TR0N: Translator Networks for 0-Shot Plug-and-Play Conditional Generation

Zhaoyan Liu [*1,2] **Noël Vouitsis** [*1] **Satya Krishna Gorti** [1] **Jimmy Ba** [2,3] **Gabriel Loaiza-Ganem** [1]

Abstract

We propose TR0N, a highly general framework to turn pre-trained unconditional generative models, such as GANs and VAEs, into conditional models. The conditioning can be highly arbitrary, and requires only a pre-trained auxiliary model. For example, we show how to turn unconditional models into class-conditional ones with the help of a classifier, and also into text-to-image models by leveraging CLIP. TR0N learns a lightweight stochastic mapping which "translates" between the space of conditions and the latent space of the generative model, in such a way that the generated latent corresponds to a data sample satisfying the desired condition. The translated latent samples are then further improved upon through Langevin dynamics, enabling us to obtain higher-quality data samples. TR0N requires no training data nor fine-tuning, yet can achieve a zero-shot FID of 10.9 on MS-COCO, outperforming competing alternatives not only on this metric, but also in sampling speed – all while retaining a much higher level of generality. Our code is available at `https://github.com/layer6ai-labs/tr0n`.

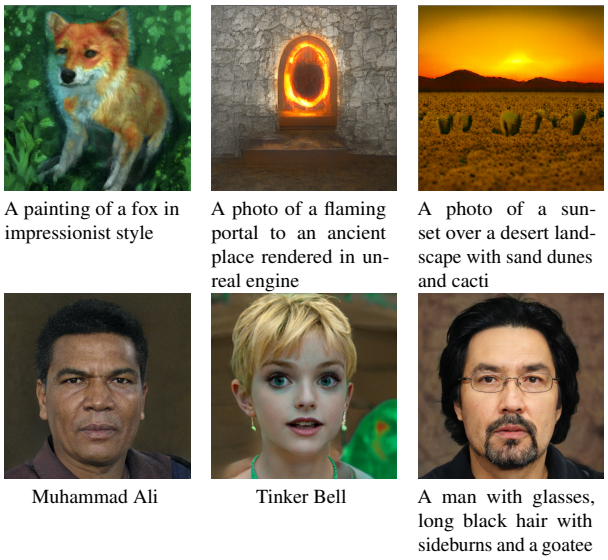


Figure 1. Images generated by TR0N from corresponding text captions, obtained by finding adequate points on the latent space of a pre-trained GAN. Neither fine-tuning nor training data are used. **Top row**: BigGAN pre-trained on ImageNet. **Bottom row**: StyleGAN2 pre-trained on FFHQ.

1. Introduction

Large machine learning models have recently achieved remarkable success across various tasks (Brown et al., 2020; Jia et al., 2021; Nichol et al., 2022; Chowdhery et al., 2022; Rombach et al., 2022; Yu et al., 2022; Ramesh et al., 2022; Saharia et al., 2022; Reed et al., 2022). Nonetheless, training such models requires massive computational resources. Properly and efficiently leveraging existing large pre-trained models is thus of paramount importance. Yet, tractably combining the capabilities of these models in a plug-and-play manner remains a generally open problem. Mechanisms to achieve this task should ideally be modular and model-agnostic, such that one can easily swap out a model component for one of its counterparts (e.g. interchanging a GAN (Goodfellow et al., 2014) for a VAE (Kingma & Welling, 2014; Rezende et al., 2014), or swapping CLIP (Radford et al., 2021) for a new state-of-the-art text/image model).

In this work, we study conditional generation through the lens of combining pre-trained models. Conditional generative models aim to learn a conditional distribution of data given some conditioning variable c. They are typically trained from scratch on pairs of data with corresponding c (e.g. images x, with corresponding class labels or text prompts fed through a language model c) (Mirza & Osindero, 2014; Sohn et al., 2015). Our goal is to take an arbitrary pre-trained unconditional pushforward generative model (Salmona et al., 2022; Ross et al., 2022) – i.e. a model G which transforms latent variables z sampled from a prior $p(z)$ to data samples $x = G(z)$ – and turn it into a conditional model. To this end, we propose TR0N, a

[*]Equal contribution [1]Layer 6 AI, Toronto, Canada [2]University of Toronto, Toronto, Canada [3]Vector Institute, Toronto, Canada. Correspondence to: Zhaoyan Liu <zhaoyan@layer6.ai>, Noël Vouitsis <noel@layer6.ai>, Satya Krishna Gorti <satya@layer6.ai>, Jimmy Ba <jba@cs.toronto.edu>, Gabriel Loaiza-Ganem <gabriel@layer6.ai>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

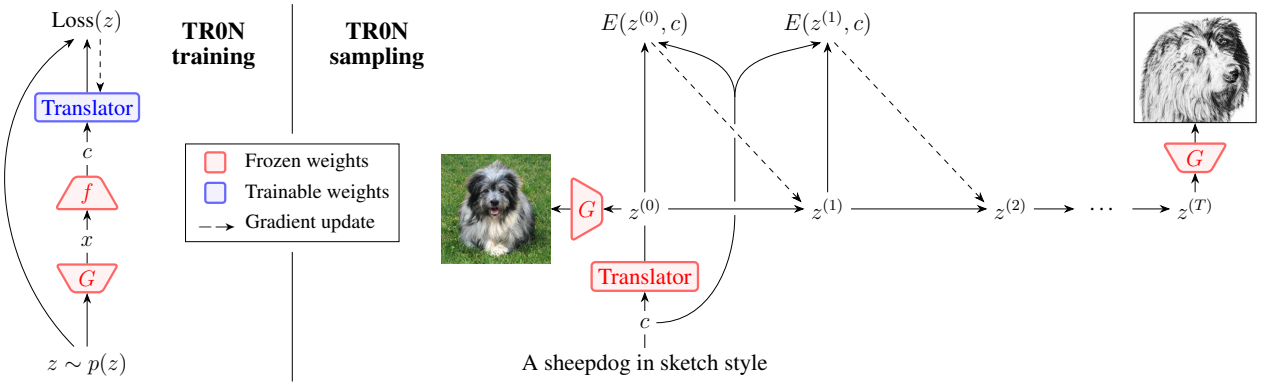


Figure 2. **Left panel**: The stochastic translator network learns to recover z from $c = f(G(z))$. **Right panel**: The (stochastic) output $z^{(0)}$ of the trained translator – which is such that $G(z^{(0)})$ "roughly satisfies" condition c – initializes Langevin dynamics over $E(z, c)$ which further improves the sample so as to better match c. In the depicted example, G is a GAN trained on ImageNet, f the CLIP image encoder, c the CLIP text embedding corresponding to the given text prompt, and $E(z, c)$ the negative cosine similarity between $f(G(z))$ and c.

highly general framework to make pre-trained unconditional generative models conditional. TR0N assumes access to a pre-trained auxiliary model f that maps each data point x to its corresponding condition $c = f(x)$, e.g. f could be a classifier, or a CLIP encoder. TR0N also assumes access to a function $E(z, c)$ such that latents z for which $G(z)$ "better satisfies" a condition c are assigned smaller values. Using this function, for a given c, TR0N performs T steps of gradient minimization of $E(z, c)$ over z to find latents that, after applying G, will generate desired conditional data samples.

However, we show that naïvely initializing the optimization of E is highly suboptimal. With this in mind, TR0N starts by learning a network that we use to better initialize the optimization process. We refer to this network as the translator network since it "translates" from a condition c to a corresponding latent z such that $E(z, c)$ is small, essentially amortizing the optimization problem. Importantly, the translator network is trained *without fine-tuning G or f nor using a provided dataset*. In this sense, TR0N is a zero-shot method wherein the only trainable component is a lightweight translator network. Importantly, TR0N avoids the highly expensive training of a conditional model from scratch and is model-agnostic: we can use any G and any f, which also makes it straightforward to update any of these components whenever a newer state-of-the-art version becomes available. We outline the procedure to train the translator network on the left panel of Figure 2.

Once the translator network is trained, we use its output to initialize the optimization of E. This reclaims any performance lost due to the amortization gap (Cremer et al., 2018; Kim et al., 2018), resulting in better local optima and faster convergence than naïve initialization. In reality, TR0N is a stochastic method: the translator network is a conditional distribution $q_\theta(z|c)$ that assigns high density to latents z such that $E(z, c)$ is small, and we add noise during the gradient optimization of E, which allows us to interpret TR0N as sampling with Langevin dynamics (Welling & Teh, 2011) using an efficient initialization scheme. We exemplify how to sample with TR0N on the right panel of Figure 2.

Our contributions are: (i) introducing translator networks and a particularly efficient parameterization of them, allowing for various ways to initialize Langevin dynamics; (ii) framing TR0N as a highly general framework, whereas previous related works mostly focus on a single task with specific choices of G and f; and (iii) showing that TR0N empirically outperforms competing alternatives across tasks in image quality and computational tractability, while producing diverse samples; and that it can achieve an FID (Heusel et al., 2017) of 10.9 on MS-COCO (Lin et al., 2014).

2. Background

Joint text/image models In this work, we leverage pre-trained joint text/image models as a particular choice for both the auxiliary model f and to construct E, enabling TR0N to be conditioned on either free-form text prompts or on image semantics. Recent joint text/image models such as CLIP learn a joint representation space $\mathcal{C}_{\text{CLIP}}$ for images and texts. CLIP includes an image encoder $f^{\text{img}} : \mathcal{X} \to \mathcal{C}_{\text{CLIP}}$ and a text encoder $f^{\text{txt}} : \mathcal{T} \to \mathcal{C}_{\text{CLIP}}$, where $\mathcal{X}$ is the space of images and $\mathcal{T}$ is the space of text prompts, which are trained in such a way that images and texts that are semantically aligned are mapped to similar representations. More specifically, CLIP is such that the negative cosine similarity $U_{\text{sim}}(f^{\text{img}}(x), f^{\text{txt}}(y))$ is small for semantically aligned image/text pairs $(x, y) \in \mathcal{X} \times \mathcal{T}$, and large for semantically unaligned pairs, where $U_{\text{sim}}(c', c) = -c^\top c' / (\|c'\|_2 \|c\|_2)$.

Pushforward models We use the term *pushforward model* to refer to any generative model whose samples

$x \in \mathcal{X}$ can be obtained as $x = G(z)$, where $z \in \mathcal{Z}$ is a latent variable sampled from some (typically not trainable) prior $p(z)$, and $G : \mathcal{Z} \to \mathcal{X}$ is a neural network. Many models fall into this category, including generative adversarial networks (GANs), variational autoencoders (VAEs), normalizing flows (Dinh et al., 2017; Durkan et al., 2019) and variants thereof (Brehmer & Cranmer, 2020; Caterini et al., 2021; Ross & Cresswell, 2021), and more (Tolstikhin et al., 2018; Loaiza-Ganem et al., 2022). We focus on GANs and VAEs since they use a low-dimensional latent space $\mathcal{Z}$, which will later make the translator network's task easier. Our main goal is to turn a pre-trained unconditional pushforward model $(p(z), G)$ into a conditional model $(p(z|c), G)$.

