Realistic Evaluation of Deep Semi-Supervised Learning Algorithms

Avital Oliver[*] **Augustus Odena**[*] **Colin Raffel**[*] **Ekin D. Cubuk & Ian J. Goodfellow**
Google Brain
`{avitalo,augustusodena,craffel,cubuk,goodfellow}@google.com`

Abstract

Semi-supervised learning (SSL) provides a powerful framework for leveraging unlabeled data when labels are limited or expensive to obtain. SSL algorithms based on deep neural networks have recently proven successful on standard benchmark tasks. However, we argue that these benchmarks fail to address many issues that SSL algorithms would face in real-world applications. After creating a unified reimplementation of various widely-used SSL techniques, we test them in a suite of experiments designed to address these issues. We find that the performance of simple baselines which do not use unlabeled data is often underreported, SSL methods differ in sensitivity to the amount of labeled and unlabeled data, and performance can degrade substantially when the unlabeled dataset contains out-of-distribution examples. To help guide SSL research towards real-world applicability, we make our unified reimplemention and evaluation platform publicly available.[2]

1 Introduction

It has repeatedly been shown that deep neural networks can achieve human- or super-human-level performance on certain supervised learning problems by leveraging large collections of labeled data. However, these successes come at a cost: Creating these large datasets typically requires a great deal of human effort (to manually label examples), pain and/or risk (for medical datasets involving invasive tests) or financial expense (to hire labelers or build the infrastructure needed for domain-specific data collection). For many practical problems and applications, we lack the resources to create a sufficiently large labeled dataset, which limits the wide-spread adoption of deep learning techniques.

An attractive approach towards addressing the lack of data is **semi-supervised learning** (SSL) [6]. In contrast with supervised learning algorithms, which require labels for all examples, SSL algorithms can improve their performance by also using unlabeled examples. SSL algorithms generally provide a way of learning about the structure of the data from the unlabeled examples, alleviating the need for labels. Some recent results [32, 50, 39] have shown that in certain cases, SSL approaches the performance of purely supervised learning, even when a substantial portion of the labels in a given dataset has been discarded. These results are demonstrated by taking an existing classification dataset (typically CIFAR-10 [31] or SVHN [40]) and only using a small portion of it as labeled data with the rest treated as unlabeled. The accuracy of a model trained with SSL with labeled and unlabeled data is then compared to that of a model trained on only the small labeled portion.

These recent successes raise a natural question: Are SSL approaches applicable in "real-world" settings? In this paper, we argue that this de facto way of evaluating SSL techniques does not address this question in a satisfying way. Our goal is to more directly answer this question by proposing a new

---

[*]Equal contribution
[2]`https://github.com/brain-research/realistic-ssl-evaluation`

experimental methodology which we believe better measures applicability to real-world problems. Some of our findings include:

- When given equal budget for tuning hyperparameters, the gap in performance between using SSL and using only labeled data is smaller than typically reported.

- Further, a large classifier with carefully chosen regularization trained on a small labeled dataset with no unlabeled data can reach very good accuracy. This demonstrates the importance of evaluating different SSL algorithms on the same underlying model.

- In some settings, pre-training a classifier on a different labeled dataset and then retraining on only labeled data from the dataset of interest can outperform all SSL algorithms we studied.

- Performance of SSL techniques can degrade drastically when the unlabeled data contains a different distribution of classes than the labeled data.

- Different approaches exhibit substantially different levels of sensitivity to the amount of labeled and unlabeled data.

- Realistically small validation sets would preclude reliable comparison of different methods, models, and hyperparameter settings.

Separately, as with many areas of machine learning, direct comparison of approaches is confounded by the fact that minor changes to hyperparameters, model structure, training, etc. can have an outsized impact on results. To mitigate this problem, we provide a unified and modular software reimplementation of various state-of-the-art SSL approaches that includes our proposed evaluation techniques.

The remainder of this paper is structured as follows: In section 2, we enumerate the ways in which our proposed methodology improves over standard practice. In section 3, we give an overview of modern SSL approaches for deep architectures, emphasizing those that we include in our study. Following this discussion, we carry out extensive experiments (section 4) to better study the real-world applicability of our own reimplementation of various SSL algorithms. We restrict our analysis to image classification tasks as this is the most common domain for benchmarking deep learning models. Finally, we conclude (section 5) with concrete recommendations for evaluating SSL techniques.

2   Improved Evaluation

In this work, we make several improvements to the conventional experimental procedures used to evaluate SSL methods, which typically proceed as follows: First, take a common (typically image classification) dataset used for supervised learning and throw away the labels for most of the dataset. Then, treat the portion of the dataset whose labels were retained as a small labeled dataset $\mathcal{D}$ and the remainder as an auxiliary unlabeled dataset $\mathcal{D}_{UL}$. Some (not necessarily standard) model is then trained and accuracy is reported using the unmodified test set. The choice of dataset and number of retained labels is somewhat standardized across different papers. Below, we enumerate ways that we believe this procedure can be made more applicable to real-world settings.

P.1   A Shared Implementation. We introduce a shared implementation of the underlying architectures used to compare all of the SSL methods. This is an improvement relative to prior work because though the datasets used across different studies have largely become standardized over time, other experimental details vary significantly. In some cases, different reimplementations of a simple 13-layer convolutional network are used [32, 39, 50], which results in variability in some implementation details (parameter initialization, data preprocessing, data augmentation, regularization, etc.). Further, the training procedure (optimizer, number of training steps, learning rate decay schedule, etc.) is not standardized. These differences prevent direct comparison between approaches. All in all, these issues are not unique to SSL studies; they reflect a larger reproducibility crisis in machine learning research [28, 23, 13, 35, 38].

P.2   High-Quality Supervised Baseline. The goal of SSL is to obtain better performance using the combination of $\mathcal{D}$ and $\mathcal{D}_{UL}$ than what would be obtained with $\mathcal{D}$ alone. A natural baseline to compare against is the same underlying model (with modified hyperparameters) trained in a fully-supervised manner using only $\mathcal{D}$. While frequently reported, this baseline is occasionally omitted. Moreover, it is not always apparent whether the best-case performance has been eked out of the fully-supervised model (e.g. Laine & Aila [32] and Tarvainen & Valpola [50] both report a fully supervised baseline

with ostensibly the same model but obtain accuracies that differ between the two papers by up to 15%). To ensure that our supervised baseline is high-quality, we spent 1000 trials of hyperparameter optimization to tune both our baseline as well as all the SSL methods.

P.3 Comparison to Transfer Learning. In practice a common way to deal with limited data is to "transfer" a model trained on a separate, but similar, large labeled dataset [12, 51, 9]. This is typically achieved by initializing the parameters of a new model with those from the original model, and "fine-tuning" this new model using the small dataset. While this approach is only feasible when an applicable source dataset is available, it nevertheless provides a powerful, widely-used, and rarely reported baseline to compare against.

P.4 Considering Class Distribution Mismatch. Note that when taking an existing fully-labeled dataset and discarding labels, all members of $\mathcal{D}_{UL}$ come from the same classes as those in $\mathcal{D}$. In contrast, consider the following example: Say you are trying to train a model to distinguish between ten different faces, but you only have a few images for each of these ten faces. As a result, you augment your dataset with a large unlabeled dataset of images of random people's faces. In this case, it is extremely unlikely that any of the images in $\mathcal{D}_{UL}$ will be one of the ten people the model is trained to classify. Standard evaluation of SSL algorithms neglects to consider this possibility. This issue was indirectly addressed e.g. in [32], where labeled data from CIFAR-10 (a natural image classification dataset with ten classes) was augmented with unlabeled data from Tiny Images (a huge collection of images scraped from the internet). It also shares some characteristics with the related field of "domain adaptation", where the data distribution for test samples differs from the training distribution [4, 16]. We explicitly study the effect of differing class distributions between labeled and unlabeled data.

