Mean Teacher

Semi-Supervised Learning with Learned Regularization

Sam Spilsbury, Aalto University

Desert Island

A motivating Example

You take a holiday on a beach island.
You have 10,000 photos of coconut trees.

You want to classify them into
different types of trees.
But after labelling 100 of them,
you are tired.
Can you use the 100 labelled images and
99900 other images to label the rest?

Semi Supervised Learning

- Using the 100 images alone will probably overfit.
- Regularization on just 100 images will probably not save us.
- Maybe the other images can help.

Semi Supervised Learning

More formal definition

- We have a set S, with disjoint subsets L (the labelled set) and U the unlabelled set, such that $L \cup U \subset S$, but $L \not\subset U$.
- Oracle function $EX(x, L) = 0, \ldots, n$ (class labels).
- Oracle function $EX(x, U) = -1, \forall x \in U$
- Objective: Find some $h(x)$ using both $EX(x, L)$ and $EX(x, U)$ that minimizes the generalization error(h, D) on the true distribution.

Semi Supervised Learning

Deep Learning Hat

- Ideas
  - Autoencoders?
  - Variational Autoencoders?
  - GANs? [a]

[a] GAN's can be used for SSL! See https://paperswithcode.com/paper/improved-techniques-for-training-gans

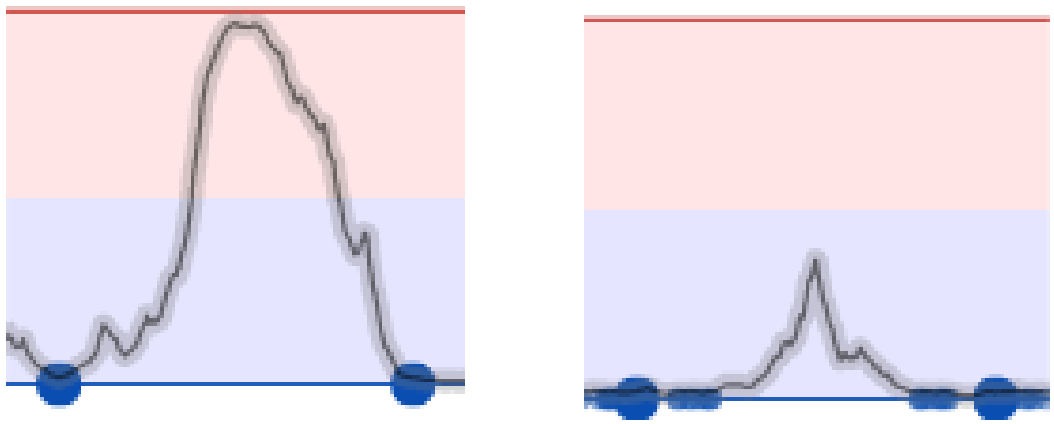
Semi Supervised Learning

Alternative

- We can still overfit to latent variable space ...
- Use U to learn to regularize h learned on L!

