Thesis Proposal

Learning Transformation via Implicit Generative Models

Chun-Liang Li

November

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Barnabás Póczos (Chair), CMU,
Jeff Schneider, CMU,
Ruslan Salakhutdinov, CMU,
Phillip Isola, MIT

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Implicit Generative Model, Adversarial Networks, Deep Generative Model

Abstract

One of the fundamental problems in machine learning and statistics is learning generative models of data. Explicit generative models by modeling the distribution density have been widely studied in different applications. However, using combinations of simple parametric distributions is usually difficult to model complex data, such as natural images. *Implicit generative models* (IGM) regain attention with explosion of interests recently, which approximates sampling processes instead of probabilistic densities. By combining this with deep neural networks, IGMs have yielded impressive empirical performance.

Learning IGMs is a process which transforms a known source distribution to data distributions. The source distribution can be a simple parametric distribution, or the distribution for the other source of data (e.g. conditional generation). While there are new algorithms for learning IGMs, its theoretical properties are under explored. The first thrust of this thesis is to understand statistical guarantees of learning IGMs. By connecting IGMs with two-sample test, we propose a new generic algorithm that can be built on the top of many existing algorithms and bring performance improvement. We also study the statistical properties of learning IGMs, such as weak* topology and continuity, which is a step forward in building its theoretical foundations. On the other hand, there are different types of data that we are interested in, such as text, images, point clouds or meshes. We develop algorithms for learning IGMs on those data by exploiting their underlying structures.

Beyond learning to sample from data distributions via IGM, we study conditional generation via IGMs by learning transformation from the conditioning to the target. This opens up to broader and more practical applications in the real-world. Various problems from different domains can be solved via IGMs, including style transformations of languages, linear inverse problems of images, and deformations of 3D models.

Finally, instead of simply learning IGM transformation via powerful models, such as deep neural networks, we want to leverage human priors into model designs to not only reduce the parametrized model size, which saves computational overhead, but also lead to interpretable results. Two cases will be studied, including encoding prior knowledge into differentiable and non-differentiable methods.

To summarize, we study three major aspects of transformations via IGMs, including *noise to data*: learning to model data directly, *data to data*: learning to adapt from one domain to the other, and *priors to data*: learning to generate with interpretation.

Contents

Chapter 1

Introduction

In machine learning and statistics, the heart of unsupervised learning lies in learning generative models that describes probabilistic distribution of the data. Learning generative models has demonstrated success in various applications, including but not limited to image captioning [111], image processing [43, 56], science simulation [65], chemical design [53], anomaly detection and art creation [23]. Learning good generative models also benefits other major tasks in machine learning, for example, data augmentation in supervised and semi-supervised learning [92], data imputation in unsupervised learning [113], planning [25] in reinforcement learning.

There are two categories of generative models, prescribed (explicit) and implicit models [15]. *Explicit generative models* explicitly model density of the underlying distribution, which is widely studied in last decades. Take a coin-toss as an example. The outcome could be modeled by a Bernoulli distribution. In Bayesian statistics, graphical models [49] are widely studied with different mixture and hierarchical structures, where each node is modeled by some parametric distributions. However, explicit models, which use combinations of simple parametric distributions to model densities, do not have much success in modeling complex , structured and high dimensional data, such as natural images. In non-parametric statistics, kernel density estimation [105] only assumes limited smoothness for modeling data density, but it usually suffers from curse of dimensionality in practice.

In many applications, we are actually interested in sampling from the distribution instead of measuring the density. The other category of generative model is *implicit generative model* (IGM), which models the *sampling process* of underlying distributions. In the coin-toss example, from physics, the outcome is determined by every initial conditions (environments, impulse, angles). The randomness is not from the Bernoulli distribution but from the randomness of those initialization. The physics system *transforms* these randomness into coin-toss outcomes. Following the same idea, IGM models the sampling by modeling a transformation process, which transforms a prior randomness into data of interest. The study of IGMs with neural networks can be dated back to MacKay [66]. It regains attentions with recent breakthrough in deep learning [29] with different successful applications aforementioned.

The first goal of this thesis will focus on advancing the understanding of the learnt IGM transformation from a statistical perspective and improving existing algorithms. Next, we

study on how to learn IGMs from different types of data and how to learn transformations between different modalities. Last but not least, we explore how to leverage human prior knowledge and incoporate them into IGM model, which provides interpretable generative models.

In Chapter 2, we revisit classical two-sample test in statistics and introduce its connection with learning IGMs. From this perspective, we propose a generic algorithm MMD GAN [59], which can be built on the top of most existing GAN algorithms with improvements brought by the tool for two-sample tests, maximum mean discrepancy [34]. In addition to using kernel tools to improve learning IGMs, IGMs can be used to improve kernel learning [61], which is a long studied problem, the other way round. The improved kernel from IGMs can be used to improve IGMs learning further.

In Chapter 3, we study how to learn IGMs on different data types. Firstly, we study the effectiveness of gradient penalty regularized IGM model on learning to generate texts [76]. We show that gradient penalty encourages leave-one-out (LOO) conditional distribution matching, which follows the standard assumptions in language structures. However, the performance is barely satisfactory compared with standard language models. We aim to address this issue by incorporating more language structures in addition to weak LOO conditioning. The second data type that we tried to tackle is point clouds [60] by utilizing ideas inspired from Hierarchical Bayesian. Next, we propose to move one step further from point cloud to learning to generate meshes.

Although deep learning provides a family of models, which has large capacity with developed tools to train and regularize it, there are many existing human-designed models that incorporate strong prior knowledges. For example, many well-crafted graphics rendering pipelines proposed in computer graphics are based on physics rules. We plan to explore a learnt transformations by utilizing human prior knowledges.

In Chapter 4, we study conditional generations via IGMs, which learns more useful transformations in real-world applications. We propose a projection network, which learns a projected transformation to project noisy images onto natural images. By incorporating these into standard iterative algorithms (e.g. ADMM), we can solve a family of linear inverse problems without retraining. Similar to Chapter 3, we want to explore conditional generation in text generation, including text style transformation and machine translation beyond image data.

Chapter 2

Learning Implicit Generative Models

Assume we are given data $\{x_i\}_{i=1}^n$, where $x_i \in \mathcal{X}$ and $x_i \sim \mathbb{P}_r$, we are interested in sampling from $\mathbb{P}_r$. For sampling, it is not necessary to estimate the density of $\mathbb{P}_r$. Instead, Implicit Generative Model (IGM) trains a generator g_θ parameterized by θ to transform samples $z \sim \mathbb{P}_\mathcal{Z}$, where $z \in \mathcal{Z}$ is a base distribution or is called as *noise*, into $g_\theta(z) \sim \mathbb{Q}_\theta$ such that $\mathbb{Q}_\theta \approx \mathbb{P}_r$. The core problem of learning IGMs is to define a *distance* between $\mathbb{Q}_\theta$ and $\mathbb{P}_r$ if we can only access samples from these two distributions. We then train g_θ by optimizing the defined distance. Generative Adversarial Network (GAN) [30] trains an auxiliary network f_ϕ to estimate the distance between $\mathbb{P}_r$ and $\mathbb{Q}_\theta$. Different probabilistic (pseudo) metrics have been studied [4, 30, 78, 118] under GAN framework. In addition to adversarial loss in *dual form*, the learning IGMs in *primal* is also studied [48, 69, 99]. Next, we will use GAN and IGM interchangably if the IGM referred to is trained via adversarial loss.

2.1 IGMs and Two-Sample Test

The heart of learning IGMs is to measure distances between distributions of generators and data via samples. Distinguishing two distributions by finite samples is known as *Two-Sample Test* in statistics. One way to conduct two-sample test is via kernel maximum mean discrepancy (MMD) [33]. Given two distributions $\mathbb{P}$ and $\mathbb{Q}$, and a kernel k, the square of MMD distance is defined as

$$M_k(\mathbb{P}, \mathbb{Q}) = \|\mu_\mathbb{P} - \mu_\mathbb{Q}\|_\mathcal{H}^2 = \mathbb{E}_\mathbb{P}[k(x, x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x, y)] + \mathbb{E}_\mathbb{Q}[k(y, y')].$$

Theorem 1. *[33] Given a kernel k, if k is a characteristic kernel, then $M_k(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$.*

Example One example of characteristic kernel is Gaussian kernel $k(x, x') = \exp(\|x - x'\|^2)$. Based on Theorem 1, [21, 62] propose generative moment-matching network (GMMN), which trains g_θ by

$$\min_\theta M_k(\mathbb{P}_r, \mathbb{Q}_\theta), \tag{2.1}$$

with a fixed Gaussian kernel k rather than training an additional discriminator f as GAN.

2.1.1 MMD with Kernel Learning

In practice we use finite samples from distributions to estimate MMD distance. Given $X = \{x_1, \cdots, x_n\} \sim \mathbb{P}$ and $Y = \{y_1, \cdots, y_n\} \sim \mathbb{Q}$, one estimator of $M_k(\mathbb{P}, \mathbb{Q})$ is

$$\hat{M}_k(X, Y) = \frac{1}{\binom{n}{2}} \sum_{i \neq i'} k(x_i, x_i') - \frac{2}{\binom{n}{2}} \sum_{i \neq j} k(x_i, y_j) + \frac{1}{\binom{n}{2}} \sum_{j \neq j'} k(y_j, y_j').$$

Because of the sampling variance, $\hat{M}(X, Y)$ may not be zero even when $\mathbb{P} = \mathbb{Q}$. We then conduct hypothesis test with null hypothesis $H_0 : \mathbb{P} = \mathbb{Q}$. For a given allowable probability of false rejection α, we can only reject H_0, which imply $\mathbb{P} \neq \mathbb{Q}$, if $\hat{M}(X, Y) > c_\alpha$ for some chose threshold $c_\alpha > 0$. Otherwise, $\mathbb{Q}$ passes the test and $\mathbb{Q}$ is indistinguishable from $\mathbb{P}$ under this test. Please refer to [33] for more details.

Intuitively, if kernel k cannot result in high MMD distance $M_k(\mathbb{P}, \mathbb{Q})$ when $\mathbb{P} \neq \mathbb{Q}$, $\hat{M}_k(\mathbb{P}, \mathbb{Q})$ has more chance to be smaller than c_α. Then we are unlikely to reject the null hypothesis H_0 with finite samples, which implies $\mathbb{Q}$ is not distinguishable from $\mathbb{P}$. Therefore, instead of training g_θ via (2.1) with a pre-specified kernel k as GMMN, we consider training g_θ via

$$\min_\theta \max_{k \in \mathcal{K}} M_k(\mathbb{P}_r, \mathbb{Q}_\theta), \tag{2.2}$$

which takes different possible characteristic kernels $k \in \mathcal{K}$ into account. On the other hand, we could also view (2.2) as replacing the fixed kernel k in (2.1) with the *adversarially learned kernel* $\arg\max_{k \in \mathcal{K}} M_k(\mathbb{P}_r, \mathbb{Q}_\theta)$ to have stronger signal where $\mathbb{P} \neq \mathbb{Q}_\theta$ to train g_θ. We refer interested readers to [26] for more rigorous discussions about testing power and increasing MMD distances.

However, it is difficult to optimize over all characteristic kernels when we solve (2.2). By [31, 33] if f is a injective function and k is characteristic, then the resulted kernel $\tilde{k} = k \circ f$, where $\tilde{k}(x, x') = k(f(x), f(x'))$ is still characteristic. If we have a family of injective functions parameterized by ϕ, which is denoted as f_ϕ, we are able to change the objective to be

$$\min_\theta \max_\phi M_{k \circ f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta), \tag{2.3}$$

Here we consider the case that combining Gaussian kernels with injective functions f_ϕ, where $\tilde{k}(x, x') = \exp(-\|f_\phi(x) - f_\phi(x)'\|^2)$. One example function class of f is $\{f_\phi | f_\phi(x) = \phi x, \phi > 0\}$, which is equivalent to the kernel bandwidth tuning. Next, we abuse the notation $M_{f_\phi}(\mathbb{P}, \mathbb{Q})$ to be MMD distance given the composition kernel of Gaussian kernel and f_ϕ in the following. Note that [35] considers the linear combination of characteristic kernels, which can also be incorporated into the discussed composition kernels. A more general kernel is studied in [108].

2.1.2 Properties of MMD with Kernel Learning

Arjovsky et al. [4] discuss different distances between distributions adopted by existing deep learning algorithms, and show many of them are discontinuous, such as Jensen-Shannon divergence [30] and Total variation [118], except for Wasserstein distance. The

discontinuity makes the gradient descent infeasible for training. From (2.3), we train g_θ via minimizing $\max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta)$. Next, we show $\max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta)$ also enjoys the advantage of being a continuous and differentiable objective in θ under mild assumptions.

Assumption 2. *$g : \mathcal{Z} \times \mathbb{R}^m \to \mathcal{X}$ is locally Lipschitz, where $\mathcal{Z} \subseteq \mathbb{R}^d$. We will denote $g_\theta(z)$ the evaluation on (z, θ) for convenience. Given f_ϕ and a probability distribution $\mathbb{P}_z$ over $\mathcal{Z}$, g satisfies Assumption 2 if there are local Lipschitz constants $L(\theta, z)$ for $f_\phi \circ g$, which is independent of ϕ, such that $\mathbb{E}_{z\sim\mathbb{P}_z}[L(\theta, z)] < +\infty$.*

Theorem 3. *The generator function g_θ parameterized by θ is under Assumption 2. Let $\mathbb{P}_r$ be a fixed distribution over $\mathcal{X}$ and Z be a random variable over the space $\mathcal{Z}$. We denote $\mathbb{Q}_\theta$ the distribution of $g_\theta(Z)$, then $\max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta)$ is continuous everywhere and differentiable almost everywhere in θ.*

If g_θ is parameterized by a feed-forward neural network, it satisfies Assumption 2 and can be trained via gradient descent as well as propagation, since the objective is continuous and differentiable followed by Theorem 3.

Theorem 4. *(weak* topology) Let $\{\mathbb{P}_n\}$ be a sequence of distributions. Considering $n \to \infty$, under mild Assumption, $\max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{P}_n) \to 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P}_r$, where $\xrightarrow{D}$ means* converging in distribution *[105].*

Theorem 4 shows that $\max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{P}_n)$ is a sensible cost function to the distance between $\mathbb{P}_r$ and $\mathbb{P}_n$. The distance is decreasing when $\mathbb{P}_n$ is getting closer to $\mathbb{P}_r$, which benefits the supervision of the improvement during the training.

2.2 MMD GAN

To approximate (2.3), we use neural networks to parameterize g_θ and f_ϕ with expressive power. For g_θ, the assumption is locally Lipschitz, where commonly used feed-forward neural networks satisfy this constraint. Also, the gradient $\bigtriangledown_\theta (\max_\phi f_\phi \circ g_\theta)$ has to be bounded, which can be done by clipping ϕ [4] or gradient penalty [37]. The non-trivial part is f_ϕ has to be injective. For an injective function f, there exists an function f^{-1} such that $f^{-1}(f(x)) = x, \forall x \in \mathcal{X}$ and $f^{-1}(f(g(z))) = g(z), \forall z \in \mathcal{Z}$[1], which can be approximated by an autoencoder. In the following, we denote $\phi = \{\phi_e, \phi_d\}$ to be the parameter of discriminator networks, which consists of an encoder f_{ϕ_e}, and train the corresponding decoder $f_{\phi_d} \approx f^{-1}$ to regularize f. The objective (2.3) is relaxed to be

$$\min_\theta \max_\phi M_{f_{\phi_e}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z}))) - \lambda \mathbb{E}_{y\in\mathcal{X}\cup g(\mathcal{Z})} \|y - f_{\phi_d}(f_{\phi_e}(y))\|^2. \tag{2.4}$$

Note that we ignore the autoencoder objective when we train θ, but we use (2.4) for a concise presentation. We note that the empirical study suggests autoencoder objective is not necessary to lead the successful GAN training as we will show in Section 2.2.4, even though the injective property is required in Theorem 1.

