FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

CLASS INCREMENTAL OBJECT DETECTION

By

SIQUAN ZHU

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2025

Siquan Zhu defended this dissertation on March 24, 2025.

The members of the supervisory committee were:

Adrian Barbu
Professor Directing Dissertation

Gordon Erlebacher
University Representative

Jinfeng Zhang
Committee Member

Joshua Loyal
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

TABLE OF CONTENTS

List of Tables . . . vi

List of Figures . . . vii

Abstract . . . ix

1 Introduction **1**

1.1 The Incremental Object Detection Task . . . 1

1.2 Challenges in Incremental Object Detection . . . 1

1.3 List of Contribution . . . 2

1.4 Related Works . . . 2

1.4.1 Two-stage detectors . . . 3

1.4.2 One-stage detectors . . . 4

1.4.3 Incremental learning . . . 5

1.4.4 Object proposal methods . . . 6

1.4.5 Class-agnostic object detection . . . 8

2 Incremental Training of an Objectness-based Region Proposal Generator **10**

2.1 Setup and Notation . . . 10

2.2 Method Overview . . . 10

2.3 Preliminary Knowledge . . . 11

2.3.1 Integral image . . . 11

2.3.2 Running averages . . . 12

2.3.3 Nelder-Mead optimization . . . 13

2.3.4 Proposals with Mask R-CNN . . . 13

2.3.5 Class-agnostic proposals with Grounding-DINO . . . 13

2.4 Incremental Learning with Integral Image Features and RAVE . . . 14

2.5 Training a Binary Classifier Using RAVE . . . 15

2.6 Generating Anchor Proposals . . . 17

2.7 Multi-scale Classification . . . 18

2.7.1 Proportional allocation of candidates . . . 19

2.7.2 Best threshold combination for two classifiers . . . 19

2.7.3 Incremental Nelder-Mead optimizer . . . 20

3 Anchor Proposal Generator Experiments **22**

3.1 Datasets 22

3.2 Metrics 22

3.3 Training Details 23

3.3.1 Preprocessing 23

3.3.2 Faster-RCNN model 23

3.3.3 Foreground-background classifier 23

3.3.4 Multi-scale classification 24

3.4 Experiments 25

3.4.1 Ablation study 25

3.4.2 Comparison with existing anchor generation methods 26

4 Class-incremental Object Detection with Probabilistic PCA Classifiers **28**

4.1 Method Overview 28

4.2 Preliminary Knowledge 28

4.2.1 ROI-Align 28

4.2.2 Probabilistic PCA 30

4.2.3 CLIP 32

4.3 Incremental Multi-class Object Detection with PPCA 33

4.3.1 Collecting RAVEs for the multi-class classification model 33

4.3.2 Training PPCA 34

4.4 Classifying Anchors with PPCA 34

5 Class Incremental Object Detection Experiments **36**

5.1 Datasets 36

5.2 Metrics 36

5.2.1 Computation of mAP 37

5.3 Training Details 37

5.3.1 Mask R-CNN proposals 37

5.3.2 Grounding-DINO proposals 38

5.3.3 Building PPCA models 38

5.3.4 Predicting boxes 38

5.4 Experiments 39

5.4.1 Catastrophic forgetting 39

5.4.2 Ablation study . . . 41
5.4.3 Comparison . . . 43

6 Conclusion **44**

Bibliography . . . 46

Biographical Sketch . . . 49

LIST OF TABLES

2.1 Notations used in this thesis. 10

3.1 Test detection rate and number of anchors per image for different classifier combinations. 26

3.2 Comparison of different anchor generation method 27

5.1 Mean Average Precision (mAP@0.5) at different stages and groups of ILSVRC2015 for the incremental PPCA model. 40

5.2 Mean Average Precision (mAP(0.5)) at different stages and groups of ILSVRC2015 for the Faster-RCNN model. 40

5.3 Ablation study results for different numbers of sampling points. 41

5.4 Ablation study results for different numbers of principal components (npc). 42

5.5 Comparison of different box proposal methods. 42

5.6 Performance comparison across different methods with COCO eval2017.(mAP(0.5:0.95)/mAP(0.5)) 43

LIST OF FIGURES

2.1 Diagram of the proposed anchor generation framework. 11

2.2 Training process of the binary classifier. 17

4.1 PPCA-based Object detector framework. 28

4.2 Diagram of PPCA classifier 32

5.1 First row shows the G_DINO box proposals. Second row shows the detection result of our method after incremental training from all 180 classes. 39

LIST OF ALGORITHMS

1 IntegralImage 12
2 Incremental multi-scale RAVE update 15
3 Anchor generator 18
4 Threshold Selection for two classifiers 20
5 ROI_feature 30
6 Incremental multi-class RAVE update 34
7 PPCA 34
8 Classify anchors of one image with PPCA 35

ABSTRACT

Class-incremental object detection presents significant challenges, particularly in addressing the computational complexity and memory demands associated with continuously training models as new object categories are introduced. In this study, we propose a novel framework that mitigates these challenges by eliminating the need for complex neural networks and extensive hyperparameter tuning. Our approach utilizes the integral image technique to efficiently collect anchor features from feature maps and employs the RAVE method to store pertinent information, facilitating real-time object detection. We train a multi-scale online linear regression classifier with L2 regularization to categorize anchor proposals into foreground and background. Additionally, we introduce three strategies for threshold selection to optimize the balance between detection rate and anchor quantity. This method significantly reduces computational overhead and enables incremental learning without the need to retrain the model from scratch. Experimental evaluations show that our framework yields promising results for the anchor proposal generator compared to widely used existing methods.

Furthermore, in the second phase of our experiments, we conducted an extensive comparative analysis of different anchor proposal methods to assess their effectiveness in class-incremental object detection. By leveraging the strong generalization capabilities of large-scale pre-trained models, we enhanced our framework's ability to adapt to new object categories while mitigating catastrophic forgetting. Specifically, we incorporated the CLIP feature extractor to improve multi-class classification and integrated it with RAVE-based storage to train a Probabilistic Principal Component Analysis (PPCA) model for classifying anchor proposals.

To further refine feature quality, we applied the concept of ROI Align to eliminate misalignment in feature extraction, ensuring more precise localization of object features. Additionally, we incrementally computed mean Average Precision (mAP) across different learning stages, allowing for a more detailed evaluation of detection performance over time.

The results demonstrate that our proposed framework not only achieves competitive detection rates but also significantly reduces computational and memory overhead, making it well-suited for real-world applications where new object categories continuously emerge. By eliminating the need for complex neural network retraining, our approach offers an efficient and scalable solution for class-incremental object detection, paving the way for future advancements in real-time detection systems.

CHAPTER 1

INTRODUCTION

1.1 The Incremental Object Detection Task

In traditional object detection tasks, models typically rely on fixed datasets and corresponding annotations. However, we often encounter new data, classes, and scenarios, necessitating a more sophisticated approach. Hence, the Incremental Object Detection (IOD) task was proposed to address some of these complexities. IOD contains methods to train object detectors gradually by adding new object categories without retraining on the whole dataset. This approach enables the model to adapt to changes in the environment or task requirements dynamically, to gradually recognize novel objects, or to handle evolving object distributions without retraining from scratch. In practice, the process involves dividing the classes to be learned into several batches, each containing distinct classes. The model is initially trained on a subset of classes, and subsequently, when new data is added, the model is not allowed to utilize data from previous classes. This constraint makes IOD more challenging, requiring models to continually evolve and adapt to novel data increments without access to previous training data.

1.2 Challenges in Incremental Object Detection

As we explore the challenges associated with incremental learning in object detection, several key issues emerge that require careful consideration.

- **Catastrophic Forgetting**: Catastrophic forgetting occurs when the model forgets information about previously learned classes while adapting to new ones. Catastrophic forgetting occurs when a well-trained detector on class A is fine-tuned for a different class B, resulting in the loss of knowledge about class A and suboptimal detection results for that class.

- **Class Imbalance**: As new classes are introduced incrementally, the dataset may become imbalanced, leading to challenges in training the model to effectively recognize and differentiate between new and existing classes.

- **Data Efficiency**: Incremental learning requires efficient utilization of limited labeled data for new classes. Developing methods that make the most out of the available data is a constant challenge.

- **Online Learning**: Enabling models to learn incrementally in an online setting, where new data becomes available continuously, requires specialized algorithms and frameworks.
- **Computational Complexity**: Managing computational resources becomes challenging as the model accumulates knowledge over multiple increments. Strategies to ensure efficient training and inference are essential.

1.3 List of Contribution

This work brings the following key contributions:

- **Novel Framework for Class-Incremental Object Detection.** Introduced a new framework that efficiently handles class-incremental object detection without the need for complex neural networks and extensive hyperparameter tuning.
- **Integration of the Integral Image Technique.** Utilized the integral image method to effectively gather anchor features from feature maps, enhancing the efficiency of the detection process.
- **Use of RAVE for Information Storage.** Employed the RAVE technique to store relevant model information, facilitating real-time object detection and efficient memory management.
- **Multi-Scale Online Linear Regression Classifier.** Developed a multi-scale online linear regression classifier with L2 regularization for initial foreground-background classification of anchor proposals.
- **Threshold Selection Strategies.** Introduced three innovative strategies for selecting thresholds for each classifier, optimizing the trade-off between the detection rate and the number of anchor proposals.
- **Overcome catastrophic forgetting using PPCA models.** We employed a PPCA-based approach combined with the RAVE mechanism to enable online updates of PPCA parameters. This method helps retain previously learned information while adapting to new data, effectively reducing catastrophic forgetting.
- **Comparison with Other Methods.** We applied our online PPCA model to various class-agnostic box proposal methods and compared the results with the state-of-the-art class-incremental object detection architectures.

1.4 Related Works

In this section, we provide an overview of significant contributions relevant to our study, focusing on four main areas: two-stage detectors, one-stage detectors, incremental learning, and object proposal methods.

1.4.1 Two-stage detectors

One-stage detectors and two-stage detectors are two primary categories of object detection algorithms. One-stage detectors, such as YOLO (Redmon et al., 2016) and SSD (Liu et al., 2016), operate by predicting bounding boxes and class labels in a single pass through the network, making them faster and more suitable for real-time applications. In contrast, two-stage detectors, like Faster R-CNN (Ren et al., 2015), first generate region proposals and then classify these proposals in a separate step, typically resulting in higher accuracy but at the cost of increased computational complexity. The main difference lies in their architecture: one-stage detectors streamline the detection process into one step, while two-stage detectors involve a two-step procedure that enhances precision but requires more processing time.

In Faster-RCNN (Ren et al., 2015), the authors introduced a modification to the Fast R-CNN (Girshick, 2015) model by incorporating a new region proposal module. They found that the feature maps used by region-based detectors can also be used for generating region proposals. By just adding convolutional layers that simultaneously regress the anchors (a set of fixed initial bounding box guesses) and a measure of objectness for each location on the feature map, this implementation, compatible with GPU processing, significantly accelerated the proposal generation process. More details on anchors will be given in Section 2.1. Comparing the efficiency of generating proposals with other popular methods, their Region Proposal Network (RPN) exhibited remarkable speed, requiring only 10ms per image. In contrast, Selective Search (Uijlings et al., 2013) demanded 2 seconds per image, and EdgeBoxes (Zitnick and Dollár, 2014) took 0.2 seconds per image. Additionally, the authors also proposed a new way to address the multi-scale and object size problem. Instead of employing traditional approaches such as pyramids of images and feature maps or pyramids of filters with multiple scales, they adopted a more efficient strategy. They utilized pyramids of reference boxes directly within the regression functions, which led to significant time savings during the training process.

In the Cascade-RCNN paper (Cai and Vasconcelos, 2018), the authors highlight a potential vulnerability in the traditional R-CNN architecture. They identify a discrepancy between the distribution of proposals in the training and inference phases. This arises from intuitively supplying all boxes with an intersection over union (IoU) greater than 0.5 with the ground truth to the regressor as positives during training. However, in the inference phase, where the ground truth is unknown, all boxes are indiscriminately input to the regressor. The authors emphasize that increasing the

IoU threshold during proposal screening does not effectively address this issue, as it may lead to overfitting problems and exacerbate the mismatch. To tackle this challenge, they propose a multi-stage structure. Each stage takes the region proposals from the previous stage through an ROI pooling layer (Ren et al., 2015) based on a higher IoU threshold. After that, a prediction head is employed to predict the class and bounding box. Subsequently, this newly selected bounding box and the feature map are used to train another prediction head. This iterative process is repeated for three different prediction heads, significantly enhancing the quality of proposals.

1.4.2 One-stage detectors

You Only Look Once (YOLO) (Redmon et al., 2016) revolutionizes object detection by framing the task as a regression problem, directly mapping image pixels to bounding box coordinates and class probabilities simultaneously. This innovative approach unifies disparate components of traditional object detection systems, such as Deformable Part Models (DPM) (Girshick et al., 2015), into a single neural network. The model partitions the input image into an $S \times S$ grid, designating each grid cell responsible for detecting an object if its center falls within that cell. The bounding box predictions encompass five key elements: $(x, y, w, h, score)$. The (x, y) coordinates represent the box center relative to the grid cell bounds, while (w, h) are predicted relative to the entire image. The score reflects the Intersection over Union (IOU) between the predicted and ground truth boxes. Additionally, each grid cell predicts C conditional class probabilities, $P(Class_i|Object)$. These probabilities are conditioned on the grid cell containing the object. During testing, the multiplication of conditional class probabilities and individual box confidence predictions yields class-specific confidence scores for each box, offering a comprehensive and efficient real-time object detection framework.

