Generative Visual Prompt: Unifying Distributional Control of Pre-Trained Generative Models

Chen Henry Wu, Saman Motamed, Shaunak Srivastava, Fernando De la Torre
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA
{chenwu2,ftorre}@cs.cmu.edu, {saman.moatamed,shaunak1999}@gmail.com

Abstract

Generative models (e.g., GANs, diffusion models) learn the underlying data distribution in an unsupervised manner. However, many applications of interest require sampling from a particular region of the output space or sampling evenly over a range of characteristics. For efficient sampling in these scenarios, we propose **Generative Visual Prompt** (PromptGen), a framework for distributional control over pre-trained generative models by incorporating knowledge of other off-the-shelf models. PromptGen defines control as energy-based models (EBMs) and samples images in a feed-forward manner by approximating the EBM with invertible neural networks, avoiding optimization at inference. Our experiments demonstrate how PromptGen can efficiently sample from several unconditional generative models (e.g., StyleGAN2, StyleNeRF, diffusion autoencoder, NVAE) in a controlled or/and de-biased manner using various off-the-shelf models: (1) with the CLIP model as control, PromptGen can sample images guided by text, (2) with image classifiers as control, PromptGen can de-bias generative models across a set of attributes or attribute combinations, and (3) with inverse graphics models as control, PromptGen can sample images of the same identity in different poses. (4) Finally, PromptGen reveals that the CLIP model shows a "reporting bias" when used as control, and PromptGen can further de-bias this controlled distribution in an iterative manner.[1]

1 Introduction

Generative models learn the underlying high-dimensional data distribution and have achieved promising performance on image synthesis [4, 75, 36, 21]. Though being well praised, they still face two main limitations. 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 [60, 31]. 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 [23, 68, 31] and explicit conditions [43, 69]. 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 [58], 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 weights fixed. Given its conceptual similarity to prompting [42, 85, 86, 30, 22], we term our framework as Generative Visual Prompt (PromptGen). PromptGen requires **no training data**, and the only supervision comes from off-the-shelf models that help define the control. Specifically,

[1]The code is available at https://github.com/ChenWu98/Generative-Visual-Prompt.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

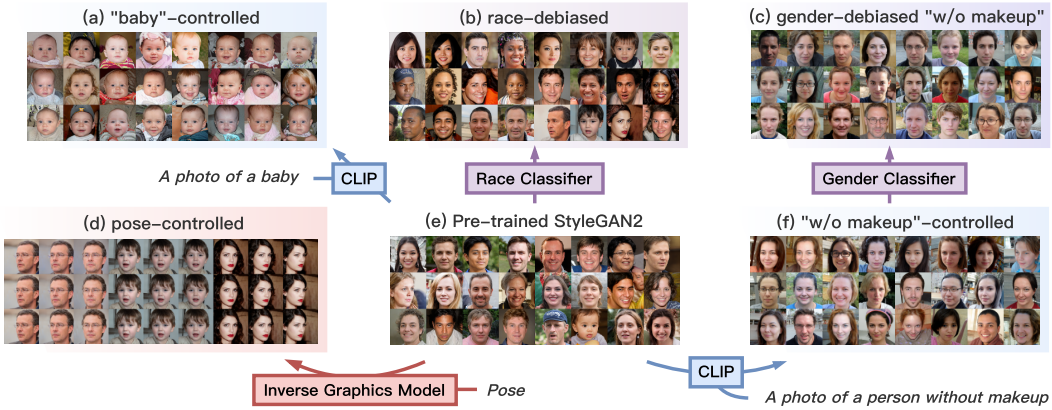


Figure 1: PromptGen uses different off-the-shelf models (e.g., CLIP [58], inverse graphics models, and classifiers) to control the output distribution of a pre-trained and fixed generative model (e.g., StyleGAN2). Colored boxes (e.g., the blue CLIP) indicate the control. See the text for details.

PromptGen allows the user to define controls using an energy-based model (EBM) approximated by an invertible neural network (INN). Unlike methods that require optimization at inference in EBM sampling [49, 14, 45, 52], 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 [35] without any labels, while we also show results for StyleNeRF [21], diffusion autoencoder [57], and NVAE [75] 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 [58], 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 [49] / LACE [52]	Guided DDPM [10]	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 [40], invertible neural networks [11, 12], and diffusion models [25, 71]. In particular, generative models trained on large amounts of unlabeled data, e.g., BigGAN [4], StyleGANs [35, 36, 34, 64], Glow [39], and diffusion models [51, 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 [43, 69] or local editing of the learned representation, e.g, "style" codes [1, 56, 46, 68, 80]. For de-biasing, existing methods include local editing of "style" codes [60, 31] and importance sampling for either training [20] or inference [29]. 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 tasks have been shown by a recent trend of works in multiple research areas across vision and language [59, 48, 63, 81, 61]. Moreover, one can fine-tune generative models [17] for domain adaptation, which is orthogonal to PromptGen: since PromptGen maps a generative model to another generative model (Algorithm 1), fine-tuning can be applied before or after PromptGen training. We leave this exploration to future studies.

Previous methods usually sample from EBM [41] with Markov Chain Monte Carlo (MCMC) [73, 76, 14, 19, 13, 45]. Among them, plug-and-play generative models (PPGMs) [49] and LACE [52] 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 [51, 67] or not [25, 26, 50]. 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 [54] to sample from physical systems, which is later used by [78] to solve inverse problems with two composed INNs. Notably, [54] and [78] do not leverage a low-dimensional latent space, and [78] requires training a separate model for each image sample. In this paper, we use INN to model arbitrary EBMs for various generative models.

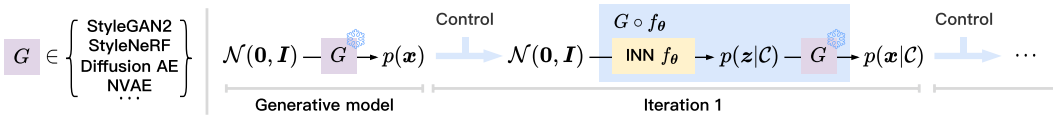


Figure 2: Overview of Generative Visual Prompt (PromptGen). Given a pre-trained generative model G and a control $\mathcal{C}$, we learn a distribution $p(\boldsymbol{z}|\mathcal{C})$ in G's latent space, while keeping G fixed. Each control $\mathcal{C}$ can have multiple components, e.g., $\mathcal{C} = \{$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.

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: Generative Visual Prompt (PromptGen)

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 $G : \mathbb{R}^d \to \mathcal{X}$, which is a feed-forward network

3.1 Latent-Space EBM for Distributional Control

The plug-and-play generative model [49] 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 [52]. We define the latent-space EBM following similar formulation as [49, 19, 52], with some new energy functions. We then extend the formulation

to incorporate the moment constraint [7], which was adopted for language modeling [37], but unlike [37], we define the moment constraint in the latent space to accommodate generative vision models.

We define a control $\mathcal{C}$ as M independent properties $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$, 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

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

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 [58], we define the CLIP energy as the cosine distance between the embeddings of the image and the text $\boldsymbol{t}$, averaged over L differentiable augmentations [84, 46]:

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

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, 37] 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

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

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 $\boldsymbol{\mu} = \big(|\mathcal{A}|^{-1}, \ldots, |\mathcal{A}|^{-1}\big)$ where $\mathcal{A}$ is the set of races. It means that we would like to find a distribution that 1) stays close to the original, pre-trained generative model G and 2) mitigates G's bias. In this paper, we generalize the moment constraint to the latent space, and approximate the above objective as (detailed in Appendix B.6):

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

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

Notably, to optimize $\hat{\boldsymbol{\beta}}$ in Eq. (7), we leverage existing gradient-based optimization tools such as stochastic gradient descent (SGD) and Adam [38].

