Boosting the Performance of Semi-Supervised Learning with Unsupervised Clustering

By

BOAZ LERNER

Under the supervision of

PROF. DAPHNA WEINSHALL

THE HEBREW
UNIVERSITY
OF JERUSALEM

Faculty of Computer Science and Engineering
THE HEBREW UNIVERSITY OF JERUSALEM

A dissertation submitted to the Hebrew University of Jerusalem as a partial fulfillment of the requirements of the degree of MASTER OF SCIENCE in the Faculty of Computer Science and Engineering.

DECEMBER 2020

Abstract

Recently, Semi-Supervised Learning (SSL) has shown much promise in leveraging unlabeled data while being provided with very few labels. In this study, we show that ignoring the labels altogether for whole epochs intermittently during training can significantly improve performance in the small sample regime. More specifically, we propose to train a network on two tasks jointly. The primary classification task is exposed to both the unlabeled and the scarcely annotated data, whereas the secondary task seeks to cluster the data without any labels. As opposed to hand-crafted pretext tasks frequently used in self-supervision, our clustering phase utilizes the same classification network and head in an attempt to relax the primary task and propagate the information from the labels without overfitting them. On top of that, the self-supervised technique of classifying image rotations is incorporated during the unsupervised learning phase to stabilize training. We demonstrate our method's efficacy in boosting several state-of-the-art SSL algorithms, significantly improving their results and reducing running time in various standard semi-supervised benchmarks, including 92.6% accuracy on CIFAR-10 and 96.9% on SVHN, using only 4 labels per class in each task. We also notably improve the results in the extreme cases of 1,2 and 3 labels per class, and show that features learned by our model are more meaningful for separating the data.

Dedication and acknowledgements

There are several people who made my journey much more meaningful and fun, and to whom I wish to dedicate this work. Firstly, I would like to thank my advisor, Professor Daphna Weinshall who supported me throughout the process and motivated me during hard times. I learned a lot from her vast experience, and greatly enjoyed our weekly meetings.

I would also like to thank my friends from the lab and from other labs as well, for your encouragement, fruitful conversations and most importantly, fun lunch breaks.

And finally, to my lovely fiancée, Yael, who has always been by my side, and believed in me all along the way.

TABLE OF CONTENTS

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

In recent years we have seen huge improvement in the performance of deep learning methods in various computer vision tasks. However, most models require large amounts of annotated data. Collecting this data is a tedious and expensive process. Moreover, learning from so many labels is very different from the way that we as humans learn, and from the way we conceive of intelligence. Therefore, it seems desirable, in our journey towards strong AI, to develop models that rely less on data annotated by humans, and are capable of extracting features in an unsupervised manner.

Semi-Supervised Learning (SSL) is an attempt to tackle this issue and bridge the gap between supervised and unsupervised learning. The typical setting of the problem is that we are given a small amount of labeled data and a possibly large amount of unlabeled data, where both are sampled from the same or similar distributions. Most techniques derive an objective which is split into two separate terms for labeled and unlabeled data [52, 56]. In this way, every gradient step is influenced by the labels. [39] is unusual in that respect, as their method iterates between supervised learning with only labeled data, and unsupervised learning with only unlabeled data. However, their unsupervised stage relies exclusively on fixed pseudo-labels obtained in the supervised stage. Hence, this stage can be considered 'supervised learning with noisy labels'.

In this study, we wish to benefit from real unsupervised learning, undistracted by a small number of possibly uncharacteristic training points, within the SSL framework. The approach is illustrated in Fig. 3.1. We start by describing a clustering algorithm that can be easily integrated with any deep SSL method. This algorithm serves as a secondary

task to the principal classification task. Unlike [39], the pseudo labels (targets) may be propagated from the real labels seen during classification, but they are likely to change during this clustering phase. Also differing from some self-supervised auxiliary tasks used in [2, 56], we use the exact same architecture (network and head) for clustering. Our main goal is to add to the learning protocol a phase which does not depend on the labels. When learning the secondary task, the goal is to separate the data into clusters without assigning names to those clusters. At the same time, self-supervision [17] is used to stabilize and accelerate training during the clustering phase.

Our main contributions in this work are the following:

- Devise a new approach for semi-supervised learning, which is based on solving an unsupervised clustering task together with a classification task.
- Show how realizing this approach by integrating a deep clustering algorithm with three existing SSL algorithms can boost their performance and surpass state-of-the-art results on 3 benchmark datasets.

CHAPTER 2

BACKGROUND & REALTED WORK

In the first part of this chapter, we give a high level background on our main topic of semi-supervised learning. Next, we survey some significant developments in the various fields discussed through our work, and connect our framework to some other resembling frameworks.

2.1 Background

Broadly speaking, machine learning is traditionally divided into two major tasks: supervised learning and unsupervised learning. In supervised learning, we are provided with data points and their corresponding ground-truth output values, and we aim to learn a classifier or regressor that is able to predict the output value for unseen inputs. In unsupervised learning, only the data points are provided, and we are trying to infer a certain structure of the data. E.g., in unsupervised clustering we wish to learn a mapping from the data points to some groups, where each group should contain inputs that are similar to each other in some sense.

Semi-supervised learning attempts to incorporate extra information from one of the above worlds in order to benefit the other. For instance, we may be given extra unlabeled data for a classification problem or some cues regarding the similarity between certain inputs for a clustering problem. However, the extra information might be less structured and more domain specific.

Most SSL research is focused on classification problems. An implied condition neces-

sary for semi-supervised classification methods to benefit from unlabeled data is that $p(x)$ has information on $p(y|x)$. x here denotes a data point and y denotes an optional label. If that holds, we should be able to learn from the unlabeled data about $p(x)$ and thereby strengthen our knowledge about $p(y|x)$ [47]. When this is not the case, the extra unlabeled data is useless for our classification task, and we can discard it. In light of the prominent success of SSL, we can infer that the interaction between these two probabilitiy distributions do exist in most real problems, while the level in which they interact and the exact form of interaction vary across different data and domains.

As a result, some assumptions about this interaction are normally undertaken. The most widely recognized assumptions are the following:

- **Smoothness assumption**: Every two points that are close by in the input space should share the same label. This assumption is also true for standard supervised classification, but can be exploited more significantly in the SSL setting, as it implies transitivity in the labels. Namely, an unlabeled sample which is close to a labeled sample in the input space, can be given the same label and propagate that information further to other unlabeled samples in its vicinity.
- **Low density assumption**: The decision boundary of the classifier should not pass through high density areas. This assumption is somewhat implied by the above smoothness assumption.
- **Manifold assumption**: The data lie approximately on a manifold of much lower dimension than the input space. This assumption is used in many other fields of machine learning. For example, in the vision domain, the input space of natural images is a huge space. However, obviously not every combination of pixels in that space can constitute a natural image. Hence, their intrinsic dimension is much lower. In the case of SSL, it encourages a shift in the attention from primarily being focused on the labels, to being focused on finding the low-dimensional manifold. If we further assume that each class lies on a slightly different manifold, we can try to develop representations for each of these manifolds using just the input data (without labels). In this task the unlabeled data is identically helpful compared with the labeled data. Once achieved, hopefully we can simply label the different manifolds according to the few labeled instances.

Most semi-supervised classification methods assume at least one of the above assumptions. In the next section, we explore the various means of practically implementing those theoretical concepts.

2.2 Related Work

In this work we employ several methods from various fields, including semi-supervised learning, deep clustering and self-supervision. We therefore review the most prominent recent developments in each of those fields. In addition, we present some closely related paradigms and draw the lines between our framework and theirs.

2.2.1 Semi-Supervised Learning

As mentioned in previous section, semi-supervised learning refers to a family of algorithms aimed at learning from both labeled and unlabeled data. Thanks to the relative simplicity of collecting big unlabeled datasets without manual labeling, the field has seen an increasing interest in the last few years. As a result, the performance gap between supervised and unsupervised methods has been consistently diminishing, with the number of labels used to achieve comparable results getting considerably smaller. Given the vast amount of techniques proposed in the literature, we review only the latest and most influential works, and refer the reader to [7] for a comprehensive survey.