In Single Shot MultiBox Detector (SSD)(Liu et al., 2016), object detection is accomplished through a single deep neural network. This network discretizes the output space of bounding boxes, creating a collection of default boxes characterized by diverse aspect ratios and scales at each feature map location. Subsequently, scores are generated for each default box, and adjustments are produced accordingly. It leverages any image classification network (truncated before any classification layers) followed by an auxiliary structure, represented by a set of convolutional neural networks (CNN). In contrast, YOLO employs fully connected layers. The concept of default boxes in SSD resembles the anchor boxes utilized in Faster-RCNN. Notably, these default boxes are applied across various feature maps with distinct resolutions. This approach enables the efficient discretization of the space associated with potential output box shapes, as it accommodates diverse default box shapes across multiple feature maps.

In (Lin et al., 2017), the authors introduce a novel loss term named Focal Loss, which adds a factor $(1 - p_t)^r$ to the standard cross-entropy criterion. They categorize object detection methods into two groups: one-stage detectors, known for their speed and simplicity, and two-stage detectors, recognized for their higher accuracy. The authors claim that the difference is caused by the foreground-background imbalance in training data. The Focal Loss addresses this issue by concentrating training on a sparse set of challenging examples, preventing the dominance of numerous easy negatives during the training process. The authors demonstrate the effectiveness of their approach by constructing a simple network, Retina-Net, which matches the speed of previous one-stage detectors while surpassing the accuracy of all state-of-the-art two-stage detectors.

In the paper "End-to-End Object Detection with Transformers," (Carion et al., 2020) the authors redefine the object detection task as a set prediction problem, eliminating the need for generating numerous anchors or proposals and the need to use Non-Maximum Suppression to prune these bounding boxes. Instead, the approach directly obtains the final predictions for each input image. The main contribution of this paper is a new transformer architecture and a new set-based global loss that forces unique predictions. The architecture comprises four key components: a convolutional neural network for feature generation, like most object detection models; a transformer encoder for learning global information; a transformer decoder for generating proposed bounding boxes; and a bipartite matching loss, which facilitates the optimization of the entire model. Notably, the proposed loss function draws inspiration from the Hungarian algorithm. The authors treat the box prediction problem as a bipartite graph matching problem, utilizing a modified classification loss and regression loss as costs to train the whole model.

1.4.3 Incremental learning

In the paper "Incremental Learning of Object Detectors without Catastrophic Forgetting," (Shmelkov et al., 2017) the authors tackle the challenge of catastrophic forgetting in incremental learning with Convolutional Neural Networks (CNNs). To mitigate this issue, the authors introduce a novel loss function for training a model incrementally with new classes. Their proposed approach involves a two-network architecture when learning a new class. The first network incorporates a frozen copy of the original detector, used for detecting the original classes and computing the distillation loss. The second network is dedicated to detecting the added class while simultaneously predicting both the new classes and the original classes when combined with the first network. This distillation loss is computed by the following function:

$$L_{dist}(y_A, t_A, y_B, t_B) = \frac{1}{N|C_A|} \sum [(\overline{y}_A - \overline{y}_B)^2 + (t_A - t_B)^2], \tag{1.1}$$

where the summation is taken over the RoIs(refers to a specific area in an image likely to contain an object), N is the number of RoIs samples, B represents the new group, $|C_A|$ is the number of classes of old group A, y_A, y_B are the unnormalized objectness logits (how likely the box contains an object), while $\overline{y}_A$, $\overline{y}_B$ are the centered logits and t_A,t_B represents bounding box regression outputs.

The idea of using distillation to preserve knowledge from previous tasks has been widely explored in the context of incremental learning. For example, in (Feng et al., 2022), the authors emphasize the importance of learning responses from both the classification and regression heads of object detection models. Their method, Elastic Response Distillation (ERD), improves upon existing knowledge distillation techniques by elastically learning classification and regression responses to preserve localization information better. The authors also present the Elastic Response Selection (ERS) strategy, which selects the most relevant responses for distillation through statistical analysis, helping to further mitigate the issue of catastrophic forgetting.

In (Dong et al., 2021), the authors addressed the challenge of catastrophic forgetting in class-incremental object detection (CIOD). Unlike prior works that rely on the co-occurrence of base and novel classes in training data, their method considers a more realistic scenario where base classes are entirely absent from the novel class dataset. To tackle this, they introduced a blind sampling strategy that selects relevant samples from unlabeled in-the-wild data based on the responses of a base-class model and a novel-class model. They further proposed a dual-teacher distillation framework, in which knowledge from both base-class and novel-class teacher models is transferred to a student model using sampled in-the-wild data. Their method includes a class remodeling step to treat irrelevant classes as background, image-level distillation using region-of-interest (RoI) masks, and instance-level distillation via response heatmaps.

1.4.4 Object proposal methods

In Selective Search (Uijlings et al., 2013), the authors address the challenge of generating high-quality object proposals for object recognition tasks. They found a way to combine segmentation and exhaustive search, resulting in a novel data-driven selective search method. The proposed approach offers several advantages, including the capacity to capture objects across diverse scales through a hierarchical algorithm, the employment of various strategies to generate grouped regions, and a notable advantage in terms of computational efficiency. The methodology begins by leveraging the (Felzenszwalb and Huttenlocher, 2004) technique to generate all initial regions, followed by iterative grouping and saving these regions based on their similarities until the entire image forms a single region. Notably, the authors introduce four distinct methods for computing the similarity between pairs of regions, considering factors such as color space, texture, size, and the degree of overlap. The high-quality regions computed by Selective Search could be easily adapted to various object detection methods.

In (Zitnick and Dollár, 2014), the authors present an innovative approach to generating object bounding box proposals by utilizing edges. Their idea relies on the observation that the likelihood of a bounding box containing an object is correlated with the number of contours entirely enclosed within it. The authors introduce a unique scoring mechanism for bounding boxes based on edges. This technique employs a Structured Edge Detector (Dollár and Zitnick, 2014) to obtain a dense edge response map, followed by a Non-Maximum Suppression orthogonal to the edge map to get edge peaks. Subsequently, they construct edge groups by combining 8-connected edges until the sum of their orientation differences surpasses a predefined threshold. The authors then calculate the affinity between all pairs of groups and generate contours based on this information. Finally, an objectness score is computed, using the number of edges contained within the box minus those that are part of contours overlapping the box's boundary. This objectness score could be used to improve the quality of proposed boxes computed by some popular proposal-generating methods like sliding windows.

In Multiscale Combinatorial Grouping (MCG) (Pont-Tuset et al., 2016), the authors introduce an optimized normalized cuts algorithm that speeds up the contour globalization process. They also propose a hierarchical segmentation method that effectively integrates multi-scale information. By combining these multi-scale regions, MCG generates accurate object proposals by efficiently exploring their combinatorial space. Additionally, the paper introduces a faster version of MCG, Single-scale Combinatorial Grouping (SCG), which delivers competitive proposal results in under five seconds per image, offering parameter-free generalization and a ranked set of object proposals suitable for various proposal counts.

1.4.5 Class-agnostic object detection

Significant improvements in class-agnostic object detection have been achieved in (Maaz et al., 2022) by training with image-text pairs and utilizing language as a supervisory signal. This approach enables the detection of generic objects across various domains, such as natural images, satellite images, and even artistic representations like sketches and paintings. Multiscale Vision Transformers (MViTs) take advantage of different types of data to learn the common characteristics of objects, which helps them detect a variety of objects without relying on predefined class labels. The architecture has also been improved by incorporating multi-scale deformable attention and late fusion techniques, which make object localization more accurate and efficient. These models have demonstrated state-of-the-art performance in class-agnostic object detection and have been successfully applied to various downstream tasks, such as open-world object detection, saliency detection, and self-supervised learning.

In (Liu et al., 2024), the authors present Grounding DINO, an open-set object detection system that integrates the DINO detector(Zhang et al., 2022),—a Transformer-based framework utilizing self-attention mechanisms for accurate and robust detection without relying on predefined anchor boxes—with grounded pre-training. Grounded pre-training aligns vision and language models by training on image-text pairs, enabling the model to detect objects based on textual descriptions. This enhances the model's ability to handle open-set detection tasks.

The key innovation is integrating language into a closed-set detector to handle open-set scenarios, improving concept generalization for unseen objects. Grounding DINO achieves this by dividing the process into three phases: feature enhancement, language-guided query selection, and cross-modality decoding. It is pre-trained on large-scale datasets, including object detection, grounding, and caption data. Evaluations on various benchmarks (COCO, LVIS, ODinW, RefCOCO) show that Grounding DINO significantly outperforms existing models.

In (Lim et al., 2024), the authors introduce DiPEx (Dispersing Prompt Expansion), a method aimed at improving class-agnostic object detection (OD). While existing vision-language model (VLM)–based approaches have made significant progress in object localization, they often struggle with low recall rates, especially in scenarios with diverse object types and complex contexts. To address this issue, DiPEx employs a self-supervised learning strategy to expand a set of non-overlapping hyperspherical prompts to enhance recall rates and improve performance in downstream tasks like out-of-distribution (OOD) object detection. DiPEx begins by self-training generic parent prompts and selects the ones with the highest semantic uncertainty for further expansion. The child prompts generated capture more fine-grained semantics. Dispersion losses are applied to ensure high inter-class discrepancy among the child prompts while maintaining semantic consistency between parent-child pairs. To prevent excessive growth of the prompt set, DiPEx uses maximum angular coverage (MAC) as a criterion to terminate the expansion process. Extensive experiments on MS-COCO and LVIS datasets show that DiPEx outperforms other prompting methods.

CHAPTER 2

INCREMENTAL TRAINING OF AN OBJECTNESS-BASED REGION PROPOSAL GENERATOR

2.1 Setup and Notation

Anchors: Anchors typically represent a set of fixed initial bounding box guesses. These anchor boxes are placed densely over the entire image, often grid-like, and serve as potential candidate regions for objects of interest. During training, the model predicts offsets (translations and scales) and objectness scores for each anchor to refine their positions and determine whether they contain objects.

The following Table 2.1 introduces the notations used in this project.

Table 2.1: Notations used in this thesis.

$\mathbf{M}$	the feature map
$\mathbf{J}$	the integral image
$\mathbf{F}$	the feature of a bounding box
W, H	the width and height of a feature map
w, h	the width and height of a bounding box
s	the scale of a bounding box where $s = \lfloor \log_2 (w \times h) \rfloor$
n	number of samples
K	number of classes
$\mathbf{x}$, $\mathbf{y}$	feature vector and corresponding labels
$\boldsymbol{\mu}_x$, $\boldsymbol{\mu}_y$	mean of vector x and y
$\mathbf{S}_{xx}$, $\mathbf{S}_{xy}$	running average of squares (see Eq. 2.3)
$\mathbf{X}$, $\mathbf{Y}$	features and labels in a batch
R	a running averages model $(n, \boldsymbol{\mu}_x, \boldsymbol{\mu}_y, \mathbf{S}_{xx}, \mathbf{S}_{xy})$ (Section 2.3.2)
$\boldsymbol{\beta}$	the coefficient vector in OLS
$\mathbb{D}$	the set of multi-scale running averages (RAVEs)

2.2 Method Overview

This project employs a structured approach to object detection, incorporating a backbone network for feature extraction and a multi-scale anchor generation strategy. Instead of a single binary

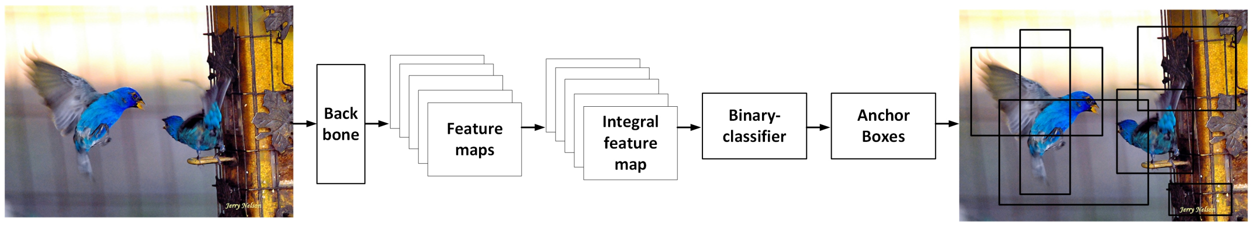

Figure 2.1: Diagram of the proposed anchor generation framework.

classifier, we train multiple binary classifiers tailored for objects of different scales. Additionally, we introduce three distinct methods for selecting the classification threshold of each classifier, enhancing the adaptability and robustness of the detection process. The backbone network extracts features from an image of size $w \times h$ into a map of size $\frac{w}{32} \times \frac{h}{32} \times p$, where p is the number of feature maps (channels). Notably, we introduce an innovative training methodology to train the classifiers using Running Averages (RAVE) (Sun et al., 2024) on features obtained using the integral image. An overview of this process is illustrated in Figure 2.1.

