Identifying Wrongly Predicted Samples: A Method for Active Learning

Rahaf Aljundi Nikolay Chumerin Daniel Olmeda Reino

Toyota Motor Europe

1. Introduction

These supplementary materials are structured as follows: Section 2: detailed derivation of our kernel for multi-class classification; Section 3: estimation of the computational cost of our criterion in comparison to other criteria; Section 4: additional details on experimental setting; Section 5: extra empirical analysis and additional results on both balanced and imbalanced settings; Section 6: details on deployment setting of semantic segmentation experiments.

2. Derivation

In the main paper, Section 3.1, we consider an example of a multi-class classification problem with a softmax cross entropy loss and construct our kernel as follows:

$$K_\theta(x_i, x_j) = (\nabla_\theta f(x_i;\theta)^\intercal (p_i - y_i)) \cdot (\nabla_\theta f(x_j;\theta)^\intercal (p_j - y_j)) . \quad (1)$$

Here is the derivation of this kernel. Starting from

$$K_\theta(x_i, x_j) = ((\nabla_\theta f(x_i;\theta))^\intercal \ell') \cdot ((\nabla_\theta f(x_j;\theta))^\intercal \ell') . \quad (2)$$

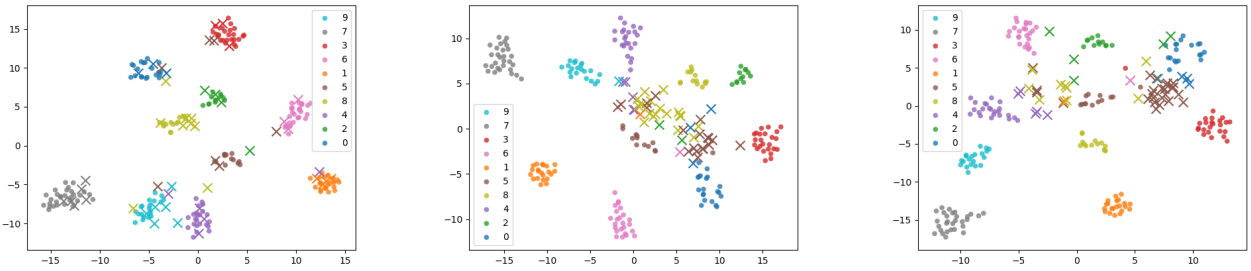
The output of the softmax function for each class c is defined as $p^c = \frac{\exp(f^c(x;\theta))}{\sum_k \exp(f^k(x;\theta))}$, which gives the probability of a class c given $f^c(x;\theta)$ the output of the class c logit. We define $p = [p^1, \dots, p^k]^\intercal$ as the output probability vector for sample x. The cross entropy loss of one sample can be then written as $\ell = -y \cdot \log(p) = \sum_c -y^c \log(p^c)$, where $y = [y^1, \dots, y^k]^\intercal$ is a one-hot encoded ($y^c \in \{0, 1\}, \sum_c y^c = 1$) vector. Following this, the derivative of the softmax cross entropy loss w.r.t. the neural network function output $f(x;\theta)$ is defined as $\ell' = p - y$. By replacing the derivative of the loss in (2) we get the kernel in (1).

