Deep Reference Priors: What is the best way to pretrain a model?

Yansong Gao [*1] Rahul Ramesh [*2] Pratik Chaudhari [3]

Abstract

What is the best way to exploit extra data—be it unlabeled data from the same task, or labeled data from a related task—to learn a given task? This paper formalizes the question using the theory of reference priors. Reference priors are objective, uninformative Bayesian priors that maximize the mutual information between the task and the weights of the model. Such priors enable the task to maximally affect the Bayesian posterior, e.g., reference priors depend upon the number of samples available for learning the task and for very small sample sizes, the prior puts more probability mass on low-complexity models in the hypothesis space. This paper presents the first demonstration of reference priors for medium-scale deep networks and image-based data. We develop generalizations of reference priors and demonstrate applications to two problems. First, by using unlabeled data to compute the reference prior, we develop new Bayesian semi-supervised learning methods that remain effective even with very few samples per class. Second, by using labeled data from the source task to compute the reference prior, we develop a new pretraining method for transfer learning that allows data from the target task to maximally affect the Bayesian posterior. Empirical validation of these methods is conducted on image classification datasets. Code is available at https://github.com/grasp-lyrl/deep_reference_priors.

[*]Equal contribution [1]Applied Mathematics and Computational Sciences, University of Pennsylvania [2]Computer and Information Science, University of Pennsylvania [3]Electrical and Systems Engineering, University of Pennsylvania. Correspondence to: Yansong Gao <gaoyans@sas.upenn.edu>, Rahul Ramesh <rahulram@seas.upenn.edu>, Pratik Chaudhari <pratikac@seas.upenn.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

1. Introduction

Exploiting extra data, e.g., labeled data from a related task, or unlabeled data from the same task, is a powerful way of reducing the number of training data required to learn a given task. This idea lies at the heart of burgeoning fields like transfer, meta-, semi- and self-supervised learning, and these fields have developed a wide variety of methods to incorporate such extra information. To give a few examples, methods for transfer learning fine-tune a representation that was pretrained on labeled data from another—ideally related—task. Methods for semi-supervised learning pretrain the representation using unlabeled data, which may come from the same task or from other related tasks, before using the labeled data. In this paper, we ask the question: what is the *best* way to exploit extra data for learning a task? In other words, if we have *some* pool of data—be it labeled or unlabeled, from the same task, or from another task—what is the *optimal* way to pretrain a representation?

As posed, the answer to the question above depends upon the downstream task that we seek to solve. But we can ask a more reasonable question by recognizing that a pretrained representation can be thought of as a Bayesian prior (or a sample from it). Fundamentally, a prior restricts the set of models that can be fitted upon the task. So we could instead ask: *how to best use the extra data to restrict the set of models that we could fit on the desired task*. This paper formalizes the question using the concept of reference priors and makes the following contributions.

(1) We **formalize the problem of "how to best pretrain a model"** using the theory of reference priors, which are objective, uninformative Bayesian priors computed by maximizing the mutual information between the task and the weights. We show how these priors maximize the KL-divergence between the posterior computed from the task and the prior, on average over the distribution of the unknown future data. This allows the samples from the task to maximally influence the posterior. We discuss how reference priors are supported on a discrete set of atoms in the weight space. We **develop a method to compute reference priors for deep networks**. To our knowledge, this is the **first instantiation of reference priors for deep networks that preserves their characteristic discrete nature**.

(2) We **formalize semi-supervised learning as computing a reference prior** where the learner is given access to a pool of unlabeled data and seeks to compute a prior using this data. This formulation sheds light upon the **theoretical underpinnings of existing state of the art methods such as FixMatch**. We show that techniques such as consistency regularization and entropy minimization which are commonly used in practice can be directly understood using the reference prior formulation.

(3) We **formalize transfer learning as building a two-stage reference prior** where the learner gets access to data in two stages and computes a prior that is optimal for data from the second stage. Such a prior has the flavor of ignoring certain parts of the weight space depending upon whether data from the first stage was similar to that from the second stage, or not. This formulation is useful because it is an information-theoretically optimal way to pretrain using a source task for the goal of transferring to the target task. This objective is closely related to the predictive Information Bottleneck principle.

(4) We show an empirical study of our formulations on the CIFAR-10 and CIFAR-100 datasets. We show that **our methods to compute reference priors provide results that are competitive with state of the art** methods for semi-supervised learning, e.g., we obtain an **accuracy of 85.45% on CIFAR-10 with 5 labeled samples/class**. We obtain significantly better accuracy than well-tuned fine-tuning for transfer learning, even for very small sample sizes.

2. Background

2.1. Setup

Consider a dataset $\hat{P}_n = \{(x_i, y_i)\}_{i=1}^n$ with n samples that consists of inputs $x_i \in \mathbb{R}^d$ and labels $y_i \in \{1, \dots, C\}$. Each sample of this dataset is drawn from a joint distribution $P(x, y)$ which we define to be the "task". We will use the shorthand $x^n = (x_1, \dots, x_n)$ and $y^n = (y_1, \dots, y_n)$ to denote all inputs and labels. Let $w \in \mathbb{R}^p$ be the weights of a probabilistic model which evaluates $p_w(y \mid x)$. We will use a random variable z with a probabilistic model $p_w(z)$ when we do not wish to distinguish between inputs and labels.

Given a prior on weights $\pi(w)$, Bayes law gives the posterior $p(w \mid x^n, y^n) \propto p(y^n \mid x^n, w)\pi(w)$. The Fisher Information Matrix (FIM) $g \in \mathbb{R}^{p \times p}$ has entries $g(w)_{kl} =$

$$\frac{1}{n}\sum_{i=1}^n \sum_{y=1}^C p_w(y \mid x_i)\partial_{w_k} \log p_w(y \mid x_i)\partial_{w_l} \log p_w(y \mid x_i).$$

It can be used to define the Jeffreys prior $\pi_J(w) \propto \sqrt{\det g(w)}$. Jeffreys prior is reparameterization invariant, i.e., it assigns the same probability to a set of models irrespective of our choice of parameterization of those models. It is an uninformative prior, e.g., it imposes some generic structure on the problem (reparameterization invariance).

2.2. Reference Priors

To make the choice of a prior more objective, Bernardo (1979) suggested that uninformative priors should maximize some divergence, say the Kullback-Leibler (KL) divergence $\mathrm{KL}(p(w \mid z), \pi(w)) = \int \mathrm{d}w\, p(w \mid z) \log\left(p(w \mid z)/\pi(w)\right)$, between the prior and the posterior for data z. The rationale for doing so is to allow the data to dominate the posterior rather than our choice of the prior. Since we do not know the data *a priori* while picking the prior, we should maximize the *average* KL-divergence over the data distribution $p(z)$. This amounts to maximizing the mutual information

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}}\, I_\pi(w; z) \\ &:= \int \mathrm{d}z\, \mathrm{d}w\, p(z)p(w \mid z) \log \frac{p(w \mid z)}{\pi(w)} \\ &= H(w) - H(w \mid z) \end{aligned} \tag{1}$$

where $p(z) = \int \mathrm{d}w\, \pi(w)p(z \mid w)$ and $H(w) = -\int \mathrm{d}w\, \pi(w) \log \pi(w)$ is the Shannon entropy; the conditional entropy $H(w \mid z)$ is defined analogously. Mutual information is a natural quantity for measuring the amount of missing information about w provided by data z if the initial belief was π. The prior $\pi^*(w)$ is known as a reference prior. It is invariant to a reparameterization of the weight space because mutual information is invariant to reparameterization. The reference prior does not depend upon the samples $\hat{P}_n$ but only depends on their distribution P.

The objective to calculate reference prior π^* above may not be analytically tractable and therefore Bernardo also suggested computing n-reference priors. We call n the "order" and deliberately overload the notation for the number of samples n; the reason will be clear soon.

$$\pi_n^* = \operatorname{argmax}_\pi I_\pi(w; z^n) = H(w) - H(w \mid z^n), \tag{2}$$

using n samples and then setting $\pi^* := \lim_{n \to \infty} \pi_n^*$ under appropriate technical conditions (Berger et al., 1988). Reference priors are asymptotically equivalent to Jeffreys prior for one-dimensional problems. In general, they differ for multi-dimensional problems but it can be shown that Jeffreys prior is the continuous prior that maximizes the mutual information (Clarke and Barron, 1994).

2.3. Blahut-Arimoto algorithm

The Blahut-Arimoto algorithm (Arimoto, 1972; Blahut, 1972) is a method for maximizing functionals like (1) and leads to iterations of the form $\pi^{t+1}(w) \propto \exp\left(\mathrm{KL}(p(z \mid w), p(z))\right) \pi^t(w)$. It is typically implemented for discrete variables, e.g., in the Information Bottleneck (Tishby et al., 1999). In this case, maximizing mutual information is a convex problem and therefore the BA algorithm is guaranteed to converge. Such discretization is difficult for high-dimensional deep networks. We therefore implement the BA algorithm using particles; see Remark 2.