2.3 Preliminary Knowledge

In this section, we will cover three key concepts essential for understanding our approach to object detection: **Integral Image**, which enables efficient computation of image features; **Running Averages**, a statistical method that refines data and dynamically updates model parameters; and **Nelder-Mead Optimization**, a gradient-free algorithm for minimizing objective functions in multidimensional spaces. These concepts form the foundation of our detection framework.

2.3.1 Integral image

The Faster R-CNN approach samples anchors at each location, obtaining only a subset of all possible anchors. To generate anchors of all possible sizes on the feature map, we instead leverage the integral image over the feature maps generated by the backbone network.

The integral image of an image $\mathbf{M}$ is a summed-area table, where each value

$$\mathbf{J}(x,y) = \sum_{\substack{x\prime \le x \\ y\prime \le y}} \mathbf{M}(x', y') \tag{2.1}$$

denotes the sum of values of $\mathbf{M}$ up to position (x, y). It can be computed efficiently using Algorithm 1.

Algorithm 1 IntegralImage

```
**Input:** $\mathbf{M} \in \mathbb{R}^{W \times H \times p}$
Initialize $\mathbf{J}(x, y) \leftarrow \mathbf{M}$ for all $x, y$
**for** $x \in \{2, \ldots, W\}$ **do**
    **for** $y \in \{2, \ldots, H\}$ **do**
        $\mathbf{J}(x, y) \leftarrow \mathbf{M}(x, y) + \mathbf{J}(x-1, y) + \mathbf{J}(x, y-1) - \mathbf{J}(x-1, y-1)$
    **end for**
**end for**
**Output:** $\mathbf{J}$
```

Subsequently, the sum of values of $\mathbf{M}$ over any rectangle can be computed using the integral image values of its four corners.

$$\sum_{\substack{x_0 < x\prime \le x_1 \\ y_0 < y\prime \le y_1}} \mathbf{M}(x', y') = \mathbf{J}(x_1, y_1) + \mathbf{J}(x_0, y_0) - \mathbf{J}(x_1, y_0) - \mathbf{J}(x_0, y_1). \tag{2.2}$$

Finally, the average feature value over a rectangular area is simply obtained as the sum of features divided by the rectangle's size. This method facilitates the inclusion of diverse shape and size anchors, facilitating robust object detection across the feature map.

2.3.2 Running averages

Object detection often incurs significant computational costs, primarily attributed to the high-dimensional features and the substantial requirement for training data samples. We draw inspiration from the running averages (RAVE) method (Sun et al., 2024) in response to this computational burden.

RAVE adopts a setting where the feature information $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^p \times \mathbb{R}^C, i = 1, ..., n$ is condensed into running averages $(n, \boldsymbol{\mu}_x, \boldsymbol{\mu}_y, \mathbf{S}_{xx}, \mathbf{S}_{xy})$, where n is the sample size, and $\boldsymbol{\mu}_x = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$, $\boldsymbol{\mu}_y = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i$, $\mathbf{S}_{xx} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T$, $\mathbf{S}_{xy} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i\mathbf{y}_i^T$. These values can be updated incrementally with a batch of size b with values $(\mathbf{X}_b, \mathbf{Y}_b)$ for instance,

$$\begin{aligned} \boldsymbol{\mu}_x^{(n+b)} &= \frac{n}{n+b}\boldsymbol{\mu}_x^{(n)} + \frac{1}{n+b}\mathbf{X}_b^T\mathbf{1}, \\ \mathbf{S}_{xx}^{n+b} &= \frac{n}{n+b}\mathbf{S}_{xx}^{n} + \frac{1}{n+b}\mathbf{X}_b^T\mathbf{X}_b, \end{aligned} \tag{2.3}$$

and similarly for $\boldsymbol{\mu}_y^{n+b}$ and $\mathbf{S}_{xy}^{n+b}$. These running averages could be used for model estimation in methods such as OLS, Feature Selection with Annealing (FSA), Lasso, and Elastic Net, as described in (Sun et al., 2024). We leverage these running averages to mitigate the computational complexity associated with object detection tasks while maintaining model performance.

2.3.3 Nelder-Mead optimization

The Nelder-Mead method (Singer and Nelder, 2009) is a gradient-free optimization technique utilized to identify the minimum or maximum of an objective function within a multi-dimensional space. It is particularly useful when the gradient of the objective function is either unknown or difficult to compute. The method uses the concept of a simplex, which is a special polytope of $n+1$ vertices in n dimensions. By iteratively adjusting the simplex through a series of transformations, including reflection, expansion, contraction, and shrinkage, the Nelder-Mead algorithm explores the parameter space to locate the optimal solution while adapting to the local landscape of the objective function.

2.3.4 Proposals with Mask R-CNN

Mask R-CNN (He et al., 2017) is an extension of Faster R-CNN that not only detects objects but also generates pixel-wise segmentation masks for each object. It uses a Region Proposal Network (RPN) to generate region proposals and then applies a fully convolutional network (FCN) to predict segmentation masks for each proposal. This architecture is particularly effective for tasks that require both object detection and instance segmentation, providing precise boundaries for each detected object.

In our project, we leverage the robustness of Mask R-CNN's segmentation capability to generate class-agnostic box proposals for our data. This allows us to use high-quality proposals without class bias, which are then passed through the subsequent stages of the pipeline for classification.

2.3.5 Class-agnostic proposals with Grounding-DINO

Grounding DINO (Liu et al., 2024) is a recent technique in object detection that enables the generation of high-quality, class-agnostic object proposals. It combines the power of vision transformers (ViTs) and contrastive learning to produce proposals that are not biased towards any specific object category. The method works by grounding object queries on image features, allowing for precise localization of potential object regions, independent of predefined class labels.

In Grounding DINO, the model is trained to match object queries to feature regions of the image, learning to produce object proposals without the need for specific class annotations. This is achieved through a combination of supervised contrastive learning and query-guided attention mechanisms, which ensure that the generated proposals are robust and accurate across various object types. The class-agnostic nature of these proposals makes them highly flexible and suitable for downstream tasks like multi-class object detection, where the goal is to classify and localize objects from a large variety of categories.

For this project, the proposals generated by Grounding DINO will serve as the starting point for the second stage of the object detection pipeline, which involves a multi-class classification task. These class-agnostic proposals will provide a diverse set of candidate regions that will be further refined and classified in the subsequent stages of the detection process. By utilizing class-agnostic proposals, the model can focus on accurately localizing and classifying objects across multiple categories, enhancing overall detection performance in complex environments.

The key advantage of using Grounding DINO is its ability to generate high-quality object proposals without relying on class-specific supervision, making it an ideal technique for generating versatile proposals in multi-class object detection tasks.

2.4 Incremental Learning with Integral Image Features and RAVE

Consider a $W \times H \times p$ feature map $\mathbf{M}$ generated by the backbone network. Initially, we compute the integral image (2.1). Subsequently, for any rectangle anchor with the top-left corner (x_0, y_0) and width w, height h, its corresponding feature $\mathbf{F} \in \mathbb{R}^p$ could be computed as

$$\mathbf{F}_I(x_0, y_0, w, h) = [\mathbf{J}(x_0, y_0) + \mathbf{J}(x_1, y_1) - \mathbf{J}(x_1, y_0) - \mathbf{J}(x_0, y_1)]/(w \times h), \tag{2.4}$$

where $x_1 = x_0 + w, y_1 = y_0 + h$. This approach allows us to generate a diverse set of anchors with different shapes. In the context of incremental learning settings, efficiently storing and updating vast anchor features is crucial. Thus, we leverage the RAVE method to update running averages for the positive and negative anchor candidates, one image at a time. The RAVEs are then utilized to train a binary classifier that will prune away most anchors that do not contain objects. Moreover, we observed that using separate classifiers for different scale anchors yields superior results compared to training a single classifier for the entire task. Therefore, we maintain separate RAVEs for different sizes of the corresponding anchors for training multiple classifiers. This entire process, described in Algorithm 6, can be implemented efficiently in batches, requiring minimal time for execution.

Algorithm 2 Incremental multi-scale RAVE update

```
Input: Feature map $\mathbf{M} \in \mathbb{R}^{W\times H\times p}$, RAVEs for positive samples $\mathbb{D}_s^{(p)}$, RAVEs for negative samples $\mathbb{D}_s^{(n)}$, $s = 0, ..., S$
Output: Updated RAVEs $\mathbb{D}_s^{(p)}, \mathbb{D}_s^{(n)}, s = 0, ..., S$
1: Compute $\mathbf{J} = \text{IntegralImage}(\mathbf{M}) \in \mathbb{R}^{W\times H\times p}$ using Algorithm 1
2: for $w \in \{1, 2, \dots, W\}$ do
3:   for $h \in \{1, 2, \dots, H\}$ do
4:     compute scale $s = \lfloor \log_2 (w \times h) \rfloor$
5:     for $x_0 \in \{1, 2, \dots, W - w\}$ do
6:       for $y_0 \in \{1, 2, \dots, H - h\}$ do
7:         $x_1 = x_0 + w$
8:         $y_1 = y_0 + h$
9:         $\mathcal{A} = (x_0, y_0, x_1, y_1)$
10:        compute $\mathbf{F} = [\mathbf{J}(x_0, y_0) + \mathbf{J}(x_1, y_1) - \mathbf{J}(x_1, y_0) - \mathbf{J}(x_0, y_1)]/(w \times h) \in \mathbb{R}^p$
11:        if
12:        $\mathcal{A}$ has $IoU \geq 0.5$ with a GT box then
13:          update RAVE $\mathbb{D}_s^{(p)}$ using eq (2.3) with $\mathbf{x} = \mathbf{F}, y = 1$
14:        else
15:          update RAVE $\mathbb{D}_s^{(n)}$ using eq (2.3) with $\mathbf{x} = \mathbf{F}, y = -1$
16:        end if
17:      end for
18:    end for
19:  end for
20: end for
```

2.5 Training a Binary Classifier Using RAVE

In the realm of running-average-based prediction methods, the online Ordinary Least Squares (OLS) method stands out for its simplicity and effectiveness. Recall that in OLS, the coefficient vector $\boldsymbol{\beta}$ is obtained by solving the system $\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}$. Replacing the $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$ by their running-averages $\mathbf{S}_{xx} = \mathbf{X}^T\mathbf{X}/n, \mathbf{S}_{xy} = \mathbf{X}^T\mathbf{y}/n$ we obtain

$$\boldsymbol{\beta} = \mathbf{S}_{xx}^{-1}\mathbf{S}_{xy}. \tag{2.5}$$

This way, the OLS linear model $\boldsymbol{\beta}$ could be obtained online. While the OLS is aimed originally for regression, it can also be used for binary classification using $y \in \{-1, 1\}$.

Due to the significant imbalance in the data distribution in object detection tasks, it is advantageous to combine the regression loss of positive and negative samples with appropriate weights. Additionally, applying an L2 penalty helps enhance the robustness of the model.

The weighted OLS loss function, denoted as $L(\boldsymbol{\beta})$, is defined as follows:

$$\begin{aligned} L(\boldsymbol{\beta}) &= \frac{w_p}{n_p} \sum_{i, y_i = 1} \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right)^2 + \frac{w_n}{n_n} \sum_{i, y_i = -1} \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right)^2 + \lambda \|\boldsymbol{\beta}\|^2 \\ &= \frac{w_p}{n_p} \|\mathbf{X}_p \boldsymbol{\beta} - \mathbf{y}_p\|^2 + \frac{w_n}{n_n} \|\mathbf{X}_n \boldsymbol{\beta} - \mathbf{y}_n\|^2 + \lambda \|\boldsymbol{\beta}\|^2 \end{aligned} \tag{2.6}$$

In practice, we set $w_p = w_n = 1$.

