A. Derivation of the Teacher's Update Rule

In this section, we present the detailed derivation of the teacher's update rule in Section 2.

Mathematical Notations and Conventions. Since we will work with the chain rule, we use the standard Jacobian notations.[3] Specifically, for a differentiable function $f : \mathbb{R}^m \to \mathbb{R}^n$, and for a vector $x \in \mathbb{R}^m$, we use the notation $\frac{\partial f}{\partial x} \in \mathbb{R}^{n \times m}$ to denote *the Jacobian matrix* of f, whose dimension is $n \times m$. Additionally, when we mention the Jacobian of a function f at multiple points such as x_1 and x_2, we will use the notations of $\left.\frac{\partial f}{\partial x}\right|_{x=x_1}$ and $\left.\frac{\partial f}{\partial x}\right|_{x=x_2}$.

Furthermore, by mathematical conventions, a vector $v \in \mathbb{R}^n$ is treated as a *column matrix* – that is, a matrix of size $n \times 1$. For this reason, the gradient vector of a multi-variable real-valued function is actually the transpose of of its Jacobian matrix. Finally, all multiplications in this section are standard matrix multiplications. If an operand is a vector, then the operand is treated as a column matrix.

Dimension Annotations. Understanding that these notations and conventions might cause confusions, in the derivation below, we annotate the dimensions of the computed quantities to ensure that there is no confusion caused to our readers. To this end, we respectively use $|S|$ and $|T|$ to denote the dimensions of the parameters θ_S, θ_T. That is, $\theta_S \in \mathbb{R}^{|S| \times 1}$ and $\theta_T \in \mathbb{R}^{|T| \times 1}$.

We now present the derivation. Suppose that on a batch of unlabeled examples x_u, the teacher samples the pseudo labels $\widehat{y}_u \sim T(x_u; \theta_T)$ and the student uses $(x_u, \widehat{y}_u)$ to update its parameter θ_S. In expectation, the student's new parameter is $\mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\eta_S} \text{CE}(\widehat{y}_u, S(x_u; \theta_S))\right]$. We will update the teacher's parameter to minimize the student's cross-entropy on a batch of labeled data a this expected parameter. To this end, we need to compute the Jacobian:

$$\underbrace{\frac{\partial R}{\partial \theta_T}}_{1 \times |T|} = \frac{\partial}{\partial \theta_T} \text{CE}\left(y_l, S\Big(x_l; \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\eta_S} \text{CE}(\widehat{y}_u, S(x_u; \theta_S))\right]\Big)\right) \tag{4}$$

To simplify our notation, let us define

$$\underbrace{\bar{\theta}'_S}_{|S| \times 1} = \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\eta_S} \text{CE}(\widehat{y}_u, S(x_u; \theta_S))\right] \tag{5}$$

Then, by the chain rule, we have

$$\begin{aligned}
\underbrace{\frac{\partial R}{\partial \theta_T}}_{1 \times |T|} &= \frac{\partial}{\partial \theta_T} \text{CE}\left(y_l, S\Big(x_l; \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\eta_S} \text{CE}(\widehat{y}_u, S(x_u; \theta_S))\right]\Big)\right) \\
&= \frac{\partial}{\partial \theta_T} \text{CE}\left(y_l, S\left(x_l; \bar{\theta}'_S\right)\right) \\
&= \underbrace{\left.\frac{\partial \text{CE}\left(y_l, S\left(x_l; \bar{\theta}'_S\right)\right)}{\partial \theta_S}\right|_{\theta_S = \bar{\theta}'_S}}_{1 \times |S|} \cdot \underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S| \times |T|}
\end{aligned} \tag{6}$$

The first factor in Equation 6 can be simply computed via back-propagation. We now focus on the second term. We have

$$\begin{aligned}
\underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S| \times |T|} &= \frac{\partial}{\partial \theta_T} \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \nabla_{\eta_S} \text{CE}(\widehat{y}_u, S(x_u; \theta_S))\right] \\
&= \frac{\partial}{\partial \theta_T} \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)}\left[\theta_S - \eta_S \cdot \left(\left.\frac{\partial \text{CE}\left(\widehat{y}_u, S(x_u; \theta_S)\right)}{\partial \theta_S}\right|_{\theta_S = \theta_S}\right)^\top\right]
\end{aligned} \tag{7}$$

[3] Standard: https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant

Note that in Equation 7 above, the Jacobian of CE $(\widehat{y}_u, S(x_u;\theta_S))$, which has dimension $1 \times |S|$, needs to be transposed to match the dimension of θ_S, which, as we discussed above, conventionally has dimension $|S| \times 1$.

