FixMatch

Sohn, Berthelot, *et al*, Google Research

July 30, 2020

What is artificial labeling?

Approach 1: Pseudo-labeling

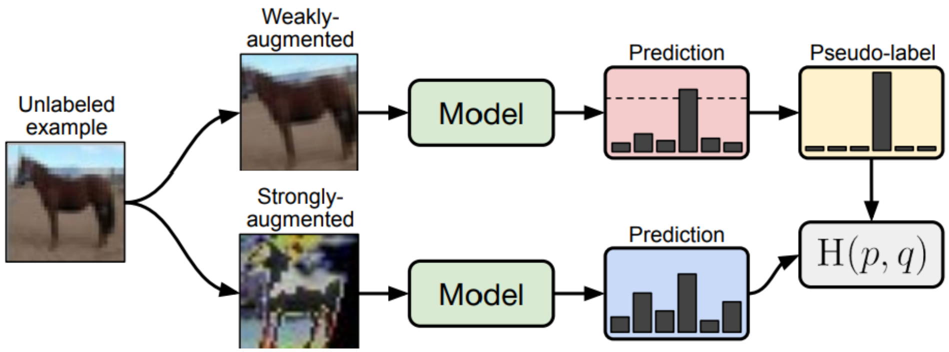
- Take a training model that you would use for fully-labeled examples and throw in some unlabeled images.
- At every weights update, assign the class label for the class with the maximum probability

Approach 2: Consistency regularization

- Sajjadi *et al*: "Given any training sample, a model's prediction should be the same under any random transformation of the data and perturbations to the model"
- So if you run the model multiple times, disturbing the dropout techniques, randomized pooling schemes, choosing which hidden nodes are activated, which class label is most frequently activated?

Approach 3: FixMatch: find a way to use both approaches

What are the basic elements of FixMatch?



- Base the label on a weakly-augmented (only flip-and-shift) image as a target
- Feed the model a strongly-augmented image, using UDA, ReMixMatch, CutOut, CTAugment, RandAugment

An artificial label is only given if there is high probability for one of the classes. Then the model is trained to require consistency with the strongly-augmented image's prediction.

Notation

- L classes
- A batch of labeled examples $\chi = \{(x_b, p_b) : b \in (1, ..., B)\}$ where x_b is the training input and $p_b = (0, 0, ..., 0, 1, 0, ...)$ is a one-hot label vector
- An unlabeled batch $\mathcal{U} = \{u_b : b \in (1, ..., \mu B)\}$ where the hyperparameter μ determines the relative batch sizes
- $p_m(y|x)$ is the predicted class distribution
- The cross-entropy between two probability distributions p and q is

$$H(p, q) = -\sum_{x \in \chi} p(x) \log q(x)$$

- Weak augmentation $\alpha(\cdot)$
- Strong augmentation $\mathcal{A}(\cdot)$

Background: Pseudo-labeling

Train the black-box model with a standard supervised classification loss for the labeled images.

- It's a very old idea
- Keep whatever the model gives the highest probability to for unlabeled data as long as it's above a certain threshold τ, so define $\hat{q}_b = \arg\max(q_b)$ as the one-hot classification that would result.
- This results in the following loss function

$$\text{loss}_{\text{unlabeled}} = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau) H(\hat{q}_b, q_b) \tag{2}$$

This formation encourages the model towards low entropy (or high confidence) on unlabeled data.

Background: consistency regularization originally

Model is trained with standard supervised classification loss and the sum of a stochastic loss term for each unlabeled image b:

$$\text{loss}_{\text{unlabeled}} = \sum_{b=1}^{\mu B} \|p_m(y|\alpha(u_b)) - p_m(y|\alpha(u_b))\|_2^2 \tag{1}$$

This isn't a typo. Both α and p_m are stochastic and will yield different values. But what does this really mean? It's really just summing the squared difference between two prediction vectors given the stochastic model p_m and the stochastic weak augmentation of each unlabeled image.

Background: alternative formulations for consistency regularization

$$\text{loss}_{\text{unlabeled}} = \sum_{b=1}^{\mu B} \|p_m(y|\alpha(u_b)) - p_m(y|\alpha(u_b))\|_2^2 \tag{1}$$

- Replace the weak augmentation α with an adversarial augmentation or a stronger augmentation
- Rather than using two fresh stochastic implementations of the model for the prediction vector, replace one with a running average or past prediction
- replace the ℓ_2-like loss with cross-entropy
- Use the consistency regularization within a larger SSL pipeline

Introducing FixMatch

For the labeled images, apply cross-entropy to the weakly augmented images (so p_m is the known one-hot classification vector):

$$\ell_S = \frac{1}{B}\sum_{b=1}^{B} H(p_b, p_m(y|\alpha(x_b))) \tag{3}$$

For unlabeled data (including stripped labels from above)

1. Find an artificial label by computing the model's predicted class distribution given a weakly-augmented version of the unlabeled image: $q_b = p_m(y|\alpha(u_b))$ and use the one-hot max for the $\hat{q}_b$ as before
2. Then enforce the cross-entropy loss against the model's output for a strongly-augmented form of u_b:

$$\ell_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) > \tau) H(\hat{q}_b, p_m(y|\mathcal{A}(u_b))) \tag{4}$$

FixMatch Loss

$$\ell = \ell_S + \lambda_u \ell_u$$

In other implementations of artificial labeling, the weight λ_u is typically increased as the training progresses; however, that was not needed for FixMatch possibly because of the threshold not being met early on. As the model progresses, there are fewer cases that don't meet the threshold τ. This serves as a justification to ignore the low-threshold guesses in early training.

