Appendix A
Synthetic Datasets Experiments

The synthetic datasets were used in our study to validate empirically the limitations of standard label smoothing. In these experiments, we sampled 100 datasets using different random seeds from a generative model. Table V shows the average results of these experiments including our proposed instance-based label smoothing methods over the 100 datasets. The ECE and cwECE metrics were calculated at 15 equal-width bins. We also present the average hyper-parameter values corresponding to each method. These parameters were tuned to get the best validation cross-entropy loss on each sampled dataset. It can be easily observed that our proposed methods ILS1, ILS2 and ILS are outperforming both No LS and Standard LS in all the performance metrics. ILS is slightly more overfitted to the validation set than ILS1 as it has one more hyper-parameter tuned over the validation set. For this reason, ILS has slightly lower accuracy, cross-entropy loss and ECE than ILS1. We believe that increasing the validation set size would even enhance more the performance of ILS.

A. *Dataset Generative model*

The generative model was shown in Figure 1, where the sum of the probability density functions of six Gaussian circles, 2-D Gaussian distributions with equal variances, of three classes, is plotted. Each two of these Gaussian circles are used to represent one class in the dataset. The classes are overlapped with different degrees to show that some classes could be more similar to each other. Additionally, one Gaussian circle for each class has 80% of the probability density of this class generative model, representing more dense area of instances, while the other circle has only 20% density, representing a sparse area of instances. This generative model is designed to ensure that neural networks would overfit the sparse area more than the dense area. Hence, using different smoothing factors is better for different feature space regions. Each training set includes 50 instances per class (150 instances in total). The validation set has the same size as the training set, while the testing set has 5000 instances per class. The instances of each class are randomly drawn out of the distribution of the two Gaussian circles of the corresponding class using SciPy multivariate normal sampler [3].

B. *Network Training*

A feed-forward neural network architecture was trained on each of the datasets in various modes (No LS, Standard LS, ILS) in addition to the Bayes-Optimal model computed given the generative model of the datasets. The network architecture consists of five hidden layers of 64 neurons each. ReLU activations were used in all the network layers. We used Adam optimizer with 10^{-2} learning rate. The learning rate was selected based on the learning rate finder [4] results along a range of 10^{-7} to 1. The networks were trained for a maximum of 500 epochs and an early stopping of 10 epochs. The networks from the epoch of the minimum validation loss were finally used in the evaluation.

C. *Hyper-parameter tuning*

We used the set of $\{0.001, 0.005, 0.01, 0.03, 0.05, ..., 0.19\}$ for the possible smoothing factor values (ϵ) for LS and ILS2 methods. Also, the temperature value used with the teacher network final softmax in ILS2, and ILS was tuned from the set $\{1, 2, 4, 6\}$. In ILS1 and ILS, we set values of $\{0.75, 0.775, 0.8, 0.825, 0.85\}$ and $\{0.75, 1, 1.25, 1.5, 1.75, 2\}$ for the hyper-parameters P_1 and P_2, respectively. The best ϵ , P_1, P_2, and/or temperature values were selected for each dataset based on the best validation cross-entropy loss.

D. *Self-distillation Experiments*

In this set of experiments, we distilled a student network from a teacher of the same architecture but trained with different modes (NoLS, LS, ILS). The student was trained using the same optimizer and learning rate as the teacher network to minimize the log-loss with the teacher network predictions. The temperature values of both the student and teacher networks were set to 1. The experiment was repeated for the 100 network replicas trained on the 100 sampled training sets as illustrated previously. The average results are reported in Table VI including the teacher trained with ILS.

Appendix B
Smoothing Factor Value in ILS

We generated multiple smoothing factor curves that can be used to decide the values of the smoothing factor corresponding to different true class predictions. As explained in Section IV-A, the aim of these curves is to find the suitable amount of smoothing for each group of instances such that the network can predict similar true class probabilities as the Bayes-Optimal model. Figure 6 shows examples for the smoothing factor curves given the Bayes-Optimal predictions on synthetic datasets. To ensure that a similar behavior could be obtained with different experiment setup, we changed the neural network capacity, the training set size, and the separation between classes to observe the shape of these curves. First, we generated the smoothing factor curve on the same synthetic dataset experiment setup used in the main paper (Normal Synthetic Data Setup). Then, we tried out different variants of this setup like using a shallow network of a single hidden layer on the same generative synthetic dataset (Shallow Network), using larger training set of 200 instances per class instead of the originally used 50 instances (Bigger Training Set), and using different separation distances among the classes of the generative model of the dataset as shown in Figure 5 (Different Synthetic Dataset). We observed that the behavior of the obtained curves could be well approximated using a quadratic function of a shift parameter (P_1) on the x-axis and coefficient (P_2) representing the slope of the quadratic term.