Regularization

A closer analysis

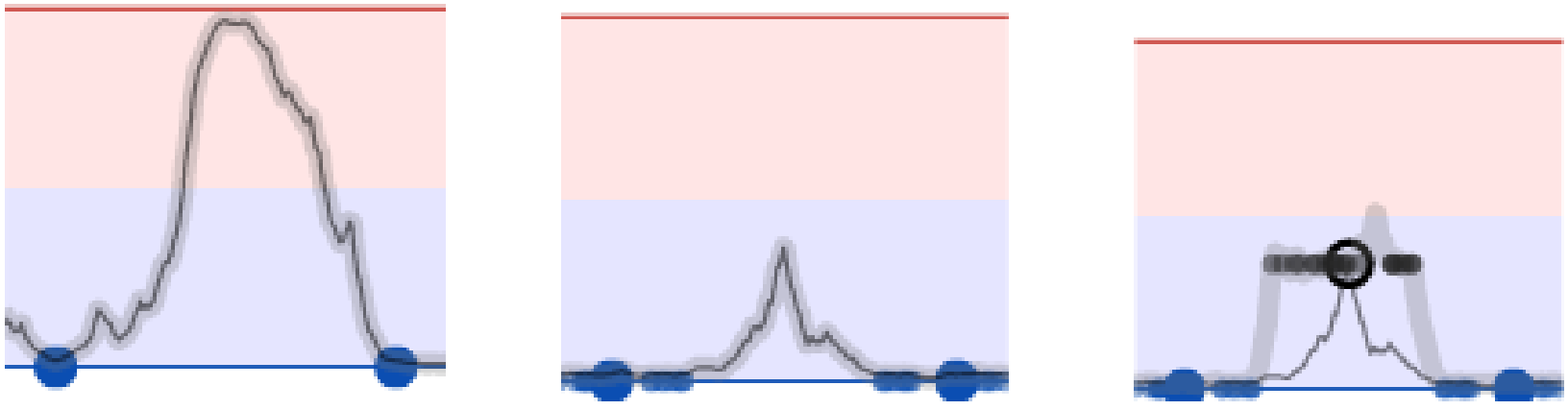


(a) Overfitting (b) Regularization

Figure: Effect of Regularization

Regularization

Regularizing using unlabelled data

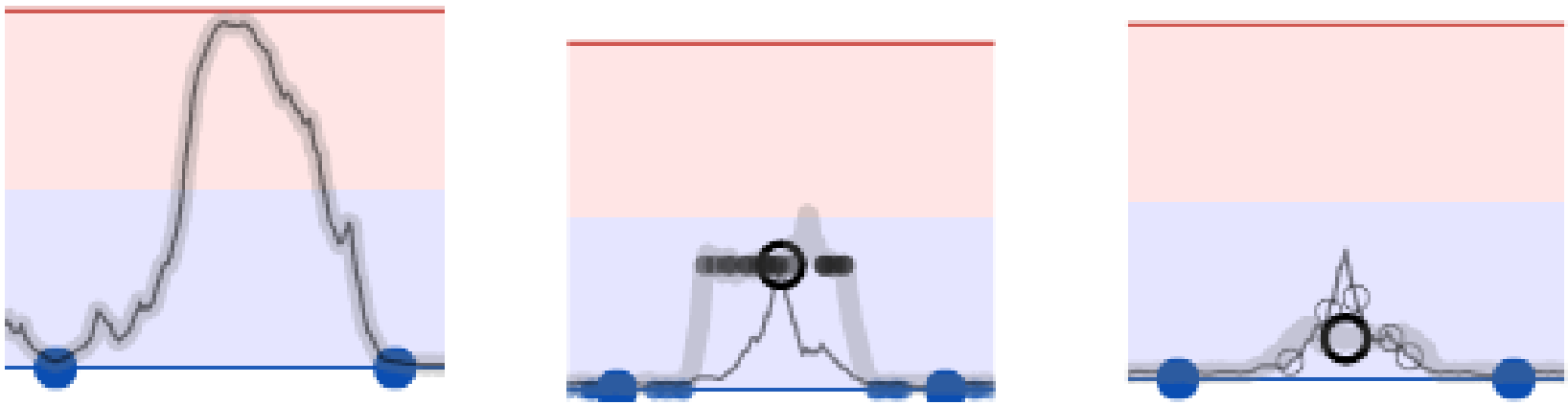


(a) Overfitting (b) Regularization (c) Consistency Cost

Figure: Just using the "unlabelled data" may end up introducing errors due to bad predictions

