**Noname manuscript No.**
(will be inserted by the editor)

# R2-D2: Repetitive Reprediction Deep Decipher for Semi-Supervised Deep Learning

**Guo-Hua Wang · Jianxin Wu**

Received: date / Accepted: date

**Abstract** Most recent semi-supervised deep learning (deep SSL) methods used a similar paradigm: use network predictions to update pseudo-labels and use pseudo-labels to update network parameters iteratively. However, they lack theoretical support and cannot explain why predictions are good candidates for pseudo-labels in the deep learning paradigm. In this paper, we propose a principled end-to-end framework named deep decipher (D2) for SSL. Within the D2 framework, we prove that pseudo-labels are related to network predictions by an exponential link function, which gives a theoretical support for using predictions as pseudo-labels. Furthermore, we demonstrate that updating pseudo-labels by network predictions will make them uncertain. To mitigate this problem, we propose a training strategy called repetitive reprediction (R2). Finally, the proposed R2-D2 method is tested on the large-scale ImageNet dataset and outperforms state-of-the-art methods by 5 percentage points.

**Keywords** semi-supervised learning · deep learning · image classification

## 1 Introduction

Deep learning has achieved state-of-the-art results on many visual recognition tasks. However, training these models often needs large-scale datasets such as ImageNet (Russakovsky et al., 2015). Nowadays, it is easy to collect images by search engines, but image annotation is expensive and time-consuming. Semi-supervised learning (SSL) is a paradigm to learn a model with a few labeled data and massive amounts of unlabeled data. With the help of unlabeled data, the model performance may be improved.

With a supervised loss, unlabeled data can be used in training by assigning pseudo-labels to them. Many state-of-the-art methods on semi-supervised deep learning used pseudo-labels implicitly. Temporal Ensembling (Laine and Aila, 2017) used the moving average of network predictions as pseudo-labels. Mean Teacher (Tarvainen and Valpola, 2017) and Deep Co-training (Qiao et al., 2018) employed another network to generate pseudo-labels. However, they produced or updated pseudo-labels in ad-hoc manners. Although these methods worked well in practice, there are few theories to support them. A mystery in deep SSL arises: why can predictions work well as pseudo-labels?

In this paper, we propose an end-to-end framework called deep decipher (D2). Inspired by Yi and Wu (2019), we treat pseudo-labels as variables and update them by back-propagation, which are also learned from data. The D2 framework specifies a well-defined optimization problem, which can be properly interpreted as a maximum likelihood estimation over two set of variables (the network parameters and the pseudo-labels). Within deep decipher, we prove that there exists an exponential relationship between pseudo-labels and network predictions, leading to a theoretical support for using network predictions as pseudo-labels. Then, we further analyze the D2 framework and prove that pseudo-labels will become flat (i.e., their entropy is high) during training and there is an equality constraint bias in it. To mitigate these problems, we propose a simple but effective strategy, repetitive reprediction (R2). The improved D2 framework is named

This work is supported by the National Natural Science Foundation of China (61772256, 61921006).

J. Wu is the corresponding author.

Guo-Hua Wang
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China.
E-mail: wangguohua@lamda.nju.edu.cn

Jianxin Wu
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China.
E-mail: wujx2001@gmail.com

R2-D2 and obtains state-of-the-art results on several SSL problems.

Our contributions are as follows.

- We propose D2, a deep learning framework that deciphers the relationship between predictions and pseudo-labels. D2 updates pseudo-labels by back-propagation. To the best of our knowledge, D2 is the first deep SSL method that learns pseudo-labels from data end-to-end.
- Within D2, we prove that pseudo-labels are exponentially transformed from the predictions. Hence, it is reasonable for previous works to use network predictions as pseudo-labels. Meanwhile, many SSL methods can be considered as special cases of D2 in terms of certain aspects.
- To further boost D2's performance, we find some shortcomings of D2. In particular, we prove that pseudo-labels will become flat during the optimization. To mitigate this problem, we propose a simple but effective remedy, R2. We tested the R2-D2 method on ImageNet and it outperforms state-of-the-arts by a large margin. On small-scale datasets like CIFAR-10 (Krizhevsky and Hinton, 2009), R2-D2 also produces state-of-the-art results.

## 2 Related Works

We first briefly review deep SSL methods and the related works that inspired this paper.

Lee (2013) is an early work on training deep SSL models by pseudo-labels, which picks the class with the maximum predicted probability as pseudo-label for unlabeled images and tested only on a samll-scale dataset MNIST (LeCun et al., 1998). Label propagation (Zhu and Ghahramani, 2002) can be seen as a form of pseudo-labels. Based on some metric, label propagation pushes the label information of each sample to the near samples. Weston et al. (2012) apply label propagation to deep learning models. Lu and Peng (2013) use label propagation to solve the exhaustively propagating pairwise constraints problem. Iscen et al. (2019) use the manifold assumption to generate pseudo-labels for unlabeled data. However, their method is complicated and relies on other SSL methods to produce state-of-the-art results.

Several recent state-of-the-art deep SSL methods can be considered as using pseudo-labels implicitly. Temporal ensembling (Laine and Aila, 2017) proposes making the current prediction and the pseudo-labels consistent, where the pseudo-labels take into account the network predictions over multiple previous training epochs. Extending this idea, Mean Teacher (Tarvainen and Valpola, 2017) employs a secondary model, which uses the exponential moving average weights to generate pseudo-labels. Virtual Adversarial Training (Miyato et al., 2018) uses network predictions as pseudo-labels, then they want the network predictions under adversarial perturbation to be consistent with pseudo-labels. Deep Co-Training (Qiao et al., 2018) employs many networks and uses one network to generate pseudo-labels for training other networks.

We notice that they all use the network predictions as pseudo-labels but a theory explaining its rationale in the deep learning context is missing. Within our D2 framework, we demonstrate that pseudo-labels will indeed be related to network predictions. That gives a support to using network predictions as pseudo-labels. Moreover, pseudo-labels of previous works were designed manually and ad-hoc, but our pseudo-labels are updated by training the end-to-end framework. Many previous SSL methods can also be considered as special cases of the D2 framework in terms of certain aspects in these methods.

There are some previous works in other fields that inspired this work. Deep label distribution learning (Gao et al., 2017) inspires us to use label distributions to encode the pseudo-labels. Tanaka et al. (2018) studies the label noise problem. They find it is possible to update noisy labels to make them more precise during the training. PENCIL (Yi and Wu, 2019) proposes an end-to-end framework to train the network and optimize the noisy labels together. Our method is inspired by PENCIL (Yi and Wu, 2019). In addition, inspired by Liu et al. (2018), we analyze our algorithm from the gradient perspective. [1]

## 3 The R2-D2 Method

We define the notations first. Column vectors and matrices are denoted in bold (e.g., $\mathbf{x}, \mathbf{X}$). When $\mathbf{x} \in \mathbb{R}^d$, $x_i$ is the $i$-th element of the vector $\mathbf{x}$, $i \in [d]$, where $[d] := \{1, 2, \ldots, d\}$. $\mathbf{w}_i$ denote the $i$-th column of matrix $\mathbf{W} \in \mathbb{R}^{d\times l}$, $i \in [l]$. And, we assume the dataset has $N$ classes.

### 3.1 Deep decipher (D2)

Figure 1 shows the D2 pipeline, which is inspired by Yi and Wu (2019). Given an input image $\mathbf{x}$, D2 can employ any backbone network to generate feature $\mathbf{f} \in \mathbb{R}^D$. Then, the linear activation $\hat{\mathbf{y}} \in \mathbb{R}^N$ is computed as $\hat{\mathbf{y}} = \mathbf{W}^\mathsf{T}\mathbf{f}$, where $\mathbf{W} \in \mathbb{R}^{D\times N}$ are weights of the FC layer and we omit the bias term for simplicity. The softmax function is denoted as $\sigma(\mathbf{y}) : \mathbb{R}^N \to \mathbb{R}^N$ and $\sigma(\mathbf{y})_i = \frac{\exp(y_i)}{\sum_{j=1}^N \exp(y_j)}$. Then, the prediction $\hat{\mathbf{p}}$ is calculated as $\hat{\mathbf{p}} = \sigma(\hat{\mathbf{y}})$ , hence

$$\hat{p}_n = \sigma(\hat{\mathbf{y}})_n = \sigma(\mathbf{W}^\mathsf{T}\mathbf{f})_n = \frac{\exp(\mathbf{w}_n^\mathsf{T}\mathbf{f})}{\sum_{i=1}^N \exp(\mathbf{w}_i^\mathsf{T}\mathbf{f})} . \tag{1}$$

[1] Preliminary studies of the proposed R2-D2 method appeared as a conference presentation (Wang and Wu, 2020), available at `https://arxiv.org/abs/1908.04345`.

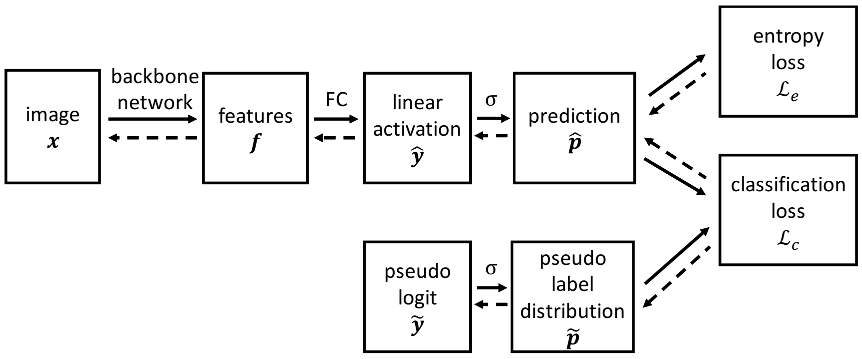


Figure 1: The pipeline of D2. Solid lines and dashed lines represent the forward and back-propagation processes, respectively.

We define $\tilde{\mathbf{y}}$ as the pseudo logit which is an unconstrained variable and *can* be updated by back-propagation. Then, the pseudo label is calculated as $\tilde{\mathbf{p}} = \sigma(\tilde{\mathbf{y}})$ and it is a valid probability distribution.

