Deep Co-Training for Semi-Supervised Image Recognition

(2018)

Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, Alan Yuille

Notes

1 Introduction

In this paper the authors explore using co-training, where they train different neural network using different views of the input data by generating different and new views from the original data using adversarial examples, and compare the predictions of these network.

Given an image dataset $D = S \cup U$ where the images in S are labeled and the images in U are not, the task if to build classifiers on the C categories that appear in the S, and the goal of this paper is to use the unlabeled data U to help learning on S, for this the authors try to adpat the co-training framework in a deep learning setting, this framework assumes that each data x in D has two views: $x = (v_1, v_2)$ and each view is sufficient for learning an effective model (like different data source, augmentations or representations), given $\mathcal{X}$, a distribution drawn from D, co-training assumes that f_1 and f_2 trained on views v_1 and v_2 have consistent prediction on $\mathcal{X}$:

$$f(x) = f_1(v_1) = f_2(v_2), \quad \forall x = (v_1, v_2) \sim \mathcal{X}$$

So co-training proposes a dual view self training algorithm: first we learn seperate classifiers for each view on S and then gradually add the prediction of the two classifiers on U to S, in this framework we push the models to make similar predictions on U and S, the this can lead to colapsed models, so it is necessary to have a force that pushes the neworks away to balance the co-training assumption that pull them together, for this the authors add a new contraint called *View Difference Constraint* that encourages the networks to be different:

$$\exists \mathcal{X}' : f_1(v_1) \neq f_2(v_2), \forall x = (v_1, v_2) \sim \mathcal{X}'$$

The authors model the co-training assumption by minimizing the Jensen-Shannon divergence between the predictions of the two networks on U, and to avoid the colapsing of the two networks they impose different views by training each network to be resistance to adversarial examples, so that the networks provide different and complementary information about the data.

2 Deep Co-Training

2.1 Co-Training

Given a dataset $D = S \cup U$ with labeled S and unlabeled data U, the two views $v_1(x)$ and $v_2(x)$ are convolutionnal representation of x before the final fully-connected layers $f_i(.)$, on the labeled examples in S we use standard cross entropy loss to minimizes the expected loss $\mathbb{E}[\mathcal{L}_{\text{sup}}]$:

$$\mathcal{L}_{\text{sup}}(x, y) = H(y, f_1(v_1(x))) + H(y, f_2(v_2(x)))$$

In the co-training, we want the two predictions ($p_1(x) = f_1(v_1(x))$ and $p_2(x) = f_2(v_2(x))$) using the two networks, i.e. two different fully connected layer, on the two different views to be close on the unlabaled data, so the objective is to minimize the difference modeled by the Jensen-Shannon divergence on U to minimize the expected loss $\mathbb{E}[\mathcal{L}_{\text{cot}}]$:

$$\mathcal{L}_{\text{cot}}(x) = H\left(\frac{1}{2}\left(p_1(x) + p_2(x)\right)\right) - \frac{1}{2}\left(H\left(p_1(x)\right) + H\left(p_2(x)\right)\right)$$

2.2 View Difference Constrain

The important aspect in co-training is having two different views of the input that present complementatry information about it, using the above learning objectives wil only encourage the network to output the same predictions on $D = S \cup U$ but will not push the networks to be different and complementary, to achieve this the authors create a new set of images D' where $p_1(x) \neq p_2(x), \forall x \in \mathcal{D}'$ that are generated by adversarial examples, and given that $p_1(x) = p_2(x), \quad \forall x \in \mathcal{D}$, we have $\mathcal{D} \cap \mathcal{D}' = \emptyset$, to generate the new images of the new set $\mathcal{D}' = \{g(x) | x \in \mathcal{D}\}$, we need a simple generative method $g(x)$, so that the generated images are similar to natural images ($g(x) - x$ is small) but in the same time $p_1(x) \neq p_2(x)$, and adversarial examples have these propreties, so to avoid the network colapsing, we train each network to be resistant to the adversatial examples if the second network by minimizing the CE of the prediction between the output of network having as input the clean examples x and the output of the net having as an input the generated example $g(x)$:

$$\mathcal{L}_{\text{dif}}(x) = H\left(p_1(x), p_2\left(g_1(x)\right)\right) + H\left(p_2(x), p_1\left(g_2(x)\right)\right)$$

