Prompt Generation: Unified Distributional Control of Pre-Trained Generative Vision Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Generative models (e.g., GANs, VAEs) learn the underlying data distribution in an unsupervised manner. However, many applications of interest require sampling from a specific region of the generative model's output space or evenly over a range of characteristics. To allow efficient sampling in these scenarios, we propose Prompt Generation (PromptGen), a framework to achieve distributional control over pre-trained generative models by incorporating knowledge of arbitrary off-the-shelf models. PromptGen defines control as an energy-based model (EBM) and samples images in a feed-forward manner by approximating the EBM with invertible neural networks, avoiding optimization at inference. We demonstrate how PromptGen can control several generative models using various off-the-shelf models: (1) with the CLIP model, PromptGen can sample images guided by text, (2) with image classifiers, PromptGen can de-bias generative models across a set of attributes, and (3) with inverse graphics models, PromptGen can sample images of the same identity in different poses. (4) Finally, PromptGen reveals that the CLIP model shows "reporting bias" when used as control, and PromptGen can further de-bias this controlled distribution in an iterative manner.[1]

1 Introduction

Generative models (e.g., GANs, VAEs) learn the underlying data distribution in high-dimensional spaces and achieve promising performance on image synthesis [4, 71, 35, 21]. Though being well praised, they still face two main criticisms. First, since generative models are typically trained in an unsupervised way, they lack controllability, meaning that it is unclear how to sample from a specific region of the space. Second, generative models are prone to inherit the imbalance and bias of training data [57, 30]. For instance, StyleGAN2 is more likely to produce images of white individuals, see Figure 1(e). Previous works have studied these challenges separately, and typical methods include editing of "style" codes [22, 64, 30] and explicit conditions [41, 65]. However, these methods are either model-dependent (i.e., requiring a well-structured style space) or label-intensive (i.e., requiring all training samples to be labeled for explicit conditions), limiting their generality and practical use.

To address the above challenges, this paper advocates a unified formulation, *distributional control of generative models*, which enables controllability by incorporating the knowledge of off-the-shelf models (e.g., CLIP [55], classifiers, or inverse graphics models [16]). Based on this unified view, we propose to learn distributions in the latent space of a pre-trained generative model while keeping the pre-trained parameters fixed. Given its conceptual similarity to prompt learning [40, 79, 70, 80, 29], we term our framework as Prompt Generation (PromptGen). PromptGen requires **no training data**, and the only supervision comes from off-the-shelf models that help define the control. Specifically, PromptGen allows the user to define controls using an energy-based model (EBM) approximated

[1] The code will be made publicly available.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

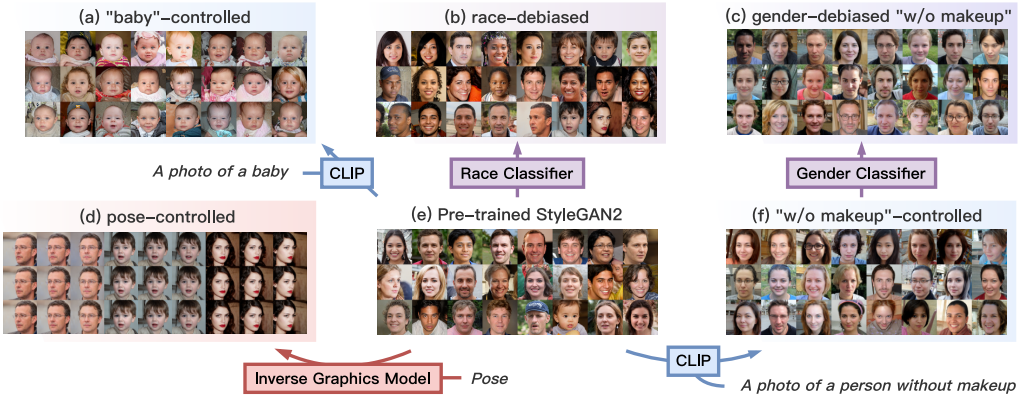


Figure 1: PromptGen uses different off-the-shelf models (e.g., CLIP [55], inverse graphics models, and classifiers) to control the output distribution of a pre-trained and fixed generative model (e.g., StyleGAN2). Colored boxes (e.g., the blue CLIP) indicate the control. See the text for details.

by an invertible neural network (INN). Unlike methods that require optimization at inference in EBM sampling [47, 14, 43, 50], PromptGen samples images in a **feed-forward** manner, which is highly efficient. Moreover, PromptGen **stands alone** at inference, meaning that we can discard the off-the-shelf models, which define the control, after training.

We illustrate the benefits of PromptGen with several experiments. Figure 1 demonstrates our main findings with StyleGAN2 trained on FFHQ [34] without any labels, while we also show results for StyleNeRF [21] and NVAE [71] on non-facial data in Section 4. Figure 1(a) illustrates how we can sample from StyleGAN2 based on text descriptions such as "a photo of a baby". Figure 1(b) shows that PromptGen can leverage a race classifier (potentially trained on a different dataset) to sample uniformly across all races, de-biasing the pre-trained StyleGAN2. Figure 1(d) illustrates that PromptGen can generate images of the same identity in different poses, guided by a pose regressor.

Finally, it is worth pointing out that PromptGen not only offers **generality** for algorithmic design and **modularity** for control composition, but also enables **iterative** controls when some controls are contingent on others. For instance, one may train a text-controlled distribution and then de-bias this distribution. To achieve this, we view the composition of the INN and the generative model as a new "generative model" to be controlled. Figure 1(f) illustrates that PromptGen reveals "reporting bias" of the CLIP model [55], where "without makeup" is – perhaps surprisingly – positively correlated with female, and Figure 1(c) shows that PromptGen can further mitigate this bias with iterative control.

