

# GAN-based Semi-supervised Learning

Xiangli Yang

November 9, 2018

- 1 Semi-supervised Learning
- 2 Generative Adversarial Nets
- 3 GAN-based Semi-supervised Learning
- 4 Summary

- 1 Semi-supervised Learning
- 2 Generative Adversarial Nets
- 3 GAN-based Semi-supervised Learning
- 4 Summary

# What is semi-supervised learning?

**Semi-supervised Learning (SSL)** is a class of machine learning techniques that make use of both labeled and unlabeled data for training.

**Goal:** Using both labeled and unlabeled data to build better learners, than using each one alone.

- Semi-supervised Learning (SSL) learns from both
  - ▶ labeled data (expensive and scarce)
  - ▶ unlabeled data (cheap and abundant)

# What is semi-supervised learning? (Cont.)

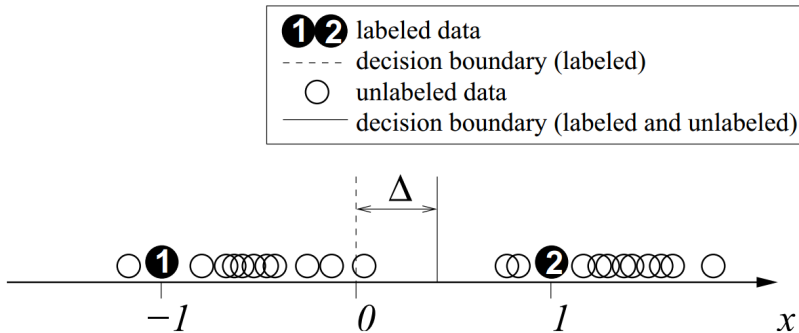
■ Given  $i, i, d$  samples from an unknown distribution  $p(\mathbf{x}, y)$  over  $\mathbf{x} \in \Omega$  and  $y \in \mathbb{Z}$  organized as

- ▶ a labeled set:  $\mathcal{L} = \mathcal{X}_l \cup \mathcal{Y}_l = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$
- ▶ an unlabeled set:  $\mathcal{U} = \mathcal{X}_u = \{\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+u}\}$ , and  $\mathcal{L} \ll \mathcal{U}$

■ Output missing labels  $\hat{\mathcal{Y}}_u = \{\hat{y}_{l+1}, \dots, \hat{y}_{l+u}\}$  that largely agree with true missing labels

$$\mathcal{Y}_u = \{y_{l+1}, \dots, y_{l+u}\}$$

# How can unlabeled data ever help?



- Figure:**
- Assuming each class is a coherent group (e.g. Gaussian)
  - With and without unlabeled data: decision boundary shift

- ▶ Suppose the data is well-modeled by a mixture density :

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{l=1}^L \alpha_l \cdot p(\mathbf{x}|\theta_l)$$

where  $\sum_{l=1}^L \alpha_l = 1$  and  $\boldsymbol{\theta} = \{\theta_l\}$

The class labels are viewed as random quantities and are assumed chosen conditioned on the selected mixture component  $m_i \in \{1, 2, \dots, L\}$  and possibly on the feature value, i.e. according to the probabilities  $p[c_i|x_i, m_i]$ <sup>1</sup>

---

<sup>1</sup>[Miller and Uyar, 1997]

# Why unlabeled data can be helpful? (Cont.)

- ▶ Thus, the optimal classification rule for this model is the maximum a posteriori rule:

$$\begin{aligned} S(\mathbf{x}) &= \arg \max_k p(c_i = k | x_i) \\ &= \arg \max_k \sum_j p(c_i = k, m_i = j | x_i) \\ &= \arg \max_k \sum_j p[c_i = k | m_i = j, x_i] p[m_i = j | x_i] \end{aligned} \tag{1}$$

$$\text{where } p[m_i = j | x_i] = \frac{\alpha_j f(x_i | \theta_j)}{\sum_{l=1}^L \alpha_l f(x_i | \theta_l)}$$

Since this rule is based on the a posteriori class probabilities, one can argue that learning should focus solely on estimating these probabilities. However, if the classifier truly implements , then implicitly it has been assumed that the estimated mixture density accurately models the feature vectors. **If this is not true, suggests that even in the absence of class labels, the feature vectors can be used to better learn a posteriori probabilities via improved estimation of the mixture-based feature density.**



- 1 Semi-supervised Learning
- 2 Generative Adversarial Nets**
- 3 GAN-based Semi-supervised Learning
- 4 Summary

# Generative Adversarial Nets

## GANs<sup>2</sup> framework:

Training generative model's through an objective function that implements a two-player zero sum game between a discriminator  $D$  and a generator  $G$ .

