Generative Adversarial Nets with Labeled Data by Activation Maximization

Zhiming Zhou [1] **Shu Rong** [2] **Han Cai** [1] **Weinan Zhang** [1] **Yong Yu** [1] **Jun Wang** [3]

Abstract

In this paper, we study the impact and role of multi-class labels on adversarial training for generative adversarial nets (GANs). Our derivation of the gradient shows that the current GAN model with labeled data still results in undesirable properties due to the overlay of the gradients from multiple classes. We thus argue that a better gradient should follow the intensity and direction that maximize each sample's activation on one and the only one class in each iteration, rather than weighted-averaging their gradients. We show, mathematically, that the proposed activation-maximized adversarial training (AM-GAN) is a general one covering two major complementary solutions exploring labeled information. Additionally, we investigate related metrics for evaluating generative models. Empirical experiments show that our approach has achieved the best Inception score (8.34) compared with previously reported results. Moreover, our adversarial training produces faster convergence with no mode collapse observed.

1. Introduction

Generative adversarial nets (GANs) (Goodfellow et al., 2014) as a new way of generating samples has recently drawn much attention. Generally, GAN consists of two distinct neural networks competing with each other: the generator network aims to generate samples in order to approximate the underlying data distribution, whereas the discriminator network aims to distinguish a sample as to whether it is real or generated from the generator.

Since then, variants of GANs have been proposed. Denton et al. (2015) used cascade Laplacian Pyramid to generate images in a coarse-to-fine fashion. Radford et al. (2015) proposed a class of deep convolution network structure which makes GAN training more stable. Larsen et al. (2015) combined variational autoencoder (Kingma & Welling, 2013) and GAN to provide better training. Che et al. (2016) proposed to add mode regularizer and train GAN in a manifold-diffusion fashion. Metz et al. (2016) proposed to unroll discriminator's optimization for generator's objective. Arjovsky & Bottou (2017) proposed to introduce annealing noise to the input of discriminator to relieve the zero measure of intersection problem.

Recent work, Wasserstein GAN (Arjovsky et al., 2017), leveraged the Wasserstein distance as an alternative objective to the log-likelihood of the traditional GAN. Such a new objective successfully avoids the gradient vanishing problem when the distributions of the real data and the generated one are significant different.

SeqGAN (Yu et al., 2016) extended GAN to be applied to sequential, discrete data such as text or music MIDI data by using policy gradient. There is also a potential to make use of the generator of GAN to form a manifold in the data space (Zhu et al., 2016; Brock et al., 2016). Related applications include interactive image navigation, image editing, image inpainting etc.

Besides directly using GAN to generate samples, GAN has been leveraged as an additional loss in various tasks to push the output to follow a certain distribution. Ledig et al. (2016); Nguyen et al. (2016b) showed improved high-resolution and photo-realistic image generation results via introducing a log-likelihood term of a GAN discriminator's estimation as real images.

Although GAN has been applied in various applications, its high quality results were mostly shown with a combination of other methods (Nguyen et al., 2016b), or with a specially-designed multi-level network structure (Zhang et al., 2016; Huang et al., 2016). It is because the training of GAN itself still has quite a few problems, including the difficulty of convergence, mode collapse, and low sample quality (Goodfellow, 2016).

In order to improve GAN training, we also see another branch of research on incorporating additional information. For instance, the conditional version of GAN (Mirza & Osindero, 2014) is used in various supervised and unsupervised conditional generation tasks. Isola et al. (2016); Zhang et al. (2016) showed conditional GAN can be used to generate photo-realistic images, from text descriptions or abstract maps. Under supervised settings, discriminator takes the input pairs of samples, where the discriminator can be viewed as an automatically learned loss function.

[1]Shanghai Jiao Tong University, Shanghai, China [2]Yitu Tech, Shanghai, China [3]University College London, London, United Kingdom. Correspondence to: Zhiming Zhou <heyohai@apex.sjtu.edu.cn>.

Another useful information to add is class labels (Springenberg, 2015; Salimans et al., 2016). CatGAN (Springenberg, 2015) built the discriminator that learns to provide high entropy class probability for each generated sample while the generator learns to make the discriminator predict low class entropy for each generated sample. Furthermore, Salimans et al. (2016) found that introducing class labels can significantly improve sample quality. However, the reason behind is still not well understood (Goodfellow, 2016).

In this paper, theoretically, we show how class labels help GAN training from the perspective of class-aware gradient. Our derivations indicate that the gradient of GAN with class labels tends to refine each sample towards a certain class. We find the previous formulation of using class labels (Salimans et al., 2016) still results in unreasonable gradient because of the overlay of the gradients from multiple classes. We next propose Activation Maximization Generative Adversarial Network (AM-GAN) that dynamically assigns each sample only one class during the training for a clearer guidance to optimize the generator. A theoretic analysis justifies that AM-GAN can be viewed as a weighted combination of the cross-entropy version of CatGAN (Springenberg, 2015) and the GAN model with class label proposed in (Salimans et al., 2016).

In our experiments on synthetic data, our model show faster convergence comparing with the GAN model with class label (Salimans et al., 2016). In the real-word data experiment, we observe sample quality improvement in terms of Inception score from 8.09 of (Salimans et al., 2016) to 8.34 of our model as well as convergence speedup without model collapse observed.

2. Preliminaries: GAN with Labeled Data

In the original GAN formulation (Goodfellow et al., 2014), the loss functions of the generator G and discriminator D are given as

$$\begin{aligned} L_D^{\text{ori}} &= -\mathbb{E}_{x\sim p_{\text{data}}}\big[\log D_r(x)\big] - \mathbb{E}_{z\sim p_z(z)}\big[\log(1-D_r(G(z)))\big] \\ &= -\mathbb{E}_{x\sim p_{\text{data}}}\big[\log D_r(x)\big] - \mathbb{E}_{x\sim G}\big[\log(1-D_r(x))\big], \\ L_G^{\text{ori}} &= -\mathbb{E}_{z\sim p_z(z)}\big[\log D_r(G(z))\big] = -\mathbb{E}_{x\sim G}\big[\log D_r(x)\big]. \end{aligned} \tag{1}$$

where the discriminator D performs two-class classification of real and generated data, and $D_r(x)$ represents the probability of sample x coming from real data.

The framework has been generalized to multi-class cases where each sample x has its corresponding class label $y\in\{1,\ldots,K,K+1\}$ with the $(K+1)$-th label corresponding to generated samples (Salimans et al., 2016).

In this case, the discriminator D, given an input sample x, outputs a $(K+1)$-dimensional vector of logits $l(x)=[l_1(x),\ldots,l_{K+1}(x)]$, which can be further translated into class probability distribution by applying the softmax function σ: $D(x) \triangleq \sigma(l(x)) = [\sigma_1(l(x)),\ldots,\sigma_{K+1}(l(x))]$ with $\sigma_i(l(x))=\frac{\exp(l_i(x))}{\sum_{k=1}^{K+1}\exp(l_k(x))}$.

Given the class label y, the target class probability distribution for D, is denoted as $v(y) = [v_1(y),\ldots,v_{K+1}(y)]$ where $v_i(y)=0$ if $i\neq y$ and $v_i(y)=1$ if $i=y$.

