Gradient Layer: Enhancing the Convergence of Adversarial Training for Generative Models

Atsushi Nitanda[1,2]
atsushi_nitanda@mist.i.u-tokyo.ac.jp

Taiji Suzuki[1,2,3]
taiji@mist.i.u-tokyo.ac.jp

[1]Graduate School of Information Science and Technology, The University of Tokyo
[2]Center for Advanced Intelligence Project, RIKEN
[3]PRESTO, Japan Science and Technology Agency

Abstract

We propose a new technique that boosts the convergence of training generative adversarial networks. Generally, the rate of training deep models reduces severely after multiple iterations. A key reason for this phenomenon is that a deep network is expressed using a highly non-convex finite-dimensional model, and thus the parameter gets stuck in a local optimum. Because of this, methods often suffer not only from degeneration of the convergence speed but also from limitations in the representational power of the trained network. To overcome this issue, we propose an additional layer called the *gradient layer* to seek a descent direction in an *infinite-dimensional space*. Because the layer is constructed in the infinite-dimensional space, we are not restricted by the specific model structure of finite-dimensional models. As a result, we can get out of the local optima in finite-dimensional models and move towards the global optimal function more directly. In this paper, this phenomenon is explained from the functional gradient method perspective of the gradient layer. Interestingly, the optimization procedure using the gradient layer naturally constructs the deep structure of the network. Moreover, we demonstrate that this procedure can be regarded as a discretization method of the gradient flow that naturally reduces the objective function. Finally, the method is tested using several numerical experiments, which show its fast convergence.

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

1 Introduction

Generative adversarial networks (GANs) [5] are a promising scheme for learning generative models. GANs are trained by a discriminator and a generator in an adversarial way. Discriminators are trained to classify between real samples and fake samples drawn from generators, whereas generators are trained to mimic real samples. Although training GANs is quite difficult, adversarial learning succeeded in generating very impressive samples [16], and there are many subsequent studies [11, 17, 14, 4, 23]. Wasserstein GANs (WGANs) [3] are a variant to remedy the mode collapse that appears in the standard GANs by using the Wasserstein distance [21], although they also sometimes generate low-quality samples or fail to converge. Moreover, an improved variant of WGANs was also proposed [6] and it succeeded in generating high-quality samples and stabilizing WGANs. Although these attempts have provided better results, there is still scope to improve the performance of GANs further.

One reason for this difficulty stems from the limitation of the representational power of the generator. If the discriminator is optimized for the generator, the behavior is solely determined by the samples produced from that generator. In other words, for a generator with a poor representational power, the discriminator terminates its learning in the early stage and consequently results in having low discriminative power. However, for a finite-dimensional parameterized generator, the ability to generate novel samples to cheat the discriminators is limited. In addition, the highly non-convex structure of the deep neural network for the generator prevents us from finding a direction for improvement. As a result, the trained parameter gets stuck in a local optimum and the training procedure does not proceed any more.

In this study, we propose a new learning procedure to

overcome the issues of limited representational power and local optimum by introducing a new type of layer called a *gradient layer*. The gradient layer finds a direction for improvement in an infinite-dimensional space by computing the *functional gradient* [12] instead of the ordinary gradient induced by a finite-dimensional model. Because the functional gradient used for the gradient layer is not limited in the tangent space of a finite-dimensional model, it has much more freedom than the ordinary finite-dimensional one. Thanks to this property, our method can break the limit of the local optimum induced by the strong non-convexity of a finite-dimensional model, which gives much more representational power to the generator. We theoretically justify this phenomenon from the functional gradient method perspective and rigorously present a convergence analysis. Interestingly, one iteration of the method can be recognized as inserting one layer into the generator and the total number of iterations is the number of inserted layers. Therefore, our learning procedure naturally constructs the deep neural network architecture by inserting gradient layers. Although, gradient layers can be inserted into an arbitrary layer, they are typically stacked on top of the generator in the final training phase to improve the generated sample quality.

Moreover, we provide another interesting perspective of the gradient layer, i.e., discretization of the gradient flow in the space of probability measures. In Euclidean space, the steepest descent which is the typical optimization method, can be derived by discretizing the gradient flow that naturally produces a curve to reduce the objective function. Because the goal of GANs is to generate a sequence of probability measures moving to the empirical distribution by training samples, it is natural to consider a gradient flow in the space of probability measures defined by a distance between generated distribution and the empirical distribution and to discretize it in order to construct practical algorithms. We show that the functional gradient method for optimizing the generator in the function space is such a discretization method; in other words, the gradient flow can be tracked by stacking gradient layers successively.

The recently proposed SteinGAN [22] is closely related to our work and has a similar flavor, but it is based on another strategy to track gradient flow. That is, since that discretization is mimicked by a fixed-size deep neural network in SteinGAN, it may have the same limitation as typical GANs. By contrast, our method directly tracks the gradient flow in the final phase of training GANs to break the limit of the finite-dimensional generator.

2 Brief Review of Wasserstein GANs

In this section, we introduce WGANs and their variants. Although our proposed gradient layer is applicable to various models, we demonstrate how it performs well for the training of generative models; in particular, we treat Wasserstein GANs as a main application in this paper. Let us start from briefly reviewing WGANs.

