1 The Architectures of Networks

In Table 1, we deliver the CNNs architecture used in our experiments. We describe the operation in the format of “filter size / type / # of output channel”. Note that the network architecture of LLP-GAN for SVHN and CIFAR-100 is the same as that for CIFAR-10. In particular, *Dense* means fully connected layer. *Transpose_conv* is the deconvolution layer. In our model, we choose ReLU as the activation function.

Following a standard setting in the previous work [1, 4], we perform 11-way softmax on the 10-dimensional output in the last fully connected layer. In detail, we add an extra dimension to the output and fix its value as zero, which is a form of over-parameterization. Then, we apply 11-way softmax to the 11-dimensional vector.

Table 1: Network architectures in LLP-GAN.

Generator		Discriminator	
MNIST	CIFAR-10	MNIST	CIFAR-10
Input 28×28 or 32×32 monochrome or RGB image			
			dropout 0.2
Dense-BN 500	Dense-BN 4*4*512	5×5 conv. 32 3×3 conv. 64	3×3 conv. 64 3×3 conv. 64 3×3 conv. 64
			dropout 0.5
Dense-BN 500	5×5 Transpose_conv-BN 256	1×1 conv. 32	3×3 conv. 128 3×3 conv. 128 3×3 conv. 128
			dropout 0.5
Dense-BN 784	5×5 Transpose_conv-BN 128	Dense 1024	3×3 conv. 256 1×1 conv. 128 1×1 conv. 64
			global meanpooling 8
	5×5 Transpose_conv 3	Dense 10	Dense 10
		11-way softmax (over-parameterization)	

The CNNs architectures used in the baselines for MNIST and CIFAR-10 are given in Table 2, which are the same as that in [3]. Besides, the CNNs used in the baselines for SVHN and CIFAR-100 are the same as that in [2].

Table 2: The baseline’s architectures.

MNIST	CIFAR-10
Input 28×28 or 32×32 monochrome or RGB image	
5×5 conv. ReLU 32	3×3 conv. BN LeakyReLU 96 3×3 conv. BN LeakyReLU 96 3×3 conv. BN LeakyReLU 96
2×2 max-pooling stride 2 BN	2×2 max-pooling stride 2 BN
3×3 conv. BN ReLU 64 3×3 conv. BN ReLU 64	3×3 conv. BN LeakyReLU 192 3×3 conv. BN LeakyReLU 192 3×3 conv. BN LeakyReLU 192
2×2 max-pooling stride 2 BN	2×2 max-pooling stride 2 BN
3×3 conv. 128 BN ReLU 1×1 conv. BN ReLU 10	3×3 conv. BN LeakyReLU 192 1×1 conv. BN LeakyReLU 192 1×1 conv. BN LeakyReLU 10
global meanpool BN	global meanpool BN
Dense-BN 10	
10-way softmax	

2 More results on Performance

2.1 Binary Case

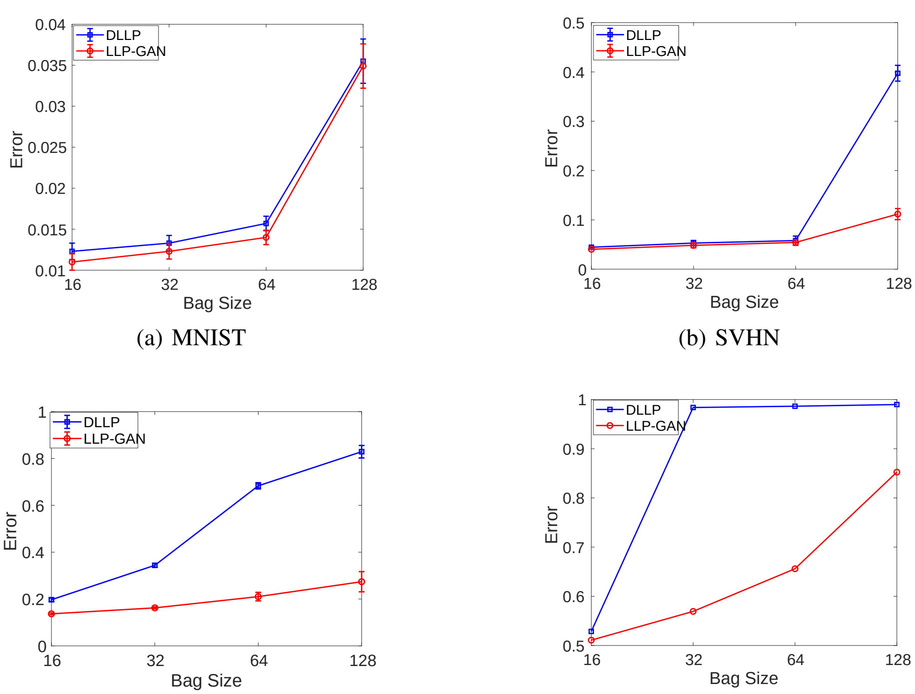
In addition to the comparison between DLLP and LLP-GAN, we investigate results of other two representative LLP solvers: InvCal and alter-$\propto$SVM. Because they are originally designed for binary problem, we randomly select two classes and merely conduct binary classification on all datasets with four algorithms. The comparison on test error rates is displayed in Table 3.