In the training, the D2 framework is initialized as follows. Firstly, we train the backbone network using only labeled examples, and use this trained network as the backbone network and FC in Figure 1. For labeled examples, $\tilde{\mathbf{y}}$ is initialized by $K\mathbf{y}$, in which $K = 10$ and $\mathbf{y}$ is the groundtruth label in the one-hot encoding. Note that $\tilde{\mathbf{y}}$ of labeled examples will *not* be updated during D2 training. For unlabeled examples, we use the trained network to predict $\tilde{\mathbf{y}}$. That means we use the FC layer activation $\hat{\mathbf{y}}$ as the initial value of $\tilde{\mathbf{y}}$. The process of initializing pseudo-labels is called predicting pseudo-labels in this paper. In the testing, we use the backbone network with FC layer to make predictions and the branch of pseudo-labels is removed.

Our loss function consists of $\mathscr{L}_c$ and $\mathscr{L}_e$. $\mathscr{L}_c$ is the classification loss and defined as $KL(\hat{\mathbf{p}}||\tilde{\mathbf{p}})$ as in Yi and Wu (2019), which is different from the classic KL-loss $KL(\tilde{\mathbf{p}}||\hat{\mathbf{p}})$. $\mathscr{L}_c$ is used to make the network predictions match the pseudo-labels. $\mathscr{L}_e$ is the entropy loss, defined as $-\sum_{j=1}^{N} \hat{p}_j \log(\hat{p}_j)$. Minimizing the entropy of the network prediction can encourage the network to peak at only one category. So our loss function is defined as

$$\begin{aligned}\mathscr{L} &= \alpha\mathscr{L}_c + \beta\mathscr{L}_e \\ &= \alpha\sum_{j=1}^{N}\hat{p}_j\left[\log(\hat{p}_j) - \log(\tilde{p}_j)\right] - \beta\sum_{j=1}^{N}\hat{p}_j\log(\hat{p}_j),\end{aligned} \tag{2}$$

where $\alpha$ and $\beta$ are two hyperparameters. Although there are two hyperparameters in D2, we always set $\alpha = 0.1$ and $\beta = 0.03$ in all our experiments.

Then, we show that we can decipher the relationship between pseudo-labels and network predictions in D2, as shown by Theorem 1.

**Theorem 1** *Suppose D2 is trained by SGD with the loss function $\mathscr{L} = \alpha\mathscr{L}_c + \beta\mathscr{L}_e$. Let $\hat{\mathbf{p}}$ denote the prediction by the network for one example and $\hat{p}_n$ is the largest value in $\hat{\mathbf{p}}$. After the optimization algorithm converges, we have $\tilde{p}_n \to \exp(-\frac{\mathscr{L}}{\alpha})(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$.*

*Proof* First, the loss function can be rewritten as

$$\begin{aligned}\mathscr{L} = (\alpha-\beta)\sum_{j=1}^{N}\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_j\log\left(\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_j\right) \\ -\alpha\sum_{j=1}^{N}\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_j\log(\tilde{p}_j).\end{aligned} \tag{3}$$

It is easy to see

$$\begin{aligned}\frac{\partial\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_j}{\partial\mathbf{w}_n} = \mathbb{I}(j=n)\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_j\mathbf{f} \\ -\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_j\sigma\left(\mathbf{W}^{\mathsf{T}}\mathbf{f}\right)_n\mathbf{f},\end{aligned} \tag{4}$$

in where $\mathbb{I}(\cdot)$ is the indicator function. Now we can compute the gradient of $\mathscr{L}$ with respect to $\mathbf{w}_n$:

$$\begin{aligned}\frac{\partial\mathscr{L}}{\partial\mathbf{w}_n} = (\alpha-\beta)\sum_{j=1}^{N}\Bigg[\frac{\partial\sigma(\hat{\mathbf{y}})_j}{\partial\mathbf{w}_n}\log\left(\sigma(\hat{\mathbf{y}})_j\right) \\ +\sigma(\hat{\mathbf{y}})_j\frac{\partial\log\left(\sigma(\hat{\mathbf{y}})_j\right)}{\partial\mathbf{w}_n}\Bigg] \\ -\alpha\sum_{j=1}^{N}\frac{\partial\sigma(\hat{\mathbf{y}})_j}{\partial\mathbf{w}_n}\log(\tilde{p}_j)\end{aligned} \tag{5}$$

$$\begin{aligned}&= (\alpha-\beta)\sum_{j=1}^{N}\left[\mathbb{I}(j=n)-\sigma(\hat{\mathbf{y}})_n\right]\sigma(\hat{\mathbf{y}})_j\mathbf{f}\log\left(\sigma(\hat{\mathbf{y}})_j\right) \\ &\quad+(\alpha-\beta)\sum_{j=1}^{N}\sigma(\hat{\mathbf{y}})_j\left(\mathbb{I}(j=n)\mathbf{f}-\sigma(\hat{\mathbf{y}})_n\mathbf{f}\right) \\ &\quad-\alpha\sum_{j=1}^{N}\left[\mathbb{I}(j=n)\sigma(\hat{\mathbf{y}})_j\mathbf{f}-\sigma(\hat{\mathbf{y}})_j\sigma(\hat{\mathbf{y}})_n\mathbf{f}\right]\log(\tilde{p}_j)\end{aligned} \tag{6}$$

$$= (\alpha-\beta)\sigma(\hat{\mathbf{y}})_n\log\left(\sigma(\hat{\mathbf{y}})_n\right)\mathbf{f}$$

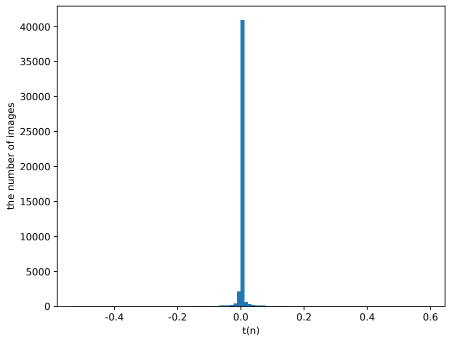


Figure 2: Distribution of $t(n)$ on the whole CIFAR-10 dataset at the end of the D2 training. $t(n)$ is defined as $(\alpha-\beta)\log(\hat{p}_n)-\alpha\log(\tilde{p}_n)-\mathscr{L}$. We can see $t(n)=0$ for almost all images, where $n$ is calculated according to each image.

$$
\begin{aligned}
&-(\alpha-\beta)\sum_{j=1}^{N}\sigma(\hat{\mathbf{y}})_j\log\left(\sigma(\hat{\mathbf{y}})_j\right)\sigma(\hat{\mathbf{y}})_n\mathbf{f}\\
&+(\alpha-\beta)\sigma(\hat{\mathbf{y}})_n\mathbf{f}-(\alpha-\beta)\sigma(\hat{\mathbf{y}})_n\mathbf{f}\sum_{j=1}^{N}\sigma(\hat{\mathbf{y}})_j\\
&-\alpha\sigma(\hat{\mathbf{y}})_n\log(\tilde{p}_n)\mathbf{f}+\alpha\sum_{j=1}^{N}\sigma(\hat{\mathbf{y}})_j\log(\tilde{p}_j)\sigma(\hat{\mathbf{y}})_n\mathbf{f}
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
=&(\alpha-\beta)\sigma(\hat{\mathbf{y}})_n\log(\sigma(\hat{\mathbf{y}})_n)\mathbf{f}-\alpha\sigma(\hat{\mathbf{y}})_n\log(\tilde{p}_n)\mathbf{f}\\
&-\sigma(\hat{\mathbf{y}})_n\mathbf{f}\left[(\alpha-\beta)\sum_{j=1}^{N}\sigma(\hat{\mathbf{y}})_j\log\left(\sigma(\hat{\mathbf{y}})_j\right)\right.\\
&\left.-\alpha\sum_{j=1}^{N}\sigma(\hat{\mathbf{y}})_j\log(\tilde{p}_j)\right]\\
&+(\alpha-\beta)\sigma(\hat{\mathbf{y}})_n\mathbf{f}-(\alpha-\beta)\sigma(\hat{\mathbf{y}})_n\mathbf{f}
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
=&[(\alpha-\beta)\log(\sigma(\hat{\mathbf{y}})_n)-\alpha\log(\tilde{p}_n)]\,\sigma(\hat{\mathbf{y}})_n\mathbf{f}\\
&-\mathscr{L}\sigma(\hat{\mathbf{y}})_n\mathbf{f}
\end{aligned} \tag{9}
$$

$$
=[(\alpha-\beta)\log(\sigma(\hat{\mathbf{y}})_n)-\alpha\log(\tilde{p}_n)-\mathscr{L}]\,\sigma(\hat{\mathbf{y}})_n\mathbf{f} \tag{10}
$$

$$
=[(\alpha-\beta)\log(\hat{p}_n)-\alpha\log(\tilde{p}_n)-\mathscr{L}]\,\hat{p}_n\mathbf{f}\,. \tag{11}
$$

During training, we expect the optimization algorithm can converge and finally $\frac{\partial\mathscr{L}}{\partial\mathbf{w}_n}\to\mathbf{0}$. Because $\mathbf{f}$ will not be $\mathbf{0}$, we conclude that $[(\alpha-\beta)\log(\hat{p}_n)-\alpha\log(\tilde{p}_n)-\mathscr{L}]\,\hat{p}_n\to 0$. Because $\sum_{i=1}^{N}\hat{p}_i=1$, consider the fact that $\hat{p}_n$ is the largest value in $\{\hat{p}_1,\hat{p}_1,\dots,\hat{p}_N\}$, then $\hat{p}_n\not\to 0$ at the end of training. So we have $[(\alpha-\beta)\log(\hat{p}_n)-\alpha\log(\tilde{p}_n)-\mathscr{L}]\to 0$, which easily translates to $\tilde{p}_n\to\exp(-\frac{\mathscr{L}}{\alpha})(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$. □

We would like to show experimental results for verifying Theorem 1. Let $t(n)$ denote $(\alpha-\beta)\log(\hat{p}_n)-\alpha\log(\tilde{p}_n)-\mathscr{L}$. Now, consider a single sample, suppose $\hat{\mathbf{p}}$ will get the largest value at $n$ where $n\in\{1,2,\dots,N\}$. Then it is expected that $\hat{p}_n\to 1$ and $t(n)\to 0$ at the end of training. Figure 2 shows the distribution of $t(n)$ on the whole CIFAR-10 dataset, where $n$ is calculated according to different samples. The distribution is almost gathered around 0. So we also observed empirically that $\tilde{p}_n\to\exp(-\frac{\mathscr{L}}{\alpha})(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$, where $n$ is the class predicted by the network.

