Image and Vision Computing 112 (2021) 104213

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Discrepant collaborative training by Sinkhorn divergences

Yan Han [a,b,*], Soumava Kumar Roy [a,b], Lars Petersson [a,b], Mehrtash Harandi [b,c]

[a] Australian National University, New Acton, ACT 2601, Australia
[b] DATA61 CSIRO, Acton, ACT 2601, Australia
[c] Monash University, Clayton, VIC 3800, Australia

ARTICLE INFO

Article history:
Received 10 January 2021
Received in revised form 12 May 2021
Accepted 13 May 2021
Available online 21 May 2021

Keywords:
Co-training
Noisy labels
Weak-supervised learning

ABSTRACT

Deep Co-Training algorithms are typically comprised of two distinct and diverse feature extractors that simultaneously attempt to learn task-specific features from the same inputs. Achieving such an objective is, however, not trivial, despite its innocent look. This is because homogeneous networks tend to mimic each other under the collaborative training setup. Keeping this difficulty in mind, we make use of the newly proposed $S_\in$ divergence to encourage diversity between homogeneous networks. The $S_\in$ divergence encapsulates popular measures such as maximum mean discrepancy and the Wasserstein distance under the same umbrella and provides us with a principled, yet simple and straightforward mechanism. Our empirical results in two domains, classification in the presence of noisy labels and semi-supervised image classification, clearly demonstrate the benefits of the proposed framework in learning distinct and diverse features. We show that in these respective settings, we achieve impressive results by a notable margin.

1. Introduction

In this paper, we address the issue of learning diverse yet distinct models in a general Co-Training [1] (Co-Tr) framework. More specifically, we equip the Co-Tr learning paradigm with a novel discrepancy module that results in learning different yet complementary views of the input; thereby enhancing the overall discriminative ability of the entire Co-Tr module.

Designing diverse models is a long-standing problem in machine learning despite several breakthroughs [2–4]. The prime example is the boosting algorithm and its variants, where a number of different, weak classifiers are learned sequentially. However, in the era of deep learning, the capacity of boosting algorithms to produce strong classifiers have been vastly superseded, and more appropriate alternative methods to enforce diversity in a deep network need to be investigated.

To address the aforementioned need, what started as a simple learning of two conditionally independent views of classifying web-data [1], has now been employed across a wide variety of tasks such as image classification [5,6], text classification [4], email classification [7], and natural language processing [8] etc. In our preliminary study, we have employed Co-Tr framework with MMD to learn from noisy labels. Our work proved that improved Co-Tr could learn better from noisy labels. Co-Tr has also been used in a semi-supervised setting [6],

We stress that the diversity is a requirement for the success of the Co-Tr framework [9]. The predominant techniques to induce diversity in Co-Tr are (a) use of different network architectures [10], (b) random initialization schemes [11] in each of the individual networks, and (c) training each network with different sets of samples [5, 6]. While these tactics have achieved significant improvements in learning different but complementary views of the input, they suffer from a few setbacks. First, two or more networks with different architectures must be highly compatible with each other in order to jointly learn different yet complementary features given the same input data. Moreover, random initialization of the networks does not guarantee that the learnt features will indeed be diverse, distinct and informative enough for the task at hand. As a remedy, the recent work of Qiao et al utilized adversarial examples during training along with random initialization [6]. Nevertheless, one cannot guarantee the learning of complementary features as the distribution of the adversarial examples is very similar to the original training images. Interestingly, several recent works used identical homogeneous networks as the feature extractor units for co-training [5,12]. Nonetheless, these methods do not explicitly enforce a discrepancy between the homogeneous networks and may thereby fail in capturing inherent discriminative features across the different views.

In this paper, we address the above concern by explicitly enforcing a diversity constraint in the Co-Tr framework such that the underlying networks do learn different yet complementary views (or features) of the input even though they have the same architecture and different

* Corresponding author at: Australian National University, New Acton, ACT 2601, Australia.
E-mail addresses: yan.han@anu.edu.au (Y. Han), soumava.kumarroy@anu.edu.au (S.K. Roy), lars.petersson@data61.csiro.au (L. Petersson), mehrtash.harandi@monash.au (M. Harandi).

https://doi.org/10.1016/j.imavis.2021.104213
0262-8856/

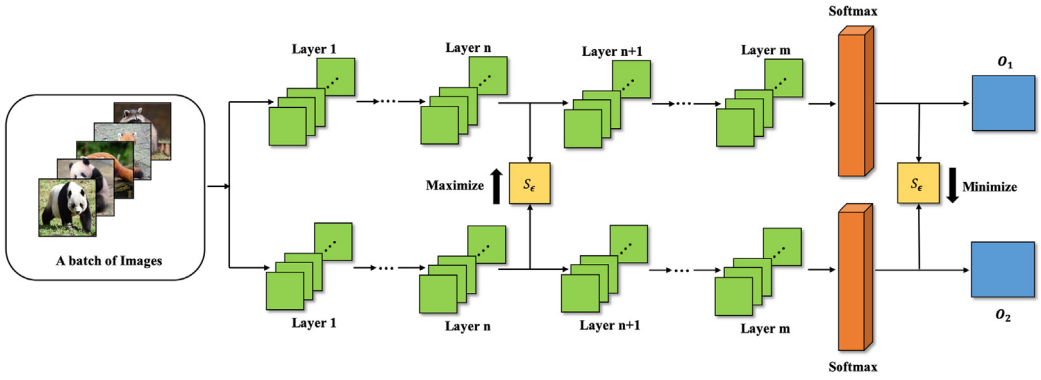

Fig. 1. Deep Co-Training with Discrepancy schematic. Propagation of Discrepant Collaborative Training. A batch of images are fed into homogeneous networks independently. The first $\mathcal{S}_\in$ module (orange box on left) is placed after *Layer n* between two networks. The second $\mathcal{S}_\in$ (orange box on right) module is placed after *softmax* layer. We maximize the first $\mathcal{S}_\in$ module to learn diverse features in each of the network, while the second $\mathcal{S}_\in$ is minimized so as to learn the same class distributions.

initialization. Both the two networks f and g have the same structure (Fig. 1). They are fed with same batch of input data and are trained on the same task: image classification. However, they will be updated by different losses. Our proposed module (yellow boxes in Fig. 1) helps to keep the homogeneous networks different from each other and improve the image classification accuracy. O_1 and O_2 are prediction outputs from networks. For the test stage, we gather the outputs of network f and g for each test image. We then select the larger value (most confident) as the final prediction of the test image. More details are explained in Section 5. We apply $\mathcal{S}_\in$ divergence [13] to measure diversity between networks. One can simply tune $\mathcal{S}_\in$ into Maximum Mean Discrepancy (MMD) or Optimal Transport (OT) distance by changing $\in$. The $\mathcal{S}_\in$ module can be seamlessly added to any part of a Co-Tr framework. This module drives the networks to robustly learn from different views but keeping the consistency of the outputs at same time. Fig. 1 provides a brief schematic of our proposed methodology.

Our major contributions are as follows:

- An explicit method to disentangle learnt representations via an $\mathcal{S}_\in$ based discrepancy module.
- An empirical study of the effectiveness of two well known variants of $\mathcal{S}_\in$ divergence measures, including Maximum Mean Discrepancy and Wasserstein Distance.
- An extensive set of experiments demonstrating the advantage of such a $\mathcal{S}_\in$ module across two different settings; *(i)* noisy labeled image classification, and *(ii)* semi-supervised image classification.

2. Extension of previous work

We extend our previous work to a more general module by applying more tasks, datasets and divergences. Our previous work can be considered as a special case of our proposed work here. Our theoretical contribution is to formulate the problem using Sinkhorn divergence. Sinkhorn divergence is a family of divergences where MMD and Wasserstein distance are considered as instances of this family. Our new work aims to cover more general notion of distance measures under the family of Sinkhorn divergences. As shown in Tables 3 and 4, the new method outperforms previous work by a large margin. Moreover, we also show how our contribution can be used for semi-supervised learning.

Table 1
Notation for symbols.