Most recent approaches revolve around the two fundamental concepts of entropy minimization and consistency regularization. An intuitive assumption is that the decision boundaries of a classifier should not pass in highly dense areas - the low-density assumption. Minimizing the entropy of a model's prediction on unlabeled data is a common approach to facilitate this heuristic. It can be done explicitly [19] or quite often implicitly by psuedo-labeling [30, 39, 45], a method that assigns an artificial label to an unlabeled image and trains the network to predict that label.

The most basic approach that uses pseudo-labelling is called self-training. The idea is to train in a supervised manner on the labeled data, with the labeled data getting larger in each iteration by appending to it some new unlabeled samples we choose to pseudo-label. Often, this process continues iteratively until all data is labeled. The various methods differ in the way they choose which unlabeled data to label, the stopping criteria and the way in which they re-use the pseudo-labelled data. In [15], a pool of unlabeled samples which is the closest to labeled data is formed. Then, those images which the model is the most uncertain about are selected to be labeled. Their intuition for the first step is clearly linked to the smoothness assumption detailed in Section 2.1, while the intuition for the second step is that the less certain samples are more informative for the model to get better. This is contrary to more common heuristics which simply prefer the most confident samples. For instance, in [21], they propose to weight a sample according

to the current confidence of the model, but also according to the effect of selecting it on the accuracy of the model on the originally labeled data. If a pseudo-labeled sample will cause a major deviation in the performance on the labeled data, it might imply that the pseudo-labelling was wrong. To tackle the same problem of wrong pseudo labels, [31] proposed to increase the weight of pseudo samples as training progresses. The reason is that as the model gets better during training, its pseudo-labels become more reliable.

The second common concept of consistency regularization refers to the assumption that small perturbations to the data should not affect its semantics, and hence the label. It is reasonable then to force the model to output consistent predictions to all perturbed versions of the same sample. Consequently, a lot of recent research makes use of complex augmentation strategies [2, 10, 11, 52]. FixMatch [45], which is the most relevant to our work, combines both approaches and will be thoroughly discussed in Section 3.1.2.

The two concepts discussed above are closely linked to the first two assumptions we discussed in section 2.1. The third assumption regarding the low-dimensional manifold in which the data resides is often assumed in an approach that takes advantage of the huge development in generative models. The most commonly used generative models in general, and within the framework of semi-supervised classification in particular are: generative adversarial networks (GANs) [18] and variational auto-encoders (VAEs) [27].

GANs consist of two neural networks. The first one is called the generator, whose goal is to generate images resembling the data distribution, and hence to fool the second network, the discriminator, that the image came from the real distribution. The discriminator's goal then is to distinguish between images coming from the real data distribution and "fake" images generated by the generator. Its output is a scalar reflecting the probability of the image coming from the real distribution. The simplest extension of GANs for SSL [41] extends the discriminator's output space to have $K+1$ outputs, where K is the number of classes. The first K outputs represent the probabilities for each class, and the last output corresponds to the probability of the image being generated by the generator. Then, [41] proposed to combine the cross-entropy loss with the regular adversarial loss (only the last output is used) for labeled data, and use the adversarial loss alone for unlabeled data. After training, the discriminator is used as the classifier by ignoring the last output and predict according to the highest probability among the first K outputs. In *triple adversarial networks* [33], the discriminator/classifier is separated into two distinct networks. The discriminator then is back to its usual form, only distinguishing between real and fake images, with the slight change that it works on the joint distribution $p(x, y)$ (x-image, y-label) instead of the marginal $p(x)$. Namely,

the input to the discriminator is an image-label pair. In every iteration, three batches are sampled: one from the real labeled data, another one from the generator and the last one from the classifier which outputs a pseudo-label to unlabeled images. The intuition is that the joint distribution can be factorized in two ways. One way, $p(x,y) = p(x)p(y|x)$, is approximated by the classifier and the other, $p(x,y) = p(y)p(x|y)$, is approximated by the generator.

VAEs are type of auto-encoders [40] with some added constraint on the prior distribution over the latent space. The common prior used is the normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$. Given an image x, the encoder models the posterior distribution $p(z|x)$ (z is a latent code) and attempts to approximate the prior by outputting a mean and covariance of the Gaussian distribution. Then, a latent code is sampled via a reparameterization trick to keep all operations differentiable, and a new image is generated by the decoder. The loss is a combination of the regular reconstruction loss of auto-encoders and a D_{KL} loss between the outputted encoder's Gaussian distribution and the prior. One of the most prominent methods applying VAEs within the framework of SSL is introduced by Kingma et. al [26]. They propose a two-steps model. The first step is a preprocessing representation learning step, where a regular VAE is trained in a standard manner on the entire data (labeled and unlabeled). Then, in the second step they train a VAE whose latent representation is augmented with a label vector. For the labeled data, this vector is the one-hot version of the true label and for the unlabeled data it is treated as an additional latent variable. In addition to the decoder, another classification network is introduced to infer the labels' predictions.

2.2.2 Deep Clustering

The task of unsupervised clustering is a long-standing problem and a highly challenging one, especially when it meets the high-dimensionality of images. Classical algorithms, such as k-means [25] and Gaussian Mixture Models (GMM) [4], struggle in this domain as the raw data is not very informative, and thus the need for succinct and meaningful representation of images is critical. In recent years, deep clustering frameworks have become increasingly competent in solving this task. One type of such frameworks divides training into two steps. The first step focuses on learning good representations and the second step finetunes the network's parameters with some clustering objective [23, 53]. The probably most known work performing that is Deep Embedded Clustering (DEC) [51]. Firstly, an encoder network is initialized by pre-training on a reconstruction task alongside a decoder (i.e an auto-encoder) which is afterwards discarded. Then, cluster

centroids in the embedding space are iteratively computed and refined until convergence. More recently, [48] utilizes representations learned from a pretext self-supervised task in order to extract the nearest neighbors of each image. Then, they train a different network to output similar features for an image and those nearest neighbors.

Other methods use representations learned in a supervised way. Guérin et al. [20] conduct extensive experiments with various CNN architectures pre-trained on ImageNet [12]. They use those pre-trained networks as features extractors and apply several classical clustering algorithms on these features. In [22], they use the pre-trained network for initializing the clusters' centroids. Then, they finetune this network's parameters together with a k-means objective.

However, more typical approach is to jointly learn image features alongside cluster assignments by training a deep network with some clustering loss in an end-to-end fashion. The coupling of learning image features and clusters together allows the deep network to better adapt its image features to the task of clustering. Joint Unsupervised Learning (JULE) [54] trains a model in an end-to-end fashion by iteratively merging clusters of deep representations and updating the network's parameters in a hierarchical fashion. Recently, Invariant Information Clustering (IIC) [24] proposed a novel information-theory approach for clustering, in which they maximize the mutual information between deep embeddings obtained by two different random augmentations applied to the same image, and rely on the natural characteristics of the mutual information loss to produce a clustering of the data.

2.2.3 Self-Supervised Learning

Self-supervised learning is an approach to learning in an unsupervised manner by solving a pretext supervised task. The supervisory signals are gathered automatically from the data without the need for manual labeling. The task is designed in a way that implicitly requires learning of useful image representation, e.g. predicting the relative position of patches in an image [14], or solving jigsaw puzzles [36]. A more recent method [17] predicts image rotations (RotNet) and has also been used as an auxiliary task to stabilize and improve training in semi-supervised [56] and image generation tasks [8]. Similalry, our method also employs RotNet for this purpose.