To find the estimated values of $\boldsymbol{\beta}$, we start with the derivative of the loss function $L(\boldsymbol{\beta})$:

$$\frac{\partial L}{\partial \boldsymbol{\beta}}(\boldsymbol{\beta}) = \frac{2w_p}{n_p} \mathbf{X}_p^T (\mathbf{X}_p \boldsymbol{\beta} - \mathbf{y}_p) + \frac{2w_n}{n_n} \mathbf{X}_n^T (\mathbf{X}_n \boldsymbol{\beta} - \mathbf{y}_n) + 2\lambda \boldsymbol{\beta}. \tag{2.7}$$

Setting the gradient to zero yields:

$$\frac{w_p}{n_p} \mathbf{X}_p^T \mathbf{X}_p \boldsymbol{\beta} - \frac{w_p}{n_p} \mathbf{X}_p^T \mathbf{y}_p + \frac{w_n}{n_n} \mathbf{X}_n^T \mathbf{X}_n \boldsymbol{\beta} - \frac{w_n}{n_n} \mathbf{X}_n^T \mathbf{y}_n + \lambda \boldsymbol{\beta} = 0. \tag{2.8}$$

Assume to add $\mathbf{1}_n$ to the first column of $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}, \; \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}. \tag{2.9}$$

Rearranging the terms gives us the linear system:

$$\begin{bmatrix} \frac{w_p}{n_p} + \frac{w_n}{n_n} + \lambda & (w_p \boldsymbol{\mu}_x^p + w_n \boldsymbol{\mu}_x^n)^T \\ w_p \boldsymbol{\mu}_x^p + w_n \boldsymbol{\mu}_x^n & w_p \mathbf{S}_{xx}^p + w_n \mathbf{S}_{xx}^n + \lambda \mathbf{I}_p \end{bmatrix} \boldsymbol{\beta} = \begin{bmatrix} w_p \boldsymbol{\mu}_y^p + w_n \boldsymbol{\mu}_y^n \\ w_p \mathbf{S}_{xy}^p + w_n \mathbf{S}_{xy}^n \end{bmatrix} \tag{2.10}$$

To apply this method, we first maintain separate RAVEs for the positive samples and for the negative samples, denoted as $\mathbb{D}_s^{(p)}$, $\mathbb{D}_s^{(n)}$. From these RAVEs, we can then directly derive the mean sum of squares $\mathbf{S}_{xx}^p$, $\mathbf{S}_{xx}^n$ as well as the mean vectors $\boldsymbol{\mu}_x^p$ and $\boldsymbol{\mu}_x^n$ for the positive and negative samples, respectively. In Figure 2.2 is shown a diagram of the procedure.

Experimental findings suggest that refining the foreground-background classifier through multiple OLS models, each tailored to specific anchor scales yields superior performance. Given the RAVEs for multiple anchor scales, we train online weighted OLS for each scale s individually and save the corresponding $\boldsymbol{\beta}_s$ for the multi-scale detection process. This approach not only enhances adaptability but also augments the accuracy of object detection models, particularly in contexts characterized by varying scales and complex backgrounds.

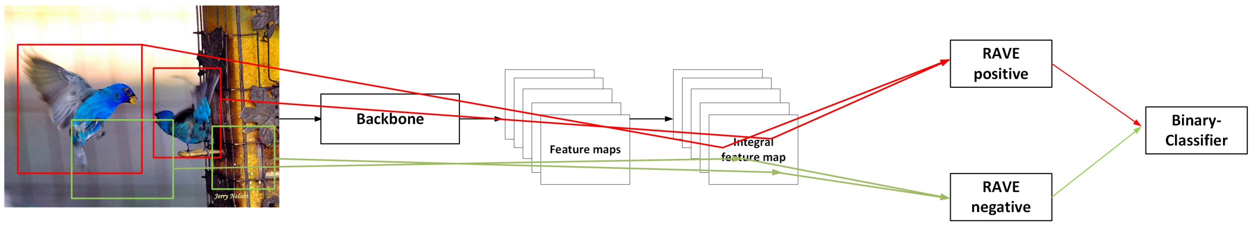

Figure 2.2: Training process of the binary classifier.

2.6 Generating Anchor Proposals

In this section, we describe the process of generating and selecting proposed anchors based on the feature map $\mathbf{M} \in \mathbb{R}^{W \times H \times p}$, which is output from the backbone network, and the multi-scale binary classifier C_s, previously introduced in Section 2.4. The goal is to efficiently propose anchors that are likely to contain objects, which are then refined for object detection.

Given that the anchor classifier C_s is linear, we exploit the property that $C_s(\mathbf{J}(\mathbf{M})) = \mathbf{J}(C_s(\mathbf{M}))$, allowing us to speed up the computation process. Based on that, we directly input the feature map $\mathbf{M}$ into the anchor classifier C_s to produce a score map $\mathbf{Z}$. The score map Z assigns an objectness score to each location of the feature map, corresponding to the likelihood of the location containing an object.

To further accelerate the process, we construct an integral image $\mathbf{J}_s(x, y)$ for each scale s considered by the classifier using Eq 2.2.

For each scale s, we compute the average score using integral image for each anchor of a specific width w and height h, which serves as the objectness score of the anchors. Anchors are assigned to a scale based on their area, where the scale s is determined by the condition $s = \lfloor \log_2 (w \times h) \rfloor$. Anchors with objectness scores that exceed a predefined threshold are retained as potential candidates. This selection criterion helps filter out low-quality proposals, keeping only those that are likely to contain objects.

After selecting the anchors for a given scale, we proceed to gather the proposed anchors along with their corresponding scores. This process is repeated iteratively for all desired anchor sizes across multiple scales, ensuring a robust set of anchor proposals.

Finally, to remove redundant anchor proposals and refine the set of candidates, we apply Non-Maximum Suppression (NMS) independently for each scale classifier. The anchor proposals are generated from resized augmentations of the original image. Before performing NMS, these anchors are resized back to their corresponding dimensions of the original image, ensuring that the suppression process is effective for each classifier. NMS suppresses anchors with lower scores that overlap significantly with higher-scoring anchors, ensuring that only the most confident and non-overlapping proposals are retained. This results in a final set of anchor proposals, which are then passed to the subsequent stages of the object detection pipeline for further processing and refinement.

Algorithm 3 Anchor generator

```
Input: Feature map $\mathbf{M} \in \mathbb{R}^{W \times H \times p}$, anchor scales $S$, binary anchor classifiers $C_s, s \in S$, thresholds $T = \{\tau_s, s \in S\}$
Output: Anchors $\mathcal{A}_s, s \in S$
1: for scale $s \in S$ do
2:   Compute score map $\mathbf{M}_s = \mathbf{M}\boldsymbol{\beta}_s$
3:   Compute $\mathbf{J} = \text{IntegralImage}(\mathbf{M}_s) \in \mathbb{R}^{W \times H}$ using Algorithm 1
4:   $\mathcal{A}_s = \varnothing$
5:   for $(w, h) \in \{1, ..., W\} \times \{1, ..., H\}$ do
6:     if $\lfloor \log_2 (wh) \rfloor = s$ then
7:       for $(x_0, y_0) \in \{1, ..., W - w\} \times \{1, ..., H - h\}$ do
8:         Compute objectness score: $o = \frac{I(x_0,y_0)+I(x_0+w,y_0+h)-I(x_0,y_0+h)-I(x_0+w,y_0+h)}{wh}$
9:         if $o > \tau_s$ then
10:          $\mathcal{A}_s \leftarrow \mathcal{A}_s \cup \{(x_0, y_0, x_0 + w, y_0 + h, o)\}$
11:        end if
12:      end for
13:    end if
14:  end for
15:  $\mathcal{A}_s \leftarrow NMS(\mathcal{A}_s)$
16: end for
```

2.7 Multi-scale Classification

Since objects can vary in scale, we trained multiple classifiers independently for each scale, making the selection of thresholds for each classifier crucial. We present three possible methods for choosing these thresholds and compare their effectiveness. Additionally, to optimize computational resources for subsequent training, we need to limit the total number of predicted boxes generated by all classifiers. In this project, we restrict each image to approximately 1,000 candidate boxes.

2.7.1 Proportional allocation of candidates

A natural approach is to distribute the quota of the 1,000 candidate boxes proportional to the number of boxes generated by each classifier. The quota for each classifier $q_s = \frac{n_s}{\sum n_s} \times 1000$, where n_s is the number of boxes of the scale s classifier. We then arrange the prediction boxes produced by each classifier in descending order according to their corresponding scores and select the allocated number of boxes for each classifier. This method has the advantage of comprehensively retaining objects of varying scales and eliminates the need to establish a threshold for each classifier, facilitating incremental filtering of prediction boxes. However, a potential disadvantage is that we might allocate some quota to classifiers with poor performance, which could result in insufficient selection from more effective classifiers, ultimately impacting the final detection rate.

2.7.2 Best threshold combination for two classifiers

During the research process, we experimented with various combinations of classifiers and evaluated the detection rates achieved by the predicted boxes from each combination. Our findings indicate that for this dataset, it is not necessary to utilize all classifiers to attain a relatively high detection rate. Notably, without considering the classifier thresholds, we can theoretically detect all ground-truth boxes using at least two classifiers. Consequently, we implemented a straightforward algorithm that employs all scores as thresholds to identify the combination that yields the highest detection rate. The algorithm is described in Algorithm 4 below.

Algorithm 4 Threshold Selection for two classifiers

```
Input: Total number of images $n_i$, number $n_d$ of desired proposals per image, total number $\boldsymbol{n_{pos}}$ of ground truth boxes
Sorted scores $\boldsymbol{s_1}$, $\boldsymbol{s_2}$ for all anchors generated by the two classifiers
Sorted scores $\boldsymbol{d_1}$, $\boldsymbol{d_2}$ and the corresponding box ids $\boldsymbol{j_1}$, $\boldsymbol{j_2}$ for all detections generated by the two classifiers
Output: Best thresholds $\tau_1, \tau_2$ for the two classifiers.
1:  $N = n_{pos} \cdot n_d$
2:  for $i = 1$ to $|\boldsymbol{d_1}|$ do
3:      $t_1 \leftarrow \boldsymbol{d_1}[i]$
4:      $n_1 = |\boldsymbol{s_1} \geq t_1|$
5:      if $n_1 > N$ then
6:          continue {Skip iteration if the limit is exceeded}
7:      end if
8:      $n_2 = \min(|\boldsymbol{s_2}|, N - n_1)$
9:      $t_2 = \boldsymbol{s_2}[|\boldsymbol{s_2}| - n_2]$
10:     $A_1 = \{\boldsymbol{j_1}[i], \boldsymbol{d_1}[i] \geq t_1\}$ {IDs from detection set 1 with scores above threshold}
11:     $A_2 = \{\boldsymbol{j_2}[i], \boldsymbol{d_2}[i] \geq t_2\}$ {IDs from detection set 2 with scores above threshold}
12:     $A = A_1 \cup A_2$ {The union of the two detection sets}
13:     $d_r = \frac{|A|}{n_{pos}}$ {Detection rate as ratio of unique IDs to total positives}
14:     if $i = 1$ or $d_r > d_{max}$ then
15:         $\tau_1 = t_1, \tau_2 = t_2, d_{max} = d_R$
16:     end if
17: end for
```

2.7.3 Incremental Nelder-Mead optimizer

The proposed approach has S classifiers, each requiring a threshold. To find the combinations of thresholds for the classifiers to maximize the detection rate and minimize the number of anchor proposals per image simultaneously, we leverage the Nelder-Mead optimization method.

To control the trade-off between detection rate and number of anchors per image, we define two functions $f : \mathbb{R}^S \to \mathbb{R}$ and $g : \mathbb{R}^S \to \mathbb{R}$, both defined over the S threshold combinations of the classifiers, and where f computes the corresponding detection rate, and g computes the average (per image) of the total number of anchors with scores greater than the corresponding thresholds. Then we use the Nelder-Mead method to optimize the objective function:

$$M_{loss} = f - \alpha g \tag{2.11}$$

where α is a hyperparameter.

To update the thresholds for each classifier using the Nelder-Mead method in an incremental manner, we need to gather relevant information from the data to update the functions in Equation (2.11). To update the detection rate, we collect all the features of ground truth bounding boxes. Subsequently, the score matrix can be computed by multiplying these features with the corresponding $\boldsymbol{\beta}$. By giving the score matrix with the corresponding thresholds to function f, we obtain the updated detection rate. To determine the number of anchors exceeding the threshold per image, we can take a subsample of the dataset whenever new classes are introduced. For each classifier, we divide the objectness scores into 100 equally spaced intervals within a specified range and calculate how many anchor proposals fall into each interval. By applying the threshold combination and anchor count matrix to function g, we can easily compute the number of anchor proposals with scores above the corresponding threshold for each classifier, and then sum these counts across all classifiers average over the images. In practice, we set tuning the α and limit each classifier to not generate more than 1000 candidate boxes to further reduce the number of prediction boxes.

CHAPTER 3

ANCHOR PROPOSAL GENERATOR EXPERIMENTS

3.1 Datasets

The following datasets are used in this project.

The **Common Objects in Context (COCO)** dataset (Lin et al., 2014) comprises a comprehensive compilation of over 200,000 images spanning 80 object categories. It includes annotations for various tasks such as object detection, segmentation, and captioning.

The **ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)** (Russakovsky et al., 2015) comprises a vast collection of 456,567 training images and 20,121 validation images with bounding box annotations for object detection, featuring 200 different class categories. However, 17 out of the 200 classes are duplicated in COCO, and among the remaining 183 classes, 3 have fewer than 150 images. Consequently, we choose to work with the remaining 180 classes for our experiments. We removed classes without corresponding test data and those with insufficient training data.

In this project, we use a Faster-RCNN model pretrained on the COCO dataset to obtain a robust backbone. This pre-trained backbone is then utilized in our method for training and evaluation on the 20 classes that are randomly picked from ILSVRC2015 dataset.

3.2 Metrics

In our experiments, the distribution of anchor proposals is highly imbalanced, with only 2% of all proposals being positive samples. At this stage, our primary goal is to maximize the detection rate while keeping the number of anchor proposals within a reasonable limit. Therefore, we use recall (the detection rate of objects) as the evaluation metric.

