Smooth Neighbors on Teacher Graphs for Semi-supervised Learning

Yucen Luo[†], Jun Zhu[†], Mengxi Li[‡], Yong Ren[†], Bo Zhang[†]
[†] Department of Computer Science & Technology, Tsinghua University, Beijing, China
[‡] Department of Electronic Engineering, Tsinghua University, Beijing, China
{luoyc15@mails, dcszj@mail, limq14@mails,renyong15,dcszb@mail}.tsinghua.edu.cn

Abstract

The paper proposes an inductive deep semi-supervised learning method, called *Smooth Neighbors on Teacher Graphs (SNTG)*. At each iteration during training, a graph is dynamically constructed based on predictions of the teacher model, i.e., the implicit self-ensemble of models. Then the graph serves as a similarity measure with respect to which the representations of "similar" neighboring points are learned to be smooth on the low dimensional manifold. We achieve state-of-the-art results on semi-supervised learning benchmarks. The error rates are 9.89%, 3.99% for CIFAR-10 with 4000 labels, SVHN with 500 labels, respectively. In particular, the improvements are significant when the labels are scarce. For non-augmented MNIST with only 20 labels, the error rate is reduced from previous 4.81% to 1.36%. Our method also shows robustness to incorrect labels.

1 Introduction

Recently due to the great advances of deep learning, remarkable results have been achieved on semi-supervised learning (SSL) [8, 13, 9]. Among these works, perturbation-based methods [12, 9, 15, 11] have demonstrated great promise, which enforce smooth predictions of unlabeled data under different perturbations. Consider a training set $\mathcal{D}$ consists of N examples, out of which L have labels. Let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{L}$ be the labeled set and $\mathcal{U} = \{x_i\}_{i=L+1}^{N}$ be the unlabeled set where the input $x_i \in \mathcal{X}$ and the label $y_i \in \{1, 2, ..., K\}$. We aim to learn $f : \mathcal{X} \to [0, 1]^K$ parameterized by $\theta \in \Theta$ by solving:

$$\min_{\theta} \sum_{i=1}^{L} \ell(f(x_i; \theta), y_i) + \lambda R(\theta, \mathcal{L}, \mathcal{U}), \tag{1}$$

where ℓ is the supervised loss (e.g., cross-entropy loss) and $f(x; \theta)$ denotes the predictive distribution $p\,(y|x; \theta)$. The regularization term R is used to leverage unlabeled data and $\lambda \geq 0$ is the coefficient. All perturbation-based methods fit in Eq. (1) by defining R as a consistency loss:

$$R_C(\theta, \mathcal{L}, \mathcal{U}) = \sum_{i=1}^{N} \mathbb{E}_{\xi', \xi}\; d\big(\tilde{f}(x_i; \theta', \xi'), f(x_i; \theta, \xi)\big), \tag{2}$$

where $\tilde{f}$ is a "teacher" with parameters θ' and random perturbations ξ', similarly, f is a "student" with θ and ξ, and $d(\cdot, \cdot)$ denotes the divergence between two distributions (e.g., l_2 distance or KL divergence) [15]. The perturbations include the input noise and the network dropout. $\tilde{f}$ is defined as an implicit ensemble of previous student models and is expected to give better predictions than the student. $\tilde{f}(x)$ can be seen as the training targets for the unlabeled inputs and the student model is supposed to predict consistently with $\tilde{f}(x)$. Below are several ways to define the teacher model $\tilde{f}$:

Π **model [9].** $\tilde{f}$ shares the same parameters with f, i.e., $\theta' = \theta$ in Eq. (2). It evaluates the network twice under different realizations of i.i.d. perturbations ξ' and ξ and minimizes their l_2 distance.

Temporal ensembling (TempEns) [9]. To reduce the variance of the targets, TempEns maintains an exponentially moving average (EMA) of previous predictions over epochs as $\tilde{f}$. The ensemble

Table 1: Error rates (%) on benchmark datasets without augmentation, averaged over 10 runs.