EBMs and Langevin dynamics We will later formalize the goal of TR0N as sampling from a distribution $p(z|c)$ defined only up to proportionality, i.e. $p(z|c) \propto e^{-\beta E(z,c)}$, where $E : \mathcal{Z} \times \mathcal{C} \to \mathbb{R}$ is called the energy function, and the hyperparameter $\beta > 0$ controls the degree to which small values of $E(z, c)$ correspond to large values of $p(z|c)$, and vice-versa. We hereafter refer to this formulation as an energy-based model (EBM). While the energy function in EBMs is typically learnable (Xie et al., 2016; Du & Mordatch, 2019), in our work we define and fix an energy function that allows us to enforce the requirement that "applying G to a sample from $p(z|c)$ satisfies condition c". Langevin dynamics is a method that allows us to sample from EBMs by constructing a Markov chain $(z^{(0)}, z^{(1)}, \dots)$ given by

$$z^{(t+1)} = z^{(t)} - \frac{\beta\lambda^{(t)}}{2}\nabla_z E\left(z^{(t)}, c\right) + \sqrt{\lambda^{(t)}}\epsilon^{(t)}, \quad (1)$$

where the sequence $(\lambda^{(0)}, \lambda^{(1)}, \dots)$ is a hyperparameter, and $\epsilon^{(t)} \sim \mathcal{N}(\epsilon; 0, I)$. Under mild conditions and by sending $\lambda^{(t)}$ to 0 at an appropriate rate, the limiting distribution of this Markov chain as $t \to \infty$ is $p(z|c)$. Langevin dynamics can be interpreted as gradient descent on E with added noise, and has been successfully applied to sample and train deep EBMs, where in practice it is common to deviate from theory and set $\lambda^{(t)} = \lambda > 0$ for all t (i.e. a single scalar hyperparameter λ is used) for improved empirical performance. Also, while in theory convergence does not depend upon the starting point $z^{(0)}$, in practice this choice can greatly speed up convergence (Hinton, 2002; Nijkamp et al., 2020; Yoon et al., 2021), just as with gradient descent (Boyd & Vandenberghe, 2004; Glorot & Bengio, 2010).

3. TR0N

3.1. Plug-and-play components of TR0N

TR0N requires three key components to ensure that it can operate as a plug-and-play framework. First, TR0N takes an arbitrary pre-trained pushforward model $(p(z), G)$. TR0N also assumes access to a pre-trained auxiliary model $f : \mathcal{X} \to \mathcal{C}$ that maps data to its corresponding condition. For example, if our goal is to condition on class labels, f would be a classifier, and $\mathcal{C}$ the space of probability vectors of appropriate length. If we aim to condition on text, f could be given by the CLIP image encoder f^{img} – although we will see later that a different choice of f led us to improved empirical performance in this setting – and $\mathcal{C}$ the latent space of CLIP, $\mathcal{C}_{\text{CLIP}}$. The final component of TR0N is a function $E : \mathcal{Z} \times \mathcal{C} \to \mathbb{R}$ which measures how much $G(z)$ satisfies condition c, an intuitive choice being

$$E(z, c) = U(f(G(z)), c), \quad (2)$$

where $U : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ measures discrepancy between conditions, for example: when f is a classifier, U could be the categorical cross entropy; and when f is the image encoder from CLIP, U could be the negative cosine similarity, U_{sim}. However, other choices of E are possible, as we will show in our experiments.

3.2. Overview of TR0N

Translator networks TR0N uses the aforementioned components to train the translator network which, given c, aims to output a z with small $E(z, c)$. This can be intuitively understood as amortizing the minimization of E with a neural network so as to not have to run a minimizer from scratch for every c. Since there can be many latents z for which $G(z)$ satisfies c (i.e. $E(z, c)$ is small), we propose to have the translator be a distribution $q_\theta(z|c)$, parameterized by θ. This way, the translator can assign high density to all the latents z such that $E(z, c)$ is small. We will detail how we instantiate $q_\theta(z|c)$ with a neural network in subsection 3.4, but highlight that any choice of conditional density is valid. Importantly, since we have access to the unconditional model $(p(z), G)$, we can generate synthetic data $G(z)$ with $z \sim p(z)$; and since we have access to f, we can obtain the condition corresponding to $G(z)$, namely $c = f(G(z))$. Together, this means that the translator can be trained through maximum likelihood *without the need for a provided training dataset*, through

$$\theta^* = \arg\min_\theta \mathbb{E}_{p(z)}\left[-\log q_\theta\left(z|c = f(G(z))\right)\right]. \quad (3)$$

We summarize the above objective in Algorithm 1.

Error correction The translator is trained to stochastically recover z from $c = f(G(z))$, so that intuitively it places high densities on latents which have low $E(z, c)$ values. Yet, the translator is not directly trained to minimize E, and thus having an error correction step, over which E is explicitly optimized, is beneficial to further improve its output. Thus, for a given c, we run T steps of gradient descent on $E(z, c)$ over z, which we initialize with the help of the translator. Initializing optimization with $q_{\theta^*}(z|c)$ rather than

Algorithm 1 TR0N training

Input: $p(z)$, G, f, $q_\theta(z|c)$, and batch size B
while not converged **do**
  Sample $z_i \sim p(z)$ for $i = 1, \dots, B$
  $c_i \leftarrow f(G(z_i))$ for $i = 1, \dots, B$
  $\Delta \leftarrow \nabla_\theta \frac{1}{B} \sum_{i=1}^{B} -\log q_\theta(z_i|c_i)$
  Use Δ to update θ, e.g. with ADAM (Kingma & Ba, 2015)
end while

Algorithm 2 TR0N sampling

Input: c, $q_{\theta^*}(z|c)$, M, E, T, β, λ, G
Output: $G(z^{(T)})$
Initialize $z^{(0)}$ with $q_{\theta^*}(z|c)$, e.g. $z_m \sim q_{\theta^*}(z|c)$ for $m = 1, \dots, M$, and $z^{(0)} = \arg\min_{z_m} E(z_m, c)$
for $t = 0$ **to** $T - 1$ **do**
  Sample $\epsilon^{(t)} \sim \mathcal{N}(\epsilon; 0, I)$
  $z^{(t+1)} = z^{(t)} - \frac{\beta\lambda}{2} \nabla_z E\left(z^{(t)}, c\right) + \sqrt{\lambda}\epsilon^{(t)}$
end for

naïvely (e.g Gaussian noise) significantly speeds up convergence, and as we will see in our experiments, can also lead to better local optima. Importantly, we can use the translator in various ways to initialize optimization. For example, we can sample M times from $q_{\theta^*}(z|c)$, and use the sample with the lowest $E(z, c)$ value (which would be impossible if the translator was deterministic). We will detail another way to leverage the translator network to initialize optimization in subsection 3.4. In practice, we add Gaussian noise to gradient descent. Together with the stochasticity of the translator, this ensures diverse samples. Lastly, we transform the final latent, $z^{(T)}$, through G to obtain a conditional sample from TR0N. We summarize this procedure in Algorithm 2.

3.3. TR0N as an EBM sampler

TR0N can be formalized as sampling from an EBM with Langevin dynamics. Defining the distribution $p(z|c)$, which we call the *conditional prior*, as $p(z|c) \propto e^{-\beta E(z,c)}$, Algorithm 2 uses Langevin dynamics (1) to sample from $p(z|c)$, initialized with the help of $q_{\theta^*}(z|c)$. Thus, TR0N can be interpreted as a sampling algorithm for the conditional push-forward model $(p(z|c), G)$. Again, G remains fixed throughout, and conditioning is achieved only through the prior $p(z|c)$. In this view, the translator network $q_{\theta^*}(z|c)$ can be understood as a rough approximation to $p(z|c)$, as both of these distributions assign large densities to latents z for which $E(z, c)$ is small. This is precisely why the translator provides a good initialization for Langevin dynamics: the more $z^{(0)}$ "comes from $p(z|c)$", the faster (1) will converge.

Why maximum-likelihood? If our goal is for the translator to be close to the conditional prior, i.e. $q_{\theta^*}(z|c) \approx p(z|c)$, then a natural question is why train the translator through (3), which does not involve $p(z|c)$, rather than by minimizing some discrepancy between these two distributions? The answer is that, since the target $p(z|c)$ is specified only up to proportionality and true samples from it are not readily available (better sampling from $p(z|c)$ is in fact what we designed TR0N to achieve), minimizing commonly-used discrepancies such as the KL divergence or the Wasserstein distance is not tractable. The only discrepancy we are aware of that could be used in this setting is the Stein discrepancy, which has also been used to train EBMs (Grathwohl et al., 2020). However, in preliminary experiments we observed very poor results by attempting to minimize this discrepancy. In contrast, the maximum-likelihood objective (3) is straightforward to optimize, and obtained strong empirical performance in our experiments.

3.4. GMMs to parameterize translator networks

While clearly any choice of conditional density model $q_\theta(z|c)$ can be used in TR0N, we choose a Gaussian mixture model (GMM), as it has several advantages that we will discuss shortly. More specifically, we use a neural network, parameterized by η, which maps conditions $c \in \mathcal{C}$ to the mean $(\mu_{\eta,k}(c))_{k=1}^K \in \mathcal{Z}^K$ and weight $w_\eta(c) \in \mathbb{R}^K$ parameters of a Gaussian mixture, i.e.

$$q_\theta(z|c) = \sum_{k=1}^{K} w_{\eta,k}(c) \mathcal{N}(z; \mu_{\eta,k}(c), \text{diag}(\sigma^2)), \quad (4)$$

where $w_\eta(c)$ has positive entries which add up to one (enforced with a softmax), and $\theta = (\eta, \sigma)$, i.e. σ is learnable. We use a simple multilayer perceptron with multiple heads to parameterize this neural network.