2.2.4 Few-Shot Learning

In our context of very small sample of labeled data, it's compelling to compare to the newly rising paradigm of few-shot learning (FSL) [50]. Typically FSL is formulated as an N-way-K-shot classification problem, where N is the number of classes and K is the number of samples from each class, which tends to be very small. While humans can normally solve this problem pretty easily, this is a very challenging task for machines. Hence, usually a prior knowledge is leveraged. This prior can be in the form of extra N-way-K-shot tasks (with different classes), as in the framework of meta learning where the goal is to learn how to learn [49], or a pre-trained model as in the framework of transfer learning [37]. The key distinction between this study and FSL is the type of prior. FSL typically leverages extra supervisory information from data sampled from different distribution than the data's in hand whereas SSL focuses in leveraging extra unlabeled data assumed to come from the same distribution as the labeled one.

2.2.5 Active Learning

Similarly to our main goal of learning with as few labeled data as possible, active learning is a paradigm which attempts to minimize the number of labels provided to the learner. In this framework, there is an interaction between the learner and the real world. The learner starts with unlabeled data only (or very few labeled data), and during training it can request to label some specific instances of the data. Essentially, labelling a data point is assumed to have a certain cost (usually the same cost for all data points) and the goal is to prioritize the data for labelling in a way that minimizes the amount of resources utilized. Different methods mainly differ in the scoring function they use for determining which unlabeled instance to query in each step. The most basic idea is to query the instances about which the model is least certain how to label. The uncertainty can take into account the probability the model gives to the most likely class [32], both the most likely and second most likely classes' probabilities [42], or the entire vector of probabilities, e.g. using the entropy. While both active learning framework and our framework make use of both labeled data and unlabeled data, in this study we focus on standard SSL, which is static with respect to the data. We fix the labeled data in the beginning and don't allow interaction for acquiring labels during training. We can think of our method as an attempt to perform well no matter which labels are chosen, and of active learning as a method to choose the labels wisely in order to improve performance.

CHAPTER 3

OUR METHOD

In this chapter, we present our novel approach for semi-supervised learning when the data is scarcely annotated. We discuss the different components of our proposed method for realizing this approach and give intuition to why it should work.

3.1 SSL with Very Few Labels

In this section we present an SSL algorithm that alternates between two phases: unsupervised clustering and semi-supervised classification. The method is designed especially to perform well in the very-few-labels scenario.

As the SSL building block we use in our experiments a variety of recent state-of-the-art algorithms, including FixMatch [45] which is used in most of the experiments, as well as MixMatch [3] and UDA [52]. Our main goal is to equip them with an additional unsupervised mechanism, in order to reduce their susceptibility to outliers in the small labeled sample. To this end we describe an effective unsupervised deep clustering method, a variant of MMDC [44]. This building block can also be replaced in order to improve performance.

We start by describing each component separately, and then describe their integration into a single coherent model.

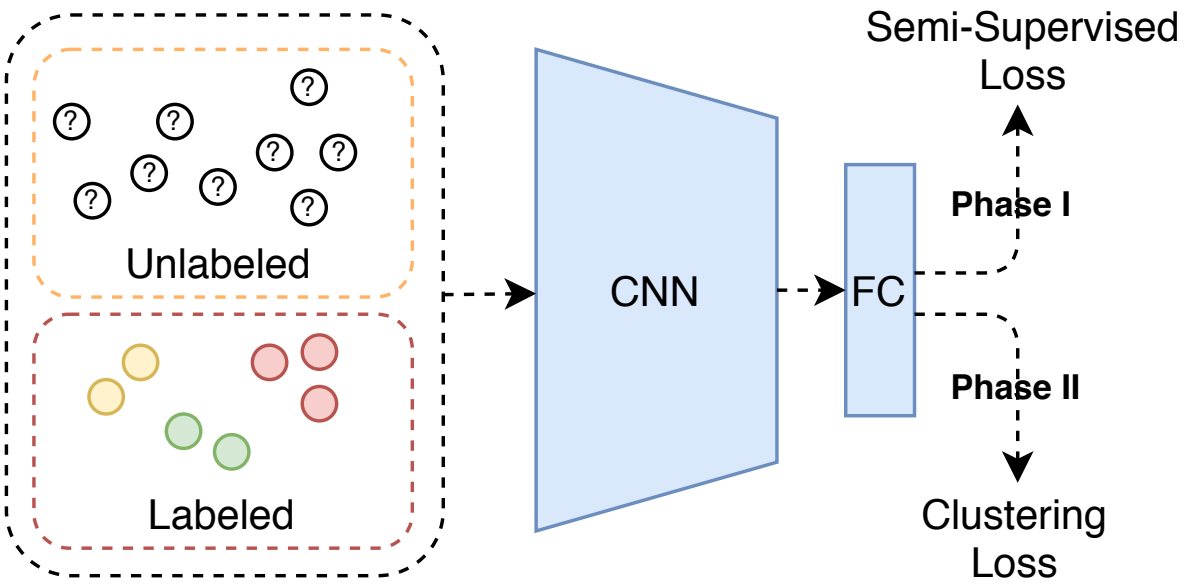

Figure 3.1: Our method iterates between two phases. In the first phase we train our model with the available labeled and unlabeled data points using a semi-supervised learning algorithm. In the second phase we train the same model to cluster all the data points, without using any labels.

3.1.1 Unsupervised Clustering - MMDC

In a typical supervised classification setting we are given pairs of images and labels $\{x_i, y_i\}_{i=1}^n$, and train a parameterized model f_θ by solving:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i), \tag{3.1}$$

where ℓ is some loss function. In the setting of unsupervised clustering, where ground truth labels $y_1, ..., y_n$ are not available, we can attempt to learn them alongside the model's parameters:

$$\min_{\theta, y_1, ..., y_n} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i). \tag{3.2}$$

Without additional constraints, this optimization procedure is prone to suffer from mode collapse, where all images are assigned the same label $y_1 = \cdots = y_n$. To overcome this susceptibility, we add a constraint to the optimization formulation that explicitly prevents this from happening:

$$\forall k \in [K] \quad \sum_{i=1}^{n} 1_{y_i = k} \geq \alpha \frac{n}{K}, \quad 0 < \alpha \leq 1. \tag{3.3}$$

Here K denotes the number of classes in the dataset and α is a hyper-parameter. Eq. 3.3 guarantees that each cluster has a minimal number of images assigned to it.

In order to solve this optimization problem, we adapt a similar approach to the one taken in [5], where the task of representation learning is addressed. In [5], the problem of feature collapse is tackled by fixing static feature vectors $\{y_i\}_{i=1}^n$ at the beginning of training. Throughout the training, the algorithm learns the model's parameters and a one-to-one mapping $P : [n] \to [n]$ from images $\{x_i\}_{i=1}^n$ to those fixed targets.

In our case, we are interested in clustering the data, and hence it is reasonable to set the targets to be one-hot vectors in $\mathbb{R}^k$, which we denote by $e_1, \ldots, e_k$. To enforce (3.3), the targets are set to $T = \{y_i | \forall k \in [K], \sum_{i=1}^n \mathbb{1}_{y_i = e_k} = \alpha \frac{n}{K}\}$, with $\alpha \frac{n}{K}$ target instances per cluster. Before training begins, each target is randomly assigned to an image in the dataset. Note that some images may not be paired up with a target as there might be more images than targets.

The optimization problem is solved stochastically one mini-batch at a time, where each iteration consists of two steps. Given a mini-batch $X_b = \{x'_1, \ldots, x'_b\} \subseteq X$ of b images and their current targets $T_c = \{t'_1, \ldots, t'_c\} \subseteq T$ $(b \geq c)$, the first step finds the best assignment of targets to images, denoted by $P^* : [c] \to [b]$, while the network's parameters are kept fixed. This is accomplished by minimizing the following objective with the Hungarian method [29]:

$$P^* = \arg\min_P \sum_{i=1}^{c} ||f_\theta(x'_{P(i)}) - t'_i||_2^2. \tag{3.4}$$

Recall that not all images are necessarily assigned a target. Among the unassigned images, only those with confidence exceeding a certain threshold are assigned to pseudo-targets. A similar approach is adopted in [6, 45]. For an unassigned image x'_k to be considered confident, we require that $||f_\theta(x'_k) - e_{\arg\max f_\theta(x'_k)}||_2^2 < \rho$, where ρ is a hyper-parameter. In this case, the pseudo-target assigned to x'_k is $\tilde{y}_k = e_{\arg\max f_\theta(x'_k)}$.