Models	MNIST (L=100)	SVHN (L=1000)	CIFAR10 (L=4000)
Ladder network [12]	0.89±0.50	–	20.40±0.47
Improved GAN [13]	0.93±0.065	8.11±1.3	18.63±2.32
ALI [5]	–	7.42±0.65	17.99±1.62
Triple GAN [10]	0.91±0.58	5.77±0.17	16.99±0.36
GoodSemiBadGAN [4]	0.795±0.098	4.25±0.03	14.41±0.03
Π model [9]	0.89±0.15*	5.43±0.25	16.55±0.29
Π+SNTG (**ours**)	**0.66±0.07**	4.22±0.16	13.62±0.17
VAT [11]	1.36	5.77	14.82
VAT+Ent	–	4.28	13.15
VAT+Ent+SNTG (**ours**)	–	**4.02±0.20**	**12.49±0.36**

output is defined as $\tilde{F}^{(t)}(x_i) = \alpha \tilde{F}^{(t-1)}(x_i) + (1-\alpha) f^{(t)}(x_i;\theta,\xi)$, where $f^{(t)} : \mathcal{X} \to [0,1]^K$ is the student model at training epoch t and α is the pre-defined momentum. The target given by $\tilde{f}$ for x_i at epoch t is the debias correction of $\tilde{F}^{(t)}$, i.e., $\tilde{f}^{(t)}(x_i) = \tilde{F}^{(t)}(x_i)/(1-\alpha^t)$. Since the targets are based on EMA, the network only needs to be evaluated once, leading to a speed-up for Π model.

Mean teacher (MT) [15]. Instead of averaging previous predictions every epoch, MT updates the targets more frequently to form a better teacher, i.e., it averages parameters θ every iteration: $\theta' \leftarrow \alpha\theta' + (1-\alpha)\theta$. It also evaluates the network twice by the teacher and the student respectively.

VAT [11]. Instead of l_2 distance, VAT defines R_C as KL divergence between the model prediction and that of the input under adversarial perturbation ξ'_{adv}: $R_C = \sum_{i=1}^{N} \mathrm{KL}(\tilde{f}(x_i;\theta) \| f(x_i;\theta,\xi'_{adv}))$. It is assumed that a model trained under the worst-case (adversarial) perturbations will generalize well [11]. VAT resembles Π model but distinguishes itself in the distance metric and the type of perturbations. $\tilde{f}$ can be seen as the teacher while f with ξ'_{adv} is treated as the student.

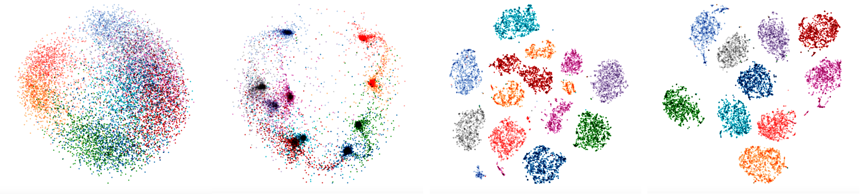
These approaches aim to fuse the inputs into coherent clusters by smoothing the mapping function locally. However, they only regularize the output to be smooth near each single data point, while ignoring the connections between data points, therefore not fully utilizing the information in unlabeled data structure, such as clusters or manifolds. An extreme situation may happen where the function is smooth in the vicinity of each unlabeled point but not smooth in the vacancy among them. This artifact could be avoided if the unlabeled data structure is taken into consideration. The connections between similar data points will help the fusing of clusters become tighter and more effective.

We present *Smooth Neighbors on Teacher Graphs* (SNTG), which considers the neighboring structure to induce smoothness on the data manifold in some local clusters. By defining a teacher graph based on the targets generated by the teacher, our model encourages features to be invariant when some perturbations are added to the neighboring points on the graph. We propose a doubly stochastic sampling algorithm to reduce the computational cost with large mini-batch sizes. Our method can be easily incorporated into existing deep SSL methods including both generative and discriminative approaches because SNTG does not increase the number of network parameters. We demonstrate significant performance improvements over state-of-the-art results while the extra time cost is negligible. SNTG is still effective with few labels or noisy labels.

2 Our Approach

In this section, we formalize SNTG by answering two key questions: (1) How to define the graph and neighbors? (2) How to induce the smoothness of neighboring points using the graph?

