Diffusion Models

Scalable structure generation for molecular design and prediction

October 3rd, 2024

Generative modeling of biomolecules

- Put simply, generative models are machine learning models which learn to sample from an underlying distribution.
- Generating valid, three-dimensional molecular structures is an important goal for drug design and molecular modeling more widely.
  - Example: given some conditions (binding with ligand, secondary structure, amino acid sequence), can we determine a corresponding structure.
- Recent successes of deep generative models in biomolecules:
  - Structure Design: FrameFlow/FrameDiff, Chroma, RFDiffusion
  - Structure Prediction: Alpha Fold 3, DiffDock, DiffPack, FlowPack

Earlier deep generative models

- Variation Autoencoders (VAEs) and Generative Adversarial Networks (GANs)

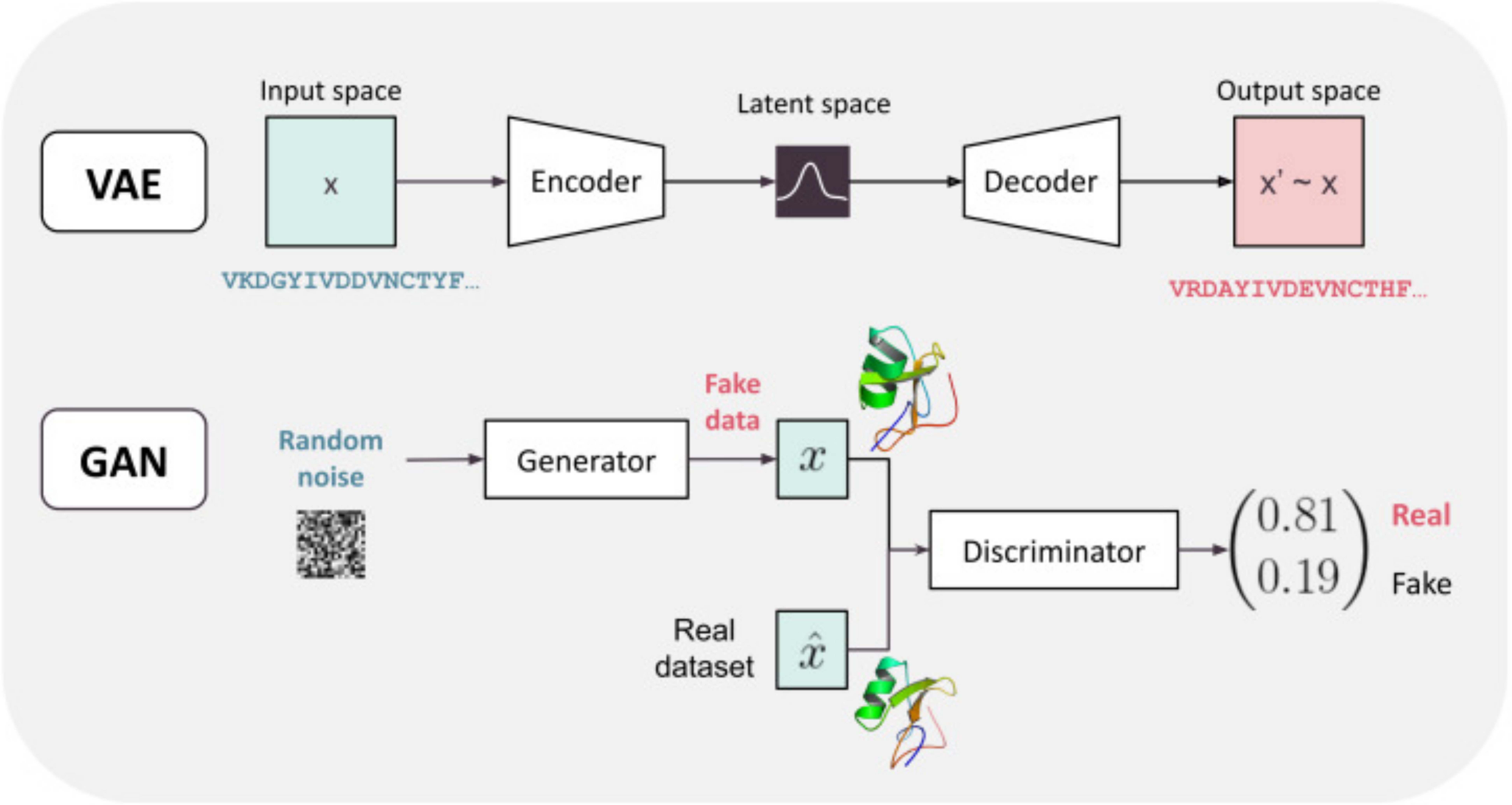
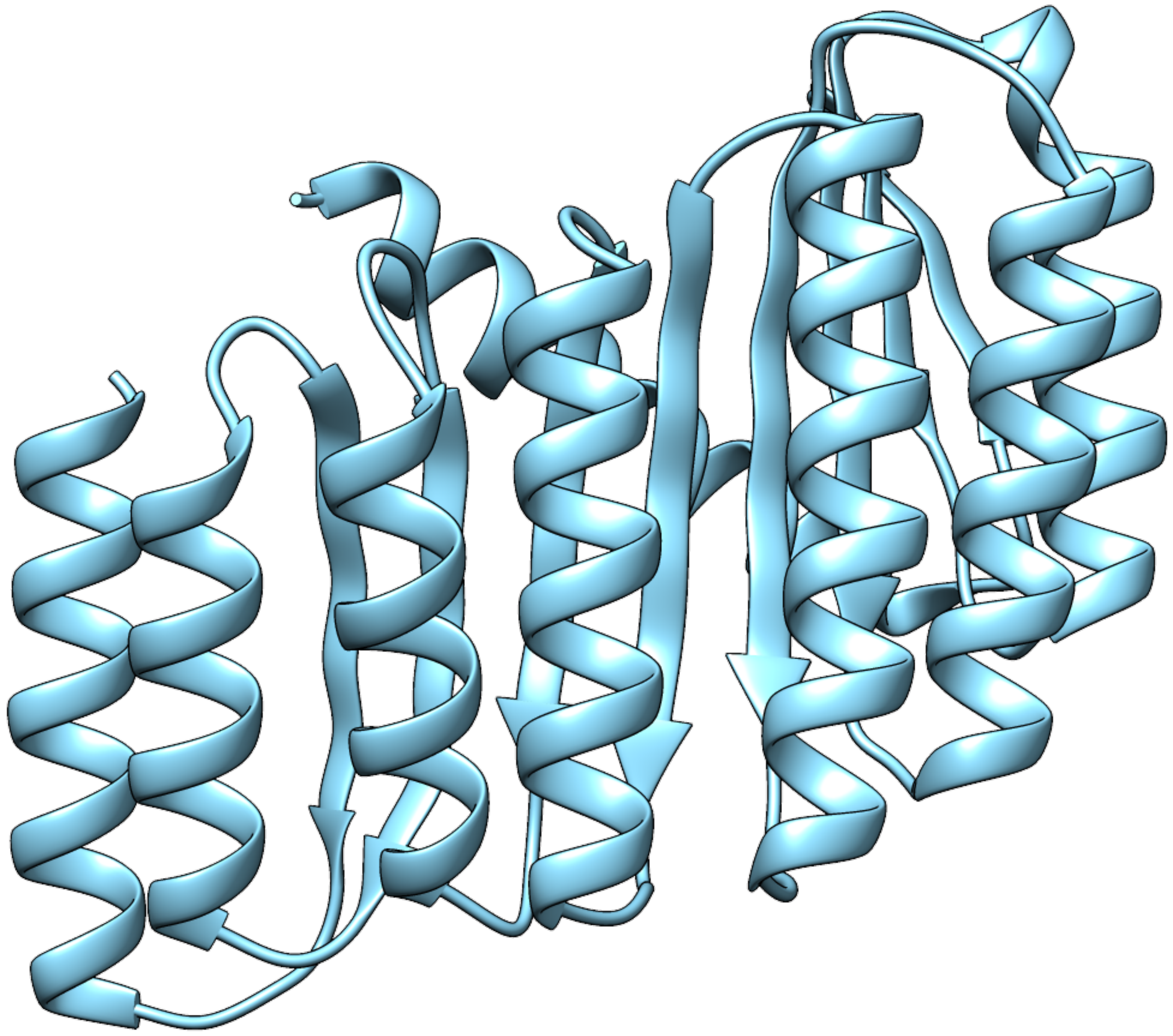