With the above, the loss functions can be written in the form of cross-entropy:

$$\begin{aligned} L_D^{\text{lab}} &= -\mathbb{E}_{(x,y)\sim p_{\text{data}}}\big[\log D_y(x)\big] - \mathbb{E}_{x\sim G}\big[\log D_{K+1}(x)\big] \\ &= \mathbb{E}_{(x,y)\sim p_{\text{data}}}\big[H\big(v(y), D(x)\big)\big] \\ &\quad + \mathbb{E}_{x\sim G}\big[H\big(v(K{+}1), D(x)\big)\big], \end{aligned} \tag{2}$$

$$\begin{aligned} L_G^{\text{lab}} &= -\mathbb{E}_{x\sim G}\big[\log \textstyle\sum_{i=1}^K D_i(x)\big] \triangleq -\mathbb{E}_{x\sim G}\big[\log D_r(x)\big] \\ &= \mathbb{E}_{x\sim G}\big[H\big([1,0],[D_r(x), D_{K+1}(x)]\big)\big], \end{aligned} \tag{3}$$

with $l_r(x) \triangleq \log\sum_{i=1}^K \exp(l_i(x))$ which can be viewed as the overall real logit assembled from the K real class logits, and $D_r(x) \triangleq \sum_{i=1}^K D_i(x) = \sigma(l_r(x))$ is the overall probability of real. H is the cross-entropy, defined as $H(p,q)=-\sum_i p_i \log q_i$. We refer to the above formulation as LabGAN (using class labels) throughout this paper.

It is worth mentioning that in the above formulation we adopt $-\log(D_r(x))$ as an alternative of $\log(1-D_r(x))$ for the generator's loss (Goodfellow et al., 2014). The rational is that the original loss function of the generator $\log(1-D_r(x))$ may suffer from the gradient vanishing problem (Goodfellow et al., 2014; Arjovsky & Bottou, 2017). The motivation of using the negative logarithm $-\log(D_r(x))$ is that: while giving a difference gradient scale, it always preserves the same gradient direction as $\log(1-D_r(x))$.

The recent work (Arjovsky & Bottou, 2017), however, suggests that a potential conflict may happen when using the negative logarithm as the loss function. We did not find empirical evidence from our experiment, and a related discussion is included in Appendix D. Further study on the subject is beyond the scope of this paper and we shall leave it for future work.

3. Activation Maximization GAN

This section is organized as follows. In Section 3.1, we shall first conduct a theoretical analysis on the existing label informed GAN (Salimans et al., 2016) (as reformulated in Eqs. (2) and (3)) and how class information indeed helps the GAN learning in that case. We use it to motivate our work and in Section 3.2 we present our activation maximization model. In Section 3.3, we analysis the new gradient behaviors of proposed methods. Furthermore, we compare our formulation and discuss the links to related work in Section 3.4. Later in Section 3.5, we further extend AM-GAN to a soft and more general version. Finally, we propose a new metric called AM score for evaluating the generated samples and compare it with the Inception score and the MODE score in Section 3.6.

3.1. Class-aware Gradients

Using class labels during the training is found to improve the quality of generated images empirically (Salimans et al., 2016). To our best knowledge, the underlying mechanism of how class labels help is not fully understood and with only several guesses (Goodfellow, 2016).

In this paper, we reformulate it using cross-entropy and our derivation of its gradient sheds some light on why class labels help improve the quality of generated images. Before going into the details, we first introduce the following lemma.

Lemma 1. *With l being the logits vector and σ being the softmax function as defined in Section 2, let $\sigma(l)$ be the current softmax probability distribution and $\hat{p}$ denote any target probability distribution, then*

$$-\frac{\partial H\big(\hat{p},\sigma(l)\big)}{\partial l} = \hat{p} - \sigma(l). \tag{4}$$

Proof.

$$-\left(\frac{\partial H\big(\hat{p},\sigma(l)\big)}{\partial l}\right)_k = -\frac{\partial H\big(\hat{p},\sigma(l)\big)}{\partial l_k} = \frac{\partial \sum_i \hat{p}_i \log \sigma(l)_i}{\partial l_k}$$

$$= \frac{\partial \sum_i \hat{p}_i \log \frac{\exp(l_i)}{\sum_j \exp(l_j)}}{\partial l_k} = \frac{\partial \sum_i \hat{p}_i \big(l_i - \log \sum_j \exp(l_j)\big)}{\partial l_k}$$

$$= \frac{\partial \sum_i \hat{p}_i l_i}{\partial l_k} - \frac{\partial \log\big(\sum_j \exp(l_j)\big)}{\partial l_k} = \hat{p}_k - \frac{\exp(l_k)}{\sum_j \exp(l_j)}$$

$$\Rightarrow -\frac{\partial H\big(\hat{p},\sigma(l)\big)}{\partial l} = \hat{p} - \sigma(l).$$

□

Empirically, the loss function L_G^{lab} is estimated by drawing samples from G. For a given sample x from G, the loss is $L_G^{\text{lab}}(x) = H([1,0],[D_r(x), D_{K+1}(x)])$, defined in Eq. (3). With Lemma 1 the gradient of $L_G^{\text{lab}}(x)$ w.r.t. the logits vector $l(x)$ is given as:

$$\begin{aligned} -\frac{\partial L_G^{\text{lab}}(x)}{\partial l_k(x)} &= -\frac{\partial H\big([1,0],[D_r(x), D_{K+1}(x)]\big)}{\partial l_k(x)} \\ &= -\frac{\partial H\big([1,0],\sigma\big([l_r(x), l_{K+1}(x)]\big)\big)}{\partial l_r(x)} \frac{\partial l_r(x)}{\partial l_k(x)} \\ &= \big(1 - D_r(x)\big)\frac{D_k(x)}{D_r(x)}, \quad k \in \{1,\dots,K\}, \end{aligned} \tag{5}$$

$$\begin{aligned} -\frac{\partial L_G^{\text{lab}}(x)}{\partial l_{K+1}(x)} &= -\frac{\partial H\big([1,0],\sigma\big([l_r(x), l_{K+1}(x)]\big)\big)}{\partial l_{K+1}(x)} \\ &= -D_{K+1}(x) = -\big(1 - D_r(x)\big). \end{aligned} \tag{6}$$

With the above, the gradient of $L_G^{\text{lab}}(x)$ w.r.t. x thus is:

$$-\frac{\partial L_G^{\text{lab}}(x)}{\partial x} = \sum_{k=1}^{K} -\frac{\partial L_G^{\text{lab}}(x)}{\partial l_k(x)}\frac{\partial l_k(x)}{\partial x} - \frac{\partial L_G^{\text{lab}}(x)}{\partial l_{K+1}(x)}\frac{\partial l_{K+1}(x)}{\partial x}$$

$$= \big(1 - D_r(x)\big)\left(\sum_{k=1}^{K}\frac{D_k(x)}{D_r(x)}\frac{\partial l_k(x)}{\partial x} - \frac{\partial l_{K+1}(x)}{\partial x}\right)$$

$$= \big(1 - D_r(x)\big)\sum_{k=1}^{K+1}\alpha_k^{\text{lab}}(x)\frac{\partial l_k(x)}{\partial x}, \tag{7}$$

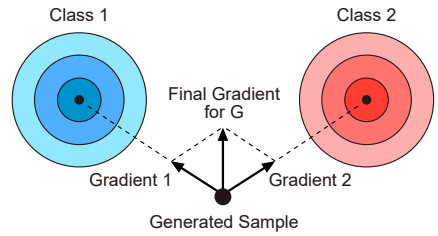


Figure 1. The problem of overlayed gradient of LabGAN (Salimans et al., 2016) from multi-mode real data. We assume the logit is built based on the distance between the gradient sample and the class center.

where

$$\alpha_k^{\text{lab}}(x) = \begin{cases} \frac{D_k(x)}{D_r(x)} & k \in \{1,\dots,K\} \\ -1 & k = K{+}1 \end{cases}. \tag{8}$$

From the formulation, we see that the overall gradient w.r.t generated example x is $(1{-}D_r(x))$. This is consistent with the original GAN (Goodfellow et al., 2014) when no label information is given. The gradient on real is then further distributed to each real class logit according to its current fraction of probability $\frac{D_k(x)}{D_r(x)}$.

