FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

From Classification to Detection: Adapting Semi-Supervised Frameworks to Object Detection

Alexandre Ferreira Nunes

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Ricardo Cruz

July 18, 2025

From Classification to Detection: Adapting Semi-Supervised Frameworks to Object Detection

Alexandre Ferreira Nunes

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: José Maria Corte Real da Costa Pereira
External Examiner: João Tiago Ribeiro Pinto
Supervisor: Ricardo Pereira de Magalhães Cruz

July 18, 2025

Resumo

A tarefa de deteção de objetos é fundamental na área de visão por computador, exigindo simultaneamente a classificação e a localização precisa de objetos em imagens. Embora os modelos de Deep Learning tenham alcançado resultados notáveis neste domínio, o seu desempenho está fortemente condicionado sobre a disponibilidade de grandes volumes de dados anotados, cuja obtenção é trabalhosa e dispendiosa. Semi-supervised Learning (SSL) surge como uma alternativa promissora, aproveitando dados não anotados para reduzir a dependência de anotações manuais. Apesar do progresso significativo das metodologias de SSL na classificação de imagens, a aplicação direta das mesmas metodologias à deteção de objetos permanece pouco explorada, uma vez que requer adaptações não triviais para lidar com as diferenças estruturais e as complexidades adicionais da tarefa de deteção.

Esta tese investiga a viabilidade e eficácia da adaptação de algoritmos SSL bem sucedidos na classificação de imagens, como o MixMatch, ReMixMatch e FixMatch, para o contexto de deteção de objetos. Um aspeto importante deste trabalho é a modificação interna de um modelo de deteção de objetos, especificamente o FCOS, para incorporar um novo conceito chamado Loss Masks, que permite o modelo ignorar regiões das imagens, e consequentemente ter um treino mais controlado, reduzindo o erro introduzido por trabalhar com imagens sem anotações. A investigação foca na identificação de princípios transferíveis e na resoluções dos desafios que surgem durante o processo de adaptação, na avaliação empírica do desempenho, e na proposta de novas metodologias conceptuais. As experiências realizadas nos conjuntos de dados KITTI e COCO demonstraram que todos os métodos foram adaptados com sucesso à tarefa de deteção de objetos, superando os modelos treinados apenas com os dados anotados. Entre os métodos, o ReMixMatch obteve os melhores resultados, com melhorias de 9% mAP no KITTI e 5% mAP no COCO.

Adicionalmente, os métodos adaptados foram aplicados a um cenário real de condução autónoma utilizando dados LiDAR. Ao projetar 3D points clouds em imagens 2D range-view, os mesmos métodos foram avaliados nesta modalidade, produzindo novamente resultados positivos em todas as metolodogias. A análise comparativa entre a deteção baseada em imagens RGB e a baseada em LiDAR forneceu conclusões interessantes e destacou direções promissoras para futuras investigações.

De um modo geral, este estudo confirma a viabilidade da adaptação de metodologias de classificação à deteção de objetos e destaca a sua eficácia no aproveitamento de dados não anotados, resultando em treinos de modelos de deteção de objetos mais escaláveis e económicos.

Palavras-chave: semi-supervised learning, deteção de objetos, visão por computador, deep learning

Abstract

Object detection is a primary task in computer vision that requires both accurate classification and precise localization of objects within images. While deep learning models have achieved remarkable performance in this domain, their success heavily relies on large-scale annotated datasets, which creation is both labor-intensive and costly. Semi-supervised learning (SSL) presents a promising alternative by leveraging unlabeled data to reduce dependency on manual annotations. Despite the significant progress of SSL techniques in image classification, the direct application of the same frameworks to object detection remains underexplored, as it requires non-trivial adaptations to handle the structural differences and added complexities of the detection task.

This thesis investigates the feasibility and effectiveness of adapting successful SSL algorithms originally developed for image classification, such as MixMatch, ReMixMatch, and FixMatch, to object detection tasks. An important aspect of this adaptation was the internal modification of the FCOS object detection model to incorporate Loss Masks, which allows the model to ignore regions of unlabeled images, enabling finer control over label noise during training. The research focuses on identifying transferable principles and addressing the challenges that arise during the adaptation process, evaluating empirical performance, and proposing new conceptual frameworks. The experiments performed on the KITTI and COCO datasets demonstrated that all the SSL frameworks were successfully adapted to object detection, outperforming the fully supervised baselines. Among them, ReMixMatch achieved the best results, with improvements of 9% mAP on KITTI and 5% mAP on COCO.

Furthermore, the adapted frameworks were applied to a real-world autonomous driving scenario using LiDAR data. By projecting 3D point clouds into 2D range-view images, the same SSL methods were evaluated in this alternative modality, again yielding positive results across all frameworks. Comparative analysis between RGB-based and LiDAR-based detection provided additional insights and highlighted promising directions for future research.

Overall, this study confirms the viability of adapting classification frameworks to object detection and highlights their effectiveness in leveraging unlabeled data to reduce annotation costs, leading to more scalable and cost-efficient object detection pipelines.

Keywords: semi-supervised learning, object detection, computer vision, deep learning

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice. The specific Sustainable Development Goals mentioned have the following names:

SDG 3 Ensure healthy lives and promote well-being for all at all ages.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

SDG 11 Make cities and human settlements inclusive, safe, resilient, and sustainable.

SGD	Target	Contribution	Performance Indicators and Metrics
3	3.8	Semi-supervised object detection supports medical imaging, such as X-ray diagnostics, by lowering the barrier to deploying AI in hospitals where labeled data is scarce, thus helping expand access to quality health services.	Number of AI-assisted diagnoses.
9	9.4	Semi-supervised learning promotes efficient and sustainable data use by reducing reliance on fully labeled datasets, enabling scalable AI development with large volumes of unlabeled data.	Number of object detection models trained with partially labeled or unlabeled datasets.
	9.5	Adapting SSL methods to object detection advances research and enables high-performance models without large private datasets, broadening AI access across the world.	Performance gap between SSL and fully supervised methods.
11	11.2	SSL in object detection enables safer and more cost-effective autonomous transport by leveraging unlabeled urban data for model training.	Number of autonomous vehicles on the road.

Acknowledgements

This thesis is the outcome of a journey filled with ups and downs, with moments of doubt, but also moments of discovery and growth. It was only possible not just through determination, but through the support and encouragement of those around me.

I am deeply grateful to my supervisor, Prof. Ricardo Cruz, for the consistent support, thoughtful feedback, and availability throughout the entire process.

To my family, parents, sisters, and brother, thank you for your patience, love, and faith in me. Your support has been the anchor that kept me moving forward and aiming further.

And to my friends, thank you for your encouragement, late-night conversations, and the many shared moments that brought me relief and joy, even in the most stressful times.

Alexandre Nunes

This work was supported by Portuguese National Funds through AICEP, E. P. E., and the Innovation and Digital Transition Program (COMPETE2030), under Project ATLAS – Trusted Autonomous Navigation, with funding reference 01/RPA/2022-C679908640-00009887.

"The secret of getting ahead is getting started."

Mark Twain

Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network
AP	Average Precision
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
EMA	Exponential Moving Average
FFN	Feed Forward Network
FCOS	Fully Convolutional One-Stage
FPN	Feature Pyramid Network
GIoU	Generalized Intersection over Union
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
mAP	Mean Average Precision
MSE	Mean Squared Error
NMS	Non-Maximum Suppression
PR	Precision-Recall
ReLU	Rectified Linear Units
RPN	Region Proposal Network
SSL	Semi-supervised Learning
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Context

Object detection is a fundamental computer vision task that aims to find and classify objects in an image. This task is a central element in various applications, including video surveillance, robotics, autonomous driving, and medical imaging. Unlike image classification, which involves one label prediction per image, object detection must also handle spatial localization by predicting bounding boxes and associating them with appropriate class labels.

Deep learning [1] models have become the dominant approach on object detection tasks, exhibiting excellent performance and the ability to handle complex scenarios and backgrounds. However, they typically require large-scale annotated data to perform well. Preparing such datasets requires significant manual effort, including drawing bounding boxes and annotating each object with a class label, which is costly and time-consuming [2].

Semi-supervised learning (SSL) offers an alternative by leveraging labeled and unlabeled data to avoid labeling the whole dataset. A common SSL approach is pseudo-labeling, where a model trained only with the labeled subset generates predicted labels on the unlabeled data. Only predictions with high confidence, based on a filtering threshold, are treated as pseudo-labels, converting unlabeled data into labeled data. This iterative process allows the model to continue learning while minimizing noise and improving performance [3].

1.2 Motivation

The growing need for minimal dependency on labeled data and the vast availability of unlabeled data present immense potential for semi-supervised learning. There are many real-world applications in which vast quantities of visual unlabeled data are collected continuously. For example, an autonomous vehicle fleet constantly collects video frames and sensor data as it moves through diverse environments. Video surveillance networks, including traffic and public security surveillance systems, produce vast amounts of content daily. In medical care, hospitals gather enormous

reserves of medical images, such as X-rays, where the majority remain unlabeled due to the time and cost associated with having them labeled by an expert.

SSL has already performed impressively in image classification. Methods such as MixMatch [4] and FixMatch [5] have shown comparable performance to fully supervised models on only partial access to labeled data. In recent years, the same applied techniques and principles have been deeply explored in the literature and shown promising results in more complex tasks such as object detection [6].

1.3 Problem

Despite the significant progress made in SSL for image classification, the direct application of the same frameworks to object detection remains underexplored, as object detection introduces a new set of challenges that are inherently different from the single-label output of classification tasks.

Most state-of-the-art SSL techniques for detection have been specifically crafted for the detection task or include model architecture-specific variations [7–9], often incorporating limited generalizability and reusability across architectures. While many methods are inspired by FixMatch [5], a broad class of classification SSL methods, such as MixMatch [4] and ReMixMatch [10], employ distinct principles and techniques that could offer valuable insights for detection.

This thesis investigates the problem of directly transferring semi-supervised classification algorithms to object detection. The main goal is to investigate which of the successful classification-based SSL algorithms' principles and components can be successfully transferred to the object detection task, and what are the limitations and challenges that occur throughout the process. By focusing on the adaptation process itself, this thesis contributes not only with empirical comparisons but also with new conceptual frameworks.

1.4 Research Questions

The following research questions will guide this investigation:

- Can semi-supervised learning algorithms developed for image classification be effectively adapted for object detection tasks?
- What challenges arise when adapting classification-based SSL methods to object detection, and how can they be addressed?

Hypothesis Semi-supervised learning algorithms originally designed for classification, when properly adapted, can significantly reduce the amount of labeled data required for object detection tasks while maintaining reasonable performance.

1.5 Document Structure

The document is organized as follows:

Chapter 2 provides all the essential background knowledge for the development of the thesis, including an overview of deep learning, convolutional neural networks, object detection paradigms, the FCOS detection framework, learning paradigms, evaluation metrics, and data augmentations.

Chapter 3 presents a review of the state of the art in semi-supervised learning for image classification. It categorizes and explains key methods such as consistency regularization, self-training, hybrid approaches, generative models, and graph-based techniques.

Chapter 4 presents a review of the state of the art in semi-supervised learning for object detection. It discusses how the object detection task has been addressed in the literature and how classification-based SSL ideas have been adapted to this domain.

Chapter 5 outlines the methodology adopted in this thesis. It describes the datasets, the warm-up stage, the adaptation of various classification SSL methods, including MixMatch, ReMixMatch, and FixMatch, to the object detection context, and concludes with the approach to explore 2D LiDAR projections.

Chapter 6 presents experimental results and analysis. It includes ablation studies and a discussion of the empirical findings about the research questions and hypotheses.

Chapter 7 concludes the thesis with an overview of the main findings and contributions, finishing with potential directions for future research.

Chapter 2

Background Knowledge

2.1 Deep Learning

In recent years, deep learning has become the dominant approach for solving complex problems in computer vision. These models use large, multilayered neural networks to learn patterns directly from raw data, outperforming traditional object detection methods that used sliding windows and hand-crafted features. While these methods were somewhat effective, they often struggled with accuracy and robustness in the presence of diverse objects and complex backgrounds [11].

Convolutional Neural Networks (CNNs) [12] are a subset of deep neural networks, designed to extract spatial hierarchies of features from images efficiently. Instead of connecting every neuron to every pixel in the input, each layer uses filters, also called kernels, that scan over the image and process regions of the image at a time. Each filter slides across the width and height of the input image, detecting local patterns such as edges, corners, or textures. This design drastically reduces the number of parameters and focuses the network on spatially local relationships that are meaningful in computer vision. CNNs typically consist of three main types of layers:

- Convolutional layers, which apply filters to extract feature maps from the input. Often interleaved with activation functions such as the Rectified Linear Unit (ReLU), which introduces non-linearity and allows the network to learn more complex patterns.
- Pooling layers, which reduce the spatial dimensions of feature maps, helping to control overfitting and improve computational efficiency.
- Fully connected layers, similar to the traditional artificial neural networks (ANN) layers. However, rather than receiving raw data as input, they process abstract, high-level patterns learned through the preceding convolution and pooling layers.

CNNs progressively transform input images into increasingly abstract feature representations through this layered processing pipeline. This pipeline makes them especially powerful for visual tasks such as image classification, object detection, and semantic segmentation, where they have become a foundational tool in computer vision [13].

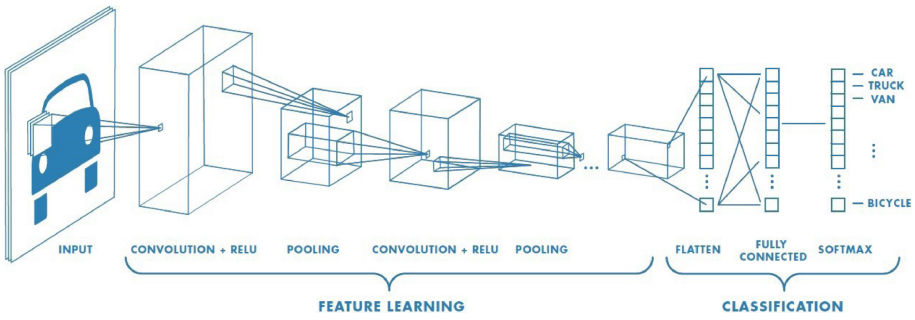


Figure 2.1: Diagram with an example of a CNN architecture. Convolutional and pooling layers extract features and detect patterns from input images. These are followed by fully connected layers that perform specific tasks, e.g., classification, semantic segmentation, or object detection.

2.2 Convolutional Neural Networks for Object Detection

Object detectors typically rely on an extended modular CNN architecture composed of three key components: the backbone, the neck, and the detection head.

Backbone The backbone is responsible for extracting high-level features from input images. It processes raw input images and outputs feature maps that encode high-level representations. Common backbones include CNNs like VGG [14], ResNet [15], and MobileNet [16].

Neck The neck is typically placed between the backbone and the detection head, primarily enhancing the model's ability to detect objects at different scales. It aggregates multi-scale feature maps, enabling effective spatial and semantic information fusion across varying resolutions. A popular neck architecture is the Feature Pyramid Network (FPN) [17], which builds a top-down pyramid with lateral connections to merge features from different backbone levels.

Head Performs the final task-specific prediction. In the case of object detection, this includes classifying object categories and regressing bounding box coordinates. The design of the detection head can significantly vary depending on whether it follows a one-stage or two-stage detection paradigm.

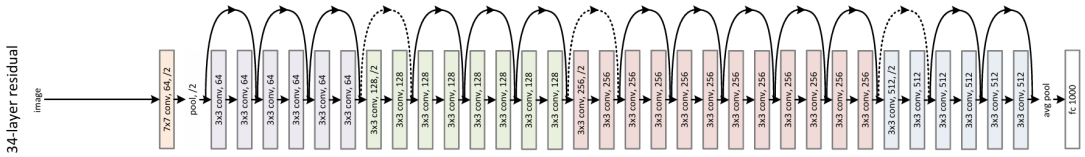


Figure 2.2: Diagram of the ResNet architecture, a widely used CNN backbone known for using residual connections to ease the training of deep networks.

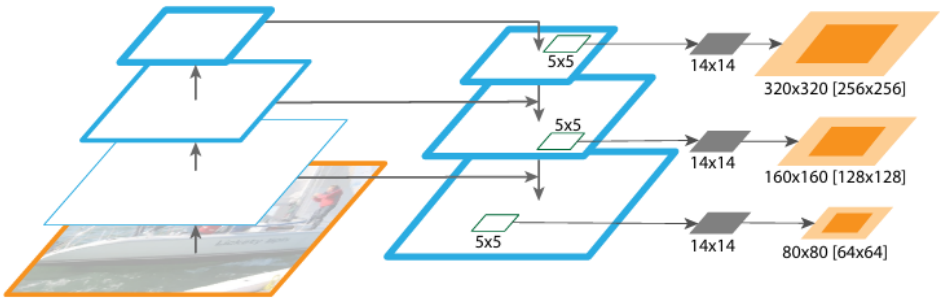


Figure 2.3: Diagram of the Feature Pyramid Network (FPN) architecture, enhancing object detection at multiple scales by combining features from different layers of the backbone through a top-down pathway with lateral connections.

2.3 Object Detection Paradigms

Object detection is a computer vision task that localizes and classifies objects in images. It combines object localization, which identifies the location of objects using bounding boxes, with object classification, which categorizes them. Unlike image classification, which only determines the presence of an object, object detection identifies and categorizes multiple objects within an image, providing both their location and type as seen in Figure 2.4. Three well-known paradigms will be briefly described below.

Two-stage Object Detection exemplified by Faster R-CNN [18], involves a two-step process that combines region proposal and object detection. In the first stage, a Region Proposal Network (RPN), which is a fully convolutional network, generates region proposals, which are candidate bounding boxes that potentially contain objects. In the second stage, a Fast R-CNN detector evaluates each region proposal, performing object classification and bounding box refinements.

Each region proposal generated by the RPN is followed by an "objectness" score, which indicates the likelihood of the region containing an object. Each region is then ranked based on the confidence of its classification. R-CNN filters out regions with a certain Intersection over Union (IoU) overlap with a higher-scoring selected region, retaining only the top-ranking.

One-stage Object Detection as seen in YOLO [19], simplifies the detection process by treating object localization and classification as a single regression task. In this approach, a single neural network processes the entire image in one forward pass, directly predicting both bounding boxes and class probabilities. This design leads to a significant speed advantage, as the model does not rely on an intermediate region proposal step, as in two-stage methods.