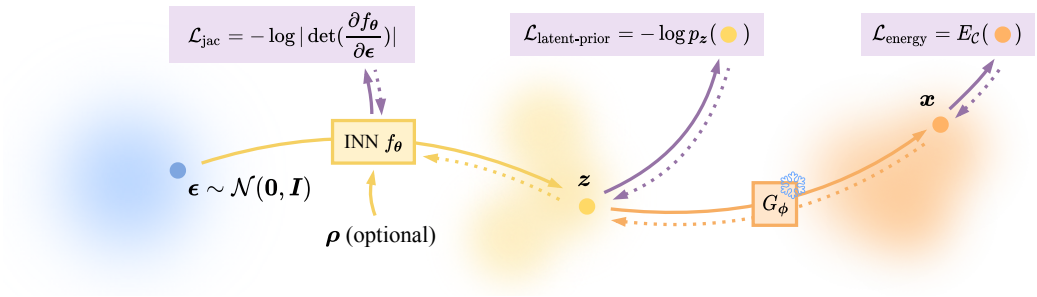


Figure 3: Illustration for Algorithm 2. In the forward pass (solid curves), we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, map $\boldsymbol{\epsilon}$ to latent code $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ with an INN $f_{\boldsymbol{\theta}}$, 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 $\nabla_{\boldsymbol{\theta}}\Big(-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}}(\boldsymbol{x})\Big)$

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 [54], 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 $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and mapping it to $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$, and (2) $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ has a closed-form solution:

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

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

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

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

PromptGen in a class-embedding space (Figure 5(e)) Previous works [4, 64] have shown that *class conditioning* boosts generative models' performances on ImageNet [62]. 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):

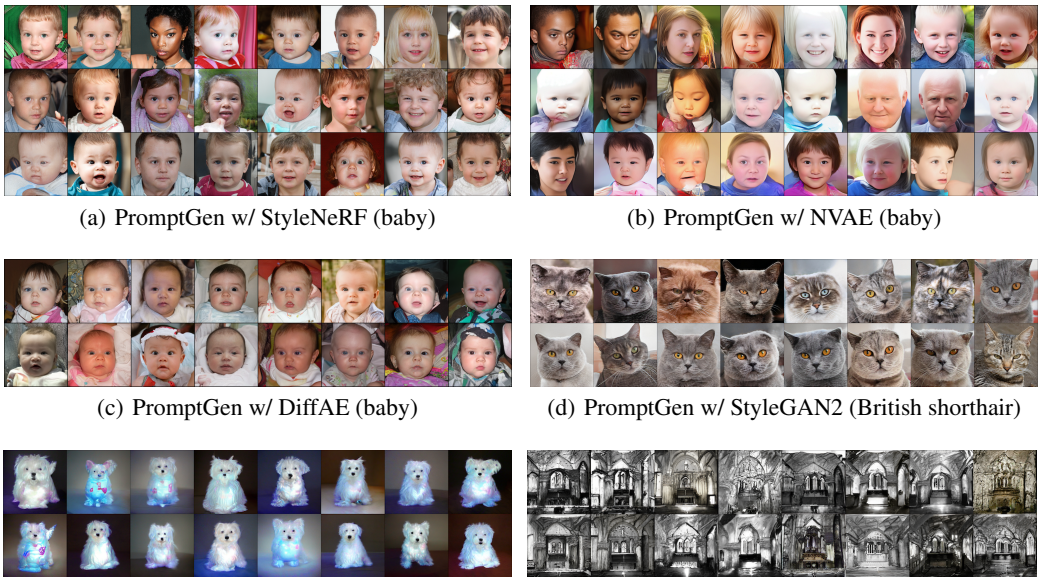
$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\boldsymbol{\xi}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I}),\boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}),\boldsymbol{y}=h_{\boldsymbol{\theta}}(\boldsymbol{\xi}),\boldsymbol{x}=G(\boldsymbol{z},\boldsymbol{y})}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

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 To generalize the control to continuous values, e.g., the scene parameters $\boldsymbol{\rho}$ in Eq. (4), we condition the INN on $\boldsymbol{\rho}$. The condition is modeled by replacing $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ with $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \boldsymbol{\rho})$. Space limited, we provide details and derivations in Appendix B.5. This extension results in a similar architecture to StyleFlow [1], but StyleFlow [1] uses MLE training on image-label pairs and is only applicable to explicit condition, which is not capable of modeling EBMs. When the condition is modeled by the equivariant operations proposed by [79], PromptGen also satisfies the homomorphism property [79] in the latent space.

(a) StyleCLIP [56] (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 [35] 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 (baby) (b) PromptGen w/ NVAE (baby)

(c) PromptGen w/ DiffAE (baby) (d) PromptGen w/ StyleGAN2 (British shorthair)

(e) PromptGen w/ BigGAN trained on ImageNet 512^2

Figure 5: PromptGen is applicable to different generative models in various domains. Figure 5(a), Figure 5(c), and Figure 5(b) are PromptGen applied to StyleNeRF [21], diffusion autoencoder [57], and NVAE [75]. Figure 5(d) is PromptGen applied to StyleGAN2 [36] trained on AFHQ-Cats [6]. Figure 5(e) is the extension to the embedding space of BigGAN [4] on ImageNet [62]. All images are resized for visualization. See Appendix C for results on more datasets and text descriptions.

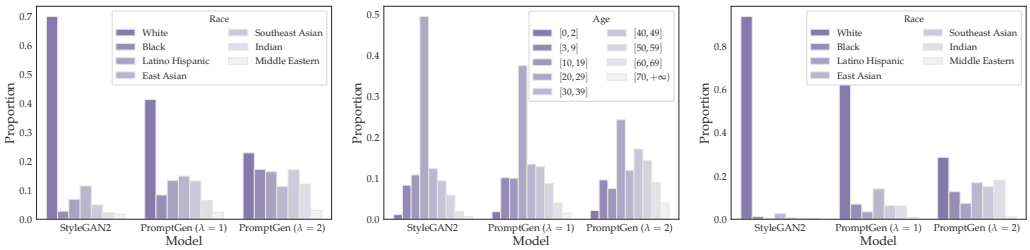
4 Experiments

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 $\boldsymbol{t}$ using the pre-trained CLIP model [58] (the ViT-B/32 version). 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 [56] and StyleGAN2-NADA [17]. 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] These results show that (1) locally editing the latent code (i.e., StyleCLIP) is not always an effective method for controlling generative models (e.g., not all images' latent code can be locally edited into a baby); (2) domain adaptation (i.e., StyleGAN-NADA) is not effective in seeking modes in a generative model. Figure 5(a), Figure 5(b), and Figure 5(c) show how PromptGen applies to StyleNeRF [21], NVAE [75], and diffusion autoencoder (DiffAE; [57]). Although these generative models are all trained on FFHQ, baby images sampled from them have distinct characteristics. Figure 5(d) shows the ability of PromptGen to generate cats of a particular species. Figure 5(e) shows the extension (Section 3.2) of our PromptGen to the class-embedding space of BigGAN [4], a class-conditional GAN trained on ImageNet [62]. We observe that PromptGen helps BigGAN generate images with complex text descriptions, which are out of BigGAN's training image distribution; however, the diversity seems to be limited in this extension.



(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.

4.2 De-Biasing Pre-Trained 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 [33] 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 [32] 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 shows that PromptGen de-biases the race and age effectively.

Existing de-biasing baselines consider *binary* attributes [31]. For a fair comparison with them, we adopted their setting to use binary classifiers trained on CelebA [47] 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.

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

Table 2: Comparison with baselines for de-biasing binary attributes and their correlations. Following [31], we use classifiers on CelebA [47]. Baseline performances are copied from [31]. PromptGen has competitive performance, even in the cases where FairStyle achieves nearly perfect performance.

	FFHQ (binary attributes)					
	$\mathbb{D}_{\text{KL}}^{\text{gender}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{eyeglasses}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{blond hair}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{age+gender}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{age+eyeglasses}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{gender+eyeglasses}}\downarrow$
FFHQ (real data)	0.015	0.186	–	0.246	0.355	0.242
StyleGAN2 [36]	0.018	0.180	–	0.279	0.384	0.250
StyleFlow [1]	0.023	0.061	–	0.214	0.162	0.121
FairGen [72]	4.21 $\times 10^{-4}$	7.07 $\times 10^{-4}$	–	0.0373	0.0330	0.00185
FairStyle [31]	**3.20** $\times 10^{-7}$	**0**	–	0.0257	0.0157	**0.000241**
PromptGen (ours)	1.71 $\times 10^{-5}$	1.72 $\times 10^{-5}$	0.0008	**0.000558**	**0.000415**	0.000628

Figure 7: Using inverse graphics model DECA [16] and scene parameters $\boldsymbol{\rho}_{\text{pose}} \in \text{SO}(3)$, PromptGen controls the pose of StyleGAN2 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 [66], pi-GAN [5], GIRAFFE [53], and StyleNeRF [21] are from [21]; results for DiscoFaceGAN (DFG) [9] and GAN-Control [69] are from [69].

	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

4.3 Pose-Guided Face Synthesis

With an inverse graphics model, PromptGen can control the pose of faces generated by StyleGAN2. We used the DECA model [16], which infers the parameters of FLAME [44], a parametric facial graphics model. We set $\boldsymbol{\rho} \in \text{SO}(3)$ as FLAME's three neck poses and used the conditional INN extension introduced in Section 3.2. To enable generating different poses of the same identity (ID), we propose 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)

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

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 conditions. We computed the FID score [24] of each model using Clean-FID [55]; following [36], 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 [69], 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 Decomposing Complex Control via Energy Composition

Energy composition $E_{\mathcal{C}}(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i)$ in Eq. 1 allows us to decompose a complex controls into simple ones. For instance, Figure 8(a) shows that $\boldsymbol{y}_1 =$ <u>photo of a bald black man with beard</u> ($\lambda_1 = 6000$) generates good samples of men with beard but not all of them have a dark-skin tone; however, by decomposing it as $\boldsymbol{y}_1 =$ <u>photo of a bald man</u> ($\lambda_1 = 1500$), $\boldsymbol{y}_2 =$ <u>photo of a black man</u> ($\lambda_1 = 3000$), and $\boldsymbol{y}_3 =$ <u>photo of a man with beard</u> ($\lambda_1 = 1500$), we have better control (on average),

as shown in Figure 8(b). Note that the random seed for Figure 8(a) and Figure 8(b) is the same, and hence the identities are preserved and mostly the skin tone has changed.

