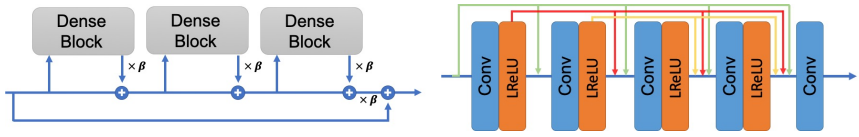
A Implementation Details

Architecture As introduced in Sect. 2.3, we adopt a model architecture that consists of a chain of modules. Each module consists of two streams, a local processing stream and a global processing stream. In terms of the backbone architecture of the two streams, the local processing stream consists of residual-in-residual dense blocks (RRDB) [87] as shown in Figure 8. Unlike traditional RRDB, which comes without normalization and deliberately removes batch normalization in particular, we apply weight normalization [73] to all convolution layers. The global processing stream consists of a sequence of dense layers, known as a mapping network [41]. The outputs from the mapping network apply adaptive instance normalization (AdaIN) [35, 25] to the output produced by each RRDB.

Details of RRDB In Figure 8, we show the inner workings of each RRDB, which contains dense blocks and residual connections.



Residual-in-Residual Dense Block (RRDB) Dense Block

Figure 8: (a) Inner workings of Residual-in-Residual Dense Blocks (RRDBs), which comprises of dense blocks (details in (b)). β is the residual scaling parameter. (b) Inner workings of dense blocks.

Global and Local Stream Effects In Figure 9a we can see that by varying the latent code input to the local stream only, local textures details like the distribution of the colour on the parrot's wing change while in Figure 9b, by varying the latent code input to the global stream only, global features like the colour tone change.

Varying Latent Code To The Local Stream

Varying Latent Code To The Global Stream

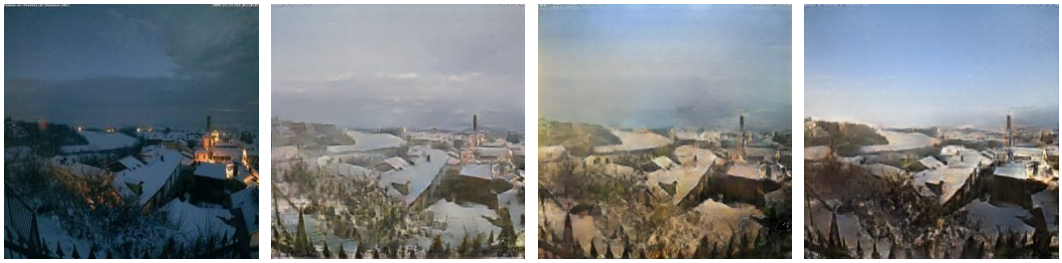
Figure 9: Varying latent code input to (a) the local stream: capturing different local spatial features like the distribution of colour spots on the image. In this example, the wing and the chest of the parrot change from red to blue. (b) the global stream: capturing diverse global features like the colour tone of the image. In this example, the tone changes from red to blue and to green.

B Experiment Details and More Results

We include more details and results on night-to-day, $16\times$ super-resolution, colourization and image decompression in the following pages. For tasks that only show a single sample from the models, we also show different samples for the same input as videos on the project website.

B.1 Night-to-day

We compare our method, CHIMLE, to the best prior IMLE-based method, cIMLE [53], and general-purpose conditional image synthesis methods, BicycleGAN [105], MSGAN [60], DivCo [56], MoNCE [95] and NDM [6] on the Transient Attributes Dataset [46]. As indicated by the quantitative metric in Table 1 and 2, DDRM failed on this task; so we omit the qualitative samples for it.



Input BicycleGAN MSGAN DivCo

MoNCE NDM cIMLE CHIMLE (Ours)