The proposed algorithm is similar to GAN [30], which aims to optimize two neural networks g_θ and f_ϕ in a minmax formulation, while the meaning of the objective is different.

[1]Note that injective is not necessary invertible.

Algorithm 1 Caption

Input: α the learning rate, c the clipping parameter, B the batch size, n_c the number of iterations of discriminator per generator update.
while θ has not converged **do**
  for $t = 1, \dots, n_c$ **do**
    Sample a minibatches $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X})$ and $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$
    $g_\phi \leftarrow \nabla_\phi M_{f_{\phi_e}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z}))) - \lambda \mathbb{E}_{y \in \mathcal{X} \cup g(\mathcal{Z})} \|y - f_{\phi_d}(f_{\phi_e}(y))\|^2$
    $\phi \leftarrow \phi + \alpha \cdot \mathrm{RMSProp}(\phi, g_\phi)$
    $\phi \leftarrow \mathrm{clip}(\phi, -c, c)$
  end for
  Sample a minibatches $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X})$ and $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$
  $g_\theta \leftarrow \nabla_\theta M_{f_{\phi_e}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z})))$
  $\theta \leftarrow \theta - \alpha \cdot \mathrm{RMSProp}(\theta, g_\theta)$
end while

In [30], f_{ϕ_e} is a discriminator (binary) classifier to distinguish two distributions. In the proposed algorithm, distinguishing two distribution is still done by two-sample test via MMD, but with an adversarially learned kernel parametrized by f_{ϕ_e}. g_θ is then trained to pass the hypothesis test. Because of the similarity of GAN, we call the proposed algorithm *MMD GAN* [59]. We present an implementation with the weight clipping in Algorithm **??**, but one can easily extend to other Lipschitz approximations, such as gradient penalty [37].

Encoding Perspective of MMD GAN: Besides from using kernel selection to explain MMD GAN, the other way to see the proposed MMD GAN is viewing f_{ϕ_e} as a feature transformation function, and the kernel two-sample test is performed on this transformed feature space (i.e., the code space of the autoencoder). The optimization is finding a manifold with stronger signals for MMD two-sample test. From this perspective, [62] is the special case of MMD GAN if f_{ϕ_e} is the identity mapping function. In such circumstance, the kernel two-sample test is conducted in the original data space.

Connection with WGAN: If we composite f_ϕ with linear kernel instead of Gaussian kernel, and restricting the output dimension h to be 1, we then have the objective

$$\min_\theta \max_\phi \|\mathbb{E}[f_\phi(x)] - \mathbb{E}[f_\phi(g_\theta(z))]\|^2. \tag{2.5}$$

Parameterizing f_ϕ and g_θ with neural networks and assuming $\exists \phi' \in \Phi$ such $f'_\phi = -f_\phi, \forall \Phi$, recovers Wasserstein GAN (WGAN) [4] [2]. If we treat $f_\phi(x)$ as the data transform function, WGAN can be interpreted as first-order moment matching (linear kernel) while MMD GAN aims to match infinite order of moments with Gaussian kernel form Taylor expansion [62]. Theoretically, Wasserstein distance has similar theoretically guarantee as Theorem 1, 3 and 4. In practice, [5] show neural networks does not have enough capacity to approximate Wasserstein distance. [77] also propose McGAN that matches second order moment from the primal-dual norm perspective. However, the proposed algorithm requires matrix

[2] Theoretically, they are not equivalent but the practical neural network approximation results in the same algorithm.

(tensor) decompositions because of exact moment matching [116], which is hard to scale to higher order moment matching. On the other hand, by giving up exact moment matching, MMD GAN can match high-order moments with kernel tricks.

Difference from Other Works with Autoencoders: Energy-based GANs [117, 118] also utilizes the autoencoder (AE) in its discriminator from the energy model perspective, which minimizes the reconstruction error of real samples x while maximize the reconstruction error of generated samples $g_\theta(z)$. In contrast, MMD GAN uses AE to approximate invertible functions by minimizing the reconstruction errors of *both* real samples x and generated samples $g_\theta(z)$. Also, [4] show EBGAN approximates total variation, with the drawback of discontinuity, while MMD GAN optimizes MMD distance. The other line of works [48, 62, 69] aims to match the AE codespace $f(x)$, and utilize the decoder $f_{dec}(\cdot)$. [48, 69] match the distribution of $f(x)$ and z via different distribution distances and generate data (e.g. image) by $f_{dec}(z)$. [62] use MMD to match $f(x)$ and $g(z)$, and generate data via $f_{dec}(g(z))$. The proposed MMD GAN matches the $f(x)$ and $f(g(z))$, and generates data via $g(z)$ directly as GAN. [100] is similar to MMD GAN but it considers KL-divergence without showing continuity and weak* topology guarantee.

2.2.1 Feasible Set Reduction

Theorem 5. *For any f_ϕ, there exists f'_ϕ such that $M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta) = M_{f'_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta)$ and $\mathbb{E}_x[f_\phi(x)] \succeq \mathbb{E}_z[f_{\phi'}(g_\theta(z))]$.*

With Theorem 5, we could reduce the feasible set of ϕ during the optimization by solving

$$\min_\theta \max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta) \quad s.t. \;\; \mathbb{E}[f_\phi(x)] \succeq \mathbb{E}[f_\phi(g_\theta(z))]$$

which the optimal solution is still *equivalent* to solving (2.2).

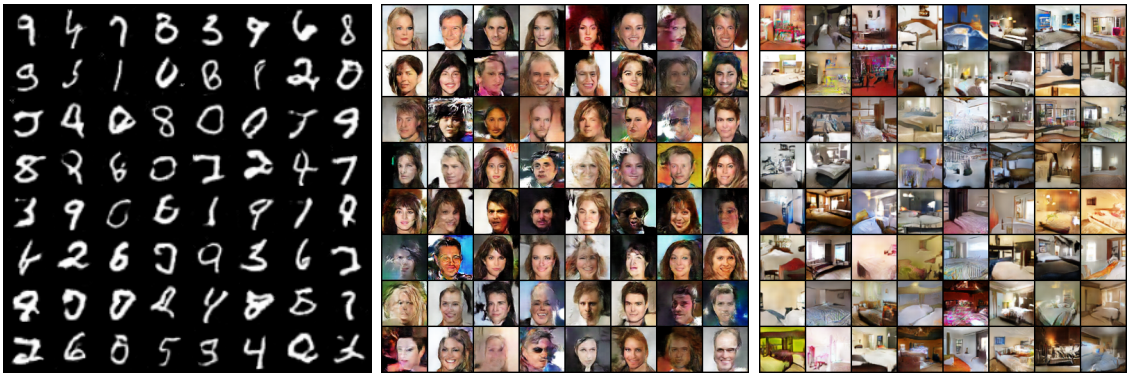
However, it is hard to solve the constrained optimization problem with backpropagation. We relax the constraint by ordinal regression [39] to be

$$\min_\theta \max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta) + \lambda \min\big(\mathbb{E}[f_\phi(x)] - \mathbb{E}[f_\phi(g_\theta(z))], 0\big),$$

which only penalizes the objective when the constraint is violated. In practice, we observe that reducing the feasible set makes the training faster and stabler.

2.2.2 Better Lipschitz Approximation and Necessity of Auto-Encoder

We used weight-clipping for Lipschitz constraint in Assumption 2. Another approach for obtaining a discriminator with the similar constraints that approximates a Wasserstein distance is [37], where the gradient of the discriminator is constrained to be 1 between the generated and data points. Inspired by [37], an alternative approach is to apply the gradient constraint as a regularizer to the witness function for a different IPM, such as the MMD. This idea was first proposed in [6] for the Energy Distance (in parallel to our submission), which was shown in [32] to correspond to a gradient penalty on the witness function for any RKHS-based MMD.



(a) WGAN MNIST (b) WGAN CelebA (c) WGAN LSUN

(d) MMD GAN MNIST (e) MMD GAN CelebA (f) MMD GAN LSUN

Figure 2.1: Generated samples from WGAN and MMD GAN on MNIST, CelebA, and LSUN bedroom datasets.

2.2.3 Other Related GAN Works

In addition to the discussed works, there are several extended works of GAN. [92] proposes using the linear kernel to match first moment of its discriminator's latent features. [97] considers the variance of empirical MMD score during the training. Also, [97] only improves the latent feature matching in [92] by using kernel MMD, instead of proposing an adversarial training framework as we studied. [7] uses Wasserstein distance to match the distribution of autoencoder loss instead of data. One can consider to extend [7] to higher order matching based on the proposed MMD GAN. A parallel work [6] use energy distance, which can be treated as MMD GAN with different kernel. However, there are some potential problems of its critic. More discussion can be referred to [32].

2.2.4 Preliminary Results

There are several representative extensions of GANs. We consider recent state-of-art WGAN [4] based on DCGAN structure [87], because of the connection with MMD GAN discussed aforementioned. The results are shown in Figure 3.4. For MNIST, the digits generated from WGAN in Figure 2.1a are more unnatural with peculiar strikes. In Con-

trary, the digits from MMD GAN in Figure 2.1d enjoy smoother contour. Furthermore, both WGAN and MMD GAN generate diversified digits, avoiding the mode collapse problems appeared in the literature of training GANs. For CelebA, we can see the difference of generated samples from WGAN and MMD GAN. Specifically, we observe varied poses, expressions, genders, skin colors and light exposure in Figure 2.1b and 2.1e. By a closer look (view on-screen with zooming in), we observe that faces from WGAN have higher chances to be blurry and twisted while faces from MMD GAN are more spontaneous with sharp and acute outline of faces. As for LSUN dataset, we could not distinguish salient differences between the samples generated from MMD GAN and WGAN.

Quantitative Analysis

To quantitatively measure the quality and diversity of generated samples, we compute the inception score [92] on CIFAR-10 images. The inception score is used for GANs to measure samples quality and diversity on the pretrained inception model [92]. Models that generate collapsed samples have a relatively low score. Table 2.1 lists the results for $50K$ samples generated by various unsupervised generative models trained on CIFAR-10 dataset. The inception scores of [19, 92, 104] are directly derived from the corresponding references.

Although both WGAN and MMD GAN can generate sharp images as we show in Figure 3.4, our score is better than other GAN techniques except for DFM [104]. This seems to confirm empirically that higher order of moment matching between the real data and fake sample distribution benefits generating more diversified sample images. Also note DFM appears compatible with our method and combing training techniques in DFM is a possible avenue for future work.

Method	Scores $\pm$ std.
Real data	$11.95 \pm .20$
DFM [104]	7.72
ALI [19]	5.34
Improved GANs [92]	4.36
MMD GAN	**6.17** $\pm$ .07
WGAN	$5.88 \pm .07$
GMMN	$3.47 \pm .03$

Table 2.1: Inception scores

2.3 Implicit Kernel Learning

One key component of the proposed MMD GAN is kernel learning (2.2), which replies on the feature map learning parametrized by deep neural networks combined with Gaussian kernels. Although the composition kernel with a learned feature embedding f_φ is powerful,

choosing a good base kernel k is still crucial in practice [9]. Different base kernels for MMD GAN, such as rational quadratic kernel [9] and distance kernel [6], have been studied. Instead of choosing it by hands, this base kernels can be learnt via IGMs.

We start from reviewing basic properties of kernels. Given data $x \in \mathbb{R}^d$, kernel methods compute the inner product of the feature transformation $\phi(x)$ in a high-dimensional Hilbert space H via a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is defined as $k(x, x') = \langle \phi(x), \phi(x') \rangle_H$, where $\phi(x)$ is usually high or even infinitely dimensional. If k is shift invariant (i.e. $k(x, y) = k(x-y)$), we can represent k as an expectation with respect to a spectral distribution $\mathbb{P}_k(\omega)$.

Bochner's theorem [91] A continuous, real valued, symmetric and shift-invariant function k on $\mathbb{R}^d$ is a positive definite kernel if and only if there is a positive finite measure $\mathbb{P}_k(\omega)$ such that

$$k(x - x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x-x')} d\mathbb{P}_k(\omega) = \mathbb{E}_{\omega \sim \mathbb{P}_k} \left[e^{i\omega^\top (x-x')} \right].$$

2.3.1 Kernel Learning with IGMs

We restrict ourselves to learning shift invariant kernels. According to that, learning kernels is equivalent to learning a spectral distribution by optimizing

$$\begin{aligned} &\arg\max_{k \in \mathcal{K}} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') k(x, x') \right] = \\ &\arg\max_{k \in \mathcal{K}} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') \mathbb{E}_{\omega \sim \mathbb{P}_k} \left[e^{i\omega^\top (x-x')} \right] \right], \end{aligned} \tag{2.6}$$

where F is a task-specific objective function and $\mathcal{K}$ is a set of kernels. (2.12) covers many popular objectives, such as kernel alignment [28] and MMD distance [33]. Existing works [79, 107] learn the spectral density $\mathbb{P}_k(\omega)$ with *explicit* forms via parametric or non-parametric models. When we learn kernels via (2.12), it may not be necessary to model the density of $\mathbb{P}_k(\omega)$, as long as we are able to estimate kernel evaluations $k(x - x') = \mathbb{E}_\omega[e^{i\omega^\top (x-x')}]$ via sampling from $\mathbb{P}_k(\omega)$ [88]. Follow the idea of IGMs, we propose an *Implicit Kernel Learning (IKL)* method by modeling $\mathbb{P}_k(\omega)$ via an implicit generative model $h_\psi(\nu)$, where $\nu \sim \mathbb{P}(\nu)$, which results in

$$k_\psi(x, x') = \mathbb{E}_\nu \left[e^{i h_\psi(\nu)^\top (x-x')} \right]. \tag{2.7}$$

We then reduce (2.12) to solve

$$\arg\max_{\psi} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') \mathbb{E}_\nu \left(e^{i h_\psi(\nu)^\top (x-x')} \right) \right]. \tag{2.8}$$

The gradient of (2.8) can be represented as

$$\sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \mathbb{E}_\nu \left[\nabla_\psi F_i(x, x') e^{i h_\psi(\nu)^\top (x-x')} \right].$$

Thus, (2.8) can be optimized via sampling x, x' from data and ν from the base distribution to estimate gradient as shown above (SGD) in every iteration. Next, we discuss the parametrization of h_ψ to satisfy Bochner's Theorem, and describe how to evaluate IKL kernel in practice.

Symmetric $\mathbb{P}_k(\omega)$ To result in real valued kernels, the spectral density has to be symmetric, where $\mathbb{P}_k(\omega) = \mathbb{P}_k(-\omega)$. Thus, we parametrize $h_\psi(\nu) = \texttt{sign}(\nu) \circ \tilde{h}_\psi(\texttt{abs}(\nu))$, where $\circ$ is the Hadamard product and $\tilde{h}_\psi$ can be any unconstrained function if the base distribution $\mathbb{P}(\nu)$ is symmetric (i.e. $\mathbb{P}(\nu) = \mathbb{P}(-\nu)$), such as standard normal distributions.

Kernel Evaluation Although there is usually no closed form for the kernel evaluation $k_\psi(x, x')$ in (2.7) with fairly complicated h_ψ, we can evaluate (approximate) $k_\psi(x, x')$ via sampling finite number of random Fourier features $\hat{k}_\psi(x, x') = \hat{\phi}_{h_\psi}(x)^\top \hat{\phi}_{h_\psi}(x')$, where $\hat{\phi}_{h_\psi}(x)^\top = [\phi(x; h_\psi(\nu_1)), \ldots, \phi(x; h_\psi(\nu_m))]$, and $\phi(x; \omega)$ is the evaluation on ω of the Fourier transformation $\phi(x)$ [88].