P.5 Varying the Amount of Labeled *and* Unlabeled Data. A somewhat common practice is to vary the size of $\mathcal{D}$ by throwing away different amounts of the underlyling labeled dataset [48, 43, 45, 50]. Less common is to vary the size of $\mathcal{D}_{UL}$ in a systematic way, which could simulate two realistic scenarios: First, that the unlabeled dataset is gigantic (e.g. using billions of unlabeled natural images on the Internet to augment a natural image classification task); or second, that the unlabeled dataset is also relatively small (e.g. in medical imaging, where both obtaining and labeling data is expensive).

P.6 Realistically Small Validation Sets. An unusual artefact of the way artificial SSL datasets are created is that often the validation set (data used for tuning hyperparameters and not model parameters) is significantly larger than the training set. For example, the standard SVHN [40] dataset has a validation set of roughly 7,000 labeled examples. Many papers that evaluate SSL methods on SVHN use only 1,000 labels from the training dataset but retain the full validation set. The validation set is thus over seven times bigger than the training set. Of course, in real-world applications, this large validation set would instead be used as the training set. The issue with this approach is that any objective values (e.g. accuracy) used for hyperparameter tuning would be significantly noisier across runs due to the smaller sample size from a realistically small validation set. In these settings, extensive hyperparameter tuning may be somewhat futile due to an excessively small collection of held-out data to measure performance on. In many cases, even using cross-validation may be insufficient and additionally incurs a substantial computational cost. The fact that small validation sets constrain the ability to select models is discussed in [6] and [14]. We take this a step further and directly analyze the relationship between the validation set size and variance in estimates of a model's accuracy.

3 Semi-Supervised Learning Methods

In supervised learning, we are given a training dataset of input-target pairs $(x, y) \in \mathcal{D}$ sampled from an unknown joint distribution $p(x, y)$. Our goal is to produce a prediction function $f_\theta(x)$ parametrized by θ which produces the correct target y for previously unseen samples from $p(x)$. For example, choosing θ might amount to optimizing a loss function which reflects the extent to which $f_\theta(x) = y$ for $(x, y) \in \mathcal{D}$. In SSL we are additionally given a collection of unlabeled input datapoints $x \in \mathcal{D}_{UL}$, sampled from $p(x)$. We hope to leverage the data from $\mathcal{D}_{UL}$ to produce a prediction function which is more accurate than what would have been obtained by using $\mathcal{D}$ on its own.

From a broad perspective, the goal of SSL is to use $\mathcal{D}_{UL}$ to augment $f_\theta(x)$ with information about the structure of $p(x)$. For example, $\mathcal{D}_{UL}$ can provide hints about the shape of the data "manifold" which can produce a better estimate of the decision boundary between different possible target values.

A depiction of this concept on a simple toy problem is shown in fig. 1, where the scarcity of labeled data makes the decision boundary between two classes ambiguous but the additional unlabeled data reveals clear structure which can be discovered by an effective SSL algorithm.
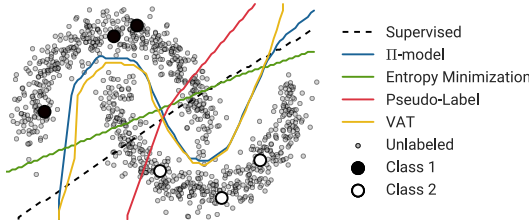

Figure 1: Behavior of the SSL approaches described in section 3 on the "two moons" dataset. We omit "Mean Teacher" and "Temporal Ensembling" (appendix A.1.2) because they behave like Π-Model (appendix A.1.1). Each approach was applied to a MLP with three hidden layers, each with 10 ReLU units. When trained on only the labeled data (large black and white dots), the decision boundary (dashed line) does not follow the contours of the data "manifold", as indicated by additional unlabeled data (small grey dots). In a simplified view, the goal of SSL is to leverage the unlabeled data to produce a decision boundary which better reflects the data's underlying structure.

A comprehensive overview of SSL methods is out of the scope of this paper; we refer interested readers to [53, 6]. Instead, we focus on the class of methods which solely involve adding an additional loss term to the training of a neural network, and otherwise leave the training and model unchanged from what would be used in the fully-supervised setting. We limit our focus to these approaches for the pragmatic reasons that they are simple to describe and implement and that they are currently the state-of-the-art for SSL on image classification datasets. Overall, the methods we consider fall into two classes: Consistency regularization, which enforces that realistic perturbations of data points $x \in \mathcal{D}_{UL}$ should not significantly change the output of $f_\theta(x)$; and entropy minimization, which encourages more confident predictions on unlabeled data. We now describe these methods in broad terms. See appendix A for more detail, and additional references to other SSL methods.

Π-Model: The simplest setting in which to apply consistency regularization is when the prediction function $f_\theta(x)$ is itself stochastic, i.e. it can produce different outputs for the same input x. This is quite common in practice during training when $f_\theta(x)$ is a neural network due to common regularization techniques such as data augmentation, dropout, and adding noise. Π-Model [32, 46] adds a loss term which encourages the distance between a network's output for different passes of $x \in \mathcal{D}_{UL}$ through the network to be small.

Mean Teacher: A difficulty with the Π-model approach is that it relies on a potentially unstable "target" prediction, namely the second stochastic network prediction which can rapidly change over the course of training. As a result, [50] proposed to obtain a more stable target output $\bar{f}_\theta(x)$ for $x \in \mathcal{D}_{UL}$ by setting the target to predictions made using an exponential moving average of parameters from previous training steps.

Virtual Adversarial Training: Instead of relying on the built-in stochasticity of $f_\theta(x)$, Virtual Adversarial Training (VAT) [39] directly approximates a tiny perturbation r_{adv} to add to x which would most significantly affect the output of the prediction function.

Entropy Minimization (EntMin): EntMin [21] adds a loss term applied that encourages the network to make "confident" (low-entropy) predictions for all unlabeled examples, regardless of their class.

Pseudo-Labeling: Pseudo-labeling [34] proceeds by producing "pseudo-labels" for $\mathcal{D}_{UL}$ using the prediction function itself over the course of training. Pseudo-labels which have a corresponding class probability which is larger than a predefined threshold are used as targets for a standard supervised loss function applied to $\mathcal{D}_{UL}$.

4 Experiments

In this section we cover issues with the evaluation of SSL techniques. We first address **P.1** and create a unified reimplementation of the methods outlined in section 3 using a common model architecture and training procedure. Our goal is *not* to produce state-of-the-art results, but instead to provide a rigorous comparative analysis in a common framework. Further, because our model architecture and training hyperparameters differ from those used to test SSL methods in the past, our results are not directly comparable to past work and should therefore be considered in isolation (see appendix D for

Table 1: Test error rates obtained by various SSL approaches on the standard benchmarks of CIFAR-10 with all but 4,000 labels removed and SVHN with all but 1,000 labels removed, using our proposed unified reimplementation. "Supervised" refers to using only 4,000 and 1,000 labeled datapoints from CIFAR-10 and SVHN respectively without any unlabeled data. VAT and EntMin refers to Virtual Adversarial Training and Entropy Minimization respectively (see section 3).

