FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Exploring Label Efficiency with Semi-Supervision and Self-Supervision Methods

Francisco Gonçalves Cerqueira

Mestrado em Engenharia Informática e Computação

Supervisor: Ricardo Pereira de Magalhães Cruz

July 16, 2024

Exploring Label Efficiency with Semi-Supervision and Self-Supervision Methods

Francisco Gonçalves Cerqueira

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: João Pedro Carvalho Leal Mendes Moreira
External Examiner: Hélder Filipe Pinto de Oliveira
Supervisor: Ricardo Pereira de Magalhães Cruz

July 16, 2024

Resumo

Os modelos de deep learning requerem um volume colossal de dados para uma generalização eficaz e para alcançar alto desempenho, envolvendo não apenas a recolha de dados, mas, mais criticamente, o processo de anotação de dados. Esta tarefa intensiva em mão-de-obra de anotar cada elemento nos dados, seja veículos, pedestres ou outras entidades, exige uma força de trabalho substancial. Consequentemente, a quantidade de dados não-anotados normalmente supera a quantidade de dados anotados, e torna-se vantajoso encontrar maneiras de aproveitar este recurso não utilizado.

Há duas grandes famílias de abordagens que enfrentam este desafio incorporando dados não-anotados durante o processo de treino sob a suposição de que a previsão de um rótulo deve permanecer consistente, independentemente das transformações aplicadas à amostra correspondente: as aprendizagens semi-supervisionadas e auto-supervisionadas (self-supervised). Por exemplo, ao não ter certeza se uma imagem representa um peão ou um ciclista, é razoável esperar que uma leve rotação da imagem não altere a sua classificação. Métodos de semi-supervisão necessitam de dados anotados, mas conseguem fazer também uso de dados não-anotados para melhorar o treino; nomeadamente usando as previsões do modelo para estes dados cujas anotações são desconhecidas para gerar pseudo-anotações nos casos em que o modelo tenha grande confiança nas mesmas. Abordagens de auto-supervisão, por outro lado, operam na ausência de dados anotados, utilizando tarefas secundárias para criar anotações artificiais a partir dos próprios dados, o que ajuda o modelo a aprender representações úteis. Por exemplo, o modelo pode ser treinado para prever o ângulo de rotação aplicado a uma imagem, aprendendo assim a extrair características importantes. Este tipo de abordagens pode posteriormente utilizar os dados anotados para ajustar o modelo, ajudando-o a aprimorar ainda mais sua compreensão e representação dos dados, melhorando, em última instância, o seu desempenho em tarefas subsequentes.

O estudo visa fazer contribuições inovadoras através de uma análise comparativa abrangente de métodos de semi-supervisão e auto-supervisão, com o objetivo principal de avaliar a sua aplicação prática e desempenho na área de Condução Autónoma, adaptando-os para tarefas mais particulares deste domínio, como a segmentação semântica. Vários conjuntos de dados públicos estão disponíveis, sendo o KITTI um dos mais populares na literatura. Como objetivo adicional, foi desenvolvida uma framework de software, permitindo que outros projetos beneficiem destas técnicas.

Os resultados indicam que estes métodos melhoram significativamente a eficiência das anotações, sendo a aprendizagem semi-supervisionada particularmente adequada em cenários com quantidades moderadas de dados rotulados e a aprendizagem auto-supervisionado em situações com grandes volumes de dados não anotados. No contexto de Condução Autónoma, esses métodos demonstraram melhorias em segmentação semântica, exibindo robustez e adaptabilidade. Este estudo conclui que a integração destes métodos pode reduzir substancialmente a dependência de dados anotados, abrindo caminho para modelos mais eficientes e escaláveis. O código encontra-se disponível em `https://github.com/xico2001pt/exploring-label-efficiency`.

Palavras-chave: aprendizagem semi-supervisionada · aprendizagem uto-supervisionada · label efficiency · condução autónoma · visão por computador · deep learning · redes neuronais · segmentação semântica

Abstract

Deep learning models demand a colossal volume of data for effective generalization and achieving superior performance, involving not only data collection but, more critically, the process of data annotation. This labor-intensive task of labeling each element within the data, be it vehicles, pedestrians, or other entities, requires a substantial workforce. Consequently, the amount of unlabeled data typically surpasses the amount of labeled data, and it becomes advantageous to find ways to leverage this untapped resource.

There are two major families of approaches that tackle this challenge by incorporating unlabeled data during the training process under the assumption that the prediction of a label should remain consistent despite transformations applied to the corresponding sample: Semi-Supervised and Self-Supervised Learning. For example, while not sure whether an image depicts a pedestrian or a cyclist, it is reasonably expected that a slight rotation of the image should not alter its classification. Semi-supervision methods require annotated data but can also utilize unlabeled data to improve training; specifically by using the model's predictions for these unlabeled data to generate pseudo-labels in cases where the model has high confidence. Self-supervision frameworks, on the other hand, rely solely on unlabeled data, employing pretext tasks to create artificial labels from the data itself, which help the model learn useful representations. For instance, the model might be trained to predict the rotation angle applied to an image, thereby learning to extract meaningful features. Approaches like Supervised Learning can later utilize the annotated data to fine-tune the model, helping it further refine its understanding and representation of the data, ultimately improving its performance on downstream tasks.

The study aims to make innovative contributions through an extensive comparative analysis of semi-supervision and self-supervision methods, with the primary objective of assessing their practical application and performance within the challenging domain of Autonomous Driving, adapting them for more particular tasks, such as semantic segmentation. Several public datasets are available, with KITTI being one of the most popular in research. As an additional objective, a practical software framework was developed, enabling other projects to benefit from these techniques.

The results indicate that these methods significantly improve label efficiency, with semi-supervised learning particularly adequate in scenarios with moderate amounts of labeled data and self-supervised learning in situations with large volumes of unlabeled data. In Autonomous Driving, these methods demonstrated marked improvements in semantic segmentation, showcasing robustness and adaptability. This study concludes that integrating these methods can substantially reduce the dependency on labeled data, paving the way for more efficient and scalable models. The code is available at `https://github.com/xico2001pt/exploring-label-efficiency`.

Keywords: semi-supervised learning · self-supervised learning · label efficiency · autonomous driving · computer vision · deep learning · artificial neural networks · semantic segmentation

Acknowledgements

Completing this thesis has been a journey filled with support, encouragement, and more than a few late nights. First and foremost, I want to express my deepest gratitude to my family. To my Mom and Dad, thank you for your unwavering love, belief in me, even when I doubted myself, and the countless sacrifices you made. To my sister and cousin, your sense of humor has been my refuge in moments of stress. And to the rest of my family, I am eternally grateful for the countless ways you've all been there for me.

A heartfelt thank you to my supervisor, Ricardo Pereira de Magalhães Cruz, whose guidance, wisdom, and dedication have been invaluable. Your insights and constructive critiques have pushed me to refine my work and strive for excellence. Your mentorship has been instrumental in shaping my academic journey, and for that, I am profoundly thankful.

To the incredible friends I've made over the past five years at FEUP, you have been my second family. Thank you for the laughs, the late-night work sessions, and the shared moments of triumph and despair. You made this experience not only bearable but genuinely enjoyable. I couldn't have asked for better companions to share this chapter of my life.

Finally, to my furry companions who have provided comfort and joy throughout this journey: Daisy, my loyal dog, and all my cats, both past and present. Your companionship and occasional disruptions have reminded me to take breaks and enjoy the simple things in life.

Each one of you, in your unique way, has positively influenced this journey, and I am forever grateful.

Francisco Gonçalves Cerqueira

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

"Be clearly aware of the stars and infinity on high.
Then life seems almost enchanted after all."

Vincent Van Gogh

Contents

List of Figures

List of Tables

Abbreviations

AUROC	Area Under the Receiver Operating Characteristic Curve
CE	Cross-entropy
CNN	Convolutional Neural Network
CV	Computer Vision
DSC	Dice similarity coefficient
EMA	Exponential Moving Average
FCN	Fully Convolutional Network
GAN	Generative Adversarial Network
IoU	Intersection over Union
ML	Machine Learning
MAE	Mean Absolute Error
MSE	Mean Squared Error
NLP	Natural Language Processing
PCA	Principal Component Analysis
PSPNet	Pyramid Scene Parsing Network
ReLU	Rectified Linear Unit
SelfSL	Self-Supervised Learning
SemiSL	Semi-Supervised Learning
SL	Supervised Learning
SVM	Support Vector Machine
TL	Transfer Learning
UL	Unsupervised Learning
VGG	Visual Geometry Group
WSL	Weakly-Supervised Learning

Chapter 1

Introduction

1.1 Context

Autonomous driving has emerged as a transformative paradigm in the automotive industry, driven by the pursuit of enhancing road safety and providing a more enriching driving experience. In recent years, considerable research and development efforts have been dedicated to equipping vehicles with perceptive systems capable of autonomous decision-making.

The Society of Automotive Engineers has established a widely adopted standard to classify the levels of automation in autonomous driving systems [1]. This framework comprises six levels, ranging from Level 0 to Level 5. Level 0 signifies a vehicle without driving automation, relying entirely on the driver for operation. Progressing through Levels 1-2, autonomous driving systems begin to assist human drivers, but the drivers remain responsible for monitoring the environment. Only in Levels 3-5 does the system take over the monitoring activity, achieving varying degrees of automation. Figure 1.1 represents a visualization of these levels.

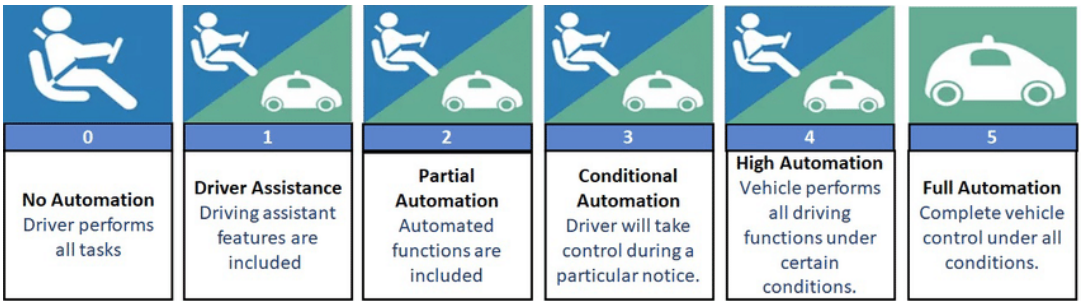


Figure 1.1: SAE J3016 levels of driving automation. Source: [2]

As this industry progresses toward achieving full automation at Level 5, where there is no necessary interaction between the driver and the system, artificial intelligence and deep learning have become indispensable tools, demanding large quantities of training data to refine algorithms and ensure the robustness of autonomous driving systems.

1.2 Motivation

Traditional deep learning methods assume that all data used for training is labeled, and this assumption becomes a bottleneck in the context of autonomous driving, as the demand for extensive labeled data, especially as autonomy levels increase, poses challenges in terms of annotation speed, cost, and susceptibility to errors [3]. This was particularly evident in the project **THEIA – Automated Perception Driving**[1], a joint initiative involving University of Porto, INESC TEC and Bosch Portugal. The primary objective of this collaboration was to explore and create intelligent perception algorithms for Autonomous Driving. However, the process of collecting the dataset within this project encountered significant delays, leading to an overwhelming workload. Multiple iterations were required to ensure the effective gathering of data. Despite earnest efforts, the precision and accuracy of annotations were not consistently achieved, posing additional challenges to the project. This limitation underscores the need for innovative approaches to effectively leverage unlabeled data, reducing dependency on traditional, labor-intensive annotation processes.

1.3 Objectives

This research aims to delve into state-of-the-art semi-supervision and self-supervision methods, tailoring their application to the specific demands of autonomous driving. These algorithms were initially evaluated on datasets commonly used in this literature, specifically CIFAR10 [4] and SVHN [5]. They were then adapted for a more prominent task in autonomous driving, namely semantic segmentation, using KITTI [6] and Cityscapes [7]. The intention behind addressing this type of task is that it represents a gradual step beyond classification tasks – in fact, it can be seen as pixel-by-pixel classification. Nevertheless, some special considerations were necessary, particularly in the application of data augmentation, which is a significant component of these techniques.

The primary focus lies in conducting thorough experiments to assess and compare these methodologies, dissecting their strengths and weaknesses. Beyond the experimental phase, the goal is to construct a software framework that implements pertinent losses associated with these techniques and serves as a valuable baseline for broader applications across various projects and fields.

In essence, this dissertation strives to confront the challenges stemming from the scarcity of labeled data, particularly in the realm of autonomous driving. Adapting these methods to tasks intrinsic to this domain, such as semantic segmentation, stands as a focal point, with a dedicated performance evaluation.

1.4 Main Contributions

This dissertation aims to make several key contributions to the fields of deep learning and autonomous driving:

[1] https://noticias.up.pt/2021/12/06/u-porto-bosch-projeto-de-investigacao-28-milhoes-de-euros/

1. Review and summary of Semi-Supervised and Self-Supervised Learning, their categories, and respective methods.
2. A comparative study evaluating the effectiveness of these methods. This includes assessing the label efficiency of these approaches and determining their ability to enhance model performance with limited annotated data.
3. Implementation and adaptation of these methods to address the specific challenges of autonomous driving.
4. Development of a software framework containing relevant losses tailored for these methods and auxiliary functions, allowing broader accessibility.

1.5 Document Structure

The remainder of this document is structured as follows:

- **Chapter 2** provides a comprehensive background on deep learning for computer vision and its learning paradigms. It also covers classification, semantic segmentation, and data augmentation topics.
- **Chapter 3** delves into the category-wise review of Semi-Supervised Learning, exploring its categories and methods.
- **Chapter 4** provides a review of Self-Supervised Learning, covering generative, contrastive, and adversarial methods. It also discusses specific techniques of the contrastive category.
- **Chapter 5** outlines the methodology used for conducting all the experiments, including a detailed description of the datasets employed, the process of splitting unlabeled data, and the training cycle of each used learning paradigm. Additionally, it thoroughly explains the implementations and adaptations of the experiments.
- **Chapter 6** presents the results of the experiments and provides a comprehensive discussion. It compares the performance across different methods, addressing observed trends or anomalies.
- Concluding the thesis, **Chapter 7** summarizes the research's key findings, contributions, and implications. It also outlines potential avenues for future work and improvements in the field.

Chapter 2

Background Knowledge

2.1 Deep Learning for Computer Vision

2.1.1 Classification

The classification task within the domain of Machine Learning (ML) constitutes a fundamental process that aims to categorize data instances into distinct predefined classes or categories predicated upon the discernible patterns and features inherent in the dataset. It stands as a cornerstone in Computer Vision (CV), Natural Language Processing (NLP), and various other domains where data analysis and decision-making based on categorization are crucial.

Typically, the classification process involves employing one-hot encoding, where each class is represented by a probability indicating the likelihood of a particular instance belonging to that class. This method enables models to assign scores to each class based on their predictions.

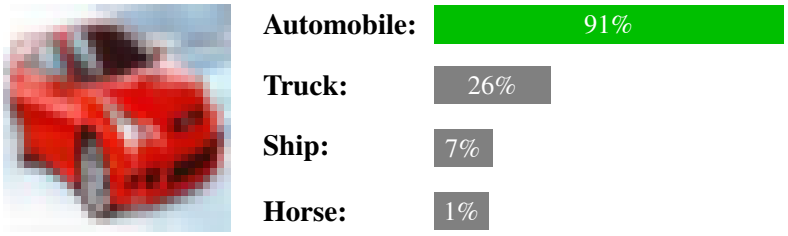


Figure 2.1: Image classification example, showcasing a CIFAR-10 [4] dataset image of an automobile alongside its classification probabilities, indicating the likelihood of the image being identified within a set of categories.

2.1.1.1 Architectures

Classification models often employ diverse architectures designed to extract meaningful features and make accurate predictions. These architectures typically follow a pattern of spatial reduction while increasing the depth of extracted features.

AlexNet Introduced by Krizhevsky et al. (2012) [8], AlexNet revolutionized Convolutional Neural Networks (CNNs) for high-resolution image recognition in the ImageNet [9] competition, playing a pivotal role in popularizing deep learning for image classification tasks. It introduced concepts like deeper network layers, parallel pathways, and sophisticated pooling techniques to enhance performance. AlexNet's architecture, depicted in Figure 2.2, consists of eight layers: five convolutional and three fully connected. Key innovations include the use of Rectified Linear Units (ReLUs) for faster training compared to saturating neurons (e.g., *tanh*) and multi-GPU support, reducing training time and enabling larger model sizes. Additionally, it introduced overlapping pooling, reducing errors and making overfitting more challenging.

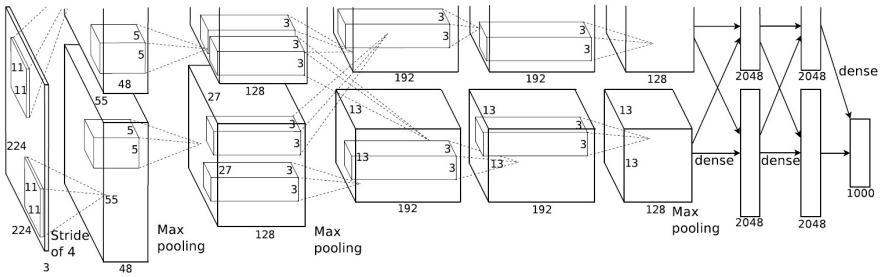


Figure 2.2: AlexNet architecture. Source: [8]

VGG The Visual Geometry Group (VGG) [10] is a convolutional neural network architecture consisting of 16 or 19 layers with small 3×3 convolutional filters, followed by max-pooling layers. After the convolutional and pooling layers, VGG integrates fully connected layers to amalgamate extracted features and make final predictions. The architecture's uniformity and simplicity, represented in Figure 2.3, contribute to its widespread adoption and understanding within the deep learning community. Despite its depth, VGG is known for its ease of implementation, serving as a foundational model for various computer vision applications, showcasing impressive performance on image classification benchmarks like ImageNet [9].

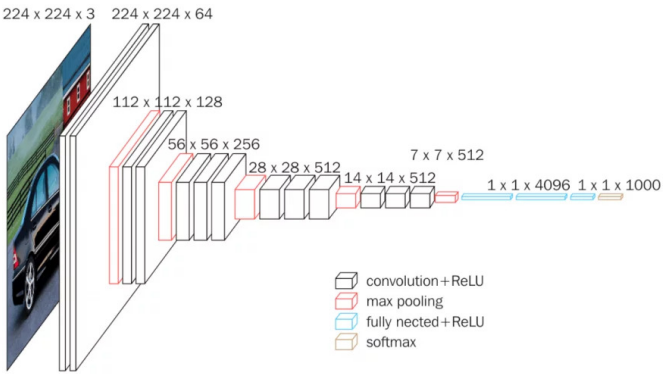


Figure 2.3: VGG16 architecture. Source: [11]

GoogLeNet GoogLeNet [12] diverges from conventional architectures like AlexNet and VGG by introducing the concept of inception modules, depicted in Figure 2.4, which are composed of multiple convolutional layers of different sizes running in parallel. This design aims to capture features at various scales while maintaining computational efficiency. Unlike sequential architectures, GoogLeNet employs a network with multiple branches, allowing for parallel information processing. This model also integrates global average pooling, reducing the number of parameters significantly.