WGAN is a powerful generative model based on the 1-Wasserstein distance, defined as the L^1 minimum cost of transporting one probability distribution to the other. Let $\mathcal{X} \subset \mathbb{R}^v$ and $\mathcal{Z} \subset \mathbb{R}^h$ be a compact convex data space and a hidden space, respectively. A typical example of $\mathcal{X}$ is the image space $[0,1]^v$. For a noise distribution μ_n on $\mathcal{Z}$, WGAN learns a data generator $g : \mathcal{Z} \to \mathcal{X}$ to minimize an approximation to the 1-Wasserstein distance between the data distribution μ_D and the push-forward distribution $g_\sharp \mu_n$, which is a distribution that the random variable $g(z)$ follows when $z \sim \mu_n$ (in other words, the distribution obtained by applying a coordinate transform g to $z \sim \mu_n$). That is, WGAN can be described as the following min max problem by using a Kantrovich-Rubinstein duality form of the 1-Wasserstein distance:

$$\min_{g \in \mathcal{G}} \max_{f \in \mathcal{F}} \mathcal{L}(f, g) \stackrel{\text{def}}{=} \mathbb{E}_{x \sim \mu_D}[f(x)] - \mathbb{E}_{z \sim \mu_n}[f \circ g(z)],$$

where $\mathcal{G}$ is the set of generators and $\mathcal{F}$ is an approximate set to the set of 1-Lipschitz continuous functions called *critic*. In WGANs, $\mathcal{G}, \mathcal{F}$ are parameterized neural networks $\{g_\theta\}, \{f_\tau\}$ and the problem is solved by alternate optimization: maximizing and minimizing $\mathcal{L}(f_\tau, g_\theta)$ with respect to τ and θ, alternately.

In practice, to impose the Lipschitz continuity on critics f_τ, penalization techniques were explored. For instance, the original WGANs [3] use weight clipping $\|\tau\|_\infty \leq c$, which implies the upper-bound on the norm of $\nabla_\tau f_\tau$ and makes it Lipschitz continuous. However, it was pointed out in a subsequent study [6] that such a restriction seems to be unnatural and sometimes leads to a low-quality generator or a failure to converge. In the same study, an improved variant of WGANs called WGAN-GP was proposed, which succeeded in stabilizing the optimization process and generating high-quality samples. WGAN-GP [6] adds the gradient penalty $(\|\nabla_{\tilde{x}} f_\tau(\tilde{x})\|_2 - 1)^2$ to the objective function in the training phase of critics, where $\tilde{x}$ is a random interpolation between a training example $x \sim \mu_D$ and a generated sample $g(z) \sim g_{\theta\sharp}\mu_n$, i.e., $\tilde{x} \leftarrow \epsilon x + (1-\epsilon) g(z)$ ($\epsilon \sim U[0,1]$: uniform distribution). DRAGAN [10] is a similar method to WGAN-GP, although it is based on a different motivation. DRAGAN also uses the gradient penalty, but the penalty is imposed on a neighborhood of the data manifold by a

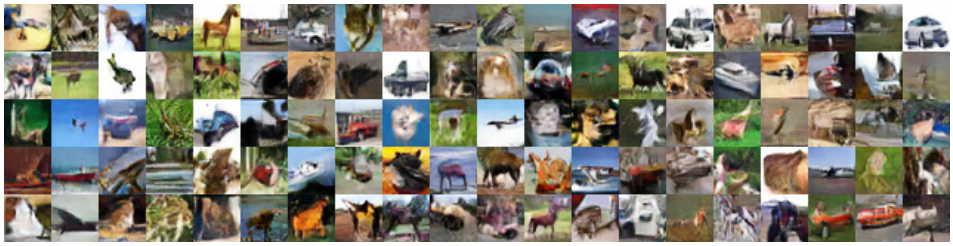

Figure 1: Random samples drawn from the generator trained by Algorithm 1 on the CIFAR-10 dataset.

random perturbation of a training example.

WGAN and its variants are learned by alternately optimizing f_τ and g_θ, as stated above. We can regard this learning procedure as a problem of minimizing $\mathcal{L}(g_\theta) \stackrel{\mathrm{def}}{=} \max_\tau\{\mathcal{L}(f_\tau, g_\theta) - \lambda R_\tau\}$, where R_τ is a penalty term. Let $\mathcal{L}(f_\tau, g_\theta) - \lambda R_\tau$ attain its maximum value at τ_* for g_θ. Then, the gradient $\nabla_\theta \mathcal{L}(g_\theta)$ is the same as $-\mathbb{E}_{\mu_n}[\nabla_\tau f_{\tau_*}^\top \nabla_\theta g_\theta(z)]$ by the envelope theorem [13] when both terms are well-defined. The differentiability of $\mathcal{L}(g_\theta)$ with respect to θ almost everywhere is proved in [3] under a reasonable assumption. Hence, we can apply the gradient method to this problem by approximating this gradient with finite particles generated from μ_n. However, because it is difficult to obtain f_{τ_*}, we run the gradient method for several iterations on training a critic instead of exactly computing f_{τ_*} at each g_θ. We can notice that this learning procedure is quite similar to that of the standard GAN [5].

3 Gradient Layer

In the usual training procedure of WGANs, though more general maps are admissible for the original purpose, generators are parameterized by finite-dimensional space as described in the previous section, and the parameter may get stuck in a local optimum induced by this restriction, or the speed of convergence may reduce. In this work, we propose a gradient layer that accelerates the convergence and breaks the limit of finite-dimensional models. This layer is theoretically derived by the infinite-dimensional optimization method. We first explain the high-level idea of the gradient layer that strictly improves the ability of generator and why our method enhances the convergence of training WGANs.