Table 1: Comparison between methods.

	StyleFlow [1]	PPGM [47] / LACE [50]	Guided DDPM [24]	PromptGen
Arbitrary control (e.g., CLIP)	✗	✓	✓	✓
Low-dimensional latent space	✓	✓	✗	✓
Stands alone at inference	✓	✗	✗	✓
Feed-forward (i.e., no inference-time optim.)	✓	✗	✗	✓
Iterative distributional control	✗	✗	✗	✓

2 Related Work

Over the past few years, generative models have gained the ability to generate images with high visual quality. A few of the most widely used methods include generative adversarial networks (GANs) [18], VAE [38], invertible neural networks [11, 12], and diffusion models [24, 67]. In particular, generative models trained on large amounts of unlabeled data, e.g., BigGAN [4], StyleGANs [34, 35, 33, 61], Glow [37], and diffusion models [49, 10], achieve promising image synthesis results.

Despite their success, controllability and de-biasing are still two fundamental challenges generative models face. For controllability, existing methods include explicit conditioning at training [41, 65]

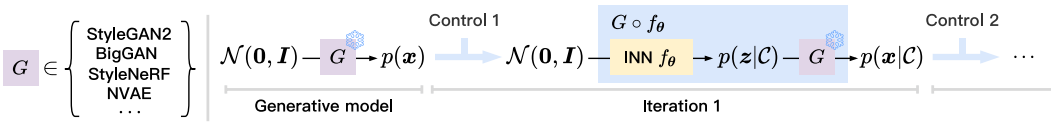


Figure 2: Overview of our Prompt Generation (PromptGen) framework. Given a pre-trained generative model G and a control $\mathcal{C}$, we learn a distribution $p(\boldsymbol{z}|\mathcal{C})$ in G's latent space, while keeping G fixed. Each control $\mathcal{C}$ can have multiple components, e.g., $\mathcal{C} = \{$text = "a photo of a baby", gender = `male`$\}$. PromptGen views the composition $G \circ f_{\boldsymbol{\theta}}$ as a new "generative model" for iterative control.

or local editing of the learned representation, e.g, "style" codes [1, 54, 44, 64, 74]. For de-biasing, existing methods include local editing of "style" codes [57, 30] and importance sampling for either training [20] or inference [28]. Existing works study these problems separately, each requiring a specific design for the task studied. On the contrary, PromptGen is a unified framework for arbitrary controls defined by off-the-shelf models. The benefits of unifying related tasks have been shown by a recent trend of works in multiple research areas across vision and language [56, 46, 60, 75, 58]. Moreover, one can fine-tune pre-trained generative models [17] to achieve domain adaptation, which is orthogonal to PromptGen: since PromptGen outputs a generative model (Algorithm 1), fine-tuning can be applied afterwards. However, we leave this exploration to future studies.

Previous methods usually sample from EBM [39] with Markov Chain Monte Carlo (MCMC) [69, 72, 14, 19, 13, 43]. Among them, plug-and-play generative models (PPGMs) [47] and LACE [50] define latent-space EBMs. However, MCMC requires inference-time optimization, which is inefficient and requires the off-the-shelf models to be available at inference; this is also the criticism for diffusion models, regardless of being gradient-guided [49, 63] or not [24, 25, 48]. In contrast, PromptGen achieves efficient, feed-forward sampling. Table 1 shows a comparison with previous methods. Our INN training is similar to that proposed by [52] to sample from physical systems, which is later used by [73] to solve inverse problems with two composed INNs. Notably, [52] and [73] do not leverage a low-dimensional latent space, and [73] requires training a separate model for each image sample. In this paper, we use INN to model arbitrary EBMs for various generative models.

3 Method

Figure 2 illustrates our PromptGen framework. To begin, the user selects a pre-trained generative model G. PromptGen then lets the user specify a control $\mathcal{C}$ as an energy-based model (EBM) $p(\boldsymbol{z}|\mathcal{C})$. We train an INN $f_{\boldsymbol{\theta}}$ to approximate $p(\boldsymbol{z}|\mathcal{C})$. PromptGen views the functional composition $G \circ f_{\boldsymbol{\theta}}$ as a new generative model and can perform iterative control. Algorithm 1 describes the overall procedure.

Algorithm 1: Prompt Generation

Input: Generative model $G : \mathbb{R}^d \to \mathcal{X}$
repeat
1. **Input:** control $\mathcal{C}$ of the current iteration
2. Define an EBM $p(\boldsymbol{z}|\mathcal{C})$ for $\mathcal{C}$ (Section 3.1)
3. Train an INN $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^d$ to approximate $p(\boldsymbol{z}|\mathcal{C})$ (Section 3.2)
4. $G \leftarrow G \circ f_{\boldsymbol{\theta}}$

until *user stops the iteration*
return $G : \mathbb{R}^d \to \mathcal{X}$

3.1 Latent-Space EBM for Distributional Control

The plug-and-play generative model [47] was first proposed to use a latent-space EBM for controllable image synthesis. If a fixed generative model is used, then theoretically, any image-space EBM can be viewed as a latent-space EBM, as shown by [19] and [50]. We define the latent-space EBM following similar formulation as [47, 19, 50], with some new energy functions. We then extend the formulation to incorporate the moment constraint [36], which was adopted for language modeling, but unlike [36], we define the moment constraint in the latent space to accommodate generative vision models.

