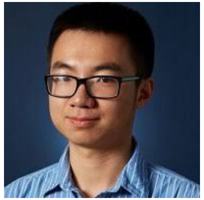
CS294-158 Deep Unsupervised Learning

Lecture 8b: Semi-Supervised Learning - Primer

Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas

UC Berkeley

Realistic Evaluation of Semi-Supervised Learning

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]

Outline

- Realistic Evaluation of Semi-Supervised Learning
  - pi-model
  - Temporal Ensembling
  - Mean Teacher
  - Virtual Adversarial Training

What is Semi-Supervised Learning?

$$(x, y) \sim p(x, y)$$

What is Semi-Supervised Learning?

$$(x, y) \sim p(x, y)$$

Supervised Learning

$$\max \quad E_{(x,y)\sim p(x,y)}\left[p(y|x)\right]$$

What is Semi-Supervised Learning?

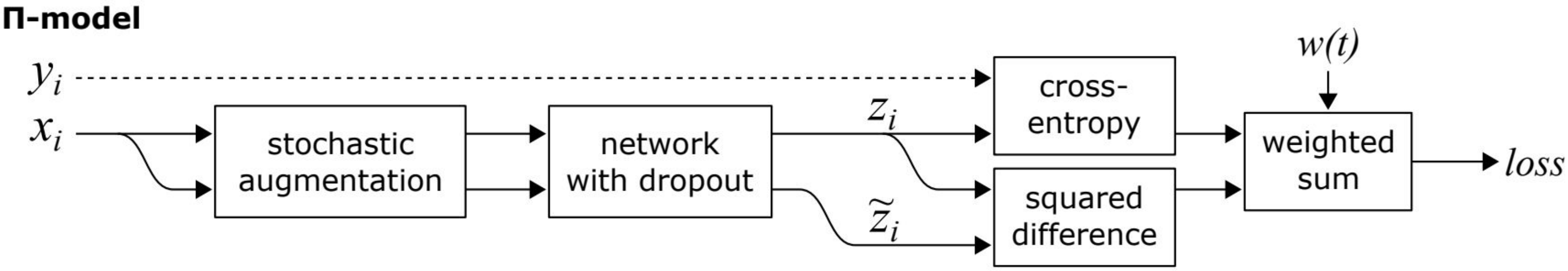
$$(x, y) \sim p(x, y)$$

Supervised Learning

$$\max \quad E_{(x,y)\sim p(x,y)} \left[p(y|x)\right]$$

$$D_U : x \sim p(x), D_S : (x, y) \sim p(x, y)$$

pi-Model



[Temporal Ensembling for Semi-Supervised Learning]

pi-Model

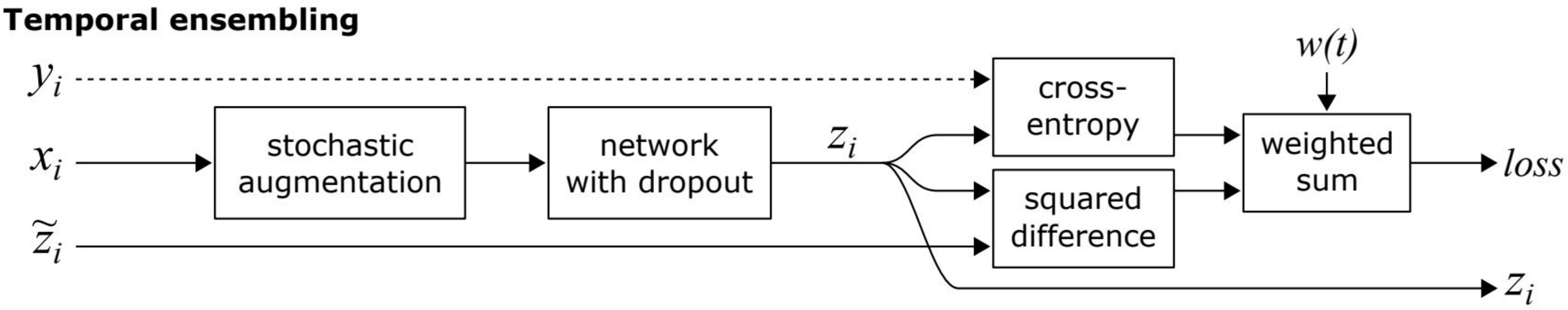
Algorithm 1 Π-model pseudocode.

Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function

  for t in $[1, num\_epochs]$ **do**
    for each minibatch B **do**
      $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ evaluate network outputs for augmented inputs
      $\tilde{z}_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ again, with different dropout and augmentation
      $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$ ▷ supervised loss component
          $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
      update θ using, e.g., ADAM ▷ update network parameters
    end for
  end for
  return θ

Temporal Ensembling for Semi-Supervised Learning

Temporal Ensembling



[Temporal Ensembling for Semi-Supervised Learning]

Temporal Ensembling

Algorithm 2 Temporal ensembling pseudocode. Note that the updates of Z and $\tilde{z}$ could equally well be done inside the minibatch loop; in this pseudocode they occur between epochs for clarity.

Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: α = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function

```
Z ← 0_[N×C]                                          ▷ initialize ensemble predictions
z̃ ← 0_[N×C]                                          ▷ initialize target vectors
for t in [1, num_epochs] do
    for each minibatch B do
        z_{i∈B} ← f_θ(g(x_{i∈B}, t))                 ▷ evaluate network outputs for augmented inputs
        loss ← − 1/|B| Σ_{i∈(B∩L)} log z_i[y_i]      ▷ supervised loss component
                + w(t) 1/(C|B|) Σ_{i∈B} ||z_i − z̃_i||²  ▷ unsupervised loss component
        update θ using, e.g., ADAM                   ▷ update network parameters
    end for
    Z ← αZ + (1 − α)z                                ▷ accumulate ensemble predictions
    z̃ ← Z/(1 − α^t)                                  ▷ construct target vectors by bias correction
end for
return θ
```

Where:

$$Z \leftarrow \mathbf{0}_{[N\times C]}, \quad \tilde{z} \leftarrow \mathbf{0}_{[N\times C]}$$

$$z_{i\in B} \leftarrow f_\theta(g(x_{i\in B}, t))$$

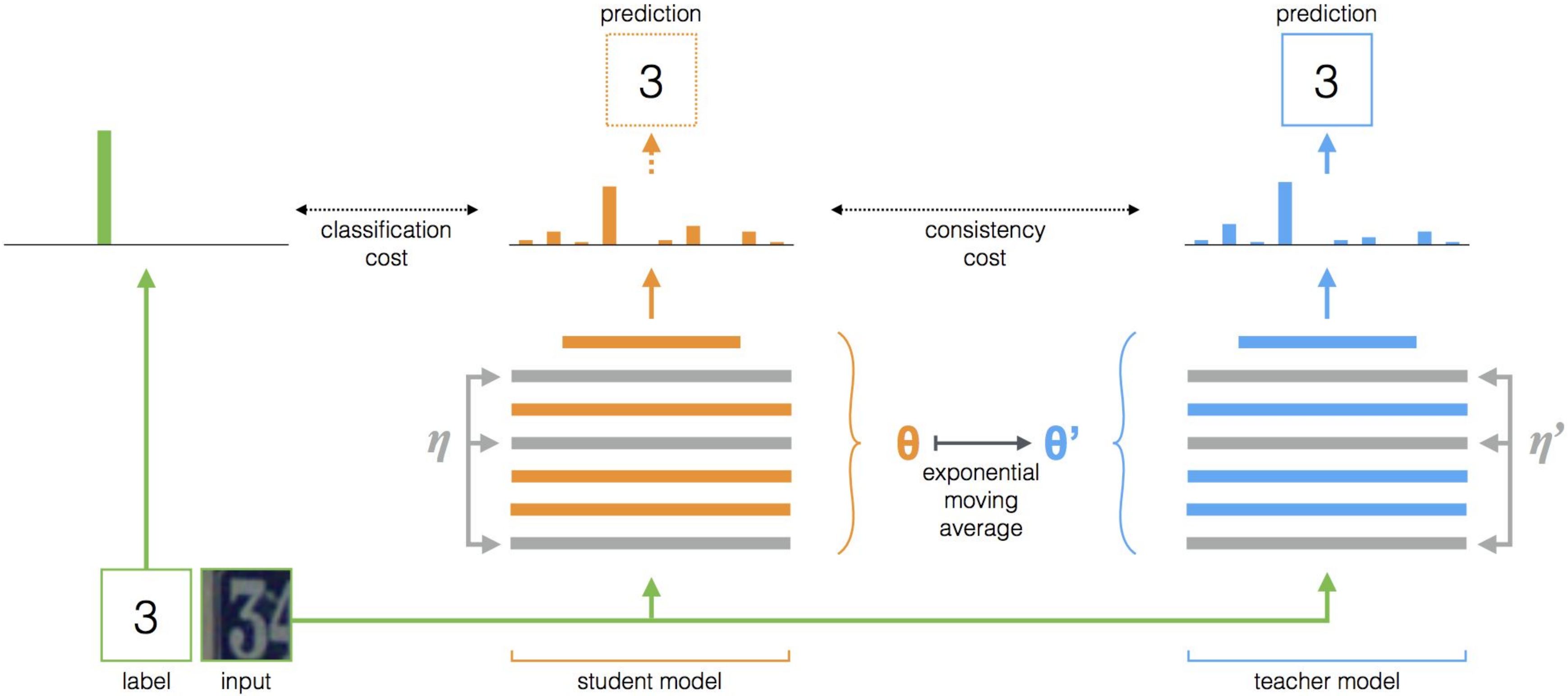
$$loss \leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap L)} \log z_i[y_i] + w(t)\frac{1}{C|B|}\sum_{i\in B} ||z_i - \tilde{z}_i||^2$$