More about augmentation

Weak augmentation is just flipping randomly (except house number pics) and translating a little

Strong augmentation: they use Cutout (like dropout but at the input state) followed by either RandAugment or CTAugment

RandAugment

- You feed RandAugment a list of transformations – perhaps color inversion, translation, contrast, etc. and it randomly applies one of those to each image in the minibatch.
- There's a hyperparameter that controls the magnitude of the disturbance.
- The authors randomly selected this hyperparameter from a fixed range at each training step for best results.

CTAugment

- Rather than choosing random magnitudes during training, CTAugment learns the optimal magnitudes.
- A wide range of values are binned with each bin's weight initialized at 1.
- The weight determines the likelihood that the bin is chosen at each training step.
- Weights are updated by best performance on labeled samples (two choices used for each labeled example).

Other important factors

- Low-label regimes depend heavily on the amount of regularization
- Deep learning is always dependent on architecture, optimizer, training schedule...

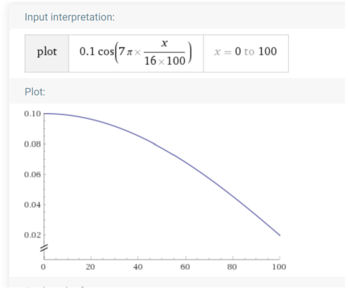
So when you justify a new SSL algorithm without considering these other pieces, you're missing critical information.

Approaches here

The authors used weight decay regularization, and found that the Adam optimizer resulted in worse performance and instead used the SGD with momentum. They used a cosine learning rate decay:

$$\text{learning rate} = \eta \cos\left(\frac{7\pi k}{16K}\right) \tag{5}$$

where η is the initial learning rate, k is the current step, and K is the number of training steps.


Input interpretation:

plot $0.1\cos\left(7\pi\times\frac{x}{16\times 100}\right)$ $x = 0$ to 100

Plot:

Background: optimization schemes (Wiki thanks)

Objective: minimize an objective function of this form

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w)$$

Stochastic gradient descent: update after each training example

$$w := w - \eta \nabla Q_i(w)$$

For this algorithm to converge, some clever manipulations are typically needed for the training rate η

Stochastic gradient ascent with momentum: Remember the update Δw at each training step, and use a linear combination of the new update and the prior. The α below is the **loss coefficient for weight decay regularization**.

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w$$

Background: optimization schemes Slide 2 (Wiki thanks)

Averaged stochastic gradient descent: Calculate the running average weight over all training,

$$\bar{w} = \frac{1}{t} \sum_{i=0}^{t-1} w_i$$

and after all training examples replace the weight vector with this average.

AdaGrad (Adaptive gradient) Algorithm: Works will with space data and parameters. The formulation is complex, but has the result that parameters with few updates receive the greatest learning rates. Setting: optimization of stochastic objections with high-dimensional parameter spaces

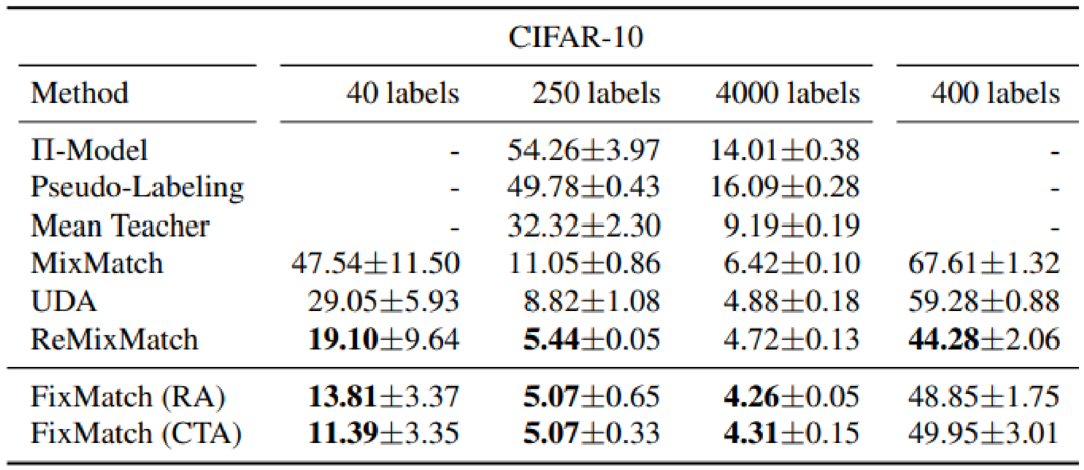
Background: optimization schemes Slide 3

