Temporal Ensembling for Semi-Supervised Learning
(2017)

Samuli Aila, Timo Laine
Notes

1 Introduction

In this work the authors propose two models for semi-supervised leaning, the Pi model where when given a unlabled example, we'll have two kinds of prediction z and $\hat{z}$, each one is depending on a different input x due to the usage of dropout and data augmentation, and the unsupervised loss is the MSE between these two predictions, and to avoid having to apply two forward passes each time to calculate the unsupervised loss, they propose the second model, temporal ensembling where they compare the prediction of the ulabeled example to an exponentially weighted average of all the previous prediction, this is illustrated in the figure bellow:

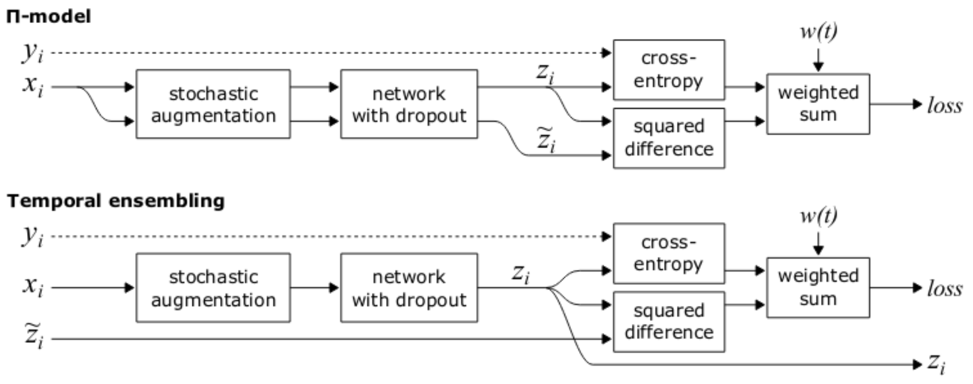

Figure 1: Structure of the training pass in our methods. Top: Π-model. Bottom: temporal ensembling. Labels y_i are available only for the labeled inputs, and the associated cross-entropy loss component is evaluated only for those.

2 Π Model

Duringtraining, we evaluate the network for each training input x_i twice, resulting in prediction vectors z_i and $\hat{z}_i$. In this case the loss function consists of two components. The first component is the standard cross-entropy loss, evaluated for labeled inputs only. The second component, evaluated for all inputs, penalizes different predictions for the same training input x_i by taking the mean square difference between the prediction vectors z_i and $\hat{z}_i$, given that the training is stochasitc with the usage of dropout, the two outputs of the same input under the same network will gives us different results, which we want to minimize to get stable results.

The supervised and unsupervised losses are combined by a weigthing factor $\omega(t)$, the weight for the unsupervised loss start with zero and increase up to 1 after a given number of epochs (80 out of 300) following this equation: $\exp\left[-5(1-T)^2\right]$.

The pseudo code:

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 $\theta$
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 $\theta$ using, e.g., ADAM                   ▷ update network parameters
    end for
  end for
  return $\theta$
```

3 Temporal Ensembling

To avoid having to apply two forward passes through the network for the unsupervised loss, the authors propose to aggregate the predictions of multiple previous netwrok evaluations into an ensemble prediction, this will give a speed up of a factor of two, but also the training targets are going to be less noisy.

After every training epoch, the network outputs z_i are accumulated into ensemble outputs Z_i by updating $\tilde{Z}_i \leftarrow \alpha Z_i + (1-\alpha)z_i$ where α is a momentum term that controls how far the ensemble reaches into training history. Because of dropout regularization and stochastic augmentation, Z thus contains a weighted average of the outputs of an ensemble of networks f from previous training epochs, with recent epochs having larger weight than distant epochs. For generating the training targets $\tilde{z}$, we need to correct for the startup bias in by dividing by factor $(1-\alpha^t)$.

The pseudo code:

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: $\alpha$ = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters $\theta$
Require: $g(x)$ = stochastic input augmentation function
  $Z \leftarrow \mathbf{0}_{[N\times C]}$                           ▷ initialize ensemble predictions
  $\tilde{z} \leftarrow \mathbf{0}_{[N\times C]}$                           ▷ initialize target vectors
  for $t$ in $[1, num\_epochs]$ do
    for each minibatch $B$ do
      $z_{i\in B} \leftarrow f_\theta(g(x_{i\in B}, t))$                  ▷ evaluate network outputs for augmented inputs
      $loss \leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap L)} \log z_i[y_i]$          ▷ supervised loss component
            $+ w(t)\frac{1}{C|B|}\sum_{i\in B} ||z_i - \tilde{z}_i||^2$      ▷ unsupervised loss component
      update $\theta$ using, e.g., ADAM                   ▷ update network parameters
    end for
    $Z \leftarrow \alpha Z + (1-\alpha)z$                       ▷ accumulate ensemble predictions
    $\tilde{z} \leftarrow Z/(1-\alpha^t)$                          ▷ construct target vectors by bias correction
  end for
  return $\theta$
```

