FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

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

Francisco Gonçalves Cerqueira

DISSERTATION PLANNING

Mestrado em Engenharia Informática e Computação

Supervisor: Ricardo Pereira de Magalhães Cruz

February 5, 2024

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

Francisco Gonçalves Cerqueira

Mestrado em Engenharia Informática e Computação

February 5, 2024

Resumo

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

As aprendizagens semi-supervisionadas e auto-supervisionadas enfrentam este desafio incorporando dados não rotulados 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. 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 exploram dados rotulados e não rotulados e incorporam essa perspicácia usando as previsões como pseudo-rótulos e atribuindo-os a quaisquer transformações de imagem quando a confiança do modelo nessas previsões é suficientemente alta. Abordagens de auto-supervisão operam na ausência de dados anotados, introduzindo um termo na função de perda que penaliza classificações inconsistentes para a mesma imagem modificada. 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 no domínio de Condução Autónoma, adaptando-os para tarefas mais proeminentes, como a segmentação semântica. Vários conjuntos de dados públicos estão disponíveis, sendo o KITTI um dos mais populares na literatura. Um pacote de software prático será desenvolvido como um objetivo adicional, permitindo que outros projetos e áreas beneficiem destas técnicas.

Palavras-chave: semi-supervised learning · self-supervised learning · label efficiency · autonomous driving · computer vision · deep learning · semantic segmentation

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.

Semi-supervised and self-supervised learning 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. 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 exploit labeled and unlabeled data and incorporate this insight by using the predictions as pseudo-labels and attributing them to any image transformations when the model's confidence in those predictions is sufficiently high. Self-supervision frameworks operate in the absence of annotated data, introducing a term in the loss function that penalizes inconsistent classifications for the same modified image. 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 prominent tasks, such as semantic segmentation. Several public datasets are available, with KITTI being one of the most popular in research. A practical software package will be developed as an additional objective, enabling other projects and fields to benefit from these techniques.

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

Acknowledgements

This chapter has been intentionally left blank.

Francisco Gonçalves Cerqueira

"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
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.

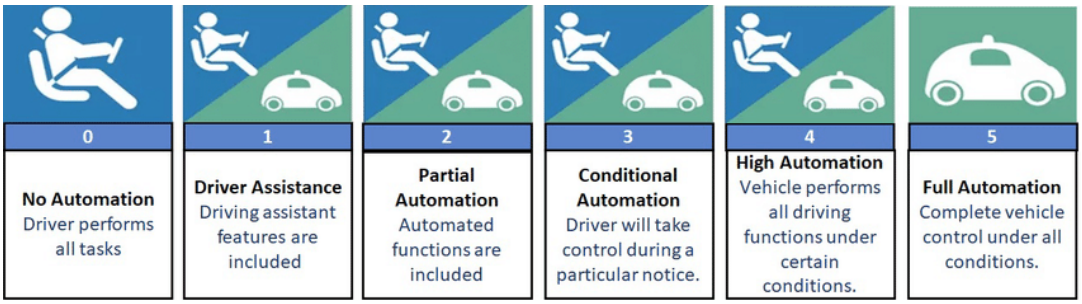

Figure 1.1: SAE J3016 levels of driving automation. Source: [2]

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 Faculdade de Engenharia da Universidade do Porto 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 will be initially evaluated on datasets commonly used in this literature, such as CIFAR10 [4]. They will then be adapted for more prominent tasks in autonomous driving, namely semantic segmentation and lane estimation. We intend to address these types of tasks because they represent a gradual step beyond classification tasks – in fact, we can think of these tasks as pixel-by-pixel classification. Nevertheless, some special considerations will be 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 package that implements pertinent losses associated with these techniques and serves as a valuable resource 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 evaluation of their performance underlining the practical implications of these advancements.

1.4 Main Contributions

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

[1] https://www.inesctec.pt/pt/projetos/theia

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 when applied to the specific challenges of autonomous driving. This includes assessing the label efficiency of these approaches and determining their ability to enhance model performance with limited annotated data.

3. Development of a software package containing relevant losses tailored for these methods and other relevant auxiliary functions.

4. Establishment of an open-source repository hosting the developed code for 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 topics such as classification, semantic segmentation, and data augmentation.

- **Chapter 3** delves into the category-wise review of semi-supervised learning, exploring its categories and methods.

- **Chapter 4** provides a category-wise review of self-supervised learning, covering generative, contrastive, and adversarial methods. It also discusses specific methods within each category.

- **Chapter 5** outlines the proposed work for this research, presenting an overview of the planned comparative study and software development. Additionally, it covers the preliminary work conducted thus far and outlines the anticipated work plan for the upcoming phases.

- Concluding the thesis, **Chapter 6** succinctly encapsulates the essential discoveries and anticipated research outcomes.

Chapter 2

Background Knowledge

2.1 Deep Learning for Computer Vision

2.1.1 Classification

The classification task within the domain of machine learning 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.

