Machine Vision and Applications (2020) 31:49
https://doi.org/10.1007/s00138-020-01096-z

ORIGINAL PAPER

Semi-supervised learning using adversarial training with good and bad samples

Wenyuan Li[1] · Zichen Wang[1] · Yuguang Yue[2] · Jiayun Li[1] · William Speier[1] · Mingyuan Zhou[2] · Corey Arnold[1]

Received: 5 December 2019 / Revised: 14 May 2020 / Accepted: 24 June 2020

Abstract
In this work, we investigate semi-supervised learning (SSL) for image classification using adversarial training. Previous results have illustrated that generative adversarial networks (GANs) can be used for multiple purposes in SSL . Triple-GAN, which aims to jointly optimize model components by incorporating three players, generates suitable image-label pairs to compensate for the lack of labeled data in SSL with improved benchmark performance. Conversely, Bad (or complementary) GAN optimizes generation to produce complementary data-label pairs and force a classifier's decision boundary to lie between data manifolds. Although it generally outperforms Triple-GAN, Bad GAN is highly sensitive to the amount of labeled data used for training. Unifying these two approaches, we present unified-GAN (UGAN), a novel framework that enables a classifier to simultaneously learn from both good and bad samples through adversarial training. We perform extensive experiments on various datasets and demonstrate that UGAN: (1) achieves competitive performance among other GAN-based models, and (2) is robust to variations in the amount of labeled data used for training.

Keywords Semi-supervised learning · Generative adversarial network · Image classification · Adversarial training

1 Introduction

With recent progress in deep learning, large labeled training datasets are becoming increasingly important [1,5,13,22]. However, labeling such datasets is expensive and time-consuming. Semi-supervised learning (SSL) aims to leverage large amounts of unlabeled data to boost model performance. Various SSL methods have been proposed using deep learning and proven to be successful. Weston et al. [37] employed a manifold embedding technique using a pre-constructed graph of unlabeled data; Rasmus et al. [32] used a specially designed auto-encoder to extract essential features for classification; Kingma and Welling [12] developed a variational autoencoder by maximizing the variational lower bound of both labeled and unlabeled data; Miyato et al. [26] proposed virtual adversarial training (VAT), which helped find a deep classifier that had a good prediction accuracy and was less sensitive to data perturbation toward the adversarial direction.

Recently, generative adversarial networks (GANs) [9] have demonstrated their capability in SSL frameworks [4,7,15,17,19,20,34]. GANs are a powerful class of deep generative models that can represent data distributions over natural images [24,31]. Specifically, a GAN is formulated as a two-player game, where the generator G takes a random vector z as input and produces a sample $G(z)$ in the data space, while the discriminator D identifies whether a certain sample comes from the true data distribution $p(x)$ or the generator. The training procedure of GAN is to solve a minimax problem:

$$\begin{aligned}\min_G \max_D U(D,G) &= E_{x\sim p(x)}[\log(D(x))]+ \\ &= E_{z\sim p_z(z)}[\log(1-D(G(z)))]\end{aligned} \tag{1}$$

where $p_z(z)$ is a simple distribution (e.g., uniform or gaussian) and U denotes the utility function.

As an extension, Salimans et al. [34] first proposed feature-matching GANs (FM-GANs) to solve an SSL problem.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00138-020-01096-z) contains supplementary material, which is available to authorized users.

✉ Wenyuan Li
liwenyuan.zju@gmail.com

✉ Corey Arnold
cwarnold@ucla.edu

[1] University of California, Los Angeles, USA

[2] University of Texas, Austin, USA

Published online: 19 July 2020

Suppose we have a classification problem that requires classifying a data point x into one of K possible classes. A standard classifier takes x as input and outputs a K-dimensional vector of logits $\{l_1, \ldots, l_K\}$. Salimans et al. extended the standard classifier by simply adding samples from a GAN's G to the dataset, labeling them as a new "generated" class $y = K+1$, and correspondingly increasing the classifier's output dimension from K to $K+1$. They also found that using feature matching loss in G improved classification performance. The $(K+1)$-class discrimination objective with feature matching loss in G led to strong empirical results.

Empirically, FM-GANs demonstrate good performance on SSL classification tasks; however, the generated images from the generator are low-quality, i.e., the generator may create visually unrealistic images. Li et al. [19] realized that the generator and the discriminator in FM-GANs may not be optimal at the same time. Intuitively, assuming the generator can create good samples, the discriminator should identify these samples as fake samples as well as predict the correct class for them. To address this problem, they proposed a three-player game, Triple-GAN, to simultaneously achieve superior classification results and obtain a good image generator. Triple-GAN consisted of a generator G, a discriminator D, and a separate classifier C. C and G were two conditional networks that generated pseudo labels given real data, and pseudo data given real labels. To jointly evaluate the quality of the samples from the two conditional networks, D was used to distinguish whether a data-label pair was from the real labeled dataset or not. The improvements achieved by Triple-GAN were more significant as the number of labeled data decreased, suggesting that the generated data-label pairs can be used effectively to train the classifier. Meanwhile, Dai et al. [4] realized the same problem of the generator, but instead gave theoretical justifications of why using "bad" samples from the generator could boost SSL performance. Loosely speaking, they defined samples that form a complement set of the true data distribution in feature space as "bad" samples. By carefully defining the generator loss, the generator could create "bad" samples that forced C's decision boundary to lie between the data manifolds of different classes, which in turn improved generalization of C. Their model was called Bad GAN, which achieved better performance on multiple benchmark datasets compared to Triple-GAN.

Most recently, Li et al. [21] performed a comprehensive comparison between Triple-GAN and Bad GAN. They illustrated the distinct characteristics of the images the models generated, as well as each model's sensitivity to various amount of labeled data used for training. Furthermore, they showed that in the case of low amounts of labeled data, Bad GAN's performance decreased faster than Triple-GAN, and both models' performance was contingent on the selection of labeled samples; in other words, selecting non-representative samples would deteriorate the classification performance.