However, this speed comes at a cost: YOLO is generally less accurate than two-stage detectors, particularly in terms of localization. As noted by the authors, YOLO tends to produce more localization errors but is less prone to generating false positives, especially in background regions.

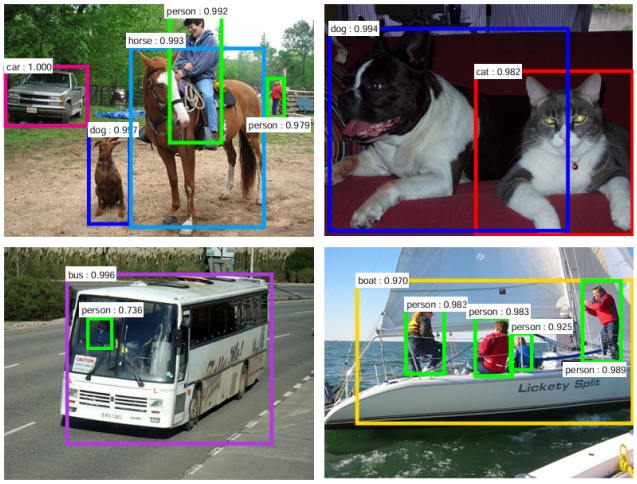


Figure 2.4: Outputs from an object detection model, displaying predicted bounding boxes and corresponding class labels.

The architecture divides the input image into an $S \times S$ grid, where S is a tunable hyperparameter with a trade-off of localization accuracy and computational efficiency. If the center of an object falls within a grid cell, that cell becomes responsible for detecting the object. Each grid cell predicts B bounding boxes, each with a confidence score that reflects both the probability of an object being present and the accuracy of the bounding box. During training, this confidence score is supervised to match the IoU between the predicted and ground truth boxes. At inference time, the model outputs this confidence score as an estimate without access to ground truth.

Set-based Object Detection such as DETR [20], introduces a new approach to object detection by modeling it as a direct set prediction problem, unlike traditional methods, which introduce some bias by relying on a predefined grid or anchors, set-based models predict the entire set of objects in an image simultaneously. One-stage algorithms often predict multiple bounding boxes for the same object when the object spans multiple grid cells. To address this, a post-processing step is typically used to eliminate redundant predictions. DETR, however, removes the need for such a step by employing bipartite matching during training. Using the Hungarian algorithm [21], it matches ground-truth bounding boxes to predicted ones, encouraging accurate predictions while penalizing errors. This approach reduces the likelihood of repeated predictions during inference.

The architecture of DETR consists of three main components: a CNN backbone that extracts a compact feature representation, a transformer encoder-decoder [22] that models interactions between objects using self-attention, and a feed-forward network (FFN) that produces the final predictions. Using self-attention allows DETR to effectively handle set prediction constraints, such as eliminating duplicate predictions, by modeling all pairwise interactions between elements. DETR predicts all objects simultaneously and is trained end-to-end.

2.4 Fully Convolutional One-Stage (FCOS)

FCOS [23] is an object detection model that builds upon the success of fully convolutional architectures initially popularized in semantic segmentation. The method follows a one-stage architecture and reformulates object detection as a per-pixel prediction problem, allowing the network to predict object locations directly from feature maps without predefined anchors. This approach reduces complexity and has demonstrated competitive performance compared to anchor-based methods. The method leverages an FPN to construct a multi-scale representation of the input image. The FPN outputs five different latent spaces, which are responsible for detecting objects at various scales, where each location matches a region of the original image.

The FCOS head consists of two branches: a classification head and a regression head, where each location of the feature map is treated as a potential object center. For training, the ground-truth boxes are assigned to all feature map locations inside the box area, and for each of these locations, the network predicts:

- Location object class
- The offsets from the location to the top, bottom, left, and right sides of the bounding box
- Centerness score, which indicates how close the location is to the center of the object

Centerness The centerness score is introduced due to the observation of low-quality predictions far from the object's center. To address that, FCOS introduced the centerness concept computed as:

$$\text{centerness} = \sqrt{\frac{\min(l,r)}{\max(l,r)} \cdot \frac{\min(t,b)}{\max(t,b)}} \tag{2.1}$$

where l, r, t, b are the distances from the pixel to the four sides of the ground-truth bounding box.

Classification Head Outputs a tensor of logits for each spatial location and for each class. The logits correspond to raw scores that require a final sigmoid to map to the range of $[0,1]$. However, at inference, the centerness is also used to calculate the confidence score, where the following formula can calculate the final value:

$$score = \sqrt{\sigma(logit) \cdot \sigma(centerness)} \tag{2.2}$$

where $\sigma(\cdot)$ denotes the sigmoid function.

Regression Head Predicts the spatial localization of the objects. Each location in the feature map outputs not only a 4D vector representing the distances to the left, top, right, and bottom sides of the bounding box but also the centerness score mentioned previously.

Loss The FCOS loss function is defined per feature map location and combines three components: classification, bounding box regression, and centerness. Let N be the number of foreground locations in the feature maps, the total loss is computed as:

$$\ell = \frac{1}{N}\left(\ell_{\text{cls}} + \ell_{\text{reg}} + \ell_{\text{ctr}}\right) \tag{2.3}$$

Classification Loss (ℓ_{cls}) Uses the FocalLoss [24], which was introduced to address a significant issue in one-stage object detectors of extreme imbalance between foreground and background classes during training. This loss function modifies the standard cross-entropy loss by down-weighting the contribution of easy negatives, enabling the model to focus more on the complex, incorrectly classified examples.

Regression Loss (ℓ_{reg}) Regression loss is computed through the Generalized Intersection over Union (GIoU) [25]. Contrary to IoU, which is only defined when the ground-truth and prediction boxes overlap, GIoU provides gradients even if the boxes do not overlap.

Centerness Loss (ℓ_{ctr}) The centerness loss is computed using Binary Cross Entropy (BCE) between the predicted centerness logits and the ground-truth centerness, which is calculated by the ground-truth boxes' coordinates. Unlike standard cross-entropy, which is used when choosing between multiple classes, BCE is a variant designed for cases where the output is a single number representing a probability, such as centerness.

2.5 Learning Paradigms

Although deep learning models can learn directly from raw data, their performance often depends on the availability of large labeled datasets. This dependency has driven research towards alternative approaches, leading to the exploration of different learning paradigms.

Supervised learning remains the most commonly used paradigm, where models are trained on datasets in which each input is paired with a corresponding ground-truth label. This setting enables direct error correction during training and typically yields high performance when sufficient labeled data is available.

Unsupervised learning involves discovering underlying patterns, structures, or representations in data without access to any labels. Common objectives include clustering, dimensionality reduction, and generative modeling. While powerful in specific contexts, e.g, Natural Language

Processing (NLP), purely unsupervised approaches often struggle with high-level semantic understanding.

Semi-supervised learning bridges these two extremes by using a small amount of labeled data together with a much larger pool of unlabeled examples. SSL methods are particularly appealing in domains like computer vision, where unlabeled images are plentiful but precise annotations, e.g., bounding boxes, are costly to obtain.

Self-supervision learning is a paradigm in which the model learns from unlabeled data by solving auxiliary tasks that generate pseudo-labels from the data itself. These tasks are designed to encourage the model to learn meaningful features that can later be transferred to downstream tasks with minimal fine-tuning. An example task would be to train the model to predict what rotation was applied to an image [26].

2.6 Metrics

This section presents the metrics used in this thesis to evaluate the performance of the object detection models, which require not only correctly identifying objects but also precisely delineating their locations through bounding boxes.

Intersection over Union (IoU) Intersection over Union is a fundamental metric used to evaluate how well the predicted bounding box overlaps with the ground truth bounding box. An illustration is shown in Figure 2.5, and the formula is defined as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.4}$$

The IoU value ranges between 0 and 1, where 1 indicates perfect congruence between the predicted and actual bounding boxes. In practice, a threshold, commonly set at 0.5, determines whether a detection is considered a true positive. This thresholding is also helpful in post-processing techniques such as Non-Maximum Suppression (NMS), which removes redundant bounding boxes that refer to the same object, retaining only the box with the highest confidence score.

Precision and Recall Precision and recall are metrics derived from the concepts of true positives (TP), false positives (FP), and false negatives (FN). Precision measures the proportion of correct positive predictions among all positive predictions made, while recall assesses the proportion of actual positives correctly identified by the model. These metrics are defined as:

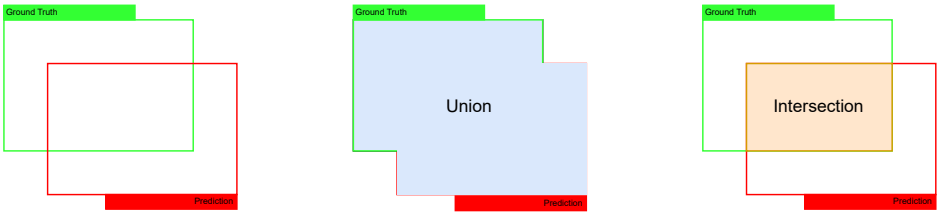


Figure 2.5: Illustration of the Intersection over Union (IoU) metric. A higher IoU means a larger intersection between the predicted and ground truth regions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.5}$$

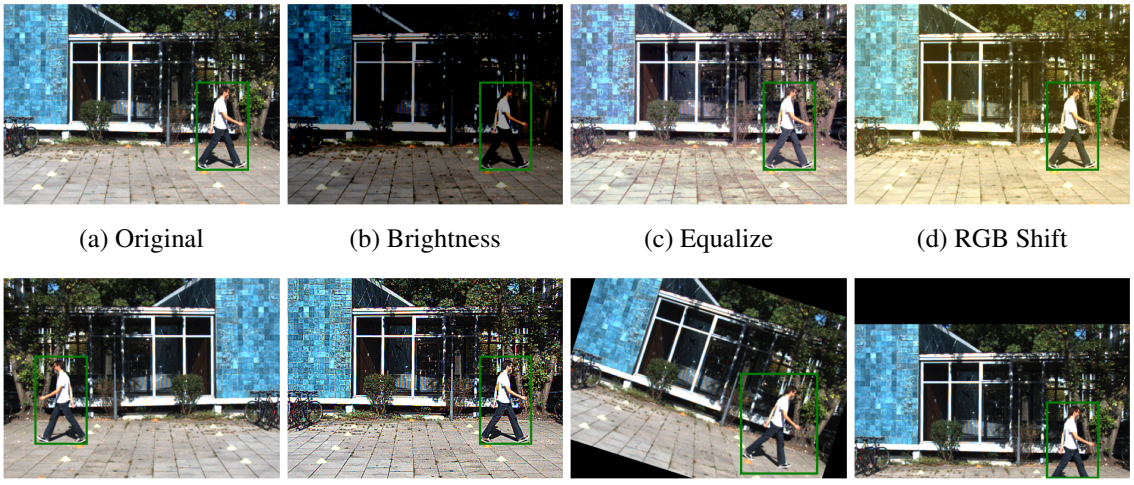
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.6}$$

Average Precision (AP) Summarizes the relationship between precision and recall by calculating the area under the precision-recall (PR) curve. The PR curve is a plot of precision against recall at various confidence score thresholds. As the threshold varies, the resulting curve illustrates the model trade-off between precision and recall. The area under this curve is the AP, which effectively quantifies the model's overall detection quality for a specific class and IoU threshold.

Mean Average Precision (mAP) represents the average of AP values computed across all object classes. It provides a single scalar value that reflects both the classification and localization performance of the model. For a dataset with C classes, mAP is calculated as:

$$\text{mAP} = \frac{1}{C} \sum_{i=1}^{C} \text{AP}_i \tag{2.7}$$

COCO AP@[0.50:0.95] The COCO benchmark [27] employs a stricter evaluation protocol by calculating AP across multiple true positive IoU thresholds, from 0.50 to 0.95 in increments of 0.05. This metric, denoted as AP@[0.50:0.95], or mAP@[0.50:0.95] when averaged over classes, evaluates localization robustness by penalizing models that are only capable of coarse localization and rewarding those that consistently perform well across a range of localization strictness levels. For example, a model achieving high AP at IoU=0.50 but low AP at IoU=0.95 can reliably identify objects with approximate localization but struggles to output tightly fitted bounding boxes required for precise detection. By averaging AP over ten distinct thresholds, AP@[0.50:0.95] provides a complete assessment of a model's ability to balance accuracy and spatial precision, making it a standard for state-of-the-art object detection evaluation and also the metric used in this thesis.



(a) Original (b) Brightness (c) Equalize (d) RGB Shift

(e) Horizontal Flip (f) Sharpen (g) Rotate (h) Translate Y

Figure 2.6: Illustration of some augmentation examples. In spatial augmentations, both the image and the bounding boxes are affected.

2.7 Data Augmentation

Data augmentation is a standard used technique in computer vision that is used to increase dataset diversity and improve generalization by artificially creating new training samples. It helps prevent overfitting and encourages the model to be robust to input variations. A set of augmentation examples is illustrated in Figure 2.6.

2.7.1 Types of augmentations

- **Weak Augmentations** applies minimal changes that preserve most of the original image structure. They help add some diversity to the training data relatively easily. These augmentations are especially useful for generating pseudo-labels, as they keep the image close to its original form and help maintain consistency.
- **Strong Augmentations** involves more aggressive and diverse transformations that can substantially change the appearance of the image, although preserving its semantic content. Typically used to enhance the model's generalization by forcing it to learn high-level patterns.

In the context of object detection, augmentations must be applied not only to the images but also to the bounding boxes. Most augmentations can be trivially applied, e.g., color transformations do not modify anything in the ground-truth labels. However, some more complex augmentation algorithms may require additional complexity, e.g,. rotations and translations. For the present work, all augmentations applied used the framework Albumentations [28] that already offers a vast list of augmentations with support to both images and bounding boxes.

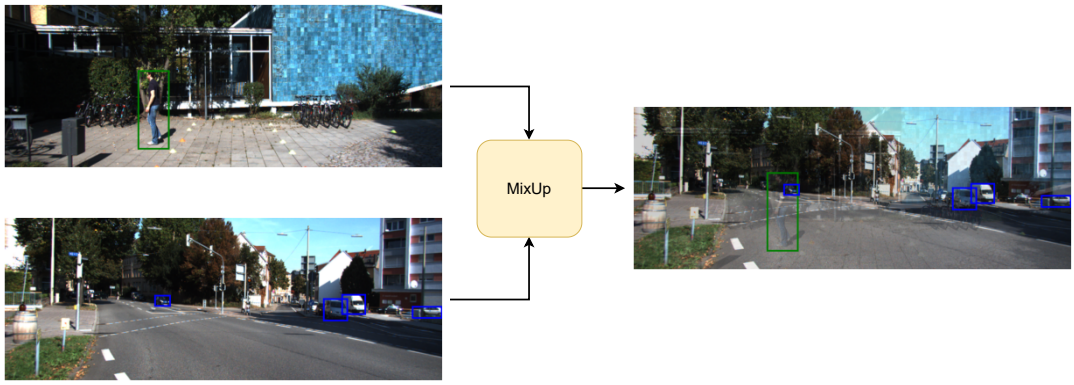


Figure 2.7: Diagram with a MixUp augmentation example. The images are interpolated linearly by a factor λ, and the bounding boxes are merged.

2.7.2 MixUp

MixUp [29] is a strong augmentation that creates new artificial samples by blending two images and their corresponding labels through linear interpolation, as illustrated in Figure 2.7. Originally designed for classification tasks, MixUp can be adapted for object detection, where the most trivial way to implement it is to simply merge the different images' bounding boxes into a single set. However, this naive approach overlooks the fact that objects from the less dominant image are harder for the model to detect. To address this, it is a good approach to assign weights to the bounding boxes proportional to λ and $1-\lambda$. Given two input images x_i and x_j with their corresponding labels y_i and y_j, MixUp generates a new sample $(\tilde{x}, \tilde{y})$ as follows:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j \tag{2.8}$$

where $\lambda \in [0,1]$ is sampled from a Beta distribution, $\text{Beta}(\alpha, \alpha)$, for a chosen hyperparameter $\alpha > 0$. Overall, the main goal of MixUp is to encourage the model to behave linearly between examples.

2.7.3 RandAugment

RandAugment [30] is another strong data augmentation strategy that enhances model generalization by applying a sequence of randomly selected transformations from a fixed set of augmentation operations. Each transformation is chosen uniformly at random and applied with the same intensity, controlled by a magnitude parameter. The only two hyperparameters are the number of different transformations N applied per image and the magnitude M, which controls the strength of each transformation. The augmentation pool typically includes transformations such as rotation, translation, shearing, color adjustments, and more. These operations are applied sequentially to each input image, introducing diverse variations that make the model more robust to real-world distortions and perturbations.

2.8 LiDAR

LiDAR technology is a common sensor technology used in autonomous vehicles that is able to operate independently of ambient light, making it more resilient to atmospheric variations and particularly effective in low-light or nighttime conditions. In contrast to RGB cameras, which are passive sensors, LiDAR sensors are active sensors that emit their own signals and estimate object distances in three dimensions by analyzing the phase delay of the returned signals. However, it also has certain limitations, the resolution of LiDAR is inherently restricted, and the data it captures is represented as a sparse set of 3D points. This sparsity makes it challenging to capture some details or small objects when compared to high-resolution cameras. Furthermore, as an active sensor, LiDAR may face difficulties in accurately detecting specific materials, such as transparent surfaces or those with low reflectivity. In practice, LiDAR sensors are frequently combined with cameras to complement one another's strengths. Cameras excel in object recognition but are poor at distance estimation, whereas LiDAR provides precise distance measurements but struggles with detailed object recognition.

Approaches to processing LiDAR data generally fall into three domains: treating points as raw 3D data, converting points into voxels, or projecting the point cloud onto a 2D plane as images. The first two methods typically incur significant computational overheads, which demand substantial resources and are often unsuitable for real-time applications.

Projection-based methods leverage the maturity of 2D convolutional neural networks (CNNs) by transforming the 3D point cloud into 2D images. After running the model, the 2D results can be projected back onto the corresponding points in the 3D cloud. The quality of this projection directly influences the final results. The most common techniques are:

Bird's-Eye View (BEV) This approach simplifies the data by disregarding the height dimension, effectively creating a top-down view. BEV is advantageous for its straightforward integration with map data, e.g., road layout, and consistent object sizes, regardless of distance. However, this method sacrifices height-related information, which may reduce accuracy, e.g., when detecting pedestrians near other objects.