Theorem 1 tells us $\tilde{p}_n$ converges to $\exp(-\frac{\mathscr{L}}{\alpha})(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$ during the optimization. And at last, we expect that $\tilde{p}_n=\exp(-\frac{\mathscr{L}}{\alpha})(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$, in which $n$ is the class predicted by the network. In other words, we have deciphered that there is an exponential link between pseudo-labels and predictions. From $\tilde{p}_n\to\exp(-\frac{\mathscr{L}}{\alpha})(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$, we notice that $\tilde{p}_n$ is approximately proportional to $\hat{p}_n^{1-\frac{\beta}{\alpha}}$. That gives a theoretical support to use network predictions as pseudo-labels. And, it is required that $1-\frac{\beta}{\alpha}>0$ to make pseudo-labels and network predictions consistent. We must set $\alpha>\beta$. In our experiments, if we set $\alpha<\beta$, the training will indeed fail miserably.

Next, we analyze how $\tilde{\mathbf{y}}$ is updated in D2. With the loss function $\mathscr{L}$, the gradients of $\mathscr{L}$ with respect to $\tilde{y}_n$ is

$$
\frac{\partial\mathscr{L}}{\partial\tilde{y}_n}=\sum_{k=1}^{N}\frac{\partial\mathscr{L}}{\partial\tilde{p}_k}\frac{\partial\tilde{p}_k}{\partial\tilde{y}_n} \tag{12}
$$

$$
=-\alpha\sum_{k=1}^{N}\frac{\sigma(\hat{\mathbf{y}})_k}{\tilde{p}_k}\left(\mathbb{I}(k=n)\sigma(\tilde{\mathbf{y}})_k-\sigma(\tilde{\mathbf{y}})_k\sigma(\tilde{\mathbf{y}})_n\right) \tag{13}
$$

$$
=-\alpha\sum_{k=1}^{N}\frac{\sigma(\hat{\mathbf{y}})_k}{\sigma(\tilde{\mathbf{y}})_k}\left(\mathbb{I}(k=n)\sigma(\tilde{\mathbf{y}})_k-\sigma(\tilde{\mathbf{y}})_k\sigma(\tilde{\mathbf{y}})_n\right) \tag{14}
$$

$$
=-\alpha\sum_{k=1}^{N}\sigma(\hat{\mathbf{y}})_k\left(\mathbb{I}(k=n)-\sigma(\tilde{\mathbf{y}})_n\right) \tag{15}
$$

$$
=-\alpha\sum_{k=1}^{N}\sigma(\hat{\mathbf{y}})_k\mathbb{I}(k=n)+\alpha\sum_{k=1}^{N}\sigma(\hat{\mathbf{y}})_k\sigma(\tilde{\mathbf{y}})_n \tag{16}
$$

$$
=-\alpha\sigma(\hat{\mathbf{y}})_n+\alpha\sigma(\tilde{\mathbf{y}})_n\sum_{k=1}^{N}\sigma(\hat{\mathbf{y}})_k \tag{17}
$$

$$
=-\alpha\sigma(\hat{\mathbf{y}})_n+\alpha\sigma(\tilde{\mathbf{y}})_n\,. \tag{18}
$$

By gradient descent, the pseudo logit $\tilde{\mathbf{y}}$ is updated by

$$
\tilde{\mathbf{y}}\leftarrow\tilde{\mathbf{y}}-\lambda\frac{\partial\mathscr{L}}{\partial\tilde{\mathbf{y}}}=\tilde{\mathbf{y}}-\lambda\alpha\sigma(\tilde{\mathbf{y}})+\lambda\alpha\sigma(\hat{\mathbf{y}})\,, \tag{19}
$$

where $\lambda$ is the learning rate for updating $\tilde{\mathbf{y}}$. The reason we use one more hyperparameter $\lambda$ rather than the overall learning rate is because the magnitude of $\frac{\partial\mathscr{L}}{\partial\tilde{\mathbf{y}}}=-\alpha\sigma(\hat{\mathbf{y}})+\alpha\sigma(\tilde{\mathbf{y}})$ is much smaller than that of $\tilde{\mathbf{y}}$ (in part due to the sigmoid transform) and the overall learning rate is too small to update the pseudo logit (cf. Figure 3). We set $\lambda=4000$ in all our experiments.

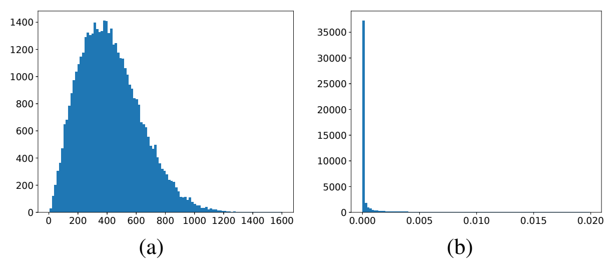


Figure 3: (a) shows the distribution of the number of images versus $\|\tilde{\mathbf{y}}\|_2$ in CIFAR-10. (b) shows the distribution of the number of images versus $\|\frac{\partial \mathscr{L}}{\partial \tilde{\mathbf{y}}}\|_2$. Note that the ranges of x-axis are *different* between (a) and (b). From the figure, we can see the magnitude of $\frac{\partial \mathscr{L}}{\partial \tilde{\mathbf{y}}} = -\alpha\sigma(\hat{\mathbf{y}}) + \alpha\sigma(\tilde{\mathbf{y}})$ is far less than that of $\tilde{\mathbf{y}}$. So we use one more hyperparameter $\lambda$ rather than the overall learning rate to update the pseudo logit $\tilde{\mathbf{y}}$.

The updating formulas in many previous works can be considered as special cases of that of D2. In Temporal Ensembling (Laine and Aila, 2017), the pseudo-labels $\tilde{\mathbf{p}}$ is a moving average of the network predictions $\hat{\mathbf{p}}$ during training. The updating formula is $\mathbf{P} \leftarrow \alpha\mathbf{P} + (1-\alpha)\hat{\mathbf{p}}$. To correct for the startup bias, the $\tilde{\mathbf{p}}$ needs to be divided by the factor $(1-\alpha^t)$, where $t$ is the number of epochs. So the updating formula of $\tilde{\mathbf{p}}$ is $\tilde{\mathbf{p}} \leftarrow \mathbf{P}/(1-\alpha^t)$. In Mean Teacher (Tarvainen and Valpola, 2017), the $\tilde{\mathbf{p}}$ is the prediction of a teacher model which uses the exponential moving average weights of the student model. Tanaka et al. (2018) proposed using the running average of the network predictions to estimate the groundtruth of the noisy label. However, their updating formula were designed manually and ad-hoc. In contrast, we treat pseudo-labels as updatable variables like the network parameters. These variables are learned by minimizing a well-defined loss function (cf. equation 2). From a probabilistic perspective, it is well known that minimizing the KL loss is equivalent to maximum likelihood estimation, in which the backbone network's architecture defines the estimation's functional space while SGD optimizes over these variables (both the network parameters and the pseudo-labels). We do not need to manually specify how the pseudo-labels are generated. This process is natural and principled.

### 3.2 An illustrative example

Now, we use a toy example to explain how the D2 framework works intuitively. Inspired by Liu et al. (2018), we use the LeNet (LeCun et al., 1998) as backbone structure and add two FC layers, in which the first FC layer learns a 2-D feature and the second FC layer projects the feature onto the class space. The network was trained on MNIST. Note that MNIST has 50000 images for training. We only used 1000 images as labeled images to train the network.

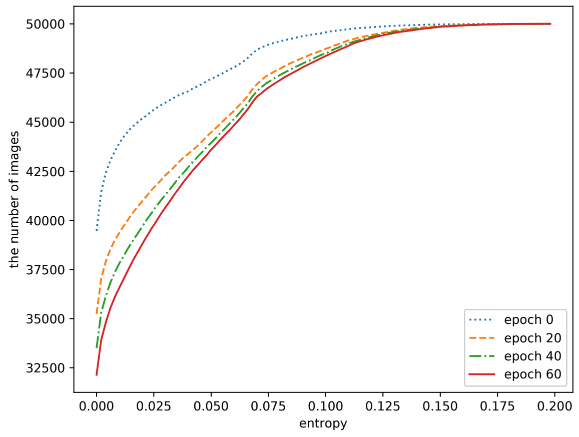


Figure 5: Cumulative distribution of the number of pseudo-labels versus the entropy. For each point $(n, e)$ on the line, it means there are $n$ images whose label entropies are less than $e$. Pseudo-labels will become flat as the D2 framework is trained more epochs. This figure is best viewed in color.

Figure 4a depicts the 2-D feature distribution of these 1000 images. We observe that features belonging to the same class will cluster together. Figure 4b shows the feature distribution of both these 1000 labeled and other 49000 unlabeled images. Although the network did not train on the unlabeled images, features belonging to the same class are still roughly clustered.

Pseudo-labels in our D2 framework are probability distributions and initialized by network predictions. As Figure 4b shows, features near the cluster center will have confident pseudo-labels and can be learned safely. However, features at the boundaries between clusters will have a pseudo-label whose corresponding distribution among different classes is flat rather than sharp. By training D2, the network will learn confident pseudo-labels first. Then it is expected that uncertain pseudo-labels will become more and more precise and confident by optimization. At last, each cluster will become more compact and the boundaries between different classes' features will become clear. Figure 4d depicts the feature distribution of all images after D2 training. Because the same class features of unlabeled images get closer, the same class features of labeled images will also get closer (cf. Figure 4c). That is how unlabeled images help the training in our D2 framework.

### 3.3 Repetitive reprediction (R2)

Although D2 has worked well in practice (cf. Table 1 column a), there are still some shortcomings in it. We will discuss two major ones. To mitigate these problems and further boost the performance, we propose a simple but effective strategy, repetitive reprediction (R2), to improve the D2 framework.

