Supplementary Information: Mitigating Confirmation Bias in Semi-supervised Learning via Efficient Bayesian Model Averaging

Reviewed on OpenReview: `https://openreview.net/forum?id=PRrKOaDQtQ&`

A Theoretical Connections to Generalization Bounds

In the training of BaM-, we adopt a variational approach and maximize the evidence lower-bound (ELBO),

$$\text{ELBO} = \mathbb{E}_q \log p(Y|X;\theta) - KL(q(\theta|\phi)\|p(\theta))$$

with notation consistent with those used in Section 4.1 of the main text. We motivate this approach by using Corollary 1 below, derived from the PAC-Bayes framework (for the complete proof, see Appendix A.1 below). The statement shows the generalization error bounds on the variational posterior in the SSL setting and suggests that this generalization error is upper bounded by the negative ELBO. This motivates our approach, i.e. by maximizing the ELBO we improve generalization by minimizing the upper bound to the generalization error. The second term on the right side of the inequality characterises the SSL setting, and vanishes in the supervised setting.

Corollary 1 *Let $\mathcal{D}$ be a data distribution where i.i.d. training samples are sampled, of which N_l are labeled, $(x,y) \sim \mathcal{D}^{N_l}$ and N_u are unlabeled, $(x,\hat{y}) \sim \mathcal{D}^{N_u}$ where $\hat{y}_i$ (y_i) denotes the model-assigned pseudo-labels (true labels) for input x_i. For the negative log likelihood loss function ℓ, assuming that ℓ is sub-Gaussian (see Germain et al. (2016)) with variance factor s^2, then with probability at least $1-\delta$, the generalization error (denoted as $\mathcal{L}^{\ell}_{\mathcal{D}}$) of the variational posterior $q(\theta|\phi)$ is given by,*

$$\mathbb{E}_{q(\theta|\phi)}\mathcal{L}^{\ell}_{\mathcal{D}}(q) \leq \frac{1}{N}[-\text{ELBO}] - \mathbb{E}_{q(\theta|\phi)}\left[\frac{1}{N_u}\sum_{i=1}^{N_u}\log\frac{p(\hat{y}_i|x_i;\theta)}{p(y_i|x_i;\theta)}\right] + \frac{1}{N}\log\frac{1}{\delta} + \frac{s^2}{2} \tag{1}$$

A.1 Proof of Corollary 1

PAC-Bayesian theory aims to provide bounds on the generalization risk under the assumption that samples are i.i.d.. While PAC-Bayesian bounds typically apply to bounded losses, Germain et al. (2016) extends them to unbounded losses (which is necessary in our work since it uses the unbounded log-loss). Their theoretical result is reproduced below in Theorem 1, under assumptions for sub-Gaussian losses. These bounds commonly assume the supervised setting; in this proof, our goal is to extend them to the semi-supervised learning setting where in addition to labels from an empirical set, our loss is also mediated by pseudo-labels assigned by the model.

Let $(X,Y) \sim \mathcal{D}^N$ denote N i.i.d. training samples from the data distribution $\mathcal{D}$, where $(X,Y) = \left\{\{(x_i,y_i)\}_{i=1}^{N_l}, \{(x_i,\hat{y}_i)\}_{i=1}^{N_u}\right\}$ represents the set of N_l labeled input-output pairs and N_u pairs of unlabeled input and their model-assigned pseudo-labels $\hat{y}$. Here, $N_l + N_u = N$. Let the loss function be $\ell(f,x,y) \to \mathbb{R}$, the generalization risk on distribution $\mathcal{D}$, i.e. $\mathcal{L}^{\ell}_{\mathcal{D}}(f)$, and the empirical risk on the training set, i.e. $\mathcal{L}^{\ell}_{X,Y}(f)$, is given by:

$$\mathcal{L}^{\ell}_{\mathcal{D}}(f) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\ell(f,x,y); \quad \mathcal{L}^{\ell}_{X,Y}(f) = \frac{1}{N_l}\sum_{i=1}^{N_l}\ell(f,x_i,y_i) + \frac{1}{N_u}\sum_{i=1}^{N_u}\ell(f,x_i,\hat{y}_i)$$

On assumptions that the loss is sub-Gaussian (see Boucheron et al. (2013) section 2.3), i.e. a loss function ℓ is sub-Gaussian with variance s^2 under a prior π and $\mathcal{D}$ if it can be described by a sub-Gaussian random variable $v = \mathcal{L}^{\ell}_{\mathcal{D}}(f) - \ell(f, x, y)$, the generalization error bounds are given by Theorem 1 (Germain et al., 2016) below.

Theorem 1 *(Corollary 4 from Germain et al. (2016)) Let π be the prior distribution and $\hat{\rho}$ be the posterior. If the loss is sub-Gaussian with variance factor s^2, with probability at least $1-\delta$ over the choice of $(X, Y) \sim \mathcal{D}^N$,*

$$\mathop{\mathbb{E}}_{f\sim\hat{\rho}} \mathcal{L}^{\ell}_{\mathcal{D}}(f) \leq \mathop{\mathbb{E}}_{f\sim\hat{\rho}} \mathcal{L}^{\ell}_{X,Y}(f) + \frac{1}{N}\left(KL(\hat{\rho}\|\pi) + \log(1/\delta)\right) + \frac{1}{2}s^2$$