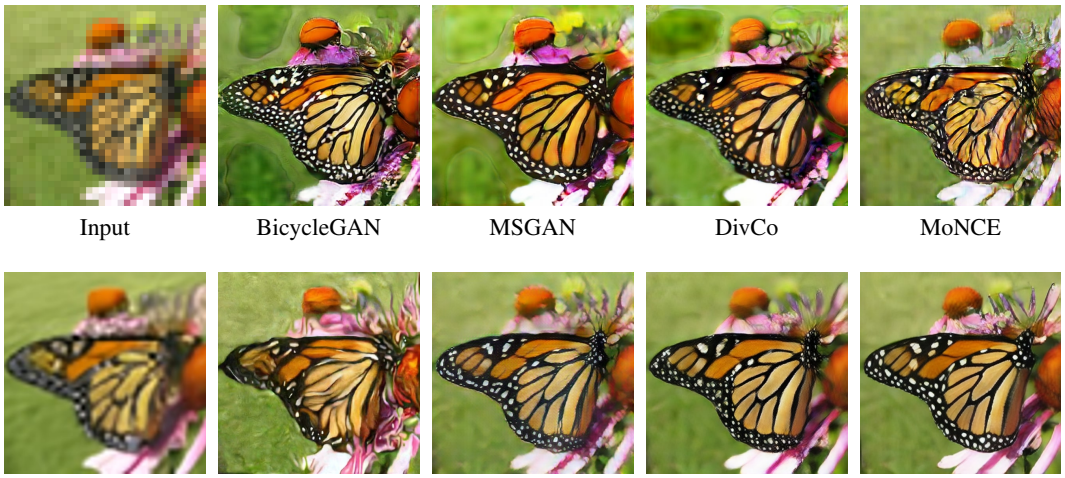
Input | BicycleGAN | MSGAN | DivCo

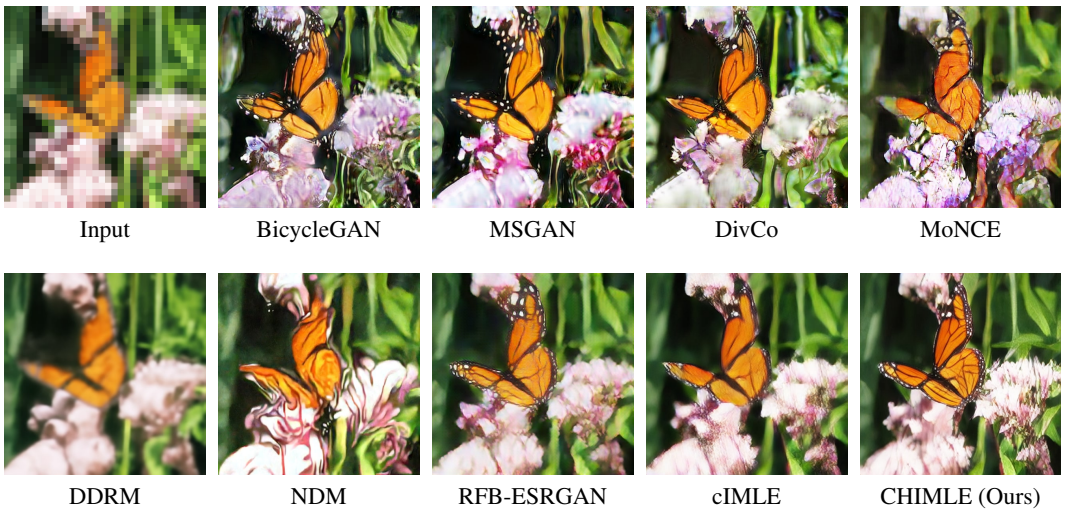
MoNCE | NDM | cIMLE | CHIMLE (Ours)

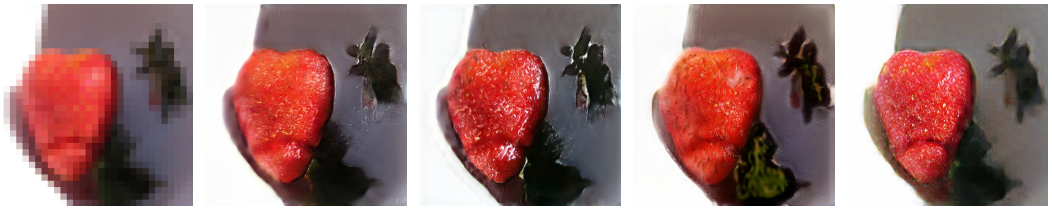
B.2 $16\times$ Super-Resolution

Given a low-resolution input image, the goal is to generate different possible higher-resolution output images that are all geometrically consistent with the input. Applications include photo enhancement, remote sensing and medical imaging. We apply our method to the more challenging [5] setting of $16\times$ upscaling, where the input image resolution is 32×32 and the output image resolution is 512×512.

We select three categories from the ILSVRC-2012 dataset and compare our method, CHIMLE, to the best prior IMLE-based method, cIMLE [53], and general-purpose conditional image synthesis methods, BicycleGAN [105], MSGAN [60], DivCo [56], MoNCE [95], DDRM [42] and NDM [6], and a task-specific method, RFB-ESRGAN [76].