- ▶ **Discriminator  $D$** : a function aiming to tell apart real from fake input data.
- ▶ **Generator  $G$** : a function that is optimized to generate input data (from noise) that “fools” the discriminator.

---

<sup>2</sup>[Goodfellow et al., 2014]

- ▶ The “game” that the generator produces an example from random noise that has the potential to fool the discriminator.
- ▶ The discriminator is then presented a few real data examples, together with the examples produced by the generator, and it’s task is to classify them as “real” or “fake”.

- ▶ Afterwards, the discriminator is rewarded for correct classification and the generator for generating examples that did fool the discriminator.
- ▶ Both models are then updated and the next cycle of the game begins.

This process can be **formalized** as follows.

- ▶ Let  $\mathcal{X} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$  be a dataset of provided “real” inputs with dimensionality  $I$  (i.e.  $\mathbf{x} \in \mathbb{R}^I$ ).
- ▶ Let  $D$  denote the mentioned discriminative function and  $G$  denote the generator function.
- ▶  $G$  maps random vectors  $\mathbf{z} \in \mathbb{R}^Z$  to generated inputs  $\tilde{\mathbf{x}} = G(\mathbf{z})$  and we assume  $D$  to predict the probability of example  $\mathbf{x}$  being present in the dataset

$$\mathcal{X} : p(y = 1 | \mathbf{x}, D) = \frac{1}{1 + e^{-D(\mathbf{x})}}.$$

- ▶  $D$  and  $G$  play the following two-player minimax game with value function  $V(D, G)$ :

$$\begin{aligned} \min_G \max_D V(D, G) = \min_G \max_D \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} [\log p(y = 1 | \mathbf{x}, D)] \\ + \mathbb{E}_{\mathbf{z} \sim P(\mathbf{z})} [\log(1 - p(y = 1 | G(\mathbf{z}), D))] \end{aligned} \quad (2)$$

where  $P(\mathbf{z})$  is an arbitrary noise distribution.

- ▶ If both the generator and the discriminator are deep neural networks then they can be trained by alternating stochastic gradient descent (SGD) steps on the objective functions, effectively implementing the two player game described above.

# Architecture of GANs

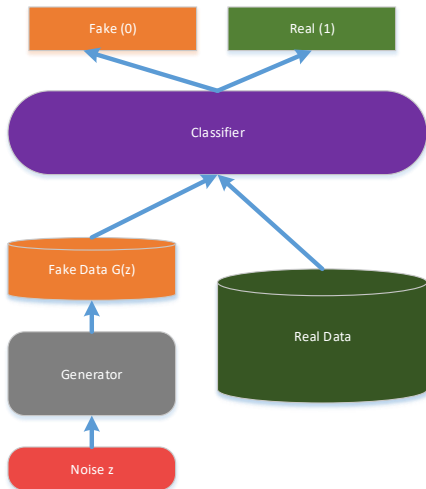


Figure: GANs

---

---

**for** number of training iterations **do**

**for**  $k$  steps **do**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)}))) \right].$$

**end for**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(\mathbf{z}^{(i)}))).$$

**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

---

<sup>3</sup>[Goodfellow et al., 2014]



- 1 Semi-supervised Learning
- 2 Generative Adversarial Nets
- 3 GAN-based Semi-supervised Learning**
- 4 Summary

- ▶ Consider a standard classifier for classifying a data point  $\mathbf{x}$  into one of  $K$  possible classes.
- ▶ Such a model takes in  $\mathbf{x}$  as input and outputs a  $K$ -dimensional vector of logits  $\{l_1, \dots, l_K\}$ ,
- ▶ Applying the softmax:  $p_{\text{model}}(y = j | \mathbf{x}) = \frac{\exp(l_j)}{\sum_{k=1}^K \exp(l_k)}$ .
- ▶ Trained by minimizing the cross-entropy between the observed labels and the model predictive distribution  $p_{\text{model}}(y | \mathbf{x})$ .