Range View This method provides a frustum-like 2D perspective, closely aligning with how LiDAR naturally captures its data. Range view offers a compact and intuitive representation of the scene, making it particularly effective for object detection and segmentation tasks.

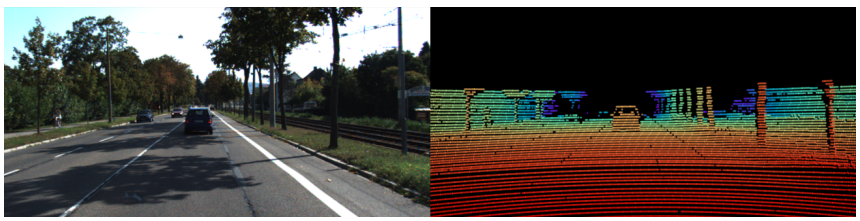


Figure 2.8: Illustration of LiDAR Range View Projection.

Chapter 3

Review on Semi-Supervision for Classification

3.1 Introduction

Semi-supervised learning techniques have achieved remarkable success in classification tasks, demonstrating their ability to leverage both labeled and unlabeled data effectively. As object detection is a generalization of classification with the added complexity of localization, it is important to study SSL foundations in classification prior to extending them to more complex tasks.

This chapter provides a comprehensive review of state-of-the-art SSL methods in image classification. The goal is to establish a baseline knowledge of such approaches and their underlying principles that will inform and motivate their application to object detection in the following chapters. The methodologies discussed are drawn from several surveys [31–35], and are organized into five primary categories: consistency regularization methods (Section 3.2), self-learning methods (Section 3.3), generative methods (Section 3.4), graph-based methods (Section 3.5) and hybrid methods (Section 3.6), each offering a unique approach to leverage unlabeled data. Each section begins with a brief introduction, followed by some example frameworks.

3.2 Consistency Regularization Methods

The core idea of consistency regularization is to minimize the predictive variance under realistic perturbations in either the input space or the model space. The predictive variance is generally measured as the discrepancy between two network outputs, and by minimizing the resulting loss, it ensures that the model becomes more resilient to variations introduced during training, thereby improving its generalization performance. The different types of perturbations used in consistency regularization can be grouped into three distinct levels: input-level perturbations, network-level perturbations, and training-level perturbations. Each level targets a specific aspect of the learning process to introduce controlled variability, encouraging the model to make consistent predictions despite these changes.

- **Input-Level Perturbations**: Involves directly modifying the input data to create variations that the model must learn to handle consistently. Common techniques include:
  - Additive Noise: Introducing random noise to input data, such as Gaussian noise, to simulate imperfections.
  - Random Augmentations: Applying transformations like rotations, flips, color adjustments, or cropping to the input images, mimicking real-world variability.
  - Adversarial Training: Generating intentionally modified inputs, called adversarial examples, that are designed to slightly modify the data in a way that challenges the model's robustness.
- **Network-Level Perturbations**: Focus on introducing variability in the structure of the neural network itself. For example:
  - Layers/Nodes Dropout: Temporarily removing specific layers, neurons, or connections during training, prevents the model from relying too heavily on particular neurons or paths, making the network more robust.
- **Training-Level Perturbations**: These involve changes applied during the training process to promote stability and consistency across training epochs.
  - Stochastic Weight Averaging (SWA): Averaging weights from multiple checkpoints across different training steps to smooth out the optimization landscape and improve generalization.
  - Exponential Moving Averages (EMA): Maintaining a weighted average of model parameters over training iterations, where recent parameters are given higher weights, ensuring a stable and robust model.

Several metrics may be used to quantify the loss, including the Mean Squared Error (MSE), Kullback-Leibler (KL) divergence, and Jensen-Shannon (JS) divergence. The most popular is MSE. If we consider C as the number of classes, x and $\tilde{x}$ two different augmentations of the same input, and f and $\tilde{f}$ two networks derived from any model perturbation, the losses can be defined as follows:

$$\text{Input Pertubation :} \quad d_{\mathbf{MSE}}(f(x_i), f(\tilde{x}_i)) = \frac{1}{C}\sum_{i=1}^{C}(f(x_i) - f(\tilde{x}_i))^2 \tag{3.1}$$

$$\text{Model Pertubation :} \quad d_{\mathbf{MSE}}(f(x_i), \tilde{f}(x_i)) = \frac{1}{C}\sum_{i=1}^{C}(f(x_i) - \tilde{f}(x_i))^2 \tag{3.2}$$

Ladder Network [36] The method is built upon the structure of a Teacher-Student model and inspired by a Deep Denoising AutoEncoder [37]. There are two main processing paths, called encoders: a "clean" encoder and a "corrupted" encoder, along with a decoder that aims to reconstruct the clean features. The clean encoder processes the input data directly, while the corrupted encoder injects Gaussian noise at each layer, simulating variations in data. The decoder then uses the outputs from the corrupted encoder to reconstruct the original clean features. Through this process, the model learns to "denoise" the corrupted input, encouraging it to produce stable and generalized representations of the input data that can ignore irrelevant noise. The clean encoder can be viewed as the teacher, while the decoder acts as the student.

The method distinguishes itself from denoising autoencoders not only by applying the learning task at every layer, not just the inputs, but also by incorporating latent skip connections that link corresponding layers in the encoder and decoder allowing the decoder to combine information from earlier layers, that captures simpler features, with information from later layers regarding more complex features.

Training relies on a consistency loss, for a dataset of unlabeled data, the loss, ℓ, is calculated as the mean squared error (MSE) between the activations z from the clean encoder and the reconstructed activations $\tilde{z}$ from the decoder. The error is computed across all layers, L, with each layer contribution weighted by a specific factor λ_l derived by tuning, which reflects the importance of that layer to the total loss. This loss is then combined with the supervised loss from the labeled dataset, measured in a standard supervised loss.

$$\ell_{unlabled} = \sum_{l \in L} \lambda_l \times d_{MSE}(z^{(l)}, \tilde{z}^{(l)}) \tag{3.3}$$

Π Model [38] Represents a simplification of the Ladder Networks, which eliminates the complex encoder-decoder architecture mentioned previously. It consists of a single neural network, which is trained to produce consistent predictions under stochastic perturbations applied to the inputs or the model itself. The core idea remains, the model should achieve consistent predictions when the same input is processed multiple times, even if some processing steps include injecting randomness or noise. This consistency helps generalization, encouraging similar predictions in related images. The method is applied to both labeled and unlabeled data during training. For labeled data, the model still relies on a standard supervised goal, such as minimizing cross-entropy loss. For unlabeled data, it focuses solely on this consistency training by generating two predictions $\tilde{y_1}$ and $\tilde{y_2}$ from different perturbations applied either on the input or the model. To balance the two losses and guide the learning process, the Π Model gradually increases the weight of the unlabeled loss, w, as training progresses. Initially, the model emphasizes labeled data, allowing it to stabilize and learn base patterns before fully incorporating the unlabeled data.

$$\ell = \ell_{labeled} + w \times \ell_{unlabled}, \qquad \ell_{unlabled} = d_{MSE}(\tilde{y}_1, \tilde{y}_2) \tag{3.4}$$

3.3 Self-Training Methods

Self-training works by initially training the model on the labeled data. Once trained, the model generates predictions on unlabeled data and selects samples with high confidence scores, treating them as "pseudo-labeled" examples and adding them to the labeled dataset in subsequent training iterations. This process of Entropy Minimization is also supported by the cluster assumption, which states that the decision boundary should not cross high-density regions but instead lie in low-density regions. By iteratively incorporating high-confidence samples, self-training implicitly pushes the decision boundary toward low-density regions, aligning with this assumption. A limitation of such methods is that the model cannot correct its own mistakes, which means that incorrect or biased predictions may be reinforced over time, leading to confident yet inaccurate results.

Pseudo-labeling [39] The basic idea is to first train the model on labeled data using regular supervised learning and cross-entropy loss. Then, for the unlabeled data, the same model is used to make predictions. If the model is confident about certain predictions, above a set confidence threshold, those predictions are treated as if they were correct labels and added to the labeled dataset to help train the model further.

The process is repeated every epoch until the model stops producing confident predictions, allowing the model to continue to learn without labeled examples. However, if the initial labeled examples are not good enough or the threshold is not correctly tuned, there is a high risk of incorporating low-quality pseudo-labels, leading the model to reinforce incorrect patterns.

The formula (3.5) explains the pseudo-labeling process, where the class of an input is determined by the highest predicted class, if it is above a threshold τ. The loss function follows a standard supervised learning approach. For labeled examples, the true label serves as the ground truth, while for unlabeled examples, the pseudo-label, $\tilde{y}$, is treated as the ground truth. As commonly observed in other methods, a weighting factor is used to balance the contributions of the supervised and unsupervised losses.

$$\tilde{y}_{class} = \begin{cases} 1 & \text{if } class = argmax\ f(x) \text{ and } f(x)_{class} \geq \tau, \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

Co-training [40] Co-training framework assumes each input x in the dataset has two different and complementary views, v_1 and v_2, and each view is sufficient for training a good classifier. The different views may be multi-modal data or can be generated by injecting noise or by applying different augmentations. The method involves training two classifiers, each on an independent view of the data, where each classifier iteratively labels unlabeled examples it is confident about,

and these labels are used to augment the training set of the other classifier. However, in order for the idea to work, it is important that the classifiers are not strongly correlated, otherwise their potential to provide each other with useful information is limited. The basic idea of co-training can be expanded from dual-view to triple or multi-view. For example, in Tri-training, three classifiers are trained together, with labels assigned to the unlabeled data when two of them agree on the predictions and the confidence scores are higher than a threshold. Let f_1 and f_2 represent two classifiers trained on different views of the inputs. For the labeled dataset, the standard cross-entropy loss H is used. For the unlabeled dataset, a natural similarity measure, such as the Jensen-Shannon divergence, is used. Denoting $\tilde{y_1}$ as the output of the first model on the first view of the data, and $\tilde{y_2}$ as the output of the second model on the other view, the losses can be expressed as follows:

$$\ell_{labeled} = H(y, \tilde{y_1}) + H(y, \tilde{y_2}) \tag{3.6}$$

$$\ell_{unlabeled} = H\left(\frac{1}{2}(\tilde{y_1} + \tilde{y_2})\right) - \frac{1}{2}\left(H(\tilde{y_1}) + H(\tilde{y_2})\right) \tag{3.7}$$

3.4 Generative Methods

Generative semi-supervised learning models aim to capture the underlying data-generating process by learning $p(x)$, the probability distribution of the input data. This allows generative models to produce realistic data samples similar to the input distribution, which means they capture key features and patterns in the data. When labels are incorporated, these models can condition on y to estimate $p(x|y)$, or the probability of an input given a class, which is particularly useful for classification tasks.

Generative Adversarial Networks [41] GANs combine a generative model and a discriminative model in a competitive framework. The method consists of a generator, G, that creates synthetic data and a discriminator, D, that tries to distinguish between real data and synthetic data generated by G. These models are trained simultaneously: the discriminator is trained to maximize its accuracy in identifying real versus fake data, while the generator is trained to "fool" the discriminator, generating data that is realistic.

Variational Autoencoders [42] VAEs are a type of generative model that learns the structure of data through latent variables. A VAE consists of two main components: an encoder and a decoder. The encoder takes the input, compresses it, and maps it into a distribution within the latent space, typically a Gaussian distribution. The decoder then uses samples from this distribution to

reconstruct the original input. During training, VAEs optimize a dual objective, learning how to best encode the data and then how to decode it back as accurately as possible.

3.5 Graph-Based Methods

The labeled and unlabeled data points can be considered as nodes of a graph, and the objective is to propagate the labels from the labeled nodes to the unlabeled ones by using the similarity of two nodes x_1 and x_2, which is reflected by the edge e_{12} between the two nodes. The process involves two main steps:

1. **Graph Construction:** A graph is constructed using all data points, both labeled and unlabeled, where edges are created based on similarity measures such as Euclidean distance or a Gaussian kernel, and neighbor selection methods like K-Nearest Neighbors are used to determine which edges to include.

2. **Label Propagation:** Once the graph is built, labels from the labeled nodes are propagated to unlabeled nodes based on the graph structure.

The underlying assumption is that nodes connected by edges with high weights are likely to share the same label, aligning with the manifold assumption, which says that data points close to each other on a low-dimensional manifold should have similar labels [43].

3.6 Hybrid Methods

In recent years, several hybrid methods that integrate concepts from multiple semi-supervised learning approaches have been introduced into holistic frameworks. These methods basically achieve better performances by leveraging the strengths of different strategies. This category is the primary focus of this thesis, and each method will be discussed in greater detail. Moreover, all methods presented in this section will have their own adaptation for the task of object detection.

MixMatch [4] Combines consistency regularization and entropy minimization into a unified loss function. The core idea is to make effective use of unlabeled data by encouraging the model to produce consistent and confident predictions on similar samples, improving the ability of generalization, as is represented in the Figure 3.1. Given an unlabeled input example, u, the model first applies K different weak augmentations such as random crops, flips, or weak color jitters, denoted as $\alpha_1(u), \alpha_2(u), ..., \alpha_K(u)$. For each augmented view, the model predicts the class probabilities that will be averaged to output a final pseudo label $\tilde{q}$. Formally, the pseudo-label is given by:

$$\tilde{q} = \frac{1}{|K|} \sum_{k=1}^{K} f(\alpha_k(u)) \tag{3.8}$$

However, in order to encourage the model to output confident predictions, an entropy minimization procedure called sharpening is used. It uses temperature scaling to adjust the distribution

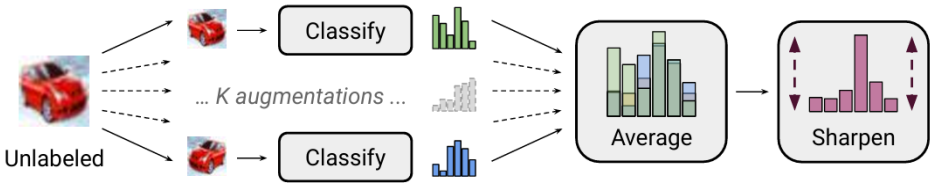


Figure 3.1: MixMatch generates K weakly augmented versions of each unlabeled instance and feeds them to the model. These predictions are averaged to output a pseudo-label, which is further adjusted by using a post-processing step called *Sharpen*, that performs entropy minimization.

of class probabilities, making the most probable class more pronounced. This process can be easily visualized in Figure 3.1.

$$\text{Sharpen}(p_i) = \frac{p_i^{1/T}}{\sum_j p_j^{1/T}} \tag{3.9}$$

Once pseudo-labels are generated, both labeled and unlabeled examples are mixed using the MixUp [29] augmentation, which blends the image-label pairs to create synthetic examples that will be fed to train the model. This synthetic interpolation regularizes the model by encouraging linear behavior between examples, which helps prevent overfitting and improves generalization. The combined loss is represented in the formula below, where the labeled loss uses cross-entropy loss, represented by H, and the unlabeled loss uses an L2 loss. Being X and U a batch of labeled and unlabeled samples respectively, and p, q are its corresponding labels, being q a pseudo-label, the formulas are as followed:

$$\ell_x = \frac{1}{|X|} \times \sum_{x,p \in X} H(p, f(\alpha(x))) \tag{3.10}$$

$$\ell_u = \frac{1}{|U|} \times \sum_{u,q \in U} ||q - f(\alpha(u))||^2 \tag{3.11}$$

ReMixMatch [10] Extends MixMatch by introducing three main improvements: distribution alignment, augmentation anchoring, and a self-supervised auxiliary task based on rotation prediction. These enhancements aim to improve the class balance in predictions, reduce the confirmation bias in pseudo-labels, and encourage the learning of generalizable features.

Distribution alignment addresses the issue of class imbalance in pseudo-labels by aligning the model predictions on unlabeled data with the marginal label distribution estimated from labeled data. Specifically, given a prediction q, for an unlabeled sample u, the method adjusts q by multiplying it with the ratio of the empirical class distributions followed by a normalization. The ratio mentioned is computed using a moving average of the class distributions of past pseudo-labels, $\hat{p}(y)$, and the empirical class distribution of the labeled data, $p(y)$, that is obtained before training. This adjustment ensures that pseudo-labels follow a distribution closer to that of the labeled

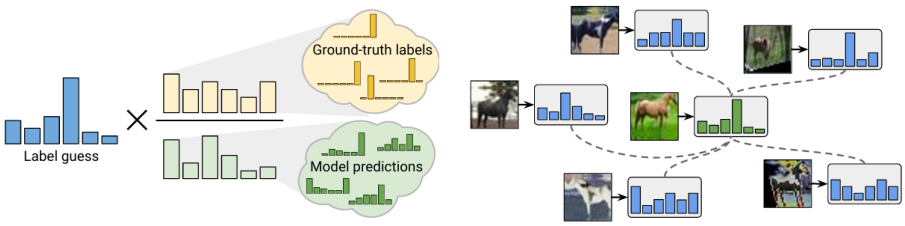


Figure 3.2: Diagram with an overview of the improvements introduced by ReMixMatch. The left image relates to the distribution alignment procedure, where the class distribution of the pseudo-labels is encouraged to follow the class distribution of labeled data, addressing problems in class imbalance. On the right, the new form of consistency regularization is introduced, where multiple strong augmentations are trained in a weak augment pseudo-label.

dataset, hopefully leading to better pseudo-labels. Given $p(y)$ and $\hat{p}(y)$, the adjustment is defined as follows:

$$q = f(u) \tag{3.12}$$

$$\hat{q} = Normalize(q \times p(y)/\hat{p}(y)) \tag{3.13}$$

Augmentation anchoring replaces the consistency regularization part of MixMatch. Instead of enforcing consistency across weak augmentations, ReMixMatch uses weak augmentations only to generate a pseudo-label to be used on training with several strong augmented versions. The framework still uses MixUp and the sharpening procedure, introducing RandAugment [30] and CTAugment as other forms of strong augmentation. Specifically, for each unlabeled example, ReMixMatch first generates a pseudo-label using a weakly augmented version of the input. Then, it creates K strongly augmented versions of the same input, which are mixed using MixUp and trained to match the generated pseudo-label.

Lastly, the authors introduce an auxiliary rotation prediction task, inspired by self-supervised learning. For each unlabeled image, a randomly rotated version is generated using one of four fixed angles (0º, 90º, 180º, 270º). The model is then trained to predict the applied rotation as a four-class classification problem. This encourages the model to learn meaningful features from the structure of the input data, improving its generalization ability.