Dataset	# Labels	Supervised	Π-Model	Mean Teacher	VAT	VAT + EntMin	Pseudo-Label
CIFAR-10	4000	20.26 ± .38%	16.37 ± .63%	15.87 ± .28%	13.86 ± .27%	13.13 ± .39%	17.78 ± .57%
SVHN	1000	12.83 ± .47%	7.19 ± .27%	5.65 ± .47%	5.63 ± .20%	5.35 ± .19%	7.62 ± .29%

a full comparison). We use our reimplementation as a consistent testbed on which we carry out a series of experiments, each of which individually focusses on a single issue from section 2.

4.1 Reproduction

For our reimplementation, we selected a standard model that is modern, widely used, and would be a reasonable choice for a practitioner working on image classification. This avoids the possibility of using an architecture which is custom-tailored to work well with one particular SSL technique. We chose a Wide ResNet [52], due to their widespread adoption and availability. Specifically, we used "WRN-28-2", i.e. ResNet with depth 28 and width 2, including batch normalization [25] and leaky ReLU nonlinearities [36]. We did not deviate from the standard specification for WRN-28-2 so we refer to [52] for model specifics. For training, we chose the ubiquitous Adam optimizer [29]. For all datasets, we followed standard procedures for regularization, data augmentation, and preprocessing; details are in appendix B.

Given the model, we implemented each of the SSL approaches in section 3. To ensure that all of the techniques we are studying are given fair and equal treatment and that we are reporting the best-case performance under our model, we carried out a large-scale hyperparameter optimization. For every SSL technique in addition to a "fully-supervised" (not utilizing unlabeled data) baseline, we ran 1000 trials of Gaussian Process-based black box optimization using Google Cloud ML Engine's hyperparameter tuning service [18]. We optimized over hyperparameters specific to each SSL algorithm in addition to those shared across approaches.

We tested each SSL approach on the widely-reported image classification benchmarks of SVHN [40] with all but 1000 labels discarded and CIFAR-10 [31] with all but 4,000 labels discarded. This leaves 41,000 and 64,932 unlabeled images for CIFAR-10 and SVHN respectively when using standard validation set sizes (see appendix B). We optimized hyperparameters to minimize classification error on the standard validation set from each dataset, as is common practice (an approach we evaluate critically in section 4.6). Black-box hyperparameter optimization can produce unintuitive hyperparameter settings which vary unnecessarily between different datasets and SSL techniques. We therefore audited the best solutions found for each dataset/SSL approach combination and hand-designed a simpler, unified set of hyperparameters. Hyperparameter values were selected if they were shared across different SSL approaches and achieved comparable performance to those found by the tuning service. After unification, the only hyperparameters which varied across different SSL algorithms were the learning rate, consistency coefficient, and any hyperparameters unique to a given algorithm (e.g. VAT's ϵ hyperparameter). An enumeration of these hyperparameter settings can be found in appendix C.

We report the test error at the point of lowest validation error for the hyperparameter settings we chose in table 1. We use this hyperparameter setting without modification in all of our experiments.

4.2 Fully-Supervised Baselines

By using the same budget of hyperparameter optimization trials for our fully-supervised baselines, we believe we have successfully addressed item **P.2**. For comparison, table 2 shows the fully-supervised baseline and SSL error rates listed in prior studies. **We find the gap between the fully-supervised baseline and those obtained with SSL is smaller in our study than what is generally reported in the literature.** For example, [32] report a fully-supervised baseline error rate of 34.85% on CIFAR-10 with 4000 labels which is improved to 12.36% using SSL; our improvement for the same approach went from 20.26% (fully-supervised) to 16.37% (with SSL).

5

Table 2: Reported change in error rate from fully-supervised (no unlabeled data) to SSL. We do not report results for VAT because a fully-supervised baseline was not reported in [39]. We also did not include the SVHN results from [46] because they use 732 labeled examples instead of 1000.

Method	CIFAR-10 4000 Labels	SVHN 1000 Labels
Π-Model [32]	$34.85\% \rightarrow 12.36\%$	$19.30\% \rightarrow 4.80\%$
Π-Model [46]	$13.60\% \rightarrow 11.29\%$	–
Π-Model (ours)	$20.26\% \rightarrow 16.37\%$	$12.83\% \rightarrow 7.19\%$
Mean Teacher [50]	$20.66\% \rightarrow 12.31\%$	$12.32\% \rightarrow 3.95\%$
Mean Teacher (ours)	$20.26\% \rightarrow 15.87\%$	$12.83\% \rightarrow 5.65\%$

Table 3: Comparison of error rate using SSL and transfer learning. VAT with Entropy Minimization was the most performant method on CIFAR-10 in our experiments. "No overlap" refers to transferring from an ImageNet model which was not trained on classes which are similar to the classes in CIFAR-10 (see section 4.3 for details).

Method	CIFAR-10 4000 Labels
VAT with Entropy Minimization	13.13%
ImageNet → CIFAR-10	12.09%
ImageNet → CIFAR-10 (no overlap)	12.91%

We can push this line of enquiry even further: Can we design a model with a regularization, data augmentation, and training scheme which can match the performance of SSL techniques without using any unlabeled data? Of course, comparing the performance of this model to SSL approaches with different models is unfair; however, we want to understand the upper-bound of fully-supervised performance as a benchmark for future work.

After extensive experimentation, we chose the large Shake-Shake model of [17] due to its powerful regularization capabilities. We used a standard data-augmentation scheme consisting of random horizontal flips and random crops after zero-padding by 4 pixels on each side [22], as well as cutout regularization with a patch length of 16 pixels [11]. Training and regularization was as in [17], with a learning rate of 0.025 and weight decay factor of 0.0025. On 4,000 labeled data points from CIFAR-10, this model obtained an average test error of **13.4%** over 5 independent runs. This result emphasizes the importance of the underlying model in the evaluation of SSL algorithms, and reinforces our point that **different algorithms must be evaluated using the same model to avoid conflating comparison.**

4.3   Transfer Learning

Further to the point of item **P.2**, we also studied the technique of transfer learning using a pre-trained classifier, which is frequently used in limited-data settings but often neglected in SSL studies. We trained our standard WRN-28-2 model on ImageNet [10] downsampled to 32x32 [7] (the native image size of CIFAR-10). We used the same training hyperparameters as used for the supervised baselines reported in section 4.1. Then, we fine-tuned the model using 4,000 labeled data points from CIFAR-10. As shown in table 3, the resulting model obtained an error rate of **12.09%** on the test set. **This is a lower error rate than any SSL technique achieved using this network, indicating that transfer learning may be a preferable alternative when a labeled dataset suitable for transfer is available.** Note that we did not tune our model architecture or hyperparameters to improve this transfer learning result - we simply took the baseline model from our SSL experiments and used it for transfer learning. This suggests that the error rate of 12.09% is a conservative estimate of the potential performance of transfer learning in this setting.

Note that ImageNet and CIFAR-10 have many classes in common, suggesting that this result may reflect the best-case application of transfer learning. To test how much this overlap affected transfer learning performance, we repeated the experiment after removing the 252 ImageNet classes (listed in appendix F) which were similar to any of the CIFAR-10 classes. Performance degraded moderately to 12.91%, which is comparable to the best SSL technique we studied. We also experimented with transfer learning from ImageNet to SVHN, which reflects a much more challenging setting requiring substantial domain transfer. We were unable to achieve convincing results when transferring to SVHN which suggests that the success of transfer learning may heavily depend on how closely related the two datasets are. More concretely, it primarily demonstrates that the transfer learning on a separate, related, and labeled dataset can provide the network with a better learning signal than SSL can using unlabeled data. We are interested in exploring the combination of transfer learning and SSL in future work.

