains constant as the confidence level increases indicates the stability of the model. In contrast, the F1 value achieved with the original images is 0,88 and the confidence level is 0,264. When the confidence level exceeds 0,5, the F1 value drops.

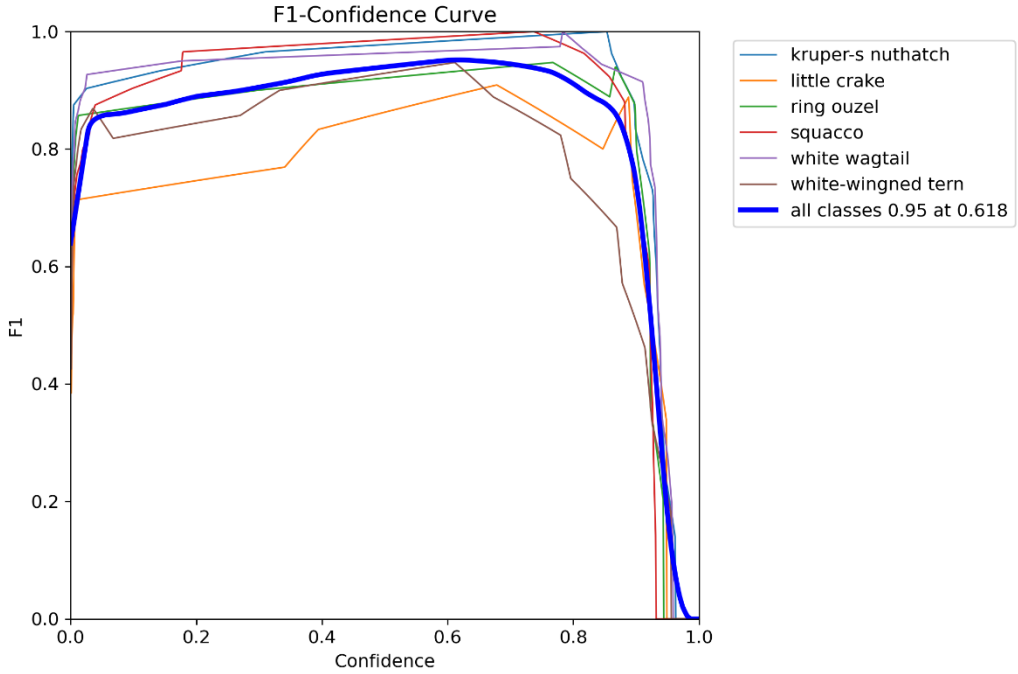

Figure 4.22. F1 Curve of YOLOv8 using Proposed Model in the Training Phase

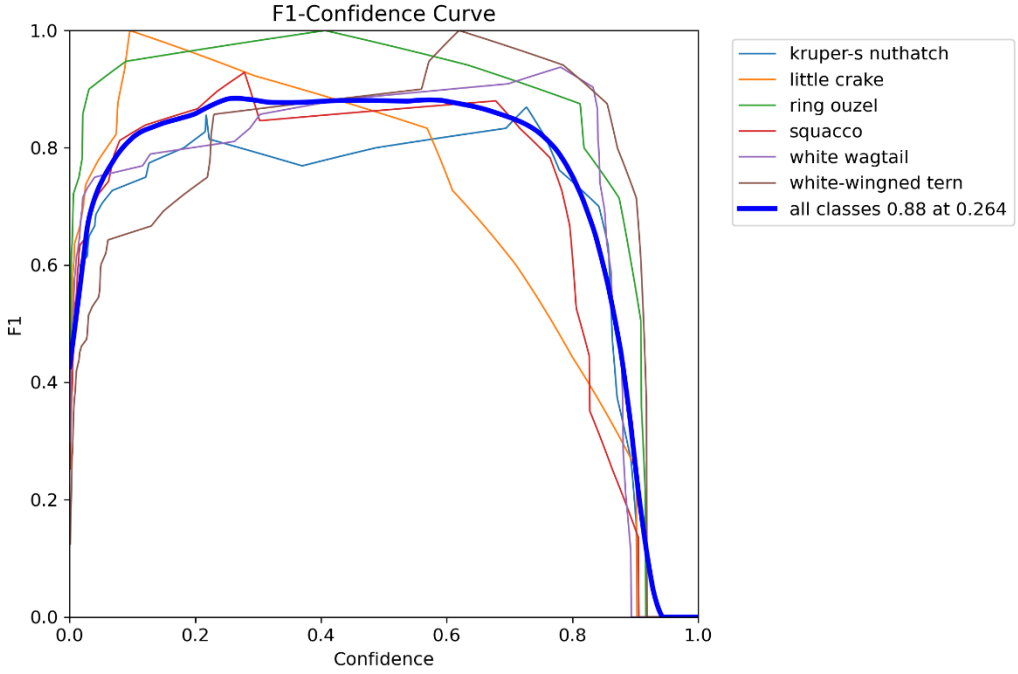

Figure 4.23. F1 Curve of BaselineYOLOv8 in the Training Phase

Tables 4.14 and 4.15 show the results of the proposed method and the original images including number of instances, precision, recall, mAP50 and mAP50-95 for each class. The proposed method outperformed the original YOLOv8 in 4 classes in terms of precision. In addition, the proposed model achieved 100% recall in 5 classes. Looking at the results in terms of mAP50, an improvement of over 10% was achieved, especially in the White Wagtail dataset. The mAP50-95 results show that the proposed method is clearly superior in 5 classes. It is particularly striking that the improvement in the Squacco dataset is over 93%.

Table 4.14. Class-based YOLOv8 Validation Results using Proposed Model

Class	Instances	Precision	Recall	mAP50	mAP50-95
Kruper-s Nuthatch	14	**0,972**	**1**	**0,995**	**0,762**
Little Crake	5	0,810	**1**	0,962	**0,769**
Ring Ouzel	9	0,875	**1**	0,984	0,724
Squacco	14	**0,986**	**1**	**0,995**	**0,817**
White Wagtail	19	**0,938**	**1**	**0,995**	**0,765**
White-wingned Tern	10	**1**	0,881	0,972	**0,851**

Table 4.15. Class-based YOLOv8 Validation Results using Original Data

Class	Instances	Precision	Recall	mAP50	mAP50-95
Kruper-s Nuthatch	14	0,780	0,916	0,913	0,584
Little Crake	5	**1**	0,817	**0,995**	0,682
Ring Ouzel	9	**0,959**	0,869	**0,995**	**0,773**
Squacco	14	0,927	**1**	0,932	0,422
White Wagtail	19	0,774	0,882	0,895	0,629
White-wingned Tern	10	0,760	**1**	**0,995**	0,716

The results given in Table 4.13 are also illustrated in Figure 4.24 and 4.25.

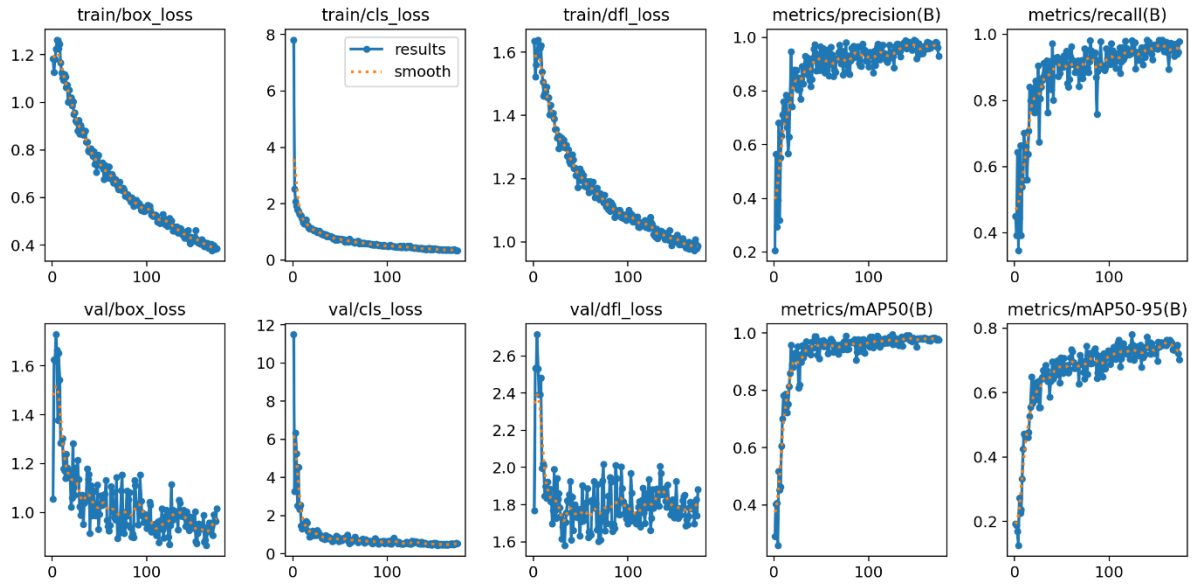

Figure 4.24. Overall results of YOLOv8 on Training and Validation Sets using Proposed Model

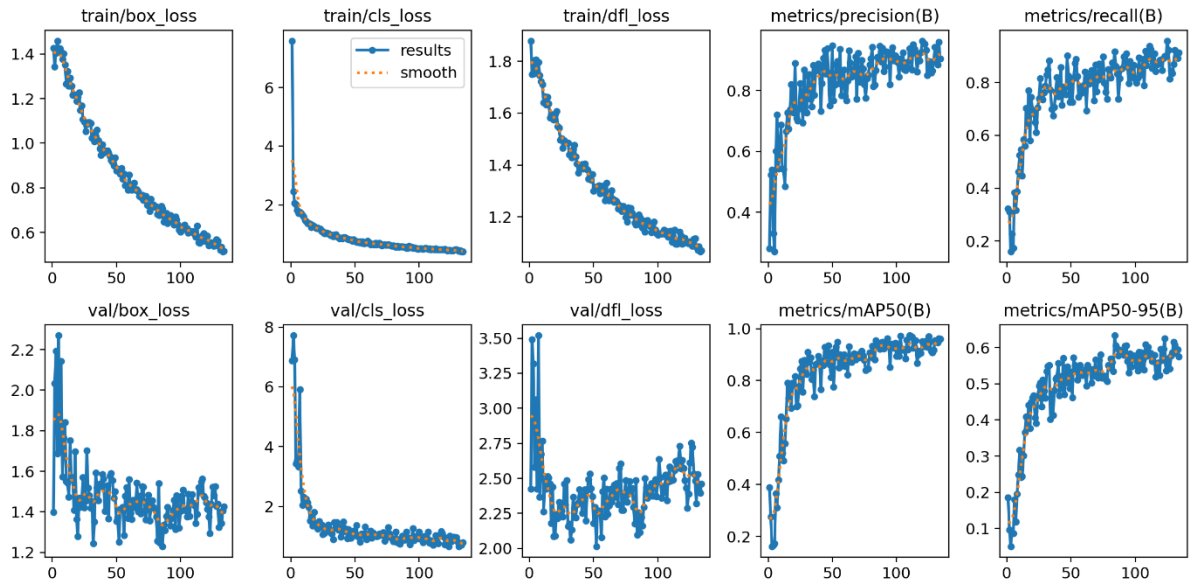

Figure 4.25. Overall Results of BaselineYOLOv8 on Training and Validation Sets

Figures 4.26 and 4.27 show some examples of detected objects in the training phase using the proposed model and the original images, respectively.

