Supplementary Materials: Regularization via Structural Label Smoothing

Weizhi Li
Arizona State University

Gautam Dasarathy
Arizona State University

Visar Berisha
Arizona State University

1 Description of the Synthetic Dataset

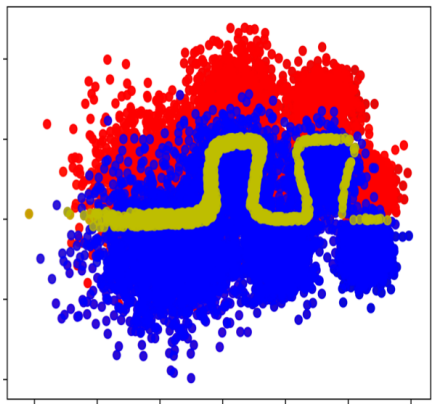


Figure 1: Illustration of the synthetic data with the highlighted optimal decision boundary

To create a binary classification synthetic dataset, we first generate two-dimensional features S_0 and S_1 from a Gaussian mixture model (GMM) for Class 0 and Class 1. Specifically, we have $S_0 \sim \sum_{i=1}^{6} \frac{1}{6}\mathcal{N}(\mathbf{u}_i^0, \mathbf{\Sigma}_i^0)$ and $S_1 \sim \sum_{i=1}^{6} \frac{1}{6}\mathcal{N}(\mathbf{u}_i^1, \mathbf{\Sigma}_i^1)$ where $\mathbf{u}_i$ and $\mathbf{\Sigma}_i$ are mean vector and covariance matrices of a Gaussian distribution. This amounts to a GMM with 6 components. We subsequently add 126 noisy features, each of them sampled from a Gaussian distribution $\mathcal{N}(0.05, 0.083^2)$. Consequently, we generate a 128-dimensional synthetic dataset where only 2 dimensions are useful for classification. The synthetic dataset, plotted only along the two-dimensional features that are useful for classification is shown in Figure 1. The parameters used to generate synthetic data are described below:

$$\mathbf{u}_1^0 = [-0.60, 0.20]^T, \quad \mathbf{\Sigma}_1^0 = \begin{bmatrix} 0.070 & 0 \\ 0 & 0.070 \end{bmatrix}, \quad \mathbf{u}_1^1 = [-0.60, -0.20]^T, \quad \mathbf{\Sigma}_1^1 = \begin{bmatrix} 0.070 & 0 \\ 0 & 0.070 \end{bmatrix}$$

$$\mathbf{u}_2^0 = [-0.25, 0.20]^T, \quad \mathbf{\Sigma}_2^0 = \begin{bmatrix} 0.047 & 0 \\ 0 & 0.047 \end{bmatrix}, \quad \mathbf{u}_2^1 = [-0.25, 0.20]^T, \quad \mathbf{\Sigma}_2^1 = \begin{bmatrix} 0.047 & 0 \\ 0 & 0.047 \end{bmatrix}$$

$$\mathbf{u}_3^0 = [0.10, 0.70]^T, \quad \mathbf{\Sigma}_3^0 = \begin{bmatrix} 0.031 & 0 \\ 0 & 0.031 \end{bmatrix}, \quad \mathbf{u}_3^1 = [0.10, 0.30]^T, \quad \mathbf{\Sigma}_3^1 = \begin{bmatrix} 0.031 & 0 \\ 0 & 0.031 \end{bmatrix}$$

$$\mathbf{u}_4^0 = [0.45, 0.20]^T, \quad \mathbf{\Sigma}_4^0 = \begin{bmatrix} 0.021 & 0 \\ 0 & 0.021 \end{bmatrix}, \quad \mathbf{u}_4^1 = [0.45, -0.20]^T, \quad \mathbf{\Sigma}_4^1 = \begin{bmatrix} 0.021 & 0 \\ 0 & 0.021 \end{bmatrix}$$

$$\mathbf{u}_5^0 = [0.80, 0.70]^T, \quad \mathbf{\Sigma}_5^0 = \begin{bmatrix} 0.014 & 0 \\ 0 & 0.014 \end{bmatrix}, \quad \mathbf{u}_5^1 = [0.80, 0.30]^T, \quad \mathbf{\Sigma}_5^1 = \begin{bmatrix} 0.014 & 0 \\ 0 & 0.014 \end{bmatrix}$$