3.1 High-level idea of gradient layer

Here, we explain gradient layer with intuitive motivation. It is inserted into the generator g in WGANs. We now focus on minimizing $\mathcal{L}(f, g)$ with respect to g under a fixed critic f, that is, we consider the problem $\min_g \mathcal{L}_f(g) \stackrel{\mathrm{def}}{=} \mathbb{E}_{\mu_n}[-f(g(z))]$. Let us split g into two neural networks $g = g_1 \circ g_2$ at arbitrary layer where a new layer is to be inserted. Our purpose is to specify the form of layer ϕ that reduces the objective value by perturbations of inputs $g_2(z)$, i.e., $\mathcal{L}_f(g_1 \circ \phi \circ g_2) \leq \mathcal{L}_f(g)$. Since $\mathcal{L}_f(g_1 \circ \phi \circ g_2)$ is regarded as the integral $\mathbb{E}_{z' \sim g_{2\sharp}\mu_n}[-f(g_1(\phi(z')))]$ with respect to the push-forward distribution $g_{2\sharp}\mu_n$, this purpose is achieved by transporting the input distribution of ϕ along the gradient field $\nabla_{z'} f(g_1(z'))$. Therefore, we propose a gradient layer G_η with one hyperparameter $\eta > 0$ as a map that transforms an input z' to

$$G_\eta(z') = z' + \eta \nabla_{z'} f(g_1(z')). \tag{1}$$

Because the gradient layer depends on the parameters τ, θ of the upper layers f, g_1, we specify the parameter as $G_\eta^{\tau,\theta}$ if needed.

Applying the gradient layer recursively, it further progresses and achieves a better objective. The computation of the gradient layer is quite simple. Actually, simply taking the derivative is sufficient, which can be efficiently executed. Because too many gradient layers would lead to overfitting to the critic f, we stop stacking the gradient layer after an appropriate number of steps. Indeed, if $f \circ g_1$ is Lipschitz continuous, $id + \eta f \circ g_1$ for sufficiently small η is an injection because $(id + \eta f \circ g_1)(z) = (id + \eta f \circ g_1)(z')$ implies $\|z - z'\|_2 \leq \eta \mathcal{L}_{f \circ g_1} \|z - z'\|_2$ where $\mathcal{L}_{f \circ g_1}$ is the Lipschitz constant. Thus, a topology of $\mathrm{supp}(g_{2\sharp}\mu_n)$ is preserved and early stopping is justified. Then, this layer efficiently generates high-quality samples for the critic and the overall adversarial training procedure can be also boosted.

3.2 Powerful optimization ability

Because the gradient layer directly transports inputs as stated above, it strictly improves the objective value if there is room for optimization, unlike finite-dimensional models that may be trapped in local optima induced by the restriction of generators. Indeed, when the gradient layer cannot move inputs, i.e., $G_\eta(z') = z'$, the gradient $\nabla_{z'} f(g_1(z'))$ vanishes on $\mathrm{supp}(g_{2\sharp}\mu_n)$ and there is no chance to improve the

objective value by optimizing g_2 because of the chain rule of derivatives. We now explain this phenomenon more precisely. Let us first consider the training of g_2 in the usual way. We denote by w_2 the parameter of g_2. As stated in the previous section, w_2 is updated by using the gradient

$$\mathbb{E}_{\mu_n}[J_{w_2}^{\top} g_2(z) \nabla_{z'} f(g_1(g_2(z)))], \tag{2}$$

where z' is the input to g_1 and $J_{w_2} g_2(z)$ is the Jacobian matrix of g_2 with respect to w_2. We immediately notice that this gradient (2) is the inner product of $J_{w_2}^{\top} g_2(\cdot)$ and $\nabla_{z'} f(g_1(g_2(\cdot)))$ in $L^2(\mu_n)$-space, and the latter term is the output of the gradient layer. Therefore, when the gradient layer cannot move the inputs, the gradient with respect to w_2 also vanishes. However, even if the gradient vanishes, the gradient layer G_η can move the inputs in general. Thus, whereas the optimization of g_2 may get stuck in a local optimum or be slowed down in this case, the gradient layer strictly improves the quality of the generated samples for the upper layers, because $\nabla_{z'} f(g_1(g_2(z)))$ does not vanish. This is the reason the gradient layer has a greater optimization ability than finite-dimensional models.

3.3 Algorithm description

The overall algorithm is described in this subsection. We adopt WGAN-GP as the base model to which gradient layer is applied. Let us denote by $R_{f_\tau}(\tilde{x})$ a gradient penalty term. In a paper on the improved WGANs [6], the use of a two-sided penalty $(\|\nabla_{\tilde{x}} f_\tau(\tilde{x})\|_2 - 1)^2$ is recommended. However, we also allow the use of the one-sided variant $(\max(\|\nabla_{\tilde{x}} f_\tau(\tilde{x})\|_2 - 1, 0))^2$. As for the place in which the gradient layer is inserted, we can propose several possibilities, e.g., inserting the gradient layer into (i) the top and (ii) the bottom of the layers of the generator. The latter usage is described in the appendix.