3. Computational Cost Estimation

In this section, we compare the computational cost of estimating our criterion, Eq.3 in the main paper, and our approximated criterion, Eq.10 in the main paper, to that of estimating the criterion of `MC-Dropout` [4]. The comparison is done on per sample basis i.e., we will estimate the computational cost of estimating each criterion for a given pool sample. Let's denote the cost of the forward pass of a neural network with C and assume that the backward pass is roughly of a similar cost. `MC-Dropout` [4] requires multiple forward passes to estimate the model uncertainty of predictions (Monte Carlo sampling). Given number of samples n, the cost of estimating the `MC-Dropout` criterion, assuming we forward through the full network, is approximately nC. For `IWPS`, we limit the minimization of the loss to few iterations $\mathcal{T}$. As such, the minimization of the loss requires approximately $2\mathcal{T}C$. In addition, we need to estimate the loss on the holdout set after the pseudo loss minimization. We use a small holdout set that can be combined in one mini batch of size N^h and its loss can be estimated with cost $\approx N^h C$. Following this, `IWPS` computational cost is approximately $2\mathcal{T}C + N^h C$. For `IWPS-app`, it requires mainly computing the gradients of the pool sample which will be of cost $2C$, note that the gradient of the holdout set $\nabla_\theta \ell_v(\theta^s)$ needs to be pre-computed only once for all the pool samples and can be stored from the last validation step, hence not considered here. Further, when limiting the criterion estimation to the last layer, which presumably has a forward pass cost close to $\frac{1}{L}C$ with L the number of layers in the deployed neural network, the cost of `IWPS` will be $\left(N^h + 1 + \frac{1}{L} + \frac{2(\mathcal{T}-1)}{L}\right) C$, as we only need to propagate the sample through the full network once, with the cost of the first iteration $(1 + \frac{1}{L})C$. For `IWPS-app` the approximate cost will be $(1 + \frac{1}{L})C$.

4. Deployment Settings

For our method `IWPS`, in all experiments we limit the number of iterations of pseudo loss minimization to 3 iterations and use a learning rate of 0.001 for the fully connected network and 0.0001 for ResNet. These parameters were set based on the accuracy of selecting wrongly predicted samples in an initial experiment on the initial training sets. For both `MC-Dropout` and `BALD`, we use Monte Carlo sampling with $n = 10$ number of samples. For `MC-Dropout`, we use uncertainty, as defined in [4], as the ranking crite-



(a) (b) (c)

Figure 1: tSNE visualization of initial training samples and selected pool samples on MNIST benchmark with 200 training samples and 50 selected as annotation step. (a) `Random` (b) `IWPS` (c) `IWPS-app`. Dots are training data while each cross ($\times$) represents a selected pool sample.

Figure 2: Examples of selected pool samples using `IWPS` criterion, based on: MNIST, KMNIST, SVHN and Cifar10 balanced datasets from top to bottom.

rion. Dropout is placed on each fully connected layer, for both architectures, with $p = 0.2$. For `Coreset`, we use the features of the last layer and the greedy solver[1].

On the different studied datasets, models were trained using ADAM optimizer and a learning rate of 0.001 with early stopping on the validation set. We use a mini batch size of 50 for the fully connected network and 64 for ResNet. We split the training set into train, validation and pool and report results on the test set.

[1] github.com/google/active-learning/blob/master/sampling˙methods/kcenter˙greedy.py