We define a control $\mathcal{C}$ as M independent properties $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_M\}$, e.g., $\boldsymbol{y}_1$ can be a text description and $\boldsymbol{y}_2$ can be an attribute. The controllability can be defined as the EBM (detailed in Appendix B.1)

$$p(\boldsymbol{x}|\mathcal{C}) = \frac{p_{\boldsymbol{x}}(\boldsymbol{x})e^{-E_{\mathcal{C}}(\boldsymbol{x})}}{Z_X}, \quad E_{\mathcal{C}}(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i), Z_X = \int_{\boldsymbol{x}'} p_{\boldsymbol{x}}(\boldsymbol{x}')e^{-E_{\mathcal{C}}(\boldsymbol{x}')}d\boldsymbol{x}', \quad (1)$$

which reweights the image prior $p_{\boldsymbol{x}}(\boldsymbol{x})$ with energy $E_{\mathcal{C}}(\boldsymbol{x})$, where images with smaller energy are preferred. Using a pre-trained generative model $G : \mathbb{R}^d \to \mathcal{X}$ that maps a latent code $\boldsymbol{z}$ to an image $\boldsymbol{x}$, the image prior $p_{\boldsymbol{x}}(\boldsymbol{x})$ is defined by sampling a latent code $\boldsymbol{z}$ from $p_{\boldsymbol{z}} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and mapping it to $\boldsymbol{x} = G(\boldsymbol{z})$. Appendix B.2 shows that this EBM is equivalent to the latent-space EBM

$$p(\boldsymbol{z}|\mathcal{C}) = \frac{p_{\boldsymbol{z}}(\boldsymbol{z})e^{-E_{\mathcal{C}}(G(\boldsymbol{z}))}}{Z}, \quad E_{\mathcal{C}}(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i), Z = \int_{\boldsymbol{z}'} p_{\boldsymbol{z}}(\boldsymbol{z}')e^{-E_{\mathcal{C}}(G(\boldsymbol{z}'))}d\boldsymbol{z}'. \quad (2)$$

Latent-space EBM allows us to use any off-the-shelf model to specify the control. The following are some examples that are discussed in this paper (explained in Appendix B.1):

Classifier energy: Given a classifier $P(\cdot|\boldsymbol{x})$ and the target class a that we want to sample images from, we define the classifier energy as $E_{\text{classifier}}(\boldsymbol{x}, a) = -\log P(a|\boldsymbol{x})$.

CLIP energy: Using the CLIP model [55], we define the CLIP energy as the cosine distance between the embeddings of the image and the text $\boldsymbol{y}$, averaged over L differentiable augmentations [78, 44]:

$$E_{\text{CLIP}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{L}\sum_{l=1}^{L}\left(1 - \cos\left\langle \text{CLIP}_{\text{img}}\big(\text{DiffAug}_l(\boldsymbol{x})\big), \text{CLIP}_{\text{text}}(\boldsymbol{y})\right\rangle\right). \quad (3)$$

Inverse graphics energy: Given an inverse graphics model, $f_{\mathcal{X}\to\mathcal{P}}$, which infers image parameters (e.g., pose and expression), and the target parameters $\boldsymbol{\rho}$, we define the inverse graphics energy as

$$E_{\text{inv-graphics}}(\boldsymbol{x}, \boldsymbol{\rho}) = d\big\langle f_{\mathcal{X}\to\mathcal{P}}(\boldsymbol{x}), \boldsymbol{\rho}\big\rangle^2, \quad (4)$$

where $d\langle\cdot,\cdot\rangle$ is the geodesic distance between the inferred parameters and the target parameters.

Moment constraint: Some controls cannot be *directly* defined by off-the-shelf models, and the moment constraint [7, 36] is one of them. Given a mapping $\boldsymbol{\gamma} : \mathcal{X} \to \mathbb{R}^K$ (e.g., $\boldsymbol{\gamma}$ can be a classifier that outputs the probability simplex), the moment constraint defines the target distribution $p(\boldsymbol{x}|\mathcal{C})$ as

$$p(\boldsymbol{x}|\mathcal{C}) = \underbrace{\arg\min_{p(\boldsymbol{x}|\mathcal{C})} \mathbb{D}_{\mathsf{KL}}(p(\boldsymbol{x}|\mathcal{C})\|p_{\boldsymbol{x}}(\boldsymbol{x}))}_{\text{Deviation from the pre-trained distribution}}, \quad \text{s.t.} \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\mathcal{C})}\big[\boldsymbol{\gamma}(\boldsymbol{x})\big] = \boldsymbol{\mu}}_{\text{Moment constraint}}, \quad (5)$$

where $\boldsymbol{\mu}$ is the user-specified vector. For example, if we want to generate images that are uniformly distributed across races, we may use a race classifier as $\boldsymbol{\gamma}$ and define $\boldsymbol{\mu} = \big(|\mathcal{A}|^{-1}, \dots, |\mathcal{A}|^{-1}\big)$ where $\mathcal{A}$ is the set of races. In this paper, we generalize the moment constraint to the latent space, and approximate the above objective as (detailed in Appendix B.6):

$$p(\boldsymbol{z}|\mathcal{C}) = \frac{p_{\boldsymbol{z}}(\boldsymbol{z})\exp\Big(\hat{\boldsymbol{\beta}}^\top\boldsymbol{\gamma}\big(G(\boldsymbol{z})\big)\Big)}{Z}, \quad Z = \int_{\boldsymbol{z}'} p_{\boldsymbol{z}}(\boldsymbol{z}')\exp\Big(\hat{\boldsymbol{\beta}}^\top\boldsymbol{\gamma}\big(G(\boldsymbol{z}')\big)\Big)d\boldsymbol{z}', \quad (6)$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \mathbb{E}_{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(N)}\overset{\text{i.i.d.}}{\sim}p_{\boldsymbol{z}}(\boldsymbol{z}),\boldsymbol{x}^{(j)}=G(\boldsymbol{z}^{(j)})}\left\|\frac{\sum_{j=1}^{N}\exp\big(\boldsymbol{\beta}^\top\boldsymbol{\gamma}(\boldsymbol{x}^{(j)})\big)\boldsymbol{\gamma}(\boldsymbol{x}^{(j)})}{\sum_{j'=1}^{N}\exp\big(\boldsymbol{\beta}^\top\boldsymbol{\gamma}(\boldsymbol{x}^{(j')})\big)} - \boldsymbol{\mu}\right\|_2^2. \quad (7)$$