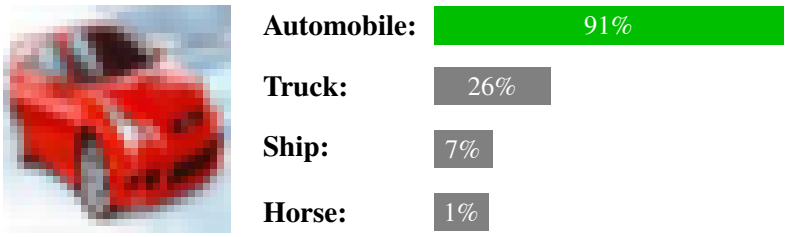

Figure 2.1: Image classification example, showcasing a CIFAR-10 [4] dataset image of an automobile alongside its classification probabilities, indicating the likelihood of the image being identified within a set of categories.

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) [5], AlexNet revolutionized Convolutional Neural Networks (CNNs) for high-resolution image recognition in the ImageNet [6] 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.

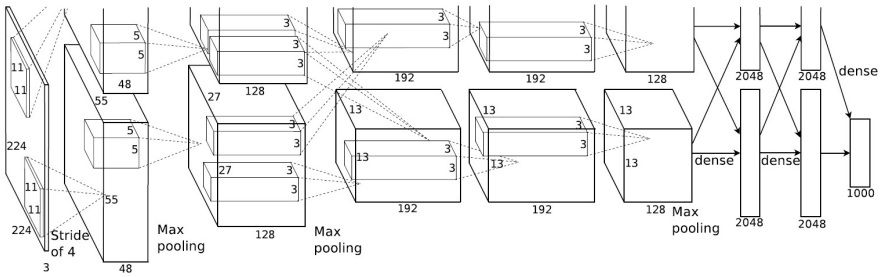

Figure 2.2: AlexNet architecture. Source: [5]

VGG The Visual Geometry Group (VGG) [7] 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 [6].

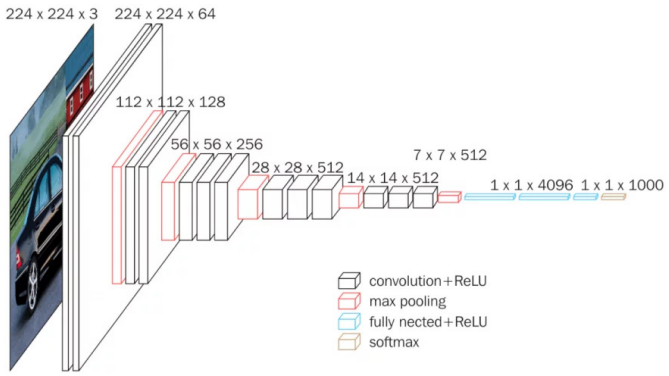

Figure 2.3: VGG16 architecture. Source: [8]

GoogLeNet GoogLeNet [9] 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.

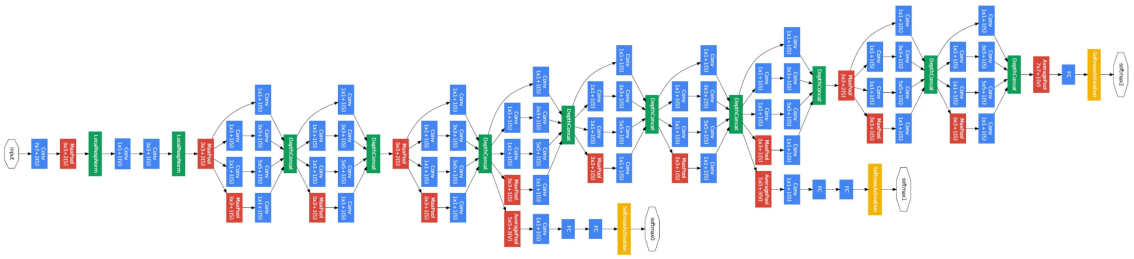

Figure 2.4: GoogLeNet architecture. Source: [9]

ResNet ResNet [10] 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.

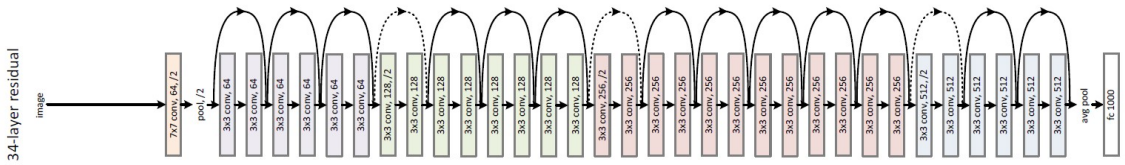

Figure 2.5: ResNet architecture. The dotted shortcuts increase dimensions. Source: [10]

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 Mean Squared Error (MSE) is a popular loss within semi-supervision methods adapted for classification problems. 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: [11]

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) [12] 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 [13]. Its distinctive U-shaped structure, depicted in Figure 2.8, features an encoder path for downsampling and a symmetric decoder path with multiple upsampling layers using learnable filters. The key

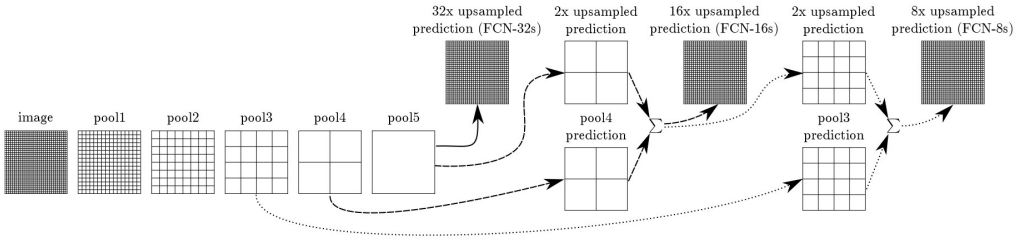

Figure 2.7: Fully Convolutional Network architecture. Source: [12]

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.