Building the graph with the teacher model. Most existing graph-based SSL methods [1, 16] depend on a distance metric in the input space $\mathcal{X}$, which is typically low-level (e.g., pixel values of images). For natural images, pixel distance cannot reflect semantic similarity well. Instead, we use the distance in the label space $\mathcal{Y}$, and treat the data points from the same class as neighbors. However, an issue is that the true labels of unlabeled data are unknown. We address it by learning a teacher graph using the targets generated by the teacher model. Self-ensembling is a good choice for constructing the graph because the ensemble predictions are expected to be more accurate than the outputs of current classifier. Formally, for $x_i \in \mathcal{D}$, a target prediction $\tilde{f}(x_i)$ is given by the teacher defined in the previous section. Denote the *hard* target prediction as $\tilde{y}_i = \mathrm{argmax}_k[\tilde{f}(x_i)]_k$ where $[\cdot]_k$ is the k-th component of the vector, indicating the probability that x_i is of class k. We build the graph as follows: $W_{ij} = \mathbb{1}[\tilde{y}_i = \tilde{y}_j]$, where $\mathbb{1}$ is the indicator function. W_{ij} measures the similarity



(a) Π model (b) SNTG (c) Π model (d) SNTG

Figure 1: (a,b) are the embeddings of CIFAR-10 test data projected to 2-D using PCA. (c,d) are the 2-D embeddings of MNIST test data using t-SNE. Each color denotes a class.

between x_i and x_j and those pairs with nonzero entries are treated as "neighbors". Here we restrict $W_{ij} \in \{0, 1\}$ to construct a 0-1 sparse graph.

Guiding the low dimensional mapping. Generally, a deep classifier (i.e., the student) f can be decomposed as $f = g \circ h$, where $h : \mathcal{X} \to \mathbb{R}^p$ is the mapping from the input space to the penultimate layer and $g : \mathbb{R}^p \to [0, 1]^K$ is usually parameterized by a fully-connected layer with softmax. Due to the hierarchical nature of deep networks, $h(x)$ can be seen as a low-dimensional feature of the input. And the feature space is expected to be linearly separable, as shown in the common practice that a following linear classifier g suffices. In terms of approximating the semantic similarity of two instances, the Euclidean distance of $h(x_i)$ and $h(x_j)$ is more suitable than that of $f(x)$ which represents class probabilities. Hence we use the graph to guide $h(x)$, making them distinguishable among classes. Given a $N \times N$ similarity matrix W of the sparse graph, we define our SNTG loss as

$$R_S(\theta, \mathcal{L}, \mathcal{U}) = \sum_{x_i, x_j \in \mathcal{D}} \ell_G(h(x_i; \theta), h(x_j; \theta), W_{ij}) \quad (3)$$

Here we utilize the contrastive Siamese networks [3], which can learn an invariant mapping and perform well in metric learning and face verification [7, 14]. Specifically, the loss is defined as:

$$\ell_G = \begin{cases} \|h(x_i) - h(x_j)\|^2 & \text{if } W_{ij} = 1 \\ \max\left(0, m - \|h(x_i) - h(x_j)\|\right)^2 & \text{if } W_{ij} = 0 \end{cases} \quad (4)$$

where $m > 0$ is a pre-defined margin. The margin loss is to constrain neighboring points to have consistent embeddings. Consequently, the neighbors are encouraged to have consistent predictions while the non-neighbors (i.e., the points of different classes) are pushed apart from each other with a minimum distance m.

One interpretation of why the proposed method works well is that SNTG explores more information in the teacher and improves the target quality. The teacher graph leads to better abstract representations in a smooth and coherent feature space and then aids the student f to give more accurate predictions. In turn, an improved student contributes to a better teacher model which can provide more accurate targets. Another perspective is that SNTG implements the *manifold* assumption for classification which underlies the loss ℓ_G, i.e.,, the points of same class are encouraged to concentrate together on sub-manifolds. The perturbation-based methods only keep the decision boundaries far away from each unlabeled data point while our method encourages the unlabeled data points to form tighter clusters, leading the decision boundaries to locate between the clusters.