In the second step of the optimization scheme, we update the model's parameters with a gradient step, which minimizes the distance of the model's outputs from the targets or pseudo-targets found in the first step. Specifically, if we denote the unassigned confident images by C, and the batch images being processed by $S = Im(P^*) \cup C$, then in the second step each image in S is augmented r times with a stochastic function g, where all r versions of the same image are matched to the same target. This is formulated as minimizing the following objective w.r.t. θ:

$$\mathcal{L}_c = \frac{1}{|S|r} \sum_{i \in S} \sum_{j=1}^{r} ||f_\theta(g(x'_i)) - \tilde{y}_i)||_2^2, \tag{3.5}$$

where $\tilde{y}_i$ is either the target or the pseudo-target of image x'_i. The objective is minimized via stochastic gradient descent. Note that unassigned images with low confidence are ignored and do not influence the optimization. As in [5], we implement f_θ as a ConvNet and normalize its output such that $||f_\theta(x)||_2 = 1$.

To further enhance the representation capabilities of the model, which may in turn facilitate better clustering, we train the same model on an additional auxiliary task. Specifically, we employ the self-supervised task of predicting image rotations (RotNet) [17], as it has a proven record of efficiently improving ConvNets representations in a variety of tasks [8, 16, 56]. We do this by feeding the penultimate layer of f_θ into another head, which is used to generate the rotation predictions for the RotNet task.

The full clustering algorithm is detailed below in Alg. 1.

3.1.2 Semi-Supervised Classification

Semi-supervised classification is carried out in most of our experiments by adopting the FixMatch method, as it currently yields state-of-the-art results when relying on a small labeled sample. FixMatch combines the two commonly used heuristics discussed in Section 2.2.1, *consistency regularization* and *pseudo-labelling*. They are expressed as part of the loss function applied to unlabeled data during the training of a neural network, while labeled data is used to optimize the standard cross-entropy loss.

Formally, given a batch of b images $X = \{x_1, \ldots, x_b\}$ and their labels $Y = \{y_1, \ldots, y_b\}$, and another batch of b' unlabeled images $U = \{u_1, \ldots, u_{b'}\}$, FixMatch predicts the class distribution of the network's output on a weakly-augmented expansion of the unlabeled batch, and uses these predictions as hard pseudo-labels for a strongly-augmented expansion of the same images. Thus, if we denote the network by f_θ and the stochastic weak and strong augmentation functions by g and q respectively, the pseudo-label of image u_i becomes $y'_i = \max(f_\theta(g(x_i))$, and the loss term on the unlabeled batch can be written as:

$$\mathcal{L}_u = \frac{1}{|\{i \in [b'] \mid y'_i \geq \tau\}|} \sum_{i=1}^{b'} \mathbb{1}(y'_i \geq \tau) H(f_\theta(q(u_i)), y'_i).$$

Above, H denotes the cross-entropy loss and τ denotes a hyper-parameter that determines the threshold confidence above which the image will be considered in the update of the network's parameters (similar to ρ defined above). The loss on labeled data is simply:

$$\mathcal{L}_s = \frac{1}{b} \sum_{i=1}^{b} H(f_\theta(g(x_i)), y_i), \tag{3.6}$$

Algorithm 1 Unsupervised Clustering

INPUT: $X = \{x_i\}_{i=1}^{n}$ - unlabeled dataset
f_θ - convnet with two heads and parameters θ
K - number of clusters
g - stochastic augmentation function
λ_i - learning rate at epoch i
r - number of times g is applied to an image
α - ratio of dataset that will have targets
ρ - maximal distance to be considered confident

$T \leftarrow [\,]$
for k=1 to K **do** ▷ initialize targets
  for i=1 to $\alpha\frac{n}{K}$ **do**
    append e_k to T ▷ e_k is the kth unit vector in $\mathbb{R}^K$
  end for
end for
$\forall i \in [\alpha n] \quad A(x_i) = T[i]$ ▷ initialize assignments
for *i=1...epochs* **do**
  for *j=1...iters* **do**
    sample a batch $(\{x'_i\}_{i=1}^{b}, \{t'_i\}_{i=1}^{c})$ ▷ $b \geq c$
    compute P^* with Eq. 3.4
    $\forall i \in [c] \quad A(x_{P^*(i)}) = t'_i$ ▷ update assignments
    update the parameters with gradient step of Eq. 3.5:

$$\theta \leftarrow \theta - \lambda_i \nabla_\theta \mathcal{L}_c$$

  end for
  for *j=1...iters* **do**
    sample a batch X_b
    $\forall d \in \{0°, 90°, 180°, 270°\}$, rotate X_b d degrees
    update parameters with a gradient step of the
    cross-entropy loss $\mathcal{L}_r$:

$$\theta \leftarrow \theta - \lambda_i \nabla_\theta \mathcal{L}_r$$

  end for
end for

and the total loss is a combination of them both: $\mathcal{L}_s + \lambda_u \mathcal{L}_u$, where λ_u denotes a hyper-parameter balancing the weights of the two terms.

The weak augmentations used in the algorithm include the standard flip and shift transformations. First, the images are flipped horizontally with 50% probability, and then they are randomly translated by up to 12.5% vertically and horizontally. As strong

augmentations, two variants of AutoAugment [10] are used, RandAugment [11] and CTAugment [2]. Both are followed by Cutout [13].

3.1.3 Integrated Method

The basic idea underlying our method is that if the number of labels is small, it benefits a classification algorithm to occasionally refrain from taking the labels into account. To achieve this goal, our method alternates traditional semi-supervised training epochs with full epochs that optimize the unsupervised loss in (3.5) while ignoring the labels. This design aims to learn meaningful features, which can compensate for the shortage of labels and help the model generalize better. As a result, the model is less susceptible to overfitting the few labeled datapoints, especially when they do not agree well with the total data distribution.

Specifically, we use the same network architecture and weights to solve the two different tasks described above jointly. To this end the algorithm alternates between FixMatch epochs and clustering epochs. We also perform several RotNet warmup epochs, as this has proven useful for accelerating the learning.

Given the information propagated from the labels during the semi-supervised phase, clustering can be seen as a surprisingly easier task that attempts to separate the data without giving names to the different clusters thus created. Along the way, the mini-batch permutation optimization gives the network a chance to swiftly switch the targets of images whose confidence level is too low. Then, in the next supervised phase, the algorithm tries again to give those clusters names and refine the boundaries between them. This cycle is repeated until convergence.

Alg. 2 summarizes the method. We make the code available in github[1]. In the next section, we show its effectiveness in semi-supervised learning with a small labeled sample. In these experiments, the semi-supervised step is realized with more out-of-the-box SSL algorithms for comparison.

[1] `https://github.cs.huji.ac.il/boazler/SSClustering`

Algorithm 2 Boosted SSL

INPUT: $U = \{u_i\}_{i=1}^{n}$ - unlabeled dataset
$(X, Y) = \{x_i, y_i\}_{i=1}^{m}$ - labeled dataset
SSL_ALGO - some deep SSL algorithm
C_ALGO - our clustering algorithm from 1
f_θ - convnet with two heads and parameters θ
g - stochastic augmentation function
$\lambda_{i,j}$ - learning rate at iteration i, epoch j
r - number of times g is applied to an image
α - ratio of dataset that will have targets
ρ - maximal distance to be considered confident

for *i=1...iters* **do**
  for *j=1...*e_1 **do**
    run $SSL\_ALGO(X, Y, U, f_\theta)$ for one epoch
  end for
  for *j=1...*e_2 **do**
    run $C\_ALGO(U, c, f_\theta, g, \lambda_{i,j}, r, \alpha, \rho)$ for one epoch ▷ c is the number of classes
  end for
end for

CHAPTER 4

EXPERIMENTAL EVALUATION

In this chapter we empirically evaluate our method on various datasets. We show how it can boost the performance of several SSL algorithms and decrease their running time significantly. Also, we show its superiority when employing FixMatch as the SSL module, and discuss directions for improving the results further.