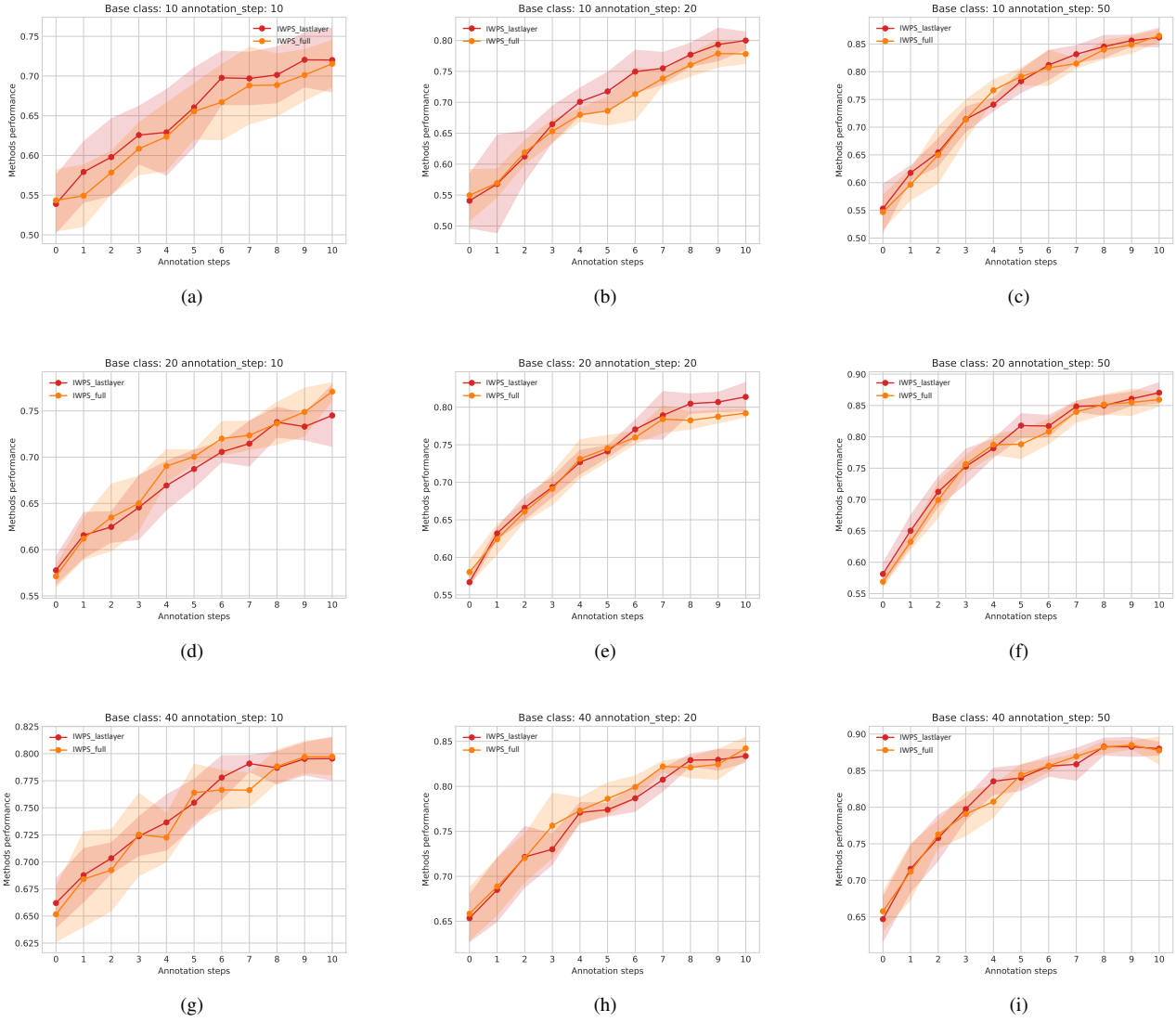


Figure 3: Performance (mean acc. and std. dev.) of IWPS on different configurations of MNIST imbalanced benchmark when optimizing the pseudo loss, Eq.2 in the main paper, on the last layer only vs. the full network.

5. Additional Experiments and Ablation on Image Classification

5.1. Comparison of Optimizing the Pseudo Loss on the Full Network vs. the Last Layer

In our experiments, we limit the optimization of the loss of each pool sample given its pseudo label, Eq.2 in the main paper, to the classification layer. Here, we compare optimizing the full network to the optimization of the last layer only. Figure 3 shows the performance of both variants on MNIST imbalanced benchmark with different base class sizes (10, 20, 40) and varied annotation step sizes. As it can be seen there is no significant difference in our method performance when the optimization of the pseudo loss is limited to the last layer as opposed to the full model.

5.2. Holdout Set Size and Alternatives

In the main paper, we show an ablation of the holdout size size on SVHN imbalanced setting. Figure 7a and 7b report the results of using different holdout set sizes on the MNIST balanced and imbalanced settings respectively. Similar to the notes made in the main paper, only the very smallest size, one sample per class deteriorates the performance of IWPS. However, there is no significant difference between the other sizes.