3 Experiments

We provide results on widely adopted benchmarks. Following [13], we randomly sample 100, 1000 and 4000 labels for MNIST, SVHN and CIFAR-10, respectively. We further test with fewer labels for non-augmented MNIST as well as SVHN and CIFAR-10 with standard data augmentation. The results are averaged over 10 runs with different seeds for data splits. We use the same network architecture and hyper-parameters to our baselines. Main results are presented in Tables 1, 2.Our method surpasses previous state-of-the-arts by a large margin. More results and details on experimental setup are in Appendix A.

Table 2: Errors(%) on MNIST without augmentation.

Models	20 labels	50 labels
FM-GAN [13]	16.77±4.52	2.21±1.36
Triple-GAN [10]	4.81±4.95	1.56±0.72
Π model [9]	6.32±6.90*	1.02±0.37*
Π+SNTG (**Ours**)	**1.36±0.78**	**0.94±0.42**

Note that for VAT, its best results are achieved with an additional entropy minimization (Ent) regularization [6], leading to a much stronger baseline. We evaluate our method under its best setting VAT+Ent, and observe a further improvement with SNTG, e.g., from 13.15% to 12.49% and from 10.55% to 9.89% on CIFAR-10 without or with augmentation, respectively. In fact, we observe that Ent can also improve the performance of other baselines if it was added along with SNTG. But to focus on the efficacy of SNTG, we did not illustrate the results here.

Fewer labels. Notably, as shown in Tables 2, 4 and 3, when labels are scarce, e.g., MNIST with 20 labels (only 2 each class), SVHN with 250 labels and CIFAR-10 with 1000 labels, the improvements are even more significant. Since the labeled data only accounts for a small part, adding a strong regularizer SNTG in this case helps the model learn faster by exploring the unlabeled data structure.

Noisy Labels. SNTG can not only benefit from unlabeled data, but also learn from noisy supervision. We did experiments on supervised SVHN to show the tolerance to incorrect labels. True labels on the training set are replaced by random labels following [9]. Fig. **??** shows that TempEns+SNTG retains over 93% accuracy even when 90% of the labels are noisy while TempEns alone only obtains 73% [9]. Thus, our SNTG regularization improves the robustness and generalization performance.

Visualization of embeddings We finally visualize the embeddings of our algorithm and Π model on test data under the same settings (CIFAR-10 with 4000 labels and MNIST with 100 labels, both without augmentation). Fig. 1 shows the representations $h(x) \in \mathbb{R}^{128}$ projected to 2 dimension using PCA or tSNE. The learned representations of our model are more concentrated within clusters and are potentially easier to separate for different classes. The visualization is also consistent with our assumption and analysis.

4 Conclusions and Future Work

We developed a simple but effective SNTG for SSL that builds graphs using the teacher model and enforces smoothness of neighbor points on it. Empirically, it outperforms all baselines and achieves new state-of-the-art results. As a byproduct, we also learn an invariant mapping on a low dimensional manifold. SNTG can handle extreme cases with fewer labels and noisy labels.

References

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(Nov):2399–2434, 2006.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a “siamese" time delay neural network. In *NIPS*, 1994.

[4] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *NIPS*, 2017.

[5] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

[6] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2005.

[7] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[8] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[9] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

[10] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *NIPS*, 2017.

[11] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[12] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

[13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[15] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

[16] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *ICML*, 2008.

This paper is a short version. See `https://arxiv.org/pdf/1711.00258.pdf` for more details.

A Experimental Setup

MNIST. It contains 60,000 gray-scale training images and 10,000 test images from handwritten digits 0 to 9. The input images are normalized to zero mean and unit variance.

SVHN. Each example in SVHN is a 32×32 color house-number images and we only use the official 73,257 training images and 26,032 test images following previous work. The augmentation of SVHN is limited to random translation between $[-2, 2]$ pixels.

CIFAR-10. The CIFAR-10 dataset consists of 32×32 natural RGB images from 10 classes such as airplanes, cats, cars and horses. We have 50,000 training examples and 10,000 test examples. The input images are normalized using ZCA following previous work [9]. We use the standard way of augmenting the CIFAR-10 dataset including horizontal flips and random translations.