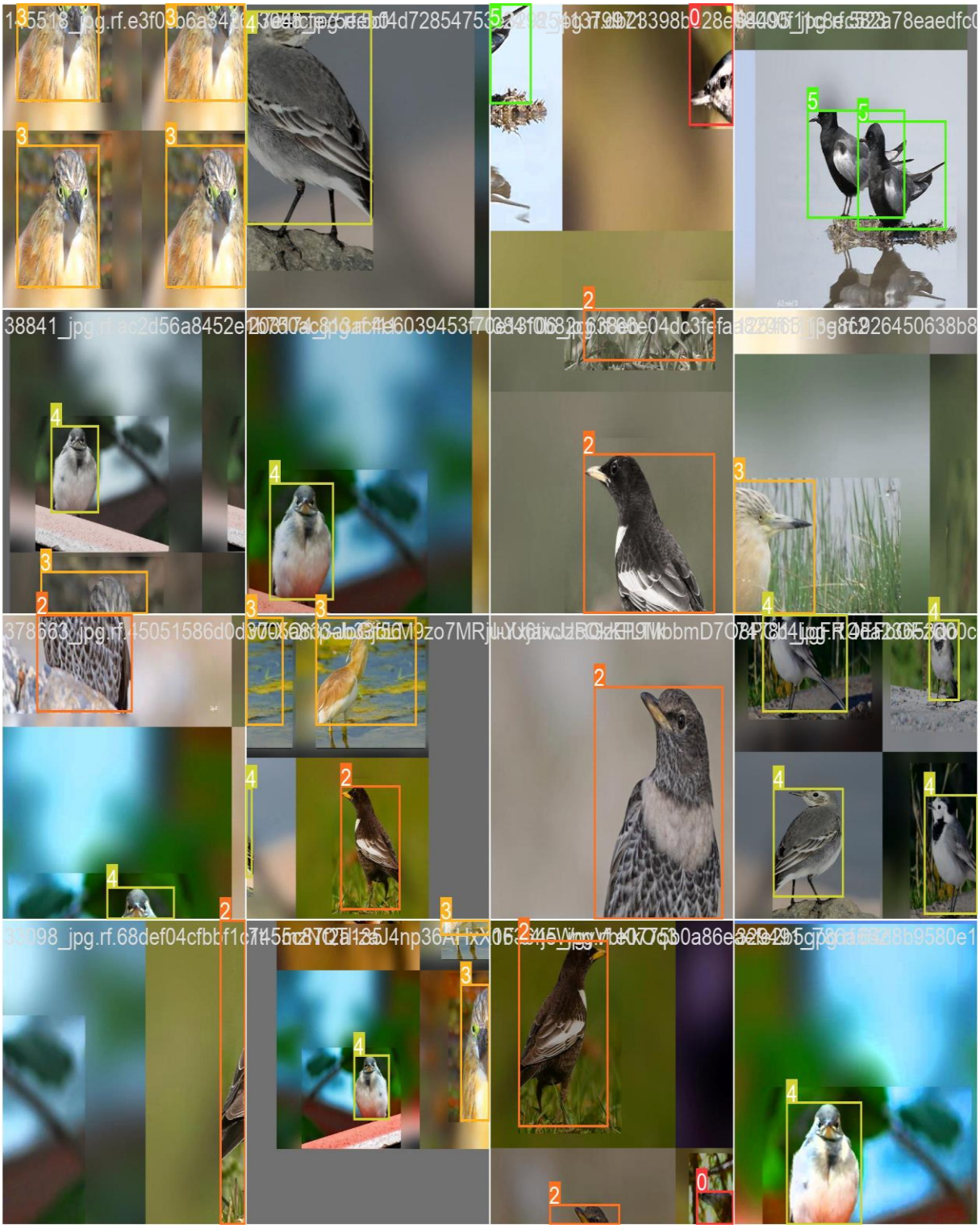

Figure 4.26. Some Examples of Detected Object in the Training of YOLOv8 using Proposed Model

Figure 4.27. Some Examples of Detected Object in the Training of Baseline YOLOv8

Figures 4.28 and 4.29 show predicted objects and boundary boxes in the validation phase using the proposed method and the ground truth data, respectively. The comparison of the figures shows that the proposed method detects the bird species completely and correctly.

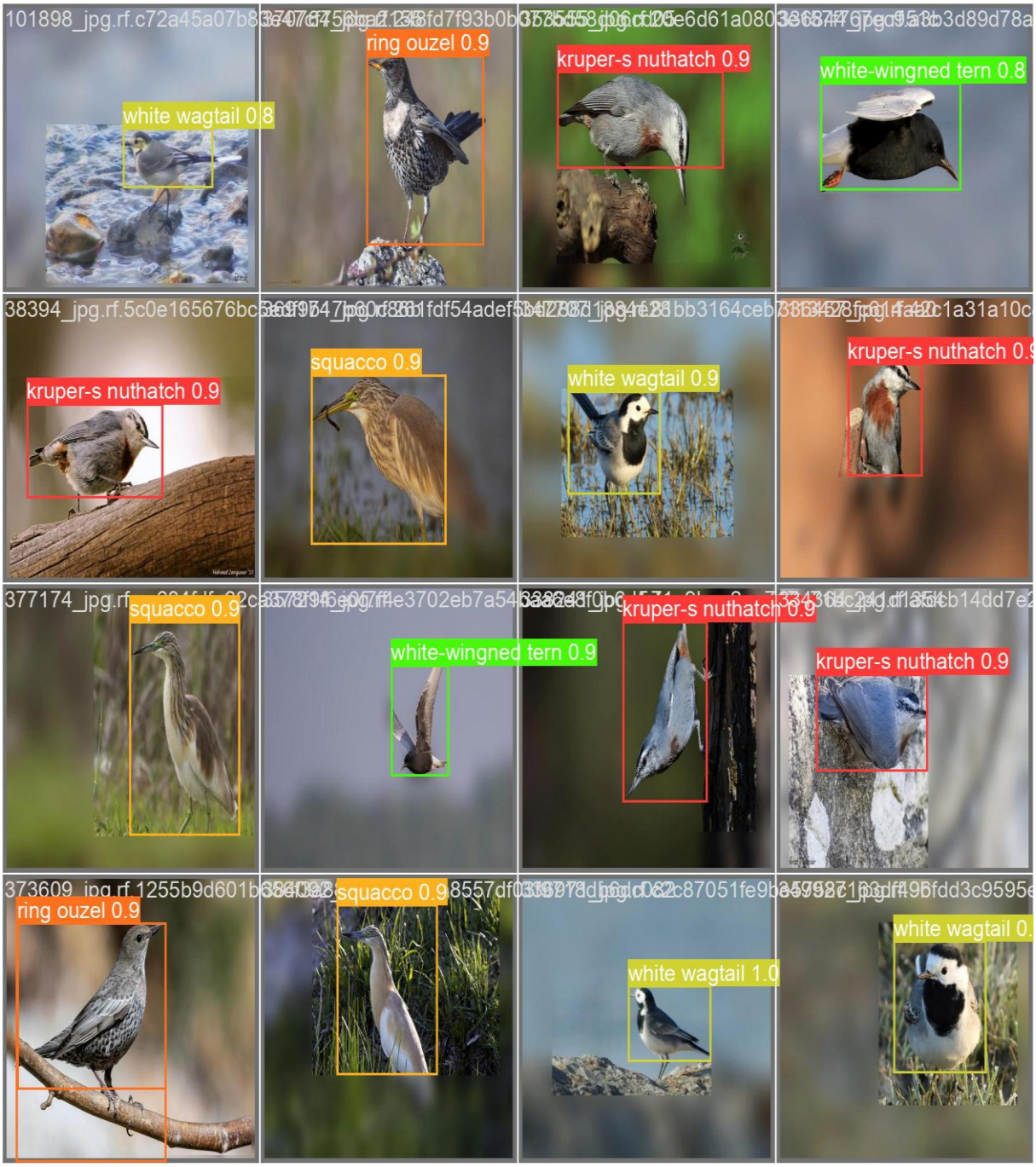

Figure 4.28. Some Examples of Detected Objects in the Validation Phase of YOLOv8 using Proposed Model