Regularization

Regularizing using unlabelled data



(a) Overfitting

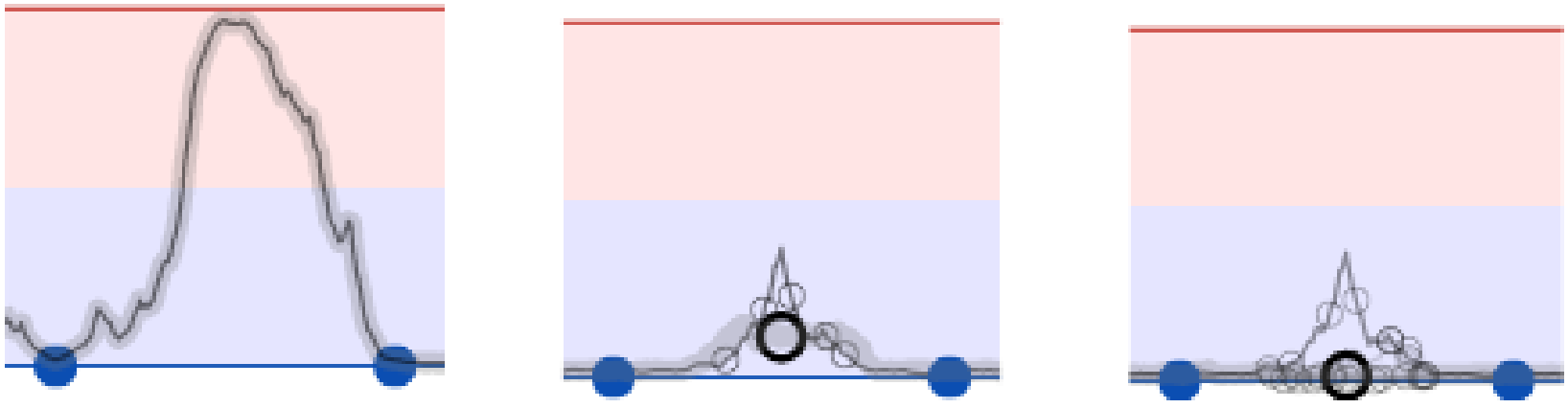
(b) Consistency Cost

(c) Regularized Consistency Cost

Figure: We can try to regularize the "teacher" that tries to predict targets for the unlabelled data

Regularization

Regularizing using unlabelled data



(a) Overfitting

(b) Regularized Consistency Cost

(c) Ensemble Approach

Figure: By using an ensemble of models we can regularize even more

Regularization

What have we learned?

- Regularization is a good thing
- Taking into account our unlabelled data can be useful
- Ensemble models have a regularizing effect

Teacher / Student Models

Making use of what we have learned

- Two "models", "teacher" T and "student" S[a]
- Ideally, T is more *conservative* than S, eg, it is more regularized via noise, dropout or some other mechanism.
- S is more *liberal* than T, in that it does not have as much regularization and can learn more about the limited amount of data.
- But, via *consistency cost* $C(T(x), S(x))$, S pays a penalty if it strays too far from T.

[a] They can be the same network, but we require that $T(x) \neq S(x)$

Teacher / Student Models

Prior Art

- Π model (Finnish Invention). [RBH+15]
  - Sample from network twice, with some noise mechanism $S(x) = N(x) + \epsilon_1$, $T(x) = N(x) + \epsilon_2$.
  - Cost is Classification cost plus weighted consistency cost $\text{loss}(S(x), y) + w(t)C(S(x), T(x))$
  - Classification cost for unlabelled examples $(y = -1)$ is defined to be zero.
  - Consistency cost weight $w(t)$ increases as time goes on, as training continuous we care more about being consistent on the training data S than being right on the labelled data L.

Teacher / Student Models

Π *model*, *PyTorch*

```
model = DropoutMLP()
criterion = CrossEntropyLoss(ignore_index=-1)
consistency = MSELoss()

for epoch in range(epochs):
  for label, data in loader:
    out = model(data)
    again = model(data)

    assert out != again

    w = weight(epoch)
    loss = criterion(out, label) + w * consistency(out, again)
    loss.backward()
    optimizer.step()
```

Teacher / Student Models

Prior Art

- Problem: $T(x)$ is stochastic, it would be nicer to use an ensemble and incorporate all our regularization knowledge
- Enter *Temporal Ensembling* (Another Finnish Invention). [LA17]