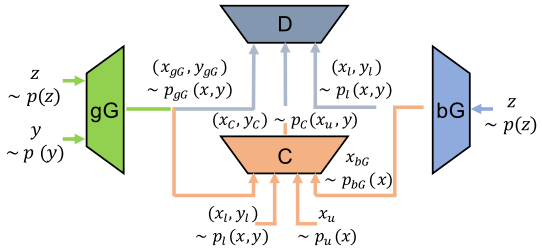

Fig. 1 Network architecture of UGAN. UGAN consists of four components: (1) a bad generator, bG, generates "bad" samples; (2) two conditional networks, gG and C, that generate pseudo labels given real data, and pseudo data given real labels; and (3) a separate discriminator, D, that distinguishes the generated data-label pair from the real data-label pair. "CE" denotes the cross entropy loss for supervised learning, while "BCE" denotes the binary cross-entropy loss that distinguish the real data and fake data generated by bG

In this paper, we present unified-GAN (UGAN), a semi-supervised learning framework that unifies both good and bad generated samples and takes advantage of them through adversarial training. Inspired by Triple-GAN and Bad GAN, we find that good and bad synthetic samples can be used for complementary purposes. Generated good image-label pairs can be used to train the classifier, while the bad samples can force the decision boundary to be between the data manifold of different classes. Hence, we leverage both good and bad generated samples in the proposed UGAN and achieve further performance improvement in SSL. Overall, our main contributions of this paper are: (1) we propose a novel SSL framework, UGAN, which simultaneously trains a good and a bad generator through adversarial training and takes advantage of both generated samples to boost SSL performance; (2) we analyze our proposed UGAN, theoretically prove its global optimum, and additionally put UGAN in the Expectation-Maximization (EM) framework and validate its non-increasing divergence property; and (3) we do extensive experiments to show that UGAN can improve upon Triple-GAN and Bad GAN classification results in SSL and show the effectiveness of the model with different amounts of labeled data.

2 Related work

Besides the aforementioned FM-GAN [34], Triple-GAN [19], and Bad GAN [4], several previous studies have also incorporated the idea of adversarial training in SSL. CatGAN [35] substituted the binary discriminator in standard GAN with a multi-class classifier and trained both the generator and discriminator using information theoretical criteria on unlabeled data. Virtual adversarial training (VAT) [26] effectively smoothed the classifier output distribution by seeking virtual adversarial samples. In adversarial learned inference

[6], the inference network approximated the posterior of latent variables given true data in an unsupervised manner. Another line of work has focused on manifold regularization [2]. Kumar et al. [15] estimated the manifold gradients at input data points and added an additional regularization term to a GAN, which promoted invariance of the discriminator to all directions in the data space. Lecouat et al. [17] achieved competitive results by performing manifold regularization using approximate Laplacian norm that was easily computed within a GAN.

Apart from adversarial training, there have been other efforts in SSL recently. One class of the most successful algorithms in SSL is based on pseudo labels [10,16,30,32,36]. Pseudo labels are artificial labels generated by the model, which play the same role as labels of manually annotated data. Γ model [32] evaluated unlabelled data with and without noise, and applied a consistency cost between the two predictions. It assumed a dual role as a teacher and a student. The teacher generated targets of unlabeled data, which were then used to train a student. Since the model itself generated the targets, they could be incorrect. To alleviate the problem, Π model [16] added noise at the inference time, and consequently a noisy teacher could yield more accurate targets. Π model was further improved by Temporal Ensembling [16], which maintained an exponential moving average (EMA) prediction for each of the training examples. Consequently, the EMA prediction of each example was formed by an ensemble of the model's current version and those earlier versions that evaluated the same example. This ensembling improved the quality of the predictions, and using the predictions as teacher signals improved results. Mean Teacher [36] averaged model weights to form a target-generating teacher model. Unlike Temporal Ensembling, Mean Teacher worked with large datasets and on-line learning, which was able to improve the speed of learning and classification accuracy simultaneously.

Our proposed UGAN is mainly inspired by Triple-GAN and Bad GAN, these models can be used for complementary purposes. We restrict our discussion to GAN-based model in most part of this paper. Nevertheless, it has a connection with those "teacher" models, as will be seen in Sect. 3, our model provides a smart way to generate input-label pairs and use them as teaching signals to improve the SSL results.

3 Method

To outline our approach, we consider the following SSL problem. Given a relatively small labeled set $(x_l, y_l) \sim p_l(x, y)$, where $y \in \{1, 2, \ldots, K\}$ is the label space for classification, and a large unlabeled set $x_u \sim p_u(x)$, the goal is to utilize the large amount of unlabeled data to predict the labels y of the unseen samples. Suppose the true data distribution is denoted as $p(x, y)$, we aim to obtain a classifier that can approximate the conditional distribution $p_C(y|x) \approx p(y|x)$. To achieve this, we will use an adversarial training process that enables the classifier to learn from both good and bad samples. Specifically, a good generator is able to generate good image-label pairs to train the classifier, while a bad generator generates samples that force the classifier's decision boundary between the data manifolds of different classes. As will be shown, our model takes advantage of both good and bad synthetic samples and improves the SSL results in a wide range of labeled training data.

3.1 Adversarial training process with four players

Our model consists of four parts: (1) a good generator, gG, that characterizes the conditional distribution $p_{gG}(x|y) \approx p(x|y)$; (2) a bad generator, bG, that takes in a latent vector z and outputs "bad" samples [4]; (3) a classifier, C, that characterizes the conditional distribution $p_c(y|x) \approx p(y|x)$; and (4) a discriminator, D, that distinguishes whether a pair of data (x, y) comes from the true distribution $p(x, y)$ or not. All the components are parameterized as neural networks, as shown in Fig. 1a.