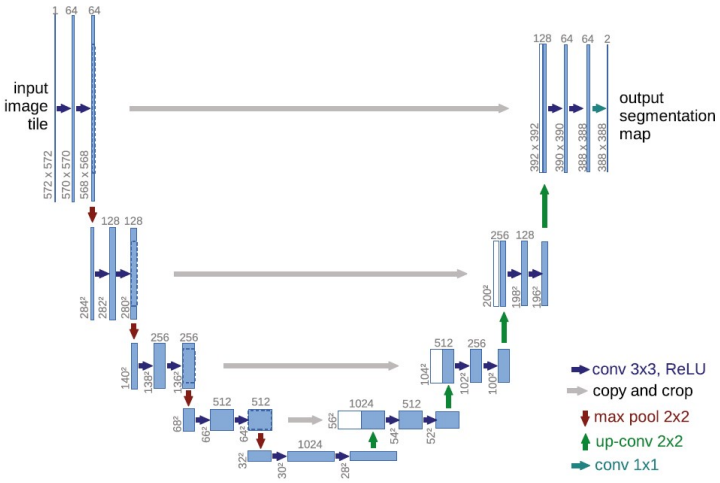

Figure 2.8: U-Net architecture. Source: [13]

SegNet Similar to U-Net, SegNet [14] (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) [15] 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.

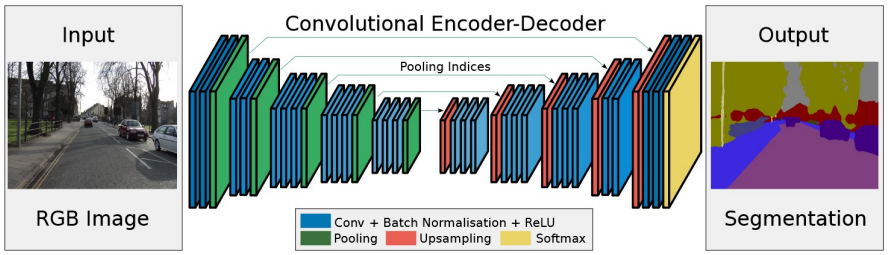

Figure 2.9: SegNet architecture. Source: [14]

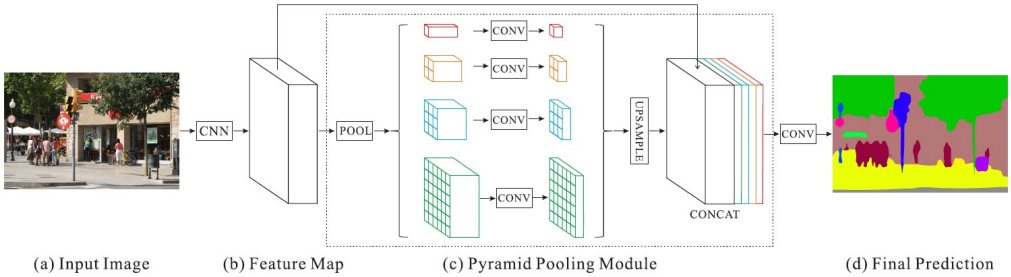

Figure 2.10: Pyramid Scene Parsing Network architecture. Source: [15]

DeepLabv3 DeepLabv3 [16] 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.

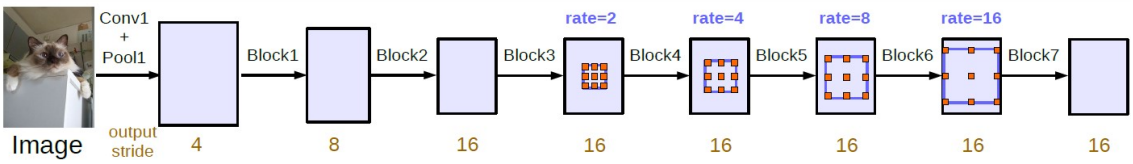

Figure 2.11: DeepLabv3 architecture. Source: [16]

Segmenter Transformers have also been applied in segmentation tasks [17]. 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.

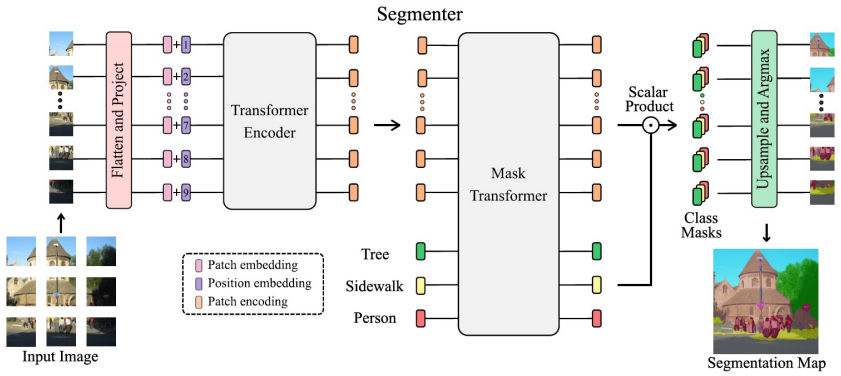

Figure 2.12: Segmenter architecture. Source: [17]

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 [18] 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

There are two main categories of data augmentation techniques:

- **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 [19]. 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.