3.2 Approximating EBM with Invertible Neural Network

Given the functional form of the EBM $p(\boldsymbol{z}|\mathcal{C})$, our next step is to approximate it with an efficient sampling network. To achieve this, we train a distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ that minimizes the KL divergence $\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z})\|p(\boldsymbol{z}|\mathcal{C}))$. Estimating $\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z})\|p(\boldsymbol{z}|\mathcal{C}))$ requires easily sampling $\boldsymbol{z} \sim p_{\boldsymbol{\theta}}$ and tractably computing $p_{\boldsymbol{\theta}}(\boldsymbol{z})$. Inspired by [52], we model $p_{\boldsymbol{\theta}}$ with an INN $f_{\boldsymbol{\theta}}$ that defines a bijection $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$, which has two merits besides the invertibility: (1) one can easily sample $\boldsymbol{z}$ by sampling $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and mapping it to $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$, and (2) $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ has a closed-form solution:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \log\mathcal{N}(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{I}) - \log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial\boldsymbol{\epsilon}})|, \quad \boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}). \quad (8)$$

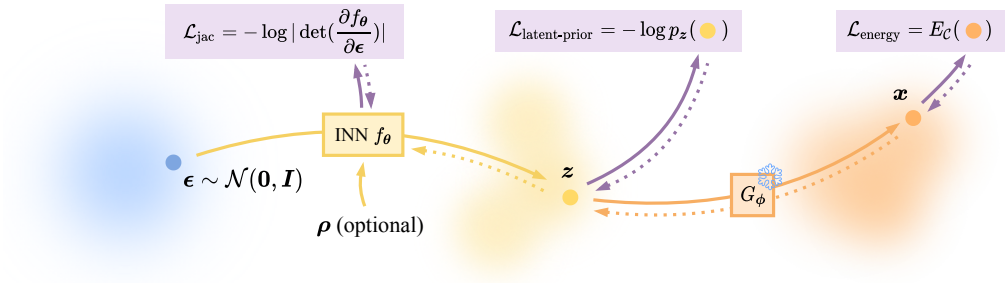


Figure 3: Illustration for Algorithm 2. In the forward pass (solid curves), we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, map $\boldsymbol{\epsilon}$ to latent code $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ with an INN $f_{\boldsymbol{\theta}}$, and map $\boldsymbol{z}$ to an image $\boldsymbol{x} = G(\boldsymbol{z})$ with a fixed generative model G. Dashed curves show the gradients. Details are provided in Section 3.2.

Algorithm 2: Approximating Latent-Space EBM with INN

while *not converged* **do**

1. Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
2. Map $\boldsymbol{\epsilon}$ to latent code $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$
3. Map $\boldsymbol{z}$ to an image $\boldsymbol{x} = G(\boldsymbol{z})$
4. Optimize $\boldsymbol{\theta}$ with gradient $\nabla_{\boldsymbol{\theta}}\Big(-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}}(\boldsymbol{x})\Big)$

Based on these properties of INN, we can rewrite our KL divergence objective $\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z})\|p(\boldsymbol{z}|\mathcal{C}))$ as (full derivations in Appendix B.3) the following form:

$$\begin{aligned}\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z})\|p(\boldsymbol{z}|\mathcal{C})) &= \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}(\boldsymbol{z}),\boldsymbol{x}=G(\boldsymbol{z})}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z})}{p_{\boldsymbol{z}}(\boldsymbol{z})e^{-E_{\mathcal{C}}(\boldsymbol{x})}/Z}\Big] \\ &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}),\boldsymbol{x}=G(\boldsymbol{z})}\Big[-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}}(\boldsymbol{x})\Big] - \mathbb{H}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})} + \log Z.\end{aligned} \tag{9}$$

Since $\mathbb{H}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})}$ and $\log Z$ are independent of $\boldsymbol{\theta}$, our training objective becomes

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}),\boldsymbol{x}=G(\boldsymbol{z})}\Big[-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}}(\boldsymbol{x})\Big]. \tag{10}$$

Figure 3 gives an illustration of our process, and Algorithm 2 describes the algorithmic details.

PromptGen for generative models with a class-embedding space (Figure 5(d)) Previous works [4, 61] have shown that *class conditioning* boosts generative models' performances on diverse data such as ImageNet [59]. Specifically, class-conditioned generative models map a latent code $\boldsymbol{z}$ and a class embedding $\boldsymbol{y}$ to $\boldsymbol{x} = G(\boldsymbol{z}, \boldsymbol{y})$. To extend PromptGen to these models, we train an INN $h_{\boldsymbol{\theta}}$ to map $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\boldsymbol{I})$ to $\boldsymbol{y} = h_{\boldsymbol{\theta}}(\boldsymbol{\xi})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation of G's class embeddings. The motivation for defining the distribution of $\boldsymbol{\xi}$ as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\boldsymbol{I})$ but not $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is that we want the learned INN $h_{\boldsymbol{\theta}}$ to be volume-preserving, which is easier to train. The training objective is to minimize $\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{y})\|p(\boldsymbol{z}, \boldsymbol{y}|\mathcal{C}))$, which is equivalent to (full derivations in Appendix B.4)

$$\begin{aligned}\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\boldsymbol{\xi}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I}),\boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}),\boldsymbol{y}=h_{\boldsymbol{\theta}}(\boldsymbol{\xi}),\boldsymbol{x}=G(\boldsymbol{z},\boldsymbol{y})}\Big[&-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) \\ &- \log|\det(\frac{\partial h_{\boldsymbol{\theta}}}{\partial \boldsymbol{\xi}})| - \log p_{\boldsymbol{y}}(\boldsymbol{y}) + E_{\mathcal{C}}(\boldsymbol{x})\Big],\end{aligned} \tag{11}$$

where $p_{\boldsymbol{y}}(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\boldsymbol{I})$ is the estimated class-embedding distribution.