Example 1 (Estimating the bias of a coin). To ground intuition, consider the estimation of the bias of a coin $w \in [0, 1]$ using n trials. If z denotes the number of heads (which is a sufficient statistic), we have $p(z \mid w) = w^z(1-w)^{n-z}n!/(z!(n-z)!)$. For $n = 1$, since we know that $I(w; z^1) \leq \log 2$ with this one bit of information, we can see that $\pi_1^*(z) = (\delta(w) + \delta(1-w))/2$ is the reference prior that achieves this upper bound. This result is intuitive: if we *know* that we have only one observation, then the optimal uninformative prior should put equal probability mass on the two exhaustive outcomes $w = 0$ (heads) and $w = 1$ (tails). We can numerically calculate π_n^* for different values of n using the BA algorithm (Fig. 1).

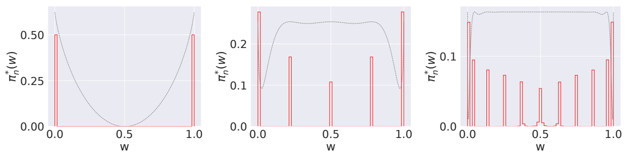


Figure 1. We calculated the **reference prior for the coin-tossing model** for $n = 1, 10, 50$ (from left to right) using the Blahut-Arimoto algorithm. Atoms are critical points of the gray line which is $\mathrm{KL}(p(z^n), p(z^n \mid w))$. The prior is discrete for finite order $n < \infty$ (Mattingly et al., 2018). Atoms of the prior are maximally different from each other, e.g., for $n = 1$, they are on opposite corners of the parameter space. As the number of samples increases, the separation between atoms of the prior reduces. The prior converges to Jeffreys prior $\pi_J(w) \propto (w(1-w))^{-1}$ as $n \to \infty$.

3. Methods

This section discusses a key property of reference priors that enables us to calculate them numerically, namely that they are supported on a discrete set in the weight space (§3.1). It then formulates reference priors for semi-supervised (§3.3) and transfer learning (§§3.4 and 3.5).

3.1. Existence and discreteness of reference priors

Rigorous theoretical development of reference priors has been done in the statistics literature. We focus on their applications. We however mention some technical conditions under which our development remains meaningful.

A reference prior does not exist if $I_\pi(w; z^n)$ is infinite (Berger et al., 1988). For the concept of a reference prior to remain meaningful, we make the following technical assumptions. (i) π is supported on a compact set $\Omega \subset \mathbb{R}^p$, and (ii) if $p_\pi(z^n) = \int_\Omega \mathrm{d}w\, \pi(w) p(z^n \mid w)$ is the marginal, then $\mathrm{KL}(p_w, p_\pi)$ is a continuous function of w for any π. Under these conditions, the n-order prior π_n^* exists and $I_{\pi_n}(w; z^n)$ is finite; see (Zhang, 1994, Lemma 2.14). Now assume that π_n^* exists and is unique up to a set of measure zero. Let $\Omega_n = \{w \in \Omega : \pi_n^*(w) > 0\}$ be the support of π_n^* and z^n be a discrete random variable with C atoms. If $\{p(z^n \mid w) : w \in \Omega_n\}$ is compact, then π_n^* is discrete with no more than C atoms (Zhang, 1994, Lemma 2.18)).

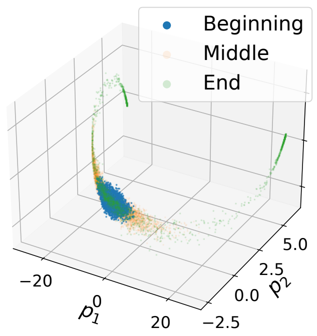


Figure 2. **Reference prior (green) for binary classification on MNIST**. A three-dimensional embedding of the probability distributions of $K = 3000$ atoms in the reference prior after 50,000 iterations of the BA algorithm (green) for a binary classification problem on MNIST (digits 3 vs. 5). Particles were initialized randomly (blue) and they are nearby in this embedding because at initialization, the logits of each particle are uniformly distributed. Orange shows particle locations after 5,000 iterations. As the reference prior objective in (2) is optimized, the particles increasingly make more diverse predictions (orange) and towards the end (green) these particles spread apart in the prediction space.

Remark 2 (Blahut-Arimoto algorithm with particles). Since the optimal prior is discrete, we can maximize the mutual information directly by identifying the best set of atoms. We set the prior have the form $\pi_n^* = \sum_{i=1}^K K^{-1}\delta(w - w^i)$ where $\{w^1, \ldots, w^K\}$ are the K atoms. We call these atoms "particles". Using standard back-propagation, we can then compute the gradient of the objective in (2) with respect to each particle (note that each particle's gradient depends upon all other particles).

3.2. Visualizing the reference prior for deep networks

One cannot directly visualize the high-dimensional particles w in π_n^*. But we can think of each particle w as representing a probability distribution $f(w) \in \mathbb{R}^{nC}$ given by

$$\left(\sqrt{p_w(y=1 \mid x_1)}, \sqrt{p_w(y=2 \mid x_1)}, \ldots, \sqrt{p_w(y=C \mid x_n)}\right).$$

and use a method for visualizing such distributions developed in Quinn et al. (2019) that computes a principal component analysis (PCA) of such vectors $\{f(w^1), \ldots, f(w^K)\}$ shown in Fig. 2. See Appendix C for more details.

This experiment demonstrates that we can instantiate reference priors for deep networks in a scalable fashion even for a large number of particles K. It provides a visual understanding of how atoms of the prior are diverse models in prediction space, just like the atoms in Fig. 1.

How to choose the number of atoms K in the reference prior? Each particle in this paper is a deep network, so must be careful to ensure that we do not maintain an unduly large number of atoms in the prior. Abbott and Machta (2019) suggest a scaling law for K in terms of the number of samples n, e.g., $K \sim n^{4/3}$ for a problem with two biased coins. We will instead treat K as a hyper-parameter. This choice is motivated from the emergent low-dimensional structure of the green particles in Fig. 2; see the further analysis in in §4.4.

Remark 3 (Variational approximations of reference priors). Nalisnick and Smyth (2017) maximize a lower bound on $I_\pi(w; z)$ and replace the term $p(z) = \int \mathrm{d}w\, \pi(w) p(z \mid w)$ in (1) by the so-called VR-max estimator $\max_w \log p(z \mid w)$ where the maximum is evaluated across a set of samples from $\pi(w)$ (Li and Turner, 2016). They use a continuous variational family parameterized by neural networks. However, reference priors are supported on a discrete set. Using a continuous variational family, e.g., a Gaussian distribution, to approximate π^*_n is computationally beneficial but it is detrimental to the primary purpose of the prior, namely to discover diverse models. This is also seen in Fig. 2 where it would be difficult to construct a variational family whose distributions put mass mostly on the green points. We therefore do not use variational approximations.

Remark 4 (Reference prior depends upon the number of samples and its atoms are diverse models). (1) encourages the likelihood $p(z^n \mid w)$ of atoms in the reference prior to be maximally different from that of other atoms. This gives us intuition as to why the prior should have finite atoms. Consider the covering number in learning theory (Bousquet et al., 2003) where we endow the model space with a metric that measures disagreement between two hypotheses over n samples. Smaller the number of samples n, smaller the covering number, and smaller the effective set of models considered. The reference prior is similar. If we only have few samples n, then it is not possible for the likelihood in Bayes law to distinguish between a large set of models and assign them different posterior probabilities. The prior therefore puts probability mass only on a finite set of atoms, and just like the coin-tossing experiment in Example 1, these atoms have diverse outputs on the n samples. This ability of the prior to select a small set of representative models is extremely useful for training deep networks with few data and it was our primary motivation.

