# Time-Consistent Self-Supervision for Semi-Supervised Learning

**Tianyi Zhou** [*1] **Shengjie Wang** [*1] **Jeff A. Bilmes** [1]

## Abstract

Semi-supervised learning (SSL) leverages unlabeled data when training a model with insufficient labeled data. A common strategy for SSL is to enforce the consistency of model outputs between similar samples, e.g., neighbors or data augmentations of the same sample. However, model outputs can vary dramatically on unlabeled data over different training stages, e.g., when using large learning rates. This can introduce harmful noises and inconsistent objectives over time that may lead to concept drift and catastrophic forgetting. In this paper, we study the dynamics of neural net outputs in SSL and show that selecting and using first the unlabeled samples with more consistent outputs over the course of training (i.e., "*time-consistency*") can improve the final test accuracy and save computation. Under the time-consistent data selection, we design an SSL objective composed of two self-supervised losses, i.e., a consistency loss between a sample and its augmentation, and a contrastive loss encouraging different samples to have different outputs. Our approach achieves SOTA on several SSL benchmarks with much fewer computations.

## 1 Introduction

Semi-supervised learning (SSL) tackles a challenging problem commonly encountered in real-world applications, i.e., how one should train a reliable machine learning model with limited amounts of labeled samples but a rich reserve of unlabeled samples. For many tasks, collecting accurate labeled data is expensive and error-prone as it requires human labor, while unlabeled data is cheap and abundant. Commonly used methodology in the early days of SSL includes label and measure propagation (Zhu & Ghahramani, 2002; Zhou et al., 2003; Subramanya & Bilmes, 2011) and manifold regularization (Niyogi, 2013), which encourage the label consistency within a local neighborhood on the embedded data manifold (i.e., the model should produce similar label distributions for samples geodesically close to each other), and where the manifold is approximated by a graph. This encourages the model to be locally smooth on the manifold.

Recent deep learning shows that a trained neural net not only works as an accurate classifier but also provides hidden-layer features that can capture manifold structure at multiple scales (Verma et al., 2019a; Zeiler & Fergus, 2014) Moreover, with prior knowledge of data and inductive bias embedded in the neural net architecture and training strategies, e.g., convolutions and data augmentations (Zhang et al., 2018; DeVries & Taylor, 2017; Cubuk et al., 2019a;b), we can train a more locally consistent model with better generalization performance. This motivates recent progress on self-supervision (Raina et al., 2007; Trinh et al., 2019; Zhang et al., 2016; Noroozi & Favaro, 2016) and SSL (Sajjadi et al., 2016; Iscen et al., 2019; Miyato et al., 2019; Athiwaratkun et al., 2019; Laine & Aila, 2017; Berthelot et al., 2019; Oliver et al., 2018), in which a self-supervised loss (which does not require ground-truth labels) defined on data augmentations can encourage a neural network's local smoothness on the data manifold. For example, the loss can encourage that a sample and its augmentations should have similar outputs, or different samples (and their augmentations) should have different outputs. The former is usually called "consistency regularization" (Rasmus et al., 2015), while the latter is regarded as a "contrastive loss" (Chopra et al., 2005) or a "triplet loss" (Weinberger & Saul, 2009). Neural networks trained by this strategy have achieved substantial improvements on SSL (Sajjadi et al., 2016; Berthelot et al., 2020) and unsupervised learning (Caron et al., 2018; He et al., 2019) tasks without leveraging the weighed neighborhood graphs commonly used in earlier works. More recently, MixMatch (Berthelot et al., 2019) and ReMixMatch (Berthelot et al., 2020) utilize a consistency loss with data augmentations and, with a delicate combination of regularization techniques, gets significantly improved SSL performance on several datasets, thus demonstrating the effectiveness of combining data augmentation and self-supervision.

The self-supervised loss defined on data augmentations can be explained as an ordinary supervised loss with a pseudo target (Lee, 2013; Miyato et al., 2019) generated by the neu-

[*]Equal contribution [1]University of Washington, Seattle. Correspondence to: Tianyi Zhou <tianyizh@uw.edu>, Shengjie Wang <wangsj@uw.edu>, Jeff A. Bilmes <bilmes@uw.edu>.

*Proceedings of the $37^{th}$ International Conference on Machine Learning*, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

ral net itself or a self-ensemble (Tarvainen & Valpola, 2017; Laine & Aila, 2017). For a consistency loss, the pseudo target for an unlabeled sample can be the class distribution of the sample's random augmentation(s) generated by the neural net. For a contrastive loss, we can use softmax to normalize the similarities between a sample and a subset of other samples (including one of its own augmentations). The pseudo target for the softmax output has value 1 for the sample's augmentation and 0 for the other samples (van den Oord et al., 2018). Although pseudo targets achieved on data augmentations tend to be accurate, the above techniques cannot guarantee time-consistent pseudo targets for unlabeled samples. At different training stages, the output of a neural net for a sample can change dramatically due to the non-smoothness of activation functions, the complexity of network structures, and sophisticated learning rate schedules (Smith, 2017; Loshchilov & Hutter, 2017; Li & Arora, 2019). In such a case, when we use self-supervised losses with varying pseudo targets for the same sample, the optimization objectives may keep changing over training steps. This inconsistency of the objective over time can considerably slow down the training progress or even make it diverge, resulting in serious concept drift and training failure.

Another common strategy for SSL is to select samples whose predictions have high confidence. However, the confidence can still change drastically over time for the same aforementioned reason. Moreover, a neural network can often simultaneously overconfident and incorrect on training samples. Indeed, our later empirical analysis shows that confidence is an unreliable metric for selecting samples. In addition, samples with high confidence tend to be less informative to the future training since the gradient is proportional to the gap between current outputs and the pseudo target, which is small if the confidence is high.

In this paper, we study the dynamics of neural net outputs during SSL. We analyze possible catastrophic forgetting on labeled data caused by adding an unlabeled sample to training, from which we derive a "time-consistency (TC)" metric $c^t(x)$ for an individual sample $x$ at training step[1] $t$ that can be used to select informative unlabeled data with more time-consistent pseudo targets in self-supervision. Specifically, $c^t(x)$ is an negative exponential moving average of $a^t(x)$ (defined below) over training history before $t$:

$$a^t(x) \triangleq D_{KL}(p^{t-1}(x)||p^t(x)) + \left|\log \frac{p^{t-1}_{y^{t-1}(x)}(x)}{p^t_{y^{t-1}(x)}(x)}\right|, \quad (1)$$

where $D_{KL}(\cdot||\cdot)$ is the Kullback–Leibler divergence, $p^t(x)$ and $y^t(x) \triangleq \operatorname{argmax}_y p^t_y(x)$ are the output distribution over classes and the predicted class label of $x$ at step $t$. Intuitively, the KL-divergence between output distributions measures how consistent the output is between two consecutive steps, while the log odds ratio for the predicted class $y^{t-1}(x)$ measures the change of confidence on the predicted class $y^{t-1}(x)$. A moving average of $a^t(x)$ naturally captures inconsistency over time quantify, based on the history, how much change will occur to the learning objective when selecting $x$ and its pseudo target for future training.