2.3 Training

So the objective function is a linear combination of the above equations with a number of weighting factors λ:

$$\mathcal{L} = \mathbb{E}_{(x,y)\in\mathcal{S}}\mathcal{L}_{\text{sup}}(x, y) + \lambda_{\text{cot}}\mathbb{E}_{x\in\mathcal{U}}\mathcal{L}_{\text{cot}}(x) + \lambda_{\text{dif}}\mathbb{E}_{x\in\mathcal{D}}\mathcal{L}_{\text{dif}}(x)$$

And here is the algorithm (note: each network have different steams of data)

Algorithm 1: One Iteration of the Training Loop of Deep Co-Training

1 **Data Sampling** Sample data batch $b_1 = (x_{b_1}, y_{b_1})$ for p_1 and $b_2 = (x_{b_2}, y_{b_2})$ for p_2 from $\mathcal{S}$ s.t. $|b_1| = |b_2| = b$. Sample data batch $b_u = (x_u)$ from $\mathcal{U}$.

2 **Create Adversarial Examples** Compute the adversarial examples $g_1(x)$ of p_1 for all $x \in x_{b_1} \cup x_u$ and $g_2(x)$ of p_2 for all $x \in x_{b_2} \cup x_u$ using FGSM [24].

3 $\mathcal{L}_{\text{sup}} = \frac{1}{b}\Big[\sum_{(x,y)\in b_1} H(y, p_1(x)) + \sum_{(x,y)\in b_2} H(y, p_2(x))\Big]$

4 $\mathcal{L}_{\text{cot}} = \frac{1}{|b_u|}\sum_{x\in b_u}\Big[H\Big(\frac{1}{2}\big(p_1(x) + p_2(x)\big)\Big) - \frac{1}{2}\Big(H\big(p_1(x)\big) + H\big(p_2(x)\big)\Big)\Big]$

5 $\mathcal{L}_{\text{dif}} = \frac{1}{b + |b_u|}\Big[\sum_{x\in x_1\cup x_u} H(p_1(x), p_2(g_1(x))) + \sum_{x\in x_2\cup x_u} H(p_2(x), p_1(g_2(x)))\Big]$

6 $\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_{\text{cot}}\mathcal{L}_{\text{cot}} + \lambda_{\text{dif}}\mathcal{L}_{\text{dif}}$

7 **Update** Compute the gradients with respect to $\mathcal{L}$ by backpropagation and update the parameters of p_1 and p_2 using gradient descent.

We can also have multiple views instead of only two, and calculate the losses between each pairs of the views:

$$\mathcal{L}_{\text{ fake } n-\text{ view}}(t) = \sum_{i=1}^{n/2}\mathcal{L}\left(v_{2i-1}, v_{2i}, B_i(t)\right)$$

Here we only consider $n/2$ independent dual views (v1 and v2, then v3 and v4 ...), one alternative is to shuffle a list l of n views at each training iterations, and then using the same loss between pairs, but this time at each iteration will be comparing different pairs:

$$\mathcal{L}_{n\text{ -view}}(t) = \sum_{i=1}^{n/2}\mathcal{L}\left(v_{l_{2i-1}}, v_{l_{2i}}, B_i(t)\right)$$

3 Experiments

- SVHN and CIFAR: using warmup scheme for the hyperparameters of the unsupervised losses: $\lambda = \lambda_{\max} \cdot \exp\left(-5(1 - T/80)^2\right)$ and cos anneling for the learning rate: $lr = 0.05 \times (1.0 + \cos((T - 1) \times \pi/600))$.
- Imagenet: first train the two network using only supervised data for 600 epochs ($\tilde{6}0$ in case of full supervision), the lr is 0.1 and divided by 10 at the 301st epoch, and then they train the model using both supervised and unsupervised examples and set $\lambda = \lambda_{\max}$ directely and they train on both for 20 epoch using a lr scheduler : $lr = 0.005 \times (1.0 + \cos((T - 1) \times \pi/20))$