Symbols	Corresponding notations
$\mathcal{S}_\in$	Sinkhorn
O_1	Prediction output of first network
O_2	Prediction output of second network
$\in$	Smoothing parameter for Sinkhorn divergence
$\mathcal{U}$ and $\mathcal{V}$	positive random measures of unit mass over a metric space $\mathcal{X}$
$\{\boldsymbol{u}_i^{\mathcal{U}}\}_{i=1}^n$ and $\{\boldsymbol{v}_i^{\mathcal{V}}\}_{i=1}^m$	samples drawn from $\mathcal{U}$ and $\mathcal{V}$
$\mathbf{K}$	cost matrix
α and β	finite discrete measures
α_{x_i}	the probability of x_i to be at point α_i
π^*	Best transport plan
$\Pi(\boldsymbol{u},\boldsymbol{v})$	Set of transportation plans
$c(\boldsymbol{u},\boldsymbol{v})$	Cost function to move a unit of mass from $\mathcal{U}$ to $\mathcal{V}$
$\mathbf{C} \in \mathbb{R}^{n\times m}$	Matrix of cost function
$\otimes$	Product of the marginals
f and g	networks
θ and $\widehat{\theta}$	parameters of networks
η	learning rate
T	epoch number
l	the layer number for $\mathcal{S}_\in$
$\mathcal{D}$	Training set
$3\overline{\mathcal{D}}$	Mini-batch of training set
$\mathcal{D}_1$ and $\mathcal{D}_2$	Discrepancy module
Loss_f	Total loss to update network f
Loss_g	Total loss to update network g
L_M^f	Conventional supervised loss for network f
L_M^g	Conventional supervised loss for network g
L_D	Diversity loss
L_C	Consistency loss
λ_D	Combination weights for the diversity loss
λ_C	Combination weights for the consistency loss
$\boldsymbol{X}_i$	Input data
$\boldsymbol{A}_i$	Features extracted from lth layer of network f
$\boldsymbol{B}_i$	Features extracted from lth layer of network g
$\boldsymbol{z}_i^f$	Softmax output of network f
$\boldsymbol{z}_i^g$	Softmax output of network g

1. We extend the specific MMD divergence to Sinkhorn divergence which is a family of divergences, which can be further tuned to MMD or Wasserstein distance by tuning the hyper parameter

which is more general and flexible compared to our previous work. For applying our module to different tasks, one can always tries to achieve better performance by tuning the hyper parameter easily. And our experiment results shows that Sinkhorn Divergence provides better and robust performance than MMD divergence in most cases which means one can always achieve competitive results without tuning hyper parameters.

2. More datasets. We tested our methods on more and lager datasets such as Clothing1M [14] and ImageNet [15]. In our previous work, we focused on more common datasets for noisy labels problems such as MNIST, CIFAR10, CIFAR100, CUB200 and CARS196. In this work we not only include all those datasets but also studied on large scale datasets Clothing1M and ImageNet, results are shown in Section 6.
3. Extension to semi-supervised learning task. By applying the module to semi-supervised task, we prove that our module is a general solution for the co-training framework. Since the co-training framework has been applied to numerous task (domain adaptation [16], image classification [6], data segmentation [17], tag-based image search [18] and many more), one could apply our module to any tasks using the co-training framework and expect improvements.
4. Experiments with different settings are provided. Here, we also considered extreme noise rates (such as 80%) on ImageNet. By setting the noise rate to 80%, one expects an extremely difficult condition to learn from labels. Even under this stringent noise condition, our proposed method outperformed the other baseline by a large margin.

3. Related work

3.1. Co-training

Blum et al [1] successfully used a Co-Tr framework to solve the problem of web page classification. Co-Tr algorithm is to learn two (or more) models with distinction and diversity. It has been applied to a large variety of tasks ranging from domain adaptation [16], image classification [6], data segmentation [17], tag-based image search [18] and many more. Similar to the general framework of learning in Co-Tr, Learning to Teach [19] applies an inherent feedback sharing mechanism between a teacher and student network(s) to learn distinct features. On the other hand, model distillation algorithms [10,19] make use of an additional *mimicry* loss to align final class-specific posterior distribution of the student network. As an example, Zhang et al [20] employ the Kullback–Leibler divergence as a mimicry loss to match the probability estimates. In order to learn from unlabeled data, Chen et al [16] makes use of an auto-encoder to Co-Tr framework to reconstruct synthetic segmentation labels. All the applications of Co-Tr require a diversity between the homogeneous networks to function properly and this serves as the intuition and motivation behind our work. To this end, we propose and develop a method that directly targets and addresses the issue of diversity via an explicit discrepancy mechanism.

3.2. Discrepancy measurement

Our main objective in this work is to introduce discrepancy between the homogeneous networks. Hence, we need to measure the dissimilarity between two sets of samples, i.e., the generated features of homogeneous networks. Here, this is achieved by statistical measures. For the design and development of several algorithms of machine learning and computer vision, comparing and matching probability distributions between two different domains is a fundamental building block [21,22]. Several divergences such as Kullback–Leibler [23], Jensen-Shannon [24], *etc* have successfully been used in learning similar/dissimilar distributions across various domains. They are appreciated for their computational simplicity, but they suffer from the major shortcoming of not metrizising weak-convergence [25]. It has been shown earlier that MMD [26,27] is a highly nontrivial choice as well as the Optimal Transport (OT) distance which has been long known to be a powerful tool to compare probability distributions with non-overlapping supports. Both MMD and OT have the ability to metrizise weak-convergence, but they enjoy different characteristics. MMD can be efficiently and robustly estimated from a small number of samples of the measures, as a closed form solution exists for MMD. OT on the other hand, takes into account the underlying geometry while is computationally expensive. Thanks to [13], this cost is largely mitigated by settling for cheaper approximations obtained through strongly convex regularizers, in particular entropy. [13] introduce the Sinkhorn loss $\mathcal{S}_\epsilon$ with a smoothing parameter ϵ. When $\epsilon \to 0$, $\mathcal{S}_\epsilon$ is reduced to a pure Wasserstein distance, and conversely when $\epsilon = +\infty$, it leads to MMD.

3.3. Learning from noisy datasets and semi-supervised learning

Learning from a clean dataset is not considered as a difficult task any longer. Recently, there has observed a growing surge in the interest of studying the robustness of any machine learning algorithm against noisy labeled images and unlabeled images. Regard of learning from noisy labeled images, MentorNet [28] trains an additional StudentNet network to select clean labels which is in-turn used to further guide the main training process. If a clean and unbiased validation set is not available, MentorNet will discover new data-driven sample-weight schemes from data which can be updated according to feedback from StudentNet. Ren et al [29] follow a meta-learning paradigm and use a clean validation set to re-weight the training samples. Importance weights for training samples which result in the decrease of the loss in a clean validation set are increased, while the weights of those that result in the increase of the loss are decreased during the training process. One of the major drawbacks of [29] is the calculation of the clean validation set based importance weights after every gradient update of the network, which increases the time complexity of the overall algorithm. On the other hand, Decoupling [12] trains two different sub-networks with the examples that are confusing to both of them during the course of training.[1] Similarly, Co-Tr [5] trains two different networks with one selecting the clean examples, i.e. the examples with lower classification loss, for the other in an intertwined fashion. However, without any explicit discrepancy module between the networks that enforce the features learnt to be distinct and different, the solution learnt by the two aforementioned algorithm is not optimal. Motivation of semi-supervised learning is similar to learning from noisy labels. Regard of semi-supervised learning tasks, [30] presented the mutual-exclusivity loss and [31] presented the entropy minimization which are applied in one of the most famous semi-supervised learning methods – self-training technique.

[1] Both the networks produce different predictions with high confidence.

4. Preliminaries

Algorithm 1

Input $\{\boldsymbol{u}_i^{\mathcal{U}}\}_{i=1}^n$ and $\{\boldsymbol{v}_i^{\mathcal{V}}\}_{i=1}^m$ denote samples drawn from $\mathcal{U}$ and $\mathcal{V}$ szd1, cost function $c(x, y)$, for example, $c(x, y) = \|x - y\|^2$

Output Optimal transport plan π^*

Algorithm `Sinkhorn's Algorithm [13]`

Define $a_i \overset{\text{def}}{=} \frac{1}{\sum_{j=1}^m b_j e^{\frac{-c(x_i,y_j)}{\epsilon}} \beta_j}$ and $b_j \overset{\text{def}}{=} \frac{1}{\sum_{i=1}^n a_i e^{\frac{-c(x_i,y_j)}{\epsilon}} \alpha_i}$, cost matrix $\mathbf{K}$ where $\mathbf{K_{ij}} = e^{-\frac{c(s_i,y_j)}{\epsilon}}$

Define Consider optimal transport between two finite discrete measures $\alpha \overset{\text{def}}{=} \sum_{i=1}^n \alpha_i \sigma_{x_i}$ and $\beta \overset{\text{def}}{=} \sum_{j=1}^m \beta_i \sigma_{y_j}$, σ_{x_i} is the probability of x_i to be at point α_i, for iid samples, it would be 1/n and 1/m

for $l = 1, \ldots,$ **do**

$a^{l+1} = \frac{1}{\mathbf{K}(\mathbf{b}^{(l)} \odot \beta)}$ and $b^{l+1} = \frac{1}{\mathbf{K^T}(\mathbf{a}^{(l+1)} \odot \alpha)}$

$\odot$ represents Hadamard product.

