Generative Flow Networks

2023-10-20

Contents

Introduction 1
General structure 1
Training 3
How does it work in practice? 4
References 9

Introduction

Citation example Y. Bengio et al. (2023)

123456

Diffusion models seem to have taken the world by storm due to their amazing generative powers. That is not their only advantage, however. With a bit of a tongue in the proverbial cheek, statistical methods can be traditionally classified as either inflexible (classical stats), computationally expensive (MCMC), or non-analytical (boosted trees). From this perspective, **diffusion models** are a significant outlier since they are extremely flexible, provide access to the full posterior (and conditional) distributions, and are computationally less expensive than many of the competing methods.

General structure

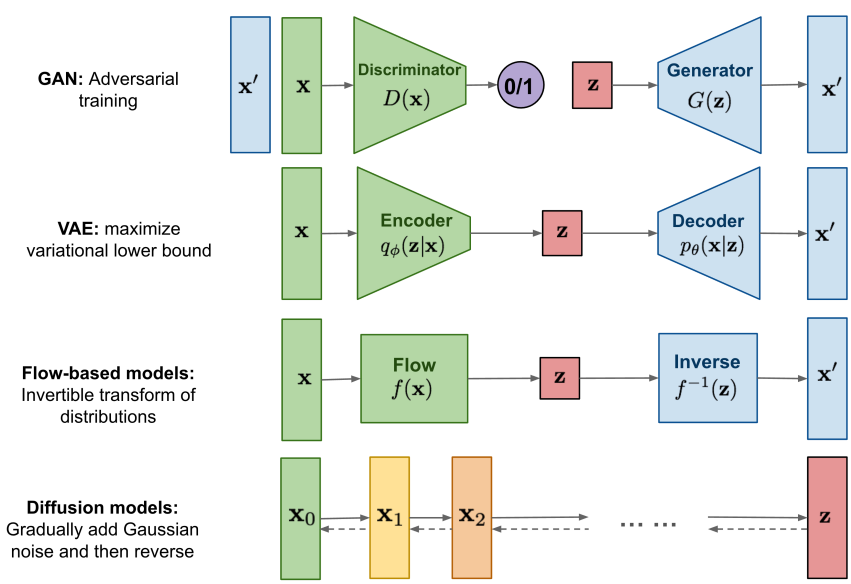

Figure 1: *An overview of generative models* (source)

Diffusion models are composed of two separate processes, **the forward** and **the backward**.

Forward diffusion

In general - a diffusion process can be characterized by a *Markov diffusion kernel* given a final state $\pi(x)$:

$$q(x_t|x_{t-1}) = T_\pi(x_t|x_{t-1}; \beta_t) \tag{1}$$

Since it's a length-one Markov process, we have for the full joint:

$$q(x_{t=0:T}) = q(x_0) \prod_{i=1}^{T} q(x_i|x_{i-1}) \tag{2}$$

Usually, we use a Gaussian diffusion, for which the posterior is closed form (c.f. the 1-step update in *Kalman filtering*):

$$q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}) \tag{3}$$

β_is are called the **variance schedule**, with usually $\beta_1 < \beta_2 < ... < \beta_T$.

Moreover, due to the property of the Gaussian distribution:

$$\alpha_t := 1 - \beta_t$$

$$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_t$$

$$q(x_t|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\ x_0, (1-\alpha_t)\mathbf{I}), \tag{4}$$

meaning any state in the forward process can be expressed knowing *just the initial one* and the *variance schedule*. In general, the theoretical underpinning of *Langevin dynamics* guarantees any smooth distribution can be corrupted into Gaussian noise, meaning the initial distribution can be almost arbitrarily complex, giving the model its expressive power.

Reverse diffusion

The reverse process is characterized by a new transition probability $p(x)$. Its starting point is the stationary distribution at the final time T:

$$\pi(x_T) := p(x_T) \tag{5}$$

Like before:

$$p(x_{t=T:0}) = p(x_T) \prod_{i=T-1}^{1} p(x_i + 1|x_i) \tag{6}$$

For Gaussian (and binomial), the joint is still in the same family; however, there is no closed-form for the parameters μ and Σ. These need to be estimated, in this case via neural networks.