PromptGen with conditional INN When the control is specified by continuous values, e.g., the scene parameters $\boldsymbol{\rho}$ in Eq. (4), it is desirable to condition the INN on $\boldsymbol{\rho}$ because it helps in generalizing to all continuous values. Specifically, the conditioning is implemented by simply replacing $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ with $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \boldsymbol{\rho})$. Full derivations are in Appendix B.5. Note that PromptGen with conditional INN results in a similar architecture to StyleFlow [1], but StyleFlow [1] uses MLE training on image-label pairs and is only applicable to explicit conditioning. Therefore, StyleFlow is not capable of modeling EBMs, which are required for CLIP control and moment constraint.

(a) StyleCLIP [54] (b) StyleGAN2-NADA [17] (c) PromptGen (ours) w/ StyleGAN2

Figure 4: Image synthesis based on text description, guided by the CLIP model. As the pre-trained generative model, we use StyleGAN2 trained on FFHQ 1024^2 [34] with truncation $\psi = 0.7$. The text description used in this experiment is a photo of a baby. StyleCLIP requires optimization at inference, while StyleGAN2-NADA and PromptGen do not. All images are resized for visualization.

(a) PromptGen w/ StyleNeRF (text: a photo of a baby) (b) PromptGen w/ NVAE (text: a photo of a baby) (c) PromptGen w/ StyleGAN2 (text: a photo of a British shorthair)

(d) PromptGen w/ BigGAN trained on ImageNet 512^2, truncation $\psi = 0.4$

Figure 5: PromptGen is applicable to different generative models in various domains. Figure 5(a) and Figure 5(b) are PromptGen applied to StyleNeRF [21] and NVAE [71] for baby face synthesis. Figure 5(c) is PromptGen applied to StyleGAN2 [35] trained on AFHQ-Cats [6]. Figure 5(d) is the extension to BigGAN [4] trained on ImageNet [59]. All images are resized for visualization. See Appendix C for results on more datasets and text descriptions.

4 Experiments

This section describes the experimental validation of PromptGen. See Appendix A for experiments on synthetic data and Appendix C, D, E, and F for additional experiments on images and 3D meshes. Additional experimental details are provided in Appendix B.8.

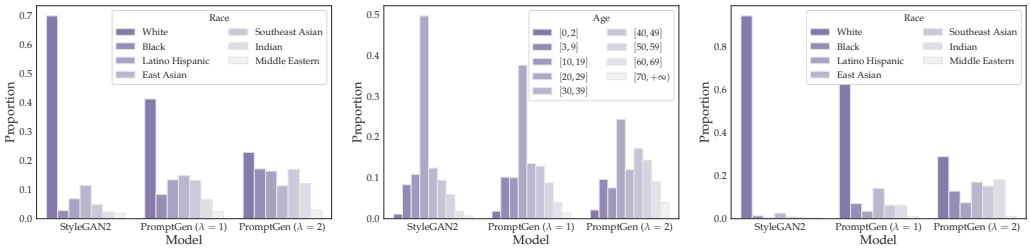
4.1 Image Synthesis based on Text Description

This experiment illustrates the capability of PromptGen to sample images from a generative model driven by a text description $\boldsymbol{y}$ using the pre-trained CLIP model [55]. We used the CLIP energy from Eq. (3), with text descriptions such as a photo of a baby.

Figure 4 shows a comparison between PromptGen and two previous CLIP-guided image generation methods, StyleCLIP [54] and StyleGAN2-NADA [17]. All methods used StyleGAN2 as the backbone. We observe that PromptGen generates diverse and high-quality images of babies, while StyleCLIP struggles in controllability and image quality, and StyleGAN2-NADA generates baby-like adults.[2]

Figure 5(a) and Figure 5(b) display how our PromptGen framework applies to StyleNeRF [21], a NeRF-based GAN model, and NVAE [71], a deep variational autoencoder model. We observe that PromptGen with StyleNeRF (Figure 5(a)) has higher diversity than PromptGen with StyleGAN2 (Figure 4(c)), and PromptGen with NVAE has lower image quality. Figure 5(c) shows the ability of PromptGen to generate cat faces based on the text description. Figure 5(d) shows the results for the extension (Section 3.2) of our PromptGen framework to BigGAN [4], a class-conditional GAN trained on ImageNet [59] with a class-embedding space. Text descriptions used for this experiment are shown below each set of images. We observe that PromptGen helps BigGAN generate images with complex text descriptions, which are out of BigGAN's pre-training image distribution.

[2]Since we care about distributions, ***none*** *of the images in this paper are cherry- or lemon-picked.*



(a) Race dist. (FFHQ) (b) Age dist. (FFHQ) (c) Race dist. (MetFaces)

Figure 6: With the moment constraint, PromptGen de-biases StyleGAN2 (on FFHQ and MetFaces 1024^2, truncation $\psi = 0.7$). See Appendix D for image samples and Table 5 for quantitative results.

Table 2: Comparison with baselines for de-biasing binary attributes and their correlations. Following [30], we use classifiers on CelebA [45]. Baseline performances are copied from [30]. PromptGen has competitive performance, even in the cases where FairStyle achieves nearly perfect performance.