Adam (Adaptive moment estimation): running averages of the gradients and the second moments (approx variance?) of the gradients used:

- Initialize first and second moments, m_0 and v_0 at 0. Set forgetting rates $\alpha = 0.0001$, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$
- Until convergence, at each time step do:
  - $g_t \leftarrow \nabla_w Q_t(w_{t-1})$ (gradients)
  - $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t$ (biased first moment estimate)
  - $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (biased second moment estimate)
  - $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ (bias-corrected)
  - $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ (bias-corrected)
  - $w_t \leftarrow w_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$

Ablation study

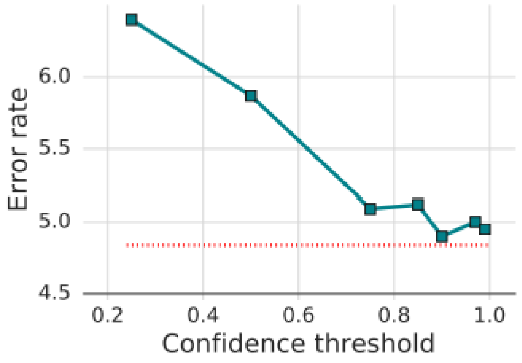
Starting point: 4.84% error rate on a *particular* 250 label split from the CIFAR-10 using CTAugment.



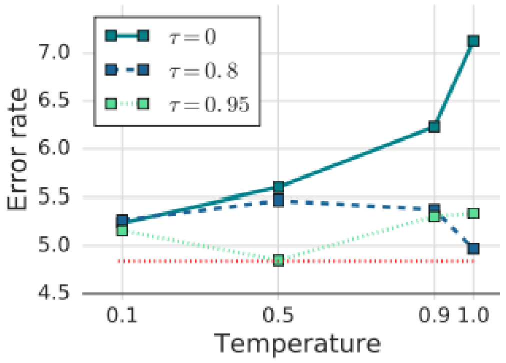
	CIFAR-10			
Method	40 labels	250 labels	4000 labels	400 labels
Π-Model	-	54.26±3.97	14.01±0.38	-
Pseudo-Labeling	-	49.78±0.43	16.09±0.28	-
Mean Teacher	-	32.32±2.30	9.19±0.19	-
MixMatch	47.54±11.50	11.05±0.86	6.42±0.10	67.61±1.32
UDA	29.05±5.93	8.82±1.08	4.88±0.18	59.28±0.88
ReMixMatch	**19.10**±9.64	**5.44**±0.05	4.72±0.13	**44.28**±2.06
FixMatch (RA)	**13.81**±3.37	**5.07**±0.65	**4.26**±0.05	48.85±1.75
FixMatch (CTA)	**11.39**±3.35	**5.07**±0.33	**4.31**±0.15	49.95±3.01

Ablation 1: Sharpening and Thresholding

Mess with the threshold, or don't replace your pseudo prediction with a one-hot – introduce a sharpening dependent on a temperature T. As $T \to 0$ the one-hot labeling is replaced. They say they found no better performance as long as the threshold is somewhere between 0.95 and 0.99.



(b)



(c)

Ablation 2: Augmentation

In some cases, CTAugmentation worked better than random, but not always.

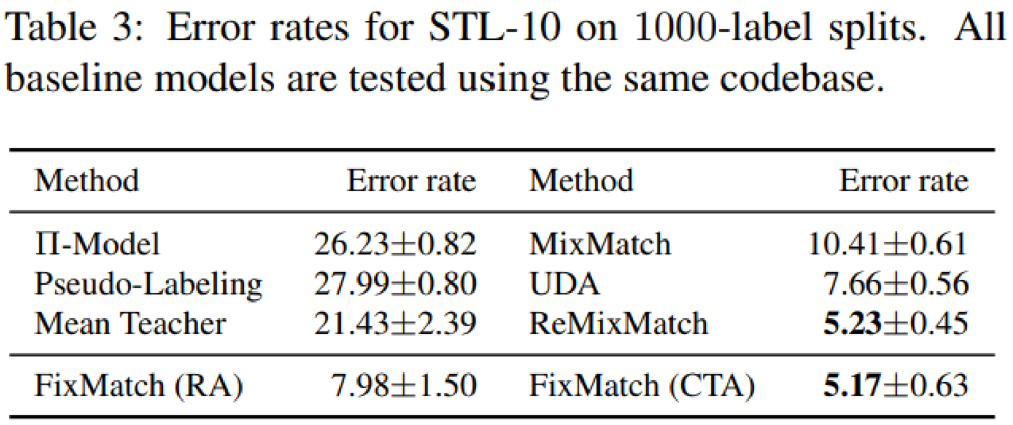


Table 3: Error rates for STL-10 on 1000-label splits. All baseline models are tested using the same codebase.

Method	Error rate	Method	Error rate
Π-Model	26.23±0.82	MixMatch	10.41±0.61
Pseudo-Labeling	27.99±0.80	UDA	7.66±0.56
Mean Teacher	21.43±2.39	ReMixMatch	**5.23**±0.45
FixMatch (RA)	7.98±1.50	FixMatch (CTA)	**5.17**±0.63

Ablation 3: CutOut

Interestingly, CutOut without augmentation or augmentation alone had the same result which was worse than CutOut with Augmentation.