As such, the gradient naturally takes the labels into consideration: for a sample from G, the higher probability on a certain class k leads to the larger step size towards the direction of increasing the corresponding logit for the class. As such, individually, the gradient from the sample tends to refine it towards being one of the classes in a probabilistic sense.

Recall there are also similar inspirations in related work. Denton et al. (2015) showed that the result could be significantly better, if GAN was trained with separated classes. Salimans et al. (2016) found the mini-batch features that capture similar mini-batch statistics between real and fake samples could help improve image quality. Feature matching (Salimans et al., 2016), adding an additional loss function (Larsen et al., 2015; Che et al., 2016) and instance noise (Arjovsky & Bottou, 2017) may also have effect on helping samples escape from the sticking state with little valid gradient.

3.2. The Proposed AM-GAN

However, the weighted average gradients across multiple label predictors does have a problem when the generated sample is ambiguous among classes, i.e. at the "fork of roads". The overlayed gradient from multiple classes may guide the sample to an unreasonable direction, as illustrated in Figure 1.

We thus hypothesize that in a multiple exclusive classes setting, a good sample from G should be classified to one class by D with a high confidence, i.e. with a sharp probability distribution over classes rather than a flat one or a

weighted one. And we should encourage G to generate such samples.

In other words, rather than pushing samples to be real, judged by each label predictors, we could actually assign a target class for each generated sample x and maximize its activation.

A natural choice of target class could be the class that is currently of the maximal probability estimated by D, i.e. $y_{\max}(x) = \operatorname{argmax}_{i\in\{1,\dots,K\}} D_i(x)$. Therefore, the target class probability distribution for G is $v(x) \triangleq v(y_{\max}(x))$. Mathematically, the loss of G becomes:

$$\begin{aligned} L_G^{\text{AM}} &= -\mathbb{E}_{x\sim G}\big[\log D_i(x)\big|y_{\max}(x)=i\big] \\ &= \mathbb{E}_{x\sim G}\big[H\big(v(x), D(x)\big)\big]. \end{aligned} \tag{9}$$

It can be viewed as we are training the generator to perform activation maximization (Nguyen et al., 2016a) on class logits of D, with dynamically and automatically determined target class for each generated sample. We name our model Activation Maximization Generative Adversarial Network (AM-GAN).

The $-log(D_r(x))$ alternative GAN can be viewed as: every sample is trying to activate the *real* class. With classes label provided, we can hence assign (dynamic) label for each sample to provide better guidance.

3.3. AM-GAN Gradient

Similar to Section 3.1, here we analyze the gradient of G's new loss function:

$$-\frac{\partial L_G^{\text{AM}}(x)}{\partial l_k(x)} = -\frac{\partial H\big(v(x), D(x)\big)}{\partial l_k(x)} = v_k(x) - D_k(x), \quad k\in\{1,\dots,K{+}1\} \tag{10}$$

$$\begin{aligned} -\frac{\partial L_G^{\text{AM}}(x)}{\partial x} &= \sum_{k=1}^{K+1} -\frac{\partial L_G^{\text{AM}}(x)}{\partial l_k(x)}\frac{\partial l_k(x)}{\partial x} \\ &= \sum_{k=1}^{K+1}\big(v_k(x)-D_k(x)\big)\frac{\partial l_k(x)}{\partial x} = \sum_{k=1}^{K+1}\alpha_k^{\text{AM}}(x)\frac{\partial l_k(x)}{\partial x}, \end{aligned} \tag{11}$$

where

$$\alpha_k^{\text{AM}} = \begin{cases} 1 - D_k(x) & \text{if } k = y_{\max}(x) \\ -D_k(x) & \text{otherwise (include } k = K{+}1) \end{cases}. \tag{12}$$

In AM-GAN, only the gradient w.r.t. the logit $l_{y_{\max}(x)}$ will be positive (encouraged), and the others will have negative gradient (discouraged). As a result, the sample will be refined towards the class that is currently of the largest probability, and at the same time, being far away from other classes.

3.4. Relations to the Existing Work

3.4.1. Compare with CatGAN and LabGAN

CatGAN trains a K-class discriminator with labeled data or unlabeled data. And for generated images, the discriminator tries to distribute its probability uniformly on each class by maximizing $\mathbb{E}_{x\sim G}[H(D(x))]$, while the generator tries to make the distribution of low entropy (i.e. sharp), minimizing $\mathbb{E}_{x\sim G}[H(D(x))]$.

We show the proposed AM-GAN has close relationship to CatGAN (Springenberg, 2015) and LabGAN (Salimans et al., 2016). Specifically, the AM-GAN is the cross-entropy version of CatGAN that combined with LabGAN by introducing an additional fake class.

To see this, we first describe a simple refactoring lemma for cross-entropy with $K{+}1$ logits.

Lemma 2. *Given $v = [v_1,\dots,v_{K+1}]$, $v_{1:K} \triangleq [v_1,\dots,v_K]$, $v_r \triangleq \sum_{k=1}^K v_k$, $R(v) \triangleq v_{1:K}/v_r$ and $F(v) \triangleq [v_r, v_{K+1}]$, let $\hat{p} = [\hat{p}_1,\dots,\hat{p}_{K+1}]$, $p = [p_1,\dots,p_{K+1}]$, then we have:*

$$H\big(\hat{p}, p\big) = \hat{p}_r H\big(R(\hat{p}), R(p)\big) + H\big(F(\hat{p}), F(p)\big). \tag{13}$$

Proof.

$$\begin{aligned} H(\hat{p},p) &= -\textstyle\sum_{k=1}^K \hat{p}_k \log p_k - \hat{p}_{K+1}\log p_{K+1} \\ &= -\hat{p}_r \textstyle\sum_{k=1}^K \frac{\hat{p}_k}{\hat{p}_r}\log(\frac{p_k}{p_r}p_r) - \hat{p}_{K+1}\log p_{K+1} \\ &= -\hat{p}_r \textstyle\sum_{k=1}^K \frac{\hat{p}_k}{\hat{p}_r}(\log\frac{p_k}{p_r} + \log p_r) - \hat{p}_{K+1}\log p_{K+1} \\ &= -\hat{p}_r \textstyle\sum_{k=1}^K \frac{\hat{p}_k}{\hat{p}_r}\log\frac{p_k}{p_r} - \hat{p}_r\log p_r - \hat{p}_{K+1}\log p_{K+1} \\ &= \hat{p}_r H\big(R(\hat{p}), R(p)\big) + H\big(F(\hat{p}), F(p)\big). \end{aligned}$$

□

With Lemma 2, we can decompose the loss function of G in AM-GAN into two terms.

$$\begin{aligned} L_G^{\text{AM}}(x) &= H\big(v(x), D(x)\big) \\ &= v_r(x)\cdot \underbrace{H\Big(R\big(v(x)\big), R\big(D(x)\big)\Big)}_{\text{cross-entropy CatGAN}} + \underbrace{H\Big(F\big(v(x)\big), F\big(D(x)\big)\Big)}_{\text{LabGAN}}. \end{aligned} \tag{14}$$

The first term can be viewed as the cross-entropy version of $L_G^{\text{cat}}(x)$: generator of CatGAN directly optimizes entropy $H(R(D(x)))$ to make each sample be one class, while AM-GAN achieves this by the first term of its decomposed loss $H(R(v(x)), R(D(x)))$ in terms of cross-entropy with given target distribution.