Now, since θ_S in Equation 7 does not depend on θ_T, we can leave it out of subsequent derivations. Also, to simplify notations, let us define *the gradient*

$$\underbrace{g_S(\widehat{y}_u)}_{|S|\times|1|} = \left(\frac{\partial \text{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\theta_S} \right)^\top \tag{8}$$

Then, Equation 7 becomes

$$\underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S|\times|T|} = -\eta_S \cdot \frac{\partial}{\partial \theta_T} \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)} \Big[\underbrace{g_S(\widehat{y}_u)}_{|S|\times 1} \Big] \tag{9}$$

Since $g_S(\widehat{y}_u)$ has no dependency on on θ_T, except for via $\widehat{y}_u$, we can apply the REINFORCE equation [75] to achieve

$$\begin{aligned}
\underbrace{\frac{\partial \bar{\theta}_S^{(t+1)}}{\partial \theta_T}}_{|S|\times|T|} &= -\eta_S \cdot \frac{\partial}{\partial \theta_T} \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)} \left[g_S(\widehat{y}_u) \right] \\
&= -\eta_S \cdot \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)} \Big[\underbrace{g_S(\widehat{y}_u)}_{|S|\times 1} \cdot \underbrace{\frac{\partial \log P\left(\widehat{y}_u | x_u;\theta_T\right)}{\partial \theta_T}}_{1\times|T|} \Big] \\
&= \eta_S \cdot \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)} \Big[\underbrace{g_S(\widehat{y}_u)}_{|S|\times 1} \cdot \underbrace{\frac{\partial \text{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)}{\partial \theta_T}}_{1\times|T|} \Big]
\end{aligned} \tag{10}$$

Here, the last equality in Equation 10 is is due to the definition of the cross-entropy loss, which is the negative of the log-prob term in the previous line.

Now, we can substitute Equation 10 into Equation 6 to obtain

$$\begin{aligned}
\underbrace{\frac{\partial R}{\partial \theta_T}}_{1\times|T|} &= \underbrace{\frac{\partial \text{CE}\left(y_l, S\left(x_l;\bar{\theta}'_S\right)\right)}{\partial \theta_S}\bigg|_{\theta_S=\bar{\theta}'_S}}_{1\times|S|} \cdot \underbrace{\frac{\partial \bar{\theta}'_S}{\partial \theta_T}}_{|S|\times|T|} \\
&= \eta_S \cdot \underbrace{\frac{\partial \text{CE}\left(y_l, S\left(x_l;\bar{\theta}'_S\right)\right)}{\partial \theta_S}\bigg|_{\theta_S=\bar{\theta}'_S}}_{1\times|S|} \cdot \mathbb{E}_{\widehat{y}_u \sim T(x_u;\theta_T)} \Big[\underbrace{g_S(\widehat{y}_u)}_{|S|\times 1} \cdot \underbrace{\frac{\partial \text{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)}{\partial \theta_T}}_{1\times|T|} \Big]
\end{aligned} \tag{11}$$

Finally, we use Monte Carlo approximation for every term in Equation 11 using the sampled $\widehat{y}_u$. In particular, we approximate $\bar{\theta}'_S$ with the parameter obtained from θ_S by updating the student parameter on $(x_u, \widehat{y}_u)$, i.e., $\theta'_S = \theta_S - \eta_S \cdot \nabla_{\theta_S} \text{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)$), and approximate the expected value in the second term with the same using $\widehat{y}_u$. With these approximation, we obtain the gradient $\nabla_{\theta_T} \mathcal{L}_u(\theta_T, \theta_S)$ from Equation 1:

$$\begin{aligned}
\nabla_{\theta_T} \mathcal{L}_l &= \eta_S \cdot \underbrace{\frac{\partial \text{CE}\left(y_l, S\left(x_l;\theta'_S\right)\right)}{\partial \theta_S}}_{1\times|S|} \cdot \underbrace{\left(\frac{\partial \text{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right)}{\partial \theta_S}\bigg|_{\theta_S=\theta_S} \right)^\top}_{|S|\times 1} \cdot \underbrace{\frac{\partial \text{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)}{\partial \theta_T}}_{1\times|T|} \\
&= \underbrace{\eta_S \cdot \left(\left(\nabla_{\theta'_S} \text{CE}\left(y_l, S(x_l;\theta'_S)\right) \right)^\top \cdot \nabla_{\theta_S} \text{CE}\left(\widehat{y}_u, S(x_u;\theta_S)\right) \right)}_{\text{A scalar} := h} \cdot \nabla_{\theta_T} \text{CE}\left(\widehat{y}_u, T(x_u;\theta_T)\right)
\end{aligned} \tag{12}$$

B. Pseudo Code for Meta Pseudo Labels with UDA

In this section, we present the pseudo code for Meta Pseudo Labels where the teacher is trained with an extended objective to include the UDA loss. We emphasize that the UDA objective is applied *on the teacher*, while the student still only learns from the pseudo labeled data given by the teacher. The pseudo code can be found in Algorithm 1.

