Proceedings of Machine Learning Research 157, 2021 ACML 2021

A Two-Stage Training Framework with Feature-Label Matching Mechanism for Learning from Label Proportions [*]

Haoran Yang HRYANG@SE.CUHK.EDU.HK
The Chinese University of Hong Kong, Sha Tin, Hong Kong

Wanjing Zhang WANJINGZ123@GMAIL.COM
Central University of Finance and Economics, Beijing, China

Wai Lam WLAM@SE.CUHK.EDU.HK
The Chinese University of Hong Kong, Sha Tin, Hong Kong

Abstract

In this paper, we study a task called Learning from Label Proportions (LLP). LLP aims to learn an instance-level classifier given a number of bags and each bag is composed of several instances. The label of each instance is concealed and what we know is the proportion of each class in each bag. The lack of instance-level supervision information makes the model struggle for finding the right direction for optimization. In this paper, we solve this problem by developing a two-stage training framework. First, we facilitate contrastive learning to train a feature extractor in an unsupervised way. Second, we train a linear classifier with the parameter of the feature extractor fixed. This framework performs much better than most baselines but is still unsatisfactory when the bag size or the number of classes is large. Therefore, we further propose a **F**eature-**L**abel **M**atching **m**echanism (FLMm). FLMm can provide a roughly right optimization direction for the classifier by assigning labels to a subset of instances selected in this bag with a high degree of confidence. Therefore, the classifier can more easily establish the correspondence between instances and labels in the second stage. Experimental results on two benchmark datasets, namely CIFAR10 and CIFAR100, show that our model is far superior than baseline models, for example, accuracy increases from 43.44% to 61.25% for bag size 128 on CIFAR100. Code is available at `https://github.com/LHRYANG/LLP_FLMm`.

Keywords: Learning from Label Proportions, Contrastive Learning, Matching Mechanism, Multiple Instance Learning

1. Introduction

Weakly supervised learning (WSL) Zhou (2017) aims to utilize *limited* or *incomplete* sources to establish a powerful model. For example, semi-supervised learning (SSL) Chapelle et al. (2010) is a kind of WSL problem where the dataset is composed of a small amount of labeled data and a large amount of unlabeled data. In this paper, we delve into another WSL problem - Learning from Label Proportions (LLP) Patrini et al. (2014); Tsai and Lin (2019); Yu et al. (2015), which has many real-world applications, e.g., spam email filtering Quadrianto et al. (2009), disease diagnosis Hernández-González et al. (2018); Tokunaga et al. (2020), political elections Fish and Reyzin (2017), etc. The LLP dataset comprises a number of bags and each bag is composed of several instances and

[*] The work described in this paper is substantially supported by a grant from the Asian Institute of Supply Chains and Logistics, the Chinese University of Hong Kong.

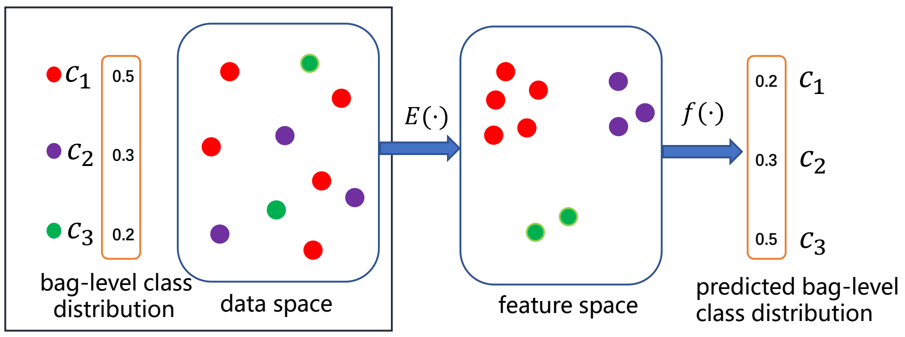

Figure 1: A general training paradigm of LLP tasks. The black square contains an example of a bag with size 10. $E(\cdot)$ and $f(\cdot)$ represent the feature extractor and the classifier respectively.

equipped with information on the proportion of each class. An example is illustrated in Figure 1. Given these bag-level information, the goal is to build an instance-level classifier.

The main difficulty for LLP is that the model is intrinsically harder to train since it can not easily find the right direction for optimization. To gain some insight of this phenomenon, an intuitive explanation is provided. In Figure 1, instances in the bag are firstly encoded by a feature extractor $E(\cdot)$, then a classifier $f(\cdot)$ is applied to calculate the predicted probability of each instances. The predicted bag-level class distribution is the average of all instances' probabilities. To train the model, the Kullback Leibler divergence (KLD) between the ground truth and the predicted bag-level class distributions is usually used as the loss function. Suppose we have already established a powerful $E(\cdot)$, then the instances with the same label will be clustered together in the feature space. In this situation, the classifier $f(\cdot)$ may assign the *same* label (may not be the correct label) to a cluster of instances belonging to the same class. In this example, $f(\cdot)$ classifies the purple points correctly but wrongly classifies the red points to c_3 and green points to c_1. When calculating the KLD loss, we can see that there is a difference between 0.2 and 0.5 which means that c_1 and c_3 account for the whole KLD loss. So the optimizer tries to narrow down this difference. However, this only reveals that green points and red points are not likely to belong to c_1 and c_3 respectively without telling which instances should be classified to c_1 and c_3. Therefore, the classifier has to explore all possible combinations between instances and labels, making it difficult to converge to the optimum. Moreover, two factors aggravate this phenomenon. The first factor is the training scheme. Recall that the above analysis is based on the premise that $E(\cdot)$ is well trained in advance. If $E(\cdot)$ and $f(\cdot)$ are jointly trained in an end-to-end training scheme, not only the $f(\cdot)$ but also the $E(\cdot)$ (usually a huge network) will not be well optimized. The second factor is the bag size and the number of classes. The growth of the number of classes or the bag size results in much more combinations between instances and labels. In other words, the search space will become huge. That's why the performance of previous models Yu et al. (2013); Dulac-Arnold et al. (2019); Liu et al. (2021) is attenuated significantly in situations where the number of class categories or the bag size is large.