Cutout To address the challenge of object occlusion frequently observed in various computer vision tasks, the Cutout [20] 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.

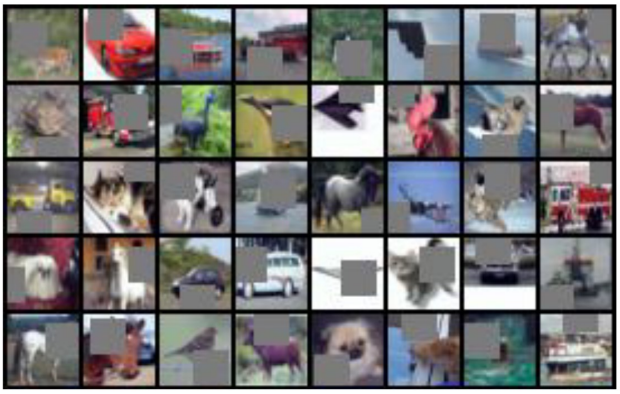

Figure 2.13: Cutout applied to images from the CIFAR-10 [4] dataset. Source: [20]

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 [21] 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 [22] is a technique 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 at hand. The available operations include rotation, translation, shearing, sample pairing, Cutout, and color adjustment.

RandAugment Utilizing only two hyperparameters, namely the quantity of augmentation operations and their respective magnitudes, the RandAugment [23] 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) [24] 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.

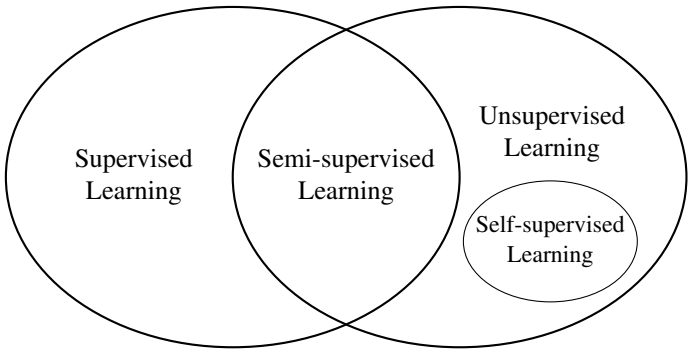

Figure 2.14: A visual representation of different learning paradigms: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Self-supervised Learning. This diagram illustrates the overlapping areas and distinctions between these fundamental approaches to machine learning.

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 [25]. 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 [26].

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 [27]. 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) [28, 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 [29], 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 [30].

- **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 [31], 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 [32], which combine predictions from multiple models or annotators to reduce the impact of inaccuracies, are common approaches to mitigate this problem [33, 34].

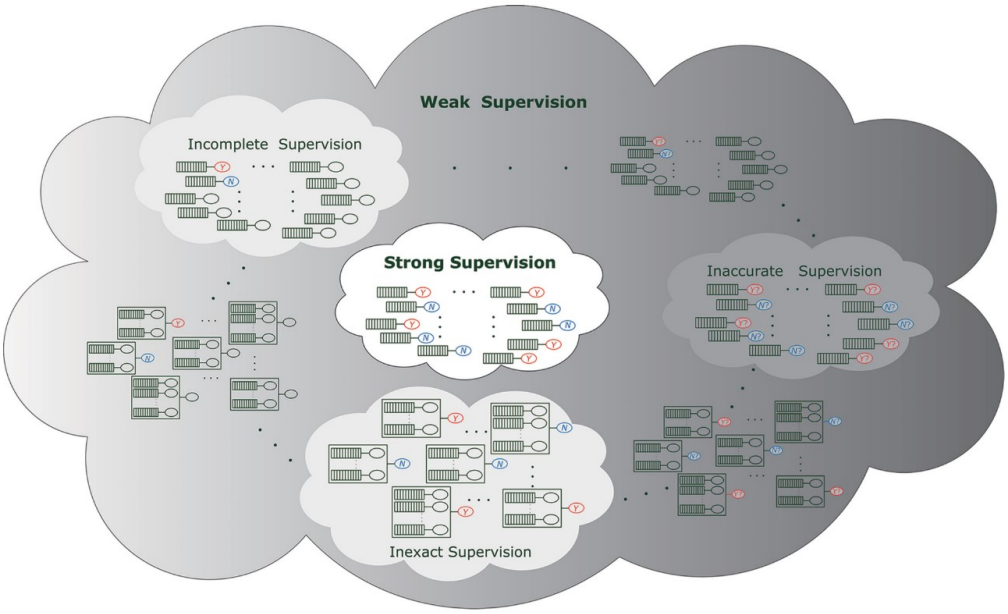

Figure 2.15: Illustration of the three typical types of weak supervision. Bars denote feature vectors, boxes represent instance-level samples, and "?" implies the inaccuracy of the label. The interpolated sub-graphs represent scenarios with mixed types of weak supervision. Source: [35]

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 [36]. 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 the acquisition of 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 [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 [38].
- **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 [39].
- **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 [40].

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 [36].