$$Z \leftarrow \alpha Z + (1-\alpha)z, \quad \tilde{z} \leftarrow Z/(1-\alpha^t)$$

Temporal Ensembling for Semi-Supervised Learning

Mean Teacher

Mean Teachers are better role models



Virtual Adversarial Training

$$\boldsymbol{r}_{\text{adv}} = -\epsilon \boldsymbol{g} / \|\boldsymbol{g}\|_2 \text{ where } \boldsymbol{g} = \nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}; \hat{\boldsymbol{\theta}})$$

$$\boldsymbol{r}_{\text{v-adv}} = \underset{\boldsymbol{r}, \|\boldsymbol{r}\| \leq \epsilon}{\arg\max} \, \mathrm{KL}[p(\cdot \mid \boldsymbol{x}; \hat{\boldsymbol{\theta}}) \| p(\cdot \mid \boldsymbol{x} + \boldsymbol{r}; \hat{\boldsymbol{\theta}})]$$

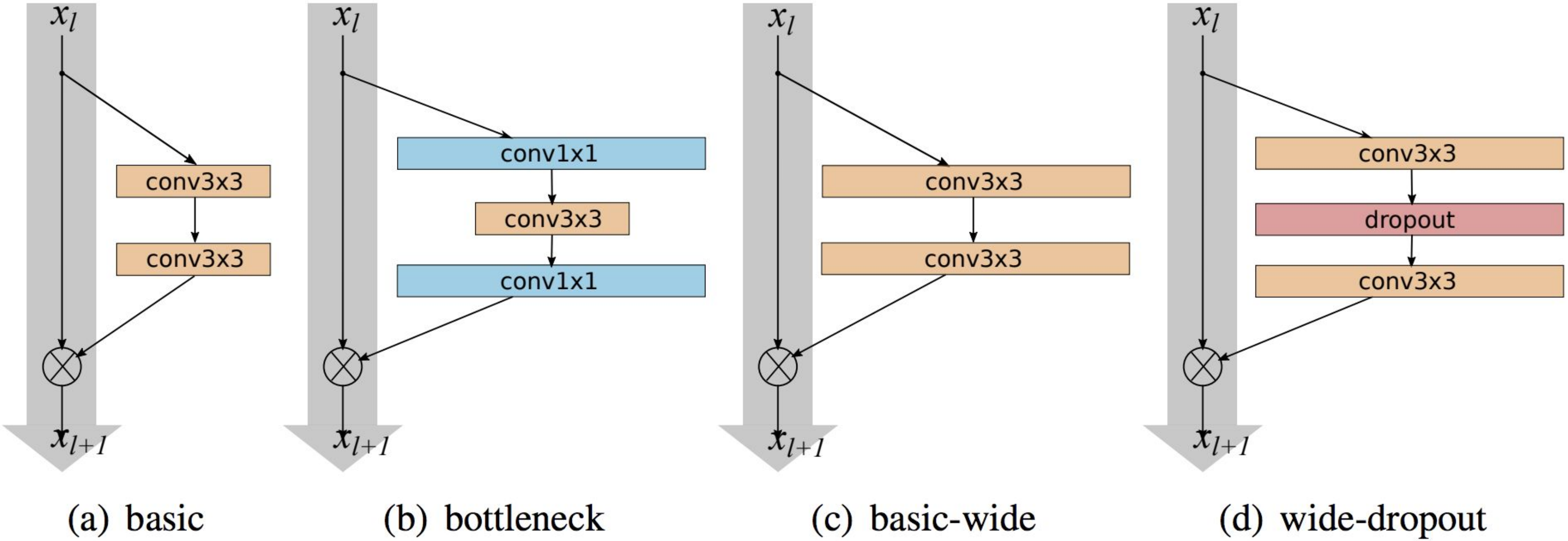
Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning