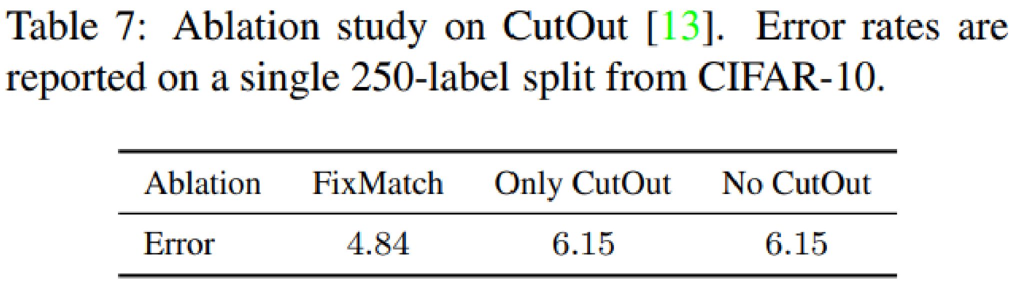


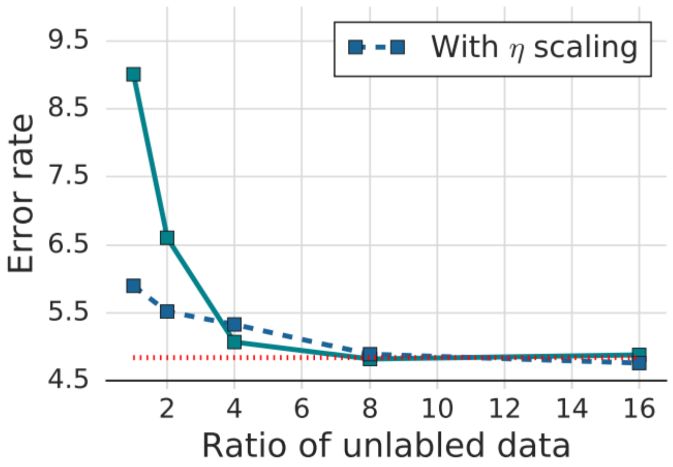
Table 7: Ablation study on CutOut [13]. Error rates are reported on a single 250-label split from CIFAR-10.

Ablation	FixMatch	Only CutOut	No CutOut
Error	4.84	6.15	6.15

Ablation 4: Weak/strong augmentation

- Weak $\rightarrow$ strong for artificial labels leads to divergence
- Strong $\rightarrow$ weak leads to unstable performance with poor accuracy

Ablation 5: How much unlabeled data?



Recall that if B is the batch size of the labeled data, then μB is batch size of the unlabeled data. The ratio they are plotting is

$$\mu = \frac{\text{unlabeled}}{\text{labeled}}$$

and in the dotted line the learning rate scales linearly with the batch size. I don't follow the claim that error decreases with *more* unlabeled data. That doesn't agree with their results. What am I missing?

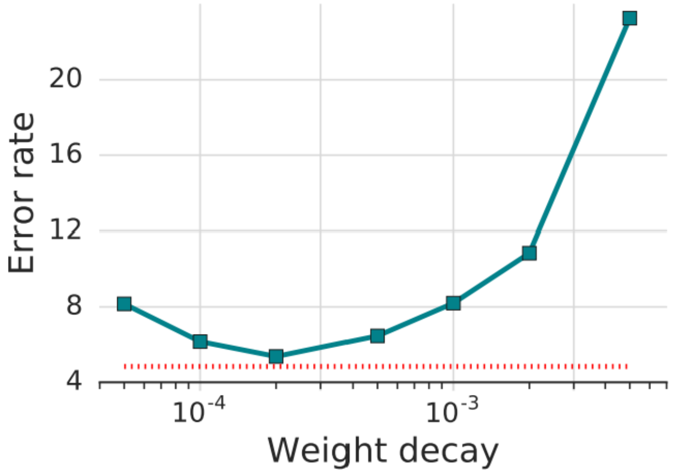
Ablation 6: Optimization and learning rates

They justify the choices already stated – also they say linear learning rate decay worked as well as the cosine we already talked about.