A natural training target is the likelihood of the original data $p(x_0)$ as given by the learned reverse process' distribution, obtained by marginalizing the full joint:

$$p(x_0) = \int_{\mathcal{X}} p(x_{t=T:0})\ dx_1....dx_T \tag{7}$$

This is computationally intractable! A trick is to use annealed importance sampling - comparing the relative probability of the backward - p - and the forward trajectories q, where the latter are known in closed form:

$$p(x_0) = \int_{\mathcal{X}} q(x_{t=1:T})\ p(x_T)\ \prod_{t=1}^{T} \frac{p(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\ dx_1....dx_T \tag{8}$$

In the limit of very small β, both directions become the same.

Training

We maximize the **expected** log likelihood of the original data $p(x_0)$ under the original true distribution $q(x_0)$:

$$L(p,q) = \mathbb{E}_{q(x_0)}[p(x_0)] = \int_{\mathcal{X}} q(x_0)\ \log p(x_T)\ dx_0 \tag{9}$$

$$= -\ \mathbb{E}_q \left[\underbrace{D(q(x_T|x_0)||p(x_T))}_{L_T} + \underbrace{\sum_{t=2}^{T} D(q(x_{t-1}|x_t,x_0)||p(x_{t-1}||x_t))}_{L_{t-1}} - \underbrace{\log p(x_0|x+1)}_{L_0} \right] \tag{10}$$

It can be *lower bounded* by a closed-form expression:

$$L \geq K = -\sum_{t=2}^{T} \int q(x_0,x_t)\ D(q(x_{t-1}|x_t,x_0)\ ||\ p(x_{t-1}|x_t))\ dx_0,\ dx_t \tag{11}$$

$$+H_q(X_T|X_0) - H_q(X_1|X_0) - H_p(X_T), \tag{12}$$

where D is the K-L divergence and H denotes the (conditional) entropies. Hence maximizing the latter maximizes the former.

Additionally, conditioning the forward process posteriors on x_0 gives us the closed-forms

$$q(x_{t-1}|x_t,x_0) = \mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t \mathbf{I}) \tag{13}$$

$$\tilde{\mu}_t := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\ x_0 + \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_t}\ x_t$$

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

The goal of training is therefore to **estimate the reverse Markov transition densities** $p(x_{t-1}|x_t)$:

$$\hat{p}(x_{t-1}|x_t) = \underset{p}{\text{argmax}} K$$

As mentioned, in the case of the Gaussian and binomial, the reverse process stays in the same family, therefore the task amounts to estimating the parameters.

Variance schedule

Since β_t is a free parameter, it can be learned simultaneously with the whole K optimization procedure, freezing the other variables and optimizing on β. Alternatively, the first paper also treated a sequence as a hyperparameter and used a simply linearly increasing β. This is also the approach of *Ho et al.*

Estimating p

Our goal is to learn the following:

$$p(x_{t-1}|x_t) = \mathcal{N}(\mu(x_t, t), \Sigma(x_t, t)), \quad t \in [T, 0]$$

which corresponds to estimating $\mu(x_t, t), \Sigma(x_t, t)$.

For the variance, it is simply set to be isotropic with the diagonal entries either **fixed** at a constant (either β_t or $\tilde{\beta}_t$) or **learned**. The latter proved to be unstable in Ho et al., but has since been successfully implemented.

The mean is, of course, learned in all implementations and proceeds as follows: by inspecting the L_{t-1} term in the likelihood in Training, the minimizing mean can be expressed by the reparametrization:

$$x_t = \sqrt{\bar{\alpha}_t}\ x_0 + \sqrt{1 - \bar{\alpha}_t}\ \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \mathbf{I})$$

$$\mu(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \tag{14}$$

meaning we instead learn the estimator of the **noise** in the mean term at step t: $\epsilon_\theta(x_t, t))$ from the state x_t.

This can be further simplified by setting t to be *continuous* on $[1, T]$ and optimizing the following simplified objective for $\epsilon_\theta(x_t, t)$:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, x_0, \epsilon}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2\right] \tag{15}$$

This can be optimized using standard optimization techniques, such as gradient descent. We can now state the two algorithms needed to first, train the noise estimator ϵ_θ, and then, obtain a sample from the reverse diffusion process:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Architecture

Ho et al. used a variation of an U-net called PixelCNN++ to estimate $\epsilon_\theta(x_t, t))$:

How does it work in practice?

Example - recovery

Example - generation

Ho et al. used the modified network for both **conditional** and **unconditional** generation.