Apart from the supervised loss, the overall loss function in ReMixMatch incorporates four components that contribute to both improved performance and training stability. Two of these components are MixUp related losses: one applied to mixed labeled examples and the other to mixed unlabeled examples. The remaining two components include a pre-MixUp consistency loss, added to improve training stability, and the rotation prediction loss introduced earlier as a self-supervised auxiliary task. Let $\hat{X}$ represent the batch of labeled data after MixUp, $\hat{U}$ the batch of unlabeled data after MixUp, and U the batch of weakly augmented unlabeled data before MixUp. These loss components are formally defined in Equations 3.14, 3.15, and 3.16 and the relative importance of each loss component is controlled by the hyperparameters $\lambda_{\hat{U}}$, λ_U, and λ_r:

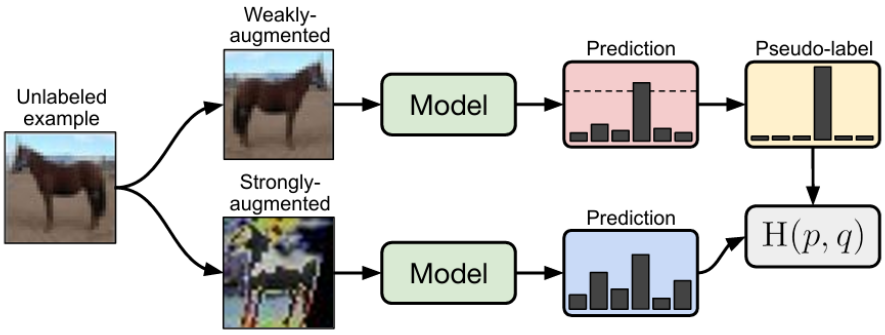


Figure 3.3: Diagram with a overview of FixMatch. For each unlabeled example, two synthetic samples are created via weak and strong augmentations. The weak augmentation is used to calculate a pseudo-label, and the strongly augmentation image is used to train with that pseudo-label.

$$\sum_{x,p\in\hat{X}} \mathrm{H}(p, f(\theta(x))) \tag{3.14}$$

$$\lambda_{\hat{U}} \times \sum_{u,q\in\hat{U}} \mathrm{H}(q, f(\theta(u))) + \lambda_U \times \sum_{u,q\in U} \mathrm{H}(q, f(\theta(u))) \tag{3.15}$$

$$\lambda_r \times \sum_{u\in U} \mathrm{H}(r, f(\mathrm{Rotate}(u, r))) \tag{3.16}$$

where θ represents the strong augmentation procedure.

FixMatch [5] Focuses on simplicity and proposes a much simpler method represented in Figure 3.3, that combines consistency regularization and pseudo-labeling. For labeled data, the standard cross-entropy loss is applied, being computed between the model prediction and the true label. For unlabeled data, FixMatch follows a multi-step process to create pseudo-labels for training. For each unlabeled input, the method first applies a weak augmentation, such as a simple crop or flip, and passes this version through the model to get the predicted class probabilities. If the probability of the most likely class exceeds a confidence threshold, τ, it is assigned as a pseudo-label. With the pseudo-label assigned, the next step is to apply a strong augmentation, such as through RandAugment [30] and color jitter, to create an artificial version of that unlabeled example. These strongly augmented examples are then treated as labeled data, considering the pseudo-label as the true label, allowing us to also use the same cross-entropy loss. Let $\tilde{x}_{weak}$ and $\tilde{x}_{strong}$ represent augmentations obtained from a weak augmentation function, α, and a strong augmentation function, θ, respectively. The losses can be defined as follows:

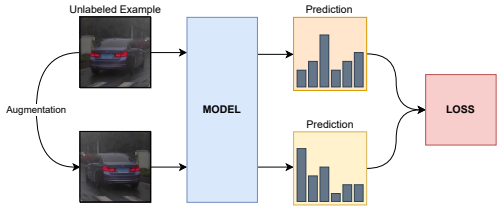
$$\ell_{labeled} = H(y, f(\tilde{x}_{weak})) \tag{3.17}$$

$$\ell_{\text{unlabeled}} = \mathbf{1}_{[argmax f(\tilde{x}_{\text{weak}}) \geq \tau]} H(f(\tilde{x}_{\text{weak}}), f(\tilde{x}_{\text{strong}})) \tag{3.18}$$

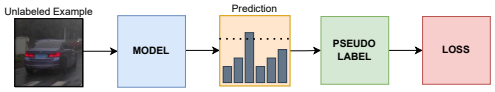
3.7 Summary

This chapter highlighted the different approaches to handling semi-supervision in classification tasks. Among these, hybrid methodologies have consistently demonstrated superior performance, as they integrate the strengths of multiple techniques to overcome individual limitations.

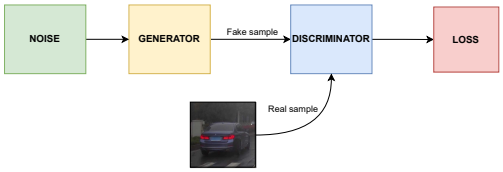
The Figure 3.4 illustrates the high-level conceptual framework of each methodology category. Consistency regularization methods emphasize the importance of robust model predictions under perturbations. Self-training approaches demonstrate the potential of iteratively refining predictions using pseudo-labels, though they come with challenges related to confirmation bias. Hybrid techniques have emerged as a solution to these limitations, combining multiple strategies to create more robust learning frameworks. Generative methods and graph-based techniques, although not deeply explored, also provide different and valuable approaches to deal with unlabeled data.



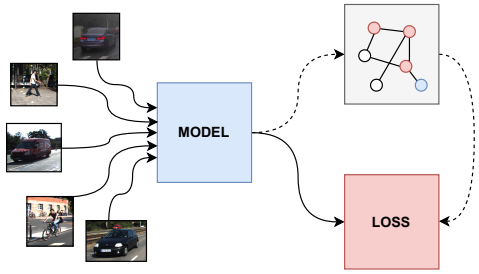
(a) Consistency Regularization Methods



(b) Self Training Methods



(c) Generative Methods



(d) Graph-based Methods

Figure 3.4: Categories of semi-supervised learning methodologies.

Chapter 4

Review on Semi-Supervision for Detection

4.1 Introduction

In the previous chapter, a state-of-the-art review was performed on semi-supervised learning methods applied to image classification tasks. Building on that foundation, this chapter shifts the focus to dense tasks. The literature review revealed that SSL has been explored more extensively in semantic segmentation than in object detection. This observation partly motivated the decision to focus on object detection as the target task.

Despite not being directly addressed in this thesis, semantic segmentation insights are still considered potentially useful, as both tasks can eventually share similar challenges and may benefit from common solutions. Therefore, this section includes a review of the state of the art in both object detection and semantic segmentation. While object detection involves locating and classifying individual objects within an image, semantic segmentation is a related task with the goal of assigning a class label to each pixel, providing a more detailed understanding of the scene.

4.2 Semantic Segmentation

Dense FixMatch [44] FixMatchSeg [45] both frameworks extend FixMatch [5] beyond image classification, maintaining the use of pseudo-labeling and consistency regularization through data augmentations. Additionally, the authors also include the Mean Teacher framework to improve robustness against noisy pseudo-labels and class imbalances.

In dense tasks like semantic segmentation, augmentations such as rotations or translations will not only transform the input image but also cause corresponding changes in the spatial structure of the output predictions. For example, a rotation applied to the input will result in a rotated segmentation output, making the predictions from augmented views incompatible with the original pseudo-labels. To address this, the core idea is to align the reference frames of pseudo-labels generated from weakly augmented views with predictions from strongly augmented views. Dense

FixMatch addresses this by aligning the reference frames of pseudo-labels generated from weakly augmented views with predictions from strongly augmented views. It achieves this alignment by applying an inverse geometric transformation to the weakly augmented predictions and then reapplying the strong augmentation. This approach ensures compatibility of predictions across augmentation types and enables a pixel-level consistency loss. FixMatchSeg, on the other hand, simplifies the data augmentation pipeline by carefully managing how augmentations are applied. To prevent misalignment between images and their corresponding labels, it applies only geometric transformations uniformly across both the weak and strong augmentation paths. This ensures that spatial relationships are preserved. For the strong view, FixMatchSeg avoids any additional geometric changes and instead focuses on intensity-based augmentations like sharpness and contrast adjustments or Gaussian blur. This approach maintains consistency between weak and strong predictions, eliminating the need for extra alignment procedures.

The supervised loss evaluates the model predictions by comparing them to the corresponding ground truth labels at each spatial location. The loss is averaged across all spatial locations within each labeled example and further averaged over the batch of labeled examples. The simplified formula is shown below, where S is the total number of spatial locations in the output. The function, H, is the loss function used to compute the error between $f(x_i)$, the model prediction, and $\alpha(y_i)$, the ground truth label at location i, with α representing any required spatial transformation applied to align the labels with the predictions.

$$\ell = \frac{1}{S}\sum_{i=1}^{S} H(f(x_i), \alpha(y_i)) \tag{4.1}$$

Regarding the unsupervised loss, it is designed to enforce consistency between the model predictions under different augmentations of the same unlabeled input. This loss compares the predictions from strongly augmented views of the input to pseudo-labels generated from weakly augmented views. The comparison is performed at each spatial location, and the loss is averaged across all locations within each unlabeled input and over all unlabeled examples in the batch. The formula is shown below, where the term $match(\bar{y}_i; \alpha, A)$ represents the operation that aligns pseudo-labels $\bar{y}_i$ from a weakly-augmented view to the reference frame of the strongly-augmented view. It applies the inverse of the weak augmentation geometric transformation, α, to the pseudo-labels and then reapplies the strong augmentation transformation, A.

$$\ell = \frac{1}{S}\sum_{i=1}^{S} H(f(x_i), match(y_i, \alpha, A)) \tag{4.2}$$

CorrMatch [46] presents a framework that leverages correlation maps to improve pseudo-label accuracy and semantic consistency through two unique label propagation strategies. The correlation maps are used to propagate labels across pixels and regions, thus expanding high-confidence pseudo-label regions while maintaining accuracy. The label propagation process is performed in a

two-stage process. The first stage, pixel propagation, computes global pairwise pixel similarities using correlation maps constructed from feature vectors extracted by the network encoder. The similarity is then integrated into predictions to enhance the semantic understanding of unlabeled data. The second stage, region propagation, refines pseudo-labels using class-agnostic shapes derived from correlation maps. These shapes, which encapsulate local spatial information, are matched with high-confidence regions to adjust and expand pseudo-labels.

The supervised loss employs standard cross-entropy to train the model on labeled data, while the unsupervised loss combines three components: hard pseudo-label loss, ℓ^h, soft prediction consistency loss, ℓ^s, and a novel correlation-based loss, ℓ^c, where λ_1, λ_2, λ_3 are hyperparameters that balance the contributions of these components.

$$\ell = \lambda_1 \ell^h + \lambda_2 \ell^s + \lambda_3 \ell^c \tag{4.3}$$

Cross-Consistency Training (CCT) [47] The framework adapts consistency regularization techniques, a common SSL approach for classification, to semantic segmentation. While consistency training enforces invariance in predictions under perturbations of the inputs, CCT observes that for semantic segmentation, the cluster assumption is more prominent in the encoder latent space rather than the input space. At the input level, the cluster assumption is violated because low-density regions do not align with class boundaries. In contrast, the encoder output preserves the cluster assumption, with class boundaries exhibiting high average distances between feature representations, corresponding to low-density regions. This key observation forms the foundation of their approach, where perturbations are applied to the encoder outputs rather than the raw inputs, enabling more effective learning and class separation in the latent space.

The method consists of a shared encoder and multiple decoders. The encoder processes input images and generates latent feature representations. These features are then passed to a main decoder and multiple auxiliary decoders. The main decoder is trained in a supervised manner using labeled examples. Simultaneously, the auxiliary decoders are fed perturbed versions of the encoder latent outputs. These perturbations are applied to the encoder latent space to challenge the auxiliary decoders with augmented versions of the same data.

The core idea is to enforce consistency between the predictions of the main decoder and the auxiliary decoders, and the overall loss function for training combines both the supervised and unsupervised components. The supervised loss is calculated using the labeled data, typically through cross-entropy, which compares the predicted segmentation map to the ground truth pixel labels. For unlabeled data, the unsupervised loss measures the discrepancy between the predictions of the main decoder and the auxiliary decoders. This discrepancy is calculated using a mean squared error (MSE) between the probability distributions of the decoders' outputs. Additionally, since the auxiliary decoders are used only during training, the inference process remains efficient, involving only the shared encoder and the main decoder.

4.3 Object Detection

Unbiased Teacher v2 [48] generalizes semi-supervised object detection to both anchor-free and anchor-based detectors and introduces the Listen2Student mechanism to improve bounding box regression by refining the pseudo-labels' quality. This framework primarily addresses two challenges in semi-supervised object detection. First, it adapts pseudo-labeling techniques to anchor-free object detectors, which predict object bounding boxes without predefined anchor boxes. Anchor-free models often use a "centerness" score to evaluate how central a point is to an object. However, under semi-supervised conditions, this centerness score becomes unreliable. To overcome this, the authors select pseudo-labels based solely on classification scores, ignoring centerness. Additionally, instead of using advanced label assignment techniques, which tend to fail when pseudo-labels have noisy bounding boxes, the method chooses a simpler approach that assigns all pixels within a bounding box as foreground. The second major enhancement involves bounding box regression, which predicts the precise boundaries of the detected objects. Existing methods use confidence scores to select pseudo-labels for regression, but these can be misleading because they do not account for boundary-specific errors. To solve this, the authors introduce a mechanism called Listen2Student. This mechanism adds an extra branch to the model to predict uncertainty for each boundary (top, bottom, left, right) of a bounding box. During training, a Teacher-Student paradigm is employed, and the predictions of each one are compared for each boundary. If the teacher predicted uncertainty for a boundary is lower than that of the student, that boundary is used to calculate the regression loss. Otherwise, it is ignored to prevent the student from learning possible noisy information. This selective process ensures that the student only learns from reliable pseudo-labels.

The training process begins with a "burn-in" stage, where the models are trained using labeled data to initialize them. Following this, in the main training phase, the teacher generates pseudo-labels from unlabeled data, and the student uses these labels along with the labeled data to refine its predictions. The regression and classification branches are improved simultaneously, with the Teacher parameters being updated via EMA.

The supervised loss is computed using labeled data and consists of two components: a classification loss and a regression loss. For the classification branch, the loss measures the difference between the predicted class probabilities and the ground truth class labels. For the regression branch, the loss evaluates the error in bounding box predictions relative to the ground truth bounding boxes. The classification loss is typically implemented using the cross-entropy loss, while the regression loss employs the negative power log-likelihood loss (NPLL) to handle the bounding box predictions.

Regarding the unsupervised loss, it is computed on unlabeled data using pseudo-labels generated by the Teacher model. For the classification loss, pseudo-labels with confidence scores above a threshold are used to guide the Student learning. For the regression loss, the Listen2Student mechanism is used, which selects boundaries for training based on the relative uncertainties of the teacher and student.

DenSe Learning [49] The framework is built around a Teacher-Student paradigm and has some similarities with FixMatch [5], but introduces several additional mechanisms to improve overall performance. Both labeled and unlabeled images are weakly augmented, with the teacher model generating pseudo-labels for the unlabeled images.

To control the quality of the pseudo-labels, DSL employs an Adaptive Filtering mechanism that partitions pseudo-labels into three categories: foreground, background, and ignorable regions. Unlike simpler methods that use a single threshold, DSL uses multiple thresholds to classify image regions more accurately, ensuring that predictions between the two thresholds are considered uncertain and excluded from gradient propagation. A mechanism called MetaNet is further applied to those pseudo-labels, which reevaluates the pseudo-labels by comparing their features with class-level representations derived from labeled data. If the features of a pseudo-labeled instance deviate significantly from the corresponding class representation, it is reassigned to the ignorable region category.

Beyond Adaptive Filtering and MetaNet, DSL introduces three additional components: the Aggregated Teacher, Patch Shuffling, and a scale loss. The Aggregated Teacher is introduced to address the limitation of Exponential Moving Average (EMA) teachers, where parameter aggregation treats each layer independently. This mechanism uses shared recurrent layers to explicitly link hidden states across layers, promoting more stable knowledge propagation. Patch Shuffling is a novel strong augmentation technique that splits an image into patches and shuffles them, removing the dependency of foreground objects on their surrounding context. Finally, the scale loss ensures consistent outputs among scales through downsampling, improving the model's robustness to object scaling variations.

MUM [50] The authors identify that typical interpolation-regularization (IR) methods, such as MixUp [29] and CutMix [51], create ambiguity in the labels when applied to object detection. These methods interpolate input images, which can blur object boundaries or mix objects with backgrounds, resulting in challenges for generating reliable pseudo-labels that account for both classification and localization tasks. To address this, the framework proposed preserves object localization by reassembling mixed features into their original positions while retaining the benefits of IR.

The MUM method works by splitting images into tiles and mixing these tiles across images to create mixed training samples. After the mixed images are passed through the network, the resulting feature maps are "unmixed," restoring the tiles to their original geometrical positions before feeding them to the detection head. This ensures that the spatial and contextual integrity required for localization is preserved while still leveraging the benefits of IR for regularization.

MUM is integrated into a teacher-student framework, a common architecture in semi-supervised learning. The teacher generates pseudo-labels for weakly augmented images, while the student is trained on strongly augmented mixed images created using MUM. The teacher network is updated via exponential moving average (EMA) of the student weights, ensuring that the teacher improves over time.

STAC [52] The framework is one of the first attempts to extend semi-supervised learning strategies, originally designed for image classification, to the more complex task of object detection. This work is particularly relevant to this thesis, as it closely aligns with the research objective of adapting classification-based SSL methods to object detection.

STAC proposes a two-stage training pipeline that leverages two principles from classification-based SSL: pseudo-labeling and consistency regularization, highly inspired by FixMatch [5]. In the first stage, a standard object detector model is trained on the labeled data until convergence. The trained model is then used to generate pseudo-labels for the unlabeled images. To ensure high-quality labels, it applies a confidence filtering mechanism, keeping only bounding boxes with a confidence threshold above a high threshold.

In the second stage, the model is retrained using both the original labeled data and the pseudo-labeled unlabeled data. During this phase, strong data augmentations are applied to the unlabeled images to enforce consistency in predictions.