	FFHQ (binary attribute)			
	$\mathbb{D}_{\text{KL}}^{\text{gender}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{age+gender}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{age+eyeglasses}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{gender+eyeglasses}}\downarrow$
FFHQ (real data)	0.015	0.246	0.355	0.242
StyleGAN2 [35]	0.018	0.279	0.384	0.250
StyleFlow [1]	0.023	0.214	0.162	0.121
FairGen [68]	4.21×10^{-4}	0.0373	0.0330	0.00185
FairStyle [30]	$\mathbf{3.20} \times 10^{-7}$	0.0257	0.0157	**0.000241**
PromptGen (ours)	1.71×10^{-5}	**0.000558**	**0.000415**	0.000628

4.2 De-biasing Generative Models

An important problem in generative models is to generate fair distributions w.r.t a set of attributes of interest. For instance, Figure 6(a) shows that StyleGAN2 generates images with bias across races and ages. PromptGen de-biases StyleGAN2 models trained on FFHQ 1024^2 and MetFaces 1024^2 [32] in terms of *categorical* attributes, using the moment constraint defined in Eq. (5) and Eq. (6). As control, we used a classifier trained on FairFace 224^2 [31] as $\boldsymbol{\gamma}$. We defined $\boldsymbol{\mu} = \left(|\mathcal{A}|^{-1}, \dots, |\mathcal{A}|^{-1}\right)$, where $\mathcal{A}$ is the set of races. Similar to the energy weights λ_i defined in Eq. (2), we propose to rescale the trained $\hat{\boldsymbol{\beta}}$ as $\lambda\hat{\boldsymbol{\beta}}$. Figure 6 visualizes the attribute distribution before and after PromptGen de-biasing, which shows that PromptGen de-biases the race and age distributions effectively.

Existing de-biasing baselines consider *binary* attributes [30]. For a fair comparison with them, we adopted their setting to use binary classifiers trained on CelebA [45] for de-biasing and evaluation. Since classifiers trained on CelebA also suffer from the spurious correlation between attributes, we did *not* use the moment constraint for this experiment. Instead, since PromptGen allows conditional image generation with the classifier energy, we generated the same number of samples conditioned on each attribute or attribute combination. Table 2 shows that PromptGen has competitive performance on de-biasing for attributes and attribute combinations. In cases where the previous best FairStyle has nearly perfect performance, the margin between PromptGen and FairStyle is only a few samples.

4.3 Pose-Guided Face Synthesis

With an inverse graphics model, PromptGen can control the pose of faces generated by StyleGAN2 trained on FFHQ 1024^2. Specifically, we used the DECA model [16], which infers the parameters of FLAME [42], a parametric facial graphics model with six pose parameters. We set $\boldsymbol{\rho} \in \text{SO}(3)$ as FLAME's three neck poses (but we kept the z-axis rotation fixed since FFHQ does not contain such variation) and used the conditional INN extension introduced in Section 3.2. To enable generating different poses of the same identity (ID), we propose an additional ID energy using the IR-SE50 model [8]. Specifically, given a canonical pose $\boldsymbol{\rho}_0$, we define $\boldsymbol{z}_0 = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \boldsymbol{\rho}_0)$, and the ID energy is defined as (detailed in Appendix B.7)

$$E_{\text{ID}}(\boldsymbol{x}_0, \boldsymbol{x}) = 1 - \cos\left\langle R(\boldsymbol{x}_0), R(\boldsymbol{x})\right\rangle, \quad \boldsymbol{x}_0 = G(\boldsymbol{z}_0), \boldsymbol{x} = G(\boldsymbol{z}), \tag{12}$$

(a) Using the DECA model [16] and scene parameters $\boldsymbol{\rho}_{\text{pose}} \in \text{SO}(3)$, PromptGen controls StyleGAN2.

(b) Samples from models that are explicitly 3D, copied from [21]

Figure 7: Using inverse graphics model, PromptGen samples faces in different poses from StyleGAN2 (trained on FFHQ 1024^2, truncation $\psi = 0.7$) while preserving the identity. All images are resized for visualization.

Table 3: Pose-controlled face generation. In this experiment, PromptGen uses StyleGAN2 as the pre-trained generative model. Results for GRAF [62], pi-GAN [5], GIRAFFE [51], and StyleNeRF [21] are from [21]; results for DiscoFaceGAN (DFG) [9] and GAN-Control [65] are from [65].

	StyleGAN2	GRAF	pi-GAN	GIRAFFE	StyleNeRF	DFG	GAN-Control	PromptGen
Resolution	1024^2	256^2	256^2	256^2	1024^2	256^2	512^2	1024^2
Pose	✗	✓	✓	✓	✓	✓	✓	✓
FID↓	3	71	85	35	8	13	6	4
Dist. w/ same ID↓	–	–	–	–	–	0.83	0.68	0.45
Dist. w/ diff. ID	–	–	–	–	–	1.73	1.90	1.37

where R is the IR-SE50 model [8] that computes face embeddings. Figure 7 shows that PromptGen generates faces of the same ID in different poses, even without being explicitly trained with poses as condition. We reported the FID score [23] of each model, computed with Clean-FID [53]; following [35], we did not use the truncation trick when computing the FID score. Table 3 shows that PromptGen outperforms existing models in terms of the FID score. Following [65], we then reported the average IR-SE50 [8] embedding distances for images with the same ID and with different IDs. Results show that PromptGen achieves the best ID preservation, with a slight sacrifice of ID diversity.

4.4 Iterative Distributional Control via Functional Composition

This section discusses an interesting bias that our PromptGen reveals about the CLIP model. Figure 8(a) shows images generated by PromptGen (with StyleGAN2) using the CLIP model with the

(a) PromptGen (iteration 1). Image synthesis with text description a photo of a person without makeup. Gender distribution: female: 81.6%; male: 18.4%.