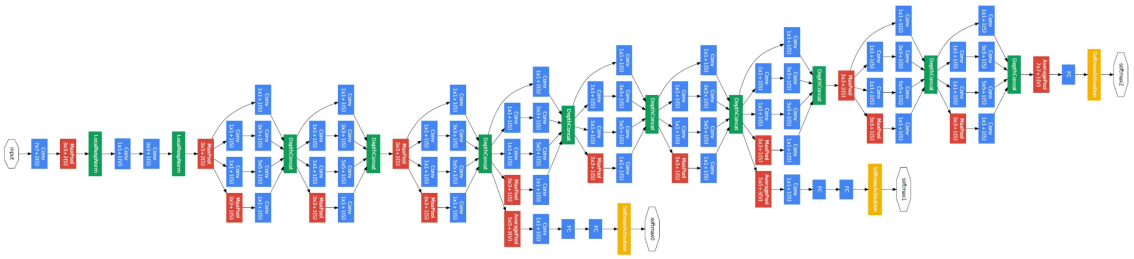


Figure 2.4: GoogLeNet architecture. Source: [12]

ResNet ResNet [13] stands out from traditional architectures due to its ingenious use of residual connections, addressing the vanishing gradient problem. This approach introduces skip connections that allow the flow of information through the network layers, enabling the training of deep models while mitigating degradation issues. The residual blocks, depicted in Figure 2.5, contain shortcut connections, allowing the learning of residual mappings instead of the original mappings. ResNet architectures achieve exceptional performance without suffering from diminishing accuracy with increasing depth, unlike previous models.

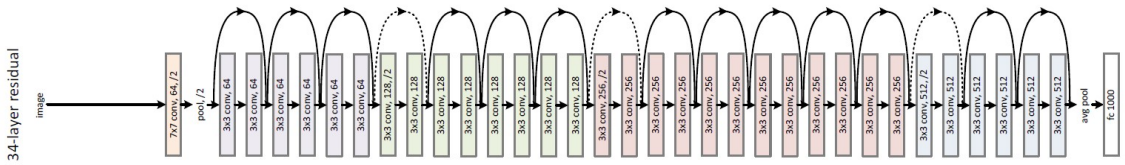


Figure 2.5: ResNet architecture. The dotted shortcuts increase dimensions. Source: [13]

2.1.1.2 Metrics

While evaluating classification models, various metrics serve as benchmarks to assess their performance.

Accuracy One of the most straightforward metrics is accuracy, representing the ratio of correctly predicted instances to the total number of predicted instances. It is calculated using the formula

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}. \tag{2.1}$$

Precision Precision measures the accuracy of predictions of the positive class. It is defined as the ratio of correctly predicted positive observations to the total predicted positives, formally denoted as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{2.2}$$

Recall Recall (also known as sensitivity) exhibits the model's ability to find all the relevant cases within the dataset. It calculates the ratio of correctly predicted positive observations to all actual positives, being defined as

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{2.3}$$

F1 Score A "harmonic mean" of precision and recall is designated by F1 Score, providing a balance between these two metrics. It can be calculated using the formula

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.4}$$

AUROC The Area Under the Receiver Operating Characteristic Curve (AUROC), also abbreviated as AUC or ROC AUC, summarizes the Receiver Operating Characteristic curve into a numerical value, measuring how well a model distinguishes between different classes by examining its true positive and false positive rates across various classification thresholds. An AUROC score of 1 implies perfect discrimination, meaning the model perfectly separates positive and negative instances regardless of the threshold chosen, whereas a score of 0.5 indicates that the model performs no better than random chance guessing.

2.1.1.3 Losses

Various loss functions are employed to gauge the disparity between predicted values (q) and actual values (p), guiding the model toward optimal predictions. Here, $N_{\mathcal{D}}$ represents the number of samples in the dataset, p_i denotes the actual class probability vector for the i-th sample, and q_i signifies the predicted class probability vector for the i-th sample.

Cross-entropy Cross-entropy (CE) is a widely used loss that measures the divergence between the predicted values and the actual distribution. For a multi-class classification problem, the CE loss is given as

$$\mathrm{H}(p,q) = -\frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \sum_{c=1}^{N_{\mathcal{C}}} p_{ic} \cdot \log(q_{ic}), \tag{2.5}$$

where $N_{\mathcal{C}}$ denotes the number of classes, and p_{ic} and q_{ic} represent the true value and the predicted probability, respectively, for sample i belonging to class c.

Mean Squared Error The loss of semi-supervision methods often operates in the latent space, which is continuous, not the output space; therefore, Mean Squared Error (MSE) is a popular

loss within these methods. It calculates the average of the squared differences between predicted values and actual values, described as

$$\mathrm{MSE}(p,q) = \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \|p_i - q_i\|_2^2 . \tag{2.6}$$

Mean Absolute Error Mean Absolute Error (MAE) computes the average of the absolute differences between predicted values and true values, defined as

$$\mathrm{MAE}(p,q) = \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \|p_i - q_i\|_1 . \tag{2.7}$$

2.1.2 Semantic Segmentation

Semantic segmentation is a Computer Vision task focusing on pixel-level classification in digital images, partitioning an image into semantically meaningful regions by assigning categorical labels to individual pixels. Unlike other image recognition tasks that merely classify the entire image or detect objects within it, semantic segmentation delves into the granular level, allowing for a meticulous understanding of the image's content and structure.

Figure 2.6: Semantic Segmentation mask example from Cityscapes dataset. Source: [7]

2.1.2.1 Architectures

Typically, semantic segmentation architectures adopt an encoder-decoder structure, leveraging various neural network designs to achieve accurate pixel-level predictions. Therefore, several architectures have been developed, each with distinct characteristics and performance metrics.

Fully Convolutional Network The Fully Convolutional Network (FCN) [14] architecture, represented in Figure 2.7, introduced the concept of end-to-end learning for pixel-wise classification tasks. It revolutionized semantic segmentation by utilizing fully convolutional layers to preserve spatial information throughout the network.

U-Net U-Net is widely recognized for its use in biomedical image segmentation [15]. Its distinctive U-shaped structure, depicted in Figure 2.8, features an encoder path for downsampling

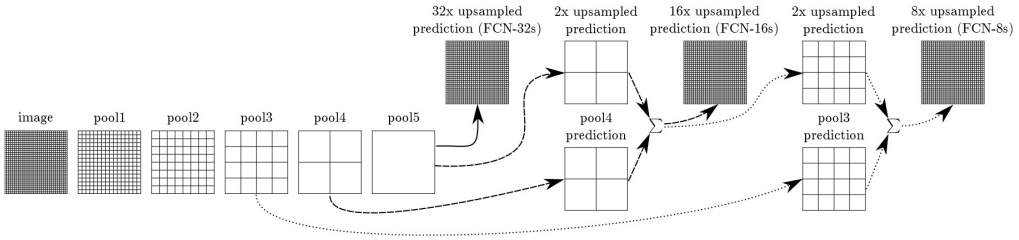


Figure 2.7: Fully Convolutional Network architecture. Source: [14]

and a symmetric decoder path with multiple upsampling layers using learnable filters. The key innovation lies in incorporating skip connections between the corresponding encoder and decoder layers to preserve location details and enable the network to capture global context accurately.

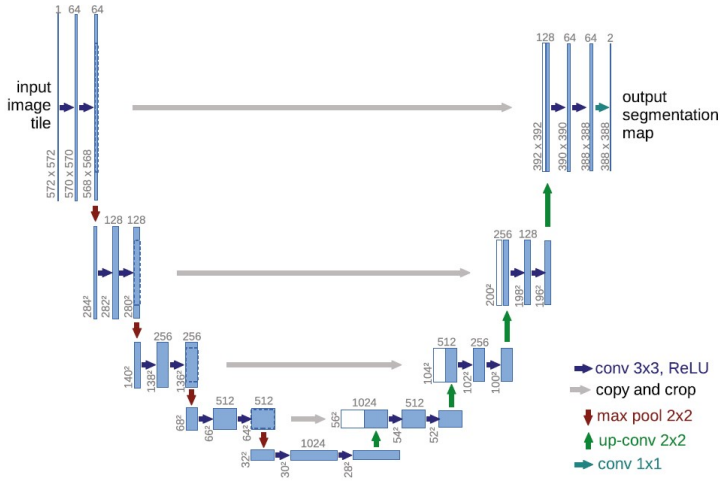


Figure 2.8: U-Net architecture. Source: [15]

SegNet Similar to U-Net, SegNet [16] (Figure 2.9) adopts an encoder-decoder structure for semantic segmentation. However, it diverges in its approach by transferring the pooling indices from each encoder layer to the respective symmetric layer in the decoder instead of the entire map feature, resulting in less memory usage. This model also introduced the concept of "max-unpooling" to revert the max-pooling operation, i.e., obtain the pooling indices used during the encoding phase to perform upsampling. This process helps to retrieve spatial information lost during the pooling operation, contributing to the reconstruction of the segmented output with finer details.

Pyramid Scene Parsing Network As doing heavy processing at every scale is quite computationally intensive, Pyramid Scene Parsing Network (PSPNet) [17] proposes the innovative use of pyramid pooling modules. These modules, represented in Figure 2.10, enable the network to gather information from various scales within an image to effectively handle diverse object sizes, capturing fine details and global context to enhance its understanding of complex scenes.

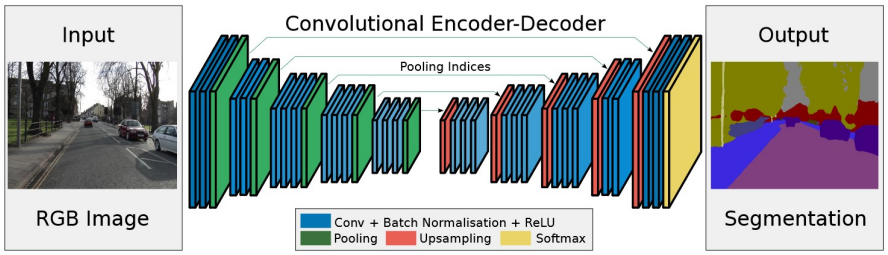


Figure 2.9: SegNet architecture. Source: [16]

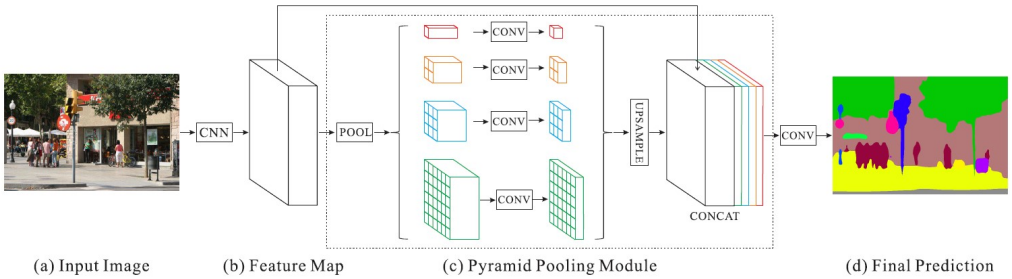


Figure 2.10: Pyramid Scene Parsing Network architecture. Source: [17]

DeepLabv3 DeepLabv3 [18] utilizes Atrous (dilated) convolutions to widen the receptive field without increasing the number of parameters. These convolutions capture local and global information at various scales by creating gaps between kernel elements. Varying the distance between each weight, designated as the dilation rate, enables the model to capture fine details with lower rates and broader context with higher rates, aiding accurate semantic segmentation by handling multi-scale features efficiently. Figure 2.11 illustrates DeepLabv3's Atrous convolution utilization, displaying the integration of diverse dilation rates for enhanced segmentation performance.

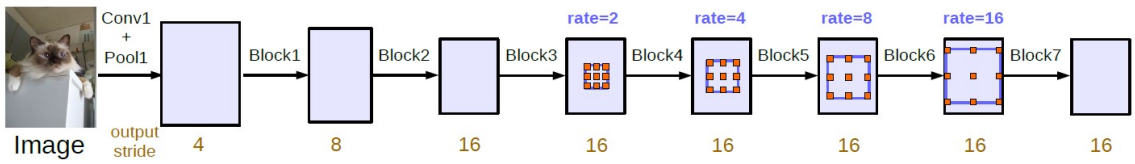


Figure 2.11: DeepLabv3 architecture. Source: [18]

Segmenter Transformers have also been applied in segmentation tasks [19]. This involves dividing an image into patches, encoding their positional information, and using an attention-based transformer to determine the probable class for each patch. While offering enhanced interpretability through attention masks, transformers entail higher computational costs when applied to images due to the extensive calculations involved in attending to multiple patches across different layers. Figure 2.12 depicts the architecture of Segmenter, a transformer model for semantic segmentation.

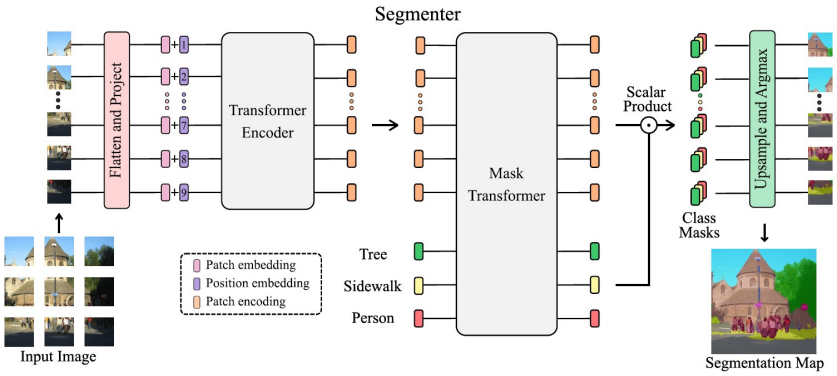


Figure 2.12: Segmenter architecture. Source: [19]

2.1.2.2 Metrics

This section presents an overview of widely utilized metrics in semantic segmentation, elucidating their strengths, limitations, and formulations.

Pixel Accuracy Pixel Accuracy is a simple metric that quantifies the proportion of correctly predicted pixels to the total number of pixels within the evaluation set, providing a basic understanding of model performance. While offering a straightforward measure of overall accuracy, this metric encounters limitations in scenarios characterized by significant class imbalances or localized errors in segmented images. Its formal definition is expressed as

$$\text{Pixel Accuracy} = \frac{\text{Number of Correctly Predicted Pixels}}{\text{Total Number of Pixels}}. \tag{2.8}$$

Jaccard Index The Jaccard Index, called Intersection over Union (IoU), is a pivotal metric measuring the overlap between predicted (A) and ground truth (B) segmentation masks. It evaluates the similarity of the areas by calculating the ratio of the intersection area to the union area, mathematically described as

$$\text{Jaccard Index} = \text{IoU} = \frac{\text{Intersection Area}}{\text{Union Area}} = \frac{|A \cap B|}{|A \cup B|}. \tag{2.9}$$

Higher IoU values indicate better agreement between the model's predictions and the actual regions of interest. This metric provides a more nuanced evaluation than Pixel Accuracy, especially in scenarios where precise delineation of boundaries and accurate localization of objects are crucial.

Dice Coefficient The Sørensen–Dice coefficient, often termed Dice similarity coefficient (DSC) or F1 score, stands as another fundamental metric in evaluating the performance of models for image segmentation tasks. It quantifies the similarity between the predicted (A) and ground truth (B) segmentation masks, emphasizing the balance between false positives and false negatives. The

formula for calculating the Dice coefficient is represented as

$$\text{Dice Coefficient} = \frac{2 \times |A \cap B|}{|A| + |B|}. \tag{2.10}$$

Like the Jaccard Index, the Dice coefficient is particularly advantageous in scenarios where precise boundary delineation and accurate localization are critical for model assessment. It mitigates the sensitivity to class imbalance by focusing on the overlap of segmented regions rather than individual pixels, offering a balanced measure of segmentation quality.

2.1.2.3 Losses

Cross-entropy and Dice Loss are two widely used loss functions for semantic segmentation. These loss functions are calculated by comparing predicted (q) and ground truth (p) values. In the context of semantic segmentation, H and W represent the height and width of the images, respectively, $N_{\mathcal{D}}$ represents the number of samples in the dataset, and p_{ijc} and q_{ijc} denote the actual value and the predicted probability, respectively, for the i-th sample of the j-th pixel belonging to class c.

Cross-entropy The Cross-entropy loss function for semantic segmentation extends the conventional CE used in classification tasks to pixel-wise prediction scenarios. It measures the dissimilarity between the predicted pixel-wise probabilities and the ground truth labels across the image. The adapted CE loss is expressed as

$$\mathrm{H}(p,q) = -\frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \sum_{j=1}^{H \times W} \sum_{c=1}^{N_{\mathcal{C}}} p_{ijc} \cdot \log(q_{ijc}). \tag{2.11}$$

Dice Loss The Dice loss [20] is a loss function adapted from the Dice coefficient metric, focusing on the overlap between predicted and ground truth segmentation masks and emphasizing the balance between false positives and false negatives. The Dice loss aims to maximize the Dice coefficient and is defined as

$$\text{Dice Loss}(p,q) = 1 - \frac{2\sum_{i=1}^{N_{\mathcal{D}}} \sum_{j=1}^{H \times W} \sum_{c=1}^{N_{\mathcal{C}}} p_{ijc} \cdot q_{ijc} + \varepsilon}{\sum_{i=1}^{N_{\mathcal{D}}} \sum_{j=1}^{H \times W} \sum_{c=1}^{N_{\mathcal{C}}} p_{ijc} + \sum_{i=1}^{N_{\mathcal{D}}} \sum_{j=1}^{H \times W} \sum_{c=1}^{N_{\mathcal{C}}} q_{ijc} + \varepsilon}, \tag{2.12}$$

where ε is a smoothing factor to prevent division by zero.

2.1.3 Data Augmentation

Data augmentation is a technique widely employed in Machine Learning and Computer Vision to artificially increase the size of a dataset by applying various transformations to the existing data. The primary goal is to enhance the model's generalization capability, reduce overfitting, and improve performance by exposing the model to diverse variations of the original data.

2.1.3.1 Categories

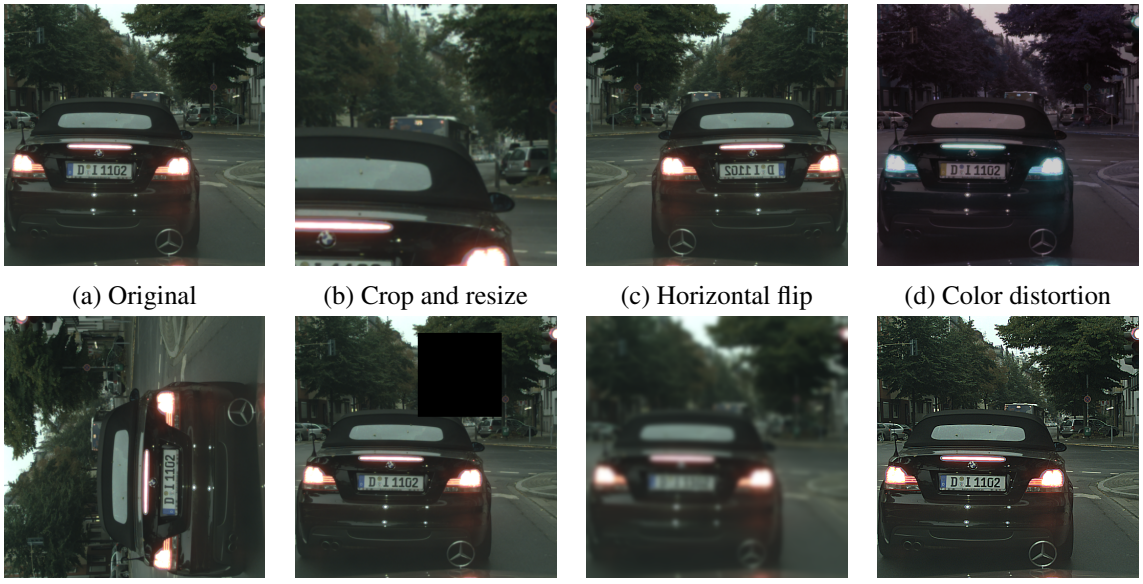
Several of the methods that will be explored divide the augmentations into two groups of intensity:

- **Weak augmentations** involve applying minor modifications to the input data without significantly changing its underlying characteristics. Examples of weak augmentations include horizontal or vertical flipping, scaling, and cropping.
- **Strong augmentations** introduce more significant modifications to the data, often leading to more diverse and varied samples. Examples of strong augmentation techniques include rotation, translation, color transformation, shearing, contrast adjustment, noise, and distortion.

2.1.3.2 Techniques

Several notable data augmentation techniques have been developed to improve the robustness and generalization of Machine Learning models.