Next, we demonstrate two example applications covered by (2.8), where we can apply IKL, including kernel alignment and maximum mean discrepancy (MMD).

2.3.2 MMD GAN with IKL

The compositional kernel considered in Section 2.2 is

$$k_\varphi(x, x') = \exp(-\|f_\varphi(x) - f_\varphi(x)'\|^2). \tag{2.9}$$

By replacing the Gaussian kernel to IKL, we extend (2.9) to be $k_{\psi,\varphi} = k_\psi \circ f_\varphi$ with the form

$$k_{\psi,\varphi}(x, x') = \mathbb{E}_\nu \left[e^{i h_\psi(\nu)^\top (f_\varphi(x) - f_\varphi(x'))} \right]. \tag{2.10}$$

We then extend the MMD GAN objective to be

$$\min_\theta \max_{\psi,\varphi} M_{\psi,\varphi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta), \tag{2.11}$$

where $M_{\psi,\varphi}$ is the MMD distance (2.3) with the IKL kernel (2.10). Clearly, for a given φ, the maximization over ψ in (2.11) can be represented as (2.12) by letting $F_1(x, x') = 1$, $F_2(x, y) = -2$ and $F_3(y, y') = 1$. In what follows, we will use for convenience $k_{\psi,\varphi}$, k_ψ and k_φ to denote kernels defined in (2.10), (2.7) and (2.9) respectively.

2.3.3 Property of MMD GAN with IKL

As proven by Arjovsky and Bottou [3], some probability distances adopted by existing works (e.g. Goodfellow et al. [30]) are not *weak* (i.e. $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$ then $D(\mathbb{P}_n \| \mathbb{P}) \to 0$), which cannot provide better signal to train g_θ. Also, they usually suffer from discontinuity, hence it cannot be trained via gradient descent at certain points. We prove that $\max_{\psi,\varphi} M_{\psi,\varphi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta)$ is a continuous and differentiable objective in θ and *weak* under mild assumptions as Assumption 2 used by Theorem 3 and Theorem 4.

Algorithm 2 MMD GAN with IKL

Input: η the learning rate, B the batch size, n_c number of f, h updates per g update, m the number of basis, λ_{GP} the coefficient of gradient penalty, λ_h the coefficient of variance constraint.
Initial parameter θ for g, φ for f, ψ for h
Define $\mathcal{L}(\psi, \varphi) = M_{\psi,\varphi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta) - \lambda_{GP}(\|\nabla_{\hat{x}} f_\varphi(\hat{x})\|_2 - 1)^2 - \lambda_h(\mathbb{E}_\nu[\|h_\psi(\nu)\|^2] - u)^2$
while θ has not converged **do**
  for $t = 1, \dots, n_c$ **do**
    Sample $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X}),\ \{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z}),\ \{\nu_k\}_{k=1}^m \sim \mathbb{P}(\nu)$
    $(\psi, \varphi) \leftarrow \varphi + \eta \mathrm{Adam}\left((\psi, \varphi), \nabla_{\psi,\varphi}\mathcal{L}(\psi, \varphi)\right)$
  end for
  Sample $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X}),\ \{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z}),\ \{\nu_k\}_{k=1}^m \sim \mathbb{P}(\nu)$
  $\theta \leftarrow \theta - \eta \mathrm{Adam}(\theta, \nabla_\theta M_{\psi,\varphi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta))$
end while

Assumption 6. *$g_\theta(z)$ is locally Lipschitz and differentiable in θ; $f_\varphi(x)$ is Lipschitz in x and $\varphi \in \Phi$ is compact. $f_\varphi \circ g_\theta(z)$ is differentiable in θ and there are local Lipschitz constants, which is independent of φ, such that $\mathbb{E}_{z\sim\mathbb{P}_z}[L(\theta, z)] < +\infty$. The above assumptions are adopted by Arjovsky et al. [4]. Lastly, assume given any $\psi \in \Psi$, where Ψ is compact, $k_\psi(x, x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right]$ and $|k_\psi(x, x')| < \infty$ is differentiable and Lipschitz in (x, x') which has an upper bound L_k for Lipschitz constant of (x, x') given different ψ.*

Theorem 7. *Assume function g_θ and kernel $k_{\psi,\varphi}$ satisfy Assumption 6, $\max_{\psi,\varphi} M_{\psi,\varphi}$ is weak, that is, $\max_{\psi,\varphi} M_{\psi,\varphi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_n) \to 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P}_\mathcal{X}$. Also, $\max_{\psi,\varphi} M_{\psi,\varphi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta)$ is continuous everywhere and differentiable almost everywhere in θ.*

Lemma 8. *Assume $\mathcal{X}$ is bounded. Let $x, x' \in \mathcal{X}$, $k_\psi(x, x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right]$ is Lipschitz in (x, x') if $\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right] < \infty$, which is variance since $\mathbb{E}_\nu\left[h_\psi(\nu)\right] = 0$.*

We penalize $\lambda_h(\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right] - u)^2$ as an approximation of Lemma 8 in practice to ensure that assumptions in Theorem 7 are satisfied. The algorithm with IKL and gradient penalty [9] is shown in Algorithm 2.

2.3.4 Preliminary Results

We consider image and text generation tasks for quantitative evaluation. For image generation, we evaluate the inception score [92] on CIFAR-10 [50]. We use DCGAN [87] and expands the output of f_φ to be 16-dimensional as Bińkowski et al. [9]. For text generation, we consider a length-32 character-level generation task on Google Billion Words dataset. The evaluation is based on Jensen-Shannon divergence on empirical 4-gram probabilities (JS-4) of the generated sequence and the validation data as used by Gulrajani et al. [37], Heusel et al. [40], Mroueh et al. [76]. The model architecture follows Gulrajani et al. [37] in using ResNet with 1D convolutions. We train every algorithm $10,000$ iterations for comparison.

For MMD GAN with fixed base kernels, we consider the mixture of Gaussian ker-

nels $k(x, x') = \sum_q \exp(-\frac{\|x-x'\|^2}{2\sigma_q^2})$ [59] and the mixture of RQ kernels $k(x, x') = \sum_q (1 + \frac{\|x-x'\|^2}{2\alpha_q})^{-\alpha_q}$.

Lastly, for learning base kernels, we compare IKL with SM kernel [107] f_φ, which learns mixture of Gaussians to model kernel spectral density. It can also be treated as the *explicit generative model counter part* of the proposed IKL.

In both tasks, $\mathbb{P}(\nu)$, the base distribution of IKL, is a standard normal distribution and h_ψ is a 3-layer MLP with 32 hidden units for each layer. Similar to the aforementioned mixture kernels, we consider the mixture of IKL kernel with the variance constraints $\mathbb{E}[\|h_\psi(\nu)\|^2] = 1/\sigma_q$, where σ_q is the bandwidths for the mixture of Gaussian kernels. Note that if h_ψ is an identity map, we recover the mixture of Gaussian kernels. We fix λ_h to be 10 and resample $m = 1024$ random features for IKL in every iteration. For other settings, we follow [9].

Results and Discussion

We compare MMD GAN with the proposed IKL and different fixed kernels. We repeat the experiments 10 times and report the average result with standard error in Table 2.2. Note that for inception score the larger the better; while JS-4 the smaller the better. We also report WGAN-GP results as a reference. Sampled images on larger datasets are shown in Figure 2.2.

Method	Inception Scores (↑)	JS-4 (↓)
Gaussian	6.726 ± 0.021	0.381 ± 0.003
RQ	6.785 ± 0.031	0.463 ± 0.005
SM	6.746 ± 0.031	0.378 ± 0.003
IKL	$\mathbf{6.876 \pm 0.018}$	$\mathbf{0.372 \pm 0.002}$
WGAN-GP	6.539 ± 0.034	0.379 ± 0.002

Table 2.2: Inception scores and JS-4 divergece results.

Pre-defined Kernels Bińkowski et al. [9] show RQ kernels outperform Gaussian and energy distance kernels on image generation. Our empirical results agree with such finding: RQ kernels achieve 6.785 inception score while for Gaussian kernel it is 6.726, as shown in the left column of Table 2.2. In text generation, nonetheless, RQ kernels only achieve 0.463 JS-4 score[3] and are not on par with 0.381 acquired by Gaussian kernels, even though it is still slightly worse than WGAN-GP. These results imply *kernel selection is task-specific*. On the other hand, the proposed IKL learns kernels in a data-driven way, which results in the best performance in both tasks. In CIFAR-10, although Gaussian kernel is worse than RQ, IKL is still able to transforms $\mathbb{P}(\nu)$, which is Gaussian, into a powerful kernel, and

[3]For RQ kernels, we searched 10 possible hyperparameter settings and reported the best one in Appendix, to ensure the unsatisfactory performance is not caused by the improper parameters.

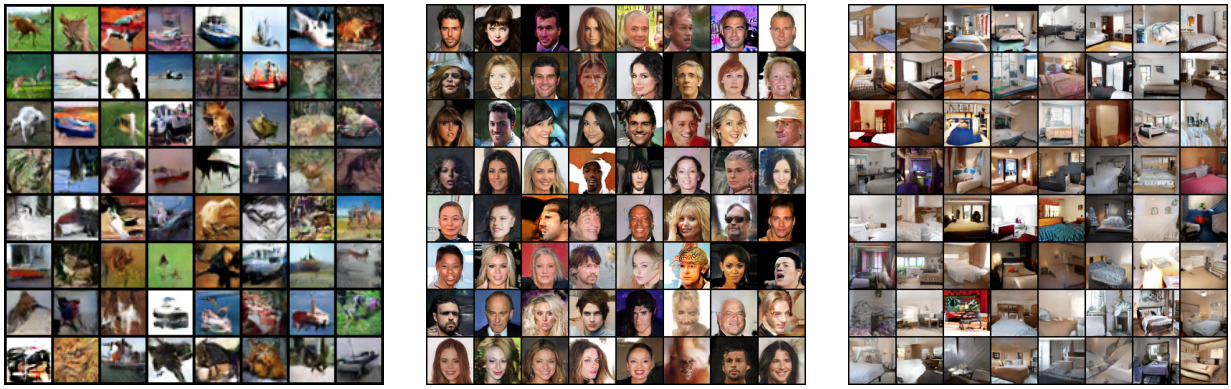


Figure 2.2: Samples generated by MMDGAN-IKL on CIFAR-10, CELEBA and LSUN dataset.

outperforms RQ on inception scores (6.876 v.s. 6.785). For text generation, from Table 2.2 and Figure 2.3, we observe that IKL can further boost Gaussian into better kernels with substantial improvement. Also, we note that the difference between IKL and pre-defined kernels in Table 2.2 is significant based on the t-test at 95% confidence level.

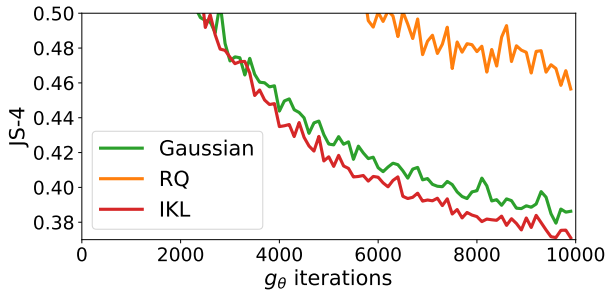


Figure 2.3: Convergence of MMD GANs with different kernels on text generation.

Learned Kernels The SM kernel [107], which learns the spectral density via mixture of Gaussians, does not significantly outperforms Gaussian kernel as shown in Table 2.2, since Li et al. [59] already uses equal-weighted mixture of Gaussian formulation. It suggests that proposed IKL can learn more complicated and effective spectral distributions than simple mixture models.

Study of Variance Constraints In Lemma 8, we prove bounding variance $\mathbb{E}[\|h_\psi(\nu)\|^2]$ guarantees k_ψ to be Lipschitz as required in Theorem 7. We investigate the importance of this constraint. In Figure 2.4, we show the training objective (MMD), $\mathbb{E}[\|h_\psi(\nu)\|^2]$ and the JS-4 divergence for training MMD GAN (IKL) without variance constraint, i.e. $\lambda_h = 0$. We could observe the variance keeps going up without constraints, which leads exploded MMD values. Also, when the exploration is getting severe, the JS-4 divergence starts increasing, which implies MMD cannot provide meaningful signal to g_θ. The study justifies the validity of Theorem 7 and Lemma 8.

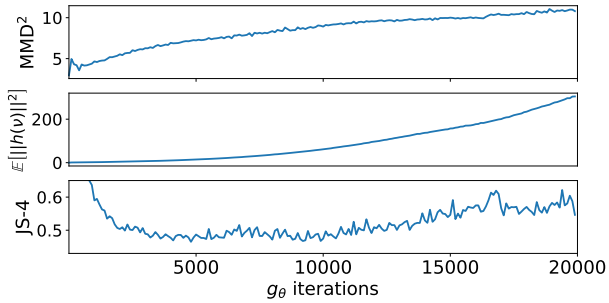


Figure 2.4: Learning MMD GAN (IKL) without the variance constraint on Google Billion Words datasets for text generation.

2.3.5 Proposed Work

Using IKL, we have seen that it further bring improvement over MMD GAN with Gaussian kernels without selecting kernels by hands. The idea of learning IKL via IGMs could also be applied to other standard applications with kernels if the problem can be formulated as

$$\begin{aligned} &\arg\max_{k\in\mathcal{K}} \sum_{i=1} \mathbb{E}_{x\sim\mathbb{P}_i, x'\sim\mathbb{Q}_i} \left[F_i(x,x')k(x,x')\right] = \\ &\arg\max_{k\in\mathcal{K}} \sum_{i=1} \mathbb{E}_{x\sim\mathbb{P}_i, x'\sim\mathbb{Q}_i} \left[F_i(x,x')\mathbb{E}_{\omega\sim\mathbb{P}_k}\left[e^{i\omega^\top(x-x')}\right]\right], \end{aligned} \tag{2.12}$$

where MMD is one of the special case of (2.12). We would like to investigate those and publish a paper by wrapping up these studies.

Chapter 3

Learning on Different Data Type

In Chapter 2, we discuss a generic algorithm for learning IGMs. One of the most widely studied data type in IGMs or GANs is image [3, 29, 37, 59, 75, 87]. For different data types, we should utilize their characteristic properties to design a better generative algorithm. Recently, some data types have also been explored by literatures, such as speech [54], text [114], video [101] and 3D voxels [109]. In this Chapter, we make a step towards learning IGMs on two different data types, including text [76] and point clouds [60].

3.1 Learning to Generate Text

In addition to using simple language model to generate text [70], Sampling sentences via continuous codes gains attention recently [10]. When moving to neural generators of *discrete sequences* generative adversarial networks theory and practice are still not very well understood. Maximum likelihood pre-training or augmentation, in conjunction with the use of reinforcement learning techniques were proposed in many recent works for training GAN for discrete sequences generation [41, 89, 114]. Other methods included using the Gumbel Softmax trick [52] and the use of auto-encoders to generate adversarially discrete sequences from a continuous space [119]. *End to end* training of GANs for discrete sequence generation is still an open problem [84]. Empirical successes of end to end training have been reported within the framework of WGAN-GP [37], using a proxy for the Wasserstein distance via a *pointwise gradient penalty* on the critic. In Chapter 2.3, we have seen using gradient penalty [37] in GAN formulation for text generation. In addition to the explanation based on optimality conditions of Wasserstein distance in dual form [37], we present a different explanation of the effectiveness of using gradient penalty.