Beginning from Theorem 1, the generalization error of the variational posterior $q(\theta|\phi)$ under the negative log likelihood loss and prior $p(\theta)$ in our SSL setting can be found by replacing $\hat{\rho}$ with $q(\theta|\phi)$ and ℓ with $\ell^{nll}(f, x_i, y_i) := -\log p(y_i|x_i;\theta)$;

$$\begin{aligned}
\mathop{\mathbb{E}}_{f\sim q} \mathcal{L}^{\ell}_{\mathcal{D}}(f) &\leq \mathop{\mathbb{E}}_{f\sim q} \mathcal{L}^{\ell}_{X,Y}(f) + \frac{1}{N}\left(KL(q(\theta|\phi)\|p(\theta)) + \log(1/\delta)\right) + \frac{1}{2}s^2 \\
&= \mathbb{E}_q\left[\frac{1}{N_u}\sum_{i=1}^{N_u} -\log p(\hat{y}_i|x_i;\theta) + \frac{1}{N_l}\sum_{i=1}^{N_l} -\log p(y_i|x_i;\theta)\right] \\
&\qquad + \frac{1}{N}\left(KL(q(\theta|\phi)\|p(\theta)) + \log(1/\delta)\right) + \frac{1}{2}s^2 \\
&= \mathbb{E}_q\left[\frac{1}{N_u}\sum_{i=1}^{N_u} -\log \frac{p(\hat{y}_i|x_i;\theta)}{p(y_i|x_i;\theta)} + \frac{1}{N_u}\sum_{i=1}^{N_u} -\log p(y_i|x_i;\theta) + \frac{1}{N_l}\sum_{i=1}^{N_l} -\log p(y_i|x_i;\theta)\right] \\
&\qquad + \frac{1}{N}\left(KL(q(\theta|\phi)\|p(\theta)) + \log(1/\delta)\right) + \frac{1}{2}s^2 \\
&= \mathbb{E}_q\left[\frac{1}{N_u}\sum_{i=1}^{N_u} -\log \frac{p(\hat{y}_i|x_i;\theta)}{p(y_i|x_i;\theta)} + \frac{1}{N}\sum_{i=1}^{N} -\log p(y_i|x_i;\theta)\right] + \frac{1}{N}\left(KL(q(\theta|\phi)\|p(\theta)) + \log(1/\delta)\right) + \frac{1}{2}s^2 \\
&= \mathbb{E}_q\left[-\frac{1}{N_u}\sum_{i=1}^{N_u} \log \frac{p(\hat{y}_i|x_i;\theta)}{p(y_i|x_i;\theta)}\right] + \mathbb{E}_q\left[-\frac{1}{N}\log p(Y|X;\theta)\right] + \frac{1}{N}\left(KL(q(\theta|\phi)\|p(\theta)) + \log(1/\delta)\right) + \frac{1}{2}s^2 \\
\mathop{\mathbb{E}}_{f\sim q} \mathcal{L}^{\ell}_{\mathcal{D}}(f) &\leq \mathbb{E}_q\left[-\frac{1}{N_u}\sum_{i=1}^{N_u} \log \frac{p(\hat{y}_i|x_i;\theta)}{p(y_i|x_i;\theta)}\right] + \frac{1}{N}\left[-\text{ELBO}\right] + \frac{1}{N}\log(1/\delta) + \frac{1}{2}s^2
\end{aligned}$$

where we define $\text{ELBO} = \mathbb{E}_{q(\theta|\phi)}\left[\log p(Y|X;\theta)\right] - \text{KL}(q(\theta\,|\,\phi)\|P(\theta))$. In the later sections, Appendix C.1, we will see that this is the term we are maximizing in our loss objective. By maximizing the ELBO, i.e. minimizing the negative ELBO, we are minimizing the upper bound to the generalization error of our variational posterior. The first term in the last line adds a divergence measure between the pseudo-label prediction distribution and the ground truth distribution — in the fully supervised setting $\hat{y}_i \to y_i$ and this term vanishes.

B Exploring SSL methods without a selection metric

In the main text, we have looked at two popular SSL methods that are based on a selection metric. More recently, there is also a family of SSL methods based upon representation learning (Assran et al., 2021; Chen et al., 2020b; Caron et al., 2021), where the model is first trained to produce useful representations and subsequently fine-tuned on limited labeled data. This give rise to some interesting questions: Is confirmation bias still an issue in SSL methods without a selection metric? And if so, can BaM- or other approximate Bayesian techniques be used to mitigate confirmation bias? We explore these questions in this section, using PAWS (Assran et al., 2021) as a canonical example for the family of SSL methods based on representation learning approach. Details about the training loss used in PAWS are explained in Appendix B.1. The main difference PAWS has in contrast to the methods studied above is in its prediction generation technique —

instead of a classification head, PAWS uses a non-parametric nearest neighbour classification method over its representations.

B.1 SSL methods based on representation learning

PAWS. More recently, there has been a newer family of SSL methods based upon visual representation learning Chen et al. (2020a;b); Assran et al. (2021). We use PAWS Assran et al. (2021) as a canonical example of a SSL method from this family. A key difference of PAWS as compared to other selection-metric based SSL methods is the lack of a parametric classfier layer. Instead, predictions are derived from a non-parametric soft-nearest-neigbour classifier based on representations. Let $z_1 = f(\alpha_1(x))$ and $z_2 = f(\alpha_2(x))$ be the representations for the two views from the backbone encoder, their pseudo-labels (q_1, q_2) and the unlabeled loss are given by:

$$q_i = \pi_d(z_i, \{z_s\}) = \sum_{s=1}^{B} \frac{d(z_i, z_s) \cdot y_s}{\sum_{s'=1}^{B} d(z_i, z_{s'})}; \quad L_{\mathrm{u}} = \frac{1}{2\mu B} \sum_{u=1}^{\mu B} H(\rho_t(q_{1,u}), q_{2,u}) + H(\rho_t(q_{2,u}), q_{1,u}) \tag{2}$$

where $\{z_s\}_{s=1}^{B}$ is the set of representations of the labeled examples, $d(a, b) = \exp(a \cdot b/(\|a\|\|b\|\tau_p))$ is a similarity metric with temperature hyperparameter τ_p and all other symbols have the same meanings defined before in Section 3 of the main text. The combined training loss for PAWS is $L_u + L_{\text{me-max}}$ where the latter is a regularization term $L_{\text{me-max}} = H(\bar{q})$ that seeks to maximize the entropy of the average of predictions $\bar{q} := (1/(2\mu B)) \sum_{u=1}^{\mu B} (\rho_t(q_{1,u}) + \rho_t(q_{2,u}))$.