Recall is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.1}$$

In our setting, an anchor is considered a true positive (TP) if its Intersection over Union (IoU) with a ground truth bounding box exceeds 0.5; otherwise, it is classified as a false positive (FP). By definition, the sum of true positives and false negatives corresponds to the total number of objects.

3.3 Training Details

3.3.1 Preprocessing

All input images are resized to have both width and height within the range from 800 to 1333 pixels. Additionally, the images are normalized using the following image mean and standard deviation: mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]. Finally, image padding with zeros is implemented to ensure that the new image dimensions are multiples of 32 for both width and height.

3.3.2 Faster-RCNN model

To compare with the Faster R-CNN model, we initially train a Faster R-CNN on the PASCAL-VOC dataset, utilizing a pre-trained ResNet50 backbone previously trained on the COCO dataset. In this project, we specify the last three layers out of five as the trainable layers in the backbone, while freezing the remaining layers. Subsequently, a Region Proposal Network (RPN) layer is employed to generate 15 anchors per location. The anchor sizes are defined as $\sqrt{h \times w}$ and in this project, they are set to (32,64,128,256,512), with aspect ratios set to (0.5,1.0,2.0). Furthermore, the first four feature maps from the feature pyramid network (FPN) are utilized for training purposes. During training, we employ the Stochastic Gradient Descent (SGD) optimizer, with a scheduled learning rate. The initial learning rate is set to 0.01, and it's multiplied by 0.33 for every 3 epochs.

The model is trained for 15 epochs until convergence, with a batch size of 1. Finally, we retain the parameters that yield the best Mean Average Precision (MAP) result for further evaluation and comparison.

3.3.3 Foreground-background classifier

After completing the training of the base Faster R-CNN model, we extract the feature generator from it for our method. In practice, we retain only the feature map that downsamples the original image by a factor of 32, plus the first convolutional layer in RPN. We found that this configuration works better for a linear model. This feature generator is frozen and will not be trained as part of our method.

To enhance the robustness of the classifier, we use data augmentation on the input images. This augmentation involves resizing the images into 20 different sizes based on the shortest side, with the width and height ranging from 200 to 800 pixels on an exponential grid.

Given a feature map and the corresponding ground truth (GT) box location, we construct an integral image according to Eq. (2.2). We go through combinations of width w and height h values with aspect ratio $w/h \in (0.2, 15)$ to calculate the corresponding box feature average values. For every box, we compute the IoU with respect to all GT boxes and retain the highest IoU value. Subsequently, we compute the scale of the box using the formula $s = \lfloor \log_2(wh) \rfloor$ and update the corresponding positive RAVE by including features of boxes with IoU greater than 0.7 and the negative RAVE for those with IoU less than 0.2. We continually update the RAVEs while iterating over all box sizes and input images.

During testing, we still scale the image and examine all combinations of length and width within the integral image. Each combination is passed to the corresponding classifier based on the anchor scale, yielding a score for each candidate box. After gathering all candidate boxes, we filter them according to the threshold set by each classifier. Subsequently, we restore them to their original size based on their corresponding scales and perform Non-Maximum Suppression (NMS) separately on the boxes obtained from each classifier according to their scale prior to restoration.

3.3.4 Multi-scale classification

In the process of merging anchor boxes obtained from each classifier after applying NMS, we employed and compared the merging strategies discussed in Section 2.7.

Proportional allocation based on the number of candidates per classifier. Since our goal is to produce approximately 1000 anchor boxes per image, we first record the number of candidates generated by each classifier for each image. We then distribute the final anchor boxes produced by each classifier proportional to these numbers, as specified in Section 2.7.1. This method is computationally efficient and achieves a good detection rate.

Combination of the best two classifiers. During the experiment, we observed that it is sufficient to combine two classifiers to achieve the optimal detection rate. Specifically, combining the classifier with scale $s = 3$ and the classifier with scale $s = 4$, or alternatively, the classifier with scales 2 and 4, yielded the highest detection rate. To determine the optimal thresholds for each classifier in the combination, we used Algorithm 4. In total, the sampled 20 classes contain 9767 training images and 11223 ground truth boxes. We set the desired number of proposals per image to 900 on the training set to ensure efficient processing and limited proposals in the test phase.

Incremental Nelder-Mead optimizer

Another approach for determining the threshold for each classifier is to employ an optimization algorithm, such as the Nelder-Mead optimization algorithm. In this project, we constructed an objective function comprising two sub-functions, f, and g, and introduced a hyperparameter α to control the weight of the function g. For the function g, we first created 100 equally spaced score intervals ranging from -20 to 20. Then, for each class, we randomly selected 10 images and calculated how many proposals fell within each interval, enabling rapid estimation of the average number of proposals per image for any given threshold.

In practice, we have 10 classifiers, so we initialized the Nelder-Mead algorithm with a ten-dimensional unit simplex, setting 0 as the initial value and assigning $\alpha = 0.000003$. To further control the number of proposals, we imposed an additional constraint, limiting each individual classifier to select a maximum of 1000 proposals based on their corresponding scores.

‘

3.4 Experiments

We start with an ablation study to show the usefulness of our proposed approach, then continue with a comparison with existing anchor proposal methods.

3.4.1 Ablation study

To evaluate the effectiveness of our proposed approach, we conducted an ablation study comparing the performance of using no classifier, a single classifier, and the combination of two classifiers for further filtering the anchor boxes.

No Classifier. As a baseline, we used the raw anchor boxes without applying any further filtering. This approach indicates the highest detection rate we could get, with a large number of false positives.

Single Classifier. We then applied each of the trained scale-specific classifiers individually to filter the anchor boxes. While this helps to prune some false positives over the baseline, the results were limited by the capacity of a single classifier to capture all relevant aspects of the data.

Combination of Two Classifiers. Finally, we combined two of the best-performing classifiers, as described in Section 2.7.2. This approach resulted in the most significant improvement, with a noticeable reduction in false positives and more accurate localization. By leveraging the strengths of both classifiers, we achieved a more robust filtering process that led to better overall detection performance.

In Table 3.1 are shown the results of the test dataset. They highlight the trade-off between recall and the number of anchors per image. Without any classifiers (No clf), the recall is the highest at 92.43%, but this comes at the cost of a significantly higher number of anchors per image (266,894). This indicates that while detection performance is optimal, the computational and storage costs are also substantial.

When individual classifiers are applied, the number of anchors per image is significantly reduced, but this comes at the cost of recall. For example, with clf 2, the recall drops to 59.05%, while the number of anchors per image is constrained to 1,000. Similarly, with clf 3 and clf 4, the recall reaches 79.35% and 84.91%, respectively, with anchor counts of 943 and 613. These results demonstrate our ability to limit the number of anchors to within 1,000, but this constraint leads to a reduction in recall.

However, when combining classifiers (e.g., clf 2 & 4 and clf 3 & 4), there is an improvement in recall, reaching 89.43% and 86.73%, respectively, while the number of anchors per image remains relatively moderate (1,056 and 1,004). This indicates that combining classifiers can strike a balance between recall and anchor efficiency, achieving better detection performance while keeping the number of anchors at a manageable level.

Table 3.1: Test detection rate and number of anchors per image for different classifier combinations.

20 classes	No clf	clf 2	clf 3	clf 4	clf 2 & 4	clf 3 & 4
recall	92.43%	59.05%	79.35%	84.91%	89.43%	86.73%
n_anchors/img	266894	1000	943	613	1056	1004

3.4.2 Comparison with existing anchor generation methods

This section evaluates different screening strategies for the foreground-background classifier that are proposed in this paper, which are proportional allocation of candidates (Section 2.7.1), best threshold combination for two classifiers (Section 2.7.2) and the incremental Nelder-Mead optimizer (Section 2.7.3). Moreover, it compares them with several methods, including some widely used object proposal method such as: the region proposal network in Faster-RCNN (Ren et al., 2015), Edgeboxes (Zitnick and Dollár, 2014), Selective Search (Uijlings et al., 2013), Mask-RCNN (He et al., 2017), and Grounding DINO (Liu et al., 2024).

The results of the test dataset are summarized in Table 3.2, where the detection rates and the average number of anchors per image for each method are reported.

Among these, Selective Search demonstrates the highest detection rate of 96.17%, but it comes with the cost of the largest number of anchors per image, at 1491. In contrast, Edgeboxes achieves the lowest detection rate of 30.29%, although it requires significantly fewer anchors, with an average of only 435 per image. The Faster-RCNN approach achieved a detection rate of 79.60%, with an average of 920 anchors per image.

The proportional allocation strategy achieves a detection rate of 86.43% with 996 anchors per image, while the combo of two classifiers method improves upon this, reaching 89.43% with 1,056 anchors per image. The Nelder-Mead approach performs similarly to proportional allocation, achieving a detection rate of 88.69% with 1,063 anchors per image.

Interestingly, MRCNN and Grounding DINO (GDINO) show detection rates of 81.86% and 83.33%, respectively. However, both methods come with significant computational demands. MRCNN requires a large number of parameters ($4.4\cdot 10^7$), while GDINO demands the highest parameter count ($1.72\cdot 10^8$). These results highlight the trade-offs between detection performance, the number of anchors, and model complexity. While methods like Selective Search show high detection rates, they also comes with a large number of false positives. On the other hand, methods like Grounding DINO achieve competitive detection rates with fewer anchors, albeit at the cost of significantly higher model complexity. Overall, these results demonstrate that a well-designed screening strategy can significantly enhance detection performance, but this improvement comes with varying computational demands.

Table 3.2: Comparison of different anchor generation method

20 classes	Prop.	Combo of 2	N-M	FRCNN	FRCNN1	Edge	Selective	MRCNN	GDINO
recall	86.43	89.43	88.69	76.79	79.60	30.29	96.17	81.86	83.33
n_anchors	996	1056	1063	998	920	435	1491	30	11
n_parameters	10250	2050	10250	$2.7\cdot 10^7$	$2.7\cdot 10^7$	0	0	$4.4\cdot 10^7$	$1.72\cdot 10^8$

CHAPTER 4

CLASS-INCREMENTAL OBJECT DETECTION WITH PROBABILISTIC PCA CLASSIFIERS

4.1 Method Overview

In this chapter, we commence from the second stage of object detection. Given a set of anchor proposals, we extract their corresponding features using a model specifically designed for multi-class classification. For each individual category, we train a dedicated Probabilistic Principal Component Analysis (PPCA) model. Furthermore, we employ the Running Averages (RAVE) method to update these PPCA models online, ultimately refining the final set of predicted bounding boxes. A structural overview of this process is illustrated in Figure 4.1.

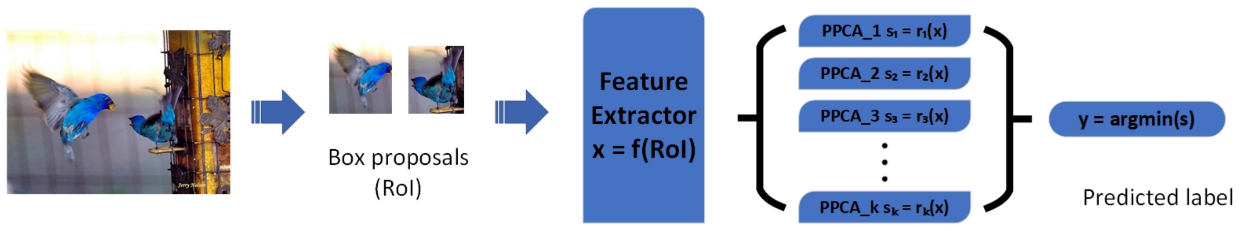

Figure 4.1: PPCA-based Object detector framework.

4.2 Preliminary Knowledge

4.2.1 ROI-Align

In modern object detection, Region of Interest (ROI) Align is a vital algorithm used to address misalignment issues in feature extraction. This method is especially useful in architectures like Faster R-CNN (Ren et al., 2015), where accurate feature representation from regions of interest is crucial. Unlike the traditional ROI Pooling method that quantizes coordinates, ROI Align uses bilinear interpolation to retain spatial precision, ensuring that features are aligned more accurately.