Until a and b converge to (a^*, b^*) the exponential scaling of a solution of the dual problem.

end

The best transport plan:

$\pi^* = diag(a^* \odot \alpha)\mathbf{K}diag(b^* \odot \beta)$

Notation. All notations used are shown in Table 1. Throughout this paper, we use bold lower-case letters (*e.g*, $\boldsymbol{x}$) and bold upper-case letters (*e.g*, $\boldsymbol{X}$) to represent column vectors and matrices respectively. $[\boldsymbol{x}]_i$ denotes the ith element of the vector $\boldsymbol{x}$. $\mathbf{I}_n$ represents the $n \times n$ identity matrix. $\|\boldsymbol{X}\|_F = \sqrt{\text{Tr}(\boldsymbol{X}^\top \boldsymbol{X})}$ represents the Frobenius norm of the matrix $\boldsymbol{X}$, with Tr$(\cdot)$ indicating the trace of the matrix $\boldsymbol{X}$. $\boldsymbol{X}^\top$ denotes the transpose of $\boldsymbol{X}$. $\mathcal{P}(\mathcal{U})$ and $\mathcal{P}(\mathcal{V})$ represent the set of probability measures on two metric spaces $\mathcal{U}$ and $\mathcal{V}$. $\{\boldsymbol{u}_i^{\mathcal{U}}\}_{i=1}^n$ and $\{\boldsymbol{v}_i^{\mathcal{V}}\}_{i=1}^m$ denote n and m i.i.d. samples drawn from $\mathcal{U}$ and $\mathcal{V}$, respectively.

In this work, we use $\mathcal{S}_\epsilon$ [13] which can be used over general spaces $\mathcal{X}$, instead of only the Euclidean space $\mathbb{R}^d$ and not only the 1-Wasserstein distance. We first calculate OT between two probability distributions $\mathcal{U}$ and $\mathcal{V}$ (assume $\mathcal{U}$ and $\mathcal{V}$ are positive random measures of unit mass over a metric space $\mathcal{X}$) which is defined as the solution of the, possibly infinite dimensional, linear program:

$$\mathcal{OT}_c(\boldsymbol{u}, \boldsymbol{v}) = \min_{\pi \in \prod(\boldsymbol{u}, \boldsymbol{v})} \int_{\mathcal{X} \times \mathcal{X}} c(\boldsymbol{u}, \boldsymbol{v}) d\pi(\boldsymbol{u}, \boldsymbol{v}) \tag{1}$$

where $\Pi(\boldsymbol{u}, \boldsymbol{v})$ is the set of transportation plans (or couplings) of the joint probability distribution over the product space $\mathcal{U} \times \mathcal{V}$ with marginals $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. Here, $c(\boldsymbol{u}, \boldsymbol{v})$ is the cost function to move a unit of mass from $\mathcal{U}$ to $\mathcal{V}$. The cost function c needs to be defined for every pair $\boldsymbol{u}_i^{\mathcal{U}}, \boldsymbol{v}_j^{\mathcal{V}}$; and thus can be represented as a matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$. Therefore, the total cost incurred is $\langle \pi, \mathbf{C} \rangle = \sum_{ij} \pi_{ij} \mathbf{C}_{ij}$. When $\mathcal{X}$ is equipped with a distance $d_{\mathcal{X}}$, a typical choice is $c(\boldsymbol{u}, \boldsymbol{v}) = d_{\mathcal{X}}(\boldsymbol{u}, \boldsymbol{v})^p$ with $p > 0$ and thus we obtain the so-called p-Wasserstein distance between probability measures. However, because the above equation is not differentiable, we refer to the regularized optimal transport problem [32] defined by

$$\begin{aligned} \mathcal{W}_\epsilon(\mathbf{u}, \mathbf{v}) &= \min_{\pi \in \Pi(\mathbf{u}, \mathbf{v})} \langle \pi, \mathbf{C} \rangle + \epsilon \, \text{KL}(\pi | \mathbf{u} \otimes \mathbf{v}) \\ &= \min_{\pi \in \Pi(\mathbf{u}, \mathbf{v})} \int_{\mathcal{U} \times \mathcal{V}} \text{c}(\mathbf{u}, \mathbf{v}) \, d\pi(\mathbf{u}, \mathbf{v}) + \epsilon \, \text{KL}(\pi | \mathbf{u} \otimes \mathbf{v}) \ , \end{aligned} \tag{2}$$

where KL$(\pi | \mathbf{u} \otimes \mathbf{v})$ denotes the KL divergence between π and $\mathbf{u} \otimes \mathbf{v}$. $\mathbf{u} \otimes \mathbf{v}$ represents the product of the marginals (or the probability measures) $\mathbf{u}$ and $\mathbf{v}$. When $\text{c}(\mathbf{u}, \mathbf{v}) = D^p(\mathbf{u}, \mathbf{v})$; we obtain the entropy regularized p-Wasserstein distance as shown below:

$$\mathcal{W}_\epsilon^p(\mathbf{u}, \mathbf{v}) = \left(\min_{\pi \in \Pi(\mathbf{u}, \mathbf{v})} \int_{\mathcal{U} \times \mathcal{V}} \text{D}^p(\mathbf{u}, \mathbf{v}) \, d\pi(\mathbf{u}, \mathbf{v}) \right)^{1/\text{p}} + \epsilon \, \text{KL}(\pi | \mathbf{u} \otimes \mathbf{v}) \tag{3}$$

Similar to [13], the *Sinkhorn* loss is defined as Eq. (4) and algorithm of finding best transport plan π^* is given in Algorithm 1.

$$\mathcal{S}_\epsilon(\mathbf{u}, \mathbf{v}) = \mathcal{W}_\epsilon(\mathbf{u}, \mathbf{v}) - \frac{1}{2} \left(\mathcal{W}_\epsilon(\mathbf{u}, \mathbf{u}) + \mathcal{W}_\epsilon(\mathbf{v}, \mathbf{v}) \right) \tag{4}$$

In this paper, we set $p = 2$. $\mathcal{S}_\epsilon$ has the following behavior in ϵ:

1. as $\epsilon \to 0, \mathcal{S}_\epsilon(\mathbf{u}, \mathbf{v}) \to 2\mathcal{W}_\epsilon(\mathbf{u}, \mathbf{v})_c$;
2. as $\epsilon \to \infty, \mathcal{S}_\epsilon(\mathbf{u}, \mathbf{v}) \to \text{MM}\mathcal{D}_{-c}(\mathbf{u}, \mathbf{v})$, where MM$\mathcal{D}_{-c}$ is the Maximum Mean Distance whose kernel is the cost function.

4.1. Maximum mean distance

An empirical estimate of MMD between $\mathcal{U}$ and $\mathcal{V}$ is obtained as

$$\text{MM}\mathcal{D}(\mathcal{U}, \mathcal{V}) = \left\| \frac{1}{n} \sum_{i=1}^n \Phi(\mathbf{u}_i) - \frac{1}{m} \sum_{j=1}^m \Phi(\mathbf{v}_j) \right\|_H^2 \tag{5}$$

Here, H denotes the induced Reproducing Kernel Hilbert space [33]) and $\|\cdot\|_H$ denotes its norm. MM$\mathcal{D}(\mathcal{U}, \mathcal{V})$ is a measure of overlap between $\mathcal{U}$ and $\mathcal{V}$ such that increase (or decrease) in overlap results in decrease (or increase) in MM$\mathcal{D}(\mathcal{U}, \mathcal{V})$. $\Phi(\mathbf{p})$ represents a functional mapping of the input $\mathbf{p}$ to a high-dimensional space. By the kernel trick, the form in Eq. (5) can be written as;

$$\mathrm{MM}\mathcal{D}(\mathcal{U},\mathcal{V}) = \frac{1}{n^2}\sum_{i}^{n}\sum_{i'}^{n} k(\mathbf{u}_i,\mathbf{u}_{i'}) - \frac{1}{nm}\sum_{i}^{n}\sum_{j}^{m} k(\mathbf{u}_i,\mathbf{v}_j) + \frac{1}{m^2}\sum_{j}^{m}\sum_{j'}^{m} k\left(\mathbf{u}_j,\mathbf{v}_{j'}\right) \tag{6}$$

In all our experiments, we have employed the Gaussian kernel

$$k(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{\sigma}\right) . \tag{7}$$

5. Methodology

Here, we present our proposed methodology, i.e., Discrepant Collaborative Training (DCT). We formulate DCT as a cohort of two homogeneous networks f and g with their learnable parameters represented as θ and $\widehat{\theta}$ respectively (see Fig. 1 for more details.). Our total loss is contributed by three parts which are: L_M, L_D and L_C. L_M is the basic loss function for different tasks (for example, cross entropy loss for image classification task). L_D is discrepancy loss to increase diversity. L_C is the consistency loss which keeps consistency between networks.

We now define our total loss function as follows:

$$\mathrm{Loss}_f = \mathrm{L}_M^f - \lambda_D \mathrm{L}_D + \lambda_C \mathrm{L}_C \tag{8}$$

$$\mathrm{Loss}_g = \mathrm{L}_M^g - \lambda_D \mathrm{L}_D + \lambda_C \mathrm{L}_C \tag{9}$$

λ_D and λ_C are the combination weights for the diversity and the consistency loss (i.e. L_D and L_C) respectively. Eqs. (8) and (9) are used to update f and g using stochastic gradient decent optimizer.