Since knowing the Bayes-Optimal model is practically impossible, we also plotted the same smoothing factor curves

[3]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.multivariate_normal.html

[4]https://github.com/davidtvs/pytorch-lr-finder

TABLE V

COMPARISON BETWEEN THE EVALUATED METHODS. ACC DENSE/SPARSE REPRESENT THE ACCURACY ON THE TEST INSTANCES SAMPLED FROM THE DENSE/SPARSE REGIONS SEPARATELY. THE AVG HYPER-PARAMETER VALUES REPRESENT THE AVERAGE VALUES OF THE DIFFERENT PARAMETERS CORRESPONDING TO EACH METHOD OVER THE 100 SAMPLED SYNTHETIC DATASETS.

				Test Performance						**Avg Hyper-parameter Values**			
Method	**Temp Scaling**	**Avg Temperature**	**Validation Loss**	**Acc**	**Acc Dense**	**Acc Sparse**	**Cross Entropy**	**ECE**	**cwECE**	**Scaling Temp**	ϵ	P_1	P_2
Bayes-Optimal	No	1.000	0.4423	*81.98%*	82.94%	78.14%	*0.4444*	*0.0067*	*0.0234*	-	-	-	-
No LS	No	1.000	0.5222	78.45%	81.43%	66.54%	0.5589	0.0426	0.0668	-	-	-	-
No LS	Yes	1.202	0.5092	78.45%	81.43%	66.54%	0.5446	0.0311	0.0632	-	-	-	-
Standard LS	No	1.000	0.4851	79.44%	81.17%	72.51%	0.5318	0.0293	0.0723	-	0.061	-	-
Standard LS	Yes	0.946	0.4797	79.48%	81.24%	72.44%	0.5238	0.0263	0.0701	-	0.079	-	-
Standard LS-0.2	No	1.000	0.5073	79.39%	81.11%	72.53%	0.5439	0.0650	0.0785	-	0.200	-	-
Standard LS-0.2	Yes	0.804	0.4788	79.39%	81.11%	72.53%	0.5301	0.0241	0.0707	-	0.200	-	-
ILS1	No	1.000	0.4700	**80.18%**	81.49%	74.93%	**0.5041**	**0.0235**	0.0495	-	-		
ILS1	Yes	0.942	0.4654	80.16%	81.48%	74.88%	0.5052	0.0268	0.0497	-	-		
ILS2	No	1.000	0.4804	79.56%	81.50%	73.37%	0.5232	0.0287	0.0496	11.48	0.0494	-	-
ILS2	Yes	0.906	0.4744	79.58%	**81.54%**	73.02%	0.5201	0.0259	0.0494	9.63	0.0698	-	-
ILS	No	1.000	0.4628	80.13%	81.42%	**74.97%**	0.5089	0.0269	0.0449	2.91	-	1.63	0.80
ILS	Yes	1.033	0.4618	80.14%	81.45%	74.91%	0.5087	0.0278	**0.0446**	2.73	-	1.59	0.81

TABLE VI

COMPARISON BETWEEN USING TEACHER TRAINED WITH/WITHOUT LS IN SELF-DISTILLATION.

	Student Performance			
Teacher	**Accuracy**	**CrossEntropy**	**ECE**	**cwECE**
Bayes-Optimal	*81.92%*	*0.4451*	*0.0137*	*0.0338*
No LS	78.47%	0.5592	**0.0405**	**0.0649**
Standard LS	78.43%	0.5621	0.0408	0.0726
ILS	**79.57%**	**0.5201**	0.0473	0.0661