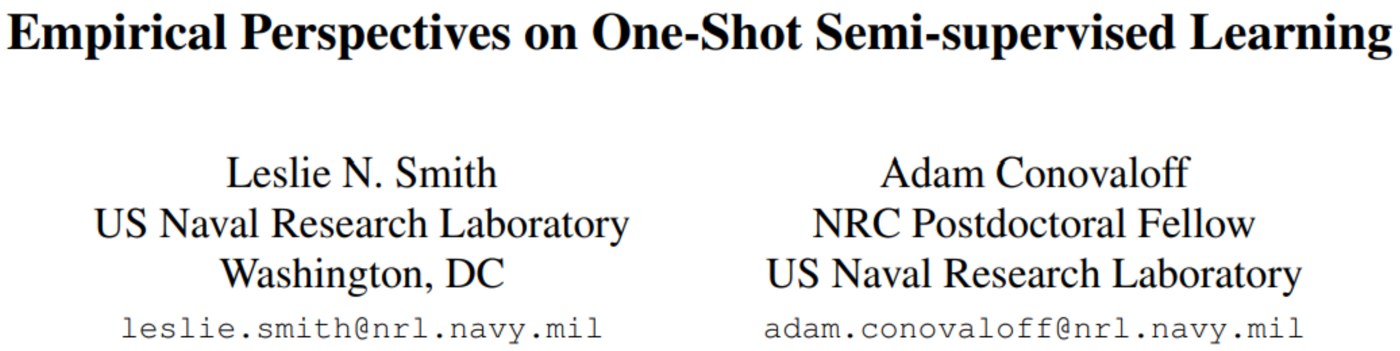
Ablation 7: Weight decay is crucial

Recall: **Stochastic gradient ascent with momentum**: Remember the update Δw at each training step, and use a linear combination of the new update and the prior. The α below is the **loss coefficient for weight decay regularization**.

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w$$



Experiments building on FixMatch

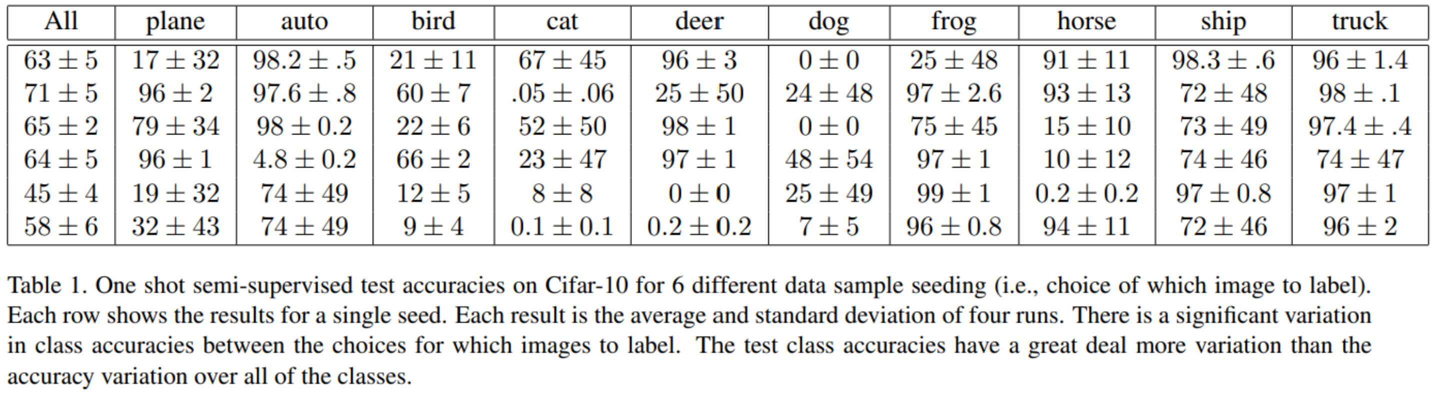


Empirical Perspectives on One-Shot Semi-supervised Learning

Leslie N. Smith
US Naval Research Laboratory
Washington, DC
leslie.smith@nrl.navy.mil

Adam Conovaloff
NRC Postdoctoral Fellow
US Naval Research Laboratory
adam.conovaloff@nrl.navy.mil

Six implementations using 4 runs each show very different performance, especially by class



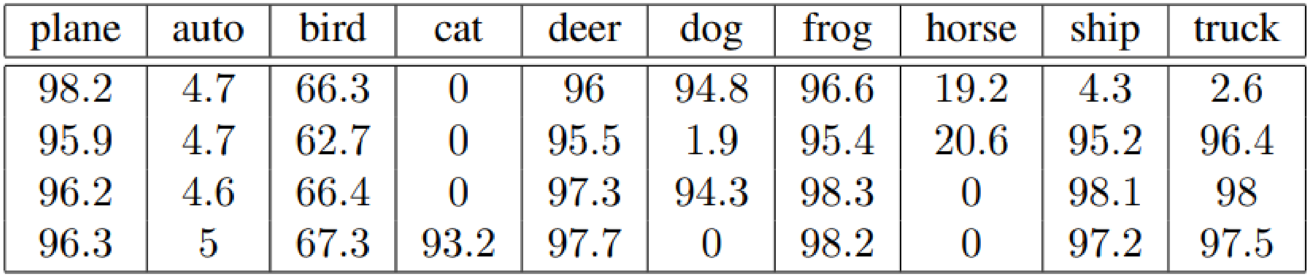
All	plane	auto	bird	cat	deer	dog	frog	horse	ship	truck
63 ± 5	17 ± 32	$98.2 \pm .5$	21 ± 11	67 ± 45	96 ± 3	0 ± 0	25 ± 48	91 ± 11	$98.3 \pm .6$	96 ± 1.4
71 ± 5	96 ± 2	$97.6 \pm .8$	60 ± 7	$.05 \pm .06$	25 ± 50	24 ± 48	97 ± 2.6	93 ± 13	72 ± 48	$98 \pm .1$
65 ± 2	79 ± 34	98 ± 0.2	22 ± 6	52 ± 50	98 ± 1	0 ± 0	75 ± 45	15 ± 10	73 ± 49	$97.4 \pm .4$
64 ± 5	96 ± 1	4.8 ± 0.2	66 ± 2	23 ± 47	97 ± 1	48 ± 54	97 ± 1	10 ± 12	74 ± 46	74 ± 47
45 ± 4	19 ± 32	74 ± 49	12 ± 5	8 ± 8	0 ± 0	25 ± 49	99 ± 1	0.2 ± 0.2	97 ± 0.8	97 ± 1
58 ± 6	32 ± 43	74 ± 49	9 ± 4	0.1 ± 0.1	0.2 ± 0.2	7 ± 5	96 ± 0.8	94 ± 11	72 ± 46	96 ± 2