The first part of the loss function, L_M, is different for each specific network. For example, if we are considering a supervised learning task, L_M will be a conventional supervised loss that trains the network to predict the correct labels for the training instances. To increase the diversity between f and g, we apply our $\mathcal{S}_\in$ in the middle of our two networks (see Fig.1) which forms the second part of the total loss which we call discrepancy loss.

$$\mathrm{L}_D = \mathcal{S}_\in(\mathbf{A}_i, \mathbf{B}_i) \tag{10}$$

where $\mathbf{A}_i = f_{\theta(1:l)}(\mathbf{X}_i)$ and $\mathbf{B}_i = g_{\widehat{\theta}(1:l)}(\mathbf{X}_i)$ for the i^{th} input $\mathbf{X}_i$, l denotes the layer where the discrepant $\mathcal{S}_\in$ module is inserted, and $\theta(1:l)$ and $\widehat{\theta}(1:l)$ represent the parameters of the two networks f and g up until layer l. It is to be noted that L_D is calculated irrespective of the presence or absence of noisy labels within the mini-batch of images. Even though we want both f and g to learn diverse and distinct features, the final class probability distribution learnt by f and g should not be very different from each other. Thus, we use another $S_\in$ to explicitly reduce the discrepancy between $\mathbf{z}_i^f$ and $\mathbf{z}_i^g$ for every $\mathbf{X}_i$ as shown below:

$$\mathrm{L}_C = \mathcal{S}_\in\left(\mathbf{z}_i^f, \mathbf{z}_i^g\right), \tag{11}$$

where $\mathbf{z}_i^f = f_\theta(\mathbf{X}_i)$ and $\mathbf{z}_i^g = g_{\widehat{\theta}}(\mathbf{X}_i)$.[2]

5.1. Extension to learning from noisy labels

5.1.1. Settings

We refer to the settings of [5] for the task of learning from noisy labels. The input for f and g is always the same mini-batch of training/testing set. Labels of all samples in mini-batch are randomly corrupted based on noise rate. The outputs of f and g for each sample are predictions of the class (each network will predict one most possible class where the sample is from). During the test step, we gather the outputs of f and g for each sample. For example, if the dataset is CIFAR10, final outputs of f and g are both $1 * 1$ for each test image. We select the larger value as the final prediction of the test image.

Now, we show how we introduce our DCT design to the Co-Tr framework to deal with noisy labels Fig.2. As elaborated in the previous section, we employ the total loss defined in Eqs. (8) and (9). Similar to [5,27,28], networks will select possible clean examples based on classification loss. More specifically, classification loss for $\mathbf{X}_i$ for f and g are shown below:

$$\mathrm{L}_f(\mathbf{X}_i) = -\log\left(\frac{\exp\left(\mathbf{z}_i^f\right)}{\sum_1^m \exp\left(\mathbf{z}_j^f\right)}\right) \qquad \mathrm{L}_g(\mathbf{X}_i) = -\log\left(\frac{\exp\left(\mathbf{z}_i^g\right)}{\sum_1^m \exp\left(\mathbf{z}_j^g\right)}\right) , \tag{12}$$

where $\mathbf{z}_i^f = f_\theta(\mathbf{X}_i)$ and $\mathbf{z}_i^g = g_{\widehat{\theta}}(\mathbf{X}_i)$.[3] Similar to [5], within each mini-batch examples which produce the $\mathbf{R}$ lowest L_g and L_f will be selected for f and g separately. The loss to update θ_f is given as

$$\mathrm{L}_M^f = \sum_{i=1}^{R} \mathrm{L}_f(\mathbf{X}_i) \quad \forall \mathbf{X}_i \in \mathcal{D}_g \quad , \tag{13}$$

where $\mathcal{D}_g$ represents a set of images that results in the R lowest L_g calculated in Eq. (12). Similarly, we calculate the loss to update θ_g as

$$\mathrm{L}_M^g = \sum_{i=1}^{R} \mathrm{L}_g(\mathbf{X}_i) \quad \forall \mathbf{X}_i \in \mathcal{D}_f \quad , \tag{14}$$

where $\mathcal{D}_f$ represents a set of images that results in the R lowest L_f as calculated in Eq. (12). L_D and L_C are the same as discussed in the previous section. Algorithm 2 provides the pseudo code of our proposed DCT algorithm for the noisy label task.

5.2. Extension to semi-supervised learning

5.2.1. Settings

We choose [6] as our baseline for the task of semi-supervised learning. Here, we denote $\mathcal{D} = \mathcal{S} \cup \mathcal{U}$ as our dataset where images in $\mathcal{S}$ are labeled and images in $\mathcal{S}$ are not. For the training of semi-supervised task, both networks have access to $\mathcal{S}$ and $\mathcal{U}$. We feed two networks with same mini-batch which contains both labeled and unlabelled samples. The network is trained to make similar predictions on training samples (no matter labeled or not) and make different predictions on adversarial samples. We simply add the divergence modules to the co-training framework to force the networks to extract different features but similar softmax outputs on training samples.

We also define f_{fc} and g_{fc} as the final fully-connected layer that classify the softmax outputs $\mathbf{z_f}$ and $\mathbf{z_g}$ to one of the categories in $\mathcal{S}$. L_1 is the semi supervised learning loss [6] defined as:

$$\mathrm{L}_\mathrm{M}^\mathrm{f} = \mathrm{L}_\mathrm{M}^\mathrm{g} = \mathrm{L}_\mathrm{sup} + \mathrm{L}_\mathrm{cot} + \mathrm{L}_\mathrm{diff} \tag{15}$$

For the above equation, we define:

$$\mathrm{L}_\mathrm{sup}(\mathbf{X}, y) = \mathrm{H}\left(y,\ f_{fc}\left(p_f(\mathbf{X})\right)\right) + \mathrm{H}\left(y, g_{fc}\left(p_g(\mathbf{X})\right)\right) \tag{16}$$

$p_f(\mathbf{x}) = \mathbf{z_f}$ and $p_g(\mathbf{x}) = \mathbf{z_g}$ (see Fig.1), for any data $(\mathbf{X}, y)$ in $\mathcal{S}$ where y is the label for $\mathbf{X}$ and $\mathrm{H}(\mathbf{m},\mathbf{n})$ is the cross entropy between distribution $\mathbf{m}$ and $\mathbf{n}$.

[2] Usually it is the output of the softmax layer for each of f and g.

[3] Usually it is the output of the softmax layer for each of f and g.