Image from: https://www.mdpi.com/1422-0067/22/21/11741

Biomolecular structures are complicated



RFDiffusion generated structure

Deep Generative Model Wishlist

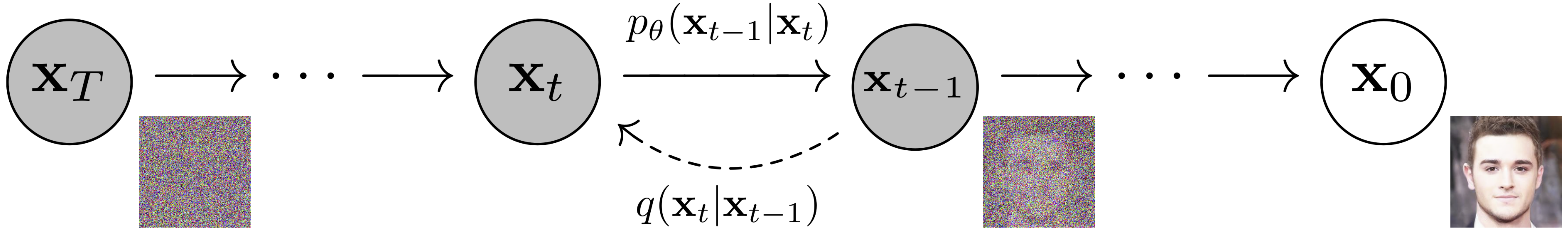
1. Incremental generation
    - Break the generation process into smaller steps that are easier to learn
2. Family of interpolating distributions
    - Intermediate steps in the generation process have defined distributions which interpolate from the data distribution to the latent distribution
3. Simulation-free training
    - Training at generation step t does not require stepping through all previous steps (typically by using marginal distributions targeting full distribution)