Traditional Techniques In traditional data augmentation, the manipulation of images includes transformations in spatial structure, appearance, and quality while preserving labels, effectively generating new images to expand the dataset [21]. Spatial transformations encompass rotations, flips, scaling, and deformations. Appearance augmentation adjusts characteristics such as brightness and contrast in image intensities. Finally, image quality manipulation involves adjustments in parameters like blurriness, sharpness, and noise levels.



(a) Original (b) Crop and resize (c) Horizontal flip (d) Color distortion

(e) Rotation (f) Cutout (g) Gaussian Blur (h) Sharpness

Figure 2.13: Examples of data augmentation techniques using an image from the Cityscapes [7] dataset.

Cutout To address the challenge of object occlusion frequently observed in various computer vision tasks, the Cutout [22] method randomly conceals rectangular areas within input images while training. By employing this technique, the model is prompted to prioritize alternative features, thereby improving its capacity for generalization and bolstering its resilience against occlusions.

MixUp Blending pairs of images, denoted as x_1 and x_2, along with their corresponding labels y_1 and y_2 in a weighted fashion, the MixUp [23] technique introduces a data augmentation approach. This method generates new training instances by combining the features and targets according to the following equations:

$$\lambda \sim \text{Beta}(\alpha, \alpha) \tag{2.13}$$

$$x' = \lambda x_1 + (1-\lambda)x_2 \tag{2.14}$$

$$y' = \lambda y_1 + (1-\lambda)y_2 \tag{2.15}$$

In this context, λ is sampled from a Beta distribution where both parameters are assigned to α, ensuring a symmetrical distribution shape. The variables x' and y' symbolize the resultant composite image and label, respectively. This blending encourages the model to learn from interpolated data points between different samples, enhancing its generalization ability.

AutoAugment By employing Reinforcement Learning, AutoAugment [24] is a technique that automatically searches for optimal data augmentation policies. It explores various augmentation strategies and evaluates their impact on the model's performance, tailoring augmentation policies specific to the dataset and task. The available operations include rotation, translation, shearing, sample pairing, Cutout, and color adjustment.

RandAugment Utilizing only two hyperparameters, namely the number of augmentation operations and their respective magnitudes, the RandAugment [25] method enhances training data through the application of diverse augmentation operations. These operations encompass color transformations, rotations, and translations, and their random and sequential implementation is controlled by the specified hyperparameters, providing a straightforward means to augment training data.

CTAugment Berthelot et al. (2019) [26] proposes CTAugment to alleviate the need for supervised optimization and hyperparameter tuning. Like RandAugment, it employs random sampling of transformations during training but distinguishes itself by dynamically determining magnitudes for each transformation during the training phase.

2.2 Learning Paradigms

While this thesis focuses on Semi-Supervised and Self-Supervised Learning, it is crucial to grasp the array of approaches available for training models, discern the distinctions between these learning paradigms, and determine the contexts in which each method finds its appropriateness.

Given the proportion of labeled data, denoted by η, where $0 \leq \eta \leq 1$, and the proportion of unlabeled data, represented by γ, where $0 \leq \gamma \leq 1$ and thus $\eta + \gamma = 1$, the learning paradigms illustrated in Figure 2.14 can be formally characterized using these two parameters. Supervised Learning relies solely on annotated data and, as such, can be delineated as $\eta = 1$ (implying $\gamma = 0$). Unsupervised Learning and Self-Supervised Learning exclusively utilize unlabeled data, characterized by $\gamma = 1$ (thus $\eta = 0$). Semi-Supervised Learning amalgamates varying proportions of both labeled and unlabeled data and is expressed as $0 < \eta < 1$ (with $\gamma = 1 - \eta$).

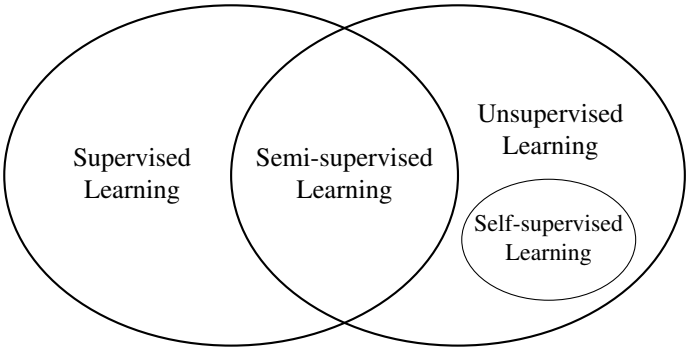


Figure 2.14: A visual representation of different learning paradigms: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning, and Self-Supervised Learning. This diagram illustrates the overlapping areas and distinctions between these fundamental approaches to Machine Learning.

2.2.1 Supervised Learning

Supervised Learning (SL) is the dominant methodology in Machine Learning, as the availability of the labeled data provides clear criteria for model optimization [27]. The approach involves learning a mapping function between a set of inputs and one or more output variables, relying only on labeled data. With SL, the model's performance heavily depends on the quantity and quality of the annotations, being more prone to issues such as generalization error, adversarial attacks, and spurious correlations. This makes it less appropriate for scenarios where annotating data is expensive to acquire or requires a substantial workforce, like in the context of Autonomous Driving [3]. Within Supervised learning, tasks can be broadly categorized into:

- **Classification** tasks, where the goal is to assign inputs into predefined classes, as it involves predicting a discrete label for input data. For instance, email spam detection, image classification, and sentiment analysis are typical examples of classification tasks. Examples of models used for classification tasks include Logistic Regression, Decision Trees, and Neural Networks.

- **Regression** tasks that establish a relationship between input variables and continuous numerical output values. Predicting housing prices, stock market trends, or temperature forecasts are examples of regression problems. Some examples of models used for regression tasks include Linear Regression, Support Vector Machines (SVMs), and Neural Networks.

2.2.2 Unsupervised Learning

It is also possible for a model to learn when no ground-truth labels are given. This approach is called Unsupervised Learning (UL), where the formal framework is based on the model's goal to obtain valuable representations of the input that would be later used for decision-making. Examples of UL techniques are:

- **Clustering**, aiming to organize data by identifying inherent similarities among its elements and segmenting a dataset into distinct groups where items within each group exhibit higher resemblance and notably differ from those in other groups. Essentially, it entails grouping objects based on shared characteristics while emphasizing group differences. For example, k-means clustering is a popular algorithm for partitioning data into distinct clusters based on feature similarity.
- **Dimensionality Reduction**, where the number of input variables or features is reduced while preserving essential information. This technique can be used to address issues like the curse of dimensionality, which can lead to increased computational complexity and overfitting in models. Principal Component Analysis (PCA) and word embeddings like Word2Vec are commonly used methods for dimensionality reduction.

2.2.3 Semi-Supervised Learning

Semi-Supervised Learning (SemiSL) is a paradigm that uses labeled and unlabeled data during the same training instance, seeking to alleviate the need for labeled data while leveraging from unlabeled data [28].

The architecture of SemiSL models typically incorporates a loss function comprising a supervised loss term, computed using labeled data, alongside one or more unsupervised loss terms calculated from the unlabeled data. Moreover, certain semi-supervised techniques integrate a regularization term, often inherently embedded within the previously mentioned losses [29]. This is one of the advantages in opposition to Self-Supervised Learning, as it allows for flexibility in adapting the strength of the regularization provided by the unlabelled data. However, while SelfSL enables the model to be fine-tuned to any downstream task, SemiSL is confined to the specific task type predetermined by the chosen method.

2.2.4 Weakly-Supervised Learning

Zhou, Z. (2018) [30, p. 3] describes Weakly-Supervised Learning (WSL) as the "process of learning from noisy or poorly labeled data", like social networks, where there is a lack of guarantee

that the users' hashtags are appropriately assigned, and discriminates WSL in three different but not mutually exclusive types, as illustrated in Figure 2.15:

- **Incomplete supervision** concerns the situation where only a tiny fraction of the data is labeled, typically because humans annotate it or there is an associated cost. Approaches to this scenario include Semi-Supervised Learning, Self-Supervised Learning, or Active Learning [31], where an "oracle", typically a human, is actively enquired to get the ground truth labels for unlabeled instances.
- **Inexact supervision** encapsulates the object-level annotations to image-level annotations, aiming to accommodate scenarios where object-level annotations are scarce or imprecise. This case is also known as Multi-instance Learning, which is typically used when data fits this particular category, as it allows for more accurate predictions by considering multiple instances of an object instead of just a single example [32].
- **Inaccurate supervision** englobes situations where the provided labels may deviate from ground truth or exhibit varying degrees of imprecision and can stem from multiple sources, including human annotator errors, ambiguous labeling criteria, or inherent difficulties in defining the ground truth itself. While *crowdsourcing* is a popular paradigm used as a cost-saving way to collect labels [33], the workers usually come from a large society and individually provide labels based on their judgment. Furthermore, there may exist "*spammers*" who assign random tags and "adversaries" who give incorrect answers deliberately. Majority-voting strategies with theoretical support in ensemble methods [34], which combine predictions from multiple models or annotators to reduce the impact of inaccuracies, are common approaches to mitigate this problem [35, 36].

2.2.5 Self-Supervised Learning

Self-Supervised Learning (SelfSL) is considered a subset of Unsupervised Learning because it falls under the broader category of Machine Learning techniques where the model learns from data without explicit labels [38]. However, while UL concentrates on detecting specific data patterns, SelfSL creates artificial labels, as the fundamental idea of this approach is to use some part of inputs or transformed inputs in the dataset as targets. The typical training process involving SelfSL comprises two main stages, but only the first utilizes self-supervision.

The first phase is designated as "pretext task training" and focuses on acquiring feature representations. This phase involves engaging in pre-designed tasks, referred to as pretext tasks, which serve as secondary objectives. These pretext tasks enable the model to learn in an unsupervised manner by automatically generating labels from an unlabeled dataset, thereby facilitating the extraction of meaningful features. Some examples of pretext tasks are:

- **Geometric transformations**, where the image can be flipped or rotated. The network should predict properties associated with the transformation applied, like the rotation angle or whether it is mirrored or not [39].

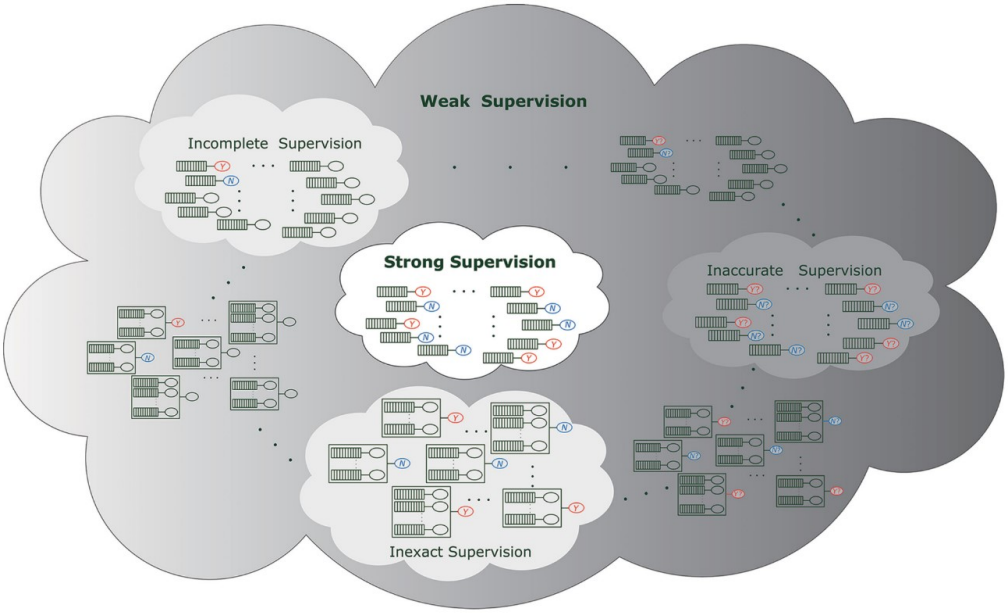


Figure 2.15: Illustration of the three typical types of weak supervision. Bars denote feature vectors, boxes represent instance-level samples, and "?" implies the inaccuracy of the label. The interpolated sub-graphs represent scenarios with mixed types of weak supervision. Source: [37]

- **Color transformations**, where the network should be able to predict changes in color space, such as grayscale conversion, color augmentation, or predicting the presence of specific color alterations [40].
- **Jigsaw puzzles**, where an image is cut into pieces and shuffled. The network must predict the correct arrangement of those pieces by understanding the relationship between the patches [41].
- **Missing patch prediction**, where a portion of the input image is intentionally occluded. The network's objective is to reconstruct the missing part based on the surrounding context and information available in the image [42].

The second and last stage is called "downstream task training", where the model parameters learned in the first phase are transferred to another downstream task, combining Transfer Learning to fine-tune the model parameters and finally evaluating the quality of the features learned by SelfSL. The fine-tuning of the parameters can either involve adjusting the entirety of the network's parameters or selectively fine-tuning specific layers, depending on the requirements and nature of the downstream task.

This approach might seem similar to SemiSL when using annotated data in the second phase. However, it is essential to distinguish that pre-training a backbone using unlabeled data, as in UL, and fine-tuning it later to a specific downstream task using labeled data, as in SL, is commonly designated by "two-stage self-supervision" and is not considered SemiSL, as labeled and unlabeled samples were used in different training instances [38].

2.2.6 Transfer Learning

While Transfer Learning (TL) can be used with supervised, semi-supervised, or unsupervised approaches [43], it is a potent methodology for utilizing the remaining labeled data of the self-supervised training. At its core, TL aims to benefit from the learned representations using a source task to expedite learning on a target task, particularly when labeled data for the target task is scarce or unavailable.

Additionally, Transfer Learning can take diverse forms, from fine-tuning the entire pre-trained model on the target task to selectively adapting specific layers or components of the model architecture. This flexibility allows practitioners to tailor the learning process according to the nuances and complexities inherent in the target task, thus optimizing the model's adaptability and performance in the new domain.

Chapter 3

Review on Semi-Supervision

While Section 2.2.3 explains the concept of Semi-Supervised Learning, this section presents a detailed analysis of the categories and methods that constitute it.

In the context of this chapter, the underlying premise is that the dataset $\mathcal{D}$ comprises a combination of partially labeled and partially unlabeled images, denoted as $\mathcal{D} = \mathcal{X} \cup \mathcal{U}$. Notably, the cardinality of dataset $\mathcal{D}$, represented as $N_{\mathcal{D}}$, signifies its amalgamation of labeled and unlabeled instances. The annotated subset of the dataset is expressed as $\mathcal{X} = \{x_i \mid i \in [1, N_{\mathcal{X}}]\}$, where $N_{\mathcal{X}}$ denotes the number of labeled examples and x_i represents individual examples. Furthermore, the set $\mathcal{Y}$ is defined as the ensemble of specific labels attributed to the partially labeled instances in $\mathcal{X}$, encompassing the diverse classes. This set, denoted as $\mathcal{Y} = \{y_i \mid i \in [1, N_{\mathcal{X}}]\}$, consists of $N_{\mathcal{C}}$ distinct classes, where y_i denotes individual labels corresponding to the x_i sample. The complementary portion encompasses the unlabeled instances in the dataset, formulated as $\mathcal{U} = \{u_j \mid j \in [1, N_{\mathcal{U}}]\}$, where $N_{\mathcal{U}}$ signifies the number of unlabeled examples and u_j represents unlabeled instances. The function $f(\cdot)$ is used to represent the model's inference, mapping input instances to their corresponding predictions.

3.1 Categories

Following an iterative analysis of the studied methods, it becomes possible to extract common underlying categories that form the basis of these methods. Those categories are not mutually exclusive, as approaches can aggregate more than one of these concepts.

3.1.1 Consistency Regularization

Initially proposed in seminal works [44, 45, 46], consistency regularization has emerged as a pivotal technique in enhancing the performance of Semi-Supervised Learning algorithms. This approach utilizes unlabeled data, grounded on the principle that a robust model should produce consistent predictions when presented with augmented versions of the same input (i.e., if we rotate an object, the object should still be classified as being the same object, even if we do not know what the object is). Commonly, this augmentation can be achieved through diverse techniques,

including domain-specific data transformations [46, 45, 47, 48], adversarial perturbations [49], dropout [46, 50], or Cross-entropy-based modifications [49, 48].

In scenarios where labeled data is available, consistency evaluation involves comparing the model's predictions with the ground truth labels to assess the model's performance and alignment with the provided labeled data. Conversely, consistency regularization uses loss functions to measure the disparity between the model's predictions for a perturbed input and its original counterpart. A basic expression capturing this concept is illustrated in Equation (3.1), where two unique stochastic augmentations, $\mathcal{A}^{(1)}(u)$ and $\mathcal{A}^{(2)}(u)$, are employed on an unlabeled example u, as done in prior work [46].

$$\left\| f(\mathcal{A}^{(1)}(u)) - f(\mathcal{A}^{(2)}(u)) \right\|_2^2 \tag{3.1}$$

Augmentation Anchoring Berthelot et al. (2019) [26] introduced a specific implementation of consistency regularization known as augmentation anchoring. Unlike general consistency regularization, which compares predictions of different augmentations equally, augmentation anchoring establishes a "weak" augmentation as an anchor point. The input undergoes this weak augmentation first, and then stronger augmentations are applied to the same input. The model is trained to ensure that the predictions from these strong augmentations are consistent with the predictions from the weakly augmented anchor. This approach enhances training stability and effectiveness, allowing the use of a standard Cross-entropy loss instead of the Mean Squared Error loss commonly used in consistency regularization. The key difference lies in the structured approach to augmentation: starting with a weak anchor and comparing it against stronger augmentations to enforce prediction coherence.

3.1.2 Entropy Minimization

Entropy minimization is an entropy regularization strategy used to enforce that the classifier's decision boundary should not pass through high-density regions of the marginal data distribution. Grandvalet & Bengio (2005) [51] advocate for utilizing unlabeled data to ensure well-separated classes, achieved by encouraging the model's output distribution to exhibit low entropy, signifying high-confidence predictions on unlabeled data. This principle can be accomplished by using a Cross-entropy loss term.

However, there is a risk of overfitting to low-confidence data points by outputting large logits, leading to overly confident predictions [52]. Therefore, in isolation, entropy minimization might not yield competitive SemiSL results, although its integration with diverse other approaches has shown promise in achieving state-of-the-art results [47, 26, 53, 54].

Temperature Sharpening An alternative methodology frequently employed involves modifying the categorical distribution [55], a technique referred to as "temperature sharpening" or "temperature scaling". MixMatch [47] and ReMixMatch [26] achieve this by employing a "sharpening" function on the target distribution for unlabeled data to reduce the entropy of the label distribution.

This operation is articulated as

$$\text{Sharpen}(p,T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{N_C} p_j^{\frac{1}{T}} , \tag{3.2}$$

where p is the categorical distribution and T is a hyperparameter regulating the "temperature". As $T \to 0$, the output progressively converges towards a Dirac ("one-hot") distribution, thereby encouraging the model to generate predictions characterized by lower entropy.

3.1.3 Pseudo-labeling

Pseudo-labeling is an influential technique in Semi-Supervised Learning that harnesses the power of the model itself to create surrogate labels for unlabeled data [56, 57]. This approach revolves around utilizing a "hard" label determined by the arg max of the model's output while retaining only those artificial labels with maximum class probabilities surpassing a predefined threshold [58]. Letting $p_u = f(u)$ for a given unlabeled example u, this process employs the loss function

$$\frac{1}{N_\mathcal{U}} \sum_{u \in \mathcal{U}} \mathbb{1}\left(\max(p_u) \geq \tau\right) \mathrm{H}\left(\arg\max(p_u), p_u\right), \tag{3.3}$$

where τ represents the specified threshold. Notably, this reliance on hard labels establishes a close connection to entropy minimization [51, 45], where the model is encouraged to predict high-confidence outputs on unlabeled data.