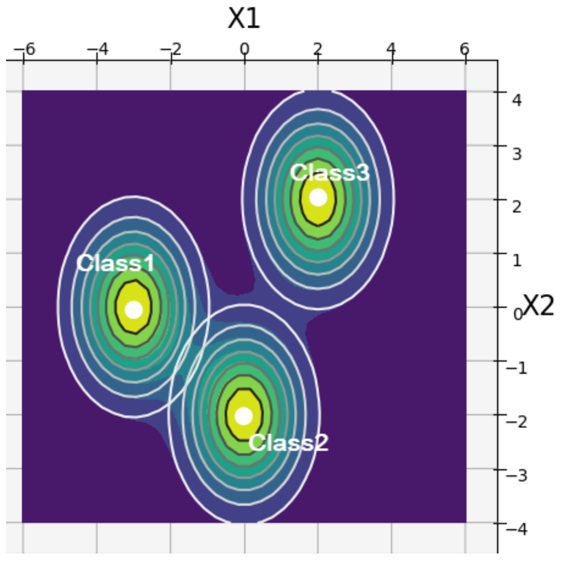

Fig. 5. Heat-map of the total density function of a generative model of 3-classes synthetic dataset with different separation between the classes.

given the No-LS with temperature scaling network predictions on the same synthetic dataset setup as shown in Figure 7. Additionally, we generated the same curves on different multi-class real datasets with different number of instances, and classes obtained from OpenML [5]. A sample of these curves is shown in Figure 9 and others could be found in our repository [6].Interestingly, similar shapes for the smoothing factor curves to that shown in Figure 6 could be still observed. There could be some observable mismatches that can not be approximated using the quadratic family of functions especially on the curves of the real datasets. These mismatches occur in regions with very few instances and thus irrelevant in practice. These regions are mostly the left-side parts of the mapping curves (predicted probabilities less than 0.5). Figures 8 and 10 show histograms of predicted probabilities of the actual class on the normal synthetic dataset setup by the Bayes-Optimal and NoLS+TempS networks, and some real datasets by the NoLS+TempS network. It can be observed that the majority of the predictions are highly confident. For instance, in volcanoes dataset, all the true class probability predictions are > 0.94. We choose the No-LS with temperature scaling network as it showed the best class-wise calibration performance. Hence, we will choose the instance-specific smoothing factor value based on the true class prediction of this network.

To select good ranges for P_1 and P_2, we performed a grid search on a long range of values ranging from 0.7 to 0.975 for P_1, and from 1 to 10 for P_2 on the synthetic datasets. The summary of the this experiments is shown in Figure 11, where the plot shows the average cross-entropy loss on the testing set for each combination of P_1 and P_2 in ILS1. This plot helped in narrowing down the promising ranges of P_1 and P_2 in our evaluation experiments on the synthetic data. Surprisingly, the best hyper-parameter settings ($P_1 = 0.8$, $P_2 = 2$), in Figure 11, were found to be approximating the smoothing factor curve in Figure 4 to a very good extent. The scatter plot also confirms the good selection for the quadratic family of functions in Equation 3 to approximate the smoothing factor curves because changing the values of either P_1 or P_2 from the closest matching function parameters ($P_1 = 0.8$, $P_2 = 2$) leads to increase in the test loss. Hence, it would be easy to choose a suitable function out of this family by tuning the values of P_1 and P_2 on a separate validation set.