$$\mathbf{u}_6^0 = [1.15, 0.20]^T, \quad \mathbf{\Sigma}_6^0 = \begin{bmatrix} 0.009 & 0 \\ 0 & 0.009 \end{bmatrix}, \quad \mathbf{u}_6^1 = [1.15, -0.20]^T, \quad \mathbf{\Sigma}_6^1 = \begin{bmatrix} 0.009 & 0 \\ 0 & 0.009 \end{bmatrix}$$

2 Complete results for cross-entropy on test sets along training

The figures below show the cross-entropy loss, computed on the test set, as a function of epoch, for the SVHN, CIFAR-10, and CIFAR-100 data for 3 smoothing strength values. The results are shown for cases where batchnorm is used in addition to the label smoothing regularizer and for the case where it is not.

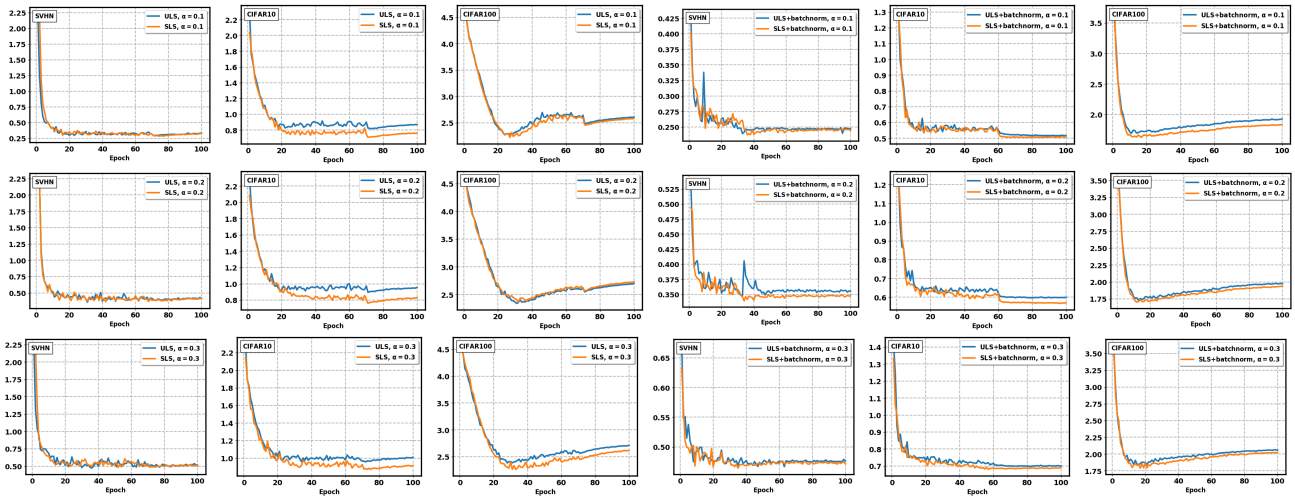


Figure 2: Complete results for cross-entropy on test sets along training.

3 Extra Experimental Results on the CIFAR-10

In additional to the MobileNetV2 (Sandler et al., 2018), we also train ResNet-18 (He et al., 2016), Vgg (Simonyan and Zisserman, 2014), and Googlenet (Szegedy et al., 2015) on the CIFAR-10 dataset without any regularization. Similar to the experiments described in Section 4.3 of the main paper, we compute the CSR of each cluster for the CIFAR-10 training set by the LASS algorithm described in (Arpit et al., 2017) and examine the complexity of the decision boundary learned by MobileNet-v2, ResNet-18, Vgg and Googlenet. Furthermore, for the CIFAR-10 clustering previously described, we compute the error difference (ED) between test error and training error. The correlation matrix between each pair of CSR and ED is shown in Table 1. The results show a strong correlation