3.2 Methods

3.2.1 Pi-Model

Pi-Model [46], or Π-Model, is one of the simplest semi-supervision methods regarding classification tasks, using consistency regularization to encourage consistent network output between different transformations of the same input. It uses a time-dependent weighting function for the unsupervised loss term while taking advantage of data augmentation and network dropout as regularization techniques.

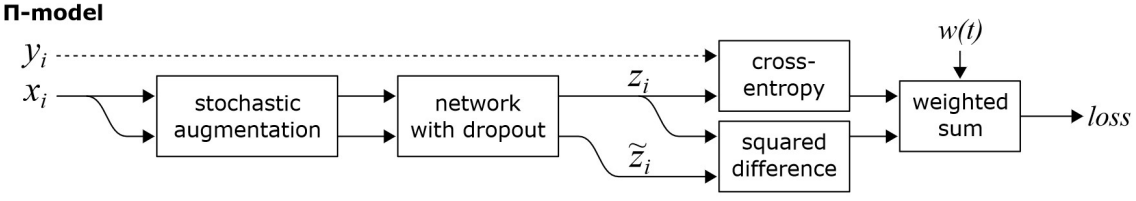


Figure 3.1: Diagram representing the loss computation for Π-Model. Distinct stochastic augmentations are applied to an input instance to enforce consistency regularization. The cross-entropy loss is also evaluated if the input is a labeled sample. Source: [46]

Consistency Regularization In each training pass (epoch), the method starts by generating two different perturbations for each sample, labeled or unlabeled, by applying stochastic weak augmentations. Subsequently, the model produces the predictions z and $\tilde{z}$ for each perturbed input, which will be used on the loss function to measure the consistency between the model's outputs. Consistent regularization ensures that the predictions for the same sample should be as similar as possible when subjected to different augmentations.

Network Dropout Dropout [59] is a regularization technique that randomly deactivates a fraction of neurons in the network during each forward and backward pass. In addition to consistency regularization, network dropout is applied when predicting each perturbed input, introducing an element of uncertainty in the model's predictions, further promoting its adaptability, and reducing overfitting.

Loss Function The loss function, expressed in Equation (3.4), comprises a supervised component evaluated only for the labeled data and an unsupervised component assessed for the entire data. The first component uses a standard Cross-entropy loss, where the target label y is compared with the model's prediction of the first transformation z. The second component penalizes different forecasts for the same input using a mean square difference between the augmented inputs z and $\tilde{z}$. The unsupervised loss weighting function $w(t)$ ramps up, starting from zero up to a fixed weight after a specific number of epochs, following a Gaussian curve. This ramp-up revealed great importance in preventing the network from getting stuck in a degenerate solution because the supervised term dominates the loss in the early stages of the training. Given the two distinct stochastic predictions $z_u = f(\mathcal{A}_{\text{Weak}}{}^{(1)}(u))$ and $\tilde{z}_u = f(\mathcal{A}_{\text{Weak}}{}^{(2)}(u))$, the consistency loss can be described as

$$\mathcal{L} = \frac{1}{N_{\mathcal{X}}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathrm{H}\left(y, f(\mathcal{A}_{\text{Weak}}(x))\right) + w_t \frac{1}{N_{\mathcal{C}} N_{\mathcal{D}}} \sum_{u\in\mathcal{D}} \|z_u - \tilde{z}_u\|_2^2 . \tag{3.4}$$

3.2.2 Temporal Ensembling

Due to efficiency issues, Laine & Aila (2017) [46] also proposed a second version of Π-Model named Temporal Ensembling, which enables a single network evaluation per epoch during training, yielding an approximate $2\times$ speedup compared to the Π-Model.

Π-Model Limitations In Π-Model, the process of evaluating both branches could equally be done in two distinct phases: initially classifying the training set without adjusting the classifier weights and subsequently training the network under various augmentations and dropout, utilizing the resultant predictions as targets for the unsupervised loss component. A notable drawback arises from the reliance on a single network evaluation for generating the predicted targets that are susceptible to rapid fluctuations and consequently become noisy. The training efficiency is also a concern, as each input is transformed twice.

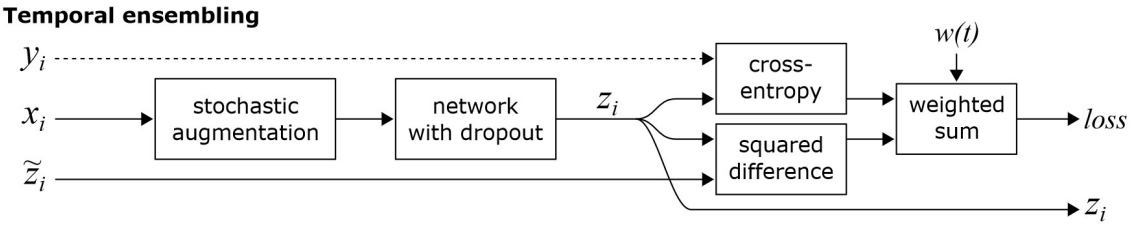


Figure 3.2: Diagram representing the loss computation for Temporal Ensembling. A stochastic augmentation is applied to the input instance to enforce consistency regularization later. The Cross-entropy loss is also evaluated if the input is a labeled sample. The ensemble prediction $\tilde{z}$ incorporates the Exponential Moving Average mechanism for a more responsive adaptation to the evolving patterns in the training data. Source: [46]

Ensemble Prediction Temporal ensembling alleviates these limitations by aggregating the predictions from multiple prior network evaluations into an ensemble prediction by leveraging the Exponential Moving Average (EMA). This reduces the computational overhead as only a single forward pass is performed in each iteration, as EMA can accumulate the predictions of the past epochs. The ensemble prediction $\tilde{z}$ can be described as

$$\tilde{z} = \alpha\tilde{z} + (1-\alpha)z, \tag{3.5}$$

where α is a momentum parameter that regulates the extent to which the ensemble reflects the training history and thus represents a weighted average of the output from previous training epochs, with recent epochs having a larger weight than distant epochs. However, a startup bias is associated with the generation of training target $\tilde{z}$. To address this issue, a corrective measure is employed by dividing the process by the factor $(1-\alpha^t)$, a strategy also used in Adam [60] and mean-only batch normalization [61]. The corrected version of the ensemble prediction should be expressed as

$$\tilde{z} = \frac{\alpha\tilde{z} + (1-\alpha)z}{(1-\alpha^t)}. \tag{3.6}$$

Loss Function In Temporal Ensembling, the loss function undergoes modification from the original Π-Model due to incorporating the ensemble prediction mechanism. The loss function now includes a term that penalizes the divergence between the model's prediction for a sample and the ensemble prediction $\tilde{z}$ generated from past evaluations. The supervised component remains unchanged, evaluating the Cross-entropy loss for labeled data. However, the unsupervised component now involves the mean square difference between the model prediction $f(u)$ and the ensemble prediction $\tilde{z}_u$ for each unlabeled input u. The updated loss function for Temporal Ensembling can be expressed as

$$\mathcal{L} = \frac{1}{N_{\mathcal{X}}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathrm{H}(y, f(\mathcal{A}_{\mathrm{Weak}}(x))) + w_t \frac{1}{N_{\mathcal{C}} N_{\mathcal{D}}} \sum_{u\in\mathcal{D}} \|f(\mathcal{A}_{\mathrm{Weak}}(u)) - \tilde{z}_u\|_2^2. \tag{3.7}$$

3.2.3 MixMatch

MixMatch [47] is a "holistic" approach that incorporates the concepts of pseudo-labeling, consistency regularization, and entropy minimization while assuming that the task to be solved is image classification.

Pseudo-Labeling The authors introduce a label-guessing algorithm to create target labels for the unlabeled data, as described in Figure 3.3. MixMatch produces K augmented images for a given unlabeled image by applying a stochastic weak augmentation. The classifier makes a prediction for each augmented image, which is then averaged and regularized using temperature "sharpening", an entropy minimization technique described in Section 3.1.2, generating the label guess for all the augmented images.

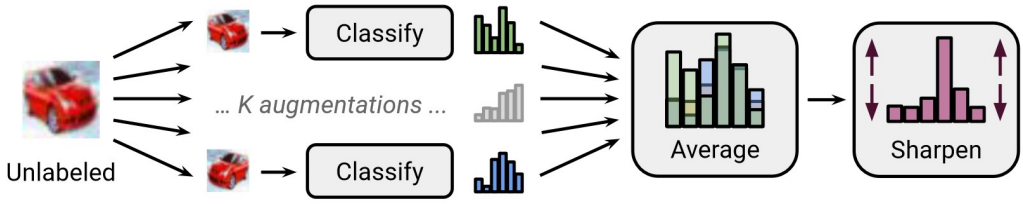


Figure 3.3: MixMatch pseudo-labeling algorithm diagram. For a single unlabeled sample, K augmented images are generated using stochastic data augmentation and fed to the classifier to predict their label. Then, the average of the obtained distribution is "sharpened" by adjusting the distribution's temperature. Source: [47]

Data Augmentation The MixMatch method itself is a data augmentation technique that combines the previous pseudo-labeling approach, producing a processed set of augmented annotated examples $\mathcal{X}'$ with the respective labels and a set of transformed unlabeled samples with "guessed" labels $\mathcal{U}'$ from a set of labeled data $\mathcal{X}$, the respective labels $\mathcal{Y}$ and a set of unlabeled data $\mathcal{U}$, formally defined as

$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{Y}, \mathcal{U}, T, K, \alpha), \tag{3.8}$$

where T, K, and α are hyperparameters that describe the "sharpening temperature", the number of augmentations, and the Beta distribution for MixUp [23], respectively. Using the terminology as in Figure 3.4, $\hat{\mathcal{X}}$ is the result of applying stochastic weak data augmentations to $\mathcal{X}$ followed by its combination with the respective labels $\mathcal{Y}$, and $\hat{\mathcal{U}}$ is the result of applying the pseudo-labeling algorithm to $\mathcal{U}$, which includes all the augmented images used in the pseudo-labeling process. Optionally, it is possible to create a composite set denoted as $\mathcal{W}$, which combines the two separate collections, specifically the concatenation of $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ followed by a shuffle operation, as denoted in Equation (3.9). This composite set can subsequently be employed to apply the MixUp method described by Zhang et al. (2017) [23], whereas Equation (3.11) intentionally uses the remainder of $\mathcal{W}$ that was not used while computing $\mathcal{X}'$ on Equation (3.10). The conjunction of these three equations defines how MixUp is utilized to generate the returned data sets, potentially enhancing

the generalization of the model by facilitating the mixing of labeled examples with unlabeled examples. Not performing this last step is the equivalent of assigning $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ to $\mathcal{X}'$ and $\mathcal{U}'$, respectively.

$$\mathcal{W} = \text{Shuffle}(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})) \tag{3.9}$$

$$\mathcal{X}' = \left\{ \text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i) \mid i \in \left[1, |\hat{\mathcal{X}}|\right] \right\} \tag{3.10}$$

$$\mathcal{U}' = \left\{ \text{MixUp}(\hat{\mathcal{U}}_j, \mathcal{W}_{j+|\hat{\mathcal{X}}|}) \mid j \in \left[1, |\hat{\mathcal{U}}|\right] \right\} \tag{3.11}$$

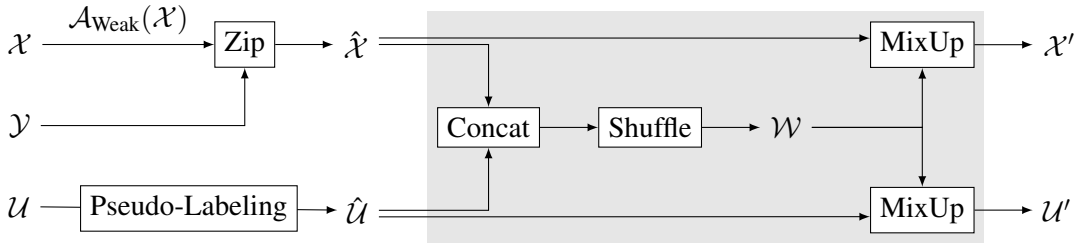


Figure 3.4: Diagram of the data augmentation technique used in MixMatch. A pseudo-labeling step is applied for the unlabeled data to create target labels, while the labeled data only benefits from weak data augmentation. The grey area represents the last optional step of combining them to apply MixUp to both samples, used to generalize the model better.

MixUp Modification The second term of the MixUp method always belongs to the shuffled set $\mathcal{W}$, making it impossible to predict if that image was initially labeled or unlabeled and creating a problem because Equation (2.13) can generate numbers bigger than 0.5, allowing the attribution of a larger weight to the second image, along with the label, and making it impossible to compute the individual losses correctly, as the supervised loss $\mathcal{L}_{\mathcal{X}}$ must prioritize the labeled examples and unsupervised loss $\mathcal{L}_{\mathcal{U}}$ must prioritize the unlabeled examples. To fix this, the authors introduced a modified version of MixUp, where after generating the value of λ with Equation (2.13), its value gets re-assigned to the maximum of λ and its complementary, defined as

$$\lambda = \max(\lambda, 1-\lambda), \tag{3.12}$$

ensuring that the first term of Equations (2.14) and (2.15) has a greater or equal weight than the second term.

Loss Function A standard semi-supervised loss is used during training, combining both supervised and unsupervised loss terms, expressed in Equation (3.13). The supervised loss term calculates the error between the model's predictions and the ground truth labels for the labeled data using a CE loss. In contrast, the unsupervised loss term is a MSE loss on predictions and guessed labels p from $\mathcal{U}'$. The MSE loss is used as it is less sensitive to incorrect predictions, unlike the Cross-entropy, and is often used as the unsupervised loss term in Semi-Supervised Learning [46, 50, 45].

$$\mathcal{L} = \frac{1}{|\mathcal{X}'|} \sum_{(x,y)\in\mathcal{X}'} \mathrm{H}(y, f(x)) + \lambda_{\mathcal{U}} \frac{1}{N_{\mathcal{C}}|\mathcal{U}'|} \sum_{(u,p)\in\mathcal{U}'} \|p - f(u)\|_2^2 \tag{3.13}$$

The hyperparameter $\lambda_{\mathcal{U}}$ weights the contribution of the unlabeled examples.

3.2.4 ReMixMatch

ReMixMatch [26] is an extension of the MixMatch methodology that introduces two novel techniques: distribution alignment and augmentation anchoring. Moreover, a novel augmentation strategy called CTAugment is employed to generate strongly augmented versions of input images. Lastly, ReMixMatch modifies the loss function to integrate these innovative techniques effectively while incorporating a self-supervised loss related to a rotation prediction task.

Augmentation Anchoring ReMixMatch innovates upon MixMatch's methodology by introducing augmentation anchoring, a specific implementation of consistency regularization. When confronted with an unlabeled sample, ReMixMatch establishes an initial "anchor" by subjecting the input to a weak augmentation and generates K strongly augmented versions derived from that same input, as illustrated in Figure 3.5, which would later be used to enforce coherence between those strongly-augmented versions and the "anchor". Notably, through experimentation, Augmentation Anchoring ensured stability and enabled the replacement of MixMatch's unlabeled-data MSE loss with a standard CE loss. Augmentation anchoring exhibited significant advantages with increased augmentations, diverging from MixMatch, which achieved its optimal performance with fewer augmentations.

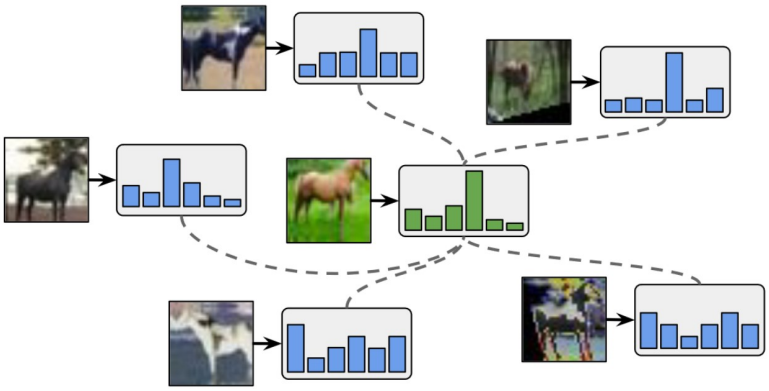


Figure 3.5: ReMixMatch augmentation anchoring uses the prediction for a weakly augmented image (green, middle) as the target for predictions on strong augmentations of the same image (blue). Source: [26]

CTAugment Creating robust augmentations is integral to enhancing model performance. However, the challenge lies in efficiently generating these augmentations. While AutoAugment [24]

achieves high validation accuracy, its reliance on labeled data for policy learning presents difficulties in scenarios with limited labels, such as SemiSL. RandAugment [25] addresses this by randomly sampling transformations but requires complex hyperparameter tuning on small labeled datasets. To solve this, ReMixMatch introduces CTAugment as an innovative augmentation that dynamically infers transformation magnitudes during training to enable aggressive data augmentation, as discussed in Section 2.1.3.2.

Distribution Alignment The concept of distribution alignment extends the framework's capability beyond entropy minimization by incorporating fairness into the training process. The model's predictions on unlabeled data are aggregated into a running average distribution denoted as $\tilde{p}$ over the last 128 batches. Leveraging this, the predicted distribution $(p = f(u))$ for an unlabeled example u is adjusted by scaling it using the ratio $g/\tilde{p}$, where g is the ground-truth distribution of the provided labeled data, and subsequently "sharpened" to ensure a valid probability distribution. This approach, illustrated in Figure 3.6, mitigates the biases induced by a non-uniform class distribution in the dataset.

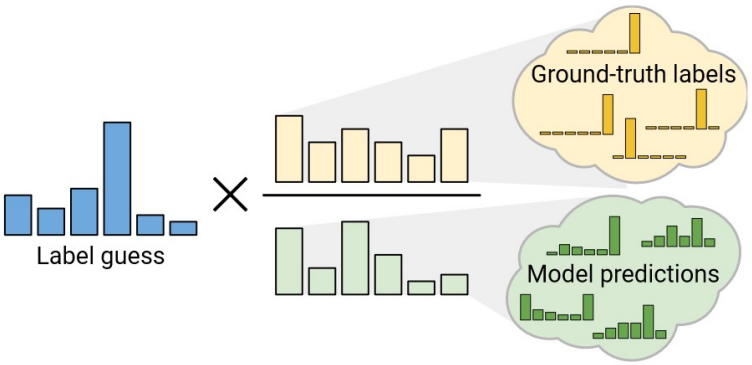


Figure 3.6: ReMixMatch distribution alignment diagram. The guessed label distributions are adjusted according to the empirical ground-truth class distribution ratio divided by the average model predictions on unlabeled data. Source: [26]

Pseudo-Labeling The pseudo-labeling algorithm amalgamates the outlined concepts of augmentation anchoring and distribution alignment. Initially, ReMixMatch generates a weakly augmented version and K strongly augmented image renditions. Subsequently, the classifier predicts the weak transformation. Afterward, distribution alignment is employed for that prediction, followed by two "sharpening" procedures: normalization to maintain a valid probability distribution after applying distribution alignment (special case where $T = 1$) and regular temperature "sharpening", as in MixMatch, to ensure entropy minimization.