4.4 Class Distribution Mismatch

Now we examine the case where labeled and unlabeled data come from the same underlying distribution (e.g. natural images), but the unlabeled data contains classes not present in the labeled data. This setting violates the strict definition of semi-supervised learning given in section 3, but as outlined in item **P.4** it nevertheless represents a common use-case for SSL (for example, augmenting a face recognition dataset with unlabeled images of people not in the labeled set). To test this, we synthetically vary the class overlap in our common test setting of CIFAR-10. Specifically, we perform 6-class classification on CIFAR-10's animal classes (bird, cat, deer, dog, frog, horse). The unlabeled data comes from four classes — we vary how many of those four are among the six labeled classes to modulate class distribution mismatch. We also compare to a fully supervised model trained using no unlabeled data. As before, we use 400 labels per class for CIFAR-10, resulting in 2400 labeled examples.

Our results are shown in fig. 2. **We demonstrate the surprising result that adding unlabeled data from a mismatched set of classes can actually *hurt* performance compared to not using any unlabeled data at all** (points above the black dotted line in fig. 2). This implies that it may be preferable to pay a larger cost to obtain labeled data than to obtain unlabeled data if the unlabeled data is sufficiently unrelated to the core learning task. However, we did not re-tune hyperparameters these experiments; it is possible that by doing so the gap could narrow.

4.5 Varying Data Amounts

Many SSL techniques are tested only in the core settings we have studied so far, namely CIFAR-10 with 4,000 labels and SVHN with 1,000 labels. However, we argue that varying the amount of labeled data tests how performance degrades in the very-limited-label regime, and also at which point the approach can recover the performance of training with all of the labels in the dataset. We therefore ran experiments on both SVHN and CIFAR with different labeled data amounts; the results are shown in fig. 4. In general, the performance of all of the SSL techniques tends to converge as the number of labels grows. On SVHN, VAT exhibits impressively consistent performance across labeled data amounts, where in contrast the performance of Π-Model is increasingly poor as the number of labels decreases. As elsewhere, we emphasize that these results only apply to our specific architecture and hyperparameter settings and may not provide general insight into each algorithms' behavior.

Another possibility is to vary the amount of unlabeled data. However, using the CIFAR-10 and SVHN datasets in isolation places an upper limit on the amount of unlabeled data available. Fortunately, SVHN is distributed with the "SVHN-extra" dataset, which adds 531,131 additional digit images and which was previously used as unlabeled data in [50]. Similarly, the "Tiny Images" dataset can augment CIFAR-10 with eighty million additional unlabeled images as done in [32], however it also introduces a class distribution mismatch between labeled and unlabeled data because its images are not necessarily from the classes covered by CIFAR-10. As a result, we do not consider Tiny Images for auxiliary unlabeled data in this paper.

We evaluated the performance of each SSL technique on SVHN with 1,000 labels and varying amounts of unlabeled data from SVHN-extra, which resulted in the test errors shown in fig. 3. As expected, increasing the amount of unlabeled data tends to improve the performance of SSL techniques. However, we found that performance levelled off consistently across algorithms once 80,000 unlabeled examples were available. Furthermore, performance seems to degrade slightly for Pseudo-Labeling and Π-Model as the amount of unlabeled data increases. More broadly, **we find surprisingly different levels of sensitivity to varying data amounts across SSL techniques.**

4.6 Small Validation Sets

In all of the experiments above (and in recent experiments in the literature that we are aware of), hyperparameters are tuned on a labeled validation set which is significantly larger than the labeled portion of the training set. We measure the extent to which this provides SSL algorithms with an unrealistic advantage compared to real-world scenarios where the validation set would be smaller.

We can derive a theoretical estimate for the number of validation samples required to confidently differentiate between the performance of different approaches using Hoeffding's inequality [24]:

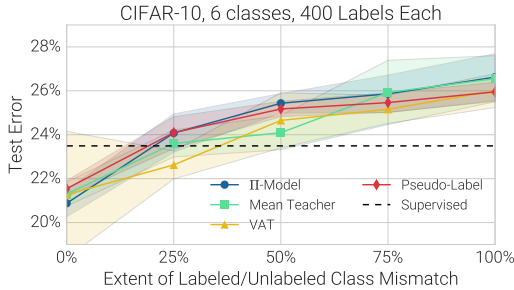$$\mathbf{P}(|\bar{V} - \mathbb{E}[V]| < p) > 1 - 2\exp(-2np^2) \tag{1}$$

Figure 2: Test error for each SSL technique on CIFAR-10 (six animal classes) with varying overlap between classes in the labeled and unlabeled data. For example, in "25%", one of the four classes in the unlabeled data is not present in the labeled data. "Supervised" refers to using no unlabeled data. Shaded regions indicate standard deviation over five trials.
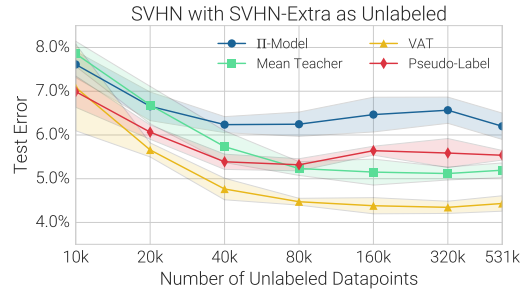
Figure 3: Test error for each SSL technique on SVHN with 1,000 labels and varying amounts of unlabeled images from SVHN-extra. Shaded regions indicate standard deviation over five trials. X-axis is shown on a logarithmic scale.
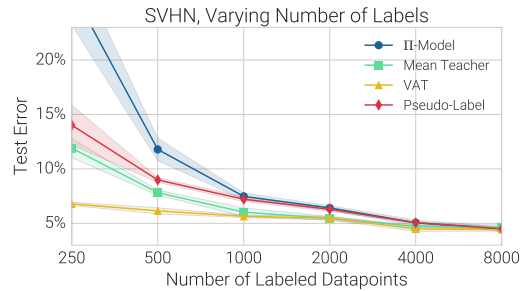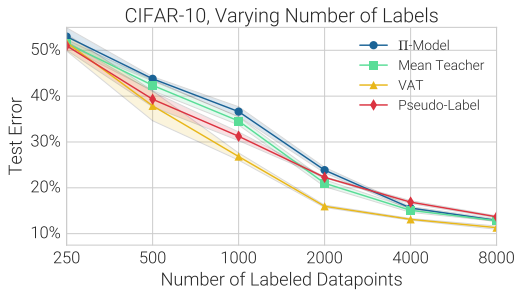

Figure 4: Test error for each SSL technique on SVHN and CIFAR-10 as the amount of labeled data varies. Shaded regions indicate standard deviation over five trials. X-axis is shown on a logarithmic scale.

where in our case $\bar{V}$ is the empirical estimate of the validation error, $\mathbb{E}[V]$ is its hypothetical true value, p is the desired maximum deviation between our estimate and the true value, and n is the number of examples in the validation set. In this analysis, we are treating validation error as the average of independent binary indicator variables denoting whether a given example in the validation set is classified correctly or not. As an example, if we want to be 95% confident that our estimate of the validation error differs by less than 1% absolute of the true value, we would need nearly 20,000 validation examples. This is a disheartening estimate due to the fact that the difference in test error achieved by different SSL algorithms reported in table 1 is often close to or smaller than 1%, but 20,000 is many times more samples than are provided in the training sets.