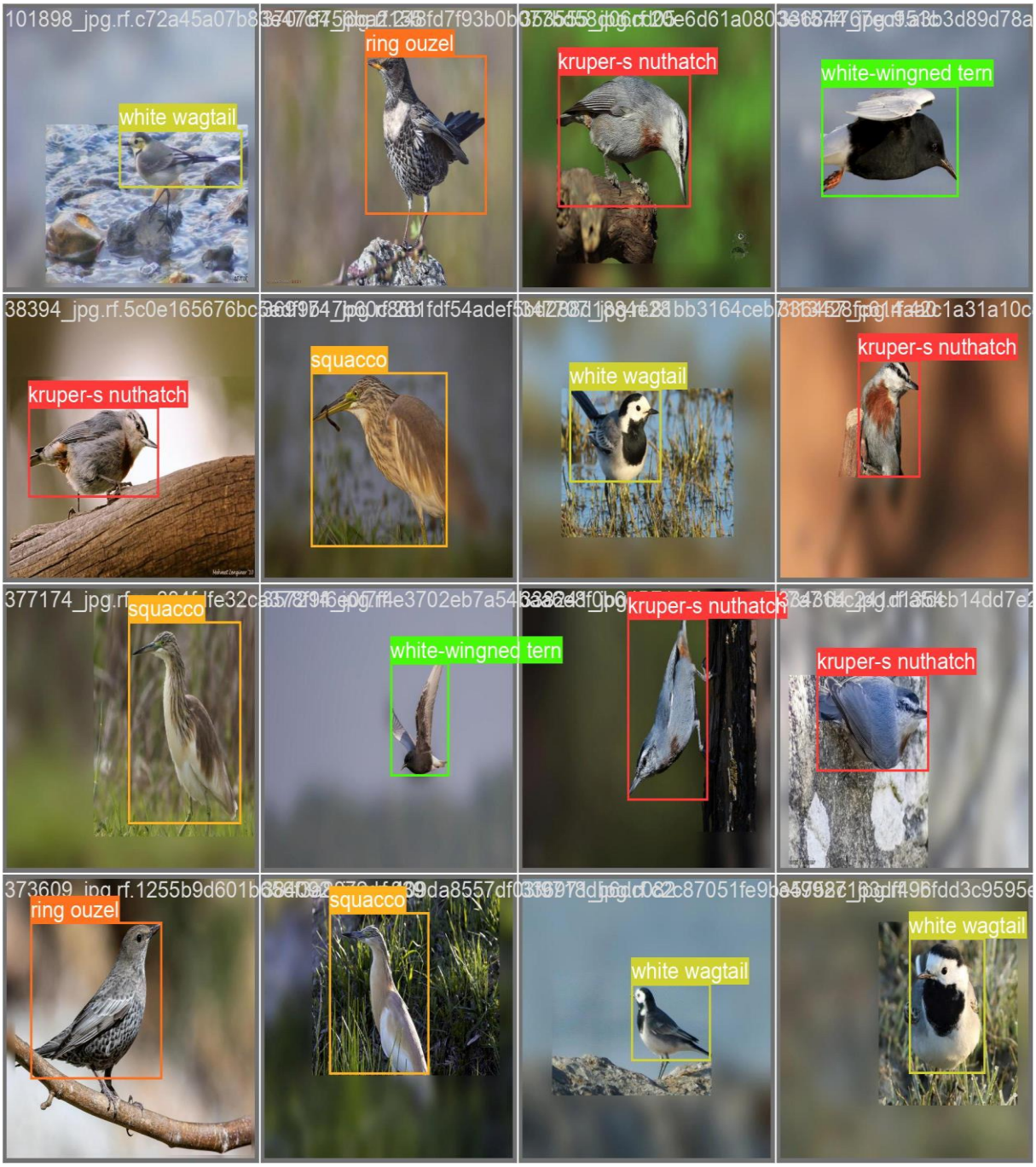

Figure 4.29. Ground Truth Validation Data for YOLOv8 using Proposed Model

Figures 4.30 and 4.31 show predicted objects and boundary boxes in the validation phase using the original bird images and the ground truth data, respectively. When the figures are compared, it is seen that there are certain objects that YOLOv8 fails to detect and classifies incorrectly in the original bird images.

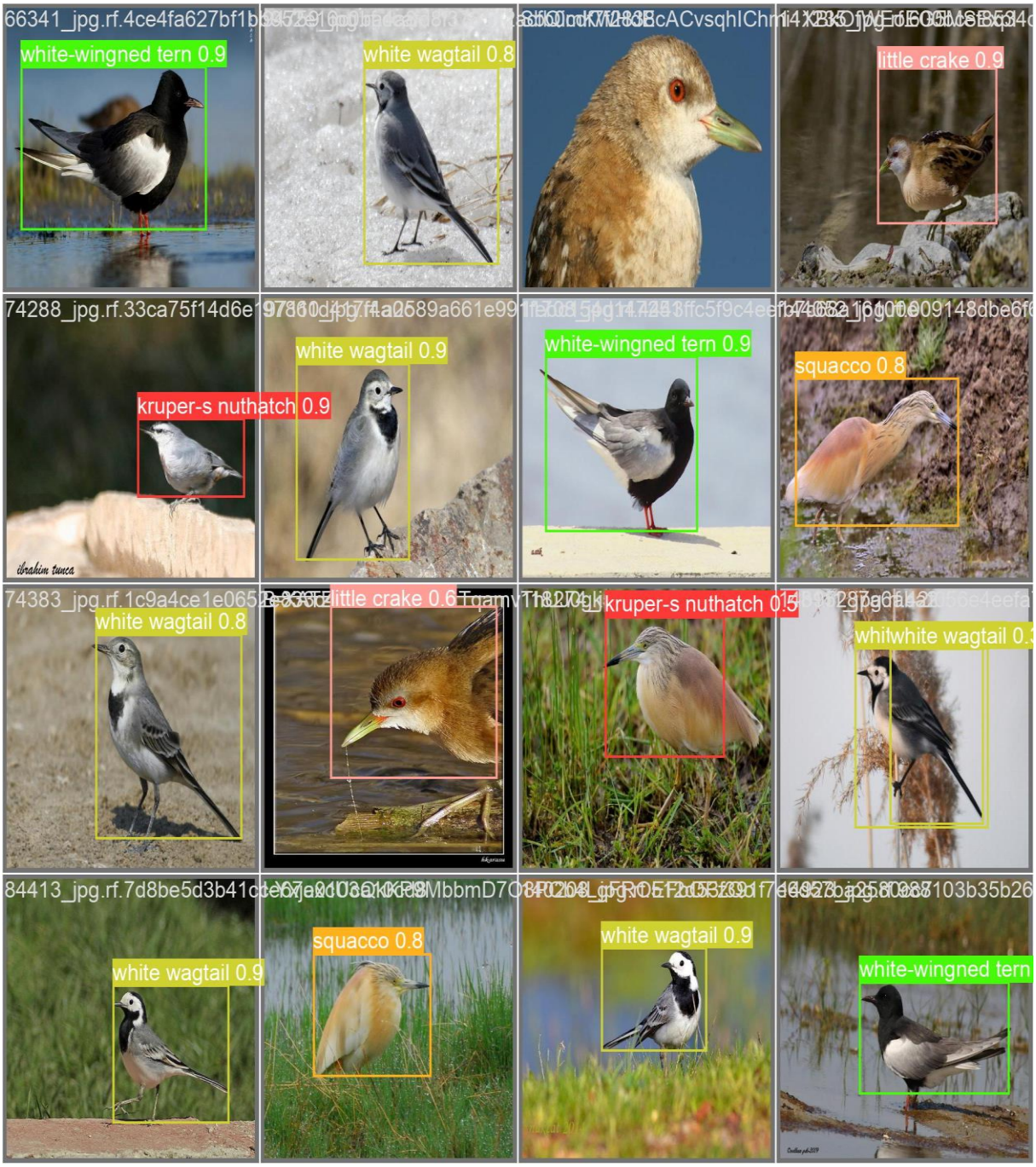

Figure 4.30. Some Examples of Detected Objects in the Validation Phase of Baseline YOLOv8