4.1 Datasets

We evaluate our method on three common SSL benchmarks, see Table 4.1. We should mention that STL-10 contains 100K extra unlabeled images which come from a slightly different distribution than the labeled data, including some extra classes not represented in the 5000 labeled images. Hence, this dataset is more challenging and requires more labeled data to perform well. SVHN contains roughly 530K extra labeled images as well which we don't use in our experiments. It is also slightly imbalanced, having a ratio of approximately $1:3$ between the number of instances in the least represented class to the number in the most represented one.

4.2 Implementation Details

In all of the experiments we used the WideResNet (WRN) architecture [55], replicating the setup described in [45]. More specifically, for the CIFAR-10 and SVHN datasets we

Name	Classes	Train/Test Size	Dimension
CIFAR-10 [28]	10	50000 / 10000	$32 \times 32 \times 3$
SVHN [35]	10	73257 / 26032	$32 \times 32 \times 3$
STL-10 [9]	10	5000 / 8000	$96 \times 96 \times 3$

Table 4.1: Datasets used in our experiments. For STL-10, the 100K extra unlabeled images were used as well.

used WRN-28-2, and for STL-10 we used WRN-37-2. In the SSL phase, we kept the exact same hyper-parameters as in the original SSL algorithm being employed, while for the clustering phase, the learning rate and weight decay were reduced to 0.01 and 0.0001 respectively (from 0.03 and 0.0005 in FixMatch). The clustering hyper-parameter ρ was set to 0.2, and the α hyper-parameter was set to 1 for Cifar-10, and 0.6 for SVHN and STL-10. As in most other contemporary SSL methods, we stored and evaluated the model with exponential moving average of the weights over the training and a decay of 0.999.

Image augmentation: during the SSL phase, we took care to always apply the exact same augmentations as used in the original SSL method used to realize the SSL phase. Specifically in the experiments with FixMatch, we used Control Theory Augment [2] that achieved the best results in most scenarios. In the unsupervised phase, we used the customary flip followed by crop, after the application of random color jitter to each pixel. We used the same flip and crop transformation in both phases: horizontal flip with probability 0.5, followed by cropping the mirror padded image to the original size.

Unless otherwise mentioned, we trained our model for 200 iterations, each comprising multiple passes over the data in the SSL phase (10 for Cifar-10 and 5 for all other datasets), followed by one pass in the clustering phase.

In all the experiments whose results are reported below, in order to ensure a fair comparison, we used the exact same partitions into labeled and unlabeled data as used in [45]. Therefore, whenever we present results while replicating experiments reported in [45], we use the results reported there for all the methods but our own. When using different existing algorithms, we first made sure that our implementation of those methods yielded comparable results to those reported in the original manuscripts, in order to avoid running all the various experiments anew.

4.3 Results

	CIFAR-10		STL-10	SVHN	
Method	40 labels	250 labels	1000 labels	40 labels	250 labels
Π-Model [38]	-	54.26 ± 3.97	26.23 ± 0.82	-	18.96 ± 1.92
Pseudo-Labeling	-	49.78 ± 0.43	27.99 ± 0.80	-	20.21 ± 1.09
Mean Teacher [46]	-	32.32 ± 2.30	21.43 ± 2.39	-	3.57 ± 0.11
MixMatch	47.54 ± 11.50	11.05 ± 0.86	10.41 ± 0.61	42.55 ± 14.53	3.98 ± 0.23
UDA	29.05 ± 5.93	8.82 ± 1.08	7.66 ± 0.56	52.63 ± 20.51	5.69 ± 2.76
ReMixMatch	19.10 ± 9.64	5.44 ± 0.05	5.23 ± 0.45	3.34 ± 0.20	2.92 ± 0.48
FixMatch (RA)	13.81 ± 3.37	$\mathbf{5.07} \pm 0.65$	7.98 ± 1.50	3.96 ± 2.17	2.48 ± 0.38
FixMatch (CTA)	11.39 ± 3.35	$\mathbf{5.07} \pm 0.33$	5.17 ± 0.63	7.65 ± 7.65	2.64 ± 0.64
Ours	$\mathbf{7.39} \pm 0.61$	5.51 ± 0.25	$\mathbf{4.78} \pm 0.29$	$\mathbf{3.09} \pm 0.54$	$\mathbf{2.30} \pm 0.03$

Table 4.2: Error rates for the 3 datasets used in our study: CIFAR-10, STL-10 and SVHN. Results are reported for varying amounts of labels, denoting the total number of labeled points from all classes.

Classification Results with FixMatch

In Table 4.2, we report the results of our method when applied to the three datasets used in our study, with various amounts of labels. We ran the algorithm with 5 different partitions, the exact same partitions used in [45]. Due to the large variance in the 40 labels setting, we repeated the experiment 3 times for each partition. Hence, the standard deviation reported in our results is the standard deviation (STD) of the means over the different partitions. As expected for such small partitions, it is rather large.

As can be seen in Table 4.2, our method is very effective in the very small sample regime with 4 labels per class, where its relative advantage over the alternative methods is quite high. Its added value is less pronounced when using a total of 250 labels. Still, when learning to classify the more challenging STL-10 dataset with a setting identical to the one described in [45]—we used the same 5 folds of 1000 labeled images each, and the additional 100K unlabeled images—our method once again outperforms all the results reported in [45].

Finally, in order to push our method to its limit, we experimented with even fewer labels: 10, 20, 30, and also 100 - an intermediate number between 40 and 250. This very small sample regime is not systematically investigated in [45]. The results for these experiments are presented in Fig. 4.1, as well as a detailed per-partition results for the

20-labels CIFAR-10 experiment in Fig. 4.2. Clearly, as long as the number of labeled points is smaller than 250, our method is quite beneficial.

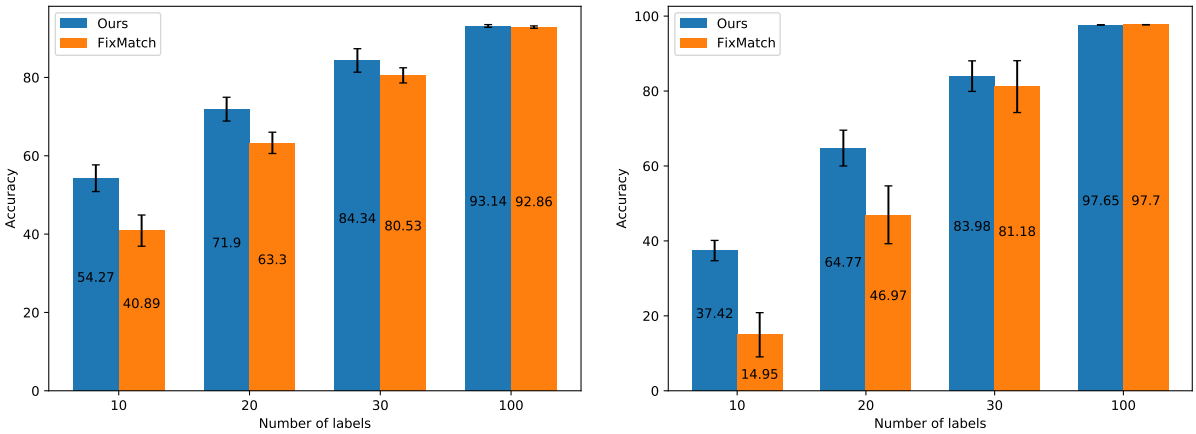

Figure 4.1: Classification accuracy, comparing our method to FixMatch with CTA, using identical protocols. Left: CIFAR-10, right: SVHN. The numbers inside the columns denote the mean accuracy across different partitions and runs.