For this project, a technique inspired by ROI Align is applied to compute the features of anchor proposals on the feature map. The goal is to extract precise features for each anchor while maintaining accurate localization. Instead of quantifying the coordinates of the anchor boxes, which could lead to misalignment, the coordinates are mapped directly onto the feature map. Bilinear interpolation is then used to extract features at non-integer positions, which helps preserve spatial details. The feature for each anchor box is computed by averaging the features at $Z * Z$ sampling points within the anchor region. Given an anchor box with coordinates (x_0, y_0) and (x_1, y_1) in the image space and a feature map M, the coordinates are first transformed to the feature map space by considering the scale factor s between the image and the feature map. The mapped coordinates are:

$$x' = \frac{x_0}{s}, \quad y' = \frac{y_0}{s}, \quad x'' = \frac{x_1}{s}, \quad y'' = \frac{y_1}{s}.$$

After determining the location of the anchor box on the feature map, we divide the region into a grid of $Z \times Z$ sampling points. These points are evenly distributed within the anchor box. For each sampling point (x_i, y_i), bilinear interpolation is applied to calculate the feature value at that location. The bilinear interpolation formula for a non-integer coordinate (x, y) is:

$$M(x, y) = (1-\alpha)(1-\beta)M_1 + \alpha(1-\beta)M_2 + (1-\alpha)\beta M_3 + \alpha\beta M_4, \tag{4.1}$$

where: $\alpha = x - \lfloor x \rfloor$ and $\beta = y - \lfloor y \rfloor$ are the fractional parts of the x and y coordinates, M_1, M_2, M_3, M_4 are the feature values at the four nearest grid points surrounding (x, y). Once the feature values at all N sampling points are obtained, they are averaged to compute the final feature for the anchor box:

$$\mathbf{F} = \frac{1}{N} \sum_{i=1}^{Z*Z} M_i(x_i, y_i). \tag{4.2}$$

This ensures the features extracted from each anchor box are aligned correctly, avoiding the misalignment that can occur when using discrete grid-based methods.

Algorithm 5 ROI_feature

```
Input: anchor box $\mathcal{A} = (x_0, y_0, x_1, y_1)$, factor $s$, number of sampling points $Z * Z$, feature map $M$
Output: ROI features $\mathbf{F}$
1: Transform the anchor box coordinates to the feature map space
$x' = \frac{x_0}{s}, \quad y' = \frac{y_0}{s}, \quad x'' = \frac{x_1}{s}, \quad y'' = \frac{y_1}{s}$
2: Divide the anchor box region into $Z * Z$ evenly distributed sampling points
For each sampling point $(x_i, y_i)$ within the anchor box:
    Calculate the position of the sampling point $(x_i, y_i)$ based on the anchor box coordinates
3: Apply bilinear interpolation to obtain feature values for each sampling point
for each sampling point $(x_i, y_i)$ do
    Compute the fractional parts $\alpha = x_i - \lfloor x_i \rfloor$ and $\beta = y_i - \lfloor y_i \rfloor$
    Determine the four nearest grid points: $(M_1, M_2, M_3, M_4)$ surrounding $(x_i, y_i)$
    Use bilinear interpolation formula to compute the feature at $(x_i, y_i)$:
    $M(x_i, y_i) = (1-\alpha)(1-\beta)M_1 + \alpha(1-\beta)M_2 + (1-\alpha)\beta M_3 + \alpha\beta M_4$
end for
4: Compute the final feature vector by averaging the values at all sampling points
$\mathbf{F} = \frac{1}{Z*Z} \sum_{i=1}^{Z*Z} M(x_i, y_i)$
Return: $\mathbf{F}$
```

4.2.2 Probabilistic PCA

As an incremental classification model, Probabilistic PCA (PPCA) excels at modeling class distributions and providing accurate probability scores, enabling precise categorization of object features across multiple classes. The effectiveness of this approach has been demonstrated in previous research (Wang and Barbu, 2022).

The PPCA model assumes that a high-dimensional observation $\mathbf{x} \in \mathbb{R}^d$ lies in a lower dimensional space parameterized by a lower-dimensional latent variable $\mathbf{t} \in \mathbb{R}^q$, where $q \ll d$, with added Gaussian noise. The relationship between the observation $\mathbf{x}$ and latent variable $\mathbf{t}$ is expressed as:

$$\mathbf{x} = \mathbf{W}\mathbf{t} + \boldsymbol{\mu} + \boldsymbol{\epsilon}. \tag{4.3}$$

Here, $\mathbf{W} \in \mathbb{R}^{d \times q}$ is a projection matrix that maps the latent variable $\mathbf{t} \in \mathbb{R}^q$ to the observation space, $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean of the observations, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$ represents Gaussian noise. The latent variable $\mathbf{t}$ follows a standard normal distribution $\mathbf{t} \sim \mathcal{N}(0, \mathbf{I}_q)$.

Given this model, the conditional distribution of $\mathbf{x}$ given $\mathbf{t}$ is:

$$\mathbf{x}|\mathbf{t} \sim \mathcal{N}(\mathbf{W}\mathbf{t} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}_d). \tag{4.4}$$

By integrating out the latent variable $\mathbf{t}$, we derive the marginal distribution of $\mathbf{x}$ as a multivariate Gaussian:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}_d). \tag{4.5}$$

In order to leverage this method in a multi-class classification task, we characterize class k by a Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k = \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}_d$. For classification, we compute the likelihood of an observation $\mathbf{x}$ belonging to class k:

$$p(\mathbf{x}|y = k) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right). \tag{4.6}$$

In practice, we simplify this by focusing on the Mahalanobis distance (McLachlan, 1999) between $\mathbf{x}$ and the mean of class k:

$$r_k(x) = (\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k). \tag{4.7}$$

A smaller value of $r_k(x)$ indicates that $\mathbf{x}$ is more likely to belong to class k.

Training the PPCA model for each class requires estimating the class-specific mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. The mean $\boldsymbol{\mu}_k$ is calculated as the average of the observations in class k:

$$\boldsymbol{\mu}_k = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i. \tag{4.8}$$

To estimate the covariance matrix $\boldsymbol{\Sigma}_k$, we first compute the sample covariance matrix from the class observations using Singular Value Decomposition (SVD). The SVD decomposition is expressed as:

$$\mathbf{V}\mathbf{D}\mathbf{V}^T = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T. \tag{4.9}$$

In this formulation, defining $\mathbf{L}$ as the first q columns of $\mathbf{V}$, corresponding to the principal components that capture the most variance in the data, the PPCA covariance matrix is approximated to:

$$\boldsymbol{\Sigma}_k \approx \mathbf{L}\mathbf{D}\mathbf{L}^T + \lambda \mathbf{I}_d. \tag{4.10}$$

Here, λ represents a small regularization term (e.g., $\lambda = 0.01$). The covariance matrix of class k can be derived from RAVE using the equation:

$$\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T = \frac{n}{n-1}(\mathbf{S}_{xx_k} - \boldsymbol{\mu}_{x_k}\boldsymbol{\mu}_{x_k}^T) \tag{4.11}$$

Theorem 1. *In (Wang and Barbu, 2022), the score in* (4.7) *can be reformulated as:*

$$r(\mathbf{x}, \boldsymbol{\mu}, \mathbf{L}, \mathbf{d}) = \frac{\|\mathbf{x} - \boldsymbol{\mu}\|^2}{\lambda} - \frac{\|\mathbf{u}(x)\|^2}{\lambda}, \tag{4.12}$$

where $\mathbf{u}(\mathbf{x}) = diag\left(\sqrt{\frac{\mathbf{d}}{\mathbf{d}+\lambda}}\right)\mathbf{L}^T(\mathbf{x} - \boldsymbol{\mu})$*, and the determinant is given by:*

$$\log|\boldsymbol{\Sigma}| = (d - q)\log\lambda + \sum_{i=1}^{q}\log(d_i + \lambda). \tag{4.13}$$

Here, diag($\mathbf{v}$) constructs a diagonal matrix from the elements of the vector $\mathbf{v}$, and $\sqrt{\mathbf{v}}$ applies the square root element-wise to the vector. Computing $r(\mathbf{x})$ using (4.12) can be 10 to 100 times faster than (4.7), as (4.12) only involves multiplication with the $q \times d$ matrix $\mathbf{L}^T$, where q is typically 10 to 100 times smaller than d.

In figure 4.2 is a diagram of the PPCA classifier.

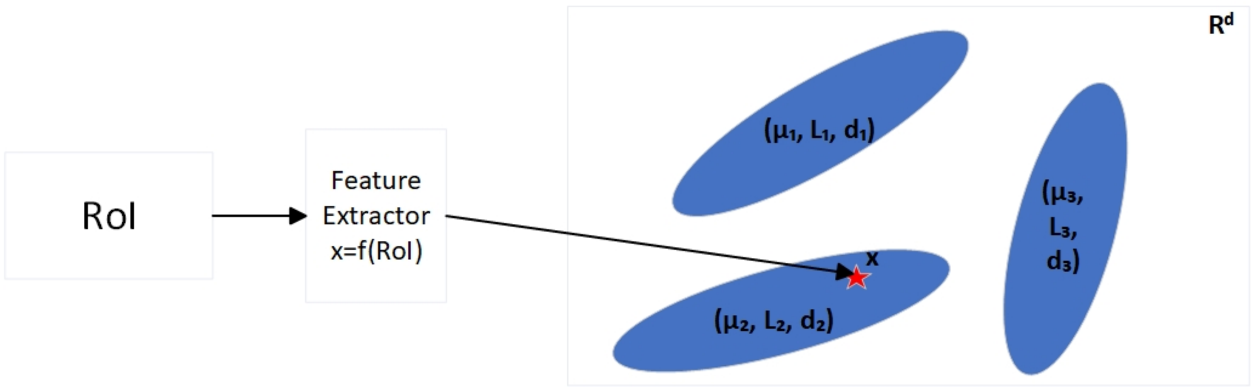

Figure 4.2: Diagram of PPCA classifier

4.2.3 CLIP

The Contrastive Language–Image Pretraining (CLIP) model (Radford et al., 2021) represents a significant advancement in the integration of vision and language modalities. CLIP is designed to learn visual concepts from natural language descriptions, enabling it to perform a wide range of tasks, such as zero-shot classification and image retrieval, without the need for additional task-specific training.

At the core of CLIP's approach is the use of a large dataset of image-caption pairs, which allows the model to learn rich representations through a contrastive objective. In this framework, both image and text encoders are trained to project inputs into a shared embedding space, where the cosine similarity between their representations serves as the basis for their alignment. Specifically, given an image and a set of text descriptions, CLIP aims to maximize the cosine similarity for matching pairs while minimizing it for non-matching pairs.

One of the most notable features of CLIP is its ability to generalize to unseen tasks, thanks to its extensive pretraining on diverse data sources. This enables CLIP to be applied to various tasks directly with little or no additional fine-tuning, distinguishing it from traditional supervised models that often require task-specific datasets. Given this capability, we plan to utilize CLIP as a feature extractor in our project, allowing us to efficiently harness its rich visual and textual representations for our multi-class classification objectives.

4.3 Incremental Multi-class Object Detection with PPCA

In this section, we will explain the process of collecting RAVEs from anchor proposals and utilizing them to train the PPCA model. We will also detail how the trained PPCA model is applied to perform multi-class classification on anchor proposals.

4.3.1 Collecting RAVEs for the multi-class classification model

In this process, based on experimental findings, we discovered that the CLIP model generates features more suitable for multi-class classification tasks. Therefore, at this stage, we will extract features from the original images again using the pre-trained CLIP model, obtaining feature maps that downsample the original image by a factor of 32, similar to the previous feature extractor.

Building on the methodology outlined earlier, we will recollect RAVEs by using the ground truth boxes along with the corresponding CLIP features. According to Equation (2.3), we will now only focus on collecting the mean vector $\boldsymbol{\mu}_{\boldsymbol{x}}^{\boldsymbol{k}}$ and the covariance matrix $\boldsymbol{S}_{\boldsymbol{xx}}^{\boldsymbol{k}}$ for each class k, as these are sufficient for the calculation of the covariance matrix. For this multi-class classification task involving k categories, we stored k RAVEs. By iterating over all ground truth boxes, we build a comprehensive set of RAVEs to train the PPCA model. The following algorithm 6 outlines the process of collecting RAVEs and preparing them for training the PPCA model.

Once the RAVEs for each category have been collected, we proceed to compute the projection matrix $\mathbf{L}$ for a given number of principal components using Equations (4.9),(4.10), and (4.11).

Algorithm 6 Incremental multi-class RAVE update

```
Input: Feature map $\mathbf{M} \in \mathbb{R}^{W \times H \times p}$, ground truth boxes $\mathcal{A} = \{a_1, ..., a_n\}$, RAVEs for positive samples $\mathbb{D}_k^{(p)}$,
Output: Updated RAVEs $\mathbb{D}_k^{(p)}, k = 1, ..., K$
1: for $i = 1$ to $n$ do
2:     Compute $\mathbf{F} = ROI(a_i, \mathbf{M})$ using Alg. 5
3:     Update $\boldsymbol{\mu}_x^p$ and $\mathbf{S}_{xx}^p$ in RAVE $\mathbb{D}_k^{(p)}$ using Eq (2.3) with $\mathbf{x} = \mathbf{F}$
4: end for
```

4.3.2 Training PPCA

After collecting the RAVEs for all samples, we extract the necessary components for training the PPCA model as outlined in Section 4.2.2. Specifically, we use the covariance matrix of each RAVE to compute the projection matrix $\mathbf{L}$ and diagonal matrix $\mathbf{d}$ according to Eqns(4.9), (4.10) and (4.11). Along with the mean vector $\boldsymbol{\mu}_x$, these extracted components are stored in PPCA models $\mathbb{D}_k^{ppca} = (\boldsymbol{\mu}_k, \mathbf{L}_k, \mathbf{d}_k), k = 1, ..., K$, where each entry corresponds to a specific category, including positive samples for each class $k \geq 1$. Algorithm 7 demonstrates the whole process to train a PPCA model.