Data Augmentation The ReMixMatch algorithm, depicted in Figure 3.7, modifies the data augmentation technique utilized by MixMatch discussed on page 26. The updated algorithm substitutes weak augmentation of labeled examples with strong augmentation, updates the pseudo-labeling approach, and generates a new set, denoted as $\hat{\mathcal{U}}_1$, which comprises the first heavily augmented version for each unlabeled image without employing MixUp. The method, which has the same parameters as MixMatch, can be formalized as

$$\mathcal{X}', \mathcal{U}', \hat{\mathcal{U}}_1 = \text{ReMixMatch}(\mathcal{X}, \mathcal{Y}, \mathcal{U}, T, K, \alpha). \tag{3.14}$$

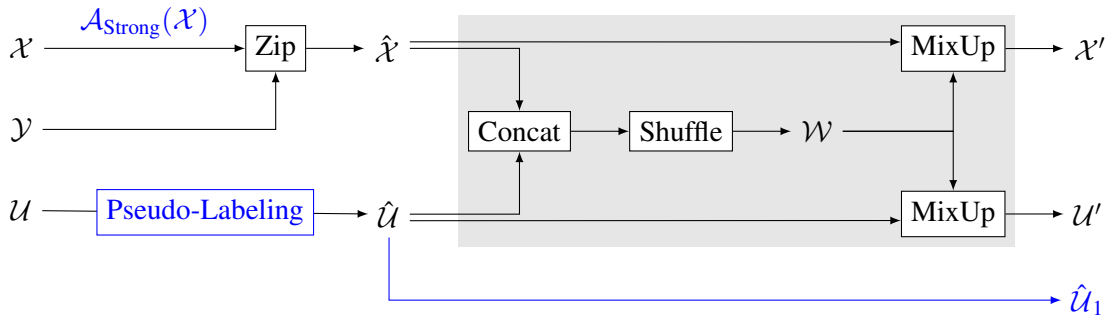


Figure 3.7: Diagram of the data augmentation technique used in ReMixMatch. A pseudo-labeling step is applied for the unlabeled data to create target labels, while the labeled data only benefits from strong data augmentation. The grey area represents the last optional step of combining them to apply MixUp to both samples, used to generalize the model better. A subset $\hat{\mathcal{U}}_1$ contains the first heavily augmented version of each unlabeled image without applying MixUp. Changes to MixMatch are highlighted in blue.

Rotation Prediction In line with recent advancements integrating Self-Supervised Learning techniques into Semi-Supervised Learning, ReMixMatch introduces rotation prediction as an augmentation to the loss function. The concept leverages the approach proposed by Gidaris et al. (2018) [62] and Zhai et al. (2019) [63], where each image $u \in \hat{\mathcal{U}}_1$ undergoes rotation via $\text{Rotate}(u, r)$, with the rotation angle r uniformly sampled from $r \sim \{0, 90, 180, 270\}$. The model is then tasked with predicting the degree of rotation as a four-class classification problem, encouraging it to learn invariant features relationships present within the data.

Loss Function ReMixMatch modifies the MixMatch loss function to include terms for the $\hat{\mathcal{U}}_1$ batch and for the rotation prediction task. All MSE losses are replaced with CE losses, and the scalars with the dataset sizes are removed. The new loss function is defined as

$$\begin{aligned}\mathcal{L} = &\sum_{(x,y)\in\mathcal{X}'} \text{H}(y, f(x)) + \lambda_{\mathcal{U}} \sum_{(u,p)\in\mathcal{U}'} \text{H}(p, f(u)) + \lambda_{\hat{\mathcal{U}}_1} \sum_{(u,p)\in\hat{\mathcal{U}}_1} \text{H}(p, f(u)) \\ &+ \lambda_r \sum_{(u,p)\in\hat{\mathcal{U}}_1} \text{H}(r, f(\text{Rotate}(u, r))),\end{aligned} \tag{3.15}$$

where $\lambda_{\hat{\mathcal{U}}_1}$ represents the weight on the first heavily-augmented samples and λ_r represents the weight on the rotation loss.

3.2.5 FixMatch

Among the techniques presented, FixMatch [53] stands out as one of the simplest yet most accurate methods, employing both consistency regularization and pseudo-labeling.

Pseudo-Labeling FixMatch utilizes a pseudo-labeling algorithm to assign target labels to unlabeled data, as illustrated in Figure 3.8. This algorithm generates both weakly and strongly augmented versions for each unlabeled input. The model then assesses both versions, employing the weakly augmented one as a pseudo-label to ensure consistency with the prediction of the strongly augmented version. Strong transformations may involve CTAugment [26] or RandAugment[25] followed by the Cutout [22] technique. The pseudo-label is determined by selecting the artificial label with the highest class probability, and it is utilized only if this probability exceeds a specified hyperparameter τ.

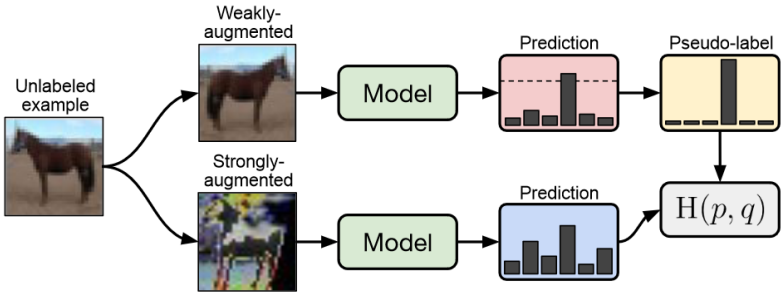


Figure 3.8: FixMatch pseudo-labeling algorithm diagram. For a single unlabeled sample, both weakly and strongly augmented images are generated using stochastic data augmentation and fed to the classifier for label prediction. The prediction from the weakly-augmented image serves as a pseudo-label and is compared with the strongly-augmented version. Source: [53]

Loss Function During training, FixMatch employs a standard semi-supervised loss that combines supervised and unsupervised terms, as expressed in Equation (3.16). The supervised loss term computes the error between the model's predictions and the ground truth labels for annotated data using a CE loss term. On the other hand, the unsupervised loss term employs a CE loss on the model predictions for strongly augmented images and the generated pseudo-labels $q = f(\mathcal{A}_{\text{Weak}}(u))$, where u represents an unlabeled image.

$$\begin{aligned}\mathcal{L} = &\frac{1}{|\mathcal{X}|}\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathrm{H}\left(y, f(\mathcal{A}_{\text{Weak}}(x))\right) \\ &+ \lambda_{\mathcal{U}}\frac{1}{|\mathcal{D}|}\sum_{u\in\mathcal{D}} \mathbb{1}_{[\max(q)\geq\tau]}\mathrm{H}\left(\arg\max(q), f(\mathcal{A}_{\text{Strong}}(u))\right)\end{aligned} \tag{3.16}$$

The hyperparameter $\lambda_{\mathcal{U}}$ weights the contribution of the unlabeled examples, while τ is a scalar denoting the threshold above which the pseudo-label is retained.

3.3 Summary

Semi-Supervised learning harnesses the power of unlabeled data to enhance the model's knowledge and boost its performance. This approach relies mainly on techniques such as Consistency Regularization, which ensures model predictions remain consistent across different versions of the same sample; Entropy Minimization, compelling the model to make confident predictions; and Pseudo-Labeling, which converts model predictions into actionable hard labels.

As shown in Table 3.1, the studied methods are founded upon consistency regularization as their primary approach. However, they diverge in incorporating supplementary techniques to enhance model robustness and efficacy. Certain methodologies opt for also utilizing heavy data augmentation, which has been shown to produce better results [48]. Conversely, approaches like ReMixMatch integrate components of Self-Supervised Learning, introducing secondary tasks to enrich the model's knowledge base.

While pseudo-labeling does induce a form of entropy minimization, its primary objective differs. Consequently, FixMatch is not classified under the entropy minimization category since it does not directly penalize model uncertainty.

Table 3.1: Comparison of Semi-supervision methods. CR, EM, PL, W-S Aug., MSE, and CE are abbreviations for Consistency Regularization, Entropy Minimization, Pseudo-Labeling, Weak-Strong Augmentation, Mean Squared Error, and Cross-entropy. Table adapted from USB [64].

Method	CR Loss	CR	EM	PL	W-S Aug.	Self-Supervised
Π-Model	MSE	✓				
Temporal Ensembling	MSE	✓				
MixMatch	MSE	✓	✓	✓		
ReMixMatch	CE	✓	✓	✓	✓	Rotation
FixMatch	CE	✓		✓	✓	

Chapter 4

Review on Self-Supervision

While Section 2.2.5 has elucidated various aspects of Self-Supervised Learning, this section dives into its methods and categories.

In the scope of this chapter, the dataset $\mathcal{U}$ comprises solely unlabeled instances. The dataset is denoted as $\mathcal{U} = \{u_i \mid i \in [1, N_\mathcal{U}]\}$, where $N_\mathcal{U}$ is the number of unlabeled examples and u_i represents individual unlabeled examples. Regarding model inference representation, $f(\cdot)$ is utilized as the base encoder, responsible for encoding the input data, while $g(\cdot)$ serves as the projection head, typically discarded after the Self-Supervised learning task.

4.1 Categories

In machine learning, models are traditionally classified into two primary categories: generative and discriminative. These distinctions arise from their approaches to understanding data distributions. Generative models, operating on the joint distribution $\mathrm{P}(\mathcal{X}, \mathcal{Y})$ of input data $\mathcal{X}$ and target labels $\mathcal{Y}$, focus on calculating the probability $\mathrm{P}(\mathcal{X} \mid \mathcal{Y} = y)$, capturing how inputs might be generated given specific outcomes. Conversely, discriminative models aim to model the conditional probability $\mathrm{P}(\mathcal{Y} \mid \mathcal{X} = x)$, centering on predicting labels given particular inputs [65].

Self-Supervised Learning comprises three primary categories, diverging in terms of their model architectures and objectives: generative, contrastive, and adversarial. Broadly, their architectures can be generalized into two main components: the generator and the discriminator. Furthermore, a decomposition exists within the generator between an encoder and a decoder. Figure 4.1 provides a comprehensive conceptual comparison of these categories.

4.1.1 Generative

Generative approaches focus on reconstructing individual samples [66, 67, 68]. The primary objective is to train models to generate realistic samples similar to unlabeled data through the use of generative models. Typically, these methods employ architectures consisting of an encoder and a decoder, where the encoder extracts meaningful explicit representations through specifically designed pretext tasks, and the decoder reconstructs the input from these representations, minimizing

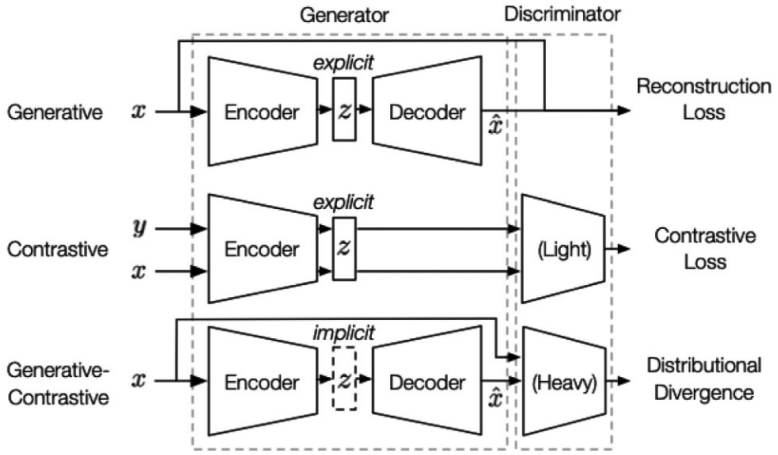


Figure 4.1: Conceptual comparison between self-supervision categories: Generative, Contrastive, and Generative-Contrastive. Source: [65]

a reconstruction loss. This measure quantifies the fidelity of the generated output to the original input, enhancing the model's ability to generate more realistic samples that closely resemble the unlabeled data. Those representations are then leveraged for downstream tasks.

These approaches encompass various tasks such as the *cloze test*, where models anticipate absent words in a sentence, and graph generation, where models construct graphs mirroring provided data structures. Methods in this domain rely on diverse models such as autoregressive, flow-based, auto-encoding, and hybrid generative models to accomplish these tasks.

However, pixel-level generation, involving the meticulous reconstruction of individual image elements, demands significant computational resources and may not be imperative for effective representation learning, especially when higher-level abstractions or features suffice for the intended tasks.

4.1.2 Contrastive

Contrastive approaches emphasize learning representations by contrasting positive samples against a pool of negative or dissimilar samples [69, 70]. The fundamental objective is to encourage similar representations for data augmentations or views of the same instance while pushing representations from different instances apart in a shared latent space, leading to improved discriminative power in the learned representations.

Typically, contrastive methods utilize a pretext task where pairs of augmented views or samples from the same instance are considered positive examples, and pairs from different instances act as negatives. Through a contrastive loss function, the model learns to maximize agreement between positive pairs and minimize agreement between negative pairs in the learned embedding space. However, contrastive learning often requires carefully selecting suitable augmentation strategies to ensure effective representation learning.

4.1.3 Adversarial

Adversarial representation learning, or generative-contrastive representation learning, combines elements from generative and contrastive approaches. Unlike traditional generative techniques focusing on reconstructing individual samples, this method aims to reconstruct the original data distribution, although it maintains an encoder-decoder structure. The presence of the decoder in adversarial methods demands that representations encapsulate all necessary information for reconstructing inputs, thereby being "reconstructive".

In contrast to the generative aspect, this approach borrows from contrastive methods by employing a discriminator that discerns between real and generated samples. This guides the generator in producing samples resembling the authentic data distribution. Hence, the primary training goal is to minimize divergence in distributions by instructing the generator to generate outputs indistinguishable from real data as perceived by the discriminator.

In self-supervised settings, adversarial representation learning finds applications in, for example, Generative Adversarial Networks (GANs) [71], which requires capturing crucial high-level features inherent in the data distribution.

Adversarial approaches excel at scenarios where modeling the complete data distribution is critical for subsequent tasks [65]. However, this approach tends to be computationally intensive and susceptible to issues like "mode collapse" – the generator only produces a single output type – or unstable behavior.

4.2 Methods

This study primarily focuses on exploring and applying contrastive methods. The emphasis on this approach stems from recognizing two key factors: firstly, the adaptability of other methodological categories in various scenarios that may potentially lead to trivial outcomes, and secondly, the widespread usage of contrastive methods in the field.

4.2.1 Rotation Prediction

A simple way to learn visual representations without relying on annotations is by creating secondary tasks that would encourage the model to learn robust features that capture the spatial configuration of the objects within the image. One simple yet effective method that follows this concept is rotation prediction [62].

Pretext Task In this approach, each image is rotated by one of four predefined angles: 0°, 90°, 180°, or 270°. The goal is to correctly classify the rotation angle of each transformed image, forcing the network to develop an understanding of the object's orientation and structure.

Loss Function The model must output the probabilities for the four rotation classes. Then, the network minimizes the Cross-entropy loss, measuring the discrepancy between the predicted and the true rotation angles. This loss can be defined as

$$\mathcal{L} = \frac{1}{|\mathcal{U}|} \sum_{(u,r)\in\mathcal{U}} \mathrm{H}\left(r, f(\mathrm{Rotate}\,(u,r))\right). \tag{4.1}$$

4.2.2 SimCLR

SimCLR [69] (Simple Framework for Contrastive Learning of Visual Representations) is a Self-Supervised Learning method that leverages contrastive learning to learn visual representations. The key idea behind SimCLR, depicted in Figure 4.2, is to maximize the agreement between differently augmented views of the same data example while minimizing the agreement between views of different data examples. This is achieved through a contrastive loss function, specifically the normalized temperature-scaled cross-entropy (NT-Xent) loss.

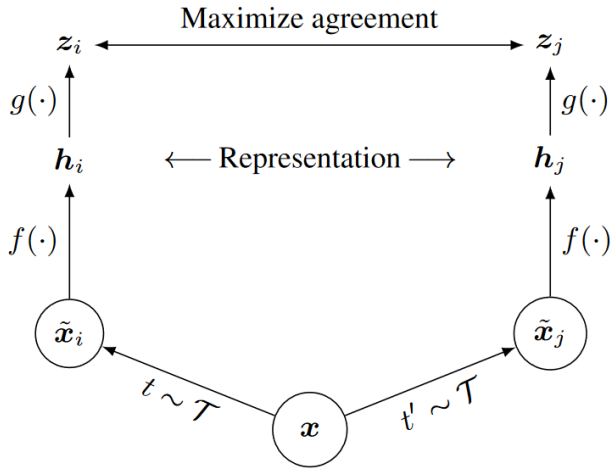


Figure 4.2: Diagram representation of the SimCLR method. Two data augmentations create correlated views of each data example. The encoder and projection head maximize agreement between these views using contrastive loss. Post-training, the projection head is discarded, and the encoder is used for downstream tasks. Source: [69]

Data Augmentation SimCLR heavily relies on data augmentation to create different views of the same image. Common augmentations include random cropping, color distortions, and Gaussian blur. These augmentations are designed to produce varied yet semantically consistent views of an image to encourage the model to learn robust features.

Architecture The architecture of SimCLR consists of a base encoder network followed by a projection head. The base encoder extracts feature representations from augmented images. The projection head, a small multi-layer perceptron, maps the encoded representations to a latent space

where the contrastive loss is applied. In the end, only the encoder is retrieved, while the projection head is thrown away.

Loss Function During training, a batch of images is augmented twice, resulting in two sets of views. The model then processes these views through the encoder and projection head to obtain their latent representations. The NT-Xent loss is computed to maximize the similarity between representations of the same image (positive pairs) and minimize the similarity between representations of different images (negative pairs). For a positive pair of samples (i, j), the loss function can be formally described as

$$\mathcal{L}_{i,j} = -\log \frac{\exp\left(\text{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp\left(\text{sim}\left(z_i, z_k\right)/\tau\right)}, \tag{4.2}$$

where $\text{sim}(z_i, z_j)$ is the cosine similarity between the latent vectors z_i and z_j, τ is a temperature parameter, and $2N$ is the total number of augmented examples in the batch. This method benefits from large batch sizes and strong data augmentation.

4.2.3 BYOL

Bootstrap Your Own Latent [72] (BYOL) is a self-supervision method that learns visual representations without relying on negative samples. Instead, BYOL employs two neural networks and utilizes their predictions to achieve the same objective.

Data Augmentation BYOL uses the same set of image augmentations as in SimCLR to create varied yet semantically consistent views of the same image, helping the model learn robust features. Each view is fed to a different network, aiming to maximize their similarity.

Architecture BYOL comprises two networks: the online and target networks. While both networks include an encoder and a projection head, an additional prediction head is added to the online network. Therefore, the target network shares the same architecture as the online network but with no prediction head. Figure 4.3 contains a visual representation of this architecture.

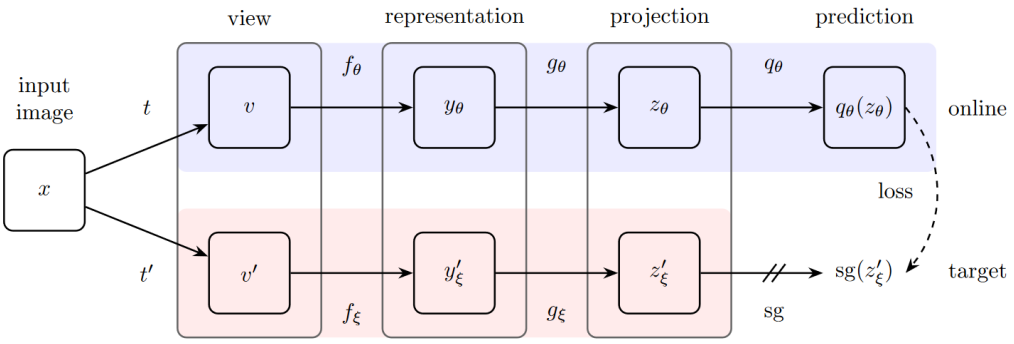


Figure 4.3: Diagram representation of the BYOL's architecture. Source: [72]

Target Network The target network is updated using a momentum-based moving average of the online network's parameters, ensuring that the target network evolves more smoothly and consistently than the online network. This mechanism prevents the model from predicting the same representation for all inputs.

Loss Function The loss function in BYOL aims to maximize the cosine similarity between the normalized predictions from the online network and the normalized target projections. Specifically, for a given pair of augmented views x_i and x_j, the loss is computed as

$$\mathcal{L}_{i,j} = -2\left(\text{sim}\left(q_\theta\left(z_i\right), z_i^{'}\right) + \text{sim}\left(q_\theta\left(z_j\right), z_j^{'}\right)\right), \tag{4.3}$$

where $\text{sim}(z_i, z_j)$ is the cosine similarity between the latent vectors z_i and z_j, q_θ symbolizes the prediction from the online network, z_i and z_j represent the projections from the online network, and $z_i^{'}$ and $z_j^{'}$ the projections from the target network.