To help the model converge to its optimum quickly, we propose a two-stage training framework where the first stage focuses on training a feature extractor and the second stage is responsible for training a linear classifier. In the first stage, we utilise SimCLR Chen et al. (2020a,b), a contrastive learning framework, to train the feature extractor. SimCLR is an unsupervised feature extraction technique and can generate as good a feature space as that based on supervised learning. The generated features have the following property: inputs belonging to the same class are mapped closer and inputs belonging to different classes are mapped far apart in feature space. With a well-trained feature extractor, in the second stage, we only need to optimize a simple linear classifier which is much easier than optimizing the feature extractor and classifier jointly. However, as the bag size or the number of classes becomes larger, it is still difficult for the classifier to converge using the bag-level KLD loss. The reason has been explained in the above paragraph. Therefore, we further put forward a **F**eature-**L**abel Matching mechanism (FLMm). This mechanism is motivated by the desired property that the instances with the same label should be clustered together if the pretrained feature extractor is powerful enough. Based on this property, in the second stage, a linear classifier even without well trained can generate *analogous* probability distributions for instances with the *same* label, although it is possible that the classifier makes an incorrect prediction. FLMm takes advantage of the probability similarity between the predicted bag-level class distribution and the ground truth bag-level class distribution to assign labels to a subset of instances elaborately selected in this bag. Then, instance-level supervised training is conducted on the selected instances and their assigned labels. We show that FLMm can provide a a roughly right optimization direction to speed up the convergence of the classifier. Our proposed two-stage training framework with FLMm can achieve dramatic improvements compared with previous proposed models on two image classification datasets CIFAR10 and CIFAR100.

2. Related Work

2.1. Learning from Label Proportions

Practically, it is not only time-consuming but also expensive to obtain a large number of labeled data, not to mention privacy issue. In this context, learning from label proportions (LLP), which trains the classifier only using the label proportion at bag level rather than the independent label of each instance, comes into being, and has been widely applied to many fields including the census Patrini et al. (2014), medical research Hernández-González et al. (2018) and object recognition Kück and de Freitas (2005). Previously works can roughly be divided into two categories: SVM-based and deep learning methods. SVM-based methods including InvCal Rueping (2010), $\propto$SVM Yu et al. (2013), NPSVM Qi et al. (2016) aim to model LLP as a SVM optimization problem. InvCal puts forward a "super instance" concept which is the mean of each bag and each super instance is associated with a soft label. However, the mean representation may sometimes not be a good representation of the whole bag which could result in poor performance. Instead, Yu et al. propose $\propto$SVM which explicitly models the unknown label for each instance in a bag. Besides, they also provide two algorithms to optimize their model by alternatively fixing labels and fixing classifier parameters. Qi propose NPSVM which tries to determine labels of instances according to two nonparallel hyper-planes under the supervision of label proportion information. However, all the above methods can only solve the binary classification LLP problem and lag behind in the era of deep learning. Therefore, Dulac-Arnold et al. (2019) put forward ROT which can be applied to general nonlinear multiclass setting. To learn a smooth classifier which is consistent when the input is perturbed, Tsai and Lin (2019) introduce consistency regularization as an auxiliary loss term. Liu et al. (2019) integrate generative adversarial network (GAN) into the LLP framework which achieves boosting improvement. Besides these two categories, there are also some works that attempt to use cluster Stolpe and Morik (2011) and label propagation Poyiadzi et al. (2018) techniques, which will not be described here.

2.2. Contrastive Learning

There have been many methods emerging during the last few years for unsupervised feature extraction since the amount of unlabeled data is substantial. For instance, VAE Kingma and Welling (2014), discrete VAE van den Oord et al. (2017) and NVAE Vahdat and Kautz (2020), which are based on the variational bayesian inference, try to reconstruct the original input after compressing it into a low-dimensional Gaussian distribution and achieve impressive performance especially in computer vision domain. In natural language domain, generalised language models, such as GPT-2 Radford et al. (2018), GPT-3 Brown et al. (2020) and BERT Devlin et al. (2019) are proposed and trained by defining different tasks, , predicting next word or masked word and judging whether two sentences are adjacent. Recently, contrastive learning Chen et al. (2020a); He et al. (2020); Khosla et al. (2020); Oord et al. (2018); Tian et al. (2020); Yang et al. (2021) has demonstrated its power for feature extraction in unsupervised way so we want to take advantage of it to achieve fast convergence as well as excellent feature extraction in LLP. SimCLR Chen et al. (2020a) generates two transformed images from the same image and introduces a contrastive loss which is capable of bringing same class features together and keeping different classes features away from each other. Experiment on supervised image classification tasks shows that pretrained feature extractor coupled with a linear classifier surpass the end-to-end training framework. Iter et al. (2020) utilise contrastive learning for pretraining language models which is beneficial to improve discourse performance. The above successes inspire us to solve the LLP problem by a two-stage training procedure.