CIFAR10	Mobile-CSR	Res-CSR	Vgg-CSR	Google-CSR	Mobile-ED	Res-ED	Vgg-ED	Google-ED
Mobile-CSR	1	0.9549	0.8901	0.9524	0.7574	0.8167	0.8382	0.7549
Res-CSR	0.9549	1	0.9390	0.9289	0.7031	0.7914	0.7808	0.7145
Vgg-CSR	0.8901	0.9390	1	0.8924	0.5612	0.6492	0.6854	0.6350
Google-CSR	0.9524	0.9289	0.8924	1	0.7208	0.8039	0.8300	0.7630
Mobile-ED	0.7574	0.7031	0.5612	0.7208	1	0.8286	0.8095	0.7691
Res-ED	0.8167	0.7914	0.6492	0.8039	0.8286	1	0.8250	0.7826
Vgg-ED	0.8382	0.7808	0.6854	0.8300	0.8095	0.8520	1	0.8775
Google-ED	0.7549	0.7145	0.6350	0.7630	0.7691	0.7826	0.8775	1

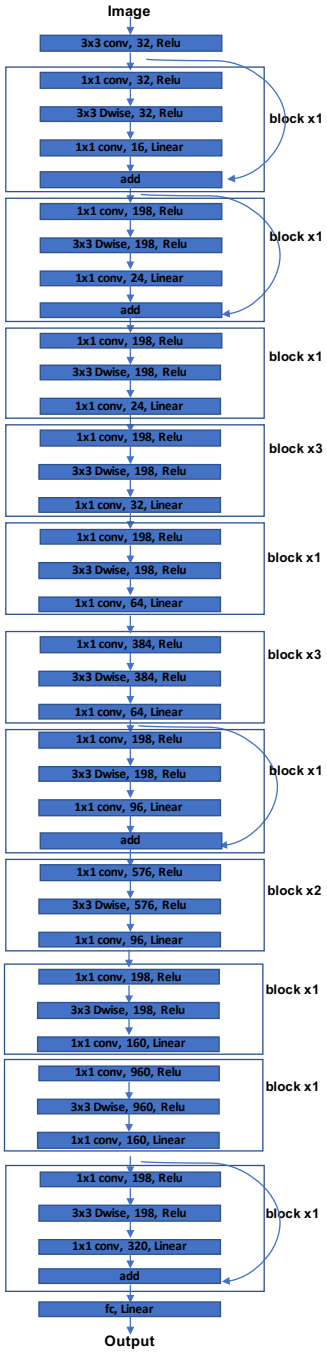
Table 1: Correlation matrices among CSR and error difference (ED) between test error and training error from MobileNetV2, ResNet-18, Vgg, and Googlenet for CIFAR-10.

among the CSRs and ED from different networks. This experimentally provides some additional evidence for the claims in (Goodfellow et al., 2014) which shows that adversarial examples generalize over different neural networks: an adversarial example misclassified by one network is often misclassified by others. This also indicates that the MobileNetV2, ResNet-18, Vgg, and the Googlenet learn similar decision boundary and therefore implies that our proposed structural label smoothing potentially works on the other neural network architectures in addition to the MobileNetV2.

4 Neural Network Architectures

The architectures of the auto-encoder and the MobileNetV2 (Sandler et al., 2018) used in the experiments described in Section 4.2 of the main paper are shown in Figure 3(a) and (b).

(a) Auto-encoder



(b) MobileNetV2

Figure 3: Illustrations of the (a) auto-encoder and (b) MobileNetV2 with each box containing kernel size, convolution (conv)/full-connection (fc)/deconvolution (deconv)/depthwise convolution (Dwise) operation, output features/feature channels, stride size and activation function.

5 Study on number of clusters used for Structural Label Smoothing (SLS)

Our empirical results indicate that SLS performs better than ULS consistently regardless of the number of clusters chosen. We added additional experiments to confirm this and the results are in Table 2. This table shows the performance of SLS and ULS on the synthetic dataset for different number of clusters. Fixing α, β and the number of clusters, we train the same neural network for five trials and generate the mean error rate and the lowest error rate of the five trials. As shown in Table 2, SLS consistently performs better than ULS for different numbers of clusters. We also perform the same analysis on a phoneme classification task (see Section 6) and SLS again outperforms ULS.