We have also discussed deploying a subset of the train-

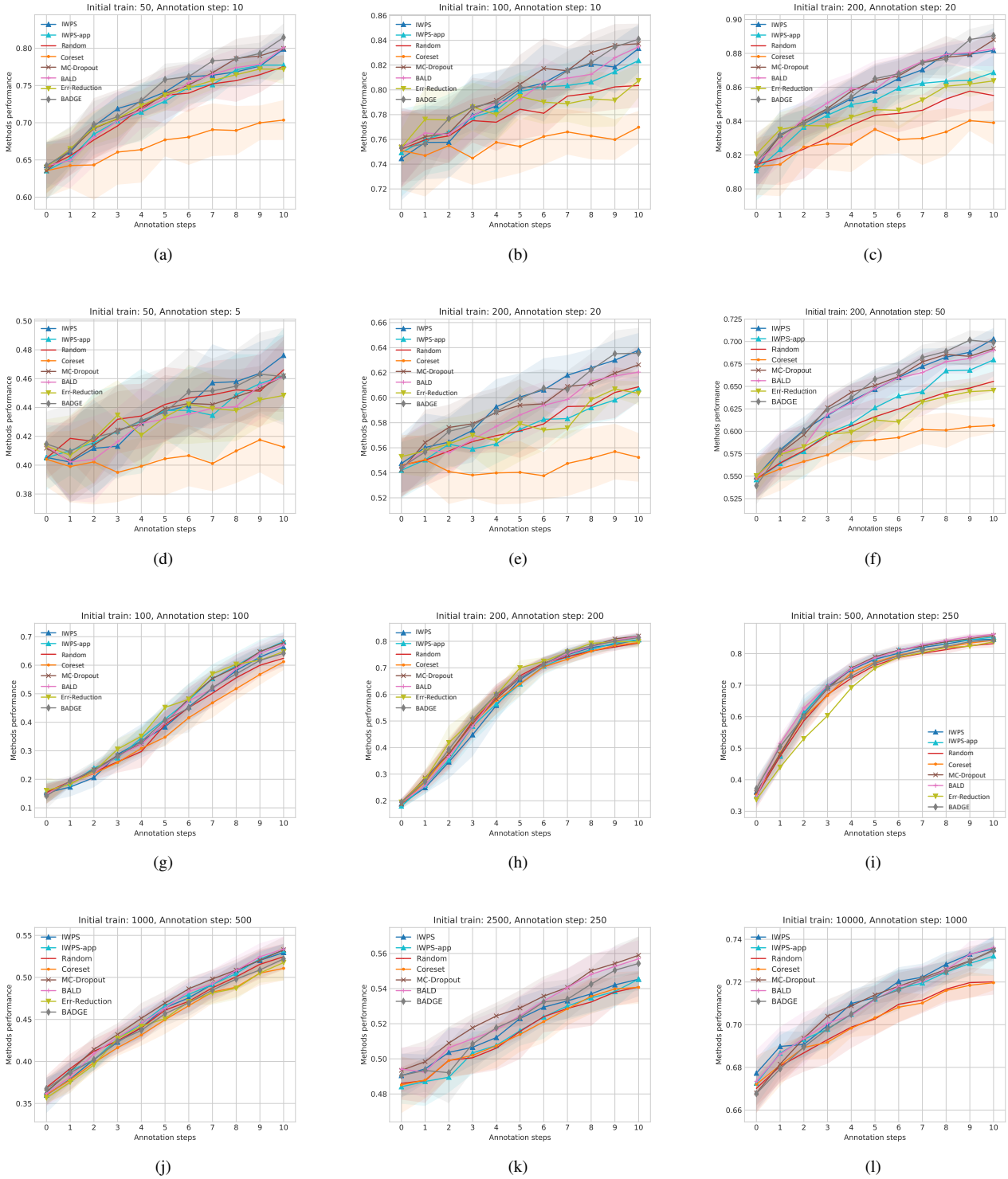


Figure 4: Additional results on different configurations of the balanced setting (mean acc. and std. dev.). First row shows MNIST, second shows KMNIST, third row shows SVHN and fourth row shows Cifar10 results. For Cifar10, unfortunately not all results of BADGE [1] and Err-Reduction [6] were ready.