3. Proposed Model

3.1. Overview

First of all, we describe some commonly used notations and give a formal definition of LLP problem. Suppose the label set is $\mathbb{C}$ which contains K classes $(c_1, c_2, .., c_K)$. The training dataset is represented by $\mathbb{D}_{train}$, which is composed of n instances $(x_1, x_2, ..., x_n)$ together with their corresponding label vectors $(\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_n})$ where $\mathbf{y_i} \in \mathbb{R}^K$ is a one-hot vector. Each bag $\mathcal{B}_i = (x_1^i, x_2^i, \cdots, x_{m_i}^i)$ consists of m_i instances sampled from $\mathbb{D}_{train}$. Then we can obtain the bag class distribution $\mathbf{p}_i$ given by

$$\mathbf{p}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathbf{y}_j \tag{1}$$

We can see that the k-th element $\mathbf{p}_i^k$ of $\mathbf{p}_i$ is the proportion of instances belonging to the class k with the constraint $\sum_{k=1}^{K} \mathbf{p}_i^k = 1$.

Given a bunch of such bags with their corresponding bag-level class distributions $\{(\mathcal{B}_i, \mathbf{p}_i)\}_i$, LLP aims to learn an instance-level classifier which will be tested on $\mathbb{D}_{test}$ after training. There are two points that should be noticed: (1) In real scenarios, the bag size m_i may be different for different bags. Here, just as previous works did, all bags are assumed to have the same size, uniformly expressed as m. If $m = 1$, LLP degrades to the normal supervised learning problem. We shall see in detail later, m has a significant impact on the performance of the model. (2) During training time, the label of each instance in a bag is not accessible while in testing time, each instance is provided with the ground truth label so as to evaluate the performance of LLP models.

3.2. Extractor Training with Contrastive Learning

Compared with other unsupervised learning families, e.g., auto-encoder Baldi (2011), variational auto-encoder Kingma and Welling (2014), which extract low-dimensional features by reconstructing the input, contrastive learning Chen et al. (2020a); Tian et al. (2019) takes the relationship between different instances into consideration. It tries to minimize the distance between instances with same label and maximize the distance between those with different labels. However, the label information is not really needed by generating two different *views* of the same input so that it is capable of extracting features of input in an unsupervised way.

3.2.1. FEATURE EXTRACTOR

Firstly, an instance x is transformed into two augmented instances by some transformations $x^a = T_i(x)$, $x^b = T_j(x)$, $T_i, T_j \in \mathbb{T}$. Here, we denote the transformation set by $\mathbb{T}$ which comprises several commonly used transformations, e.g., ResizedCrop, HorizontalFlip, ColorJitter, Grayscale. We follow the same setting provided in *SupContrast* Khosla et al. (2020). Specifically, T_i, which randomly crops the instance and then resizes that back to the original instance size, is always selected. T_j is sampled from $\mathbb{T}$ randomly. The augmented instances are then fed into the encoder $E(\cdot)$ to generate the representation vectors $\mathbf{z}^a = E(x^a)$, $\mathbf{z}^b = E(x^b)$. The encoder allows various choices of network architectures and here we adopt ResNet50 He et al. (2016) architecture and note that ResNet50 is not pretrained.

3.2.2. CONTRASTIVE LOSS

To optimize the feature extractor, contrastive loss Chen et al. (2020a) is utilized. Concretely, given a batch of instances $(x_1, x_2, \cdots, x_N)$, $2N$ augmented instances denoted by $(x_1^a, x_1^b, x_2^a, x_2^b, \cdots, x_N^a, x_N^b)$ are generated by applying the transformation. And then, $2N$ instances are inputted into the extractor $E(\cdot)$ to generate $2N$ features $(\mathbf{z}_1^a, \mathbf{z}_1^b, \mathbf{z}_2^a, \mathbf{z}_2^b, \cdots, \mathbf{z}_N^a, \mathbf{z}_N^b)$. For a specific vector $\mathbf{z}_i^a$, the positive pair is $(\mathbf{z}_i^a, \mathbf{z}_i^b)$, and $\mathbf{z}_i^a$ with the other remaining $2N-2$ features forms $(2N-2)$ negative pairs. For $\mathbf{z}_i^b$, it is the same as $\mathbf{z}_i^a$. The loss takes the following form

$$\mathcal{L}_i^a = -log\frac{exp(\mathbf{z}_i^a \cdot \mathbf{z}_i^b/\tau)}{exp(\mathbf{z}_i^a \cdot \mathbf{z}_i^b/\tau) + \sum_{j\neq i}(exp(\mathbf{z}_i^a \cdot \mathbf{z}_j^a/\tau) + exp(\mathbf{z}_i^a \cdot \mathbf{z}_j^b/\tau))} \tag{2}$$

$$\mathcal{L}_i^b = -log\frac{exp(\mathbf{z}_i^b \cdot \mathbf{z}_i^a/\tau)}{exp(\mathbf{z}_i^b \cdot \mathbf{z}_i^a/\tau) + \sum_{j\neq i}(exp(\mathbf{z}_i^b \cdot \mathbf{z}_j^a/\tau) + exp(\mathbf{z}_i^b \cdot \mathbf{z}_j^b/\tau))} \tag{3}$$

$$\mathcal{L} = \sum_{i=1}^{N}(\mathcal{L}_i^a + \mathcal{L}_i^b) \tag{4}$$

where $\cdot$ represents the dot product between two vectors and τ denotes a temperature parameter.

After training, we obtain the vector representation $\mathbf{z}_i$ in low-dimensional feature space for each instance x_i by concatenating $\mathbf{z}_i^a$ and $\mathbf{z}_i^b$. These vectors have a property: the distance between two vectors is small if two instances have the same label, otherwise the distance is large (See Figure 3 and Figure 4 as an illustration). This property lays the foundation for Stage 2 which is described below.