Regularization	α	β	# of Clusters	Mean± std	Lowest Err.
None	-	-	-	24.82±0.19	24.65
ULS	0.1	-	-	24.63±0.14	24.50
SLS	0.1	0.4	16	**24.55±0.09**	**24.49**
SLS	0.1	0.4	32	**24.48±0.11**	**24.29**
SLS	0.1	0.3	64	**24.63±0.10**	**24.48**
ULS	0.2	-	-	25.02±0.29	24.59
SLS	0.2	0.3	16	**24.53±0.06**	**24.45**
SLS	0.2	0.2	32	**24.42±0.07**	**24.33**
SLS	0.2	0.3	64	**24.73±0.12**	**24.50**
ULS	0.3	-	-	25.67±0.13	25.47
SLS	0.3	0.3	16	**24.77±0.06**	**24.67**
SLS	0.3	0.3	32	**24.55±0.03**	**24.53**
SLS	0.3	0.3	64	**24.82±0.13**	**24.68**

Table 2: Test error rate (%) for different cluster number chosen for the SLS on the synthetic data.

6 Experimental results on phoneme classification

We compare the performance of SLS to ULS on a speech dataset. Table 3 shows the mean error rate and the lowest error rate for SLS and ULS for a phoneme classification task on a randomly-subsampled version of the TIMIT dataset. We sample the original TIMIT training dataset and validation dataset to 50000 samples and 10000 samples respectively with ten phoneme classes. The ten classes are sil (silence), aa, ae, ah, ao, aw, ax, ay, b, and cl. We extract Mel-frequency cepstral coefficients (MFCC) features and train a 5-layer neural network for 200 epochs with the same initialization for SLS and ULS. The experiments are run 4 times and the analysis for the number of clusters is also performed. As shown in Table 3, SLS consistently outperforms ULS for different numbers of clusters and smoothing strengths alpha.

Regularization	α	β	# of Clusters	Mean± std	Lowest
None	-	-	-	30.63±0.23	30.43
ULS	0.1	-	-	30.24±0.19	30.04
SLS	0.1	0.8	32	**30.13±0.03**	**29.70**
SLS	0.1	0.8	64	**30.09±0.15**	**29.93**
SLS	0.1	0.6	128	**30.05±0.22**	**29.75**
ULS	0.2	-	-	30.09±0.18	29.95
SLS	0.2	0.2	32	**29.98±0.20**	**29.80**
SLS	0.2	0.2	64	**29.95±0.15**	**29.74**
SLS	0.2	0.4	128	**29.96±0.39**	**29.56**
ULS	0.3	-	-	30.19±0.19	29.96
SLS	0.3	0.2	32	**30.16±0.11**	30.10
SLS	0.3	0.4	64	**30.15±0.12**	29.97
SLS	0.3	0.4	128	**30.12±0.20**	**29.91**

Table 3: Test error rate (%) for the subsampled TIMIT dataset. Better algorithms are highlighted in bold for each α and number of cluster.

7 Algorithm for the SLS

In Algorithm 1 we provide a listing of the steps required to implement SLS. It is free to use any embedding methods and clustering methods in the DataEmbedding subroutine and the DataClustering subroutine.

Algorithm 1 Structural Label Smoothing

Input: training examples S and cluster number C

DataEmbedding(S)$\rightarrow \hat{S}$ where $\hat{S}$ is embedded data
DataClustering($\hat{S}$, C)$\rightarrow \hat{S}_c, w_c$ where $\hat{S}_c$ is embedded data falling in the cluster c and w_c is the weight of the cluster c.
Generate Euclidean minimum spanning tree (MST) on the $\hat{S}_c$ for each cluster c.
For each cluster c, set $C_{ij}(c)$ to be the number of edges in the MST connecting points from class i and j in cluster c.
if Binary class classification **then**
    Use the estimated lower bound using on the BER in (12) as a proxy for the BER.
else
    Use the estimated lower bound using on the BER in (13) as a proxy for the BER.
Set the average smoothing strength α and Bias error bias reduction strength β
Infer smoothing strength $\hat{\alpha}_c$ for each cluster c using (10).
Construct the loss function (14) of the structural label smoothing for network training.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.