B.2 Exploring other approximate Bayesian techniques

Since BaM- replaces the classification head and PAWS does not have one, we incorporate BaM- into PAWS by using a Bayesian last layer in the encoder and "Bayesian marginalizing" over the representations (also see Appendix B.4.1 for detailed formulation). Nonetheless, apart from Bayesian marginalization, BaM- is originally desirable towards improving uncertainty estimates in the selection metric. Since PAWS does not use a selection metric, instead of Bayesian marginalizing or aggregating over just one layer, one could seek to aggregate over the entire network. This can be most commonly acheived by approximate bayesian techniques such as model ensembling approaches Lakshminarayanan et al. (2016), which has been highly successful at improving uncertainty estimation. However, training multiple networks would add immense computational overhead. To perform this aggregation computational efficiently, we instead explored weight averaging approaches, such as Stochastic Weight Averaging (Izmailov et al., 2018) (SWA) and Exponential Moving Averaging (EMA) (Tarvainen & Valpola, 2017; He et al., 2020; Grill et al., 2020) and studied their role in mitigating confirmation bias during pseudo-labeling. Weight averaging however differs from traditional bayesian model averaging approaches typically used in ensembles since we are averaging in the weight-space (i.e. averaging model weights) instead of function-space (i.e. averaging predictions). The approximation of SWA to Fast Geometric Ensembles (Garipov et al., 2018) has been shown by previous works (Izmailov et al., 2018); however, to the best of our knowledge the connection of EMA to ensembling has not been formally shown.

Weight averaging during pseudo-labeling We maintain a separate set of non-trainable weights θ_g containing the aggregated weight average of θ_h which is used to produce pseudo-labels throughout training. In SWA, we update them via $\theta_g \leftarrow (n_a\theta_g + \theta_f)/(n_a + 1)$ at every iteration, where n_a represents the total number of models in the aggregate and θ_f are the trainable parameters of our backbone encoder as before. In practice, we switch on SWA only after some amount of training. In EMA, this update is $\theta_g \leftarrow \gamma\theta_g + (1-\gamma)\theta_f$, where γ is a momentum hyperparameter controlling how much memory θ_g should retain at each iteration (see Appendix C.2 for pseudocode). While EMA has been previously explored in the context of SSL (Tarvainen & Valpola, 2017), to the best of our knowledge SWA has not been explored in SSL and more importantly, their link to mitigating confirmation bias in pseudo-labeling has not been explicitly shown.

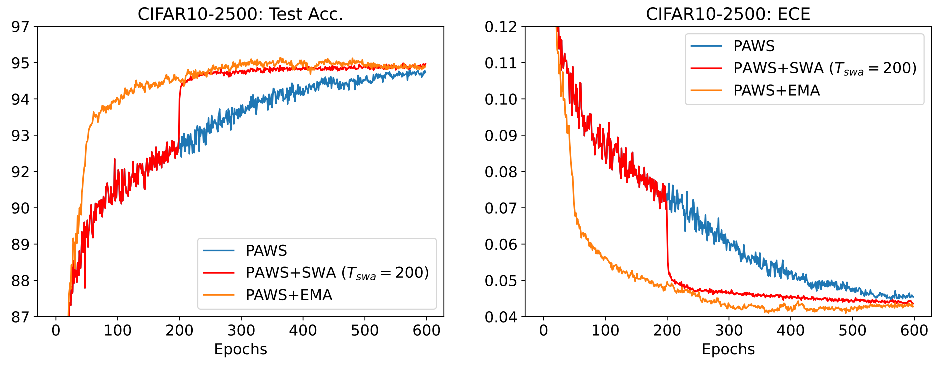

Figure 1: **Weight averaging techniques on PAWS for CIFAR-10 with 2500 labels** displays highly consistent relationships between increasing test accuracy and decreasing ECE.

B.3 BaM- is useful, but weight averaging is more effective when a selection metric is absent

Results are shown in Table 1; while BaM- gave some improvements over the baselines across all benchmarks, the improvements from weight averaging approaches are consistently larger. This is unsurprising, since BaM- relies on stochasticity only in the last layer while weight averaging aggregates over the entire model. Since PAWS does not use a selection metric and simply “accepts all unlabeled samples”, the “selection purity” is absent and thus we investigate the issue of confirmation bias by tracking the accuracy and model calibration on a held-out set as shown in Fig. 1. Incorporating weight averaging results in better accuracies on the held-out set compared to the baseline at every stage of training. This suggests that even in the absence of a selection metric, proper uncertainty estimates (or improved model calibration) can lead to improved SSL performance. The effectiveness of improved uncertainty estimates is particularly evident through PAWS+SWA—when SWA was switched on (at $T_{\text{swa}} = 200$ epochs), model calibration was immediately improved (i.e. the ECE quickly dips) and the accuracy also correspondingly spikes. Furthermore, we also observe a significant improvement in convergence rates resulting from the weight averaging techniques, i.e. better accuracies can be attained in just a third of the number of training iterations of the baseline. Ablations on T_{swa} and momentum schedules are presented in **??**.

Effective also in large-scale datasets like ImageNet. A strength of representation learning SSL methods is in large-scale datasets like ImageNet, where PAWS outperforms selection metric based SSL methods (i.e. 73.9% vs 71.5% on the best performing selection metric based method, FixMatch for the ImageNet-10% benchmark). In this work, we use the 10% labels setting of Assran et al. (2021) with slight modifications to the default setting in order to fit on our system (see Appendix D.2 for implementation details) and explored weight averaging techniques SWA and EMA, given their effectiveness over BaM- in PAWS.

Table 2: **ImageNet-10%** showing “Top-1 test accuracy (%) / ECE” for PAWS and our methods, trained for 200 epochs and fine-tuned with a linear head.

	IN-10%
PAWS	74.4 / 0.186
+SWA (ours)	74.5 / 0.186
+EMA (ours)	**75.1** / **0.183**

Table 1: “Test accuracy (%) / ECE” for PAWS (Assran et al., 2021) and our methods. ‘BaM-” refers to Bayesian Model averaging via a BNN final layer, while +SWA and +EMA are using weight averaging techniques. For each benchmark, results are averaged over 3 random dataset splits.