The first usage is stacking gradient layers on the top of the generator, except for normalization to fine-tune the generator in the final phase. Although a normalization term such as tanh is commonly stacked on generators to bound the output range of the generators, gradient layers are typically applied before the normalization layer. Since tanh is a fixed function, it is no problem to combine tanh with critics by reinterpreting $\mathcal{F}$ and $\mathcal{X}$. The gradient layer directly handles the generated samples, so that it may significantly improve the sample quality. Because the gradient $\nabla_x f_\tau(x)$ of the critic with respect to data variables provides the direction to improve the quality of the current generated samples, it is expected that we can obtain better results by tracking the gradient iteratively. To compute the output from the gradient layer for a completely new input, we need to reproduce the computation of the gradient layers, which can be realized by saving the history of the parameters of critics and stacking the gradient layers using these parameters. The concrete procedure is described in Algorithm 1. When executing Algorithm 1, the parameter of g_θ is fixed, so that the push-forward measure $g_{\theta\sharp}\mu_n$ is treated as a base probability measure and we denote it by μ_g. Because the gradient layers depend on the history of the parameters in this case, we specify the parameter to be used: G_η^τ. For the parameter τ and the gradient v, we denote by $\mathcal{A}(\tau, v)$ one step of a gradient-based method such as SGD with momentum, Adam [9], and RMSPROP [20]. From the optimization perspective, we show that Algorithm 1 can be regarded as an approximation to the functional gradient method. From this perspective, we show fast convergence of the method under appropriate assumptions where the objective function is smooth and the critics are optimized in each loop. This theoretical justification is described later. Although Algorithm 1 has a great optimization ability, applying the algorithm to large models is difficult because it requires the memory to register parameters; thus, we propose its usage for fine-tuning in the final phase of training a WGAN-GP. After the execution of Algorithm 1, we can generate samples by using the history of critics, the learning rate, and the base distribution as described in Algorithm 2.

Algorithm 1 Finetuning WGAN-GP

Input: The base distribution $\mu_g = g_\sharp \mu_n$, the mini-batch size b, the number of iterations T, the initial parameters τ_0 of the critic, the number of iterations T_0 for the critic, the regularization parameter λ, and the learning rate η for gradient layers.

for $k = 0$ **to** $T - 1$ **do**
  $\tau \leftarrow \tau_k$
  for $k_0 = 0$ **to** $T_0 - 1$ **do**
    $\{x_i\}_{i=1}^b \sim \mu_D^b$, $\{z_i\}_{i=1}^b \sim \mu_g^b$, $\{\epsilon_i\}_{i=1}^b \sim U[0,1]^b$
    $\{z_i\}_{i=1}^b \leftarrow \{G_\eta^{\tau_k} \circ \cdots \circ G_\eta^{\tau_1}(z_i)\}_{i=1}^b$
    $\{\tilde{x}_i\}_{i=1}^b \leftarrow \{\epsilon_i x_i + (1-\epsilon_i) z_i\}_{i=1}^b$
    $v \leftarrow \nabla_\tau \frac{1}{b} \sum_{i=1}^b [f_\tau(z_i) - f_\tau(x_i) + \lambda R_{f_\tau}(\tilde{x}_i)])$
    $\tau \leftarrow \mathcal{A}(\tau, v)$
  end for
  $\tau_{k+1} \leftarrow \tau$
end for
Return $(\tau_1, \ldots, \tau_T)$.

Algorithm 2 Data Generation for Algorithm 1

Input: the seed drawn from base measure $z \sim \mu_g = g_\sharp \mu_n$, the history of parameters $\{\tau_k\}_{k=1}^T$, and the learning rate η for gradient layers.

Return the sample $G_\eta^{\tau_T} \circ \cdots \circ G_\eta^{\tau_1}(z)$.

4 Functional Gradient Method

In this section, we provide mathematically rigorous derivation from the functional gradient method [12] perspective under the Fréchet differentiable (functional differentiable) assumption on $\mathcal{L}$. That is, we consider an optimization problem with respect to a generator in an infinite-dimensional space. For simplicity, we focus on the case where the gradient layer is stacked on top of a generator g and we treat $g_\sharp \mu_n$ as the base measure μ_g. Thus, in the following we omit the notation g in $\mathcal{L}(f, \phi \circ g)$. Let $L^2(\mu_g)$ be the space of $L^2(\mu_g)$-integrable maps from $\mathbb{R}^v$ to $\mathbb{R}^v$, equipped with the $\langle \cdot, \cdot \rangle_{L^2(\mu_g)}$-inner product: for $\forall \phi_1, \forall \phi_2 \in L^2(\mu_g)$,

$$\langle \phi_1, \phi_2 \rangle_{L^2(\mu_g)} = \mathbb{E}_{\mu_g}[\phi_1(z) \top \phi_2(z)].$$

To learn WGAN-GP, we consider the infinite-dimensional problem:

$$\min_{\phi \in L^2(\mu_g)} \max_{f_\tau \in \mathcal{F}} \mathcal{L}(f_\tau, \phi) - \lambda R_{f_\tau},$$

where R_{f_τ} is a gradient penalty term. To achieve this goal, we take a Gâteaux derivative along a given map $v \in L^2(\mu_g)$, i.e., a directional derivative along v. Let us denote $\max_{f_\tau \in \mathcal{F}} \{\mathcal{L}(f_\tau, \phi) - \lambda R_f\}$ by $\mathcal{L}(\phi)$ and $\arg\max_{f_\tau \in \mathcal{F}} \{\mathcal{L}(f_\tau, \phi) - \lambda R_{f_\tau}\}$ by f^*_ϕ and the corresponding parameter by τ^*_ϕ, i.e., $f^*_\phi = f^*_{\tau_\phi}$ for $\phi \in L^2(\mu_g)$. If every $f \in \mathcal{F}$ is Lipschitz continuous and differentiable, we can find that by the envelope theorem and Lebesgue's convergence theorem this derivative takes the form:

$$\frac{d}{dt}\mathcal{L}(\phi + tv)\Big|_{t=0} = -\mathbb{E}_{\mu_g}[\nabla_x f^*_\phi(x)|^\top_{x=\phi(z)} v(z)].$$