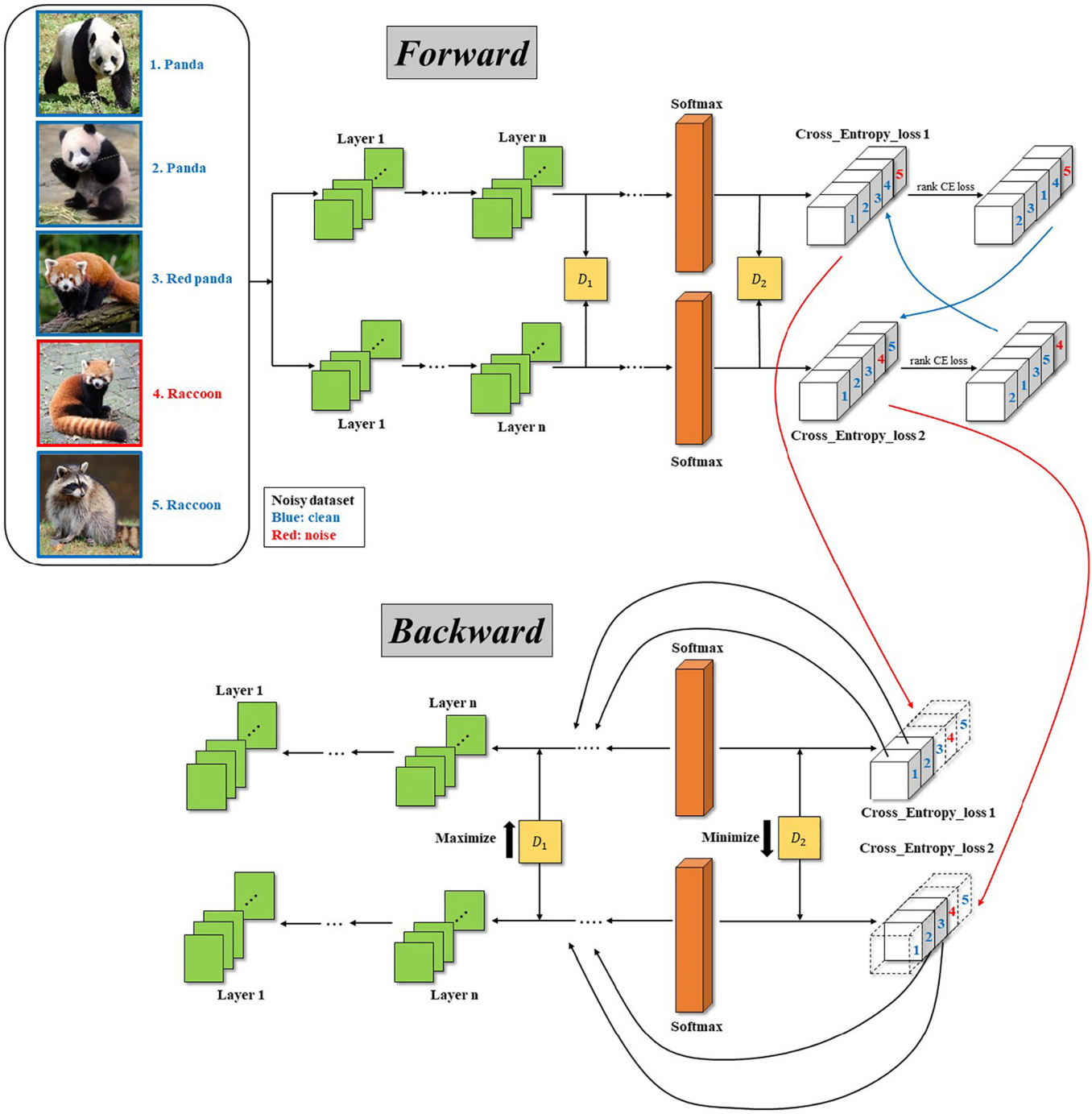

Fig. 2. Forward (top) and backward (bottom) propagation of Discrepant Collaborative Training. [FORWARD] Five images are fed into sub-networks independently, four (blue) are correctly labeled and one (red) is corrupted with noise. The first discrepancy module is placed after *Layer n* between two networks. The second discrepancy module is placed after *softmax* layer. Then, the five images will be ranked according to its own cross entropy loss calculated by each network. Then the two networks exchange information of the ranking. [BACKWARD] According to the ranking information provided by its"peer" network, each network chooses only a few images with smaller loss value to update itself. D_1 and D_2 are Sinkhorn divergence modules. We maximize the diversity of the first discrepancy module to learn diverse features in each of the network, while the diversity of the second module is minimized so as to learn the same class distributions. [27].

$$\mathrm{L}_{\mathrm{cot}}(x) = \mathrm{H}\left(\frac{1}{2}\left(p_f(\mathbf{X}) + p_g(\mathbf{X})\right)\right) - \frac{1}{2}\mathrm{H}\left(p_f(\mathbf{X}) + p_g(\mathbf{X})\right) \tag{17}$$

where $x \in \mathcal{U}$ and $\mathrm{H}(\mathbf{m})$ is the entropy of $\mathbf{m}$. $\mathrm{L}_{\mathrm{cot}}$ is the Jensen-Shannon divergence between $f_{fc}(p_f(\mathbf{X}))$ and $g_{fc}(p_g(\mathbf{X}))$.

We then generate a set of adversarial examples [22] $\mathcal{D}' = \{\mathrm{h}(\mathbf{X}) | \mathbf{X} \in \mathcal{D}\}$. We employ several constraints for the adversarial examples: (i) $f_{fc}(p_f(\mathrm{h}(\mathbf{X}))) \neq g_{fc}(p_g(\mathrm{h}(\mathbf{X})))$, (ii) $p_f(\mathrm{h}(\mathbf{X})) = p_f(\mathbf{X})$ but $p_g(\mathrm{h}(\mathbf{X})) \neq p_g(\mathbf{X})$, which implies that $\mathrm{h}(\mathbf{X})$ is an adversarial example of g that fools network g but not network h (and vice versa), and (iii) $\mathrm{h}(\mathbf{X}) - \mathbf{X}$ is small. $\mathrm{L}_{\mathrm{diff}}$ is finally defined as:

$$\mathrm{L}_{\mathrm{diff}}(x) = \mathrm{H}\left(p_f(\mathbf{X}), p_g(\mathrm{h}(\mathbf{X}))\right) - \mathrm{H}\left(p_f(\mathrm{h}(\mathbf{X})), p_g(\mathbf{X})\right) \tag{18}$$

Similarly, L_D and L_C will be the same discussed in the previous section. Algorithm provides the pseudo code of our proposed DCT algorithm for the task of semi-supervised learning.

Algorithm 2 Discrepant Collaborative Training on Learning from Noisy labels referred to [5]

Data: θ and $\hat{\theta}$ (parameters of network f and g), learning rate η, epoch number T, l the layer number for $\mathcal{S}_\epsilon$.

Algorithm `DCT with noisy labels`

for $t = 1, \ldots, T$ **do**

  Shuffle training set $\mathcal{D}$;

  Fetch mini-batch $\bar{\mathcal{D}}$ from $\mathcal{D}$;

  Obtain $\mathcal{D}_f = \arg\min_{\bar{\mathcal{D}}} l(f, \bar{\mathcal{D}})$; sample R(T) instances with small cross entropy loss

  Obtain $\mathcal{D}_g = \arg\min_{\bar{\mathcal{D}}} l(g, \bar{\mathcal{D}})$; sample R(T) instances with small cross entropy loss

  Update $\theta = \theta - \eta \Big[\lambda_M \nabla_\theta \mathrm{L}_{\mathrm{M}}^{\mathrm{f}}(\theta, \mathcal{D}_g) - \lambda_D \nabla_\theta \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_\theta \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;

  Update $\hat{\theta} = \hat{\theta} - \eta \Big[\lambda_M \nabla_{\hat{\theta}} \mathrm{L}_{\mathrm{M}}^{\mathrm{g}}(\hat{\theta}, \mathcal{D}_f) - \lambda_D \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;

end

Algorithm 3 Discrepant Collaborative Training on Semi-Supervised Learning [6]

Data: θ and $\hat{\theta}$ (parameters of network f and g), learning rate η, epoch number T, l the layer number for $\mathcal{S}_\epsilon$.

Algorithm `DCT for semi-supervised learning`

for $t = 1, \ldots, T$ **do**

  Shuffle training set $\mathcal{D}$;

  Fetch mini-batch $\bar{\mathcal{S}}$ from $\mathcal{S}$;mini-batch $\bar{\mathcal{U}}$ from $\mathcal{U}$;$\bar{\mathcal{D}} = \bar{\mathcal{S}} \bigcup \bar{\mathcal{U}}$;

  Create Adversarial Examples Compute the adversarial examples $h_1(x)$ and $h_2(x)$ for $x \in \bar{\mathcal{D}}$ using FGSM [34]

  Calculate $\mathrm{L_M}$ $\mathrm{L_M^f} = \mathrm{L_M^g}$ (see Eq.15);

  Update $\theta = \theta - \eta\Big[\lambda_M \nabla_\theta \mathrm{L}_{\mathrm{M}}^{\mathrm{f}}(\theta, \bar{D}) - \lambda_D \nabla_\theta \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_\theta \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;

  Update $\hat{\theta} = \hat{\theta} - \eta\Big[\lambda_M \nabla_{\hat{\theta}} \mathrm{L}_{\mathrm{M}}^{\mathrm{g}}(\hat{\theta}, \bar{D}) - \lambda_D \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;

end

Table 2
Network Structure trained by MNIST, CIFAR10, CIFAR100 and SVHN. The slope of all LReLU functions in the network are set to 0.01. Number of training classes for the dataset is *K*.

#Layer	Input image
1	3×3 conv, 128 LReLU
2	3×3 conv, 128 LReLU
3	3×3 conv, 128 LReLU
	2×2 max-pool, stride 2
	dropout, $p = 0.25$
4	3×3 conv, 256 LReLU
5	3×3 conv, 256 LReLU
6	3×3 conv, 256 LReLU
	2×2 max-pool, stride 2
	dropout, $p = 0.25$
7	3×3 conv, 512 LReLU
8	3×3 conv, 256 LReLU
9	3×3 conv, 128 LReLU
	avg-pool
10	fc-layer $128 \rightarrow K$
	softmax

Table 3
Details of the various datasets used in the evaluation of *DCT*.

Dataset	# of train images	# of test images	# of class	Image size
MNIST	60,000	10,000	10	28×28
CIFAR10	50,000	10,000	10	32×32
CIFAR100	50,000	10,000	100	32×32
SVHN	73,257	26,032	10	32×32
CUB200-2011	5864	5924	200	227×227
CARS196	8054	8131	196	227×227

Table 5
Comparison between our proposed DCT algorithm with baseline algorithms for CUB200-2011 and CARS196 in terms of accuracy on the test set (%). Co-T stands for Co-teaching [5].

Noise	Dataset	Cross Entropy	Co-T [5]	DCT ($\epsilon = +\infty$)	DCT ($\epsilon = 0$)
Symmetric-50%	CUB200–2011	40.80	54.64	59.24	60.56
Symmetric-20%	CUB200–2011	63.78	72.34	74.57	75.44
Symmetric-50%	CARS196	38.86	66.75	67.80	69.15
Symmetric-20%	CARS196	71.76	86.00	86.62	87.51

Table 6
Comparison between our proposed DCT algorithm with popular baseline algorithms. We report the average accuracy (%) from 5 runs using DCT. CIFAR100+ means it is trained using data augmentation, otherwise not.

Dataset	GAN [30]	S-T [30]	π Model [40]	T-E [40]	D-C [6]	DCT ($\epsilon = +\infty$)	DCT ($\epsilon = 0$)
CIFAR10	81.37	88.71	87.64	87.84	90.97	91.45	91.89
CIFAR100	–	–	56.57	–	61.23	61.77	61.98
CIFAR100+	–	–	–	–	65.37	65.84	66.16
SVHN	91.89	–	95.18	95.58	96.39	96.52	96.78

S-T stands for Stochastic Transformations. T-E stands for Temporal Ensembling. D-C stands for Deep Co-Training. "–" means the original papers did not report the corresponding accuracy.