	CIFAR-10	CIFAR-100	
	2500 labels	4000 labels	10000 labels
PAWS (repro)	95.3±0.1 / 0.046±0.001	71.5±0.3 / 0.232±0.005	75.6±0.1 / 0.198±0.002
BaM-PAWS	95.4±0.1 (↑0.1) / 0.046±0.001	72.5±0.3 (↑1.0) / 0.228±0.003	76.5±0.2 (↑0.9) / 0.195±0.002
PAWS+SWA	95.6±0.1 (↑0.3) / 0.046±0.001	**74.5**±0.3 (↑3.0) / **0.193**±0.002	**78.4**±0.4 (↑2.8) / **0.164**±0.003
PAWS+EMA	**95.8**±0.0 (↑0.5) / **0.042**±0.000	73.5±0.2 (↑2.0) / **0.193**±0.005	77.2±0.1 (↑1.6) / 0.168±0.003

Results are shown in Table 2, where our methods provide consistent improvements in calibration and test accuracy over the baseline.

B.4 Further experimental details

B.4.1 Improving calibration in PAWS with BaM- (i.e. BaM-PAWS)

Because of the lack of a classifier layer, incorporating BaM- requires some modifications. Instead, we use a BaM- in the last layer in the encoder (which is a linear layer) and pseudo-labels are obtained by "Bayesian marginalizing" over the representations. More specifically, let h be the Bayesian version of this layer, parameterized by θ_h and let v be the input to this layer, pseudo-labels of BaM-PAWS (i.e. Eq. (2)) are defined as:

$$q_i = \pi_d(\hat{z}, \{z_s\}) = \sum_{s=1}^{B} \frac{d(\hat{z}, z_s) \cdot y_s}{\sum_{s'=1}^{B} d(\hat{z}, z_{s'})}; \quad \hat{z} = (1/M) \sum_{m}^{M} h(v, \theta_h^{(m)}) \tag{3}$$

where M is the number of samples taken from the Bayesian layer.

B.4.2 Further training details for BaM-PAWS, PAWS+EMA, PAWS+SWA

BaM-PAWS. Unlike BaM- in selection-metric based SSL methods, we do not use a separate optimizer and simply impose a one-minus-cosine scheduler (i.e. scheduler (2) from the next paragraph) for the coefficient to the KL-divergence loss that goes from 0 to 1 in T epochs where we simply picked the best from $T \in \{50, 100\}$.

PAWS+SWA and PAWS+EMA. We delay the onset of SWA to after some amount of training has elapsed (i.e. T_{swa} epochs) since it is undesirable to include the randomly initialized weights to the aggregate. We set $T_{swa} = 200$ and $T_{swa} = 100$ for CIFAR-10 and CIFAR-100 respectively. After SWA was switched on, the weight aggregate was simply updated after every batch iteration. In contrast, EMA has a natural curriculum to "forget" older model parameters since more recent parameters are given more weight in the aggregate. We experimented with two schedules for γ: 1) using a linear warm-up from 0 to 0.996 in 50 epochs and then maintaining γ at 0.996 for the rest of training and 2) using a one-minus-cosine scheduler starting from 0.05 and decreasing to 0 resulting in a 0.95 to 1 range for γ. We visualize these schedulers and include ablation studies on them and on T_{swa} in **??**.

C Formulation details and pseudocode of our methods

C.1 Bayesian model averaging via a BNN final layer

Following the notations in Section 4.1, we denote the BNN layer to be h and an input embedding to this layer to be v in this section. We assume a prior distribution on weights $P(\theta_h)$ and seek to calculate the posterior distribution of weights given the empirical/training data, $P(\theta_h|\mathcal{D}_{\mathcal{X}})$, where $\mathcal{D}_{\mathcal{X}} := (X, Y)$, which can then be used to compute the posterior predictive during inference. As exact Bayesian inference is intractable for neural networks, we adopt a variational approach following (Blundell et al., 2015) to approximate the posterior with a Gaussian distribution parameterized by ϕ, $q(\theta|\phi)$. From now, we will drop the h index in θ_h for brevity. To learn the variational approximation, we seek to minimize the Kullback-Leibler (KL) divergence between the Gaussian variational approximation and the posterior:

Algorithm 1 PyTorch-style pseudocode for Bayesian model averaging in UDA or FixMatch.

```
# f: backbone encoder network
# h: bayesian classifier
# KL_loss: KL term in evidence lower-bound
# H: cross-entropy loss
# Q: quantile parameter
# num_samples: number of weight samples
# method: ‘UDA‘ or ‘FM‘
# shp: sharpen operation

q_list = []
for (xl, labels), xu in zip(labeled_loader, unlabeled_loader):
    x_lab, x_uw, x_us = weak_augment(xl), weak_augment(xu), strong_augment(xu)
    z_lab, z_uw, z_us = f(x_lab), f(x_uw), f(x_us) # get representations
    mean_uw, std_uw = bayes_predict(h, z_uw)
    q_list.pop(0) if len(q_list) > 50 # keep 50 most recent quantiles
    q_list.append(quantile(std_uw, Q))
    mask = std_uw.le(q_list.mean())

    # compute losses
    loss_kl = KL_loss(h) # prior-dependent (data-independent) loss
    loss_lab = H(h(z_lab), labels)
    if method == ‘UDA‘:
        loss_unlab = H(h(z_us), shp(mean_uw)) * mask # sharpened soft pseudo-labels
    elif method == ‘FM‘:
        loss_unlab = H(h(z_us), mean_uw.argmax(-1)) * mask # hard pseudo-labels
    loss = loss_lab + loss_unlab + loss_kl
    loss.backward()
    optimizer.step()

def bayes_predict(h, z):
    outputs = stack([h(z).softmax(-1) for _ in range(num_samples)] # sample weights
    return outputs.mean(), outputs.std() # mean and std of predictions
```

$$\begin{aligned}
\phi^* &= \text{argmin}_\phi \text{KL}\left(q(\theta|\phi) \,\|\, p(\theta|X,Y)\right)\\
&= \text{argmin}_\phi \int q(\theta\,|\,\phi)\text{log}\frac{q(\theta\,|\,\phi)}{p(\theta\,|\,X,Y)}d\theta\\
&= \text{argmin}_\phi \mathbb{E}_{q(\theta|\phi)}\text{log}\frac{q(\theta\,|\,\phi)p(Y|X)}{p(Y|X;\theta)p(\theta)}\\
&= \text{argmin}_\phi \mathbb{E}_{q(\theta|\phi)}\left[\text{log}q(\theta\,|\,\phi) - \text{log}p(Y|X;\theta) - \text{log}p(\theta)\right] + \text{log}p(Y|X)\\
&= \text{argmin}_\phi \text{KL}(q(\theta\,|\,\phi)\|p(\theta)) - \mathbb{E}_{q(\theta|\phi)}\text{log}p(Y|X;\theta) + \text{log}p(Y|X)\\
&= \text{argmin}_\phi \left([-\text{ELBO}] + \text{log}p(Y|X)\right)
\end{aligned}$$