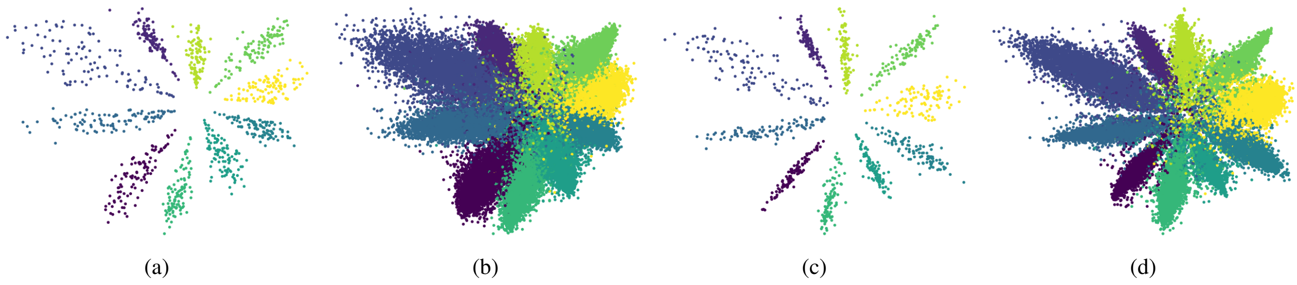


Figure 4: Feature distribution on MNIST. First, LeNet was trained by labeled data. (a) shows the the feature distribution of labeled images. Points with the same color belong to the same class. (b) shows the feature distribution of both labeled and unlabeled images. Then, we used LeNet as the backbone network and trained the D2 framework. After training, (c) and (d) show the feature distribution of labeled images and all images, respectively. This figure needs to be viewed in color.

First, we expect pseudo-labels can become more confident along with D2's learning process. Unfortunately, we observed that more and more pseudo-labels become flat during training (cf. Figure 5). Below, we prove Theorem 2 to explain why this adverse effect happens.

**Theorem 2** *Suppose D2 is trained by SGD with the loss function* $\mathscr{L} = \alpha\mathscr{L}_c + \beta\mathscr{L}_e$. *If* $\tilde{p}_n = \exp(-\frac{\mathscr{L}}{\alpha})\,(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$, *we must have* $\tilde{p}_n \le \hat{p}_n$.

*Proof* First, according to the loss function we defined, we have

$$\mathscr{L} = \alpha\sum_{j=1}^{N}\hat{p}_j\left[\log(\hat{p}_j)-\log(\tilde{p}_j)\right]-\beta\sum_{j=1}^{N}\hat{p}_j\log(\hat{p}_j) \tag{20}$$

$$\ge -\beta\sum_{j=1}^{N}\hat{p}_j\log(\hat{p}_j) \tag{21}$$

$$\ge -\beta\sum_{j=1}^{N}\hat{p}_j\log(\hat{p}_n) \tag{22}$$

$$= -\beta\log(\hat{p}_n)\,, \tag{23}$$

where $\hat{p}_n$ is the largest value in $\{\hat{p}_1,\hat{p}_2,\ldots,\hat{p}_N\}$. Then, from $\tilde{p}_n = \exp\left(-\frac{\mathscr{L}}{\alpha}\right)\hat{p}_n^{1-\frac{\beta}{\alpha}}$ and $\mathscr{L} \ge -\beta\log(\hat{p}_n)$, we have

$$\tilde{p}_n = \exp\left(-\frac{\mathscr{L}}{\alpha}\right)\hat{p}_n^{1-\frac{\beta}{\alpha}} \le \exp\left(\frac{\beta\log(\hat{p}_n)}{\alpha}\right)\hat{p}_n^{1-\frac{\beta}{\alpha}} = \hat{p}_n\,. \tag{24}$$

□

We show that $\tilde{p}_n \le \hat{p}_n$ holds in experiments. With $\tilde{p}_n = \exp(-\frac{\mathscr{L}}{\alpha})\hat{p}_n^{1-\frac{\beta}{\alpha}}$, if $\tilde{p}_n \le \hat{p}_n$, that yields $\exp(-\frac{\mathscr{L}}{\alpha})\hat{p}_n^{1-\frac{\beta}{\alpha}} \le \hat{p}_n$. Then we can get $\hat{p}_n \ge \exp(-\frac{\mathscr{L}}{\beta})$. Figure 6 shows $\hat{p}_n$ versus $\exp(-\frac{\mathscr{L}}{\beta})$, in which $\beta = 0.03$. For a specific loss value, if $p_n$ is above the function curve, $\tilde{p}_n$ is smaller than $p_n$. Figure 6 shows the scatter plot of $(\mathscr{L},\hat{p}_n)$ at the end of the D2 training on CIFAR-10. Almost all points are above the curve. That means if $\tilde{p}_n \to \exp(-\frac{\mathscr{L}}{\alpha})\hat{p}_n^{1-\frac{\beta}{\alpha}}$, $\tilde{p}_n$ will be smaller than $\hat{p}_n$.

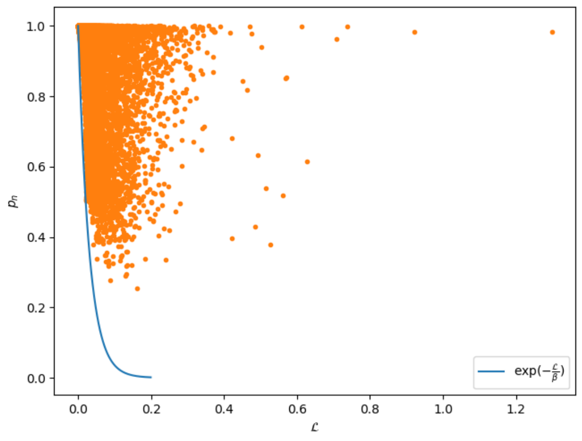


Figure 6: The curve of $\exp(-\frac{\mathscr{L}}{\beta})$ and $(\mathscr{L},p_n)$ at the end of the D2 training on CIFAR-10. This figure is best viewed in color.

From Theorem 1, we get $\tilde{p}_n \to \exp(-\frac{\mathscr{L}}{\alpha})\,(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$, where $\hat{\mathbf{p}}$ gets the largest value at $\hat{p}_n$. And Theorem 2 tells us if $\tilde{p}_n = \exp(-\frac{\mathscr{L}}{\alpha})\,(\hat{p}_n)^{1-\frac{\beta}{\alpha}}$ then $\tilde{p}_n$ will be smaller than $\hat{p}_n$. Because $\tilde{\mathbf{p}}$ and $\hat{\mathbf{p}}$ are probability distributions, if $\tilde{\mathbf{p}}$ and $\hat{\mathbf{p}}$ get their largest value at $n$, $\tilde{\mathbf{p}}$ is more flat than $\hat{\mathbf{p}}$ when $\tilde{p}_n \le \hat{p}_n$. That is, along with the training of D2, there is a tendency that pseudo-labels will be more flat than the network predictions.

Second, we find an unsolicited bias in the D2 framework. From the updating formula, we can get

$$\sum_{i=1}^{N}\tilde{y}_i \leftarrow \sum_{i=1}^{N}\tilde{y}_i-\lambda\alpha\sum_{i=1}^{N}\sigma(\tilde{\mathbf{y}})_i+\lambda\alpha\sum_{i=1}^{N}\sigma(\hat{\mathbf{y}})_i \tag{25}$$

$$=\sum_{i=1}^{N}\tilde{y}_i-\lambda\alpha+\lambda\alpha \tag{26}$$

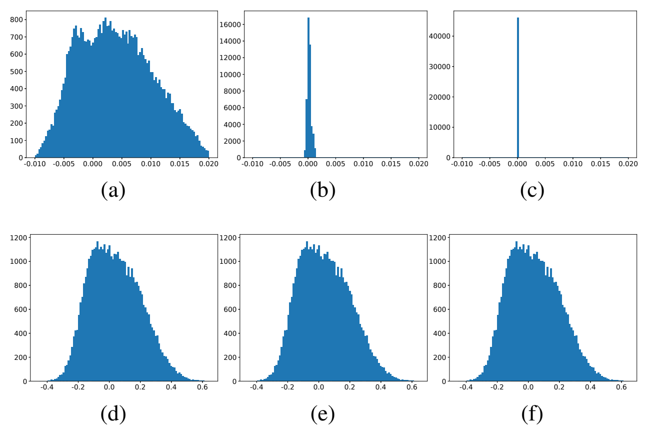


Figure 7: During the D2 training, the distribution of $\sum_{i=1}^{N}\hat{y}_i$ on the unlabeled samples at 100, 200, 300 epoch on CIFAR-10 are showed by (a), (b), (c), respectively. Note that $\sum_{i=1}^{N}\hat{y}_i$ will get more and more concentrated with training. The distributions of $\sum_{i=1}^{N}\tilde{y}_i$ on the unlabeled samples at 100, 200, 300 epoch on CIFAR-10 are showed by (d), (e), (f), respectively. According to our analysis, $\sum_{i=1}^{N}\tilde{y}_i$ will not change. Experimental results are consistent with our theoretical analysis.

$$= \sum_{i=1}^{N}\tilde{y}_i \,. \tag{27}$$

That is, $\sum_{i=1}^{N}\tilde{y}_i$ will *not* change after initialization. Although we define $\tilde{\mathbf{y}}$ as the variable which is unconstrained, the softmax function and SGD set an equality constraint for it. On the other hand, in practice, $\sum_{i=1}^{N}\hat{y}_i$ become more and more concentrated (cf. Figure 7). Later, we will use an ablation study to demonstrate this bias is harmful.

We propose a repetitive reprediction (R2) strategy to overcome these difficulties, which repeatedly perform repredictions (i.e., using the prediction $\hat{\mathbf{y}}$ to re-initialize the pseudo-labels $\tilde{\mathbf{y}}$ several times) during training D2. The benefits of R2 are two-fold. First, we want to make pseudo-labels confident. According to our analysis, the network predictions are sharper than pseudo-labels when the algorithm converges. So repredicting pseudo-labels can make them sharper. Second, $\sum_{i=1}^{N}\tilde{y}_i$ will not change during D2 training. Reprediction can reduce the impact of this bias. Furthermore, the validation accuracy often increase during training. A repeated reprediction can make pseudo-labels more accurate than that of the last reprediction.