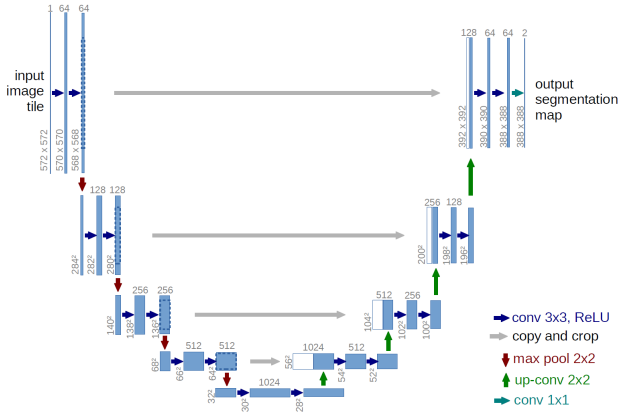

Figure 2: *The original U-net*

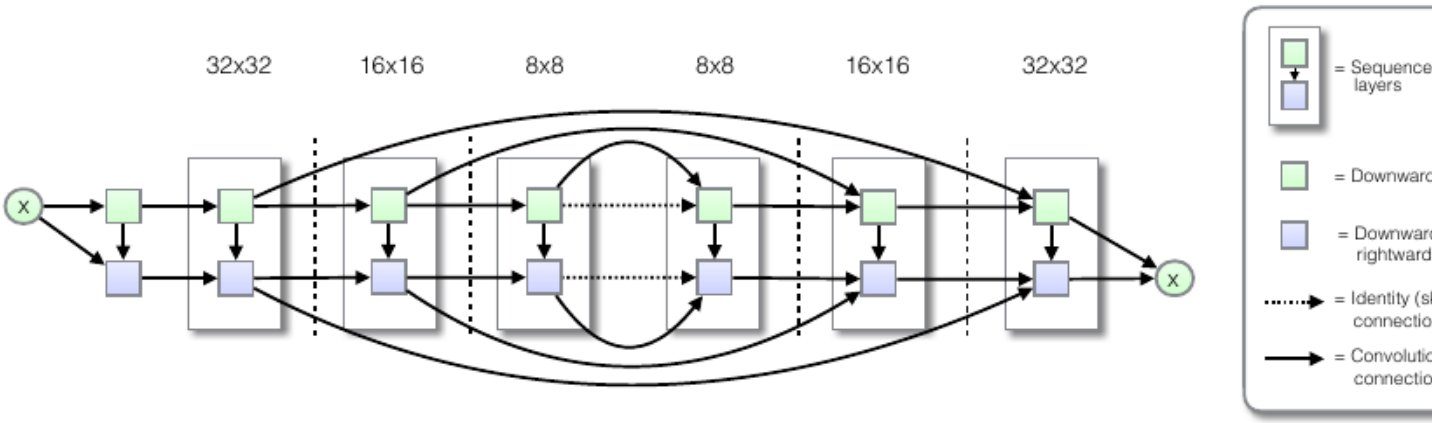

Figure 3: *PixelCNN++ structure*

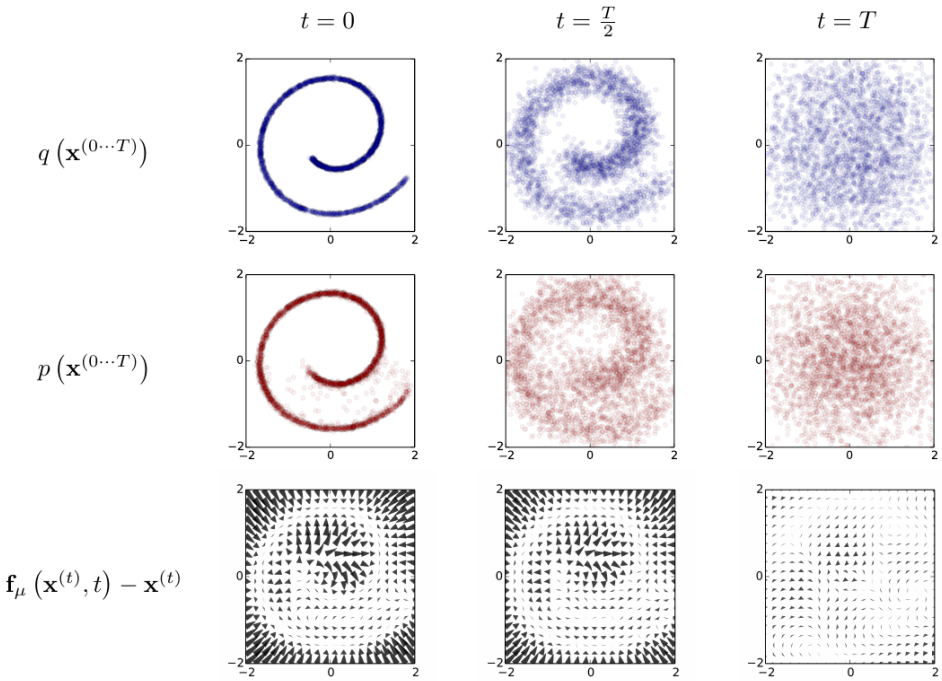

Figure 4: *Recovery (middle row) of a spiral distribution (top) using a Gaussian diffusion model. The bottom row represents the "drift term", i.e. the field controlling the mean for the "particles" in the next step of the reverse process.*

The *unconditional generation* was performed by estimating x_0 - the end result of the reverse process - from initially random x_T. The following figure shows the order in which features crystallize when the *initial state is itself sampled from various points of the reverse process.* The training dataset was CIFAR10.

For *conditional generation,* the authors selected a x_t from the actual distribution and sampled from the predictive posterior $p(x_0|x_t)$. The following figure shows the results conditioned on the bottom-right quadrant, for a network trained on CelebA-HQ:

Imagen & DALL-E 2

Ho and Salimans improved the above procedure by introducing the notion of *guiding* the model during training on labeled data, i.e. estimating $\epsilon_\theta(x_t, t \mid y)$, where y are the labels. The crux of the approach is training both on conditional and unconditional objectives at once by randomly setting the class label to a null class with some predetermined probability. Likewise, the samples are drawn from a convex combination (with the same coefficient) of both p-s.

This was used by Nichol et al. in GLIDE, which uses information extracted from text to do the guiding, combining a transformer with the previously described architecture.

This approach has been used to construct Imagen, which uses additional diffusion models to up-sample the image created by the guided diffusion process. The text embeddings are provided by a *pretrained* transformer's encoder.

The other diffusion-based model to make waves recently - DALL-E 2 - uses a bit more complex approach:

First, it re-uses a model called CLIP (Contrastive Language-Image Pre-training) to construct a *mapping between captions and images* (top of the schematic above). In practice, the net result of this is a *joint embedding of text and images* in a representation space.