4.2.4 MoCo

Momentum Contrast [70], abbreviated to MoCo, is a SelfSL method that uses a dynamic queue to address the limitations of memory and batch size in contrastive learning, as represented in Figure 4.4. It also uses a moving-averaged encoder to provide stable representations for the queue.

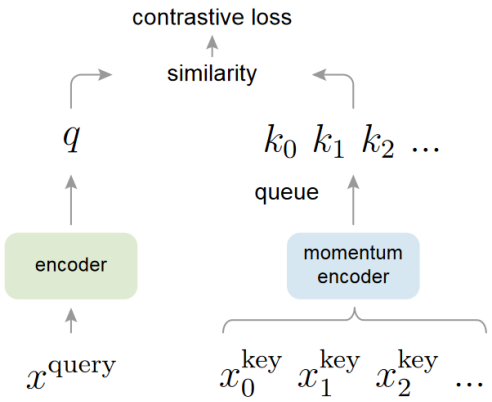


Figure 4.4: Diagram representation of the MoCo method. Source: [70]

Memory Bank MoCo maintains a dynamic queue of feature representations. As new data samples are processed, their representations are enqueued and the oldest representations are dequeued. This allows MoCo to store a large number of negative samples without the need to have a large batch size.

Momentum Encoder Instead of using a single encoder, MoCo employs two encoders: a *query* encoder and a *key* encoder. While the query encoder is updated with the standard backpropagation,

the key encoder is updated using a weighted average between the query and key encoder parameters. This ensures that the key encoder evolves more smoothly and consistently, providing stable representations.

Loss Function This method uses a contrastive loss to train the encoders called InfoNCE, maximizing the similarity between the query representation and the corresponding key representation (positive pair) while minimizing the similarity between the query representation and the remaining key representations in the queue (negative pairs). For a positive pair of samples (i, j), the loss function can be formally described as

$$\mathcal{L}_{i,j} = -\log \frac{\exp\left(\text{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{K} \exp\left(\text{sim}\left(z_i, q_k\right)/\tau\right)}, \tag{4.4}$$

where q_k is the k-th query representation, $\text{sim}(z_i, z_j)$ denotes the cosine similarity between the latent vectors z_i and z_j, τ is a temperature parameter, and K is the total number of samples in the queue.

Chapter 5

Methodology

5.1 Datasets

This study used publicly available datasets to assess the methods' effectiveness on classification and semantic segmentation tasks. All the datasets were normalized to their mean and standard deviation.

CIFAR-10 A widely-used dataset in the field of Computer Vision is CIFAR-10 [4], consisting of 60,000 32×32 color images with ten distinct classes, with 6,000 images per class. This dataset is usually associated with image classification task, where the goal is to classify each image into one of the ten predefined classes. While the dataset was originally composed of a train split of 50,000 pictures and 10,000 images for the test split, a validation set was constructed, retrieving 5,000 samples from the train split and leading to a final train split of 45,000 samples.

SVHN The Street View House Numbers [5] dataset, shortly SVHN, is a real-world image dataset for digit recognition obtained from house numbers in *Google Street View* images. When discarding the "extra" training set, it comprises 99,289 labeled images of digits in the street view, where 73,257 belong to the training set and the remaining to the test set. Each image is of size 32×32 pixels, similar to CIFAR-10. Consequently, the original dataset is divided into three segments – training, validation, and testing, where the validation set is created by extracting 10% of the samples from the training data. SVHN is frequently employed in research to assess the performance of machine learning algorithms designed for digit recognition tasks.

Cityscapes Representing a cornerstone in semantic urban scene understanding within Computer Vision and Autonomous Driving research, the Cityscapes [7] dataset offers meticulously annotated high-resolution images capturing diverse urban street scenes across several German cities. These annotations encompass detailed pixel-level semantic segmentation labels for urban objects such as roads, vehicles, and pedestrians. The dataset contains 500 testing images and 2,975 fine-annotated

training images, of which 500 are extracted for validation. It also has an extra training set containing 19,998 images with coarser annotations, often utilized for broader scene understanding tasks. Each image, originally 2048×1024, was resized to 512×512 pixels. Although the dataset has annotations for 30 classes, some were discarded and grouped with the void class, leaving a total of 20 classes.

KITTI The KITTI dataset [6] is also a prominent dataset used extensively in Computer Vision and Autonomous Driving, capturing various urban scenes. The dataset is especially noted for its diverse scenarios, including variable lighting conditions, dynamic objects, and differing road types. For this work, the semantic segmentation dataset, which contains a total of 200 annotated samples, was split into a training set of 140 images, a validation set with 20 images, and a test set with 40 images, each with an original resolution of 1242×375 pixels, later resized to 621×188. The dataset contains annotations for 20 classes.

Table 5.1: Summary of datasets settings used for classification and semantic segmentation tasks in the thesis.

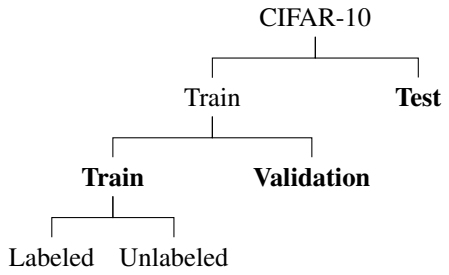
Task	Dataset	Training set	Validation set	Test set	Image size	Classes
Classification	CIFAR-10	45,000	5,000	10,000	32×32	10
	SVHN	65,931	7,326	26,032	32×32	10
Semantic Segmentation	Cityscapes	2,475 + 19,998	500	500	512×512	20
	KITTI	140 + 7,481	20	40	621×188	20

5.2 Unlabeled Splitting

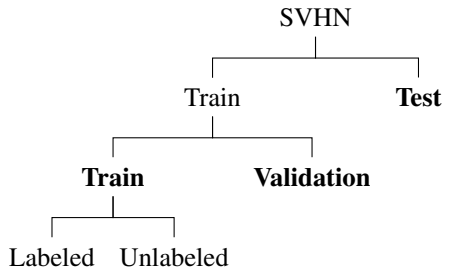
This study generated an additional split within each dataset to simulate the labeled and unlabeled sets required for Semi-Supervised Learning and Self-Supervised Learning.

For the classification datasets CIFAR-10 and SVHN, this was accomplished by further partitioning the samples within the training split based on a parameter indicating the number of labeled samples. This means the complementary set used as the unlabeled set consists of samples not included in the labeled set.

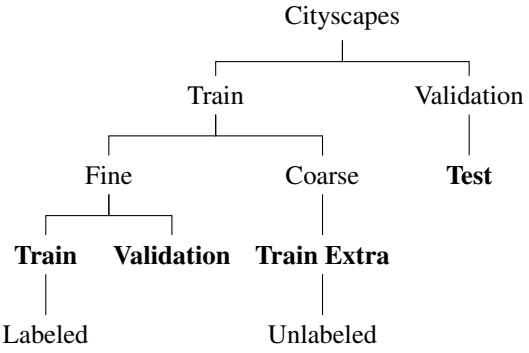
A different part of the dataset was used for the semantic segmentation datasets. In the case of Cityscapes, following precedent literature [73], the unlabeled samples are drawn from the "Coarse" set, while the labeled set is sourced from the "Fine" set. Regarding the KITTI dataset, the train set is maintained for the labeled set. In contrast, the training set of the object detection benchmark is used as the unlabeled set, ignoring its annotations. These partitions are visually depicted in Figure 5.1.



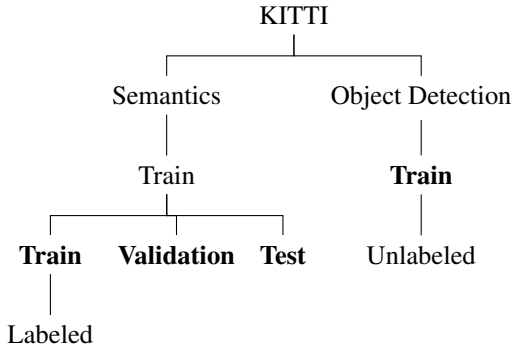
(a) CIFAR-10 dataset partition. As it does not contain a validation set, it is constructed using data from the training set.



(b) SVHN dataset partition. As it does not contain a validation set, it is constructed using data from the training set.



(c) Cityscapes dataset partition. The train and validation sets derive from the "Fine" annotated set. An additional training set containing coarse annotated images was used as the unlabeled set.



(d) KITTI dataset partition. The train set was split into the train, validation, and test sets. The train set of the object detection benchmark was used as the unlabeled set.

Figure 5.1: Datasets partition tree. The bold nodes represent the final train, validation, and test sets.

5.3 Training Cycle

Distinct training cycles were employed for each learning paradigm due to their differing approaches to data utilization.

5.3.1 Supervised Learning

In Supervised Learning, all the samples used during the training are annotated. Therefore, a typical training cycle was employed, where the whole dataset is iterated in each epoch, comparing the model predictions to the respective known targets. An additional parameter was introduced to control the number of samples to use, allowing for performance evaluation when employing subsets of the dataset. Furthermore, a validation loss was used to monitor the model's performance on the validation set during training. At the end of the training cycle, the model with the lowest validation loss was selected, which helped to reduce overfitting.

5.3.2 Semi-Supervised Learning

For the SemiSL paradigm, both labeled and unlabeled data are used simultaneously in the same training instance, meaning the training cycle must be robust to their availability – handle situations where one of the sets is exhausted before the other – and scale – what happens when one of the sets is more extensive or the same size. Three potential training cycles were considered regarding these situations.

Full Set Cycle The first approach involves iterating through the labeled and unlabeled sets entirely before progressing to a new epoch, utilizing the larger set even when the smaller set ended prematurely. Although straightforward, this approach creates situations where one set is unavailable, introducing an imbalance where the model only learns from either the labeled or unlabeled set as soon as the other exhausts itself.

Epoch	1			2			
Iteration	1	2	3	1	2	3	...
Labeled Batch	L1	L2	–	L1	L2	–	
Unlabeled Batch	U1	U2	U3	U1	U2	U3	

Figure 5.2: Exemplification of the "Full Set Cycle" approach. The labeled and unlabeled sets contain two and three batches, respectively. When the labeled set is exhausted at the second iteration, it will remain empty until the next epoch.

Full Set Cycle with Repetition This approach is similar to the first one but repeats the smaller set whenever it is exhausted, addressing the previously mentioned issue. However, due to repeated exposure, this would risk overfitting the smaller set. Also, it deviates from the conventional definition of an epoch, where only one complete pass should be made through the entire training dataset.

Epoch	1			2			...
Iteration	1	2	3	1	2	3	
Labeled Batch	L1	L2	L1	L1	L2	L1	
Unlabeled Batch	U1	U2	U3	U1	U2	U3	

Figure 5.3: Exemplification of the "Full Set Cycle with Repetition" approach. The labeled and unlabeled sets contain two and three batches, respectively. When the labeled set is exhausted at the second iteration, it will repeat itself until it is exhausted again or the epoch ends.

Early Stop Cycle The third and last approach, ultimately chosen for implementation, addresses these issues by terminating an epoch upon exhausting the smaller set, also leading to a smaller training duration. There are two ways of handling the unlabeled data. The first way – used in this study – is to restart the cycle. Although some data from the larger would remain unused, periodic shuffling would attenuate this limitation, increasing the diversity of the samples of different epochs. The second way would be to continue from where it ended in the previous epoch, assuring that the whole dataset was used before repeating it.

Epoch	1		2		...
Iteration	1	2	1	2	
Labeled Batch	L1	L2	L1	L2	
Unlabeled Batch	U1	U2	U1	U2	

(a) The unlabeled batch is restarted when a new epoch starts.

Epoch	1		2		...
Iteration	1	2	1	2	
Labeled Batch	L1	L2	L1	L2	
Unlabeled Batch	U1	U2	U3	U1	

(b) The unlabeled batch continues from where it was left over when a new epoch starts.

Figure 5.4: Exemplification of the "Early Stop Cycle" approach. The labeled and unlabeled sets contain two and three batches, respectively. The epoch ends when the labeled set is exhausted at the second iteration.

5.3.3 Self-Supervised Learning

In Self-Supervised Learning, the data used during training does not contain any annotations. For this reason, this paradigm's training cycle did not rely on a validation loss to monitor performance during training. After completing the self-supervised training, the model's weights, pre-trained in this manner, are loaded and then fine-tuned on downstream tasks with labeled data.

A common approach in SelfSL is *linear evaluation*, where a linear classifier is trained on top of the frozen pre-trained features to assess the quality of the learned representations, providing a quick assessment of the feature extraction capabilities of the pre-trained model. However, in this

work, linear evaluation was not performed, as the main focus was on the fine-tuning process and its impact on the performance of downstream tasks.

5.4 Experimental Setup

5.4.1 Experiments

The efficacy of the methods was evaluated in different tasks, datasets, and models. Specifically, the experiments varied the amount of labeled data on CIFAR-10 and SVHN for classification, as well as Cityscapes and KITTI for semantic segmentation. The methods were first implemented and evaluated in the classification datasets to assert the correctness of the implementation by comparing the results. Therefore, they were adapted to be compatible with other dense tasks by performing the changes mentioned in Section 5.4.3.

Classification For classification, Wide ResNet [74] (specifically, "WRN-28-2") was employed due to its widespread use in various semi-supervised methods. This architecture corresponds to a ResNet with a depth of 28 and a width of 2. However, as Wide ResNet has a smaller backbone representation, the Self-Supervised Learning approach was not achieving its full potential, as it heavily depends on the strength of the backbone. For this reason, a second model, ResNet50, was added to enhance performance.

For CIFAR-10 experiments, the number of labeled examples varies between 1,000, 2,000, and 4,000, a subset of what is considered the standard practice [47, 26]. For SVHN, a similar subset is used, containing 250, 500, or 1,000 labeled examples. As also common in the literature, the metric used to evaluate the model's performance was the Top-1 Accuracy.

Semantic Segmentation For semantic segmentation, the DeepLabV3 model with two different backbones – ResNet101 and MobileNetV3-Large – was used. A common practice of using fractions with the power of 2 was used to vary the number of labeled partitions [73]. For Cityscapes, 1/2, 1/4, 1/8, 1/16, and 1/32 were used. Only the labeled partitions 1/2, 1/4, and 1/8 were used for KITTI, as it is a smaller dataset. As also common in the literature, the metric used to evaluate the model's performance was the IoU using a macro reduction.

5.4.2 Implementation Details

This section details the choices made when implementing and adapting the studied methods. The validation loss for all the experiments was the CE loss. To ensure a fair comparison, the same data augmentations on labeled data were used across SL, SemiSL, and SelfSL for the same dataset. While some of the hyperparameters used for each dataset across the learning paradigms are shown in Table 5.2 and described in the following sections, consulting the code repository is advised for a complete understanding of its implementations and the hyperparameters used.

Table 5.2: Hyperparameters used for the different learning paradigms across different datasets. The table details the number of epochs, labeled and unlabeled batch sizes (BS), and EMA decay rates when updating the model's parameters.

Method	Dataset	Epochs	Labeled BS	Unlabeled BS	EMA Decay
Supervised Learning	CIFAR-10	200	128	–	–
	SVHN	200	128	–	–
	Cityscapes	120	16	–	–
	KITTI	120	16	–	–
Semi-Supervised Learning	CIFAR-10	1000	16	112	0.999
	SVHN	1000	16	112	0.999
	Cityscapes	120	1	3	0.99
	KITTI	120	2	6	0.99
Self-Supervised Learning	CIFAR-10	200	–	128	–
	SVHN	100	–	128	–
	Cityscapes	100	–	16	–
	KITTI	100	–	16	–

5.4.2.1 Supervised Learning

To start, a fully supervised baseline was trained to measure the baseline performance for each labeled partition. An experiment was also performed using the complete training set to determine the highest possible accuracy. Additionally, more baseline experiments were conducted using Pytorch's pre-trained models. The default pre-trained weights in PyTorch were trained on ImageNet-1K [75], which, in this same dataset, achieved a Top-1 Accuracy of 76.13% for ResNet-50, 74.04% for MobileNetV3-Large, and 77.37% for ResNet-101. However, This was not performed using Wide ResNet, as no pre-trained model was available for this architecture.

For classification, the recommended settings were used when training both models [74]: batch size of 128, SGD optimizer with Nesterov momentum of 0.9, weight decay of 0.0005, and trained for 200 epochs. For CIFAR-10, the learning rate started at 0.1 and dropped by 0.2 at 60, 120, and 160 epochs. For SVHN, it started at 0.01 and is dropped by 0.1 at the epochs 80 and 120. Regarding the Cityscapes dataset, the settings proposed by Xu et al. (2017) [76] were used: the batch size of 16, SGD optimizer with the momentum of 0.9, and a polynomial policy to dynamically decay the learning rate along the whole training. The initial learning rate was set to 0.007, and the weight decay to 0.0005. For KITTI, the Adam optimizer was used with a learning rate of 0.001, with no scheduler, and the batch size was set to 8.

The transformations used to train the labeled data were random cropping and horizontal flip for CIFAR-10, random crop for SVHN, and resize and random crop for Cityscapes and KITTI.

5.4.2.2 Semi-Supervised Learning

The standard SemiSL evaluation protocols [52, 47] were followed to ensure a fair comparison between methods. Generally, uniform settings, such as the number of epochs and batch size, were applied across the methods. However, some settings were tailored to the specific requirements of each technique, such as the optimizer or scheduler. In scenarios where the same batch size could

not be maintained due to hardware limitations, the ratio between the batch sizes of the labeled and unlabeled sets was kept consistent to ensure that the same amount of data was used.

Apart from the number of training epochs and batch size, the hyperparameters used were those specified in each method's paper, including optimizers and schedulers, to replicate the reported performance. Semi-Supervised Learning uniquely benefited from using a weighted mean when updating the model's parameters. Two types of transformations were used in this paradigm. The same set of transformations used for the labeled data was applied for weak augmentations, except for MixMatch, which also utilized Gaussian Blur. The set of transformations for strong augmentations included those used for the labeled data extended with RandAugment and Cutout methods.

While all the methods benefited from data shuffling at the beginning of each epoch, data shuffling was disabled for the Temporal Ensembling method, as it required maintaining the same order of data throughout the training process.

5.4.2.3 Self-Supervised Learning

A consistent comparison, similar to the one used in Semi-Supervised Learning, was employed in this paradigm. This involved using the same batch size and number of epochs across all methods. Additionally, the hyperparameters specified in each method's original paper, including optimizers and schedulers, were used to replicate the reported performance accurately. The transformations used by the methods comprise random cropping, horizontal flipping, color jittering, and grayscaling.

An exceptional characteristic of Self-Supervised Learning methods is that they typically achieve state-of-the-art performance when the input images are significantly large. Consequently, the images from the CIFAR-10 and SVHN datasets were resized to 224×224, as recommended in previous studies [69, 70, 72], including during the fine-tuning process. The images from the Cityscapes and KITTI datasets retained their original sizes since they were already sufficiently large.

For the fine-tuning process, the same settings as in Supervised Learning were applied, with the exception of the learning rate. Since the model parameters already contain useful information from the SelfSL phase, a lower learning rate was used to prevent the model from diverging from what it had learned. This adjustment led to improved results.

Finally, although SimCLR particularly benefits from a high batch size, due to memory limitations, a smaller batch size than recommended in the paper was used in our experiments, which may have negatively affected the results.

5.4.3 Methods Adaptation

We implemented specific changes described below to adapt methods for dense tasks, particularly semantic segmentation, while maintaining a consistent set of hyperparameters across all methods.

In the context of Self-Supervised Learning, no significant modifications were required, as this learning paradigm is inherently agnostic to the task type, focusing solely on learning robust representations. Conversely, the Semi-Supervised Learning methods, initially designed for classification tasks, necessitated careful adaptation. Ensuring consistency in the transformations applied was crucial; thus, we avoided separated geometrical transformations that could alter the masks. Instead, we used the same geometrical transformations to all the transformed versions of the unlabeled data, preserving mask consistency in these versions.