We then present an empirical study of TC for unlabeled data selection and compare this with using confidence. We artificially split a fully labeled dataset into labeled and unlabeled subsets, and train a neural net by only using the labeled set. We observe that the model tends to produce consistent outputs on some unlabeled samples but frequently changes its prediction on others. We further find that the time-consistent outputs (the former) are usually correct predictions while the latter contains more mistakes. The observations lead us to a natural curriculum (Bengio et al., 2009; Zhou & Bilmes, 2018) for SSL that selects unlabeled samples with higher TC at each training step while gradually increasing the selection budget. We then introduce "TC-SSL" that adopts the curriculum to train a neural net by minimizing the consistency loss and contrastive loss defined on augmentations of unlabeled samples (using learned policies of AutoAugment (Cubuk et al., 2019a)). In experiments, we show that TC-SSL outperforms the very recent MixMatch and other SSL approaches on three datasets (CIFAR10, CIFAR100, and STL10) under various labeled-unlabeled splittings and significantly improves SSL efficiency, i.e., consistently using $< 20\%$ training batches of what the best baseline needs. These results portend well for TC-SSL as a modern SSL methodology.

### 1.1 Related Work

Classic SSL methods enforce samples to have similar label distributions in every local region of the data manifold — this approach is taken by label/measure propagation (Zhu & Ghahramani, 2002; Zhu et al., 2003; Zhou et al., 2003; Subramanya & Bilmes, 2011; Bengio et al., 2006; Iscen et al., 2019) and manifold regularization (Niyogi, 2013; Belkin & Niyogi, 2002). They rely on similarity measures between samples to find the local neighborhood for every sample. One can think of these methods as encouragements of "spatial consistency". It is not always clear, however, how to determine sample pair similarity for a given data set and model. E.g., which feature space do we compute the similarity in? What similarity metric performs best? Recently, graph neural net (GNN) based methods (Kipf & Welling, 2017; Veličković et al., 2018; Verma et al., 2019b) can apply graph-based SSL on neural net architectures and gets compelling performance on SSL of graph data. The data in these works consists of natural graph structures, while on arbitrary data, generating the graph is itself a challenging problem.

Recent SSL research encourages spatial consistency by combining data augmentation techniques and self-supervised

[1] We use superscripts to index the training step in this paper.

losses. In (Rasmus et al., 2015; Sajjadi et al., 2016), the authors propose "consistency regularization" to encourage the sample and its augmentations to have similar neural net outputs. "Contrastive loss" (Chopra et al., 2005; van den Oord et al., 2018) and "triplet loss" (Weinberger & Saul, 2009; Schroff et al., 2015), on the contrary, encourage different samples (and their augmentations) to have distant outputs. MixMatch (Berthelot et al., 2019) combines several techniques, i.e., consistency loss + sharpening the pseudo target distribution averaged over multiple augmentations + MixUp (Zhang et al., 2018), which significantly improve SSL performance on several datasets. Compared to graph-based SSL, it integrates multiple augmentation methods (Berthelot et al., 2020; Cubuk et al., 2019a;b) to provide stronger self-supervision. Our work also adopts a similar strategy but additionally enforces the time-consistency of selected samples at every SSL step, which is an orthogonal technique that can be seamlessly integrated into MixMatch.

## 2 Time-Consistency of Unlabeled Sample

We define the time-consistency (TC) of a sample $x$ at step $t$ as an exponential moving average of $-a^t(x)$ over time, i.e.,

$$c^t(x) = \gamma_c(-a^t(x)) + (1-\gamma_c)c^{t-1}(x) \tag{2}$$

where $\gamma_c \in [0,1]$ is a discount factor. We negate $a^t(x)$ so that larger $c^t(x)$ means better time-consistency. For classification tasks, we show that the changes in labeled samples' losses are bounded by $a^t(x)$ on unlabeled data. In other words, choosing time-consistent unlabeled samples mitigates catastrophic forgetting of labeled sample.

We denote the labeled data set as $\mathcal{L}$, an unlabeled data set as $\mathcal{U}$, and an unlabeled sample as $x' \in \mathcal{U}$. Let $f^t(x)$ be the final layer output (before softmax) on sample $x$ with network parameters $\theta^t$ at step $t$ of training, so $p^t(x) = \text{softmax}(f^t(x))$. Let $\boldsymbol{y}(x)$ be the one-hot label vector for sample $x$, and $\ell(x;\theta^t)$ be the cross entropy loss between the class label and the softmax output of network with parameter $\theta^t$. We consider two cases: (1) we train the network using only the labeled samples as a gradient step, i.e., $\theta^{t+1} = \theta^t + \eta\sum_{x\in\mathcal{L}}\nabla_\theta\ell(x;\theta^t)$, where $\eta$ is the learning rate; and (2) we add an unlabeled sample $x'$ to the gradient step, i.e., $\hat{\theta}^{t+1} = \theta^t + \eta(\sum_{x\in\mathcal{L}}\nabla_\theta\ell(x;\theta^t) + \nabla_\theta\ell(x';\theta^t))$. In (2), when calculating $\ell(x';\theta^t)$, we use a one-hot label $\boldsymbol{y}^t(x')$ that has value one in position $y^t(x')$ (recall $y^t(x') = \text{argmax}_j\, p^t_j(x')$, where $p^t_j(x')$ is class-$j$'s probability in distribution $p^t(x')$) and value zero elsewhere — which is a "winner take all" or a "pseudo" target. The Taylor expansion of labeled sample loss $\ell(x,\theta)$ defined on $\theta_a$ and evaluated at $\theta_b$ is:

$$g_{\theta_a}(\theta_b) = \Big[\sum\nolimits_{x\in\mathcal{L}}\ell(x;\theta_a) + \nabla_\theta\ell(x;\theta_a)(\theta_b-\theta_a)\Big] + o((\theta_b-\theta_a)^2) \tag{3}$$

We can measure the forgetting effect of adding $x' \in \mathcal{U}$ to the training set by looking at the changes in loss over labeled samples $x \in \mathcal{L}$. Ideally, $x'$ should not cause vital changes to the loss over labeled samples, which should remain small. If $\theta_a$ is close to $\theta_b$, it is reasonable to omit the second and higher order terms of the Taylor expansion in Eq. (3). Doing so, we calculate the change of loss for labeled data by

$$\begin{aligned}
&\frac{1}{\eta}\left|\sum_{x\in\mathcal{L}}\Big[\ell(x;\theta^{t+1}) - \ell(x;\hat{\theta}^{t+1})\Big]\right| \\
=&\frac{1}{\eta}\left|g_{\theta^t}(\theta^{t+1}) - g_{\theta^t}(\hat{\theta}^{t+1})\right| \approx \left|\nabla_\theta\ell(x';\theta^t)\sum_{x\in\mathcal{L}}\nabla_\theta\ell(x;\theta^t)\right| \\
\approx&\left|\frac{\partial\ell(x';\theta^t)}{\partial\theta^t}\frac{\partial\theta^t}{\partial t}\right| = \left|\frac{\partial\ell(x';\theta^t)}{\partial t}\right| = \left|\frac{\partial\ell(x';\theta^t)}{\partial f^t(x')}\frac{\partial f^t(x')}{\partial t}\right| \\
=&\left|(\boldsymbol{y}^t(x') - p^t(x'))\frac{\partial f^t(x')}{\partial t}\right|
\end{aligned} \tag{4}$$