Teacher / Student Models

Temporal Ensembling

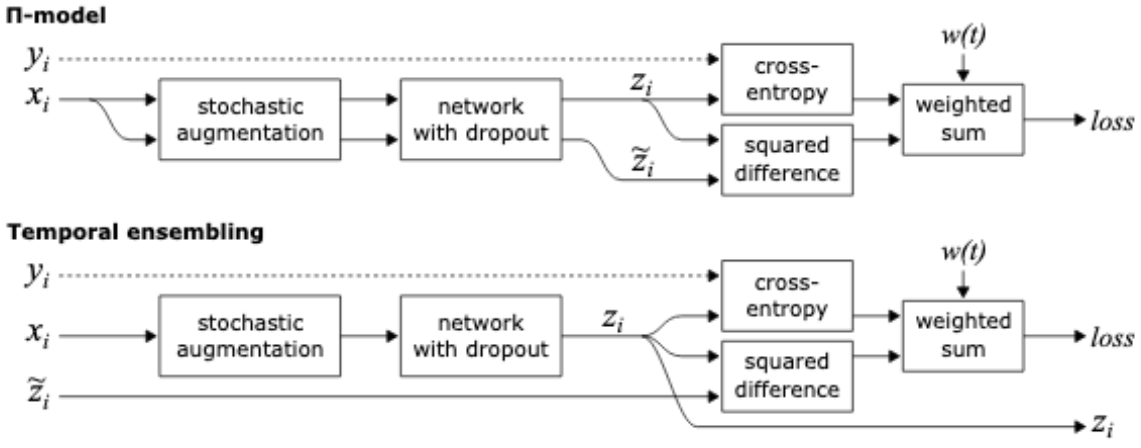


Figure: Π model vs Temporal Ensembling

Teacher / Student Models

Temporal Ensembling

- Similar idea for loss, classification cost plus consistency cost:
  $\text{loss}(S(x), y) + w(t)C(S(x), T(x))$
- However, use an ensemble of consistency targets, eg, an exponential
  moving average of ensemble predictions.
- In practice, we store the "mean prediction" for a given data point, eg
  $Z_i = \alpha Z_i + (1 - \alpha)\bar{z}_i$, where $\bar{z}_i$ is the incoming prediction from S.

Teacher / Student Models

Temporal Ensembling model, PyTorch

```
model = DropoutMLP()
criterion = CrossEntropyLoss(ignore_index=-1)
consistency = MSELoss()
Z = torch.tensor(
  np.random.uniform(size=(n_training_points, n_classes))
)

for epoch in range(epochs):
  for i, (label, data) in enumerate(loader):
    out = model(data)
    assert out != again
    w = weight(epoch)
    loss = criterion(out, label) + w * consistency(out, Z[i])
    loss.backward()
    optimizer.step()
    Z[i] = alpha * Z[i] + (1 - alpha) * out
```

Teacher / Student Models

Temporal Ensembling

- Benefits
  - Only need to evaluate T once per input on each epoch, instead of sampling many times to get ensemble of noise
  - Target Z_i is less noisy than with Π model.
- Drawbacks
  - Need a "past" to compare with
  - Only see the ensemble information once per data point in the training set, therefore, once per epoch. This leads to slow training.

Mean Teacher

Basic Idea

- Finnish Invention! [TH]
- Measure consistency cost for student loss as before.
- Two networks, update teacher weights from student weights directly, instead of updating mean prediction.