(b) PromptGen (iteration 2). Gender de-biasing ($\lambda = 2$) for the distribution learned in iteration 1. Gender distribution: female: 49.3%; male: 50.7%.

Figure 8: A curious case of the CLIP model: with description a photo of a person without makeup, PromptGen generates more female images than male, showing that CLIP learns a "reporting bias" (i.e., people are more likely to *say* "a person without makeup" when the person is a female). Besides revealing this "reporting bias", PromptGen can also mitigate this bias via an iterative control, allowing us to de-bias the text-controlled distribution. All images are resized for visualization.

text description a photo of a person without makeup, where more females are generated than males. This bias should not be attributed to *image pre-training data* because images contain a bias in the opposite direction, i.e., men are less likely to have makeup. We argue that this bias should be explained by CLIP having learned a "reporting bias" in the *multi-modal (vision-language) pre-training data*: people are more likely to *say* "a person without makeup" when the person is a female.

Besides revealing the above "reporting bias", PromptGen can also mitigate this bias via an iterative distributional control, enabled by the functional composition in Algorithm 1. Specifically, in the second iteration, we de-biased the gender distribution of $G \circ f_{\boldsymbol{\theta}}$ instead of G, where $f_{\boldsymbol{\theta}}$ is the INN learned for the text control. For de-biasing, we used the moment constraint, where $\hat{\boldsymbol{\beta}}$ is trained based on $G \circ f_{\boldsymbol{\theta}}$. Figure 8(b) shows that females and males are uniformly distributed after the de-biasing control in the second iteration, and all generated images do not have makeup, as originally requested.

4.5 Inference Latency

We compared our PromptGen and the plug-and-play generative model (PPGM) [47] in terms of the inference latency. A difference between the two methods is that PPGM uses Langevin dynamics [72] to optimize over the latent space during inference, while PromptGen samples images in a feed-forward manner. In this experiment, we used PPGM and our PromptGen to approximate the same EBM with CLIP energy. Inference times in Table 4 was estimated on an NVIDIA RTX A4000 GPU. Table 4 shows that our PromptGen has the highest performance and efficiency. Compared to our PromptGen, PPGM requires $100\times$ inference time to achieve comparable results.

Table 4: Comparison between PromptGen and PPGM, when approximating the *same* EBM with CLIP energy. n is the number of inference-time optimization steps used by PPGM. All models used StyleGAN2 as the pre-trained generative model. The text descriptions are a photo of a {baby, boy, girl}. Inference time was estimated on an NVIDIA RTX A4000 GPU.

	PPGM ($n = 10$)	PPGM ($n = 50$)	PromptGen (ours)
CLIP energy (baby)↓	0.7327	0.7134	**0.7038**
CLIP energy (girl)↓	0.7257	**0.7184**	0.7199
CLIP energy (boy)↓	0.7263	0.7114	**0.7081**
Inference time per sample (sec.)↓	4.4	21.5	**0.2**
Back-propagation through CLIP at inference	10	50	0
When is it equal to the EBM?	$n \to \infty$	$n \to \infty$	$\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z}) \| p(\boldsymbol{z}\|\mathcal{C})) = 0$

5 Conclusions and Future Work

This paper proposes PromptGen, a unified framework to learn latent distributions for distributional control of pre-trained generative models. PromptGen leverages the knowledge of various off-the-shelf models and, unlike previous methods, it does not require the availability of these models at inference. PromptGen can sample images in a feed-forward manner, which is more efficient than methods that require optimization at inference. PromptGen offers the generality for algorithmic design and modularity for control composition, and it also enables iterative controls. Experiments validate that PromptGen applies to various generative models (StyleGAN2, StyleNeRF, NVAE), control types (continuous, discrete, and moment constraint), off-the-shelf models (CLIP, classifiers, and inverse graphics models), and data domains (faces, churches, animals, ImageNet, and landscapes).

Limitation and future work We provide an error analysis in Appendix F. Since PromptGen builds upon pre-trained generative models, it is inevitably restricted by the original generative model's coverage [3, 27]. Except for the case of generative models with a class-embedding space, PromptGen focuses on mode-seeking and mode-reweighting instead of domain adaptation. One possible direction is to fine-tune the pre-trained generative model [17] after PromptGen learning to achieve domain adaptation. Also, the control performance depends on the nature of off-the-shelf models that provide knowledge about the control. PromptGen helps us understand these off-the-shelf models, e.g., the CLIP model spuriously correlates female with the mention of "without makeup" (Figure 8(a)), and we also show that it has difficulty in modeling linguistic negation (Appendix F). In subsequent work, it can be beneficial to explore learnable energy functions [15, 26] (besides our moment constraint for de-biasing), which may provide fine-grained control with less bias.

References

[1] Rameen Abdal, Peihao Zhu, Niloy Jyoti Mitra, and Peter Wonka. StyleFlow: Attribute-conditioned exploration of StyleGAN-generated images using conditional continuous normalizing flows. *TOG*, 2021.

[2] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and U. Köthe. Guided image generation with conditional invertible neural networks. *ArXiv*, 2019.

[3] David Bau, Jun-Yan Zhu, Jonas Wulff, William S. Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a GAN cannot generate. *ICCV*, 2019.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *ICLR*, 2019.

[5] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. *CVPR*, 2021.

[6] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. *CVPR*, 2020.

[7] Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Found. Trends Commun. Inf. Theory*, 2004.

[8] Jiankang Deng, J. Guo, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. *CVPR*, 2019.

[9] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3D imitative-contrastive learning. *CVPR*, 2020.

[10] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. *NeurIPS*, 2021.

[11] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *ICLR, Workshop Track Proceedings*, 2015.

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *ICLR*, 2017.

[13] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *NeurIPS*, 2020.

[14] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *NeurIPS*, 2019.

[15] Jesse H. Engel, Matthew D. Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. *ICLR*, 2018.