Input BicycleGAN MSGAN DivCo MoNCE

DDRM NDM RFB-ESRGAN cIMLE CHIMLE (Ours)

Input BicycleGAN MSGAN DivCo MoNCE

DDRM NDM RFB-ESRGAN cIMLE CHIMLE (Ours)

B.3 Image Colourization

Given a grayscale image, the goal is to generate colours for each pixel that are consistent with the content of the input image. Applications include the restoration of old photos, for example those captured by black-and-white cameras or those captured by colour cameras that has become washed out over time. Since there could be many plausible colourings of the same grayscale image, the goal of multimodal colourization is to generate different plausible colourings, which would provide the user the ability to choose the preferred colouring among them.

We compare our method, CHIMLE, to the best prior IMLE-based method, cIMLE [53], and general-purpose conditional image synthesis methods, BicycleGAN [105], MSGAN [60], DivCo [56], MoNCE [95], DDRM [42] and NDM [6], and a task-specific method, InstColorization [79], on a subset of two categories from ILSVRC-2012 and the Natural Color Dataset (NCD) [3].

Input BicycleGAN MSGAN DivCo MoNCE

DDRM NDM InstColorization cIMLE CHIMLE (Ours)

Input BicycleGAN MSGAN DivCo MoNCE

DDRM NDM InstColorization cIMLE CHIMLE (Ours)

Input BicycleGAN MSGAN DivCo MoNCE

DDRM NDM InstColorization cIMLE CHIMLE (Ours)

Input BicycleGAN MSGAN DivCo MoNCE

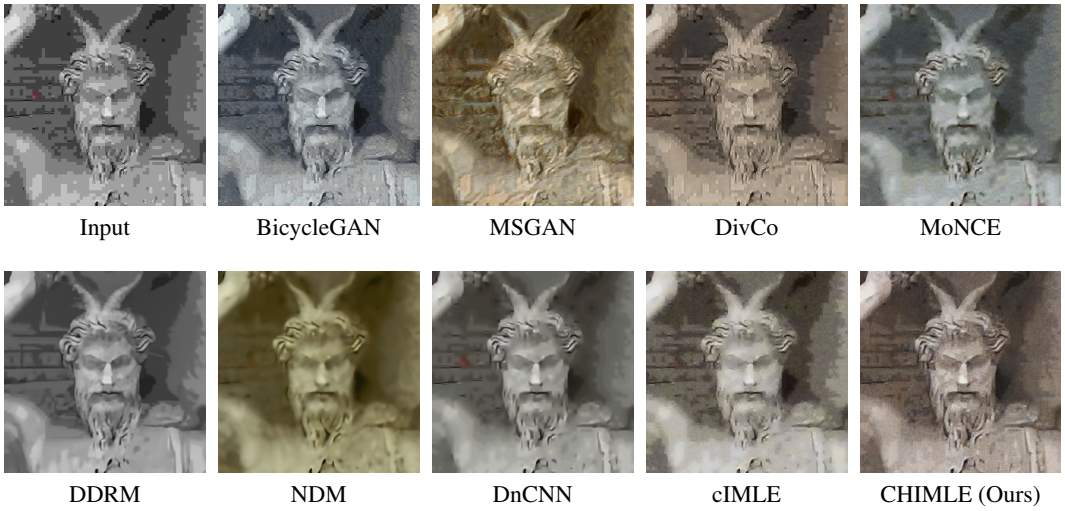
DDRM NDM InstColorization cIMLE CHIMLE (Ours)

