TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING

School of Industrial and Management Engineering, Korea University

Hyeonji Kim

KOREA UNIVERSITY

DMQA hcai Human-Centered Artificial Intelligence Center

Contents

Research Purpose

❖ **Temporal Ensembling for Semi-Supervised Learning (ICLR 2017)**

- NVIDIA에서 연구하였으며 2022년 8월 5일 기준으로 1625회 인용됨

Published as a conference paper at ICLR 2017

TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING

Samuli Laine
NVIDIA
slaine@nvidia.com

Timo Aila
NVIDIA
taila@nvidia.com

ABSTRACT

In this paper, we present a simple and efficient method for training deep neural networks in a semi-supervised setting where only a small portion of training data is labeled. We introduce self-ensembling, where we form a consensus prediction of the unknown labels using the outputs of the network-in-training on different epochs, and most importantly, under different regularization and input augmentation conditions. This ensemble prediction can be expected to be a better predictor for the unknown labels than the output of the network at the most recent training epoch, and can thus be used as a target for training. Using our method, we set new records for two standard semi-supervised learning benchmarks, reducing the (non-augmented) classification error rate from 18.44% to 7.05% in SVHN with 500 labels and from 18.63% to 16.55% in CIFAR-10 with 4000 labels, and further to 5.12% and 12.16% by enabling the standard augmentations. We additionally obtain a clear improvement in CIFAR-100 classification accuracy by using random images from the Tiny Images dataset as unlabeled extra inputs during training. Finally, we demonstrate good tolerance to incorrect labels.

Research Purpose

❖ **Temporal Ensembling for Semi-Supervised Learning (ICLR 2017)**

- 다수의 신경망 모델을 앙상블 하면 단일 네트워크보다 더 좋은 성능을 낼 수 있음
- 앙상블 기법은 **단일 모델**에서도 dropout, dropconnect, stochastic depth regularization method 등을 사용하여 간접적으로 적용되어 왔음
- 본 논문에서는 단일 모델에서, **학습 과정에서 서로 다른 epoch, regularization, augmentation을 통해 나온 결과**를 **앙상블**에 적용하고자 함

Research Purpose

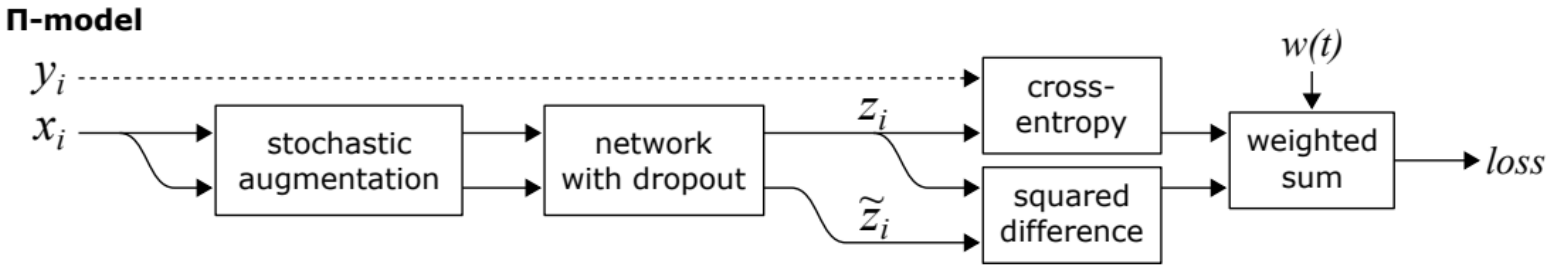
❖ **본 논문에서는 semi-supervised learning 상황에서 사용할 수 있는 Π-model과 temporal ensembling 방법을 제안함**

- 앙상블 기법을 훈련 데이터 일부에만 레이블이 있는 semi-supervised learning 상황에서 활용하고자 함
- 학습 과정에서 얻은 **앙상블 예측 값**을 **unlabeled 데이터의 training target**으로 사용함 (**Self-ensembling**)
- 본 논문에서는 학습 중 self-ensembling의 두 가지 구현을 제안함
  1) **Π-model:** 동일한 입력 데이터에 서로 다른 augmentation과 dropout이 적용되었더라도 **일관된 결과**를 출력하도록 학습함
  2) **Temporal ensembling:** 학습 과정에서 얻은 **앙상블 예측 값**을 unlabeled 데이터의 training target으로 사용함

Proposed Method

❖ Π-MODEL (1/2)

- 동일한 입력 데이터 x_i에 대해 서로 다른 두 개의 augmentation과 dropout을 적용함
  - ✓ 학습 하는 동안 입력 데이터는 두 개의 네트워크에 각각 forward됨
- 두 가지 loss를 사용함
  - ✓ **Cross-entropy loss:** labeled 데이터에 대해서만 계산
  - ✓ **Squared difference loss:** labeled, unlabeled 데이터 모두에 대해 계산
    - 동일한 입력 데이터에 대해 다른 augmentation과 dropout이 적용되었더라도 **일관된 출력 결과**가 나오도록 함
- 두 가지 loss를 가중치 $w(t)$를 사용해 weighted sum을 함