(a) CLIP with one complex sentence (b) CLIP with three simple sentences

Figure 8: Decomposing complex controls (e.g., photo of a bald black man with beard) into simpler ones improves control performance. See details in Section 4.4.

4.5 Iterative Distributional Control via Functional Composition

This section discusses an interesting bias that PromptGen reveals about the CLIP model. Figure 9(a) shows images generated by PromptGen (with StyleGAN2) with the CLIP model and the 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* since 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 *vision-language pre-training data*: people are more likely to *say* "a person without makeup" when the person is a female (detailed analysis in Appendix F).

Besides revealing the above "reporting bias", PromptGen can also mitigate this bias via an iterative 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 with $\hat{\boldsymbol{\beta}}$ trained for $G \circ f_{\boldsymbol{\theta}}$. Figure 9(b) shows that females and males are uniformly distributed after the moment constraint in the second iteration.

(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 9: 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". 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.

4.6 Inference Latency

We compared our PromptGen and the plug-and-play generative model (PPGM) [49] in terms of the inference latency. PPGM uses Langevin dynamics [76] to optimize over the latent space at 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 were 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. Moreover, learning $f_{\boldsymbol{\theta}}$ in PromptGen requires training-time optimization, and this amortized optimization is useful when one wants to reuse a controlled distribution many times.

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}.

	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, by leveraging the knowledge of other off-the-shelf models. Unlike previous approaches, which require optimization for generating each sample at inference, PromptGen trains an invertible neural network to approximate the latent distribution. Thus, PromptGen can be viewed as an amortization of the inference-time optimization into a feed-forward neural network, allowing for efficient, feed-forward sampling at inference. Further, this amortization also makes PromptGen not require the availability of those off-the-shelf models at inference. In practice, PromptGen offers generality for algorithmic design and modularity for control composition, and it also enables iterative controls via functional composition. Experiments validate that PromptGen applies to various generative models (StyleGAN2, StyleNeRF, diffusion autoencoder, and 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 and a discussion on societal impact in Section 6. PromptGen is restricted by the pre-trained generative model's coverage [3, 28]. Notably, since PromptGen focuses on mode-seeking and reweighting instead of domain adaptation, the support set of the output distribution will not change after PromptGen learning. We find it interesting to further explore combining PromptGen and domain adaptation [17] to achieve better generalization and adaptability. Also, PromptGen depends on the off-the-shelf models that provide knowledge about the control (Appendix F). In subsequent work, it can be beneficial to explore how to learn energy functions [15, 27] (besides our moment constraint used for de-biasing), which may provide fine-grained control with less bias.

6 Societal Impact

Like any technology, generative models have both benefits and drawbacks for society; therefore, it is essential to investigate and understand these issues. On the positive side, we have demonstrated that PromptGen may be used to uncover biases learnt by text-image models (like CLIP) and to de-bias text-image models and pre-trained generative models, suggesting that PromptGen may be a helpful tool for fair AI if used properly. The efficient inference provided by PromptGen also helps reduce the computational expense, which has a positive impact on the environment. However, improved controllability inevitably makes it easier to synthesize targeted images, which might have negative societal implications such as the creation of deceptive media (like DeepFakes [77, 74]) or privacy leakage (like identity-conditioned human face synthesis). To battle these cases, we expect researchers and practitioners to use current technologies that can detect fake media and mitigate privacy leakage. We also encourage practitioners to consider these risks when using PromptGen to develop systems for machine learning, visual understanding, and media synthesis.

Acknowledgments and Disclosure of Funding

The authors would like to thank the anonymous reviewers, Zoltán Ádám Milacski, Jianchun Chen, and Shubhra Aich for their valuable feedback on drafts of this paper.

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. *SIGGRAPH*, 2022.

[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] Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyu Zhang, Heng Gong, and Bing Qin. A distributional lens for multi-aspect controllable text generation. *EMNLP*, 2022.

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

[24] 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.

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

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

[27] 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.

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

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

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

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

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

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

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

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

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

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

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

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

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

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

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

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

[44] 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.

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

[46] 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.

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

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

[49] 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.

[50] 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.

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

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

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

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

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

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

[57] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. *CVPR*, 2022.

[58] 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.

[59] 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.

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

[61] 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.

[62] 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.

[63] 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.

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

[65] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: Open dataset of CLIP-filtered 400 million image-text pairs. *ArXiv*, 2021.

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

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

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

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

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

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

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

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

[74] Cristian Vaccari and Andrew Chadwick. Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news. *Social Media + Society*, 6, 2020.

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

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

[77] Mika Westerlund. The emergence of Deepfake technology: A review. *Technology Innovation Management Review*, 9(11), 2019.

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

[79] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Interpretable transformations with encoder-decoder networks. *ICCV*, 2017.

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

[81] 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.

[82] 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.

[83] 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.

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

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

[86] 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.

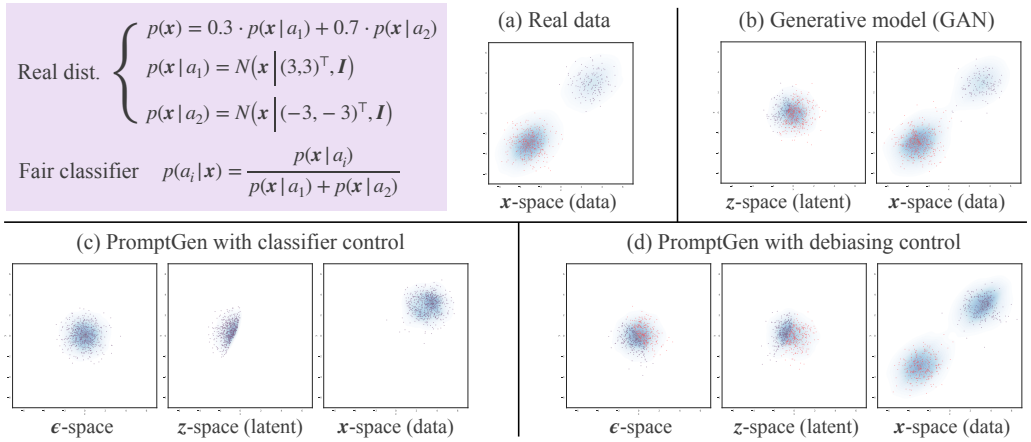


Figure 10: PromptGen on synthetic data. (a) The real distribution is a Gaussian mixture distribution biased towards one mode. We derive a closed-form fair classifier. (b) A GAN trained on this real distribution. (c) With the fair classifier as control, PromptGen learns a distribution concentrated in a specific region. (d) We derive the moment constraint following Section 3.1 and Section 4.2, and PromptGen learns to up-weight the under-represented regions.

A PromptGen on Synthetic Data

We demonstrate the behavior of our PromptGen with two-dimensional synthetic data, using GAN as the generative model. This synthetic experiment is illustrated in Figure 10. Specifically, we created a "real" distribution that is a Gaussian mixture distribution biased towards one mode. We derive a closed-form fair classifier based on this distribution, detailed in the purple block in Figure 10. When a GAN is trained to approximate this distribution, its outputs are biased towards the over-represented mode, as shown by Figure 10(b). Figure 10(c) illustrates the controllability experiment: using the fair classifier as control, PromptGen learns a distribution concentrated in a specific region of the output space. Figure 10(d) illustrates the de-biasing experiment: using the moment constraint (Section 3.1 and Section 4.2), PromptGen upweights the under-represented regions of the output space.

B Method Details and Derivations

B.1 Controllability as EBM

In Section 3, we defined a control $\mathcal{C}$ as M independent properties $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$. For example, $\boldsymbol{y}_1$ can be a text description, and $\boldsymbol{y}_2$ can be an attribute. In this part, we first elaborate on why controllability can be formed as EBMs in Eq. (1). We then provide concrete examples of the distributions derived from different energy functions defined in Section 3.1.