Apart from the repredictions, we also reduce the learning rate to boost the performance. If the D2 framework is trained by a fixed learning rate as in Yi and Wu (2019), the loss $\mathscr{L}$ did not descend in experiments (cf. Figure 8). Reducing the learning rate can make the loss descend (cf. Figure 9). We can get some benefits from a lower loss. On one hand, $\mathscr{L}_c$ is the KL divergence between pseudo-labels and the network predictions. Minimizing this term makes pseudo-labels as

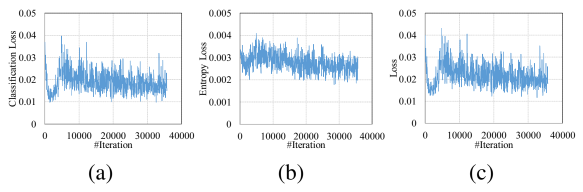


Figure 8: (a), (b), (c) show how $\mathscr{L}_c$, $\mathscr{L}_e$, $\mathscr{L}$ change by training D2 without R2 on CIFAR-10 (column a in Table 1), respectively. With a fixed learning rate, it is difficult for these loss terms to decrease.

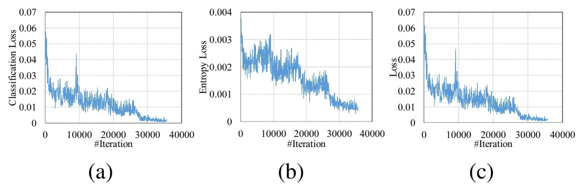


Figure 9: (a), (b), (c) show how $\mathscr{L}_c$, $\mathscr{L}_e$, $\mathscr{L}$ change by training D2 with R2 on CIFAR-10 (column e in Table 1), respectively. Reprediction occurs at 8750, 17500, and 26250 iterations. After each reprediction, we decrease the learning rate.

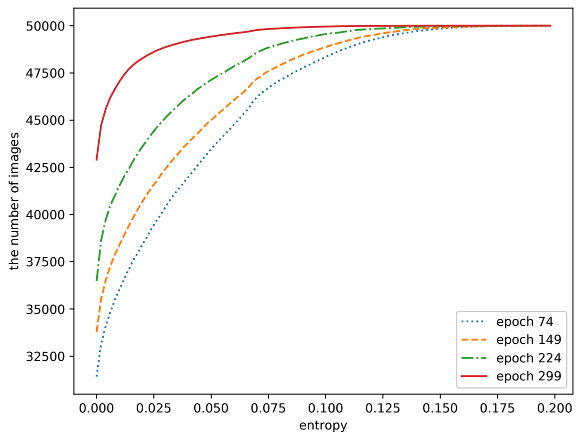


Figure 10: Cumulative distribution of the number of pseudo-labels versus the entropy after using the repetitive reprediction (R2) strategy on CIFAR-10. For each point $(n, e)$ on the line, it means there are $n$ images whose label entropies are less than $e$. Using the R2 strategy can make pseudo-labels sharper at the end of training. This figure is best viewed in color.

sharp as the network predictions. On the other hand, minimizing $\mathscr{L}_e$ can decrease the entropy of network predictions. So when it comes to the next reprediction, pseudo-labels will be more confident according to sharper predictions.

Finally, repredicting pseudo-labels frequently is harmful for performance. By using the R2 strategy every epoch, the network predictions and pseudo-labels are always the same and D2 cannot optimize pseudo-labels anymore. In CIFAR-10 experiments, we repredict pseudo-labels every 75 epochs and reduce the learning rate after each reprediction. Figure 10 shows that using the R2 strategy can make pseudo-labels more confident at the end of training.

### 3.4 The overall R2-D2 algorithm

Now we propose the overall R2-D2 algorithm. The training can be divided into three stages. In the first stage, we only use labeled images to train the backbone network with cross entropy loss as in common network training. In the second stage, we use the backbone network trained in the first stage to predict pseudo-labels for unlabeled images. Then we use D2 to train the network and optimize pseudo-labels together. It is expected that this stage can boost the network performance and make pseudo-labels more precise. But according to our analysis, it is not enough to train D2 by only one stage. With the R2 strategy, D2 will be repredicted and trained for several times. In the third stage, the backbone network is finetuned by all images whose labels come from the second stage. For unlabeled images, we pick the class which has the maximum value in pseudo-labels and use the cross entropy loss to train the network. And pseudo-labels are not updated anymore. For labeled images, we use their groundtruth labels.

In general, R2-D2 is a simple method. It requires only one single network (versus two in Mean Teacher) and the loss function consists of two terms (versus three in Mean Teacher). The training processes in different stages are identical (share the same code), just need to change the value of two switch variables.

## 4 Experiments

In this section, we use four datasets to evaluate our algorithm: ImageNet (Russakovsky et al., 2015), CIFAR-100 (Krizhevsky and Hinton, 2009), CIFAR-10 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011). We first use an ablation study to investigate the impact of the R2 strategy. We then report the results on these datasets to compare with state-of-the-arts. We also conduct experiments that use R2-D2 to finetune other SSL methods. At last, we evaluate R2-D2 under the realistic setting. All experiments were implemented using the PyTorch (Paszke et al., 2019) framework and run on a computer with TITAN Xp GPU.

### 4.1 Implementation details

Note that we trained the network using stochastic gradient descent with Nesterov momentum 0.9 in all experiments. We set $\alpha = 0.1$, $\beta = 0.03$ and $\lambda = 4000$ on *all* datasets, which shows the robustness of our method to these hyperparameters. Other hyperparameters (e.g., batch size, learning rate, and weight decay) were set according to different datasets.

**ImageNet** is a large-scale dataset with natural color images from 1000 categories. Each category typically has 1300 images for training and 50 for evaluation. Following the prior work (Qiao et al., 2018; Sajjadi et al., 2016; Pu et al., 2016; Tarvainen and Valpola, 2017), we uniformly choose 10% data from training images as labeled data. That means there are 128 labeled data for each category. The rest of training images are considered as unlabeled data. We test our model on the validation set. The backbone network is ResNet-18 (He et al., 2016a). The data augmentation we used is the same as that of Tarvainen and Valpola (2017), which includes random rotation, random resized crop to $224 \times 224$, random horizontal flip and color jittering.

In the first stage, we trained ResNet-18 (He et al., 2016a) on 4 GPUs with the labeled data. We trained for 60 epochs with the weight decay of $5 \times 10^{-5}$. Because the labeled dataset only contains 128000 images, the batch size was set as 160 to make the parameters update more times. The learning rate was 0.1 at the beginning and decreased by cosine annealing (Loshchilov and Hutter, 2017) so that it would reach 0 after 75 epochs.

In the second stage, we trained for 60 epochs on 4 GPUs. We set the batch size as 800, 200 of which were labeled. The learning rate was 0.12 and did not change in this stage. During this stage, we found that pseudo-labels would be more accurate. Note that the capacity of ResNet-18 is small and it is hard for ResNet-18 to remember all pseudo-labels. To make pseudo-labels more accurate when repredicting, we finetuned the model using the dataset with pseudo-label. We finetuned for 60 epochs with initial learning rate 0.12 and decayed it with cosine annealing (Loshchilov and Hutter, 2017) so that is would reach 0 after 65 epochs.

Repeating the second stage, we used the network at the end of last stage to repridict the pseudo-labels of unlabeled images. Then we trained the network and optimize pseudo-labels for 30 epochs with learning rate 0.04. Other settings were the same as the second stage.

In the third stage, we used the pseudo-labels at the end of last stage to finetune the model. We finetuned for 60 epochs with initial learning rate 0.04 and decayed it with cosine annealing (Loshchilov and Hutter, 2017) so that is would reach 0 after 65 epochs.

**CIFAR-100** contains $32 \times 32$ natural images from 100 categories. There are 50000 training images and 10000 testing images in CIFAR-100. Following Laine and Aila (2017); Qiao et al. (2018); Iscen et al. (2019), we use 10000 images (100 per class) as labeled data and the rest 40000 as unlabeled data. We report the error rates on the testing images. The backbone network is ConvLarge (Laine and Aila, 2017).

The data augmentation contained random translations, random horizontal flip and cutout (DeVries and Taylor, 2017).

In the first stage, we trained the ConvLarge network on 1 GPU with 10000 labeled images. To make the parameters update more times, we set the batch size as 20 and trained the network for 300 epochs. So the parameters of the network could update 500 times per epoch and update 150000 times totally in this stage. The initial learning rate was 0.05 and decreased by cosine annealing (Loshchilov and Hutter, 2017) so that it would reach 0 after 350 epochs. The weight decay was set as 0.0002.

In the second stage, we optimized the network and pseudo-labels for 300 epochs on 4 GPUs. The batch size was 512, in which 128 images were labeled and others were unlabeled. The learning rate was 0.04 and did not change in this stage.

Repeating the second stage, we repredicted pseudo-labels at 0, 75, 150, 225 epoch. After each reprediction, we optimized the network and pseudo-labels for 75 epochs. The learning rate were set as 0.04, 0.03, 0.02, 0.01, respectively. Other settings were the same as the second stage.

In the third stage, we finetuned the network for 50 epochs with batch size 512. The learning rate was 0.01 at the beginning and decreased by cosine annealing (Loshchilov and Hutter, 2017) so that it would reach 0 at the end.

**CIFAR-10** contains $32 \times 32$ natural images from 10 categories. Following Laine and Aila (2017); Miyato et al. (2018); Tarvainen and Valpola (2017); Qiao et al. (2018); Robert et al. (2018), we use 4000 images (400 per class) from 50000 training images as labeled data and the rest images as unlabeled data. We report the error rates on the full 10000 testing images. The backbone network is Shake-Shake (Gastaldi, 2017). The data augmentation contained random translations, random horizontal flip and cutout (DeVries and Taylor, 2017).