Our GMM choice for the stochastic translator has four important benefits: (i) It is a very lightweight model, and thus achieves our goal of being much more tractable to train than any of the pre-trained components G and f, which we once again highlight remain fixed throughout. (ii) Sampling from a GMM is very straightforward and can be done very quickly. (iii) Empirically, we found that using more complicated density models $q_\theta(z|c)$ such as normalizing flows did not result in improved performance. We hypothesize that, since Langevin dynamics acts as an error correction step, $q_{\theta^*}(z|c)$ just needs to approximate, rather than perfectly recover, $p(z|c)$. (iv) Finally, taking $q_\theta(z|c)$ as a GMM allows using the translator to initialize Langevin dynamics in ways that are not straightforward to extend to a non-GMM setting. In particular, we found that sometimes (when diversity is not as paramount), rather than initializing (1) as described

in Algorithm 2, better performance could be achieved by directly using the GMM parameters. That is, we initialize at the GMM mean, $z^{(0)} = \sum_k w_{\eta^*,k}(c)\mu_{\eta^*,k}(c)$. Note that the mean of more complex distributions might not be so easily computable. Further, we found that when initializing this way, optimizing the weights and means directly yielded better performance, i.e we write $z^{(t)}$ as $z^{(t)} = \sum_k w_k^{(t)}\mu_k^{(t)}$, and perform Langevin dynamics as

$$(w^{(t+1)}, \mu^{(t+1)}) = \\ (w^{(t)}, \mu^{(t)}) - \frac{\beta\lambda}{2}\nabla_{(w,\mu)}E\left(z^{(t)}, c\right) + \sqrt{\lambda}\epsilon^{(t)}, \tag{5}$$

where $w^{(0)} = w_{\eta^*}(c)$, $\mu_k^{(0)} = \mu_{\eta^*,k}(c)$ for $k = 1, \ldots, K$, and the size of $\epsilon^{(t)}$ is appropriately changed from (1).

3.5. TR0N for Bayesian inference

In some settings, the auxiliary model f might provide a probabilistic model $p(c|x)$. For example, when f is a classifier, $p(c|x) = f_c(x)$.[1] Combined with the pushforward model, this provides a latent/data/condition joint distribution $p(z, x, c) = p(z)\delta_{G(z)}(x)p(c|x)$, where $\delta_{G(z)}(x)$ denotes a point mass on x at $G(z)$. For Bayesian inference, it might be of interest to sample from the corresponding posterior $p(x|c)$, which is equivalent to sampling from $p(z|c)$ and transforming the result through G. That is, in this scenario, the conditional prior $p(z|c)$ is a proper posterior distribution of latents given a condition. TR0N can sample from this posterior by using specific choices of β and E. While these choices provide a probabilistically principled way of combining $(p(z), G)$ and f into a conditional model, we find that non-Bayesian choices obtain stronger empirical results. We nonetheless believe that TR0N being compatible with Bayesian inference is worth highlighting. Due to space constraints, we include additional details in Appendix A.

4. Related Work

Several methods aim to obtain a conditional generative model by combining pre-trained models, although none of them shares all of the advantages of TR0N. Notably, almost all the works we discuss below are shown to work for a single task, unlike TR0N which is widely applicable.

Non-zero-shot methods Zhou et al. (2021a) and Wang et al. (2022) leverage CLIP to train text-to-image models without text data, but unlike TR0N, still require a training dataset of images and relatively longer training times. Wang & Torr (2022) propose a method to turn a classifier into a conditional generative model which also requires training data to train a masked autoencoder. Nie et al. (2021) condition GANs through a similar EBM as us, but use data to train f, do not condition on text, and do not use translator networks. Zhang & Agrawala (2023) add conditioning to pre-trained diffusion models (Ho et al., 2020), but require training data to do so.

Deterministic optimization The works of Nguyen et al. (2016), Liu et al. (2021), Patashnik et al. (2021), and Li et al. (2022b) can be thought of as deterministic versions of our EBM, where rather than sampling from $p(z|c) \propto e^{-\beta E(z,c)}$, the energy $E(z, c)$ is directly minimized over z. These methods do not account for the fact that there can be many latents z such that $G(z)$ satisfies condition c, and thus can be less diverse than TR0N. Additionally, these methods do not have a translator network, and with the exception of FuseDream (Liu et al., 2021), naïvely initialize optimization, resulting in reduced empirical performance and needing more gradient steps for optimization to converge. We also note that FuseDream's initialization scheme – which we detail in Appendix B for completeness – requires many forward passes through G and f, and remains much more computationally demanding than TR0N's.

Stochastic methods Ansari et al. (2021) apply Langevin dynamics on the latent space of a GAN, but do so to iteratively refine its samples, rather than for conditional sampling. Nguyen et al. (2017) use a similar EBM to ours, but do not use a translator network and initialize Langevin dynamics naïvely, once again resulting in significantly decreased empirical performance as compared to TR0N. Wu et al. (2022) also define a similar EBM to ours, which is approximated with a normalizing flow for each different c, meaning that a different model has to be trained for each condition, resulting in a method that is far less scalable than TR0N. Finally, Pinkney & Li (2022) propose clip2latent, which can be understood as using a diffusion model instead of a GMM as the translator network, making clip2latent more expensive to train than TR0N. Importantly, they perform no error correction step whatsoever, and thus do not leverage important information contained in the gradient of E.

5. Experiments

All our experimental details are provided in Appendix B.

5.1. Conditioning on class labels

We demonstrate TR0N's ability to make an unconditional model on CIFAR-10 (Krizhevsky, 2009) into a class-conditional one. To highlight the flexibile plug-and-play nature of TR0N, we use two different pushforward models G: an NVAE (Vahdat & Kautz, 2020), and an AutoGAN (Gong et al., 2019) – we use this somewhat non-standard choice of GAN since most publicly available GANs pre-trained

[1] We slightly abuse notation here and use c interchangeably as either a one-hot vector, or as the corresponding integer index.

Figure 3. Samples from NVAE (**first panel**), TR0N:NVAE+ResNet50 (**second panel**), AutoGAN (**third panel**), and TR0N:AutoGAN+ResNet50 (**fourth panel**). Rows on the second and fourth panels correspond to classes: TR0N learns to diversely sample in a class-conditional way, while retaining the image quality of the underlying unconditional model. Best viewed while zoomed-in.

on CIFAR-10 are class-conditional. Here, $\mathcal{C}$ is the space of probability vectors of length 10, we take f as a ResNet50 classifier (He et al., 2015), and use E as in (2) with U given by the cross-entropy loss, $U_{\text{ent}}(c', c) := -\sum_j c_j \log c'_j$.

Figure 3 shows qualitative results: we can see that, for both pushforward models, TR0N not only obtains samples from each of the 10 classes, but that it achieves this without sacrificing neither image quality nor diversity.

We also make quantitative comparisons between each unconditional model (i.e. NVAE and AutoGAN) and the resulting conditional models provided by TR0N. To make the comparison equitable, we sample unconditionally from TR0N models by first sampling one of the 10 classes uniformly at random, and then sampling from the corresponding conditional. Results are shown in Table 1, by measuring image quality and diversity through both the FID score and the inception score (Salimans et al., 2016), and the quality of conditioning through the average probability that the ResNet50 assigns to the intended class of TR0N samples. TR0N not only makes the models conditional as these probabilities are very close to 1, especially for the AutoGAN-based model, but it also improves their FID and inception scores (IS): TR0N leverages the classifier f not only to make a conditional model, but also to improve upon its underlying pre-trained pushforward model.

Table 1 also includes some ablations: (i) removing the error correction (Langevin dynamics) step altogether, which results in heavily degraded FID and IS for the NVAE-based model, and much worse conditioning for both models; (ii) removing the translator, which is equivalent to a stochastic version (i.e. with Langevin dynamics instead of gradient descent) of the method of Nguyen et al. (2016), and which significantly hurts FID, IS, and conditioning performance, highlighting the relevance of translator networks; (iii) using a deterministic translator rather than a stochastic one (see Appendix B for details), which significantly hurts FID and IS due to a lack of diversity since Langevin dynamics is always initialized at the same point for a given condition; and (iv) using ADAM instead of gradient descent to update latents in Algorithm 2, which not only removes the formal interpretation of TR0N as an EBM sampler, but also worsens performance across metrics. Finally, we include additional results in Appendix C using the Bayesian choice of β and E mentioned in subsection 3.5.

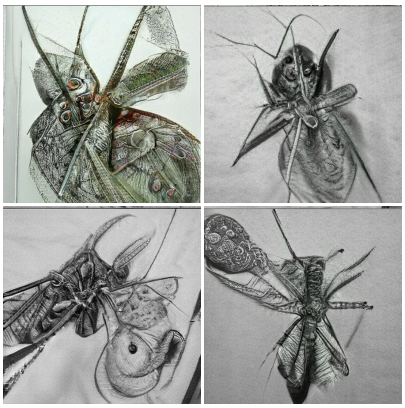
Table 1. FID, IS, and average probability assigned to the intended class of generated samples by a ResNet50 on CIFAR-10. "no EC" stands for "no error correction", "no T" for "no translator", "DT" for "deterministic translator", and "ADAM" and for changing the optimizer in Langevin dymanics.

Model	FID ↓	IS ↑	Avg. prob. ↑
NVAE	41.70	6.95	–
TR0N:NVAE+ResNet50	**19.79**	**8.64**	**0.75**
TR0N:NVAE+ResNet50 (no EC)	40.80	6.95	0.20
TR0N:NVAE+ResNet50 (no T)	36.74	7.75	0.54
TR0N:NVAE+ResNet50 (DT)	77.11	7.15	0.48
TR0N:NVAE+ResNet50 (ADAM)	20.24	8.31	0.51
AutoGAN	12.45	8.53	–
TR0N:AutoGAN+ResNet50	**10.69**	**8.91**	0.95
TR0N:AutoGAN+ResNet50 (no EC)	11.00	8.66	0.41
TR0N:AutoGAN+ResNet50 (no T)	14.30	8.37	0.68
TR0N:AutoGAN+ResNet50 (DT)	123.23	5.48	**0.97**
TR0N:AutoGAN+ResNet50 (ADAM)	11.08	8.88	0.93

5.2. Conditioning on text

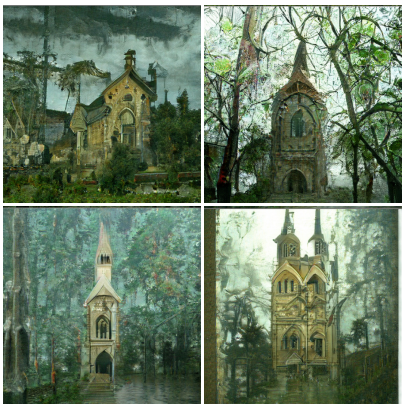
Natural images We now show TR0N's capability to turn unconditional models into text-to-image models. Here, we use $\mathcal{C}$ as the latent space of CLIP, $\mathcal{C}_{\text{CLIP}}$, and to condition on a text prompt $y \in \mathcal{T}$, we simply use the text encoder, $c = f^{\text{txt}}(y)$. First, we take G as a BigGAN[2] (Brock et al., 2018) pre-trained on ImageNet (Deng et al., 2009), and use two different choices of f leveraging CLIP. The first choice is simply the image encoder of CLIP, f^{img}. We focus our

[2]While BigGAN is a class-conditional model, it is not text-conditional. We include the class condition on $\mathcal{Z}$ and think of the GAN as unconditional. See Appendix B for details.