4 Experiments

All networks were trained using Adam with a maximum learning rate of $_{max}$= 0.003, except for temporal ensembling in the SVHN case where a maximum learning rateof $_{max}$ 0.001 worked better. Adam momentum parameters were set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as suggested in the paper. The maximum value for the unsupervised loss component was set to $w_{\max} \cdot M/N$, where M is the number of labeled inputs and N is the total number of training inputs.For -model runs, $w_{\max}$= 100 in all runs except for CIFAR-100 with Tiny Images where $w_{\max}$= 300. For temporal ensembling $w_{\max}$= 30in most runs. For the corruptedlabel test in Section 3.5 we usedwmax= 300for 0% and 20% corruption, and $w_{\max}$= 3000 forcorruption of 50% and higher. For basic CIFAR-100 runs $w_{\max}$= 100, and for CIFAR-100w ith Tiny Images $w_{\max}$= 100. The accumulation decay constant of temporal ensembling was set to $\alpha = 0.6$ in all runs.

Table 1: CIFAR-10 results with 4000 labels, averages of 10 runs (4 runs for all labels).

	Error rate (%) with # labels	
	4000	All (50000)
Supervised-only	35.56 ± 1.59	7.33 ± 0.04
with augmentation	34.85 ± 1.65	6.05 ± 0.15
Conv-Large, Γ-model (Rasmus et al., 2015)	20.40 ± 0.47	
CatGAN (Springenberg, 2016)	19.58 ± 0.58	
GAN of Salimans et al. (2016)	18.63 ± 2.32	
Π-model	16.55 ± 0.29	6.90 ± 0.07
Π-model with augmentation	12.36 ± 0.31	$\mathbf{5.56 \pm 0.10}$
Temporal ensembling with augmentation	$\mathbf{12.16 \pm 0.24}$	5.60 ± 0.10

Table 2: SVHN results for 500 and 1000 labels, averages of 10 runs (4 runs for all labels).

Model	**Error rate (%) with # labels**		
	500	1000	All (73257)
Supervised-only	35.18 ± 5.61	20.47 ± 2.64	3.05 ± 0.07
with augmentation	31.59 ± 3.60	19.30 ± 3.89	2.88 ± 0.03
DGN (Kingma et al., 2014)		36.02 ± 0.10	
Virtual Adversarial (Miyato et al., 2016)		24.63	
ADGM (Maaløe et al., 2016)		22.86	
SDGM (Maaløe et al., 2016)		16.61 ± 0.24	
GAN of Salimans et al. (2016)	18.44 ± 4.8	8.11 ± 1.3	
Π-model	7.05 ± 0.30	5.43 ± 0.25	2.78 ± 0.03
Π-model with augmentation	6.65 ± 0.53	4.82 ± 0.17	$\mathbf{2.54 \pm 0.04}$
Temporal ensembling with augmentation	$\mathbf{5.12 \pm 0.13}$	$\mathbf{4.42 \pm 0.16}$	2.74 ± 0.06

Table 3: CIFAR-100 results with 10000 labels, averages of 10 runs (4 runs for all labels).

	Error rate (%) with # labels	
	10000	All (50000)
Supervised-only	51.21 ± 0.33	29.14 ± 0.25
with augmentation	44.56 ± 0.30	26.42 ± 0.17
Π-model	43.43 ± 0.54	29.06 ± 0.21
Π-model with augmentation	39.19 ± 0.36	26.32 ± 0.04
Temporal ensembling with augmentation	$\mathbf{38.65 \pm 0.51}$	$\mathbf{26.30 \pm 0.15}$

Table 4: CIFAR-100 + Tiny Images results, averages of 10 runs.

	Error rate (%) with # unlabeled auxiliary inputs from Tiny Images	
	Random 500k	Restricted 237k
Π-model with augmentation	25.79 ± 0.17	25.43 ± 0.32
Temporal ensembling with augmentation	$\mathbf{23.62 \pm 0.23}$	$\mathbf{23.79 \pm 0.24}$