Training details. In Π model and TempEns based experiments, the network architectures (shown in Table 5) and the hyper-parameters are the same as our baselines. Following [9], we apply mean-only batch normalization with momentum 0.999 to all layers and use leaky ReLU with $\alpha = 0.1$. The network is trained for 300 epochs using Adam Optimizer with mini-batches of size $n = 100$ and maximum learning rate 0.003 (exceptions are that temporal ensembling for SVHN uses 0.001 and MNIST uses 0.0001). We use the default Adam momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also ramp up the learning rate and the regularization term during the first 80 epochs with weight $w(t) = \exp\left[-5(1 - \frac{t}{80})^2\right]$ and ramp down the learning rate during the last 50 epochs. The ramp down function is $\exp\left[-12.5(1 - \frac{300-t}{50})^2\right]$. The regularization coefficient of consistency loss R_C is $\lambda_1 = 100$ for Π model and $\lambda_1 = 30$ for temporal ensembling (exception is that SVHN with $L = 250$ uses $\lambda_1 = 50$).

For comparison with Mean Teacher and VAT, we keep the same architecture and hyper-parameters settings with the corresponding baselines [15, 11]. Their network architectures are the same as shown in Table 5 but differ in some hyper-parameters such as weight normalization, training epochs and mini-batch sizes, which are detailed in their papers. We just add our SNTG loss along with their regularization R_C and keep other settings unchanged.

SNTG loss only needs three extra hyper-parameters: the regularization parameter λ_2, the margin m and the number of sub-sampled pairs s. We fix m and s, only tune λ_2. In all our experiments, the margin m in Eq. (4) is set to $m = 1$ when $\|h(x_i) - h(x_j)\|^2$ is averaged by feature dimension p. We sample half the number of mini-batch size pairs of (x_i, x_j) for computing ℓ_G, e.g., $s = 50$ for mini-batch size $n = 100$. The regularization coefficient λ_2 for SNTG loss R_S is set to $k\lambda_1$ where k is chosen from $\{0.2, 0.4, 0.5, 0.6, 1.0\}$ using the validation set and we use default ratio $k = 0.4$ for most settings. SNTG does not increase the number of model parameters and the run time is almost the same to baselines.

Code reproducing our results will be available at `https://github.com/xinmei9322/SNTG` soon.

B Doubly stochastic sampling approximation

Our overall objective is the sum of two components. The first one is cross-entropy loss on the labeled set $\mathcal{L}$, and the second is the regularization term, which encourages the smoothness for each single point (i.e., R_C) as well as for the neighboring points (i.e., R_S). Alg. 1 presents the pseudo-code.

As our model belongs to deep networks, we train it using SGD [2] with mini-batches. We construct the sub-graph in a random mini-batch B of size n to estimate R_S in Eq. (3). We need to compute W_{ij} for all the data pairs $(x_i, x_j) \in B$, which is of size n^2 in total. Although this step is fast, the computation of $\|h(x_i) - h(x_j)\|$ related to W_{ij} is $O(p)$ and then the overall computational cost is $O(n^2 p)$, which is slow for large n. We instead use doubly stochastic sampled data pairs to construct W_{ij} for Eq. (4), which is still an unbiased estimation of R_S. Specifically, in each iteration, we sample a mini-batch B and then sub-sample $s \leq n^2$ data pairs S from B.

Algorithm 1 Mini-batch training of SNTG for SSL

```
Input: $x_i, y_i, w(t)$ = unsupervised weight ramp-up function
$f_\theta(x)$ = neural network with trainable parameters $\theta$
for $t$ in [1, numepochs] do
  for each minibatch $B$ do
    $f_i \leftarrow f_\theta(x_{i\in B})$ evaluate network outputs
    $\tilde{f}_i \leftarrow \tilde{f}(x_{i\in B})$ given by the teacher model
    for $(x_i, x_j)$ in a minibatch pairs $S$ from $B$ do
      Compute $W_{ij}$ according to Section 2
    end for
    loss $\leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap\mathcal{L})}\log[f_i]_{y_i}$
    $+w(t)\Big[\lambda_1\frac{1}{|B|}\sum_{i\in B} d(\tilde{f}_i, f_i) + \lambda_2\frac{1}{|S|}\sum_{i,j\in S}\ell_G(h(x_i), h(x_j), W_{ij})\Big]$
    update $\theta$ using optimizers
  end for
end for
return $\theta$
```

Table 3: Error rates (%) on CIFAR-10 with standard augmentation, averaged over 10 runs.