[Π-Model]

Proposed Method

❖ Π-MODEL (2/2)

- 두 가지 loss를 위한 가중치 $w(t)$는 시간에 따라 변화함
  - ✓ 0에서 부터 **ramp-up 방식**으로 **증가**함
  - ✓ 초반에는 supervised loss를 중점적으로 학습하고, 점점 unsupervised loss의 비중이 높아짐
  - ✓ Unsupervised loss component를 충분히 천천히 학습하지 않으면 유의미한 학습 결과를 얻을 수 없음

Algorithm 1 Π-model pseudocode.

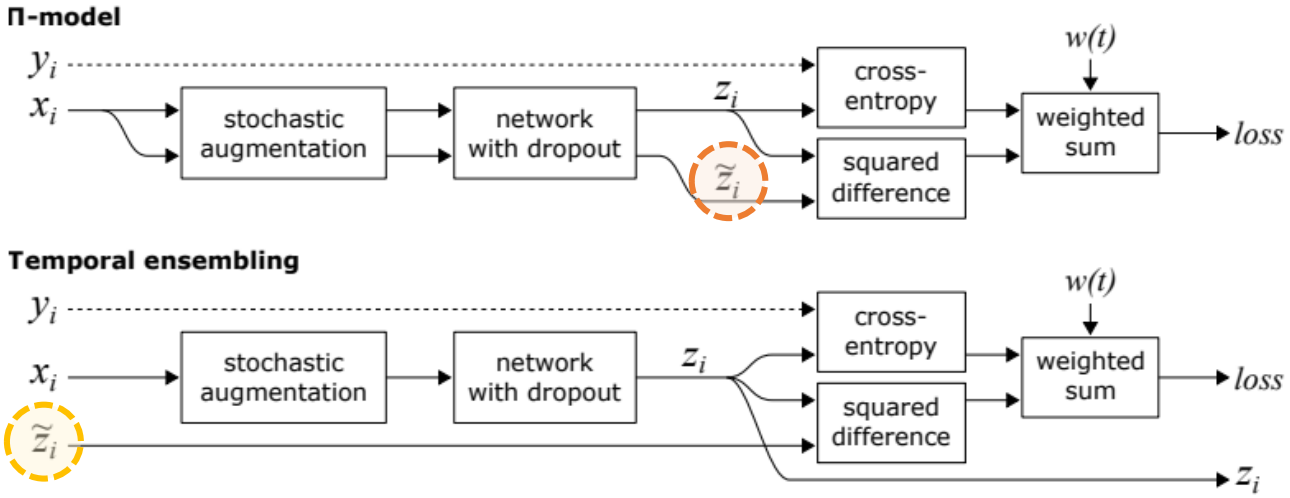
Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic 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
    $\quad + w(t)\frac{1}{C|B|}\sum_{i\in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
    update θ using, e.g., ADAM ▷ update network parameters
  end for
end for
return θ

[Π-Model pseudocode]

Proposed Method

❖ TEMPORAL ENSEMBLING (1/2)

- Π-Model의 unsupervised learning을 위한 training target $\tilde{z}_i$는 네트워크에 대한 **single evaluation을** 기반으로 하기 때문에 **noisy**하다는 단점이 존재함
- Temporal ensembling에서는 각 입력 데이터에 대해, **이전까지 저장된 예측 값을 ensembling**한 값을 $\tilde{z}_i$로 사용함
  - ✓ Π-Model에서의 $\tilde{z}_i$에 비해 덜 noisy함



[Π-Model과 temporal ensembling 비교]

Proposed Method

❖ TEMPORAL ENSEMBLING (2/2)

- 각 epoch에서 하나의 입력 데이터에 대해 한 번의 forward만 수행하기 때문에 **학습 속도가 빠름**
- 그러나 각 입력 데이터에 대해 $\tilde{z}_i$를 저장해야 하기 때문에 **별도의 메모리 공간**이 필요함
- **Momentum parameter α**를 이용해 이전까지의 정보를 얼마나 사용할지 조정할 수 있음

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

$Z \leftarrow \mathbf{0}_{[N \times C]}$ ▷ 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 θ 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 θ

[Temporal ensembling pseudocode]

Experiment

❖ CIFAR-10, SVHN, CIFAR-100 dataset

- 기존 semi-supervised learning 방법론보다 두 제안 방법론 모두 좋은 성능을 보임
- Temporal ensembling이 Π-model 보다 더 좋은 성능을 보이며 학습 속도도 2배 정도 빠름
- 전체 레이블을 모두 사용할 경우에는 성능이 더욱 향상됨