[5]https://www.openml.org/

[6]https://github.com/mmaher22/Instance-based-label-smoothing

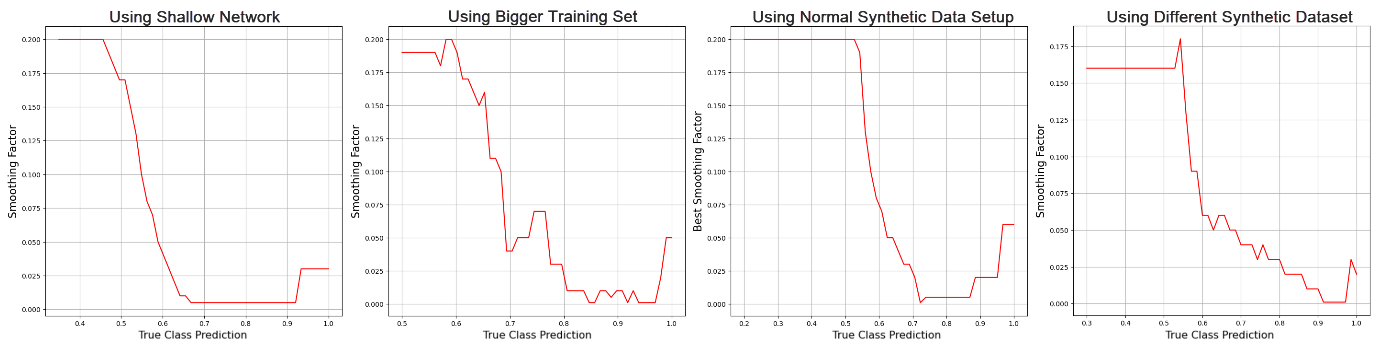

Fig. 6. Examples for the best smoothing factor curves corresponding to the predicted probability of the true class of the Bayes-Optimal model on different synthetic dataset setup.

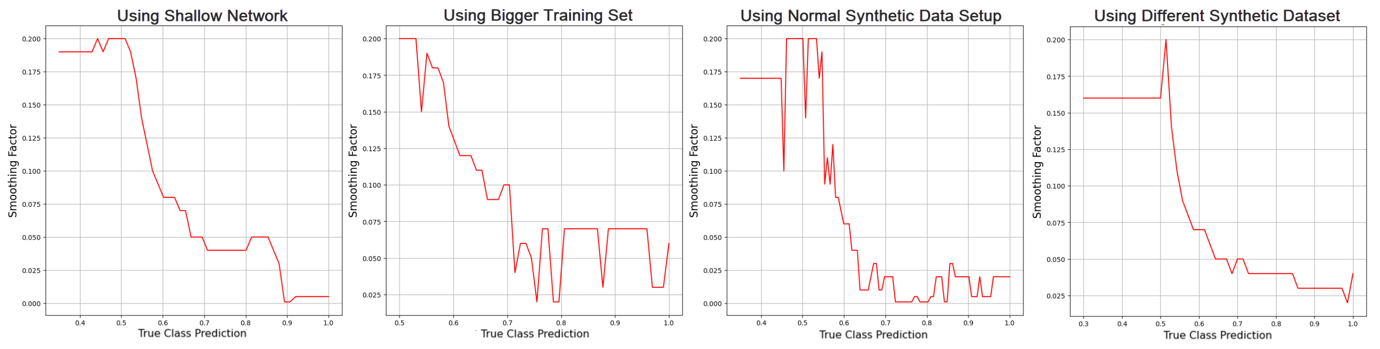

Fig. 7. Examples for the best smoothing factor curves corresponding to the predicted probability of the true class of the No-LS network + Temperature Scaling on different synthetic dataset setup.

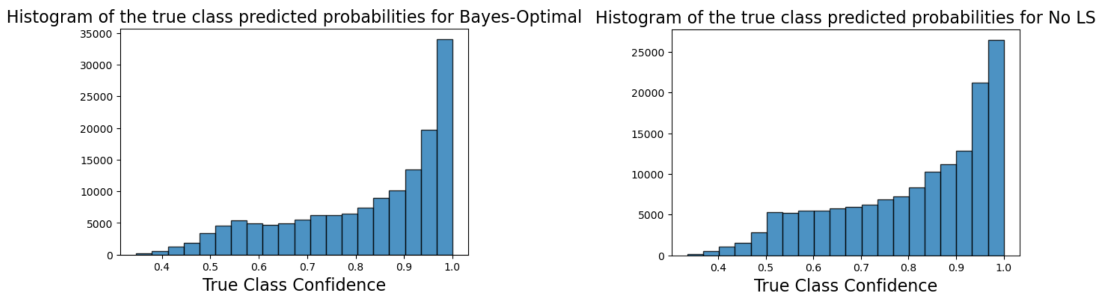

Fig. 8. Histograms of the true class predicted probabilities for BO, NoLS+TempS models on the normal synthetic dataset setup.

Appendix C
Real Datasets Experiments

In this set of experiments, we used the CIFAR10/100 [7] and SVHN [8] datasets. For all datasets, we used the original training and testing splits of 50K and 10K images respectively for CIFAR10/100, and 604K and 26K images for SVHN dataset. The training splits were further divided into (45K-5K) images and (598K-6K) images as training and validation splits for CIFAR10/100 and SVHN, respectively. For data augmentation, we normalized both CIFAR10/100 and SVHN datasets and we used horizontal flips and random crops for the training splits of CIFAR10/100. The experiments were repeated two times with different seeds for the whole training process including the training/validation splitting, and the average results are reported. The critical difference diagrams for Nemenyi two tailed test given significance level $\alpha = 5\%$ for the average ranking of the evaluated methods and the number of tested datasets in terms of test accuracy, cross-entropy loss, ECE and cwECE are summarized in Figures 12,13, 14 and 15, respectively.