3.1.1 Sobolev IPM

We will start by recalling some definitions on Sobolev Spaces. We assume in the following that $\mathcal{X}$ is compact and consider functions in the Sobolev space $W^{1,2}(\mathcal{X}, \mu)$:

$$W^{1,2}(\mathcal{X}, \mu) = \left\{ f : \mathcal{X} \to \mathbb{R}, \int_{\mathcal{X}} \|\nabla_x f(x)\|^2 \mu(x) dx < \infty \right\},$$

We restrict ourselves to functions in $W^{1,2}(\mathcal{X},\mu)$ vanishing at the boundary, and note this space $W_0^{1,2}(\mathcal{X},\mu)$. Note that in this case: $\|f\|_{W_0^{1,2}(\mathcal{X},\mu)} = \sqrt{\int_{\mathcal{X}} \|\nabla_x f(x)\|^2 \mu(x)dx}$ defines a semi-norm. We can similarly define a dot product in $W_0^{1,2}(\mathcal{X},\mu)$, for $f,g \in W_0^{1,2}(\mathcal{X},\mu)$:

$$\langle f,g\rangle_{W_0^{1,2}(\mathcal{X},\mu)} = \int_{\mathcal{X}} \langle \nabla_x f(x), \nabla_x g(x)\rangle_{\mathbb{R}^d} \mu(x)dx.$$

Hence we define the following Sobolev IPM, by restricting the critic of the mean discrepancy to the Sobolev unit ball :

$$\mathcal{S}_{\boldsymbol{\mu}}(\mathbb{P},\mathbb{Q}) = \sup_{f\in W_0^{1,2}, \|f\|_{W_0^{1,2}(\mathcal{X},\boldsymbol{\mu})}\leq 1} \left\{ \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) \right\} \tag{3.1}$$

Let $F_{\mathbb{P}}$ and $F_{\mathbb{Q}}$ be *the cumulative distribution functions* of $\mathbb{P}$ and $\mathbb{Q}$ respectively. We have:

$$\mathbb{P}(x) = \frac{\partial^d}{\partial x_1 \dots \partial x_d} F_{\mathbb{P}}(x), \tag{3.2}$$

We note

$$D = \frac{\partial^d}{\partial x_1 \dots \partial x_d}$$

and

$$D^{-i} = \frac{\partial^{d-1}}{\partial x_1 \dots \partial x_{i-1} \partial x_{i+1} \dots \partial x_d}, \text{ for } i = 1 \dots d.$$

D^{-i} computes the $d-1$ partial derivative excluding the variable i.

Our main result is presented in Theorem 9.

Theorem 9 (Sobolev IPM)**.** *Assume that $F_{\mathbb{P}}$, and $F_{\mathbb{Q}}$ and its d derivatives exist and are continuous: $F_{\mathbb{P}}$ and $F_{\mathbb{Q}} \in C^d(\mathcal{X})$. Define the differential operator D^- :*

$$D^- = (D^{-1}, \dots D^{-d}).$$

For $x = (x_1, \dots x_{i-1}, x_i, x_{i+1}, \dots x_d)$, let $x^{-i} = (x_1, \dots x_{i-1}, x_{i+1}, \dots x_d)$.

The Sobolev IPM given in Equation (3.1) *has the following equivalent forms:*

1. *Sobolev IPM as comparison of high order partial derivatives of CDFs. The Sobolev IPM has the following form:*

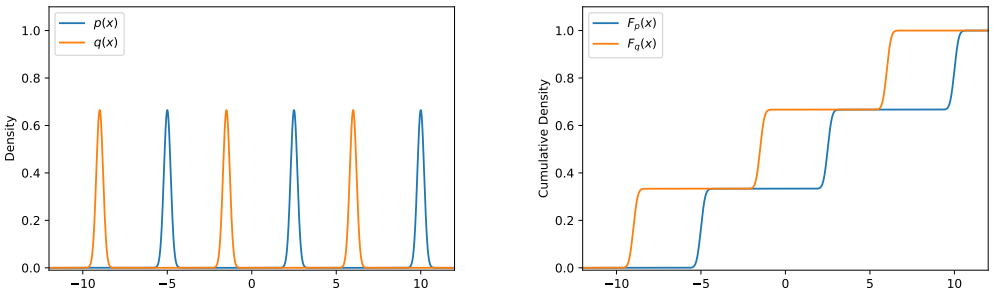
$$\mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q}) = \frac{1}{d}\sqrt{\int_{\mathcal{X}} \frac{\sum_{i=1}^d (D^{-i}F_{\mathbb{P}}(x) - D^{-i}F_{\mathbb{Q}}(x))^2}{\mu(x)} dx}.$$

2. *Sobolev IPM as comparison of weighted (coordinate-wise) conditional CDFs. The Sobolev IPM can be written in the following equivalent form:*

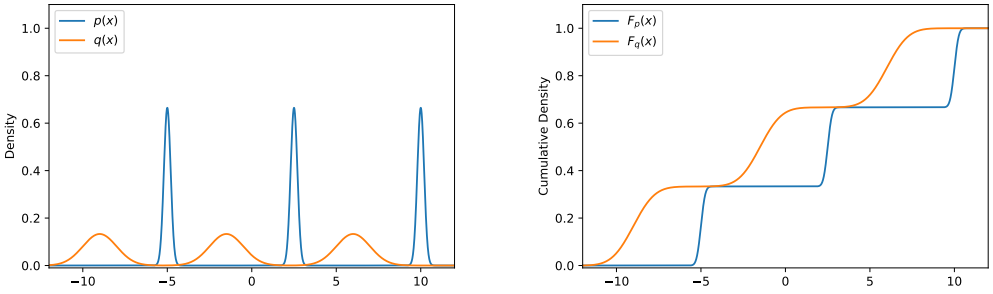
$$\mathcal{S}_{\mu}^2(\mathbb{P},\mathbb{Q}) = \frac{1}{d^2}\mathbb{E}_{x\sim\mu} \sum_{i=1}^d \left(\frac{\mathbb{P}_{X^{-i}}(x^{-i}) F_{\mathbb{P}_{[X_i|X^{-i}=x^{-i}]}}(x_i) - \mathbb{Q}_{X^{-i}}(x^{-i}) F_{\mathbb{Q}_{[X_i|X^{-i}=x^{-i}]}}(x_i)}{\mu(x)} \right)^2.$$

3. *The optimal critic f^* satisfies the following identity:*

$$\nabla_x f^*(x) = \frac{1}{d\mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q})} \frac{D^- F_{\mathbb{Q}}(x) - D^- F_{\mathbb{P}}(x)}{\mu(x)}, \mu - \text{ almost surely.} \tag{3.3}$$



(a) Smoothed discrete densities: PDF versus CDF of smoothed discrete densities with non overlapping supports.



(b) Smoothed Discrete and Continuous densities: PDF versus CDF of a smoothed discrete density and a continuous density with non overlapping supports.

Figure 3.1: In the GAN context for example in text generation, we have to match a (smoothed) discrete real distribution and a continuous generator. In this case, the CDF matching enabled by Sobolev IPM gives non zero discrepancy between a (smoothed) discrete and a continuous density even if the densities have disjoint supports. This ensures non vanishing gradients of the critic.

Sobolev IPM / Cramér Distance and 1-Wasserstein in one Dimension. In one dimension, Sobolev IPM is the Cramér Distance (for μ uniform on $\mathcal{X}$, we note this $\mu := 1$). While Sobolev IPM in one dimension measures the discrepancy between CDFs, the one dimensional p-Wasserstein distance measures the discrepancy between inverse CDFs:

$$\mathcal{S}^2_{\mu:=1}(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} (F_{\mathbb{P}}(x) - F_{\mathbb{Q}}(x))^2 dx \text{ versus } W_p^p(\mathbb{P},\mathbb{Q}) = \int_0^1 |F^{-1}_{\mathbb{P}}(u) - F^{-1}_{\mathbb{Q}}(u)|^p du,$$

Consider for instance two point masses $\mathbb{P} = \delta_{a_1}$ and $\mathbb{Q} = \delta_{a_2}$ with $a_1, a_2 \in \mathbb{R}$. The rational behind using Wasserstein distance for GAN training is that for far distributions Wasserstein distance provides some signal [4]. In this case, it is easy to see that $W_1^1(\mathbb{P},\mathbb{Q}) = \mathcal{S}^2_{\mu:=1} = |a_1 - a_2|$, while $\mathcal{F}^2_{\mu:=1}(\mathbb{P},\mathbb{Q}) = 2$. As we see from this simple example that **CDF** comparison is more suitable than PDF (e.g. f-divergence) for comparing distributions on **discrete spaces**.

3.1.2 Sobolev GAN

Now we turn to the problem of learning IGMs with Sobolev IPM. As defined at 2, given the "real distribution" $\mathbb{P}_r \in \mathcal{P}(\mathcal{X})$, our goal is to learn a generator $g_\theta : \mathcal{Z} \to \mathcal{X}$, such that for $z \sim \mathbb{P}_z$, the distribution of $g_\theta(z)$ is close to the real data distribution $\mathbb{P}_r$, where $\mathbb{P}_z$ is a fixed distribution on $\mathcal{Z}$ (for instance $z \sim \mathcal{N}(0, I)$). We note $\mathbb{Q}_\theta$ for the "fake distribution" of $g_\theta(z), z \sim \mathbb{P}_z$. Consider $\{x_i, i = 1 \dots N\} \sim \mathbb{P}_r$, $\{z_i, i = 1 \dots N\} \sim \mathcal{N}(0, I)$, and $\{\tilde{x}_i, i = 1 \dots N\} \sim \mu$. We consider these choices for μ:

1. $\mu = \frac{\mathbb{P}_r + \mathbb{Q}_\theta}{2}$ i.e $\tilde{x} \sim \mathbb{P}_r$ or $\tilde{x} = g_\theta(z), z \sim \mathbb{P}_z$ with equal probability $\frac{1}{2}$.
2. μ_{GP} is the implicit distribution defined by the interpolation lines between $\mathbb{P}_r$ and $\mathbb{Q}_\theta$ as in [37] i.e : $\tilde{x} = ux + (1-u)y, x \sim \mathbb{P}_r, y = g_\theta(z), z \sim \mathbb{P}_z$ and $u \sim \text{Unif}[0, 1]$.

Sobolev GAN can be written as follows:

$$\min_{g_\theta} \sup_{f_p, \frac{1}{N}\sum_{i=1}^N \|\nabla_x f_p(\tilde{x}_i)\|^2 = 1} \hat{\mathcal{E}}(f_p, g_\theta) = \frac{1}{N}\sum_{i=1}^N f_p(x_i) - \frac{1}{N}\sum_{i=1}^N f_p(g_\theta(z_i))$$

For any choice of the parametric function class $\mathcal{H}_p$, note the constraint by $\hat{\Omega}_S(f_p, g_\theta) = \frac{1}{N}\sum_{i=1}^N \|\nabla_x f_p(\tilde{x}_i)\|^2$. For example if $\mu = \frac{\mathbb{P}_r + \mathbb{Q}_\theta}{2}$, $\hat{\Omega}_S(f_p, g_\theta) = \frac{1}{2N}\sum_{i=1}^N \|\nabla_x f_p(x_i)\|^2 + \frac{1}{2N}\sum_{i=1}^N \|\nabla_x f_p(g_\theta(z_i))\|^2$. Note that, since the optimal theoretical critic is achieved on the sphere, we impose a sphere constraint rather than a ball constraint. Similar to [75] we define the Augmented Lagrangian corresponding to Sobolev GAN objective and constraint

$$\mathcal{L}_S(p, \theta, \lambda) = \hat{\mathcal{E}}(f_p, g_\theta) + \lambda(1 - \hat{\Omega}_S(f_p, g_\theta)) - \frac{\rho}{2}(\hat{\Omega}_S(f_p, g_\theta) - 1)^2 \tag{3.4}$$

where λ is the Lagrange multiplier and $\rho > 0$ is the quadratic penalty weight. We alternate between optimizing the critic and the generator. We impose the constraint when training the critic only. Given θ, we solve $\max_p \min_\lambda \mathcal{L}_S(p, \theta, \lambda)$, for training the critic. Then given the critic parameters p we optimize the generator weights θ to minimize the objective $\min_\theta \hat{\mathcal{E}}(f_p, g_\theta)$. See Algorithm 3.

Relation to WGAN-GP. WGAN-GP can be written as follows:

$$\min_{g_\theta} \sup_{f, \|\nabla_x f_p(\tilde{x}_i)\| = 1, \tilde{x}_i \sim \mu_{GP}} \hat{\mathcal{E}}(f_p, g_\theta) = \frac{1}{N}\sum_{i=1}^N f_p(x_i) - \frac{1}{N}\sum_{i=1}^N f_p(g_\theta(z_i))$$

The main difference between WGAN-GP and our setting, is that WGAN-GP enforces *pointwise constraints* on points drawn from $\mu = \mu_{GP}$ via a point-wise quadratic penalty $(\hat{\mathcal{E}}(f_p, g_\theta) - \lambda\sum_{i=1}^N (1 - \|\nabla_x f(\tilde{x}_i)\|)^2)$ while we enforce that constraint on average as a Sobolev norm, allowing us the coordinate weighted conditional CDF interpretation of the IPM.

3.1.3 Preliminary Results

In this Section, we present an empirical study of Sobolev GAN in character level text generation. In addition to WGAN-GP [37], we also compare Fisher GAN, which matches

Algorithm 3 Sobolev GAN

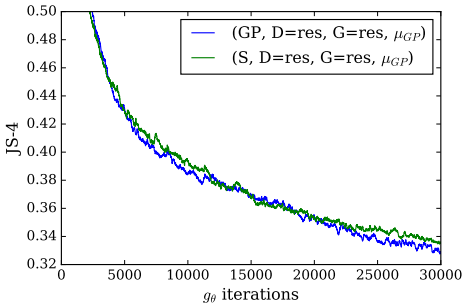
Input: ρ penalty weight, η Learning rate, n_c number of iterations for training the critic, N batch size
Initialize $p, \theta, \lambda = 0$
repeat
  for $j = 1$ **to** n_c **do**
    Sample a minibatch $x_i, i = 1 \dots N, x_i \sim \mathbb{P}_r$
    Sample a minibatch $z_i, i = 1 \dots N, z_i \sim \mathbb{P}_z$
    $(g_p, g_\lambda) \leftarrow (\nabla_p \mathcal{L}_S, \nabla_\lambda \mathcal{L}_S)(p, \theta, \lambda)$
    $p \leftarrow p + \eta$ ADAM (p, g_p)

    $\lambda \leftarrow \lambda - \rho g_\lambda$ ▷ SGD rule on λ with learning rate ρ
  end for
  Sample $z_i, i = 1 \dots N, z_i \sim \mathbb{P}_z$
  $d_\theta \leftarrow \nabla_\theta \hat{\mathcal{E}}(f_p, g_\theta) = -\nabla_\theta \frac{1}{N} \sum_{i=1}^N f_p(g_\theta(z_i))$
  $\theta \leftarrow \theta - \eta$ ADAM (θ, d_θ)
until θ converges

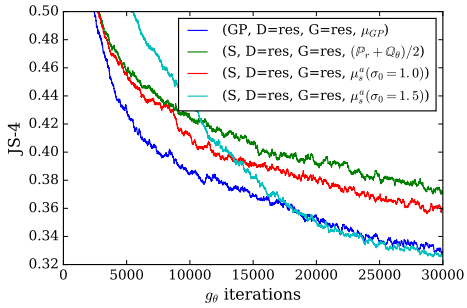
distributions via PDF Our empirical study on end to end training of character-level GAN for text generation is articulated on four dimensions **(loss, critic, generator, $\boldsymbol{\mu}$)**. matching. (1) the loss used (**GP**: WGAN-GP [37], **S**: Sobolev or **F**: Fisher) (2) the architecture of the critic (ResNets or RNN) (3) the architecture of the generator (ResNets or RNN or RNN with curriculum learning) (4) the sampling distribution $\boldsymbol{\mu}$ in the constraint. We train a character-level GAN on Google Billion Word dataset and follow the same experimental setup used in [37]. The generated sequence length is 32 and the evaluation is based on Jensen-Shannon divergence on empirical 4-gram probabilities (JS-4) of validation data and generated data. JS-4 may not be an ideal evaluation criteria, but it is a reasonable metric for current character-level GAN results, which is still far from generating meaningful sentences.