STAC is relevant not only for its results, which show strong performance gains on benchmarks such as MS COCO [27], but also because it demonstrates a successful direct transfer of SSL techniques from classification to detection, which represents a challenge and opportunity explored in this thesis.

4.4 Summary

Dense tasks, by their nature, require spatially consistent outputs, which present challenges when applying semi-supervised learning methods that were originally designed for image classification. This chapter highlighted a set of state-of-the-art architectures in both semantic segmentation and object detection, emphasizing how SSL methods have been adapted to these tasks.

Rather than directly transferring semi-supervised learning methods from image classification, most approaches in dense prediction tasks introduce new ideas to address the spatial structure and context of the tasks. Nevertheless, several frameworks draw inspiration from successful classification-based methods, particularly FixMatch [5], achieving notable performance gains. This opens the door to exploring other classification-inspired techniques such as MixMatch [4] and ReMixMatch [10], which remain underexplored in this context. Among dense tasks, semantic segmentation has received more attention and often involves simpler SSL frameworks compared to object detection. For this reason, the primary focus of this thesis is placed on the more complex and less-explored problem of semi-supervised object detection.

Chapter 5

Methodology

5.1 Introduction

This chapter describes the methodology used to adapt and evaluate the proposed semi-supervised learning (SSL) frameworks for object detection. It begins by presenting the datasets used, how they were split for the experiments, and the pretraining applied, referred to as warm-up. It then introduces the adaptive filter component, which serves as a common foundation for all adapted frameworks, and explains how loss masks are used to manage pseudo-label quality. The chapter then details the specific adaptations made to each SSL method – FixMatch, MixMatch, and ReMixMatch. Finally, the chapter concludes with a case study applying the adapted frameworks to 2D LiDAR range-view projections. This part explores the generalization of the approach to a real-world autonomous driving scenario, offering insights into how SSL performs with alternative input modalities beyond RGB images.

5.2 Datasets

KITTI [53] Released in 2012, it has become a standard benchmark for evaluating object detection algorithms in real-world driving environments. The dataset was collected using a vehicle-mounted sensor suite comprising stereo cameras capturing urban, rural, and highway scenes around Karlsruhe, Germany.

The dataset includes 7,481 training images and 7,518 test images, with a total of 80,256 labeled objects. Although eight different object classes were annotated, only the "Car" and "Pedestrian" classes are officially evaluated, as they have a sufficient number of labeled instances for meaningful benchmarking. However, for the purpose of this work, the "Cyclist" class was also included.

All available images were merged into a unified set and subsequently divided into three subsets: a test set with 1,000 images, a labeled set, with 1-10% of the images depending on the experiment, and an unlabeled set consisting of the remaining images, as illustrated in Figure 5.1.

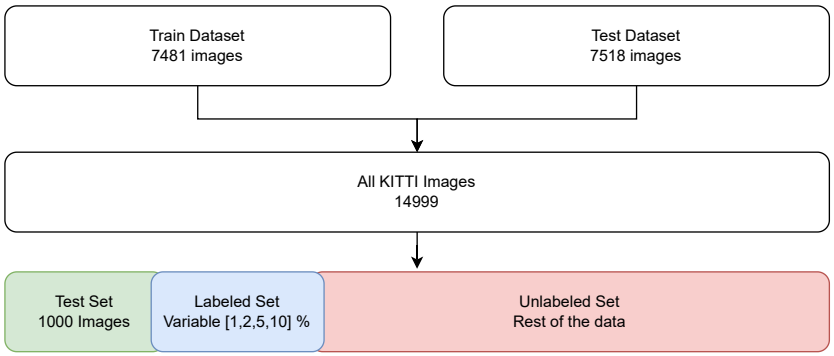


Figure 5.1: Illustration of the data splits for the KITTI dataset folds. In order to increase the unlabeled data compared to the labeled data, as seen in other datasets and in the literature, part of the labeled data is used as unlabeled data.

COCO [27] Introduced in 2014, the Microsoft Common Objects in Context (COCO) dataset has become one of the most widely used benchmarks for object detection and other tasks. The dataset includes predefined splits: over 118,000 images in the `train2017` set, around 123,000 in the `unlabeled2017` set, and 5,000 images in the `val2017` set, with more than 1,5 million object instances labeled across 80 object categories.

In this study, the `val2017` split is used for evaluation and validation, while the `train2017` split serves as the labeled set, sampled according to the experiment proportions (1%, 2%, 5%, or 10%). The `unlabeled2017` split is used entirely as the unlabeled data. One important note about the results over this dataset is the fact that in the literature, the authors use the unused portion of the `train2017` as unlabeled data, e.g., using 10% of labeled data and adding the remaining 90% to the unlabeled set. This was not performed due to computational constraints. While this may limit the potential performance, as the unlabeled data could possibly double its size, it was considered that it would not affect the validity of the study analysis and comparisons. A visual diagram is provided in Figure 5.2.

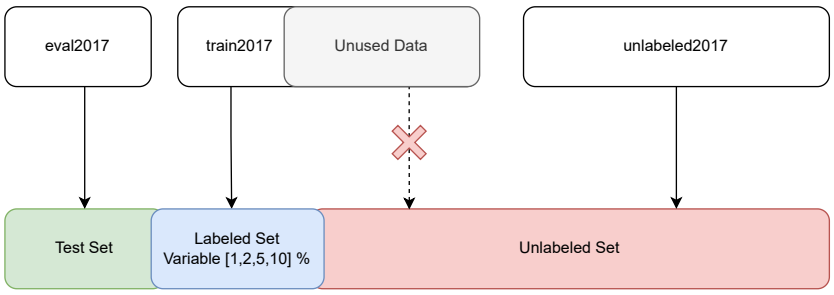


Figure 5.2: Illustration of the data splits for the COCO dataset folds. The dataset is already divided into labeled and unlabeled splits. As seen in the literature, only 1%–10% of the `train2017` data is used as labeled, with the remaining typically added to the unlabeled set. This merging step was omitted here due to computational limitations.

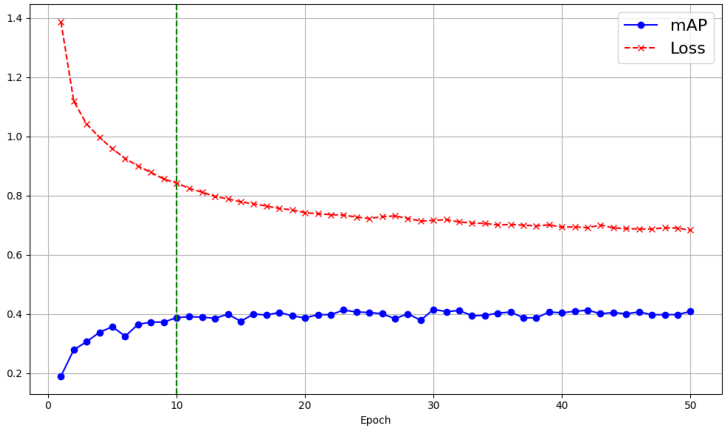


Figure 5.3: Evolution of mAP when training on the labeled subset of the KITTI dataset only. The warm-up duration of 10 epochs was selected empirically by running multiple 50 epoch experiments. Beyond 10 epochs, no substantial performance gains are observed, and continuing the training would only lead to overfitting, which degrades the capability of generalization.

5.3 Warm-up

Following practices observed in semi-supervised object detection frameworks such as DSL [49] and STAC [52], the model should be initially trained exclusively on the labeled data. In the early stages of training, the pseudo-labels generated for the unlabeled data are typically of low quality. If these noisy labels are used too early, it can misguide the model, leading to unstable training and degraded performance. Although models can eventually recover from such noise, this typically results in slower convergence and may prevent the model from reaching its full potential.

To address this, the frameworks cited before perform a two-stage training pipeline. In the first stage, the model is trained only with the labeled data until convergence. With the trained model, it generates pseudo-labels for each unlabeled example and moves into the second stage of the training, where the model is trained with all the unlabeled data together with just the generated pseudo-labels. Although it is a valid approach, this work follows a different approach, also seen in the literature, where the pseudo-labels are generated on the fly, at each iteration, rather than once at the start. This online pseudo-labeling approach has the main consequence of being much more computationally intensive, but it allows the quality of the pseudo-labels to improve as the model learns from the unlabeled data.

In summary, a warm-up phase was incorporated, during which training was conducted exclusively on the labeled data. This allows the model to first learn solid and reliable representations before being exposed to the potentially noisy pseudo-labeled data. By doing so, when the unlabeled data is incorporated, the model is capable of generating more accurate pseudo-labels, leading to an overall more efficient training.

In our experiments, the duration of the warm-up phase was determined empirically by monitoring the convergence behavior of the model trained solely on the labeled data. Figure 5.3 demonstrates this process for the KITTI dataset, where the warm-up continues until the model stops

showing clear improvements, which suggests that the labeled data has been mostly used to its full learning potential. To minimize the risk of slight overfitting to the labeled subset, the number of warm-up epochs was kept as low as possible while still allowing the model to achieve the best performance with labeled data only.

Specifically, a warm-up period of 10 epochs is used for the KITTI dataset, while 20 epochs are applied for COCO. These values balance the need for solid representations a priori with the objective of early integration of unlabeled data to maximize the benefits of semi-supervised learning.

5.4 Adaptive Filter

In semi-supervised object detection, the effectiveness of a framework depends on the quality of the pseudo-labels and the ability to handle and filter noise.

In image classification, the output space is relatively simple, consisting of a single categorical label per image. This simplicity allows consistency regularization techniques such as enforcing that the model predictions remain consistent under data augmentations, for example, an image of a cat should remain classified as a cat regardless of whether the image is flipped or cropped, making consistency-based training relatively straightforward and robust.

In contrast, object detection presents a more complex output space. It requires predicting not only one but multiple object categories and their corresponding bounding box coordinates. As a result, the application of consistency regularization is far more challenging. If a pseudo-labeled bounding box is inaccurately placed or incorrectly classified, and the model is trained to replicate this prediction across augmentations, it risks being penalized for failing to imitate an incorrect prediction. This is particularly problematic when the pseudo-label includes a bounding box in a region where no object exists, as the model will be explicitly penalized for failing to hallucinate the same false positive. As seen in Figure 5.4, object detection is inherently more sensitive to label noise than classification. Techniques that perform well in semi-supervised classification cannot be directly applied to object detection without careful adaptation.

All classification frameworks that will be adapted to object detection rely on consistency regularization, and it was observed that without addressing the limitations of consistency regularization first, the models failed to converge, with performance deteriorating further at each epoch.

To address this limitation, all the frameworks will be using an Adaptive Filter that will control what areas of the images will be considered in the training and in the loss. This process is inspired by DSL [49] where the authors observed that using a single confidence threshold to separate foreground and background instances during pseudo-labeling introduces significant label noise. When the threshold is set too high, many true positive samples may be incorrectly excluded and treated as background, thereby discarding possible valuable patterns. Conversely, a low threshold increases the risk of including background regions as false positives, further corrupting the training data. To address this, the authors propose a dual-threshold strategy that more effectively distinguishes between high-confidence foreground, ambiguous regions, and clear background, leading

Figure 5.4: Raw output examples of the model to be considered as pseudo-labels. The model will fail to learn if all bounding boxes are included in the training, as there is a lot of noise. This requires a type of threshold, however, it is not direct as there is a trade-off between recall and precision.

to improved detection performance by reducing both false positives and false negatives. It uses the following formula to classify each pixel of the image:

$$pixel_{h,w} = \begin{cases} Foreground & \text{if } p_{h,w} > \tau_2 \\ IgnorableRegion & \text{if } \tau_1 \leq p_{h,w} \leq \tau_2 \\ Background & \text{if } p_{h,w} < \tau_1 \end{cases} \tag{5.1}$$

where τ_1 is the background threshold, τ_2 is the foreground threshold and $p_{h,w}$ is the object confidence on that position.

Pixels with prediction scores between τ_1 and τ_2 fall into an "ignorable" region, where those areas are excluded from the loss computation, meaning that their associated gradients are not backpropagated. This prevents the model from reinforcing potentially noisy pseudo-labels.

Furthermore, rather than using a fixed τ_2 for all classes, a class-specific threshold, τ_2^k, is computed on the fly. This value is derived based on the amount of predicted objects of each class for a specific image, scaled by a parameter ($\beta = 0.05$) and a reference threshold ($\tau = 0.70$). This allows the model to have slightly different thresholds for each image, depending on the amount of predicted objects, allowing it to achieve better performances as a fixed foreground threshold is too restrictive or too permissive depending on the classes. In the KITTI dataset, the class distribution is not uniform, and some classes, e.g., Cyclist, have a lower number of samples and lower confidence scores compared to the Car class. The formula is shown below, where k is the class of the threshold being computed, h, w are the dimensions of the feature map, N_{pos} the total number of locations, and $p_{h,w}$ the score on that specific location.

$$\tau_2^k = \left(\frac{\sum_{h,w} \mathbf{1}_{\{\bar{p}^*_{h,w} == k\}} p_{h,w}}{N_{\text{pos}}} \right)^{\beta} \times \tau \tag{5.2}$$

Figure 5.5: Diagram with example pseudo-labels and the corresponding mask. On the left, the model's raw predictions are displayed, where after applying the Adaptive Filter, only the confident boxes are kept, resulting in the middle image. Alongside the pseudo-labels, the Adaptive Filter also outputs a loss mask, excluding areas where an uncertain box was placed. The illustrated mask may indicate excessive ignorable areas, but that is directly related to the tuning of the background and foreground thresholds. As the background threshold increases, more boxes will be considered as background instead of ignorable regions, resulting in a trade-off between learning signals and noise.

The final result after applying the Adaptive Filter will be the filtered pseudo-labels, bounding boxes, and a feature map mask that will be used in the loss computation by indicating which are the confident pixels that will be penalized and have their gradients propagated. The Figure 5.5 represents an example of the final pseudo-labels and their corresponding masks after applying the adaptive filter.

5.5 Loss Masks

The main goal of loss masks is to provide finer control over the pseudo-labels and not wrongly penalize the model for predicting a bounding box in uncertain areas. For each training image, a corresponding loss mask is generated and integrated into the loss computation. This mechanism offered some technical challenges in this work due to the multi-scale nature of the FPN alongside the FCOS architecture. FCOS generates predictions at multiple scales using a Feature Pyramid Network (FPN). Each FPN level is responsible for detecting objects of different sizes, meaning that a single, uniform loss mask cannot simply be computed and resized to fit all levels. Instead, loss masks must be computed separately for each scale level.

To understand how these masks are implemented, it is helpful to recall how loss is computed in FCOS. The input image is passed through a backbone and then through the FPN, which produces five different feature maps, each corresponding to a different object scale. While FCOS is described as making "per-pixel predictions," these pixels refer to locations in the feature maps, not the original image resolution. Therefore, when applying a loss mask to exclude certain bounding boxes, it is necessary to first determine which FPN level and which feature map location generated that bounding box.

Each loss mask is effectively a binary map for a specific FPN level, indicating which spatial locations should be considered or ignored during training. While it is conceptually helpful to

visualize the mask as covering the original image, in practice, the mask operates at the resolution of each feature map level in the latent space, as represented in Figure 5.6.

The process becomes more complex when data augmentations are applied after the loss mask has been computed. Augmentations such as Cutout, Random Crop, or Rotation change the spatial layout of the image, which means that the associated loss masks must also be transformed accordingly. For example, if Cutout removes a 100x100 pixel region in the input image, the corresponding region must be correctly scaled and mapped for each FPN level, taking into account the downsampling factor. This introduces potential sources of error due to interpolation or rounding when aligning augmentations and masks across different resolutions, so the used augmentations were carefully selected.

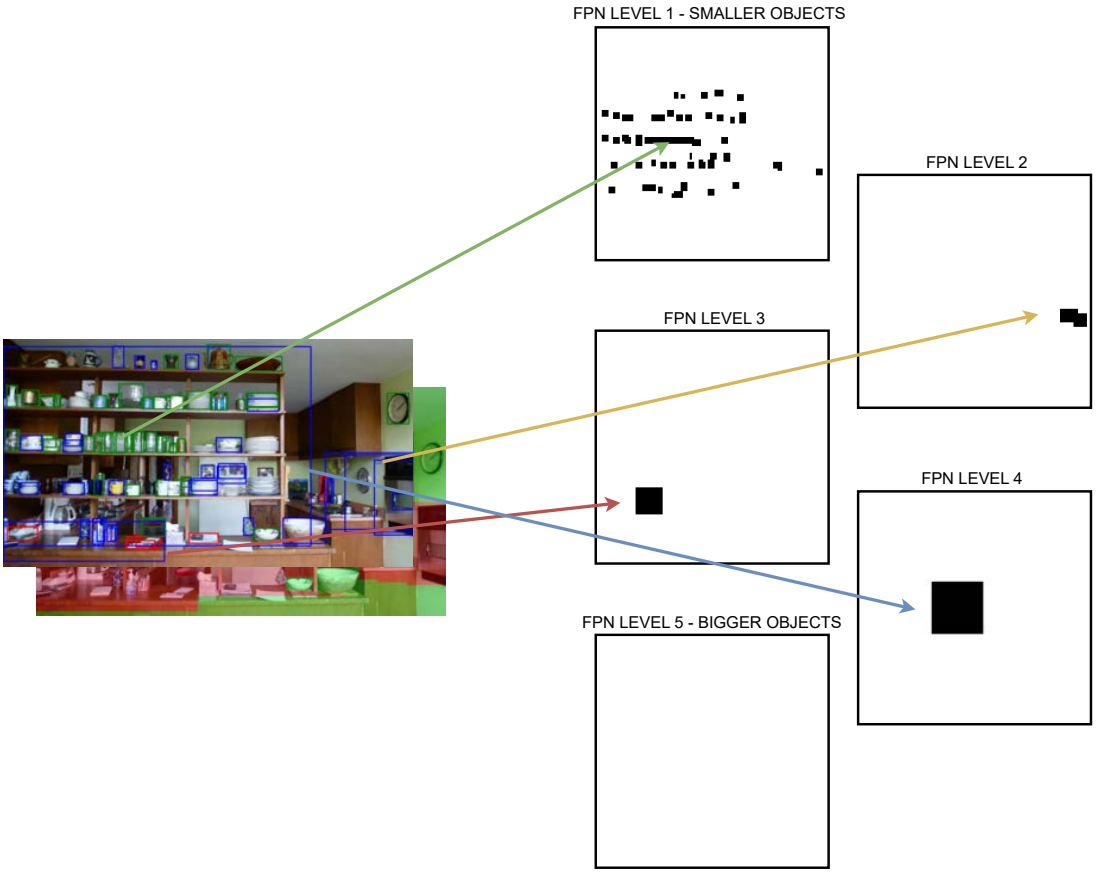


Figure 5.6: Illustration of how the loss mask operates internally. On the left, the raw predictions of the model are shown for a sample MS-COCO image. Underneath, there is a representation of the loss mask that has been used until now with green/red colors to explain how the mechanism works. However, internally, it has a different representation. On the right, the internal representation of the loss mask is displayed: white pixels indicate locations in the feature maps that will contribute to the loss, while black pixels are the ones excluded, where, for illustration purposes, the boxes were considered all uncertain. The colored arrows illustrate how bounding boxes are mapped across different FPN levels, where the ones with confidence scores falling between the thresholds τ_1 and τ_2 have their corresponding feature map locations masked out to avoid contributing noisy gradients during training.