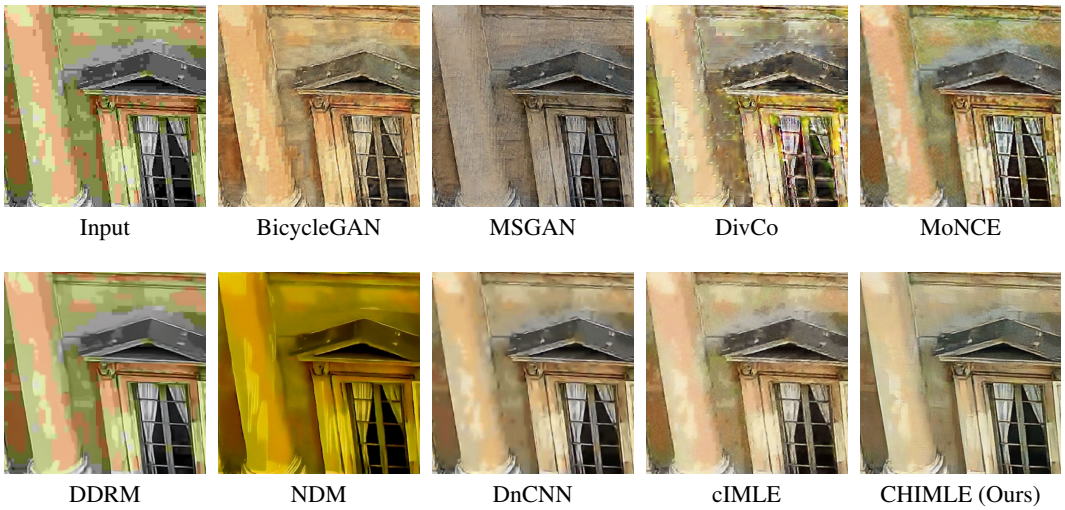
B.4 Image Decompression

Given an image compressed aggressively with a standard lossy codec (e.g.: JPEG), the goal is to recover the original uncompressed image. Note that this task is different from image compression, where both the encoding and decoding model can be learned. Here, the encoding model is fixed (e.g.: JPEG), and only the decoding model is learned.

Image decompression is of practical interest since most images are saved in lossy compressed formats; noticeable artifacts may have been introduced during compression and the original uncompressed images have been lost. It would be nice to have the capability to recover an image free of artifacts. Because compression causes irreversible information loss, multiple artifact-free images are possible. With user guidance, the most preferred version can be selected and saved for future use.

To generate training data, we compressed each image from the RAISE1K [20] dataset using JPEG with a quality of 1%. We compare our method, CHIMLE, to the best prior IMLE-based method, cIMLE [53], and general-purpose conditional image synthesis methods, BicycleGAN [105], MSGAN [60], DivCo [56], MoNCE [95], DDRM [42] and NDM [6], and a task-specific method, DnCNN [97].





C Training Stability

In Figure 10, we visualize the output of CHIMLE for a test input image over the course of training, please refer to the project webpage for the full video. As shown in the video, the output quality improves steadily during training, thereby demonstrating training stability.

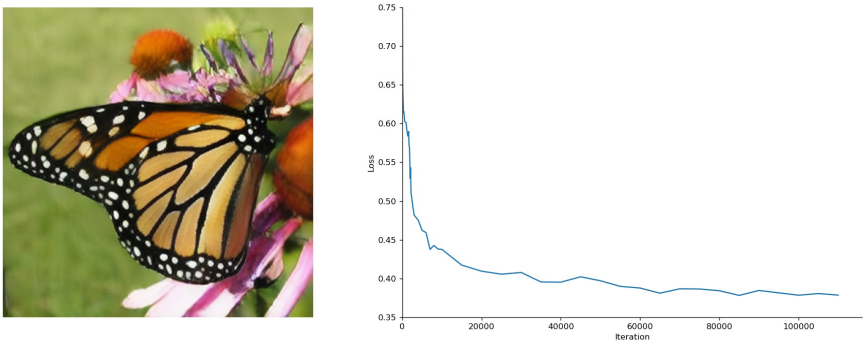


Figure 10: Output of the model at the end of training accompanied by the loss curve over the course of training. See the project webpage for the video showing the output of model while it trains. As shown in the video, the image quality steadily improves and the loss steadily decreases, which shows that training is stable.

D Mode Forcing of Diffusion Models

Consider a discrete-time diffusion model [77, 32]. We denote the data as $\mathbf{x}_0$ and the latent variables as $\mathbf{x}_{1:T}$. The forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is a fixed Markov chain, which satisfies $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}) = q(\mathbf{x}_t|\mathbf{x}_{t-1})\ \forall 1 \leq t \leq T$. The reverse process $p_\theta(\mathbf{x}_{0:T})$ is a parameterized Markov chain that we learn, which satisfies $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_{t:T}) = p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\ \forall 1 \leq t \leq T$. The likelihood the model assigns to the data is $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$, which is intractable. So, to train a diffusion model, we cannot maximize likelihood directly and instead maximize the variational lower bound $\mathcal{L}(q, \theta)$, which is a lower bound on the log-likelihood $\log p_\theta(\mathbf{x}_0)$:

$$\mathcal{L}(q,\theta) = \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \tag{3}$$

$$= \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_0)p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \tag{4}$$

$$= \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_0)\prod_{t=1}^T p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right] \tag{5}$$

$$= \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_0)\prod_{t=1}^T p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \tag{6}$$

$$= \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0) + \sum_{t=1}^T \log \frac{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \tag{7}$$

$$= \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0) - \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right] \tag{8}$$

$$= \log p_\theta(\mathbf{x}_0) - \sum_{t=1}^T \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right] \tag{9}$$