The second term actually equals to the loss function of G in LabGAN, i.e. $L_G^{\text{lab}}(x)$ in Eq. (3).

$$\begin{aligned} H\Big(F\big(v(x)\big), F\big(D(x)\big)\Big) &= H\Big(\big[1,0\big], \big[D_r(x), D_{K+1}(x)\big]\Big) \\ &= L_G^{\text{lab}}(x). \end{aligned} \tag{15}$$

Similar analysis applies to the loss function of D in AM-GAN, $L_D^{\text{AM}}(x)$, and it is included in Appendix A. We also extended AM-GAN to unlabeled data and introduced loss for further avoiding mode collapse, analogy to CatGAN, in Appendix B and Appendix C, where similar analysis also included.

3.4.2. ACTIVATION MAXIMIZATION

Our work is also closely related to activation maximization that is originally used to visualize a neuron (Nguyen et al., 2016a). In our case, we use it to ensure the quality of generated samples by pushing them to be one of the classes. In their formulation, the target neuron corresponds to the target class in our case, which is automatically determined for each sample.

The maximized activation of one neuron is not necessary of high quality. Traditionally people introduce various priors. In AM-GAN, with the exist of fake class, the adversarial process of GAN training ensured the sample quality.

3.5. AM-GAN with Softmax Smoothing

A potential concern is that directly assigning the target to the class, which currently has the maximum probability judged by D, may result in an unstable objective. To handle this issue, we further propose a soft version of AM-GAN which takes a soft selection among K classes by softmax function with temperature t.

The new target probability distribution for G is $v(x,t) \triangleq \hat{\sigma}(l(x),t)$, where $\hat{\sigma}_i(l(x),t) = \frac{\exp(l_i(x)/t)}{\sum_{k=1}^{K}\exp(l_k(x)/t)}$ for $i \in \{1,\cdots,K\}$ and $\hat{\sigma}_{K+1}(l(x),t) = 0$. We call this soft version of AM-GAN as SAM-GAN. When $t{=}0^+$, SAM-GAN is approaching AM-GAN. Similar analysis can be applied to SAM-GAN as is done in previous sections.

Specially, SAM-GAN can be viewed as a coarse-to-fine class selection scheme for generator training: when sample has evenly distributed class probability or has several large components with similar value scale, the gradients may be provided from multiple classes; when the generated sample approaches a certain class, the gradient offered from this class will dominate others, similar to the gradient in AM-GAN.

3.6. AM Score and Reference Distribution

One of the difficult problems in generative models is how to evaluate them (Theis et al., 2015). The "being one class" principle in fact was used for that purpose. In this section, we first present two related existing metrics, i.e., the Inception score (Salimans et al., 2016) and the MODE score (Che et al., 2016), and point out their drawbacks when the training (reference) data class is not evenly distributed. Then we present the new AM score which solves such a problem, and we further suggest using a accordingly pretrained classifier for each dataset.

3.6.1. INCEPTION SCORE

As a recently proposed metric for evaluating the performance of a generator, the Inception score is found to be well correlated with human evaluation (Salimans et al., 2016), where a pretrained publicly-available Inception model C is introduced. By applying the Inception model to each generated image x and getting the corresponding class probability distribution judged by C, i.e. $C(x)$, the Inception score is calculated via:

$$\text{Inception score} = \exp\Big(\mathbb{E}_x\big[\text{KL}\big(C(x) \parallel \bar{C}^G\big)\big]\Big),$$

where $\mathbb{E}_x$ is the short of $\mathbb{E}_{x\sim G}$ and $\bar{C}^G = \mathbb{E}_x[C(x)]$ is the overall class probability distribution of the generated samples judged by C, and KL denotes the Kullback-Leibler divergence and is defined as:

$$\begin{aligned}\text{KL}(p \parallel q) &= \textstyle\sum_i p_i \log\frac{p_i}{q_i} = \sum_i p_i \log p_i - \sum_i p_i \log q_i \\ &= -H(p) + H(p,q).\end{aligned}$$

A particular drawback of the Inception score is it does not take into account the prior distribution of the labels. An extended measure, the MODE score, is proposed in (Che et al., 2016), which is calculated via:

$$\text{MODE score} = \exp\Big(\mathbb{E}_x\big[\text{KL}\big(C(x) \parallel \bar{C}^{\text{train}}\big)\big] - \text{KL}(\bar{C}^G \parallel \bar{C}^{\text{train}})\Big),$$

where the overall class probability distribution $\bar{C}^{\text{train}}$ from the training data has been added as a reference point. However, the MODE score and the Inception score are, in fact, equivalent. To see it, we introduce the following lemma.

Lemma 3. *Let $p(x)$ be the class probability distribution of the sample x that from a certain data distribution, and $\bar{p}$ denote the reference probability distribution, then*

$$\mathbb{E}_x\big[H\big(p(x),\bar{p}\big)\big] = H\big(\mathbb{E}_x\big[p(x)\big],\bar{p}\big). \tag{16}$$

Proof.

$$\begin{aligned}&\mathbb{E}_x\big[H\big(p(x),\bar{p}\big)\big] = \mathbb{E}_x\big[-\textstyle\sum_i p_i(x)\log\bar{p}_i\big] \\ &= -\textstyle\sum_i \mathbb{E}_x[p_i(x)]\log\bar{p}_i = -\sum_i \big(\mathbb{E}_x[p(x)]\big)_i \log\bar{p}_i \\ &= H\big(\mathbb{E}_x\big[p(x)\big],\bar{p}\big).\end{aligned}$$

□

With Lemma 3, we have

$$\begin{aligned}\log(\text{Inception score}) &= \mathbb{E}_x\big[\text{KL}(C(x) \parallel \bar{C}^G)\big] \\ &= \mathbb{E}_x\big[H\big(C(x),\bar{C}^G\big)\big] - \mathbb{E}_x\big[H\big(C(x)\big)\big] \\ &= H\big(\mathbb{E}_x\big[C(x)\big],\bar{C}^G\big) - \mathbb{E}_x\big[H\big(C(x)\big)\big] \\ &= H(\bar{C}^G) + (-\mathbb{E}_x\big[H\big(C(x)\big)\big]),\end{aligned}$$

$$\begin{aligned}\log(\text{MODE score}) &= \mathbb{E}_x\big[\text{KL}\big(C(x) \parallel \bar{C}^{\text{train}}\big)\big] - \text{KL}(\bar{C}^G \parallel \bar{C}^{\text{train}}) \\ &= \mathbb{E}_x\big[H\big(C(x),\bar{C}^{\text{train}}\big)\big] - \mathbb{E}_x\big[H\big(C(x)\big)\big] \\ &\quad - H(\bar{C}^G,\bar{C}^{\text{train}}) + H(\bar{C}^G) \\ &= H(\bar{C}^G) + (-\mathbb{E}_x\big[H\big(C(x)\big)\big]),\end{aligned}$$

$$\Rightarrow \text{Inception score} = \text{MODE score}, \tag{17}$$

where we see that the required $\bar{C}^{\text{train}}$ is canceled out. Thus, they both consist of two entropy terms: the first term encourages the overall class probability distribution formed

by generated samples to be uniformly distributed (large entropy), and the second one encourages the class probability distribution of each generated sample to be sharp (low entropy).