We assume that the samples from both real data $p(x)$ and real label $p(y)$ can be easily obtained.[1] In our model, gG produces a pseudo input-label pair by first drawing $y \sim p(y)$ and latent vector $z \sim p(z)$ (we use a uniform distribution for z in our experiments), and then generating $x_{gG} \sim p_{gG}(x|y, z)$. bG generates bad samples by transforming the latent vector $z \sim p(z)$ as in a traditional GAN to obtain $x_{bG} \sim p_{bG}(x|z)$. C takes in four different types of samples (i.e., labeled data, unlabeled data, samples from gG, and samples from bG) and produces pseudo labels y for them following the conditional distribution $p_C(y|x)$. For the labeled data x_l, and the gG generated samples x_{gG}, we expect C to put them into the right class (i.e., either the class y_l of the labeled data x_l, or the conditional labels y based on which x_{gG} are generated). For the generated samples from bG $x_{bG} \sim p_{bG}(x|z)$, and unlabeled data $x_u \sim p_u(x)$, we expect C to put them into the $(K+1)$th class (i.e., the "fake" class) and one of the K classes of real data, respectively. Due to the fact that the softmax layer is over-parameterized, we can still model C with K neurons at the output layer by modifying the loss function (see details in Supplementary Appendix A). D accepts the input-label pairs generated by both C $(x_C, y_C) \sim p(x_u)p_C(y|x_u)$, and gG $(x_{gG}, y_{gG}) \sim p(y)p_{gG}(x|y)$, and the pairs from the labeled data distribution $(x_l, y_l) \sim p_l(x, y)$ for judgement. D treats the labeled data pairs as positive samples, while the pairs from both gG and C as negative. We refer the loss function

[1] In semi-supervised learning, $p(x)$ is the empirical distribution of inputs and $p(y)$ is assumed same to the distribution of labels on labeled data, which is uniform in our experiments.

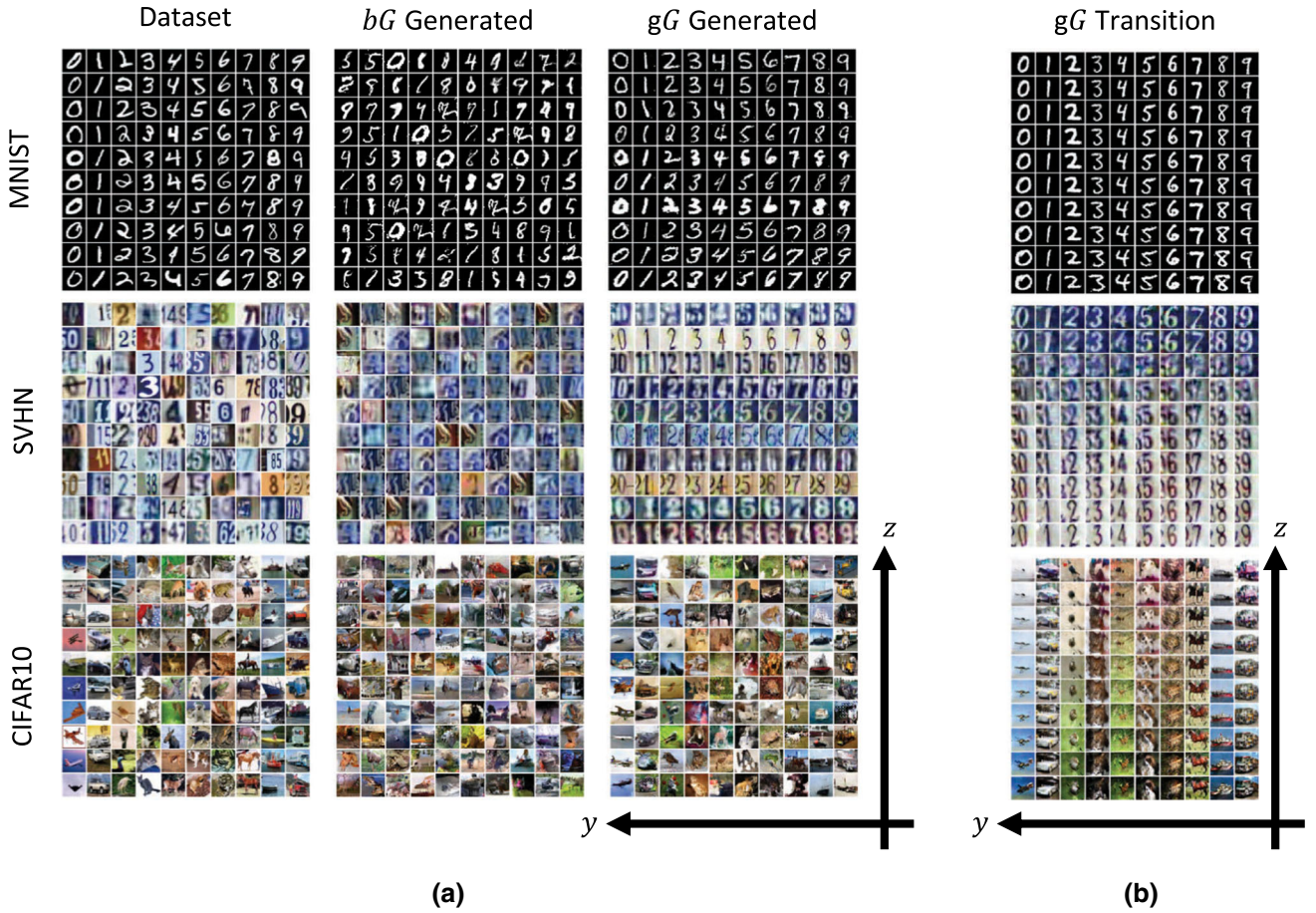

Fig. 2 **a** Left: randomly selected data from datasets; mid: bG generated images; right: gG generated images sampled by varying the class label y in the horizontal axis and the latent vectors z in the vertical axis. **b** Class-conditional latent space interpolation. The vertical axis is the direction for latent vector interpolation, while the horizontal axis for varying the class labels

of gG as[2]

$$L_{gG} = \mathbb{E}_{x,y\sim p_{gG}(x,y)}[\log(1 - p_D(x, y)] \tag{2}$$

The loss function of bG is

$$L_{bG} = -\mathcal{H}(p_{bG}(x)) + \left\|\mathbb{E}_{x\sim p_u(x)}(\boldsymbol{f}(x)) - \mathbb{E}_{x\sim p_{bG}(x)}(\boldsymbol{f}(x))\right\|_2^2 \tag{3}$$

where $-\mathcal{H}(p_{bG}(x))$ measures the negative entropy of bG generated samples. $-\mathcal{H}(p_{bG}(x))$ is used to avoid collapsing while increasing the coverage of bG. The second term is feature matching loss, where $\boldsymbol{f}(x)$ denotes a feature map of an intermediate layer of C. D's loss function becomes

$$\begin{aligned} L_D = &-\mathbb{E}_{x,y\sim p_l(x,y)}[\log(p_D(x, y)] \\ &-\frac{1}{2}\mathbb{E}_{x,y\sim p_{gG}(x,y)}[\log(1 - p_D(x, y)] \\ &-\frac{1}{2}\mathbb{E}_{x,y\sim p_C(x,y\le K)}[\log(1 - p_D(x, y)] \end{aligned} \tag{4}$$

where D treats the labeled data as positive samples, and the pseudo input-label pairs from both gG and C as negative samples. Finally, the loss function of C consists of four components,

$$\begin{aligned} L_{C_1} &= -\mathbb{E}_{x,y\sim p_l(x,y)}[\log(p_C(y|x, y \le K)] \\ L_{C_2} &= -\mathbb{E}_{x,y\sim p_{gG}(x,y)}[\log(p_C(y|x, y \le K)] \\ L_{C_3} &= -\mathbb{E}_{x\sim p_u(x)}[\log(1 - p_C(y = K + 1|x)] \\ L_{C_4} &= -\mathbb{E}_{x\sim p_{bG}(x)}[\log(p_C(y = K + 1|x)] \end{aligned} \tag{5}$$