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

[CIFAR-10 results]

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

[SVHN results]

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	$\mathbf{38.65 \pm 0.51}$	$\mathbf{26.30 \pm 0.15}$

[CIFAR-100 results]

Experiment

❖ CIFAR-100 and Tiny Images

- CIFAR-100 데이터에 Tiny Images 데이터 셋의 레이블이 없는 데이터를 추가하여 두 가지 테스트를 진행함
  1) 무작위로 추가 데이터를 구성: 대부분 CIFAR-100 범주에 해당하지 않음
  2) 범주를 제한하여 추가 데이터를 구성: CIFAR-100 범주에 해당하는 이미지들을 포함
- 레이블이 없는 추가 데이터를 사용한 결과 성능이 향상됨
- 추가 데이터를 CIFAR-100에 있는 범주로 제한해도 성능이 향상되지 않음

  → 꼭 동일한 클래스의 데이터가 아닌 적절한 unlabeled 데이터를 활용하는 것으로도 충분함

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

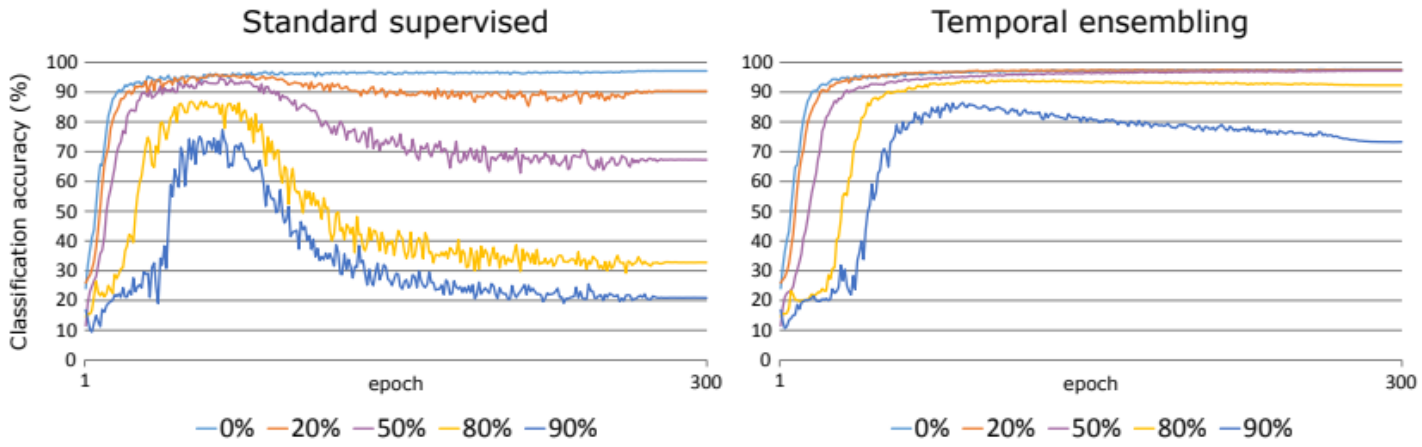
	Error rate (%) with # unlabeled auxiliary inputs from Tiny Images	
	Random 500k	Restricted 237k
Π-model with augmentation	25.79 ± 0.17	25.43 ± 0.32
Temporal ensembling with augmentation	$\mathbf{23.62 \pm 0.23}$	$\mathbf{23.79 \pm 0.24}$

[CIFAR-100 + Tiny Images results]

Experiment

❖ Tolerance to Incorrect Labels

- 일부 데이터에 무작위로 레이블을 할당하여 잘못된 레이블을 포함된 데이터 셋으로 실험을 진행함
- 일반적인 supervised learning에서는 잘못된 레이블의 비율이 증가함에 따라 성능이 급격하게 하락함
- Temporal ensembling은 잘못된 레이블이 포함된 경우에도 강건한 성능을 보임



[잘못된 레이블의 비율에 따른 성능]

Conclusion

❖ Π-Model & Temporal ensembling

- 본 논문에서는 기존 다수의 모델들을 앙상블하는 것과 다르게, 단일 모델에서 서로 다른 epoch, regularization, augmentation을 통해 나온 결과들을 앙상블 함
- Π-model은 동일한 입력 데이터에 서로 다른 augmentation과 dropout이 적용되었더라도 일관된 결과를 출력하도록 학습함
- Temporal ensembling은 학습 과정에서 얻은 앙상블 예측 값을 unlabeled 데이터의 training target으로 사용하여 Π-model 보다 성능을 향상시키고, 학습 시간을 줄임
- 기존 semi-supervised 방법론 보다 좋은 성능을 보였으며 unlabeled 데이터가 포함된 상황 뿐만 아니라 모든 데이터에 레이블이 있는 경우에도 좋은 성능을 보임
- 부정확한 레이블이 포함되어 있어도 강건한 성능을 보임


Thank You