Table 1. One shot semi-supervised test accuracies on Cifar-10 for 6 different data sample seeding (i.e., choice of which image to label). Each row shows the results for a single seed. Each result is the average and standard deviation of four runs. There is a significant variation in class accuracies between the choices for which images to label. The test class accuracies have a great deal more variation than the accuracy variation over all of the classes.

1. You'd best be sure to label the quintessential image (see dog versus cat with differing one-shot images)
2. In-class variance paints a very different picture from the overall measurement that has smoothed this out.

Now look at the variation within one example of 4 runs – the starting image alone doesn't explain variation

This is the break-out from row 4 above – so in these 4 runs we are always starting with the same picture



plane	auto	bird	cat	deer	dog	frog	horse	ship	truck
98.2	4.7	66.3	0	96	94.8	96.6	19.2	4.3	2.6
95.9	4.7	62.7	0	95.5	1.9	95.4	20.6	95.2	96.4
96.2	4.6	66.4	0	97.3	94.3	98.3	0	98.1	98
96.3	5	67.3	93.2	97.7	0	98.2	0	97.2	97.5

A look at convergence by class – first row

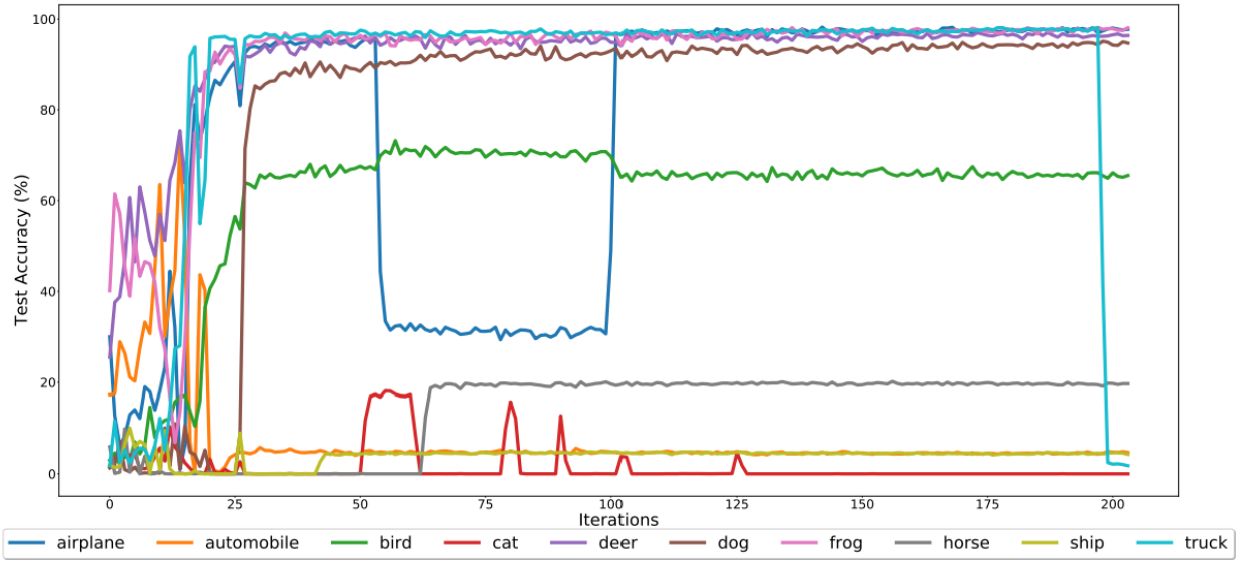


Figure 1. Class test accuracies on Cifar-10 during training with seed=3 (i.e., corresponds to the results displayed in the first row of Table 2).