Therefore, $-\nabla_x f^*_\phi(x)|_{x=\phi(\cdot)}$ can be regarded as a Fréchet derivative (functional gradient) in $L^2(\mu_g)$ and we denote it by $\nabla_\phi \mathcal{L}(\phi)$, which performs like the usual gradient in Euclidean space. Using this notation, the optimization of $\mathcal{L}(\phi)$ can be accomplished by Algorithm 3, which is a gradient descent method in a function space. Because the functional gradient has the form $-\nabla_x f^*_\phi \circ \phi$, each iteration of the functional gradient method with respect to ϕ is $\phi \leftarrow \phi + \eta \nabla_x f^*_\phi \circ \phi = (id + \eta \nabla_x f^*_\phi) \circ \phi$, where η is the learning rate. We notice here that this iteration is the composition of a perturbation map $id + \eta \nabla_x f^*_\phi$ and a current map ϕ and is nothing but stacking a gradient layer $G^{\tau_*}_\eta$ on $\phi(z)$. In other words, the functional gradient method with respect to ϕ, i.e., Algorithm 3, is the procedure of building a deep neural network by inserting gradient layers, where the total number of iterations is the number of layers. Moreover, we notice that if we view $\nabla_x f^*_\phi$ as a perturbation term, this layer resembles that of *residual networks* [7] which is one of the state-of-the-art architectures in supervised learning tasks.

However, executing Algorithm 3 is difficult in practice because the exact optimization with respect to a critic f to compute $\mathcal{L}(\phi)$ is a hard problem. Thus, we need an approximation and we argue that Algorithm 1 is such a method. This point can be understood as follows. Roughly speaking, it maximizes $\mathcal{L}(f, \phi)$ with respect to f in the inner loop under fixed $\phi = G^{\tau_k}_\eta \circ G^{\tau_{k-1}}_\eta \circ \cdots \circ G^{\tau_1}_\eta$ to obtain an approximate solution τ_{k+1} to τ_* and minimizes that with respect to ϕ in the outer loop by stacking $G^{\tau_{k+1}}_\eta$, which is an approximation to $G^{\tau_*}_\eta$. Thus, Algorithm 1 is an approximated method, but we expect it to achieve fast convergence owing to the powerful optimization ability of the functional gradient method, as shown later. In particular, it is more effective to apply the algorithm in the final phase of training WGAN-GP to fine-tune it, because the optimization ability of parametric models are limited.

Algorithm 3 Functional Gradient Descent

Input: the initial generator g and the learning rate η.
$\phi_0 \leftarrow g$
for $k = 0$ **to** $T - 1$ **do**
  $\phi_{k+1} \leftarrow \phi_k - \eta \nabla_\phi \mathcal{L}(\phi_k)$
end for
Return the function: ϕ_T.

5 Convergence Analysis

Let us provide convergence analysis of Algorithm 3 for the problem of the general form: $\min_\phi \mathcal{L}(\phi)$. The convergence can be shown in an analogous way to that for the finite-dimensional one. To prove this, we make a smoothness assumption on the loss function. We now describe a definition of the smoothness on a Hilbert space whose counterpart in finite-dimensional space is often assumed for smooth non-convex optimization methods.

Definition 1. *Let h be a function on a Hilbert space $(\mathcal{Z}, \langle , \rangle_\mathcal{Z})$. We call that h is L-smooth at z in U if h is differentiable at z and it follows that $\forall z' \in U$.*

$$|h(z') - h(z) - \langle \nabla_z h(z), z' - z \rangle_\mathcal{Z}| \leq \frac{L}{2} \|z' - z\|^2_\mathcal{Z}.$$

The following definition and proposition provide one condition leading to Lipschitz smoothness of $\mathcal{L}$. Let us denote by $\| \cdot \|_{L^\infty(\mu_g)}$ the sup-norm $\|\psi\|_{L^\infty(\mu_g)} = \sup_{\mathrm{supp}(\mu_g)} \|\psi(z)\|_2$ and by $B^\infty_r(\phi)$ a ball of center ϕ and radius r. Let $\hat{\mathcal{L}}(f, \psi) = \mathcal{L}(f, \psi) - \lambda R_f$. In the following we assume f^*_ψ is uniquely defined for $\psi \in L^2(\mu_g)$ and L-smoothness with respect to the input x.

Definition 2. *For positive values r and L, we call that $\mathcal{L}$ is (r, L)-regular at ϕ when the following condition is satisfied; For $\forall \psi \in B_r^\infty(\phi)$, $\hat{\mathcal{L}}(f^*_{\psi'}, \psi)$ is L-smooth at ψ with respect to ψ' in $B_r^\infty(\psi)$.*

Proposition 1. *If $\mathcal{L}$ is (r, L)-regular at ϕ, then $\mathcal{L}$ is $2L$-smooth at ϕ in $B_r^\infty(\psi)$.*

We now show the convergence of Algorithm 3. The following theorem gives the rate to converge to the stationary point.

Theorem 1. *Let us assume the norm of the gradient $\|\nabla_x f^*_\phi(x)\|_2$ is uniformly bounded by α and assume $\mathcal{L}$ is L-smooth at ϕ in $B_r^\infty(\phi)$ for $\forall \phi \in L^2(\mu_g)$. Suppose we run Algorithm 3 with constant learning rate $\eta \leq \min\{1/L, r/\alpha\}$. Then we have for $T \in \mathbb{Z}_+$*

$$\min_{k \in \{0,\dots,T-1\}} \|\nabla_\phi \mathcal{L}(\phi_k)\|^2_{L^2(\mu_g)} \leq \frac{2}{\eta T}(\mathcal{L}(\phi_0) - \mathcal{L}_*),$$

where $\mathcal{L}_* = \inf_\phi \mathcal{L}(\phi)$.