Table 7
Comparison of our proposed CCT against two different baselines on Clothing1M dataset: Co-Teaching [45] and JoCoR [46]. Accuracy(%) is reported as the average of 5 runs.

Method	Co-Teaching	JoCoR	DCT(OURS)
Accuracy	68.5	69.8	**70.1**

The best results are shown in bold.

6. Experiments

6.1. Dataset

We verify the effectiveness of our approach on six benchmark datasets: MNIST [35], CIFAR10 [36], CIFAR100 [36], SVHN [37], CUB200–2011 [38], CARS196 [39], Clothing1M [15] on different tasks. Table 3 provides more details regarding the datasets.

6.2. Setup

Table 2 shows the CNN architecture used for experiments on MNIST, CIFAR10, CIFAR100 and SVHN. For experiments in the weakly-supervised learning setup, we follow the well-acknowledged standard settings of "Temporal Ensembling" [40] and "Virtual Adversarial Training" [41]. Further, we use Adam optimizer with details of hyper-parameters such as learning rate and weight decay are provided in supplementary material. We choose [5] as our baseline for the Noisy Label task and we follow their detailed settings. For the task of Semi-Supervised learning, we follow the settings of [6]. For experiments on two fine-grained image recognition datasets (i.e. CUB200–2011 and CARS196), we use the Inception-V1 [42] architecture, pretrained on Imagenet [43]. Here, we use RMSProp and the Adam optimizer for the CUB200–2011 [38] and CARS196 [39] datasets respectively. The initial learning rate for both optimizers is set to 0.0001.

Note: We report the optimal value of the hyper-parameters λ_2 and λ_3, batch size, and number of epochs used in DCT for all the datasets in the supplementary material. The method trains two networks simultaneously and out of the two predictions, we select the prediction that has the higher confidence.

Table 4
Comparison between our proposed DCT algorithm with popular baseline algorithms. We report the average accuracy in (%) from 5 runs of DCT.

Noise	Dataset	F-C [44]	Decoupling [12]	M-Net [28]	Co-T [5]	DCT ($\epsilon = +\infty$)	DCT ($\epsilon = 0$)
Pairflip-45%	MNIST	0.24	58.03	80.88	87.63	88.54	**90.85**
Symmetric-50%	MNIST	79.61	81.15	90.05	91.32	94.21	**95.85**
Symmetric-20%	MNIST	98.82	95.70	96.70	97.25	98.54	**99.07**
Pairflip-45%	CIFAR10	6.61	48.80	58.14	72.62	72.91	**74.34**
Symmetric-50%	CIFAR10	59.83	51.49	71.10	74.02	78.50	**80.4**
Symmetric-20%	CIFAR10	84.55	80.44	80.76	82.32	85.41	**87.2**
Pairflip-45%	CIFAR100	1.60	26.05	31.60	34.811	35.33	**36.00**
Symmetric-50%	CIFAR100	41.04	25.80	39.00	41.37	42.11	**43.4**
Symmetric-20%	CIFAR100	**61.87**	**44.52**	**52.13**	**54.23**	**56.11**	**57.8**

F-C stands for F-correction, M-Net stands for MentorNet, and Co-T stands for Co-Teaching.
The best results are shown in bold.

Table 8
Comparison of our proposed CCT against two different baselines on Mini-Imagenet: Co-Teaching [45] and JoCoR [46]. Accuracy(%) is reported as the average of 5 runs.

Noise rate	20%	40%	80%
Co-Teaching	54.1	45.3	37.0
JoCoR	57.2	49.1	40.4
DCT(OURS)	58.4	50.8	42.1

Table 9
The accuracy (%) of $\mathcal{S}_{\epsilon=0}$ (i.e. 2-Wasserstein Distance) for various setups (using CUB200–2011 dataset contaminated by symmetric noise with rate of 50%.)

λ_2	λ_3	DCT(%)	Descriptions
0	0	54.64	Co-teaching [5]
0.001	0	57.44	Diversity loss only
0	0.001	56.52	Consistency loss only
0.001	0.001	60.56	Optimal Case
0.001	0.005	58.75	–
0.001	0.0005	58.25	–
0.005	0.001	58.22	–
0.0005	0.001	59.45	–

Table 10
Study of the importance of ϵ for $\mathcal{S}_\epsilon$ for CUB200-2011 dataset in the presence of 50% symmetric noisy labels. The average accuracy (%) is reported after 5 different runs of DCT. WD stands for Wasserstein Distance. We fix $\lambda_2 = \lambda_3 = 0.001$.

Condition	$\epsilon = 0$ (2-WD)	$\epsilon = 1$	$\epsilon = 5$	$\epsilon = 1000$	$\epsilon = \infty$ (MMD)	Co-Teaching [5]
Accuracy	60.56	60.25	59.85	59.41	59.24	54.64

6.3. Learning from noisy labels

6.3.1. Noise transition

Here we test our proposed algorithm on the task of noisy-supervised image classification. More specifically and following the protocols of [5, 44], we manually corrupt all the clean datasets by a noise transition matrix $\mathbf{Q} \in \mathbb{R}^{K\times K}$, where $\mathbf{Q}_{ij} = Pr(\widetilde{\mathbf{y}} = j|\mathbf{y} = i)$ gives the probability that noisy label (i.e. $\widetilde{y}$) is flipped to a value j from clean label (i.e. y) value of i. In this paper, we set up two different noise transition situations: (1) *Pair* flipping, and (2) *Symmetric* flipping. Exemplar noise transition matrix for each of the pair and symmetric flipping is shown in Fig. 3 (a) and (b), where the value of noise rate ζ is set 45% and 50% respectively.

In our experiments, we run and test our proposed DCT in three different noise conditions, which are specified as follows: (1) 45% pair flip noise; (2) 50% symmetric flip noise; (3) 20% symmetric flip noise. We choose high noise rate values, i.e., 45% and 50%, as this paper mainly focuses on the robustness of DCT and the value of working with noisy data. However one should definitely set the value ϵ lower than 50% for pair flip noise, otherwise the neural networks will need additional information to learn discriminative features. We further test on 20% symmetric flip noise so as to verify the robustness of our proposed DCT algorithm under lower noise conditions.

6.3.2. Baseline algorithms

For MNIST, CIFAR10 and CIFAR100, we compare our results against the following baseline methods: (i) F-correction [44], which makes use of a noise transition matrix to correct the softmax predictions; (ii) Decoupling [12], which select those samples where the underlying networks are highly non-confident of their predictions and use them to update the parameters; (iii) MentorNet [28], where noisy instances for the student networks are filtered out by a pre-trained additional teacher network to learn robustly under noisy labels. (iv) Co-Teaching [5], which trains two identical homogeneous networks where one selects the best possible clean instances for the other and vice-versa. Results of the above baselines are reported from [5]. Furthermore, we consider two baseline models trained with (a) conventional cross-entropy loss and (b) Co-Teaching algorithms so as to verify the robust performance of our proposed DCT method on CUB200–2011 and CARS196 datasets.

6.4. Results analysis and discussion

Here, we provide a detailed insight regarding the performance of DCT and compare it against the several baseline algorithms mentioned before across the different datasets for different settings of noise.

6.4.1. MNIST

As observed in Table 4, all baseline methods obtain competitive results against each other in the case of a low noise rate (i.e. symmetric 20%). Our DCT method achieve a competitive 99.07% against the second best F-correction method (which achieves 98.82% accuracy on the test set). Further one can observe a drop in performance when the noise rate is increased to 50% for the F-correction and the Decoupling baseline methods. However both MentorNet and Co-Teaching are quite robust when dealing with large values of the noise rate. Further our proposed DCT algorithm obtains the state-of-the-art accuracy of 95.85%, thereby outperforming all the baselines and the current best algorithm (i.e. Co-Teaching) by 4.53%. A further increase in the complexity of the noise (i.e. pair-flip noise with $\zeta = 45\%$) results in a considerable drop in the classification accuracy for both F-correction and Decoupling methods. On the other hand, we again observe that DCT outperforms all the baselines by a significant margin. More specifically, we outperform MentorNet by 9.97% in terms of accuracy on the test set. Moreover, we notice that when $\epsilon = 0$ ($\mathcal{S}_\epsilon$ becomes the Wasserstein Distance) the performance is improved somewhat compared to when $\epsilon = +\infty$ (the $\mathcal{S}_\epsilon$ loss becomes MMD). The same conclusion can be drawn from the experiments on the other datasets as well.