This theoretical analysis may be unrealistic due to the assumption that the validation accuracy is the average of independent variables. To measure this phenomenon empirically, we took baseline models trained with each SSL approach on SVHN with 1,000 labels and evaluated them on validation sets with varying sizes. These synthetic small validation sets were sampled randomly and without overlap from the full SVHN validation set. We show the mean and standard deviation of validation error over 10 randomly-sampled validation sets for each method in fig. 5. For validation sets of the same size (100%) as the training set, some differentiation between the approaches is possible. **However, for a realistically-sized validation set (10% of the training set size), differentiating between the performance of the models is not feasible.** This suggests that SSL methods which rely on heavy hyperparameter tuning on a large validation set may have limited real-world applicability. Cross-validation can help with this problem, but the reduction of variance may still be insufficient and its use would incur an N-fold computational increase.

A possible objection to this experiment is that there may be strong correlation between the accuracy of different SSL techniques when measured on the same validation set. If that is indeed the case, then principled model selection may be possible because all that is necessary is choosing the better of a class of models, not estimating each model's expected error as an exact number. In order to account for this objection, in fig. 6 we show the mean and standard deviation of the *difference in validation error* between each SSL model and Π-model (chosen arbitrarily as a point of comparison).
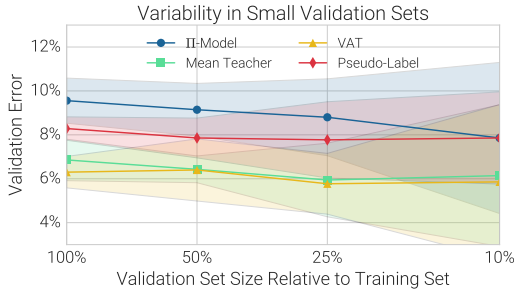
Figure 5: Average validation error over 10 randomly-sampled nonoverlapping validation sets of varying size. For each SSL approach, we re-evaluated an identical model on each randomly-sampled validation set. The mean and standard deviation of the validation error over the 10 sets are shown as lines and shaded regions respectively. Models were trained on SVHN with 1,000 labels. Validation set sizes are listed relative to the training size (e.g. 10% indicates a size-100 validation set). X-axis is shown on a logarithmic scale.
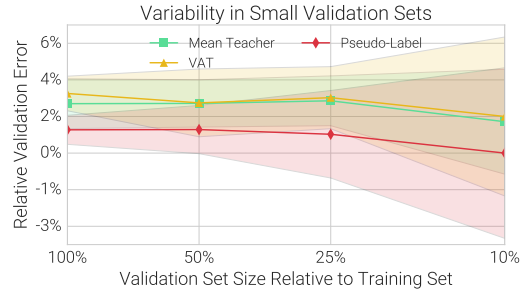
Figure 6: Average and standard deviation of relative error over 10 randomly-sampled nonoverlapping validation sets of varying size. The experimental set-up is identical to the one in fig. 5, with the following change: The mean and standard deviation are computed over the difference in validation error compared to Π-model, rather than the absolute validation error. X-axis is shown on a logarithmic scale.

For realistically small validation sets under this setting, the overlap between the error bounds with small validation sets still surpasses the difference between the error for different models. Thus, we still argue that with realistically small validation sets, model selection may not be feasible.

5 Conclusions and Recommendations

Our experiments provide strong evidence that standard evaluation practice for SSL is unrealistic. What changes to evaluation should be made to better reflect real-world applications? Our recommendations for evaluating SSL algorithms are as follows:

- Use the exact same underlying model when comparing SSL approaches. Differences in model structure or even implementation details can greatly impact results.

- Report well-tuned fully-supervised and transfer learning performance where applicable as baselines. The goal of SSL should be to significantly outperform the fully-supervised settings.

- Report results where the class distribution mismatch systematically varies. We showed that the SSL techniques we studied all suffered when the unlabeled data came from different classes than the labeled data — a realistic scenario that to our knowledge is drastically understudied.

- Vary both the amount of labeled and unlabeled data when reporting performance. An ideal SSL algorithm is effective even with very little labeled data and benefits from additional unlabeled data. Specifically, we recommend combining SVHN with SVHN-Extra to test performance in the large-unlabeled-data regime.

- Take care not to over-tweak hyperparameters on an unrealistically large validation set. A SSL method which requires significant tuning on a per-model or per-task basis in order to perform well will not be useable when validation sets are realistically small.

Our discoveries also hint towards settings where SSL is most likely the right choice for practitioners:

- When there are no high-quality labeled datasets from similar domains to use for fine-tuning.

- When the labeled data is collected by sampling i.i.d. from the pool of the unlabeled data, rather than coming from a (slightly) different distribution.

- When the labeled dataset is large enough to accurately estimate validation accuracy, which is necessary when doing model selection and tuning hyperparameters.

SSL has seen a great streak of successes recently. We hope that our results and publicly-available unified implementation help push these successes towards the real world.

Acknowledgements

References

[1] Bachman, Philip, Alsharif, Ouais, and Precup, Doina. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, 2014.

[2] Belkin, Mikhail and Niyogi, Partha. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, 2002.

[3] Belkin, Mikhail, Niyogi, Partha, and Sindhwani, Vikas. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[4] Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. *Machine learning Journal*, 2010.

[5] Bengio, Yoshua, Delalleau, Olivier, and Le Roux, Nicolas. *Label Propagation and Quadratic Criterion*, chapter 11. MIT Press, 2006.

[6] Chapelle, Olivier, Scholkopf, Bernhard, and Zien, Alexander. *Semi-Supervised Learning*. MIT Press, 2006.

[7] Chrabaszcz, Patryk, Loshchilov, Ilya, and Hutter, Frank. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[8] Coates, Adam and Ng, Andrew Y. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, 2011.

[9] Dai, Andrew M. and Le, Quoc V. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, 2015.

[10] Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[11] DeVries, Terrance and Taylor, Graham W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[12] Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

[13] Fedus, William, Rosca, Mihaela, Lakshminarayanan, Balaji, Dai, Andrew M., Mohamed, Shakir, and Goodfellow, Ian. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. *International Conference on Learning Representations*, 2018.

[14] Forster, Dennis, Sheikh, Abdul-Saboor, and Lücke, Jörg. Neural simpletrons: Learning in the limit of few labels with directed generative networks. *Neural computation*, (Early Access): 1–62.

[15] Gammerman, Alexander, Vovk, Volodya, and Vapnik, Vladimir. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

[16] Ganin, Yaroslav, Ustinova, Evgeniya, Ajakan, Hana, Germain, Pascal, Larochelle, Hugo, Laviolette, François, Marchand, Mario, and Lempitsky, Victor. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 2016.

[17] Gastaldi, Xavier. Shake-shake regularization. *Fifth International Conference on Learning Representations (Workshop Track)*, 2017.

[18] Golovin, Daniel, Solnik, Benjamin, Moitra, Subhodeep, Kochanski, Greg, Karro, John, and Sculley, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[19] Goodfellow, Ian J., Courville, Aaron, and Bengio, Yoshua. Spike-and-slab sparse coding for unsupervised feature discovery. In *NIPS Workshop on Challenges in Learning Hierarchical Models*, 2011.