If the learning rate is not too large, we may use the difference $(f^{t+1}(x') - f^t(x'))$ to approximate $\partial f^t(x')/\partial t$. After some algebra (details given in Appendix B), we can bound the changes in loss for labeled samples by $a^t(x')$:

$$\begin{aligned}
&\left|(\boldsymbol{y}^t(x') - p^t(x'))\frac{\partial f^t(x')}{\partial t}\right| \\
\approx&\left|(\boldsymbol{y}^t(x') - p^t(x'))(f^{t+1}(x') - f^t(x'))\right| \\
\le& D_{KL}(p^t(x')||p^{t+1}(x')) + \left|\log\frac{p^{t+1}_{y^t(x')}(x')}{p^t_{y^t(x')}(x')}\right| = a^t(x').
\end{aligned} \tag{5}$$

Therefore, by selecting an unlabeled sample $x'$ with high TC $c^t(x')$ (a smoothed estimate of $-a^t(x')$ using exponential moving average to eliminates noise), we will not suffer from a rapid surge on the labeled sample loss if $\sum_{x\in\mathcal{L}}\ell(x;\theta^t) \approx 0$ at step $t$. Thereby, $x'$ and its pseudo target will not result in catastrophic forgetting of learned samples. On the other hand, for $x'$ itself, we expect its pseudo target and objective to be consistent after being added to training. A large $c^t(x')$ indicates that both $p^t(x')$ and the confidence on pseudo-class $y^t(x')$ vary little over training history. Note the above analysis of forgetting effect on labeled data also extends and can be recursively applied to the unlabeled data that have been correctly predicted and selected for training. Hence, selecting unlabeled samples with large TC can avoid catastrophic forgetting of learned data.

### 2.1 Empirical Evidence

We empirically investigate the quality of unlabeled samples selected by the time-consistency metric. Ideally, in SSL training, we expect the selected samples to have pseudo targets no different from the unknown ground truths. On CIFAR-10, we randomly select 15000 samples for training and use the remaining 35000 samples for validation, where the latter is considered as unlabeled data in the SSL setting. We train a WideResNet-28-2 (Zagoruyko & Komodakis, 2016) (28 layers, width factor of 2, 1.5-million parameters) on the 15000 samples by fully supervised training, and

inspect the 35000 samples to check if the time-consistency metric can effectively prune out the samples with wrong predictions on the model while selecting samples with the right labels. We compare time-consistency against confidence as the metric to select unlabeled samples in Figures 1,2 and 3. More empirical studies with different parameters and on CIFAR-100 can be found in Appendix A, which show patterns similar to what we present here.

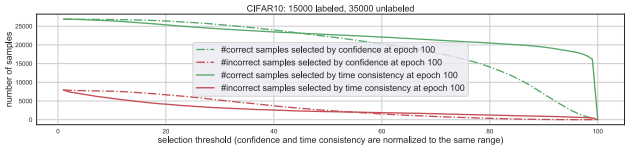


*Figure 1.* Computed time-consistency and confidence at epoch 100. The x-axis shows the validation samples selected using different thresholds on the two metrics (normalized to $[0, 100]$). The y-axis reports correct v.s. incorrect predictions over the selected samples.

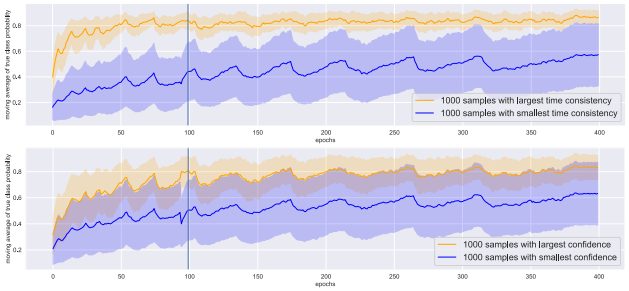


*Figure 2.* Computed time-consistency (top) and confidence (bottom) at epoch 100. Select the top 1000 and bottom 1000 validation samples based on the two metrics. Compare the moving average of true class probability of the selected samples across epochs.

We report the number of correct/incorrect predictions over the validation samples selected by various thresholds on TC in Figure 1 and compare it with confidence $\max_y p(y|x)$, i.e., the highest probability among classes. We normalize TC and confidence values to be in $[0, 100]$ and so, the threshold set to 20 means that we select the validation samples whose TC is greater than 20% of the maximum TC value. At high thresholds, time-consistency can more effectively identify the correctly predicted samples than confidence.

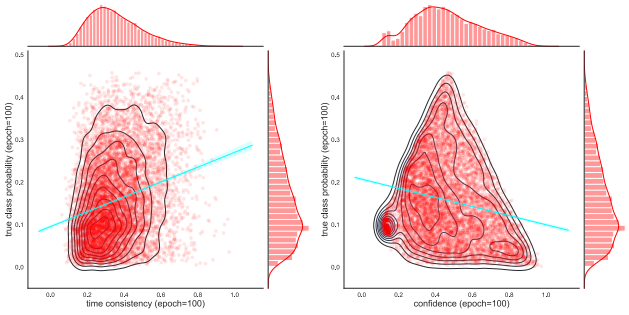


*Figure 3.* Scatter plot and correlation (cyan lines) of the true class's probability (y-axis) for incorrectly predicted validation samples with TC (**left**) or confidence (**right**) on the x-axis at epoch 100.

In Figure 2, we select the top 1000 and bottom 1000 samples in the validation set based on the two criteria computed at epoch 100 and report the performance on those samples across training epochs. Compared to confidence, the performance on the top 1000 samples selected by TC is significantly better than the bottom 1000 ones across most epochs. This again suggests that TC is a more powerful criteria for identifying correctly-predicted samples. It also shows that TC can be predictive for future training dynamics.

In Figure 3, we plot all incorrectly predicted samples in the validation set based on their performance and selection criteria (either TC or confidence) computed at epoch 100. Comparing the two, we find many fewer samples with high TC that have low true-class probability than we find using the confidence (see the lower-right regions in Figure 3). This suggests that TC could potentially prune incorrectly-predicted samples more effectively.

## 3 Time-Consistent SSL

In this section, we will first introduce the training objective of TC-SSL at each step and explain the importance of selecting time-consistent samples to the training stability when using this objective. We then present the TC-SSL algorithm, which selects samples by their TC according to a curriculum that selects more unlabeled samples as training progresses. We then discuss some practical modifications to TC-SSL that bring further improvements.