Virtual Adversarial Training

Algorithm 1 Mini-batch SGD for $\nabla_\theta \mathcal{R}_{\text{vadv}}(\theta)|_{\theta=\hat{\theta}}$, with a one-time power iteration method.

1) Choose M samples of $x^{(i)} (i = 1, \ldots, M)$ from dataset $\mathcal{D}$ at random.
2) Generate a random unit vector $d^{(i)} \in R^I$ using an iid Gaussian distribution.
3) Calculate r_{vadv} via taking the gradient of D with respect to r on $r = \xi d^{(i)}$ on each input data point $x^{(i)}$:
$$g^{(i)} \leftarrow \nabla_r D\left[p(y|x^{(i)}, \hat{\theta}), p(y|x^{(i)} + r, \hat{\theta})\right]\Big|_{r=\xi d^{(i)}},$$
$$r_{\text{vadv}}^{(i)} \leftarrow g^{(i)} / \|g^{(i)}\|_2$$
4) **Return**
$$\nabla_\theta \left(\frac{1}{M} \sum_{i=1}^{M} D\left[p(y|x^{(i)}, \hat{\theta}), p(y|x^{(i)} + r_{\text{vadv}}^{(i)}, \theta)\right] \right) \Bigg|_{\theta=\hat{\theta}}$$