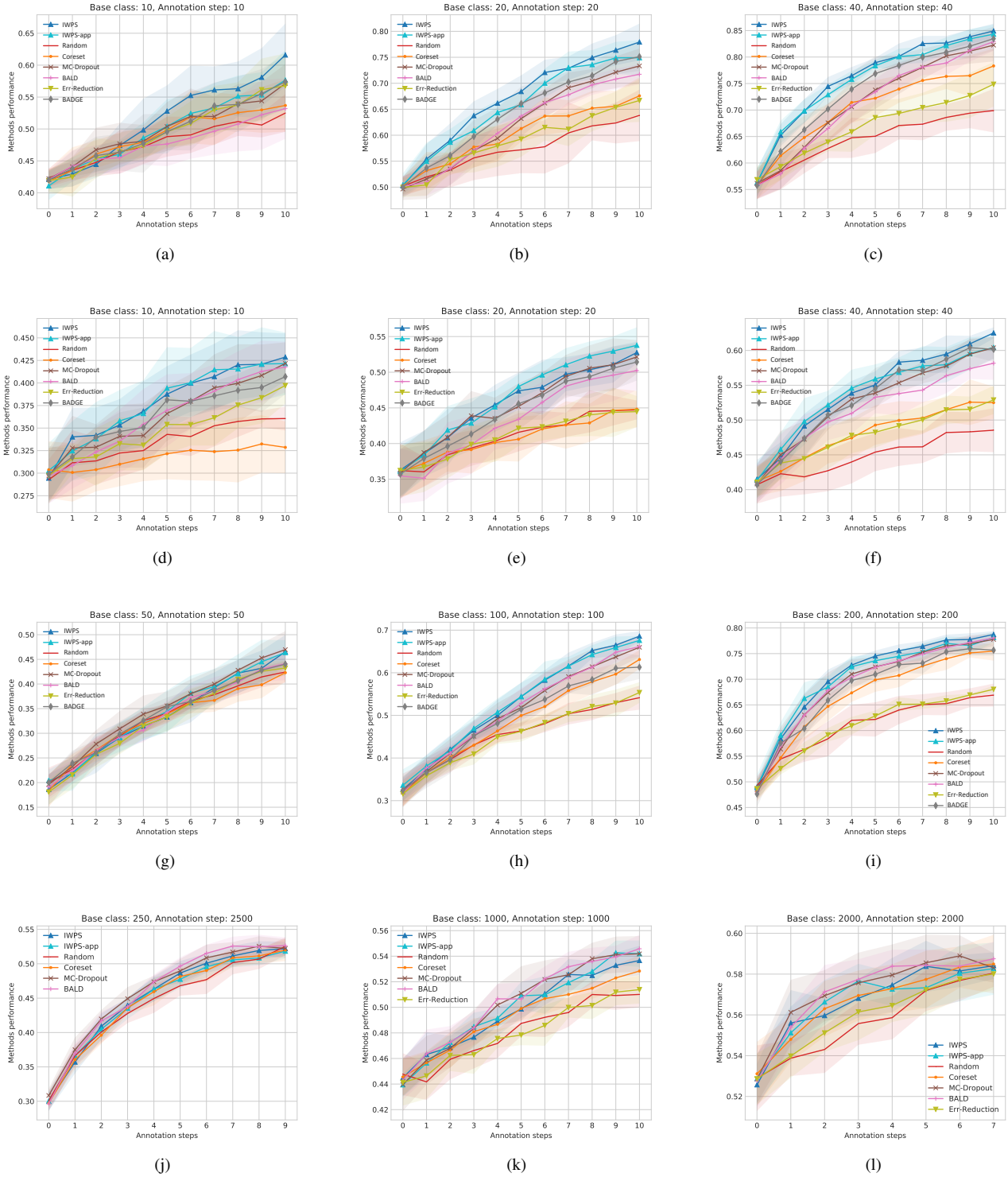
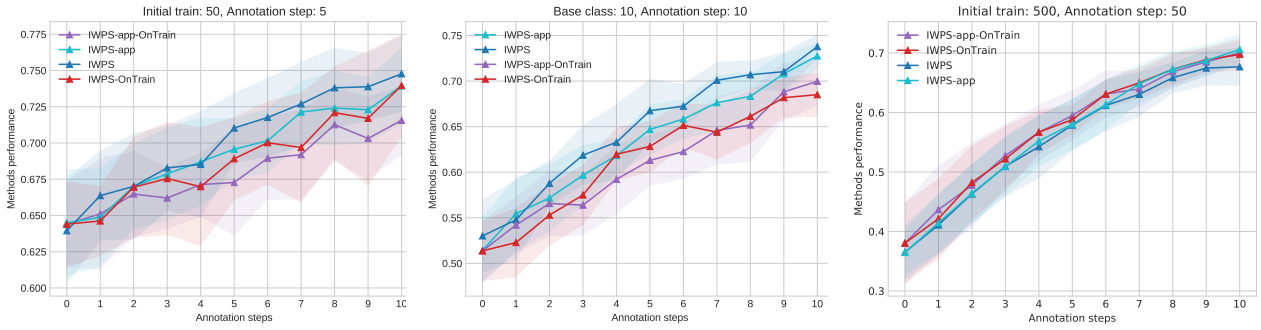