2.2.6 Transfer Learning

While Transfer Learning (TL) can be used with supervised, semi-supervised, or unsupervised approaches [41], 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 [42, 43, 44], 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. Commonly, this augmentation can be achieved through diverse techniques, including domain-specific data

transformations [44, 43, 45, 46], adversarial perturbations [47], dropout [44, 48], or Cross-entropy-based modifications [47, 46].

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 [44].

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

3.1.1.1 Augmentation Anchoring

Berthelot et al. (2019) [24] proposed a specific implementation of consistency regularization called augmentation anchoring, which establishes an "anchor" by subjecting the input to a weak augmentation. Then, it generates strong augmented versions from that same input, which would later be used to enforce coherence in the predictions compared to the weakly augmented sample. Augmentation anchoring's experimental validation demonstrated remarkable stability, allowing the replacement of the Mean Squared Error loss with a more straightforward standard Cross-entropy loss.

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) [49] 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 [50]. 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 [45, 24, 51, 52].

3.1.2.1 Temperature Sharpening

An alternative methodology frequently employed involves modifying the categorical distribution [53], a technique referred to as "temperature sharpening" or "temperature scaling". MixMatch [45] and ReMixMatch [24] 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 [54, 55]. 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 [56]. 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 [49, 43], where the model is encouraged to predict high-confidence outputs on unlabeled data.

3.2 Methods

3.2.1 Pi-Model

Pi-Model [44], 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.

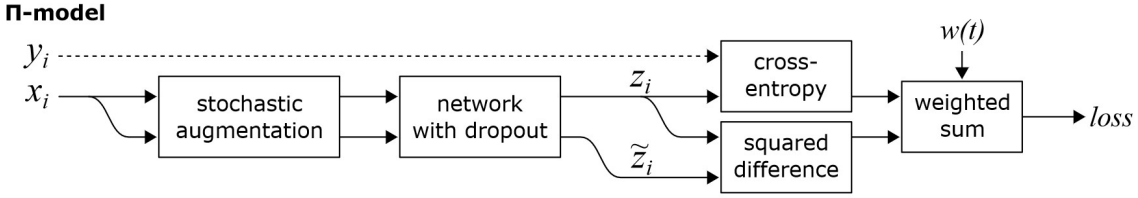

Figure 3.1: Diagram representing the loss computation for Π-Model. Distinct stochastic augmentations are applied to an input instance to enforce consistency regularization. The cross-entropy loss is also evaluated if the input is a labeled sample. Source: [44]

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 [57] 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) [44] 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.

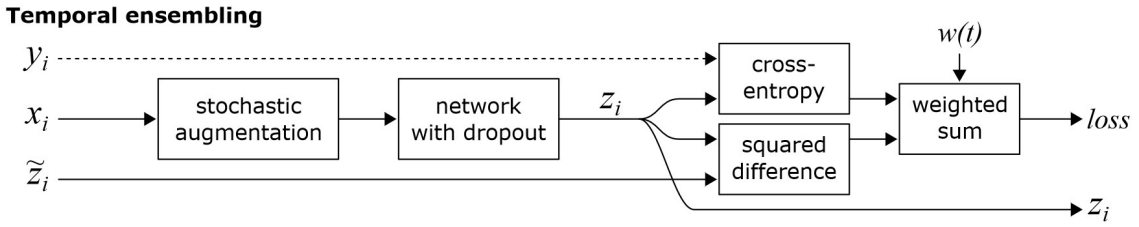

Figure 3.2: Diagram representing the loss computation for Temporal Ensembling. A stochastic augmentation is applied to the input instance to enforce consistency regularization later. The Cross-entropy loss is also evaluated if the input is a labeled sample. The ensemble prediction $\tilde{z}$ incorporates the Exponential Moving Average mechanism for a more responsive adaptation to the evolving patterns in the training data. Source: [44]

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 [58] and mean-only batch normalization [59]. 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

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

3.2.3 MixMatch

MixMatch [45] 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 3.1.2.1, generating the label guess for all the augmented images.

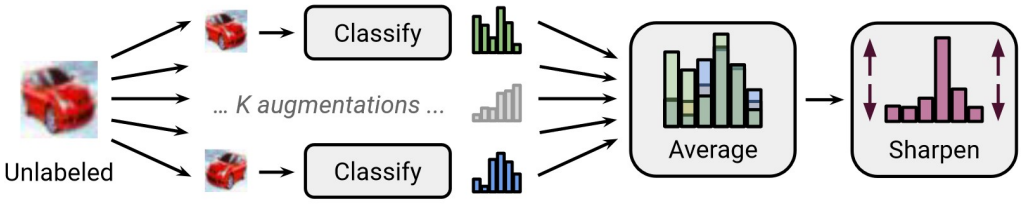

Figure 3.3: MixMatch pseudo-labeling algorithm diagram. For a single unlabeled sample, K augmented images are generated using stochastic data augmentation and fed to the classifier to predict their label. Then, the average of the obtained distribution is "sharpened" by adjusting the distribution's temperature. Source: [45]

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 [21], 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) [21], 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.