[16] Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. *TOG*, 2021.

[17] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. StyleGAN-NADA: CLIP-guided domain adaptation of image generators. *ArXiv*, 2021.

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

[19] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *ICLR*, 2020.

[20] Aditya Grover, Kristy Choi, Rui Shu, and Stefano Ermon. Fair generative modeling via weak supervision. *ICML*, 2020.

[21] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. StyleNeRF: A style-based 3D aware generator for high-resolution image synthesis. *ICLR*, 2022.

[22] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. GANSpace: Discovering interpretable GAN controls. *NeurIPS*, 2020.

[23] 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. *NIPS*, 2017.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[25] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *NeurIPS Workshop*, 2021.

[26] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, Xiaodan Liang, Lianhui Qin, Haoye Dong, and Eric P. Xing. Deep generative models with learnable knowledge constraints. *NeurIPS*, 2018.

[27] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. *ECCV*, 2020.

[28] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. MaGNET: Uniform sampling from deep generative network manifolds without retraining. *ICLR*, 2022.

[29] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser Nam Lim. Visual prompt tuning. *ArXiv*, 2022.

[30] Cemre Karakas, Alara Dirik, Eylül Yalçınkaya, and Pinar Yanardag. FairStyle: Debiasing StyleGAN2 with style channel manipulations. *ArXiv*, 2022.

[31] Kimmo Kärkkäinen and Jungseock Joo. FairFace: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. *WACV*, 2021.

[32] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *NeurIPS*, 2020.

[33] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *NeurIPS*, 2021.

[34] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019.

[35] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. *CVPR*, 2020.

[36] Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to controlled text generation. *ICLR*, 2021.

[37] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *NeurIPS*, 2018.

[38] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *ICLR*, 2014.

[39] Yann LeCun, Sumit Chopra, Raia Hadsell, Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. *Tutorial*, 2006.

[40] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *EMNLP*, 2021.

[41] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. Controllable text-to-image generation. *NeurIPS*, 2019.

[42] Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *TOG*, 2017.

[43] Nan Liu, Shuang Li, Yilun Du, Joshua B. Tenenbaum, and Antonio Torralba. Learning to compose visual relations. *NeurIPS*, 2021.

[44] Xingchao Liu, Chengyue Gong, Lemeng Wu, Shujian Zhang, Haoran Su, and Qiang Liu. FuseDream: Training-free text-to-image generation with improved CLIP+GAN space optimization. *ArXiv*, 2021.

[45] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

[46] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. *CVPR*, 2020.

[47] Anh M Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. *CVPR*, 2017.

[48] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *ArXiv*, 2021.

[49] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *ICML*, 2021.

[50] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. *NeurIPS*, 2021.

[51] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. *CVPR*, 2021.

[52] Frank Noé, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 2019.

[53] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in GAN evaluation. *CVPR*, 2022.

[54] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and D. Lischinski. StyleCLIP: Text-driven manipulation of StyleGAN imagery. *ICCV*, 2021.

[55] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *ICML*, 2021.

[56] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text Transformer. *JMLR*, 2020.

[57] Vikram V. Ramaswamy, Sunnie S. Y. Kim, and Olga Russakovsky. Fair attribute classification through latent space de-biasing. *CVPR*, 2021.

[58] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Manfred Otto Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *ArXiv*, 2022.

[59] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

[60] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. *ArXiv*, 2021.

[61] Axel Sauer, Katja Schwarz, and Andreas Geiger. StyleGAN-XL: Scaling StyleGAN to large diverse datasets. *ArXiv*, 2022.

[62] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. *NeurIPS*, 2020.

[63] Vikash Sehwag, Caner Hazirbas, Albert Gordo, Firat Ozgenel, and Cristian Canton Ferrer. Generating high fidelity data from low-density regions using diffusion models. *ArXiv*, 2022.

[64] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. InterFaceGAN: Interpreting the disentangled face representation learned by GANs. *TPAMI*, 2022.

[65] Alon Shoshan, Nadav Bhonker, Igor Kviatkovsky, and Gérard Medioni. GAN-Control: Explicitly controllable GANs. *ICCV*, 2021.

[66] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. *ICCV*, 2021.

[67] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 2019.

[68] Shuhan Tan, Yujun Shen, and Bolei Zhou. Improving the fairness of deep generative models without retraining. *ArXiv*, 2020.

[69] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. *ICML*, 2008.

[70] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *NeurIPS*, 2021.

[71] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. *NeurIPS*, 2020.

[72] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. *ICML*, 2011.

[73] Jay Whang, Erik M. Lindgren, and Alexandros G. Dimakis. Composing normalizing flows for inverse problems. In *ICML*, 2021.

[74] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN inversion: A survey. *ArXiv*, 2021.

[75] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *ArXiv*, 2022.

[76] Haotian Yang, Hao Zhu, Yanru Wang, Mingkai Huang, Qiu Shen, Ruigang Yang, and Xun Cao. FaceScape: A large-scale high quality 3D face dataset and detailed riggable 3D face prediction. *CVPR*, 2020.

[77] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *ArXiv*, 2015.

[78] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. *NeurIPS*, 2020.

[79] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *ArXiv*, 2021.

[80] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. *CVPR*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   In Section 5 and Appendix F.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   In Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   In Section 3 and Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes]
   In Section 3 and Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   We will make our code publicly available on GitHub, with instruction for each experiment. We made great efforts to ensure that most experiments will be run in one line of command.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   In Appendix B.8; other training details will be included in the code release.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
   In Appendix B.8.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   We have cited the original papers.
   (b) Did you mention the license of the assets? [Yes]
   License will be included in the code.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   We do not have new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   All data we use are published datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
   All data we use are published datasets.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   We do not have user study.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   We do not have user study.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
   We do not have user study.