Model	1000 labels	2000 labels	4000 labels	All labels
Supervised-only			34.85±1.65	6.05±0.15
Π model [9]	31.65±1.20*	17.57±0.44*	12.36±0.31	5.56±0.10
Π+SNTG (**ours**)	21.23±1.27	14.65±0.31	11.00±0.13	**5.19±0.14**
TempEns [9]	23.31±1.01*	15.64±0.39*	12.16±0.24	5.60±0.10
TempEns+SNTG (**ours**)	**18.41±0.52**	**13.64±0.32**	10.93±0.14	5.20±0.14
VAT [11]	–	–	11.36	5.81
VAT+Ent [11]	–	–	10.55	–
VAT+Ent+SNTG (**ours**)	–	–	**9.89±0.34**	–

Table 4: Error rates (%) on SVHN with translation augmentation, averaged over 10 runs.

Model	250 labels	500 labels	1000 labels	All labels
Supervised-only	42.65±2.68	22.08±0.73	14.46±0.71	2.81±0.07
Π model [9]	9.93±1.15*	6.65±0.53	4.82±0.17	2.54±0.04
Π+SNTG (**ours**)	5.07±0.25	4.52±0.30	**3.82±0.25**	**2.42±0.05**
TempEns [9]	12.62±2.91*	5.12±0.13	4.42±0.16	2.74±0.06
TempEns+SNTG (**ours**)	5.36±0.57	4.46±0.26	3.98±0.21	2.44±0.03
Mean Teacher [15]	4.35±0.50	4.18±0.27	3.95±0.19	2.50±0.05
Mean Teacher+SNTG (**ours**)	**4.29±0.23**	**3.99±0.24**	3.86±0.27	2.42±0.06
VAT [11]	–	–	5.42	–
VAT+Ent [11]	–	–	3.86	–
VAT+Ent+SNTG (**ours**)	–	–	3.83±0.22	–

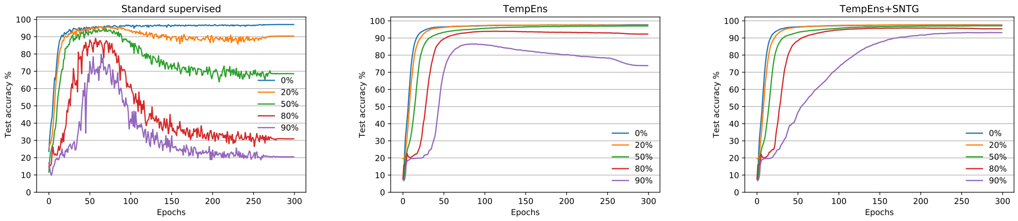


Figure 2: Test accuracy on supervised SVHN with noisy labels. Different colors denote the percentages of corrupted labels. With standard supervised training (left), the model suffers a lot and overfits to the incorrect information in labels. TempEns (middle) shows the resistance to the corruption but still has a drop in accuracy when the portion of randomized labels increases to 90%. Adding SNTG shows almost perfect robustness even when 90% labels are corrupted.

Table 5: The network architecture we used in all experiments.

Input: $32 \times 32 \times 3$ image ($28 \times 28 \times 1$ for MNIST)
Gaussian noise $\sigma = 0.15$
3×3 conv. 128 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 128 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 128 lReLU ($\alpha = 0.1$) same padding
2×2 max-pool, dropout 0.5
3×3 conv. 256 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 256 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 256 lReLU ($\alpha = 0.1$) same padding
2×2 max-pool, dropout 0.5
3×3 conv. 512 lReLU ($\alpha = 0.1$) valid padding
1×1 conv. 256 lReLU ($\alpha = 0.1$)
1×1 conv. 128 lReLU ($\alpha = 0.1$)
Global average pool 6×6 (5×5 for MNIST) $\rightarrow 1{\times}1$
Fully connected $128 \rightarrow 10$ softmax