Algorithm 7 PPCA

```
Input: RAVEs for positive samples $\mathbb{D}_k^{(p)}$, $k = 1, ..., K$, rank $q$
Output: Updated PPCA models $\mathbb{D}_k^{ppca}$, $k = 1, ..., K$
1: for $\boldsymbol{\mu}$,$\mathbf{S}_{xx} \in \mathbb{D}_k^{(p)}$ do
2:     Compute covariance matrix
           $$\mathbf{S} = \mathbf{S}_{xx} - \boldsymbol{\mu}\boldsymbol{\mu}^T \tag{4.14}$$
3:     Decompose $\mathbf{S} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ by SVD
4:     Set $\mathbf{L}$ as the first $q$ columns of $\mathbf{U}$ and $\mathbf{d}$ the corresponding $q \times q$ submatrix of $\mathbf{D}$
5:     Set $\mathbb{D}_k^{(ppca)} = (\boldsymbol{\mu}, \mathbf{L}, \mathbf{d}), k = 1, ..., K$
6: end for
```

4.4 Classifying Anchors with PPCA

In the detection phase, we input the original images into the pre-trained CLIP backbone to generate new feature maps. Using the anchor proposals collected by Grounding-DINO (Liu et al., 2024), we extract the corresponding anchor features from these new maps. Each feature is passed through the previously computed PPCA model $\mathbb{D}_k^{ppca}$ to obtain probability scores for its classification across different categories. The category with the lowest score is assigned to the corresponding anchor box. After processing all features, the final detections are obtained. The following Algorithm 8 illustrates how we classify the anchors with PPCA.

Algorithm 8 Classify anchors of one image with PPCA

```
**Input:** G_DINO proposals $\mathcal{A} = \{a_1, ..., a_n\}$, PPCA Models $(\boldsymbol{\mu}_k, \mathbf{L}_k, \mathbf{d}_k), k = 1, ..., K$, feature map $\mathbf{M} \in \mathbb{R}^{W \times H \times p}$.
**Output:** Predicted classes $\mathbf{c} = (c_i)_{i=1,...,n}$
for $i = 1$ to $n$ do
    Compute $\mathbf{x}_i = ROI(a_i, \mathbf{M})$ using Alg. 5
    Obtain class $c_i = \text{argmin}_k\, r(\mathbf{x}_i, \boldsymbol{\mu}_k, \mathbf{L}_k, \mathbf{d}_k)$ using Eq. (4.12)
end for
```

CHAPTER 5

CLASS INCREMENTAL OBJECT DETECTION EXPERIMENTS

5.1 Datasets

At this stage, we continue to use the refined ILSVRC2015 (Russakovsky et al., 2015) dataset that include 180 different categories from Section 3.1. These 180 classes are divided into groups of 20, and we conduct incremental evaluations by progressively adding each group during the training process.

5.2 Metrics

In our experiments, we employ Mean Average Precision (**mAP**) as a key metric to evaluate the overall performance of the object detector.

To compute **mAP**, we first define the precision-recall curve, obtained by plotting precision against recall at different confidence score thresholds. Precision and recall are defined as:

$$\begin{aligned} \text{Precision} &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \\ \text{Recall} &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \end{aligned} \tag{5.1}$$

A proposal is classified as positive if its Intersection over Union (IoU) with a ground truth bounding box exceeds 0.5. When multiple proposals satisfy this criterion for the same ground truth object, only the one with the highest confidence score is considered a **True Positive (TP)**, while the remaining proposals, despite having an **IoU** greater than 0.5, are counted as **False Positives (FP)**. The definitions are as follows:

- **True Positive (TP)**: A proposal is considered a **TP** if it:
  - Has an **IoU** of at least 0.5 with a ground truth bounding box.
  - Has the highest confidence score among all proposals for that object.
  - Correctly predicts the object's class.
- **False Positive (FP)**: A proposal is considered an **FP** if it:

  - Does not match any ground truth object ($IoU < 0.5$).
  - Predicts the wrong class.
  - Has an $IoU \geq 0.5$ but is not the highest-scoring proposal for that object.

- **False Negative (FN)**: A ground truth object is counted as an **FN** if it is not detected by any proposal with an $IoU \geq 0.5$ and the correct class.

5.2.1 Computation of mAP

For each class, the Average Precision (AP) is computed as the area under the precision-recall curve:

$$AP_k = \int_0^1 P_k(R)dR, \tag{5.2}$$

where $P_k(R)$ represents the precision as a function of recall for class k. In practice, **AP** is typically approximated by discrete summation, averaging precision values at various recall levels.

The final **mAP** is obtained by averaging the **AP** values over all **K** object classes:

$$mAP = \frac{1}{K}\sum_{k=1}^{K} AP_k. \tag{5.3}$$

We report both **mAP(0.5)**, where **AP** is computed at a fixed **IoU** threshold of 0.5, and **mAP(0.5:0.95)**, where **AP** is averaged over multiple **IoU** thresholds from 0.5 to 0.95 in increments of 0.05. The former primarily evaluates the detection capability, while the latter imposes a stricter requirement on localization accuracy.

5.3 Training Details

5.3.1 Mask R-CNN proposals

Mask R-CNN is a traditional two-stage object detector. Due to its excellent performance in segmentation tasks, it also has the ability to generate box proposals that do not include category information. Therefore, we use the boxes generated by this model as proposals for the second-stage classification task. We utilize the `maskrcnn_resnet50_fpn` model provided by PyTorch, which is pretrained on the COCO dataset. Since our refined ILSVRC2015(Russakovsky et al., 2015) dataset in Section 3.1 excludes the portions that overlap with the COCO dataset, our results are not affected by the pretraining model having seen images of certain categories. In the experimental process, we first resize the images so that the shortest side is at least 800 pixels to obtain a sufficient number of proposals, and discard any boxes with scores below 0.2. Finally, we save all the generated boxes for use in the subsequent multi-class classification model.

5.3.2 Grounding-DINO proposals

Grounding DINO is an end-to-end transformer-based detection model designed for detecting novel objects in an open-world setting. In our experiments, we leverage the strong generalization capability of large models to generate reliable anchor proposals. Specifically, we use the pretrained model `groundingdino_swint_ogc.pth` and set the `TEXT_prompt` to "generic" to eliminate category-specific information. Additionally, we set both `box_threshold` and `text_threshold` to 0.1 to obtain a sufficient number of box proposals. Following the methodology provided by `https://github.com/IDEA-Research/GroundingDINO`, we compute and store the box proposals for each image, preparing them for subsequent multi-class classification.

5.3.3 Building PPCA models

For each input image, we calculate the features corresponding to each anchor box after ROI-align based on its ground truth information, and update the PPCA model online using the method described in Section 4.3.

5.3.4 Predicting boxes

During the evaluation phase, we pass the collected proposed boxes through the previously trained PPCA model to compute the corresponding scores as described in Section 4.4. For each image, after computing the scores for all the boxes, we remove all the boxes with PPCA score greater than 25000 to further prune the detections, then we apply Non-Maximum Suppression (NMS) to all the boxes. The negative PPCA scores are used as the NMS scores to remove potentially redundant boxes, with an IoU threshold of 0.5.

In Figure 5.1, we present several examples of the final detection results. The first row illustrates the box proposals generated by G_DINO, while the second row shows the detection results produced by our model after completing incremental training on all 180 categories. These examples demonstrate that our model is capable of handling a wide range of detection scenarios. Specifically, it effectively distinguishes between highly correlated categories, such as guacamole and burrito, accurately detects extremely small objects, such as a ladybug on a tree stump, and successfully differentiates visually similar objects like apples and oranges, even when the apples exhibit variations in color. Furthermore, leveraging the PPCA score threshold enables our model to filter out certain false positives, further improving detection reliability.

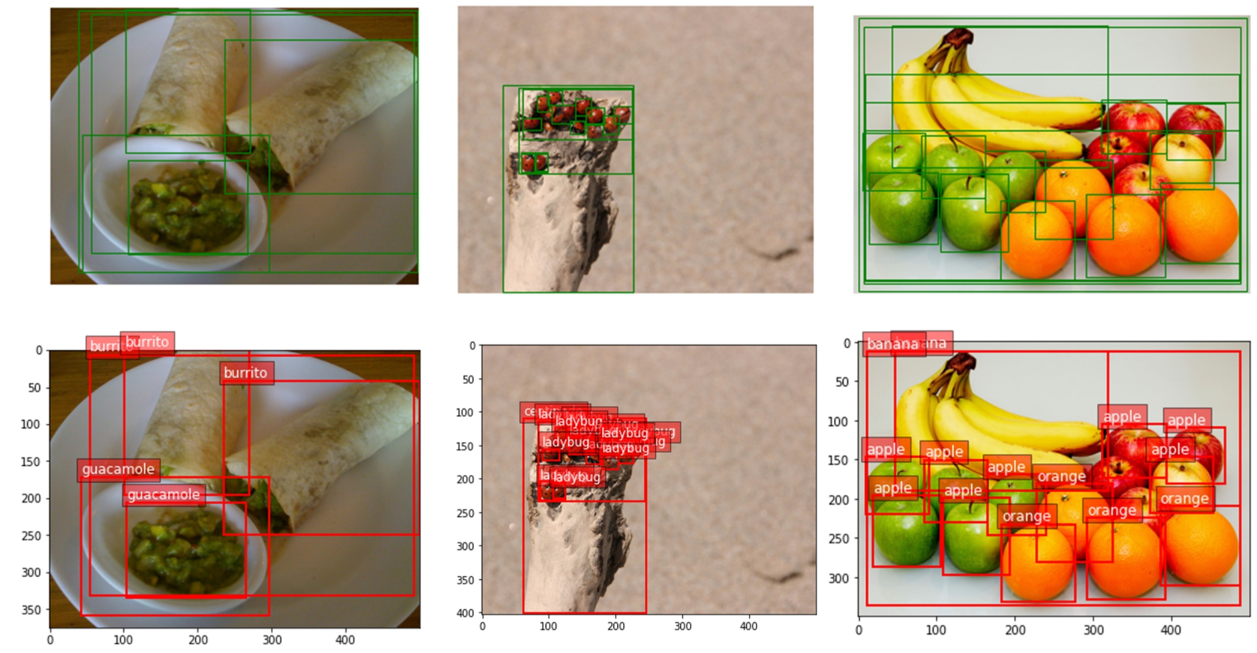

Figure 5.1: First row shows the G_DINO box proposals. Second row shows the detection result of our method after incremental training from all 180 classes.

5.4 Experiments

5.4.1 Catastrophic forgetting

During the experimental phase, we designed a series of evaluations to highlight the issue of catastrophic forgetting in incremental object detection (IOD) and to demonstrate the effectiveness of our proposed model in mitigating this challenge. Specifically, we randomly partitioned the dataset into 9 subsets, each containing 20 distinct object categories. At each evaluation stage, only the newly introduced subset was fed into the model, adhering to the standard IOD protocol. To comprehensively assess model performance, we recorded both the overall mean Average Precision (mAP) and the mAP of the obtained model for each individual subset. The results in Table 5.1 indicate a gradual decline in mAP across the incremental groups, illustrating the ssue of catastrophic forgetting while also showcasing the ability of our model to alleviate this issue significantly.

To provide a more intuitive demonstration of the impact of catastrophic forgetting on model performance, we trained a Faster R-CNN model under an incremental learning setting as a bench-

Table 5.1: Mean Average Precision (mAP@0.5) at different stages and groups of ILSVRC2015 for the incremental PPCA model.

Stage \Group	1	2	3	4	5	6	7	8	9	Overall
1	0.330	-	-	-	-	-	-	-	-	0.330
2	0.323	0.287	-	-	-	-	-	-	-	0.305
3	0.309	0.269	0.352	-	-	-	-	-	-	0.310
4	0.293	0.262	0.346	0.297	-	-	-	-	-	0.299
5	0.281	0.264	0.335	0.278	0.331	-	-	-	-	0.298
6	0.271	0.238	0.330	0.254	0.303	0.276	-	-	-	0.279
7	0.261	0.232	0.318	0.240	0.293	0.257	0.312	-	-	0.273
8	0.259	0.230	0.312	0.231	0.283	0.251	0.309	0.327	-	0.275
9	0.257	0.223	0.306	0.227	0.272	0.248	0.311	0.325	0.289	0.273

mark. Specifically, we adopted the same data partitioning strategy, using the first subset as the base dataset. During training, we froze the first two layers of the model's backbone while optimizing all other layers. The model was initially trained for five epochs with a learning rate of 0.005 until the loss converged. In each incremental step, the model was further trained for two epochs with a learning rate of 0.002 only on the new classes, progressively incorporating all 180 classes.

The final results in Table 5.2 reveal a severe manifestation of catastrophic forgetting. Although the model exhibited a slightly better ability to learn new tasks compared to our proposed approach, it rapidly lost its capability to recognize previously learned classes as the number of incremental tasks increased. Consequently, this led to a significantly low overall mean Average Precision (mAP).