3.3. Reference priors for semi-supervised learning

Consider the situation where we are **given inputs x^n, their corresponding labels y^n and unlabeled inputs x^u**. Our goal is semi-supervised learning, i.e., to use x^u to build a prior $\pi^*(w)$ that selects models that can be learned using the labeled data (x^n, y^n). Recall that since π^* is a prior, it should not depend on (x^n, y^n). Just like the construction of the reference prior in §2.2, we can maximize

$$\begin{aligned} I_\pi(y^n, x^n; w) &= \mathop{\mathbb{E}}_{x^n, (y^n \mid x^n, w), w \sim \pi} \left[\log \frac{p(y^n \mid x^n, w)}{p_\pi(y^n \mid x^n)} \right] \\ &= \mathop{\mathbb{E}}_{x^n, (y^n \mid x^n, w), w \sim \pi} \left[\log p(y^n \mid x^n, w) \right] \\ &\quad - \mathop{\mathbb{E}}_{x^n, y^n \mid x^n} \left[\log p_\pi(y^n \mid x^n) \right] \\ &= \mathop{\mathbb{E}}_{x^u} \left[H(y^u \mid x^u) \right] - \mathop{\mathbb{E}}_{x^u, w \sim \pi} \left[H(y^u \mid x^u, w) \right], \end{aligned} \tag{3}$$

where $p_\pi(y^n \mid x^n) = \int \mathrm{d}w\, \pi(w) \prod_{i=1}^n p(y_i \mid x_i, w)$ and likewise for $p_\pi(y^u \mid x^u)$. The first step is simply the definition of I_π: it is the KL-divergence of the posterior after seeing (x^n, y^n) with respect to the prior $\pi(w)$. The second step is the key idea and its rationale is as follows. If we know that inputs x^u and x^n come from the same task, then we can use samples x^u to compute the expectation over x^n. For the same reason, we can average over outputs y^u which are predicted by the network in place of the fixed labels y^n. Let us emphasize that both x^u and y^u are averaged out in the objective above. Predictions on new samples x are made using the Bayesian posterior predictive distribution

$$p(y \mid x, x^n, y^n) \propto \int \mathrm{d}w\, \pi^*_n(w) p(y \mid x, w) p(y^n \mid x^n, w). \tag{4}$$

An intuitive understanding of (3) Assume for now that we know the number of classes C (although the objective is valid even if that is not the case). If our prior has K particles, then the second term is the average of the per-particle entropy of the predictions. The objective encourages each particle w_i to predict confidently, i.e., to have a small entropy in its output distribution $p_{w_i}(y \mid x)$. The first term is the entropy of the average predictions: $p_\pi(y^n \mid x^n)$, and it is large if particles predict different outputs y^n for the same inputs x^n, i.e., they disagree with each other. We treat the constant α (which should be 1 in the definition of mutual information) as a hyper-parameter to allow control over this phenomenon. **The reference prior semi-supervised learning objective encourages particles to be dissimilar but confident models (not necessarily correct).**

3.4. Reference priors for a two-stage experiment

We first develop the idea using generic random variables z^n. Consider a situation when we **see data in two stages, first z^m, and then z^n**. How should we select a prior, and thereby the posterior of the first stage, such that the posterior of the second stage makes maximal use of the new n samples? We can extend the idea in §3.3 in a natural way to address this question. We can **maximize the KL-divergence between the posterior of the second stage and the posterior after the first stage, on average, over samples z^n.**

Since we have access to samples z^m, we need not average over them, we can compute the posterior $p(w \mid z^m)$ from these samples given the prior $\pi(w)$. First, notice that $p(w, z^n \mid z^m) = p(w \mid z^{m+n})p(z^n \mid z^m) = p(z^n \mid w)p(w \mid z^m)$. We can now write

$$\begin{aligned}\pi^*_{n \mid m} &= \operatorname*{argmax}_{\pi} I_{p(w \mid z^m)}(w; z^n) \\ &:= \int \mathrm{d}z^n\, p(z^n \mid z^m)\, \mathrm{KL}(p(w \mid z^{m+n}), p(w \mid z^m)) \\ &= \int \mathrm{d}w\, p(w \mid z^m) \int \mathrm{d}z^n\, p(z^n \mid w) \log \frac{p(z^n \mid w)}{p(z^n \mid z^m)},\end{aligned} \tag{5}$$

where $p(w \mid z^m) \propto p(z^m \mid w)\pi(w)$ and $p(z^n \mid z^m) = \int \mathrm{d}w\, p(z^n \mid w)p(w \mid z^m)$. The key observation is that if the reference prior (2) has a unique solution, we should have that the optimal $p(w \mid z^m) \equiv \pi^*_n(w)$. This leads to

$$\pi^*_{n \mid m}(w) \propto \pi^*_n(w)\, p(z^m \mid w)^{-1}. \tag{6}$$

This prior puts *less* probability on regions which have high likelihood on old data z^m whereby the posterior is maximally informed by the new samples z^n. Given knowledge of old data, the prior *downweighs regions* in the weight space that could bias the posterior of the new data. We also have $\pi^*_{n \mid m} = \pi^*_n$ for $m = 0$ which is consistent with (2). As $m \to \infty$, this prior ignores the part of the weight space that was ideal for z^m. See Appendix D.3 for an example.

Remark 5 (Averaging over z^m in the two-stage experiment). If we do not know the outcomes z^m yet, the prior should be calculated by averaging over both z^m, z^n

$$\begin{aligned}\pi^* &= \operatorname*{argmax}_{\pi} \int \mathrm{d}z^m\, p(z^m) I_{p(w \mid z^m)}(w; z^n) \\ &:= I_\pi(w; z^{m+n}) - I_\pi(w; z^m) \\ &= H(w \mid z^m) - H(w \mid z^{m+n}).\end{aligned} \tag{7}$$

The encourages multiple explanations of initial data z^m, i.e., high $H(w \mid z^m)$, so as to let the future samples z^n select the best one among these explanations, i.e., reduce the entropy $H(w \mid z^{m+n})$. It is interesting to note that neither is this two-stage prior equivalent to maximizing $I_\pi(w; z^{m+n})$, nor is it simply the optimal prior corresponding to objectives $I_\pi(w; z^m)$ or $I_\pi(w; z^n)$. Both (6) and (7) therefore indicate that two-stage priors are useful when we have *some* data *a priori*, this can be either unlabeled samples from the same task, or labeled samples from some other task.

Remark 6 (A softer version of the two-stage reference prior). The objective in (7) resembles the predictive information bottleneck (IB) of Bialek et al. (2001), or its variational version in Alemi (2020), which seek to learn a representation, say w, that maximally forgets past data while remaining predictive of future data

$$\max_{p(w \mid z^m)} I(w; z^n) - \beta I(w; z^m). \tag{8}$$

The parameter β in (8) gives this objective control over how much information from the past is retained in w. We take inspiration from this and construct a variant of (6)

$$\begin{aligned}&\pi^\beta_{n \mid m}(w) \propto \pi^*_n(w) p(z^m \mid w)^{-\beta} \quad \text{for } \beta \in (0,1). \\ \Rightarrow\ & p(w \mid z^{m+n}) \propto p(z^n \mid w) p(z^m \mid w)^{1-\beta} \pi^*_n(w).\end{aligned} \tag{9}$$

We should use $\beta = 0$ when we expect that data from the first stage z^m is similar to data z^n from the second stage. This allows the posterior to *benefit* from past samples. If we expect that the data are different, then $\beta = 1$ ignores regions in the weight space that predict well for z^m. This is similar to the predictive IB where a small β encourages remembering the past and $\beta = 1$ encourages forgetting.

3.5. Reference priors for transfer learning

Consider the two-stage experiment where in the first stage we obtain m samples (x^m_s, y^m_s) from a "source" task P^s and the second stage consists of n samples (x^n_t, y^n_t) from the "target" task P^t. Our goal is to calculate a prior $\pi(w)$ that best utilizes the target task data.

Bayesian inference for this problem involves first computing the posterior $p(w \mid x^m_s, y^m_s) \propto p(y^m_s \mid w, x^m_s)\pi(w)$ from the source task and then using it as a prior to compute the posterior for the target task $p(w \mid x^n_t, y^n_t, x^m_s, y^m_s)$. Just like §2.2, **the key idea again is to maximize the KL-divergence between the two posteriors** $\mathrm{KL}\left(p(w \mid x^n_t, y^n_t, x^m_s, y^m_s),\ p(w \mid x^m_s, y^m_s)\right)$, but averaged over samples x^m_s and x^n_t.

Case 1: Access to unlabeled data from the source x^m_s and the target task x^n_t We should average the KL-divergence over both the source and target predictions y^m_s and y^n_t and maximize

$$\mathop{\mathbb{E}}_{x^m_s, x^n_t, y^m_s \mid x^m_s, y^n_t \mid x^n_t} \mathrm{KL}\left(p(w \mid x^n_t, y^n_t, x^m_s, y^m_s), p(w \mid x^m_s, y^m_s)\right) \tag{10}$$

over the prior π. Here $p_\pi(y^m_s \mid x^m_s) = \mathbb{E}_{w \sim \pi}\, p(y^m_s \mid x^m_s, w)$ and $p_\pi(y^n_t \mid x^n_t) = \mathbb{E}_{w \sim \pi}\, p(y^n_t \mid x^n_t, w)$, respectively. Note that averages over x^m_s and x^n_t are computed using samples while averages over $y^m_s \mid x^m_s$ and $y^n_t \mid x^n_t$ are computed using the model's predictions.

