COMP 7070 Advanced Topics in AI and ML April 9, 2025

Lecture 13: Neural ODE and Variational Diffusion Models

Instructor: Yifan Chen *Scribes: Ruichen Liu, Yuxuan WU* *Proof reader: Xiong Peng*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

13.1 GLOW: Improve Shuffling

Here we first present the LU decomposition: $W = PLU$, where P is a permutation matrix, L is the lower triangle matrix, and U is the upper triangle matrix. Thus we have $|W| = |U| \cdot |L|$. However, Not all W can be represented by LU. Consider a full rank matrix W with $W_{11} = 0$ where W_{11} is the element on 1st row and 1st column of W (e.g. a permutation matrix). We have,

$$\begin{aligned} W_{11} = L_{11} \cdot U_{11} = 0 &\implies L_{11} = 0 \text{ or } U_{11} = 0 \\ &\implies L \text{ or } U \text{ will not be full rank.} \\ &\implies W = LU \text{ is not full rank.} \end{aligned} \tag{13.1}$$

Here, the conclusion contradicts our original assumption. So, we need to use P.

13.2 Variational Diffusion Models

A Hierarchical Variational Autoencoder (HVAE) is a generalization of a VAE that extends to multiple hierarchies over latent variables. Under this formulation, latent variables themselves are interpreted as generated from other higher-level, more abstract latents.

Whereas in the general HVAE with T hierarchical levels, each latent is allowed to condition on all previous latents, in this work we focus on a special case called a Markovian HVAE (MHVAE).

In a MHVAE, the generative process is a Markov chain; that is, each transition down the hierarchy is Markovian, where decoding each latent z_t only conditions on previous latent $z_t + 1$. Mathematically, we represent the joint distribution and the posterior of a Markovian HVAE as:

$$p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T) p_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t). \tag{13.2}$$

and

$$q_\phi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}) \prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}) \tag{13.3}$$

. Then, we can easily write the ELBO as:

$$\begin{aligned} \log p(x) &= \log \int \frac{p(x, z_{1-T})}{q_\phi(z_{1-T}|X)} \cdot q_\phi(z_{1-T}|X) dz_{1-T} \\ &= \log E_{q_\phi}(z_{1-T}|X) \frac{p(x, z_{1-T})}{q_\phi(z_{1-T}|X)} \\ &\geq E_{q_\phi}(z_{1-T}|X) \log \frac{p(x, z_{1-T})}{q_\phi(z_{1-T}|X)} \\ &= E_{q_\phi}(z_{1-T}|x) \log \frac{\prod_{t=0}^{T-1} p(z_t|z_{t+1}) \cdot p(z_t)}{\prod_{t=0}^{T-1} q_\phi(z_{t+1}|z_t)}. \end{aligned}$$

we can further plug our joint distribution and posterior into above Equation to produce an alternate form:

$$\mathbb{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\, \boldsymbol{z}_{1:T})}{q_\phi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x})}\right] \tag{13.4}$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(z_T)\, p_\theta(\boldsymbol{x} \mid z_1) \prod_{t=2}^{T} p_\theta(z_{t-1} \mid z_t)}{q_\phi(z_1 \mid \boldsymbol{x}) \prod_{t=2}^{T} q_\phi(z_t \mid z_{t-1})}\right]. \tag{13.5}$$

13.3 Neural ODE

Start from ResNet, we have

$$z_{l+1} = z_l + f\left(z_l, w_{l+1}\right). \tag{13.6}$$

The solution of the equation is

$$\frac{\partial z\left(t\right)}{\partial t} = f\left(z\left(t\right), t, \theta\right), \tag{13.7}$$

neural ODE is to approximate f by a neural network NN. Hence, for given initial value condition $z\left(T_0\right)$ and T_0, we have

$$z\left(T_1\right) = z\left(T_0\right) + \int_{T_0}^{T_1} NN\left(z\left(T_0\right), t, \theta\right) dt. \tag{13.8}$$