3.3. Linear Classifier Training

3.3.1. Bag-Level KLD Loss

With a well-trained $E(\cdot)$, in Stage 2, we only need to train a simple linear classifier $f(\mathbf{z}) = \sigma(\mathbf{W}\mathbf{z})$ with the knowledge of each bag's class distribution. σ represents the softmax function. Specifically, for each bag $\mathcal{B}_i = (x_1^i, x_2^i, \cdots, x_m^i)$, we firstly obtain their corresponding features $(\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m)$ and then feed them into $f(\cdot)$ to output their probability distributions $(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2, \cdot, \tilde{\mathbf{p}}_m)$, where $\tilde{\mathbf{p}}_i$ is a K-element vector. Although the label of each instance in this bag is not accessible, we can still employ KLD loss into a bag-level version by averaging the generated probabilities of each bag. We have

$$\mathbf{q}_i = \frac{1}{m}\sum_{j=1}^{m}\tilde{\mathbf{p}}_j \tag{5}$$

$$\mathcal{L}_i^{kld} = \mathbf{p}_i^{\intercal} log(\frac{\mathbf{p}_i}{\mathbf{q}_i}) \tag{6}$$

where $\mathbf{p}_i$ and $\mathbf{q}_i$ are the ground-truth and predicted bag-level class distribution respectively.

In addition, we impose an entropy regularization Grandvalet and Bengio (2004) to $\tilde{\mathbf{p}}_i$ with the purpose of encouraging it to converge to a one-hot vector whose entropy is zero, giving

$$\mathcal{L}_i^{et} = \sum_{j=1}^{m} -\tilde{\mathbf{p}}_j^{\intercal} log(\tilde{\mathbf{p}}_j) \tag{7}$$

The total loss for bag i is

$$\mathcal{L}_i = \mathcal{L}_i^{kld} + \lambda\mathcal{L}_i^{et} \tag{8}$$

3.3.2. Feature-Label Matching Mechanism

The two-stage model proposed above has exceeded most baselines. But the classification accuracy drops significantly as the bag size becomes larger, especially for large class number K. This happens because the linear classifier still fails to find the direction of optimization even if the two-stage training framework has already reduced the difficulty of optimization. To alleviate this issue, we propose a Feature-Label Matching mechanism (FLMm). FLMm is motivated by such an observation: if $E(\cdot)$ is trained perfectly, the instance clusters in the feature space, sorted in the descending order of their size, should correspond to the labels sorted in the descending order of their proportions. For example, in Figure 2, instances [1,7] form the largest cluster and they should be classified to c_1 whose proportion is the largest in this bag according to $\mathbf{p}_i$. Analogously, instances [18,23] should be classified to c_5. This kind of feature cluster-label matching can provide the classifier instance-level supervision information for optimization. A straightforward implementation is to cluster these features firstly and then assign each cluster a label. However, it has several issues. The first issue is that several clusters may have the same size which causes some confusion when assigning labels. The second issue is that the order of clusters and the order of labels are not necessarily corresponding to each other since the feature extractor is usually not perfect. A wrong cluster-label matching will mislead the classifier when optimizing.

Therefore, FLMm is designed as a fuzzy matching algorithm. Before starting to introduce this mechanism, we first define $\mathbf{h}_i$ as the *hard* generated bag-level class distribution and the j-th element

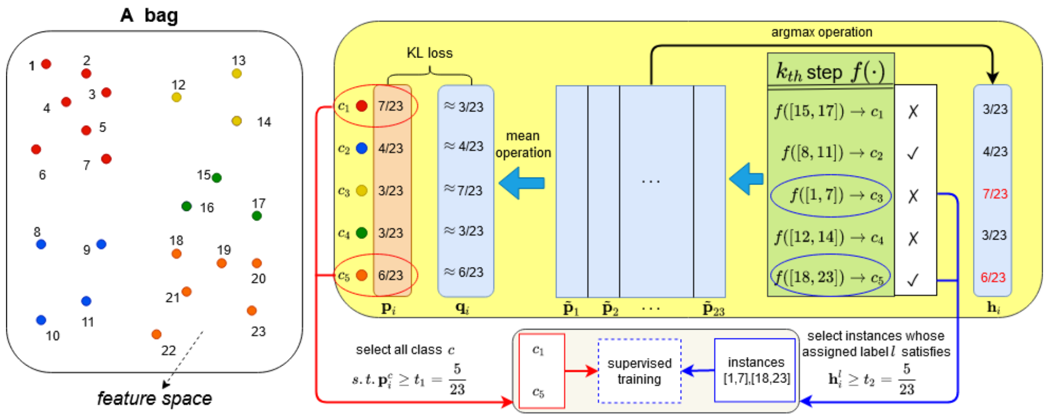

Figure 2: Given a bag of instances, we firstly obtain their corresponding feature vectors with the help of the feature extractor pretrained in Stage 1. Under ideal circumstances, features with the same label are clustered together as shown in the left part. Then, the classifier is trained by employing the bag-level KLD divergence loss (yellow background part). To further training the classifier, instances $[1, 7], [18, 23]$ are selected based on $\mathbf{h}_i$ (blue arrow) and c_1, c_5 are selected based on $\mathbf{p}_i$ (red arrow). Finally, instance-level supervised learning is conducted to optimize the classifier.

of $\mathbf{h}_i$ is

$$\mathbf{h}_i^j = \frac{1}{m}\sum_{k=1}^{m} \mathbb{1}(\arg\max(\tilde{\mathbf{p}}_k) = j) \tag{9}$$

where $\mathbb{1}(s) = 1$ if s is true, otherwise 0. Obviously, the generated bag class distribution $\mathbf{q}_i$ is a *soft* version of $\mathbf{h}_i$. Hence, values of these two vectors are similar in an ideal situation. We use $\approx a$ to represent a value close to a. FLMm aims to tell what are the most likely labels for a subset of instances with high confidence in a bag. In FLMm, firstly we select all the class c whose corresponding ground-truth probability $\mathbf{p}_i^c$ is greater than or equal to a threshold parameter t_1. The selected set of classes is denoted by C. Then, we select instances whose predicted label l satisfies $\mathbf{h}_i^l \geq t_2$, where t_2 is another threshold parameter. We denote these instances by I. We can see that the instances in I belong to *one of the class* in C with a high probability.