We denote the image prior as $p_{\boldsymbol{x}}(\boldsymbol{x})$, the only distribution that can be estimated from data when labels for the control are not provided during generative model pre-training. Given the control $\mathcal{C} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$, we resort to Bayes' theorem to rewrite the conditional distribution $p(\boldsymbol{x}|\mathcal{C})$ as

$$p(\boldsymbol{x}|\mathcal{C}) \propto p_{\boldsymbol{x}}(\boldsymbol{x}) p(\mathcal{C}|\boldsymbol{x}) \tag{13}$$

$$= p_{\boldsymbol{x}}(\boldsymbol{x}) p(\boldsymbol{y}_1|\boldsymbol{x}) \prod_{i=2}^{M} p(\boldsymbol{y}_i|\boldsymbol{x}, \boldsymbol{y}_{<i}) \tag{14}$$

$$= p_{\boldsymbol{x}}(\boldsymbol{x}) \prod_{i=1}^{M} p(\boldsymbol{y}_i|\boldsymbol{x}) \quad \text{(independence assumption).} \tag{15}$$

For Eq. (15) to be well-defined, we need to define $p(\boldsymbol{y}_i|\boldsymbol{x})$ for each $\boldsymbol{y}_i$. In order to incorporate the knowledge of arbitrary off-the-self models (besides image classifiers), we define each $p(\boldsymbol{y}_i|\boldsymbol{x})$ as

$$p(\boldsymbol{y}_i|\boldsymbol{x}) = \frac{\exp(-\lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i))}{Z_i}, \quad Z_i = \int_{\boldsymbol{y}_i'} \exp(-\lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i')) d\boldsymbol{y}_i'. \tag{16}$$

Eq. (16) says that $p(\boldsymbol{y}_i|\boldsymbol{x})$ is proportional to the exponential of an energy function $E_i(\boldsymbol{x}, \boldsymbol{y}_i)$, where $\boldsymbol{y}_i$ with lower energy has higher density or mass. Note that Eq. (16) defines a distribution over all possible values of $\boldsymbol{y}_i$ instead of all possible values of $\boldsymbol{x}$. Combining Eq. (15) and Eq. (16), we have

$$p(\boldsymbol{x}|\mathcal{C}) = \frac{p_{\boldsymbol{x}}(\boldsymbol{x})e^{-E_{\mathcal{C}}(\boldsymbol{x})}}{Z_X}, \quad E_{\mathcal{C}}(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i), Z_X = \int_{\boldsymbol{x}'} p_{\boldsymbol{x}}(\boldsymbol{x}')e^{-E_{\mathcal{C}}(\boldsymbol{x}')}d\boldsymbol{x}', \tag{17}$$

which is the same equation as Eq. (1).

In the following, we use Eq. (16) to derive the distributions from the classifier energy, CLIP energy, and inverse graphics energy defined in Section 3.1.

Classifier energy: Given a classifier $P(\cdot|\boldsymbol{x})$ and the target class a, we define the classifier energy as $E_{\text{classifier}}(\boldsymbol{x}, a) = -\log P(a|\boldsymbol{x})$. Using Eq. (16), we arrive at:

$$p_{\text{classifier}}(a|\boldsymbol{x}) = \frac{\exp(\lambda_{\text{classifier}} \log P(a|\boldsymbol{x}))}{\sum_{a'} \exp(\lambda_{\text{classifier}} \log P(a'|\boldsymbol{x}))} = \frac{P(a|\boldsymbol{x})^{\lambda_{\text{classifier}}}}{\sum_{a'} P(a'|\boldsymbol{x})^{\lambda_{\text{classifier}}}}, \tag{18}$$

which is equivalent to a temperature-adjusted distribution of the original classifier.

CLIP energy: Using Eq. (16), the CLIP energy in Eq. (3) is equivalent to

$$p_{\text{CLIP}}(\boldsymbol{t}|\boldsymbol{x}) \propto \exp\left(-\frac{\lambda_{\text{CLIP}}}{L} \sum_{l=1}^{L} \left(1 - \cos\left\langle \text{CLIP}_{\text{img}}\big(\text{DiffAug}_l(\boldsymbol{x})\big), \text{CLIP}_{\text{text}}(\boldsymbol{t}) \right\rangle \right) \right). \tag{19}$$

Inverse graphics energy: Using Eq. (16), the inverse graphics energy in Eq. (4) is equivalent to

$$p_{\text{inv-graphics}}(\boldsymbol{\rho}|\boldsymbol{x}) \propto \exp(-\lambda_{\text{inv-graphics}} d\big\langle f_{\mathcal{X}\to\mathcal{P}}(\boldsymbol{x}), \boldsymbol{\rho}\big\rangle^2). \tag{20}$$

If the geodesic distance $d\big\langle\cdot,\cdot\big\rangle$ is the Euclidean distance, then $p_{\text{inv-graphics}}(\boldsymbol{\rho}|\boldsymbol{x})$ is a Gaussian distribution whose mean is $\boldsymbol{\rho}$ and whose variance depends on the hyperparameter $\lambda_{\text{inv-graphics}}$; if the geodesic distance $d\big\langle\cdot,\cdot\big\rangle$ is the spherical distance (e.g., the distance between pose parameters defined on a unit sphere), then $p_{\text{inv-graphics}}(\boldsymbol{\rho}|\boldsymbol{x})$ is a vMF distribution.

B.2 Equivalence between Image-Space EBM and Latent-Space EBM

Proposition 1. *Define $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x})$ as $\underline{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}), \boldsymbol{x} = G(\boldsymbol{z})}$ and $p(\boldsymbol{x}|\mathcal{C})$ as $\underline{\boldsymbol{z} \sim p(\boldsymbol{z}|\mathcal{C}), \boldsymbol{x} = G(\boldsymbol{z})}$, where $p(\boldsymbol{z}|\mathcal{C})$ is defined as the following EBM:*

$$p(\boldsymbol{z}|\mathcal{C}) = \frac{p_{\boldsymbol{z}}(\boldsymbol{z})e^{-E_{\mathcal{C}}(G(\boldsymbol{z}))}}{Z}, \quad E_{\mathcal{C}}(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i E_i(\boldsymbol{x}, \boldsymbol{y}_i), Z = \int_{\boldsymbol{z}'} p_{\boldsymbol{z}}(\boldsymbol{z}')e^{-E_{\mathcal{C}}(G(\boldsymbol{z}'))}d\boldsymbol{z}'. \tag{21}$$

We have

$$p(\boldsymbol{x}|\mathcal{C}) = \frac{p_{\boldsymbol{x}}(\boldsymbol{x})e^{-E_{\mathcal{C}}(\boldsymbol{x})}}{Z_X}, \quad Z_X = \int_{\boldsymbol{x}'} p_{\boldsymbol{x}}(\boldsymbol{x}')e^{-E_{\mathcal{C}}(\boldsymbol{x}')}d\boldsymbol{x}'. \tag{22}$$

In spirit, our proof follows the proof in [52], which follows [19]. The difference between our proof and that in [52] is that we derive $p(\boldsymbol{x}|\mathcal{C})$ from $p(\boldsymbol{z}|\mathcal{C})$ while they derived $p(\boldsymbol{z}|\mathcal{C})$ from $p(\boldsymbol{x}|\mathcal{C})$.

Proof. Based on Lemma 1 in [19] and Lemma 1 in [52], $p(\boldsymbol{z}|\mathcal{C})$ is equivalent to rejection sampling with proposal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$ and acceptance probability

$$r(\boldsymbol{z}) = \frac{e^{-E_{\mathcal{C}}(G(\boldsymbol{z}))}}{M_{\mathcal{C}} \cdot Z}, \quad \text{where } \forall \boldsymbol{z}, M_{\mathcal{C}} > \frac{e^{-E_{\mathcal{C}}(G(\boldsymbol{z}))}}{Z}. \tag{23}$$

Since $p(\boldsymbol{x}|\mathcal{C})$ is defined as $\underline{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}|\mathcal{C}), \boldsymbol{x} = G(\boldsymbol{z})}$, $p(\boldsymbol{x}|\mathcal{C})$ is equivalent to rejection sampling with proposal distribution $\underline{p_{\boldsymbol{z}}(\boldsymbol{z}), \boldsymbol{x} = G(\boldsymbol{z})}$ and acceptance probability

$$r(\boldsymbol{x}) = \frac{e^{-E_{\mathcal{C}}(\boldsymbol{x})}}{M_{\mathcal{C}} \cdot Z}. \tag{24}$$

Note that the above proposal distribution $\underline{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}), \boldsymbol{x} = G(\boldsymbol{z})}$ is the same as $p_{\boldsymbol{x}}(\boldsymbol{x})$ by definition. Based on Lemma 1 in [19] and Lemma 1 in [52], we arrive at:

$$
\begin{aligned}
p(\boldsymbol{x}|\mathcal{C}) &= \frac{p_{\boldsymbol{x}}(\boldsymbol{x}) r(\boldsymbol{x})}{\mathbb{E}_{\boldsymbol{x}' \sim p_{\boldsymbol{x}}(\boldsymbol{x}')}[r(\boldsymbol{x}')]} && (25)\\
&= \frac{p_{\boldsymbol{x}}(\boldsymbol{x}) e^{-E_{\mathcal{C}}(\boldsymbol{x})}/(M_{\mathcal{C}} \cdot Z)}{\mathbb{E}_{\boldsymbol{x}' \sim p_{\boldsymbol{x}}(\boldsymbol{x}')}[e^{-E_{\mathcal{C}}(\boldsymbol{x}')}/(M_{\mathcal{C}} \cdot Z)]} && (26)\\
&= \frac{p_{\boldsymbol{x}}(\boldsymbol{x}) e^{-E_{\mathcal{C}}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}' \sim p_{\boldsymbol{x}}(\boldsymbol{x}')}[e^{-E_{\mathcal{C}}(\boldsymbol{x}')}]} && (27)\\
&= \frac{p_{\boldsymbol{x}}(\boldsymbol{x}) e^{-E_{\mathcal{C}}(\boldsymbol{x})}}{Z_X}, \quad Z_X = \int_{\boldsymbol{x}'} p_{\boldsymbol{x}}(\boldsymbol{x}') e^{-E_{\mathcal{C}}(\boldsymbol{x}')} d\boldsymbol{x}'. && (28)
\end{aligned}
$$