Some methods, however, could not be adapted for our experiments. For instance, the Temporal Ensembling method was excluded, as it relies on maintaining a weighted average of the model predictions over time. However, in semantic segmentation, where predictions are pixel-level and subject to changes due to different crops at each epoch, the same pixel can not be guaranteed to stay in the same position throughout the epochs.

Additionally, the SelfSL Rotation method was not applicable to the KITTI dataset because its images are not squared, making rotation-based tasks impractical. Similarly, in the ReMixMatch method, we set the rotation loss weight to zero to disable this pretext task, potentially reducing performance on this dataset.

Maintaining consistency between transformed versions of an image is crucial, especially for dense tasks like semantic segmentation. Therefore, we modified the RandAugment [25] technique, used in most methods, to exclude geometric transformations. This modified version, **Invariant RandAugment**, omits transformations such as shearing, translation, rotation, and sharpness. These modifications prevent mask alterations, ensuring consistent image versions and supporting effective model learning.

Chapter 6

Results & Discussion

6.1 Benchmark Results

In this section, we present the benchmark results of our experiments through the various learning paradigms. The results are divided into two main sections: Performance Metrics and Computational Efficiency.

6.1.1 Performance Metrics

This section focuses on key performance metrics, such as accuracy and mean Intersection over Union. We later discuss several aspects to compare the methods and learning paradigms by comparing these metrics.

For CIFAR-10 and SVHN, Tables 6.1 and 6.2 present the results using the Wide ResNet28-2 and ResNet50 models, respectively. Although these tables do not include the performance of a fully supervised baseline (i.e., utilizing the entire training set) for stylistic reasons, such experiments were conducted. The fully supervised baseline achieved scores of 94.04% and 88.31% for CIFAR-10 with Wide ResNet28-2 and ResNet50, respectively, and 96.94% and 95.19% for SVHN with the same models. It should be noted that MixMatch did not converge to a stable loss value on the SVHN dataset with ResNet50, resulting in extremely low performance, and thus its results are not displayed.

For Cityscapes, Tables 6.3 and 6.4 display the results using DeepLabv3 with ResNet101 and MobileNetV3-Large backbones, respectively. Similarly, for KITTI, Tables 6.5 and 6.6 show the results using the same models. It is important to highlight that the SimCLR method failed to converge on the Cityscapes dataset with the MobileNetV3-Large backbone, rendering the learned representations ineffective for fine-tuning. Additionally, experiments with the Rotation SelfSL method were not conducted for the KITTI dataset due to the non-square nature of the input images, which is a requirement for this method.

Table 6.1: Accuracy scores (%) for CIFAR-10 and SVHN. All methods were tested using Wide ResNet28-2 and the same codebase. For each column, the top-3 best-performing algorithms are depicted in **bold**.

	CIFAR-10			SVHN		
Method	1000 labels	2000 labels	4000 labels	250 labels	500 labels	1000 labels
Random Weights Init.	45.43	63.97	72.52	19.19	68.71	85.38
Π-Model	63.98	74.66	84.59	**71.40**	88.14	93.66
Temporal Ensembling	62.19	72.80	83.94	48.17	83.62	92.01
MixMatch	**69.11**	**87.52**	**89.90**	**60.20**	**95.85**	**96.06**
ReMixMatch	**77.55**	**87.90**	**90.35**	39.00	**95.68**	**96.35**
FixMatch	**78.48**	**86.10**	**90.84**	**90.12**	**94.62**	**95.54**
Rotation	45.69	52.70	64.97	21.46	22.64	38.12
SimCLR	49.60	57.66	63.73	14.04	19.76	30.22
BYOL	39.60	46.41	53.99	14.80	15.95	19.76
MoCo	43.98	50.24	61.11	17.16	19.79	24.08

Table 6.2: Accuracy scores (%) for CIFAR-10 and SVHN. All methods were tested using ResNet50 and the same codebase. For each column, the top-3 best-performing algorithms are depicted in **bold**.

	CIFAR-10			SVHN		
Method	1000 labels	2000 labels	4000 labels	250 labels	500 labels	1000 labels
Random Weights Init.	39.96	47.82	55.62	20.71	27.56	52.98
Pretrained Weights Init.	**64.47**	**69.71**	**75.30**	40.70	56.01	73.21
Π-Model	51.26	59.32	65.94	**42.69**	**83.41**	82.25
Temporal Ensembling	49.78	58.64	60.92	23.44	37.46	79.95
MixMatch	**61.62**	**73.29**	**80.45**	–	–	–
ReMixMatch	43.33	42.16	59.27	40.28	**79.92**	**85.81**
FixMatch	**61.28**	**69.72**	**81.91**	25.08	**84.06**	**91.47**
Rotation	50.07	61.35	69.38	29.45	60.94	77.88
SimCLR	39.11	44.47	51.74	**50.47**	74.14	**82.97**
BYOL	52.97	58.85	66.53	31.02	52.20	73.30
MoCo	60.74	66.77	73.24	**43.99**	69.98	75.90

Table 6.3: mIoU (%) for Cityscapes. All methods were tested using DeepLabv3 with a ResNet101 backbone and the same codebase. For each column, the top-3 best-performing algorithms are depicted in **bold**.

Method	$77_{(1/32)}$	$155_{(1/16)}$	$309_{(1/8)}$	$619_{(1/4)}$	$1238_{(1/2)}$	2475_{All}
Random Weights Init.	36.28	**43.79**	48.35	**53.24**	55.14	60.52
Pretrained Weights Init.	36.16	43.19	48.28	50.49	**55.79**	–
Π-Model	26.82	29.86	31.05	30.33	28.67	–
MixMatch	**39.08**	41.93	**49.93**	47.81	52.06	–
ReMixMatch	**41.51**	**46.29**	**51.29**	**54.96**	**56.54**	–
FixMatch	**43.35**	**48.66**	**53.07**	**55.55**	**58.03**	–
Rotation	30.92	34.11	38.83	43.63	48.28	–
SimCLR	28.79	33.15	38.99	43.42	49.03	–
BYOL	27.55	31.69	35.28	41.43	47.64	–
MoCo	35.28	38.16	41.80	46.86	51.64	–

Table 6.4: mIoU (%) for Cityscapes. All methods were tested using DeepLabv3 with a MobileNetV3-Large backbone and the same codebase. For each column, the top-3 best-performing algorithms are depicted in **bold**.

Method	$77_{(1/32)}$	$155_{(1/16)}$	$309_{(1/8)}$	$619_{(1/4)}$	$1238_{(1/2)}$	2475_{All}
Random Weights Init.	29.83	32.44	36.90	40.18	45.00	48.73
Pretrained Weights Init.	30.30	33.32	38.92	42.66	**46.73**	–
Π-Model	21.23	24.91	22.74	22.71	26.09	–
MixMatch	**33.25**	35.82	37.54	39.36	38.30	–
ReMixMatch	**35.62**	**38.99**	**43.56**	43.23	45.01	–
FixMatch	**35.87**	**39.63**	**42.97**	**44.29**	45.41	–
Rotation	28.91	33.24	36.75	42.11	46.22	–
SimCLR	–	–	–	–	–	–
BYOL	29.61	33.85	38.57	**44.35**	**47.96**	–
MoCo	33.01	**36.13**	**41.04**	**44.01**	**48.75**	–

Table 6.5: mIoU (%) for KITTI. All methods were tested using DeepLabv3 with a ResNet101 backbone and the same codebase. For each column, the top-3 best-performing algorithms are depicted in **bold**.

Method	$18_{(1/8)}$	$35_{(1/4)}$	$70_{(1/2)}$	140_{All}
Random Weights Init.	27.84	29.52	35.26	41.11
Pretrained Weights Init.	**32.10**	**33.09**	**40.37**	–
Π-Model	25.72	28.49	30.29	–
MixMatch	26.90	32.12	35.96	–
ReMixMatch	19.07	25.80	33.70	–
FixMatch	**31.33**	**37.05**	**40.37**	–
Rotation	–	–	–	–
SimCLR	21.16	27.85	34.06	–
BYOL	**28.65**	31.33	**37.26**	–
MoCo	26.73	**33.62**	36.46	–

Table 6.6: mIoU (%) for KITTI. All methods were tested using DeepLabv3 with a MobileNetV3-Large backbone and the same codebase. For each column, the top-3 best-performing algorithms are depicted in **bold**.

Method	$18_{(1/8)}$	$35_{(1/4)}$	$70_{(1/2)}$	140_{All}
Random Weights Init.	**29.63**	**33.58**	34.35	40.22
Pretrained Weights Init.	**28.53**	**34.17**	**35.83**	–
Π-Model	18.09	23.19	24.88	–
MixMatch	21.13	28.83	33.64	–
ReMixMatch	17.66	19.59	26.60	–
FixMatch	**26.43**	28.18	**35.44**	–
Rotation	–	–	–	–
SimCLR	13.98	21.66	32.39	–
BYOL	20.64	**29.54**	**34.43**	–
MoCo	19.64	26.73	32.64	–

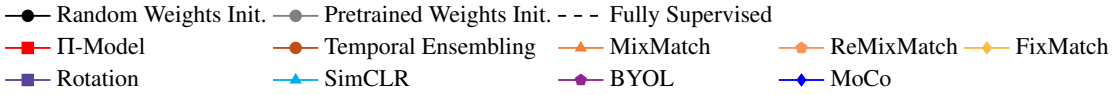
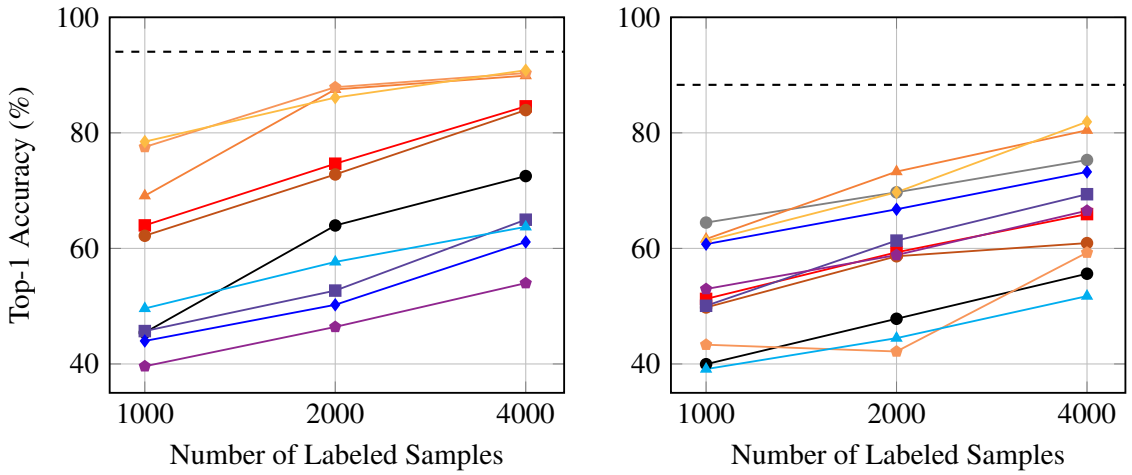


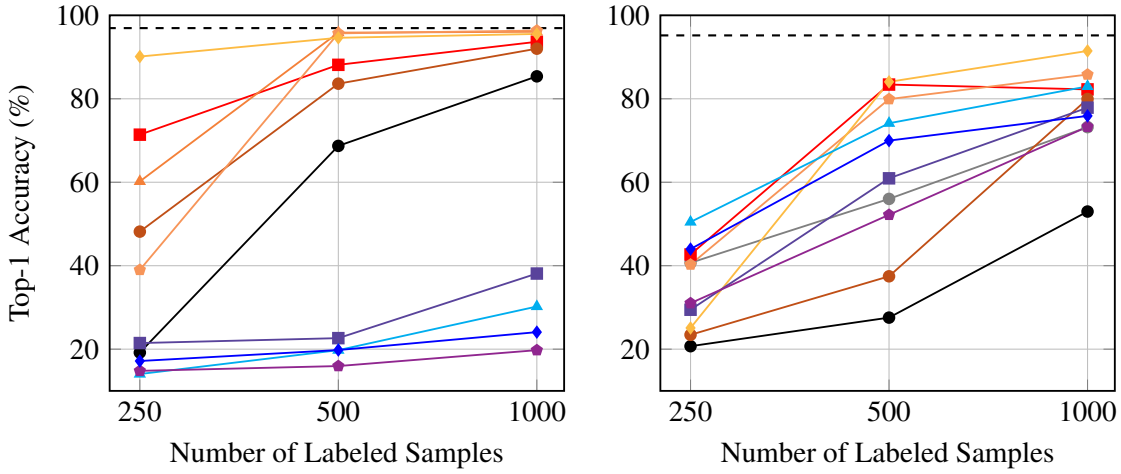
Figure 6.1: Legend for the benchmark results. Each combination of color and mark corresponds to a method. The first row represents Supervised Learning methods, the second row contains semi-supervision methods, and the third row includes self-supervision methods. Semi-supervised methods are depicted using shades of red, while self-supervised methods are shown with shades of blue.



(a) Wide ResNet28-2 model.

(b) ResNet50 model.

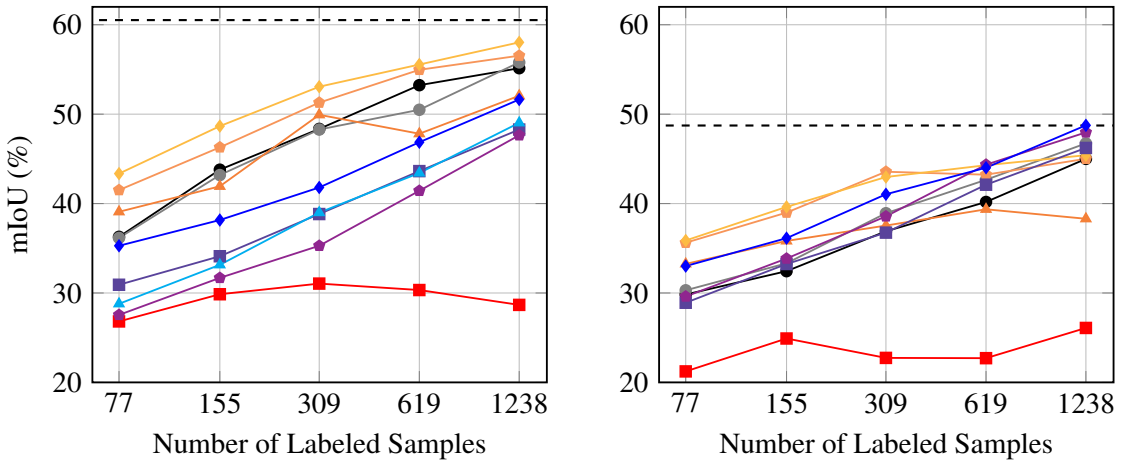
Figure 6.2: Top-1 Accuracy comparison between the methods for CIFAR-10.



(a) Wide ResNet28-2 model.

(b) ResNet50 model.

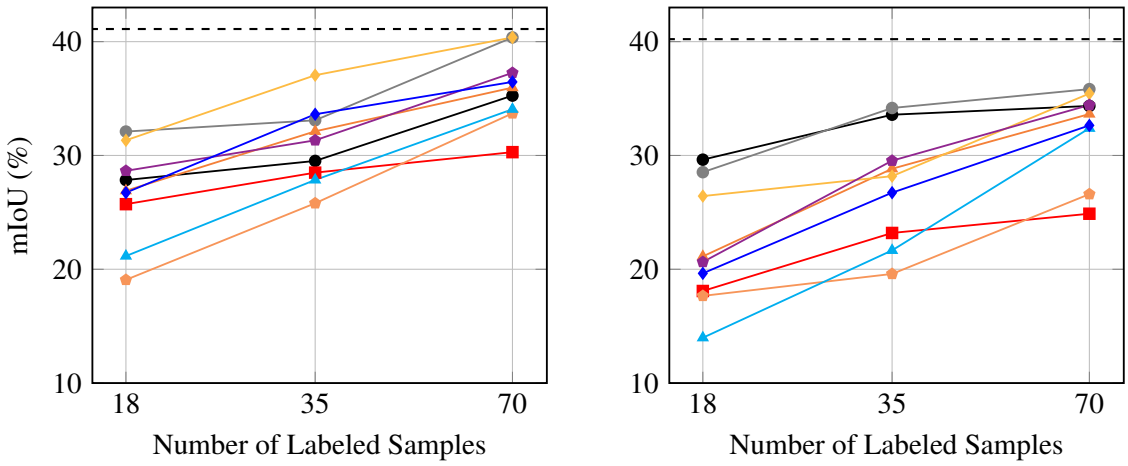
Figure 6.3: Top-1 Accuracy comparison between the methods for SVHN.



(a) ResNet101 backbone.

(b) MobileNetV3-Large backbone.

Figure 6.4: mIoU (%) comparison between the methods for Cityscapes using DeepLabv3 model.



(a) ResNet101 backbone.

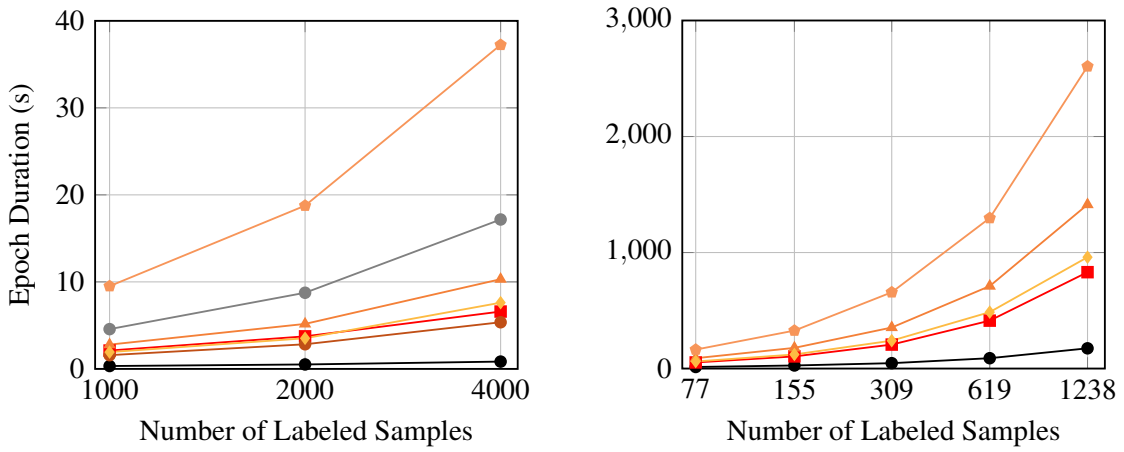
(b) MobileNetV3-Large backbone.

Figure 6.5: mIoU (%) comparison between the methods for KITTI using DeepLabv3 model.

6.1.2 Computational Efficiency

In this section, we examine the computational efficiency of the methods, focusing on the time taken per epoch. This evaluation helps to understand the trade-offs between performance and computational cost, guiding the selection of methods based on available resources and time constraints.

Tables 6.7 and 6.8 present the epoch durations for the experiments conducted on the CIFAR-10 dataset using the Wide ResNet28-2 model and the Cityscapes dataset using the ResNet101 backbone, respectively. We did not include these evaluations for every dataset and model combination, as the aim is to provide a general overview and facilitate relative comparisons between methods for each task. This approach ensures consistency in comparisons, regardless of the dataset and model used. For CIFAR-10, a row corresponding to the fine-tuning process was added, as image resizing had implications for the training duration. A separate table was also created for the self-supervision methods, as their training times are independent of the labeled data partition (Table 6.9).



(a) CIFAR-10 using Wide ResNet28-2.

(b) Cityscapes using ResNet101 as the backbone.

Figure 6.6: Mean epoch duration (s) comparison between the methods.