[2] In practice, we use $L_{gG} = -\mathbb{E}_{x,y\sim p_{gG}(x,y)}[\log(p_D(x, y)]$ to ease the training process [9].

and the total loss for C is

$$L_C = L_{C_1} + \lambda_0 L_{C_2} + \lambda_1 L_{C_3} + \lambda_2 L_{C_4} \tag{6}$$

where L_{C_1} and L_{C_2} denote the cross-entropy loss for labeled and gG generated samples, L_{C_3} forces C to put the unlabeled data into real classes, while L_{C_4} forces C to put the bG generated samples into the "fake" class. $\lambda_{0,1,2}$ is a hyperparameter used to balance each loss component.

The model defined by (2)–(6) achieves its equilibrium if and only if $p(x, y) = p_{gG}(x, y) = p_C(x, y \leq K)$. In other words, incorporating the bad samples does not change the equilibrium point of Triple-GAN (see Sect. 3.2.1). Our model consists of three adversarial parts: (1) gG tries to fool D by generating realistic images conditioned on label y; (2) C tries to fool D by generating good labels for unlabeled images; and (3) bG tries to fool C by generating images that are close to the data manifold. At convergence, D cannot distinguish both $p_{gG}(x, y)$ and $p_C(x, y)$ from the true data distribution $p(x, y)$, which indicates that we have obtained both a good gG and a good C. Bad samples from bG accelerate this process and improve the generalization of C.

One key problem of SSL is the limited amount of labeled data. A powerful D may memorize the empirical distribution of the labeled data and reject other types of samples from the true data distribution. Limited labeled data also restrict gG to explore a larger space of the true data distribution. To address this problem, we adopt the practical techniques in Li et al. [19]. We generate pseudo labels through C for some unlabeled data and use these pairs as positive samples of D. This introduces some bias to the target distribution of D, but using the EM framework to analyze the training procedure (see Sect. 3.2.2), we are able to prove the rationality of this choice. Moreover, since C converges quickly, this operation provides a way to enable gG to explore a much larger data manifold that includes both the labeled and unlabeled data information. As illustrated in Fig. 1b, C is able to provide pseudo labels for the unlabeled data, while D will judge if the pseudo labels are reliable or not. This in return will affect the evolution of gG that will take advantage of the unlabeled data to generate good images. Generated good image-label pairs that implicitly contain unlabeled data information will eventually benefit C. This works extremely well for relatively simple datasets like MNIST, and under the circumstance where only an extremely low amount of labeled data is available.

3.2 Theoretical analysis

We now give theoretical justification for our four-player game based on the loss functions as mentioned above. We mainly focus on two important properties of our model: (1) the global optimum of the game is the true distribution, which satisfies $p(x, y) = p_{gG}(x, y) = p_C(x, y|y \leq K)$; and (2) the KL divergence between the conditional density of C and the true density, $\mathrm{KL}(p(y|x)||p_C(y|x, y \leq K))$, is non-increasing after each iteration when we assume the maximum likelihood estimate (MLE) of C is obtained. A detailed proof of these properties is provided in Supplementary Appendix C.

3.2.1 Global optimum

We first show that the optimal D balances between the true data distribution and the mixture distribution defined by C and gG, as summarized in Lemma 1.

Lemma 1 *For any fixed C and gG, the optimal D of the game defined by loss functions (2)–(5) is*

$$D^*_{C,gG,bG}(x, y) = \frac{p_l(x, y)}{p_l(x, y) + p_{\frac{1}{2}}(x, y)}, \tag{7}$$

where $p_{\frac{1}{2}}(x, y) = \frac{1}{2} p_{gG}(x, y) + \frac{1}{2} p_C(x, y|y \leq K)$.

Given $D^*_{C,gG,bG}$, we can plug the optimal D^* into (4) and get a value function $V(C, gG, bG)$.

$$\begin{aligned} V_{C,gG,bG}(x, y) =& -\mathbb{E}_{x,y\sim p_l(x,y)}[\log(p_{D^*}(x, y)] \\ & - \frac{1}{2}\mathbb{E}_{x,y\sim p_{gG}(x,y)}[\log(1 - p_{D^*}(x, y)] \\ & - \frac{1}{2}\mathbb{E}_{x,y\sim p_C(x,y\leq K)}[\log(1 - p_{D^*}(x, y)] \\ =& -\mathbb{E}_{x,y\sim p_l(x,y)}[\log(\frac{p_l}{p_l + p_{1/2})}] \\ & - \frac{1}{2}\mathbb{E}_{x,y\sim p_{gG}(x,y)}[\log(\frac{p_{1/2}}{p_l + p_{1/2})}] \\ & - \frac{1}{2}\mathbb{E}_{x,y\sim p_C(x,y\leq K)}[\log(\frac{p_{1/2}}{p_l + p_{1/2})}] \end{aligned} \tag{8}$$

Now the left problem is to maximize the $V(C, gG, bG)$, so that gG and C confuse D most. For that, we have the following theorem:

Theorem 1 *The global maximum of $V(C, gG, bG)$ is achieved only when $p_l(x, y) = p_{gG}(x, y) = p_C(x, y|y \leq K)$.*

From (8), it is easy to see the global maximum is achieved if and only if $p_l(x, y) = p_{1/2}(x, y)$. By introducing the cross-entropy loss in (5) L_{C1}, we enforce $p_C(x, y|y \leq K) = p_l(x, y)$. Therefore, the global optimality will achieve if and only if $p_l(x, y) = p_{gG}(x, y) = p_C(x, y|y \leq K)$. (See more details in Supplementary Appendix C)

We now consider the case for $p_C(y = K + 1|x)$ with the following Corollary 1.