A pencil drawing of an insect, abstract, surrealism

A beach with crystal clear water and palm trees, with snow-capped mountains in the background



A painting of the middle-aged, gothic church surrounded with trees, under the rainy weather

Figure 4. Samples from TR0N:BigGAN+CLIP (BLIP). We can see samples are diverse for all captions.

comparisons against FuseDream – which to the best of our knowledge is the best performing competing method.[3]

As our second choice of f, we also leverage a pre-trained caption model $h : \mathcal{X} \to \mathcal{T}$ followed by CLIP's text encoder, i.e. $f = f^{\text{txt}} \circ h$, further demonstrating the plug-and-play nature of TR0N. The idea behind this choice is that CLIP's image and text encoders have been shown to not perfectly map images and text to the same regions of $\mathcal{C}_{\text{CLIP}}$ (Liang et al., 2022). Adding the caption model h – which maps images to text descriptions – allows us to use the text encoder within f, i.e. the same encoder used to obtain c, resulting in better matching latents. This choice of f is a novel empirical contribution for zero-shot text-to-image generation. We use BLIP (Li et al., 2022a) for the caption model h. For both choices of f, we follow FuseDream and use the negative augmented CLIP score E_{CLIP} as E, which is given by $E_{\text{CLIP}}(z, c) \coloneqq \mathbb{E}_{p(\phi)}[U_{\text{sim}}(f^{\text{img}}(\phi[G(z)]), c)]$, where $\phi[x]$ is a differentiable data-augmentation (Zhao et al., 2020) of x, and $p(\phi)$ a pre-specified distribution over data-augmentations. Like Liu et al. (2021), we find that using the data augmentations helps avoid adversarial examples with small values of $E(z, c)$ which nonetheless do not satisfy c. Note that E_{CLIP} always uses the image encoder from CLIP, regardless of which f we use to train the translator network.

We compare TR0N against FuseDream on the MS-COCO dataset, which contains text/image pairs. For each text, we generate a corresponding image with both methods, and then compute both the FID and augmented CLIP score. Results are displayed in Figure 5 for various computational budgets (the higher the budget, the bigger T, i.e. the longer Langevin dynamics is iterated for). As a consequence of FuseDream's expensive initialization scheme, TR0N can achieve similar performance much faster. This is true for our first choice of f, where TR0N uses the same components as FuseDream (red vs orange lines), emphasizing once again the relevance of the translator, as also evidenced by the light blue lines in Figure 5, which correspond to TR0N with no translator (or equivalently, FuseDream with naïve initialization). It is also true for our second choice of f (with a caption model), which allows TR0N to not only be faster than FuseDream (which cannot incorporate this f as it has no translator), but also outperform it (blue vs orange lines). We once again perform ablations over different design choices of the translator, which we include in Appendix C.

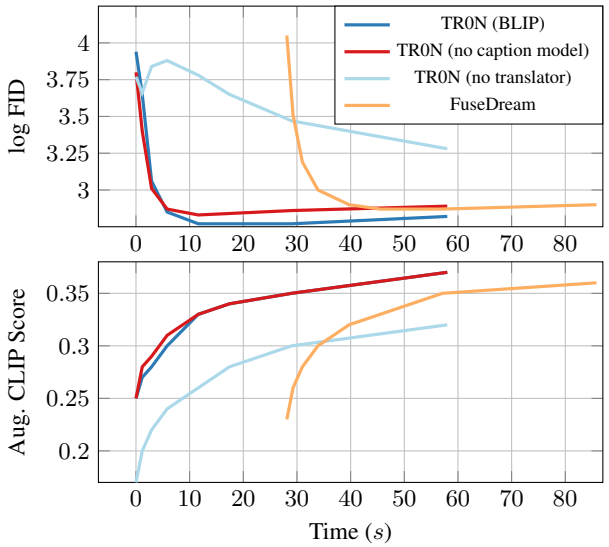


Figure 5. Comparisons between TR0N:BigGAN+CLIP and FuseDream on MS-COCO as a function of time required to generate a sample. **Top panel**: FID score (in log scale), lower is better. **Bottom panel**: augmented CLIP score, higher is better.

[3]While FuseDream is a deterministic method, its provided implementation (optionally) adds noise during gradient optimization: `https://github.com/gnobitab/FuseDream`.

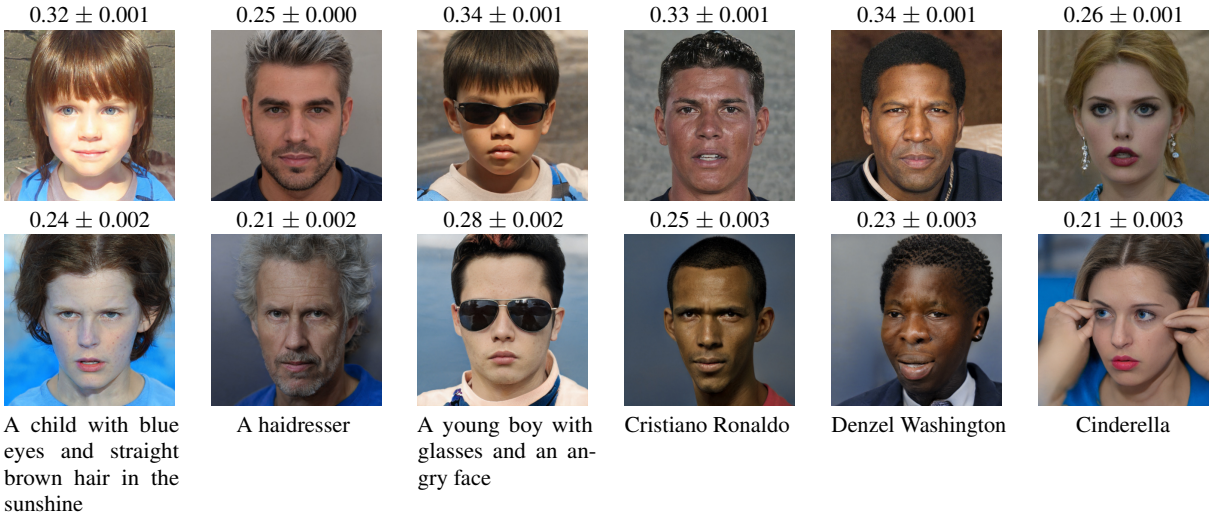


Figure 6. Comparison between TR0N:StyleGAN2+CLIP (**top row**) and clip2latent (**bottom row**). Numbers above each image correspond to average augmented CLIP score (higher is better) plus/minus standard error over 10 samples from the given caption. Thanks to the error correction step, TR0N better semantically matches the input text in its generated images than clip2latent.

Figure 1 and Figure 4 show text-to-image samples from TR0N. Although BigGAN was trained on ImageNet and remains fixed throughout, the images that TR0N manages to produce from it using text prompts are highly out-of-distribution for this dataset: TR0N's ability to efficiently leverage CLIP to explore the GAN's latent space $\mathcal{Z}$ is noteworthy. We include additional samples in Appendix C, showing both how images evolve throughout Langevin dynamics, and failure cases of TR0N.

By using the same G and version of CLIP as FuseDream, the previous experiments show that TR0N outperforms it thanks to its methodology, rather than an improved choice of networks. Yet, these networks can be improved. To further strengthen TR0N, we upgrade: G to a StyleGAN-XL (Sauer et al., 2022) – also pre-trained on ImageNet, CLIP to its LAION2B (Schuhmann et al., 2022) version, and the caption model to BLIP-2 (Li et al., 2023) (using BLIP-2 instead of BLIP as in other experiments again highlights the plug-and-play nature of TR0N). Table 2 shows quantitative results, where we can see that these updates significantly boost the performance of TR0N, to the point of making it competitive with very large models requiring text/image data and much more compute to train. While this StyleGAN-XL-based version of TR0N achieves particularly strong results on MS-COCO in terms of FID, we find that the images it produces are not consistently better, visually, than those from the BigGAN-based model. Samples and further discussion can be found in Appendix C.

Facial images To further highlight the wide applicability of TR0N, we show it can be used for other text-to-image tasks. We now use a StyleGAN2 (Karras et al., 2020) and an NVAE as G, both pre-trained on FFHQ (Karras et al., 2019).

Table 2. FID score on MS-COCO. The top part of the table shows models trained directly for text-to-image generation using paired text/image data (these are sometimes called zero-shot as they were not trained on MS-COCO, but are not zero-shot in the same way as TR0N). The bottom part shows zero-shot methods that require only pre-trained models and no provided dataset.

Model	FID ↓
DALL-E (Ramesh et al., 2021)	$\approx 27.5^{\dagger}$
StyleGAN-T (Sauer et al., 2023)	$13.9^{\dagger}$
Latent Diffusion (Rombach et al., 2022)	$12.6^{\dagger}$
GLIDE (Nichol et al., 2022)	$12.2^{\dagger}$
DALL-E 2 (Ramesh et al., 2022)	$10.4^{\dagger}$
Imagen (Saharia et al., 2022)	$7.3^{\dagger}$
Parti (Yu et al., 2022)	$\mathbf{7.2}^{\dagger}$
FuseDream (Liu et al., 2021)	16.3^{*}
TR0N:BigGAN+CLIP (BLIP)	15.0
TR0N:StyleGAN-XL+LAION2BCLIP (BLIP-2)	**10.9**

† Score as reported by the authors, not computed by us.
* Liu et al. (2021) report an FID of 21.9 since they use ADAM instead of Langevin dynamics.

We use CLIP's image encoder f^{img} as f (we do not use a caption model here as the descriptions of faces it outputs are too generic to be useful), and use the negative augmented clip score E_{CLIP} as E. We compare against clip2latent, which uses the same setup with the StyleGAN2, but with a diffusion model instead of a GMM as a translator network, and no error correction procedure.

Figure 1 and Figure 6 show qualitative results. We can see that TR0N produces images that are much more semantically aligned with the input text, which further corroborates that using a GMM as the translator is enough, while also emphasizing the relevance of error-correcting through Langevin dynamics. We highlight that the pushforward models were pre-trained on FFHQ – not CelebA (Liu et al.,

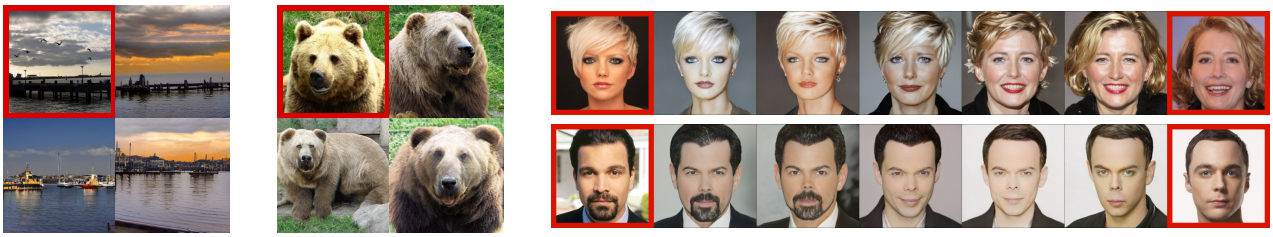


Figure 7. TR0N samples conditioning on image semantics with G as a BigGAN (**first and second panels**, x' is highlighted in red), and interpolations with G as a StyleGAN2 (**third panel**, x'_1 and x'_2 are highlighted in red).