[7]https://www.cs.toronto.edu/~kriz/cifar.html
[8]http://ufldl.stanford.edu/housenumbers/

A. *Architectures*

In these experiments, we trained the networks following their original papers as described below. The smoothing factor

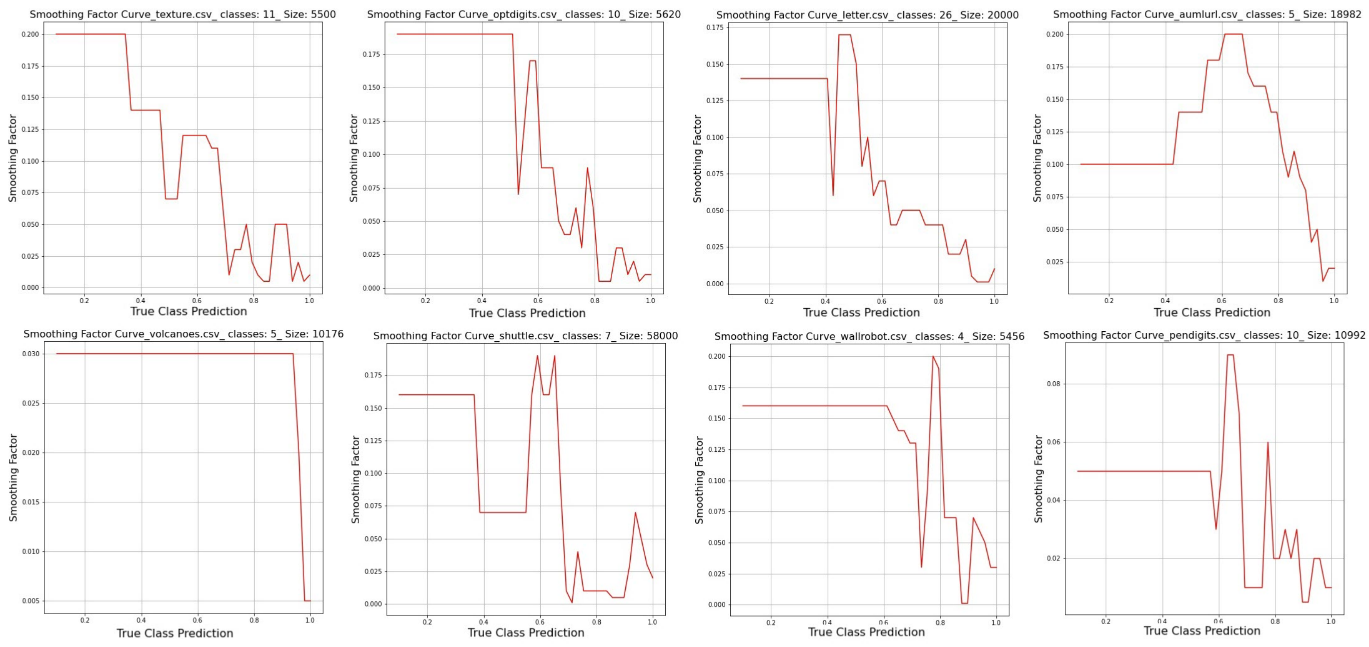

Fig. 9. Examples for the best smoothing factor curves corresponding to the predicted probability of the true class of the No-LS network + Temperature Scaling on different real datasets.