□

B.3 Derivation of Eq. (9): Approximating EBM with INN

The full derivation of Eq. (9) is given by:

$$
\begin{aligned}
&\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z}) \| p(\boldsymbol{z}|\mathcal{C}))\\
&= \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z})}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z})}{p(\boldsymbol{z}|\mathcal{C})}\Big]\\
&= \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{x}=G(\boldsymbol{z})}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z})}{p_{\boldsymbol{z}}(\boldsymbol{z}) e^{-E_{\mathcal{C}}(\boldsymbol{x})}/Z}\Big]\\
&= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\boldsymbol{I}), \boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}), \boldsymbol{x}=G(\boldsymbol{z})}\Big[\log \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\boldsymbol{I}) - \log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})|\\
&\qquad\qquad - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}}(\boldsymbol{x}) + \log Z\Big]\\
&= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\boldsymbol{I}), \boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}), \boldsymbol{x}=G(\boldsymbol{z})}\Big[-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z})\\
&\qquad\qquad + E_{\mathcal{C}}(\boldsymbol{x})\Big] - \mathbb{H}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})} + \log Z.
\end{aligned} \tag{29}
$$

B.4 Derivation of Eq. (11): Extension to Generative Models with a Class-Embedding Space

The full derivation of Eq. (11) is given by:

$$
\begin{aligned}
&\mathbb{D}_{\mathsf{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y}) \| p(\boldsymbol{z},\boldsymbol{y}|\mathcal{C}))\\
&= \mathbb{E}_{(\boldsymbol{z},\boldsymbol{y}) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y})}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y})}{p(\boldsymbol{z},\boldsymbol{y}|\mathcal{C})}\Big]\\
&= \mathbb{E}_{(\boldsymbol{z},\boldsymbol{y}) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y}), \boldsymbol{x}=G(\boldsymbol{z},\boldsymbol{y})}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y})}{p_{\boldsymbol{z},\boldsymbol{y}}(\boldsymbol{z},\boldsymbol{y}) e^{-E_{\mathcal{C}}(\boldsymbol{x})}/Z}\Big]\\
&= \mathbb{E}_{(\boldsymbol{z},\boldsymbol{y}) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y}), \boldsymbol{x}=G(\boldsymbol{z},\boldsymbol{y})}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{y})}{p_{\boldsymbol{z}}(\boldsymbol{z}) p_{\boldsymbol{y}}(\boldsymbol{y}) e^{-E_{\mathcal{C}}(\boldsymbol{x})}/Z}\Big] \quad (\boldsymbol{z} \text{ and } \boldsymbol{y} \text{ are independent})\\
&= \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 \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\boldsymbol{I}) - \log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})|\\
&\qquad + \log \mathcal{N}(\boldsymbol{\xi}|\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I}) - \log|\det(\frac{\partial h_{\boldsymbol{\theta}}}{\partial \boldsymbol{\xi}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) - \log p_{\boldsymbol{y}}(\boldsymbol{y}) + E_{\mathcal{C}}(\boldsymbol{x}) + \log Z\Big]\\
&= \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|\det(\frac{\partial h_{\boldsymbol{\theta}}}{\partial \boldsymbol{\xi}})|\\
&\qquad - \log p_{\boldsymbol{z}}(\boldsymbol{z}) - \log p_{\boldsymbol{y}}(\boldsymbol{y}) + E_{\mathcal{C}}(\boldsymbol{x})\Big] - \mathbb{H}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})} - \mathbb{H}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I})} + \log Z.
\end{aligned} \tag{30}
$$

Algorithm 3: Extension of Algorithm 2 to Conditional INN

while *not converged* **do**

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

B.5 Extension to Conditional INN

Algorithm 3 illustrates how we extend the training algorithm (Algorithm 2) to conditional INN, which helps generalize to controls specified by continuous values $\boldsymbol{\rho}$. The training objective becomes

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\rho}\sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho})}\big[\mathbb{D}_{\mathsf{KL}}\big(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{\rho})\|p(\boldsymbol{z}|\mathcal{C}_{\boldsymbol{\rho}})\big)\big], \tag{31}$$

where $\mathcal{C}_{\boldsymbol{\rho}}$ means that the control is specified by value $\boldsymbol{\rho}$. The derivation of Algorithm 3 is given by:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\rho}\sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho})}\big[\mathbb{D}_{\mathsf{KL}}\big(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{\rho})\|p(\boldsymbol{z}|\mathcal{C}_{\boldsymbol{\rho}})\big)\big] \\
&= \mathbb{E}_{\boldsymbol{\rho}\sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho}),\boldsymbol{z}\sim p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{\rho})}\Big[\log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{\rho})}{p(\boldsymbol{z}|\mathcal{C}_{\boldsymbol{\rho}})}\Big] \\
&= \mathbb{E}_{\boldsymbol{\rho}\sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho}),\boldsymbol{z}\sim p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{\rho}),\boldsymbol{x}=G(\boldsymbol{z})}\Big[\log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{\rho})}{p_{\boldsymbol{z}}(\boldsymbol{z})e^{-E_{\mathcal{C}_{\boldsymbol{\rho}}}(\boldsymbol{x})}/Z}\Big] \\
&= \mathbb{E}_{\boldsymbol{\rho}\sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho}),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon},\boldsymbol{\rho}),\boldsymbol{x}=G(\boldsymbol{z})}\Big[\log\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\boldsymbol{I}) - \log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial\boldsymbol{\epsilon}})| \\
&\qquad\qquad - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}_{\boldsymbol{\rho}}}(\boldsymbol{x}) + \log Z\Big] \\
&= \mathbb{E}_{\boldsymbol{\rho}\sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho}),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\boldsymbol{z}=f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon},\boldsymbol{\rho}),\boldsymbol{x}=G(\boldsymbol{z})}\Big[-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial\boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) \\
&\qquad\qquad + E_{\mathcal{C}_{\boldsymbol{\rho}}}(\boldsymbol{x})\Big] - \mathbb{H}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})} + \log Z.
\end{aligned} \tag{32}$$

B.6 Derivation for Latent-Space Moment Constraint

Given a mapping $\boldsymbol{\gamma} : \mathcal{X} \to \mathbb{R}^K$, moment constraint [7, 37] defines the target distribution $p(\boldsymbol{x}|\mathcal{C})$ as

$$p(\boldsymbol{x}|\mathcal{C}) = \underbrace{\arg\min_{p(\boldsymbol{x}|\mathcal{C})} \mathbb{D}_{\mathsf{KL}}(p(\boldsymbol{x}|\mathcal{C})\|p_{\boldsymbol{x}}(\boldsymbol{x}))}_{\text{Deviation from the pre-trained distribution}}, \quad \text{s.t.} \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\mathcal{C})}\big[\boldsymbol{\gamma}(\boldsymbol{x})\big] = \boldsymbol{\mu}}_{\text{Moment constraint}}, \tag{33}$$

where $\boldsymbol{\mu}$ is the user-specified constraint. Examples are provided in the main text, and we omit them here for brevity. [7] showed that Eq. (33) can be approximated to an arbitrary precision by

$$p(\boldsymbol{x}|\mathcal{C}) \propto p_{\boldsymbol{x}}(\boldsymbol{x})\exp\big(\hat{\boldsymbol{\beta}}^\top\boldsymbol{\gamma}(\boldsymbol{x})\big), \tag{34}$$

where $\hat{\boldsymbol{\beta}}$ needs to be computed. [37] estimates $\hat{\boldsymbol{\beta}}$ by solving the following regression problem:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \mathbb{E}_{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(N)}\overset{\text{i.i.d.}}{\sim}p_{\boldsymbol{x}}(\boldsymbol{x})}\left\|\frac{\sum_{j=1}^N\exp\big(\boldsymbol{\beta}^\top\boldsymbol{\gamma}(\boldsymbol{x}^{(j)})\big)\boldsymbol{\gamma}(\boldsymbol{x}^{(j)})}{\sum_{j'=1}^N\exp\big(\boldsymbol{\beta}^\top\boldsymbol{\gamma}(\boldsymbol{x}^{(j')})\big)} - \boldsymbol{\mu}\right\|_2^2. \tag{35}$$

In [37], sampling $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x})$ is straightforward since they focus on autoregressive language models. In the context of latent-variable vision generative models, where $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x})$ is implicitly defined as $\underline{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}), \boldsymbol{x} = G(\boldsymbol{z})}$, Eq. (35) is equivalent to