where in the last line, ELBO $= \mathbb{E}_{q(\theta|\phi)}\left[\text{log}p(Y|X;\theta)\right] - \text{KL}(q(\theta\,|\,\phi)\|p(\theta))$ is the evidence lower-bound which consists of a log-likelihood (data-dependent) term and a KL (prior-dependent) term. Since KL $\left(q(\theta|\phi) \,\|\, p(\theta|\mathcal{D}_\mathcal{X})\right)$ is intractable, we maximize the ELBO which is equivalent to minimizing the former up to the constant, $\text{log}p(Y|X)$.

Each variational posterior parameter of the Gaussian distribution, ϕ, consists of the mean (μ) and the standard deviation (which is parametrized as $\sigma = \log(1 + \exp(\rho))$ so that σ is always positive (Blundell et al., 2015)), i.e. $\phi = (\mu, \rho)$. To obtain a sample of the weights θ, we use the reparametrization trick (Kingma et al., 2015) and sample $\epsilon \sim \mathcal{N}(0, I)$ to get $\theta = \mu + \log(1 + \exp(\rho)) \circ \epsilon$ where $\circ$ denotes elementwise multiplication. In other words, we double the number of learnable parameters in the layer compared to a non-Bayesian approach; however, this does not add a huge computational cost since only the last layer is Bayesian and the dense backbone remains non-Bayesian.

Algorithm 1 shows the PyTorch-style pseudo-code for BaM- in selection-metric based methods (here showing asymmetric augmentation applicable for UDA or FixMatch). The main modifications upon the baseline methods include 1) computation of an additional KL loss term (between two Gaussians, i.e. the variational approximation and the prior), 2) taking multiple samples of weights to derive predictions and 3) replacing the acceptance criteria from using maximum probability to using standard deviation of predictions.

Algorithm 2 PyTorch-style pseudocode for PAWS-SWA and PAWS-EMA.

```
# f: backbone encoder network
# g: weight aggregated encoder network
# shp: sharpen operation
# snn: PAWS soft nearest neighbour classifier
# H: cross entropy loss
# ME_max: mean entropy maximization regularization loss
# N: total number of epochs
# use_swa: boolean to use SWA
# use_ema: boolean to use EMA
# swa_epochs: number of epochs before switching on SWA
# gamma: momentum parameter for EMA

g.params = f.params # initialized as copy
g.params.requires_grad = False # remove gradient computations
num_swa = 0
for i in range(N):
    for x in loader:
        x1, x2 = augment(x), augment(x) # augmentations for x
        p1, p2 = snn(f(x1)), snn(f(x2))
        q1, q2 = snn(g(x1)), snn(g(x2))

        loss = H(p1, shp(q2))/2 + H(p2, shp(q1))/2 + ME_max(cat(q1,q2))
        loss.backward()
        optimizer.step()

        if use_swa:
            if i > swa_epochs: # update aggregate
                num_swa += 1
                g.params = (g.params * num_swa + f.params) / (num_swa + 1)
            else:
                g.params = f.params # weights are just copied
        elif use_ema:
            g.params = gamma * g.params + (1-gamma) * f.params # update momentum aggregate
        else:
            g_params = f.params # PAWS baseline
```

C.2 Algorithm for SWA & EMA in PAWS

Algorithm 2 shows the pseudocode for the implementation of PAWS+SWA and PAWS+EMA in PyTorch. For brevity, we leave out details of the multicrop strategy, mean entropy maximization regularization and soft nearest neighbour classifier formulation which are all replicated from the original implementation. We defer readers to the original paper (Assran et al., 2021) for these details.

D Further implementation details

D.1 Hyperparameters for various selection-metric based methods

All selection-metric based methods in this study uses an optimization loss function of the form of Eq. (1) from the main text, with differences in the hyperparameters μ, λ, τ, ρ_t and α. We use the hyperparameters from the original implementations. For Pseudo-Labels (Lee, 2013), $\mu = 1$, $\tau = 0.95$ and $\rho_{t=0}$ (i.e. hard labels); for UDA (Xie et al., 2019), $\mu = 7$, $\tau = 0.8$, $\rho_{t=0.4}$ (i.e. soft pseudo-labels sharpened with temperature of 0.4); for FixMatch (Sohn et al., 2020), $\mu = 7$, $\tau = 0.95$, $\rho_{t=0}$ (i.e. hard labels). All methods use $\lambda = 1$. In addition, UDA and FixMatch uses asymmetric transforms for the two legs of sample and pseudo-label prediction, i.e. α_1 is a weak transform (based on the standard flip-and-shift augmentation) and α_2 is a strong transform (based on RandAugment (Cubuk et al., 2019)). Pseudo-Labels uses symmetric weak transforms for both legs.

In our calibrated versions of all these methods (i.e. "BaM-X"), we maintained the exact same hyperparameter configuration as its corresponding baseline. The only exception is the sharpening temperature of BaM-UDA, which uses $t = 0.9$ instead of $t = 0.4$ in UDA, as we found that calibration enables, and was highly effective with, the use of soft pseudo-labels).

D.2 Implementation details for ImageNet experiments

We maintain the default 64 GPU training configuration recommended by the authors and make slight modifications to the default implementations to fit on our hardware. On our set up (which does not use

Nvidia's Apex package due to installations issues), we found training to be unstable on half-precision and had to use full-precision training. In order to fit into memory, we had to decrease the number of images per class from 7 to 6, resulting in a slightly lower baseline performance from the reported for 200 epochs of training (see Table 4 in Assran et al. (2021) for the study on the correlation between the number of images per class and the final accuracy). On all ImageNet experiments on PAWS, we follow the validation and testing procedure of PAWS (Assran et al., 2021) and swept over the same set of hyperparameters during fine-tuning of the linear head. We report the ECE at the final checkpoint.