Algorithm 1 The Meta Pseudo Labels method, applied to a teacher trained with UDA [76].

Input: Labeled data x_l, y_l and unlabeled data x_u.
Initialize $\theta_T^{(0)}$ and $\theta_S^{(0)}$
for $t = 0$ **to** $N - 1$ **do**
  Sample an unlabeled example x_u and a labeled example x_l, y_l
  Sample a pseudo label $\widehat{y}_u \sim P(\cdot|x_u; \theta_T)$
  Update the student using the pseudo label $\widehat{y}_u$:

$$\theta_S^{(t+1)} = \theta_S^{(t)} - \eta_S \ \nabla_{\theta_S} \text{CE}(\widehat{y}_u, S(x_u; \theta_S))|_{\theta_S = \theta_S^{(t)}}$$

  Compute the teacher's feedback coefficient as in Equation 12:

$$h = \eta_S \cdot \left(\left(\nabla_{\theta'_S} \text{CE} \left(y_l, S(x_l; \theta_S^{(t+1)}) \right) \right)^\top \cdot \nabla_{\theta_S} \text{CE} \left(\widehat{y}_u, S(x_u; \theta_S^{(t)}) \right) \right)$$

  Compute the teacher's gradient from the student's feedback:

$$g_T^{(t)} = h \cdot \nabla_{\theta_T} \text{CE}(\widehat{y}_u, T(x_u; \theta_T))|_{\theta_T = \theta_T^{(t)}}$$

  Compute the teacher's gradient on labeled data:

$$g_{T,\text{supervised}}^{(t)} = \nabla_{\theta_T} \ \text{CE}(y_l, T(x_l; \theta_T))|_{\theta_T = \theta_T^{(t)}}$$

  Compute the teacher's gradient on the UDA loss with unlabeled data:

$$g_{T,\text{UDA}}^{(t)} = \nabla_{\theta_T} \ \text{CE} \Big(\text{StopGradient}(T(x_l); \theta_T), T(\text{RandAugment}(x_l); \theta_T) \Big) \Big|_{\theta_T = \theta_T^{(t)}}$$

  Update the teacher:

$$\theta_T^{(t+1)} = \theta_T^{(t)} - \eta_T \cdot \left(g_T^{(t)} + g_{T,\text{supervised}}^{(t)} + g_{T,\text{UDA}}^{(t)} \right)$$

end
return $\theta_S^{(N)}$ ▷ *Only the student model is returned for predictions and evaluations*

C. Experimental Details

In this section, we provide the training details for our experiments in Section 3 and Section 4.

C.1. Dataset Splits

We describe how the datasets CIFAR-10-4K, SVHN-1K, and ImageNet-10% in Section 3.2 are constructed. For CIFAR-10, we download the five training data batch files from CIFAR-10's official website.[4] Then, we load all the images into a list of 50,000 images, keeping the order as downloaded. The fisrt 5,000 images are typically reserved for validation, so we remove them. The next 4,000 images are used as labeled data. For SVHN, we download the data from the `mat` files on SVHN's official site[5], and follow the same procedure as with CIFAR-10. We note that this selection process leads to a slight imbalance in the class distribution for both CIFAR-10-4K and SVHN-1K, but the settings are the same for all of our experiments. For ImageNet, we follow the procedure in Inception's GitHub[6]. This results in 1,024 training `TFRecord` shards of approximately the same size. The order of the images in these shards are deterministic. For ImageNet-10%, we use the first 102 shards;

[4] CIFAR-10's official website: www.cs.toronto.edu/~kriz/cifar.html.
[5] SVHN's official website: ufldl.stanford.edu/housenumbers/.
[6] Inception's GitHub, which also has the code to create ImageNet's training shards in `TFRecord`: github.com/tensorflow/models/blob/master/research/inception/inception/data/download_and_preprocess_imagenet.sh.

for ImageNet-20%, we use the first 204 shards; and so on. The last 20 shards, corresponding to roughly 25,000 images, are reserved for hyper-parameters tuning (used in Section 3.3 and Section 4).

C.2. Modifications of RandAugment [13]

We modify a few data augmentation strategies as introduced by RandAugment [13]. Our modifications mostly target the SVHN dataset. In particular, we remove all rotations from the set of augmentation operations since rotation is a wrong invariance for digits such as 6 and 9. We also remove horizontal translations because they cause another wrong invariance for digits 3 and 8, e.g., when 8 is pushed half-outside the image and the remaining part looks like a 3. Table 5 presents the transformations that we keep for our datasets.