All the settings were the same with that of CIFAR-100 except the learning rate and batch size. In the first stage, we trained the Shake-Shake network on 1 GPU with 4000 labeled images. We set the batch size as 40 and trained the network for 300 epochs. The initial learning rate was 0.05 and decreased by cosine annealing (Loshchilov and Hutter, 2017) so that it would reach 0 after 350 epochs. The weight decay was set as 0.0002.

In the second stage, we optimized the network and pseudo-labels for 300 epochs on 4 GPUs. The batch size was 512, in which 128 images were labeled and others were unlabeled. The learning rate was 0.12 and did not change in this stage.

Repeating the second stage, we repredicted pseudo-labels at 0, 75, 150, 225 epoch. After each reprediction, we optimized the network and pseudo-labels for 75 epochs. The learning rate were set as 0.12, 0.08, 0.04, 0.004, respectively. Other settings were the same as the second stage.

In the third stage, we finetuned the network for 50 epochs with batch size 512. The learning rate was 0.01 at the beginning and decreased by cosine annealing (Loshchilov and Hutter, 2017) so that it would reach 0 at the end.

Table 1: Ablation studies when using different strategies to train our end-to-end framework on CIFAR-10. ($\alpha = 0.1, \beta = 0.03, \lambda = 4000$)

| | a | b | c | d | e |
|---|---|---|---|---|---|
| The 2nd stage | ✓ | ✓ | ✓ | ✓ | ✓ |
| Repeat the 2nd stage | | ✓ | ✓ | ✓ | ✓ |
| Reprediction | | | ✓ | | ✓ |
| Reducing LR | | | | ✓ | ✓ |
| Error rates (%) | 6.71 | 6.37 | 6.23 | 5.94 | 5.78 |

Table 2: Ablation studies when using different $\alpha$ to train our end-to-end framework on CIFAR-10. ($\beta = 0.03, \lambda = 4000$)

| $\alpha$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Error rates (%) | 5.78 | 5.44 | 5.81 | 5.90 | 6.11 |

Table 3: Ablation studies when using different $\beta$ to train our end-to-end framework on CIFAR-10. ($\alpha = 0.1, \lambda = 4000$)

| $\beta$ | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
|---|---|---|---|---|---|
| Error rates (%) | 5.62 | 5.75 | 5.78 | 5.83 | 5.76 |

**SVHN** dataset consists of $32 \times 32$ house number images belonging to 10 classes. The category of each image is the centermost digit. There are 73257 training images and 26032 testing images in SVHN. Following Laine and Aila (2017); Tarvainen and Valpola (2017); Miyato et al. (2018); Qiao et al. (2018), we use 1000 images (100 per class) as labeled data and the rest 72257 training images as unlabeled data. The backbone network is ConvLarge (Laine and Aila, 2017). The data augmentation consists of adding gaussian noise to images like Laine and Aila (2017); Tarvainen and Valpola (2017) and cutout (DeVries and Taylor, 2017).

The settings of learning rates and weight decay were the same as that of our training strategy for CIFAR-10. In the first stage, we trained the ConvLarge (Laine and Aila, 2017) network on 1 GPU for 180 epochs with batch size 10. In the second stage, the batch size was set as 512, in which 128 images were labeled. The network was trained for 180 epochs. Repeating the second stage, pseudo-labels were repredicted at 0, 45, 90, 135 epoch. In the third stage, we finetuned the network for 180 epochs.

## 4.2 Ablation studies

Now we validate our framework by an ablation study on CIFAR-10 with the Shake-Shake backbone and 4000 labeled images. All experiments used the same data splits and ran once. And they all used the first stage to initialize D2 and

Table 6: Error rates (%) on the validation set of ImageNet benchmark with 10% images labeled. "-" means that the original papers did not report the corresponding error rates. ResNet-18 is used.

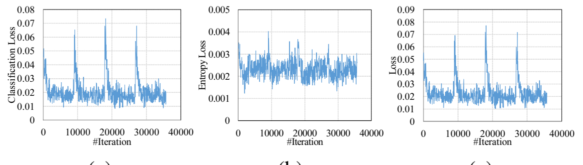
| | Method | Backbone | #Param | Top-1 | Top-5 |
|---|---|---|---|---|---|
| Supervised | 100% Supervised | ResNet-18 | 11.6M | 30.43 | 10.76 |
| | 10% Supervised | ResNet-18 | 11.6M | 52.23 | 27.54 |
| Semi-supervised | Stochastic Transformations | AlexNet | 61.1M | - | 39.84 |
| | VAE with 10% Supervised | Customized | 30.6M | 51.59 | 35.24 |
| | Mean Teacher | ResNet-18 | 11.6M | 49.07 | 23.59 |
| | Dual-View Deep Co-Training | ResNet-18 | 11.6M | 46.50 | 22.73 |
| | R2-D2 | ResNet-18 | 11.6M | **41.55** | **19.52** |
| Self-supervised + Semi-supervised | RotNet + R2-D2 | ResNet-18 | 11.6M | **40.54** | **18.76** |

Table 4: Ablation studies when using different $\lambda$ to train our end-to-end framework on CIFAR-10. ($\alpha = 0.1, \beta = 0.03$)

| $\lambda$ | 1000 | 2000 | 3000 | 4000 | 5000 |
|---|---|---|---|---|---|
| Error rates (%) | 5.85 | 5.85 | 5.82 | 5.78 | 5.53 |

Table 5: Ablation studies when using different $\mathscr{L}_c$ to train our end-to-end framework on CIFAR-10. ($\alpha = 0.1, \beta = 0.03, \lambda = 4000$)

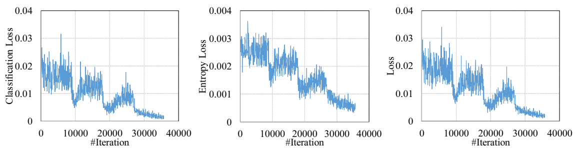
| $\mathscr{L}_c$ | $KL(\hat{\mathbf{p}}\|\|\tilde{\mathbf{p}})$ | $KL(\tilde{\mathbf{p}}\|\|\hat{\mathbf{p}})$ | $\|\|\tilde{\mathbf{p}} - \hat{\mathbf{p}}\|\|_2^2$ |
|---|---|---|---|
| Error rates (%) | 5.78 | 8.06 | 6.35 |



(a) (b) (c)

Figure 11: (a), (b), (c) show how $\mathscr{L}_c$, $\mathscr{L}_e$, $\mathscr{L}$ change by training D2 with only reprediction on CIFAR-10 (column c in Table 1), respectively. Reprediction occurs at 8750, 17500, and 26250 iterations. Without reducing the learning rate, the loss cannot decrease and network prediction keep flat. So pseudo-labels will not become sharp.

the third stage to finetune the network. Table 1 presents the results and the error rates are produced by the last epoch of the third stage. Different columns denote using different strategies to train D2 in the second stage. First, without R2 (column a), the error rate of a basic D2 learning is 6.71%, which is already competitive with state-of-the-arts. Next, we repeated the second stage without reprediction or reducing learning rate (column b). That means the network is trained by the first stage, the second stage, repeat the second stage, and the third stage. This network achieved a 6.37% error rate, which demonstrates training D2 for more epochs can boost performance and the network will not overfit easily. Repeating the second stage with reprediction (column c) could make the error rate even lower, to 6.23%. But, without reducing the learning rate, $\mathscr{L}$ did not decrease (cf. Figure 11c). On the other hand, repeating the second stage and reducing the learning rate (column d) can get better results (5.94%). However, only reducing the learning rate cannot remove the impact of the equality constraint bias. At last, applying both strategies (column e) improved the results by a large margin to 5.78%.



(a) (b) (c)

Figure 12: (a), (b), (c) show how $\mathscr{L}_c$, $\mathscr{L}_e$, $\mathscr{L}$ change by training D2 with only reducing learning rate on CIFAR-10 (column d in Table 1), respectively. Reducing learning rate occurs at 8750, 17500, and 26250 iterations. It can make loss decrease. But the learning algorithm is still impacted by the equality condition bias.

Table 2 presents the results with different $\alpha$. $\beta$ is 0.03 in all experiments. We find setting $\alpha = 0.2$ will achieve a better performance and a large $\alpha$ may degrade the performance. Table 3 shows the results with different $\beta$ when setting $\alpha = 0.1$. Compared with $\alpha$, our method is robust to $\beta$. The highest error rate is 5.83% and the lowest error rate is 5.62%. There is roughly 0.2% between them. Table 4 studies the sensitiveness of our method under different $\lambda$ which is the learning rate for updating pseudo-labels (cf. Equation 19). Intuitively, the pseudo-labels can hardly be updated by a small $\lambda$. And with a large $\lambda$, the pseudo-labels will always be the same as the predictions and thus the training will fail. In practice, we find our method is not sensitive to $\lambda$. With $\lambda = 1000$, the error rate is 5.85%, only slightly worse than $\lambda = 4000$. When setting $\lambda = 5000$, the performance is even better. Overall, R2-D2 is robust to these hyperparameters. And when apply R2-D2, we suggest that $\alpha = 0.1, \beta = 0.03$ and $\lambda = 4000$ is a safe starting point to tune these hyperparameters. All experiments in the rest of our paper used $\alpha = 0.1, \beta = 0.03, \lambda = 4000$. Please note that we did not carefully tune these hyperparameters. Error rates of R2-D2 may be lower than those reported in this paper if we tune them carefully.