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \mathbb{E}_{\boldsymbol{z}^{(1)},\ldots,\boldsymbol{z}^{(N)}\overset{\text{i.i.d.}}{\sim}p_{\boldsymbol{z}}(\boldsymbol{z}),\boldsymbol{x}^{(j)}=G(\boldsymbol{z}^{(j)})}\left\|\frac{\sum_{j=1}^N\exp\big(\boldsymbol{\beta}^\top\boldsymbol{\gamma}(\boldsymbol{x}^{(j)})\big)\boldsymbol{\gamma}(\boldsymbol{x}^{(j)})}{\sum_{j'=1}^N\exp\big(\boldsymbol{\beta}^\top\boldsymbol{\gamma}(\boldsymbol{x}^{(j')})\big)} - \boldsymbol{\mu}\right\|_2^2. \tag{36}$$

Finally, based on Proposition 1, we can generalize the moment constraint to the latent space as

$$p(\boldsymbol{z}|\mathcal{C}) = \frac{p_{\boldsymbol{z}}(\boldsymbol{z}) \exp\Big(\hat{\boldsymbol{\beta}}^\top \boldsymbol{\gamma}\big(G(\boldsymbol{z})\big)\Big)}{Z}, \quad Z = \int_{\boldsymbol{z}'} p_{\boldsymbol{z}}(\boldsymbol{z}') \exp\Big(\hat{\boldsymbol{\beta}}^\top \boldsymbol{\gamma}\big(G(\boldsymbol{z}')\big)\Big) d\boldsymbol{z}'. \tag{37}$$

Note that Eq. (36) and Eq. (37) are verbatim copies of Eq. (7) and Eq. (6), respectively.

Algorithm 4: Extension of Algorithm 3 with ID Energy (Section 4.3)

while *not converged* **do**

1. Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \boldsymbol{\rho} \sim p_{\boldsymbol{\rho}}(\boldsymbol{\rho})$
2. Map $\boldsymbol{\epsilon}$ to latent codes $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \boldsymbol{\rho})$ and $\boldsymbol{z}_0 = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \boldsymbol{\rho}_0)$
3. Map $\boldsymbol{z}$ to an image $\boldsymbol{x} = G(\boldsymbol{z})$ and $\boldsymbol{z}_0$ to an image $\boldsymbol{x}_0 = G(\boldsymbol{z}_0)$
4. Optimize $\boldsymbol{\theta}$ with gradient

$$\nabla_{\boldsymbol{\theta}}\Big(-\log|\det(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\epsilon}})| - \log p_{\boldsymbol{z}}(\boldsymbol{z}) + E_{\mathcal{C}_{\boldsymbol{\rho}}}(\boldsymbol{x}) + \lambda_{\text{ID}} E_{\text{ID}}(\boldsymbol{x}_0, \boldsymbol{x})\Big)$$

B.7 Inverse Graphics Control with Identity Energy

This subsection provides more detail on how we use an inverse graphics model, DECA [16], to control the pose of faces generated by StyleGAN2 trained on FFHQ 1024^2, as introduced in Section 4.3. Recall that we use the conditional INN extension, detailed in Algorithm 3, for this experiment. To enable generating different poses of the same identity, we add additional identity energy using the IR-SE50 model [8]. The training algorithm is detailed in Algorithm 4. Specifically, we generate a canonical latent code $\boldsymbol{z}_0 = f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \boldsymbol{\rho}_0)$ for each $\boldsymbol{\epsilon}$, where $\boldsymbol{\rho}_0$ is the canonical pose. The latent code $\boldsymbol{z}$ and the canonical latent code $\boldsymbol{z}_0$ are mapped to the image $\boldsymbol{x}$ and the canonical image $\boldsymbol{x}_0$. The identity energy in Eq. (12), copied below, encourages the embeddings of the two images to be similar:

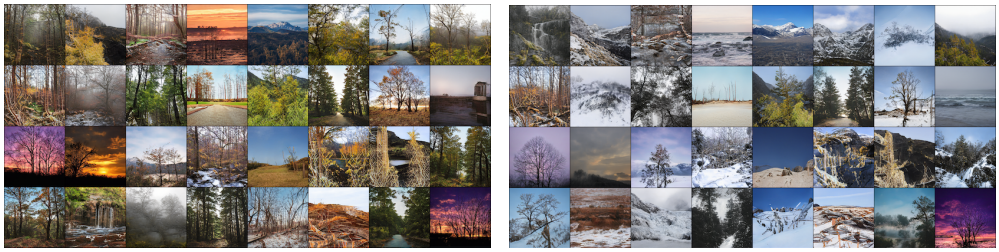
$$E_{\text{ID}}(\boldsymbol{x}_0, \boldsymbol{x}) = 1 - \cos\big\langle R(\boldsymbol{x}_0), R(\boldsymbol{x})\big\rangle, \quad \boldsymbol{x}_0 = G(\boldsymbol{z}_0), \boldsymbol{x} = G(\boldsymbol{z}). \tag{38}$$

B.8 Experimental Details

Our INN architecture contains 8 blocks. Each block consists of a soft permutation of channels [2], an affine coupling layer [12], and an ActNorm layer [39]. In the affine coupling layer, we model the sub-network as an MLP with one hidden layer, where the hidden dimension is 256 and the non-linear activation is $\text{LeakyReLU}_{0.1}$.

For the prior distribution $p_{\boldsymbol{\rho}}(\boldsymbol{\rho})$ of the pose parameter $\boldsymbol{\rho}$ in the inverse graphics experiments (Section 4.3), we sample the x-axis and y-axis rotations relative to a canonical pose $\boldsymbol{\rho}_0$, whose x-axis and y-axis rotations are 0.3 and 0, respectively. The relative pose's x-axis and y-axis rotations are uniformly sampled from $[-(1/9)\pi, (1/9)\pi]$.

Each experiment was run on an NVIDIA RTX A4000 GPU (with 16G memory). Our code implementation is based on the PyTorch framework. Our code can be found at `https://github.com/ChenWu98/Generative-Visual-Prompt`.



(a) PromptGen (description: autumn scene) (b) PromptGen (description: winter scene)

Figure 11: Additional results for image synthesis based on text description, guided by the CLIP model. As the backbone, we use StyleGAN2 trained on Landscape-HQ with truncation $\psi = 0.7$. All images are 256^2 and resized for visualization.

Table 5: Quantitative results for de-biasing categorical attributes (Figure 6). See details in Appendix D. PromptGen de-biases StyleGAN2 in terms of race, age, and gender.

	FFHQ		MetFaces		
	$\mathbb{D}_{\text{KL}}^{\text{race}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{age}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{race}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{age}}\downarrow$	$\mathbb{D}_{\text{KL}}^{\text{gender}}\downarrow$
StyleGAN2	0.860	0.597	1.624	0.546	0.019
PromptGen (ours; $\lambda = 1$)	0.286	0.357	0.687	0.397	**0.000**
PromptGen (ours; $\lambda = 2$)	**0.099**	**0.172**	**0.189**	**0.247**	0.005

C Additional Results for Image Synthesis based on Text Description

This section presents additional results of PromptGen for image synthesis from text, using the CLIP model as control. As the pre-trained generative model, we explore StyleGAN2 trained on AFHQ-Cats [6], LSUN-Churches [83], FFHQ [35], and Landscape-HQ [70]. Figure 11 and Figure 12 present more results of PromptGen with diverse text descriptions.

D Additional Results for De-Biasing Generative Models

Table 5 provides quantitative results of de-biasing StyleGAN2 across categorical attributes (Figure 6). Specifically, we used the pre-trained classifier provided by FairFace [32] to classify the attributes of the generated images. We then report the KL divergence between the attribute distribution of the generated images and the uniform distribution. Figure 13 and Figure 14 visualize the de-biasing results. We report the KL divergence between the generated distribution and the uniform distribution. Interestingly, on MetFaces, de-biasing the race results in more sculptures, and we postulate that the reason is that almost all paintings and sketches in MetFaces are for white individuals.

(a) photo of a cat with closed eyes (AFHQ-Cats)

(b) photo of a cat with eyes wide open (AFHQ-Cats)

(c) photo of a church during day (LSUN-Churches)

(d) photo of a church at night (LSUN-Churches)

(e) photo of a happy person (FFHQ)

(f) photo of a sad person (FFHQ)

(g) photo of a woman with glasses (FFHQ)

Figure 12: Additional results for image synthesis based on text description, guided by the CLIP model. As the pre-trained generative model, we use StyleGAN2 trained on FFHQ, AFHQ-Cats and LSUN-Churches datasets. The captions are the text descriptions given to the CLIP model.