Case 2: x^m_s, y^m_s are fixed and known, and we have a pool of unlabeled target data x^n_t Since we already know the labels for the source task, we will only average over x^n_t and

y_t^n and maximize

$$\mathop{\mathbb{E}}_{x_t^n, y_t^n \mid x_t^n} \mathrm{KL}\left(p(w \mid x_t^n, y_t^n, x_s^m, y_s^m), p(w \mid x_s^m, y_s^m)\right); \tag{11}$$

here $p_\pi(y_t^n \mid x_t^n) = \int \mathrm{d}w\, \pi(w) p(y_t^n \mid x_t^n, w)$.

Remark 7 (Connecting (10) and (11) to practice). Both objectives can be written down as

$$\pi^* = \operatorname*{argmax}_\pi I_\pi(w; y_t^n, x_t^n, x_s^m, y_s^m) - I_\pi(w; x_s^m, y_s^m) \tag{12}$$

with the distinction that while in Case 1, we average over all quantities, namely $p(x_s^m), p(y_s^m), p(x_t^n), p(y_t^n)$ while in Case 2, we fix x_s^m and y_s^m to the provided data from the source task. Case 2 is what is typically called transfer learning. Case 1, where one has access to *only unlabeled data* from a source task *that is different from the target task* is not typically studied in practice. Like (9), we can again introduce a coefficient β on the second term in (12) to handle the relatedness between source and target tasks.

3.6. Practical tricks for implementing reference priors

The reference prior objective is conceptually simple but it is difficult to implement it directly using deep networks and modern datasets. We next discuss some practical tricks that we have developed.

(1) Order of the reference prior n versus the number of samples Bernardo (1979) set the order of the prior n to be the same as the number of samples. We observe that both do not have to be identical and make a distinction between the two. In our expierments, we restrict the order to $n = 2, 3$. Mathematically, this amounts to computing averages in (2) or (3) over only sets of n-tuples. This significantly reduces the class of models considered in the reference prior by *pretending* that there is a small number of samples available for training the task—which is useful, and also true in practice, for over-parametrized deep networks. This choice is also motivated by the low-dimensional structure in the reference prior in Fig. 2. Note that we are *not* restricting to small order n for computational reasons, i.e., computing the expectation over all classes y^n in (3) can be done in a single forward pass.

(2) Using cross-entropy loss to bias particles towards good parts of the weight space The posterior (4) suggests that we should first compute the prior, and then weight each particle by the likelihood of the labeled data. In practice, we combine these two steps into a single objective

$$\max_\pi \gamma I_\pi(w; y^u, x^u) + \mathop{\mathbb{E}}_{w \sim \pi} \left[\log p(y^n \mid x^n, w)\right], \tag{13}$$

where γ is a hyper parameter, x^n, y^n are labeled samples. (13) allows us to directly obtain particles that both have high probability under the prior and a high likelihood. This is different from the correct Bayesian posterior (which would set $\gamma = 1$, we use $\gamma = 1/2$) but it is a trick often employed in the SSL literature. The second term restricts the search space for the particles in $\pi(w)$.

(3) Data augmentation State of the art SSL methods use heavy data augmentation, e.g., RandAugment (Cubuk et al., 2020) and CTAugment (Berthelot et al., 2019a), which both have about 20 transformations. Some are weak augmentations such as mirror flips and crops while some others are strong augmentations such as color jitter. Methods such as FixMatch (Sohn et al., 2020) or MixMatch (Berthelot et al., 2019b) use weak augmentations to get soft labels for predictions on strong augmentations.

We compute the entropy term $H(y^u \mid x^u, w)$ in (3) using the distribution $p_G(y \mid x, w) = \mathbb{E}_{g \sim G}[p_w(y \mid g(x), w)]$ where $G = G_1 \cup G_2$ is the set of weak (G_1) and strong (G_2) augmentations. Let $g_i \sim G_i$ be an augmentation and denote $p_{g_i} \equiv p_w(y \mid g_i(x), w)$ for $i \in \{1, 2\}$. In every mini-batch we use $p_G(y \mid x, w) \approx \tau p_{g_1} + (1 - \tau) p_{g_2}$ where τ is a hyper-parameter. This gives accuracy that is reasonable (about 87% for 500 samples) but a bit lower than state of the art SSL methods. We noticed that if we use an upper bound on the entropy from Jensen's inequality

$$-\mathop{\mathbb{E}}_{x^u} \int \mathrm{d}y^u\, p_G(y^u \mid x^u, w) \left[\tau \log p_{g_1} + (1 - \tau) \log p_{g_2}\right] \tag{14}$$

then we can close this gap in accuracy (see Table 1). This is perhaps because the cross-entropy terms, e.g., $-p_{g_1} \log p_{g_2}$, force the predictions of the particles to be consistent across both types of augmentations, just like the objective in FixMatch or MixMatch. Our formulation is thus useful to not only understand SSL but also to tweak it to perform as well as current methods and thereby shed light on the theoretical underpinnings of their performance.

(4) Computing $H(y^u \mid x^u, w)$ A number of SSL methods work by creating pseudolabels from weakly augmented data, which seems to be a key ingredient of good accuracy in our experience with these methods. We tried two heuristics to compute the entropy term $H(y^u \mid x^u, w)$ that are motivated by these papers. First, we follow FixMatch and only use unlabeled data with confident predictions to compute $H(y^u \mid x^u, w)$. A datum x contributes to the objective only if $\max_y p_w(y|g_1(x), w) > 0.95$. Changing this threshold does not lead to deterioration of the accuracy as we see in Table S-6, so this heuristic need not be used while building the reference prior. Second, if G_1 is the set of weak augmentations (see previous point), methods like FixMatch and MixMatch use $\operatorname{argmax}_y p(y \mid g_1(x), w)$ as a pseudo-label but do not update this using the back-propagation gradient. This prevents the more reliable predictions on G_1 from changing. As a result, the entropy term $-\tau^2 p_{g_1} \log p_{g_1}$ is a constant in (14). To normalize the terms coming from τ in (14), we set γ in (13) to $1/(1-\tau^2)$ instead of 1. We have also developed an argument to choose the appropriate value of $\tau = 1/3$ that we explain in Appendix A. This second heuristic seems essential, in Table S-6, we obtain only 10% accuracy without this heuristic.

4. Empirical Study

4.1. Setup

We evaluate on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009). For SSL, we use 50–1000 labeled samples, i.e., 5–100 samples/class and use the rest of the samples in the training set as unlabeled samples. For transfer learning, we construct 20 five-way classification tasks from CIFAR-100 and use 1000 labeled samples from the source and 100 labeled samples from the target task. All experiments use the WRN 28-2 architecture (Zagoruyko and Komodakis, 2016), same as in Berthelot et al. (2019b).

For all our experiments, the reference prior is of order $n = 2$ and has $K = 4$ particles. We run all our methods for 200 epochs, with $\tau = 1/3$ in (14) and $\alpha = 0.1$ in (3). We set $\gamma = (1-\tau^2)^{-1}$ as discussed in §3.6. For inference, each particle maintains an exponential moving average (EMA) of the weights (this is common in SSL (Tarvainen and Valpola, 2017)). Appendix A provides more details.

4.2. Semi-supervised learning

Baselines We compare to a number of recent methods such as FixMatch (Sohn et al., 2020), MixMatch (Berthelot et al., 2019b), DASH (Xu et al., 2021), SelfMatch (Kim et al., 2021), Mean Teacher (Tarvainen and Valpola, 2017), Virtual Adversarial Training (Miyato et al., 2018), and Mixup (Berthelot et al., 2019b).

Table 1 compares the accuracy of different SSL methods on CIFAR-10. We find that the reference prior approach is competitive with a number of existing methods, e.g., it is remarkably close to FixMatch on all sample sizes (notice the error bars). There is a gap in accuracy at small sample sizes (40–50) when compared to recent methods. It is important to note that these recent methods employ a number of additional tricks, e.g., FlexMatch implements curriculum learning on top of FixMatch, DASH and FlexMatch use different thresholding for weak augmentations (this increases their accuracy by 2-5%), SelfMatch has higher accuracies because of a self-supervised pretraining stage, FixMatch (CTA) outperforms its RA variant by 1.5% which indicates CTA augmentation is beneficial (we used RA). It is also extremely expensive to train SSL algorithms for 1000 epochs (all methods in Table 1 do so), we trained for 200 epochs.