A look at convergence by class – last row

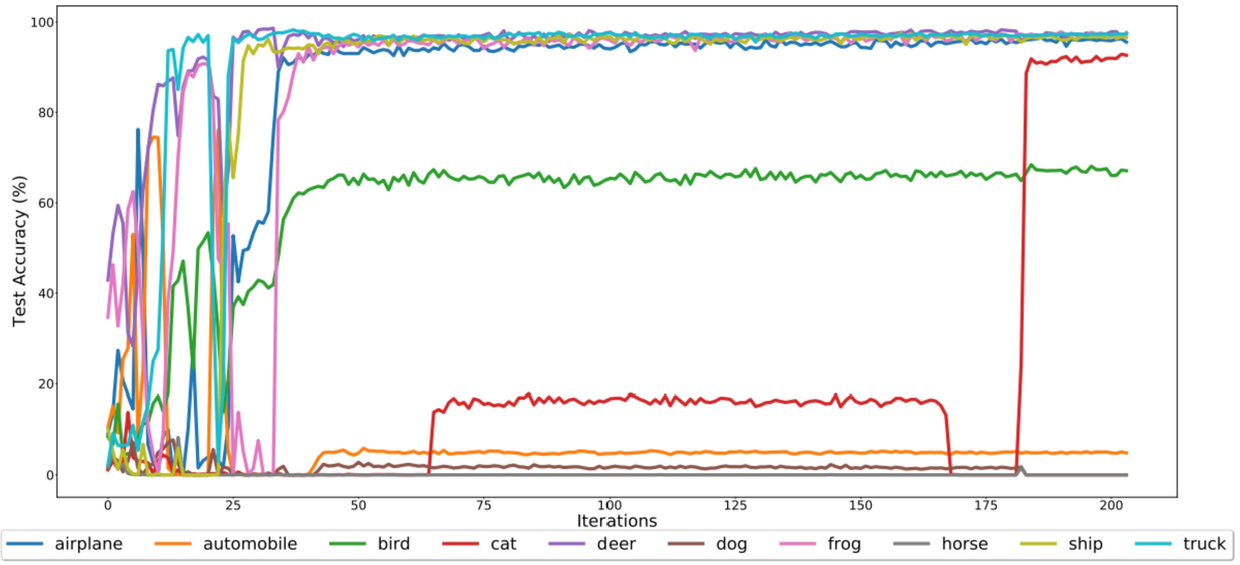
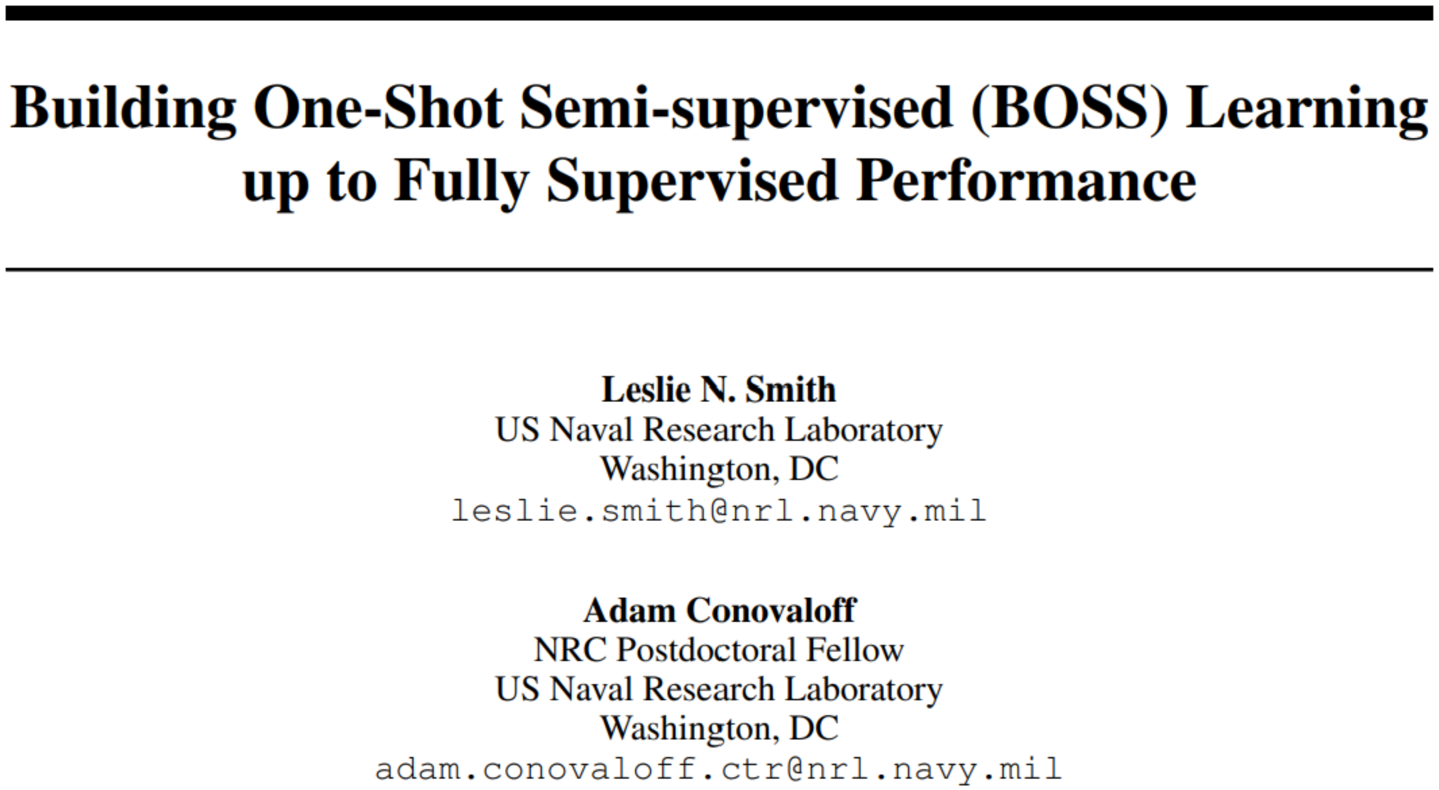