Table 5.2: Mean Average Precision (mAP(0.5)) at different stages and groups of ILSVRC2015 for the Faster-RCNN model.

Stage \Group	1	2	3	4	5	6	7	8	9	Overall
1	0.370	-	-	-	-	-	-	-	-	0.370
2	0.046	0.385	-	-	-	-	-	-	-	0.216
3	0.004	0.000	0.312	-	-	-	-	-	-	0.105
4	0.004	0.000	0.005	0.313	-	-	-	-	-	0.080
5	0.000	0.000	0.000	0.000	0.340	-	-	-	-	0.068
6	0.000	0.000	0.000	0.000	0.010	0.258	-	-	-	0.045
7	0.000	0.000	0.000	0.000	0.000	0.000	0.274	-	-	0.039
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.275	-	0.034
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.331	0.037

5.4.2 Ablation study

To better understand the impact of different components in our model, we conducted an ablation study to isolate and evaluate the contribution of each individual factor.

Number of sampling points in ROI Align. To evaluate the importance of RoI Align and the impact of different numbers of sampling points on the final detection performance, we fixed the number of principal components of the PPCA model to 100 and conducted an ablation study. We randomly selected images of 20 object categories from the whole dataset and analyzed detection results while keeping the PPCA model fixed with 100 principal components. Specifically, we examined how varying the number of sampling points affected key detection metrics, including detection rate, the number of positive samples per image ('n_pos'), the number of predicted bounding boxes per image ('n_boxes'), and the mean average precision (mAP(0.5) and mAP(0.5:0.95)) across the selected categories.

Our results in Table 5.3 show that RoI Align improves detection performance. Without RoI Align, recall was 0.6421, and mAP(0.5) and mAP(0.5:0.95) were 0.307 and 0.201, respectively. Using RoI Align with 5×5 sampling points led to the highest mAP(0.5) of 0.330 and mAP(0.5:0.95) of 0.226, along with improved recall of 0.6627. I ncreasing the sampling points beyond 6×6 did not yield further improvements, with performance declining at 14×14 and 25×25. These results indicate that RoI Align is essential for accurate detection, and 5×5 sampling points provide the best overall performance in our study.

Table 5.3: Ablation study results for different numbers of sampling points.

n_sampling points	**Recall**	**n_pos**	**n_anchor**	**mAP(0.5)**	**mAP(0.5:0.95)**
without ROI Align	0.6421	1.11	5.83	0.307	0.201
4×4	0.6357	1.10	6.77	0.308	0.210
5×5	0.6627	1.14	6.66	0.330	0.226
6×6	0.6618	1.14	6.59	0.321	0.218
7×7	0.6627	1.14	6.52	0.322	0.216
14×14	0.6613	1.15	6.37	0.303	0.199
25×25	0.6578	1.14	6.29	0.303	0.199

Number of principal components. We further analyzed the impact of the number of principal components (n_pc) in the PPCA model on detection performance. Keeping other parameters fixed, we varied npc and evaluated recall, the number of positive samples per image ('n_pos'), the number of predicted bounding boxes per image ('n_boxes'), and mean average precision (mAP) at IoU thresholds 0.5 and 0.5:0.95.

Our results in Table 5.4 show that increasing n_pc generally improves detection performance up to a certain point. With n_pc = 1, recall was the lowest at 0.5728, and mAP(0.5) and mAP(0.5:0.95) were 0.308 and 0.216, respectively. Performance improved steadily as n_pc increased, peaking at n_pc = 100 with recall of 0.6627 and mAP(0.5:0.95) of 0.226. Beyond this, further increasing n_pc resulted in a slight decline in performance. These results indicate that selecting an appropriate n_pc value is crucial, with n_pc = 100 achieving the best detection performance.

Table 5.4: Ablation study results for different numbers of principal components (npc).

n_pc	**Recall**	**n_pos**	**n_anchor**	**mAP(0.5)**	**mAP(0.5:0.95)**
1	0.5728	0.99	4.66	0.308	0.216
5	0.6096	1.05	5.15	0.321	0.221
10	0.6386	1.10	5.49	0.320	0.221
20	0.6465	1.12	5.84	0.319	0.220
50	0.6588	1.14	6.29	0.327	0.222
80	0.6598	1.14	6.56	0.328	0.224
100	0.6627	1.14	6.66	0.330	0.226
120	0.6578	1.13	6.76	0.326	0.224
150	0.6549	1.13	6.89	0.326	0.221
200	0.6504	1.12	7.02	0.322	0.221

Different box proposal method. We compared three different box proposal methods—G_DINO, Selective Search, and Mask R-CNN—to evaluate their impact on detection performance. The results are summarized in Table 5.5 below.

Table 5.5: Comparison of different box proposal methods.

Box Proposal Method	**Recall**	**n_pos**	**n_anchor**	**mAP(0.5)**	**mAP(0.5:0.95)**
Selective Search	0.7507	2.43	783.53	0.099	0.035
Mask R-CNN	0.5398	0.98	7.88	0.230	0.136
G_DINO	0.6627	1.14	6.66	0.330	0.226

The results show that Selective Search achieves the highest recall (0.7507) and the most positive samples per image (2.43), but it generates an excessive number of anchor boxes (783.53), leading to poor mAP, indicating many of its proposals are not useful. G_DINO provides the best overall performance, with the highest mAP values (0.330 for mAP(0.5) and 0.226 for mAP(0.5:0.95)). Mask R-CNN, despite having the lowest recall (0.5398), still generates relatively precise proposals, resulting in a moderate mAP (0.230 for mAP(0.5) and 0.136 for mAP(0.5:0.95)). Overall, G_DINO proves to be the most effective method, offering a good trade-off between recall and precision, while Selective Search generates too many low-quality proposals, and Mask R-CNN performs moderately without excelling in either recall or precision.

5.4.3 Comparison

To further demonstrate our model's effectiveness in mitigating catastrophic forgetting, we compared its performance with several recently proposed models for incremental object detection. However, reproducing their results was challenging due to two primary reasons: many of these models have high computational demands, making it difficult to evaluate them with our available resources, and several projects have not publicly released their source code. Given these constraints, we evaluated our model under the same experimental settings and compared our results with those reported in their respective studies.

Table 5.6 presents the results obtained on the COCO 2017 validation dataset, which consists of 5,000 images spanning 80 object categories. Following prior research, we adopted the 40+10×4 incremental learning strategy, where the first 40 categories were used as the base classes, and 10 new categories were introduced incrementally in four successive stages. The order of these 80 categories was randomly shuffled, and multiple trials were conducted to obtain the mAP(0.5:0.95)/mAP(0.5). The comparison results in Table 5.6 are based on those reported in (Feng et al., 2022).

The results indicate that while our model initially lags behind others in detection performance at the base stage, its resilience to catastrophic forgetting becomes increasingly evident as new categories are introduced. Notably, during the second and third incremental stages, our model surpasses RILODL (Li et al., 2019) and SID (Peng et al., 2021) in terms of mAP. Although its final detection performance is slightly lower than that of ERD (Feng et al., 2022), our model requires significantly fewer parameters, highlighting its efficiency in addressing catastrophic forgetting.

Table 5.6: Performance comparison across different methods with COCO eval2017.(mAP(0.5:0.95)/mAP(0.5))

	1-40	40-50	50-60	60-70	70-80	n_parameters
RILOD	0.457/ 0.663	0.254/ 0.389	0.112/ 0.173	0.105/ 0.156	0.084/ 0.125	$1.56 \cdot 10^7$
SID	0.457/ 0.663	0.346/ 0.521	0.241/ 0.380	0.146/ 0.230	0.126/ 0.233	$3.21 \cdot 10^7$
ERD	0.457/ 0.663	0.364/ 0.539	0.308/ 0.467	0.262/ 0.399	0.207/ 0.318	$2.97 \cdot 10^7$
Ours	0.283/ 0.414	0.253/ 0.375	0.227/ 0.339	0.202/ 0.303	0.180/ 0.271	$2.05 \cdot 10^7$

CHAPTER 6

CONCLUSION

In this project, we introduced a novel framework for class-incremental object detection that avoids training complex neural networks and excessive hyperparameter tuning while minimizing computational memory usage. We leveraged the integral image method to gather anchor features from the feature map and employed RAVE to store relevant information about these anchors. Next, we trained a multi-scale online linear regression classifier with an L2 penalty for initial object-background classification. We introduced three different strategies to select suitable thresholds for each classifier, controlling the trade-off between detection rate and anchor quantity.

This framework significantly reduces computational resource consumption in object detection tasks, enabling real-time detection without retraining the model when new categories are introduced. While the integral image can theoretically select all potentially object-containing anchors, in practice, we had to discard some anchors with certain aspect ratios to streamline the number of proposals, which led to some loss in detection rate.

In the second phase of our experiments, we conducted a comprehensive comparison of various anchor proposal methods to evaluate their effectiveness in class-incremental object detection. By leveraging the strong generalization capability of large-scale pre-trained models, we significantly enhanced the performance of our framework in the incremental setting. Specifically, we integrated the CLIP feature extractor, which is well-suited for multi-class classification, and combined it with RAVE-based storage to train Probabilistic Principal Component Analysis (PPCA) models for classifying anchor proposals. Additionally, we computed the mean Average Precision (mAP) incrementally to assess detection performance across different stages of category expansion.

To further improve the quality of extracted object features, we incorporated the concept of ROI Align to eliminate misalignment issues in the feature extraction process, ensuring more precise localization of object features. Furthermore, we conducted a series of ablation experiments to determine the optimal number of sampling points and the appropriate number of principal components in PPCA, ensuring an effective balance between feature representation and computational efficiency. In addition, we performed a comparative analysis of various box proposal methods to evaluate their impact on detection performance, providing further insights into the most suitable approach for our framework.

Extensive experiments on the COCO 2017 validation dataset demonstrated that while our model exhibited slightly lower detection performance at the base stage compared to existing approaches, it progressively outperformed several state-of-the-art methods in later incremental stages. In particular, our model surpassed RILOD and SID in mAP during the second and third incremental phases, highlighting its superior resistance to catastrophic forgetting. Although its final detection performance was slightly lower than that of ERD, our model achieved these results with significantly fewer parameters, underscoring its efficiency and practicality.

Overall, our findings suggest that the proposed method offers a promising solution for class-incremental object detection, balancing detection accuracy with computational efficiency. Future work will focus on further enhancing feature adaptability and exploring more effective strategies for long-term incremental learning in object detection tasks.

BIBLIOGRAPHY

Cai, Z. and Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Dollár, P. and Zitnick, C. L. (2014). Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570.

Dong, N., Zhang, Y., Ding, M., and Lee, G. H. (2021). Bridging non co-occurrence with unlabeled in-the-wild data for incremental object detection. *Advances in Neural Information Processing Systems*, 34:30492–30503.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59:167–181.

Feng, T., Wang, M., and Yuan, H. (2022). Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9427–9436.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Iandola, F., Darrell, T., and Malik, J. (2015). Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Li, D., Tasci, S., Ghosh, S., Zhu, J., Zhang, J., and Heck, L. (2019). Rilod: Near real-time incremental learning for object detection at the edge. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 113–126.

Lim, J. S., Chen, Z., Baktashmotlagh, M., Chen, Z., Yu, X., Huang, Z., and Luo, Y. (2024). Dipex: Dispersing prompt expansion for class-agnostic object detection. *arXiv preprint arXiv:2406.14924*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., et al. (2024). Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer.

Maaz, M., Rasheed, H., Khan, S., Khan, F. S., Anwer, R. M., and Yang, M.-H. (2022). Class-agnostic object detection with multi-modal transformer. In *European conference on computer vision*, pages 512–531. Springer.

McLachlan, G. J. (1999). Mahalanobis distance. *Resonance*, 4(6):20–26.

Peng, C., Zhao, K., Maksoud, S., Li, M., and Lovell, B. C. (2021). Sid: Incremental learning for anchor-free object detection via selective and inter-related distillation. *Computer vision and image understanding*, 210:103229.

Pont-Tuset, J., Arbelaez, P., Barron, J. T., Marques, F., and Malik, J. (2016). Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):128–140.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Shmelkov, K., Schmid, C., and Alahari, K. (2017). Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE international conference on computer vision*, pages 3400–3409.

Singer, S. and Nelder, J. (2009). Nelder-mead algorithm. *Scholarpedia*, 4(7):2928.

Sun, L., Wang, M., Zhu, S., and Barbu, A. (2024). A novel framework for online supervised learning with feature selection. *Journal of Nonparametric Statistics*, pages 1–27.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104:154–171.

Wang, B. and Barbu, A. (2022). Scalable learning with incremental probabilistic pca. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 5615–5622. IEEE.

Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L. M., and Shum, H.-Y. (2022). Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 391–405. Springer.

BIOGRAPHICAL SKETCH

Siquan Zhu received a Bachelor's degree in Public Finance from Jiangxi University of Finance and Economics in 2018. In 2020, Siquan earned a Master's degree in Statistics from Florida State University and continued doctoral studies in Statistics at the same institution. Under the supervision of Professor Adrian Barbu, Siquan's research focuses on incremental object detection, aiming to develop efficient methodologies for object detection in dynamic environments.