Corollary 1 *The optimal classifier C will have* $p_C(y = K + 1|x \sim p_u(x)) = 0$ *and* $p_C(y = K + 1|x \sim p_{bG}(x)) = 1$.

Corollary 1 indicates that optimal C will put bG generated images into $K + 1$ class (i.e., "fake" class), while put unlabeled data into real classes.

3.2.2 Non-increasing divergence property

Our goal is to estimate the conditional distribution $p(y|x)$ with a parameterized C modeled as $p_\theta(y|x, y \leq K)$. The objective function can be written as minimizing $\mathrm{KL}(p(y|x)||p_\theta(y|x, y \leq K))$. In the SSL setting, we only have part of the labels y, so we can thus rewrite the problem as minimizing $\mathrm{KL}(p(y_l|x)||p_\theta(y_l|x, y \leq K))$. One natural way to facilitate the model performance is using the EM algorithm to first infer the label of x_u and then update based on the complete data [28]. In our four-player game, in addition to the predicted label y_u from unlabelled data x_u, we further introduce (x_{gG}, y_{gG}) pairs from gG as latent variables, denoted as $Z = \{x_{gG}, y_{gG}, y_u\}$. We then interpret our mechanism from a variational view of the EM algorithm to illustrate the non-increasing property of the KL divergence.

Property I. Chain rule of KL divergence:

$$\begin{aligned}\mathrm{KL}(P(X, Z)||P_\theta(X, Z)) &= \mathrm{KL}(P(X)||P_\theta(X)) \\ &\quad + \mathbb{E}_{x\sim P(X)}[\mathrm{KL}(P(Z|x)||P_\theta(Z|x))].\end{aligned} \tag{9}$$

By Property I, we can rewrite our objective function as:

$$\begin{aligned}&\min_\theta \mathrm{KL}(p(y_l|x)||p_\theta(y_l|x, y \leq K)) \\ &= \min_\theta \min_{p(Z|x)} \mathrm{KL}(p(y_l, Z|x)||p_\theta(y_l, Z|x, y \leq K)),\end{aligned} \tag{10}$$

which is an iterative minimization procedure. Following the EM algorithm, we have an *E-step* and an *M-step* in UGAN. More specifically, for the *E-step* at the sth iteration, given parameters θ_s of C, we have:

$$\begin{aligned}p(Z|x) &= p_{\theta_s}(Z|x) \\ &= p_{gG}(x_{gG}, y_{gG}|x_u, x_l, y_u, y_l)p_{\theta_s}(y_u|x_u),\end{aligned} \tag{11}$$

which indicates the procedure that C first predicts labels for unlabelled data and then sends them to D and gG to generate good pseudo pairs (x_{gG}, y_{gG}). After gathering the latent variables, the *M-step* is:

$$\begin{aligned}\theta_{s+1} &= \mathrm{argmin}_\theta \mathrm{KL}(p(y_l, Z|x)||p_\theta(y_l, Z|x, y \leq K)) \\ &= \mathrm{argmax}_\theta \mathbb{E}_{(y_l,Z|x)\sim f}[\log p_\theta(y_l, Z|x, y \leq K)],\end{aligned} \tag{12}$$

where $f = p_{\theta_s}(Z|x_u)p_l(y_l|x_l)$. This will result in θ_{s+1} being the MLE based on the data at current iteration s. By applying the EM mechanism, we can inherit its non-increasing property which is stated in the following Corollary 2.

Corollary 2 *If applying the iterative procedure described in* (11) *and* (12)*, and the exact maximization can be obtained at* (12) *for each iteration, then*

$$\begin{aligned}&KL(p(y_l|x)||p_{\theta_{s+1}}(y_l|x, y \leq K)) \\ &\leq KL(p(y_l|x)||p_{\theta_s}(y_l|x, y \leq K))\end{aligned} \tag{13}$$

The non-increasing property guarantees that our classifier will be improved after each iteration under the ideal situation. Though we make some approximations during training process in practice, it still provides us a high-level justification on why the algorithm should work.

4 Experiments and discussion

We now present UGAN's performance on MNIST [18], SVHN [27], and CIFAR10 [14] datasets (see details of datasets in Supplementary Appendix D). We implement our model based on Tensorflow 1.10 [8] and optimize it on NVIDIA Titan X GPUs. The detailed architecture can be found in Supplementary Appendix E. The gG generated images are not applied until the number of epochs reaches a threshold such that gG can generate reliable image-label pairs. For MNIST and SVHN, we choose 200, while for CIFAR10 we choose 400. Batch size is an important parameter that affects model performance [21]. In our experiments, we use 50 for bG on MNIST and SVHN, 25 for bG on CIFAR10. For gG, we fix batch size as 100. All of the other hyperparameters including relative weights and parameters in Adam [11] are fixed according to [4,19,34] across all of the experiments.

4.1 Classification

We report our classification accuracy, along with other GAN-based SSL methods, on benchmark datasets in Table 1. Our results show that UGAN consistently improves performance, and achieves the best results on all of the datasets without the use of data augmentation, such as rotation, flip, etc.

To understand our model's behavior over different numbers of labeled data, we re-implemented Triple-GAN and Bad GAN, and performed an extensive investigation by varying the amount of labeled data. Following common practice, this was done by omitting different amounts of the underlying labeled dataset [29,33,34,36]. The labeled data used for training were randomly selected stratified samples unless otherwise specified. For fair comparison, we used the same network architecture for each component in all models (see Supplementary Appendix E). Table 2 shows the results of the experiments on MNIST. The similarity of our results to those reported in the original papers suggests that our reproduced models are accurate instantiations of Triple-GAN

and Bad GAN. We observe that with a medium amount of labeled data (e.g., MNIST $n = 100$), Bad GAN performs better than Triple-GAN. However, with smaller amounts of labeled data, Triple-GAN performs better, which demonstrates that it is less sensitive to the amount of labeled data than Bad GAN. UGAN inherits the good properties from both of them, resulting in a constant improvement across all cases (see results on SVHN and CIFAR10 in Supplementary Appendix F). Another interesting observation is that the selection of labeled data plays a crucial role in the low-labeled data regime, that is, selecting representative labeled data with which to train is the key to achieving good performance. This issue is further discussed in Supplementary Appendix H.