E Long-tailed CIFAR-10 and CIFAR-100

E.1 Dataset preparation

We create long-tailed versions of CIFAR-10 and CIFAR-100 following the procedure from Cao et al. (2019), i.e. by removing the number of training examples per class from the standard training set with 50,000 samples. We create the class-imbalance unlabeled dataset with an exponential decay where the severity of the imbalance is given by the imbalance ratio $\alpha = \max_i(n_{u,i})/\min_i(n_{u,i}) \in \{10, 100\}$, where $n_{u,i}$ is the number of unlabeled examples for class i. The number of samples in the most frequent class is 5,000 for CIFAR-10 and 500 for CIFAR-100. To create the labeled set, we randomly select 10% of samples *from each class*, under the constrain that at least 1 sample for each class is included in the labeled set, i.e. $n_{l,i} = \min(1, 0.1 * n_{u,i})$, where $n_{l,i}$ is the number of labeled examples for class i. The total number of labeled and unlabeled examples for the different benchmarks in CIFAR-10-LT and CIFAR-100-LT are summarized in Table 3. The test set remains unchanged, i.e. we use the original (class-balanced) test set of the CIFAR datasets with 10,000 samples.

Table 3: **Dataset statistics for CIFAR-10-LT & CIFAR-100-LT** showing total number of examples in the labeled and unlabeled datasets for each benchmark.

	CIFAR-10-LT		CIFAR-100-LT	
	$\alpha = 10$	$\alpha = 100$	$\alpha = 10$	$\alpha = 100$
Labeled	2,041	1,236	1,911	1,051
Total	20,431	12,406	19,573	10,847

We used the exact same configuration and hyperparameters as the original (non-long-tailed) CIFAR benchmarks, i.e. the architecture is WideResNet-28-2 for CIFAR-10-LT and WideResNet-28-10 for CIFAR-100-LT. All training hyperparameters used to obtain the results for FM and BaM-UDA in Table 3 of the main text also follow the ones from the original CIFAR benchmarks.

E.2 Additional classwise results

Fig. 2 shows the test accuracies for the baseline (FM) and ours (BaM-UDA) after separating samples from the test set into three groups — the "Many" group which contains classes with more than 100 samples, the "medium" group which contains classes between 10 and 100 samples and the "few" group which contains classes less than 10 samples, for the benchmark of CIFAR-100-LT with $\alpha = 10$ and 10% labels. We see that BaM-UDA outperforms FM on every group; in addition, the gap between BaM-UDA and FM increases as the number of samples are more scarce, highlighting BaM-UDA's utility in improving accuracy over the baseline in the more difficult classes.

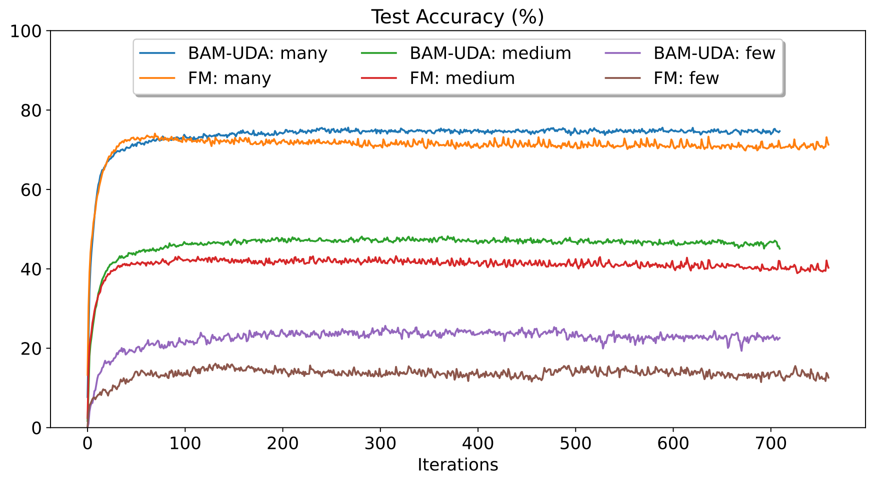

Figure 2: **CIFAR100-LT**. Accuracies for samples based on their their classwise sample frequency, "many" for classes with >100 samples, "medium" for classes between 10-100 samples and "few" for classes between <10 samples. Dataset is CIFAR-100-LT with $\alpha = 10$ and 10% labels.

F Semi-supervised Learning in Photonics Science

Semi-supervised learning is highly important and applicable to domains like Science where labeled data is particularly scarce, owing to the need for computationally expensive simulations and labor-intensive laboratory experiments for data collection. To demonstrate the effectiveness of our proposed techniques in the Science domain, we use an example problem in Photonics, adopting the datasets from Loh et al. (2022). The task we studied in this work is a 5-way classification of photonic crystals (PhC) based on their band gap sizes.

PhCs are periodically-structured materials engineered for wide ranging applications by manipulating light waves (Joannopoulos et al., 2008) and an important property of these crystals is the size of their band gap (often, engineers seek to design photonic crystals that host a substantial band gap (Christensen et al., 2020)). We adopt the dataset of PhCs from Loh et al. (2022), which consists of 32,000 PhC samples and their corresponding band gaps which had been pre-computed through numerical simulations (Johnson & Joannopoulos, 2001). Examples of PhCs and an illustration of band gap from the dataset is shown in Fig. 3. We binned all samples in the dataset into 5 classes based on their band gap sizes; since there was a preponderance (about 25,000) of samples without a band gap, we only selected 5,000 of them in order to limit the severity of class imbalance and form a new dataset with just 11,375 samples (see Fig. 3c). Notably the long-tailed distribution of this dataset is characteristic of many problems in science (and other real-world datasets), where samples with the desired properties (larger band gap) are much rarer than trivial samples. From this reduced dataset, we created two PhC benchmarks with 10% labels (PhC-10%) and 1% labels (PhC-1%), where 10% and 1% of samples *from each class* are randomly selected to form the labeled set respectively. The class-balanced test set is fixed with 300 samples per class (total of 1,500 samples), the unlabeled set consists of 9,876 samples and the labeled sets consists of 1,136 samples and 113 samples for PhC-10% and PhC-1% respectively.