### 3.1 Self-supervised loss and their time-consistency

We use two types of self-supervised losses which cooperate with each other to encourage the output consistency between similar samples on the data manifold. In particular, given a sample $x$ and one of its augmentation $G(x)$, we minimize the consistency loss defined as the difference between $f(x)$ and $f(G(x))$, i.e., the neural net outputs on these two samples. In order to obtain a pseudo target robust to unbounded noise over training steps, we follow the mean teacher (Tarvainen & Valpola, 2017) method and instead use an exponential moving average of the neural net in-training to generate the pseudo target, i.e.,

$$\overline{f^t}(x) \triangleq f(x; \overline{\theta^t}), \overline{\theta^t} \triangleq \gamma_\theta \theta^{t-1} + (1 - \gamma_\theta)\overline{\theta^{t-1}}, \quad (6)$$

where $\theta^t$ is the neural net parameters at step $t$ and $\overline{\theta^t}$ is recursively defined as the moving average of $\theta^t$ over time with discount factor $\gamma_\theta \in [0, 1]$. Although $\overline{\theta^t}$ tends to produce a smooth target over time, it cannot filter out time-inconsistent samples whose outputs change dramatically over time and their smoothed targets still suffer from large variance and low entropy. The consistency loss on $x$ is defined as

$$\ell^t_{cs}(x; \theta) \triangleq \|f(G(x); \theta) - \overline{f^t}(x)\|_p, \quad (7)$$

where we set $p = 2$ in our experiments and the above loss aims to minimize the $\ell_2$ distance between the two outputs. In order to stabilize back-propagation on $\ell^t_{cs}(x; \theta)$, $\overline{f^t}(x)$

is often treated as a constant pseudo target (Tarvainen & Valpola, 2017) (i.e., no back-propagation through $\overline{f^t}(x)$ to $x$) of $f(G(x))$. Hence, we only minimize the loss w.r.t. the current $\theta^t$. In practice, we can further consider replacing the pseudo target $\overline{f^t}(x)$ with an average over different augmentations and multiple time steps, e.g., $1/hm \sum_{t=T-h}^{T} \sum_{i=1}^{m} \overline{f^t}(G_i(x))$, which might be more accurate (Berthelot et al., 2020). For simplicity and efficiency, we do not adopt this form and use $\overline{f^t}(x)$ in our experiments.

Contrastive loss (Chopra et al., 2005; van den Oord et al., 2018) enforces the distance between augmentations of the same sample to be smaller than distances between other samples (or other samples' augmentations). Specifically, we use InfoNCE loss (van den Oord et al., 2018) defined as follows: given a dictionary $D$ of "other" data and/or their augmentations to compare with, InfoNCE loss is

$$\ell_{ct}^t(x;\theta) \triangleq -\log \frac{\exp(\cos[f(G(x);\theta), \overline{f^t}(G'(x))])}{\sum_{z\in\{G'(x)\}\cup D} \exp(\cos[f(G(x);\theta), \overline{f^t}(z)])}, \quad (8)$$

where $G(x)$ and $G'(x)$ can be either two different augmentations of $x$ or two distinct instantiations of the same random augmentation policy applied to $x$, and $\cos[x,z] \triangleq \frac{\langle x,z\rangle}{\|x\|_2 \cdot \|z\|_2}$ denotes the cosine similarity.

The contrastive loss above can be explained as a cross entropy loss, where the predicted probability distribution (computed by applying softmax to the $|D|+1$ similarities in the denominator) is defined over all the $|D|+1$ samples that measures the probability of each sample being similar to $G(x)$, and the pseudo label is 1 for the same sample's augmentation $G'(x)$ and 0 for other samples. Similar to consistency loss, all the quantities with $\overline{f^t}(\cdot)$ in Eq. 8 are treated as constants during back-propagation. Recent work (He et al., 2019) shows that minimizing the contrastive loss requires a sufficiently large dictionary size $|D|$ to achieve improvements. In our experiments, $|D|$ is often set $\geq 1000$. During training, we keep a fixed-length queue of samples' outputs in memory as the dictionary $D$ and replace the oldest outputs in the queue by the latest training batch's outputs.

These two types of self-supervised losses constitute only one part of our SSL optimization objective. The consistency and contrastive losses provide two complementary self-supervision strategies, one maximizing the absolute similarity between similar samples, while the other enforcing that the relative similarity between similar samples should be much larger than the one between distant samples. When applied together with data augmentation (which can cover a larger local region around each sample on the data manifold) and expressive neural nets (which encodes the prior of data structure and enjoys universal approximation capability), they can effectively learn a locally consistent and smooth model over the data manifold, without requiring ground truth labels for most samples. Nor does the method use similarity between all the samples (as most earlier works do), also such graph-based methods could be incorporated in our work possibly further improving performance. To best of our knowledge, TC-SSL is the first attempt to combine consistent and contrastive losses together for SSL.

The two self-supervised losses enforce local consistency but not time-consistency. For a sample or its augmentation, its pseudo targets in the loss (i.e., all the terms containing $\overline{f^t}(\cdot)$ in Eq. (7) and Eq. (8)) at different training steps us time-variant and can be inconsistent, since the neural net outputs on the sample or its nearby region may change drastically during training. There are several possible reasons for such instability on individual samples: a non-smooth activation function, a complicated neural net structure, a steep learning rate schedule, or too complicated a data augmentation strategy. When the training targets significantly change over time, the learning could be very inefficient or even diverge and suffer from concept drift. Therefore, selecting time-consistent samples for more consistent self-supervision is essential to the widely used strategy of combining data augmentation with self-supervised losses.

For a labeled sample $(x, y)$ with ground truth class $y$, we minimize their cross entropy loss, i.e.,

$$\ell_{ce}(x, y; \theta) \triangleq -\log \frac{\exp(f(G(x);\theta)[y])}{\sum_{y'} \exp(f(G(x);\theta)[y'])}, \quad (9)$$

In order to minimize the entropy of the outputs for unlabeled data (Grandvalet & Bengio, 2005; Lee, 2013), we also minimize a cross entropy loss with $y^{t-1}(x)$ as the pseudo label of each selected unlabeled data $x$, i.e.,

$$\ell_{ce}^t(x;\theta) \triangleq -\log \frac{\exp(f(G(x);\theta)[y^{t-1}(x)])}{\sum_{y'} \exp(f(G(x);\theta)[y'])}, \quad (10)$$

The overall training objective of TC-SSL at step $t$ is

$$L^t(\theta) = \frac{1}{|\mathcal{L}|} \times \sum_{(x,y)\in\mathcal{L}} \ell_{ce}(x, y; \theta) + \frac{1}{|S^t|} \times \sum_{x\in S^t} \left[\lambda_{cs}\ell_{cs}^t(x;\theta) + \lambda_{ct}\ell_{ct}^t(x;\theta) + \lambda_{ce}\ell_{ce}^t(x;\theta)\right], \quad (11)$$

where $\mathcal{L}$ denotes the set of all labeled data and $S^t \subseteq \mathcal{U}$ is the subset of unlabeled data selected by TC at step $t$ from the unlabeled data set $\mathcal{U}$, and $\lambda_{cs}$, $\lambda_{ct}$, and $\lambda_{ce}$ are weights for different types of loss that can be tuned on a validation set.