CIFAR-10 and **ImageNet**	**SVHN**
AutoContrast	AutoContrast
Brightness	Brightness
Color	Color
Contrast	Contrast
Equalize	Equalize
Invert	Invert
Sharpness	Sharpness
Posterize	Posterize
Sample Pairing	Solarize
Solarize	ShearX
Rotate	ShearY
ShearX	TranslateY
ShearY	
TranslateX	
TranslateY	

Table 5: Transformations that RandAugment uniformly samples for our datasets. We refer our readers to [12] for the detailed descriptions of these transformations.

C.3. Additional Implementation Details

To improve the stability of Meta Pseudo Labels, we use the following details in the Meta Pseudo Labels process.

Use cosine distance instead of dot product in Equation 12. The dot product h in Equation 12 has a large value range, especially at the beginning of the Meta Pseudo Labels process. Thus, in order to stabilize training, we compute h using the gradients' cosine distance. This modification requires very little modification in our code.

We give two justifications why the use of cosine distance makes sense *mathematically*. First, h in Equation 12 is on a scalar which is multiplied with the teacher's gradient with respect to θ_T. Changing dot product into cosine distance does not change the sign of h, and thus preserving the actions to increase or to decrease the probabilities of the sampled pseudo labels. Second, cosine distance's value range is much smaller than that of dot product, making the Meta Pseudo Labels updates more numerically stable. Specifically, the value range of cosine distance is $[-1, 1]$, while the value range of dot products, as observed in our experiments, is about $[-5 \times 10^4, 5 \times 10^4]$. This range also depends on the weight decay hyper-parameter.

Additionally, the dot product h, as shown in Equation 12 and as derived in Section A, results from the application of the chain rule in a so-called bi-level optimization procedure. Bi-level optimization has been applied in some past work, such as Hyper Gradient Descent [3], which also replaces dot product with cosine distance to improve the numerical stability.

Use a baseline for h in Equation 12. To further reduce the variance of h, we maintain a moving average b of h and subtract b from h every time we compute $g_T^{(t)}$ as in Equation 12. This practice is also widely applied in Reinforcement Learning literature.

While using cosine distance is very crucial to maintain the numerical stability of Meta Pseudo Labels, using the moving average baseline only slightly improves Meta Pseudo Labels's performance. We suspect that not using the moving average baseline is also fine, especially when Meta Pseudo Labels can train for many steps without overfitting.

C.4. Hyper-parameters

Optimizers. In all our experiments, the WideResNet-28-2 for CIFAR-10-4K and SVHN-1K and the ResNet-50 for ImageNet-10% and full ImageNet are updated with Nesterov Momentum with default the momentum coefficient of 0.9. The networks' learning rate follow the cosine decay [41]. Meanwhile, the EfficientNet-L2 and EfficientNet-B6-Wide for ImageNet+JFT are trained with RMSProp [66] and with an exponential decay learning rate. These are the default optimizers and learning rate schedules used for the architectures in their corresponding papers. We have only one substantial change of optimizer: when we finetune EfficientNet-L2 and EfficientNet-B6-Wide on the labeled data from ImageNet (see Section 4), we use the LARS optimizer [82] with their default parameters, i.e., momentum 0.9 and learning rate 0.001, training for 20,000 steps with a batch size of 4,096. We finetune using this optimizer instead of SGD in Noisy Student [77] because unlike Noisy Student, the student model in Meta Pseudo Labels never trains directly on any labeled example, and hence can benefit from a more "aggressive" finetuning process with stronger optimiziers.

Numerical Hyper-parameters. To tune hyper-parameters, we follow [48] and allow each method to have 128 trials of hyper-parameters. When we tune, we let each model train for up to 50,000 steps. The optimal hyper-parameters are then used to run experiments that last for much more steps, as we report below. In our experiments with Meta Pseudo Labels, training for more steps typically leads to stronger results. We stop at 1 million steps for CIFAR-10-4K and SVHN-1K, and at 0.5 million steps for ImageNet because these are the standards from past papers.

We report the hyper-parameters for our baselines and for Meta Pseudo Labels in Section 3 in Tables 6, 7, 8. We note that our settings for UDA is different from originally reported by the original UDA paper [76]. In their work, UDA [76] use a much larger batch size for their UDA objective. In our implementation of UDA, we keep these batch sizes the same. This leads to a much easier implementation of data parallelism in our framework, TensorFlow [1] running on TPU big pods. To compensate for the difference, we train all UDA baselines for much longer than the UDA paper [76]. During the training process, we also mask out the supervised examples with high confidence. Effectively, our UDA model receives roughly the same amount of training with labeled examples and unlabeled examples as the models in [76]. We have also verified that on ImageNet-10% with the augmentation policy from AutoAugment [12], our UDA implementation achives 68.77% top-1 accuracy, which is similar to 68.66% that the UDA paper [76] reported.