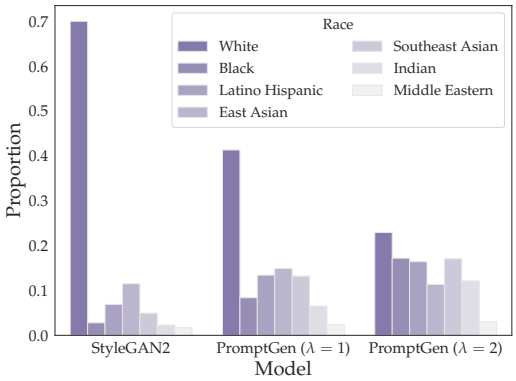
(a) Pre-trained StyleGAN2

(b) PromptGen (race de-biasing; $\lambda = 1$)

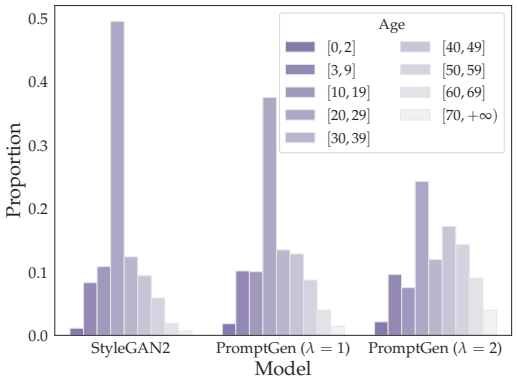
(c) PromptGen (race de-biasing; $\lambda = 2$)

(d) PromptGen (age de-biasing; $\lambda = 1$)

(e) PromptGen (age de-biasing; $\lambda = 2$)



(f) Race distribution



(g) Age distribution

Figure 13: Using our moment constraint, PromptGen de-biases the racial and age distributions of StyleGAN2 trained on FFHQ with truncation $\psi = 0.7$. All synthesized images are 1024^2 in resolution and resized for visualization. We fixed the random seed for PromptGen, so please zoom in to see detailed differences between images.

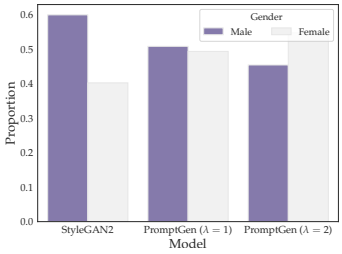
(a) Pre-trained StyleGAN2

(b) PromptGen (race de-biasing; $\lambda = 1$)

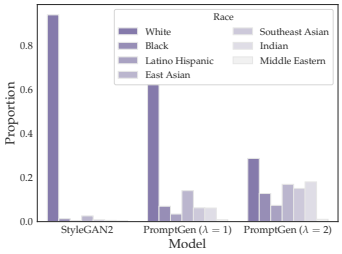
(c) PromptGen (race de-biasing; $\lambda = 2$)

(d) PromptGen (age de-biasing; $\lambda = 2$)

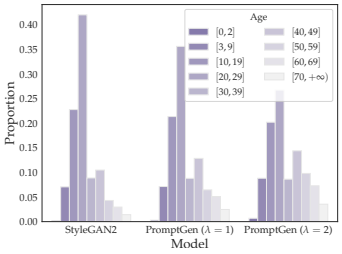
(e) PromptGen (gender de-biasing; $\lambda = 2$)



(f) Gender distribution

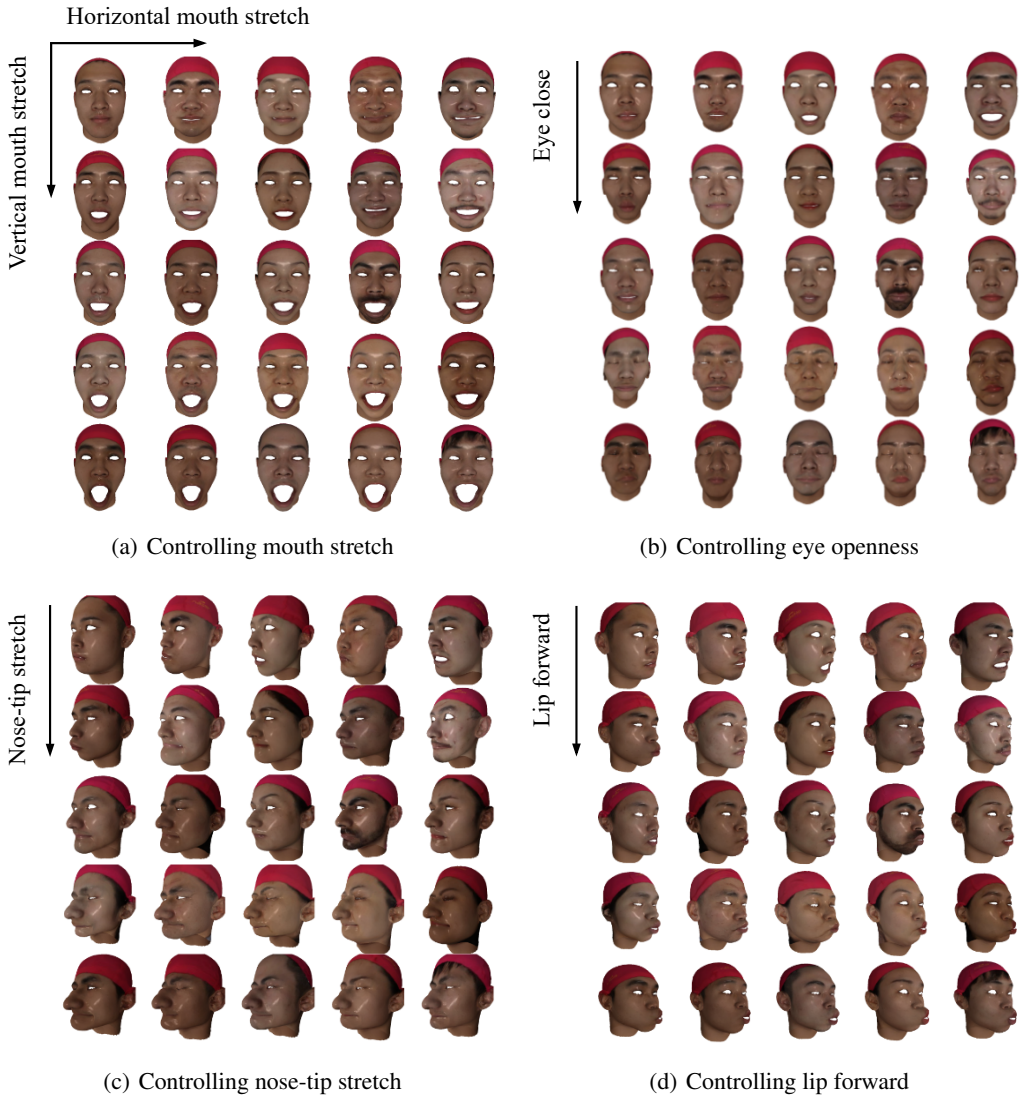


(g) Race distribution



(h) Age distribution

Figure 14: Using the moment constraint, PromptGen de-biases the racial, age, and gender distributions of StyleGAN2 trained on MetFaces with truncation $\psi = 0.7$. All images are 1024^2 and resized for visualization. Interestingly, de-biasing the race results in more sculptures, and we postulate that the reason is that almost all paintings and sketches in MetFaces are for white individuals. We fixed the random seed for PromptGen, so we recommend zooming in to see differences between images.



(a) Controlling mouth stretch (b) Controlling eye openness

(c) Controlling nose-tip stretch (d) Controlling lip forward

Figure 15: PromptGen with intra-sample signed-distance constraints that generate 3D face models with varying distances in the mouth, eye openness, nose-tip stretch, and lip forward.

E Experiments on Generative 3D Face Models

3D face modeling has been an active area of research that has recently gained major interest due to applications in virtual humans, deep faces, and digital actors. Existing 3D deep learning generative models build 3D compact representations of shape and appearance capable of modeling non-linearity (e.g., scatter effects, specularities) that is necessary for generating photo-realistic faces. Providing a generative model with the capability of generating 3D faces with a particular geometry (e.g., a specific distance between eyes or nose length) is particularly useful for technical artists when creating new characters. However, this problem remains unaddressed, partially due to the lack of publicly available 3D databases labeled with such constraints. We refer to this capability as the intra-sample constraint in 3D generative models, and this section shows how our PromptGen framework is able to achieve this by defining a new energy function based on signed distances.

We conducted the experiments on the FaceScape dataset [82], a large-scale 3D human face dataset consisting of $16,940$ topologically uniformly registered 3D face meshes along with high-quality textures. The dataset contains 847 identities with 20 expressions per identity, and each mesh consists of $26,317$ vertices of which 68 are 3D landmark vertices covering eyes, nose, mouth, jaw, and

eyebrows. With some abuse of notation, we denote a 3D face mesh as $\boldsymbol{x} \in \mathbb{R}^{3V}$ (recall that $\boldsymbol{x}$ is used to represent an image in other parts of this paper), where $V = 26,317$ is the number of vertices. Given an index l of a vertex (e.g., one of the 68 landmark vertices), the 3D coordinate of this vertex is defined as $\boldsymbol{x}[l] \in \mathbb{R}^3$. We pre-train a generative mesh model G that maps each latent code $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to a 3D mesh $\boldsymbol{x}$. PromptGen allows us to obtain geometric control on the 3D meshes. Specifically, given two landmark vertex indices l_1 and l_2, we define an intra-sample constraint that enforces the signed distance (along a given direction) between the two vertices to be s. To this effect, we define the signed-distance energy as

$$E_{\text{signed-distance}}(\boldsymbol{x}) = |d(\boldsymbol{v}_1, \boldsymbol{v}_2) - s|, \quad \boldsymbol{v}_1 = \boldsymbol{x}[l_1], \boldsymbol{v}_2 = \boldsymbol{x}[l_2] \tag{39}$$

where $d(\cdot, \cdot)$ is the signed-distance (along a given direction), and $|\cdot|$ is the absolute value.