Table 5 shows the results with different $\mathscr{L}_c$. Note that our loss function is defined as $\mathscr{L} = \alpha \mathscr{L}_c + \beta \mathscr{L}_e$. The loss function determines how the network parameters and pseudo-labels update. That means different $\mathscr{L}_c$ result in different updating formulas of pseudo-labels. The default $\mathscr{L}_c$ is $KL(\hat{\mathbf{p}}||\tilde{\mathbf{p}})$ and the updating formula is Equation 19. When set $\mathscr{L}_c = KL(\tilde{\mathbf{p}}||\hat{\mathbf{p}})$, the gradients of $\mathscr{L}$ with respect to $\tilde{y}_n$ is

$$\frac{\partial \mathscr{L}}{\partial \tilde{y}_n} = \alpha \tilde{p}_n [\log \tilde{p}_n - \log \hat{p}_n - \sum_{k=1}^{N} \tilde{p}_k (\log \tilde{p}_k - \log \hat{p}_k)] , \quad (28)$$

where $\hat{\mathbf{p}} = \sigma(\hat{\mathbf{y}})$ and $\tilde{\mathbf{p}} = \sigma(\tilde{\mathbf{y}})$. With $\mathscr{L}_c = ||\tilde{\mathbf{p}} - \hat{\mathbf{p}}||_2^2$, the gradients of $\mathscr{L}$ with respect to $\tilde{y}_n$ is

$$\frac{\partial \mathscr{L}}{\partial \tilde{y}_n} = 2\alpha \tilde{p}_n [\tilde{p}_n - \hat{p}_n - \sum_{k=1}^{N} \tilde{p}_k (\tilde{p}_k - \hat{p}_k)] . \quad (29)$$

Note that due to the sigmoid transform, $\tilde{\mathbf{p}}$ and $\hat{\mathbf{p}}$ are much smaller than $\tilde{\mathbf{y}}$, so all of them need a large $\lambda$ to update pseudo-labels. The experimental results demonstrate superior performance of R2-D2 with $\mathscr{L}_c = KL(\hat{\mathbf{p}}||\tilde{\mathbf{p}})$. It obtains 2.28% lower error rate than $KL(\tilde{\mathbf{p}}||\hat{\mathbf{p}})$ and 0.57% lower error rate than $||\tilde{\mathbf{p}} - \hat{\mathbf{p}}||_2^2$.

### 4.3 Results on ImageNet

Table 6 shows our results on ImageNet with 10% labeled samples. The setup followed that in Qiao et al. (2018). The image size in training and testing is $224 \times 224$. For the fairness of comparisons, the error rate is from single model without ensembling. We use the result of the last epoch. Our experiment is repeated three times with different random subsets of labeled training samples. The Top-1 error rates are 41.64, 41.35, and 41.65, respectively. The Top-5 error rates are 19.53, 19.60, and 19.44, respectively. R2-D2 achieves significantly lower error rates than Stochastic Transformations (Sajjadi et al., 2016) and VAE (Pu et al., 2016), although they used the larger input size $256 \times 256$. With the same backbone and input size, R2-D2 obtains roughly 5% lower Top-1 error rate than that of DCT (Qiao et al., 2018) and 7.5% lower Top-1 error rate than that of Mean Teacher (Tarvainen and Valpola, 2017). R2-D2 outperforms the previous state-of-the-arts by a large margin. The performances of Mean Teacher (Tarvainen and Valpola, 2017) with ResNet-18 (He et al., 2016a) is quoted from Qiao et al. (2018).

Self-supervised learning is another way to utilize unlabeled data. In self-supervised learning, it needs to define a pretext task to train the network. By solving the pretext task, we expect the network can learn better representations. And with the better representations, the network finetuned by a few labeled data can get a better performance than training it from scratch. Recently, RotNet (Gidaris et al., 2018) is a simple and promising self-supervised learning technique. RotNet uses recognizing the image rotation as the pretext task. We can combine R2-D2 with RotNet. First, we train the network by recognizing the image rotation ($0°$, $90°$, $180°$, $270°$) with all images (labeled images and unlabeled images). Then, we replace its FC layer by 1000-class weights of random initialization and use R2-D2 to train the network. Table 6 shows the results and using RotNet pretrained weight can improve roughly 1% without bells and whistles.

Table 7 shows our results with the ResNet-50 backbone network. The setup is the same as that of ResNet-18. And our experiment was run for once. ResNet-50 denotes the regular type (He et al., 2016a) and ResNet-50v2 denotes the pre-activation variants (He et al., 2016b). The results of Pseudo-label, VAT, VAT + EntMin, S$^4$L-Rotation, S$^4$L-Exemplar are quoted from Zhai et al. (2019). And R2-D2 is significantly better than them. Note that Zhai et al. (2019) proposed MOAM (Mix Of All Models) and got a better performance. However, they used a $4\times$ wider model as backbone network and it is not fair to compare ours with MOAM's results.

### 4.4 Results on CIFAR-100

Table 8 presents experimental results on CIFAR-100 with 10000 labeled samples. All methods used ConvLarge for fairness of comparisons and did not use ensembling. The error rate of R2-D2 is the average error rate of the last epoch over five random data splits. The results of 100% Supervised is quoted from Laine and Aila (2017). Using 10000 labeled images achieved 38.36% error rates in our experiments. With unlabeled images, R2-D2 produced a 32.87% error rate which is lower than others (e.g., Temporal Ensembling, LP (Iscen et al., 2019), Mean Teacher (Tarvainen and Valpola, 2017), LP + Mean Teacher (Iscen et al., 2019), and DCT). The performances of Mean Teacher (Tarvainen and Valpola, 2017) is quoted from Iscen et al. (2019).

### 4.5 Results on CIFAR-10

We evaluated the performance of R2-D2 on CIFAR-10 with 4000 labeled samples. Table 9 presents the results. Following Tarvainen and Valpola (2017); Robert et al. (2018), we used the Shake-Shake network (Gastaldi, 2017) as the backbone network. Overall, using Shake-Shake backbone network can achieves lower error rates than using ConvLarge. Our experiment was repeated five times with different random subsets of labeled training samples. We used the test error rates of the last epoch. After the first stage, the backbone network produced the error rates 14.90%, which is our baseline

Table 7: Error rates (%) on the validation set of ImageNet benchmark with 10% images labeled. "-" means that the original papers did not report the corresponding error rates. ResNet-50 is used.

| | Method | Backbone | Top-1 | Top-5 |
|---|---|---|---|---|
| Supervised | 100% Supervised | ResNet-50 | 23.75 | 7.23 |
| | 10% Supervised | ResNet-50 | 45.55 | 20.73 |
| Semi-supervised | Pseudo-label | ResNet-50v2 | - | 17.59 |
| | VAT | ResNet-50v2 | - | 17.22 |
| | VAT + EntMin | ResNet-50v2 | - | 16.61 |
| | $S^4$L-Rotation | ResNet-50v2 | - | 16.18 |
| | $S^4$L-Exemplar | ResNet-50v2 | - | 16.28 |
| | R2-D2 | ResNet-50 | **34.01** | **14.07** |

Table 8: Error rates (%) on CIFAR-100 benchmark with 10000 images labeled.

| | Method | Backbone | Error rates (%) |
|---|---|---|---|
| Supervised | 100% Supervised | ConvLarge | $26.42 \pm 0.17$ |
| | Using 10000 labeled images only | ConvLarge | $38.36 \pm 0.27$ |
| Semi-supervised | Temporal Ensembling | ConvLarge | $38.65 \pm 0.51$ |
| | LP | ConvLarge | $38.43 \pm 1.88$ |
| | Mean Teacher | ConvLarge | $36.08 \pm 0.51$ |
| | LP + Mean Teacher | ConvLarge | $35.92 \pm 0.47$ |
| | DCT | ConvLarge | $34.63 \pm 0.14$ |
| | R2-D2 | ConvLarge | $\mathbf{32.87 \pm 0.51}$ |

Table 9: Error rates (%) on CIFAR-10 benchmark with 4000 images labeled.

| Method | Backbone | Error rates (%) |
|---|---|---|
| 100% Supervised | Shake-Shake | 2.86 |
| Only 4000 labeled images | Shake-Shake | $14.90 \pm 0.28$ |
| Mean Teacher | ConvLarge | $12.31 \pm 0.28$ |
| Temporal Ensembling | ConvLarge | $12.16 \pm 0.24$ |
| VAT+EntMin | ConvLarge | $10.55 \pm 0.05$ |
| DCT with 8 Views | ConvLarge | $8.35 \pm 0.06$ |
| Mean Teacher | Shake-Shake | $6.28 \pm 0.15$ |
| HybridNet | Shake-Shake | 6.09 |
| R2-D2 | Shake-Shake | $\mathbf{5.72 \pm 0.06}$ |

Table 10: Error rates (%) on SVHN benchmark with 1000 images labeled.

| Method | Backbone | Error rates (%) |
|---|---|---|
| 100% Supervised | ConvLarge | $2.88 \pm 0.03$ |
| Only 1000 labeled images | ConvLarge | $11.27 \pm 0.85$ |
| Temporal Ensembling | ConvLarge | $4.42 \pm 0.16$ |
| VAdD (KL) | ConvLarge | $4.16 \pm 0.08$ |
| Mean Teacher | ConvLarge | $3.95 \pm 0.19$ |
| VAT+EntMin | ConvLarge | $3.86 \pm 0.11$ |
| VAdD (KL) + VAT | ConvLarge | $3.55 \pm 0.05$ |
| DCT with 8 Views | ConvLarge | $\mathbf{3.29 \pm 0.03}$ |
| R2-D2 | ConvLarge | $3.64 \pm 0.20$ |

using 4000 labeled samples. With the help of unlabeled images, R2-D2 obtains an error rate of 5.72%.Compared with Mean Teacher (Tarvainen and Valpola, 2017) and HybridNet (Robert et al., 2018), R2-D2 achieves lower error rate and produces state-of-the-art results.

## 4.6 Results on SVHN

We tested R2-D2 on SVHN with 1000 labeled samples. The results are shown in Table 10. Following previous works (Laine and Aila, 2017; Tarvainen and Valpola, 2017; Miyato et al., 2018; Qiao et al., 2018), we used the ConvLarge network as the backbone network. The result we report is average error rate of the last epoch over five random data splits. On this task, the gap between 100% supervised and many SSL methods (e.g., VAT+EntMin (Miyato et al., 2018), VAdD (KL)+VAT (Park et al., 2018), Deep Co-Training (Qiao et al., 2018), and R2-D2) is less than 1%. Only Deep Co-Training with 8 Views (Qiao et al., 2018) and VAdD (KL)+VAT slightly outperform R2-D2. Compared with other methods (e.g., Temporal Ensembling, Mean Teacher, and VAT, R2-D2 produces a lower error rate. Note that on the large-scale ImageNet, R2-D2 significantly outperformed Deep Co-Training. VAdD have not be evaluated on ImageNet in their paper.