Hyper-parameter	CIFAR-10	SVHN	ImageNet
Weight decay	0.0005	0.001	0.0002
Label smoothing	0	0	0.1
Batch normalization decay	0.99	0.99	0.99
Learning rate	0.4	0.05	1.28
Number of training steps	50,000	50,000	40,000
Number of warm up steps	2500	0	2000
Batch size	1024	128	2048
Dropout rate	0.4	0.5	0.2
Pseudo label threshold	0.95	0.975	0.7

Table 6: Hyper-parameters for supervised learning and Pseudo Labels.

Hyper-parameter	CIFAR-10	SVHN	ImageNet
Weight decay	0.0005	0.0005	0.0002
Label smoothing	0	0	0.1
Batch normalization decay	0.99	0.99	0.99
Learning rate	0.3	0.4	1.28
Number of training steps	1,000,000	1,000,000	500,000
Number of warm up steps	5,000	5,000	5,000
Batch size	128	128	2048
Dropout rate	0.5	0.6	0.25
UDA factor	2.5	1	20
UDA temperature	0.7	0.8	0.7

Table 7: Hyper-parameters for UDA. Unlike originally done by the UDA paper [76], we do not use a larger batch size for the UDA objective. Instead, we use the same batch size for both the labeled objective and the unlabeled objective. This is to avoid instances where some particularly small batch sizes for the labeled objective cannot be split on our computational hardware.

	Hyper-parameter	CIFAR-10	SVHN	ImageNet
Common	Weight decay	0.0005	0.0005	0.0002
	Label smoothing	0.1	0.1	0.1
	Batch normalization decay	0.99	0.99	0.99
	Number of training steps	1,000,000	1,000,000	500,000
	Number of warm up steps	2,000	2,000	1,000
Student	Learning rate	0.3	0.15	0.8
	Batch size	128	128	2048
	Dropout rate	0.35	0.45	0.1
Teacher	Learning rate	0.125	0.05	0.5
	Batch size	128	128	2048
	Dropout rate	0.5	0.65	0.1
	UDA factor	1.0	2.5	16.0
	UDA temperature	0.8	1.25	0.75

Table 8: Hyper-parameters for Meta Pseudo Labels.

D. More Detailed Analysis of Meta Pseudo Label's Behaviors

We have seen in Section 3 and Section 4 that Meta Pseudo Labels leads to strong performances on multiple image classification benchmarks. In this section, we provide further analysis of Meta Pseudo Labels and related baselines on more restricted and more controlled environments to provide better insights about Meta Pseudo Labels' behaviors.

D.1. Visualizing the Contributions of Meta Pseudo Labels

To understand the contributions of Meta Pseudo Labels (MPL), in Figure 3, we visualize the relative gains of various methods on ImageNet-10% (Section 3.2). From the figure, we have two observations. First, for a purely supervised teacher, Meta Pseudo Labels outperforms RandAugment. We suspect this is because Meta Pseudo Labels is more effective form of regularization for the student. This is very crucial for ImageNet-10%, where we only have about 128 images per class for each of the 1,000 classes. Second, UDA improves over Supervised+MPL+Finetune by 6.05% in top-1 accuracy. This is in the same ballpark with the gain that UDA+MPL delivers above UDA, which is 5.25%. As UDA's accuracy is already high, such improvement is very significant. Finally, finetuning only slightly improves over UDA+MPL. This extra performance boost is a unique advantage of Meta Pseudo Labels, since the student never directly learns from labeled data.

D.2. Meta Pseudo Labels Is An Effective Regularization Strategy

The rest of this paper uses Meta Pseudo Labels as a semi-supervised learning method. In this section, we show that Meta Pseudo Labels can behave like an effective regularization method for supervised learning. This behavior can be achieved by

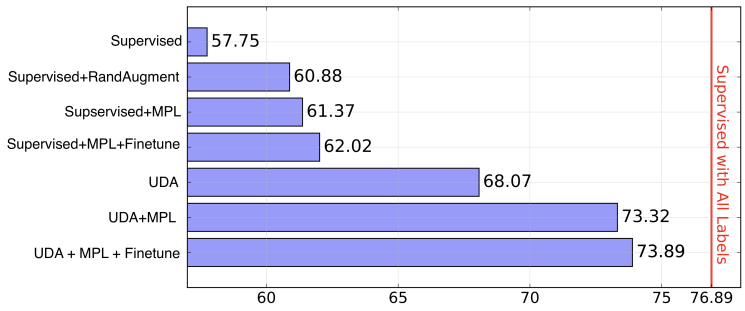

Figure 3: Breakdown of the gains of different components in Meta Pseudo Labels (MPL). The gain of Meta Pseudo Labels over UDA, albeit smaller than the gain of UDA over RandAugment, is significant as UDA is already very strong.