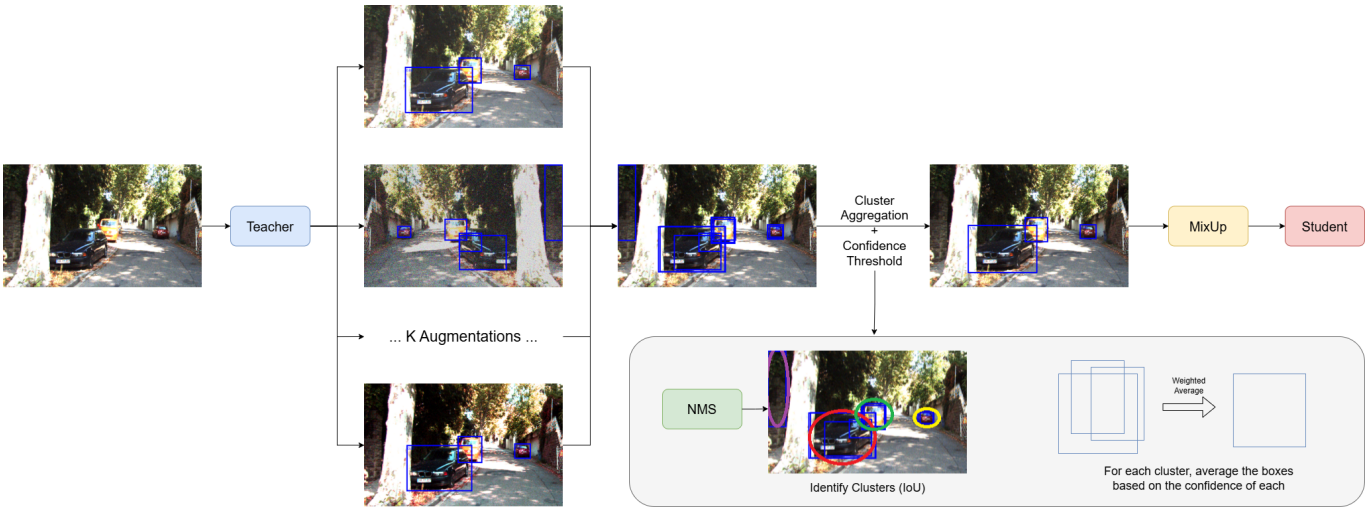


Figure 5.7: Illustration of the MixMatch adaptation to object detection. The method starts by performing K augmentations on an unlabeled example and getting the corresponding predictions. These predictions are then merged into a single set, and a cluster aggregation algorithm is used to generate pseudo-labels. Clusters are formed based on the IoU threshold, and the final pseudo-label for each cluster is computed using a weighted average of the bounding boxes, with weights proportional to their confidence scores.

5.6 MixMatch

In the following paragraphs, each of MixMatch's core techniques is described in detail, along with an explanation of how they were adapted for the object detection task. A general overview of the framework is given in the Figure 5.7.

Data Augmentation For each unlabeled example, K different augmentations were applied and passed through the model. Instead of outputting class probabilities, as in classification, the model will output a list of bounding boxes and respective class labels. The value of K used was 2, and the augmentations chosen are: horizontal flips, weak Gaussian noise, color jitter, and random brightness and contrast.

Label Guessing MixMatch averages the class probabilities of the different augmented versions, reducing the noise inherent in individual predictions. The result is a soft pseudo-label that reflects a more robust estimate of the underlying class distribution for the given input.

The object detection adaptation introduces significant modifications to address the spatial aspect inherent in the task. Unlike classification, object detection involves a localization component, meaning that predictions cannot be averaged directly. Before averaging, all boxes must be spatially aligned, specifically, it is essential to ensure that all K predictions are mapped to a common spatial reference frame. Augmentations that modify spatial structure, such as horizontal flips, must be inverted again before aggregation. This alignment guarantees that predicted bounding boxes refer to the same absolute image coordinates, making cross-augmentation comparison meaningful.

Following spatial alignment, all predicted bounding boxes and class scores are concatenated into a single set, which inherently includes redundant predictions from the multiple augmented views. In order to unify and consolidate those predictions into robust pseudo-labels, a clustering mechanism was adopted.

The process begins by applying Non-Maximum Suppression (NMS) to all bounding boxes, using the associated scores to select unique highest confidence bounding boxes, where each retained box will form a different cluster. For each retained box, the Intersection-over-Union (IoU) is computed with all remaining boxes. Boxes with IoU higher than a predefined threshold, 0.6, are assigned to the same cluster. Within each cluster, a final pseudo-box and its score are computed via a confidence-weighted average of all member boxes. This procedure, explained in the pseudo-algorithm below, enables the model to generate high-quality pseudo-labels, refined by multiple augmented views of the same image.

Algorithm 1 Label Guessing for Object Detection

Require: Unlabeled image x, number of augmentations K, IoU threshold τ

Initialize empty list of predictions: $P \leftarrow []$
for $k = 1$ to K **do**
  $x_k \leftarrow Augment(x)$
  $(B_k, S_k) \leftarrow Model(x_k)$ // Box and Score
  $(B_k, S_k) \leftarrow Augment^{-1}(B_k, S_k)$
  $P \leftarrow P \cup [(B_k, S_k)]$
end for
PL $\leftarrow$ [] // Pseudo Labels
F $\leftarrow$ NonMaximumSuppression(P)
for each box b in F **do**
  $ClusterBoxes \leftarrow []$
  $ClusterScores \leftarrow []$
  for each (b', s) in P **do**
    if IoU$(b, b') > \tau$ **then**
      $ClusterBoxes \leftarrow ClusterBoxes \cup [b']$
      $ClusterScores \leftarrow ClusterScores \cup [s]$
    end if
  end for
  $Sharpened \leftarrow [s^2$ for $s \in ClusterScores]$
  $Weights \leftarrow Sharpened / \sum Sharpened$
  $AverageBox \leftarrow \sum_j Weights_j \cdot ClusterBoxes_j$
  $PL \leftarrow PL \cup [AverageBox]$
end for
return PL

Sharpening In the original MixMatch formulation, a sharpening operation is applied to the averaged class probabilities in order to reduce their entropy. This step encourages the model to make more confident predictions, aligning the guessed labels more closely with the one-hot format used for labeled data. The sharpening is typically implemented using a temperature parameter in a softmax function, which adjusts the "peakiness" of the distribution.

FCOS does not use a softmax over the class logits but instead applies independent sigmoid activations per class. As a result, each class score is interpreted independently, rather than as part of a normalized distribution over mutually exclusive classes. This fundamental difference in how class probabilities are modeled means that typical entropy-based sharpening, which assumes a normalized distribution, is not directly applicable.

Moreover, applying sharpening indiscriminately across all feature maps to locations does not work well in object detection. In classification, the whole image is matched to a class, but in detection, the vast majority of pixels or anchors correspond to the background. Applying a sharpening function to the background would wrongly admit that every pixel belongs to a class, removing the possibility of the background.

To adapt the sharpening principle in a detection-aware manner, a confidence squaring operation is applied to the bounding boxes' scores. This operation increases the influence of high-confidence boxes during label generation by further boosting their scores, while suppressing the influence of lower-confidence boxes. Specifically, for each bounding box pseudo-label, the confidence score is updated as follows:

$$Score_{box} = \frac{Score_{box}^2}{\sum_{b \in B} Score_b^2} \tag{5.3}$$

This procedure may not correlate much with the original MixMatch formulation, as it does not perform entropy minimization, although it was inspired by it and serves as a heuristic weighting mechanism that favors more confident predictions during pseudo-label refinement.

MixUp Once pseudo-labels are assigned, MixMatch applies MixUp as a form of strong augmentation before passing the samples to the model. MixUp works by performing a linear interpolation between two data samples, including both the inputs and their labels, as presented in Section 2.7.2. In the context of object detection, there are multiple ways to merge the labels when applying the MixUp, however, naive adaptations such as simply concatenating all bounding boxes or interpolating bounding box coordinates are not ideal. The former neglects the unequal mixing of images, ignoring the fact that objects from the less dominant image are harder for the model to detect. The latter introduces noise, as the resulting interpolated bounding boxes do not accurately represent any real object.

Inspired by LossMix [54], each bounding box was assigned a weight based on the opacity of its corresponding image. Objects from the less visible image receive a lower weight in the loss function. These weights are integrated into the training pipeline through the loss mask, which was

previously binary but is now extended to a continuous range in $[0,1]$. Each value indicates the relative contribution of the corresponding anchor to the overall loss.

$$\tilde{x} = \lambda \times x_1 + (1-\lambda) \times x_2 \tag{5.4}$$

$$\tilde{b} = b_1 \cup b_2 \tag{5.5}$$

$$w_{box} = \begin{cases} \lambda & \text{if box originates from } x_1, \\ 1-\lambda & \text{if box originates from } x_2 \end{cases} \tag{5.6}$$

$$m_a = \begin{cases} m_a \times w_{box} & \text{if } a \in box \\ m_a & \text{otherwise} \end{cases} \tag{5.7}$$

where a is a feature map location and m_a is the same location in the loss mask.

Loss Function The loss function employed is identical to that used in the official FCOS implementation. All three loss components are modified to account for the soft mask, which determines both the regions of the image that should be penalized and the respective weights assigned to each region.

$$l = \sum_{a \in A} m_a \times (l_{cls}(a) + l_{reg}(a) + l_{cen}(a)) \tag{5.8}$$

where A denotes the set of all anchors, or feature maps locations, and $m_a \in [0,1]$ represents the loss mask explained in Section 5.5, that controls the contribution of the respective anchor to the final loss.

5.7 ReMixMatch

ReMixMatch builds upon the foundational ideas of MixMatch and incorporates additional mechanisms to further enhance performance. In the following paragraphs, each major component of ReMixMatch is discussed in detail, along with insights into how they were adapted and integrated within the object detection pipeline. A framework overview is presented in Figure 5.8.

Distribution Alignment Distribution alignment addresses the mismatch between the model predicted label distribution on unlabeled data and the true distribution observed in labeled data. In semi-supervised learning, models tend to become biased toward predicting certain classes more frequently. ReMixMatch mitigates this by adjusting the class probabilities of the pseudo labels such that their overall distribution matches the one on the labeled dataset. Practically, this involves computing the moving average of class distributions on labeled samples and normalizing the predictions on unlabeled data accordingly.

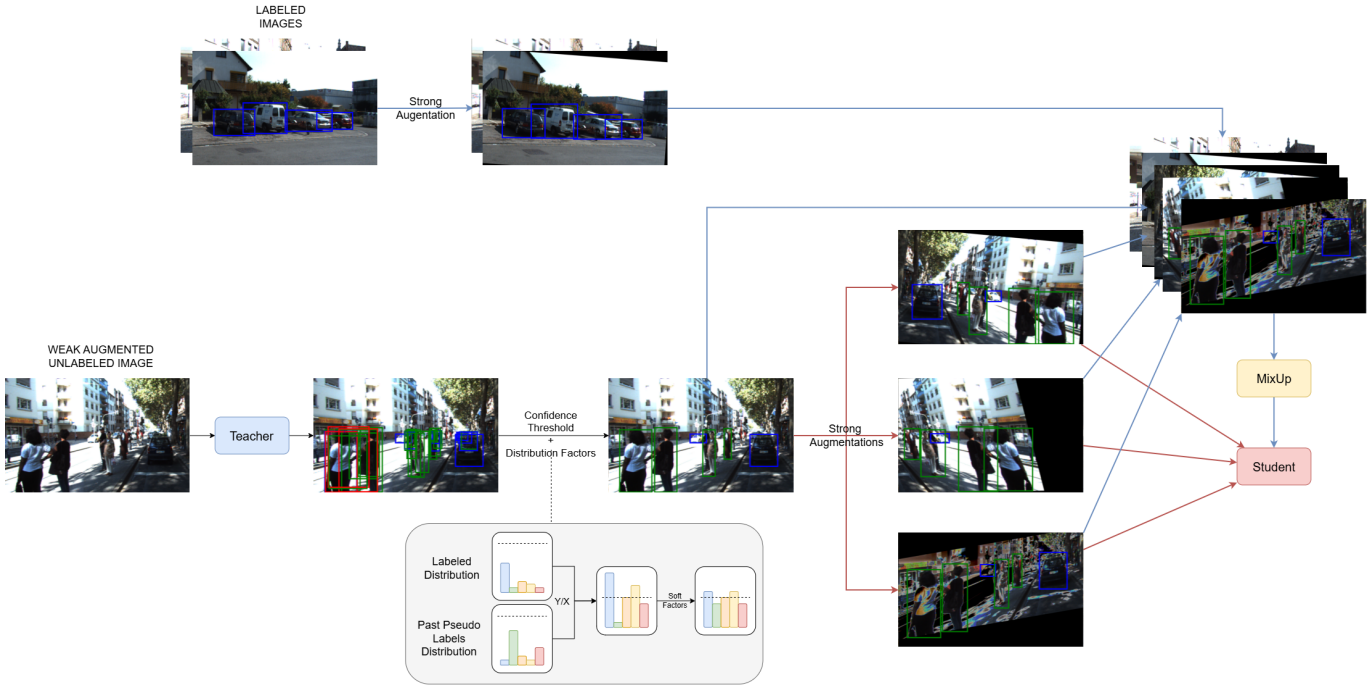


Figure 5.8: Overview of ReMixMatch for Object Detection. The model begins by generating predictions on weakly augmented unlabeled images. These predictions are refined using distribution alignment and confidence thresholding to output pseudo-labels. The Distribution Alignment works by calculating the ratio between the labeled class distributions and the moving average of past pseudo-label classes, the result will be softened to allow smooth updates. Each pseudo-labeled image is then strongly augmented K times and combined with a batch of strongly augmented labeled images. The diagram uses blue and red arrows to represent different loss paths. Blue arrows relate to loss components involving MixUp, while red arrows represent a stabilization path that uses strongly augmented images without MixUp.

In the original formulation, the class distributions of the predicted image are scaled by factors calculated through the labeled and unlabeled distributions, however, in the context of detection, the concept cannot be applied directly, instead of class distributions, there is candidate bounding boxes with a corresponding class and score. To adapt this concept to object detection, the overall methodology remains consistent with the original formulation. Specifically, the labeled distribution is computed based on labeled samples before training starts, and the alignment process is only enabled when there is at least one epoch of examples available from the current distribution to ensure statistical reliability.

To perform the adjustment, the confidence of the boxes is scaled by some calculated factors, which are calculated per class by the following formula:

$$scale_{factors} = \text{clamp}(\frac{p_{target}}{p_{model} + \varepsilon}, 0.7, 1.3) \tag{5.9}$$

where p_{target} denotes the labeled data distribution, p_{model} represents the moving average distribution of the model pseudo labels, and ε is a small noise to prevent zero division. The clamp ensures that the scaling factors are bounded between 0.7 and 1.3. This constraint is crucial, as it prevents excessive corrections that could otherwise introduce additional noise into the learning process. The overall goal with this adjustment is to slowly encourage the model to include or exclude more pseudo-labels of some classes that may be uncertain or over-confident.

Augmentation anchoring Augmentation anchoring is introduced to improve the consistency regularization aspect of MixMatch. ReMixMatch defines a single weakly augmented version of the unlabeled image as the anchor, and then enforces consistency between the anchor and the multiple strongly augmented versions of the same input. The key insight is that weak augmentations are more likely to preserve the semantic content of the input, thus providing a more stable reference point. By aligning predictions from strong augmentations to this anchor, the model is encouraged to remain robust to significant transformations while grounding its learning in more reliable signals. Instead of $K = 8$, $K = 4$ was chosen, and for the weak augmentation and strong augmentations, a Horizontal Flip and RandAugment [30] + Cutout was applied, respectively.

MixUp The MixUp strategy follows the same principles presented in the MixMatch adaptation, with a slight change that follows the original ReMixMatch formulation. MixUp is applied after strongly augmenting both the unlabeled and labeled samples.

Rotation Loss ReMixMatch introduces a rotation loss as an auxiliary self-supervised objective to improve the representation learning of the model. The idea is to encourage the network to learn more generalizable features by predicting the rotation angle (e.g., 0°, 90°, 180°, 270°) of input images. This task is particularly effective in image classification, where global semantic understanding of the image content is the key. However, in the context of object detection, global

semantic understanding loses its meaning, instead it requires precise localization. Rotation-based self-supervision focuses primarily on learning global image-level features and does not contribute meaningfully to the learning of spatially localized object representations. As such, it does not align with the goals of the detection task and was excluded from the framework.

Loss Function The overall loss function in ReMixMatch consists of multiple components, excluding the rotation loss originally proposed in the paper. The remaining three loss terms are retained and adapted to the object detection pipeline. The total loss is defined as:

$$\ell = \sum_{x,p \in X'} \lambda_{X'} \times \ell(x,p) + \sum_{u,q \in U'} \lambda_{U'} \times \ell(u,q) + \sum_{u,q \in U'_1} \lambda_{U'_1} \times \ell(u,q) \tag{5.10}$$

where X' represents labeled samples with MixUp applied, U' consists of unlabeled samples with MixUp applied, and U'_1 includes strongly augmented unlabeled examples without MixUp, used to improve stability. Notably, the function ℓ used in the formula terms represents the standard FCOS loss presented in the Formula (5.8) which also includes the mask.

5.8 FixMatch

FixMatch simplifies the previous frameworks and still achieves state-of-the-art performance in image classification. Unlike previous approaches that relied on complex consistency regularization techniques, FixMatch combines these ideas into one straightforward pipeline. To the best of our knowledge, it is also the only framework that has been directly adapted to object detection, as performed by the authors of STAC [52]. In the following section, the main components of FixMatch will be explained, and implementation choices and how they differ from those in related work will be discussed.