Mean Teacher

Architecture

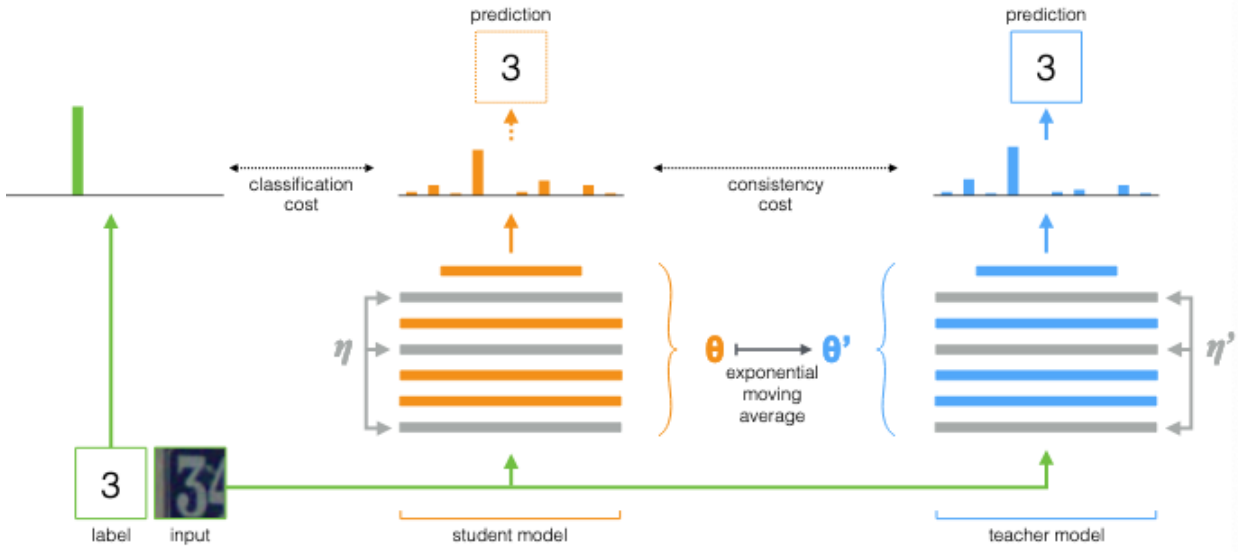


Figure: Mean Teacher Architecture

Mean Teacher

Mean Teacher, PyTorch

```
student = DropoutMLP()
teacher = DropoutMLP()
criterion = CrossEntropyLoss(ignore_index=-1)
consistency = MSELoss()

for epoch in range(epochs):
  for i, (label, data) in enumerate(loader):
    s_out, t_out = student(data), teacher(data)
    assert s_out != t_out
    w = weight(epoch)
    loss = criterion(out, label) + w *
        consistency(s_out, t_out) * (1 - exp(-25 * epoch ** 2))
    ...
    teacher.weight_ = alpha * teacher.weight_ +
      (1 - alpha) * student.weight_;
```

Mean Teacher

Why does this work?

- Evaluating consistency cost between predictions $C(S(x), T(x))$ is similar to evaluating consistency between top layer.
- Feature detection layers could still be overfitted on student!
- Mean teacher allows us to regularize all layers.
- New weights can be used right away on each batch, no need to wait entire epoch to utilize new information.

Decay Parameter

How to set α

- Too high and you trust the (probably bad) teacher model too much.
- Too low and you forget regularization information you learned in the past.
- Setting $\alpha = 0$ makes Mean Teacher into a variant of Π, but more inefficient because the Teacher never gets updated.
- Usually a good idea to vary α during training, eg, trust the student more at the start (exploration), but the teacher more at the end (regularization).
  - This is achieved by `(1 - exp(-25 * epoch ** 2))`, consistency cost schedule

Performance

Is this actually useful?

	CIFAR-10 (8% labels)	ImageNet 2012 (10% labels)
SOTA (SSL)	10.55%[MiMKI17]	35.24[PGH+16] ± 0.90%
Naive ConvNet MT	12.31 ± 0.28%	(unspecified)
ResNet MT	6.28 ± 0.15%	9.11 ± 0.12%
SOTA (all labels)	2.86%[Gas17]	3.97%[HSS18]

Table: Error % on the test set

Consistency Cost

Diving Deeper

- Consistency cost is merely defined as $MSE(T(x), S(x))$ in our model.
- Why not $KL(T(x)||S(x))$?
- Authors tried modulating between the two to find out...