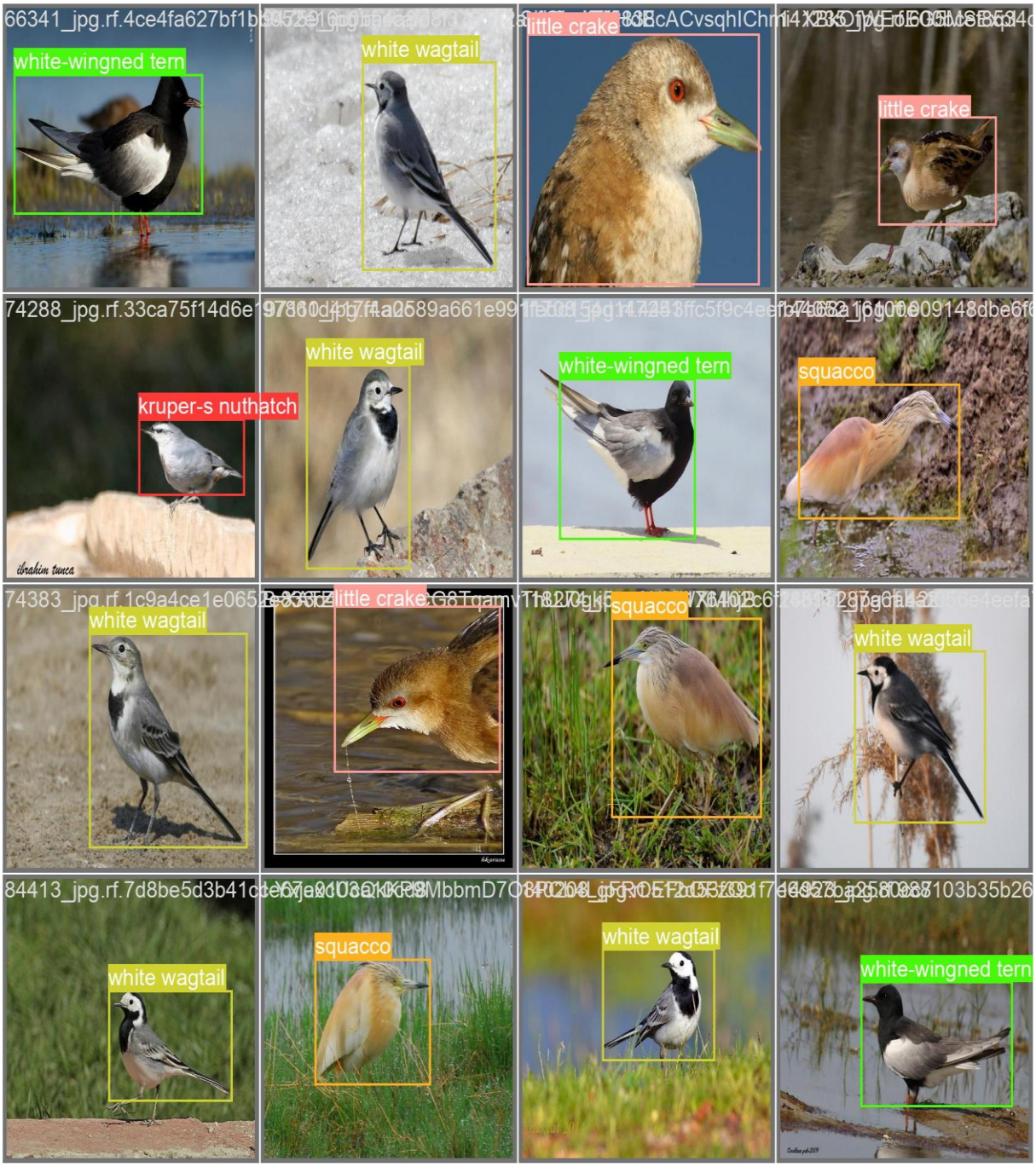

Figure 4.31. Ground Truth Validation Data for Baseline YOLOv8

Figure 4.32 and 4.33 show the normalized confusion matrices of YOLOv8 using the proposed method and the original images on the test data. The experimental results from the proposed method achieved 100% accuracy in 4 classes. On the other hand, the number of classes that were classified 100% accuracy using the original images is 3. When the original images are employed, the images belonging to the Squacco class showed worse performance compared to the proposed method. The proposed method detected the Squacco class with 100% accuracy, while it produced reduntant candidate boxes belonging to the background and White-wagtail Tern classes. In case the original images were used, the classification accuracy of YOLOv8 in the Squacco class dropped to 86%.

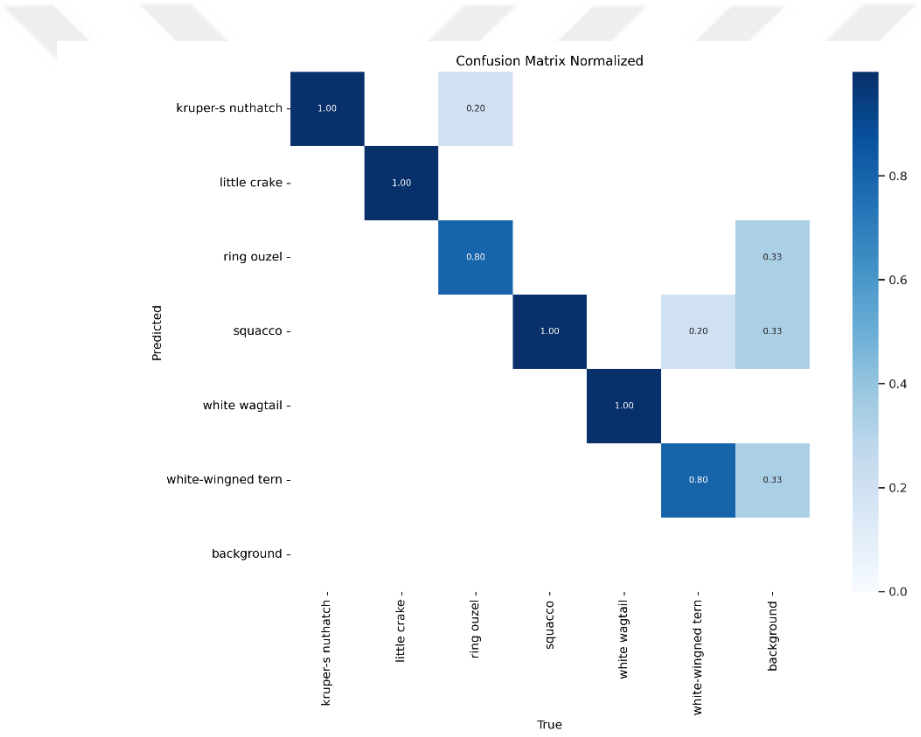

Figure 4.32. Normalized Confusion Matrix of YOLOv8 on Test Data using Proposed Model

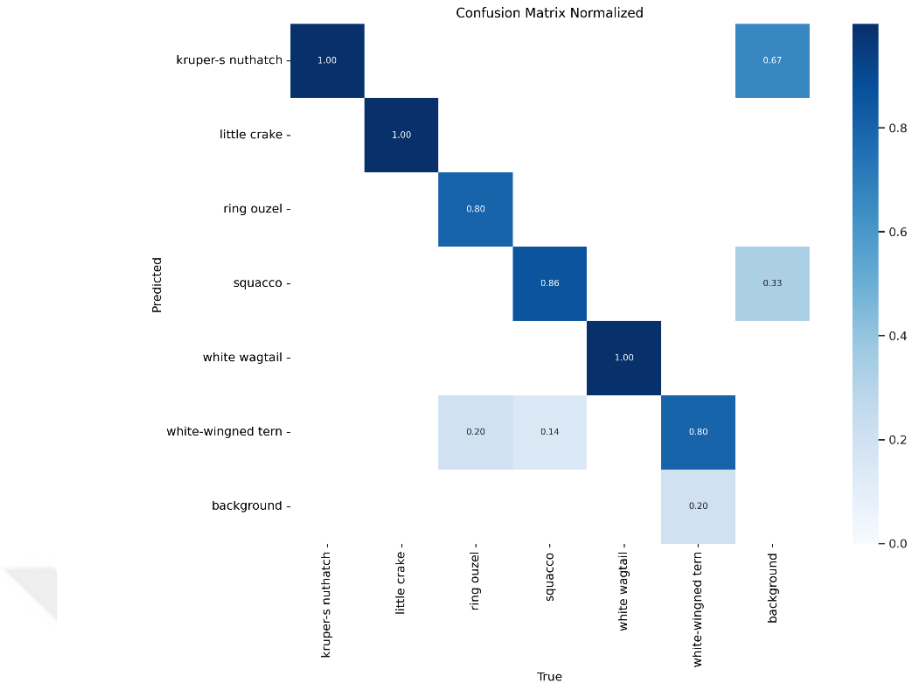

Figure 4.33. Normalized Confusion Matrix of Baseline YOLOv8 on Test Data

Figures 4.34 and 4.35 show the Precision-Recall curves obtained using the proposed method and the original images in test phase, respectively. In Figure 4.34, the curves are remarkably flat; with the exception of the Squacco class, the precision results are above 0,95. This shows that the model makes reliable predictions with high confidence. On the other hand, in Figure 4.35, the precision of the Squacco class is quite low, that is 0,71. In some classes, the precision decreases rapidly while the recall increases. This could indicate that the model also accepts predictions with low confidence.