Figure 5: *Unconditional generation from the reverse process sample*

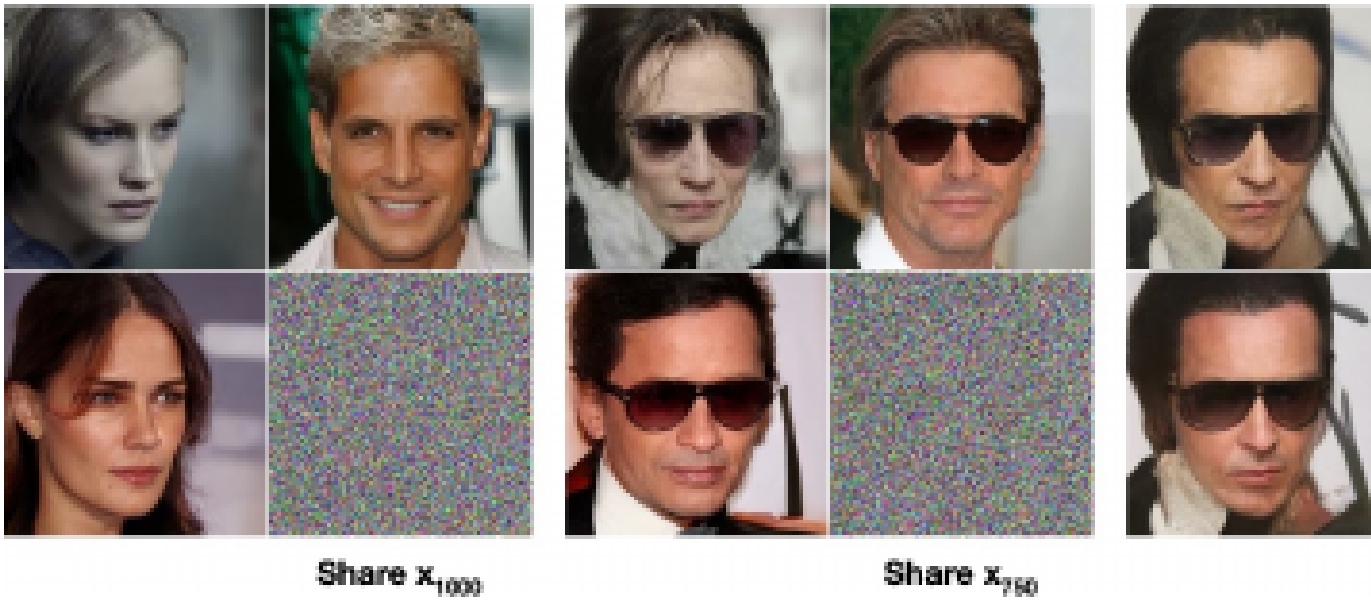

Figure 6: *Conditional generation samples from a given state:* the earlier the x_t, the more deterministic the outcome.