## 4.7 Combine R2-D2 with other SSL method

Now, we study if R2-D2 can boost other SSL methods' performance. Note that the overall R2-D2 algorithm consists of three stages. In the raw first stage, we only use the labeled im-

Table 11: Error rates (%) on CIFAR-10 benchmark with 4000 images labeled. † denotes results reported in the original papers. MT means Mean Teacher.

| Method | Backbone | Error rates (%) |
|---|---|---|
| MT + fast-SWA (1200)† | ConvLarge | 9.05 |
| MT + fast-SWA (1200) | ConvLarge | 9.70 |
| MT + fast-SWA (1200) + R2-D2 | ConvLarge | 9.27 |
| MT + SWA (1200)† | ConvLarge | 9.38 |
| MT + SWA (1200) | ConvLarge | 9.37 |
| MT + SWA (1200) + R2-D2 | ConvLarge | 9.11 |

Table 12: Error rates (%) on CIFAR-10 benchmark with 4000 labeled images and balanced/unbalanced unlabeled images. All expriments use the Shake-Shake backbone network.

| Unlabeled data | Error rates (%) | |
|---|---|---|
| | Mean Teacher | R2-D2 |
| 46000 balanced | 6.60 | 5.72 |
| 23000 balanced | 9.28 | 8.08 |
| 23000 unbalanced | 9.72 | 9.52 |

Table 13: Error rates (%) on CIFAR-10 benchmark with 4000 labeled images and open world assumption. All expriments use the Shake-Shake backbone network.

| Unlabeled data | | Error rates (%) | |
|---|---|---|---|
| CIFAR-10 | CIFAR-100 | Mean Teacher | R2-D2 |
| 46000 balanced | 0 | 6.60 | 5.72 |
| 46000 balanced | 5000 | 7.00 | 6.41 |
| 23000 balanced | 0 | 9.28 | 8.08 |
| 23000 balanced | 12000 | 9.68 | 8.99 |
| 23000 unbalanced | 0 | 9.72 | 9.52 |
| 23000 unbalanced | 12000 | 10.85 | 10.41 |
| 23000 balanced | 23000 | 28.66 | 15.32 |
| 23000 unbalanced | 23000 | 28.15 | 18.05 |

ages to train the backbone network. Combining R2-D2 with other SSL method, we can use other SSL method as the first stage in our algorithm. That means we use aother SSL method to train the network with labeled and unlabeled images. Then, we use the trained network to initialize our framework and continue to train the network by R2-D2. Table 11 presents the results and R2-D2 indeed boosts other SSL method performance. Our implementation of MT + fast-SWA (1200) (Athiwaratkun et al., 2019) achieve 9.70% error rate. And with the help of R2-D2, the error rate is 9.27% which is 0.43% lower. Combining R2-D2 with MT + SWA (1200) (Athiwaratkun et al., 2019) results in 9.11% error rate which is better than 9.37% of MT + SWA (1200).

### 4.8 Realistic evaluation of R2-D2

In this section, we evaluate R2-D2 under more realistic experiment setting. As Oliver et al. (2018) pointed out, in “real-world”, the unlabeled data may be unbalanced and even contain a different distribution of classes than the labeled data. First, we study the sensitiveness of our method when trained with unbalanced unlabeled data. Table 12 shows the results. “46000 balanced” means the typical setting that is using 4000 labeled data and 46000 unlabeled data of CIFAR-10. “23000 balanced” denotes using 23000 balanced unlabel data (2300 per class). At last, we produce 23000 unbalanced unlabeled data by random sampling. Each class contains 2770, 3452, 2042, 4062, 4047, 758, 590, 2588, 2201, 490 images, respectively. According to the experimental results, R2-D2 and Mean Teacher are more sensitive to the number of unlabeled data. When using a half but balanced unlabeled data, the performances are degraded by 2.36% and 2.68%, respectively. However, the gaps of error rates between “23000 balanced” and “23000 unbalanced” are only 1.44% and 0.44%, respectively.

Finally, we devise the experiments to simulate the situation that the unlabeled data contain a different distribution of classes than the labeled data. And we call it “open world assumption”. Because the classes of CIFAR-100 are different from that of CIFAR-10, we select some images in CIFAR-100 to add to the unlabeled data. Table 13 presents the results. The model performance can often be significantly degraded when adding CIFAR-100 images. Because we predict the pseudo-labels of unlabeled data repetitively, we can use the entropy of pseudo-labels to estimate if the unlabeled images belong to CIFAR-10. An effective remedy is to throw away 10% unlabeled data whose pseudo-label entropy are larger than others after each reprediction. With this remedy, R2-D2 achieves better performance than Mean Teacher. However, when adding 23000 CIFAR-100 images, the error rates of both methods are higher than that of only using labeled data. It is still an open problem to make sure the network indeed benefit from the unlabeled data whose distribution is different from the labeled data.

## 5 Conclusion

In this paper, we proposed R2-D2, a method for semi-supervised deep learning. D2 uses label probability distributions as pseudo-labels for unlabeled images and optimizes them during training. Unlike previous SSL methods, D2 is an end-to-end framework, which is independent of the backbone network and can be trained by back-propagation. Based on D2, we give a theoretical support for using network predictions as pseudo-labels. However, pseudo-labels will become flat during training. We analyzed this problem both theoretically and experimentally, and proposed the R2 remedy for it. At last, we tested R2-D2 on different datasets. The experiments demonstrated superior performance of our proposed methods. On large-scale dataset ImageNet, R2-D2 achieved about 5% lower error rates than that of previous state-of-the-art. In the future, we will further explore the combination of unsupervised feature learning and semi-supervised learning, and deep SSL in the open world assumption.

## References

Athiwaratkun B, Finzi M, Izmailov P, Wilson AG (2019) There are many consistent explanations of unlabeled data: Why you should average. In: The International Conference on Learning Representations (ICLR), pp 1–22

DeVries T, Taylor GW (2017) Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:170804552

Gao BB, Xing C, Xie CW, Wu J, Geng X (2017) Deep label distribution learning with label ambiguity. IEEE Transactions on Image Processing 26(6):2825–2838

Gastaldi X (2017) Shake-shake regularization of 3-branch residual networks. In: The International Conference on Learning Representations (ICLR), Workshop Track Proceedings, pp 1–5

Gidaris S, Singh P, Komodakis N (2018) Unsupervised representation learning by predicting image rotations. In: The International Conference on Learning Representations (ICLR), pp 1–16

He K, Zhang X, Ren S, Sun J (2016a) Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778

He K, Zhang X, Ren S, Sun J (2016b) Identity mappings in deep residual networks. In: The European Conference on Computer Vision (ECCV), LNCS, vol 9908, Springer, pp 630–645

Iscen A, Tolias G, Avrithis Y, Chum O (2019) Label propagation for deep semi-supervised learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 5070–5079

Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Tech. rep., University of Toronto

Laine S, Aila T (2017) Temporal ensembling for semi-supervised learning. In: The International Conference on Learning Representations (ICLR), pp 1–13

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324

Lee DH (2013) Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on Challenges in Representation Learning, ICML, vol 3, p 2

Liu Y, Song G, Shao J, Jin X, Wang X (2018) Transductive centroid projection for semi-supervised large-scale recognition. In: The European Conference on Computer Vision (ECCV), LNCS, vol 11209, Springer, pp 72–89

Loshchilov I, Hutter F (2017) SGDR: Stochastic gradient descent with warm restarts. In: The International Conference on Learning Representations (ICLR), pp 1–16

Lu Z, Peng Y (2013) Exhaustive and efficient constraint propagation: A graph-based learning approach and its applications. International Journal of Computer Vision 103(3):306–325

Miyato T, Maeda Si, Ishii S, Koyama M (2018) Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence pp 1979–1993

Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning

Oliver A, Odena A, Raffel CA, Cubuk ED, Goodfellow I (2018) Realistic evaluation of deep semi-supervised learning algorithms. In: Advances in Neural Information Processing Systems 31, pp 3235–3246

Park S, Park J, Shin SJ, Moon IC (2018) Adversarial dropout for supervised and semi-supervised learning. In: Thirty-Second AAAI Conference on Artificial Intelligence, pp 3917–3924

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp 8024–8035

Pu Y, Gan Z, Henao R, Yuan X, Li C, Stevens A, Carin L (2016) Variational autoencoder for deep learning of images, labels and captions. In: Advances in Neural Information Processing Systems 29, pp 2352–2360

Qiao S, Shen W, Zhang Z, Wang B, Yuille A (2018) Deep co-training for semi-supervised image recognition. In: The European Conference on Computer Vision (ECCV), LNCS, vol 11219, Springer, pp 142–159

Robert T, Thome N, Cord M (2018) HybridNet: Classification and reconstruction cooperation for semi-supervised learning. In: The European Conference on Computer Vision (ECCV), LNCS, vol 11211, Springer, pp 158–175

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. International Journal of Computer Vision 115(3):211–252

Sajjadi M, Javanmardi M, Tasdizen T (2016) Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In: Advances in Neural Information Processing Systems 29, pp 1163–1171

Tanaka D, Ikami D, Yamasaki T, Aizawa K (2018) Joint optimization framework for learning with noisy labels. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 5552–5560

Tarvainen A, Valpola H (2017) Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Advances in Neural Information Processing Systems 30, pp 1195–1204

Wang GH, Wu J (2020) Repetitive reprediction deep decipher for semi-supervised learning. In: Thirty-Fourth AAAI Conference on Artificial Intelligence, p in press

Weston J, Ratle F, Mobahi H, Collobert R (2012) Deep learning via semi-supervised embedding. In: Montavon G, Orr GB, Müller KR (eds) Neural Networks: Tricks of the Trade: Second Edition, Springer, pp 639–655

Yi K, Wu J (2019) Probabilistic end-to-end noise correction for learning with noisy labels. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 7017–7025

Zhai X, Oliver A, Kolesnikov A, Beyer L (2019) S4L: Self-supervised semi-supervised learning. In: The IEEE International Conference on Computer Vision (ICCV), pp 1476–1485

Zhu X, Ghahramani Z (2002) Learning from labeled and unlabeled data with label propagation. Tech. Rep. CMU-CALD-02-107, Carnegie Mellon University