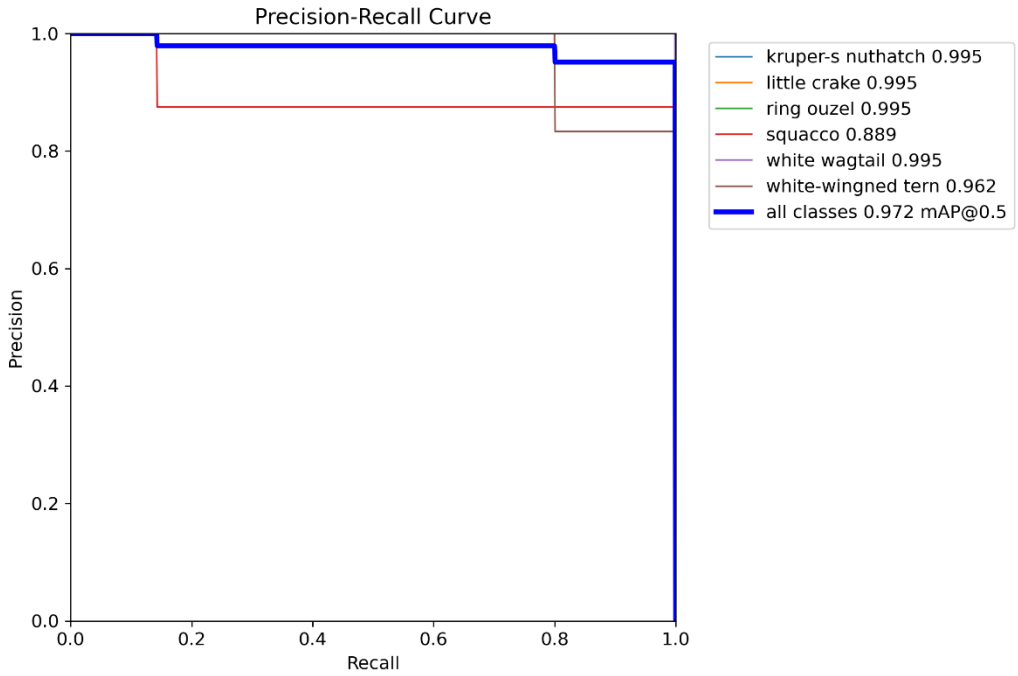

Figure 4.34. Precision-Recall Curve of YOLOv8 using Proposed Model in Test Phase

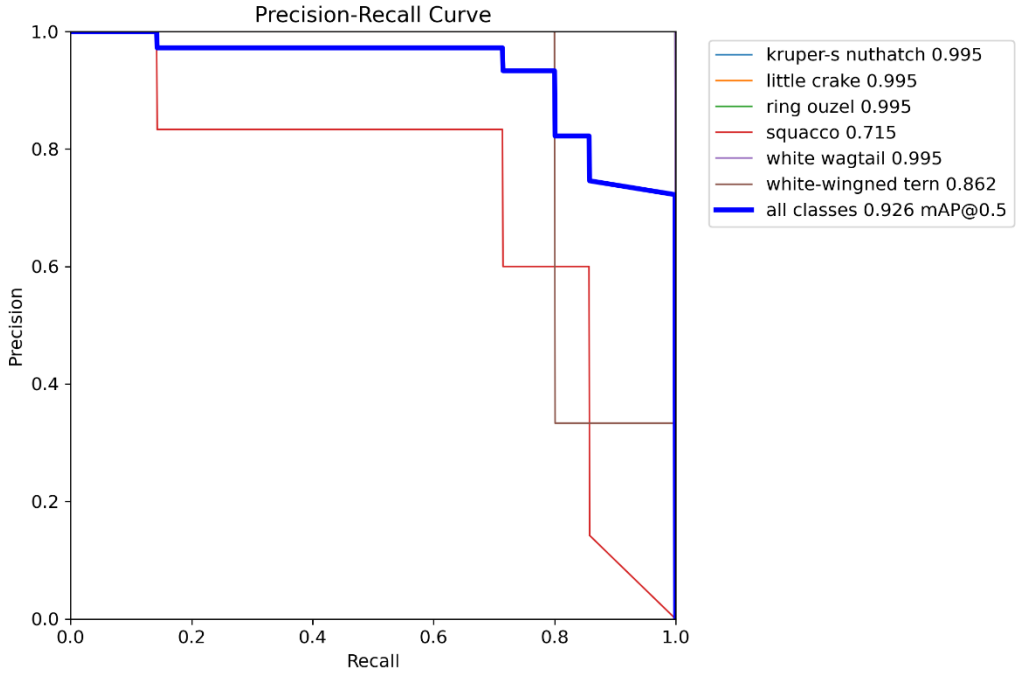

Figure 4.35. Precision-Recall Curve of Baseline YOLOv8 using in Test Phase

Figures 4.36 and 4.37 show the YOLOv8 F1 curves obtained with the proposed method and the original images in the test phase. When examining the results obtained with the proposed method, it can be seen that the model achieves an F1 value of 0,93 at a confidence level of 0,78. The fact that the F1 value remains constant as the confidence level increases indicates the stability of the model. Furthermore, the differences in F1 scores between classes are also smaller compared to the results obtained using the original YOLOv8. In contrast, the F1 score achieved with the original images is 0,88 and the confidence level is 0,60. When the confidence level increases, the F1 value drops for some classes. This shows that the model has class-based performance differences and generalizes poorly.

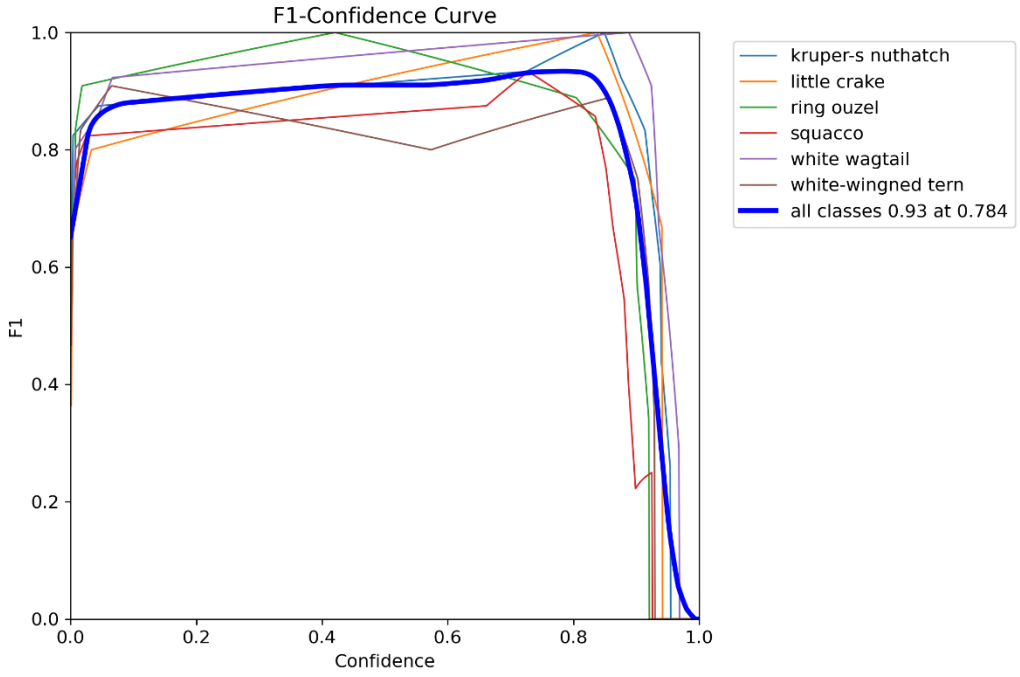

Figure 4.36. F1 Curve of YOLOv8 using Proposed Model in the Test Phase

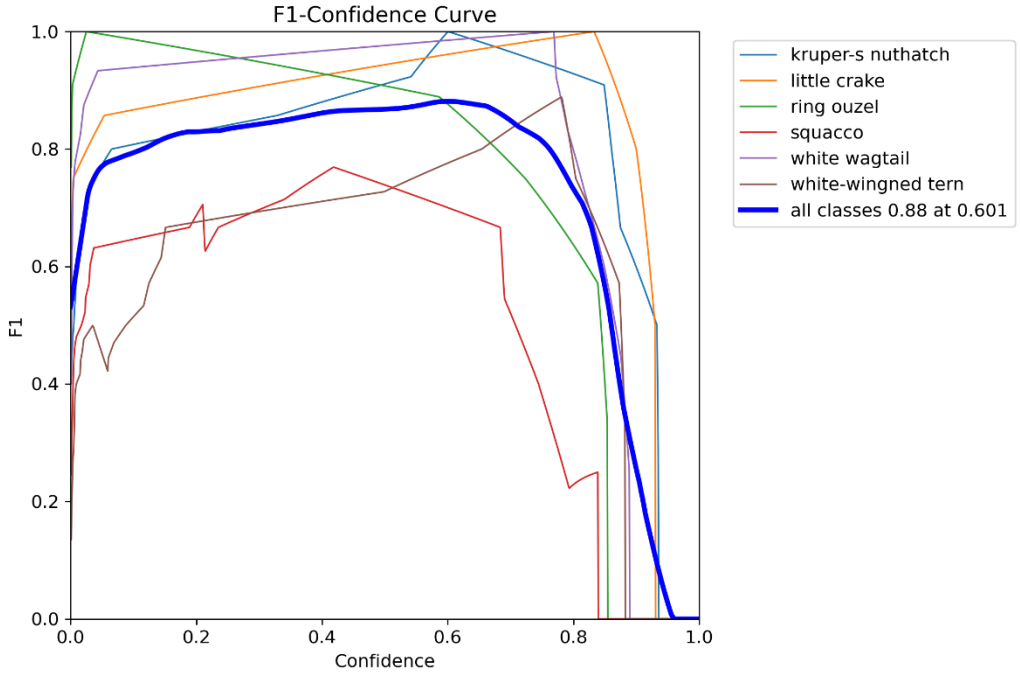

Figure 4.37. F1 Curve of Baseline YOLOv8 in the Test Phase

Figure 4.38 and 4.39 show some examples of predicted objects and boundary boxes belonging to test data using proposed model and original YOLOv8, respectively.

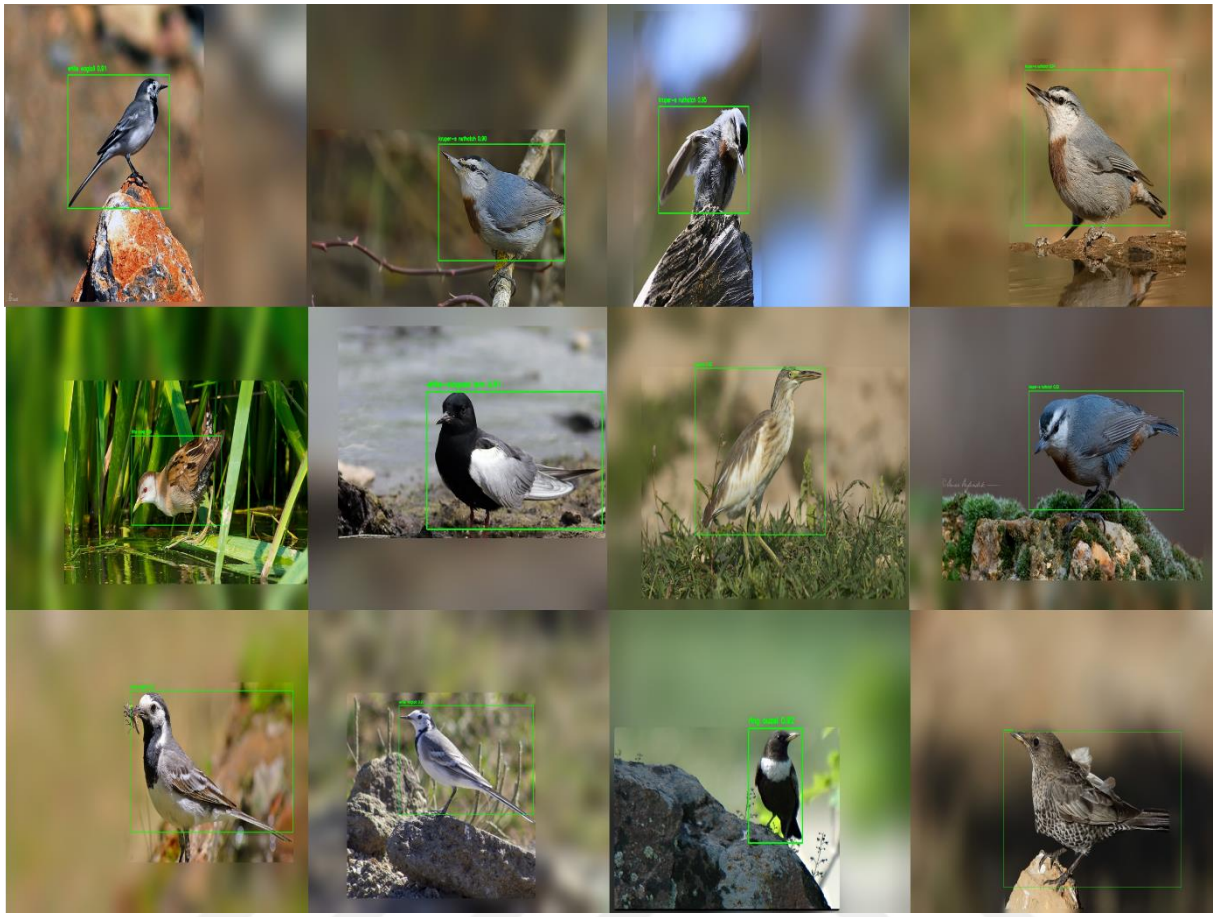

Figure 4.38. Some Examples of Detected Objects in the Test Phase of YOLOv8 using Proposed Model