### 3.2 The TC-SSL algorithm

We provide the complete description of TC-SSL in Algorithm 1. At step $t$, TC-SSL aims to select a subset of unlabeled data $S^t$ with higher TC $c^t(x)$ and then minimizes the objective in Eq. (11) w.r.t. the neural net weights $\theta^t$ by taking gradient steps. We can either select the top-$k$ unlabeled samples with the largest $c^t(x)$ as $S^t$ (line-8) or apply a weighted sampling of $S^t$ according to $\Pr(x \in S^t) \propto \exp(c^t(x))$ (line-9). The optimization of $L^t(\theta^t)$ can be one

or several steps of gradient descent or SGD (line-10). We denote the update of $\theta^t$ produced by the adopted optimizer as $\pi(\nabla_\theta L^t(\theta^t), \eta^t)$, where $\eta^t$ contains all hyperparameters of the optimizer at step $t$, e.g., the learning rate. We keep updating TC $c^t(x)$ for every unlabeled data $x \in \mathcal{U}$ (line-18) as a (negative) moving average of $a^t(x)$ (line-16), and update a moving average of $\theta^t$ over time (line-19) as the mean teacher (Tarvainen & Valpola, 2017) providing pseudo targets for the two self-supervised losses in $L^t(\theta^t)$.

**Algorithm 1** Time-Consistent SSL (TC-SSL)

1: **input:** $\mathcal{U}, \mathcal{L}, \pi(\cdot;\eta), \eta^{1:T}, f(\cdot;\theta), G(\cdot)$;
2: **hyperparameters:** $T_0, T, \lambda_{cs}, \lambda_{ct}, \lambda_{ce}, \gamma_\theta, \gamma_c, \gamma_k$;
3: **initialize:** $\theta^0, k^1$;
4: **for** $t \in \{1, \cdots, T\}$ **do**
5: **if** $t \le T_0$ **then**
6: $\theta^t \leftarrow \theta^{t-1} + \pi\left(\sum_{(x,y)\in\mathcal{L}} \nabla_\theta \ell_{ce}(x, y; \theta^{t-1}); \eta^t\right)$
7: **else**
8: $S^t = \text{argmax}_{S:S\subseteq\mathcal{U},|S|=k^t} \sum_{x\in S} c^t(x)$ or
9: Draw $k^t$ samples from $\Pr(x \in S^t) \propto \exp(c^t(x))$;
10: $\theta^t \leftarrow \theta^{t-1} + \pi\left(\nabla_\theta L^t(\theta^{t-1}); \eta^t\right)$ (ref. Eq. (11));
11: **end if**
12: $p^t(x) \leftarrow \frac{\exp(f(x;\theta^t)[y])}{\sum_{y'=1}^{C}\exp(f(x;\theta^t)[y'])},\ \forall y \in [C], x \in \mathcal{U}$;
13: **if** $t = 1$ **then**
14: $\overline{\theta^t} \leftarrow \theta^t, c^t(x) \leftarrow 0,\ \forall x \in \mathcal{U}$
15: **else**
16: Compute $a^t(x)$ (ref. Eq (1)), $\forall x \in \mathcal{U}$;
17: **end if**
18: $c^{t+1}(x) \leftarrow \gamma_c(-a^t(x)) + (1-\gamma_c)c^{t-1}(x),\ \forall x \in \mathcal{U}$;
19: $\overline{\theta^{t+1}} \leftarrow \gamma_\theta\theta^t + (1-\gamma_\theta)\overline{\theta^t}$;
20: $k^{t+1} \leftarrow (1+\gamma_k) \times k^t$;
21: **end for**

Since the TC $c^t(x)$ is an accumulated statistic over multiple time steps, we need a sufficiently long horizon to estimate it accurately and stably before using. Moreover, the change of neural net outputs on a sample in a single step can be easily perturbed by SGD randomness, the learning rate change, and neural net weight changes during early training. Therefore, we apply $T_0$ warm starting epochs (line-6) that train the neural net using only the labeled data before selecting any unlabeled data for self-supervision. In addition, there may be fewer time-consistent unlabeled samples in earlier stages without sufficient training. As more training occurs with consistent targets, the neural nets' accuracy and confidence improve, and we expect an increase in the number of time-consistent samples. Hence, over the course of training, we gradually increase the number of unlabeled samples selected into $S^t$ (line-20). This yields a curriculum for SSL that in early stages effective saves computation and avoids perturbation caused by inconsistent unlabeled data, and in later stages fully explores the information brought by the reserve of unlabeled data.

In MixMatch (Berthelot et al., 2019) and MixUp (Zhang et al., 2018), augmentation over unlabeled and labeled data can significantly improve SSL performance. One possible reason is that minimizing the self-supervised loss on linear interpolations between two samples enforces the local consistency and smoothness over larger regions of the data manifold. In TC-SSL, after single-sample augmentation (if any), we apply standard MixUp between all training samples in $\mathcal{U} \cup \mathcal{L}$, but only use the resulted MixUp samples in cross entropy loss, i.e., Eq. (9) and Eq. (10). Unlike MixMatch, we do not need to modify MixUp and simply treat each unlabeled sample as labeled with a pseudo class when mixing its target with another (labeled or unlabeled) sample. For the two self-supervised losses, we still use the single-sample augmentations before MixUp because: (1) the consistency loss already uses a soft pseudo label but MixUp further reduces its entropy; and (2) the contrastive loss aims to discriminate different samples but replacing them with their interpolations will weaken the effect.

### 3.3 Practical aspects that further improve TC-SSL

**Calibrate the time-consistency by the learning rate:** Since the learning rate $\eta^t$ can change over time, and large learning rates lead to greater changes on model outputs, the scales of $a_t(x)$ might change accordingly. Hence, we calibrate $a_t(x)$ (via $a^t(x) \leftarrow a^t(x)/\eta^t$) before using it.

**Exponential weight for exploration-exploitation trade-off:** During early training, the weighted sampling in line-9 of Algorithm 1 usually works better than selecting the top-$k^t$ samples (line-8) since the randomness avoids selecting the same samples repeatedly and encourages exploration of new unlabeled data, whose TC might be improved once being trained. In practice, we can achieve a better trade-off between exploration and exploitation by using similar techniques to Exp3 (Auer et al., 2003), i.e., instead of applying line-8 or line-9, we sample $S^t$ according to a smoothed probability $\Pr(x \in S^t) \propto \exp(\sqrt{2\log|\mathcal{U}|/|\mathcal{U}|} \times c^t(x))$ and then re-scale $a^t(x)$ for all selected $x \in S^t$ by their probability of being selected, i.e., $a^t(x) \leftarrow a^t(x)/\Pr(x \in S^t)$. Thereby, $x$ with smaller probability will get more of a chance to be selected in the future.