3.6.2. AM Score

The KL divergence is non-symmetric, and being the reference distribution, $\bar{C}^{\text{train}}$ actually should placed at the first place. We here propose to swap $\bar{C}^{\text{train}}$ with its counterpart in the two KL divergence terms Eq. (17), which leads to a more sensible metric:

$$\begin{aligned}
&\mathbb{E}_x\big[\text{KL}\big(\bar{C}^{\text{train}} \parallel C(x)\big)\big] - \text{KL}(\bar{C}^{\text{train}} \parallel \bar{C}^G) \qquad (18)\\
&= \mathbb{E}_x\big[H\big(\bar{C}^{\text{train}}, C(x)\big)\big] - H(\bar{C}^{\text{train}}) - H(\bar{C}^{\text{train}}, \bar{C}^G) + H(\bar{C}^{\text{train}})\\
&= \mathbb{E}_x\big[H\big(\bar{C}^{\text{train}}, C(x)\big)\big] + (-H(\bar{C}^{\text{train}}, \bar{C}^G)) \triangleq \text{AM score.}
\end{aligned}$$

The above defined AM score is in form of two cross-entropy terms: the first is maximized when each sample is being far away from the training data overall class distribution; the second part is maximized when the generated samples' average distribution is the same as training data. The overall class distribution indicated by the training data, i.e. $\bar{C}^{\text{train}}$, has thus been taken into account. When training data is not evenly distributed, it will be important.

3.6.3. Pretrained Classifier

It was showed the Inception score with C being the Inception model trained with ImageNet, can well correlated with human evaluation on CIFAR10. We found CIFAR10 is not evenly distributed over the ImageNet Inception model, where the entropy term on average distribution of the Inception score does not work well. With a pretrained CIFAR10 classifier, the AM score can well capture the statistics of average distribution. We hence argue that for general data, the C should be a accordingly pretrained classifier on given dataset. Note that the Inception score and the MODE score adopt an exponential transformation based on the above calculated scores in Eq. (17). With a pretrained classifier on the given dataset, we will, however, show in the experiment that without the exponential transformation, AM score is informative enough.

4. Experiments

To empirically justify our proposed model AM-GAN, we conduct experiments [1] on a synthetic dataset and two well-known labeled image datasets: CIFAR-10 and MNIST. Additionally, we investigate the behaviors of the proposed metric AM score in our experiments.

4.1. Implementation Details

On the synthetic dataset, we use the multi-layer fully-connected structure. And on CIFAR-10 and MNIST, we use the DCGAN architecture (Radford et al., 2015) with stride deconvolution.

Specifically, the generator consists of 4 deconvolution layers with stride $= 2$ and kernel size $= 5 \times 5$ while the discriminator consists of 5 convolution layers with stride $= 2$ and kernel size $= 3 \times 3$. We extend the MNIST images to 32×32 resolution during training via zero padding for sharing similar architecture as CIFAR-10.

In the generator, noise is introduced at each deconvolution or linear layer (Goodfellow, 2016). And in the discriminator, dropout is introduced between every two layers. Also we add a small constant additive Gaussian noise to each sample before feeding into the discriminator.

We use Adam optimizer with $\beta_1 = 0.5$ and exponentially decayed learning rate. The initial learning rate is 10^{-3} for CIFAR-10 and MNIST, and 10^{-5} for the synthetic dataset. It decays to a fraction of 0.1 every 200,000 iterations.

Training on CIFAR-10 usually gets quite good results within 200,000 iterations, the reported results are attained after 300,000 iterations, which takes around 20 hours with a single GeForce GTX 1080.

After tuning in the experiments, we find $\frac{1}{1.5}$ and $\frac{1}{2.0}$ are usually good choices as the temperature t for SAM-GAN. Specifically, we use $t{=}\frac{1}{2.0}$ on CIFAR-10 and MNIST, and $t{=}\frac{1}{1.5}$ on the synthetic dataset.

4.2. Synthetic: Mixture of Gaussian Dataset

To simulate multi-class labeled samples, we follow (Metz et al., 2016) and incorporate a mixture of 8 2D Gaussian with stddev 0.01 (evenly distributed along a unit circle) as our synthetic data.

As the true data model, a.k.a. the oracle, is known, we can form a direct and accurate evaluation by computing negative log-likelihood (NLL) of the true model (parameters) fitted with the generated examples (*NLL by Oracle*) (Yu et al., 2016). We plot the generated samples for the corresponding training iterations along with the density of the true model in Figure 2. It shows that generated samples in (S)AM-GAN can quickly and evenly separate to each mode nicely in the early stage of the training, and finally converge well.

We also test LabGAN with this dataset, and find its training process is less stable, and occasionally missing mode during training process. We further measure the fitness by NNL and compare it with LabGAN for different training iterations in Figure 3, where our proposed (S)AM-GANs converge much faster and provide better NLL than LabGAN.

4.3. Image Generation Results

We further evaluate our proposed models on real-world datasets. CIFAR-10 is a small dataset comprising 50,000

[1] Link for experiment code: `https://goo.gl/bJ8kHI`.

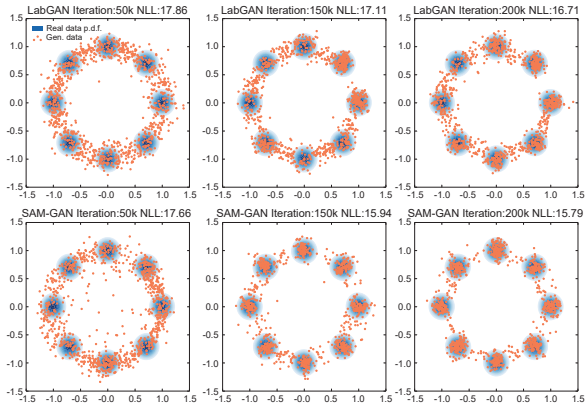


Figure 2. The generated examples along with the true density distribution on synthetic data.

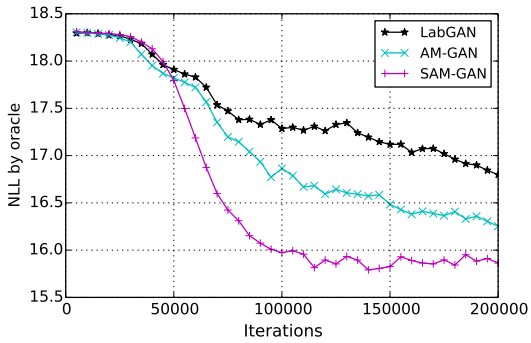
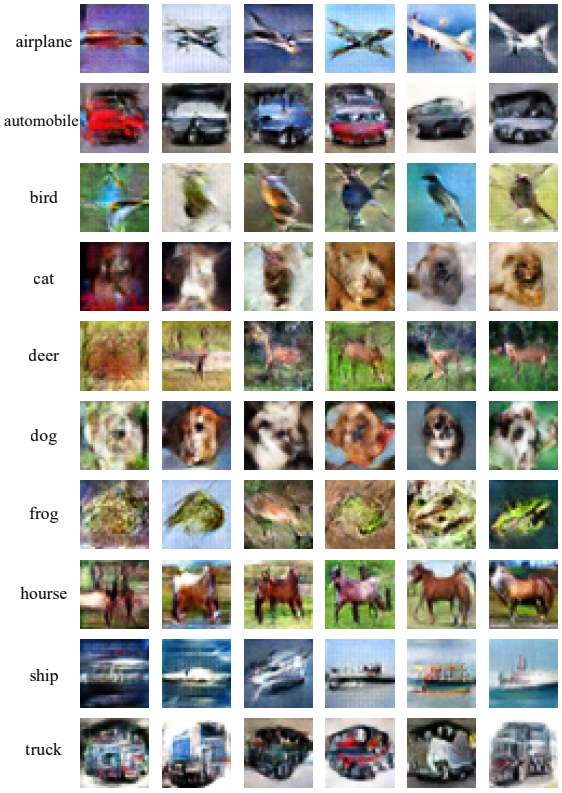


Figure 3. Training iterations on the synthetic data measured with *NNL by Oracle*.