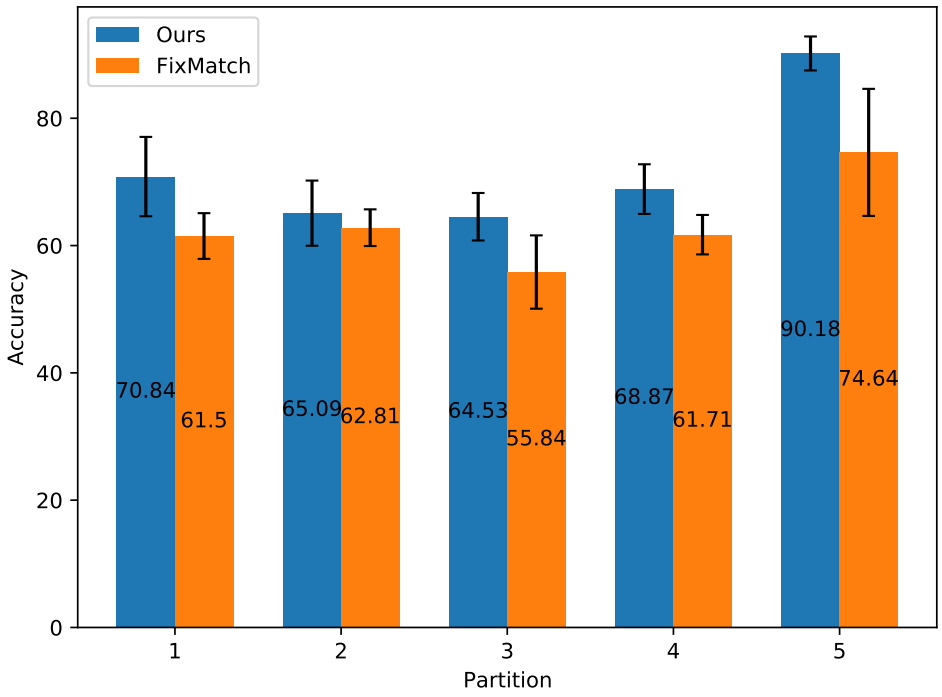

Figure 4.2: The classification accuracy for CIFAR-10 with 20 labels (2 per class), presented separately for each partition.

Clustering Accuracy Score

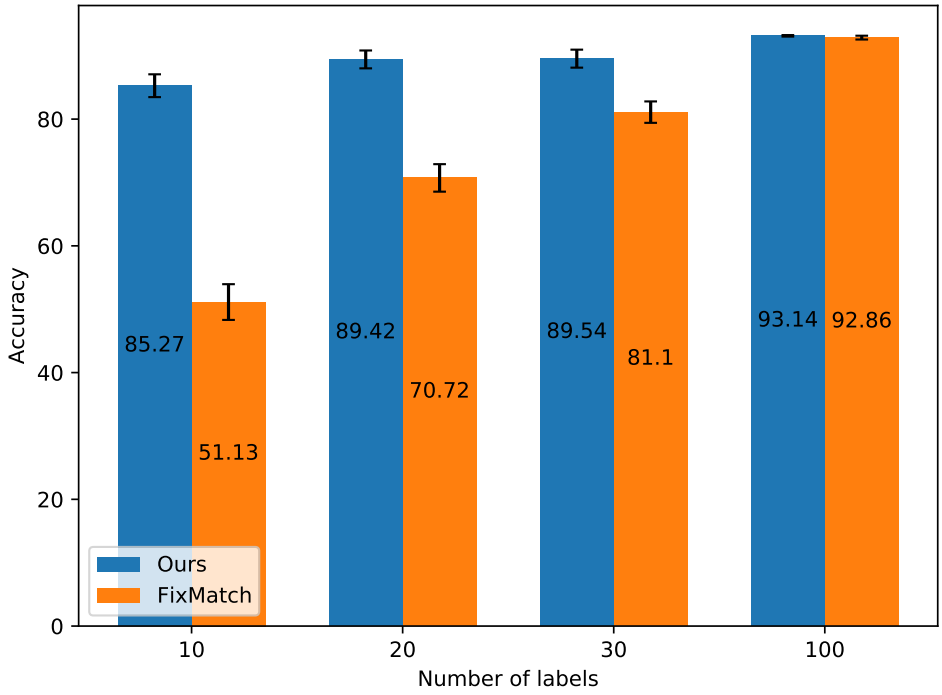

Figure 4.3: Clustering accuracy for the same experiment, the results of which are reported in Fig. 4.1.

An interesting advantage of our method is demonstrated in Fig. 4.3. The accuracy shown there is the clustering accuracy score, which is traditionally computed as the classification accuracy of the best permutation of class labels, when uniquely matched with the different clusters. More precisely, if our test images and their corresponding labels are given by $(\{x_i\}_{i=1}^m, \{y_i\}_{i=1}^m)$, and the model's predictions are given by $\{\tilde{y}_i\}_{i=1}^m$, then the clustering accuracy score is defined as:

$$\max_{P\,:\,[c]\to[c]} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}(y_i = P(\tilde{y}_i)), \tag{4.1}$$

where P denotes a permutation over the c possible classes.

Note that while the classification accuracy for the experiments with 10 and 20 labels may seem low, the data is still clustered very well by our model. Even with one label per class, the model reaches a mean clustering accuracy of over 85%, and the best partition achieves mean accuracy of over 90%. At the same time, we see in Fig. 4.1 that the mean classification accuracy is only 54.3%. The gap in accuracy may be large, but it

resides solely in the naming of the clusters. Conversely, this cannot be said about the predictions obtained by FixMatch alone. There, the gap between classification accuracy and clustering accuracy is considerably smaller, which means that FixMatch doesn't succeed in separating the classes in the extreme small sample regime. With 100 labeled examples, the classification and clustering accuracy converge to the same value for both methods.

In Fig. 4.4 we show two partitions from CIFAR-10, each with 10 labels. In one partition there is a big gap between classification accuracy and clustering accuracy, because the model confused the labels of 3 clusters as shown by the arrows. In the other partition, the model succeeded in finding the right permutation, and hence achieved 91% accuracy in both classification and clustering.

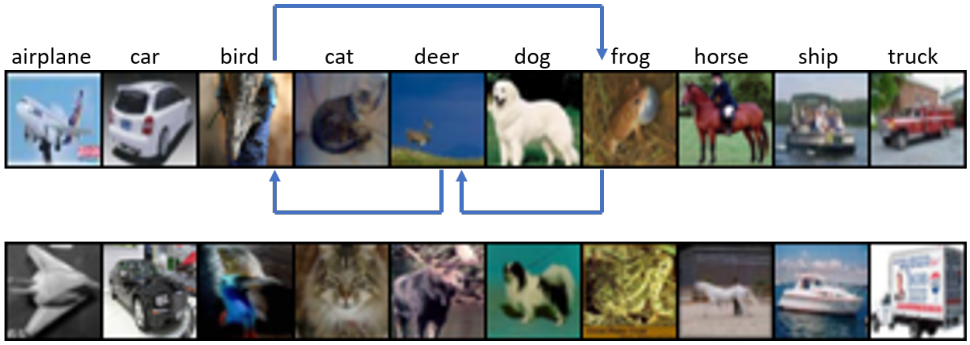

Figure 4.4: Two different partitions of CIFAR-10 with one label per class are shown above. Given the top partition, our model found the wrong permutation as illustrated with blue arrows. This error led to the gap between classification accuracy of 67.6% and clustering accuracy of 92.5%. Given the bottom partition, our model found the optimal permutation and achieved 91.0% accuracy in both classification and clustering.

Bridging this gap between classification accuracy and clustering accuracy with so few labels is a hard problem. We investigated a few heuristics in order to identify "good" permutations during or after training, but more effort is needed. In Fig. 4.5 we show the results when employing one such heuristic. After training is completed, we rotate the labeled images (in four orientations as in RotNet) and use the average prediction in order to find the k best permutations with Murty's algorithm [34]. We use rotated images because the trained model is already (over-)fitted to the small labeled sample. Note that the permutation which achieves the best performance, and which defines the clustering accuracy, always lies within the 100 best permutations (out of 10! possible permutations) according to this score in the experiments with 20 and 30 labeled examples.