For these benchmarks, we used a WideResNet-28-2 architecture, with a single channel for the first CNN layer, and made the following changes upon FixMatch and BaM-UDA. We set $\mu = 1$ and instead of 2^{20} iterations, we trained for 300 epochs (where each epoch is defined as iterating through the unlabeled set once). For BaM-UDA, we set $Q = 0.9$ and use a one-minus-cosine warm-up scheduler of 50 epochs to the KL loss coefficient (resulting in a coefficient going from 0 to 1 in 50 epochs). For each benchmark, we swept the learning rate across $\{0.01, 0.001\}$ and select the best model for both the baseline and ours. The standard image augmentations used in vision problems cannot be applied to this problem, since it destroys the scientific integrity of the data (e.g. cropping the PhC input would result in a completely different band gap profile). Instead, we used the augmentations proposed in Loh et al. (2022) (periodic translations, rotations and mirrors)

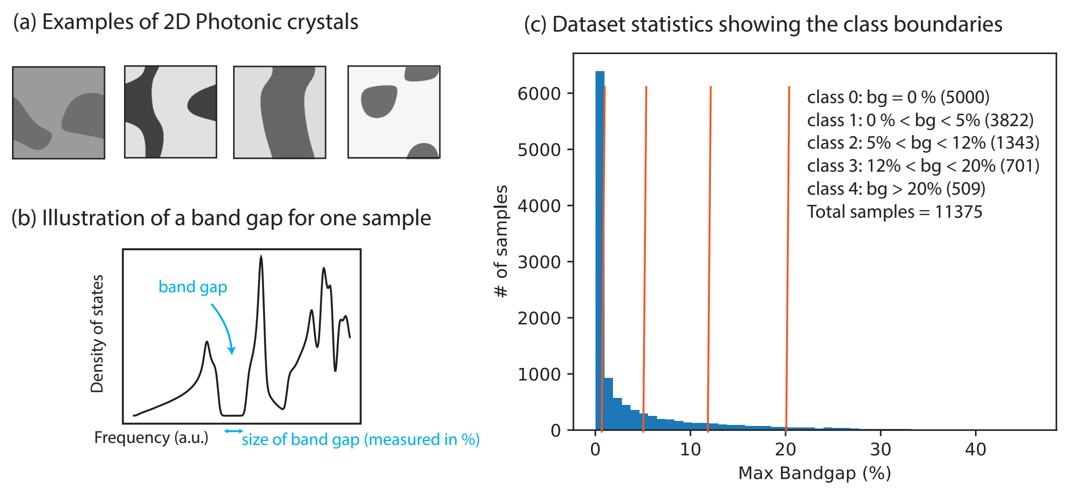

Figure 3: **Photonics dataset.** (a) Examples of 2D periodic photonic crystals (PhC), (b) illustration of the band gap size of a single PhC seen from its density-of-states, a common spectrum of interest for PhCs (Loh et al., 2022), and (c) dataset statistics for the task used in this work where the samples were binned into 5 classes based on their band gap (bg) with class boundaries detailed in the inset. Numbers in parenthesis show the number of samples in each class, showing strong class imbalance.

and applied them symmetrically (i.e. there is no distinction of strong and weak augmentations). In addition, we included these augmentations to the labeled samples as well.

G Ablation Studies

G.1 Further ablation studies on BaM-

Number of weight samples. Bayesian model averaging produces better calibrated predictions through "Bayesian marginalization", i.e. by averaging over multiple predictions instead of using a single prediction of the model. In Table 4, we show ablation studies on the number of weight samples taken from the variational posterior in the BNN layer used for "Bayesian marginalization", i.e. for computing the posterior predictive. We observe an overall trend that increasing the number of weight samples lead to better calibration and final test accuracies, providing direct evidence for the effectiveness in using Bayesian model averaging towards improving calibration. In order to limit the computational overhead arising from taking a large number of weight samples, given that pseudo-labeling happens at every iteration, we limit M to 50 in our study. The computational overhead for $M = 50$ is discussed in Appendix H, where we see that incorporating Bayesian model averaging incurs negligible computational overhead.

Table 4: Ablation of number of weight samples, M, taken from the variational posterior in BaM-UDA. Dataset is CIFAR-100 with 400 labels. Highlighted in cyan is the main configuration used.

M	Test Accuracy	ECE
2	50.0	0.403
5	58.1	0.342
10	58.5	0.336
50	59.7	0.327

A basic semi-supervised learning setup. Many SOTA semi-supervised learning methods incorporate several techniques like weak and strong augmentation, sharpening of pseudo-labels and selection criteria to achieve the best performance. An interesting study is to investigate a basic semi-supervised learning setup where many of these techniques are removed. In Table 5 of the main text, we investigated the importance of the different features in BaM- and found that even without a selection metric, BaM still leads to some performance gains. Here, we study a simple set up of semi-supervised learning, where we remove sharpening and the selection metric and compare this basic baseline with one where the last layer is replaced with a BNN via BaM-; results are shown in Table 5. Results show that performance gains via BaM- are consistent and are effective independent from the techniques commonly introduced in SSL methods.

Table 5: Basic semi-supervised learning setup without selection criteria.

	CIFAR-100-400	CIFAR-100-4000
Basic setup	53.5	74.0
Basic setup + BaM	54.6	74.9

H Computational Requirements and Additional Computational Costs

All CIFAR-10 and CIFAR-100 experiments in this work were computed using a single Nvidia V100 GPU. ImageNet experiments were computed using 64 Nvidia V100 GPUs. A key aspect of our proposed calibration methods is the requirement of adding minimal computational cost to the baseline approaches. In the following paragraphs we list the additional computational cost (based on wall-clock time on the same hardware) for our proposed methods.