Method	SVHN	CIFAR-10
GAN [36]	8.11 ± 1.30	18.63 ± 2.32
Stochastic Transformations [18]	–	11.29 ± 0.24
Π Model [19]	4.82 ± 0.17	12.36 ± 0.31
Temporal Ensembling [19]	4.42 ± 0.16	12.16 ± 0.24
Mean Teacher [35]	3.95 ± 0.19	12.31 ± 0.28
Bad GAN [20]	4.25 ± 0.03	14.41 ± 0.30
VAT [28]	3.86	10.55
Deep Co-Training with 2 Views	3.61 ± 0.15	9.03 ± 0.18
Deep Co-Training with 4 Views	3.38 ± 0.05	8.54 ± 0.12
Deep Co-Training with 8 Views	3.29 ± 0.03	8.35 ± 0.06

Table 1. Error rates on SVHN (1000 labeled) and CIFAR-10 (4000 labeled) benchmarks. Note that we report the averages of the single model error rates without ensembling them for the fairness of comparisons. We use architectures that are similar to that of Π Model [19]. "–" means that the original papers did not report the corresponding error rates. We report means and standard deviations from 5 runs.

Method	CIFAR-100	CIFAR-100+
Π Model [19]	43.43 ± 0.54	39.19 ± 0.36
Temporal Ensembling [19]	–	38.65 ± 0.51
Dual-View Deep Co-Training	$\mathbf{38.77 \pm 0.28}$	$\mathbf{34.63 \pm 0.14}$

Table 2. Error rates on CIFAR-100 with 10000 images labeled. Note that other methods listed in Table 1 have not published results on CIFAR-100. The performances of our method are the averages of single model error rates of the networks without ensembling them for the fairness of comparisons. We use architectures that are similar to that of Π Model [19]. "–" means that the original papers did not report the corresponding error rates. CIFAR-100+ and CIFAR-100 indicate that the models are trained with and without data augmentation, respectively. Our results are reported from 5 runs.

Method	Architecture	# Param	Top-1	Top-5
Stochastic Transformations [18]	AlexNet	61.1M	–	39.84
VAE [34] with 10% Supervised	Customized	30.6M	51.59	35.24
Mean Teacher [35]	ResNet-18	11.6M	49.07	23.59
100% Supervised	ResNet-18	11.6M	30.43	10.76
10% Supervised	ResNet-18	11.6M	52.23	27.54
Dual-View Deep Co-Training	ResNet-18	11.6M	**46.50**	**22.73**

Table 3. Error rates on the validation set of ImageNet benchmark with 10% images labeled. The image size of our method in training and testing is 224×224.

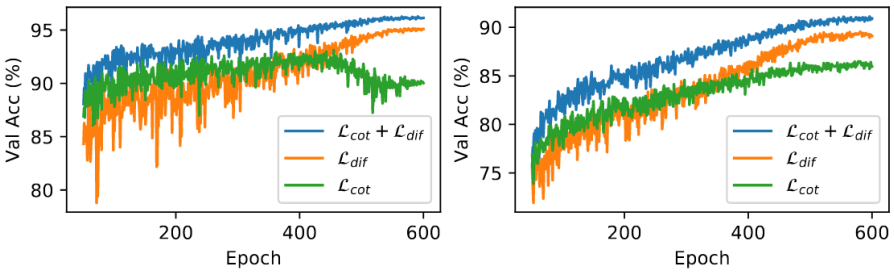

Fig. 1. Ablation study on $\mathcal{L}_{\text{cot}}$ and $\mathcal{L}_{\text{dif}}$. The left plot is the training dynamics of dual-view Deep Co-Training on SVHN dataset, and the right is on CIFAR-10 dataset. "λ_{cot}", "λ_{dif}" represent the loss functions are used alone while "$\lambda_{\text{cot}} + \lambda_{\text{dif}}$" correspond to the weighted sum loss used in Deep Co-Training. In all the cases, $\mathcal{L}_{\text{sup}}$ is used.