- ▶ Adding samples from the GAN generator  $G$  to our data set, labeling them with a new “generated” class  $y = K + 1$ , so our classifier output from  $K$ -dimension to  $K + 1$ -dimension.
- ▶ Use  $p_{model}(y = K + 1|\mathbf{x})$  to supply them probability that  $\mathbf{x}$  is fake, corresponding to  $1 - D(\mathbf{x})$
- ▶ Instead of binary classification, the discriminator employs a  $(K + 1)$ -class objective, where true samples are classified into the first  $K$  classes and generated samples are classified into the  $(K + 1)$ -th class.

- ▶ Given a labeled set  $\mathcal{L} = \{(\mathbf{x}, y)\}$ , let  $\{1, 2, \dots, K\}$  be the label space for classification.
- ▶ Let  $D$  and  $G$  denote the discriminator and generator, and  $P_D$  and  $P_G$  denote the corresponding distributions.

- ▶ Consider the discriminator objective function of GAN-based semi-supervised learning <sup>4</sup>:

$$\begin{aligned} V = \max_D & \mathbb{E}_{\mathbf{x}, y \sim p(\mathbf{x}, y)} \log P_D(y | \mathbf{x}, y \leq K) \\ & + \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \log [1 - P_D(y = K + 1 | \mathbf{x})] \\ & + \mathbb{E}_{\mathbf{x} \sim P_G} \log P_D(y = K + 1 | \mathbf{x}) \end{aligned} \quad (3)$$

where  $p(\mathbf{x})$  is the true data distribution.

- ▶ The probability distribution  $P_D$  is over  $K + 1$  classes where the first  $K$  classes are true classes and the  $(K + 1)$ -th class is the fake class.

---

<sup>4</sup>[Salimans et al., 2016]

- ▶ Consider the discriminator objective function of GAN-based semi-supervised learning <sup>3</sup>:

$$\begin{aligned}
 V = \max_D & \mathbb{E}_{\mathbf{x}, y \sim p(\mathbf{x}, y)} \log P_D(y | \mathbf{x}, y \leq K) \\
 & + \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \log [1 - P_D(y = K + 1 | \mathbf{x})] \\
 & + \mathbb{E}_{\mathbf{x} \sim P_G} \log P_D(y = K + 1 | \mathbf{x})
 \end{aligned} \tag{3}$$

where  $p(\mathbf{x})$  is the true data distribution.

- ▶ The probability distribution  $P_D$  is over  $K + 1$  classes where the first  $K$  classes are true classes and the  $(K + 1)$ -th class is the fake class.

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \log [P_D(y \leq K) | \mathbf{x}]$$

The objective function consists of three terms.

- ▶ The first term is to maximize the log conditional probability for labeled data, which is the standard cost as in supervised learning setting.
- ▶ The second term is to maximize the log probability of the first  $K$  for unlabeled data.
- ▶ The third term is to maximize the log probability of the  $(K + 1)$ -th class of generated data.

- ▶ Let  $f(x)$  be a nonlinear vector-valued function, and  $w_k$  be the weight vector for class  $k$ .
- ▶ The discriminator  $D$  is defined as

$$P_D(y = k|x) = \frac{\exp(w_k^\top f(x))}{\sum_{j=1}^{K+1} \exp(w_j^\top f(x))}$$

Since this is a form of over-parameterization,  $w_{K+1}$  is fixed as a zero vector<sup>5</sup>.

---

<sup>5</sup>[Dumoulin et al., 2017]