For the loss function, by Eq. (13.8), it's defined as

$$\begin{aligned}\mathcal{L}\left(z\left(T_1\right)\right) &= \mathcal{L}\left(z\left(T_0\right) + \int_{T_0}^{T_1} NN\left(z\left(T_0\right), t, \theta\right) dt\right) \\ &= \mathcal{L}\left(ODESolver\left(z\left(T_0\right), NN_\theta, T_0, T_1, \theta\right)\right).\end{aligned} \tag{13.9}$$

However, the computation graph obtained in the forward process is relatively complex, and the back-propagation algorithm needs to save the activations in the forward calculation process, so large amount of activations need to be stored, which is very memory-consuming. Hence, we use Adjoint Sensitivity Method which treats the backward process as a new initial value problem (IVP) of ODE, and the gradients are computed by ODE solver. Specifically, define adjoint state for $z\left(t\right)$ as

$$a\left(t\right) = \frac{\partial \mathcal{L}}{\partial z\left(t\right)}. \tag{13.10}$$

By chain rule, we have

$$a\left(t\right) = \frac{\partial \mathcal{L}}{\partial z\left(t\right)} = \frac{\partial \mathcal{L}}{\partial z\left(t+\varepsilon\right)} \frac{\partial z\left(t+\varepsilon\right)}{\partial z\left(t\right)} = a\left(t+\varepsilon\right) \frac{\partial z\left(t+\varepsilon\right)}{\partial z\left(t\right)} \tag{13.11}$$

and its derivative is computed as followings

$$\begin{aligned}\frac{\partial a\left(t\right)}{\partial t} &= \lim_{\varepsilon \to 0^+} \frac{a\left(t+\varepsilon\right) - a\left(t\right)}{\varepsilon} \\ &= \lim_{\varepsilon \to 0^+} \frac{1}{\varepsilon}\left[a\left(t+\varepsilon\right) - a\left(t+\varepsilon\right) \frac{\partial z\left(t+\varepsilon\right)}{\partial z\left(t\right)}\right] \\ &= \lim_{\varepsilon \to 0^+} \frac{1}{\varepsilon}\left[a\left(t+\varepsilon\right) - a\left(t+\varepsilon\right) \frac{\partial}{\partial z\left(t\right)}\left(z(t) + \int_t^{t+\varepsilon} f(z(s), s, \theta) ds\right)\right] \\ &= \lim_{\varepsilon \to 0^+} -\frac{a(t+\varepsilon)}{\varepsilon} \frac{\partial}{\partial z(t)} \int_t^{t+\varepsilon} f(z(s), s, \theta) ds \\ &= \lim_{\varepsilon \to 0^+} -\frac{a(t+\varepsilon)}{\varepsilon} \frac{\varepsilon \partial f(z(t), t, \theta)}{\partial z(t)} \\ &= -a(t) \frac{\partial f(z(t), t, \theta)}{\partial z(t)}.\end{aligned} \tag{13.12}$$

To calculate $\frac{\partial \mathcal{L}}{\partial \theta}$, define

$$\begin{aligned} a_{aug} &= \left[a, a_\theta = \frac{\partial \mathcal{L}}{\partial \theta}, a_t = \frac{\partial \mathcal{L}}{\partial t}\right] \\ f_{aug} &= [f, \theta, t], \end{aligned} \tag{13.13}$$

and we have

$$\frac{\partial f_{aug}}{\partial [f, \theta, t]} = \begin{bmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial t} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{13.14}$$

we can achieve

$$\begin{aligned} \frac{\partial a_{aug}}{\partial t} &= -[a(t), a_\theta(t), a_t(t)] \frac{\partial f_{aug}}{\partial [f, \theta, t]} \\ &= \left[-a(t)\frac{\partial f}{\partial z}, -a_\theta(t)\frac{\partial f}{\partial \theta}, -a_t(t)\frac{\partial f}{\partial t}\right], \end{aligned} \tag{13.15}$$

we have

$$a_\theta(t) = -a(t)\frac{\partial f(z(t), t, \theta)}{\partial \theta}, \tag{13.16}$$

and

$$a_\theta(T_0) = a_\theta(T_1) - \int_{T_1}^{T_0} a(t)\frac{\partial f(z(t), t, \theta)}{\partial \theta} dt = \frac{\partial \mathcal{L}}{\partial \theta}. \tag{13.17}$$