Next we will rewrite this in terms of KL divergences. From last page we have:

$$\mathcal{L}(q,\theta) = \log p_\theta(\mathbf{x}_0) - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right] \tag{10}$$

$$= \log p_\theta(\mathbf{x}_0) - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_{1:t}\sim q(\mathbf{x}_{1:t}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right] \tag{11}$$

$$= \log p_\theta(\mathbf{x}_0) - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_{1:t-1}\sim q(\mathbf{x}_{1:t-1}|\mathbf{x}_0)} \left[\mathbb{E}_{\mathbf{x}_t\sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1})} \left[\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right]\right] \tag{12}$$

$$= \log p_\theta(\mathbf{x}_0) - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_{1:t-1}\sim q(\mathbf{x}_{1:t-1}|\mathbf{x}_0)} \left[\mathbb{E}_{\mathbf{x}_t\sim q(\mathbf{x}_t|\mathbf{x}_{t-1})} \left[\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})}\right]\right] \tag{13}$$

$$= \log p_\theta(\mathbf{x}_0) - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_{1:t-1}\sim q(\mathbf{x}_{1:t-1}|\mathbf{x}_0)} \left[D_{KL}(q(\mathbf{x}_t|\mathbf{x}_{t-1})\|p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}))\right] \tag{14}$$

In variational inference, typically we learn both q and θ. Learning q can be seen as finding a variational lower bound $\mathcal{L}(q,\theta)$ that better approximates the log-likelihood $\log p_\theta(\mathbf{x}_0)$, and learning θ can be seen as fitting model parameters according to the variational lower bound. In diffusion models, however, the forward process q is fixed and not learnable.

As a result, whereas typical variational inference can minimize the KL divergences independently of the log-likelihood term by way of choosing q, diffusion models must trade off maximizing log-likelihood against minimizing the KL divergences, because both can only be changed through θ.

Now let's consider the arguments to each KL divergence, namely $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})$. The distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is generally a fixed Gaussian. On the other hand,

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}) \propto p_\theta(\mathbf{x}_t)p_\theta(\mathbf{x}_{0:t-1}|\mathbf{x}_t) \tag{15}$$

$$= p_\theta(\mathbf{x}_t) \prod_{t'=1}^{t} p_\theta(\mathbf{x}_{t'-1}|\mathbf{x}_{t'}) \tag{16}$$

The distribution $p_\theta(\mathbf{x}_t)$ is the marginal distribution of the sample at step t. The closer t is to 0, the closer $p_\theta(\mathbf{x}_t)$ gets to the likelihood $p_\theta(\mathbf{x}_0)$. Since the distribution of the data can be complex and therefore multimodal, $p_\theta(\mathbf{x}_t)$ can be highly multimodal as well, especially for small t. To arrive at the unnormalized density of $p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})$, we reweight $p_\theta(\mathbf{x}_t)$ according to how likely the reverse process starting from $\mathbf{x}_t$ will follow the path $\mathbf{x}_{1:t-1}$ sampled from the forward process and ultimately get to the data $\mathbf{x}_0$. So, as long as there exist multiple modes of $p_\theta(\mathbf{x}_t)$ that can ultimately transition to $\mathbf{x}_0$ through the reverse process with similar probability (which is expected if the reverse process mixes well), $p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1})$ will be multimodal.

Hence, minimizing the KL divergences would make $p_\theta(\mathbf{x}_t)$ similar to $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, which entails making $p_\theta(\mathbf{x}_t)$ unimodal since $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a fixed Gaussian. If the variance of $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ along each dimension is greater than that of the modes in the data, this will result in a model that connects the modes together by raising the probability density the model assigns to regions without data. On the other hand, if the variance of $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ along each dimension is smaller than that of the modes in the data, this will result in a model that suppresses some modes in the data by lowering the probability density the model assigns to regions with data. So minimizing the KL divergences can create spurious modes or drop real modes – we call this combined phenomenon *mode forcing* because it forces the creation or removal of modes. Therefore, diffusion models need to strike a balance between maximizing log-likelihood and forcing modes.