Annealed Smoothing of discrete $\mathbb{P}_r$ in the constraint $\boldsymbol{\mu}$. Since the generator distribution will always be defined on a continuous space, we can replace the discrete "real" distribution $\mathbb{P}_r$ with a smoothed version (Gaussian kernel smoothing) $\mathbb{P}_r \star \mathcal{N}(0, \sigma^2 I_d)$. This corresponds to doing the following sampling for $\mathbb{P}_r : x + \xi, x \sim \mathbb{P}_r$, and $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$. Note that we only inject noise to the "real" distribution with the goal of smoothing the support of the discrete distribution, as opposed to instance noise on both "real" and "fake" to stabilize the training, as introduced in [3, 45]. As it is common in optimization by continuation [72], we also anneal the noise level σ as the training progresses on a linear schedule.

Sobolev GAN versus WGAN-GP with ResNets. In this setting, we compare (WGAN-GP,G=Resnet,D=Resnet,$\mu = \mu_{GP}$) to (Sobolev,G=Resnet,D=Resnet,μ) where μ is one of: (1) μ_{GP}, (2) the noise smoothed $\mu_s(\sigma) = \frac{\mathbb{P}_r \star \mathcal{N}(0, \sigma^2 I_d) + \mathbb{Q}_\theta}{2}$ or (3) noise smoothed with annealing $\mu_s^a(\sigma_0)$ with σ_0 the initial noise level. We use the same architectures of Resnet with 1D convolution for the critic and the generator as in [37] (4 resnet blocks with



(a) Comparing Sobolev with μ_{GP} and WGAN-GP. The JS-4 are 0.3363 and 0.3302 respectively.



(b) Comparing Sobolev with different μ dominant measures and WGAN-GP. The JS-4 of $\mu_s^a(\sigma_0 = 1.5)$ is 0.3268.

Figure 3.2: Result of Sobolev GAN for various dominating measure μ, for resnets as architectures of the critic and the generator.

hidden layer size of 512). In order to implement the noise smoothing we transform the data into one-hot vectors. Each one hot vector x is transformed to a probability vector p with 0.9 replacing the one and $0.1/(dict_{size} - 1)$ replacing the zero. We then sample ϵ from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, and use softmax to normalize $\log p + \epsilon$. We use algorithm 3 for Sobolev GAN and fix the learning rate to 10^{-4} and ρ to 10^{-5}. The noise level σ was annealed following a linear schedule starting from an initial noise level σ_0 (at iteration i, $\sigma_i = \sigma_0(1 - \frac{i}{Maxiter})$, Maxiter=30K). For WGAN-GP we used the open source implementation with the penalty $\lambda = 10$ as in [37]. Results are given in Figure 3.2a for the JS-4 evaluation of both WGAN-GP and Sobolev GAN for $\mu = \mu_{GP}$. In Figure 3.2b we show the JS-4 evaluation of Sobolev GAN with the annealed noise smoothing $\mu_s^a(\sigma_0)$, for various values of the initial noise level σ_0. We see that the training succeeds in both cases. Sobolev GAN achieves slightly better results than WGAN-GP for the annealing that starts with high noise level $\sigma_0 = 1.5$. We note that without smoothing and annealing i.e using $\mu = \frac{\mathbb{P}_r+\mathbb{Q}_\theta}{2}$, Sobolev GAN is behind. Annealed smoothing of $\mathbb{P}_r$, helps the training as the real distribution is slowly going from a continuous distribution to a discrete distribution.

Fisher GAN Curriculum Conditioning versus Sobolev GAN: Explicit versus Implicit conditioning. We analyze how Fisher GAN behaves under different architectures of generators and critics. We first fix the generator to be ResNet. We study 3 different architectures of critics: ResNet, GRU (we follow the experimental setup from [84]), and hybrid ResNet+GRU [90]. We notice that RNN is unstable, we need to clip the gradient values of critics in $[-0.5, 0.5]$, and the gradient of the Lagrange multiplier λ_F to $[-10^4, 10^4]$. We fix $\rho_F = 10^{-7}$ and we use $\mu = \mu_{GP}$. We search the value for the learning rate in $[10^{-5}, 10^{-4}]$. We see that for $\mu = \mu_{GP}$ and $G =$ Resnet for various critic architectures, Fisher GAN fails at the task of text generation (Figure 3.3 a-c). Nevertheless, when using RNN critics (Fig 3.3 b, c) a marginal improvement happens over the fully collapsed state when using a resnet critic (Fig 3.3 a). We hypothesize that RNN critics enable some conditioning and factoring of the distribution, which is lacking in Fisher IPM. Finally Figure 3.3 (d) shows the result of training with recurrent generator and critic.

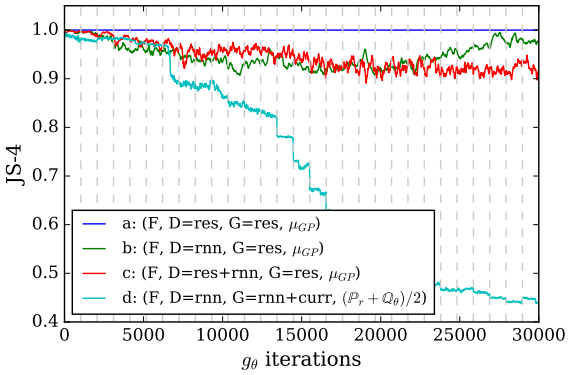


Figure 3.3: Fisher GAN with different architectures for critics: (a-c) and (d) Fisher GAN with recurrent generator and critic, trained on a curriculum conditioning.

We follow [84] in terms of GRU architecture, but differ by using Fisher GAN rather than WGAN-GP. We use $\mu = \frac{\mathbb{P}_r+\mathbb{Q}_\theta}{2}$ i.e. without annealed noise smoothing. We train (F, D=RNN,G=RNN,$\frac{\mathbb{P}_r+\mathbb{Q}_\theta}{2}$) using curriculum conditioning of the generator for all lengths ℓ as done in [84]: the generator is conditioned on $32-\ell$ characters and predicts the ℓ remaining characters. We increment $\ell = 1$ to 32 on a regular schedule (every 15k updates). JS-4 is only computed when $\ell > 4$. We see in Figure 3.3 that under curriculum conditioning with recurrent critics and generators, the training of Fisher GAN succeeds and reaches similar levels of Sobolev GAN (and WGAN-GP). Note that the need of this *explicit brute force conditioning* for Fisher GAN, highlights the *implicit conditioning* induced by Sobolev GAN via the gradient regularizer, without the need for curriculum conditioning.

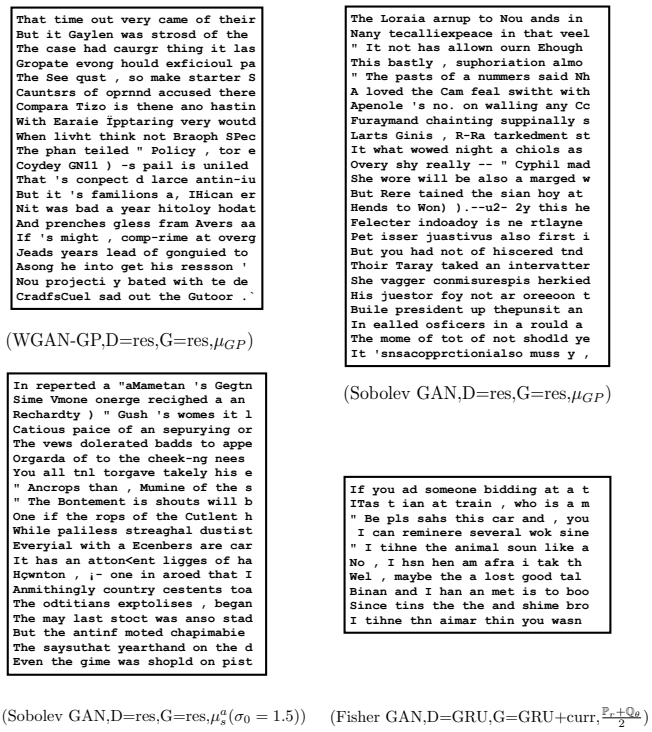


Figure 3.4: Text samples from various GANs considered in this paper.

3.1.4 Proposed Work

We investigated the effectiveness of using gradient penalty of training GAN. However, the qualitative results shown in Figure 3.4 presents significantly worse results than standard language model with word-by-word (character-by-character) supervision. In the future,

we plan to study a distance measure for comparing text with more structured information, such as grammar structure [20] and optima transport [2], than simple leave-one-out conditioning.

3.2 Learning to Generate Point Clouds

Recently, capturing 3D information is garnering attention. There are many different data types for 3D information, such as CAD, 3D meshes and point clouds. 3D point clouds are getting popular since these store more information than 2D images and sensors capable of collecting point clouds have become more accessible. These include Lidar on self-driving cars, Kinect for Xbox and face identification sensor on phones. Compared to other formats, point clouds can be easily represented as a set of points, which has several advantages, such as permutation invariance. The algorithms which can effectively learn from this type of data is an emerging field [24, 46, 85, 86, 115]. However, compared to supervised learning, unsupervised generative models for 3D data are still under explored [1, 80].

Extending existing GAN frameworks to handle 3D point clouds, or more generally set data, is not straightforward. In this section, we begin by formally defining the problem and discussing the difficulty of the problem. Circumventing the challenges, we propose a deep generative adversarial network (PC-GAN) with a hierarchical sampling and inference network for point clouds. The proposed architecture learns a stochastic procedure which can generate new point clouds as well as draw samples from point clouds without explicitly modeling the underlying density function.

3.2.1 Problem Definition and Difficulty

A point cloud for an object θ is a *set* of n low dimensional vectors $X = \{x_1, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, where d is usually 3 and n can be infinite. M different objects can be described as a collection of point clouds $X^{(1)}, ..., X^{(M)}$. A generative model for sets should be able to: (1) Sample entirely new sets according to $p(X)$, and (2) Sample more points for a given set, i.e. $x \sim p(x|X)$.

Based on De-Finetti theorem, we could factor the probability with some suitably defined θ, such as object representation of point clouds, as $p(X) = \int_\theta \prod_{i=1}^n p(x_i|\theta)p(\theta)d\theta$. In this view, the factoring can be understood as follows: Given an object, θ, the points x_i in the point cloud can be considered as i.i.d. samples from $p(x|\theta)$, an unknown latent distribution representing object θ. Joint likelihood can be expressed as:

$$p(X, \theta) = \underbrace{p(\theta)}_{\text{object}} \quad \underbrace{\prod_{i=1}^{n} p(x_i|\theta)}_{\text{points for object}} \tag{3.5}$$

One approach can be used to model the distribution of the point cloud set together, i.e., $\{\{x_i^{(1)}\}_{i=1}^n, \ldots, \{x_i^{(m)}\}_{i=1}^n\}$. In this setting, a naíve application of traditional GAN is possible through treating the point cloud as finite dimensional vector by fixing number and order

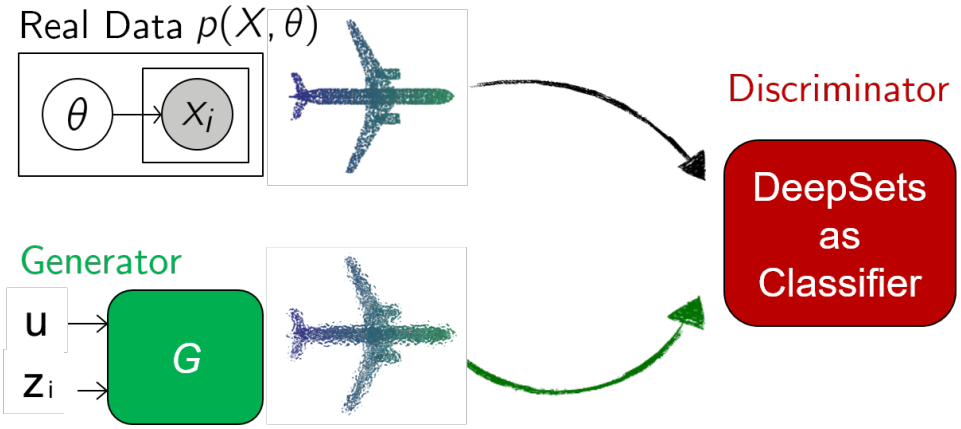


Figure 3.5: Natural extension of GAN to handle set data does not work.

of the points (reducing the problem to instances in $\mathbb{R}^{n\times 3}$) with DeepSets [115] classifier as the discriminator to distinguish real sets from fake sets. However, it would not work because the IPM guarantees behind the traditional GAN no longer hold (e.g. in case of Arjovsky et al. [4], nor are 1-Lipschitz functions over sets well-defined). The probabilistic divergence approximated by a DeepSets classifier is not clear and might be ill-defined. Counter examples for breaking IPM guarantees can be easily found as we show next.

Counter Example Consider a simple GAN [29] with a DeepSets classifier as the discriminator. In order to generate coherent sets of variable size, we consider a generator G having two noise sources: u and z_i. To generate a set, u is sampled once and z_i is sampled for $i = 1, 2, ..., n$ to produce n points in the generated set. Intuitively, fixing the first noise source u selects a set and ensures the points generated by repeated sampling of z_i are coherent and belong to same set. The setup is depicted in Figure 3.5. In this setup, the GAN minimax problem would be:

$$\min_G \max_D \mathop{\mathbb{E}}_{\substack{\theta\sim p(\theta)\\ x_i\sim p(x_i|\theta)}} \left[\log D\left(\{x_i\}\right)\right] + \mathop{\mathbb{E}}_{\substack{u\sim p(u)\\ z_i\sim p(z_i)}} \left[\log\left(1 - D\left(\{G(u, z_i)\}\right)\right)\right] \tag{3.6}$$

Now consider the case, when there exist an 'oracle' mapping T which maps each sample point deterministically to the object it originated from, i.e. $\exists T : T(\{x_i\}) = \theta$. A valid example is when different θ leads to conditional distribution $p(x|\theta)$ with non-overlapping support. Let $D = D' \circ T$ and G ignores z then optimization becomes

$$\begin{aligned}
&\min_G \max_{D'} \mathop{\mathbb{E}}_{\substack{\theta\sim p(\theta)\\ x_i\sim p(x_i|\theta)}} \left[\log D'\left(T\left(\{x_i\}\right)\right)\right] + \mathop{\mathbb{E}}_{\substack{u\sim p(u)\\ z_i\sim p(z_i)}} \left[\log\left(1 - D'\left(T\left(\{G(u, z_i)\}\right)\right)\right)\right] \\
\Rightarrow &\min_G \max_{D'} \mathop{\mathbb{E}}_{\substack{\theta\sim p(\theta)\\ x_i\sim p(x_i|\theta)}} \left[\log D'\left(\theta\right)\right] + \mathop{\mathbb{E}}_{\substack{u\sim p(u)\\ z_i\sim p(z_i)}} \left[\log\left(1 - D'\left(T\left(\{G(u)\}\right)\right)\right)\right] \\
\Rightarrow &\min_G \max_{D'} \mathop{\mathbb{E}}_{\theta\sim p(\theta)} \left[\log D'\left(\theta\right)\right] + \mathop{\mathbb{E}}_{u\sim p(u)} \left[\log\left(1 - D'\left(T\left(\{G(u)\}\right)\right)\right)\right]
\end{aligned} \tag{3.7}$$

Thus, we can achieve the lower bound $-\log(4)$ by only matching the $p(\theta)$ component, while the conditional $p(x|\theta)$ is allowed to remain arbitrary. So simply using DeepSets classifier *without any constraints* in simple GAN in order to handle sets does not lead to a valid generative model.