Method	Samples				
	50	100	250	500	1000
Mixup	-	-	52.57	63.86	74.28
VAT	-	-	63.97	73.89	81.32
Mean Teacher	-	-	52.68	57.99	82.68
MixMatch	64.21*	80.29*	88.91*	90.35*	92.25*
FixMatch (RA)	86.19 $\pm$ 3.37 (40)	90.12*	94.93 $\pm$ 0.65	93.91*	94.3*
FixMatch (CTA)	88.61 $\pm$ 3.35 (40)	-	94.93 $\pm$ 0.33	-	-
DASH (RA)	86.78 $\pm$ 3.75 (40)	-	95.44 $\pm$ 0.13	-	-
DASH (CTA)	90.84 $\pm$ 4.31 (40)	-	95.22 $\pm$ 0.12	-	-
SelfMatch	93.19 $\pm$ 1.08 (40)	-	95.13 $\pm$ 0.26	-	-
FlexMatch	95.03 $\pm$ 0.06 (40)	-	95.02 $\pm$ 0.09	-	-
Deep Reference Prior	85.45 $\pm$ 2.12	88.53 $\pm$ 0.67	92.13 $\pm$ 0.39	92.94 $\pm$ 0.22	93.48 $\pm$ 0.24

Table 1. **Classification accuracy of different semi-supervised learning methods on CIFAR-10. Note:** RA and CTA in the methods column indicate that RandAugment or CTAugment were used for augmentations. Entries with * were evaluated by us using open-source implementations from the original authors for 256 epochs. All other entries are from original papers. Entries with "(40)" indicate that 40 labeled samples were used instead of 50.

This experiment shows that our approach to SSL can obtain results that are competitive to sophisticated empirical methods without being explicitly formulated to enforce properties like label consistency with respect to augmentations. This also indicates that reference priors could be a good way to explain the performance of these existing methods, which is one of our goals in this paper.

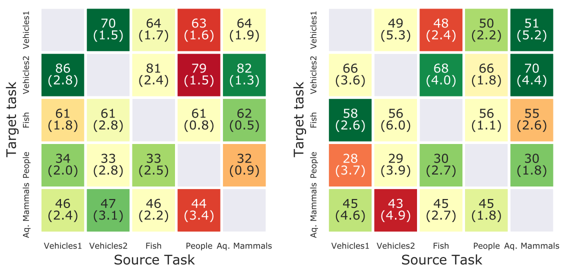
4.3. Transfer learning

Just like we did in §3.6 for SSL, we instantiate (9) and (11), by combining prior selection, pretraining on the source task and likelihood of the target task, into one objective,

$$\gamma I_\pi(w; y_t^u, x_t^u) + \mathop{\mathbb{E}}_{w\sim\pi}\left[\log p(w, y_t^n \mid x_t^n)\right] \\ + (1-\beta) \mathop{\mathbb{E}}_{w\sim\pi}\left[\log p(w, y_s^m \mid x_s^m)\right], \tag{15}$$

where $\gamma = 1/2$ and $\beta = 1/2$ are hyper-parameters, (x_s^m, y_s^m) are labeled data from the source task ($m = 1000$), (x_t^n, y_t^n) are labeled data from the target task ($n = 100$) and x_t^u are unlabeled samples from the target task (all other samples).

Baselines We use three methods: (a) fine-tuning, which is a very effective strategy for transfer learning (Dhillon et al., 2020; Kolesnikov et al., 2020) but it cannot use unlabeled target data, (b) using only labeled target data (this is standard supervised learning), and (c) using only labeled and unlabeled target data without any source data (this is simply SSL, or $\beta = 1$ in (15)). Fig. 3 compares the performance for pairwise transfer across 5 tasks from CIFAR-100. Our reference prior objective in (15) obtains much better accuracy than fine-tuning which indicates that it leverages the unlabeled target data effectively. For each task, the accuracy is much better than both standard supervised learning and semi-supervised learning using our own reference prior approach (13); both of these indicate that the labeled source data is being used effectively in (15).



Method	Task (→)	Vehicles-1	Vehicles-2	Fish	People	Aq. Mammals
Supervised Learning		42.2	63.2	56.8	31.0	42.6
Deep Reference Prior (SSL)		63.6	75.2	54.6	34.0	47.4

Figure 3. **Top: Accuracy (%) of deep reference priors (left) and fine-tuning (right) for transfer learning tasks in CIFAR-100**. Cells are colored red/green relative to the median accuracy of each row. Darker shades of green indicate that the source task is more suitable for transfer. For example, Vehicles-1 as source is the best for all tasks according to the reference prior (left) (which is optimal in theory) but fine-tuning cannot replicate this. The accuracy of cells in the left panel is better than the corresponding cells on the right, e.g., the gap in accuracy is 34.8% for Vehicles 2 → Vehicles 1. **Bottom: Accuracy (%) of supervised learning and SSL for all 5 tasks**. Each number here should be compared to the corresponding row of the matrices in the top panel, e.g., Vehicles 2 has 86% accuracy when transferred from Vehicles 1 using our transfer method (left), it has 66% accuracy from fine-tuning (right), while the same task achieves 63.2% accuracy when trained by itself using supervised learning (table first row) and 75.2% accuracy when trained using unlabeled target data (table second row). Therefore the reference prior-based transfer objective can leverage both labeled source data as well as unlabeled target data. This pattern is consistent for all tasks.

4.4. Ablation and analysis

This section presents ablation and analysis experiments for SSL on CIFAR-10 with 1000 labeled samples. We study the reference prior for different settings (i) varying the order n of the prior, (ii) varying the number of particles in the BA algorithm (K), (iii) exponential moving averaging of the weights for each particle. We also study the two entropy terms in the reference prior objective individually.

We use a reference prior of order $n = 2$ in all our experiments. We see in Table 2 that **changing the order of the prior** leads to marginal (about 1%) changes in the accuracy.

We next **vary the number of particles** in the prior in Table 3 and find that the accuracy is relatively consistent when the number of particles varies from $K = 2$ to $K = 16$. This seems surprising because a reference prior ideally should have an infinite number of atoms, when it approximates Jeffreys prior. We should not a priori expect $K = 2$ particles to be sufficient to span the prediction space of deep networks. But our experiment in Fig. 2 provides insight into this phenomenon. It shows that the manifold of diverse predictions is low-dimensional. Particles of the reference prior only need to span these few dimension and we can fruitfully implement our approach using very few particles.

Method	Order (→)	2	3	4	5
Deep Reference Prior ($K = 4$)		91.76	90.53	91.51	91.36

Table 2. The order of the reference prior has a minimal impact on the accuracy.

Method	#Particles (→)	2	4	8	16
Deep Reference Prior ($n = 2$)		91.3	91.76	89.79	90.72

Table 3. Number of particles has a minimal impact on accuracy.

Effect of exponential moving averaging (EMA) We use EMA on the weights of each particle (independently). Table 4 analyzes the impact of EMA. As noticed in other semi-supervised learning works (Berthelot et al., 2019b; Sohn et al., 2020), EMA improves the accuracy by 2-3% regardless of the number of labeled samples used.

Method	#Samples (→)	50	100	250	500	1000
EMA		85.45 ± 2.12	88.53 ± 0.67	92.13 ± 0.39	92.94 ± 0.22	93.48 ± 0.24
No EMA		82.36 ± 2.13	85.64 ± 0.43	89.75 ± 0.36	90.06 ± 1.71	91.57 ± 0.25

Table 4. Using EMA for weights of each particle is beneficial and improves accuracy by 2-3%.

The two entropy terms in the reference prior objective Fig. 4 (left) shows how, because of the entropy term $H(y^u \mid x^u)$, the accuracy of particles is quite different during training. Particles have different predictive abilities (7% range in test error) but the Bayesian posterior predictive distribution has a higher accuracy than any of them. Fig. 4 (right) tracks the two entropy terms in the objective. For large number of labeled data (500, blue) the entropy $H(y^u \mid x^u)$ which should always be higher than $H(y^u \mid x^u, w)$ in (3) is lower (this is not the case for 50 samples, red). This is likely a result of the cross-entropy term in the modified objective in (13) which narrows the search space of the particles. This experiment is an important insight into the working of existing semi-supervised learning methods as well, all of which also have a similar cross-entropy objective in their formulation. It points to the fact that at large sample-sizes, the cross-entropy loss and not the semi-supervised learning objective could dominate the training procedure.

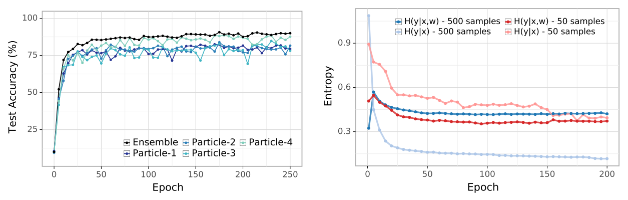


Figure 4. **(Left)** Accuracy of individual particles in the prior during training (250 labeled samples). The individual particles have diverse predictions due to the entropy term $H(y^n \mid x^n)$, the accuracy of the ensemble is larger than the accuracy of any single particle. **(Right)** Evolution of entropy terms $H(y^u \mid x^u, w)$ and $H(y^u \mid x^u)$ for two cases (500 labeled samples and 50 labeled samples). While $H(y^u \mid x^u)$ is expected to be larger than $H(y^u \mid x^u, w)$ in (3) since KL-divergence is non-negative, this is not always the case since we approximate $H(y^u \mid x^u, w)$ by an upper-bound obtained from Jensen's inequality for data augmentation as discussed in §3.6.