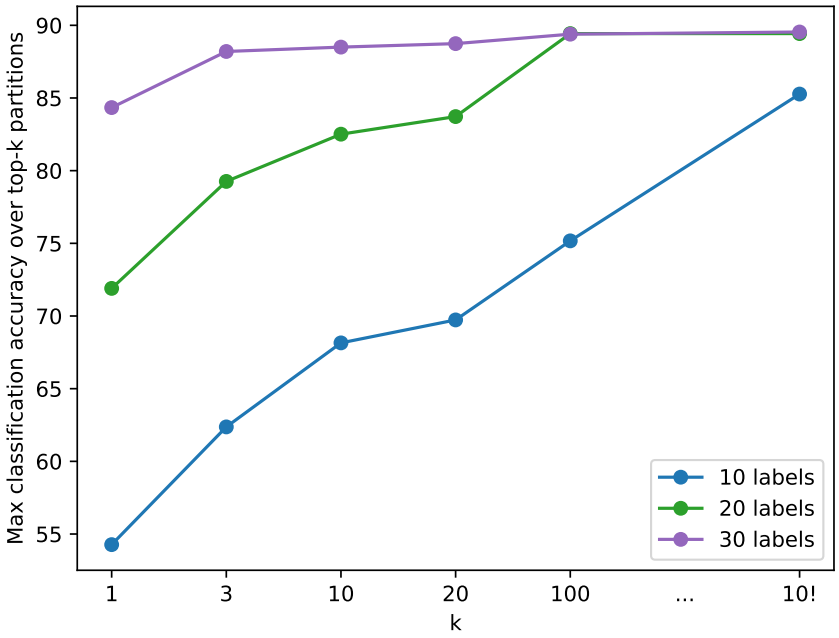

Figure 4.5: Top-k classification accuracy for CIFAR-10 with 10, 20 and 30 labels. k denotes the number of permutations considered.

Running Time Comparison

Another advantage of our method is its efficiency with respect to running time. As mentioned in Section 4.2, the reported results are obtained after 200 iterations of our method. This implies 2000 passes through the whole data, plus another 200 passes for clustering and 200 passes for RotNet. In FixMatch, with randomly sampled batches, the total number of semi-supervised batches processed to achieve the published results approaches $\sim 1M$ batches, while our method processes $\sim 220K$ batches to achieve the results shown above. Even though each clustering epoch takes a bit longer than a FixMatch epoch, due to the expensive assignment problem, the total running time of our method adds up to roughly 30% of the running time needed by FixMatch when left on its own. Similar observations hold for the other SSL algorithms, which are compared with our method next.

Other SSL algorithms

As explained in Chapter 3, our approach is general in the sense that it can use any clustering algorithm and any SSL method to address the challenging SSL problem of

classification with small labeled sample. In this section we show that interlacing our proposed clustering method with two other successful SSL methods improves their outcome, in a similar way to the previous results with FixMatch. Thus, in Table 4.3 we show how our approach boosts the performance of MixMatch [3] and UDA [52]. These experiments were conducted on CIFAR-10 with 40 labels, using the same 5 partitions as in all the other experiments. Each partition was evaluated once. As can be seen, UDA combined with our clustering mechanism almost closes its initial gap against FixMatch, when both methods use RandAugment as the generator of strong augmentations (this was the augmentation used by the original method).

Method	Error Rate
MixMatch	47.54 ± 11.50
MixMatch + Clustering	$\mathbf{35.37} \pm 6.01$
UDA	29.05 ± 5.93
UDA + Clustering	$\mathbf{14.37} \pm 3.85$

Table 4.3: Error rates of MixMatch (top) and UDA (bottom), with and without clustering, trained on CIFAR-10 with 40 labeled examples.

4.3.1 Ablation Study

We tested the contribution of two major components of the proposed method: clustering and RotNet, using CIFAR-10 with 40 labeled examples. Table 4.4 summarizes the results. As before, each of the 5 partitions used in the initial experiments is learned 3 times independently. We can see that RotNet alone does not improve the results, nor does it degrade them. However, without RotNet the clustering phase is far less stable, with a detrimental effect on the final classification outcome. In both cases, we observed a much higher variance in performance.

Method	Error Rate
FixMatch	11.39 ± 3.35
FixMatch + RotNet	11.55 ± 2.98
FixMatch + Clustering	12.15 ± 3.08
Our Method	$\mathbf{7.39} \pm 0.61$

Table 4.4: Ablation study on CIFAR-10 with 40 labeled examples.

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 Summary and Discussion

Motivated by the desire to reduce the reliance on annotated data as much as possible, we propose a new approach to semi-supervised learning, which is designed to reduce overfit when very few labels are available. The proposed method alternates between an unsupervised clustering phase that ignores the labels in the training data, and a semi-supervised classification phase that makes full use of the training labels. To this end, we utilize a variant of a new deep clustering algorithm [44]. We then demonstrate the effectiveness of the general approach by plugging into it existing SSL algorithms, achieving significantly improved performance and reduced running time. When the recent FixMatch algorithm is plugged in as the SSL module, we improve state-of-the-art results on 3 benchmarks typically used to evaluate SSL algorithms. The proposed approach is general, in that both the SSL and the clustering modules can be replaced, although in this work we experimented with a single clustering method.

From a broader perspective, our approach can take advantage of curriculum learning [1], as one might look for means to schedule the supervised and unsupervised phases in a more sophisticated manner during training. It can also benefit from active learning [43], when seeking the best permutation between labels and clusters in the course of learning, which is especially tricky when the number of labeled points per class is very small. Under the active learning framework, where the learner can opt for a specific

label of interest, we can adjust the permutation gradually by clustering the data first, subsequently asking the user to provide labels for each cluster's centroid. This way, in each round of communication the permutation can be tuned with less than one additional label per class, as only labels from uncertain clusters will be requested.

BIBLIOGRAPHY

[1] Y. BENGIO, J. LOURADOUR, R. COLLOBERT, AND J. WESTON, *Curriculum learning*, in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41–48.

[2] D. BERTHELOT, N. CARLINI, E. D. CUBUK, A. KURAKIN, K. SOHN, H. ZHANG, AND C. RAFFEL, *Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring*, arXiv preprint arXiv:1911.09785, (2019).

[3] D. BERTHELOT, N. CARLINI, I. GOODFELLOW, N. PAPERNOT, A. OLIVER, AND C. A. RAFFEL, *Mixmatch: A holistic approach to semi-supervised learning*, in Advances in Neural Information Processing Systems, 2019, pp. 5049–5059.

[4] C. M. BISHOP, *Pattern recognition and machine learning*, springer, 2006.

[5] P. BOJANOWSKI AND A. JOULIN, *Unsupervised learning by predicting noise*, arXiv preprint arXiv:1704.05310, (2017).

[6] J. CHANG, L. WANG, G. MENG, S. XIANG, AND C. PAN, *Deep adaptive image clustering*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2017, pp. 5880–5888.

[7] O. CHAPELLE, B. SCHLKOPF, AND A. ZIEN, *Semi-Supervised Learning*, The MIT Press, 1st ed., 2010.

[8] T. CHEN, X. ZHAI, M. RITTER, M. LUCIC, AND N. HOULSBY, *Self-supervised gans via auxiliary rotation loss*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 12154–12163.

[9] A. COATES, A. NG, AND H. LEE, *An analysis of single-layer networks in unsupervised feature learning*, in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 215–223.

[10] E. D. CUBUK, B. ZOPH, D. MANE, V. VASUDEVAN, AND Q. V. LE, *Autoaugment: Learning augmentation strategies from data*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 113–123.

[11] E. D. CUBUK, B. ZOPH, J. SHLENS, AND Q. 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, 2020, pp. 702–703.

[12] J. DENG, W. DONG, R. SOCHER, L.-J. LI, K. LI, AND L. FEI-FEI, *Imagenet: A large-scale hierarchical image database*, in 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[13] T. DEVRIES AND G. W. TAYLOR, *Improved regularization of convolutional neural networks with cutout*, arXiv preprint arXiv:1708.04552, (2017).