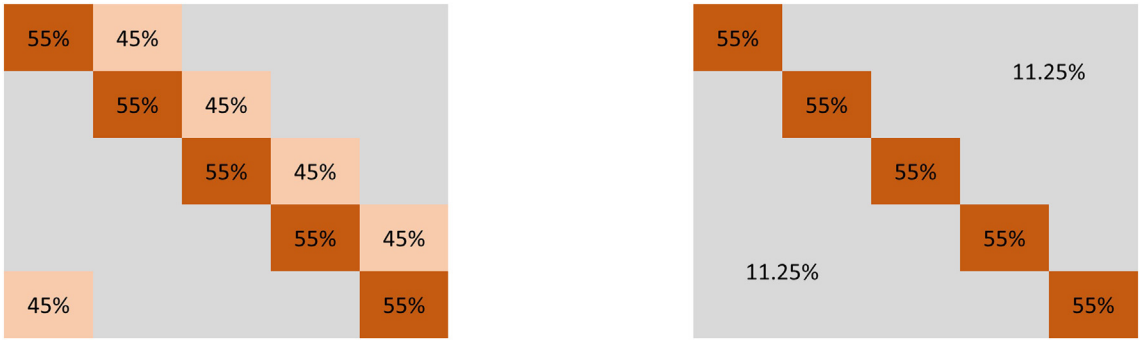

Fig. 3. Noise transition matrices [5] in the example of 5 classes.

6.4.2. CIFAR10

From Table 4, one can observe that similar results (in terms of classification accuracy) is obtained by all the baseline methods for symmetric flip noise with 20%. Unlike for the MNIST dataset, on CIFAR10, DCT outperforms all the baselines including F-correction by 2.65%. With further increase in the level and the complexity of the noise, it is easily seen that DCT outperforms all the baseline algorithms, thus further validating the design choices incorporated within DCT to learn distinct and discriminative features.

6.4.3. CIFAR100

It is observed from Table 4 that for symmetric noise with 20% noise rate, DCT outperforms all the competitive baseline algorithms apart from F-correction. It is evident that F-correction is indeed a reliable method below a certain moderate level of noise corruption. However as the complexity of noise in increased, F-correction is no longer able to cope. On the other hand in complex noise settings, DCT is significantly more robust in comparison to the baseline methods, thereby reinforcing the choice of our algorithmic design.

6.4.4. CUB200-2011

The results are reported in Table 5. In the 20% symmetric noise setting, the baseline model trained with the conventional *cross-entropy* classification loss achieves a test accuracy of 63.78%. Co-Teaching algorithm outperforms this baseline by 8.56%. Moreover, DCT outperforms all the baseline methods and achieves an accuracy of 74.57% and 75.44% with different values of noise rate for the same noise setting. A decrease in the classification accuracy is observed when the noise rate is increased to 50%. Nonetheless, DCT still outperforms the rest by a significant margin. These results clearly demonstrate the effectiveness and robustness of DCT for large scale fine-grained image recognition tasks against different value of noise corruption percentages.

6.4.5. CARS196

It is observed in Table 5 that our proposed DCT algorithm outperforms both cross-entropy baselines by a significant margin in terms of accuracy on the test set for different values of 20% and 50%. However unlike the CUBS200-2011 dataset, the performance gain observed over Co-Teaching is not substantial for the CARS196 dataset. One plausible justification that can be attributed to this observation is that CARS196 is a difficult dataset to train on in comparison to CUBS200-2011.

In summary, according to the results shown in Tables 4 and 5, we have successfully verified and demonstrated the effectiveness and robustness of our proposed DCT method, irrespective of the kind and the level of noise present in the datasets.

Note: Fine-grained image recognition datasets usually consist of labeled images which have low (and high) intra (and inter) class variances, thereby making it difficult to train models on such datasets as they are more vulnerable to noise. From the results obtained in Table 5, it is observed that both Co-Teaching and DCT significantly outperforms the cross-entropy baseline in terms of classification accuracy on the test set. This observation clearly demonstrates the need for two (or more) feature extractors in order to learn distinct and discriminative features from a fine-grained dataset in the presence of noisy labels.

6.4.6. Clothing1M

We use the noise corrupted version of the Clothing1M[4] to infer the noise rate by the clean validation set. The batch size is set to 64. Further, we report the classification accuracy after training for 51 epochs for all methods. As shown in Table 7, our DCT performs better than Co-Teaching [45] and JoCoR [46]; thus demonstrating the effectiveness of DCT over Co-Teaching and JoCoR for a large dataset.

[4] We have already used the noise corrupted version of the Clothing1M dataset in our experiments.

6.4.7. ImageNet

Jiang et al. [47] proposed to use mini-ImageNet and Stanford Cars to introduce both web and symmetric indistribution noise in a controlled manner with different noise ratios. We adopt the mini-ImageNet web noise dataset for evaluation in a real scenario with several ratios, which consists of 100 classes with 50 K (5 K) samples for training (validation). For fair comparison, we use a standard ResNet-18 [48] and follow the work of [47,49].

Table 8 illustrates the superior performance of our DCT when training in the presence of web label noise in mini-ImageNet [14]. The results demonstrate that DCT are robust to web noise and the improvements are consistenct across noise levels.

6.5. Semi-supervised learning

We also investigate the performance of our algorithm in the semi-supervised setting, in particular, on the SVHN, CIFAR10 and CIFAR100 datasets. Following [40], for SVHN, we only use 1,000 images out of the available 73,257 training images as the supervised set $\mathcal{S}$, the remaining 73,257 − 1,000 images goes in to the unsupervised set $\mathcal{U}$. For CIFAR10, we only use 4,000 images out of the 500,000 available training images in the supervised set $\mathcal{S}$ and the remaining 46,000 images form the unsupervised set $\mathcal{U}$. For CIFAR100, we use 10,000 images out of the 50,000 available training images in the supervised set $\mathcal{S}$ and the remaining 40,000 images make up the unsupervised set $\mathcal{U}$. We use the full set of test images for evaluation across all datasets. For the baseline, we follow the work by [6]. We also compare our design with: GAN [50], Stochastic Transformations [30], ∏ Model [40], Temporal Ensembling [40], Mean Teacher [51] and Deep Co-Training 2-views D-C, [6]. To support a fair comparison, we only use D-C-2-views as more views increase the number of network parameters.

6.5.1. Baselines

Baseline methods are listed in Table 6. All of the baselines methods, as well as ours, require the evaluation of multiple models, meaning that even though DCT has two homogeneous networks, it is not at a computational disadvantage.

6.5.2. Discussion of results

From Table 6 it is observed that for CIFAR10, all baseline methods reach 80% accuracy on the test set. Our DCT outperforms other methods and achieves 91.89% which beats previous state-of-the-art, Deep-Co-Training [6], by 0.92%. The same conclusion can be drawn for CIFAR100, CIFAR100 + and SVHN where our DCT outperforms previous state-of-the-art by 0.39 − 0.79%. Also, $\mathcal{S}_{\epsilon=0}$ works better than $\mathrm{S}_{\epsilon=+\infty}$, which is consistent with our results on the tasks using noisy labels. According to the results in Table 6, Deep Co-Training [6] is indeed a very good method for semi-supervised learning. Whenever we are injecting $\mathcal{S}_{\epsilon}$ into the Deep Co-Training framework, we see improvements. Moreover, these improvements do not come at the expense of adding more parameters to the network and, thanks to [13], the computational burden is affordable. Details of running times and ablation studies are provided in the next section.

7. Further discussion and analysis

In this section, we provide details of our implementation and study the robustness of our DCT design with respect to its parameters, namely λ_2 and λ_3 (see Eq. (19), below). We will first report the value of the hyper parameters used in our experiments. Then we will analyze and investigate the effect of the hyper-parameters over the performance of the proposed DCT algorithm.

7.1. Hyper-parameters

- For all experiments on MNIST, CIFAR10, CIFAR100 and SVHN, we set initial learning rate and weight decay as 0.001 and 0 respectively. The models are trained for 600 epochs (with a batch of size 128 at each iteration).
- For CUBS200-2011 and CARS196, we use RMSProp and Adam optimizer respectively. The initial learning rate for both optimizers is set to 0.0001. The models are trained for 80 and the batch size is set to 32.

7.2. Robustness of the DCT algorithm

We recall that the loss of the DCT algorithm as:

$$\text{Loss} = L_1 - \lambda_2 L_2 + \lambda_3 L_3. \tag{19}$$

Here, L_1 is the classification loss (for each network), L_2 and L_3 are diversity and consistency losses, respectively. Furthermore, λ_2 and λ_3 denote the associated weights of L_2 and L_3 respectively. Here, we analyze the robustness of the DCT algorithm with respect to its parameters. By doing so, we evaluate the performance of the DCT algorithm on the CUB200–2011 dataset by varying the values of λ_2 and λ_3. Table 9 shows the accuracy of the $\mathcal{S}_{\in=0}$ (i.e., 2-Wasserstein Distance) for the 50% symmetric noise (which is a challenging setup) for various values of λ_2 and λ_3. Some of the observations are as follows:

When $\lambda_2 = \lambda_3 = 0$, we recover the vanilla Co-Tr framework with an accuracy of 54.64%. Setting $\lambda_3 = 0.0001$ and $\lambda_2 = 0$ results in adding only the consistency loss and a modest increase in the performance (1.88% to be exact) is observed. Interestingly, by just adding the diversity loss ($\lambda_2 = 0.001$ and $\lambda_3 = 0$), the accuracy soars to 57.44%, a significant improvement over the vanilla Co-Tr. This confirms the premise of our work, i.e., the importance of diversity in Co-Tr. For example, by fixing $\lambda_2 = 0.001$ and varying λ_3 in a range from 0.0005 to 0.005, the accuracy varies in the range [58.25 %,60.56%]. Similar trends can be observed if we pick λ_2 reasonably. For example, with $\lambda_3 = 0.001$, changing λ_2 from 0.0001 to 0.005 results in accuracies in the range of [58.22 %, 59.45%].