After selecting instances I and classes C based on $\mathbf{h}_i$ and $\mathbf{p}_i$, we iteratively assign a label in C to these instances I and conduct supervised classification and then back propagation. Through this way, the classifier converges faster since the above supervised training provides a rough right direction to the optimizer. Here comes a question: what if a wrong label is assigned to some of the instances in I which have already been correctly classified ? We show that this situation will not influence much on the classifier. Consider the example in Figure 2, in the k-th step, $[18, 23]$ have already been classified to the correct label c_5 by $f(\cdot)$. But the model still need to do back propagation twice for $[18, 23]$, one for c_1 and the other for c_5 in a supervised manner. The worst case is that these two gradients are in the opposite direction. So the two gradients may cancel each other out and $f(\cdot)$ can still classify $[18, 23]$ to c_5 correctly. Since FLMm works in the worst case, it is even more valid in other cases. The full training process of the classifier is summarized in Algorithm 1.

Algorithm 1 Training Classifier for Bag $\mathcal{B}_i$

```
Require: Bag $\mathcal{B}_i = (x_1^i, x_2^i, \cdots, x_m^i)$, bag class distribution $\mathbf{p}_i$, pretrained feature extractor $E$, classifier $f$, parameter $t_1$, parameter $t_2$
 1: fix $E$
 2: $(\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m) = E(\mathcal{B}_i)$                {obtain features};
 3: $(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2, \cdots, \tilde{\mathbf{p}}_m) = f((\mathbf{z}_i, \mathbf{z}_2, \cdots, \mathbf{z}_m))$
 4: obtain $\mathbf{q}_i$ and $\mathbf{h}_i$ according to Equation 5 and Equation 9 respectively
 5: calculate $\mathcal{L}_i$ according to Equation 8 and conduct back propagation
 6: obtain the selected class set $C$ where for each $c \in C$ satisfies $\mathbf{p}_i^c \geq t_1$
 7: obtain the selected instances $I$ where for each instance $b \in I$, the predicted label $l$ satisfies $\mathbf{h}_i^l \geq t_2$
 8: for $c$ in $C$ do
 9:     suppose that the label of instances in $I$ is $c$, conduct supervised classification and then back propagation using cross entropy loss .
10: end for
```

4. Experiment and Analysis

4.1. Datasets & Evaluation Metric

We conduct experiments on two benchmark datasets CIFAR10 and CIFAR100 Krizhevsky (2012). Both datasets consist of 50000 training instances and 10000 testing instances. Each instance is a $3 \times 32 \times 32$ colored image. CIFAR10 has 10 classes and CIFAR100 has 100 classes, with each class containing 5000 and 500 training images respectively. We use the accuracy on test dataset as the evaluation metric.

4.2. Baselines

We compare our model with five baseline models:

1. VANILLA is an ordinary network which is trained in an end-to-end manner and it is optimized using the bag-level KLD loss.

2. ROT Dulac-Arnold et al. (2019) extends the idea $\propto$SVM, which tries to estimate the individual labels within each bag, to the multi-class setting. It also facilitates the method of Relax Optimal Transport (ROT) Peyré and Cuturi (2019) to make the loss function differentiable.

3. LLP-VAT Tsai and Lin (2019) aims to encourage the model to produce a decision boundary that better describes the data manifold by adding consistency regularization Berthelot et al. (2019); Verma et al. (2019). Specifically, it augments images by adding noise and the consistency loss tries to make the two generated probability distributions corresponding to original image and corrupted image close with each other.

4. LLP-GAN Liu et al. (2019) leverages generative adversarial networks Goodfellow et al. (2014) to train a classifier which can not only predict the label of instances but also can detect the difference between the real and the generated fake instances.

5. PLOT Liu et al. (2021) reduces the label noise by imposing the strict proportion consistency on the classifier and utilizes symmetric cross-entropy (SCE) Wang et al. (2019) and mixup dataset Zhang et al. (2018) to train classification network.

In addition to the above four methods, we also report the results of models trained in an *instance-level* supervised way including the end-to-end supervised model and two-stage supervised model. These two results can be regarded as the upper bound for LLP models.

Table 1: Accuracy on test dataset for CIFAR10 and CIFAR100. The best are bold.

Dataset	Methods	Bag size				
		16	32	64	128	256
CIFAR10	End-to-End Supervised			95.00		
	Two-Stage Supervised			93.03		
	VANILLA	88.77	85.02	70.68	47.48	38.69
	ROT Dulac-Arnold et al. (2019)	86.97	77.01	62.93	48.95	40.16
	LLP-VAT Tsai and Lin (2019)	89.30	85.41	72.49	50.78	41.62
	LLP-GAN Liu et al. (2019)	86.32	83.77	78.97	72.61	-
	PLOT Liu et al. (2021)	89.15	88.21	84.14	79.09	-
	Ours	**92.34**	**92.00**	**91.74**	**91.54**	**91.29**
	Ours w/o FLMm	91.99	91.60	91.20	90.64	90.05
CIFAR100	End-to-End Supervised			75.30		
	Two-Stage Supervised			69.66		
	VANILLA	58.58	48.09	20.66	5.82	2.82
	ROT Dulac-Arnold et al. (2019)	54.16	47.75	29.38	7.95	2.63
	LLP-VAT Tsai and Lin (2019)	59.47	48.98	22.84	9.40	3.29
	LLP-GAN Liu et al. (2019)	49.05	43.56	35.63	14.99	-
	PLOT Liu et al. (2021)	65.41	61.68	55.66	43.44	-
	Ours	**66.16**	**65.59**	**64.07**	**61.25**	**57.10**
	Ours w/o FLMm	55.99	51.27	46.16	32.20	12.09

4.3. Experimental Settings

We investigate the performance of different bag size $m \in \{16, 32, 64, 128, 256\}$. At each step, we sample a batch of 1024 instances to form a batch of bags. For example, if the bag size is 64, then the number of bags in a batch is 16 (equals to $\frac{1024}{64}$).