[14] C. DOERSCH, A. GUPTA, AND A. A. EFROS, *Unsupervised visual representation learning by context prediction*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1422–1430.

[15] I. DÓPIDO, J. LI, P. R. MARPU, A. PLAZA, J. M. B. DIAS, AND J. A. BENEDIKTSSON, *Semisupervised self-learning for hyperspectral image classification*, IEEE transactions on geoscience and remote sensing, 51 (2013), pp. 4032–4044.

[16] S. GIDARIS, A. BURSUC, N. KOMODAKIS, P. PÉREZ, AND M. CORD, *Boosting few-shot visual learning with self-supervision*, in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8059–8068.

[17] S. GIDARIS, P. SINGH, AND N. KOMODAKIS, *Unsupervised representation learning by predicting image rotations*, arXiv preprint arXiv:1803.07728, (2018).

[18] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial nets*, in Advances in neural information processing systems, 2014, pp. 2672–2680.

[19] Y. GRANDVALET AND Y. BENGIO, *Semi-supervised learning by entropy minimization*, vol. 17, 01 2004.

[20] J. GUÉRIN, O. GIBARU, S. THIERY, AND E. NYIRI, *Cnn features are also great at unsupervised classification*, arXiv preprint arXiv:1707.01700, (2017).

[21] Y. GUO, H. ZHANG, AND X. LIU, *Instance selection in semi-supervised learning*, in Canadian Conference on Artificial Intelligence, Springer, 2011, pp. 158–169.

[22] C.-C. HSU AND C.-W. LIN, *Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data*, IEEE Transactions on Multimedia, 20 (2017), pp. 421–429.

[23] P. HUANG, Y. HUANG, W. WANG, AND L. WANG, *Deep embedding network for clustering*, in 2014 22nd International conference on pattern recognition, IEEE, 2014, pp. 1532–1537.

[24] X. JI, J. F. HENRIQUES, AND A. VEDALDI, *Invariant information clustering for unsupervised image classification and segmentation*, in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9865–9874.

[25] T. KANUNGO, D. M. MOUNT, N. S. NETANYAHU, C. D. PIATKO, R. SILVERMAN, AND A. Y. WU, *An efficient k-means clustering algorithm: Analysis and implementation*, IEEE transactions on pattern analysis and machine intelligence, 24 (2002), pp. 881–892.

[26] D. P. KINGMA, S. MOHAMED, D. JIMENEZ REZENDE, AND M. WELLING, *Semi-supervised learning with deep generative models*, Advances in neural information processing systems, 27 (2014), pp. 3581–3589.

[27] D. P. KINGMA AND M. WELLING, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114, (2013).

[28] A. KRIZHEVSKY, G. HINTON, ET AL., *Learning multiple layers of features from tiny images*, (2009).

[29] H. W. KUHN, *The Hungarian method for the assignment problem*, Naval Research Logistics (NRL), 52 (2005), pp. 7–21.

[30] D.-H. LEE, *Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks*, in Workshop on challenges in representation learning, ICML, vol. 3, 2013.

[31] ——, *Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks*, in Workshop on challenges in representation learning, ICML, vol. 3, 2013.

[32] D. D. LEWIS AND J. CATLETT, *Heterogeneous uncertainty sampling for supervised learning*, in Machine learning proceedings 1994, Elsevier, 1994, pp. 148–156.

[33] C. LI, T. XU, J. ZHU, AND B. ZHANG, *Triple generative adversarial nets*, Advances in neural information processing systems, 30 (2017), pp. 4088–4098.

[34] K. G. MURTY, *Letter to the editor,Äîan algorithm for ranking all the assignments in order of increasing cost*, Operations research, 16 (1968), pp. 682–687.

[35] Y. NETZER, T. WANG, A. COATES, A. BISSACCO, B. WU, AND A. Y. NG, *Reading digits in natural images with unsupervised feature learning*, (2011).

[36] M. NOROOZI AND P. FAVARO, *Unsupervised learning of visual representations by solving jigsaw puzzles*, in European Conference on Computer Vision, Springer, 2016, pp. 69–84.

[37] S. J. PAN AND Q. YANG, *A survey on transfer learning*, IEEE Transactions on knowledge and data engineering, 22 (2009), pp. 1345–1359.

[38] A. RASMUS, M. BERGLUND, M. HONKALA, H. VALPOLA, AND T. RAIKO, *Semi-supervised learning with ladder networks*, in Advances in neural information processing systems, 2015, pp. 3546–3554.

[39] S.-A. REBUFFI, S. EHRHARDT, K. HAN, A. VEDALDI, AND A. ZISSERMAN, *Semi-supervised learning with scarce annotations*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 762–763.

[40] D. E. RUMELHART, G. E. HINTON, AND R. J. WILLIAMS, *Learning internal representations by error propagation*, tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[41] T. SALIMANS, I. GOODFELLOW, W. ZAREMBA, V. CHEUNG, A. RADFORD, AND X. CHEN, *Improved techniques for training gans*, Advances in neural information processing systems, 29 (2016), pp. 2234–2242.

[42] T. SCHEFFER, C. DECOMAIN, AND S. WROBEL, *Active hidden markov models for information extraction*, in International Symposium on Intelligent Data Analysis, Springer, 2001, pp. 309–318.

[43] B. SETTLES, *Active learning literature survey*, tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[44] G. SHIRAN AND D. WEINSHALL, *Multi-modal deep clustering: Unsupervised partitioning of images*, Proceedings: 25th International Conference on Pattern Recognition (ICPR), Milano Italy, January 2021.

[45] K. SOHN, D. BERTHELOT, C.-L. LI, Z. ZHANG, N. CARLINI, E. D. CUBUK, A. KURAKIN, H. ZHANG, AND C. RAFFEL, *Fixmatch: Simplifying semi-supervised learning with consistency and confidence*, arXiv preprint arXiv:2001.07685, (2020).

[46] A. TARVAINEN AND H. VALPOLA, *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results*, in Advances in neural information processing systems, 2017, pp. 1195–1204.

[47] J. E. VAN ENGELEN AND H. H. HOOS, *A survey on semi-supervised learning*, Machine Learning, 109 (2020), pp. 373–440.

[48] W. VAN GANSBEKE, S. VANDENHENDE, S. GEORGOULIS, M. PROESMANS, AND L. VAN GOOL, *Scan: Learning to classify images without labels*, in European Conference on Computer Vision, Springer, 2020, pp. 268–285.

[49] R. VILALTA AND Y. DRISSI, *A perspective view and survey of meta-learning*, Artificial intelligence review, 18 (2002), pp. 77–95.

[50] Y. WANG, Q. YAO, J. T. KWOK, AND L. M. NI, *Generalizing from a few examples: A survey on few-shot learning*, ACM Computing Surveys (CSUR), 53 (2020), pp. 1–34.

[51] J. XIE, R. GIRSHICK, AND A. FARHADI, *Unsupervised deep embedding for clustering analysis*, in International conference on machine learning, 2016, pp. 478–487.

[52] Q. XIE, Z. DAI, E. HOVY, M.-T. LUONG, AND Q. V. LE, *Unsupervised data augmentation for consistency training*, arXiv preprint arXiv:1904.12848, (2019).

[53] B. YANG, X. FU, N. D. SIDIROPOULOS, AND M. HONG, *Towards k-means-friendly spaces: Simultaneous deep learning and clustering*, in international conference on machine learning, PMLR, 2017, pp. 3861–3870.

[54] J. YANG, D. PARIKH, AND D. BATRA, *Joint unsupervised learning of deep representations and image clusters*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5147–5156.

[55] S. ZAGORUYKO AND N. KOMODAKIS, *Wide residual networks*, arXiv preprint arXiv:1605.07146, (2016).

[56] X. ZHAI, A. OLIVER, A. KOLESNIKOV, AND L. BEYER, *S4l: Self-supervised semi-supervised learning*, in Proceedings of the IEEE international conference on computer vision, 2019, pp. 1476–1485.