Diffusion models

- Several frameworks satisfy this wishlist and can achieve high-quality sampling
    - Diffusion Probabilistic Models
    - Flow Matching
    - Bayesian Flow Networks
- Diffusion Probabilistic Models (or just diffusion models) are the first and most influential
    - Main idea: inject Gaussian noise into the distribution until it reaches a normal distribution and learn to reverse the process.

Denoising Diffusion Probabilistic Models (DDPMs)

- Early (and still quite popular) framework for diffusion models
- Uses Markov chains with Gaussian transitions.
- Was one of the first diffusion models to achieve high-quality image generation.



$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

The noising and denoising process in DDPM

- Underlying distribution: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- Add Gaussian noise T times to get $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$
- Forward process is a Markov chain: $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$ where $0 < \beta_i < 1$ is the variance schedule
- Can sample at arbitrary t without stepping through MC: $\alpha_t := 1 - \beta_t$ and $\overline{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ then

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I})$$

Approaches standard normal distribution: $\mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

- Reverse process: $p_\theta(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$
- When β_t are small, the reverse process can also be written as Gaussian transitions: $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$
- Determining $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ will determine the backward process. We set $\boldsymbol{\Sigma}_\theta = \beta_t \mathbf{I}$ for simplicity.
- Training goal was originally to minimize the negative log likelihood which seeks to minimize an upper bound (plus some other terms)

$$\sum_{t>1} \mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0)} \left[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \;||\; p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \right]$$

- Let's optimize through gradient descent!
- KL divergence terms have exact formulas when distributions are normal.
  - Recall: $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \beta_t \mathbf{I})$; Just need other distribution.

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)}$$

$$\cdots$$

$$= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

where,

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_t}\beta_t}{1 - \bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t$$

- Plugging into KL divergence…

$$\mathbb{E}_q \left[\frac{1}{2\beta_t} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

- One could train a model off this using gradient descent, but there's a simpler formulation not dependent on x_t.

- Re-parameterize based on explicit formula for $q(\mathbf{x}_t \mid \mathbf{x}_0)$. Knowing x_0 allows easy sampling of x_t:

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t}\boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Then substituting the equivalent value for $\mathbf{x}_0$ gives

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1 - \overline{\alpha}_t}} \boldsymbol{\epsilon} \right)$$

- Since our model knows $\mathbf{x}_t$ at inference and needs to approximate $\tilde{\boldsymbol{\mu}}$, a good parameterization of $\boldsymbol{\mu}_\theta$ is

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

- So our model is now predicting $\boldsymbol{\epsilon}$ given $\mathbf{x}_t$ and the loss for fixed t becomes

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\beta_t\alpha_t(1-\overline{\alpha}_t)}\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]$$

- Tempting to drop the time scaling out front, so let's try it:

$$L_{\mathrm{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2$$

- This ends up working very well.

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 $\mathbf{x}_0$
```

Results

- Architecture details
  - U-Net based architecture for score network
  - T=1000 time steps
  - Time is embedded with positional encoding from Transformers and added to residual connections of U-Net
  - Variance schedule chosen to be linear with $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$

Generated images from DDPM paper

ProtDiff: an early success in protein generation

- The DDPM framework can be applied to forms of data other than images.
- Diffusion models were applied to molecular generation.
- An early success was seen in “Diffusion Probabilistic Modeling of Protein Backbones in 3D for the Motif-Scaffolding Problem”
    - Focuses on generating new, realistic protein backbones (the position of C-alpha atoms)
    - Additionally describes a method to conditionality generate proteins with prescribed positions for some backbone atoms (Motif-scaffolding)

Details for unconditional generation

- Moving into three dimensions motivates exploiting the symmetry of the problem:
  - Protein backbones that are rotated will still be considered the same
  - Typical to use equivariant networks. Authors use Equivariant Graph Neural Network (EGNN)
- Backbones are scaled so the center of mass is always at the origin and the approximate variance of points is the same the standard normal distribution
- $T=1024$
- Initial node embeddings: sinusoidal embeddings of position and time
- Initial edge embeddings: sinusoidal embedding of relative offset in sequence

$$\begin{aligned}
\mathbf{m}_{ij} &= \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right) \\
\mathbf{x}_i^{l+1} &= \mathbf{x}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right) \\
\mathbf{m}_i &= \sum_{j\neq i}\mathbf{m}_{ij} \\
\mathbf{h}_i^{l+1} &= \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)
\end{aligned}$$