**Decrease the entropy of pseudo target:** Following (Grandvalet & Bengio, 2005; Miyato et al., 2019; Berthelot et al., 2019; Lee, 2013), we also decrease the entropy of pseudo target distribution in the contrastive loss by multiplying a temperature parameter $\nu > 1$ [2] to the quantities inside the exponential in Eq. (8).

**Prune $S^t$ using the confidence statistics on a validation set:** Although TC is a better metric than confidence for unlabeled data selection, confidence is still not useless. For

[2] e.g., we use $\nu = 10$ in all experiments.

instance, it is not efficient to train models with unlabeled samples of either extremely high or low confidence since the former contributes very small gradients while the latter's pseudo target suffers from high entropy. Therefore, we remove them from $S^t$ by applying two thresholds computed on a validation set, which are the $b\%$-percentile and $(100-b)\%$-percentile of the confidence computed on the validation set [3]. We do not use the statistics on training set $\mathcal{U} \cup \mathcal{L}$ since the model can be over-confident on samples being trained.

**Duplicate labeled samples:** When applying TC-SSL with MixUp, we duplicate the labeled samples so that an unlabeled sample has higher a chance of being mixed with a labeled sample. This improves training robustness since mixing the incorrect pseudo target of an unlabeled sample with the correct label of a labeled sample is less harmful. However, unlike MixMatch, we do not duplicate the labeled samples to the same amount of unlabeled samples. Instead, we start using fewer duplicates [4] than MixMatch — this amount is comparable to the amount of unlabeled data selected in the first iteration, and we gradually reduce the duplicates as training proceeds. Thanks to the TC metric, the selected samples are likely to be correct with stable pseudo targets and we do not need to spend much compute on the labeled data to keep training robust and stable.

# 4 Experiments

In this section, we apply TC-SSL to train Wide ResNet (Zagoruyko & Komodakis, 2016) models on different labeled-unlabeled splittings of three datasets, i.e., CIFAR10, CIFAR100 (Krizhevsky & Hinton, 2009), and STL10 (Coates et al., 2011). Unless otherwise specified, we followed all the settings and train the same neural net architectures from previous works such as MixMatch (Berthelot et al., 2019) and mean teacher (Tarvainen & Valpola, 2017) to make sure that the comparison with previously published results on these datasets are fair. For CIFAR10 experiments, we train a small WideResNet-28-2 (28 layers, width factor of 2, 1.5-million parameters) and a large WideResNet-28-135 (28 layers, 135 filters per layer, 26-million parameters) for four kinds of labeled/unlabeled/validation random splittings applied to the original training set of CIFAR10, i.e., 500/44500/5000, 1000/44000/5000, 2000/43000/5000, 4000/41000/5000. For CIFAR100 experiments, we train WideResNet-28-135 for three splittings, i.e., 2500/42500/5000, 5000/40000/5000, 10000/35000/5000. For STL10 experiment, we train a deeper variant of WideResNet-28-2 with four extra residual blocks added before the output layer with the purpose of processing larger images in STL10 (compared with CIFAR images). This extra part as a whole has 256 input channels and 512 output channels and increases the total parameters from 1.5-million to 5.9-million, which however is still much smaller than the model used in MixMatch (23.8-million parameters) (Berthelot et al., 2019). We randomly draw 500 samples from the original training set as our validation set, so the splitting is 4500/100000/500. We evaluate the performance of all the trained models on the original test set of each dataset.

[3] e.g., we use $b = 10$ in experiments

[4] e.g., we use $k^1 = |S^1|$, which is $0.1|\mathcal{U}|$ in all experiments.

## 4.1 Hyperparameters

For TC-SSL in the experiments, we apply $T_0 = 10$ warm starting epochs and $T = 680$ epochs in total. Note the "epoch" here refers to one iteration in Algorithm 1 and is different from its meaning in most fully supervised training, where it refers to a full pass of the whole training set. In our case, the training samples in each epoch changes according to our curriculum of $k^t$. We apply SGD with momentum of $0.9$ and weight decay of $2 \times 10^{-5}$, and use a modified cosine annealing learning rate schedule (Loshchilov & Hutter, 2017) for multiple episodes of increasing length and decaying target learning rate, since it can quickly jump between different local minima on the loss landscape and explore more regions without being trapped by a bad local minima. In particular, we set up $12$ episodes with epochs-per-episode starting from $10$ (i.e., the warm starting episode) and increasing by $10$ after every episode until reaching epoch-$680$. The learning rate at the beginning and end of the first episode are set to $0.2$ and $0.02$, respectively. We then multiply each of them by $0.9$ after every episode. We do not heavily tune the $\lambda$-parameters and $\gamma$-parameters. For all experiments, we use $\lambda_{cs} = {}^{20}/_{C}, \lambda_{ct} = 0.2, \lambda_{ce} = 1.0, \gamma_\theta = \gamma_c = 0.99, \gamma_k = 0.005$ ($C$ is the number of classes). For data augmentation, we use AutoAugment (Cubuk et al., 2019a) learned policies for the three datasets followed by MixUp with the mixing weight sampled from $\text{Beta}(0.5, 0.5)$. We initialize $k^1 = 0.1|\mathcal{U}|$ and $\theta^0$ by Pytorch default initialization. We apply all the practical tips detailed in Section 3.3.

## 4.2 Test Accuracy of TC-SSL

For each experiment, we run 5 trials with different random seeds and report the mean and variance of the test accuracy in Table 1-3). More details can be found in Appendix C. TC-SSL achieves the lowest test error on most experiments compared with several recent baselines using a heavy combination of multiple advanced techniques. Most of our

*Table 3.* Test error rate of several methods training CNNs of different #params on **STL10**. Baselines: CutOut (DeVries & Taylor, 2017), IIC (Ji et al., 2019), MixMatch (Berthelot et al., 2019).

| Benchmark | STL10 |
|---|---|
| labeled/unlabeled | 5k/100k |
| CutOut (11.0M) | 12.74 |
| IIC (N/A) | 11.20 |
| MixMatch (23.8M) | 5.59 |
| TC-SSL (ours, **5.9M**) | **4.82** |

*Table 1.* Test error rate (mean±variance) of SSL methods training a small WideResNet and a large WideResNet on **CIFAR10**. Baselines: Pseudo Label (Lee, 2013), Π-model (Sajjadi et al., 2016), VAT (Miyato et al., 2019), Mean Teacher (Tarvainen & Valpola, 2017), MixMatch (Berthelot et al., 2019), ReMixMatch (Berthelot et al., 2020).