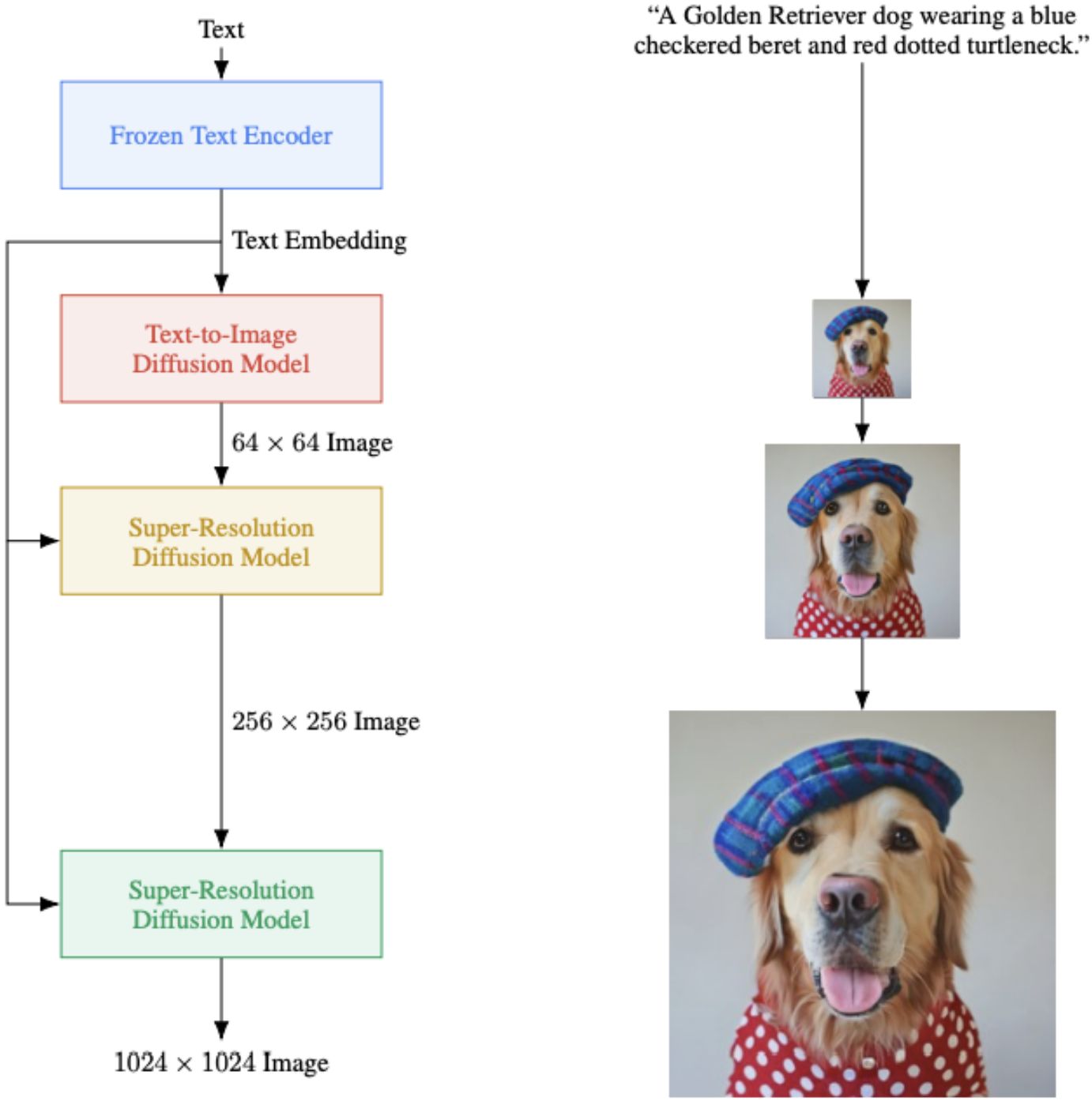

Figure 7: *Imagen structure*

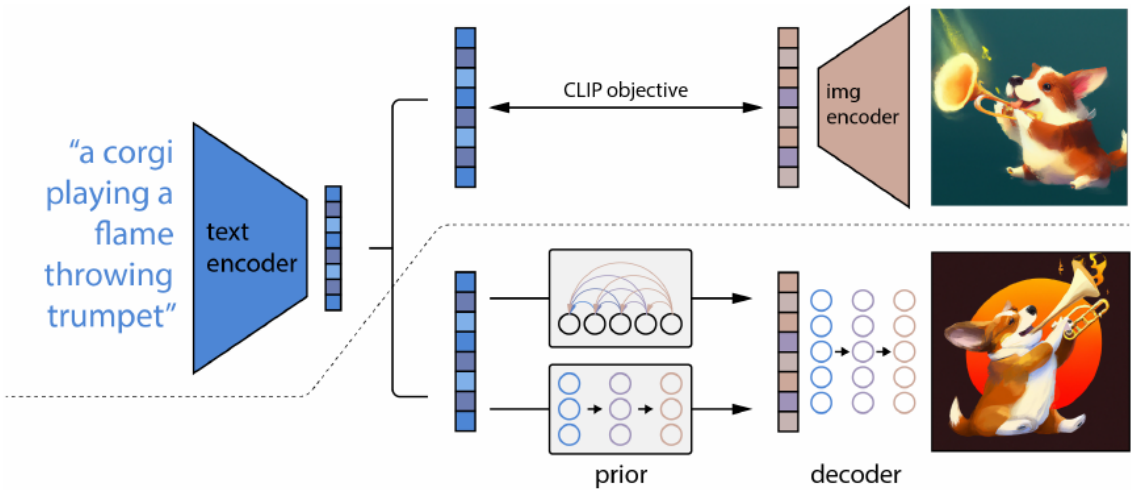

Figure 8: *DALL-E 2 architecture*

In generation, this model is frozen and a version of the GLIDE guided diffusion model is used to generate images **starting from the image representation space** (as opposed to random noise, for instance). As previously, additional up-samplers are used in the decoder, as well.

To generate from text prompts, we need to map the caption text embeddings to the above mentioned image embeddings, which are the starting point for the decoder. This is done with an *additional diffusion model* called **the prior**, which generates *multiple possible embeddings.* In other words, this is a generative model of the image embeddings given the text embeddings. The prior trains a decoder-only transformer to predict the conditional reverse process, as opposed to the U-net used in other examples.

Figure 9: *DALL-E 2: interpolation of the image embedding space by the decoder*

References

Bengio, Emmanuel, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. 2021. "Flow Network Based Generative Models for Non-Iterative Diverse Candidate Generation." In *Advances in Neural Information Processing Systems*, 34:27381–94. Curran Associates, Inc. https://papers.nips.cc/paper/2021/hash/e614f646836aaed9f89ce58e837e2310-Abstract.html.

Bengio, Yoshua, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. 2023. "GFlowNet Foundations." arXiv. https://doi.org/10.48550/arXiv.2111.09266.

Hu, Edward J., Nikolay Malkin, Moksh Jain, Katie E. Everett, Alexandros Graikos, and Yoshua Bengio.

2023. "GFlowNet-EM for Learning Compositional Latent Variable Models." In *Proceedings of the 40th International Conference on Machine Learning*, 13528–49. PMLR. https://proceedings.mlr.press/v202/hu23c.html.

Zhang, Dinghuai, Ricky T. Q. Chen, Nikolay Malkin, and Yoshua Bengio. 2023. "Unifying Generative Models with GFlowNets and Beyond." arXiv. https://doi.org/10.48550/arXiv.2209.02606.