2015) – and thus likely have not seen celebrities such as Cristiano Ronaldo, Denzel Washington, and Muhammad Ali: we believe TR0N's performance is once again noteworthy. We omit large scale quantitative comparisons here because of several reasons: First, text descriptions of FFHQ images are highly generic, which makes it challenging to compute FID against FFHQ. Second, the FID score has recently been shown to be particularly poor at evaluating facial images (Kynkäänniemi et al., 2022). We thus only include the average augmented CLIP score for the used text prompts in Figure 6. We include additional samples for the NVAE-based TR0N model in Appendix C.

5.3. Conditioning on image semantics

We follow Ramesh et al. (2022) and consider two tasks which involve conditioning on image semantics: For the first, given an image x', the goal is to generate diverse images x which share semantics with x'. Here, $\mathcal{C}$ is still the latent space of CLIP, $\mathcal{C}_{\text{CLIP}}$, and f is CLIP's image encoder, f^{img}. Instead of obtaining conditions c from a text prompt, we take $c = f^{\text{img}}(x')$. We use both BigGAN and StyleGAN2 as G, and still use the negative augmented CLIP score, E_{CLIP}, as E. For the second task, instead of computing c from a single image x', we compute it by interpolating between the encodings $f^{\text{img}}(x'_1)$ and $f^{\text{img}}(x'_2)$ of two given images, x'_1 and x'_2. Results are shown in Figure 7, where we can see that TR0N produces meaningful samples and interpolations: this highlights that TR0N allows for arbitrary conditioning – not just class labels or text prompts. We show additional samples in Appendix C.

6. Conclusions, limitations, and future work

In this paper we introduced TR0N, a highly general and simple-to-train framework to turn pre-trained unconditional generative models into conditional ones by learning a stochastic map from conditions to latents, whose output is used to initialize Langevin dynamics. TR0N is quick to sample from, outperforms competing methods, and has a remarkable ability to generate images outside of the distribution used to train G. Despite the empirical performance of TR0N being good, it is inevitably limited by that of the pre-trained model $(p(z), G)$. Diffusion models have been shown to outperform GANs, but have no low-dimensional latent space $\mathcal{Z}$ that the translator can map to, and thus applying TR0N in this setting is not straightforward.

We thus believe extending TR0N to diffusion models to be an interesting direction for future work. We also hope that our ideas can be extended to initialize Langevin dynamics in other EBM settings. Given our results on CIFAR-10 where TR0N improved upon its pre-trained unconditional model, we also believe that further exploring how large pre-trained models can be used to improve upon existing generative models – rather than endowing them with conditional capabilities – to be a promising research avenue. Finally, here we focused exclusively on generating images, but combining large pre-trained models is of interest outside of this task. For example, zero-shot conditional text generation (Su et al., 2022) is a relevant problem, and we hope that the ideas behind TR0N can be extended to this task.

Broader impact Generative models have many applications, including among others: audio generation (van den Oord et al., 2016; Engel et al., 2017), chemistry (Gómez-Bombarelli et al., 2018), neuroscience (Sussillo et al., 2016; Gao et al., 2016; Loaiza-Ganem et al., 2019), and text generation (Bowman et al., 2016; Devlin et al., 2019; Brown et al., 2020). Each of these applications can have meaningful and positive effects on society, but can also be potentially misused for unethical purposes. Text-to-image generation is no exception, and thus the possibility exists that TR0N could be misemployed to generate inappropriate or deceitful content. We do highlight however that other powerful text-to-image models exist and are publicly available, and as such we do not foresee TR0N enabling nefarious actors to abuse text-to-image models in previously unavailable ways.

Acknowledgements

We thank Harry Braviner for early discussions, Anthony Caterini for comments on a preliminary draft, and the anonymous reviewers, whose feedback helped improve our paper.

References

Ansari, A. F., Ang, M. L., and Soh, H. Refining deep generative models via discriminator gradient flow. *ICLR*, 2021.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pp. 10–21. Association for Computational Linguistics (ACL), 2016.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brehmer, J. and Cranmer, K. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33:442–453, 2020.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *ICLR*, 2018.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Caterini, A. L., Loaiza-Ganem, G., Pleiss, G., and Cunningham, J. P. Rectangular flows for manifold learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pp. 1078–1086. PMLR, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *ICLR*, 2017.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32:3608–3618, 2019.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural Spline Flows. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pp. 1068–1077. PMLR, 2017.

Gao, Y., Archer, E. W., Paninski, L., and Cunningham, J. P. Linear dynamical neural population models through nonlinear embeddings. *Advances in neural information processing systems*, 29, 2016.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Gong, X., Chang, S., Jiang, Y., and Wang, Z. Autogan: Neural architecture search for generative adversarial networks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3223–3233, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., and Zemel, R. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, pp. 3732–3747. PMLR, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pp. 4904–4916. PMLR, 2021.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pp. 2678–2687. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding Variational Bayes. *ICLR*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

Kynkäänniemi, T., Karras, T., Aittala, M., Aila, T., and Lehtinen, J. The role of imagenet classes in fr\'echet inception distance. *arXiv preprint arXiv:2203.06026*, 2022.

Li, J., Li, D., Xiong, C., and Hoi, S. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*, 2022a.

Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

Li, S., Du, Y., Tenenbaum, J. B., Torralba, A., and Mordatch, I. Composing ensembles of pre-trained models via iterative consensus. *arXiv preprint arXiv:2210.11522*, 2022b.

Liang, W., Zhang, Y., Kwon, Y., Yeung, S., and Zou, J. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *arXiv preprint arXiv:2203.02053*, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Liu, X., Gong, C., Wu, L., Zhang, S., Su, H., and Liu, Q. Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. *arXiv preprint arXiv:2112.01573*, 2021.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Loaiza-Ganem, G., Perkins, S., Schroeder, K., Churchland, M., and Cunningham, J. P. Deep random splines for point process intensity estimation of neural population data. *Advances in Neural Information Processing Systems*, 32, 2019.

Loaiza-Ganem, G., Ross, B. L., Cresswell, J. C., and Caterini, A. L. Diagnosing and fixing manifold overfitting in deep generative models. *Transactions on Machine Learning Research*, 2022. URL `https://openreview.net/forum?id=0nEZCVshxS`.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in neural information processing systems*, 29, 2016.

Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., and Yosinski, J. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4467–4477, 2017.

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., Mcgrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pp. 16784–16804. PMLR, 2022.

Nie, W., Vahdat, A., and Anandkumar, A. Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34:13497–13510, 2021.

Nijkamp, E., Hill, M., Han, T., Zhu, S.-C., and Wu, Y. N. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5272–5280, 2020.

Nukrai, D., Mokady, R., and Globerson, A. Text-only training for image captioning using noise-injected clip. *arXiv preprint arXiv:2211.00575*, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., and Lischinski, D. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2085–2094, 2021.

Pinkney, J. N. M. and Li, C. clip2latent: Text driven sampling of a pre-trained stylegan using denoising diffusion and clip. *ArXiv*, abs/2210.02347, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=1ikK0kHjvj. Featured Certification.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Ross, B. L. and Cresswell, J. C. Tractable density estimation on learned manifolds with conformal embedding flows. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Ross, B. L., Loaiza-Ganem, G., Caterini, A. L., and Cresswell, J. C. Neural implicit manifold learning for topology-aware generative modelling. *arXiv preprint arXiv:2206.11267*, 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Salmona, A., de Bortoli, V., Delon, J., and Desolneux, A. Can push-forward generative models fit multimodal distributions? *arXiv preprint arXiv:2206.14476*, 2022.

Sauer, A., Schwarz, K., and Geiger, A. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Sauer, A., Karras, T., Laine, S., Geiger, A., and Aila, T. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. *arXiv preprint arXiv:2301.09515*, 2023.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C. W., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S. R., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=M3Y74vmsMcY.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Su, Y., Lan, T., Liu, Y., Liu, F., Yogatama, D., Wang, Y., Kong, L., and Collier, N. Language models can see: Plugging visual controls in text generation. *arXiv preprint arXiv:2205.02655*, 2022.

Sussillo, D., Jozefowicz, R., Abbott, L., and Pandarinath, C. Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*, 2016.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. *ICLR*, 2018.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Wang, G. and Torr, P. H. Traditional classification neural networks are good generators: They are competitive with ddpms and gans. *arXiv preprint arXiv:2211.14794*, 2022.

Wang, Z., Liu, W., He, Q., ru Wu, X., and Yi, Z. Clip-gen: Language-free training of a text-to-image generator with clip. *ArXiv*, abs/2203.00386, 2022.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Wu, C. H., Motamed, S., Srivastava, S., and la Torre, F. D. Generative visual prompt: Unifying distributional control of pre-trained generative models. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Gsbnnc--bnw.

Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. A theory of generative convnet. In *International Conference on Machine Learning*, pp. 2635–2644. PMLR, 2016.

Yoon, S., Noh, Y.-K., and Park, F. Autoencoding under normalization constraints. In *International Conference on Machine Learning*, pp. 12087–12097. PMLR, 2021.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., and Wu, Y. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=AFDcYJKhND. Featured Certification.

Zhang, L. and Agrawala, M. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., and Han, S. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33: 7559–7570, 2020.

Zhou, Y., Zhang, R., Chen, C., Li, C., Tensmeyer, C., Yu, T., Gu, J., Xu, J., and Sun, T. Towards language-free training for text-to-image generation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17886–17896, 2021a.

Zhou, Y., Zhang, R., Chen, C., Li, C., Tensmeyer, C., Yu, T., Gu, J., Xu, J., and Sun, T. Lafite: Towards language-free training for text-to-image generation. *arXiv preprint arXiv:2111.13792*, 2021b.