5. Related Work and Discussion

Reference priors in Bayesian statistics We build upon the theory of reference priors which was developed in the objective Bayesian statistics literature Bernardo (1979); Berger et al. (1988; 2009). The main idea used in our work is that non-asymptotic reference priors allow us to exploit the finite samples from the task in a fundamentally different way than classical Bayesian inference. If the number of samples from the task available to the learner is finite, then the prior should also select only a finite number of models. Reference priors are not common in the machine learning literature. A notable exception is Nalisnick and Smyth (2017) who optimize a variational lower bound and demonstrate results on small-scale problems. The main technical distinction of our work is that we explicitly use the discrete prior instead of a variational approximation.

Information theory Discreteness is seen in many problems with an information-theoretic formulation, e.g., capacity of a Gaussian channel under an amplitude constraint (Smith, 1971), neural representations in the brain Laughlin (1981), and biological systems (Mayer et al., 2015). (Mattingly et al., 2018; Abbott and Machta, 2019) have developed these ideas to study how reference priors select "simple models" which lie on certain low-dimensional "edges" of the model space. We believe that the methods developed in our paper are effective *because* of this phenomenon. Our choice of using a small order n for the prior is directly motivated by their examples.

Semi-supervised learning Our formulation sheds light on the working of current SSL methods. For example, the reference prior can automatically enforce consistency regularization of predictions across augmentations (Tarvainen and Valpola, 2017; Berthelot et al., 2019b), as we discuss in §3.6. Similarly, minimizing the entropy of predictions on unlabeled data, either explicitly (Grandvalet et al., 2005; Miyato et al., 2018) or using pseudo-labeling methods (Lee et al., 2013; Sajjadi et al., 2016), is another popular technique. This is automatically achieved by the objective in (3). Disagreement-based methods (Zhou and Li, 2010) employ multiple models and use confident models to soft-annotate unlabeled samples for others. Disagreements in our formulation are encouraged by the entropy $H(y^n \mid x^n)$ in (3). If $p(y^n \mid x^n)$ is uniform, which is encouraged by the reference prior objective, particles disagree strongly with each other.

Transfer learning is a key component of a large number of applications today, e.g, (Devlin et al., 2019; Kolesnikov et al., 2020) but a central question that remains unanswered is how one should pretrain a model if the eventual goal is to transfer to a target task. There have been some attempts at addressing this via the Information Bottleneck, e.g., Gao and Chaudhari (2020). This question becomes particularly challenging when transferring across domains, or for small sample sizes (Davatzikos, 2019). Reference priors are uniquely suited to tackle this question: our two-stage experiment in §3.4 is the *optimal* way pretain on the source task. As our experiments show, this is better than fine-tuning in the low-sample regime §4.3.

6. Acknowledgments

This work was supported by grants from the National Science Foundation (2145164) and the Office of Naval Research (N00014-22-1-2255), and cloud computing credits from Amazon Web Services.

References

Michael C. Abbott and Benjamin B. Machta. A Scaling Law From Discrete to Continuous Solutions of Channel Capacity Problems in the Low-Noise Limit. *Journal of Statistical Physics*, 176(1):214–227, July 2019. ISSN 1572-9613. doi: 10.1007/s10955-019-02296-2.

Alexander A Alemi. Variational predictive information bottleneck. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–6. PMLR, 2020.

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

James O Berger, Jos M Bernardo, and Manuel Mendoza. *On Priors That Maximize Expected Information*. Purdue University. Department of Statistics, 1988.

James O Berger, José M Bernardo, and Dongchu Sun. The formal definition of reference priors. *The Annals of Statistics*, 37(2):905–938, 2009.

Jose M Bernardo. Reference posterior distributions for Bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):113–128, 1979.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019a.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. MixMatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 32, 2019b.

William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

Richard Blahut. Computation of channel capacity and rate-distortion functions. *IEEE transactions on Information Theory*, 18(4):460–473, 1972.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207. Springer, 2003.

Bertrand S Clarke and Andrew R Barron. Jeffreys' prior is asymptotically least favorable under entropy risk. *Journal of Statistical planning and Inference*, 41(1):37–60, 1994.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

Christos Davatzikos. Machine learning in neuroimaging: Progress and challenges. *NeuroImage*, 197:652, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT*, 2019.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2020.

Yansong Gao and Pratik Chaudhari. A free-energy principle for representation learning. In *Proc. of International Conference of Machine Learning (ICML)*, 2020.

Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. *CAP*, 367:281–296, 2005.

Byoungjip Kim, Jinho Choo, Yeong-Dae Kwon, Seongho Joe, Seungjai Min, and Youngjune Gwon. Selfmatch: Combining contrastive self-supervision and consistency for semi-supervised learning. *arXiv preprint arXiv:2101.06480*, 2021.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning. *arXiv:1912.11370 [cs]*, May 2020.

A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. PhD thesis, Computer Science, University of Toronto, 2009.

Simon Laughlin. A simple coding procedure enhances a neuron's information capacity. *Zeitschrift für Naturforschung c*, 36(9-10): 910–912, 1981.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

Yingzhen Li and Richard E. Turner. R\'enyi Divergence Variational Inference. *arXiv:1602.02311 [cs, stat]*, October 2016.

Henry H Mattingly, Mark K Transtrum, Michael C Abbott, and Benjamin B Machta. Maximizing the information learned from finite data selects a simple model. *Proceedings of the National Academy of Sciences*, 115(8):1760–1765, 2018.

Andreas Mayer, Vijay Balasubramanian, Thierry Mora, and Aleksandra M Walczak. How a well-adapted immune system is organized. *Proceedings of the National Academy of Sciences*, 112(19):5950–5955, 2015.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Eric Nalisnick and Padhraic Smyth. Variational reference priors. 2017.

Katherine N. Quinn, Colin B. Clement, Francesco De Bernardis, Michael D. Niemack, and James P. Sethna. Visualizing probabilistic models and data with intensive principal component analysis. *Proceedings of the National Academy of Sciences*, 116 (28):13762–13767, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1817218116. URL https://www.pnas.org/content/116/28/13762.

Rahul Ramesh and Pratik Chaudhari. Model Zoo: A Growing "Brain" That Learns Continually. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2022.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1908–1912. IEEE, 2016.

Joel G Smith. The information capacity of amplitude-and variance-constrained sclar Gaussian channels. *Information and control*, 18(3):203–219, 1971.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. FixMatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33, 2020.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proc. of the 37-Th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Zhongxin Zhang. *Discrete noninformative priors*. PhD thesis, Yale University, 1994.

Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439,

2010.

A. Details of the experimental setup

Architecture For experiments on CIFAR-10 and CIFAR-100 (§4), we consider a modified version of the Wide-Resnet 28-2 architecture (Zagoruyko and Komodakis, 2016), which is identical to the one used in Berthelot et al. (2019b). This architecture differs from the standard Wide-Resnet architecture in a few important aspects. The modified architecture has Leaky-ReLU with slope 0.1 (as opposed to ReLU), no activations or batch normalization before any layer with a residual connection, and a momentum of 0.001 for batch-normalization running mean and standard-deviation (as opposed to 0.1, in other words these statistics are made to change very slowly). We observed that the change to batch-normalization momentum has a very large effect on the accuracy of semi-supervised learning.

For experiments on MNIST (Appendix D.1), we use a fully-connected network with 1 hidden layer of size 32. We use the hardtanh activation in place of ReLU for this experiment; this is because maximizing the mutual information has the effect of increasing the magnitude of the activations for ReLU networks. One may use weight decay to control the scale of the weights and thereby that of the activations but in an effort to implement the reference prior exactly, we did not use weight decay in this model. Note that the nonlinearities for the CIFAR models are ReLUs.

Datasets For semi-supervised learning, we consider the CIFAR-10 dataset with the number of labeled samples varying from 50–1000 (i.e., 5–100 labeled samples per class). Semi-supervised learning experiments use all samples that are not a part in the labeled set, as unlabeled samples.

For transfer learning, we construct two tasks from MNIST (task one is a 5-way classification task for digits 0–4, and task two is another 5-way classification task for digits 5–9). For this experiment, we use labeled source data but do not use any labeled target data. This makes our approach using a reference prior similar to a purely unsupervised method.

The CIFAR-100 dataset is also utilized in the transfer learning setup (§4.3). We consider five 5-way classification tasks from CIFAR-100 constructed using the super-classes. The five tasks considered are Vehicles-1, Vehicles-2, Fish, People and Aquatic Mammals. The selection of these tasks were motivated from the fact that some pairs of tasks are known to positively impact each other (Vehicles-1, Vehicles-2), while other pairs are known to be detrimental to each other (Vehicles-2, People); see the experiments in Ramesh and Chaudhari (2022).