| Benchmark | CIFAR10 (small WideResNet-28-2) | | | | CIFAR10 (large WideResNet-28-135) | | | |
|---|---|---|---|---|---|---|---|---|
| labeled/unlabeled | 500/44500 | 1000/44000 | 2000/43000 | 4000/41000 | 500/44500 | 1000/44000 | 2000/43000 | 4000/41000 |
| Pseudo Label | $40.55 \pm 1.70$ | $30.91 \pm 1.73$ | $21.96 \pm 0.42$ | $16.21 \pm 0.11$ | - | - | - | - |
| Π-model | $41.82 \pm 1.52$ | $31.53 \pm 0.98$ | $23.07 \pm 0.66$ | $5.70 \pm 0.13$ | - | - | - | - |
| VAT | $26.11 \pm 1.52$ | $18.68 \pm 0.40$ | $14.40 \pm 0.15$ | $11.05 \pm 0.31$ | - | - | - | - |
| Mean Teacher | $42.01 \pm 5.86$ | $17.32 \pm 4.00$ | $12.17 \pm 0.22$ | $10.36 \pm 0.25$ | - | - | - | - |
| MixMatch | $9.65 \pm 0.94$ | $7.75 \pm 0.32$ | $7.03 \pm 0.15$ | $6.24 \pm 0.06$ | $8.44 \pm 1.04$ | $7.38 \pm 0.63$ | $6.51 \pm 0.48$ | $5.12 \pm 0.31$ |
| ReMixMatch | - | $\mathbf{5.73 \pm 0.16}$ | - | $5.14 \pm \mathbf{0.04}$ | - | - | - | - |
| TC-SSL (ours) | $\mathbf{9.14 \pm 0.88}$ | $6.15 \pm 0.23$ | $\mathbf{5.85 \pm 0.10}$ | $\mathbf{5.07} \pm 0.05$ | $\mathbf{6.04 \pm 0.39}$ | $\mathbf{3.81 \pm 0.19}$ | $\mathbf{3.79 \pm 0.21}$ | $\mathbf{3.54 \pm 0.06}$ |

*Table 2.* Test error rate (mean±variance) of SSL methods training WideResNet-28-135 on **CIFAR100**. Baselines: SWA (Athiwaratkun et al., 2019), MixMatch (Berthelot et al., 2019).

| Benchmark | CIFAR100 (WideResNet-28-135) | | |
|---|---|---|---|
| labeled/unlabeled | 2500/42500 | 5000/40000 | 10000/35000 |
| SWA | - | - | 28.80 |
| MixMatch | $44.20 \pm 1.18$ | $34.62 \pm 0.63$ | $25.88 \pm \mathbf{0.30}$ |
| TC-SSL (ours) | $\mathbf{31.95 \pm 0.55}$ | $\mathbf{26.98 \pm 0.51}$ | $\mathbf{22.10} \pm 0.37$ |

results are SOTA, e.g., TC-SSL trains a much smaller model (5.9M parameters) and achieves the SOTA performance on STL10. Note these improvements are achieved in the case that we do not conduct any heavy tuning or fine-grained search of hyperparameters, nor use any advanced data augmentation or any bag of tricks. The only exception when TC-SSL performs worse happens at the 1000/44000 splitting of CIFAR10 when training a small WideResNet-28-2, in which case the very recent ReMixMatch achieves slightly better results than ours. However, ReMixMatch's result was achieved when using a powerful augmentation proposed in their paper, i.e., CTAugment, and averaging over multiple (i.e., 8 vs.1 in TC-SSL) augmentations to produce the pseudo targets. According to their ablation study (Berthelot et al., 2020), these two techniques bring significant improvements. In Table 4, we further present a thorough ablation study that separately verifies the improvement caused by each component of TC-SSL.

### 4.3 Training Efficiency of TC-SSL

Since the recent MixMatch achieves much better performance than the SSL methods prior to it, we present a more detailed comparison to it in Figure 4 by using the Pytorch re-implementation of MixMatch aforementioned. We use the number of mini-batches for training in the two methods as an approximate metric of their training costs since the forward and back-propagation on them dominates the incurred computation. Since TC-SSL uses batch size of 128 while MixMatch uses 64, we divide the number of training batches in MixMatch by 2. The comparison shows TC-SSL's significant advantage in learning efficiency, e.g., it achieves higher test accuracy much earlier than MixMatch. This is because that TC-SSL selects the most time-consistent unlabeled samples that are more informative and less harmful to the training, where the selection is guided by an efficient curriculum. In contrast, MixMatch uses all the unlabeled samples since the beginning, which might introduce wrong training signals and noise, especially during earlier stages, and thus slow down the growth of test accuracy.

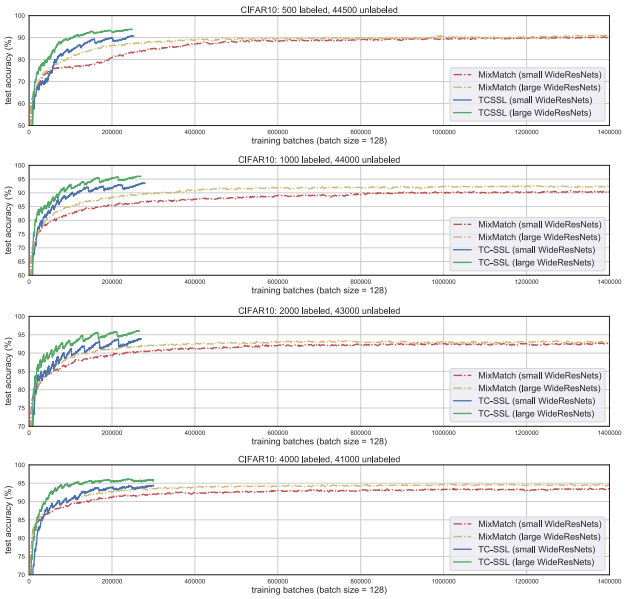


*Figure 4.* Test accuracy (%) during training small/large WideResNet by MixMatch and TC-SSL on four splittings of CIFAR10.

### 4.4 Quality of Pseudo Labels for Unlabeled Data Selected for Training in TC-SSL

One primary reason for the substantial improvements achieved by TC-SSL is that the unlabeled samples selected by time consistency have high-quality pseudo labels. To verify this, we report how the precision and recall of these selected pseudo labels together with their percentage in all the unlabeled data change during training in Figure 5.

*Table 4.* **Ablation Study:** test error rate (mean±variance) of TC-SSL variants for training WideResNet-28-135 on CIFAR10: "no consistency" removes the consistency loss in Eq. (7); "no contrastive" removes the contrastive loss in Eq. (8); "no PseudoLabel" removes the cross entropy loss for unlabeled data in Eq. (10); "no TC-selection" replaces Line 8-9 of Algorithm 1 with uniform sampling.

| labeled/unlabeled | 500/44500 | 1000/44000 | 2000/43000 | 4000/41000 |
|---|---|---|---|---|
| TC-SSL (ours) | $\mathbf{6.04 \pm 0.39}$ | $\mathbf{3.81 \pm 0.19}$ | $\mathbf{3.79 \pm 0.21}$ | $\mathbf{3.54 \pm 0.06}$ |
| TC-SSL (no consistency) | $7.51 \pm 0.56$ | $5.31 \pm 0.23$ | $3.82 \pm 0.20$ | $3.58 \pm \mathbf{0.06}$ |
| TC-SSL (no contrastive) | $7.56 \pm 0.52$ | $5.35 \pm 0.28$ | $3.96 \pm 0.25$ | $3.66 \pm 0.08$ |
| TC-SSL (no PseudoLabel) | $41.05 \pm 2.32$ | $23.64 \pm 1.17$ | $14.37 \pm 0.69$ | $9.87 \pm 0.22$ |
| TC-SSL (no TC-selection) | $12.25 \pm 0.81$ | $6.39 \pm 0.44$ | $4.68 \pm 0.35$ | $4.05 \pm 0.13$ |