A. TR0N for Bayesian inference

As mentioned in the main manuscript, in the setting where we have access to a probabilistic model $p(c|x)$, a joint distribution $p(z, x, c) = p(z)\delta_{G(z)}(x)p(c|x)$ is implied. The goal of Bayesian inference is to sample from the corresponding conditional (posterior) distribution $p(x|c)$. Since $p(z, x|c) = p(z|c)\delta_{G(z)}(x)$, sampling from $p(z|c)$ and transforming the result through G allows to obtain (z, x) pairs from $p(z, x|c)$. Discarding z, or equivalently marginalizing out z from $p(z, x|c)$, results in samples from $p(x|c)$. In other words, we only need to sample from $p(z|c)$ and transform the result through G in order to perform Bayesian inference. Furthermore, we know that $p(z|c) \propto p(z, c) = \int p(z)\delta_{G(z)}(x)p(c|x)dx = p(z)p(c|x = G(z))$, which means that TR0N can be used to sample from $p(z|c)$ by setting $\beta = 1$ and taking E as E_{Bayes}, where

$$E_{\text{Bayes}}(z, c) = -\log p(z) - \log p(c|x = G(z)). \tag{6}$$

Note that when $p(z) = \mathcal{N}(z; 0, \nu^2 I)$, the first term is just an L_2 penalty, i.e. $-\log p(z) = \frac{1}{2\nu^2}\|z\|_2^2$ (up to an additive constant that does not depend on z and is thus irrelevant for TR0N). As mentioned in the main manuscript, we find that non-Bayesian choices obtain stronger empirical results. This observation is consistent with previous works, e.g. Dhariwal & Nichol (2021) find – in a different context – that equally weighting the density $p(z)$ and classifier $p(c|x = G(z))$ terms as in E_{Bayes} is suboptimal, and that more heavily weighting the classifier term leads to improved empirical performance. As an attempt to improve performance in our experiments in Appendix C, we thus also consider sampling from a distribution proportional to $e^{-E'_{\text{Bayes}}(z,c)}$, where

$$E'_{\text{Bayes}}(z, c) = -\beta_1 \log p(z) - \beta_2 \log p(c|x = G(z)), \tag{7}$$

and β_1 and β_2 are used as hyperparameters instead of β. Note that $\beta_1 = \beta_2 = 1$ corresponds to Bayesian inference.

B. Experimental details

FuseDream's initialization FuseDream (Liu et al., 2021) uses a similar procedure to TR0N to initialize $z^{(0)}$ as described in Algorithm 2 except, since Liu et al. (2021) have no translator, they have to use the prior $p(z)$ to sample candidates z_m. As a result, they need orders of magnitude more samples M in order to obtain a decent initializer (i.e. a sample roughly matching the given condition c). This is costly, as $E_{\text{CLIP}}(z_m, c)$ has to be evaluated for every $m = 1, \ldots, M$, which requires a forward pass for G and for f^{img}. When M is very large – as is required by FuseDream, e.g. $M = 1000$ – this initialization takes a non-negligible amount of time. In particular, FuseDream uses the parameterization from (5), but since no GMM parameters are available, FuseDream uses the K latents with lowest E_{CLIP} out of the M sampled ones to initialize each $\mu_k^{(0)}$, for $k = 1, \ldots, K$, and initializes $w^{(0)}$ randomly.

Deterministic translators For our ablations using a deterministic translator, we specify a neural network $S_\theta : \mathcal{C} \to \mathcal{Z}$ instead of $q_\theta(z|c)$. We train this translator as a regressor with an L_2 loss:

$$\theta^* = \arg\min_\theta \mathbb{E}_{p(z)} \left[\|S_\theta(f(G(z))) - z\|_2^2\right]. \tag{8}$$

NVAE-based models The NVAE model is presented as having a hierarchical latent space $\mathcal{Z}' = \mathcal{Z}_0 \times \cdots \times \mathcal{Z}_L$, whose prior is given by

$$p(z') = p(z_0) \prod_{\ell=1}^{L} p(z_\ell|z_{\ell-1}), \tag{9}$$

where $z' = (z_0, \ldots, z_L)$, $p(z_0)$ is a standard Gaussian, and $p(z_\ell|z_{\ell-1}) = \mathcal{N}(z_\ell; g_\ell(z_{\ell-1}), \Sigma_\ell(z_{\ell-1}))$, where g_ℓ and Σ_ℓ are neural networks. The NVAE model also has a decoder $p(x|z_L)$. In order to produce a data sample, once z_L is sampled from (9), $G'(z_L)$ is computed to obtain a sample on $\mathcal{X}$, where G' is the mean of the decoder $p(x|z_L)$. Thus, the NVAE fits exactly into the framework of a pushforward model with $\mathcal{Z} = \mathcal{Z}_L$, $p(z)$ as the z_L-marginal of (9), and G as G'. Naïvely, we could thus have the translator be a distribution over z_L, $q_\theta(z_L|c)$, and apply TR0N. However, z_L is actually high-dimensional, and we found this approach did not work particularly well. We thus take a slightly different view of the same NVAE model, where $\mathcal{Z} = \mathcal{Z}_0$, $p(z)$ is just $p(z_0)$, and G is now a stochastic map: in order to compute $G(z_0)$, one samples z_ℓ from $p(z_\ell|z_{\ell-1})$ for $\ell = 1, \ldots, L$, until z_L is obtained, and then computes $G'(z_L)$. In this view, the NVAE is a pushforward model with a low-dimensional latent space and a stochastic G, to which we can also apply TR0N. While this approach

worked better, we found that $\mathcal{Z}_0$ generally did not provide sufficient semantic control over generated images due to the added noise from $(p(z_\ell|z_{\ell-1}))_{\ell=1}^L$, essentially rendering conditioning very hard. We thus slightly modify G as follows so as to effectively remove sources of randomness: we fix an index ℓ^*, and deterministically transform z_ℓ until ℓ^* using the mean neural networks, i.e.

$$z_\ell = g_\ell(z_{\ell-1}) \text{ for } \ell = 1, \dots, \ell^*, \tag{10}$$

and only sample the remaining entries of the hierarchy:

$$z_\ell|z_{\ell-1} \sim p(z_\ell|z_{\ell-1}) \text{ for } \ell = \ell^* + 1, \dots, L. \tag{11}$$

Now, $G(z_0)$ is given by $G'(z_L)$, where z_L is now obtained from z_0 from (10) and (11). While the formalism of Langevin dynamics is broken here since $E(z, c)$ cannot be evaluated exactly due to the randomness of G, the intuition behind TR0N remains, and we find that even if for a fixed z_0, different calls to $G(z_0)$ return different random images (i.e. G is stochastic), TR0N provides strong empirical performance in this setting. We also point out that using a single scalar λ in Langevin dynamics – which as mentioned in the main manuscript is a common practice – also formally breaks down the interpretation of EBM sampling. We thus do not see randomness in G as a fundamental limitation and argue that we are performing approximate Langevin dynamics in this setting.

On using $\mathbf{E_{CLIP}}$ and the caption model We note that much like the randomness in G as described above for the NVAE-based models, when we use E_{CLIP}, the formalism of Langevin dynamics breaks since E cannot be exactly evaluated, and only approximated by sampling data augmentations. Similarly to the NVAE case though, we find that TR0N works well regardless, and we see this as approximate Langevin dynamics. Also, the caption model h that we use to perform text conditioning using BigGAN is stochastic, which means that f is stochastic when training the translator.

BigGAN-based models As mentioned in the main manuscript, the BigGAN model is a class-conditional model and its latent space $\mathcal{Z} = \mathcal{Z}_1 \times \mathcal{Z}_2$ has a continuous component $\mathcal{Z}_1$, and a discrete component $\mathcal{Z}_2$ composed of a discrete set of tokens, one per ImageNet class. We found that, while the stochastic GMM translator was fundamental to model $\mathcal{Z}_1$, it was not so for $\mathcal{Z}_2$. We thus used a GMM only for the continuous part of the latent space, $\mathcal{Z}_1$, and a deterministic translator (trained with an L_2 loss as described above) for the discrete part, $\mathcal{Z}_2$. In practice we use a single shared translator, with GMM heads for $\mathcal{Z}_1$ and a regression head for $\mathcal{Z}_2$.

StyleGAN2-based models To ensure that we are directly comparable with clip2latent (Pinkney & Li, 2022), for StyleGAN2-based models, we define $\mathcal{Z}$ as the intermediate latent space of StyleGAN2, $\mathcal{W}$. During training, we first sample Gaussian noise and pass it through the mapping network of StyleGAN2 to obtain a latent $w \in \mathcal{W}$. Note that this procedure implicitly defines the prior $p(w)$ as a pushforward distribution whose density cannot be evaluated, although this is not a problem since training the translator (3) requires only sampling from the prior, not evaluating its density.

StyleGAN-XL-based models StyleGAN-XL is a class-conditional model like BigGAN. However, similar to StyleGAN2-based models, we define $\mathcal{Z}$ as the intermediate latent space of StyleGAN-XL, $\mathcal{W}$, which has already encoded the class conditioning. During training of the translator, to obtain a latent $w \in \mathcal{W}$, we first sample a random noise latent $z_1 \sim \mathcal{N}(z_1; 0, I)$ and a class label $z_2 \in \mathcal{Z}_2$ uniformly at random, then pass them both through StyleGAN-XL's mapping network to recover a desired sample $w \in \mathcal{W}$.

B.1. Details and hyperparameters that are shared across all experiments

Translator architecture We use an MLP as the translator consisting of two hidden layers with a residual connection in between, and ReLU activations after each hidden layer. Both hidden layers have the same number of hidden units H, which we set for each experiment. A set of heads following the second hidden layer then produce the desired outputs. For the GMM heads, we have one head to predict the K Gaussian means $(\mu_{\eta,k}(c))_{k=1}^K \in \mathcal{Z}^K$ (the head outputs the concatenated means which are then reshaped) and another head to predict the mixture weights $w_\eta(c) \in \mathbb{R}^K$. In all experiements, $K = 10$. For the regression heads, we directly predict the latent z. For the GMM translator, we also learn a separate standard deviation σ of the same size as z. To ensure its training remains stable, we instead learn a parameter ν and then recover σ as $\sigma = e^\nu + 10^{-6}$, guaranteeing that $\sigma > 10^{-6}$. We initialize the standard deviation to $\sigma = 0.01$. All other translator weights are randomly initialized using the default PyTorch (Paszke et al., 2019) linear layer initializer.

Translator training In practice, instead of sampling from $p(z)$ at each step during translator training and then feeding the latent sample through G and f as in Algorithm 1, we generate a synthetic dataset of (z, c) pairs ahead of time by sampling $N = 1,000,000$ latents $z \sim p(z)$ and recovering the corresponding conditions $c = f(G(z))$. While not necessary, we perform this step simply to speed up training the translator since, once the synthetic dataset is generated, we no longer need to perform expensive forward passes through G and f. We train the translator for 10 epochs on said synthetic dataset with a batch size $B = 16$. We find in our experiments that the maximum likelihood loss outlined in Algorithm 1 takes values with high orders of magnitude, so we scale the loss with a scalar – we use 10^{-4} – so that we can use standard learning rates schemes. We thus use ADAM to optimize the translator network with a learning rate of 10^{-4} and a cosine scheduler to anneal the learning rate to 0 throughout training. Unless otherwise stated, $p(z)$ is a standard Gaussian in our experiments.

