General response: We thank all reviewers for their constructive comments. We think there are some **misunderstandings** and we encourage the reviewers to kindly consider our response.

Reviewer # 1

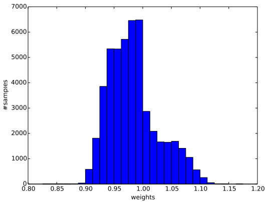
Q1: Parameter minimizing the full loss may not minimize the supervised loss. Note, "supervised loss" can be computed on the "training" or "validation" set, *i.e.*, $\mathcal{L}_S(\mathcal{D}, \theta)$ or $\mathcal{L}_S(\mathcal{V}, \theta)$. In the constraint, the program optimizes for the optimal model parameters $\theta^*(\Lambda)$ given a particular Λ, while using the "training" set's supervised loss. In the main objective, the program optimizes for Λ based on the supervised loss of the "validation" set. Note the difference in datasets. Intuitively, we look for the optimal parameters on the training set while considering the validation set result. This formalizes manual tuning of hyperparameters in prior work. Hence the bi-level optimization is reasonable.

Q2: Trivial solution $\Lambda = 0$? This is a **misunderstanding**, $\Lambda = 0$ is not a trivial solution. $\Lambda = 0$ corresponds to not using any unlabeled data. Due to few labeled data the model achieves error rate ($\downarrow$ is better) of 20.26% on CIFAR10 (4k) and 12.83% on SVHN (1k), much worse than SSL methods [29]. For more empirical evidence we initialize $\Lambda = 0$ and observe our approach to quickly deviate to a non-zero Λ (due to high $\mathcal{L}_S(\mathcal{V}, \theta)$). Theoretically, $\Lambda = 0$ is also not a trivial solution: Keep in mind that the upper-level optimizes $\mathcal{L}_S(\mathcal{V}, \theta)$, and the lower-level optimizes $\mathcal{L}_S(\mathcal{D}, \theta)$.

Q3: Minimize the unsupervised loss ... better generalization? SSL typically uses an 'unsupervised loss' to leverage unlabeled data. While the model may not generalize if the unsupervised loss is poorly designed, recent works [38, 36] empirically validate their proposed loss. Theoretical analysis of SSL has also been provided under various assumptions, *e.g.*, [6, A]. We encourage R1 to study these works which show how unsupervised losses aid generalization. Such a discussion is beyond the scope of this work. Reference: [A] P. Rigollet, "Generalization Error Bounds in Semi-supervised Classification Under the Cluster Assumption," JMLR 2007

Q4: How are the final per-sample weights distributed? Visualizations of per-example weights are color-coded in Fig. 1-2 (paper). Their evolution is shown in the appendix. Notably, unlabeled data with incorrect pseudo-labels are down-weighted near the decision boundary. We also visualize a histogram of per-example weights on CIFAR-10 in the figure to the right.

Q5: Comparison to uniform weights. We compare with baselines using uniform weights: (a) In Tab. 1, UDA and FixMatch baselines use uniform weights for all unlabeled data. (b) In Fig. 4 left (orange) we report an ablation study on UDA using a uniform weight learned by our algorithm.



Reviewer # 2

Q6: Influence function holds only when parameter weights are close to 0. This seems to be a **misunderstanding**. Please see Appendix F. We consider $\lambda_j + \epsilon$ where λ_j is the current weight for unlabeled data j and ϵ is its change. The Taylor expansion holds for any $\lambda_j > 0$ as long as $\epsilon \to 0$.

Q7: Supervised learning with Λ tuning baseline. We run FixMatch on *labeled* CIFAR-10 (4k) and SVHN (1k) data with per-example weights tuned using our method. We achieve error rates ($\downarrow$ is better) of 13.12% and 8.04% respectively, which are much worse than the results in Tab. 1. This happens as all the unlabeled data is ignored.

Q8: Per-example weights for both labeled and unlabeled data. We run FixMatch on CIFAR-10 (4k) and SVHN (1k) with per-example weights for *both labeled and unlabeled data* tuned using our method. We achieve 4.39%, 2.13% error rate respectively, which is on par with Tab. 1 results but not significantly better. This happens as labeled data only occupy a small portion and their labels are clean. Hence, tuning weights for labeled data is not always necessary.

Q9: Multiple runs for ablation study? The ablations in Fig. 4 are single run. We run experiments for Fig. 4 left (orange line) 3 times. The mean/std for 250/1k/4k split are 5.81±0.12, 5.68±0.21, and 5.32±0.13, which does not change the conclusions from the ablation study. We'll add multiple runs for all ablations.

Reviewer # 3

Q10: Motivation of per-example weights. Recent SSL works use pseudo-labels, either hard labels or soft target distributions. This is also true for "consistency based methods" (UDA, FixMatch) where each prediction is a pseudo-label for another example, as explained in L68. Label-estimates are inevitably wrong during training. We introduce per-example weights to handle this. Please see Fig. 1-2 and appendix's animations for a qualitative motivation.

Q11: Consistency-based SSL methods, or methods without label-estimate. It's a **misunderstanding** that our method is limited to pseudo-labels. Our approach can work with any unsupervised loss. For some unlabeled data, the unsupervised loss may hurt the validation performance, hence it is beneficial to introduce per-example weights.

Q12: Improvements on consistency based SSL method (UDA, FixMatch). "Consistency based methods" also involve label-estimates (soft target). Tab. 1 shows that our method improves the results in 15 out of 16 settings (except for CIFAR-10 with 4k labeled data). This is reasonable as our algorithm will become less beneficial when abundant labeled data is available. While we did not improve upon the published number of FixMatch, our number is still better than the FixMatch results reproduced using *the publicly available code*.

Q13: Improvements over single-weight baseline (Fig. 4 & Tab. 1). The black line in Fig. 4 left is the vanilla UDA baseline as in Tab. 1. The orange line in Fig. 4 left is our ablation: a single shared weight λ for all examples is learned using the program in Eq. (2). As can be seen, using per-example weights improves over a single weight.

Q14: Convergence guarantee of bi-level optimization. As discussed in L95-96, prior works have studied convergence of bi-level optimization. Details are beyond the scope.