Wide ResNet



Wide Residual Networks

Comparison

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

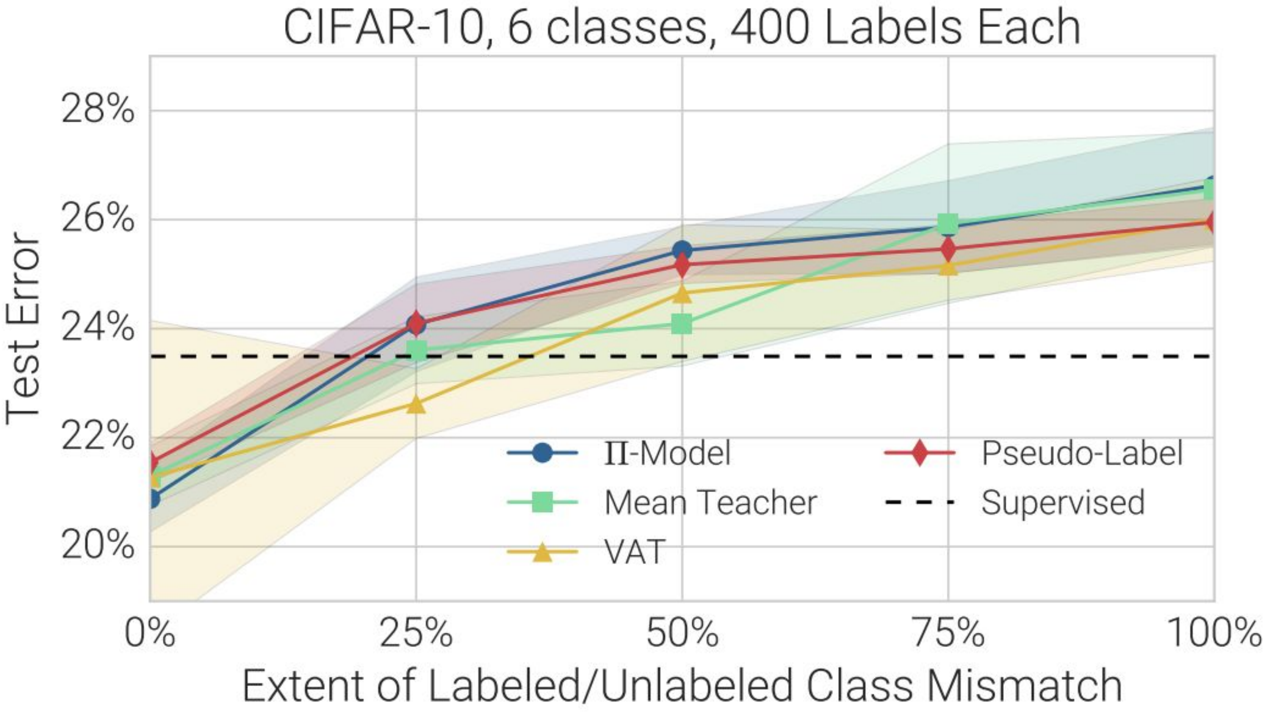
Comparison

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

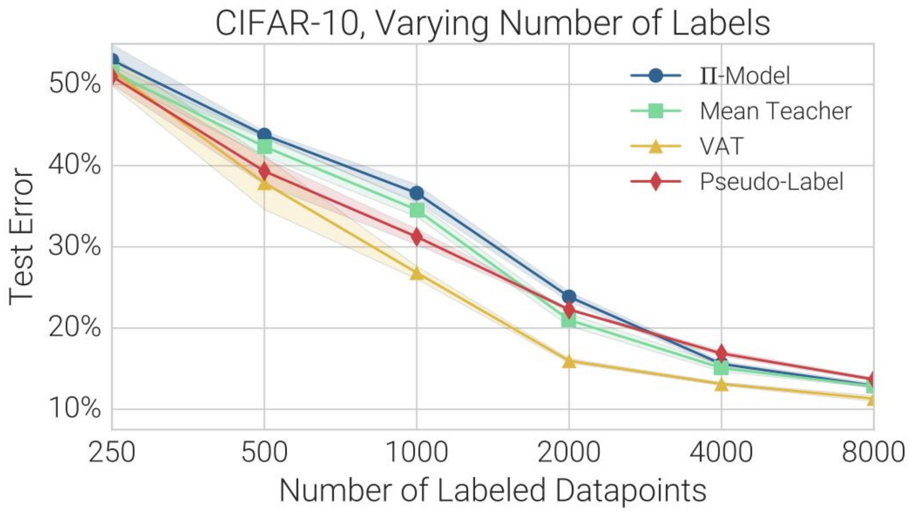
Comparison

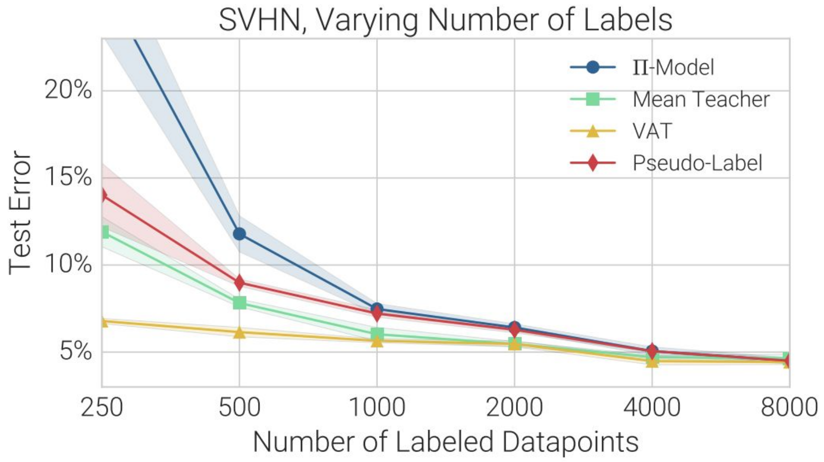
Method	CIFAR-10 4000 Labels
VAT with Entropy Minimization	13.13%
ImageNet $\rightarrow$ CIFAR-10	12.09%
ImageNet $\rightarrow$ CIFAR-10 (no overlap)	12.91%

Class Distribution Mismatch



Varying number of labels





Lessons

- Standardized architecture + equal budget for tuning hyperparameters
- Unlabeled data from a different class distribution not that useful
- Most methods don’t work well in the very low labeled-data regime
- Transferring Pre-Trained Imagenet produces lower error rate
- Conclusions based on small datasets though