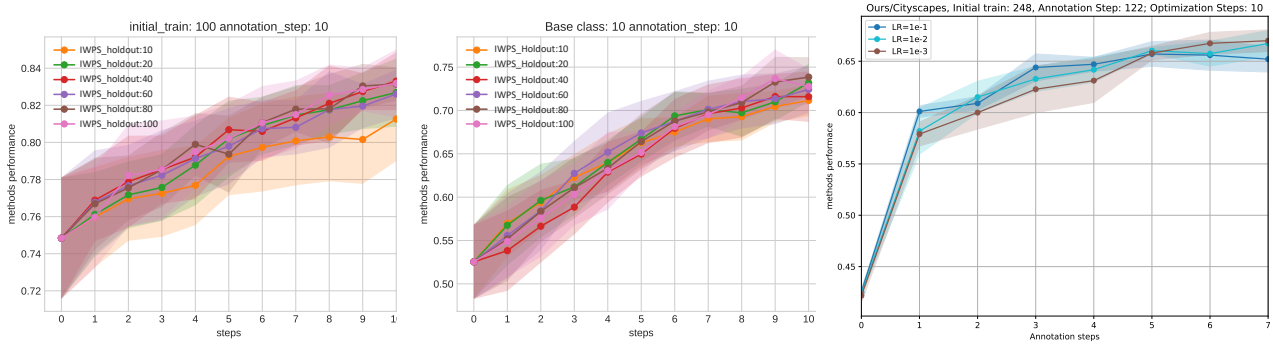