# Architecture of GAN-based SSL

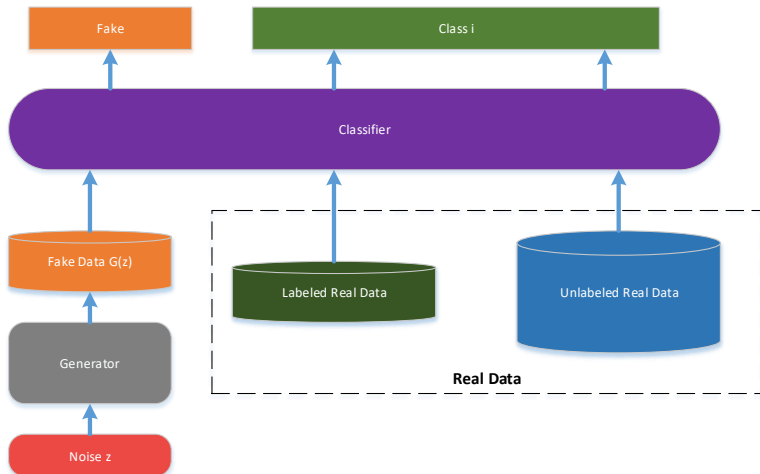


Figure: GAN-based Semi-supervised Learning

## ► Experiments on MNIST

Model	Number of incorrectly predicted test examples for a given number of labeled samples			
	20	50	100	200
DGN [Kingma et al., 2014]			333 ± 14	
Virtual Adversarial [Miyato et al., 2016]			212	
CatGAN [Springenberg, 2016]			191 ± 10	
Skip Deep Generative Model [Maaloe et al., 2016]			132 ± 7	
Ladder network [Rasmus et al., 2015]			106 ± 37	
Auxiliary Deep Generative Model [Maaloe et al., 2016]			96 ± 2	
Our model	1677 ± 452	221 ± 136	93 ± 6.5	90 ± 4.2
Ensemble of 10 of our models	1134 ± 445	142 ± 96	86 ± 5.6	81 ± 4.3

► Setups with 20, 50, 100 and 200 labeled samples, and have 5 hidden layers each.

► Results are averaged over 10 random subsets of labeled data.

---

<sup>6</sup>[Salimans et al., 2016]

## ► Experiments on CIFAR-10

Model	Test error rate for a given number of labeled samples			
	1000	2000	4000	8000
Ladder network [Rasmus et al., 2015]			20.40 $\pm$ 0.47	
CatGAN [Springenberg, 2016]			19.58 $\pm$ 0.46	
Our model	21.83 $\pm$ 2.01	19.61 $\pm$ 2.09	18.63 $\pm$ 2.32	17.72 $\pm$ 1.82
Ensemble of 10 of our models	19.22 $\pm$ 0.54	17.25 $\pm$ 0.66	15.59 $\pm$ 0.47	14.87 $\pm$ 0.89

- Discriminator: a 9 layers CNN with dropout and weight normalization.
- Generator: a 4 layers CNN with batch normalization.

## ► Experiments on SVHN

Model	Percentage of incorrectly predicted test examples for a given number of labeled samples		
	500	1000	2000
DGN [Kingma et al., 2014]		36.02 ± 0.10	
Virtual Adversarial [Miyato et al., 2016]		24.63	
Auxiliary Deep Generative Model [Maaloe et al., 2016]		22.86	
Skip Deep Generative Model [Maaloe et al., 2016]		16.61 ± 0.24	
Our model	18.44 ± 4.8	8.11 ± 1.3	6.16 ± 0.58
Ensemble of 10 of our models		5.88 ± 1.0	

► Discriminator: a 9 layers CNN with dropout and weight normalization.

► Generator: a 4 layers CNN with batch normalization.

- 1 Semi-supervised Learning
- 2 Generative Adversarial Nets
- 3 GAN-based Semi-supervised Learning
- 4 Summary**

▶ Pros:

- Achieved excellent classification performance with few labels on SHVN, MNIST, and CIFAR, and outperforming all models in the benchmark.
- Adversarial methods in the semi-supervised setting can be reused.


▶ Cons:

- The Section 3.2 where technical improvements are discussed is less clear.
- There is no clear proof that proposed methods encourage convergence or not.

-  Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. C. (2017).

Adversarially learned inference.

*international conference on learning representations.*

-  Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014).

Generative adversarial nets.

In *NIPS*.

-  Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014).

Semi-supervised learning with deep generative models.

*NIPS*.

 Maaloe, L., Sonderby, C. K., Sonderby, S. K., and Winther, O. (2016).

Auxiliary deep generative models.

*ICML.*

 Miller, D. J. and Uyar, H. S. (1997).

A mixture of experts classifier with learning based on both labelled and unlabelled data.


*In Advances in Neural Information Processing Systems 9.* MIT Press.

 Miyato, T., Maeda, S., Koyama, M., Nakae, K., and Ishii, S. (2016).

Distributional smoothing with virtual adversarial training.


*ICLR.*



-  Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. (2015).

Semi-supervised learning with ladder networks.

*NIPS.*

-  Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016).

Improved techniques for training gans.

In *NIPS.*

-  Springenberg, J. T. (2016).

Unsupervised and semi-supervised learning with categorical generative adversarial networks.

*ICLR.*