Figure 2. Another example of class test accuracies on Cifar-10 with seed=3 (i.e., corresponds to the results displayed in the last row of Table 2).

Coming next

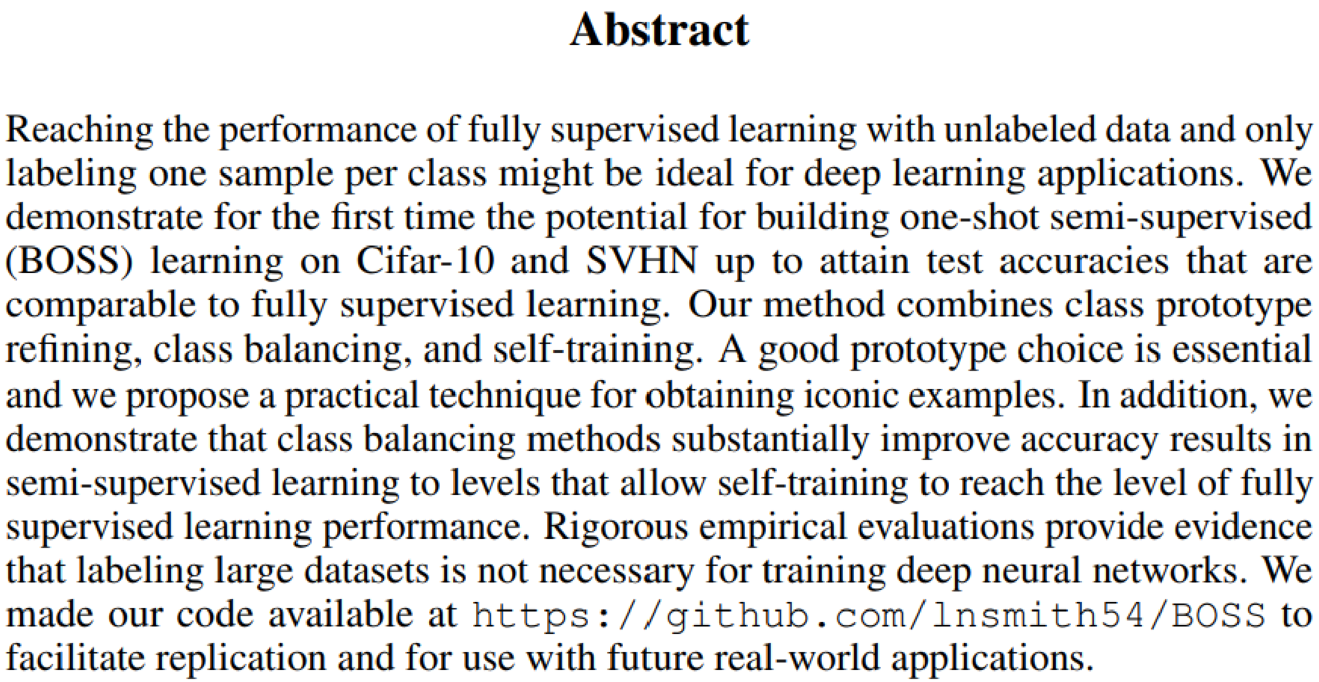


Building One-Shot Semi-supervised (BOSS) Learning up to Fully Supervised Performance

Leslie N. Smith
US Naval Research Laboratory
Washington, DC
leslie.smith@nrl.navy.mil

Adam Conovaloff
NRC Postdoctoral Fellow
US Naval Research Laboratory
Washington, DC
adam.conovaloff.ctr@nrl.navy.mil

Coming next



Abstract

Reaching the performance of fully supervised learning with unlabeled data and only labeling one sample per class might be ideal for deep learning applications. We demonstrate for the first time the potential for building one-shot semi-supervised (BOSS) learning on Cifar-10 and SVHN up to attain test accuracies that are comparable to fully supervised learning. Our method combines class prototype refining, class balancing, and self-training. A good prototype choice is essential and we propose a practical technique for obtaining iconic examples. In addition, we demonstrate that class balancing methods substantially improve accuracy results in semi-supervised learning to levels that allow self-training to reach the level of fully supervised learning performance. Rigorous empirical evaluations provide evidence that labeling large datasets is not necessary for training deep neural networks. We made our code available at `https://github.com/lnsmith54/BOSS` to facilitate replication and for use with future real-world applications.