[20] Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. In *Third International Conference on Learning Representations*, 2015.

[21] Grandvalet, Yves and Bengio, Yoshua. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, 2005.

[22] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016.

[23] Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina, and Meger, David. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Hoeffding, Wassily. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[25] Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[26] Joachims, Thorsten. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

[27] Joachims, Thorsten. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, 2003.

[28] Ke, Rosemary Nan, Goyal, Anirudh, Lamb, Alex, Pineau, Joelle, Bengio, Samy, and Bengio, Yoshua (eds.). *Reproducibility in Machine Learning Research*, 2017.

[29] Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. In *Second International Conference on Learning Representations*, 2014.

[30] Kingma, Diederik P., Mohamed, Shakir, Rezende, Danilo Jimenez, and Welling, Max. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2014.

[31] Krizhevsky, Alex. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[32] Laine, Samuli and Aila, Timo. Temporal ensembling for semi-supervised learning. In *Fifth International Conference on Learning Representations*, 2017.

[33] Lasserre, Julia A., Bishop, Christopher M., and Minka, Thomas P. Principled hybrids of generative and discriminative models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[34] Lee, Dong-Hyun. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

[35] Lucic, Mario, Kurach, Karol, Michalski, Marcin, Gelly, Sylvain, and Bousquet, Olivier. Are GANs created equal? A large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[36] Maas, Andrew L., Hannun, Awni Y., and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

[37] McLachlan, Geoffrey J. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.

[38] Melis, Gábor, Dyer, Chris, and Blunsom, Phil. On the state of the art of evaluation in neural language models. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[39] Miyato, Takeru, Maeda, Shin-ichi, Koyama, Masanori, and Ishii, Shin. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[40] Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[41] Odena, Augustus. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[42] Pereyra, Gabriel, Tucker, George, Chorowski, Jan, Kaiser, Łukasz, and Hinton, Geoffrey. Regularizing neural networks by penalizing confident output distributions. In *Fifth International Conference on Learning Representations (Workshop Track)*, 2017.

[43] Pu, Yunchen, Gan, Zhe, Henao, Ricardo, Yuan, Xin, Li, Chunyuan, Stevens, Andrew, and Carin, Lawrence. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems*, 2016.

[44] Rosenberg, Chuck, Hebert, Martial, and Schneiderman, Henry. Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, 2005.

[45] Sajjadi, Mehdi, Javanmardi, Mehran, and Tasdizen, Tolga. Mutual exclusivity loss for semi-supervised deep learning. In *IEEE International Conference on Image Processing*, 2016.

[46] Sajjadi, Mehdi, Javanmardi, Mehran, and Tasdizen, Tolga. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2016.

[47] Salakhutdinov, Ruslan and Hinton, Geoffrey E. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, 2007.

[48] Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, 2016.

[49] Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. In *Second International Conference on Learning Representations*, 2014.

[50] Tarvainen, Antti and Valpola, Harri. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems*, 2017.

[51] Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.

[52] Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[53] Zhu, Xiaojin, Ghahramani, Zoubin, and Lafferty, John D. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, 2003.

A   Semi-supervised learning methods, in more detail

There have been a wide variety of proposed SSL methods, including "transductive" [15] variants of k-nearest neighbors [27] and support vector machines [26], graph-based methods [53, 5], and algorithms based on learning features (frequently via generative modeling) from unlabeled data [2, 33, 47, 8, 19, 30, 43, 41, 48].

A comprehensive overview is out of the scope of this paper; we instead refer interested readers to [53, 6].

We now describe the methods we analyze in this paper (as described in section 3) in more detail.

A.1   Consistency Regularization

Consistency regularization describes a class of methods with following intuitive goal: Realistic perturbations $x \rightarrow \hat{x}$ of data points $x \in \mathcal{D}_{UL}$ should not significantly change the output of $f_\theta(x)$. Generally, this involves minimizing $d(f_\theta(x), f_\theta(\hat{x}))$ where $d(\cdot, \cdot)$ measures a distance between the prediction function's outputs, e.g. mean squared error or Kullback-Leibler divergence. Typically the gradient through this consistency term is only backpropagated through $f_\theta(\hat{x})$. In the toy example of fig. 1, this would ideally result in the classifier effectively separating the two class clusters due to the fact that their members are all close together. Consistency regularization can be seen as a way of leveraging the unlabeled data to find a smooth manifold on which the dataset lies [3]. This simple principle has produced a series of approaches which are currently state-of-the-art for SSL.

A.1.1   Stochastic Perturbations/Π-Model

The simplest setting in which to apply consistency regularization is when the prediction function $f_\theta(x)$ is itself stochastic, i.e. it can produce different outputs for the same input x. This is common when $f_\theta(x)$ is a neural network due to common regularization techniques such as data augmentation, dropout, and adding noise. These regularization techniques themselves are typically designed in such a way that they ideally should not cause the model's prediction to change, and so are a natural fit for consistency regularization.

The straightforward application of consistency regularization is thus minimizing $d(f_\theta(x), f_\theta(\hat{x}))$ for $x \in \mathcal{D}_{UL}$ where in this case $d(\cdot, \cdot)$ is chosen to be mean squared error. This distance term is added to the classification loss as a regularizer, scaled by a weighting hyperparameter. This idea was first proposed in [1] and later studied in [46] and [32], and has been referred to as "Pseudo-Ensembles", "Regularization With Stochastic Transformations and Perturbations" and the "Π-Model" respectively. We adopt the latter name for its conciseness. In fig. 1, the Π-Model successfully finds the correct decision boundary.

A.1.2   Temporal Ensembling/Mean Teacher

A difficulty with the Π-model approach is that it relies on a potentially unstable "target" prediction, namely the second stochastic network prediction which can rapidly change over the course of training. As a result, [50] and [32] proposed two methods for obtaining a more stable target output $\bar{f}_\theta(x)$ for $x \in \mathcal{D}_{UL}$. "Temporal Ensembling" [32] uses an exponentially accumulated average of outputs of $f_\theta(x)$ for the consistency target. Inspired by this approach, "Mean Teacher" [50] instead proposes to use a prediction function parametrized by an exponentially accumulated average of θ over training. As with the Π-model, the mean squared error $d(f_\theta(x), \bar{f}_\theta(x))$ is added as a regularization term with a weighting hyperparameter. In practice, it was found that the Mean Teacher approach outperformed Temporal Ensembling [50], so we will focus on it in our later experiments.

A.1.3   Virtual Adversarial Training

Instead of relying on the built-in stochasticity of $f_\theta(x)$, Virtual Adversarial Training (VAT) [39] directly approximates a tiny perturbation r_{adv} to add to x which would most significantly affect the output of the prediction function. An approximation to this perturbation can be computed efficiently as

$$r \sim \mathcal{N}\left(0, \frac{\xi}{\sqrt{\dim(x)}} I\right) \qquad (2)$$

$$g = \nabla_r d(f_\theta(x), f_\theta(x + r)) \tag{3}$$

$$r_{adv} = \epsilon \frac{g}{||g||} \tag{4}$$

where ξ and ϵ are scalar hyperparameters. Consistency regularization is then applied to minimize $d(f_\theta(x), f_\theta(x + r_{adv}))$ with respect to θ, effectively using the "clean" output as a target given an adversarially perturbed input. VAT is inspired by adversarial examples [49, 20], which are natural datapoints x which have a virtually imperceptible perturbation added to them which causes a trained model to misclassify the datapoint. Like the Π-Model, the perturbations caused by VAT find the correct decision boundary in fig. 1.