Optimization SGD with Nesterov momentum on a Cosine-annealed learning rate schedule with warmup was used in our experiments on CIFAR-10 and CIFAR-100. The initial learning rate was set to $0.03 \times K$ where K denotes the number of particles. The scaling factor of K exists to counteract the normalization constant in the objective from averaging across all particles. The momentum coefficient for SGD was set to 0.9 and weight decay to $5K^{-1} \times 10^{-4}$. Mixed-precision (32-bit weights, 16-bit gradients) was used to expedite training. Training was performed for 200 epochs unless specified otherwise.

Experiments on MNIST also used SGD for computing the reference prior. SGD was used with a constant learning rate of 0.001 with Nesterov's acceleration, momentum coefficient of 0.9 and weight decay of 10^{-5}.

Definition of a single Epoch Note that since we iterate over the unlabeled and labeled data (each with different number of samples), the notion of what is an epoch needs to be defined differently. In our work, one epoch refers to 1024 weight updates, where each weight update is calculated using a batch-size of 64 for the labeled data of batch size 64, and a batch-size of 448 for the unlabeled data.

Exponential Moving Average (EMA) In all CIFAR-10 and CIFAR-100 experiments, we also implement the Exponential Moving Average (EMA) (Tarvainen and Valpola, 2017). In each step, the EMA model is updated such that the new weights are the weighted average of the old EMA model weights, and the latest trained model weights. The weights for averaging used in our work (and most other methods) are 0.999 and 0.001 respectively. Note that EMA only affects the particle when it is used for testing, it does not affect how weight updates are calculated during training. We exclude batch-normalization running mean and variance estimates in EMA.

Data Augmentations We use random-horizontal flips and random-pad-crop (padding of 4 pixels on each side) as weak augmentations for the CIFAR-10 and CIFAR-100 datasets. For SSL experiments on CIFAR-10, we use RandAugment (Cubuk et al., 2020) for strong augmentations. No data augmentations were used for MNIST.

Picking the value of τ in (14) Let G_1 and G_2 be the sets of weak and strong augmentations respectively. For $g_1 \sim G_1$ and $g_2 \sim G_2$, let us write down the upper bound in (14) from Jensen's inequality in detail

$$\mathop{\mathbb{E}}_{x^u} \int \mathrm{d}y^u \left[-\tau^2 p_{g_1} \log p_{g_1} - \tau(1-\tau) p_{g_2} \log p_{g_1} - (1-\tau)\tau p_{g_1} \log p_{g_2} - (1-\tau)^2 p_{g_2} \log p_{g_2} \right].$$

The upper bound is thus a weighted sum of the entropy terms $-p_{g_1} \log p_{g_1}, -p_{g_2} \log p_{g_2}$, and cross entropy terms $-p_{g_2} \log p_{g_1}, -p_{g_1} \log p_{g_2}$. If we were to pick $\tau = 1/2$ like FixMatch, then since $(1-\tau)^2 + \tau^2 = 2\tau(1-\tau)$ for $\tau = 1/2$, the entropy and cross entropy terms will contribute equally to the loss function. However in practice, since we do not update p_{g_1} using the back-propagation gradient to protect the predictions from deteriorating on the weakly augmented images, one of the entropy terms $-p_{g_1} \log p_{g_1}$ is dropped. In such a situation, to ensure that cross entropy and entropy terms provide an equal contribution to the gradient, we would like $(1-\tau)^2 = 2\tau(1-\tau)$ which gives $\tau = 1/3$.

B. Overview of the Implementation

We provide an overview of the implementation of deep reference priors.
For more details see https://github.com/rahul13ramesh/deep_reference_priors.

Let a mini-batch from the labeled dataset be denoted by $\{(x_i, y_i)\}_{i=1}^{b}$ and a mini-batch from the unlabeled dataset be denoted by $\{(x_{i0}^u, x_{i1}^u, \cdots, x_{in}^u))\}_{i=1}^{b_u}$ where n is the order of the reference prior. Note the distinction in the two mini-batches, i.e. the unlabeled mini-batch consists of a set of n-tuples unlike the labeled mini-batch. Let g_1 and g_2 be functions that perform weak and strong augmentations respectively. The reference prior objective is used to train K particles $\{p_{w_k}\}_{k=1}^{K}$.

For the sample x^u, we compute $p(y \mid x^u, w_k)$ as follows:

$$p(y \mid x^u, w_k) = \tau p(y \mid g_1(x^u), w_k) + (1-\tau) p(y \mid g_2(x^u), w_k)$$

The reference prior loss ,requires us to compute the terms

$$\begin{aligned}
\mathbb{E}_{w \sim \pi} \left[H(y_i^u \mid x_i^u, w) \right] &= \sum_{k=1}^{K} \pi(w_k) \sum_{y \in \mathcal{Y}^n} \left(-p(y \mid x_i^u, w_k) \log(p(y \mid x_i^u, w_k)) \right) \\
&= \sum_{k=1}^{K} \pi(w_k) \sum_{y \in \mathcal{Y}^n} \left(-\prod_{j=1}^{n} p(y \mid x_{ij}^u, w_k) \right) \log \left(\prod_{j=1}^{n} p(y \mid x_{ij}^u, w_k) \right) \\
&= \sum_{k=1}^{K} \pi(w_k) \sum_{j=1}^{n} \sum_{y \in \mathcal{Y}} -p(y \mid x_{ij}^u, w_k) \log \left(p(y \mid x_{ij}^u, w_k) \right) \\
&\leq \sum_{k=1}^{K} \pi(w_k) \sum_{j=1}^{n} \sum_{y \in \mathcal{Y}} -p(y \mid x_{ij}^u, w_k) \left[\tau \log p(y \mid g_1(x_{ij}^u), w_k) + (1-\tau) \log p(y \mid g_2(x_{ij}^u), w_k) \right],
\end{aligned}$$

and

$$\begin{aligned}
H(y_i^u \mid x_i^u) &= \sum_{y^n \in \mathcal{Y}^n} -p(y^n \mid x_i^u) \log(p(y^n \mid x_i^u)) \\
&= \sum_{y^n \in \mathcal{Y}^n} - \left(\sum_{k=1}^{K} \pi(w_k) p(y^n \mid x_i^u, w_k) \right) \log \left(\sum_{k=1}^{K} \pi(w_k) p(y^n \mid x_i^u, w_k) \right).
\end{aligned}$$

In our implementation, we set $\pi(w_k) = \frac{1}{K}$. We observed no improvement in accuracy if the elements of π were trainable weights.

C. Visualizing the reference prior

We can think of each particle w as representing a probability distribution

$$\mathbb{R}^{nC} \ni f(w) = \left(\sqrt{p_w(y=1 \mid x_1)}, \sqrt{p_w(y=2 \mid x_1)}, \ldots, \sqrt{p_w(y=C \mid x_n)} \right).$$

Input data consists of a mini-batch of labeled data $\{(x_i, y_i)\}_{i=1}^{b}$ and unlabeled data $\{x_{i0}^u, x_{i1}^u, \cdots, x_{in}^u)\}_{i=1}^{b_u}$ and a user-determined order n.
Trainable weights are the weights of the K neural networks (also called particles) $\{p_{w_k}\}_{k=1}^{K}$.
Define

$$f(x, y, w) = \tau p_w(y \,|\, g_1(x)) + (1-\tau) p_w(y \,|\, g_2(x)),$$
$$f_{\log}(x, y, w) = \tau \log p_w(y \,|\, g_1(x)) + (1-\tau) \log p_w(y \,|\, g_2(x)).$$

Compute the two entropy terms as

$$h_{yw} = -\frac{1}{b_u} \sum_{i=1}^{b_u} \sum_{k=1}^{K} \frac{1}{K} \sum_{j=1}^{n} \sum_{y \in \mathcal{Y}} f(x_{ij}^u, y, w_k) f_{\log}(x_{ij}^u, y, w_k),$$
$$h_y = -\frac{1}{b_u} \sum_{i=1}^{b_u} \sum_{y^n \in \mathcal{Y}^n} \left(\frac{1}{K} \sum_{k=1}^{K} \prod_{j=1}^{n} f(x_{ij}^u, y_j^n, w) \right) \log \left(\frac{1}{K} \sum_{k=1}^{K} \prod_{j=1}^{n} f(x_{ij}^u, y_j^n, w) \right).$$

Compute the loss ℓ as

$$\ell_u = \alpha h_y - h_{yw}$$
$$\ell_x = -\frac{1}{bK} \sum_{i=1}^{b} \sum_{k=1}^{K} \log(p_{w_k}(y_i \,|\, x_i))$$
$$\ell = \ell_x - \left(\frac{1}{1-\tau^2} \right) \ell_u.$$

Figure S-5. The pseudo-code to compute the loss of one mini-batch of data while computing the reference prior.

and use a method for visualizing such distributions developed in Quinn et al. (2019) that computes a principal component analysis (PCA) of such vectors $\{f(w^1), \ldots, f(w^K)\}$. This method computes an isometric embedding of the space of probability distributions. The rationale behind the choice of $f(w)$ is that for two weight vectors w, w', the Euclidean distance between $f(w)$ and $f(w')$ is the Hellinger divergence between the respective probability distributions,

$$\begin{aligned} \|f(w) - f(w')\|^2 &= \frac{1}{2n} \sum_{i=1}^{n} \sum_{k=1}^{C} \left(\sqrt{p_w(y=k \,|\, x_i)} - \sqrt{p_{w'}(y=k \,|\, x_i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} d_H^2 \left(p_w(\cdot \,|\, x_i), p_{w'}(\cdot \,|\, x_i) \right), \end{aligned}$$

where

$$d_H^2(P, Q) = \frac{1}{2} \int \left(\sqrt{\mathrm{d}P} - \sqrt{\mathrm{d}Q} \right)^2$$

is the Hellinger distance. In other words, the prediction vector $f(w)$ maps the weights w into a $(nC)-$dimensional space. The Euclidean metric in this space corresponds to the Hellinger distance in the space of probability distributions. We can therefore compute the principal component analysis (PCA) of these vectors and project the vectors $f(w)$ into lower-dimensions to visualize them, as done in Fig. 2.