Note that the convergence rate $O(1/T)$ is the same as the gradient descent method for smooth objective in the finite-dimensional one. This means that even though the optimization is executed in the infinite-dimensional space, we do not suffer from the infinite dimensionality in terms of the convergence.

The following rough argument indicates that Algorithm 3 matches with learning WGANs. Let W_1 denote the 1-Wasserstein distance with respect to the Euclidean distance on a compact base space $\mathcal{X} \subset \mathbb{R}^v$. The following proposition is immediately shown by combining existing results [1, 18].

Proposition 2. *Let μ_g be a Borel probability measure on $\mathcal{X}$ and assume μ_g is absolutely continuous with respect to the Lebesgue measure. Then, there exists an optimal transport ψ and it follows that $W_1(\psi_{t\sharp}\mu_g, \mu_D) = (1-t)W_1(\mu_g, \mu_D)$, where $\psi_t = (1-t)id + t\psi$.*

The notion of the optimal transport is briefly introduced in Appendix. By this proposition, there exists a curve ψ_t strictly reducing distance, i.e., $dW_1(\psi_{t\sharp}\mu_g, \mu_D)/dt < 0$ if $\mu_g \neq \mu_D$. Because $\mathcal{L}$ approximates W_1, it is expected that $d\mathcal{L}(\psi_t)/dt < 0$ when μ_g differs from μ_D. Noting that $d\mathcal{L}(\psi_t)/dt = \langle \nabla_\phi \mathcal{L}(\psi_t), \psi - id \rangle_{L^2(\mu_g)}$, the functional gradient $\nabla_\phi \mathcal{L}(\psi_t) \neq 0$ does not vanish and the objective $\mathcal{L}$ may be strictly reduced by Algorithm 3.

6 Gradient Flow Perspective

In Euclidean space, the step of the steepest descent method for minimizing problems can be derived by the discretization of the gradient flow $d\gamma(t)/dt = -\nabla_x F(\gamma(t))$ where F is an objective function on Euclidean space. Because our goal is to move μ_g closer to μ_D, we should consider a gradient flow in the space of probability measures. To make this argument rigorously, we need the continuity equation that characterizes a curve of probability measures and the tangent space where velocities of curves should be contained (c.f., [2]). When these notions are provided, the gradient flow is defined immediately and it is quite natural to discretize this flow to track it well. In this section, we show that Algorithm 3 is such a natural discretization; in other words, building a deep neural network by stacking gradient layers is a discretization procedure of the gradient flow. We refer to [2] for detailed descriptions on this subject, and also refer to [15] for an original method developed by Otto.

6.1 Continuity Equation and Discretization

We denote by $\mathcal{P}$ the set of probability measures on $\mathbb{R}^v$. For $\mu \in \mathcal{P}$, let $\{\phi_t\}_{t\in[0,\delta]}$ be a curve in $L^2(\mu)$ that solves the following ordinary differential equation: for an $L^2(\phi_{t\sharp}\mu)$-integrable vector field v_t on $\mathbb{R}^v$,

$$\phi_0 = id, \quad \frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \ for \ \forall x \in \mathbb{R}^v.$$

Then, this equation derives the curve $\nu_t = \phi_{t\sharp}\mu$ in $\mathcal{P}$, which can be characterized by .

$$\frac{d}{dt}\nu_t + \nabla \cdot (v_t \nu_t) = 0. \tag{3}$$

In other words, the following equation is satisfied

$$\int_I \int_{\mathbb{R}^v} (\partial_t f(x,t) + \nabla_x f(x,t)^\top v_t) d\nu_t dt = 0,$$

for $\forall f \in \mathcal{C}_c^\infty(\mathbb{R}^v \times I)$ where $\mathcal{C}_c^\infty(\mathbb{R}^v \times I)$ is the set of $\mathcal{C}^\infty$-functions with compact support in $\mathbb{R}^v \times I$. Conversely, a *narrowly* continuous family of probability measures ν_t solving equation (3) can be obtained by transport map ϕ_t satisfying $\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x))$ [2]. Thus, equation (3) indicates that v_t drifts the probability measures ν_t. Indeed, v_t can be recognized as the tangent vector of the curve ν_t as discussed below.

Here, we focus on curves in the subset $\mathcal{P}_2 \subset \mathcal{P}$ composed of probability measures with finite second moment. Noting that there is freedom in the choice of v_t modulo divergence-free vector fields $w \in L^2(\nu_t)$ (i.e., $\nabla \cdot (w\nu_t) = 0$), it is natural to consider the equivalence class of $v \in L^2(\nu_t)$ modulo divergence-free vector fields. Moreover, there exists a unique $\Pi(v)$ that attains the minimum $L^2(\nu_t)$-norm in this class: $\Pi(v) = \arg\min_{w \in L^2(\nu_t)}\{\|w\|_{L^2(\nu_t)} \mid \nabla \cdot ((v-w)\nu_t) = 0\}$. Thus, we here introduce the definitions of the tangent space at $\mu \in \mathcal{P}_2$ as follows:

$$T_\mu \mathcal{P}_2 \stackrel{\text{def}}{=} \{\Pi(v) \mid v \in L^2(\mu)\}. \tag{4}$$

The following proposition shows that $T_\mu\mathcal{P}_2$ has the property of the tangent space on the space of probability measures, that is, a perturbation using $v_t \in T_\mu\mathcal{P}_2$ can discretize an absolutely continuous curve ν_t and v_t locally approximates *optimal transport maps*. We denote the 2-Wasserstein distance by W_2.