Table 3: Binary test error rates (%) on benchmark datasets with different bag sizes.

Dataset	Algorithm	Bag Size			
		16	32	64	128
MNIST	InvCal	0.50	0.55	1.25	0.1
	alter-pSVM	0.20	0.20	0.25	0.2
	DLLP	0.049	0.049	0.049	0.049
	LLP-GAN	**0.047**	**0.047**	**0.047**	**0.047**
CIFAR-10	InvCal	28.95	29.16	26.47	31.84
	alter-pSVM	24	26.74	30.32	27.95
	DLLP	11.31	15.83	18.96	22.59
	LLP-GAN	**1.39**	**1.61**	**11.59**	**18.29**
SVHN	InvCal	11.55	13.35	12.95	12.70
	alter-pSVM	7.05	7.95	7.95	11.15
	DLLP	**1.38**	**1.7**	3.77	24.45
	LLP-GAN	1.49	1.8	**3.46**	**9.23**

2.2 Multi-class Case

We report multi-calss error rates with the standard deviations of DLLP and LLP-GAN on benchmark datasets in Figure 1. It is based on the results in Table 1 of our paper.



(a) MNIST (b) SVHN

(c) CIFAR-10 (d) CIFAR-100

Figure 1: Multi-class test error rates (%) on benchmark datasets with different bag sizes.

2.3 DLLP with Entropy Regularization

Although DLLP with Entropy Regularization is a side contribution of our work, as claimed in the paper, we consider not to include it as a baseline. The reason is the experimental results suggest that the original DLLP has already converged to the solution with fairly low instance-level entropy, which makes the regularization term redundant. We demonstrate this statement in Figure 2.

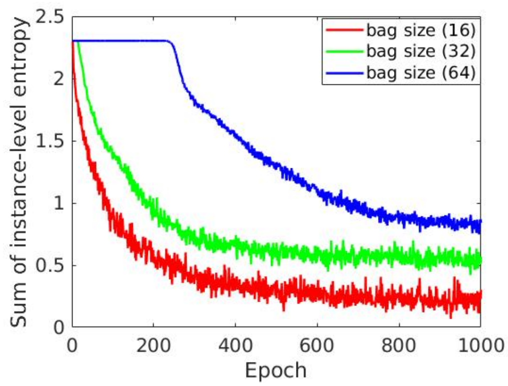


Figure 2: Sum of instance-level entropy on MNIST.

2.4 The Randomness of Bag Assignment

The distribution of proportions has an huge impact on LLP algorithm performance. Hence, fixing bag size, we randomly construct bags for multiple times and present the accuracy performance in Table 2.4. The result shows the stability of our method. Currently, we can only artificially build LLP datasets from supervised ones. However, the gap between the importance of LLP in real-life and lack of specific LLP datasets exactly suggests the meaning of our work: It is worthy of devoting efforts to further study in order to draw more attention from the community.

Table 4: The performance on accuracy with deviation under multiple random bag generations on MNIST. (Due to the time limitation, # of random are differently chosen.)

Bag Size (# of Random)	# of Errors	Accuracy (%) (Deviation)	Baseline (CNN)
16 (7)	106	98.94 (0.0285)	99.64
32 (22)	124	98.76 (0.0542)	
64 (45)	147	98.53 (0.11)	
128 (85)	335	96.65 (0.4)	

References

[1] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[2] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[3] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015.

[4] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.