To further validate that our model significantly improves the baseline model, we have performed Welch's t-test (see Supplementary Appendix G). We found that our model significantly improves Triple-GAN and Bad-GAN with a maximum p-value in order of 1e-5 for both SVHN and CIFAR10 datasets. For the MNIST dataset, we have found that for some results, our model's performance is not significantly different from the literature, such as the Bad-GAN case with 100 labeled samples. This is because the MNIST classification task is a relatively easy task, and the previous study has already achieved very good performance. The small p-values for SVHN and CIFAR10 have demonstrated that our model improves the baseline model significantly on more complex datasets.

4.2 Image generation

UGAN is able to train a gG and a bG simultaneously (see an evolution of the generated images in Supplementary Appendix I). In Fig. 2a, we show the images generated by gG and bG after training. Our gG is able to generate clear images and meaningful samples conditioned on class labels, while bG generates "bad" images that look like a fusion of samples from different classes. We quantitatively evaluate generated samples on CIFAR10 via the inception score following [34]. The value of gG generated samples is 4.19 ± 0.07, while that of bG generated samples is 3.31 ± 0.02. In addition, gG retains Triple-GAN's advantage in that it is able to disentangle classes and styles. In Fig. 2a, the gG generated images are sampled by varying the class label y in the horizontal axis and the latent vectors z in the vertical axis. The latent vector z encodes meaningful physical appearances, such as scale, intensity, orientation, color, etc., while the label y controls the semantics of the generated images. Furthermore, gG can transition smoothly from one style to another with different visual factors without losing the label information as shown in Fig. 2b. This demonstrates that gG can learn meaningful latent representations instead of simply memorizing the training data.

4.3 Hyper-parameters sensitivity analysis

We perform hyper-parameter sensitivity analysis along with some network architecture effects. We discover that the hyper-parameters used in Triple-GAN and Bad GAN are also good for UGAN. In fact, aside from batch size, we use the same hyper-parameters across all three datasets, and consistently achieve good results. UGAN is not sensitive to the learning rate due to the usage of Adam optimization as shown in Table 3. However, UGAN is quite sensitive to the batch size in training. Our results indicate the noise induced by mini-batch benefits the bG, while it hurts the gG capability to model the true data distribution. We also find that a weight-norm layer is important to ease GAN's training. UGAN doesn't usually converge when the layer is taken out. A smaller network architecture of C would not result in a significant drop in the performance. We use a C with filter size $\{32, 64, 96\}$ and get 96.27% on SVHN $n = 4000$. For details on how our hyper-parameters sensitivity analysis is performed, we defer readers to Supplementary Appendix J.

4.4 Effectiveness of good and bad generators

As discussed in Sect. 4.1, UGAN achieves consistent improvement across all the cases due to inheriting the best properties of Triple-GAN and Bad GAN. In Fig. 3a, we demonstrate a comparison of Validation Accuracy vs. Training Epochs for our implemented Triple-GAN, Bad GAN and UGAN on SVHN $n = 1000$. Note that for Triple-GAN, we trained it to 1000 epochs, but only show the first 400 epoch in the figure. Qualitatively, we observe three separate training phases:

1. In **Phase I**, the performance of Bad GAN and UGAN are worse than Triple-GAN. We speculate this is due to the fact that Triple-GAN C deals with a classification of K classes, while Bad-GAN and UGAN, C deal with $K + 1$ classes.
2. In **Phase II**, Bad GAN and UGAN start to surpass Triple-GAN, which indicates bG generated samples start to exert an effect on the classification boundary. UGAN also performs better than Bad GAN in this phase thanks to the adversarial game that requires C to produce reliable pseudo labels for unlabeled data to fool D.
3. In **Phase III**, we start to use gG generated samples to train C. UGAN surpasses both Triple-GAN and Bad GAN by a clear margin. From the perspective of C, gG generates samples that are used to complement the lack of training data in SSL, bG generated samples are used to force the decision boundary to lie in the correct place, and D requires C to keep moving itself toward the true data distribution $p(x)p_C(y|x, y \leq K) \approx p(x, y)$. All of these factors contribute to the final performance of UGAN.

Table 1 Comparison with state-of-the-art methods on three benchmark datasets

Methods	MNIST $n = 100$ (%)	SVHN $n = 1000$ (%)	CIFAR10 $n = 4000$ (%)
CatGAN [35]	98.09 ± 0.1	–	80.42 ± 0.46
ALI [6]	–	92.58 ± 0.65	82.01 ± 1.62
VAT [26]	98.64	93.17	85.13
Π Model [16]	–	94.57 ± 0.25	83.45 ± 0.29
Γ Model [32]	99.11 ± 0.50	–	79.40 ± 0.47
Mean Teacher [36]	–	96.05 ± 0.19	84.27 ± 0.31
FM-GAN [34]	99.07 ± 0.07	91.89 ± 1.3	81.37 ± 2.32
Triple-GAN [19]	99.09 ± 0.58	94.23 ± 0.17	83.01 ± 0.36
Bad-GAN [4]	99.21 ± 0.10	95.75 ± 0.03	85.59 ± 0.30
UGAN	99.21 ± 0.08	96.49 ± 0.09	85.66 ± 0.06

Only methods without data augmentation are included. Results are averaged over 10 runs and shown in terms of mean accuracy ± SD

Table 2 Test accuracy on semi-supervised MNIST

Model	Test accuracy for a given number of labeled samples			
	20 (%)	50 (%)	100 (%)	200 (%)
FM-GAN [34]	83.23 ± 4.52	97.79 ± 1.36	99.07 ± 0.07	99.10 ± 0.04
Bad GAN [4]	–	–	99.21 ± 0.10	–
Triple-GAN [19]	95.19 ± 4.95	98.44 ± 0.72	99.09 ± 0.58	99.33 ± 0.16
Bad GAN[b]	88.38 ± 3.08[a]	96.24 ± 0.16	99.17 ± 0.03	99.20 ± 0.03
Triple-GAN[b]	95.93 ± 4.45[a]	98.68 ± 1.12	99.07 ± 0.46	99.17 ± 0.08
UGAN	97.34 ± 6.86[a]	98.92 ± 0.13	99.21 ± 0.08	99.35 ± 0.05