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

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

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

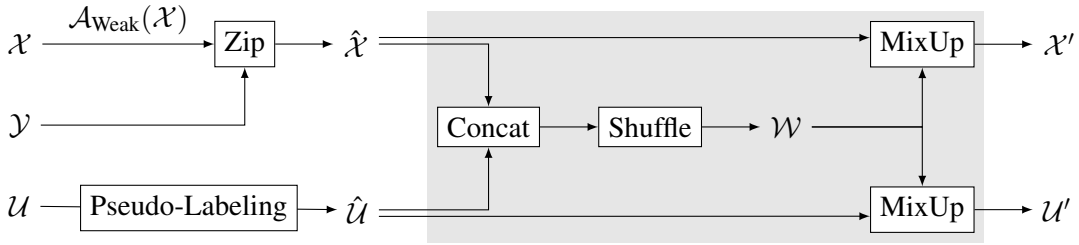

Figure 3.4: Diagram of the data augmentation technique used in MixMatch. A pseudo-labeling step is applied for the unlabeled data to create target labels, while the labeled data only benefits from weak data augmentation. The grey area represents the last optional step of combining them to apply MixUp 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 [44, 48, 43].

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

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

3.2.4 ReMixMatch

ReMixMatch [24] 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.

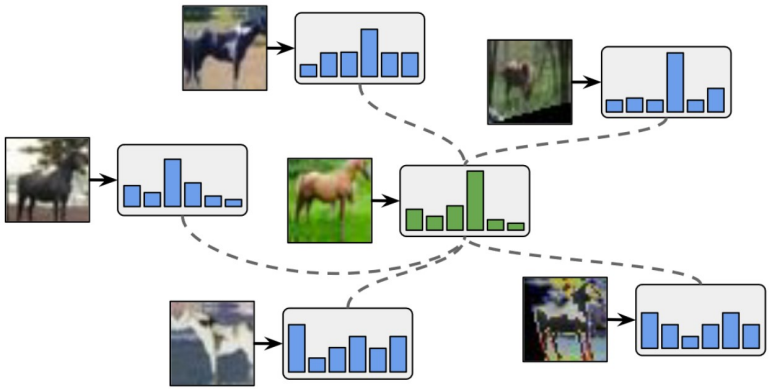

Figure 3.5: ReMixMatch augmentation anchoring uses the prediction for a weakly augmented image (green, middle) as the target for predictions on strong augmentations of the same image (blue). Source: [24]

CTAugment Creating robust augmentations is integral to enhancing model performance. However, the challenge lies in efficiently generating these augmentations. While AutoAugment [22]

achieves high validation accuracy, its reliance on labeled data for policy learning presents difficulties in scenarios with limited labels, such as SemiSL. RandAugment [23] 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.

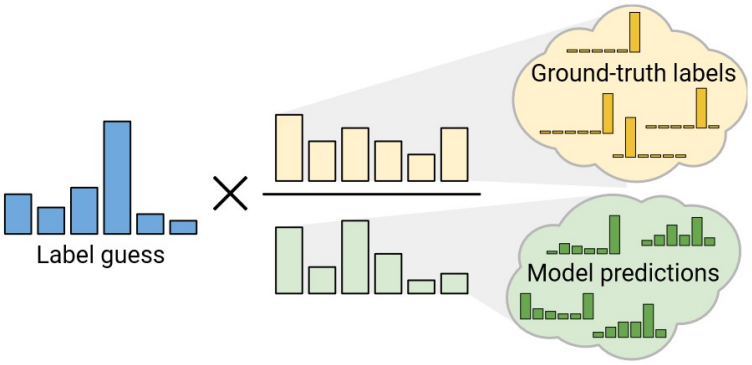

Figure 3.6: ReMixMatch distribution alignment diagram. The guessed label distributions are adjusted according to the empirical ground-truth class distribution ratio divided by the average model predictions on unlabeled data. Source: [24]

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 25. 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

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

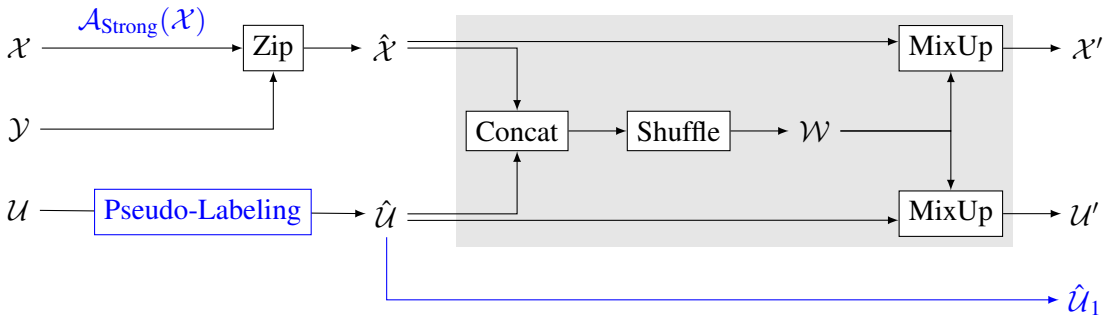

Figure 3.7: Diagram of the data augmentation technique used in ReMixMatch. A pseudo-labeling step is applied for the unlabeled data to create target labels, while the labeled data only benefits from strong data augmentation. The grey area represents the last optional step of combining them to apply MixUp 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) [60] and Zhai et al. (2019) [61], 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

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

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.

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 $P(\mathcal{X}, \mathcal{Y})$ of input data $\mathcal{X}$ and target labels $\mathcal{Y}$, focus on calculating the probability $P(\mathcal{X} \mid \mathcal{Y} = y)$, capturing how inputs might be generated given specific outcomes. Conversely, discriminative models aim to model the conditional probability $P(\mathcal{Y} \mid \mathcal{X} = x)$, centering on predicting labels given particular inputs [62].