Figure 15 presents several examples of 3D mesh control, in which random textures are applied to the generated meshes. It shows that PromptGen successfully controls the generative 3D mesh model in terms of the mouth stretch, eye openness, nose-tip stretch, and lip forward.

F Error Analysis

As we discussed in Section 5, the controlled distribution depends on (1) the pre-trained generative model's coverage and (2) the off-the-shelf models used for the control. In this section, we provide some examples of PromptGen failing to generate satisfactory images using the CLIP energy. For the pre-trained generative model, we use StyleGAN2 trained on FFHQ, AFHQ-Wild, or LSUN-Churches.

(a) photo of a man without beard (FFHQ)

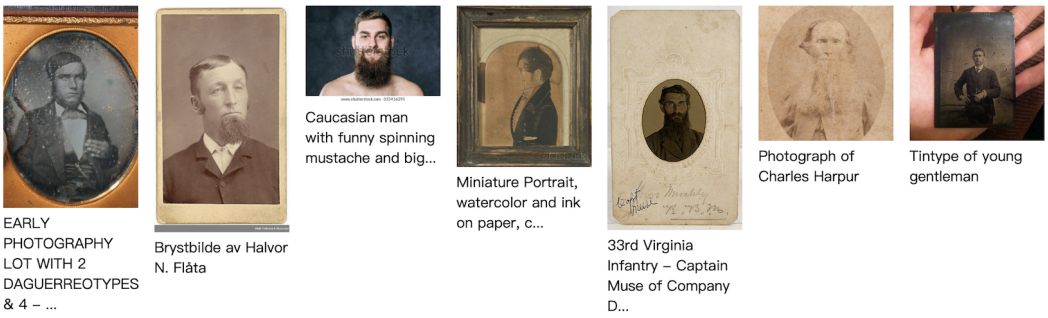
(b) a photo of a person without makeup (FFHQ)

(c) photo of a church with three windows (LSUN-Churches)

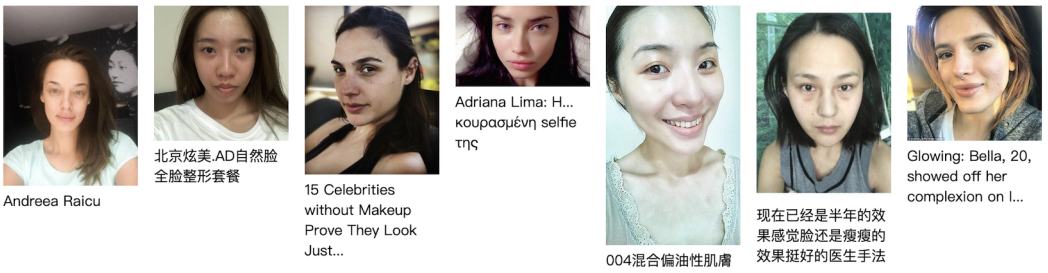
Figure 16: When the text description contains certain linguistic properties (e.g., negation, numerical reasoning), CLIP sometimes fails or shows the "reporting bias" that we discuss in Section 4.5. For a deeper understanding of these failures and biases, in Figure 17, we provide some image samples from the LAION-400M dataset [65] with CLIP retrieval, using the same text descriptions as this figure.

Our first observation is that the CLIP model sometimes fails in modeling linguistic negation. For instance, the text description photo of a man without beard results in a distribution of photos of men with beard (Figure 16(a)). Meanwhile, Figure 16(b) shows that CLIP is capable of modeling the negation of having makeup, but with the "reporting bias" discussed in Section 4.5. Moreover, CLIP seems to have difficulty in numerical reasoning, and gaining control over the count of specific objects in a scene tends to be unsuccessful. We showed this by specifying photo of a church with three windows, which did not result in the desired specification (Figure 16(c)).

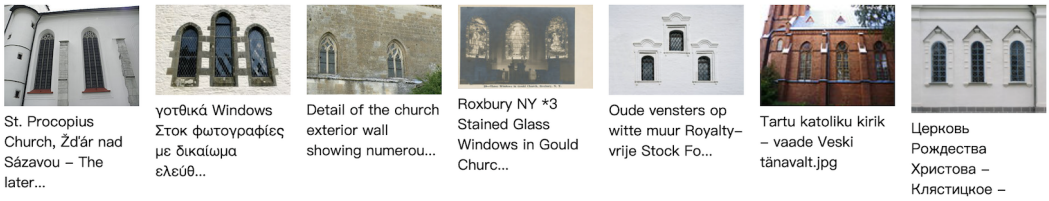
To gain a deeper understanding of the above failures and biases, in Figure 17, we provide some image samples from the LAION-400M dataset [65] with CLIP retrieval, using the same text descriptions as Figure 16. CLIP retrieval is based on the CLIP embedding similarity between the web images and text descriptions, while the original text below each individual image is not used. We observe an impressive consistency between CLIP retrieval and PromptGen: in both Figure 16(a) and Figure 17(a), most images have beard; in both Figure 16(b) and Figure 17(b), all images are female; in both Figure 16(c) and Figure 17(c), some images do not have exactly three windows. This consistency suggests that the failures and biases in Figure 16 should be mostly attributed to the CLIP model rather than to our PromptGen algorithm. We believe that our observation sheds light on the intricacy of contrastive multi-modal (vision-language) pre-training, which is worthy of being further investigated.



(a) photo of a man without beard (CLIP retrieval from LAION-400M)



(b) a photo of a person without makeup (CLIP retrieval from LAION-400M)



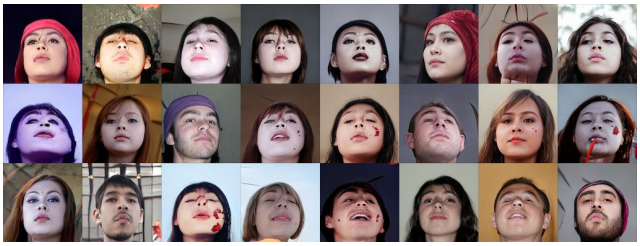
(c) photo of a church with three windows (CLIP retrieval from LAION-400M)

Figure 17: For a deeper understanding of the failures and biases illustrated in Figure 16, we provide some image samples from the LAION-400M dataset [65] with CLIP retrieval, using the same text descriptions as Figure 16. CLIP retrieval is based on the CLIP embedding similarity between the web images and text descriptions, while the original text below each individual image is not used. In Figure 17(a), most images have a beard; in Figure 17(b), all images are female; in Figure 17(c), some images do not have exactly three windows. These observations are consistent with those in Figure 16.

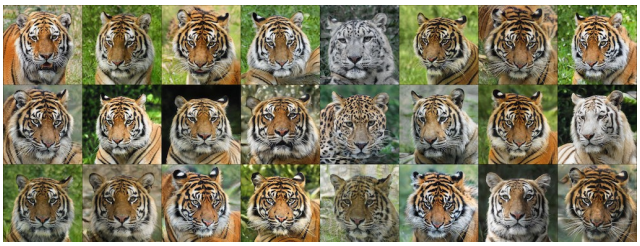
Another observation is that the control tends to fail when the text description requires sampling from low-density regions of the pre-trained generative model's output space. In other words, the control usually fails if the pre-trained generative model does not cover the mode we are trying to gain control over. For example, images faithful to photo of a person yawning and photo of a baby with long hair are not commonly observed in the FFHQ dataset and, hence, these two text descriptions result in degeneration (Figure 18(a)) or weird images (Figure 18(b)). Another example is photo of an animal from the side, which is not commonly observed in the AFHQ-Wild dataset, and Figure 18(c) shows that the generated images fail to follow this description. Even when the control is successful (e.g., when a complex description is decomposed in Figure 8(b)), sampling from low-density regions results in limited diversity (e.g., the backgrounds in Figure 8(b) look similar to each other).

Finally, we also ran some failure controls using PPGM, showing that PromptGen and PPGM reveal similar failure cases of pre-trained generative models and CLIP. Results are shown in Figure 19.

(a) photo of a person yawning (FFHQ)



(b) photo of a baby with long hair (FFHQ)



(c) photo of an animal from the side (AFHQ-Wild)

Figure 18: When the pre-trained generative model fails in covering certain modes required by the text description, unsatisfactory outputs are produced. In this figure, we show several text descriptions that require sampling from low-density regions of the pre-trained generative model's output space.

(a) photo of a man without beard (FFHQ; w/ PPGM)

(b) photo of a person yawning (FFHQ; w/ PPGM)

Figure 19: Failure modes revealed by PromptGen (Figure 16 and Figure 18) also hold for PPGM.