Iterations	5,000	10,000	15,000	30,000	150,000	300,000
Inception score	5.79	6.90	7.74	8.01	8.17	8.34
AM score	8.31	8.55	8.74	8.84	9.20	9.29

Figure 4. CIFAR-10 progress results.

training color images of size 32×32 in 10 classes. While the limited training data and low quality of some samples make it not easy to generate high quality samples on CIFAR-10 for generative models.

Figure 4 shows the generated images of (S)AM-GAN against training iterations on CIFAR-10, where our proposed model can quickly achieve fairly good sample quality for most of classes, including airplane, automobile, bird, horse, ship and truck. Note that we use horizontal clipping for data augmentation here. Both the Inception score and our new metric AM score are reported.

Additionally we test our model on MNIST. The generated images along with real images are shown in Figure 5. As we can see the generated images are highly comparable with the real ones.

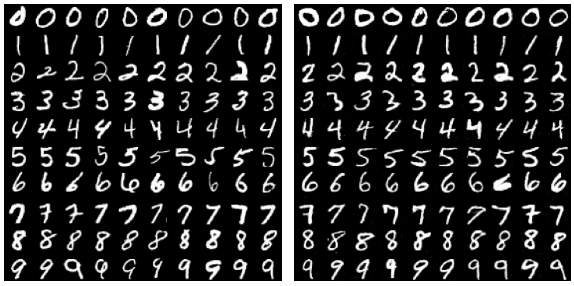
To further quantify the generation performance, we next report the generators' performance of (S)AM-GANs and several compared models based on CIFAR-10. We use the Inception score (Salimans et al., 2016), the widely accepted metric to benchmark them, while leaving the evaluation of the our proposed new metric in Section 4.4.

The overall performance on CIFAR-10 is provided in Table 1 where the following models are compared: (i) the GAN with denoising feature mapping (DFM) (Warde-Farley & Bengio, 2017), which offers the best performance on CIFAR-10 among unsupervised GANs so far; (ii) deep convolutional GAN (DCGAN) trained with labels (Wang & Liu, 2016); (iii) SteinGAN with Stein variational gradient training in GAN (Wang & Liu, 2016); (iv) LabGAN*, the full version of LabGAN proposed in (Salimans et al., 2016) enhanced with several techniques; (v) our proposed AM-GAN and its soft version. Note that the pretrained Inception model C is the same across the baselines.

Table 1. Overall Inception score performance on CIFAR-10.

Model	Score $\pm$ Std.
DFM (unsupervised) (Warde-Farley & Bengio, 2017)	7.72 ± 0.13
DCGAN with labels (Wang & Liu, 2016)	6.58
SteinGAN with labels (Wang & Liu, 2016)	6.35
LabGAN* (Salimans et al., 2016)	8.09 ± 0.07
AM-GAN (our work)	8.12 ± 0.07
SAM-GAN (our work)	8.34 ± 0.05
Real data	11.24 ± 0.12

The results in Table 1 demonstrate that our AM-GAN, particularly, its soft version, achieves the highest Inception score, which verifies our hypothesis that concentrating the gradients from multiple classes on one class or a few classes in a coarse-to-fine fashion would help the generator avoid the gradient overlay problem and improve the quality of generated samples that clearly belong to one class.



(a) Real Images (b) Generated Images

Figure 5. MNIST results.

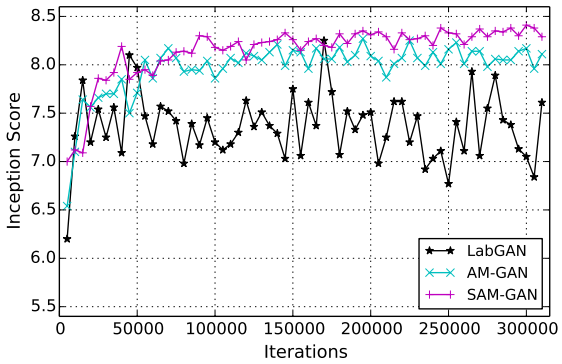


Figure 6. Training on CIFAR-10: LabGAN and (S)AM-GAN.

Figure 6 plots the Inception scores of LabGAN and (S)AM-GANs against training iterations on CIFAR-10. LabGAN is relatively unstable during training and we find sometimes it fails to converge. By contrast, we find that both AM-GAN and its soft version is more stable, and finally achieve higher Inception scores than LabGAN. Furthermore, SAM-GAN outperforms AM-GAN on both convergence rate and Inception score performance, which indicates that the coarse-to-fine class selection scheme works well in SAM-GAN.

4.4. Evaluating AM Score

In Figure 4, we have observed that the Inception score and the AM score are fairly consistent with each other when evaluating generative models. To further understand their differences, we compare them against the oracle NLL (which is considered as the accurate metric when the true data model is known) in the synthetic data. In Figure 8, we see that the AM score has a high consistency with the oracle (negative) NLL, while the Inception score is not always, particularly when the score is low.

For the toy data, we use the same pretrained classifier as C for the Inception score and the AM score. For CIFAR10, a pretrained (ImageNet) Inception model is usually used.

We show CIFAR10 is not evenly distributed across classes under the Inception model, in figure 7.A. We further found that, with the Inception model, the entropy terms of the Inception score (Eq. 17) on overall distribution can't work well: as the training goes iteratively, $H(\bar{C}^G)$ keeps oscillating as illustrated in Figure 7.B.

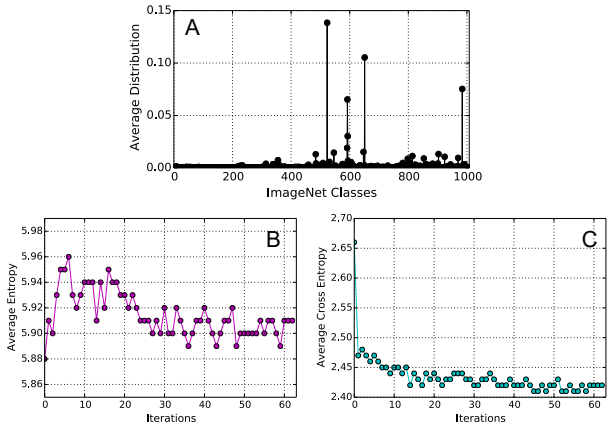


Figure 7. A: CIFAR-10 training data's overall distribution on ImageNet under the Inception model. B: The entropy term $H(\bar{D}_c^G)$ of the Inception score on overall distribution. C: The cross-entropy term $H(\bar{D}_c^{\text{train}}, \bar{D}_c^G)$ of the AM score on overall distribution.

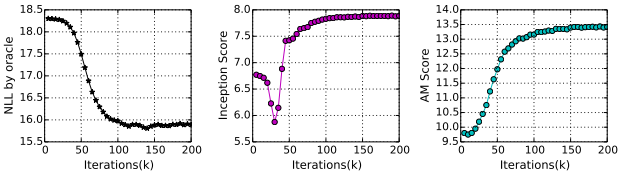


Figure 8. Metric Evaluation. Pearson correlation with negative Orale NLL: the Inception score 0.926; the AM score 0.977.

With a pretrained classifier on CIFGAR10, the AM score (Eq. 18) well captured the statistics on generated samples' overall distribution: $H(\bar{C}^{\text{train}}, \bar{C}^G)$ is stably decreasing, shown in Figure 7.C.