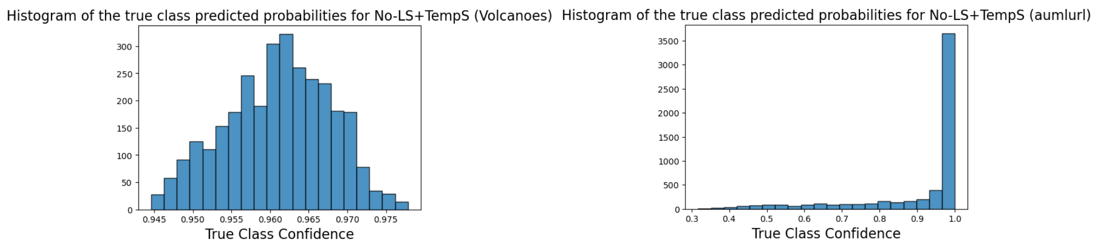

Fig. 10. Histograms of the true class predicted probabilities for BO, NoLS+TempS models on the volcanoes and amul-url datasets.

ϵ and the temperature scaling factor of the teacher network in the proposed method ILS2 were tuned from the sets $\{0.01, 0.05, 0.1, 0.15, 0.2\}$ and $\{1, 2, 4, 8, 16\}$, respectively. Also, the hyper-parameters P_1 and P_2 in ILS1 were tuned from the sets $\{0.975, 0.95, 0.925, 0.9, 0.85\}$ and $\{0.75, 1, 1.25, 1.5, 2\}$, respectively. We selected the epoch with the best validation loss to report the results.

a) LeNet: The LeNet model was used in the CIFAR10/100 experiments. We used the open source implementation of LeNet available in [9]. The network was trained for 300 epochs using the stochastic gradient descent (SGD) optimizer with 0.9 momentum and an initial learning rate of 0.1 that is reduced by a pre-defined scheduler with a rate of 10% at the 60^{th}, 120^{th}, and 160^{th} epochs. The mini-batch size and the weight decay were set to 128 and 0.0001, respectively.

b) DenseNet: The DenseNet model was used with the general CIFAR10/100 datasets experiments, and knowledge distillation experiments on Fashion-MNIST. We used the open source implementation of ResNet available in [10]. The network was trained for 300 epochs using the stochastic gradient descent (SGD) optimizer with 0.9 momentum and an initial learning rate of 0.1 that is reduced by a pre-defined scheduler with a rate of 10% at the 150^{th} and 225^{th} epochs. The mini-batch size and the weight decay were set to 64 and 0.0001, respectively.

c) ResNet: The ResNet model was used with the general CIFAR10/100 datasets experiments, and knowledge distillation experiments. We used the open source implementation of ResNet available in [11]. The network was trained for 200 epochs using the stochastic gradient descent (SGD) optimizer with 0.9 momentum and an initial learning rate of 0.1 that is reduced by a pre-defined scheduler with a rate of 10% at the 80^{th} and 150^{th} epochs. The mini-batch size and the weight decay were set to 128 and 0.0001, respectively.

[9]https://github.com/meliketoy/wide-resnet.pytorch/blob/master/networks/lenet.py

[10]https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py

[11]https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

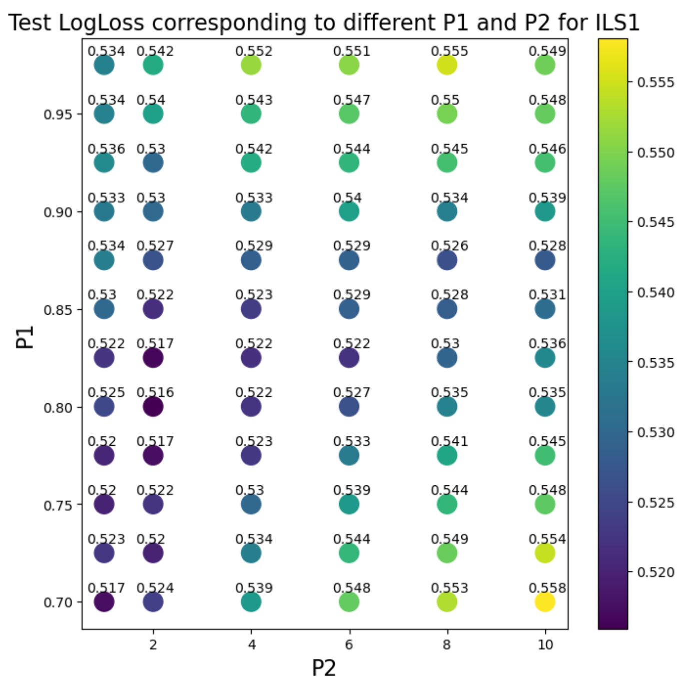

Fig. 11. The testing cross-entropy of ILS1 corresponding a grid of P_1 and P_2 parameters for the synthetic datasets.