Proposition 3 ([2]). *Let $\nu_t : I \to \mathcal{P}_2$ be an absolutely continuous curve satisfying the continuity equation with a Borel vector field v_t that is contained in $T_{\nu_t}\mathcal{P}_2$ almost everywhere $t \in I$. Then, for almost everywhere $t \in I$ the following property holds:*

$$\lim_{\delta\to 0} \frac{W_2(\nu_{t+\delta}, (id+\delta v_t)_\sharp \nu_t)}{|\delta|} = 0.$$

In particular, for almost everywhere $t \in I$ such that ν_t is absolutely continuous with respect to the Lebesgue measure, we have

$$\lim_{\delta\to 0} \frac{1}{\delta}(\mathbf{t}_{\nu_t}^{\nu_{t+\delta}} - id) = v_t \quad in\ L^2(\nu_t),$$

where $\mathbf{t}_{\nu_t}^{\nu_{t+\delta}}$ is the unique optimal transport map between ν_t and $\nu_{t+\delta}$.

This proposition suggests the update $\mu^+ \leftarrow (id+v)_\sharp\mu$ for discretizing an absolutely continuous curve in $\mathcal{P}_2$. Note that when $\mu = \phi_\sharp\nu$, $(\nu \in \mathcal{P}_2, \phi \in L^2(\nu))$, the corresponding map to μ^+ is obtained by $\phi^+_\sharp\nu = \mu^+$ where ϕ^+ is a composition as follows:

$$\phi^+ \leftarrow (id+v)\circ\phi = \phi + v\circ\phi. \tag{5}$$

So far, we have introduced the property of continuous curves in $\mathcal{P}_2$ and a method of their discretization. We notice that the above update resembles the update of Algorithm 3. Indeed, we show that the functional gradient method is nothing but a discretization method of the gradient flow derived by the functional gradient $\nabla_\phi\mathcal{L}(\phi)$.

6.2 Discretization of Gradient Flow

We here introduce the gradient flow, which is one of the most straightforward ways to understand Algorithm 3. We have explained that an absolutely continuous curve $\{\nu_t\}_{t\in I}$ in $\mathcal{P}_2$ is well characterized by the continuity equation (3) and we have seen that $\{v_t\}_{t\in I}$ in (3) corresponds to the notion of the velocity field induced by the curve. Such a velocity points in the direction of the particle flow. Moreover, the functional gradient $\nabla_\phi\mathcal{L}(\phi)(\cdot)$ points in an opposite direction to reduce the objective $\mathcal{L}$ at each particle. Thus, these two vector fields exist in the same space and it is natural to consider the following equation:

$$v_t = -\nabla_\phi\mathcal{L}(\phi_t). \tag{6}$$

This equation for an absolutely continuous curve is called the gradient flow [2] and a curve satisfying this will reduce the objective $\mathcal{L}$. Indeed, we can find by the chain rule such a curve $\{\nu_t = \phi_{t\sharp}\mu_g\}_{t\in I}$ that also satisfies the following:

$$\frac{d}{dt}\mathcal{L}(\phi_t) = -\|\nabla_\phi\mathcal{L}(\phi_t)\|^2_{L^2(\nu_t)}.$$

Recalling that ν_t can be discretized well by $\nu_{t+\delta} \sim (id - \delta\nabla_\phi\mathcal{L}(\phi_t))_\sharp\nu_t$, we notice that Algorithm 3 is a discretization method of the gradient flow (6). In other words, building deep neural networks by stacking gradient layers is such a discretization procedure.

7 Experiments

In this section, we show the powerful optimization ability of the gradient layer method empirically on training WGANs. Our implementation is done using Theano [19]. We first used three toy datasets: swiss roll, 8-gaussian, and 25-gaussian datasets (see Figure 2) to confirm the convergence behavior of the gradient layer. The sizes of toy datasets are 500, 500, and 1000, respectively. We next used the CIFAR-10 containing 50,000 images of size 32×32, and STL-10 containing 100,000 images. For STL-10 dataset, we downsample each dimension by 2, resulting image size is 48×48. We reported inception scores [17] for image datasets, which is one of the conventional scores commonly used to measure the quality of generated samples.

Toy datasets We ran Algorithm 1 without pre-training of generators (i.e., $g = id$) on toy datasets from Gaussian noise distributions with the standard deviation 0.5. We used four-layer neural networks for the critics where the dimension of hidden layers were set to 128 for swiss roll and 8-gaussian datasets and 512 for 25-gaussian dataset. We adopted one-sided penalty with regularization parameter $\lambda = 10$. The output of generator was activated by tanh. We used ADAM for training critics with parameters $\alpha = 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.9$, minibatch size $b = 50$. When we run Algorithm 1, gradient layers are stacked below tanh. The learning rates were set to $\eta = 0.1$. The number of inner iterations T_0 for training the critics was set to $5 \times datasize/b$. Figure 2 shows the results for toy datasets for running $T = 100$ iterations of generators. Although we ran the algorithm without pre-training the generators, we obtained better results only for a few iterations. This is surprising, because these toy datasets are difficult to learn and fail to converge in the standard GANs and WGANs. Whereas improved variants of these models overcome this difficulty, they usually require more than 1,000 iterations to converge.