A.2 Entropy-Based

A simple loss term which can be applied to unlabeled data is to encourage the network to make "confident" (low-entropy) predictions for all examples, regardless of the actual class predicted. Assuming a categorical output space with K possible classes (e.g. a K-dimensional $\mathrm{softmax}$ output), this gives rise to the "entropy minimization" term [21]:

$$-\sum_{k=1}^{K} f_\theta(x)_k \log f_\theta(x)_k \tag{5}$$

Ideally, entropy minimization will discourage the decision boundary from passing near data points where it would otherwise be forced to produce a low-confidence prediction [21]. However, given a high-capacity model, another valid low-entropy solution is simply to create a decision boundary which has overfit to locally avoid a small number of data points, which is what appears to have happened in the synthetic example of fig. 1 (see appendix E for further discussion). On its own, entropy minimization has not been shown to produce competitive results compared to the other methods described here [45]. However, entropy minimization was combined with VAT to obtain state-of-the-art results by [39]. An alternative approach which is applicable to multi-label classification was proposed by [45], but it performed similarly to entropy minimization on standard "one-hot" classification tasks. Interestingly, entropy *maximization* was also proposed as a regularization strategy for neural networks by [42].

A.3 Pseudo-Labeling

Pseudo-labeling [34] is a simple heuristic which is widely used in practice, likely because of its simplicity and generality – all that it requires is that the model provides a probability value for each of the possible labels. It proceeds by producing "pseudo-labels" for $\mathcal{D}_{UL}$ using the prediction function itself over the course of training. Pseudo-labels which have a corresponding class probability which is larger than a predefined threshold are used as targets for a standard supervised loss function applied to $\mathcal{D}_{UL}$. While intuitive, it can nevertheless produce incorrect results when the prediction function produces unhelpful targets for $\mathcal{D}_{UL}$, as shown in fig. 1. Note that pseudo-labeling is quite similar to entropy regularization, in the sense that it encourages the model to produce higher-confidence (lower-entropy) predictions for data in $\mathcal{D}_{UL}$ [34]. However, it differs in that it only enforces this for data points which already had a low-entropy prediction due to the confidence thresholding. Pseudo-labeling is also closely related to self-training [44, 37], which differs only in the heuristics used to decide which pseudo-labels to retain. The Pseudo-labeling paper [34] also discusses using unsupervised pre-training; we did not implement this in our experiments.

B Dataset details

Overall, we followed standard data normalization and augmentation practice. For SVHN, we converted image data to floating point values in the range [-1, 1]. For data augmentation, we solely used random translation by up to 2 pixels. We used the standard train/validation split, with 65,932 images for training and 7,325 for validation.

For any model which was to be used to classify CIFAR-10 (e.g. including the base ImageNet model for the transfer learning experiment in section 4.3), we applied global contrast normalization and ZCA-normalized the inputs using statistics calculated on the CIFAR-10 training set. ZCA normalization is a widely-used and surprisingly important preprocessing step for CIFAR-10. Data augmentation on CIFAR-10 included random horizontal flipping, random translation by up to 2 pixels, and Gaussian input noise with standard deviation 0.15. We used the standard train/validation split, with 45,000 images for training and 5,000 for validation.

Table 4: Hyperparameter settings used in our experiments. All hyperparameters were tuned via large-scale hyperparameter optimization and then distilled to sensible and unified defaults by hand. Adam's β_1, β_2, and ϵ parameters were left to the defaults suggested by [29]. *Following [50], we ramped up the consistency coefficient starting from 0 to its maximum value using a sigmoid schedule so that it achieved its maximum value at 200,000 iterations. **We found that CIFAR-10 and SVHN required different values for ϵ in VAT (6.0 and 1.0 respectively), likely due to the difference in how the input is normalized in each dataset.

Shared	
L1 regularization coefficient	0.001
L2 regularization coefficient	0.0001
Learning decayed by a factor of	0.2
at training iteration	400,000
Consistency coefficient rampup*	200,000
Supervised	
Initial learning rate	0.003
Π-Model	
Initial learning rate	0.0003
Max consistency coefficient	20
Mean Teacher	
Initial learning rate	0.0004
Max consistency coefficient	8
Exponential moving average decay	0.95
VAT	
Initial learning rate	0.003
Max consistency coefficient	0.3
VAT ϵ	6.0 or 1.0**
VAT ξ	10^{-6}
VAT + EM (as for VAT)	
Entropy penalty multiplier	0.06
Pseudo-Label	
Initial learning rate	0.003
Max consistency coefficient	1.0
Pseudo-label threshold	0.95

C Hyperparameters

In our hyperparameter search, for each SSL method, we always separately optimized algorithm-agnostic hyperparameters such as the learning rate, its decay schedule and weight decay coefficients. In addition, we optimized to those hyperparameters specific to different SSL approaches separately for each approach. In keeping with our argument in section 4.6, we attempted to find hyperparameter settings which were performant across datasets and SSL approaches so that we could avoid unrealistic tweaking. After hand-tuning, we used the hyperparameter settings summarized in table 4, which lists those settings which were shared and common to all SSL approaches.

We trained all networks for 500,000 updates with a batch size of 100. We did not use any form of early stopping, but instead continuously monitored validation set performance and report test error at the point of lowest validation error. All models were trained with a single worker on a single GPU (i.e. no asynchronous training).

Table 5: Test error rates obtained by various SSL approaches on the standard benchmarks of CIFAR-10 with all but 4,000 labels removed and SVHN with all but 1,000 labels removed. Top: Reported results in the literature; Bottom: Using our proposed unified reimplementation. "Supervised" refers to using only 4,000 and 1,000 labeled datapoints from CIFAR-10 and SVHN respectively without any unlabeled data. VAT + EntMin refers Virtual Adversarial Training with Entropy Minimization (see section 3). Note that the model used for results in the bottom has roughly half as many parameters as most models in the top (see section 4.1).

Method	CIFAR-10 4000 Labels	SVHN 1000 Labels
Π-Model [46]	11.29%	–
Π-Model [32]	12.36%	4.82%
Mean Teacher [50]	12.31%	3.95%
VAT [39]	11.36%	5.42%
VAT + EntMin [39]	10.55%	3.86%
Results above this line cannot be directly compared to those below		
Supervised	$20.26 \pm 0.38\%$	$12.83 \pm 0.47\%$
Π-Model	$16.37 \pm 0.63\%$	$7.19 \pm 0.27\%$
Mean Teacher	$15.87 \pm 0.28\%$	$5.65 \pm 0.47\%$
VAT	$13.86 \pm 0.27\%$	$5.63 \pm 0.20\%$
VAT + EntMin	$13.13 \pm 0.39\%$	$5.35 \pm 0.19\%$
Pseudo-Label	$17.78 \pm 0.57\%$	$7.62 \pm 0.29\%$

D Comparison of our results with reported results in the literature

In table 5, we show how our results compare to what has been reported in the literature. Our numbers cannot be directly compared to those previously reported due to a lack of a shared underlying network architecture. For example, our model has roughly half as many parameters as the one used in [32, 39, 50], which may partially explain its somewhat worse performance. Our findings are generally consistent with these; namely, that all of these SSL methods improve (to a varying degree) over the baseline. Further, Virtual Adversarial Training and Mean Teacher both appear to work best, which is consistent with their shared state-of-the-art status.