Consistency Cost

Interpolating between MSE and KL Divergence

- Define Distributions parameterized by τ:
  - $P_\tau = \tau p + \frac{1-\tau}{N}$
  - $Q_\tau = \tau q + \frac{1-\tau}{N}$
- Notice that as $\tau \rightarrow 0$, then we are left with $\frac{1}{N}$ for both distributions (eg, uniform).

Consistency Cost

Interpolating between MSE and KL Divergence

- Define "scaled" KL Divergence:
  - $\frac{2}{N^2\tau^2} KL(p_\tau || q_\tau)$
  - $\frac{2}{N^2\tau^2} \sum_x p_\tau(x) \ln \frac{q_\tau(x)}{p_\tau(x)}$ (Expanding KL term).
  - $\frac{2}{N^2\tau^2} \sum_x (\tau p(x) + \frac{1-\tau}{N}) \ln \frac{(\tau q(x) + \frac{1-\tau}{N})}{(\tau p(x) + \frac{1-\tau}{N})}$ (Expanding distributions)
- Take the Taylor expansion of q_τ and p_τ:
  - $D_{KL} \approx \frac{2}{N^2\tau^2} (\sum_x \frac{1}{2}\tau^2 N(p(x) - q(x))^2 + O(N^2\tau^3))$
  - As $\tau \to 0$, we have $\frac{1}{N} \sum_x (p(x) - q(x))^2$ (MSE)
  - As $\tau \to 1$, we have $\frac{2}{N^2} (\sum_x \frac{1}{2} N(p(x) - q(x))^2 + O(N^2)) \approx D_{KL}(p||q)$ (KL)

Consistency Cost

Interpolating between MSE and KL

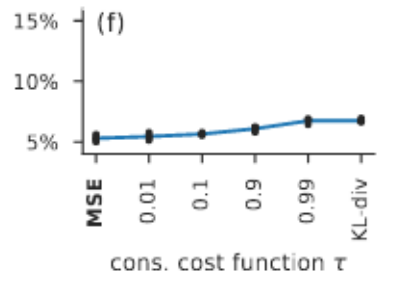


Figure: Performance of MSE vs KL, error % divergence

Consistency Cost

Interpolating between MSE and KL

- KL Divergence is a measure in difference between confidences
- MSE is a measure in difference between accuracy of predictors
- Modern NN architectures are accurate, but overconfident, so KL divergence is not going to give us "as much" information.

Consistency Cost

Effect of consistency cost

- Further studies on what Consistency Loss is actually doing [BA]
- *Penalization of output difference*
  - Can be seen as penalization of input-output Jacobian
  - $E[Q] = E_x[\,||J_x||_F^2]$, where J_x is the Jacobian of the Network's output w.r.t its inputs and F is the Frobenius Norm.
  - Has been shown to be related to generalization both theoretically [SGSR17] and empirically [NBA+18]
  - Corresponds to L2 regularization for linear models, since $J_x = W$

Consistency Cost

Effect of consistency cost

- *Penalization of weight difference*
  - Can be seen as penalization of input-weights Jacobian
  - $E[Q] = E_w[||J_w||_F^2]$, where J_w is the Jacobian of the Network's output w.r.t its weights and F is the Frobenius Norm.
  - For MSE loss, the expected trace of the Hessian of the loss $E_x[tr(H)]$ can be decomposed into a product containing the Jacobian of the weights.
  - Eigenvalues of H encode local information about the "sharpness" of the loss around a given solution. [DPBB17]

Consistency Cost

What do they look like during training

- Gradient norm of consistency loss $||\nabla_{\text{cons}}||$ remains high until the end of training and dominates the graident norm of classification loss $||\nabla_{\text{class}}||$
- For both Π and Mean-Teacher, $||\nabla_{\text{cons}}||$ is much higher than in the supervised case

Consistency Cost

Diversity

- Do the large optimization steps translate into higher prediction diversity?
- Diversity$(w_1, w_2) = \frac{1}{N} \sum^{N} |[\, y_i^{(}w_w) \neq y_i^{(}w_2)]|$ (eg, how much do the classifications differ between a model at two different training steps w_1 and w_2.
- Eg, *Diversity*(w_{170}, w_{180}) much higher for MT (7.1%) and Π (6.1%) than supervised learning (3.9%)
- SGD is still looking for solutions even by the time that training ends