Figure 5: Methods performance (mean acc. and std. dev.) with imbalanced rate of $1/20$. First row shows MNIST, second shows KMNIST, third row shows SVHN and fourth row shows Cifar10. For Cifar10, unfortunately not all results of BADGE [1] and Err-Reduction [6] were ready.



(a) (b) (c)

Figure 6: Mean accuracy and std.dev. of `IWPS` and `IWPS-app` when using a holdout set versus a subset on the training set. (a) MNIST balanced, (b) MNIST imbalanced and (c) SVHN balanced settings.



(a) (b) (c)

Figure 7: (a),(b) ablation on the size of the holdout set effect for MNIST balanced and imbalanced setting respectively. (c) ablation on the effect of the learning rate of the loss minimization step in `IWPS`.

ing set instead of a holdout set and showed the effect only on SVHN imbalanced setting due to space limit. Here we report the results of deploying a random subset of the training set versus the use of a holdout set on SVHN balanced setting 6c, in addition to MNIST (6a) balanced and MNIST imbalanced (6b) settings. As we discussed in the main paper, the differences between the two configurations are more pronounced on MNIST dataset possibly due to the simplicity of this dataset leading to zero training error especially when training on small sets as in the case of these experiments $50 - 100$.

5.3. Visualization of Selected Samples

To visualize the selected samples by both our methods, in comparison to the random selection, we use tSNE visualization of features before the last classification layer of the model trained on the initial training set. On MNIST balanced benchmark, Figure 1 shows the selected samples and the initial training points for `Random`, `IWPS` and `IWPS-app`. While `Random` selects samples that are similar to the training data and can be easily predicted by the current model, both `IWPS` and `IWPS-app` select in regions between different training points clusters, where samples are likely to be mispredicted. By adding these selected samples to the training data, the model would learn better separable features and more accurate decision boundaries.

Examples of selected images. Figure 2 shows example of top ten selected images by `IWPS` on different datasets based on the balanced setting.

5.4. Additional Results on Balanced Setting

Figure 4 presents additional results on the balanced setting for MNSIT, KMNIST, SVHN and Cifar10 datasets. As it can be seen our both variants show competitive performance, however, differences between the compared methods, in general, are not substantial.

5.5. Larger Imbalanced Rate

In the main paper, we present results on imbalanced benchmarks with an imbalanced rate of $1/10$, i.e. the underrepresented classes appear $1/10$ in comparison to the other categories. Here we repeat the same experiments with a larger imbalanced rate of $1/20$, and use a smaller balanced validation set of size $1/10$ to that of the initial train set. Figure 5 shows the performance of the compared methods on different sizes of the initial training set with varied annotation step sizes. As it can be seen that `IWPS` has significantly

better performance than other studied methods on MNIST, KMNIST and SVHN. Our improvement reaches a margin of 8%. Additionally, `IWPS-app` compares favorably to the other methods. This illustrates the ability of our method to overcome the imbalanced nature of the data and account for the under-represented categories.

6. Semantic Segmentation Settings

Dataset. We apply the proposed methods on Cityscapes [3]. It provides fine-grained labels of 19 different classes in 2945 images for training and 500 images for evaluation. We further split the training set by randomly drawing 248 and 2479 samples for our initial training and pool sets and 248 for validation. At each annotation step, we select 122 samples from the pool set and add them to the training set. The evaluation split is reserved and used as a test set.

Implementation. All methods follow the same training procedure. The deployed architecture is Deeplabv3 [2] using Resnet-50 [5] as a backbone, without auxiliary loss, initialized with Imagenet pretraining [7]. At each annotation step, we resume the training. The models are trained on random crops of 712×712, after rescaling the input images so that the maximum dimension is 2048, with batch size of 24. We apply random horizontal flipping of images and labels during training. The models are optimized using SGD with momentum 0.9, weight decay 10^{-4}, polynomial learning rate decay $\eta_i = \eta_0 \cdot (1 - \frac{i}{i_{\max}})^{\gamma}$, where η_i is learning rate at iteration i, $\eta_0 = 10^{-2}$, $\gamma = 0.9$. Optimization was stopped early if the validation loss did not improve for five consecutive epochs, for a maximum of 50. All experiments are averaged over five runs.

For `IWPS`, the minimization over the pseudo label (Eq. 2 in the main paper) is done using SGD without a momentum or weight decay and with a constant learning rate 10^{-2}, for $\mathcal{T} = 10$ steps per sample. Figure 7c shows the effect of the learning rate in the pseudo loss minimization (Eq. 2 in the main paper) on the mIoU after each annotation run. Large learning rate values improve the model in the first iterations. Smaller values translate to slower but more stable improvements, after a number of iterations. For `IWPS-app` 10 samples are randomly drawn from the hold-out set to compute the average gradient direction. For `MC-Dropout`, we use Monte Carlo sampling with $n = 10$ number of samples, and activate the dropout layer present in the network, with $p = 0.1$.

References

[1] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[5] 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.

[6] N Roy and A McCallum. Toward optimal active learning through sampling estimation of error reduction. int. conf. on machine learning, 2001.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.