E Decision boundaries found by Entropy Minimization cut through the unlabeled data

Why does Entropy Minimization not find good decision boundaries in the "two moons" figure (fig. 1)? Even though a decision boundary that avoids both clusters of unlabeled data would achieve low loss, so does any decision boundary that's extremely confident and "wiggles" around each individual unlabeled data point. The neural network easily overfits to such a decision boundary simply by increasing the magnitude of its output logits. Figure 7 shows how training changes the decision contours.
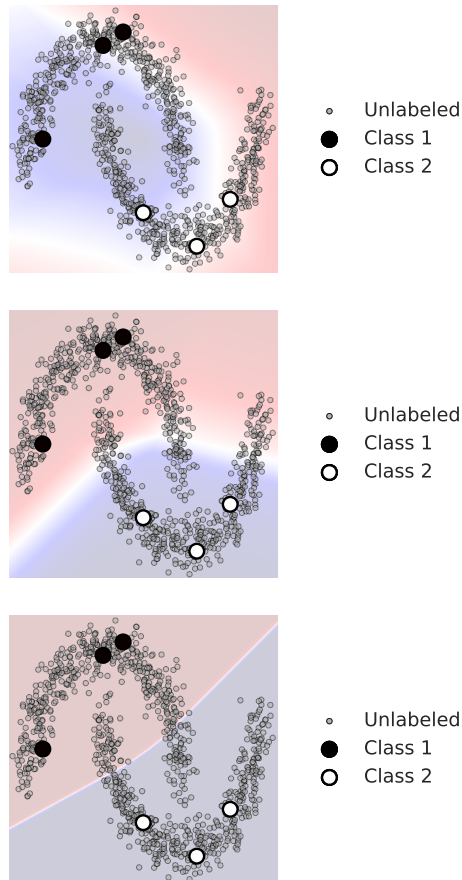
Figure 7: Predictions made by a model trained with Entropy Minimization, as made at initialization, and after 125 and 1000 training steps. Points where the model predicts "1" or "2" are shown in red or blue, respectively. Color saturation corresponds to prediction confidence, and the decision boundary is the white line. Notice that after 1000 steps of training the model is extremely confident at every point, which achieves close to zero prediction entropy on unlabeled points.

F Classes in ImageNet which overlap with CIFAR-10

Table 6: Classes in ImageNet which are similar to one of the classes in CIFAR-10. For reference, the CIFAR-10 classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

ID	Description	ID	Description	ID	Description	ID	Description
7	cock	152	Japanese spaniel	215	Brittany spaniel	283	Persian cat
8	hen	153	Maltese dog	216	clumber	284	Siamese cat
9	ostrich	154	Pekinese	217	English springer	285	Egyptian cat
10	brambling	155	Shih-Tzu	218	Welsh springer spaniel	286	cougar
11	goldfinch	156	Blenheim spaniel	219	cocker spaniel	287	lynx
12	house finch	157	papillon	220	Sussex spaniel	288	leopard
13	junco	158	toy terrier	221	Irish water spaniel	289	snow leopard
14	indigo bunting	159	Rhodesian ridgeback	222	kuvasz	290	jaguar
15	robin	160	Afghan hound	223	schipperke	291	lion
16	bulbul	161	basset	224	groenendael	292	tiger
17	jay	162	beagle	225	malinois	293	cheetah
18	magpie	163	bloodhound	226	briard	403	aircraft carrier
19	chickadee	164	bluetick	227	kelpie	404	airliner
20	water ouzel	165	black-and-tan coonhound	228	komondor	405	airship
21	kite	166	Walker hound	229	Old English sheepdog	408	amphibian
22	bald eagle	167	English foxhound	230	Shetland sheepdog	436	beach wagon
23	vulture	168	redbone	231	collie	466	bullet train
24	great grey owl	169	borzoi	232	Border collie	468	cab
30	bullfrog	170	Irish wolfhound	233	Bouvier des Flandres	472	canoe
31	tree frog	171	Italian greyhound	234	Rottweiler	479	car wheel
32	tailed frog	172	whippet	235	German shepherd	484	catamaran
80	black grouse	173	Ibizan hound	236	Doberman	510	container ship
81	ptarmigan	174	Norwegian elkhound	237	miniature pinscher	511	convertible
82	ruffed grouse	175	otterhound	238	Greater Swiss Mountain dog	554	fireboat
83	prairie chicken	176	Saluki	239	Bernese mountain dog	555	fire engine
84	peacock	177	Scottish deerhound	240	Appenzeller	569	garbage truck
85	quail	178	Weimaraner	241	EntleBucher	573	go-kart
86	partridge	179	Staffordshire bullterrier	242	boxer	575	golfcart
87	African grey	180	American Staffordshire terrier	243	bull mastiff	581	grille
88	macaw	181	Bedlington terrier	244	Tibetan mastiff	586	half track
89	sulphur-crested cockatoo	182	Border terrier	245	French bulldog	595	harvester
90	lorikeet	183	Kerry blue terrier	246	Great Dane	609	jeep
91	coucal	184	Irish terrier	247	Saint Bernard	612	jinrikisha
92	bee eater	185	Norfolk terrier	248	Eskimo dog	625	lifeboat
93	hornbill	186	Norwich terrier	249	malamute	627	limousine
94	hummingbird	187	Yorkshire terrier	250	Siberian husky	628	liner
95	jacamar	188	wire-haired fox terrier	251	dalmatian	654	minibus
96	toucan	189	Lakeland terrier	252	affenpinscher	656	minivan
97	drake	190	Sealyham terrier	253	basenji	661	Model T
98	red-breasted merganser	191	Airedale	254	pug	675	moving van
99	goose	192	cairn	255	Leonberg	694	paddlewheel
100	black swan	193	Australian terrier	256	Newfoundland	705	passenger car
127	white stork	194	Dandie Dinmont	257	Great Pyrenees	717	pickup
128	black stork	195	Boston bull	258	Samoyed	724	pirate
129	spoonbill	196	miniature schnauzer	259	Pomeranian	734	police van
130	flamingo	197	giant schnauzer	260	chow	751	racer
131	little blue heron	198	standard schnauzer	261	keeshond	757	recreational vehicle
132	American egret	199	Scotch terrier	262	Brabancon griffon	779	school bus
133	bittern	200	Tibetan terrier	263	Pembroke	780	schooner
134	crane	201	silky terrier	264	Cardigan	803	snowplow
135	limpkin	202	soft-coated wheaten terrier	265	toy poodle	814	speedboat
136	European gallinule	203	West Highland white terrier	266	miniature poodle	817	sports car
137	American coot	204	Lhasa	267	standard poodle	829	streetcar
138	bustard	205	flat-coated retriever	268	Mexican hairless	833	submarine
139	ruddy turnstone	206	curly-coated retriever	269	timber wolf	847	tank
140	red-backed sandpiper	207	golden retriever	270	white wolf	864	tow truck
141	redshank	208	Labrador retriever	271	red wolf	867	trailer truck
142	dowitcher	209	Chesapeake Bay retriever	272	coyote	871	trimaran
143	oystercatcher	210	German short-haired pointer	273	dingo	874	trolleybus
144	pelican	211	vizsla	274	dhole	895	warplane
145	king penguin	212	English setter	275	African hunting dog	908	wing
146	albatross	213	Irish setter	281	tabby	913	wreck
151	Chihuahua	214	Gordon setter	282	tiger cat	914	yawl