Our model comprises a feature extractor and a linear classifier. For the extractor, we make use of the existing implementation[1] released by Khosla Khosla et al. (2020). Specifically, the epoch number is set to 1000. The architecture of the extractor is ResNet50 He et al. (2016). Batch size is set to 512 and the temperature parameter τ is set to 0.5. SGD optimizer Bottou (2012) is used with learning rate 0.5 and momentum Sutskever et al. (2013) 0.9. Other strategies such as learning rate decay are also adopted to achieve stable training. More details of the training settings for extractor can be found in Khosla et al. (2020). After training, we can obtain a 2048 dimensional feature vector of each instance. For the classifier, the epoch number is set to 200. The balancing parameter λ is set to 0.01. The optimizer and related settings are the same as above settings for training the extractor except that the learning rate here is set to 1. The parameter t_1 used for selecting classes based on $\mathbf{p}_i$

1. https://github.com/HobbitLong/SupContrast

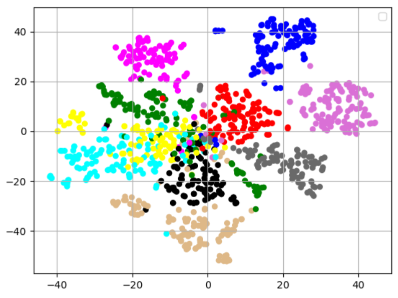
Figure 3: CIFAR10

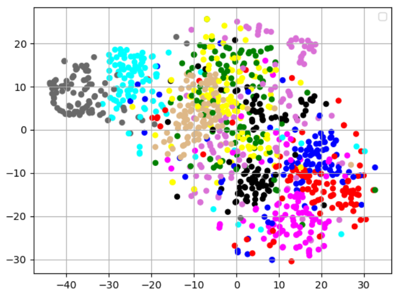
Figure 4: CIFAR100

and the parameter t_2 used for selecting instances based on $\mathbf{h}_i$ are two most important parameters to the accuracy of the classifier. For CIFAR10, both t_1 and t_2 are set to 0.1. For CIFAR100, both t_1 and t_2 are set to 0.01. We will explain how to select proper t_1 and t_2 in Section 4.4.2.

4.4. Results and Analysis

4.4.1. Classification Performance

As shown in Table 1, in supervised setting, End-to-End Supervised is superior than Two-Stage Supervised on CIFAR10 and CIFAR100. But the accuracy gap between the two models is not very large, indicating that the feature extractor is well-trained by SimCLR. End-to-End Supervised and Two-Stage Supervised provide an upper bound for LLP models.

First of all, we show that the two-stage framework even without FLMm can achieve superior performance than most of existing models. For CIFAR10, we can see that Ours w/o FLMm gains a huge improvement compared with VANILLA, ROT, LLP-VAT, LLP-GAN, PLOT on all choices of bag size. When the bag size increases, the accuracy of Ours w/o FLMm does not drop much compared with the accuracy on bag size 16 while for baselines, the accuracy drops significantly. For CIFAR100, Ours w/o FLMm is also superior than baselines except PLOT. As the bag size increases, the accuracy is still rather unsatisfying compared to that on small bag size. This shows that the number of classes also plays an important role on the performance of models and two-stage training without FLMm can only handle the bag size issue when the number of classes is small.

After adding FLMm, the accuracy is imporved on both CIFAR10 and CIFAR100. Specifically, for CIFAR10, Ours has an increase of about one percent on all bag size compared with Ours w/o FLMm. For CIFAR100, the improvement is huge on all bag size. For example, when the bag size is 16, the accuracy increases from 55.99% to 66.16%. When the bag size is 256, the accuracy increases from 12.09% to 57.10%. We can conclude that FLMm is of great importance for making the model converge towards optimum whether the bag size and the number of classes are small or large.

We use t-SNE van der Maaten and Hinton (2008) to visualise the features extracted by the extractor. For CIFAR10, we randomly sample 1024 instances and visualize their corresponding features. The result is shown in Figure 3. We can see that the features of 10 different classes are well learned which contributes the most to the high accuracy on CIFAR10. For CIFAR100, firstly, we randomly choose 10 classes and then select 100 instances for each class. The visualization

Table 2: Effects of t_1&t_2 for CIFAR10

(a) bag size:16

t_1 \ t_2	0.05(1)	0.1(2)	0.3(5)
0.05(1)	**92.48**	92.35	92.21
0.1(2)	92.39	92.34	91.56
0.3(5)	91.68	91.70	91.97

(b) bag size:32

t_1 \ t_2	0.05(2)	0.1(4)	0.3(10)
0.05(2)	92.15	92.06	91.27
0.1(4)	**92.18**	92.00	91.66
0.3(10)	91.69	91.63	91.62

(c) bag size:64

t_1 \ t_2	0.05(4)	0.1(7)	0.3(20)
0.05(4)	**92.17**	91.87	91.09
0.1(7)	92.12	91.74	91.05
0.3(20)	91.06	90.99	91.07

(d) bag size:128

t_1 \ t_2	0.05(7)	0.1(13)	0.3(39)
0.05(7)	91.09	91.29	90.60
0.1(13)	**91.89**	91.54	90.67
0.3(39)	90.55	90.62	90.52

(e) bag size:256

t_1 \ t_2	0.05(13)	0.1(26)	0.3(77)
0.05(13)	90.13	90.03	89.94
0.1(26)	**91.61**	91.29	90.09
0.3(77)	89.97	90.02	89.95

is depicted in Figure 4. We can see that most of the classes are roughly separated while some are tangled, , the green, yellow, and black classes. That's why the accuracy of two-stage models on CIFAR100 is not high. On the whole, unsupervised contrastive leaning is effective for feature extraction.