Results are averaged over 10 runs and shown in terms of mean accuracy ± SD
[a]Denotes hand selection of labeled data
[b]Denotes our implementation of the model

Table 3 Initial learning rate effect on model performance

Learning rate	$lr = 1e{-}2$	$lr = 1e{-}3$	$lr = 5e{-}4$	$lr = 3e{-}4$
Accuracy	99.13%	99.18%	99.24%	99.18%

The experiments are done on MNIST $n = 100$. Despite the differences of the training loss in the initial stage, the final results are not significant different after training 400 epochs

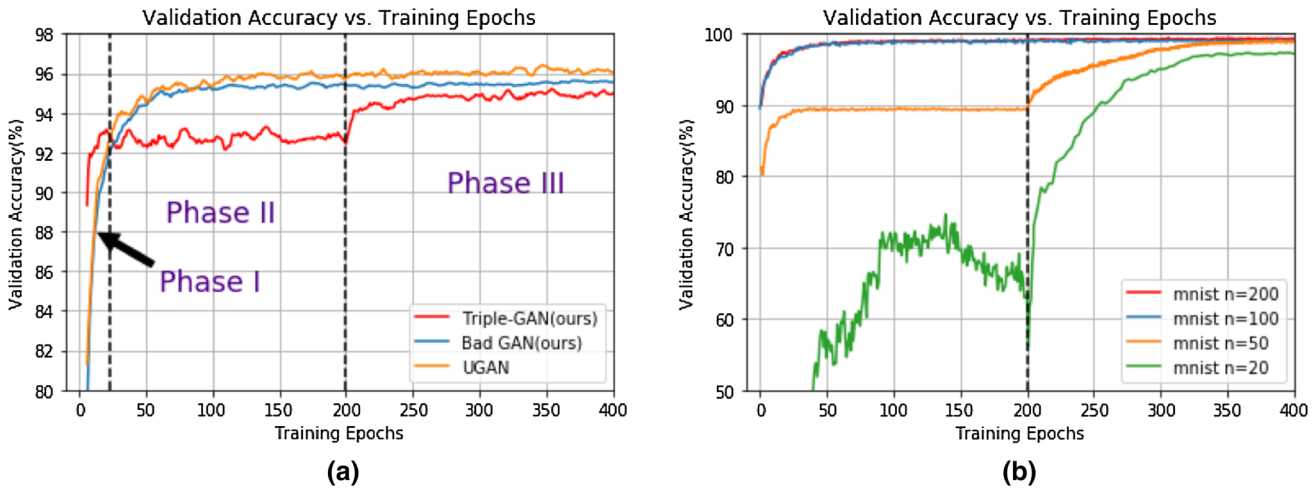

Fig. 3 **a** Comparison of validation accuracy vs. training epochs on our implemented triple-GAN, Bad GAN and UGAN. The experiments are performed on SVHN $n = 1000$. **b** UGAN validation accuracy vs. training epochs under various amounts of labeled data on MNIST

Similar observations can also be found in Supplementary Appendix K on MNIST and CIFAR10. Moreover, we hypothesize that for fewer labeled data, gG plays an important role, as gG is able to model the class-aware data distribution under weak supervision and use them to complement the lack of the training samples. While for larger labeled data, bG plays a more important role by generating complementary samples and forcing the decision boundary to lie between the data manifolds of different classes. Empirically, we show our model's validation accuracy under various amounts of labeled data on MNIST in Fig. 3b. As can be seen, when we push the number of labeled data to extremely low numbers, the training curve becomes more like that in Triple-GAN, i.e., a bump is shown clearly at epoch $= 200$ when we start to use gG generated samples to train C. However, we do not find a similar transition on SVHN and CIFAR10 (see Supplementary Appendix K). One possible explanation is that when we use too few labeled data, gG fails to model the conditional distribution due to the complexity of SVHN and CIFAR10. Note that we only used traditional techniques for training the GAN. With recent advances in generating high-quality images using GANs [3,23,25], our model may be able to achieve further performance improvements on more complex datasets with even fewer labeled data.

5 Limitation of the study

Here, we discuss some limitations of our work and provide potential research directions that could help address these limitations.

We note that we assume the marginal distribution $p(y)$ to be uniform, which is easy to sample for the generation process. However, it is not always true for other applications where $p(y)$ is no longer uniform. In these cases, we expect that the non-uniform label distribution has both effect on good sample generation and classification. For good sample generation, the generator will have difficulty capturing features for the minority class. For classification, the classifier tends to cheat by always predicting the majority class. These problems are expected to be more sever when the dataset is highly skewed. One potential future research direction is to investigate how a non-uniform label distribution will affect our model and how common data balancing methods such as upsampling and data augmentation can provide help to it.

Another area for potential investigation is to generate high-resolution, large-scale images, so that our model can be used in more complex scenarios. In this paper, we have only applied the model to relatively simple datasets with less complexity, such as MNIST (28*28), SVHN (32*32) and CIFAR10 (32*32). Part of the reason is that the model in the current form is not able to generate reliable image-label pairs on large scale. With the advancement in high-resolution large-scale image generation using GAN recently, we expect that our model will be able to applied to much more complex scenarios such as ImageNet classification, and pixel-wise segmentation.

6 Conclusions

We have presented unified-GAN (UGAN), a new GAN framework for semi-supervised learning. By learning from good and bad samples through adversarial training, we have demonstrated that our model performs better on image classification tasks across several benchmark datasets and under a range of labeled training data. We envision that UGAN can be used in a variety of scenarios, such as healthcare, where obtaining labeled data can be expensive and time-consuming. We also consider adapting UGAN to other types of data such as text (e.g., improving SSL text categorization performance for 20 newsgroups, Reuters, NYTimes, Wiki, PubMed etc.).