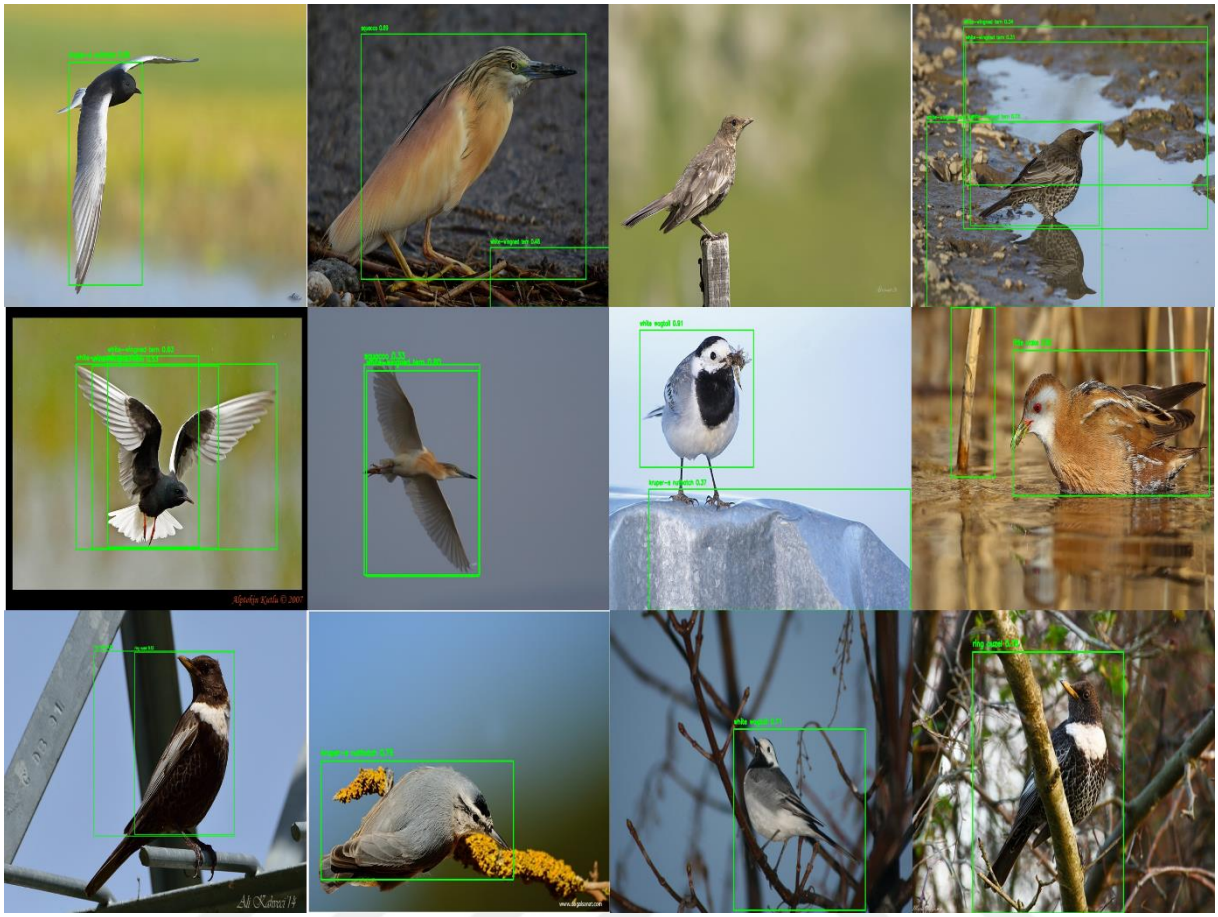

Figure 4.39. Some Examples of Detected Objects in the Test Phase of Baseline YOLOv8

4.3. Supplementary Experiments with R-CNN

In this section, additional experiments were conducted including comparisons in terms of region proposal generation time. The aim is to assess the effectiveness of the model under different conditions. Table 4.16 presents a comparison of the average region proposal generation time per image in seconds, using the proposed method and the original R-CNN, for both the training and testing phases. The proposed method demonstrated a significant reduction in average region proposal generation time in both training and testing. Specifically, the proposed method produced faster candidate regions for 3 out of 5 datasets in the training phase and for 4 out of 5 datasets in the testing phase. Figure 4.40 also shows the sum of the average training and testing times per image for each dataset.

Table 4.16. Runtime Analysis of Region Proposal Generation: Proposed Method vs. Baseline R-CNN

Dataset	Proposed		Original	
	Train	Test	Train	Test
White-backed Woodpecker	**36,763**	**49,405**	56,315	68,794
Rustic Bunting	276,010	**45,708**	**261,946**	48,328
Short-toed Treecreeper	193,142	**28,058**	**183,599**	32,740
Little Crake	**106,378**	**28,542**	116,005	31,594
White-eared Bulbul	**43,257**	27,246	50,870	31,533

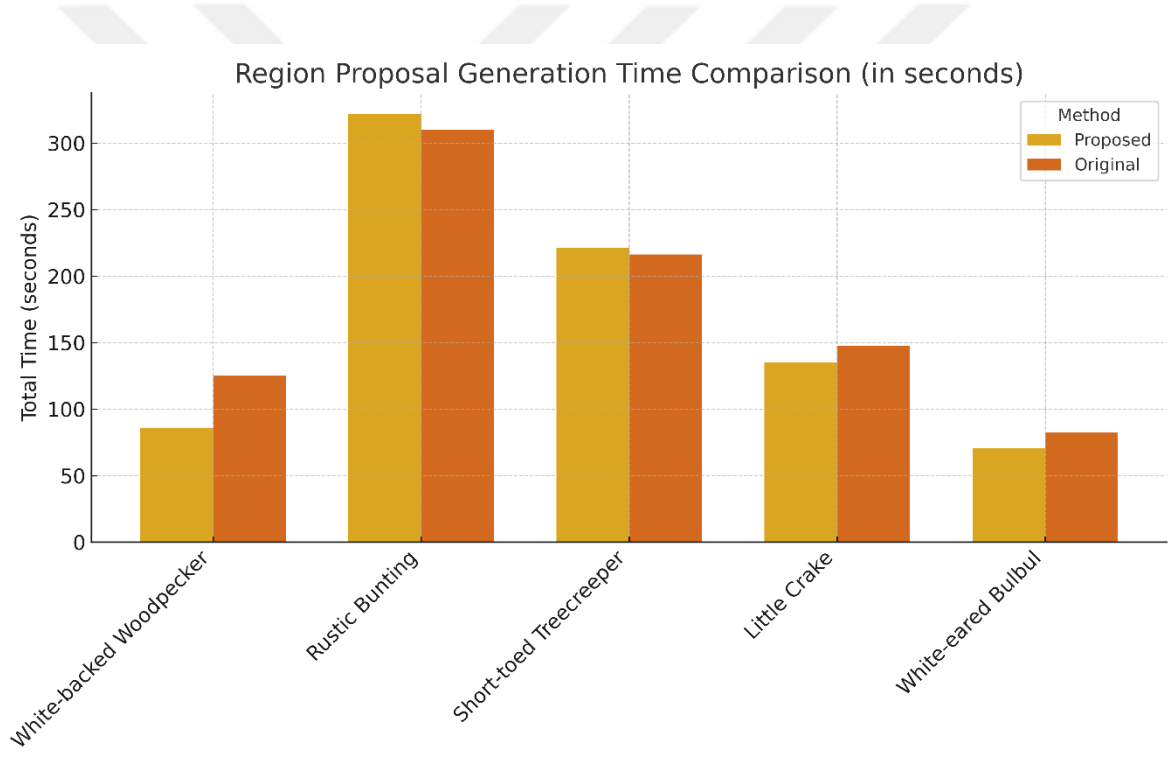

Figure 4.40. Average Region Proposal Generation Time (Train + Test) on the Five-Bird Dataset

4.4. Discussion

Previous research has shown that region proposal-based object detection methods, which utilize modified multi-stage pipelines, have been effective However, these studies have primarily focused on increasing the accuracy of the models, thus ignoring other important aspects, such as increasing the number of noise-free candidate regions and positive training samples. The authors (Roy and Sahay, 2021) presented a multi-scale deep learning method for accurate hand detection in driving video frames and real-world conditions. Their model showed promising results and increased detection accuracy on benchmark datasets. They demonstrated that a single deep Faster R-CNN can be outperformed by employing several shallow Faster R-CNNs. This is because the larger strides in later layers may cause fine-grained features to be lost in deeper networks.

Ren, Zhu, and Xiao (2018) used Faster R-CNN to detect small objects in optical remote sensing images. They chosen anchor sizes better suited for small-scale targets and enhanced RPN. They also used a high-resolution feature map by designing a structure with top-down and skip connections, helping preserve fine details during detection.

A paper (Kang, Leng, Lin, and Zi, 2017) presented en enhanced Faster R-CNN using the traditional constant false alarm rate (CFAR) method. By utilizing the object proposals generated by Faster R-CNN for the guard windows of the CFAR algorithm, this approach effectively identifies small-sized targets. Experimental results show that the model significantly improves detection performance. One effort (Xu, Li, Xie, Wu, and Wang, 2021) have introduced an innovative tunnel defect inspection method using the Mask R-CNN. In order to enhance the network's accuracy, they have incorporated a path augmentation feature pyramid network (PAFPN) and an edge detection branch.

While all these studies focus on solving detection performance in their respective fields our model aims to reduce the number of redundant regions while increasing the number of positive candidate boundary boxes and classification accuracy.

In the traditional RCNN model, the selective search generates bounding boxes for candidate objects, which are used to crop the features out of the backbone and make them the same size. The performance of the model therefore depends only on the boundary boxes generated by the selective search. The use of noise-free images reduces the number of negative candidate regions. The Gaussian blur module further improves the overlap between the ground truth and the predicted boxes. The baseline RCNN is implemented with pre-trained VGG16. Specifically, for the white-eared bulbul class, the model achieved 1215 bounding boxes, while the baseline model achieved only 147, 8.26 times less, in the test data.

