Smooth Neighbors on Teacher Graphs for Semi-supervised Learning
Supplementary Material

Yucen Luo[1] Jun Zhu[1*] Mengxi Li[2] Yong Ren[1] Bo Zhang[1]
[1] Dept. of Comp. Sci. & Tech., State Key Lab for Intell. Tech. & Sys., BNRist Lab, Tsinghua University
[2] Department of Electronical Engineering, Tsinghua University
{luoyc15, limq14, renyong15}@mails.tsinghua.edu.cn; {dcszj, dcszb}@tsinghua.edu.cn

Table 1: The network architecture 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) $\rightarrow 1\times1$
Fully connected $128 \rightarrow 10$ softmax

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 [7]. We use the standard way of augmenting the CIFAR-10 dataset including horizontal flips and random translations.

CIFAR-100. The CIFAR-100 dataset consists of 32×32 natural RGB images from 100 classes. We have 50,000 training examples and 10,000 test examples. The preprocession of inputs images are the same to CIFAR-10.

Implementation. We implemented our code mainly in Python with Theano [14] and Lasagne [4]. For comparison with VAT [9] and Mean Teacher [13] experiments, we use TensorFlow [1] to match their settings. The code for reproducing the results is available at `https://github.com/xinmei9322/SNTG`.

Training details. In Π model and TempEns based experiments, the network architectures (shown in Table 1) and the hyper-parameters are the same as our baselines [7]. We apply mean-only batch normalization with momentum 0.999 [11] to all layers and use leaky ReLU [8] with $\alpha = 0.1$. The network is trained for 300 epochs using Adam Optimizer [6] with mini-batches of size $n = 100$ and maximum learning rate 0.003 (exceptions are that TempEns for SVHN uses 0.001 and MNIST uses 0.0001). We use the default Adam momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Following [7], 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 TempEns (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 [13, 9]. Their network architectures are the same as shown in Table 1 but differ in several hyper-parameters such as weight normalization, training epochs and mini-batch sizes, which are detailed in their papers. We just add the SNTG loss along with their regularization R_C and keep other settings unchanged as in their public code.

In all our experiments, the margin m in R_S is set to $m = 1$ if we treat $\|h(x_i) - h(x_j)\|^2$ as a distance averaged by the 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

*Corresponding author.

Table 2: Comparision of error rates on MNIST with 600 labels between various classifcal SSL methods.

Methods	LGC	TSVM	LapRLS	LP	LP+kNN	DLP	*Embed*NN	MTC	PEA	SNTG (**ours**)
Error (%)	3.96	4.87	2.92	8.57	4.27	2.01	3.42	5.13	2.44	**0.45**

λ_2 of SNTG loss R_S is set to $\lambda_2 = k\lambda_1$ where k is the ratio of λ_2 to λ_1 (*i.e.*, the regularization coefficient of consistency loss R_C). k is chosen from $\{0.2, 0.4, 0.6, 1.0\}$ using the validation set and we use $k = 0.4$ for most experiments by default.

Training time. SNTG does not increase the number of neural network parameters and the runtime is almost the same to the baselines, with only extra 1-2 seconds per epoch (the baselines usually need 100-200 seconds per epoch on one GPU).

Synthetic benchmarks. The synthetic dataset experiments adopt the default settings for Π model [7] except for 0.001 maximum learning rate and 500 training epochs. We use weight normalization [11] and add Gaussian noise to each layer.

B. Rethinking Π model objective

In Π model [7], the consistency loss is defined in Eq. (2) where the teacher model shares the same parameter with the student model $\theta' = \theta$. Suppose $f(x) \in [0, 1]^K$, the consistency loss of Π model is

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

ξ' and ξ are i.i.d random noise variables, $\xi', \xi \sim p(\xi)$, then we have $\mathbb{E}_\xi f(x_i; \theta, \xi) = \mathbb{E}_{\xi'} f(x_i; \theta, \xi')$ and $\mathbb{E}_\xi \|f(x_i; \theta, \xi)\|^2 = \mathbb{E}_{\xi'} \|f(x_i; \theta, \xi')\|^2$

$$\begin{aligned} R_C &= 2 \sum_{i=1}^{N} \mathbb{E}_\xi \|f(x_i; \theta, \xi)\|^2 - \|\mathbb{E}_\xi f(x_i; \theta, \xi)\|^2 \\ &= 2 \sum_{i=1}^{N} \sum_{k=1}^{K} \mathrm{Var}_\xi \left[f(x_i; \theta, \xi) \right]_k \end{aligned}$$

where $[\cdot]_k$ is k-th component of the vector.