4.4.2. PARAMETERS t_1 AND t_2

We investigate the impact of t_1 and t_2 and provide a strategy about how to choose proper values for these two parameters. Consider two extreme cases. **Case 1:** $t_1 = t_2 = 0$, then all labels and all instances will be selected. This can mislead the direction of optimization because most of the matchings between features and labels are incorrect. **Case 2:** $t_1 > 1$ or $t_2 > 1$, then no label and no class will be selected. Therefore, FLMm no longer works. Intuitively, these two parameters cannot be too large or too small. We list the changes in accuracy given different t_1 and t_2 in Table 2 and Table 3 for CIFAR10 and CIFAR100 respectively. The numbers in brackets illustrate the threshold t_1 and t_2 from the perspective of the number of instances: for bag size m and threshold t_1, class c is selected if the number of instances belonging to c is greater than or equal to $\lceil m \times t_1 \rceil$ in this bag. It's the same for threshold t_2: if instances whose predicted label is l are selected, then the number of instances classified to l is greater than or equal to $\lceil m \times t_2 \rceil$. As shown in Table 2 and Table 3, for CIFAR10, our model is relatively stable for different values of these two parameters. We can see that for bag size 16 and 64, the best accuracy is achieved with $t_1 = 0.05$ and $t_2 = 0.05$. For other bag size, the best accuracy is achieved when $t_1 = 0.1$ and $t_2 = 0.05$. For CIFAR100, the best accuracy is achieved when $t_1 = 0.01$ and $t_2 = 0.01$ for all bag size. The other choices of parameters will lead to a decrease in accuracy. On the whole, we can set $t_1 = t_2 = \frac{1}{K}$ where K is

Table 3: Effects of t_1&t_2 for CIFAR100

(a) bag size:16

t_1 \ t_2	0.01(1)	0.1(2)	0.15(3)
0.01(1)	**66.16**	62.93	58.40
0.1(2)	62.27	61.25	58.31
0.15(3)	57.92	58.38	56.93

(b) bag size:32

t_1 \ t_2	0.01(1)	0.05(2)	0.08(3)
0.01(1)	**65.59**	63.38	60.00
0.05(2)	63.22	61.28	57.34
0.08(3)	57.60	58.33	56.29

(c) bag size:64

t_1 \ t_2	0.01(1)	0.03(2)	0.04(3)
0.01(1)	**64.07**	63.63	62.81
0.03(2)	62.24	61.78	59.98
0.04(3)	58.96	58.41	57.71

(d) bag size:128

t_1 \ t_2	0.001(1)	0.01(2)	0.03(4)
0.001(1)	58.17	58.77	59.16
0.01(2)	61.05	**61.25**	60.22
0.03(4)	55.72	56.04	52.56

(e) bag size:256

t_1 \ t_2	0.005(2)	0.01(3)	0.018(5)
0.005(2)	51.06	51.39	51.68
0.01(3)	56.39	**57.10**	56.83
0.018(5)	53.98	54.10	52.03

the number of classes to obtain relatively better accuracy. In other words, the threshold in terms of the number of instances is $\lceil m \times \frac{1}{K} \rceil$ which is the average number for each class in a bag with size m.

4.4.3. CONVERGENCE ANALYSIS

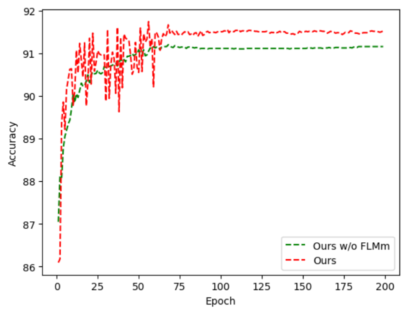

Figure 5: Acc. Curve for CIFAR10 with Bag Size 64

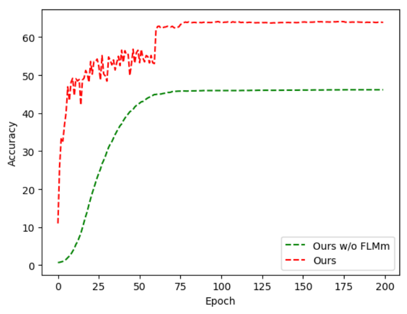

Figure 6: Acc. Curve for CIFAR100 with Bag Size 64

We also plot the trend of accuracy as epoch increases. As depicted in Figure 5 and Figure 6, in the first few epochs, the accuracy of Ours fluctuates a lot and this phenomenon may be caused by the poor quality of the classes and instances selected. After a specific epoch, the accuracy of Ours gradually becomes stable which shows the reliability of FLMm.