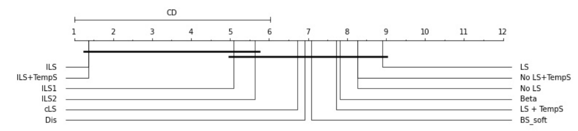

Fig. 12. The Critical Difference diagram of different methods on the real datasets in terms of the test accuracy.

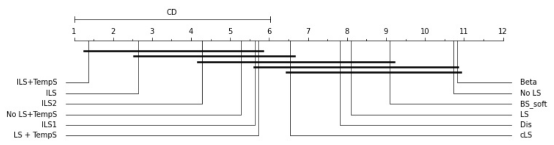

Fig. 13. The Critical Difference diagram of different methods on the real datasets in terms of the test log-loss.

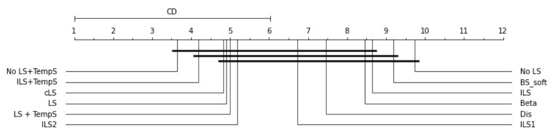

Fig. 14. The Critical Difference diagram of different methods on the real datasets in terms of the test ECE.

d) ResNet with Stochastic Depth: The ResNet with Stochastic Depth model (ResNetSD) was used in the CIFAR10/100 and SVHN experiments. We used the open source implementation of ResNetSD available in [12]. The network was trained for 500 epochs using the stochastic gradient descent (SGD) optimizer with 0.9 momentum and an initial learning rate of 0.1 that is reduced by a pre-defined scheduler with a rate of 10% at the 250^{th} and 325^{th} epochs. The mini-batch size and the weight decay were set to 128 and 0.0001, respectively.

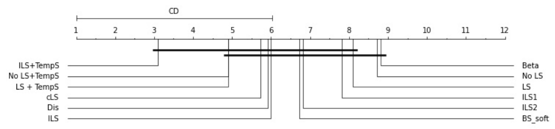

Fig. 15. The Critical Difference diagram of different methods on the real datasets in terms of the test cwECE.

e) Wide ResNet: The Wide ResNet model (ResNetW) was used in the CIFAR10/100 experiments. We used the open source implementation of ResNetW available in [13]. The network was trained for 200 epochs using the stochastic gradient descent (SGD) optimizer with 0.9 momentum and an initial learning rate of 0.1 that is reduced by a pre-defined scheduler with a rate of 20% at the 60^{th}, 120^{th}, and 160^{th} epochs. The mini-batch size, growth rate and the weight decay were set to 128, 10 and 0.0005, respectively.

f) InceptionV4: The Inception model was used with the knowledge distillation experiment on CIFAR10 dataset. We used the open source implementation of InceptionV4 available in [14]. The network was trained for 200 epochs using the stochastic gradient descent (SGD) optimizer with 0.9 momentum and an initial learning rate of 0.1 that is reduced by a pre-defined scheduler with a rate of 10% at the 30^{th}, 60^{th}, and 90^{th} epochs. The mini-batch size and the weight decay were set to 256 and 0.0001, respectively.

B. Knowledge Distillation Experiments

In these experiments, we distilled multiple student AlexNets from different teacher architectures trained with/without LS or with our proposed ILS. We used the default training and testing splits of CIFAR10/100 and Fashion-MNIST datasets. The experiments were repeated 3 times with different random seeds and the average results are reported. We used the open-source implementation available in [15]. During distillation, we trained the student network by minimizing a weighted average of the log-loss between the student and the teacher outputs, and the log-loss between the student and the hard target labels with ratios of 0.9 and 0.1, respectively. using Adam optimizer. We set the initial learning rate to 10^{-2} and the mini-batch size to 256. The network was trained for 150 epochs and the learning rate was reduced using a predefined scheduler by 10% after each 150 iterations. The temperature values were set to 20 and 1 for the teacher and student networks, respectively.

[12]https://github.com/shamangary/Pytorch-Stochastic-Depth-Resnet
[13]https://github.com/meliketoy/wide-resnet.pytorch
[14]https://github.com/weiaicunzai/pytorch-cifar100
[15]https://github.com/peterliht/knowledge-distillation-pytorch