making labeled data the same with unlabeled data in Figure 1. In this case, Meta Pseudo Labels can be seen as an adaptive form of Label Smoothing: the teacher generates soft labels on labeled data for the student, just like the way Label Smoothing smooths the hard labels to regularize the model. The main difference is that the policy in Label Smoothing is fixed, whereas the policy of the teacher in Meta Pseudo Labels is adaptive to enhance the student's performance.

To confirm the effect of Meta Pseudo Labels, we compare the method to Supervised Learning and Label Smoothing on CIFAR-10-4K and SVHN-1K. All models and settings are the same as in Section 3.2, except that we do not use RandAugment and we restrict the unlabeled data to the same set of labeled data. We choose CIFAR-10-4K and SVHN-1K for this experiment because Label Smoothing is typically already used in ImageNet models. The results are shown in Table 9. As can be seen from the table, Meta Pseudo Labels achieves 83.71% on CIFAR-10-4K and 91.89% on SVHN-1K. Both of these are significantly better than the accuracy obtained by supervised learning with and without Label Smoothing. This shows the importance of feedback in Meta Pseudo Labels.

	CIFAR-10-4K	**SVHN-1K**
Supervised	82.14 ± 0.25	88.17 ± 0.47
Label Smoothing	82.21 ± 0.18	89.39 ± 0.25
Meta Pseudo Labels	$\mathbf{83.71 \pm 0.21}$	$\mathbf{91.89 \pm 0.14}$

Table 9: Meta Pseudo Labels can be used as a regularization method for supervised learning.

D.3. Meta Pseudo Labels Is a Mechanism to Addresses the Confirmation Bias of Pseudo Labels

In this section, we show empirical evidence that Meta Pseudo Labels helps to address the teacher's confirmation bias [2] in Pseudo Labels. To this end, we analyze the *training* accuracy of the teacher and the student in Meta Pseudo Labels from our experiments for CIFAR-10-4K and ImageNet-10% in Section 3.2. In Figure 4, we plot the accuracy percentage at each training batch throughout the training process of a teacher and a student in Meta Pseudo Labels. We also plot the same data for a supervised model. From the figure, we have two observations:

- On CIFAR-10-4K (Figure 4-Left), the student's training accuracy in Meta Pseudo Labels is much lower that of the same network in Supervised Learning. As CIFAR-10-4K has very few labeled data, if the teacher converges quickly like in Supervised Learning, it will not generalize to the unlabeled data and hence will teach the student in inaccurate pseudo labels. In contrast, Figure 4-Left shows that both the teacher and student in Meta Pseudo Labels converge much slower. To see this, note that in Meta Pseudo Labels, the student's *training* accuracy is measured by how much it agrees with the teacher's pseudo labels. Therefore, the student in Meta Pseudo Labels having a lower training accuracy means that the student often disagrees with the pseudo labels that the teacher samples. This disagreement forces the teacher to constantly updates its weights to generate better pseudo labels, and makes it hard for the student to converge as the student has to learn from the teacher's changing pseudo labels. This behavior prevents both the teacher and the student from the premature convergence that causes the confirmation bias in Supervised Learning and Pseudo Labels.

- On ImageNet-10% (Figure 4-Right), the student also disagrees with the teacher's pseudo labels, as shown in the student's low training accuracy. Additionally, we observe that the teacher's training accuracy surges up faster than the supervised

model's accuracy. We suspect that this is beneficial for the student learning, since ImageNet has 1,000 classes so in order to effectively teach the student to do well on the labeled dataset, the teacher has to become more accurate. Therefore, the feedback from the student is beneficial for the teacher's learn as well. This trend of high training accuracy only changes at the end of the training procedure, where the training accuracy of Supervised Learning surpasses those of the teacher and the student in Meta Pseudo Labels. From this last sign, we suspect that the supervised model has overfitted to the small set of labeled training examples in ImageNet-10%, which will causes the confirmation bias if this supervised model is used to generate pseudo labels for another student model to learn from.