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 [63, 64, 65]. 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

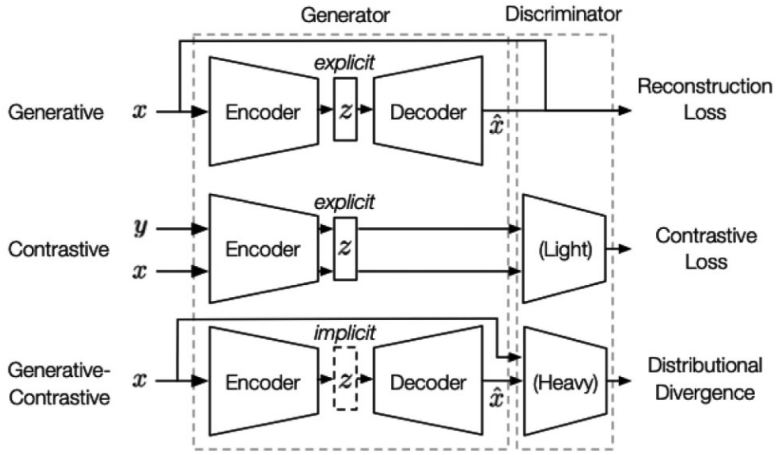

Figure 4.1: Conceptual comparison between self-supervision categories: Generative, Contrastive, and Generative-Contrastive. Source: [62]

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 [66, 67]. 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) [68], 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 [62]. 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.

The upcoming research will scrutinize several state-of-the-art methods, such as SimCLR [66], MoCo [67], BYOL [69], SimSiam [70], SwAV [71], CMC [72], and PIRL [73]. This investigation aims to unveil the unique strengths of each method, considering that each employs diverse strategies to extract meaningful representations from data.

Chapter 5

Proposed Work

5.1 Proposed Solution

As this research focuses on addressing the challenge of limited labeled data, the proposed approach entails the investigation and application of state-of-the-art techniques, specifically semi-supervision and self-supervision methods. The initial phase involves implementing and validating these methods using specified criteria, such as tasks, datasets, and models, aligned with the ones used by the authors. Subsequent phases include adapting these methods to the field of autonomous driving, incorporating suitable tasks (e.g., semantic segmentation), datasets (e.g., KITTI [74]), losses (e.g., Dice loss), and metrics (e.g., Dice coefficient), while conducting a comprehensive comparative analysis within this domain.

Simultaneously, the development of a software package is planned alongside the implementation of these methods. This package will include pertinent losses and methods supporting the integration of these techniques, with the objective of simplifying the adaptation of these methods to various domains and projects.

5.2 Preliminary Work

5.2.1 Experimental Setup

During this semester, preliminary work was undertaken to gain practical insights into the training processes of Supervised and Semi-Supervised Learning. The experimentation involved three trials evaluated on the CIFAR-10 [4] dataset, with a partition into train/validation/test sets (90% train/val split). The labeled/unlabeled split was performed using the train split, and a Wide ResNet [75] (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.

Supervised Learning was applied in two experiments, following the specifications recommended by [75]. Stochastic Gradient Descent with *Nesterov* momentum and Cross-entropy loss were used. The initial learning rate was set to 0.1, weight decay to 0.0005, dampening to 0, momentum to 0.9, and minibatch size to 128. The learning rate decreased by 0.2 at 60, 120, and

160 epoch milestones. The first experiment utilized the entire training split (45000 samples) and trained for 200 epochs, while the second experiment used only 4000 samples and trained for 300 epochs.

The third experiment employed the semi-supervised Π-Model [44] method. In contrast to the previous experiments, Adam [58] optimization was used with an initial learning rate of 0.003, and momentum parameters were set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was adjusted using exponential ramp-up and ramp-down functions as described by the authors. The model was trained for 300 epochs, and the maximum unsupervised weight w_{max} was set to 10, increasing from 0 to w_{max} using the ramp-up function. As the training loop provided labeled and unlabeled samples simultaneously, with the unlabeled samples being more than 10 times larger, the batch size for labeled data was set to 10, and the unlabeled batch size was set to 90. The final iterations, where only unlabeled batches were available, were not discarded, as this method does not require both batches to be present simultaneously.

5.2.2 Results

Table 5.1 presents the test error corresponding to the epoch with the lowest validation error for the specified hyperparameter configurations. The initial experiment employed Supervised Learning on the entire training set, yielding the highest test accuracy of 94.037% with a moderate training speed of 7.632 seconds per epoch. Subsequently, the second experiment, while showcasing a reduced test accuracy of 74.436%, demonstrated a noteworthy improvement in training speed, completing each epoch in 1.314 seconds – a nearly 20% decrease in accuracy compared to the first experiment.

The third experiment, involving the use of the Π-Model, resulted in a test accuracy of 80.192%, representing an improvement of almost 4% over the second experiment. This outcome highlights the effectiveness of the consistency regularization loss in enabling the extraction of meaningful representations. However, it is important to note that the training speed for this experiment was the highest, taking 11.925 seconds per epoch. This higher computational demand was anticipated, as the model evaluates two distinct augmentations for each sample, effectively doubling the number of original samples to 90,000. This challenge made the authors propose an alternative method named Temporal Ensembling [44], in which one of the augmentations is dismissed, addressing the computational overhead while maintaining the efficacy of consistency regularization.