The empirical analysis shows that the proposed two-stage pre-processing approach integrated into the R-CNN provides better results than those obtained with the original images. The proposed method reduced the noise in candidate regions, thus enabling the selective search algorithm to focus more intensively on positive regions. While the number of misclassified data decreases to almost 0 in both the training and testing phases, the IoU with the ground-truth boundary boxes increased to 65%. It also converges faster than the results obtained with the original images due to the noise-free input images.

In addition to evaluating the proposed method on bird datasets, we further tested its generalization capability on a non-bird object category using the Eastern Cottontail Rabbits dataset. This experiment was conducted using the R-CNN framework to remain consistent with the original training pipeline. The proposed model demonstrated faster convergence and improved bounding box quality compared to the original R-CNN baseline, indicating that its region proposal strategy remains effective even when applied to different object types and background settings.

This study makes a significant contribution by employing a two-stage preprocessing approach to decrease the search space of the boundary box generator and increase the number of positive candidate regions. As a result, the region proposal algorithm focuses solely on the noise-filtered candidate regions, reducing the number of proposals associated with selective search and aiming to eliminate redundant sub-images. On the other hand, the object detector algorithm is

performance relies on the effectiveness of the VJ model. The region proposal generator is highly responsive to filtering the search space.

To achieve the balance between detection accuracy and processing speed, the proposed method demonstrates a notable enhancement compared to baseline YOLOv8. Figures 4.24 and 4.25 illustrate that the proposed method surpasses the performance of YOLOv8 in classification, localization, distribution focal loss, precision, recall, mAP50, and mAP50-95. In both training and validation phases, it is clear that the proposed model demonstrates robustness and stability compared to baseline YOLOv8. Figures 4.38 and 4.39 indicate that the baseline YOLOv8 suffers from regression drops, generating insufficient or incorrect candidate boxes for each object, in certain objects, falling to detect some objects compared to the proposed method. This situation is an example of the localization problem of YOLOv8. The outcomes obtained by the proposed method on the test dataset are more consistent and successful.

On the other hand, when dealing with complex and crowded backgrounds, YOLOv8 may mistakenly detect the background as target objects. In contrast, our proposed method shows greater robustness than the original YOLOv8 in processing complex backgrounds, since it removes noise from areas outside the RoI.

These results emphasize the advantages of the proposed model over baseline YOLOv8 in challenging scenarios with complicated and dense backgrounds in object detection tasks and present its potential for valuable implementation in real-world contexts.

5. CONCLUSION

In this study, we have implemented the VJ algorithm to filter potential object proposals. Additionally, we proposed two adaptive padding mechanism called Adaptive Padding 1 and Adaptive Padding 2 to mitigate the possibility that the RoI detector failed to find the object regions accurately. Thus, we aimed to enhance the adaptability of the detected RoIs. We have also applied a low-pass filter to minimize noise in areas that do not contain an object and applied deep learning-based object detection tasks.

The study's primary objective is to replace numerous candidates with a reduced set of proposal boxes and improve the number of positive boundary boxes inputted into the Deep CNN for region proposal-based object detectors. The bird images used in this experiment were sourced from TRAKUS, and the ground truth boxes were manually annotated.

The proposed approach resulted in faster convergence of the R-CNN and improved the proposal generator algorithm's ability to identify more positive sub-images by efficiently filtering the search space. The findings indicate that adopting a two-stage approach significantly improves convergence speed, a higher quantity of correctly identified positive samples, and reduces the number of regions processed by the R-CNN. The proposed model achieves an impressive 55,48% reduction in region proposals and also increases the rate of positive boundary boxes by 330% during training and by 726% during testing.

This study makes a significant contribution by employing a two-stage preprocessing approach to decrease the search space of the boundary box generator and increase the number of positive candidate regions. As a result, the region proposal algorithm focuses solely on the noise-filtered candidate regions, reducing the number of proposals associated with selective search and aiming to eliminate redundant sub-images. On the other hand, the performance of the object detector algorithm relies on the effectiveness of the Haar cascade model. The region proposal generator is highly responsive to filtering the search space. In addition to improvements in object detection tasks, the proposed model includes potential applications in various real-world scenarios. UAV-based wildlife monitoring requires accurate and interpretable predictions. The

region proposal strategy of the proposed model makes it particularly suitable for ecological studies. However, because of the relatively higher computational cost of two-stage detectors, real-time applications on resource-constrained platforms, such as drones, may require further optimization.

The proposed model was also evaluated using a single-stage object detector named YOLOv8. The model separately trained a RoI detector that combines a set of weak candidate regions into a single frame and makes it more flexible utilizing an Adaptive Padding Mechanism. The proposed method aims to suppress high-frequency components by applying low pass filter-based denoising to areas outside the RoI. This way, the detector can focus on meaningful features instead of dealing with background details, making the training examples more uniform and reducing overfitting. To improve localization accuracy of YOLO by reducing grid misalignment issues the AP mechanism is utilized.

The proposed method resulted in a 3% increase in mAp50 value of YOLOv8, achieving 98.4%. Moreover, the value of mAP50-95 increased more than 23% exceeding 78%. Since the original YOLO has a speed-oriented architecture, it suffers from drops in localization accuracy, 0,677 in training and 1,247 in validation. The localization results obtained with the proposed method impressively outperformed the original YOLOv8: 0,512 in training and 0,868 in validation. Upon comparing the classification results, it is notable that the results achieved by the proposed method are more consistent: 0,496 and 0,498 in training and validation, respectively. Compared to the baseline YOLOv8, there is a 44,48% decrease in classification loss.

The objective was to employ a novel pre-processing approach as a reliable solution to enhance the performance of single-stage object detectors particularly for YOLOv8. Our observations indicate significant improvements in regression loss as well as consistency in classification results compared to the baseline YOLOv8. We aimed to prevent the edges of objects from falling outside the RoI by utilizing the Adaptive Padding mechanism.

REFERENCES

Alexe, B., Deselaers, T., & Ferrari, V. (2012). Measuring the Objectness of Image Windows. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11), 2189–2202. https://doi.org/10.1109/TPAMI.2012.28.

Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., & Malik, J. (2014, June 1). Multiscale Combinatorial Grouping. IEEE Xplore. https://doi.org/10.1109/CVPR.2014.49.

Bao, Z. (2024). The UAV Target Detection Algorithm Based on Improved YOLO V8. 264–269. https://doi.org/10.1145/3700906.3700949.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(4), 509–522. https://doi.org/10.1109/34.993558.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020, April 22). YOLOv4: Optimal Speed and Accuracy of Object Detection. ArXiv.org. https://doi.org/10.48550/arXiv.2004.10934.

Boudjit, K., & Ramzan, N. (2021). Human detection based on deep learning YOLO-v2 for real-time UAV applications. Journal of Experimental & Theoretical Artificial Intelligence, 1–18. https://doi.org/10.1080/0952813x.2021.1907793.

Carreira, J., & Sminchisescu, C. (2010). Constrained parametric min-cuts for automatic object segmentation. https://doi.org/10.1109/cvpr.2010.5540063.

Chen, C., Liu, M.-Y., Tuzel, O., & Xiao, J. (2017). R-CNN for Small Object Detection. Computer Vision – ACCV 2016, 214–230. https://doi.org/10.1007/978-3-319-54193-8_14.

Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., & Liu, Y. (2021). Deep Learning for Sensor-based Human Activity Recognition. ACM Computing Surveys, 54(4), 1–40. https://doi.org/10.1145/3447744.

Ciaparrone, G., Sánchez, L., Tabik, S., Troiano, L., Tagliaferri, R., & Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. Neurocomputing, 381, 61–88. https://doi.org/10.1016/j.neucom.2019.11.023.

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1, 886–893. https://doi.org/10.1109/cvpr.2005.177.

Dinç, B. & Kaya, Y. (2023). A Novel Hybrid Optic Disc Detection and Fovea Localization Method Integrating Region-Based Convnet and Mathematical Approach. 129(4), 2727–2748.

Eastern. (2021). *Eastern Cottontail Rabbits Object Detection Dataset*. Roboflow. https://public.roboflow.com/object-detection/eastern-cottontail-rabbits

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008, June 1). A discriminatively trained, multiscale, deformable part model. IEEE Xplore. https://doi.org/10.1109/CVPR.2008.4587597.

Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W., & Dietmayer, K. (2020). Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. IEEE Transactions on Intelligent Transportation Systems, 1–20. https://doi.org/10.1109/tits.2020.2972974.

Ferrari, A., Micucci, D., Mobilio, M., & Napoletano, P. (2022). Deep learning and model personalization in sensor-based human activity recognition. Journal of Reliable Intelligent Environments. https://doi.org/10.1007/s40860-021-00167-w.

Gai, R., Chen, N., & Yuan, H. (2021). A detection algorithm for cherry fruits based on the improved YOLO-v4 model. Neural Computing and Applications. https://doi.org/10.1007/s00521-021-06029-z.

Girshick, R. (2015). Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV), 1440–1448. https://doi.org/10.1109/iccv.2015.169.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 580–587. https://doi.org/10.1109/cvpr.2014.81.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9), 1904–1916. https://doi.org/10.1109/tpami.2015.2389824.

Helmi, A. M., Al-qaness, M. A. A., Dahou, A., & Abd Elaziz, M. (2023). Human activity recognition using marine predators algorithm with deep learning. Future Generation Computer Systems, 142, 340–350. https://doi.org/10.1016/j.future.2023.01.006.

Hoeser, T., & Kuenzer, C. (2020). Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends. Remote Sensing, 12(10), 1667. https://doi.org/10.3390/rs12101667.

Irgens, P., Bader, C., Lé, T., Saxena, D., & Ababei, C. (2017). An efficient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm. HardwareX, 1, 68–75. https://doi.org/10.1016/j.ohx.2017.03.002.

Ji, S.-J., Ling, Q.-H., & Han, F. (2023). An improved algorithm for small object detection based on YOLO v4 and multi-scale contextual information. Computers and Electrical Engineering, 105, 108490. https://doi.org/10.1016/j.compeleceng.2022.108490.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe. Proceedings of the ACM International Conference on Multimedia - MM '14. https://doi.org/10.1145/2647868.2654889.