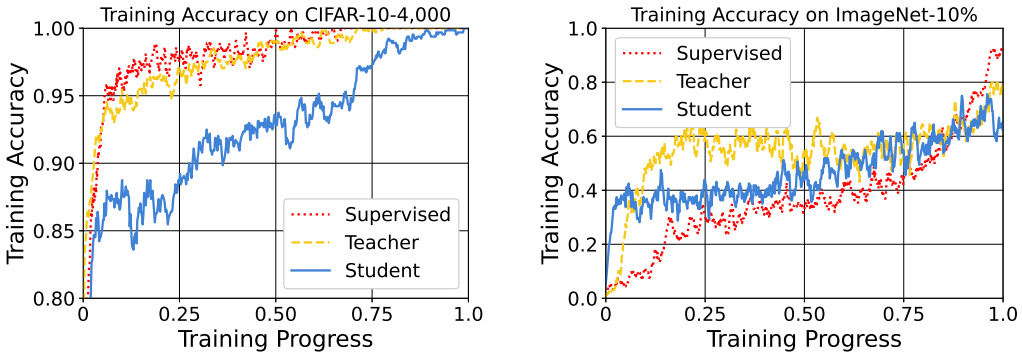

Figure 4: Training accuracy of Meta Pseudo Labels and of supervised learning on CIFAR-10-4,000 and ImageNet-10%. Both the teacher and the student in Meta Pseudo Labels have lower training accuracy, effectively avoiding overfitting.

D.4. Meta Pseudo Labels with Different Training Techniques for the Teacher

In Sections 3 and Section 4, we have presented Meta Pseudo Labels results where the teacher is trained with UDA. In Table 10, we further show that on CIFAR-10-4K, Meta Pseudo Labels improves over different teachers trained with different techniques, including Pseudo Labels [36], Mixup [85], and RandAugment. These results indicate that Meta Pseudo Labels is effective with all techniques. Additionally, the results suggest that better training techniques for the teacher tend to result in better students.

Teacher	Pseudo-Labels	Mixup [85]	RandAugment
-Meta Pseudo Labels	83.79 ± 0.11	84.20 ± 0.15	85.53 ± 0.25
+Meta Pseudo Labels	$\mathbf{84.11 \pm 0.07}$	$\mathbf{84.81 \pm 0.19}$	$\mathbf{87.55 \pm 0.14}$

Table 10: Meta Pseudo Labels's accuracy for WideResNet-28-2 on CIFAR-10-4,000, where the teacher is trained with different techniques. All numbers are mean $\pm$ std over 10 runs.

D.5. Meta Pseudo Labels with Different Amounts of Labeled Data

We study how much Meta Pseudo Labels improves as more labeled data becomes available. To this end, we experiment with 10%, 20%, 40%, 80%, and 100% of the labeled examples in ImageNet. We compare Meta Pseudo Labels with supervised learning and RandAugment. We plot the results in Figure 5. From the figure, it can be seen that Meta Pseudo Labels delivers substantial gains with less data, but plateaus as more labeled data becomes available. This result suggests that Meta Pseudo Labels is more effective for low-resource image classification problems.

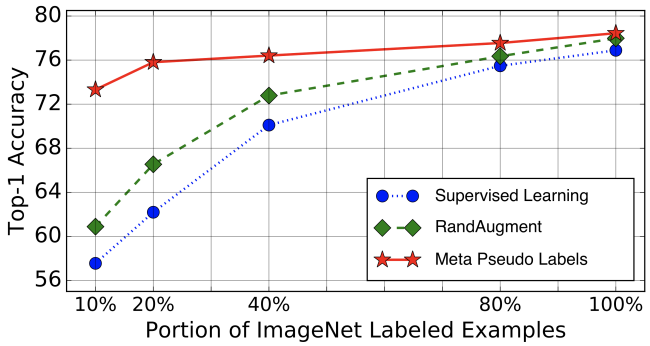

Figure 5: Performance of Supervised Learning, RandAugment, and Meta Pseudo Labels at different amounts of labeled examples.

E. Results with An Economical Version of Meta Pseudo Labels

Meta Pseudo Labels requires storing both the teacher model and the student model in memory. For model architectures with a large memory footprint, such as EfficientNet-L2 and EfficientNet-B6-Wide in our experiments, this memory footprint exceeds 16G of available memory in our accelerators. While we have implemented a hybrid data-model parallelism in Section 4 which allows us to run Meta Pseudo Labels with large model architectures, the tradeoff is a slow and expensive training procedure. To allow a more efficient training of large models with Meta Pseudo Labels, we design a more economical alternative to instantiate the teacher, termed Reduced Meta Pseudo Labels.