5. Conclusions

In this paper, we analyzed how information about class labels helps GAN training and pointed out the problem of overlayed gradients from multiple classes. To address such a problem we proposed AM-GAN which leverages the discriminator to dynamically assign the class label with the highest logit value for each sample and accordingly maximize each sample's activation on the class logit, which provides a clear gradient guidance to generator. The experiments on both synthetic data and real-world CIFAR-10 and MNIST data demonstrated the effectiveness of the proposed AM-GAN and its extensions on higher quality sample generation and faster convergence. Additionally, we proposed AM score for evaluating generative models.

For the future work, an investigation on how the gradient from non-chosen classes affects the optimization could be of interest. Combining AM-GAN with Wasserstein distance (Arjovsky et al., 2017) could also be a promising direction.

References

Arjovsky, Martin and Bottou, Léon. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, volume 2016, 2017.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Brock, Andrew, Lim, Theodore, Ritchie, JM, and Weston, Nick. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

Cao, Yun, Zhou, Zhiming, Zhang, Weinan, and Yu, Yong. Unsupervised diverse colorization via generative adversarial networks. *arXiv preprint*, 2017.

Che, Tong, Li, Yanran, Jacob, Athul Paul, Bengio, Yoshua, and Li, Wenjie. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

Denton, Emily L, Chintala, Soumith, Fergus, Rob, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pp. 1486–1494, 2015.

Goodfellow, Ian. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Huang, Xun, Li, Yixuan, Poursaeed, Omid, Hopcroft, John, and Belongie, Serge. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, Larochelle, Hugo, and Winther, Ole. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Ledig, Christian, Theis, Lucas, Huszár, Ferenc, Caballero, Jose, Cunningham, Andrew, Acosta, Alejandro, Aitken, Andrew, Tejani, Alykhan, Totz, Johannes, Wang, Zehan, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

Metz, Luke, Poole, Ben, Pfau, David, and Sohl-Dickstein, Jascha. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Nguyen, Anh, Dosovitskiy, Alexey, Yosinski, Jason, Brox, Thomas, and Clune, Jeff. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pp. 3387–3395, 2016a.

Nguyen, Anh, Yosinski, Jason, Bengio, Yoshua, Dosovitskiy, Alexey, and Clune, Jeff. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016b.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2226–2234, 2016.

Springenberg, Jost Tobias. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Theis, Lucas, Oord, Aäron van den, and Bethge, Matthias. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Wang, Dilin and Liu, Qiang. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

Warde-Farley, D. and Bengio, Y. Improving generative adversarial networks with denoising feature matching. In *ICLR*, 2017.

Yu, Lantao, Zhang, Weinan, Wang, Jun, and Yu, Yong. Seqgan: sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*, 2016.

Zhang, Han, Xu, Tao, Li, Hongsheng, Zhang, Shaoting, Huang, Xiaolei, Wang, Xiaogang, and Metaxas, Dimitris. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*, 2016.

Zhu, Jun-Yan, Krähenbühl, Philipp, Shechtman, Eli, and Efros, Alexei A. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pp. 597–613. Springer, 2016.

A. AM-GAN and CatGAN: Discriminator

A.1. AM-GAN Loss: L_D^{AM}

The loss of the discriminator in AM-GAN is

$$\begin{aligned} L_D^{\text{AM}} &= -\mathbb{E}_{(x,y)\sim p_{\text{data}}}[\log D_y(x)] - \mathbb{E}_{x\sim G}[\log D_{K+1}(x)] \\ &= \mathbb{E}_{(x,y)\sim p_{\text{data}}}\Big[H\Big(v(y), D(x)\Big)\Big] \\ &\quad + \mathbb{E}_{x\sim G}\Big[H\Big(v(K{+}1), D(x)\Big)\Big]. \end{aligned}$$

With Lemma 2, we can decompose the loss as:

$$\begin{aligned} L_D^{\text{AM}} = \; & \mathbb{E}_{(x,y)\sim p_{\text{data}}}\Big[H\big(R(v(y)), R(D(x))\big)\Big] \times v_r(r) \\ & + \mathbb{E}_{(x,y)\sim p_{\text{data}}}\Big[H\big(F(v(y)), F(D(x))\big)\Big] \\ & + \mathbb{E}_{x\sim G}\Big[H\big(R(v(K{+}1)), R(D(x))\big)\Big] \times v_r(K{+}1) \\ & + \mathbb{E}_{x\sim G}\Big[H\big(F(v(K{+}1)), F(D(x))\big)\Big]. \end{aligned}$$

Note that $\forall\, y \in \{1, \ldots, K\},\ v_r(y)$ is all equal, and we define it as $v_r(r)$.

A.2. Discriminator loss on fake sample

The L_D part connections between CatGAN and AM-GAN, in terms of decomposed losses, show as follows.

The discriminator of CatGAN maximizes the prediction entropy of each fake sample to judge the sample as being *not a class*:

$$L_D^{\text{Cat'}} = \mathbb{E}_{x\sim G}\Big[-H\big(D(x)\big)\Big].$$

In AM-GAN, as we have an extra class on fake, we can achieve this in a simpler manner by minimizing the probability on real logits.

$$L_D^{\text{AM'}} = \mathbb{E}_{x\sim G}\Big[H\big(F(v(K{+}1)), F(D(x))\big)\Big].$$

If $v_r(K{+}1)$ is not zeros, that is, when we did (negative) label smoothing (Salimans et al., 2016), we could define $R(v(K{+}1))$ to be a uniform distribution.

$$L_D^{\text{AM''}} = \mathbb{E}_{x\sim G}\Big[H\big(R(v(K{+}1)), R(D(x))\big)\Big] \times v_r(K{+}1).$$

As a result, the label smoothing part probability will be required to be uniformly distributed, similar to CatGAN. See Appendix D for discussion on label smoothing.

A.3. Discriminator loss on real sample

The loss on real sample of CatGAN is the $R(\cdot)$ part loss of AM-GAN on real samples:

$$L_D^{\text{AM'''}} = \mathbb{E}_{(x,y)\sim p_{\text{data}}}\Big[H\big(R(v(y)), R(D(x))\big)\Big] \times v_r(y),$$

despite that, in AM-GAN, it is weighed by $v_r(y)$.

B. Unlabeled Data

The previous discussion builds on the assumption that we have enough labeled data. In this section, we extend it to unlabeled data. Our solution is analogous to CatGAN (Springenberg, 2015).

B.1. Semi-supervised setting

Under semi-supervised setting, we can add the following loss to the original solution to integrate the unlabeled data (with the distribution denoted as $p_{\text{unl}}(x)$):

$$L_D^{\text{unl'}} = \mathbb{E}_{x\sim p_{\text{unl}}}\Big[H\Big(v(x), D(x)\Big)\Big].$$

B.2. Unsupervised setting

Under unsupervised setting, we need to introduce one extra loss, analogy to categorical GAN (Springenberg, 2015):

$$L_D^{\text{unl''}} = H\Big(p_{\text{ref}}, R(\mathbb{E}_{x\sim p_{\text{unl}}}[D(x)])\Big),$$

where the p_{ref} is a reference label distribution for the prediction on unsupervised data. For example, p_{ref} could be set as a *uniform* distribution, which requires the unlabeled data to make use of all the candidate class logits.

This loss can be optionally added to semi-supervised setting, where the p_{ref} could be defined as the predicted label distribution on the labeled training data $\mathbb{E}_{x\sim p_{\text{data}}}[D(x)]$.