Table 5.1: Test error rates obtained by various preliminary experiments on the standard benchmarks of CIFAR-10.

ID	Technique	Train Samples	Epochs	Epoch Duration $\left(\frac{\text{s}}{\text{epoch}}\right)$	Test Accuracy (%)
1	SL	45,000	200	7.632	94.037
2	SL	4,000	300	1.314	74.436
3	Π-Model	45,000 (4,000 labeled)	300	11.925	80.192

5.3 Work Plan

Figure 5.1 displays a Gantt chart illustrating the anticipated timeline for the development of this dissertation. The execution of the proposed work is delineated into four distinct core tasks:

1. **Semi-supervision Methods:** This task encompasses all the activities related to Semi-supervised Learning. Despite exploring new methods, the previously introduced methods must be implemented and evaluated in the context of Autonomous Driving. A critical review of these methods, highlighting their pros and cons, is also required. This task is anticipated to take slightly over two months.

2. **Self-supervision Methods:** Similar to the preceding task, this one involves the study and implementation of methods using Self-supervised Learning within the realm of autonomous driving, accompanied by a critical review of these methods. Given the absence of currently described or implemented self-supervision methods, this task is expected to take longer than the previous one, with an allocation of two and a half months.

3. **Software Package:** In parallel with the implementation of the aforementioned methods, a dedicated software package will be concurrently developed. This package aims to ease the implementation process across various domains and future research endeavors. All requisite adjustments and preparations for deployment will be meticulously carried out in the concluding stages of the project.

4. **Deliveries:** Systematic documentation of all studied methods and conducted experiments will be actively done using the dissertation document. Subsequent tasks include the composition of a paper article and the meticulous preparation of the final presentation.

This schedule incorporates a small buffer in July, allowing for potential delays or providing room for improvements if the timeline extends beyond the initial expectations.

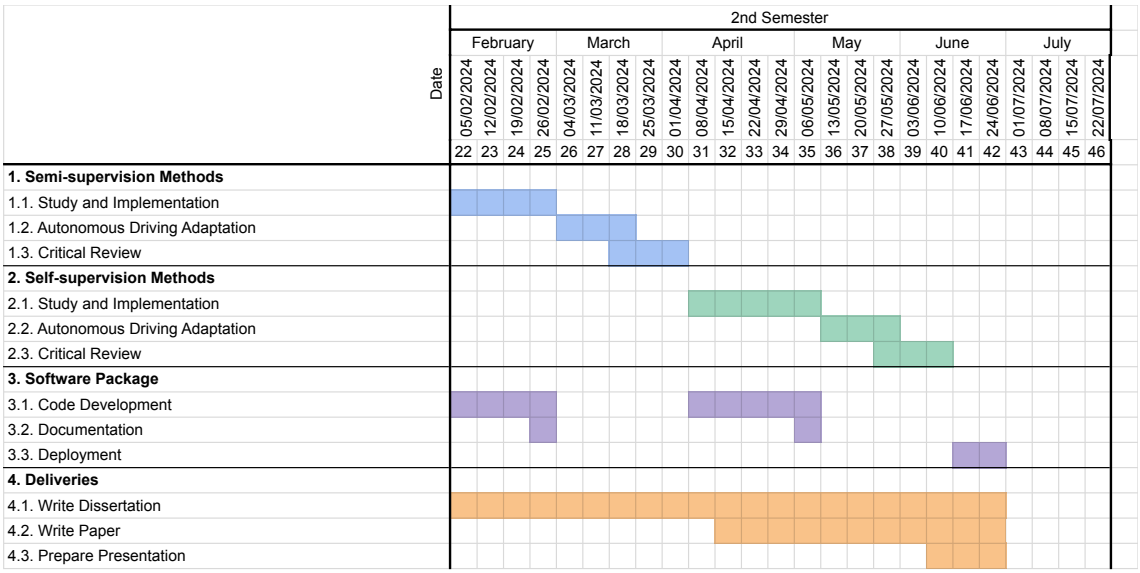

Figure 5.1: A Gantt chart illustrating the anticipated schedule for the dissertation's completion, including timelines allocated for individual tasks.

Chapter 6

Conclusions

In conclusion, this thesis delves into the efficacy of Semi-supervised and Self-supervised Learning in addressing challenges related to label efficiency. The objective is to implement and adapt methods from these paradigms to the domain of Autonomous Driving, conducting a comprehensive review and comparative study while concurrently developing a software package 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. 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 other data, annotated or not.

The proposed solution entails implementing and validating these methods based on configurations outlined in the literature. Subsequently, there is a phase dedicated to adapting them to tasks, datasets, and configurations suitable for the context of Autonomous Driving. Concurrently, the development of the software package is underway. Preliminary work has already assessed scenarios using Supervised Learning, where only a subset of the data is labeled, and compared it to a Semi-supervision approach – specifically the Pi-Model – which demonstrated superior accuracy, with the drawback of having a lower training speed. This suggests that these methods indeed hold the potential to encourage the model to learn relevant features from unlabeled data.

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.

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

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

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

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

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

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

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

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

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

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

[56] 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.

[57] 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.

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

[59] 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.

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

[61] 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.

[62] 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.

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

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

[65] 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.

[66] 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.

[67] 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.

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

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

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