In Reduced Meta Pseudo Labels, we first train a large teacher model T to convergence. Next, we use T to pre-compute all target distributions for the student's training data. Importantly, until this step, the student model has not been loaded into memory, effectively avoiding the large memory footprint of Meta Pseudo Labels. Then, we parameterize a reduced teacher T' as a small and efficient network, such as a multi-layered perceptron (MLP), to be trained the along with student. This reduced teacher T' takes as input the distribution predicted by the large teacher T and outputs a calibrated distribution for the student to learn. Intuitively, Reduced Meta Pseudo Labels works reasonably well because the large teacher T is reasonably accurate, and hence many actions of the reduced teacher T' would be close to an identity map, which can be handled by an MLP. Meanwhile, Reduced Meta Pseudo Labels retains the benefit of Meta Pseudo Labels, as the teacher T' can still adapt to the learning state of the student θ_T.

To evaluate whether Meta Pseudo Labels can scale to problems with a large number of labeled examples, we now turn to full labeled sets of CIFAR-10, SVHN and ImageNet. We use out-of-domain unlabeled data for CIFAR-10 and ImageNet. We experiment with Reduced Meta Pseudo Labels whose memory footprint allows our large-scale experiments. We show that the benefit of Meta Pseudo Labels, *i.e.,* having a teacher that adapts to the student's learning state throughout the student's learning, stil extends to large datasets with more advanced architectures and out-of-domain unlabeled data.

Model Architectures. For our student model, we use EfficinetNet-B0 for CIFAR-10 and SVHN, and use EfficientNet-B7 for ImageNet. Meanwhile, our teacher model is a small 5-layer perceptron, with ReLU activation, and with a hidden size of 128 units for CIFAR-10 and of 512 units for ImageNet.

Labeled Data. Per standard practices, we reserve 4,000 examples of CIFAR-10, 7,300 examples from SVHN, and 40 data shards of ImageNet for hyper-parameter tuning. This leaves about 45,000 labeled examples for CIFAR-10, 65,000 labeled examples for SVHN, and 1.23 million labeled examples for ImageNet. As in Section 3.2, these labeled data serve as both the validation data for the student and the pre-training data for the teacher.

Unlabeled Data. For CIFAR-10, our unlabeled data comes from the TinyImages dataset which has 80 million images [67]. For SVHN, we use the extra images that come with the standard training set of SVHN which has about 530,000 images. For ImageNet, our unlabeled data comes from the YFCC-100M dataset which has 100 million images [65]. To collect unlabeled data relevant to the tasks at hand, we use the pre-trained teacher to assign class distributions to images in TinyImages and YFCC-100M, and then keep K images with highest probabilities for each class. The values of K are 50,000 for CIFAR-10, 35,000 for SVHN, and 12,800 for ImageNet.

Baselines. We compare Reduced Meta Pseudo Labels to NoisyStudent [77], because it can be directly compared to Reduced Meta Pseudo Labels. In fact, the *only* difference between NoisyStudent and Reduced Meta Pseudo Labels is that Reduced Meta Pseudo Labels has a teacher that adapts to the student's learning state.

Methods	**CIFAR-10**	**SVHN**	**ImageNet**
Supervised	97.18 ± 0.08	98.17 ± 0.03	$84.49/97.18$
NoisyStudent	98.22 ± 0.05	$\mathbf{98.71 \pm 0.11}$	$85.81/97.53$
Reduced Meta Pseudo Labels	$\mathbf{98.56 \pm 0.07}$	$\mathbf{98.78 \pm 0.07}$	$\mathbf{86.87/98.11}$

Table 11: Image classification accuracy of EfficientNet-B0 on CIFAR-10 and SVHN, and EfficientNet-B7 on ImageNet. Higher is better. CIFAR-10 results are mean $\pm$ std over 5 runs, and ImageNet results are top-1/top-5 accuracy of a single run. All numbers are produced in our codebase and are controlled experiments.

Results. As presented in Table 11, Reduced Meta Pseudo Labels outperforms NoisyStudent on both CIFAR-10 and ImageNet, and is on-par with NoisyStudent on SVHN. In particular, on ImageNet, Meta Pseudo Labels with EfficientNet-B7 achieves a top-1 accuracy of 86.87%, which is 1.06% better than the strong baseline NoisyStudent. On CIFAR-10, Meta Pseudo Labels leads to an improvement of 0.34% in accuracy on NoisyStudent, marking a 19% error reduction.

For SVHN, we suspect there are two reasons of why the gain of Reduced Meta Pseudo Labels is not significant. First, NoisyStudent already achieves a very high accuracy. Second, the unlabeled images are high-quality, which we know by manual inspection. Meanwhile, for many ImageNet categories, there are not sufficient images from YFCC100M, so we end up with low-quality or out-of-domain images. On such noisy data, Reduced Meta Pseudo Labels's adaptive adjustment becomes more crucial for the student's performance, leading to more significant gain.