Overall, we have observed a very robust performance with DCT, outperforming vanilla Co-Tr framework consistently with ease. Here, we recommend a simple and general way to set λ_2 and λ_3 which is to set both of them as 0.001. However, one can fine tune them to achieve even better results and our suggestion is to set λ_2 a bit smaller than λ_3.

7.3. Ablation study on effect of $\in$

In this part, we study the importance of $\in$ in our DCT algorithm for the CUB200-2011 dataset in the presence of 50% symmetric noisy labels. The results are shown in Table 10. It is clearly observed that with increasing the $\in$, the performance drops. As seen, the performance drops as $\in$ is increased from 0 (i.e. 2-Wasserstein Distance) to $+\infty$ (MMD). Thereby $\mathcal{S}_{\in=0}$ (i.e. 2-Wasserstein Distance) clearly outperforms $S_{\in+\infty}$ (MMD).

8. Conclusion

A novel and effective method (i.e. Co-Tr utilizing $\mathcal{S}_{\in}$) for training deep neural networks was presented. It was demonstrated to be outperforming all other baseline methods in the vast majority of settings influenced by noisy labels, as well as in a semi-supervised setting. $\mathcal{S}_{\in}$ divergence was used, in the middle of the network, to drive the networks to learn distinct features, while the $\mathcal{S}_{\in}$ at the end enforces the networks to still learn similar class probability distributions. Multi-label classification tasks is another area with related challenges where similar approaches may prove successful.

Credit author statement

No.	Name(s) of author(s)	Highest academic degree	Author affiliations	Contribution
1	Yan Han (corresponding)	Master	Australian National University	Conceptualization; Methodology; Software; Validation; Investigation; Writing-Original Draft.
2	Soumava Kumar Roy	Master	Australian National University	Conceptualization; Methodology; Writing-Review & Editing.
3	Lars Petersson	Doctor	Commonwealth Scientific and Industrial Research Organization	Conceptualization; Methodology; Supervision; Writing-Review & Editing.
4	Mehrtash Harandi	Doctor	Monash University	Conceptualization; Methodology; Supervision; Writing-Review & Editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, Proceedings of the Eleventh Annual Conference on Computational Learning Theory, Citeseer 1998, pp. 92–100.
[2] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[3] A. Argyriou, T. Evgeniou, M. Pontil, Multi-task feature learning, Advances in Neural Information Processing Systems 2007, pp. 41–48.
[4] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Cikm, Vol. 5 (2000) 3.
[5] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, M. Sugiyama, Co-teaching: robust training of deep neural networks with extremely noisy labels, Advances in Neural Information Processing Systems 2018, pp. 8527–8537.
[6] S. Qiao, W. Shen, Z. Zhang, B. Wang, A. Yuille, Deep Co-Training for semi-supervised image recognition, The European Conference on Computer Vision (ECCV), 2018.
[7] S. Kiritchenko, S. Matwin, Email classification with co-training, Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, IBM Corp 2011, pp. 301–312.
[8] Y. Sun, L. Li, Z. Xie, Q. Xie, X. Li, G. Xu, Co-training an improved recurrent neural network with probability statistic models for named entity recognition, International Conference on Database Systems for Advanced Applications, Springer 2017, pp. 545–555.
[9] R.E. Banfield, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, Ensemble diversity measures and their application to thinning, Inform. Fusion 6 (1) (2005) 49–62.
[10] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[11] J.C. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, Vol. 65, John Wiley & Sons, 2005.
[12] E. Malach, S. Shalev-Shwartz, Decoupling "when to update" from "how to update", Advances in Neural Information Processing Systems 2017, pp. 960–970.
[13] A. Genevay, G. Peyré, M. Cuturi, Learning generative models with sinkhorn divergences, International Conference on Artificial Intelligence and Statistics, PMLR 2018, pp. 1608–1617.
[14] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, Deepfashion: powering robust clothes recognition and retrieval with rich annotations, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[15] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A.C. Berg, L. Fei-Fei, Scalable multi-label annotation, ACM Conference on Human Factors in Computing Systems (CHI), 2014.
[16] S. Chen, G. Bortsova, A.G.-U. Juarez, G. van Tulder, M. de Bruijne, Multi-task Attention-based Semi-supervised Learning for Medical Image Segmentation, arXiv preprint arXiv:1907.12303.
[17] Y. Chai, V. Lempitsky, A. Zisserman, Bicos: a bi-level co-segmentation method for image classification, 2011 International Conference on Computer Vision, IEEE 2011, pp. 2579–2586.

[18] Y. Gong, Q. Ke, M. Isard, S. Lazebnik, A multi-view embedding space for modeling internet images, tags, and their semantics, Int. J. Comp. Vision 106 (2) (2014) 210–233.
[19] G. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network, arXiv preprint arXiv:1503.02531.
[20] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, pp. 4320–4328.
[21] D.P. Kingma, M. Welling, Auto-encoding Variational Bayes, arXiv preprint arXiv:1312.6114.
[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems 2014, pp. 2672–2680.
[23] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[24] C.D. Manning, C.D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
[25] J. Feydy, T. Séjourné, F.-X. Vialard, S.-I. Amari, A. Trouvé, G. Peyré, Interpolating between Optimal Transport and MMD using Sinkhorn Divergences, arXiv preprint arXiv:1810.08278.
[26] Y. Li, K. Swersky, R. Zemel, Generative moment matching networks, International Conference on Machine Learning 2015, pp. 1718–1727.
[27] Y. Han, S. Roy, L. Petersson, M. Harandi, Learning from noisy labels via discrepant collaborative training, The IEEE Winter Conference on Applications of Computer Vision 2020, pp. 3169–3178.
[28] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, L. Fei-Fei, Mentornet: Learning Data-driven Curriculum for Very Deep Neural Networks on Corrupted Labels, arXiv preprint arXiv:1712.05055.
[29] M. Ren, W. Zeng, B. Yang, R. Urtasun, Learning to Reweight Examples for Robust Deep Learning, arXiv preprint arXiv:1803.09050.
[30] M. Sajjadi, M. Javanmardi, T. Tasdizen, Regularization with stochastic transformations and perturbations for deep semi-supervised learning, Advances in Neural Information Processing Systems 2016, pp. 1163–1171.
[31] T. Miyato, S.-i. Maeda, M. Koyama, S. Ishii, Virtual adversarial training: a regularization method for supervised and semi-supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 41 (8) (2018) 1979–1993.
[32] A. Genevay, M. Cuturi, G. Peyré, F. Bach, Stochastic optimization for large-scale optimal transport, Advances in Neural Information Processing Systems 2016, pp. 3440–3448.
[33] M.A. Alvarez, L. Rosasco, N.D. Lawrence, et al., Kernels for Vector-valued Functions: A Review, Foundations and Trends® in Machine Learning, 4 (3), 2012 195–266.
[35] Y. LeCun, The mnist Database of Handwritten Digits, http://yann.lecun.com/exdb/mnist/.
[36] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images, Tech. rep. Citeseer, 2009.
[37] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading Digits in Natural Images With Unsupervised Feature Learning, 2011.
[38] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, Tech. Rep. CNS-TR-2011-001 California Institute of Technology, 2011.
[39] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3d object representations for fine-grained categorization, 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013.
[40] S. Laine, T. Aila, Temporal Ensembling for Semi-supervised Learning, arXiv preprint arXiv:1610.02242.
[41] T. Miyato, A.M. Dai, I. Goodfellow, Adversarial Training Methods for Semi-supervised Text Classification, arXiv preprint arXiv:1605.07725.
[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, Proceedings of the IEEE conference on computer vision and pattern recognition 2015, pp. 1–9.
[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, CVPR09, 2009.
[44] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, L. Qu, Making deep neural networks robust to label noise: a loss correction approach, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017, pp. 1944–1952.
[45] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, M. Sugiyama, Co-teaching: Robust Training of Deep Neural Networks With Extremely Noisy Labels, arXiv preprint arXiv:1804.06872.
[46] H. Wei, L. Feng, X. Chen, B. An, Combating noisy labels by agreement: a joint training method with co-regularization, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020, pp. 13726–13735.
[47] L. Jiang, D. Huang, M. Liu, W. Yang, Beyond synthetic noise: Deep learning on controlled noisy labels, International Conference on Machine Learning, PMLR 2020, pp. 4804–4815.
[48] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, pp. 770–778.
[49] D. Ortego, E. Arazo, P. Albert, N.E. O'Connor, K. McGuinness, Multi-objective Interpolation Training for Robustness to Label Noise, arXiv preprint arXiv:2012.04462.
[50] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, Advances in Neural Information Processing Systems 2016, pp. 2234–2242.
[51] A. Tarvainen, H. Valpola, Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results, Advances in Neural Information Processing Systems 2017, pp. 1195–1204.