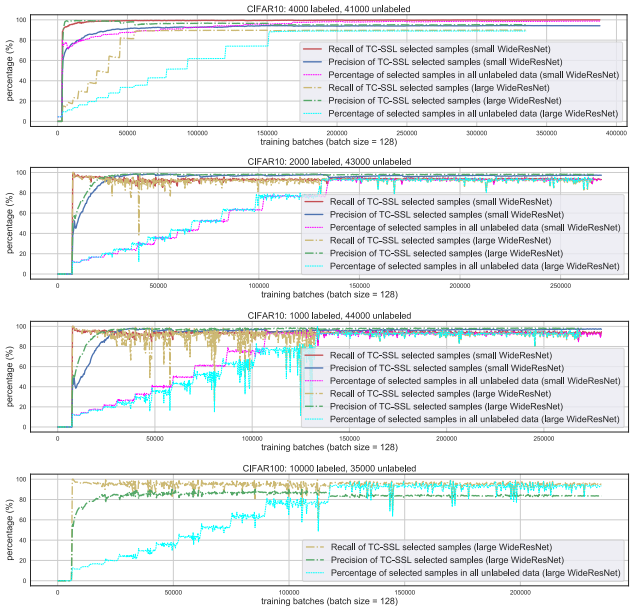


*Figure 5.* Precision and recall (%) of pseudo labels produced by the model during training for the unlabeled samples selected by TC-SSL. We also provide the percentage (%) of the selected samples in the ground set of unlabeled data $\mathcal{U}$.

In most of the four plots, the recall quickly increases to a high value close to 100% and keeps it high consistently over training, indicating that almost all the correctly-predicted unlabeled samples are selected by TC-SSL. In addition, the precision also increases quickly to a relatively high value (close to the test accuracy of fully-supervised learning that uses the whole training set), implying that most samples selected by TC-SSL have correct pseudo labels. This well explains the promising learning efficiency and final performance of TC-SSL. The noisy part (i.e., quick drops) on the recall curve when training large WideResNet is caused by the confidence thresholding mentioned in Section 3.3, which removes the extremely confident samples from $S^t$ since they provide very limited information to training (though their pseudo labels might be correct). The curves are flat during the first few training batches since they are warm-starting epochs only using labeled samples and no unlabeled data is selected. It also worth noting that the model cannot be precise on predicting all the unlabeled data in the early stages, so selecting too many unlabeled data will leads to a large portion of incorrect pseudo labels used for training, i.e., low precision. In contrast, for TC-SSL, we adopt a curriculum of gradually increasing number of selected unlabeled data, which avoid the above problem and remains a high precision on the unlabeled data selected for early training steps.

Hence, TC-SSL can maintain a precision and recall close to $100\%$ during most steps (except when it excludes some highly-confident and correctly predicted samples since they are less informative), while gradually increase the percentage of selected unlabeled samples to $100\%$.

## 5 Conclusions

We propose a metric called "time-consistency (TC)" to select unlabeled samples (with pseudo labels) for training in semi-supervised learning (SSL). We derive TC from the training dynamics of SSL and show that adding unlabeled samples with large TC to training mitigates the catastrophic forgetting on previously learned data. Moreover, TC can be efficiently computed from the byproducts of training by extremely cheap computation. We further propose a recipe of self-supervised losses and show that TC is essential to guarantee the effectiveness of self-supervision in improving SSL. Based on TC and the self-supervision recipe, we develop a curriculum learning method (TC-SSL) that incorporates more practical improvements. TC-SSL achieves state-of-the-arts performance on three SSL benckmarks and considerably improves the training efficiency. Additionally, we provide a thorough ablation study of TC-SSL and an evaluation of the quality for unlabeled samples selected in TC-SSL.

TC-SSL can be seamlessly integrated into any recent SSL approaches such as MixMatch and its variants, since our proposed curriculum of selecting unlabeled samples is orthogonal to existing techniques. In addition, TC in this paper is of general interests to other machine learning problems leveraging unlabeled data.

**Acknowledgments** This research is based upon work supported by the National Science Foundation under Grant No. IIS-1162606, the National Institutes of Health under award R01GM103544, and by a Google, a Microsoft, and an Intel research award. It is also supported by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

## References


Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*, 2019.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2003.

Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 (NeurIPS)*, pp. 585–591. 2002.

Bengio, Y., Delalleau, O., and Le Roux, N. *Label Propagation and Quadratic Criterion*, pp. 193–216. Semi-supervised learning edition, 2006.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *ICML*, pp. 41–48, 2009.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 5050–5060. 2019.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations (ICLR)*, 2020.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 539–546, 2005.

Coates, A., Lee, H., and Ng, A. Y. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pp. 215–223, 2011.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space, 2019b.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17 (NeurIPS)*, pp. 529–536. 2005.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum contrast for unsupervised visual representation learning. *ArXiv*, abs/1911.05722, 2019.

Iscen, A., Tolias, G., Avrithis, Y., and Chum, O. Label propagation for deep semi-supervised learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5065–5074, 2019.

Ji, X., Henriques, J. F., and Vedaldi, A. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *NeurIPS*, pp. 1189–1197, 2010.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

Lee, D.-H. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

Li, Z. and Arora, S. An exponential learning rate schedule for deep learning. *ArXiv*, abs/1910.07454, 2019.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Miyato, T., Maeda, S., Koyama, M., and Ishii, S. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019.

Niyogi, P. Manifold regularization and semi-supervised learning: Some theoretical analyses. *Journal of Machine Learning Research*, 14:1229–1250, 2013.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 3235–3246. 2018.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pp. 759–766, 2007.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems 28*, pp. 3546–3554. 2015.

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

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, 2017.

Subramanya, A. and Bilmes, J. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12(Nov):3311–3370, 2011.

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

Trinh, T. H., Luong, M.-T., and Le, Q. V. Selfie: Self-supervised pretraining for image embedding, 2019.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning (ICML)*, volume 97, pp. 6438–6447, 2019a.

Verma, V., Qu, M., Lamb, A., Bengio, Y., Kannala, J., and Tang, J. Graphmix: Regularized training of graph neural networks for semi-supervised learning. *ArXiv*, abs/1909.11715, 2019b.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, 2016.

Zeiler, M. and Fergus, R. Visualizing and understanding convolutional networks. In *ECCV*, number PART 1, pp. 818–833, 2014.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *ECCV*, 2016.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16 (NeurIPS 2003)*, 2003.

Zhou, T. and Bilmes, J. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *ICLR*, 2018.

Zhu, X. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, pp. 4083–4087, 2015.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, pp. 912–919, 2003.