Ensembling vs Weight Averaging

Ensembling

- High diversity between epochs indicates that you can get a substantial regularization benefit from ensembling.
- Measure
  $C_{\text{ens}} = \frac{1}{2}\text{Err}(w_1) + \frac{1}{2}\text{Err}(w_2) - \text{Err}(\text{Ensemble}(w_1, w_2))$
- C_{ens} substantially larger for Π and Mean Teacher Models.
- Means that SGD is still looking for solutions even by the time that training ends.

Ensembling vs Weight Averaging

Weight Averaging

- Can we get the same results with weight averaging?
- Average pairs of weights and evaluate performance with respect to weight distances
- Take $C_{\text{avg}} = \frac{1}{2}Err(w_1) + \frac{1}{2}Err(w_2) - Err(\frac{1}{2}w_1 + \frac{1}{2}w_2)$ to measure benefit of weight averaging.
- Paper finds that improvement from weight averaging on error ($1.2 \pm 0.2\%$) is larger than benefit of C_{ens} ($0.9 \pm 0.2\%$).
- Nicer, because the cost of averaging is lower than the cost of ensembling.

Ensembling vs Weight Averaging

Weight Averaging

- Can we get the same results with weight averaging?
- Average pairs of weights and evaluate performance with respect to weight distances
- Take $C_{\text{avg}} = \frac{1}{2}Err(w_1) + \frac{1}{2}Err(w_2) - Err(\frac{1}{2}w_1 + \frac{1}{2}w_2)$ to measure benefit of weight averaging.
- Paper finds that improvement from weight averaging on error ($1.2 \pm 0.2\%$) is larger than benefit of C_{ens} ($0.9 \pm 0.2\%$).
- Nicer, because the cost of averaging is lower than the cost of ensembling.

Stochastic Weight Averaging

- Method to average weights over training cycles in general [IPG+18]
- Use a cyclical learning rate schedule (sine curve), average all the model weights once per cycle
- (eg, every 30 epochs).

Stochastic Weight Averaging

Fast Stochastic Weight Averaging

- Similar idea to SWA
- Average once every k cycles, where $k < c$, the cycle period, starting from $l - c$.
- Use a cyclical learning rate schedule (sine curve), average all the model weights once per cycle
- (eg, every 30 epochs).

Fast Stochastic Weight Averaging

- Similar idea to SWA
- Average once every k cycles, where $k < c$, the cycle period, starting from $l - c$.
- Use a cyclical learning rate schedule (sine curve), average all the model weights once per cycle
- (eg, every 30 epochs).

Recent Criticism

- Paper from Google Brain, 2018 [OOR+18]
- Criticism of research and experimental methods used in current lines of SSL research
- Questions whether SSL is actually applicable to "real world" settings
- Proposals for improvement, new baselines.

Recent Criticism

Base Architectures

- Different SSL papers use base model architectures
- Creates difficulty in comparing SSL methods that use different baseline architectures.
- *Suggestion*: Use Wide ResNet-28-2, with LeakyReLU activation, standard procedures for regularization, data augmentation and preprocessing.

Recent Criticism

Supervised Baseline

- Different SSL papers seem have have varying results on "supervised" baselines, even for the same network architecture
- Issue appears to be that differing "hyperparameter optimization budgets" were used in tuning hyperparameters for SSL models vs supervised models.
- *Suggestion*: Use the same "hyperparameter optimization budgets" for both sets of models

Recent Criticism

Supervised Baseline

- *Suggestion*: Use the same "hyperparameter optimization budgets" for both sets of models
- Note that in doing so, Google Brain researchers found that they were not able to reproduce the "gap" that original authors of Π and Mean Teacher reported.
  - Originally published results:
    - » CIFAR-10: Improved from 34.85% to 12.36%
  - Reproduction:
    - » CIFAR-10: Improved from 20.26% to 16.37%

Recent Criticism

Compare to Transfer Learning

- SSL papers make no comparison to Transfer Learning, but this is the more "realistic" scenario
- Transfer Learning: Pre-train on ImageNet, then fine-tune on the limited labelled data
- In doing so, authors found that pre-training on ImageNet and fine-tuning on 4000 data points with CIFAR-10 gets 12.09% error, which is lower than 12.36% reported for SSL.