Then minimizing R_C is equivalent to minimizing the sum of variance of the prediction each dimension. Similar idea of variance penalty was exploited in Pseudo-Ensemble [2]. If a data point is near the decision boundary, it is likely to has a large variance since its prediction might alternate to another class when some noise is added. Minimizing the variance explicitly penalizes such alternation behavior of training data.

C. Comparison to classical SSL methods

As mentioned in Section 2, our method is different from classical graph-based SSL methods in many important aspects such as the construction of the graph and how to use it.

Table 2 is a comparion with several classical methods: (1) Label propagation (LP) [18]; (2) A variant of LP on kNN structure(LP+kNN) [12]; (3) Local and Global Consistency (LGC) [17]; (5) Transductive SVM (TSVM) [5]; (6) LapRLS [3]; (7) Dynamic Label propagation (DLP) [15]. The results of (1)-(7) are cited from [15]. We also compare with the best reported results in previously mentioned works: (8) *Embed*NN [16]; (9) the Manifold Tangent Classifier (MTC) [10]; (10) Pseudo-Ensemble [2].

While the classical graph-based methods (*e.g.*, LP, DLP and LapRLS) were the leading paradigms, with the resurgence of deep learning, recent impressive results are mostly from deep learning based SSL methods, while classical methods fall behind on performance and scalability. Furthermore, they have no reported results on challenging natural image datasets, *e.g.*, SVHN, CIFAR-10. Only one overlap is MNIST, see Table 2 for comparison. We show that our method SNTG surpasses these classical methods by a large margin.

D. Significance test of the improvements.

Table 3 shows the independent two sample T-test on the error rates of baselines and our method. All the P-values are less than significance level $\alpha = 0.01$. It indicates that the improvements of SNTG are significant.

Table 3: T-test. The top rows are the experiments without augmentation and the bottom rows are with augmentation.

Datasets & Methods		T-statistic	P-value
MNIST (L=20)	Π model v.s. Π+SNTG	20.00227	9.07043e-09
MNIST (L=100)	Π model v.s. Π+SNTG	4.34867	0.000387026
SVHN (L=1000)	VAT+Ent v.s. VAT+Ent+SNTG	4.08627	0.002732236
CIFAR-10 (L=4000)	VAT+Ent v.s. VAT+Ent+SNTG	5.90681	0.000227148
SVHN (L=250)	Π model v.s. Π+SNTG	12.31365	3.32742e-10
SVHN (L=500)	TempEns v.s. TempEns+SNTG	7.52909	3.58188e-05
CIFAR-10 (L=1000)	TempEns v.s. TempEns+SNTG	12.81875	1.73155e-10
CIFAR-10 (L=2000)	TempEns v.s. TempEns+SNTG	11.80608	6.55694e-10
CIFAR-10 (L=4000)	VAT+Ent v.s. VAT+Ent+SNTG	5.81409	0.000254937

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al.

Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.

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

[4] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, et al. Lasagne: First release., Aug. 2015.

[5] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209, 1999.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[8] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.

[9] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[10] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2011.

[11] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[12] A. Subramanya, S. Petrov, and F. Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176. Association for Computational Linguistics, 2010.

[13] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[14] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[15] B. Wang, Z. Tu, and J. K. Tsotsos. Dynamic label propagation for semi-supervised multi-class multi-label classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 425–432, 2013.

[16] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 1168–1175, 2008.

[17] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328, 2004.

[18] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. *Technical Report CMU-CALD-02-107, Carnegie Mellon University*, 2002.