6.2 Discussion

This chapter provides an analysis of the experimental results presented in the previous section. We explore the comparative performance of semi-supervised, self-supervised, and supervised methods, examining the impact of different model architectures and training strategies on overall performance and efficiency. Through this discussion, we aim to highlight key insights, address notable observations, and provide a comprehensive understanding of the strengths and limitations of each approach.

Table 6.7: Epoch duration (seconds) for CIFAR-10. All methods were tested using Wide ResNet28-2 and the same codebase.

Method	1000 labels	2000 labels	4000 labels
Supervised Learning	0.33±0.01	0.51±0.03	0.84±0.03
Fine-tunning	4.57±0.03	8.75±0.05	17.17±0.06
Π-Model	2.11±0.03	3.74±0.21	6.59±0.14
Temporal Ensembling	1.58±0.02	2.83±0.04	5.36±0.10
MixMatch	2.79±0.05	5.17±0.03	10.31±0.49
ReMixMatch	9.51±0.02	18.76±0.04	37.24±0.34
FixMatch	1.93±0.03	3.52±0.05	7.61±0.44

Table 6.8: Epoch duration (seconds) for Cityscapes. All methods were tested using DeepLabv3 (ResNet101 backbone) and the same codebase.

Method	$77_{(1/32)}$	$155_{(1/16)}$	$309_{(1/8)}$	$619_{(1/4)}$	$1238_{(1/2)}$
Supervised Learning	13.54±0.09	26.42±0.29	46.34±0.44	89.20±0.06	174.31±0.10
Π-Model	52.55±0.15	104.60±0.65	206.52±0.05	412.98±0.04	830.65±0.21
MixMatch	89.08±0.13	178.24±0.11	353.69±0.20	710.82±0.23	1413.35±0.41
ReMixMatch	163.33±0.02	327.06±0.21	657.25±9.24	1297.08±4.49	2604.54±8.18
FixMatch	60.83±0.06	121.23±0.08	240.40±0.11	486.34±9.41	959.83±0.09

Table 6.9: Epoch duration (seconds) of self-supervision methods for CIFAR-10 and Cityscapes. All methods were tested using the same codebase.

Method	Wide ResNet28-2	ResNet101
Rotation	20.757±0.078	2397.229±1.673
SimCLR	395.335±0.152	896.545±49.879
BYOL	497.620±5.741	3358.129±59.964
MoCo	209.849±0.456	1011.251±9.537

6.2.1 Intra-Paradigm Method Analysis

This section analyzes the performance of various methods within the same learning paradigm, examining their strengths and weaknesses based on key performance metrics, such as accuracy and mean Intersection over Union.

6.2.1.1 Semi-Supervised Learning

Semi-supervised learning methods generally displayed equivalent or superior performance to supervised learning, with rare instances of underperformance.

- **Π-Model:** Known for its simplicity, this method relies solely on consistency regularization. However, it exhibited lower performance compared to other semi-supervised methods.
- **Temporal Ensembling:** An enhancement of the Π-Model, Temporal Ensembling aims to achieve similar results with improvements in speed and memory efficiency. Despite this, it consistently performed worse than the Π-Model in our experiments, likely due to the disabling of data shuffling (as explained in Section 5.4.2.2), making direct comparison less fair.
- **MixMatch:** This method significantly improved performance over the baselines and the Π-Model. While not always the top performer, it occasionally achieved the best results in classification tasks.
- **ReMixMatch:** Generally the second-best semi-supervised method, ReMixMatch often outperformed earlier methods. However, it showed a notable decrease in performance on the KITTI dataset, likely due to the disabled self-supervision rotation task, which appears crucial for this method. It also showed some instability when used with lower labeled data partitions.
- **FixMatch:** The most consistent in terms of results, FixMatch outperformed other semi-supervised methods, demonstrating reliability and making it an excellent starting point for semi-supervised learning.

In summary, the performance of semi-supervised methods generally improved with an increase in the proportion of labeled data, with more pronounced differences in lower data partitions. FixMatch, in particular, demonstrated superior performance, consistently achieving higher results compared to other semi-supervised methods. Regarding label efficiency, FixMatch and ReMixMatch showed that high performance can be achieved with minimal labeled data, highlighting their potential for reducing annotation costs in practical applications.

6.2.1.2 Self-Supervised Learning

While some self-supervised learning methods showed improvements over the supervised baseline, a lot of times they did not surpass it by a significant margin.

- **Rotation:** This method serves as a good starting point for self-supervised learning due to its simplicity and consistent performance across various datasets.
- **SimCLR:** Despite not being able to use larger batch sizes due to memory constraints, as recommended by the authors, SimCLR generally performed well compared to other self-supervised methods.
- **BYOL:** Among the self-supervised methods, BYOL generally exhibited the lowest performance. However, for the semantic segmentation task, it ranked among the top three methods for some partitions.
- **MoCo:** Overall, MoCo emerged as the best self-supervision method. Notably, it was the only method to surpass the fully supervised baseline, achieving this with the Cityscapes dataset using the MobileNetV3-Large backbone.

In summary, the MoCo method stood out by outperforming other methods in most cases, although SimCLR and Rotation achieved better results in some experiments. Unlike semi-supervised learning, self-supervised methods showed less variation in their performance, with their results being more closely clustered.

6.2.2 Cross-Paradigm Evaluation

This section presents a comparative analysis between different learning paradigms, specifically on semi-supervision and self-supervision.

Performance Comparison When comparing overall performance across paradigms, Semi-Supervised Learning methods generally outperformed Self-Supervised Learning methods. This trend was consistent across various datasets and metrics, suggesting that the combination of labeled and unlabeled data in the same training instance provides a significant advantage in achieving superior results. Nevertheless, both paradigms offer substantial improvements over fully supervised baselines.

Label Efficiency The evaluation also highlights the label efficiency of different paradigms. Semi-supervision methods, particularly FixMatch, demonstrated the ability to achieve high performance with minimal labeled data, making them more label-efficient than self-supervision methods.

Data Balance In scenarios where the amount of unlabeled data significantly exceeds the amount of labeled data, self-supervised learning methods have a distinct advantage, as they are designed to leverage vast amounts of unlabeled data. Semi-supervised methods, on the other hand, depend on the batch size ratio between labeled and unlabeled data. If this ratio is not well-balanced, a significant portion of the unlabeled data may be discarded, limiting the method's ability to fully exploit the available data. Therefore, while semi-supervised methods can perform exceptionally

well with a balanced mix of labeled and unlabeled data, their efficiency may decrease when there is a large disparity between the two.

In summary, the cross-paradigm evaluation reveals that while both paradigms have their strengths, Semi-Supervised Learning methods generally offer better performance and efficiency. However, self-supervised methods could hold significant value in scenarios with a large disparity between labeled and unlabeled data.

6.2.3 Backbone Performance Comparison

This section evaluates the impact of large versus small backbones on model performance and examines how these architectures interact with self-supervised and semi-supervised learning methods.

Backbone Impact The size of the feature extraction network, or backbone, is crucial in determining the overall performance of a model. With their increased capacity and more complex architectures, larger backbones generally provide better performance by learning more intricate patterns from the data. However, they also come with increased computational costs and longer training times. In contrast, smaller backbones are more computationally efficient and faster to train, though they may not capture the same level of detail as larger ones.

Methods Effectiveness Semi-supervised methods consistently achieve better results than the baseline regardless of backbone size. These methods effectively utilize both labeled and unlabeled data, enhancing performance even with smaller backbones. On the other hand, self-supervised methods reach their full potential primarily when using larger backbones. This was particularly evident in the classification datasets, where self-supervised methods with smaller backbones often failed to surpass the baseline performance, rendering them less effective.

In summary, backbone size has a more pronounced impact on self-supervised methods. While semi-supervised methods perform well with both small and large backbones, self-supervised methods require larger backbones to achieve significant performance gains. This suggests that using Self-Supervised Learning with smaller backbones may be less beneficial and, in some cases, may not offer improvements over baseline methods.

6.2.4 Baseline Comparison with Pre-Trained Models

This section compares pre-trained models with the other learning paradigms, focusing on their convergence speed and overall performance.

6.2.4.1 Performance Enhancement

Supervised Learning Regarding performance, pre-trained models generally surpassed models with random weight initialization by a considerable margin, particularly in low data partitions and when using larger backbones. The exception was the Cityscapes dataset, where this trend was less pronounced.

Semi-Supervised Learning Semi-supervised methods consistently outperformed pre-trained models. Methods such as ReMixMatch and FixMatch achieved remarkably higher performance, demonstrating that combining labeled and unlabeled data in the same training instance is more label-efficient than using a model trained on an external dataset.

Self-Supervised Learning Unlike Semi-Supervised Learning, self-supervision methods did not achieve competitive results against the pre-trained baseline. Across the majority of datasets and model combinations, self-supervised methods rarely surpassed the pre-trained baseline, and when they did, it was only by a small margin. This suggests that representations learned from an external dataset might perform better than those learned from the same dataset. However, this conclusion depends on the choice of the external dataset, and further experiments with models pre-trained on other, especially related, datasets are needed to determine if this approach is generally better than self-supervised learning.

6.2.4.2 Convergence Speed

Empirically, pre-trained models achieved better performance in the early training stages than any learning paradigm or the random weight initialization baseline. Although further training allowed other paradigms to catch up, pre-trained models are advantageous when training resources or time are limited.

In summary, pre-trained models offer significant performance and convergence speed advantages compared to models with random weight initialization. While semi-supervised methods surpass the performance of pre-trained models, self-supervised methods, although beneficial, generally do not outperform pre-trained models. Pre-trained models are especially useful for achieving quick, high performance when training resources or time are constrained.

6.2.5 Training Time Analysis

This section examines the training time required for different learning paradigms, focusing on semi-supervised, self-supervised, and supervised methods. The analysis also considers the impact of using pre-trained models on training duration.

Supervised Learning Supervised Learning has the lowest training time among the paradigms due to its straightforward approach. It solely uses labeled data to learn without adding complexity

to exploit the data, focusing on creating relationships between the input and given targets. Fine-tuning typically takes a similar amount of time, as the main difference lies in weight initialization. However, fine-tuning can occasionally take longer if modifications, such as image resizing, are needed, as observed in the classification tasks depicted in Table 6.7.

Semi-Supervised Learning Since semi-supervision methods also leverage unlabeled data, they are expected to take longer than the supervised approach. For instance, FixMatch takes approximately five times longer than supervised learning, which is justified by its superior results. ReMixMatch, however, can take over fifteen times longer. Despite the increased duration, these methods offer substantial performance improvements, making the extended training time reasonable.

Self-Supervised Learning Self-supervision methods tend to have the longest training times. Unlike semi-supervised methods that may face batch size restrictions, they can utilize the entire unlabeled dataset in a single epoch, resulting in longer epochs and total training duration. Additionally, Self-Supervised Learning involves a second training phase for fine-tuning, further increasing the total time and resources required. Although not directly comparable to Semi-Supervised Learning due to this secondary training phase, the epoch duration for self-supervised methods is significantly higher, as shown in Table 6.9. This can be mitigated by reducing the number of epochs with more data or limiting the number of samples per epoch.

In summary, training time varies significantly across learning paradigms. Supervised Learning is the quickest due to its simplicity, while Semi-Supervised Learning, although taking longer, provides substantial performance gains. Self-Supervised Learning has the longest training duration due to its comprehensive use of unlabeled data and subsequent fine-tuning phase. The extended training times for semi-supervised and self-supervised methods are justified by their performance improvements, making them valuable despite the increased resource requirements.

6.2.6 Observations

This section highlights notable observations from the experimental results, focusing on anomalies and significant outcomes that provide deeper insights into the performance of the different learning paradigms.

Performance Fluctuations with Increased Labeled Data In some cases, increasing the number of labeled samples unexpectedly reduced performance compared to a previous partition. For example, the Π-Model in the Cityscapes dataset and ReMixMatch in the CIFAR-10 dataset exhibited such behavior. These fluctuations might be attributed to several factors, including overfitting, where the model becomes too tailored to the training data, or the introduction of noise and inconsistencies in the newly added labeled data, which could disrupt the learning process.

MoCo Surpassing Fully Supervised Baseline The MoCo self-supervised method was the only experiment that managed to surpass the fully supervised baseline (which utilizes the entire dataset and not a partition of it), which occurred specifically in the Cityscapes dataset with the MobileNetV3-Large backbone, using half of the original data. This observation underscores the potential of Self-Supervised Learning to achieve exceptional results and highlights the importance of selecting appropriate pretext tasks and architectures to maximize the benefits of self-supervision methods.

Chapter 7

Conclusions

7.1 Final Remarks

In conclusion, this thesis explored the efficacy of Semi-Supervised and Self-Supervised Learning methods in addressing challenges related to label efficiency. The objectives of implementing and adapting these methods to the domain of Autonomous Driving were completed. Our experiments, conducted on various datasets, including CIFAR-10, SVHN, Cityscapes, and KITTI, provided a comprehensive evaluation of these paradigms across different tasks and model architectures. Additionally, a software framework was concurrently developed to facilitate the integration of these approaches in various domains.

Semi-supervised approaches leverage both labeled and unlabeled data within the same training instances, employing techniques such as consistency regularization, entropy minimization, and pseudo-labeling. Their results demonstrated that semi-supervision methods, especially FixMatch and ReMixMatch, consistently outperformed both self-supervision methods and supervised baselines. This highlights the potential of semi-supervised approaches in achieving high performance with minimal labeled data, offering significant advantages in terms of label efficiency and cost-effectiveness. However, these methods are usually specific to a particular machine-learning task and require substantial effort to be adapted to other tasks.

On the other hand, Self-supervised approaches extract meaningful representations of unlabeled data by creating secondary tasks derived from data augmentation techniques, with subsequent training involving fine-tuning the model using different data, annotated or not. While not as consistently strong as semi-supervised methods, they showed promise, mainly when larger backbones were utilized. This suggests that Self-Supervised Learning can be effective, but its success heavily depends on the model architecture and the quality of the self-supervised pretext tasks.

Our analysis also revealed the importance of backbone size in determining model performance, with larger models generally performing better but at the cost of increased computational resources and training time. Furthermore, pre-trained models notably boosted convergence speed and early-stage performance, underscoring the value of leveraging external datasets for pre-training.

In essence, this research contributes to the field of Autonomous Driving by harnessing mechanisms, namely semi-supervision and self-supervision, that leverage unlabeled data, minimizing the need for the labor-intensive process of annotating data.

7.2 Future Work

While this thesis has provided valuable insights into the comparative performance of various learning paradigms, several avenues remain for future exploration:

Memory Comparison Future work could include a detailed comparison of the memory requirements for different learning methods. Understanding the memory footprint is crucial for deploying these methods in resource-constrained environments.

Autonomous Driving Tasks Extending this study to other autonomous driving tasks, such as object detection and segmentation using sensors beyond RGB cameras like LIDAR, would be beneficial. These sensors introduce additional challenges but are crucial for robust autonomous driving systems. Although projecting LIDAR data into image form is common, making these methods potentially suitable, their application in this context requires thorough investigation.

Exploring More Methods Investigating a broader range of semi-supervised and self-supervised methods could uncover new techniques that offer even greater performance and efficiency. Examples of methods that could be explored include FlexMatch [54], S4L [63], Dense FixMatch [73], FixMatchSeg [77], SimSiam [78], SwAV [79], CMC [80], and PIRL [81]. Continuous advancements in these fields warrant ongoing exploration and evaluation.

References

[1] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. On-Road Automated Driving (ORAD) Committee, April 2021.

[2] M Nadeem Ahangar, Qasim Z Ahmed, Fahd A Khan, and Maryam Hafeez. A survey of autonomous vehicles: Enabling communication technologies and challenges. *Sensors*, 21(3):706, 2021.

[3] Florian Alexander Schmidt. Crowdsourced production of AI Training Data: How human workers teach self-driving cars how to see. Working Paper Forschungsförderung 155, Düsseldorf, 2019.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. 2009.

[5] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, Spain, 2011.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 25, 2012.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Muneeb ul Hassan. Vgg16 - convolutional network for classification and detection, May 2023. `https://neurohive.io/en/popular-networks/vgg16/`, accessed on 04/12/2023.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper With Convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[16] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[18] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[19] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[20] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

[21] Khanhvi Tran, Johan Peter Bøtker, Arash Aframian, and Kaveh Memarzadeh. Artificial intelligence for medical imaging. In *Artificial Intelligence in Healthcare*, pages 143–162. Elsevier, 2020.

[22] Terrance DeVries and Graham W Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[23] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[24] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning Augmentation Strategies From Data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

[25] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical Automated Data Augmentation With a Reduced Search Space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[26] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[27] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. *Supervised Learning*, pages 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[28] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-Supervised Learning (Chapelle, O. et al., Eds.; 2006) [Book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[29] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A Survey on Deep Semi-Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):8934–8954, 2023.

[30] Veenu Rani, Syed Tufael Nabi, Munish Kumar, Ajay Mittal, and Krishan Kumar. Self-supervised Learning: A Succinct Review. *Archives of Computational Methods in Engineering*, 30(4):2761–2775, 2023.

[31] Burr Settles. Active Learning Literature Survey. 2009.

[32] James Foulds and Eibe Frank. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(1):1–25, 2010.

[33] Daren C. Brabham. Crowdsourcing as a Model for Problem Solving: An Introduction and Cases. *Convergence*, 14(1):75–90, 2008.

[34] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC press, 2012.

[35] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622, 2008.

[36] Rion Snow, Brendan O'connor, Dan Jurafsky, and Andrew Y Ng. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 254–263, 2008.

[37] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.

[38] Longlong Jing and Yingli Tian. Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

[39] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-Supervised Representation Learning by Rotation Feature Decoupling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10364–10374, 2019.

[40] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful Image Colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 649–666. Springer, 2016.

[41] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[42] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised Visual Representation Learning by Context Prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[43] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[44] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with Pseudo-Ensembles. *Advances in neural information processing systems*, 27, 2014.

[45] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning. *Advances in neural information processing systems*, 29, 2016.

[46] Samuli Laine and Timo Aila. Temporal Ensembling for Semi-Supervised Learning. In *International Conference on Learning Representations*, 2017.

[47] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. *Advances in neural information processing systems*, 32, 2019.

[48] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised Data Augmentation for Consistency Training. *Advances in neural information processing systems*, 33:6256–6268, 2020.

[49] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[50] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[51] Yves Grandvalet and Yoshua Bengio. Semi-supervised Learning by Entropy Minimization. *Advances in neural information processing systems*, 17, 2004.

[52] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

[53] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

[54] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling. *Advances in Neural Information Processing Systems*, 34:18408–18419, 2021.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[56] Geoffrey J McLachlan. Iterative Reclassification Procedure for Constructing an Asymptotically Optimal Rule of Allocation in Discriminant Analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.

[57] Henry Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[58] 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*, volume 3, page 896. Atlanta, 2013.

[59] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[60] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] Tim Salimans and Durk P Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *Advances in neural information processing systems*, 29, 2016.

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

[63] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4L: Self-Supervised Semi-Supervised Learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1476–1485, 2019.

[64] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, et al. Usb: A unified semi-supervised learning benchmark for classification. *Advances in Neural Information Processing Systems*, 35:3938–3961, 2022.

[65] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-Supervised Learning: Generative or Contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876, 2021.

[66] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18(7):1527–1554, 2006.

[67] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[68] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Advances in neural information processing systems*, 27, 2014.

[69] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[70] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[71] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*, 2015.

[72] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised Learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

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

[74] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[75] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[76] Haiming Xu, Lingqiao Liu, Qiuchen Bian, and Zhen Yang. Semi-supervised semantic segmentation with prototype-based consistency regularization. *Advances in neural information processing systems*, 35:26007–26020, 2022.

[77] Pratima Upretee and Bishesh Khanal. Fixmatchseg: Fixing fixmatch for semi-supervised semantic segmentation. *arXiv preprint arXiv:2208.00400*, 2022.

[78] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[79] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[80] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive Multiview Coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[81] Ishan Misra and Laurens van der Maaten. Self-Supervised Learning of Pretext-Invariant Representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6707–6717, 2020.