3.2.2 Point Cloud GAN

Although directly learning point cloud generation under GAN formulation is difficult as described in Section 3.2.1, given θ, learning $p(x|\theta)$ is reduced to learning a 3 dimensional distribution. Given two point clouds, one popular heuristic distance is chamfer distance [1]. On the other hand, if we treat each point cloud as a *3 dimensional distribution*, we can adopt a broader classes of probabilistic divergence $D(\mathbb{P}\|\mathbb{G})$ for training point cloud generator G_x. Instead of learning explicit densities [22, 44, 96], we could learn IGMs for point clouds via GAN-like adversarial loss. Formally, given a θ, we train a generator $G_x(z, \theta)$ such that $x = G_x(z, \theta)$, where $z \sim p(z)$, follows $\mathbb{G}$ by optimizing a (peudo) probabilistic divergence $D(\mathbb{P}\|\mathbb{G})$ between the distribution $\mathbb{G}$ of $G_x(z, \theta)$ and $p(x|\theta)$, which is denoted as $\mathbb{P}$. The full objective can be written as $\mathbb{E}_{\theta\sim p(\theta)}\left[\min_{G_x} D(\mathbb{P}\|\mathbb{G})\right]$.

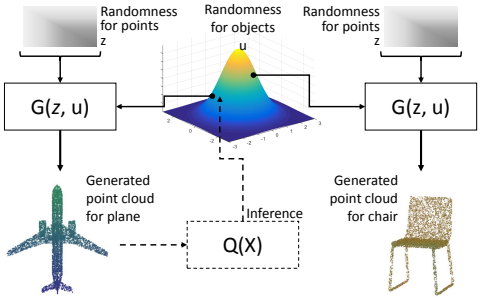


Figure 3.6: Overview of PC-GAN.

Inference Although GANs have been extended to learn conditional distributions [43, 71], they require conditioning variables to be observed, such as the one-hot label or a given image. Our θ, instead, is an unobserved latent variable for modeling different objects, which we need to infer during training. The proposed algorithm has to concurrently learn the inference network $Q(X) \approx \theta$ while we learn $p(x|\theta)$. Since X is a set of points, we can adopt Qi et al. [85], Zaheer et al. [115] for Q.

Hierarchical Sampling After training G_x and Q, we use trained Q to collect inferred $Q(X)$ and train the generator $G_\theta(u) \sim p(\theta)$ for higher hierarchical sampling, where $u \sim p(u)$ is the other noise source independent of z. In addition to layer-wise training, a joint training could further boost performance. The full generative process for sampling one point cloud could be represented as

$$\{x_i\}_{i=1}^n = \{G(z_i, u)\}_{i=1}^n = \{G_x(z_i, G_\theta(u))\}_{i=1}^n, \text{ where } z_1, \ldots, z_n \sim p(z), \text{ and } u \sim p(u).$$

We call the proposed algorithm for point cloud generation as *PC-GAN* as shown in Figure 3.6.

Different Divergences for Matching Point Clouds

To train the generator G_x using a GAN-like objective for point clouds, we need a discriminator $f(\cdot)$ to distinguishes generated samples and true samples conditioned on θ.

Combining with the inference network $Q(X)$ discussed aforementioned, the objective with IPM-based GANs can be written as

$$\mathbb{E}_{\theta\sim p(\theta)}\Bigg[\underbrace{\min_{G_x,Q}\max_{f\in\Omega_f}\mathbb{E}_{x\sim p(X|\theta)}\left[f(x)\right]-\mathbb{E}_{z\sim p(z),X\sim p(X|\theta)}\left[f(G_x(z,Q(X)))\right]}_{D(\mathbb{P}\|\mathbb{G})}\Bigg], \tag{3.8}$$

where Ω_f is the constraint for different probabilistic distances, such as 1-Lipschitz [4], L^2 ball [75] or Sobolev ball [76].

3.2.3 Tighter Solutions via Sandwiching

In our setting, each point x_i in the point cloud can be considered to correspond to single images when we train GANs over images. An example is illustrated in Figure 3.7 where samples from MMD-GAN [59] trained on CelebA consists of both good and bad faces. In case of images, when quality is evaluated, it primarily focuses on coherence individual images and the few bad ones are usually left out. Whereas in case of point cloud, to get representation of an object we need many sampled points together and presence of outlier points degrades the quality of the object. Thus, when training a generative model for point cloud, we need to ensure a much lower distance $D(\mathbb{P}\|\mathbb{G})$ between true distribution $\mathbb{P}$ and generator distribution $\mathbb{G}$ than would be needed in case of images.

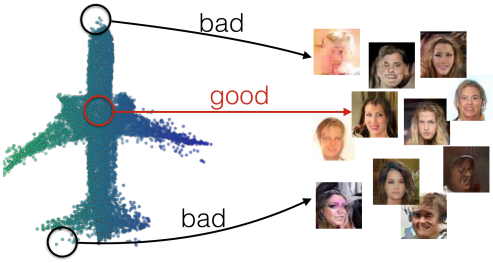


Figure 3.7: Connection between good/bad points and faces generated from a GAN.

We begin by noting that the popular Wasserstein GAN [4], aims to optimize G by $\min w(\mathbb{P},\mathbb{G})$, where $w(\mathbb{P},\mathbb{G})$ is the Wasserstein distance $w(\mathbb{P},\mathbb{G})$ between the truth $\mathbb{P}$ and generated distribution $\mathbb{G}$ of G. Many GAN works (e.g. Arjovsky et al. [4]) approximate $w(\mathbb{P},\mathbb{G})$ in dual form (a maximization problem), such as (3.8), by neural networks. The resulting estimate $W_L(\mathbb{P},\mathbb{G})$ is a lower bound of the true Wasserstein distance, as neural networks can only recover a subset of 1-Lipschitz functions [5] required in the dual form. However, finding a lower bound $W_L(\mathbb{P},\mathbb{G})$ for $w(\mathbb{P},\mathbb{G})$ may not be an ideal surrogate for solving a minimization problem $\min w(\mathbb{P},\mathbb{G})$. In optimal transport literature, Wassertein distance is usually estimated by approximate matching cost, $W_U(\mathbb{P},\mathbb{G})$, which gives us an upper bound of the true Wasserstein distance.

We propose to combine, in general, a lower bound W_L and upper bound estimate W_U by sandwiching the solution between the two, i.e. we solve the following minimization problem:

$$\min_G \quad W_U(\mathbb{P},\mathbb{G}) \qquad \text{s.t.} \quad W_U(\mathbb{P},\mathbb{G})-W_L(\mathbb{P},\mathbb{G})<\lambda \tag{3.9}$$

The problem can be simplified and solved using method of Lagrange multipliers as follows:

$$\min_G W_s(\mathbb{P},\mathbb{G}) := (1-s)W_U(\mathbb{P},\mathbb{G})+sW_L(\mathbb{P},\mathbb{G}) \tag{3.10}$$

By solving the new *sandwiched problem* (3.10), we show that under certain conditions we obtain a better estimate of Wasserstein distance in the following lemma:

Lemma 10. *Suppose we have two approximators to Wasserstein distance: an upper bound W_U and a lower W_L, such that $\forall \mathbb{P}, \mathbb{G} : (1+\epsilon_1)w(\mathbb{P},\mathbb{G}) \leq W_U(\mathbb{P},\mathbb{G}) \leq (1+\epsilon_2)w(\mathbb{P},\mathbb{G})$ and $\forall P, G : (1-\epsilon_2)w(\mathbb{P},\mathbb{G}) \leq W_L(\mathbb{P},\mathbb{G}) \leq (1-\epsilon 1)w(\mathbb{P},\mathbb{G})$ respectively, for some $\epsilon_2 > \epsilon_1 > 0$ and $\epsilon_1 > \epsilon_2/3$. Then, using the sandwiched estimator W_s from (3.10), we can achieve tighter estimate of the Wasserstein distance than using either one estimator, i.e.*

$$\exists s : |W_s(\mathbb{P},\mathbb{G}) - w(\mathbb{P},\mathbb{G})| < \min\{|W_U(\mathbb{P},\mathbb{G}) - w(\mathbb{P},\mathbb{G})|, |W_L(\mathbb{P},\mathbb{G}) - w(\mathbb{P},\mathbb{G})|\} \quad (3.11)$$

Upper Implementation

The primal form of Wasserstein distance is defined as

$$w(\mathbb{P},\mathbb{G}) = \inf_{\gamma \in \Gamma(\mathbb{P},\mathbb{G})} \int \|x-y\|_1 d\gamma(x,y),$$

where γ is the *coupling* of P and G. The Wasserstein distance is also known as optimal transport (OT) or earth moving distance (EMD). As the name suggests, when $w(\mathbb{P},\mathbb{G})$ is estimated with finite number of samples $X = x_1, \ldots, x_n$ and $Y = y_1, \ldots, y_n$, we find the one-to-one matching between X and Y such that the total pairwise distance is minimal. The resulting minimal total (average) pairwise distance is $w(X,Y)$. In practice, finding the exact matching efficiently is non-trivial and still an open research problem [82]. Instead, we consider an approximation provided by Bertsekas [8]. It is an iterative algorithm where each iteration operates like an auction whereby unassigned points $x \in X$ bid simultaneously for closest points $y \in Y$, thereby raising their prices. Once all bids are in, points are awarded to the highest bidder. The crux of the algorithm lies in designing a non-greedy bidding strategy. One can see by construction the algorithm is embarrassingly parallelizable, which is favourable for GPU implementation. One can show that algorithm terminates with a valid matching and the resulting matching cost $W_U(X,Y)$ is an ϵ-approximation of $w(X,Y)$. Thus, the estimate can serve as an upper bound, i.e.

$$w(X,Y) \leq W_U(X,Y) \leq (1+\epsilon)w(X,Y), \quad (3.12)$$

We remark estimating Wasserstein distance $w(\mathbb{P},\mathbb{G})$ with finite sample via primal form is only favorable in low dimensional data, such as point clouds. The error between $w(\mathbb{P},\mathbb{G})$ and $w(X,Y)$ is $O(1/n^{1/d})$, where d is data dimension [106]. Therefore, for high dimensional data, such as images, we cannot accurately estimate Wasserstein distance in primal and its upper bound with a small minibatch.

Finding a modified primal form with low sample complexity is also an open research problem [14, 27], and combining those into the proposed sandwiching objective for high dimensional data is left for future works.

Lower Implementation

The dual form of Wasserstein distance is defined as

$$w(\mathbb{P},\mathbb{G}) = \sup_{f \in \mathcal{L}_1} \mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim G} f(x), \quad (3.13)$$

where $\mathcal{L}_k$ is the set of k-Lipschitz functions whose Lipschitz constant is no larger than k. In practice, deep neural networks parameterized by ϕ with constraints $f_\phi \in \Omega_\phi$ [4], result in a distance approximation

$$W_L(\mathbb{P}, \mathbb{G}) = \max_{f_\phi \in \Omega_\phi} \mathbb{E}_{x\sim P} f_\phi(x) - \mathbb{E}_{x\sim G} f_\phi(x). \tag{3.14}$$

If there exists k such that $\Omega_f \subseteq \mathcal{L}_k$, then $W_L(\mathbb{P}, \mathbb{G})/k \leq w(\mathbb{P}, \mathbb{G})\ \forall P, G$ is a lower bound. To enforce $\Omega_\phi \subseteq \mathcal{L}_k$, Arjovsky et al. [4] propose a weight clipping constraint Ω_c, which constrains every weight to be in $[-c, c]$ and guarantees that $\Omega_c \subseteq \mathcal{L}_k$ for some k. However, choosing clipping range c is non-trivial in practice. Small ranges limit the capacity of networks, while large ranges result in numerical issues during the training. On the other hand, in addition to weight clipping, several constraints (regularization) have bee proposed with better empirical performance, such as gradient penalty [37] and L^2 ball [75]. However, there is no guarantee the resulted functions are still Lipschitz or the resulted distances are lower bounds of Wasserstein distance. To take the advantage of those regularization with the Lipschitz guarantee, we propose a simple variation by combining weight clipping, which always ensures Lipschitz functions.

Lemma 11. *There exists $k > 0$ such that*

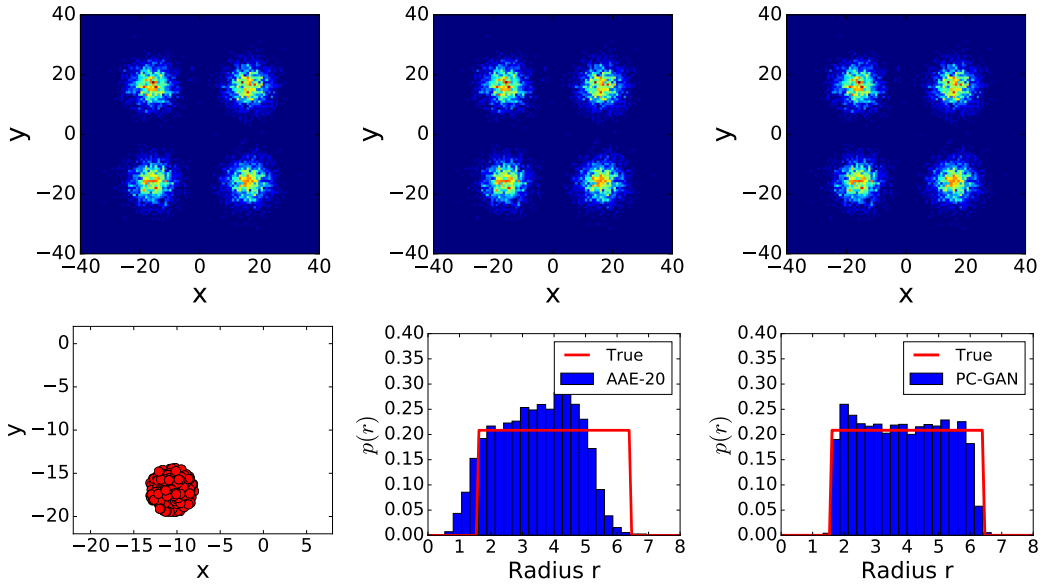
$$\max_{f\in\Omega_c\cap\Omega_\phi} \mathbb{E}_{x\sim P}[f_\phi(x)] - \mathbb{E}_{x\sim G}[f_\phi(x)] \leq \frac{1}{k} w(\mathbb{P}, \mathbb{G}) \tag{3.15}$$

Note that, if $c \to \infty$, then $\Omega_c \cap \Omega_\phi = \Omega_\phi$. Therefore, from Proposition 11, for any regularization of discriminator [37, 75, 76], we can always combine it with a weight clipping constraint Ω_c to ensure a valid lower bound estimate of Wasserstein distance and enjoy the advantage that it is numerically stable when we use large c compared with original weight-clipping WGAN [4].

In practice, we found combing L^2 ball constraint and weight-clipping leads to satisfactory performance. We also studied popular WGAN-GP [37] with weight clipping to ensure Lipschitz continuity of discriminator, but we found L^2 ball with weight clipping is faster and more numerically stable to train.

3.2.4 Preliminary Results

Evaluating performance of generative models, especially for IGMs or GANs is still an open problem. We generate 2D circle point clouds. The center of circles follows a mixture of Gaussians $\mathcal{N}\left(\{\pm 16\} \times \{\pm 16\}, 16I\right)$ with equal mixture weights. The radius of the circles was drawn from a uniform distribution *Unif*$(1.6, 6.4)$. One sampled circle is shown in Figure 3.8a. From Figure 3.8, both AAE and PC-GAN can successfully recover the center distribution, but AAE does not learn the radius distribution well even with larger latent code (20) and more parameters ($24K$). The gap of memory usage could be larger if we configure AAE to generate more points, while the model size required for PC-GAN is independent of the number of points. The reason is MLP decoder adopted by Achlioptas et al. [1] wastes parameters for nearby points. Using the much larger model (more parameters) could boost the performance.