Computational cost of Bayesian model averaging. The main additional computational cost comes from executing several (in our case, 50) forward passes through the weight samples of the BNN layer when deriving predictions for the unlabeled samples. This additional overhead is minimal since only the final layer, which consists of a small fraction of the total network weights, is (approximate) Bayesian. On the benchmarks of CIFAR-100 where gains are larger, we found the BNN versions to take only around $2-5\%$ longer in wall-clock time on the same hardware when compared to the baseline, for the same total number of iterations. This justifies the BNN layer as a plug-in calibration approach with low computational overhead. The recorded run times for 500 epochs for each method on the same hardware are shown in Table 6.

Table 6: **Wall-clock run time (in hours) for 500 epochs for each method.**

	CIFAR-10		CIFAR-100		
	250 labels	2500 labels	400 labels	4000 labels	10000 labels
FM	30.8	19.3	77.2	74.7	71.9
BaM-FM	36.5	21.5	79.7	75.1	73.7
UDA	40.8	23.8	77.8	73.7	73.2
BaM-UDA	50.1	25.6	80.6	75.4	74.5

Computational cost of SWA & EMA. Both SWA and EMA requires storing another network of the same architecture, denoted g in the main text, which maintains the aggregate (exponentially weighted aggregate) of past network weights for SWA (EMA). Rather than using representations from the original backbone, f, representations from g are used to derive the better calibrated predictions, and thus the main computational overhead comes from the second forward pass needed per iteration. We timed the methods and found that PAWS+SWA and PAWS+EMA took around 18% longer in wall clock training time (on the same hardware) when compared to the baseline PAWS method for each epoch of training. However, as shown in the main text, PAWS+SWA and PAW+EMA resulted in a significant speed up in convergence, cutting training time by $>60\%$ and additionally giving better test performances. Hence rather than an overhead, improved calibration in PAWS+SWA and PAWS+EMA in fact reduces the computation cost needed for the original approach.

I Extreme low-label settings in CIFAR-100

Table 7 compares BaM-UDA against FixMatch and UDA in the extremely low-label settings for CIFAR-100 such as 100 labels and 200 labels. These are settings with extremely small number of labels since there is only 1 label per class (the lowest possible) for the 100 labels setting and 2 labels per class for the 200 labels setting. In the extreme setting of 1 label per class, BaM-UDA gives a decent 35.2% accuracy, a whopping 27% improvement from the UDA baseline (without the last bayesian layer) and a 19% improvement from the next best method (FixMatch).

Table 7: **Extremely low labels in CIFAR-100.**

	CIFAR-100		
	100 labels	200 labels	400 labels
FM	16.0$\pm$3.3	45.6$\pm$0.3	56.4$\pm$1.6
UDA	8.2$\pm$0.7	19.0$\pm$0.1	44.1$\pm$0.7
BaM-UDA	**35.2**$\pm$1.3	**51.2**$\pm$0.2	**60.3**$\pm$0.6

J Societal Impact and Ethical Considerations

Semi-supervised learning is arguably one of the most important deep learning applications, as in real life we often have access to an abundance of unlabeled examples, and only a few labeled datapoints. Our work highlights the importance of calibration in semi-supervised learning methods which could yield benefits for real-world applications in two main ways: 1) a better deep learning model that is more data efficient and 2) a model that is better calibrated and can quantify uncertainty better. The latter is highly important for crucial societal applications such as in healthcare. However, as with any deep learning application, there may be biases accumulated during dataset collection or assimilated during the training process. This issue may be more acute in a semi-supervised setting where the small fraction of labels may highly misrepresent the ground truth data. This may lead to unfair and unjust model predictions and give rise to ethical concerns when used for societal applications.

References

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael G. Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. *CoRR*, abs/2104.13963, 2021. URL `https://arxiv.org/abs/2104.13963`.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015. URL `https://arxiv.org/abs/1505.05424`.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 02 2013. ISBN 9780199535255. doi: 10.1093/acprof:oso/9780199535255.001.0001. URL `https://doi.org/10.1093/acprof:oso/9780199535255.001.0001`.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss, 2019. URL `https://arxiv.org/abs/1906.07413`.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021. URL `https://arxiv.org/abs/2104.14294`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020a. URL `https://arxiv.org/abs/2002.05709`.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners, 2020b. URL https://arxiv.org/abs/2006.10029.

Thomas Christensen, Charlotte Loh, Stjepan Picek, Domagoj Jaković, Li Jing, Sophie Fisher, Vladimir Ceperic, John D. Joannopoulos, and Marin Soljačić. Predictive and generative machine learning models for photonic crystals. *Nanophotonics*, 9(13):4183–4192, October 2020. ISSN 2192-8614. doi: 10.1515/nanoph-2020-0197. URL https://www.degruyter.com/document/doi/10.1515/nanoph-2020-0197/html.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. URL https://arxiv.org/abs/1909.13719.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018. URL https://arxiv.org/abs/1802.10026.

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. Pac-bayesian theory meets bayesian inference. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/84d2004bf28a2095230e8e14993d398d-Paper.pdf.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2018. URL https://arxiv.org/abs/1803.05407.

J. D. Joannopoulos, S. G. Johnson, J. N. Winn, and R. D. Meade. *Photonic Crystals: Molding the Flow of Light*. Princeton University Press, 2 edition, 2008. URL http://ab-initio.mit.edu/book/.

Steven G. Johnson and J. D. Joannopoulos. Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis. *Optics Express*, 8(3):173–190, January 2001. ISSN 1094-4087. doi: 10.1364/OE.8.000173. URL https://www.osapublishing.org/oe/abstract.cfm?uri=oe-8-3-173.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick, 2015. URL https://arxiv.org/abs/1506.02557.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2016. URL https://arxiv.org/abs/1612.01474.

Dong-hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, 2013.

Charlotte Loh, Thomas Christensen, Rumen Dangovski, Samuel Kim, and Marin Soljacic. Surrogate- and invariance-boosted contrastive learning for data-scarce applications in science. *Nat Commun*, 13, 2022. URL https://doi.org/10.1038/s41467-022-31915-y.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020. URL https://arxiv.org/abs/2001.07685.

Antti Tarvainen and Harri Valpola. Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, abs/1703.01780, 2017. URL http://arxiv.org/abs/1703.01780.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation. *CoRR*, abs/1904.12848, 2019. URL http://arxiv.org/abs/1904.12848.