4.4.4. NECESSITY OF THE FIRST STAGE

Up to now, there is no evidence that the first step works. To demonstrate the pretraining of the extractor is necessary, we design a simple extractor, consisting of three convolutional layers and two fully connected layers. The results are shown in Table 4. #1, #2, #3 are all trained in an end-to-end manner (not two-stage training). We can observe that (1) model with FLMm (#2, #3) performs worse compared with that without FLMm (#1) (2) #2 (only optimizing the linear classifier in FLMm) performs better than #3 (optimizing both extractor and classifier in FLMm). The reason

Dataset	Methods	Bag size				
		16	32	64	128	256
CIFAR10	#1	85.67	77.30	58.98	37.89	27.65
	#2	46.84	41.68	37.84	34.03	28.32
	#3	42.74	35.35	31.33	27.51	19.08
CIFAR100	#1	38.64	17.07	6.84	4.63	2.59
	#2	15.49	13.68	13.19	14.39	2.92
	#3	9.41	6.38	4.09	1.92	1.25

Table 4: Results. **#1**: model trained using KLD loss. **#2**: model trained using KLD loss with FLMm. In FLMm, only the linear classifier are optimized. **#3**: model trained using KLD loss with FLMm. In FLMm, the linear classifier and extractor are jointly optimized.

for these two observation is that FLMm requires the feature space is well-organised i.e., instances with the same label should be close to each other. Otherwise, the assumption "instances in I belong to one of the class in C with a high probability" described in Section 2.3.2 will not hold. The completely incorrect matching between instances and labels will hinder the learning of the extractor and classifier.

5. Conclusion

In conclusion, it is instructive to decompose the training process into different parts to reduce the optimization complexity for LLP. Moreover, FLMm provides a straight way of associating instances of high confidence with most likely labels to guide the model to optimize towards the right direction. For future work, we plan to study a more general LLP problem where the distribution across classes of the dataset and the sampling method for constructing bags are not uniform. In addition, we will also try to give a mathematical analysis for LLP problems.

References

Pierre Baldi. Autoencoders, unsupervised learning and deep architectures. UTLW'11. JMLR.org, 2011.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2019.

Leon Bottou. Stochastic gradient descent tricks. In *Neural Networks, Tricks of the Trade, Reloaded*, pages 430–445, January 2012.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, 2020.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *ArXiv*, 2020b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019.

Gabriel Dulac-Arnold, Neil Zeghidour, M. Cuturi, Lucas Beyer, and Jean-Philippe Vert. Deep multi-class learning from label proportions. *ArXiv*, 2019.

Benjamin Fish and Lev Reyzin. On the complexity of learning from label proportions. In *International Joint Conferences on Artificial Intelligence*, page 1675–1681, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, page 529–536, 2004.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

J. Hernández-González, I. Inza, Lorena Crisol-Ortíz, María A Guembe, M. J. Iñarra, and J. Lozano. Fitting the data from embryo implantation prediction: Learning from label proportions. *Statistical Methods in Medical Research*, 27:1056 – 1066, 2018.

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. Pretraining with contrastive sentence objectives improves discourse performance of language models. In *Annual Meeting of the Association for Computational Linguistics*, pages 4859–4870, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *ArXiv*, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Hendrik Kück and Nando de Freitas. Learning about individuals from group statistics. In *Conference on Uncertainty in Artificial Intelligence*, page 332–339, 2005.

Jiabin Liu, Bo Wang, Zhiquan Qi, Yingjie Tian, and Yong Shi. Learning from label proportions with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 7169–7179, 2019.

Jiabin Liu, Bo Wang, Xin Shen, Zhiquan Qi, and Yingjie Tian. Two-stage training for learning from label proportions, 2021.

A. Oord, Y. Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, 2018.

Giorgio Patrini, Richard Nock, Paul Rivera, and Tiberio Caetano. (Almost) no label no cry. In *Advances in Neural Information Processing Systems*, pages 190–198. 2014.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11 (5-6):355–602, 2019. URL https://arxiv.org/abs/1803.00567.

R. Poyiadzi, R. Santos-Rodriguez, and N. Twomey. Label propagation for learning with label proportions. In *International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2018.

Zhiquan Qi, Bo Wang, Fan Meng, and Lingfeng Niu. Learning with label proportions via npsvm. *IEEE Transactions on Cybernetics*, PP:1–13, 08 2016. doi: 10.1109/TCYB.2016.2598749.

Novi Quadrianto, Alex J. Smola, Tibério S. Caetano, and Quoc V. Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(82):2349–2374, 2009.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

Stefan Rueping. SVM classifier estimation from group probabilities. In *International Conference on Machine Learning*, page 911–918, 2010.

Marco Stolpe and Katharina Morik. Learning from label proportions by optimizing cluster model selection. In *Machine Learning and Knowledge Discovery in Databases*, pages 349–364, 2011.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, page 1139–1147, 2013.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *Arxiv*, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SkgpBJrtvS`.

Hiroki Tokunaga, Brian Kenji Iwana, Yuki Teramoto, Akihiko Yoshizawa, and Ryoma Bise. Negative pseudo labeling using class proportion for semantic segmentation in pathology. In *European Conference on Computer Vision*, 2020.

Kuen-Han Tsai and Hsuan-Tien Lin. Learning from label proportions with consistency regularization. *Arxiv*, 2019.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, 2020.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315. 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *International Joint Conference on Artificial Intelligence*, pages 3635–3641, 7 2019.

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels, 2019.

Mouxing Yang, Yunfan Li, Zhenyu Huang, Zitao Liu, Peng Hu, and Xi Peng. Partially view-aligned representation learning with noise-robust contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1134–1143, June 2021.

Felix Yu, Dong Liu, Sanjiv Kumar, Jebara Tony, and Shih-Fu Chang. $\propto$SVM for learning with label proportions. In *Machine Learning Research*, pages 504–512, 2013.

Felix X. Yu, Krzysztof Choromanski, Sanjiv Kumar, Tony Jebara, and Shih-Fu Chang. On learning from label proportions. *Arxiv*, 2015.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

Zhi-Hua Zhou. A Brief Introduction to Weakly Supervised Learning. *National Science Review*, 5 (1):44–53, 08 2017.