References

1. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A Large-scale Video Classification Benchmark (2016). arXiv:1609.08675
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**, 2399–2434 (2006)
3. Brock, A., Donahue, J., Simonyan, K.: Large Scale Gan Training For High Fidelity Natural Image Synthesis. (2018). arXiv:1809.11096
4. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.R.: Good semi-supervised learning that requires a bad gan. In: Advances in Neural Information Processing Systems, pp. 6510–6520 (2017)
5. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
6. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference (2016). arXiv:1606.00704
7. Gan, Z., Chen, L., Wang, W., Pu, Y., Zhang, Y., Liu, H., Li, C., Carin, L.: Triangle generative adversarial networks. In: Advances in Neural Information Processing Systems, pp. 5247–5256 (2017)
8. Girija, S.S.: Tensorflow: large-scale machine learning on heterogeneous distributed systems. Software available from https://www.tensorflow.org/ (2016)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
10. Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Label propagation for deep semi-supervised learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5070–5079 (2019)
11. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 (2014)

12. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. arXiv:1312.6114 (2013)
13. Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A et al.: Openimages: A Public Dataset for Large-scale Multi-label and Multi-class Image Classification, vol. 2, p. 3. Dataset available from https://github.com/openimages (2017)
14. Krizhevsky, A., Hinton, G.: Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer (2009)
15. Kumar, A., Sattigeri, P., Fletcher, T.: Semi-supervised Learning with Gans: Manifold Invariance with Improved Inference. In: Advances in Neural Information Processing Systems, pp. 5534–5544 (2017)
16. Laine, S., Aila, T.: Temporal Ensembling for Semi-supervised Learning. arXiv:1610.02242 (2016)
17. Lecouat, B., Foo, C.-S., Zenati, H., Chandrasekhar, V.R.: Semi-supervised Learning with Gans: Revisiting Manifold Regularization. arXiv:1805.08957 (2018)
18. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
19. Li, C., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 4088–4098 (2017)
20. Li, W., Wang, Y., Cai, Y., Arnold, C., Zhao, E., Yuan, Y.: Semi-supervised Rare Disease Detection Using Generative Adversarial Network. arXiv:1812.00547 (2018)
21. Li, W., Wang, Z., Li, J., Polson, J., Speier, W., Arnold, C.: Semi-supervised Learning Based on Generative Adversarial Network: A Comparison Between Good Gan and Bad Gan Approach. arXiv:1905.06484 (2019)
22. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: common objects in context. In: European Conference on Computer Vision, pp. 740–755. Springer (2014)
23. Lucic, M., Tschannen, M., Ritter, M., Zhai, X., Bachem, O., Gelly, S.: High-Fidelity Image Generation with Fewer Labels. arXiv:1903.02271 (2019)
24. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. arXiv:1411.1784 (2014)
25. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral Normalization for Generative Adversarial Networks. arXiv:1802.05957 (2018)
26. Miyato, T., Maeda, S.I., Koyama, M., Ishii, S.: Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1979–1993 (2018)
27. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: Advances in Neural Information Processing Systems (2011)
28. Nigam, K., McCallum, A., Mitchell, T.: Semi-supervised text classification using EM. In: Semi-supervised Learning, pp. 33–56 (2006)
29. Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., Carin, L.: Variational autoencoder for deep learning of images, labels and captions. In: Advances in Neural Information Processing Systems, pp. 2352–2360 (2016)
30. Qiao, S., Shen, W., Zhang, Z., Wang, B., Yuille, A.: Deep co-training for semi-supervised image recognition. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 135–152 (2018)
31. Radford, A., Metz, L., Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434 (2015)
32. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.; Semi-supervised Learning with Ladder Networks. In: Advances in Neural Information Processing Systems, pp. 3546–3554 (2015)
33. Sajjadi, M., Javanmardi, M., Tasdizen, T.: Mutual exclusivity loss for semi-supervised deep learning. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 1908–1912. IEEE (2016)
34. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems, pp. 2234–2242 (2016)
35. Springenberg, J.T.: Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. arXiv:1511.06390 (2015)
36. Tarvainen, A., Valpola, H.: Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results. In: Advances in Neural Information Processing Systems, pp. 1195–1204 (2017)
37. Weston, J., Ratle, F., Mobahi, H., Collobert, R.: Deep learning via semi-supervised embedding. In: Neural Networks: Tricks of the Trade, pp. 639–655. Springer (2012)

Zichen Wang received the B.E. degree in information engineering from Zhejiang University, China, in 2018. He is currently a Ph.D. student in the Medical Image Informatics (MII) group at UCLA. He is advised by Prof. Corey Arnold. He is interested in machine learning analysis on biomedical images and deep learning with applications to healthcare. His current research topic focuses on digital pathology and drug discovery.

Yuguang Yue received the B.S. degree in applied mathematics from Fudan University, China, in 2015 and the M.Sc. degree in biostatistics from the University of California, Los Angeles, in 2017. He is currently pursuing his Ph.D. degree in statistics and data science in the University of Texas at Austin. His recent research interests include Bayesian statistics with a focus on probabilistic generative models such as reinforcement learning and variational auto-encoder.

Jiayun Li is a Ph.D. candidate in Computational Diagnostics group, UCLA. Her research focuses on developing weakly- or semi-supervised models to learn deep representations from large-scale whole slide image datasets, and combine histopathological features, imaging representation and clinical variables for disease progression prediction.

William Speier is an Assistant Professor in the Medical Imaging Informatics and Computational Diagnostics Laboratories at UCLA. He received his B.S. in Biomedical Engineering and Statistics and his M.S. in Computer Science from Johns Hopkins University, and his Ph.D. in Biomedical Engineering at UCLA. His research includes medical image analysis, natural language processing, and brain-computer interfaces.

Mingyuan Zhou is an associate professor of statistics in the McCombs School of Business at the University of Texas at Austin. He received his Ph.D. in electrical and computer engineering from Duke University in 2013. His research lies at the intersection of Bayesian statistics and machine learning, covering a diverse set of research topics in statistical theory and methods, probabilistic models, statistical inference for big data, and deep learning. He is currently focused on advancing both statistical inference with deep learning and deep learning with probabilistic methods. He has served as the 2018–2019 treasurer of the Bayesian Nonparametrics Section of the International Society for Bayesian Analysis, and as area chairs for leading machine learning conferences, including NeurIPS 2017-2020, ICLR 2019, and AAAI 2020.

Corey Arnold is an Associate Professor at UCLA where he is a member of the Computational Diagnostics Lab (CDx) and holds appointments in the Departments of Radiology and Pathology. He received his Ph.D. and M.S. degrees from UCLA, and his B.S. degree from the University of Wisconsin, Madison. His areas of research include machine learning, medical image analysis, computational phenotyping, and multimodal disease modeling.