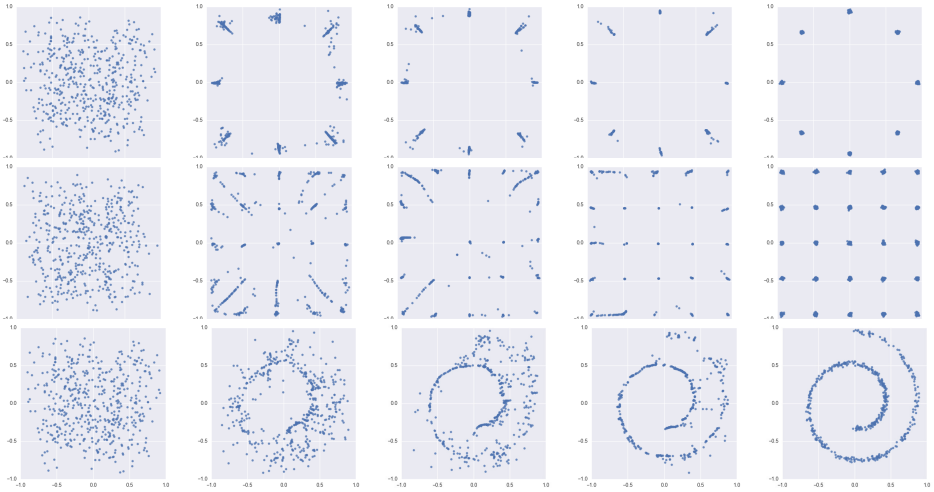

Figure 2: Generated samples by Algorithm 1 on 8-gaussian dataset for $0, 25, 50$, and 100 generator iterations (from left to right) and training data (rightmost).

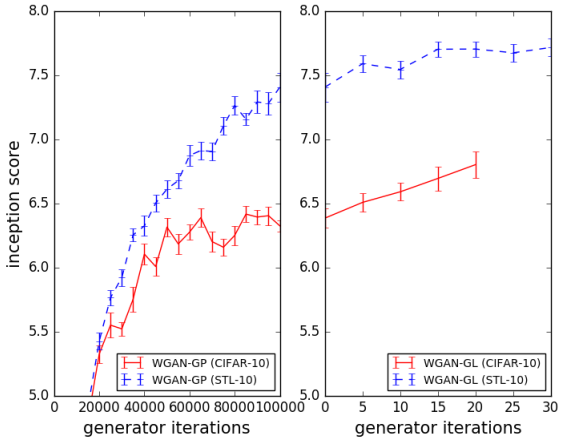

Figure 3: Left: Inception scores obtained by WGAN-GP, Right: Inception scores obtain by Algorithm 1 starting from the result of WGAN-GP.

CIFAR-10 and STL-10 We first trained WGAN-GP with a two-sided penalty ($\lambda = 10$) on the CIFAR-10 and STL-10 datasets. We used DCGAN for both the critic and the generator. The batch normalization [8] was used only for the generator. The critic and the generator were trained by using ADAM with $\alpha = 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.9$, and minibatch size $b = 64$. The number of inner iterations for training the critics were 5 and we ran ADAM for 10^5-iterations. The left side of Figure 3 shows the inception scores obtained by WGAN-GP. It seems that the learning procedure is slowed down in a final training phase, especially for CIFAR-10. The final inception score on CIFAR-10 and STL-10 are 6.32 and 7.40, respectively. We next ran Algorithm 1 starting from the result of WGAN-GP. The critics were trained by ADAM with the same parameters, except for $\alpha = 5 \times 10^{-5}$ and $T_0 = datasize/b$. The learning rates were set to 0.5 for CIFAR-10 and 0.3 for STL-10. The right side of Figure 3 shows the inception scores obtained by Algorithm 1. Note that, since we focus on the optimization ability of generators, we plotted results with the horizontal axis as the number of outer-iterations. We observed a rapid increase in the inception scores, which were improved to 6.80 and 7.71 on CIFAR-10 and STL-10, respectively.

8 Conclusion

We have proposed a gradient layer that enhances the convergence speed of adversarial training. Because this layer is based on the perspective of infinite-dimensional optimization, it can avoid local optima induced by non-convexity and parameterization. We have also provided two perspectives of the gradient layer: (i) the functional gradient method and (ii) the discretization procedure of the gradient flow. We have proven the fast convergence of the gradient layer by utilizing this perspective, and experimental results have empirically shown its reliable performance.

Acknowledgements

This work was partially supported by MEXT KAKENHI (25730013, 25120012, 26280009, 15H01678 and 15H05707), JST-PRESTO (JPMJPR14E4), and JST-CREST (JPMJCR14D7, JPMJCR1304).

References

[1] Luigi Ambrosio. Lecture notes on optimal transport problems. In *Mathematical aspects of evolving interfaces*, pages 1–52. Springer, 2003.

[2] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29*, pages 2172–2180. 2016.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*. 2014.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[10] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. How to train your DRAGAN. *arXiv preprint arXiv:1705.07215*, 2017.

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

[12] David G Luenberger. *Optimization by vector space methods*. John Wiley & Sons, 1969.

[13] Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

[14] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*, pages 271–279. 2016.

[15] Felix Otto. The geometry of dissipative evolution equations: the porous medium equation. 2001.

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

[17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[18] Vladimir N Sudakov. *Geometric problems in the theory of infinite-dimensional probability distributions*. Number 141. American Mathematical Soc., 1979.

[19] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[20] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[21] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

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

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