Recent Criticism

Data Augmentation

- In reality, unlabelled images may contain classes that are not part of the labels.
- Example: We want to classify between dogs and cats, our unlabelled images contain foxes, birds, other animals
- Reality: The unlabelled data is not "harmless", it may actually prove to be distracting, since it contains features but has the value of noise.
- *Suggestion*: Caveat approach with notice that it must not contain unlabelled images outside the class distribution =

Recent Criticism

Amount of unlabelled and labelled data

- Varying the amount of labelled and unlabelled data does not have a completely predictable result on the test error curve
- Just adding more unlabelled data does not always make things better.

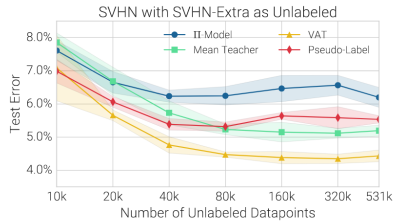


Figure: Varying labelled and unlabelled data (test error)

Recent Criticism

Validation Set Size

- Previous experiments used the rest of the "unlabelled" data as the validation set
- Validation set in Π and Mean Teacher papers is unreasonably large (7000 validation examples to 1000 labelled examples)
- Using this validation set to choose hyperparameters is unrealistic, usually validation set would be 10% of the size of the training set

References I

[BA] Pavel Izmailov Andrew Gordon Wilson Ben Athiwaratkun, Marc Finzi, *There are many consistent explanations of unlabelled data: Why you should average*, ICLR.

[DPBB17] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio, *Sharp minima can generalize for deep nets.*, ICML (Doina Precup and Yee Whye Teh, eds.), Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1019–1028.

[Gas17] Xavier Gastaldi, *Shake-shake regularization of 3-branch residual networks.*, ICLR (Workshop), OpenReview.net, 2017.

References II

[HSS18] Jie Hu, Li Shen, and Gang Sun, *Squeeze-and-excitation networks.*, CVPR, IEEE Computer Society, 2018, pp. 7132–7141.

[IPG+18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson, *Averaging weights leads to wider optima and better generalization.*, CoRR **abs/1803.05407** (2018).

[LA17] Samuli Laine and Timo Aila, *Temporal ensembling for semi-supervised learning.*, ICLR, OpenReview.net, 2017.

References III

[MiMKI17] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii, *Virtual adversarial training: a regularization method for supervised and semi-supervised learning.*, CoRR **abs/1704.03976** (2017).

[NBA+18] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein, *Sensitivity and generalization in neural networks: an empirical study.*, ICLR, OpenReview.net, 2018.

References IV

[OOR+18] Avital Oliver, Augustus Odena, Colin A. Raffel, Ekin Dogus Cubuk, and Ian J. Goodfellow, *Realistic evaluation of deep semi-supervised learning algorithms.*, NeurIPS (Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, eds.), 2018, pp. 3239–3250.

[PGH+16] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin, *Variational autoencoder for deep learning of images, labels and captions.*, NIPS (Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, eds.), 2016, pp. 2352–2360.

References V

[RBH+15] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko, *Semi-supervised learning with ladder networks.*, NIPS (Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, eds.), 2015, pp. 3546–3554.

[SGSR17] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues, *Robust large margin deep neural networks.*, IEEE Trans. Signal Processing **65** (2017), no. 16, 4265–4280.

[TH] Antti Tarvainen and Valpola Harri, *Mean teachers are better role models: Weight-averaged consistency targets improvesemi-supervised deep learning results*, NIPS.