D. Additional Experiments

D.1. Unsupervised transfer learning on MNIST

For the following experiments on MNIST, the reference prior is of order $n = 2$ and has $K = 50$ particles. We run our methods for 1024 epochs.

We first compare deep reference priors with fine-tuning for transfer learning. The parameter β controls the degree to which the posterior (9) is influenced by the target data. If we have $\beta = 1$, then the posterior is maximally influenced by target data after being pretrained on the source data. We instantiate (9), by combining prior selection, pretraining on the source task into one objective,

$$\max_\pi \gamma I_\pi(w; y^u, x^u) + (1-\beta)\, \mathbb{E}_{w\sim\pi} \log p(w; y^s \mid x^s), \tag{S-16}$$

where γ and β are hyper-parameters. Solving (S-16) requires no knowledge from target data labels, therefore the setting here is pure unsupervised clustering for target task dataset. We compare this objective to fine-tuning which adapts a model trained on labeled source to the labeled target data. In this experiment, all samples from the source task (about 30,000 images across 5 classes) were used for both the reference prior and fine-tuning.

Method # Labeled target data (→)	0	50	100	250	500
Source (0–4) to Target (5–9)					
Fine-Tuning	-	71.1	78.8	86.6	93.0
Deep Reference Prior Unsupervised Transfer	87.4	-	-	-	-
Source (5–9) to Target(0–4)					
Fine-Tuning	-	90.2	92.4	94.7	96.2
Deep Reference Prior Unsupervised Transfer	95.2	-	-	-	-

Table S-5. **Accuracy (%) of unsupervised reference-prior based transfer** (digits 0–4) to the target task (digits 5–9). We see that transfer using source and unlabeled target data using the reference prior performs as well as fine tuning with labeled source data and 250 labeled target data. Even if MNIST is a simple dataset, this is a remarkable demonstration of how effective the reference prior is at making use of both the labeled source data and unlabeled target data.

D.2. More ablation studies

§3.6 describes a few implementation tricks that we employ when computing $H(y^u \mid x^u, w)$. The unlabeled samples consist of both weak and strong augmentations of the same image x which we denote by $g_1(x)$ and $g_2(x)$ and we define $p_{g_i} \equiv p_w(y|g_i(x), w)$. The objective can be upper-bounded using Jensen's inequality as follows

$$\begin{aligned} H(y^u \mid x^u, w) &= -\mathop{\mathbb{E}}_{x^u} \int \mathrm{d}y^u\, p_G(y^u \mid x^u, w) \left[\log(p_G(y^u \mid x^u, w))\right] \\ &= -\mathop{\mathbb{E}}_{x^u} \int \mathrm{d}y^u\, p_G(y^u \mid x^u, w) \left[\log(\tau p_{g_1} + (1-\tau) p_{g_2})\right] \\ &\le -\mathop{\mathbb{E}}_{x^u} \int \mathrm{d}y^u\, p_G(y^u \mid x^u, w) \left[\tau \log p_{g_1} + (1-\tau)\log p_{g_2}\right] \end{aligned}$$

The first trick is to use the above bound from Jensen's inequality to compute $H(y^u \mid x^u, w)$. The second trick we employ is to not update $p(y \mid g_1(x), w)$ with back-propagation gradients. Table S-6 shows that both these tricks are needed to achieve good accuracy.

The third trick is to include x in the loss only if $\max p_w(y \mid g_1(x), w) > 0.95$ – an implementation detail also employed in (Sohn et al., 2020). Table S-6 shows that this has very little impact on accuracy.

Implementation trick	Accuracy (%)
Deep reference priors (All 3 tricks)	92.13
No stop gradient to p_{g_1}	10
No Jensen's inequality	86.55
No masking using probability threshold	92.35

Table S-6. We perform an ablation study over the three implementation tricks considered in §3.6 and compute the accuracy after removing each one of the tricks. The accuracy (%) is computed for 250 labeled samples, with 4 particles and using order 2.

D.3. Two-stage experiment for coin tossing

In §3.4, we consider a situation when we obtain data in two stages, first z^m, and then z^n. We propose a prior π^* (7) such that the posterior of the second stage makes the maximal use of the new n samples. In this section, we visualize π^* in the parameter space using a two-stage coin tossing experiment. Consider the estimation of the bias of a coin $w \in [0, 1]$ using two-stage $m + n$ trials. There are m trials in first stage and n trails in second stage. If z denotes the number of heads in total, we have $p(z \mid w) = w^z(1 - w)^{m+n-z}(m + n)!/(z!(m + n - z)!)$. We numerically find π^* for different values of m and n using the BA algorithm (Fig. S-6 and Fig. S-7).

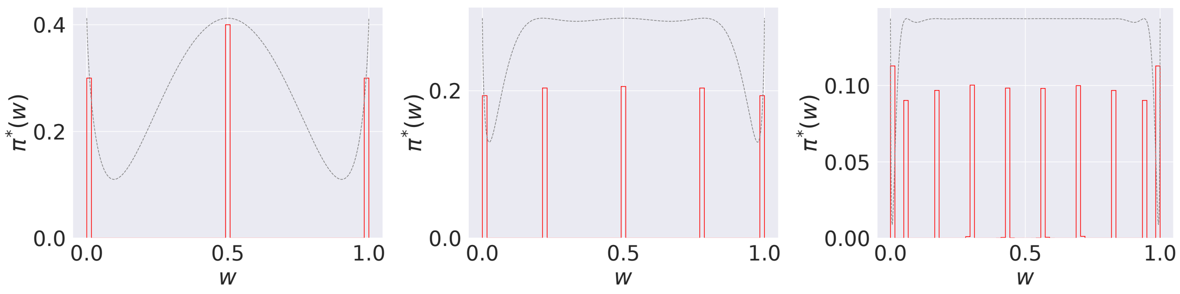


Figure S-6. **Reference prior for the two stage coin-tossing model (see (7))** for $m = 1$ and $n = 1, 10, 40$ (from left to right) computed using the Blahut-Arimoto algorithm. Atoms are critical points of the gray line which is $\mathrm{KL}(p(z^{m+n}), p(z^{m+n} \mid w)) - \mathrm{KL}(p(z^m), p(z^m \mid w))$. The prior is again discrete for finite order $n < \infty$. We see how this reference prior behaves for different values of $\alpha = m/n$, e.g., for $\alpha \to 0$ this prior π^* is close to π_n^* in (2) but there are still some differences between them. This shows that the two-stage reference prior is not the same as the single-stage reference prior.

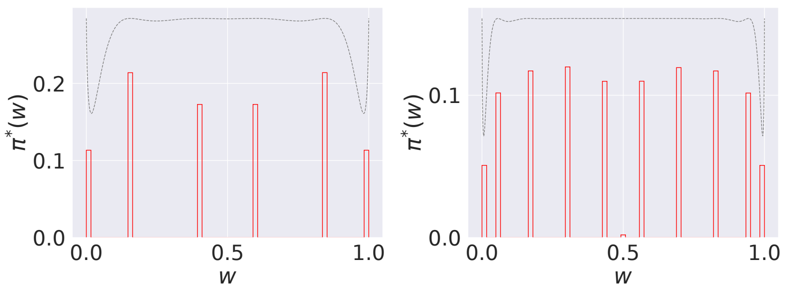


Figure S-7. **Reference prior for the two stage coin-tossing model** (see (7)) for $n = 1$ and $m = 10, 30$ (from left to right) computed using the Blahut-Arimoto algorithm. Atoms are critical points of the gray line which is $\mathrm{KL}(p(z^{m+n}), p(z^{m+n} \mid w)) - \mathrm{KL}(p(z^m), p(z^m \mid w))$.