B.3. Refactoring of unlabeled loss

The $L_D^{\text{unl'}}$ loss can also be re-factored, Lemma 2.

$$\begin{aligned} L_D^{\text{unl'}} = \; & \mathbb{E}_{x\sim p_{\text{unl}}}\Big[H\Big(R\big(v(x)\big), R\big(D(x)\big)\Big)\Big] \times v_r(x) \\ & + \mathbb{E}_{x\sim p_{\text{unl}}}\Big[H\Big(F\big(v(x)\big), F\big(D(x)\big)\Big)\Big]. \end{aligned}$$

From the equation we can easily see that it is decomposed as a LabGAN loss and a weighted cross-entropy loss to make the prediction *being one class*.

C. Mode Collapse

AM-GAN, as a multi-class extension of GAN (Goodfellow et al., 2014), has similar theoretical guarantee on convergence. Mode collapse is not observed in our experiments. As supplementary material, here we also include some discussions on mode collapse.

C.1. Class level mode collapse

Starting with CatGAN solution: the generator of CatGAN maximizes the entropy of overall distributions of generated samples to make them evenly distributed over the classes.

$$L_G^{\text{Cat''}} = H\big(\mathbb{E}_{x\sim G}[D(x)]\big).$$

Requiring generated samples evenly distributed among classes is useful to avoid mode collapse.

The global optimal of the additional loss should be consistent with the original one. Instead of using entropy, we use cross-entropy with reference distribution: minimizing the cross-entropy between average distribution of generated samples and the training samples.

$$L_G^{\text{mode}} = H\Big(R\big(\mathbb{E}_{x\sim p_{\text{data}}}[D(x)]\big), R\big(\mathbb{E}_{x\sim G}[D(x)]\big)\Big).$$

If class level mode collapse appears, this loss will play a important role to make it recover from the bad state.

$$-\frac{\partial L_G^{\text{mode}}}{\partial l_k} = R\big(\mathbb{E}_{x\sim p_{\text{data}}}[D(x)]\big)_k - R\big(\mathbb{E}_{x\sim G}[D(x)]\big)_k.$$

From the above gradient equation: if class *A* is missing, every sample will be encouraged to be refined towards *being class A*; if samples are collapsed at class *B*, every sample will be discouraged from being class *B*.

C.2. Intra-class mode collapse

The above loss can't directly avoid intra-class scale mode missing. The mostly observed or most obvious pattern of mode collapse is class level mode collapse.

Ideally, avoiding class level mode collapse may also help avoid intra-class mode collapse, and one way to reduce the risk of intra-class mode collapse is: use the unsupervised setting (Appendix sec B.2) with a proper number of class logits. Extending classes from labeled, or gradually increasing the number of classes is also a possible solution.

One practical problem of this loss is that: $E_{x\sim G}[D(x)]$ is usually approximated according to samples in one batch. When the number of classes is large, the approximation could be very inaccurate.

C.3. Data with unclear modes

We assume exclusive classes setting for AM-GAN, where we can require each sample to *be one class*. AM-GAN may not work well when data does not have clear modes. With soft assigned class labels, SAM-GAN has potential to work better in this situation.

D. Label Smoothing and $-\log(D_r(x))$

D.1. Label smoothing

Label smoothing that avoiding extreme logits value was showed to be a good regularization (Szegedy et al., 2016). Note that, label smoothing can be introduced separately for $R(\cdot)$ and $F(\cdot)$. Here we will only discuss the $F(\cdot)$ part.

A general version of label smoothing could be: (modifying the target probability of discriminator)

$$\big[\hat{D}_r^D(x), \hat{D}_{K+1}^D(x)\big] = \begin{cases} [\lambda_1, 1-\lambda_1] & x\sim G \\ [1-\lambda_2, \lambda_2] & x \sim p_{\text{data}} \end{cases}.$$

(Salimans et al., 2016) proposed to use only one-side label smoothing. That is, to only apply label smoothing for real samples: $\lambda_1 = 0$ and $\lambda_2 > 0$. The reasoning of one-side label smoothing is: applying label smoothing on fake samples will lead to fake mode on data distribution.

The authors currently have not fully understood how the reasoning effect optimization in the view of gradient. But gradient analysis show that: it is problematic to apply label smoothing to fake samples together with $\log(1-D_r(x))$ generator loss; the problem does not exist with $-\log(D_r(x))$ generator loss.

D.2. The $\log(1-D_r(x))$ generator loss

The $\log(1-D_r(x))$ generator loss with label smoothing in terms of cross-entropy is

$$L_G^{\text{log(1-D)}} = -\mathbb{E}_{x\sim G}\Big[H\big([\lambda_1, 1-\lambda_1], [D_r(x), D_{K+1}(x)]\big)\Big],$$

the negative gradient of which is

$$-\frac{\partial L_G^{\text{log(1-D)}}(x)}{\partial l_r(x)} = D_r(x) - \lambda_1,$$

$$\begin{cases} D_r(x) = \lambda_1 & \text{gradient vanishing} \\ D_r(x) < \lambda_1 & D_r(x) \text{ is optimized towards } 0 \\ D_r(x) > \lambda_1 & D_r(x) \text{ is optimized towards } 1 \end{cases}.$$

Gradient vanishing is a well know training problem of GAN. Optimizing $D_r(x)$ towards 0 or 1 is also not what desired, because the discriminator is mapping real samples to the distribution with $D_r(x) = 1-\lambda_2$.

D.3. The $-\log(D_r(x))$ generator loss

The $-\log(D_r(x))$ generator loss with target $[1-\lambda, \lambda]$ in terms of cross-entropy is

$$L_G^{\text{-log(D)}} = \mathbb{E}_{x\sim G}\Big[H\big([1-\lambda, \lambda], [D_r(x), D_{K+1}(x)]\big)\Big],$$

the negative gradient of which is

$$-\frac{\partial L_G^{\text{-log(D)}}(x)}{\partial l_r(x)} = (1-\lambda) - D_r(x),$$

$$\begin{cases} D_r(x) = 1-\lambda & \text{stationary point} \\ D_r(x) < 1-\lambda & D_r(x) \text{ towards } 1-\lambda \\ D_r(x) > 1-\lambda & D_r(x) \text{ towards } 1-\lambda \end{cases}.$$

In our experiments, we used both-side label smoothing with $\lambda = \lambda_1 = \lambda_2 = 0.75$. And we only introduced label smoothing for $F(\cdot)$, because this part is relatively easy to get overfitting.

E. The Extended (S)AM-GAN Loss

The extended loss of AM-GAN could be:

$$\begin{aligned} L_D^{\text{AM}} = \quad & \mathbb{E}_{(x,y)\sim p_{\text{data}}}\Big[H\big(v(y), D(x)\big)\Big] \\ & + \mathbb{E}_{x\sim G}\Big[H\big(v(K{+}1), D(x)\big)\Big] \\ & + \mathbb{E}_{x\sim p_{\text{unl}}}\Big[H\big(v(x), D(x)\big)\Big] \\ & + H\big(p_{\text{ref}}, R(\mathbb{E}_{x\sim p_{\text{unl}}}[D(x)])\big), \\ L_G^{\text{AM}} = \quad & \mathbb{E}_{x\sim G}\Big[H\big(v(x), D(x)\big)\Big] \\ & + H\big(R(\mathbb{E}_{x\sim p_{\text{data}}}[D(x)]), R(\mathbb{E}_{x\sim G}[D(x)])\big). \end{aligned}$$

where $v(\cdot)$ is the (softmax smoothed) vector that has value λ on the given class logit, and $1-\lambda$ on fake logit or evenly distributed on real class logits.