Jiang, N., Sheng, B., Li, P., & Lee, T.-Y. (2022). PhotoHelper: Portrait Photographing Guidance Via Deep Feature Retrieval and Fusion. IEEE Transactions on Multimedia, 25, 2226–2238. https://doi.org/10.1109/tmm.2022.3144890.

Jocher, G. (2020, May 18). YOLOv8 Documentation. Docs.ultralytics.com. https://docs.ultralytics.com/.

Kang, M., Leng, X., Lin, Z., & Ji, K. (2017). A modified faster R-CNN based on CFAR algorithm for SAR ship detection. 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP). https://doi.org/10.1109/rsip.2017.7958815.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, 60(6), 84–90.

Lamichhane, B. R., Srijuntongsiri, G., & Horanont, T. (2025). CNN based 2D object detection techniques:a review. Frontiers in Computer Science, 7. https://doi.org/10.3389/fcomp.2025.1437664

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. https://doi.org/10.48550/arxiv.2209.02976.

Li, F., Zhang, H., Xu, H., Liu, S., Zhang, L., Ni, L. M., & Shum, H.-Y. (2023). Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation. https://doi.org/10.1109/cvpr52729.2023.00297.

Li, R., Yu, J., Li, F., Yang, R., Wang, Y., & Peng, Z. (2023). Automatic bridge crack detection using Unmanned aerial vehicle and Faster R-CNN. Construction and Building Materials, 362, 129659. https://doi.org/10.1016/j.conbuildmat.2022.129659.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2017.106.

Lin, X., Sun, S., Huang, W., Sheng, B., Li, P., & Feng, D. D. (2021). EAPT: Efficient Attention Pyramid Transformer for Image Processing. IEEE Transactions on Multimedia, 1–1. https://doi.org/10.1109/tmm.2021.3120873.

Liu, F., Chen, D., Wang, F., Li, Z., & Xu, F. (2022). Deep learning based single sample face recognition: a survey. Artificial Intelligence Review. https://doi.org/10.1007/s10462-022-10240-2.

Liu, Q., Lu, Z., Gao, R., Bu, X., & Naohiko Hanajima. (2024). SimpleMask: parameter link and efficient instance segmentation. The Visual Computer. https://doi.org/10.1007/s00371-024-03451-x.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. IEEE Xplore. https://doi.org/10.1109/ICCV.1999.790410.

Luo, G., Wang, Z., Wang, Y., & Wang, K. (2024). Detection of pneumonia in chest X-ray image based on separable convolutional neural network. *Concurrency and Computation Practice and Experience*, *36*(18). https://doi.org/10.1002/cpe.8160

Masud, M., Muhammad, G., Alhumyani, H., Alshamrani, S. S., Cheikhrouhou, O., Ibrahim, S., & Hossain, M. S. (2020). Deep learning-based intelligent face recognition in IoT-cloud environment. Computer Communications, 152, 215–222. https://doi.org/10.1016/j.comcom.2020.01.050.

Mittal, U., Chawla, P., & Tiwari, R. (2022). EnsembleNet: a hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models. Neural Computing and Applications, 35(6), 4755–4774. https://doi.org/10.1007/s00521-022-07940-9.

Papageorgiou, C. P., Oren, M., & Poggio, T. (1998). A general framework for object detection. IEEE Xplore. https://doi.org/10.1109/ICCV.1998.710772.

Ranjan Sapkota, Ahmed, D., & Manoj Karkee. (2024). Comparing YOLOv8 and Mask R-CNN for instance segmentation in complex orchard environments. Artificial Intelligence in Agriculture. https://doi.org/10.1016/j.aiia.2024.07.001

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788. https://doi.org/10.1109/cvpr.2016.91.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2017.690

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Arxiv.org. https://doi.org/10.48550/arXiv.1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031.

Ren, Y., Zhu, C., & Xiao, S. (2018). Small Object Detection in Optical Remote Sensing Images via Modified Faster R-CNN. Applied Sciences, 8(5), 813. https://doi.org/10.3390/app8050813.

Roy, K., & Sahay, R. R. (2021). A robust multi-scale deep learning approach for unconstrained hand detection aided by skin segmentation. The Visual Computer. https://doi.org/10.1007/s00371-021-02157-8.

Sahin, M. E., Ulutas, H., Yuce, E., & Erkoc, M. F. (2023). Detection and classification of COVID-19 by using faster R-CNN and mask R-CNN on CT images. Neural Computing and Applications. https://doi.org/10.1007/s00521-023-08450-y.

Sahu, D., Pradhan, B. K., Wilczyński, S., Anis, A., & Pal, K. (2023). Development of an internet of things (IoT)-based pill monitoring device for geriatric patients. Elsevier EBooks, 129–158. https://doi.org/10.1016/b978-0-323-85955-4.00012-0.

Serengil, S. I., & Ozpinar, A. (2020, October 1). LightFace: A Hybrid Deep Face Recognition Framework. IEEE Xplore. https://doi.org/10.1109/ASYU50717.2020.9259802.

Sheng, B., Li, P., Ali, R., & Chen, P. (2022). Improving Video Temporal Consistency via Broad Learning System. IEEE Transactions on Cybernetics, 52(7), 6662–6675. https://doi.org/10.1109/tcyb.2021.3079311.

Solawetz, J. (2020, June 29). YOLOv5 New Version - Improvements And Evaluation. Roboflow Blog. https://blog.roboflow.com/yolov5-improvements-and-evaluation/.

Song, P., Li, P., Dai, L., Wang, T., & Chen, Z. (2023). Boosting R-CNN: Reweighting R-CNN samples by RPN's error for underwater object detection. Neurocomputing, 530, 150–164. https://doi.org/10.1016/j.neucom.2023.01.088.

Sridhar, P., Jagadeeswari, M., Sri, S., Akshaya, N., & Haritha, J. (2022). Helmet Violation Detection using YOLO v2 Deep Learning Framework. https://doi.org/10.1109/icoei53556.2022.9776661.

Sunardi, N., Yudhana, A., & Putri, A. (2023). Optimization of Breast Cancer Classification Using Faster R-CNN. Revue D Intelligence Artificielle, 37(1), 39–45. https://doi.org/10.18280/ria.370106.

Tang, S., Zhang, S., & Fang, Y. (2024). HIC-YOLOv5: Improved YOLOv5 For Small Object Detection. ArXiv (Cornell University), 6614–6619. https://doi.org/10.1109/icra57147.2024.10610273.

Tian, X., Bi, C., Han, J., & Yu, C. (2024). EasyRP-R-CNN: a fast cyclone detection model. The Visual Computer, 40(7), 4829–4841. https://doi.org/10.1007/s00371-024-03483-3.

TRAKUS. (2025, February 13). TRAKUS kuşlar: Kuş türleri detaylı bilgiler. TRAKUS. https://www.trakus.org/kods_bird/uye/?fsx=2fsdl11@d

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). Selective Search for Object Recognition. International Journal of Computer Vision, 104(2), 154–171.

Urdiales, J., Martín, D., & Armingol, J. M. (2023). An improved deep learning architecture for multi-object tracking systems. Integrated Computer-Aided Engineering, 1–14. https://doi.org/10.3233/ica-230702.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1. https://doi.org/10.1109/cvpr.2001.990517.

Vrskova, R., Kamencay, P., Hudec, R., & Sykora, P. (2023). A New Deep-Learning Method for Human Activity Recognition. Sensors, 23(5), 2816. https://doi.org/10.3390/s23052816.

Wang, C.-Y., Bochkovskiy, A., & Liao, H. (2023). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. https://doi.org/10.1109/cvpr52729.2023.00721.

Wang, J., Chen, Y., Hao, S., Peng, X., & Hu, L. (2019). Deep learning for sensor-based activity recognition: A survey. Pattern Recognition Letters, 119, 3–11. https://doi.org/10.1016/j.patrec.2018.02.010.

Wang, Y., Ahsan, U., Li, H., & Hagen, M. (2022). A comprehensive review of modern object segmentation approaches. Foundations and Trends in Computer Graphics and Vision, 13(2–3), 111–283. https://doi.org/10.1561/0600000097

Wu, P., Gu, L., Yan, X., Xie, H., Wang, F. L., Cheng, G., & Wei, M. (2022). PV-RCNN++: Semantical Point-Voxel Feature Interaction for 3D Object Detection. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.2208.13414.

Xia, Z., Huang, Z., Gu, J., & Wang, W. (2025). Surface Defect Detector Based on Deformable Convolution and Lightweight Multi-Scale Attention. *Concurrency and Computation Practice and Experience*, *37*(4-5). https://doi.org/10.1002/cpe.70003

Xin, F., Zhang, H., & Pan, H. (2023). Hybrid dilated multilayer faster RCNN for object detection. The Visual Computer, 40(1), 393–406. https://doi.org/10.1007/s00371-023-02789-y.

Xu, X., Zhao, M., Shi, P., Ren, R., He, X., Wei, X., & Yang, H. (2022). Crack Detection and Comparison Study Based on Faster R-CNN and Mask R-CNN. Sensors, 22(3), 1215. https://doi.org/10.3390/s22031215.

Xu, Y., Li, D., Xie, Q., Wu, Q., & Wang, J. (2021). Automatic defect detection and segmentation of tunnel surface using modified Mask R-CNN. Measurement, 178, 109316. https://doi.org/10.1016/j.measurement.2021.109316.

Yunus, U., Amin, J., Sharif, M., Yasmin, M., Kadry, S., & Krishnamoorthy, S. (2022). Recognition of Knee Osteoarthritis (KOA) Using YOLOv2 and Classification Based on Convolutional Neural Network. Life, 12(8), 1126. https://doi.org/10.3390/life12081126.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212–3232. https://doi.org/10.1109/tnnls.2018.2876865.

Zhou, Q., & Yu, C. (2022). Point RCNN: An Angle-Free Framework for Rotated Object Detection. Remote Sensing, 14(11), 2605. https://doi.org/10.3390/rs14112605.

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object Detection in 20 Years: A Survey. Proceedings of the IEEE, 111(3), 1–20. https://doi.org/10.1109/JPROC.2023.3238524.