(a) True Data (b) AAE (c) PC-GAN

Figure 3.8: (a) (top) the true center distribution and (bottom) one example of a circle point cloud. (b-d) are the reconstructed center and radius distributions.

We then consider ModelNet40 [110] benchmark, which contains 40 classes of objects with $9,843$ training and $2,468$ testing instances. After training PC-GAN, we run a hierarchical sampling process for sampling point clouds. In the first hierarchy, the generator G_θ, samples a object ($\theta = G_\theta(u), u \sim \mathbb{P}(u)$), while the second generator G_x samples points based on θ to form a point cloud. The randomly sampled results without given any data as input are shown in Figure 3.9. The point clouds are all smooth, structured and almost symmetric. It shows PC-GAN captures inherent symmetries and patterns in all the randomly sampled objects, even if overall object is not perfectly formed. This highlights that learning point-wise generation scheme encourages learning basic building blocks of objects.

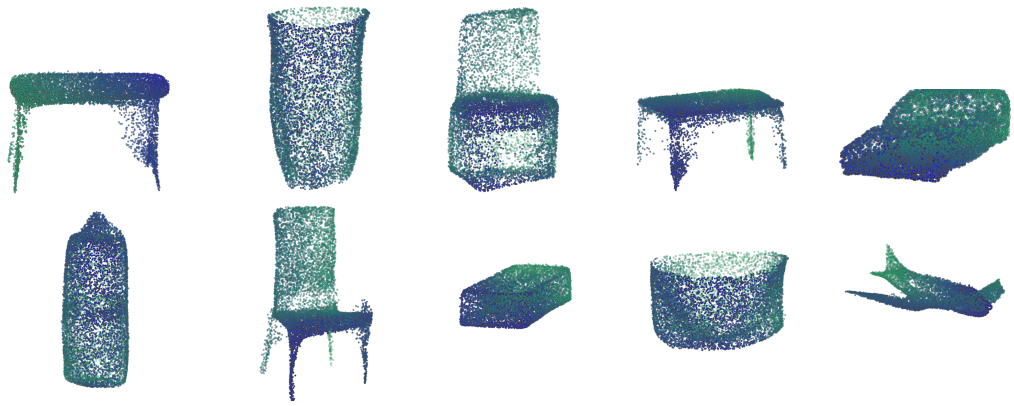


Figure 3.9: Randomly sampled objects and corresponding point cloud from the hierarchical sampling Even if there are some defects, the objects are smooth, symmetric and structured.

3.3 Proposed: Learning with Different Priors

Although point cloud is easy to operate because of its permutation invariance property, in real-world applications a 3D mesh is still desirable. There are existing algorithms to transform point cloud to mesh, we are interested in learning to generate mesh directly. A 3D mesh contains pre-specified vertices and faces. A simple way to connect developed PCGAN and mesh is learning transformation on vertices from a base template, which is closely related to mesh deformation. The idea can be treated as a first step from zero prior (e.g. Gaussian distribution) to human prior (e.g. specialized base template). Some model fitting (reconstruction) works have been explored in literatures (e.g. [36, 47]). Our goal is to explore generation and creation.

Beyond simple template, there are simple and well crafted rendering pipelines well developed in computer graphics. A natural question is how we can utilize this pipelines encoded human prior knowledge in the generator design of IGMs. The ideas have been studied by Shrivastava et al. [95], however, they only use the renderer *passively* by replacing the random noise z with the generated synthetic data. In this thesis, we want to investigate how to *actively* learn to control the render and refine the rendered output. By learning to control renderers, as a by product, we could also get interpretable generation based on the parameters of renderers. There are two major category of problems, including differentiable renders [63, 67] and non-differentiable renders [93], where we need to design different algorithms to tackle them.

Chapter 4

Conditional Transformations

Compared with the generation via transforming zero prior (e.g. Gaussian distribution) into data, conditional generation has braoder applications in practice. In computer vision, many problems can be formed as "translating" an input image to the corresponding output image. The transformationed learned by GANs achieve state-of-the-art performance on different applications, including superresolution [55], inpainting [81], image editing [43], image refinement [95] and style transformation [120]. In language, this framework also covers a broad spectrum of problems, including semantic transformation and machine translation [42, 94]. The goal of this chapter is to investigate these two problems.

4.1 Universal Transformation for Linear Inverse Problems

At the heart of many image processing tasks is a linear inverse problem, where the goal is to reconstruct an image $x \in \mathbb{R}^d$ from a set of measurements $y \in \mathbb{R}^m$ of the form $y = Ax+n$, where $A \in \mathbb{R}^{m\times d}$ is the measurement operator and $n \in \mathbb{R}^m$ is the noise. Many problems discussed at the beginning of this chapter can be modeled by this formulation. For example, in image inpainting, y is an image with masked regions and A is the linear operation applying a pixelwise mask to the original image x; in super-resolution, y is a low-resolution image and the operation A downsamples high resolution images; in compressive sensing, y denotes compressive measurements and A is the measurement matrix, e.g., a random Gaussian matrix. Linear inverse problems, like those described above, are often underdetermined, i.e., they involve fewer measurements than unknowns. Such underdetermined systems are extremely difficult to solve since the operator A has a non-trivial null space and there are an infinite number of feasible solutions but only a few of them are natural images.

There are two broad classes of methods for solving linear underdetermined problems. At one end, we design signal priors to regularize the inverse problems by constraining the infinite set of feasible solutions. Traditionally, signal priors are hand-designed based on empirical observations of images. For example, since natural images are usually sparse after wavelet transformation and are generally piecewise smooth, signal priors that constrain the

sparsity of wavelet coefficients or spatial gradients are widely used [12, 17, 18, 68, 83]. Even though these signal priors can be used in any linear inverse problems related to images and usually have efficient solvers, these signal priors are often too generic, in that many non-image signals can also satisfy the constraints. Thereby, these hand-designed signal priors cannot easily deal with challenging problems like image inpainting or super-resolution.

Instead of using a universal signal prior, a second class of method learn a *transformation* from the domain of y to the image space of x, with the help of large datasets and deep neural nets [16, 55, 81]. For example, to solve image super-resolution, low-resolution images are generated from a high-resolution image dataset, and the mapping between the corresponding image pairs are learned with a neural net [16]. Similarly, a network can be trained to solve compressive sensing problem [51, 73, 74] or image debluring [112]. These methods have achieved state-of-the-art performance in many challenging problems.

Despite their superior performance, these specially-trained transformations are designed for specific problems and usually cannot solve other problems without retraining the mapping function — even when the problems are similar. For example, a 2×-super-resolution network cannot be easily readapted to solve 4× or 8× super-resolution problems; a compressive sensing network for Gaussian random measurements is not applicable to subsampled Hadamard measurements. Training a new network for every single instance of an inverse problem is a wasteful proposition. In comparison, traditional methods using hand-designed signal priors can solve any linear inverse problems but have poorer performance on an individual problem. Clearly, a middle ground between these two classes of methods is needed.

4.1.1 One Network to Solve Them All

The proposed framework [13] is motivated by the optimization technique, alternating direction method of multipliers method (ADMM) [11], that is widely used to solve linear inverse problems with constraints. A typical first step in ADMM is to separate a complicated objective into several simpler ones by variable splitting, i.e., introducing an additional variable z that is constrained to be equal to x. This gives us the following optimization problem:

$$\min_{x,z} \ \frac{1}{2}||y - Az||_2^2 + \lambda\,\phi(x) \tag{4.1}$$

$$\text{s.t.} \ \ x = z, \tag{4.2}$$

where ϕ is a signal prior (e.g. ℓ_1 for sparsity). The scaled form of the augmented Lagrangian of (4.2) can be written as

$$\mathcal{L}(x, z, u) = \frac{1}{2}\|y - Az\|_2^2 + \lambda\,\phi(x) + \frac{\rho}{2}||x - z + u||_2^2, \tag{4.3}$$

where $\rho > 0$ is the penalty parameter of the constraint $x = z$, and u is the dual variables divided by ρ. By alternatively optimizing $\mathcal{L}(x, z, u)$ over x, z, and u, ADMM is composed

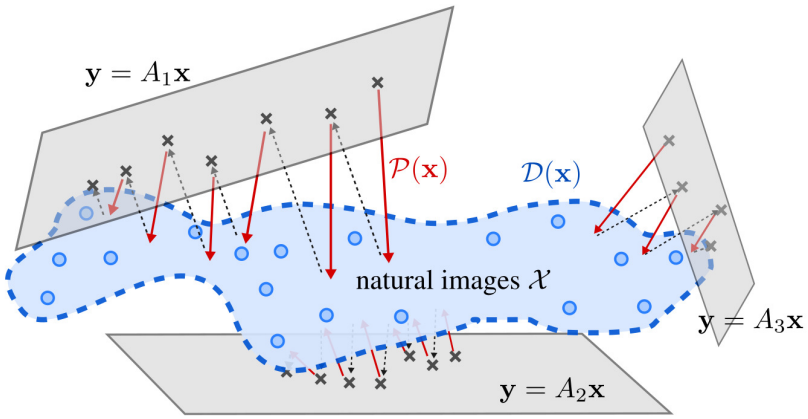


Figure 4.1: Given a large image dataset, the proposed framework learns a projection network $P(x)$:$\mathbb{R}^d \to \mathbb{R}^d$ is trained to fit the proximal operator, which enables one to solve a variety of linear inverse problems using ADMM.

of the following procedures:

$$x^{(k+1)} \leftarrow \underset{x}{\operatorname{argmin}} \frac{\rho}{2}||x - z^{(k)} + u^{(k)}||_2^2 + \lambda\,\phi(x) \tag{4.4}$$

$$z^{(k+1)} \leftarrow \underset{z}{\operatorname{argmin}} \frac{1}{2}||y - Az||_2^2 + \frac{\rho}{2}||x^{(k+1)} - z + u^{(k)}||_2^2 \tag{4.5}$$

$$u^{(k+1)} \leftarrow u^{(k)} + x^{(k+1)} - z^{(k+1)}. \tag{4.6}$$

The update of z (4.5) is a least squares problem and can be solved efficiently via algorithms like conjugate gradient descent. The update of x (4.4) is the proximal operator of the signal prior ϕ with penalty $\frac{\rho}{\lambda}$, denoted as $\mathbf{prox}_{\phi,\frac{\rho}{\lambda}}(v)$, where $v = z^{(k)} - u^{(k)}$. When the signal prior uses ℓ_1-norm, the proximal operator is simply a soft-thresholding on v. Notice that the ADMM algorithm separates the signal prior ϕ from the linear operator A. This enables us to learn a proximal operator (4.4) that can be used with any linear operator.

4.1.2 Learning Transformation as Proximal Operators

Since the signal prior only appears in the form of a proximal operator in ADMM, instead of explicitly learning a signal prior ϕ and solving the proximal operator in each step of ADMM, we propose to directly learn the proximal operator.

Let $\mathcal{X}$ represent the set of all natural images. The best signal prior is the indicator function of $\mathcal{X}$, denoted as $\mathcal{I}_{\mathcal{X}}(\cdot)$, and its corresponding proximal operator $\mathbf{prox}_{\mathcal{I}_{\mathcal{X}},\rho}(v)$ is a projection operator that projects v onto $\mathcal{X}$ from the geometric perspective— or equivalently, finding a $x \in \mathcal{X}$ such that $||x - v||$ is minimized. However, we do not have the oracle indicator function $\mathcal{I}_{\mathcal{X}}(\cdot)$ in practice, so we cannot evaluate $\mathbf{prox}_{\mathcal{I}_{\mathcal{X}},\rho}(v)$ to solve the projection operation. Instead, we propose to train a critic $\mathcal{D}$ with a large dataset to approximates $\mathcal{I}_{\mathcal{X}}$. Based on the learned classifier $\mathcal{D}$, we can learn a projection function $\mathcal{P}$ that maps a signal v to the set defined by the critic. The above procedure can also be interpreted as adversarial training used in conditional GANs []. The learned projection function $\mathcal{P}$ can then replace the proximal operator (4.4), and we simply update x via

$$x^{(k+1)} \leftarrow \mathcal{P}(z^{(k)} - u^{(k)}). \tag{4.7}$$

An illustration of the idea is shown in Figure 4.1.

There are some caveats for this approach. First, when the classification cost function of the classifier $\mathcal{D}$ is nonconvex, the overall optimization becomes nonconvex. For solving general nonconvex optimization problems, the convergence result is not guaranteed. Based on the theorems for the convergence of nonconvex ADMM [103], we provide the following theorem to the proposed ADMM framework.

Theorem 12. *Assume the function $\mathcal{P}$ solves the proximal operator* (4.4)*. If the gradient of $\phi(x)$ is Lipschitz continuous and with large enough ρ, the ADMM algorithm is guaranteed to attain a stationary point.*

We remark that, different from Arjovsky et al. [4], the Lipschitz constraint is on projection network $\mathcal{P}$ instead of the critic $\mathcal{D}$.

4.1.3 Implementation Details

To bound the gradients of $\mathcal{D}$ w.r.t. x, we truncate the weights of the network after each iteration as weight-clipping introduced in Arjovsky et al. [4]. The remaining question is the input for the projection network during the training. For each image, we train the critic and the projected via their perturbed counterparts. While adding Gaussian noise may be the simplest method to perturb an image, we found that the projection network will easily overfit the Gaussian noise and become a dedicated Gaussian denoiser. Since during the ADMM process, the inputs to the projection network, $z^{(k)} - u^{(k)}$, do not usually follow a Gaussian distribution, the overfitted projection network may fail to project the general signals produced by the ADMM process. To avoid overfitting, we generate perturbed images with two methods — adding Gaussian noise with spatially varying standard deviations and smoothing the input images. We generate the noise by multiplying a randomly sampled standard Gaussian noise with a weighted mask upsampled from a low-dimensional mask with bicubic algorithm. The weighted mask is randomly sampled from a uniform distribution ranging from $[0.05, 0.5]$. Note that the images are ranging from $[-1, 1]$. To smooth the input images, we first downsample the input images and then use nearest-neighbor method to upsample the results. The ratio to the downsample is uniformly sampled from $[0.2, 0.95]$. After smoothing the images, we add the noise described above. The projection network $\mathcal{P}$ can be thought as a denoising autoencoder with a regularization term. Compared to the typical denoising auto-encoder, which always projects a perturbed image $x_0 + n$ back to the origin image x_0, the proposed $\mathcal{P}$ may project $x_0 + n$ to the closest x in $\mathcal{X}$.

4.1.4 Preliminary Results

We evaluate the proposed framework on Celeb-1M dataset [38]. It contains a total of 8 million aligned and cropped face images of 10 thousand people from different viewing angles. We randomly select images of 73678 people as the training set and those of 25923 people as the test set. We resize the images into 64×64. We perform the following tasks:

Compressive sensing. We use $m \times d$ random Gaussian matrices of different compression ($\frac{m}{d}$) as the linear operator A. The images are vectorized into d-dimensional vectors x and multiplied with the random Gaussian matrices to form y.

Pixelwise inpainting and denoising. We randomly drop pixel values (independent of channels) by filling zeros and add Gaussian noise with different standard deviations.

Scattered inpainting. We randomly drop 10 small blocks by filling zeros. Each block is of 10% width and height of the input.
Blockwise inpainting. We fill the center 30% region of the input images with zeros.
Super resolution. We downsample the images into 50% and 25% of the original width and height using box-averaging algorithm.