TR0N sampling While Langevin dynamics can be understood as gradient descent with added noise, in practice we use gradient descent with momentum and added noise since it helps empirical performance. We set the momentum to 0.99 and add noise with $\lambda = 10^{-4}$ in all experiments. Unless otherwise stated, we use $T = 100$ steps of Langevin dynamics. We also note that while in (5) we update both w and μ through Langevin dynamics for didactic purposes, we find that applying Langevin dynamics only to μ and that deterministically updating w with ADAM to work better in practice (we use a learning rate for w of 5×10^{-3} for all experiments). Also, we find that expanding each $(w_k)_{k=1}^K$ to be of the same size as μ such that each component of μ has its own weights to be a helpful heuristic.

B.2. Conditioning on class labels

To train the translator, we set $H = 100$. For Langevin dynamics, we use $\beta = 20$ and iterate for $T = 25$ steps for AutoGAN and $T = 50$ steps for NVAE. In addition, we use the initialization described in Algorithm 2 with $M = 5$ for AutoGAN and $M = 10$ for NVAE. Also, for NVAE, we use $p(z)$ as a standard Gaussian with standard deviation of 0.7 and we use $\ell^* = 0$.

B.3. Conditioning on text

As mentioned in the main manuscript, we use two choices of f. The first uses the CLIP image encoder $f = f^{\text{img}}$. For this choice of f, taking inspiration from Zhou et al. (2021b) and Nukrai et al. (2022), we add noise during TR0N training to simulate "pseudo-text embeddings" from image embeddings in the joint latent space of CLIP, $\mathcal{C}_{\text{CLIP}}$. This helps account for the fact that the fixed condition $c = f^{\text{txt}}(y)$ uses the the text encoder, while c_i in Algorithm 1 uses the image encoder, f^{img}. More specifically, we modify (3) to:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{p(z)} \left[-\log q_\theta \left(z | c = f(G(z)) + \gamma\epsilon \right) \right], \tag{12}$$

where $\epsilon \sim \mathcal{N}(\epsilon; 0, I)$. We find $\gamma = 0.2$ to work well in these experiments.

For the second choice of f – which uses a caption model h to obtain $f = f^{\text{txt}} \circ h$ – we do not add Gaussian noise. This is because the caption model produces embeddings that are closer to the text conditions c that the translator will observe during TR0N sampling. As mentioned, we use BLIP (or BLIP-2) as the caption model and leverage nucleus sampling (Holtzman et al., 2019) to generate more diverse and semantically meaningful captions.

In all these experiments, we set $H = 2048$ in the translator's architecture. In terms of TR0N sampling, we perform Langevin dynamics as described in (5). Also, we approximate E_{CLIP} by sampling 50 times from $p(\phi)$ and then taking the average of U_{sim} across these sampled data augmentations. For BigGAN, we use $\beta = 10^5$ and for StyleGAN2, StyleGAN-XL and NVAE, we use $\beta = 2000$.

For BigGAN, we bound the continuous component $\mathcal{Z}_1$ of $\mathcal{Z}$ (see note above) to be in range $[-2, 2]$, similar to FuseDream. When training the translator, we enforce this in a differentiable way using a $\tanh$ activation at the output of the GMM mean and regression heads. Moreover, we can sample from the discrete component $\mathcal{Z}_2$ of $\mathcal{Z}$ by uniformly sampling an ImageNet class. In practice, when generating our synthetic dataset, we generate the same number of samples from each ImageNet class to ensure mode coverage when training the translator. As mentioned before, we use a deterministic translator for $\mathcal{Z}_2$, whose output we deterministically update during TR0N sampling with ADAM using a learning rate of 5×10^{-3}. We therefore only apply (5) to the continuous component $\mathcal{Z}_1$.

For StyleGAN-XL, to sample a class label, we again in practice sample the same number of samples from each ImageNet class when generating the synthetic dataset and use $p(z)$ as outlined above.

Table 3. FID, IS, and average probability assigned to the intended class of generated samples by a ResNet50 on CIFAR-10.

Model	FID ↓	IS ↑	Avg. prob. ↑
NVAE	41.70	6.95	–
TR0N:NVAE+ResNet50 ($\beta_1 = 0$, $\beta_2 = 20$)	**19.79**	**8.64**	**0.75**
TR0N:NVAE+ResNet50 ($\beta_1 = 1$, $\beta_2 = 1$)	20.12	8.40	0.51
TR0N:NVAE+ResNet50 ($\beta_1 = 1$, $\beta_2 = 20$)	20.34	8.62	0.68
AutoGAN	12.45	8.53	–
TR0N:AutoGAN+ResNet50 ($\beta_1 = 0$, $\beta_2 = 20$)	**10.69**	**8.91**	**0.95**
TR0N:AutoGAN+ResNet50 ($\beta_1 = 1$, $\beta_2 = 1$)	10.85	8.86	0.89
TR0N:AutoGAN+ResNet50 ($\beta_1 = 1$, $\beta_2 = 20$)	10.71	8.90	**0.95**

Also, for StyleGAN2, we use $p(z)$ as outlined above. For NVAE, we use $p(z)$ as a standard Gaussian with standard deviation scaled by 0.6 and we use $\ell^* = 7$.

We note that to compute all FID scores in Figure 5, we use the entire validation set of MS-COCO, which contains $40k$ text/image pairs. To compute the FID scores in Table 2, we randomly subsample $30k$ points from the validation set. We do this for valid comparisons, since the papers whose FID we report on the top part of Table 2 use only $30k$ samples. While we find it more principled to use the entire dataset, we find no major difference between these two ways of computing FID. Note also that the FID scores we report on the upper part of Table 2 are from models which produce images at a 256×256 resolution, whereas the BigGAN and StyleGAN-XL models that we use for the lower part of the table produce images at a 512×512 resolution. For the fairest possible comparison, we thus downscale the 512×512 images to 256×256 before computing the FID score.

The experiments that required timing were all run on an NVIDIA TITAN RTX. Note that a single Langevin dynamics step takes the exact same amount of time in TR0N as it does in FuseDream, as we use the same choice of E and K, so that the gains we observe in TR0N are truly due to the translator's efficient initialization, since FuseDream requires $M = 1000$ forward passes through G just to initialize Langevin dynamics. In contrast, since for TR0N we use (5), we need no forward passes through G to initialize Langevin dynamics: a single pass through the translator is enough. For these experiments, we evaluate metrics at $T \in \{0, 10, 25, 50, 100, 150, 250, 500\}$.

B.4. Conditioning on image semantics

To train the translator, we set $H = 2048$. For Langevin dynamics, we use use the same hyperparameters as in text conditioning and we also approximate E_{CLIP} the same way. In addition, we use the initialization described in Algorithm 2 with $M = 1$ for the first task (with a single image), and the mode of the GMM for the second task (interpolation). For BigGAN, we follow the same procedure as described for text conditioning to deal with the continuous and discrete components of $\mathcal{Z}$, and for StyleGAN2 we again use the $\mathcal{W}$ space as before. For the interpolation experiments, we use slerp interpolation.

C. Additional experiments

C.1. Conditioning on class labels through Bayesian inference

We compare in Table 3 the performance of E'_{Bayes} from (7) against the choice from the main manuscript for conditioning on class labels on CIFAR-10, i.e. using (2) with U as the cross entropy loss, $U_{\text{ent}}(c', c) = -\sum_j c_j \log c'_j$. Note that since $U_{\text{ent}}(f(G(z)), c) = -\log p(c|x = G(z))$, using E'_{Bayes} amounts simply to adding an L_2 penalty on z and re-weighting the classifier term. For all experiments, we use the exact same translator: only the energy function used for Langevin dynamics changes. Table 3 includes results for: the energy function from used in the main manuscript ($\beta_1 = 0, \beta_2 = 20$); the fully Bayesian choice ($\beta_1 = \beta_2 = 1$), which as previously mentioned hurts performance; and also for $\beta_1 = 1$ and a tuned value of β_2, which loses the Bayesian interpretation and does not outperform the first choice.

C.2. Ablations for text-to-image results with natural images

As mentioned in the main manuscript, we carry out an ablation over different translators so as to empirically justify our choices. Figure 8 shows results of TR0N using BigGAN for text-to-image generation on MS-COCO for the GMM translator

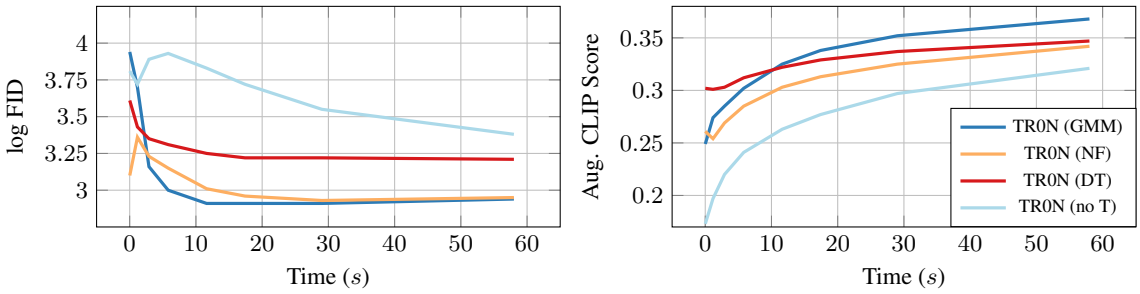


Figure 8. Ablations of translator choice for TR0N:BigGAN+CLIP (BLIP) on MS-COCO.

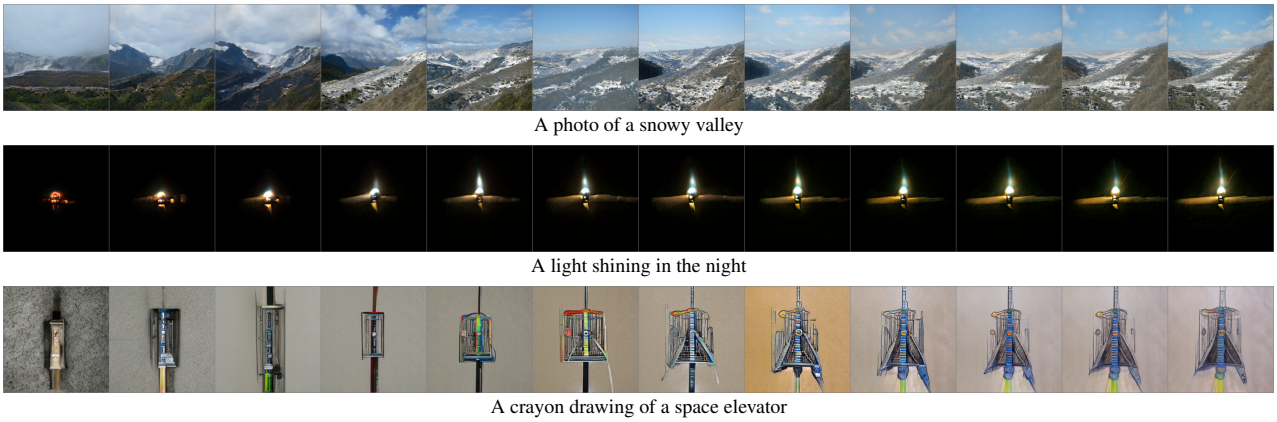


Figure 9. Evolution of samples throughout Langevin dynamics for our TR0N:BigGAN+CLIP (BLIP) model. Each row shows $G(z^{(t)})$ for increasing values of $t \in \{0, 15, 30, 45, 60, 75, 90, 140, 210, 280, 350, 500\}$ for the corresponding text prompt.

that we use on the main text, a normalizing-flow translator (marked as "NF"), a deterministic translator ("DT"), and no translator ("no T", as also shown in the main manuscript). We use the choice of f with a caption model for all translators. Due to computational constraints, we only use $10k$ generated samples to compute metrics (this change accounts for any difference with the numbers shown in the main manuscript). We can see that only the NF translator matches the GMM one in terms of FID score, but only for larger time budgets; while no method beats the GMM translator in augmented CLIP score. Once again, these results confirm our choices for the translator.