Figure 5.9: Overview of FixMatch for Object Detection. The framework creates pseudo-labels for unlabeled images by applying a confidence threshold to predictions from weakly augmented versions. These pseudo-labels are then used to enforce consistency on the same images after applying strong augmentations. In the figure, a hard-to-see car is excluded from the loss, due to a low confidence score, through the loss mask. If the model does predict a bounding box in that region, it is not penalized.

Pseudo-labels To calculate pseudo-labels, this framework simply weakly augments the unlabeled image, followed by a threshold confidence filter. The main novelty for this framework is mostly included in the first sections of this chapter, when it is described the adaptive filter for thresholding per class, and the loss masks that ended up being also reused in the others frameworks.

Strong augmentation Once pseudo-labels and their associated masks are generated, both the original image and its corresponding mask are strongly augmented. The implementation uses a combination of RandAugment and Cutout as its strong augmentation strategy, similar to the approach followed in the ReMixMatch adaptation. The model is then trained to match its prediction on the strongly augmented image to the pseudo-label derived from the weakly augmented one, thus enforcing consistency under heavy input transformations.

Loss Function Similar to MixMatch, the loss function employed is also identical to that used in the official FCOS implementation. All three loss components are modified to account for the soft mask, which determines both the regions of the image that should be penalized.

$$\ell = \sum_{a \in A} m_a \times (\ell_{cls}(a) + \ell_{reg}(a) + \ell_{cen}(a)) \tag{5.11}$$

where A denotes the set of all anchors, and $m_a \in [0,1]$ represents the loss mask explained in Section 5.5, that controls the contribution of the respective anchor to the final loss.

5.9 Case Study: Application to LiDAR 2D Projections

To further investigate how well the proposed SSL frameworks adapt to real-world scenarios, their application in the context of autonomous driving was explored. As discussed in Section 2.8, autonomous vehicles rely on multiple sensor inputs, primarily cameras and LiDAR sensors, where camera images provide rich texture and color information, while LiDAR data offers greater robustness in adverse conditions such as low lighting and fog.

LiDAR sensors generate high-resolution 3D point clouds, capturing precise spatial information about the environment. However, deep learning models designed for vision tasks typically process structured, grid-like data, such as 2D images, making raw point clouds incompatible with conventional convolutional architectures. This study explores projecting 3D LiDAR point clouds into 2D representations, enabling the direct application of SSL frameworks originally developed for image-based tasks. The KITTI dataset was used for this case study since it offers both LiDAR and RGB camera data, making it ideal for comparing model performance across different modalities. Notably, the application of semi-supervised learning to LiDAR 2D projections remains underexplored in existing literature, providing strong motivation for this case study.

Range View Projection In this study, the LiDAR point clouds were projected into the image plane of the forward-facing camera, using both the sensors and calibration data provided in the KITTI dataset. This process involved aligning the coordinate systems of the LiDAR and the camera through a series of geometric transformations.

Once transformed into the camera coordinate frame, the 3D points were projected onto the 2D image plane. Points that fell outside the image boundaries or behind the camera were filtered out. The remaining valid points were then visualized on a blank black background, creating a 2D image where each point's color corresponds to its depth value. A continuous color map was used to represent the depth, allowing distant points to appear in cooler colors and closer points in warmer tones. An example projection was presented in Section 2.8.

Note that this projection allows the use of RGB labels, since the LiDAR and RGB images are aligned within a common reference frame.

Chapter 6

Experiments & Discussion

6.1 Introduction

Following the standard evaluation protocols in the literature on semi-supervised object detection, as seen in [48, 49, 52], the performance of the models is reported under different levels of supervision. Specifically, all methods were evaluated using four labeling thresholds: 1%, 2%, 5%, and 10% of the labeled data used. The unused portion of labeled data is added to the unlabeled set, except for COCO, where computational limitations prevent this.

The primary goal of this approach is to investigate the effectiveness of leveraging large quantities of unlabeled data while minimizing reliance on labeled samples. By progressively reducing the fraction of labeled data, the aim is to quantify the extent to which the unlabeled data can compensate for limited supervision and to evaluate the robustness of semi-supervised learning methods in low-label regimes.

6.2 Implementation Details

In order to implement the concept of the loss masks (presented in Section 5.5), it required modifying the internal structure of an object detection model.

YOLOv8 The initial model selected for this task was the YOLOv8 implementation from Ultralytics [55]. While this framework offered a highly abstracted and user-friendly API, its high-level design was not ideal for the present work and was proven to be a limiting factor. Specifically, the Ultralytics implementation lacked the necessary flexibility for fine-grained control over the training loop and model internals. Semi-supervised learning typically requires training on both labeled and unlabeled data simultaneously, combining the standard supervised loss with the pseudo-label supervision derived from model predictions. However, the Ultralytics framework abstracts away the majority of the internal model flow, also providing only the final post-processed bounding box predictions. This design makes it difficult to access intermediate representations like the

multi-scale feature maps produced by the Feature Pyramid Network (FPN), which are essential for constructing and applying the loss masks.

Additionally, the loss computation itself is deeply embedded within the framework's internal abstractions. Modifying it to include custom logic, such as applying spatial loss masks selectively across anchors, would require too many changes to undocumented components and internal utilities. In the early stages of the project, considerable effort was invested in adapting the Ultralytics YOLOv8 framework to meet the requirements of semi-supervised training. This involved implementing numerous overrides and low-level modifications to insert the necessary functionality. However, these workarounds quickly became overwhelming, leading to increased instability, recurring errors, and a fragile development process that was hard to maintain over time. The effort ultimately led to the choice of transitioning towards a more open and extensible solution that supports the goals of this work.

FCOS The other chosen alternative was the Fully Convolutional One-Stage Object Detector (FCOS), presented in Section 2.4, and implemented by the PyTorch [56] `torchvision` library. This switch offered full access to the model source code and facilitated the development of a custom training loop tailored specifically for semi-supervised learning. The first step was to create a custom subclass that extends the official `torchvision.models.detection.FCOS` model. This subclass introduced new functionality for dynamically computing the spatial loss masks, where the added functionality enhances the existing anchor-matching logic of FCOS with additional steps.

The model first obtains the multi-scale feature maps from the backbone, including the Feature Pyramid Network (FPN), and generates anchors for each scale level. For an anchor to be matched to a bounding box, it must satisfy several spatial criteria: (i) it must be within the box boundaries, (ii) it lies within a radius based on the object center, and (iii) it falls within an appropriate scale range relative to the object size. In cases of multiple matches, the bounding box with the smallest area is selected. After the matching step, a binary mask is initialized with all values set to zero. The goal is to mark only the locations where the loss should be computed as valid. All unmatched anchors, considered as background, are initially marked as valid, since they contribute to the negative class signal in the classification loss. For anchors matched to bounding boxes, an additional filtering step is introduced based on prediction confidence. The model compares the predicted class scores of these anchors to a predefined, class-specific threshold, derived from the Adaptive Filter (explained in Section 5.4). Anchors with scores exceeding this threshold are considered reliable and marked as valid. Anchors with very low confidence scores, below the background threshold, set to 0.3 in this implementation, are considered as background and also marked as valid in the mask. Anchors with scores that fall between these two thresholds are excluded from loss computation, as their uncertainty may degrade the model learning. This entire procedure is implemented in our newly `FCOS.compute_masks()` function.

The output will be a spatial loss mask calculated for each image in the batch. These masks are

passed to the loss computation stage and applied consistently across all three FCOS loss components: classification, bounding box regression, and centerness. To implement this masking mechanism, it was necessary to override the `torchvision.models.detection.fcos.FCOSHead` class, specifically the function `FCOSHead.compute_loss()`, which defines the classification and regression heads responsible for loss computation. The standard FCOS loss is computed per anchor for each of the three components. The loss computation was modified by multiplying each per-anchor loss by the corresponding mask value, effectively filtering out uncertain anchors. Finally, the total loss is normalized not by the total number of anchors, but by the total number of valid anchors in the mask. This normalization ensures consistent gradient magnitudes and avoids biasing the loss due to variable masking rates.

6.3 Training Setup

All experiments were conducted using a single GPU. Due to memory constraints and the high resolution of the input images, the batch size was fixed at 16 for both labeled and unlabeled data. Training was performed for 25 epochs for KITTI and 10 for COCO unless otherwise specified. For the KITTI dataset, images were resized to 1024×378 pixels in order to preserve the original resolution. For the COCO dataset, the longer side of each image was resized to 1333 pixels, with padding applied to the shorter side to output a final resolution of 1333×1333 pixels. Throughout all experiments, the optimizer used was Adam with a fixed learning rate of 10^{-4}. The implemented frameworks followed a teacher-student paradigm, where the teacher model was updated at each epoch using an exponential moving average of the student model weights. The backbone of the model was a ResNet-50 [15] pretrained in the ImageNet dataset [57].

6.4 Frameworks Performance

Table 6.1: mAP (%) performance of different frameworks at varying labeling percentages.

	KITTI				COCO			
Method	1%	2%	5%	10%	1%	2%	5%	10%
Supervised	0.2110	0.2781	0.3599	0.4079	0.0942	0.1279	0.1905	0.2327
FixMatch	0.2340	0.3053	0.4189	0.4765	0.1337	**0.1868**	0.2377	0.2722
MixMatch	**0.2519**	0.3110	0.4022	0.4816	0.0974	0.1298	0.2249	0.2569
ReMixMatch	0.2434	**0.3266**	**0.4337**	**0.4910**	**0.1387**	0.1813	**0.2461**	**0.2823**

FixMatch As expected, FixMatch outperforms the supervised baseline at all labeling ratios, both on KITTI and COCO datasets, demonstrating the effectiveness and potential of semi-supervised adaptations for object detection. Despite being the least complex of the three frameworks adapted, relying primarily on confidence thresholding and heavy data augmentation, the method showed solid performance compared with the other frameworks.

MixMatch Starting with the KITTI dataset, MixMatch demonstrates the highest mAP in the extreme low-label regime, particularly at 1% labeled data. This result highlights the strength of its pseudo-labeling mechanism, which leverages a cluster-aggregation strategy to improve the quality of pseudo-labels and outperform other frameworks. However, as the quantity of labeled data grows, the inherent quality of the pseudo-labels naturally improves, which reduces the potential benefit of the framework techniques. Consequently, the sophisticated techniques that offer clear advantages in low-supervision settings become less impactful, and the framework loses its performance advantage. In contrast, on the COCO dataset, MixMatch underperforms globally in all experiments, and in the 1% and 2% labeled data scenarios, fails to yield performance gains. This may be justified by the lack of hyperparameter tuning in this specific dataset, which was not feasible for COCO due to computational constraints. Nevertheless, at 5% and 10% labeled data, the method begins to show modest improvements, although less noticeable when compared to the other frameworks. This suggests that MixMatch may require a baseline mAP above 20% to function effectively.

ReMixMatch ReMixMatch was the top-performing framework on both datasets in almost all experiments. Achieved a maximum 9% mAP gain from the supervised baseline using 10% labeled data on KITTI and 5% mAP gain on COCO. These results highlight the overall robustness and adaptability across datasets and labeling scenarios.

Summary All three of the semi-supervised algorithms perform very well relative to the fully supervised baseline, validating that classification frameworks may be adapted to object detection. All techniques make substantial gains, and the performance improves proportionally to the size of the labeled set. Something very interesting is the comparison of the experiments with 5% and 10% labeled data, where it is possible to observe that all semi-supervised frameworks achieve better performance on the 5% setting compared to the 10% setting of the baseline. Concretely, e.g, in KITTI, ReMixMatch on 5% labeled data achieves 0.4337 mAP, while Supervised at 10% labeled data achieves 0.4079, indicating an almost 3% mAP gain with half of the labeled set.

6.5 Ablation Studies

To better understand what drives the performance gains in the proposed frameworks, a set of ablation studies was performed. The primary goal is to identify which techniques have a positive impact, and if any of them are actually not contributing and just adding complexity.

To this end, each individual component of the framework was disabled, one at a time, while keeping all other elements unchanged. All evaluations were performed using the 10% ratio of labeled data on the KITTI dataset.

Table 6.2: Ablation study on Adaptive Filter.

Method	**With Adaptive Filter**	**Without Adaptive Filter**
FixMatch	0.4765	0.4529
MixMatch	0.4816	0.4713
ReMixMatch	0.4910	0.4554

Adaptive Filter As shown in Table 6.2, the Adaptive Filter (presented in Section 5.4) demonstrates to be an important component in the success of the frameworks. Compared to a fixed pseudo-label confidence threshold of 0.7, the use of the Adaptive Filter leads to a significant performance improvement, which highlights the need for more dynamic thresholds, where a single global threshold does not work, especially when working with class imbalance.

Interestingly, the MixMatch adaptation appears to benefit less from the Adaptive Filter than the other frameworks, showing only a minor performance drop when the adaptive mechanism is removed. This suggests that the MixMatch strategy of generating pseudo-labels from multiple augmented views of the same unlabeled image helps preserve valid pseudo-label candidates while effectively filtering out noise.

As explained in Section 5.4, the Adaptive Filter addresses the global threshold issue by dynamically adjusting the confidence threshold on a per-class and per-image basis. For example, if the model predicts only 2 instances of the person class but 20 instances of the car class in an image, it is likely that the 20 car predictions include more false positives or noisy detections. In such cases, the Adaptive Filter raises the threshold for the over-represented class while potentially lowering it for under-represented ones, improving the reliability of selected pseudo-labels.

In summary, without requiring a manual tuning of the threshold for each class, the Adaptive Filter relies on the distribution of the predicted bounding boxes to adjust the confidence thresholds, which have been proven to boost performance.

Table 6.3: Ablation study on MixMatch adaptation.

Method	**mAP**
Supervised	0.4079
MixMatch	0.4816
MixMatch with K = 1	0.4724
MixMatch with K = 4	0.4801
MixMatch without MixUp	0.4564
MixMatch without Clustering Aggregation	0.4804

MixMatch As seen in Table 6.3, to investigate the adaptation of MixMatch, there were three main components chosen to further investigate, which are the K, corresponding to the number of different augmented predictions to be averaged and create a pseudo-label. The use of the MixUp augmentation and the newly introduced Cluster Aggregation mechanism.

Starting with the K parameter, the default value chosen was $K = 2$, so the first intuition question is whether increasing K would lead to further performance gains. However, experiments with $K = 4$ showed no improvement in performance, while significantly increasing training time, making the framework less efficient and not worthwhile. On the other side, decreasing K to 1 resulted in a slight drop in performance. This is likely due to fewer potential pseudo-labels being caught since only a single prediction was performed. These observations suggest that $K = 2$ offers a balanced trade-off, providing performance gains while maintaining reasonable computational cost.

In contrast, removing the MixUp augmentation resulted in a considerable drop in performance. This aligns with previous findings in the original MixMatch work, where MixUp was highlighted as a key component for classification. The results confirm that its importance carries over to object detection tasks as well.

Finally, the Cluster Aggregation mechanism failed to display any notable impact on performance. However, this does not invalidate the averaging strategy used in MixMatch. When tested without Cluster Aggregation, the algorithm merged all predictions from the augmented views and simply applied Non-Maximum Suppression (NMS) to select the most confident boxes. This outcome suggests that a simpler strategy, retaining only the most confident predictions, may be sufficient, potentially eliminating the need for weighted averaging.

Table 6.4: Ablation study on ReMixMatch adaptation.

Method	**mAP**
Supervised	0.4079
ReMixMatch	0.4910
ReMixMatch with K = 1	0.4698
ReMixMatch with K = 8	0.4972
ReMixMatch without Distribution Alignment	0.4852
ReMixMatch without MixUp	0.4515

ReMixMatch Table 6.4 shows the results of the ablation study on the ReMixMatch adaptation, focusing on three important parts of the framework: the number of augmented views per image, K, the Distribution Alignment component, and the MixUp augmentation. The default setup uses $K = 4$. When K was reduced to 1, the performance dropped noticeably, suggesting that multiple artificial images for each unlabeled image allow the model to achieve better generalization. On the other hand, increasing K to 8 suggests a slight increase in performance. However, as K increases, it also increases the training time, $K = 8$ leads to around 50% time increase, highlighting a trade-off between performance and computational efficiency.

Removing Distribution Alignment resulted in a small drop in performance. The drop may be small because the KITTI dataset has a small number of object classes, making distribution alignment less useful. It was not possible to test this effect on the COCO dataset, which includes more classes, but it could potentially have a higher impact.

The MixUp technique showed the strongest effect. When MixUp was removed, the mAP dropped to 0.4515, a decrease of nearly 4%. This result highlights that MixUp plays a major role in improving ReMixMatch performance, as already seen in MixMatch.

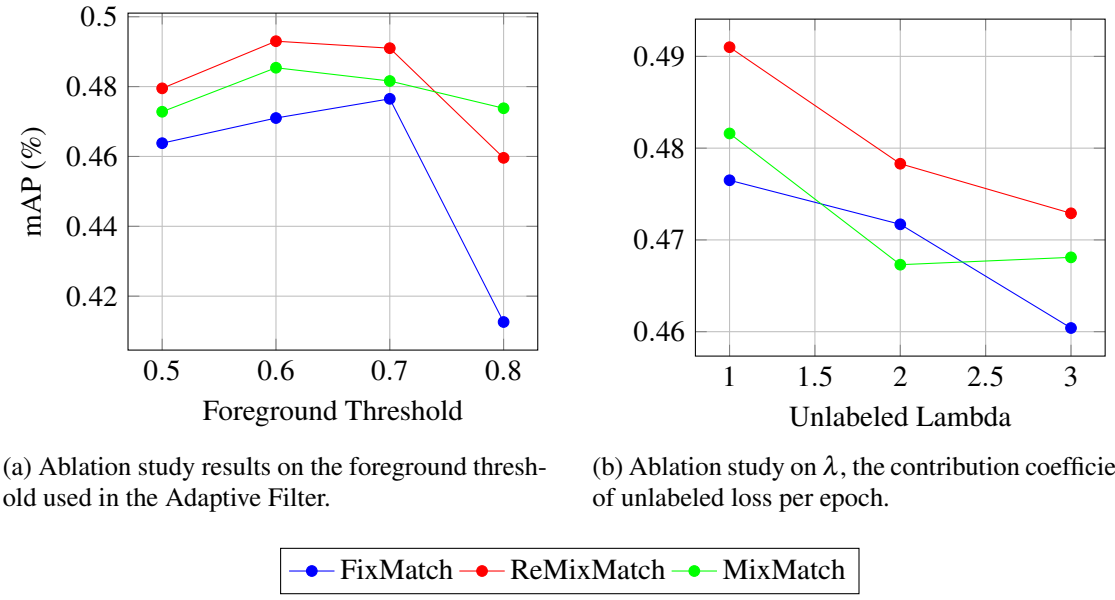
Table 6.5: Ablation study on FixMatch adaptation.