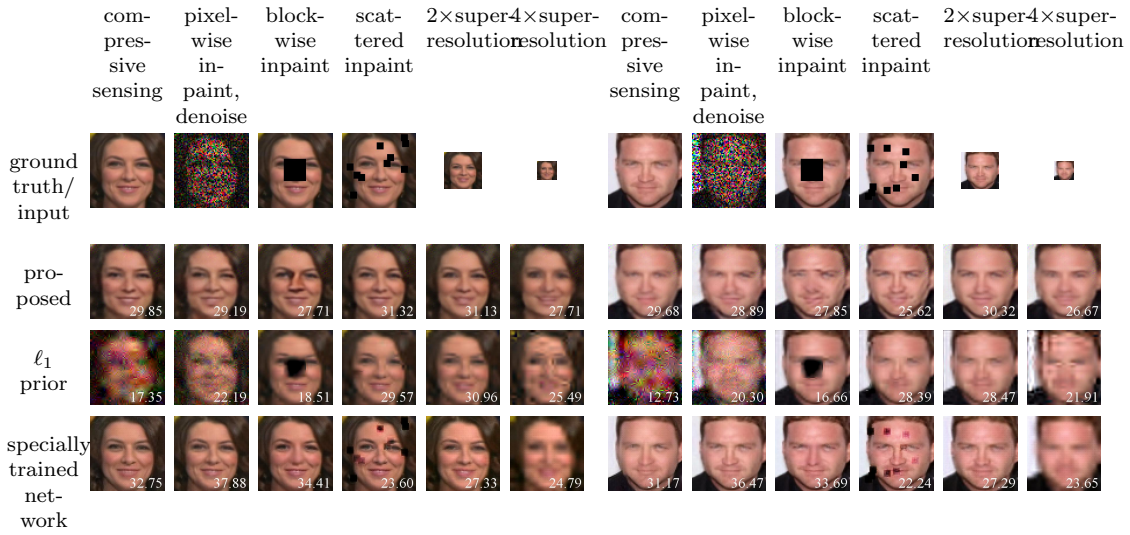


Figure 4.2: Results on MS-Celeb-1M dataset. The PSNR values are shown in the lower-right corner of each image. For compressive sensing, we test on $\frac{m}{d} = 0.1$. For pixelwise inpainting, we drop 50% of the pixels and add Gaussian noise with $\sigma = 0.1$. We use $\rho = 1.0$ on both super resolution tasks.

The result is shown in Figure 4.2. As can be seen from the results, using the proposed projection operator/network learning from datasets enables us to solve more challenging problems than using the traditional wavelet sparsity prior. While the traditional ℓ_1-prior of wavelet coefficients is able to reconstruct images from compressive measurements with $\frac{m}{d} = 0.3$, it fails to handle larger compression ratios like $\frac{m}{d} = 0.1$ and 0.03. Similar observations can be seen on pixelwise inpainting of different dropping probabilities and scattered and blockwise inpainting. In contrast, since the proposed projection network is tailored to the datasets, it enables the ADMM algorithm to solve challenging problems like compressive sensing with small $\frac{m}{d}$ and blockwise inpainting.

4.2 Proposed work: Conditional Language Generation

In language transformation, if we have one-to-one matching pairs for transformation. Seq2Seq [98] leads to state-of-the-art performance. One challenge arises when no paired information is given. Recently, similar to CycleGAN [120], unpaired language transformation is studied by Hu et al. [42], Shen et al. [94], while the building component is still the generation in one domain, such as vanilla language models. As we propose in Chapter 3.1, one direction is to investigate different structure information in the distance measure for improving language generation. An extension is to incorporate better language generation algorithm with the framework proposed by for conditional generation.

Chapter 5

Other Research

In addition to the works covered by this thesis, I have also worked on different topics during my PhD study. I briefly review my contribution in those areas.

5.1 Bayesian Optimization [58]

Bayesian Optimization (BO) is commonly used to optimize blackbox objective functions which are expensive to evaluate. A common approach is based on using Gaussian Process (GP) to model the objective function. Applying GP to higher dimensional settings is generally difficult due to the curse of dimensionality. Existing works makes strong assumptions such as the function is low-dimensional embedding or is axis-aligned additive. We generalize the existing assumption to a projected-additive assumption. Our generalization provides the benefits of i) greatly increasing the space of functions that can be modeled by our approach, which covers the previous works as special cases, and ii) efficiently handling the learning in a larger model space. We prove that the regret for projected-additive functions has only linear dependence on the number of dimensions in this general setting. Directly using projected-additive GP to BO results in a non-box constraint, which is not easy to optimize. We tackle this problem by proposing a restricted-projection-pursuit GP for BO. Our method outperforms existing approaches even when the function does not meet the projected additive assumption.

5.2 Large-Scale Kernel Learning [57]

To address the scalability issue of kernel methods, random features are commonly used for kernel approximation. However, to achieve high precision results, one might still need a large number of random features, which is infeasible in large-scale applications. Existing works address this issue by recomputing the random features of small batches in each iteration instead of pre-generating for the whole dataset and keeping them in the memory. The algorithm increases the number of random features linearly with iterations, which can reduce the approximation error. A drawback is that the large number of random features slows down the prediction and gradient evaluation. We propose two algorithms to remedy this situation by reusing old random features instead of adding new features in certain iterations. By checking the expected descent amount, the proposed algorithm selects important old features to update. The resulting procedure is simple without enhancing the complexity of the original algorithm but effective in practice.

5.3 Polynomial Optimization [102]

Matrix factorization is a core technique in many machine learning problems, yet also presents a nonconvex and often difficult-to-optimize problem. We present an approach based upon polynomial optimization techniques that both improves the convergence time of matrix factorization algorithms and helps them escape from local optima. Our method is based on the realization that given a joint search direction in a matrix factorization task, we can solve the subspace search problem (the task of jointly finding the steps to take in each direction) by solving a bivariate quartic polynomial optimization problem.

5.4 Video Anomaly Detection [64]

We study challenging anomaly detections in streaming videos under fully unsupervised settings by proposing a novel perspective to establish the connection between the heuristic unmasking procedure and multiple classifier two sample tests (MC2ST) in statistical machine leaning. Based on our analysis of the testing power of MC2ST, we present a history sampling method to increase the testing power as well as to improve the performance on video anomaly detection. We also offer a new frame-level motion feature that has better representation and generalization ability.

Chapter 6

Proposed Timeline

- Text Generation and conditional generation (∼Feb. 2019)
- Prior knowledge (∼March 2019)
- Mesh generation (∼May 2019)
- Thesis Writing (June∼July 2019)
- Thesis Defense (Aug 2019)

Bibliography

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 3.2, 3.2.2, 3.2.4

[2] David Alvarez-Melis, Tommi S Jaakkola, and Stefanie Jegelka. Structured optimal transport. *arXiv preprint arXiv:1712.06199*, 2017. 3.1.4

[3] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 2.3.3, 3, 3.1.3

[4] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. In *ICML*, 2017. 2, 2.1.2, 2.2, 2.2, 2.2.4, 6, 3.1.1, 3.2.1, 3.2.2, 3.2.3, 3.2.3, 3.2.3, 3.2.3, 4.1.2, 4.1.3

[5] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017. 2.2, 3.2.3

[6] Marc G Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017. 2.2.2, 2.2.3, 2.3

[7] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 2.2.3

[8] Dimitri P Bertsekas. A distributed asynchronous relaxation algorithm for the assignment problem. In *Decision and Control*, 1985. 3.2.3

[9] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *ICLR*, 2018. 2.3, 2.3.3, 2.3.4, 2.3.4

[10] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015. 3.1

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. 4.1.1

[12] Tony F Chan, Jianhong Shen, and Hao-Min Zhou. Total variation wavelet inpainting. *Journal of Mathematical imaging and Vision*, 25(1):107–125, 2006. 4.1

[13] Jen-Hao Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all-solving linear inverse problems using deep projection models. In *ICCV*, 2017. 4.1.1

[14] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*,

2013. 3.2.3

[15] Peter J Diggle and Richard J Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1984. 1

[16] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, 2014. 4.1

[17] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Transactions on Image Processing*, 20(7):1838–1857, 2011. 4.1

[18] David L Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995. 4.1

[19] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017. 2.2.4, **??**

[20] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*, 2016. 3.1.4

[21] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *UAI*, 2015. 2.1

[22] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration. In *3DV*, 2015. 3.2.2

[23] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017. 1

[24] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 3.2

[25] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017. 1

[26] Kenji Fukumizu, Arthur Gretton, Gert R Lanckriet, Bernhard Schölkopf, and Bharath K Sriperumbudur. Kernel choice and classifiability for rkhs embeddings of probability distributions. In *NIPS*, 2009. 2.1.1

[27] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *AISTATS*, 2018. 3.2.3

[28] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *JMLR*, 2011. 2.3.1

[29] 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*, 2014. 1, 3, 3.2.1

[30] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2, 2.1.2, 2.2, 2.3.3

[31] A. Gretton, B. Sriperumbudur, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, and K. Fukumizu. Optimal kernel choice for large-scale two-sample tests. In *NIPS*, 2012. 2.1.1

[32] Arthur Gretton. Notes on the cramer gan. `https://medium.com/towards-data-science/notes-on-the-cramer-gan-752abd505c00`, 2017. Accessed: 2017-11-2. 2.2.2, 2.2.3

[33] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 2012. 2.1, 1, 2.1.1, 2.1.1, 2.3.1

[34] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 2012. 1

[35] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *NIPS*, 2012. 2.1.1

[36] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *ECCV*, 2018. 3.3

[37] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 2.2, 2.2, 2.2.2, 2.3.4, 3, 3.1, 2, 3.1.3, 3.1.3, 3.2.3, 3.2.3

[38] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision (ECCV)*, 2016. 4.1.4

[39] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. 1999. 2.2.1

[40] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 2.3.4

[41] R. Devon Hjelm, Athul Paul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. *arXiv:1702.08431*, 2017. 3.1

[42] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. *arXiv preprint arXiv:1703.00955*, 2017. 4, 4.2

[43] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017. 1, 3.2.2, 4

[44] Bing Jian and Baba C Vemuri. A robust algorithm for point set registration using mixture of gaussians. In *ICCV*, 2005. 3.2.2

[45] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *ICLR*, 2017. 3.1.3

[46] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. *CVPR*, 2, 2017. 3.2

[47] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. *arXiv preprint arXiv:1803.07549*, 2018. 3.3

[48] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2013. 2,

2.2

[49] Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 1

[50] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 2.3.4

[51] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *CVPR*, 2016. 4.1

[52] Matt J. Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv:1611.04051*, 2016. 3.1

[53] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, 2017. 1

[54] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*, 2016. 3

[55] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 4, 4.1

[56] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 1

[57] Chun-Liang Li and Barnabás Póczos. Utilize old coordinates: Faster doubly stochastic gradients for kernel methods. In *UAI*, 2016. (document), 5.2

[58] Chun-Liang Li, Kirthevasan Kandasamy, Barnabás Póczos, and Jeff Schneider. High dimensional bayesian optimization via restricted projection pursuit models. In *AISTATS*, 2016. (document), 5.1

[59] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *NIPS*, 2017. 1, 2.2, 2.3.4, 2.3.4, 3, 3.2.3

[60] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018. 1, 3

[61] Chun-Liang Li, Wei-Cheng Chang, Youssef Mroueh, Yiming Yang, and Barnabás Póczos. Implicit kernel learning. In *AISTATS*, 2019. 1

[62] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. In *ICML*, 2015. 2.1, 2.2, 2.2

[63] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Adversarial geometry and lighting using a differentiable renderer. *arXiv preprint arXiv:1808.02651*, 2018. 3.3

[64] Yusha Liu, Chun-Liang Li, and Barnabás Póczos. Classifier two-sample test for video anomaly detections. In *BMVC*, 2018. (document), 5.4

[65] Gilles Louppe and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. *arXiv preprint arXiv:1707.07113*, 2017. 1

[66] David JC MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1995. 1

[67] Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface88*. Citeseer, 1988. 3.3

[68] Julien Mairal, Guillermo Sapiro, and Michael Elad. Learning multiscale sparse representations for image and video restoration. *Multiscale Modeling & Simulation*, 7(1):214–241, 2008. 4.1

[69] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 2, 2.2

[70] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *ISCA*, 2010. 3.1

[71] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 3.2.2

[72] Hossein Mobahi and John W. Fisher III. A Theoretical Analysis of Optimization by Gaussian Continuation. In *AAAI*, 2015. 3.1.3

[73] Ali Mousavi and Richard G Baraniuk. Learning to invert: Signal recovery via deep convolutional networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, 2017. 4.1

[74] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *Allerton Conference on Communication, Control, and Computing*, 2015. 4.1

[75] Youssef Mroueh and Tom Sercu. Fisher gan. In *NIPS*, 2017. 3, 3.1.2, 3.2.2, 3.2.3, 3.2.3

[76] Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev gan. *arXiv preprint arXiv:1711.04894*, 2017. 1, 2.3.4, 3, 3.2.2, 3.2.3

[77] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. *arxiv pre-print 1702.08398*, 2017. 2.2

[78] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016. 2

[79] Junier B Oliva, Avinava Dubey, Andrew G Wilson, Barnabás Póczos, Jeff Schneider, and Eric P Xing. Bayesian nonparametric kernel-learning. In *AISTATS*, 2016. 2.3.1

[80] Junier B Oliva, Avinava Dubey, Barnabás Póczos, Jeff Schneider, and Eric P Xing. Transformation autoregressive networks. *arXiv preprint arXiv:1801.09819*, 2018. 3.2

[81] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 4, 4.1

[82] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. Technical report, 2017. 3.2.3

[83] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising

using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003. 4.1

[84] Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv:1706.01399*, 2017. 3.1, 3.1.3, 3.1.3

[85] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 2017. 3.2, 3.2.2

[86] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 3.2

[87] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2.2.4, 2.3.4, 3

[88] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007. 2.3.1, 2.3.1

[89] Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. Adversarial generation of natural language. *arXiv:1705.10929*, 2017. 3.1

[90] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *CVPR*, 2016. 3.1.3

[91] Walter Rudin. *Fourier analysis on groups*. John Wiley & Sons, 2011. 2.3

[92] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016. 1, 2.2.3, 2.2.4, **??**, 2.3.4

[93] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. *arXiv preprint arXiv:1712.08290*, 2017. 3.3

[94] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NIPS*, 2017. 4, 4.2

[95] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 3.3, 4

[96] Johannes Strom, Andrew Richardson, and Edwin Olson. Graph-based segmentation for colored 3d laser point clouds. In *IROS*, 2010. 3.2.2

[97] Dougal J. Sutherland, Hsiao-Yu Fish Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alexander J. Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *ICLR*, 2017. 2.2.3

[98] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 2014. 4.2

[99] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017. 2

[100] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*, 2017. 2.2

[101] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene

dynamics. In *NIPS*, pages 613–621, 2016. 3

[102] Po-Wei Wang, Chun-Liang Li, and J Zico Kolter. Polynomial optimization methods for matrix factorization. In *AAAI*, 2017. (document), 5.3

[103] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*, 2015. 4.1.2

[104] D Warde-Farley and Y Bengio. Improving generative adversarial networks with denoising feature matching. In *ICLR*, 2017. 2.2.4, **??**

[105] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013. 1, 4

[106] Jonathan Weed and Francis Bach. Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance. *arXiv preprint arXiv:1707.00087*, 2017. 3.2.3

[107] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, 2013. 2.3.1, 2.3.4, 2.3.4

[108] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *AISTATS*, 2016. 2.1.1

[109] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, 2016. 3

[110] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 3.2.4

[111] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 1

[112] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 4.1

[113] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018. 1

[114] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, 2016. 3, 3.1

[115] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *NIPS*, 2017. 3.2, 3.2.1, 3.2.2

[116] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*, 2017. 2.2

[117] Shuangfei Zhai, Yu Cheng, Rogério Schmidt Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *CoRR*, 2016. 2.2

[118] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based Generative Adversarial Network. In *ICLR*, 2017. 2, 2.1.2, 2.2

[119] Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially regularized autoencoders for generating discrete structures. *CoRR*, 2017. 3.1

[120] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017. 4, 4.2