Finally, we make a note about the preliminary experiments that we mentioned in the main manuscript using Stein discrepancy to train the translator: we did not manage to stabilize training, and the obtained FIDs were an order of magnitude greater than those of the GMM translator.

C.3. Additional samples

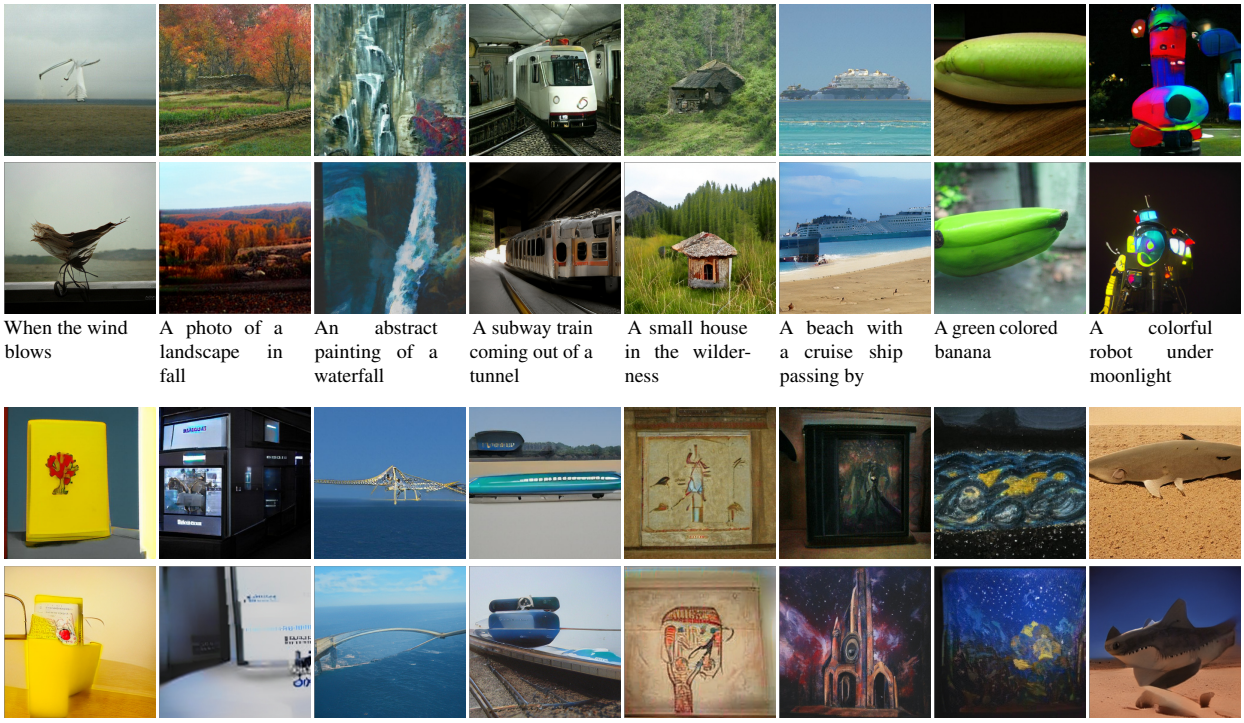
Figure 9 shows how samples evolve throughout Langevin dynamics. We can see that the output of the translator provides a sensible initialization, and that image quality improves throughout Langevin dynamics. Not surprisingly, the output of the translator more closely matches the given text prompt for in-distribution prompts (e.g. "A photo of a snowy valley") than for out-of-distribution prompts (e.g. "A crayon drawing of a space elevator"), although the initialization remains useful for all situations.

Figure 10 shows text prompts for which TR0N fails. In particular, we find that TR0N struggles to produce images which involve: written text (e.g. "A sign that says 'Deep Learning' " or "A photo of the digit number 5"), humans (e.g. "A photo of a girl walking on the street"), understanding the relationship between various objects (e.g. "A red cube on top of a blue cube"), misspelled text prompts (e.g. "A photo of a Tcennis rpacket"), or counting (e.g. "Four boxes on the table").

Figure 11 shows uncurated samples from our TR0N:BigGAN+CLIP (BLIP) and TR0N:StyleGAN-XL+LAION2BCLIP (BLIP2) models. We use out-of-distribution text-prompts we collected from various sources (the entire list of text prompts

A sign that says 'Deep Learning' | A photo of digit number 5 | A photo of a girl walking on the street | A red cube on top of a blue cube | A photo of a Tcennis rpacket | Four boxes on the table

Figure 10. Failure cases from our TR0N:BigGAN+CLIP (BLIP) model.



When the wind blows | A photo of a landscape in fall | An abstract painting of a waterfall | A subway train coming out of a tunnel | A small house in the wilderness | A beach with a cruise ship passing by | A green colored banana | A colorful robot under moonlight

A yellow book and a red vase | An American multinational technology company that focuses on artificial intelligence | A bridge connecting Europe and North America on the Atlantic Ocean, bird's eye view | A train on top of a surfboard | An ancient Egyptian painting | A painting an outer space, gothic style | A painting of a Starry Night | A shark in the desert

Figure 11. Uncurated samples from our TR0N:BigGAN+CLIP (BLIP) model (**first and third image rows**) and from our TR0N:StyleGAN-XL+LAION2BCLIP (BLIP2) model (**second and fourth image rows**).

– which we randomly subsampled to produce the figure – along with where we obtained each prompt from, is included with our code). Note that we did not curate the particular images shown in the main manuscript, although we did select text prompts for which TR0N produced good results. Although the images from the StyleGAN-XL-based model look a bit sharper (likely explaining the improved FID of this model), we find it hard to conclude than one model is consistenly better than the other. In particular, the StyleGAN-XL-based model seems to be slightly worse at conditioning (e.g. the images it produces with the same prompts as in Figure 1 and Figure 4 are worse than those from the BigGAN-based model). We hypothesize that using the style space $\mathcal{W}$ as the latent space for our StyleGAN-XL-based model might bias TR0N towards focusing more on style than matching the semantic content of the given condition.

Figure 12 shows uncurated samples from our TR0N:StyleGAN2+CLIP model. We use (randomly subsampled) out-of-distribution prompts from Pinkney & Li (2022).

Figure 13 shows samples from the TR0N:NVAE+CLIP. We do not compare this model against clip2latent since clip2latent

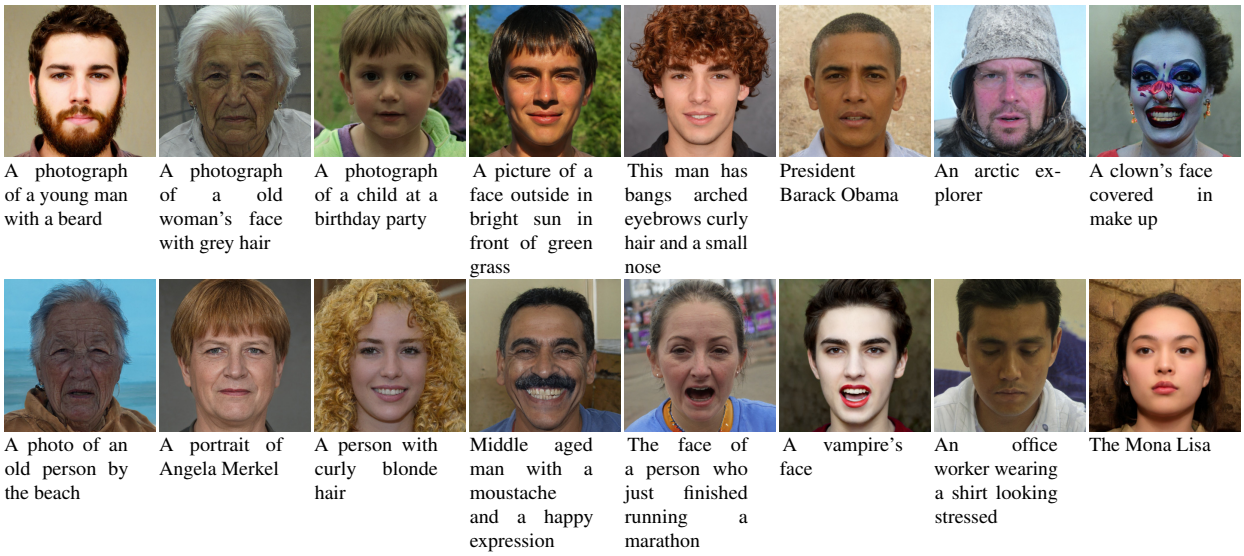


Figure 12. Uncurated Samples from our TR0N:StyleGAN2+CLIP model.

Figure 13. Samples from our TR0N:NVAE+CLIP model.

only uses StyleGAN2, and the comparison would thus not be fair. Figure 13 is thus meant to show that TR0N can also enable an NVAE model to be conditioned on text, once again highlighting the generality of TR0N.

Figure 14 shows the image semantics conditioning samples mentioned in subsection 5.3. Despite both the StyleGAN2 and the BigGAN-based models having good performance at the first task (conditioning on a single image), we find that the former is better at interpolation. Note that TR0N carries out the interpolation on $\mathcal{C}_{\text{CLIP}}$ rather than $\mathcal{Z}$, which we hypothesize might be the reason that the BigGAN-based model is not as strong at interpolation.

Figure 14. TR0N samples conditioning on image semantics with G as a StyleGAN2 (**left panels**, x' is highlighted in red); and interpolations (**right panels**, x'_1 and x'_2 are highlighted in red) with G as a StyleGAN2 (**top right panels**), and with G as a BigGAN (**bottom right panels**).