Method	**mAP**
Supervised	0.4079
FixMatch	0.4765
FixMatch without Cutout	0.4752
FixMatch with RandAugment M = 6	0.4751
FixMatch with RandAugment N = 5	0.4794

FixMatch Regarding the FixMatch ablation study, represented in Table 6.5, the primary focus was on the strong augmentations applied. The first aspect explored was the impact of the Cutout augmentation, which is a commonly used technique in the literature, and the main result was that there was no relevant impact. The next intuition was to increase the strength of the augmentations within RandAugment (explained in Section 2.7.3), where there are two possible approaches: increase the number of applied augmentations, N, or increase the intensity of each applied augmentation, M. The default settings used were $N = 3$ and $M = 3$. When increasing N to 5, the results showed an interesting improvement, whereas increasing M to 6 yielded little to no benefit. These findings slightly challenge earlier results, as the original experiments used $N = 3$, and only through ablation was it observed that applying a greater variety of augmentations could lead to better performance. This insight may also extend beyond FixMatch to ReMixMatch, which similarly employs RandAugment and was the top-performing framework overall.

Foreground Threshold One of the core elements of all frameworks is the confidence threshold. For that reason, it is important to study the impact of increasing or decreasing the threshold value. Increasing the threshold results in fewer, but more accurate pseudo-labels, potentially limiting learning due to fewer new patterns. On the other side, lowering the threshold increases the number of pseudo-labels, likely capturing more true objects but also introducing significant noise, which can prevent learning stability and potential performance.

As shown in Figure 6.1a, the optimal threshold was found to be around 0.7. Moving this value in either direction generally leads to decreased performance. There were slight increases on the ReMixMatch and MixMatch with a threshold of 0.6, signaling that both frameworks are capable of leveraging a lower threshold compared to FixMatch. Note that when a confidence threshold is mentioned, it refers to the base threshold of the Adaptive Filter, which can be slightly modified depending on the image, as seen in Formula (5.2).



(a) Ablation study results on the foreground threshold used in the Adaptive Filter.

(b) Ablation study on λ, the contribution coefficient of unlabeled loss per epoch.

Figure 6.1: Ablation studies on the hyperparameters foreground threshold and unlabeled loss coefficient.

Unlabeled Loss Coefficient A commonly studied hyperparameter in semi-supervised learning is the weight assigned to the unlabeled loss during training. This coefficient controls the contribution of the unlabeled data to the overall loss function, effectively balancing the influence of labeled and unlabeled samples in each training epoch.

Some prior works explore dynamic strategies, such as gradually increasing the weight over time to allow the model to first learn from only labeled data before incorporating potentially noisy unlabeled signals. However, this study focuses on a fixed weight throughout training. The Figure 6.1b shows that increasing the weight tends to degrade performance, likely due to the model being influenced too heavily by noisy gradients. The best results were achieved with a more conservative weight, which provides more stable updates and reduces the risk of overfitting to unreliable pseudo-labels.

Summary This ablation study provided valuable insights into the key components of each framework. Starting with the common Adaptive Filter, it is shown to be essential across all methods. In MixMatch, increasing the number of augmented views, K, did not improve performance, and the Clustering Aggregation mechanism could potentially be replaced with a simpler approach. In contrast, ReMixMatch benefited from a higher K, though this comes at the cost of increased training time. Distribution Alignment failed to impact performance in ReMixMatch, and it is likely to have a stronger impact on datasets with a larger number of object classes. For both MixMatch and ReMixMatch, the MixUp augmentation proved to be a core component, where removing it resulted in a significant performance drop. In the case of FixMatch, the study showed that the Cutout augmentation is not strictly necessary to achieve good results. It was also observed that applying a greater variety of moderate augmentations is more beneficial than fewer but stronger

ones. Lastly, two important common hyperparameters were also studied: the foreground threshold and the unlabeled loss coefficient. The optimal threshold was found to be around 0.7, although some frameworks performed well with 0.6. For the unlabeled loss coefficient, keeping it fixed at 1 provided the most stable training, while increasing it tended to reduce performance due to the influence of noisy gradients.

6.6 LiDAR Projections

This section presents the results of applying the same SSL frameworks developed in this work for RGB images to 2D LiDAR range-view projections. All the experimental setups remained the same compared to the RGB experiments to allow a direct and meaningful comparison. Table 6.6 summarizes the overall results.

While SSL frameworks consistently provided performance gains in the LiDAR, the magnitude of the improvements was reduced compared to their performance gains on RGB image data. Specifically, the best mAP gain on RGB images reached up to 9%, whereas the gains on LiDAR projections peaked at only 4% gain.

Table 6.6: KITTI LiDAR frameworks mAP (%) at varying Labeling Percentages

	KITTI LiDAR			
Method	1%	2%	5%	10%
Supervised	0.1663	0.2141	0.2843	0.3433
FixMatch	0.1966	0.2341	0.3220	0.3659
MixMatch	0.1822	0.2233	0.3063	0.3660
ReMixMatch	**0.2171**	**0.2424**	**0.3341**	**0.3833**

Frameworks Comparison The relative performance differences between SSL frameworks were less pronounced on LiDAR data compared to RGB experiments. While ReMixMatch remained the best-performing method across all labeling percentages, the margin over FixMatch and MixMatch was smaller. Interestingly, MixMatch underperformed relative to expectations, particularly at lower label ratios, where it performed worse compared to FixMatch. However, in the 10% labeled setting, it closed the gap, achieving performance comparable to FixMatch. Nevertheless, all frameworks are able to provide relevant performance gains compared to the baseline.

Reduced Performance Gains One of the main challenges when applying SSL techniques to 2D LiDAR projections is the significantly lower information density of the input data compared to RGB images. When observing a set of examples of the LiDAR projection images, some visual clues, such as textures, shadows, and borders, are missing. As objects are farther from the sensor, the density of points decreases, leading to sparser and noisier representations. This affects both supervised and semi-supervised learning, limiting the potential of the unlabeled data.

False Positives On the other hand, when comparing the model predictions on the RGB images and the LiDAR images, in LiDAR representations, there are fewer hallucinations. For example, in Figure 6.2, it is possible to note the object predictions in the sky of the image, which does not happen in the LiDAR, as there are no signals there. This suggests that LiDAR projections have better spatial reasoning, reducing absurd predictions. However, this robustness comes with a trade-off, the reduced visual richness also makes object classification more difficult. As also shown in the figure, the model correctly identifies the presence of an object on the left side, but misclassifies a trash can as a car. In summary, while LiDAR is effective at localizing objects, it struggles with fine-grained classification when used in isolation.

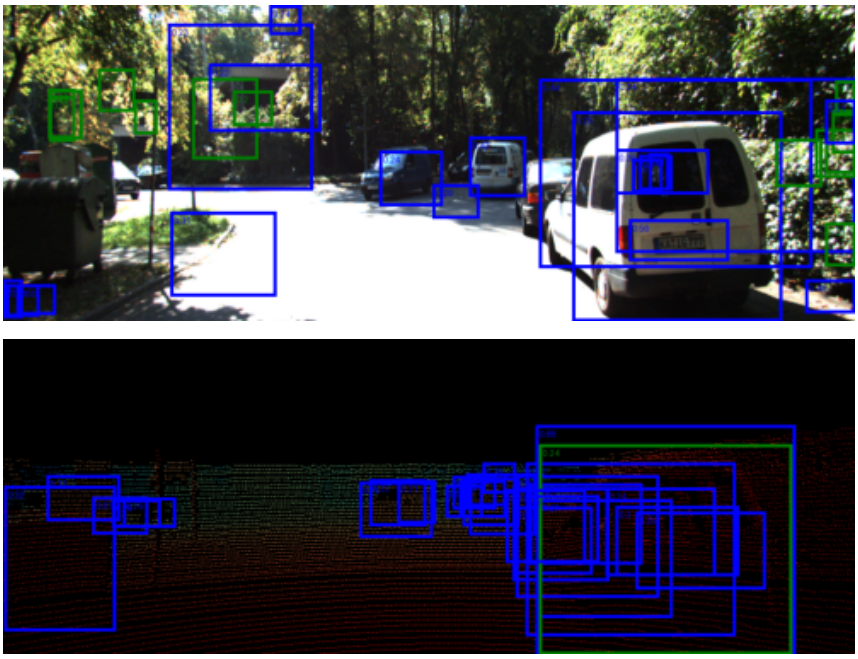


Figure 6.2: Comparison of models' predictions on RGB images versus LiDAR projections.

Chapter 7

Conclusion

7.1 Contributions

This thesis investigated the adaptation of semi-supervised learning frameworks designed initially for classification tasks to object detection. The study focused on three state-of-the-art classification methods, FixMatch, MixMatch, and ReMixMatch, that were adapted to object detection and evaluated under varying levels of labeled data availability ranging from 1% to 10%. The experiments demonstrated that all three frameworks consistently outperform fully supervised baselines, confirming the potential of leveraging large quantities of unlabeled data to reduce dependency on manual annotations. Among the methods, ReMixMatch proved to be the most robust, achieving the highest improvements in mAP across datasets and labeling regimes. Nevertheless, MixMatch was particularly effective in extremely low-label settings, while FixMatch showed competitive performance despite its relative simplicity.

Ablation studies provided further insights into which components were responsible for the performance improvements. Notably, augmentation strategies, especially MixUp, were highlighted to impact the performance gains, and the findings suggest that increasing the diversity of moderate augmentations results in greater benefits than merely intensifying augmentation strength.

Additionally, extending the analysis to 2D LiDAR range-view projections revealed that, although SSL continues to yield gains over supervised baselines, the magnitude of improvement is lower compared to RGB images. This reduced improvement is justified by the lower information density and visual richness of LiDAR data, which limits the potential improvements of SSL. However, LiDAR inputs exhibited fewer false positives, highlighting their advantage in spatial reasoning and object boundary delineation.

The main contributions of this thesis lies in the modification of an object detection model, where the source code was modified to incorporate the loss mask calculation and further application on the loss functions, the empirical evaluation of SSL frameworks originally developed for image classification when applied to object detection, followed by the conceptual frameworks developed when adapting the methods. Furthermore, the work was expanded beyond RGB images

by implementing and analyzing these frameworks in a real-world autonomous driving scenario using LiDAR range-view projections, highlighting practical applicability in multimodal perception systems. All the source code is available at https://github.com/alex-bsT/dense-ssl.

7.2 Future Work

Both RGB images and LiDAR data were explored separately, demonstrating substantial performance gains when leveraging SSL. A promising future direction is the fusion of these two sensor modalities to further enhance detection performance. One possible approach could be inspired by Co-Training [40], where two independent models, one for each sensor modality, are trained simultaneously, but their predictions are fused during inference and training. This mechanism could help reduce false positives by leveraging cross-modal agreement. A more ambitious alternative would be to modify the neural network architecture to receive both modalities as input. Specifically, this means extending the input from the standard three RGB channels to four channels, where the additional channel encodes depth information derived from LiDAR. Although more challenging, it could potentially be more efficient and yield better results compared to having to train and run two separate models.

Furthermore, a broader research direction could be understanding the impact of data augmentations in SSL for object detection. A key finding of this thesis is that augmentations are in the core of the effectiveness and generalizability of SSL models, e.g., MixUp demonstrated to be key in the performance gains from the MixMatch and ReMixMatch adaptations. Future work could systematically evaluate and compare a broader range of augmentation strategies, with some notable candidates including the Mosaic augmentation, first introduced in YOLOv4 [58], CutMix [51] which is the predecessor of Mosaic, InstaBoost [59], which is originally created for semantic segmentation although potentially adaptable to object detection and PuzzleMix [60] which is a variation of MixUp.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[2] Florian Alexander Schmidt. *Crowdsourced production of AI Training Data: How human workers teach self-driving cars how to see*. Tech. rep. 155. Hans-Böckler-Stiftung, Düsseldorf, 2019.

[3] Eric Arazo et al. *Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning*. 2020.

[4] David Berthelot et al. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[5] Kihyuk Sohn et al. "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 596–608.

[6] Yanyang Wang, Zhaoxiang Liu, and Shiguo Lian. "Semi-supervised Object Detection: A Survey on Recent Research and Progress". In: *arXiv preprint arXiv:2306.14106* (2023).

[7] Hongyu Zhou et al. "Dense Teacher: Dense Pseudo-Labels for Semi-supervised Object Detection". In: *European Conference on Computer Vision*. Springer. 2022, pp. 35–50.

[8] Xinjiang Wang et al. "Consistent-Teacher: Towards Reducing Inconsistent Pseudo-Targets in Semi-Supervised Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 3240–3249.

[9] Binbin Chen et al. "Label Matching Semi-Supervised Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14381–14390.

[10] David Berthelot et al. "ReMixMatch: Semi-supervised Learning with Distribution Alignment and Augmentation Anchoring". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HklkeR4KPB.

[11] Yibo Sun, Zhe Sun, and Weitong Chen. "The evolution of object detection methods". In: *Engineering Applications of Artificial Intelligence* 133 (2024), p. 108458.

[12] Jianxin Wu. "Introduction to Convolutional Neural Networks". In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.

[13] Keiron O'shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[14] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[15] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[16] Andrew G Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[17] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2117–2125.

[18] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2016), pp. 1137–1149.

[19] J Redmon. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[20] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229.

[21] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97.

[22] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017).

[23] Zhi Tian et al. "FCOS: Fully Convolutional One-Stage Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9627–9636.

[24] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988.

[25] Hamid Rezatofighi et al. "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 658–666.

[26] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *arXiv preprint arXiv:1803.07728* (2018).

[27] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer vision–ECCV 2014: 13th European conference, Zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*. Springer. 2014, pp. 740–755.

[28] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020), p. 125.

[29] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*. 2018.

[30] Ekin D Cubuk et al. "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 702–703.

[31] Jesper Engelen and Holger Hoos. "A survey on semi-supervised learning". In: *Machine Learning* 109 (Feb. 2020). DOI: 10.1007/s10994-019-05855-6.

[32] Xiangli Yang et al. "A Survey on Deep Semi-Supervised Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 35.9 (2022), pp. 8934–8954.

[33] YCAP Reddy, P Viswanath, and B Eswara Reddy. "Semi-supervised learning: A brief review". In: *Int. J. Eng. Technol* 7.1.8 (2018), p. 81.

[34] Jothi Prakash and Dr Nithya. "A Survey On Semi-Supervised Learning Techniques". In: *International Journal of Computer Trends and Technology* 8 (Feb. 2014). DOI: 10.14445/22312803/IJCTT-V8P105.

[35] Zhu Xiaojin. “Semi-Supervised Learning Literature Survey”. In: *Computer Sciences TR* 1530 (2008), pp. 60–60.

[36] Antti Rasmus et al. “Semi-Supervised Learning with Ladder Networks”. In: *Advances in Neural Information Processing Systems* 28 (2015).

[37] Pascal Vincent et al. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion.” In: *Journal of Machine Learning Research* 11.12 (2010).

[38] Samuli Laine and Timo Aila. “Temporal Ensembling for Semi-Supervised Learning”. In: *arXiv preprint arXiv:1610.02242* (2016).

[39] Dong-Hyun Lee et al. “Pseudo-Label: The simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks”. In: *Workshop on Challenges in Representation Learning, ICML*. Vol. 3. Atlanta. 2013, p. 896.

[40] Siyuan Qiao et al. “Deep Co-Training for Semi-Supervised Image Recognition”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 135–152.

[41] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems* 27 (2014).

[42] Yankun Chen et al. “Auto-Encoding Variational Bayes”. In: *Cambridge Explorations in Arts and Sciences* 2 (Feb. 2024). DOI: 10.61603/ceas.v2i1.33.

[43] Xiaojin Zhu and Zoubin Ghahramani. “Learning from Labeled and Unlabeled Data with Label Propagation”. In: *ProQuest Number: Information to All Users* (2002).

[44] Alessandro Pieropan, Hossein Azizpour, Atsuto Maki, et al. “Dense FixMatch: a simple semi-supervised learning method for pixel-wise prediction tasks”. In: *arXiv preprint arXiv:2210.09919* (2022).

[45] Pratima Upretee and Bishesh Khanal. “FixMatchSeg: Fixing Fixmatch for Semi-Supervised Semantic Segmentation”. In: *arXiv preprint arXiv:2208.00400* (2022).

[46] Boyuan Sun et al. “CorrMatch: Label Propagation via Correlation Matching for Semi-Supervised Semantic Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 3097–3107.

[47] Yassine Ouali, Céline Hudelot, and Myriam Tami. “Semi-Supervised Semantic Segmentation with Cross-Consistency Training”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12674–12684.

[48] Yen-Cheng Liu, Chih-Yao Ma, and Zsolt Kira. “Unbiased Teacher v2: Semi-supervised Object Detection for Anchor-free and Anchor-based Detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9819–9828.

[49] Binghui Chen et al. “Dense Learning based Semi-Supervised Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4815–4824.

[50] JongMok Kim et al. “MUM: Mix Image Tiles and Unmix Feature Tiles for Semi-Supervised Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14512–14521.

[51] Sangdoo Yun et al. “CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.

[52] Kihyuk Sohn et al. “A Simple Semi-Supervised Learning Framework for Object Detection”. In: *arXiv preprint arXiv:2005.04757* (2020).

[53] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[54] Thanh Vu et al. “Supervision Interpolation via LossMix: Generalizing Mixup for Object Detection and Beyond”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 2024, pp. 5280–5288.

[55] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. *Ultralytics YOLO*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics.

[56] Jason Ansel, Edward Yang, and He et al. “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation”. In: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS ’24)*. ACM, Apr. 2024. URL: https://docs.pytorch.org/assets/pytorch2-2.pdf.

[57] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115 (2015), pp. 211–252.

[58] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: *arXiv preprint arXiv:2004.10934* (2020).

[59] Hao-Shu Fang et al. “InstaBoost: Boosting Instance Segmentation via Probability Map Guided Copy-Pasting”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 682–691.

[60] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. “Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5275–5285.