Let $a_\theta(T_1) = 0$, $\frac{\partial \mathcal{L}}{\partial \theta}$ could be solved.

13.4 Variational Diffusion Models

13.4.1 Markovian Hierarchical Variational Autoencoders

A Hierarchical Variational Autoencoder (HVAE) is a generalization of a VAE that extends to multiple hierarchies over latent variables. Under this formulation, latent variables themselves are interpreted as generated from other higher-level, more abstract latents. Whereas in the general HVAE with T hierarchical levels, each latent is allowed to condition on all previous latents, in this work we focus on a special case called a Markovian HVAE (MHVAE). In a MHVAE, the generative process is a Markov chain; that is, each transition down the hierarchy is Markovian, where decoding each latent zt only conditions on previous latent $z_t + 1$. Mathematically, we represent the joint distribution and the posterior of a Markovian HVAE as:

$$p(\mathbf{x}, \mathbf{z}_{1:T}) = p(\mathbf{z}_T)\, p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{z}_{t-1}|\mathbf{z}_t), \tag{13.18}$$

and

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}) = q_\phi(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}). \tag{13.19}$$

The ELBO could be computed as

$$\begin{aligned} \log p(\mathbf{x}) &= \log \int \frac{p(\mathbf{x}|\mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})} q_\phi(\mathbf{z}_{1:T}|\mathbf{x})\, d\mathbf{z}_{1:T} \\ &= \log \mathbb{E}_{q\phi}[\mathbf{z}_{1:T}|\mathbf{x}] \frac{p(\mathbf{x}|\mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})} \\ &\geq \mathbb{E}_{q\phi}[\mathbf{z}_{1:T}|\mathbf{x}] \log \frac{p(\mathbf{x}|\mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})} \\ &= \mathbb{E}_{q\phi}[\mathbf{z}_{1:T}|\mathbf{x}] \log \frac{\prod_{t=0}^{T-1} p(\mathbf{z}_t|\mathbf{z}_{t+1}) p(\mathbf{z}_t)}{\prod_{t=0}^{T-1} q_\phi(\mathbf{z}_{t+1}|\mathbf{z}_t)}, \end{aligned} \tag{13.20}$$

and further we plug our joint distribution and posterior into above Equation to produce an alternative form

$$\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})}\left[\log \frac{p\left(\mathbf{x}|\mathbf{z}_{1:T}\right)}{q_\phi\left(\mathbf{z}_{1:T}|\mathbf{x}\right)}\right] = \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})}\left[\log \frac{p\left(\mathbf{z}_T\right) p_{\boldsymbol{\theta}}\left(\mathbf{x}|\mathbf{z}_1\right) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}\left(\mathbf{z}_{t-1}|\mathbf{z}_t\right)}{q_\phi\left(\mathbf{z}_1|\mathbf{x}\right) \prod_{t=2}^{T} q_\phi\left(\mathbf{z}_t|\mathbf{z}_{t-1}\right)}\right]. \tag{13.21}$$

13.4.2 From MHVAE to Variational Diffusion Models

The easiest way to think of a Variational Diffusion Model (VDM) is simply as a Markovian Hierarchical Variational Autoencoder with three key restrictions:

- The latent dimension is exactly equal to the data dimension.
- The structure of the latent encoder at each timestep is not learned; it is pre-defined as a linear Gaussian model. In other words, it is a Gaussian distribution centered around the output of the previous timestep.
- The Gaussian parameters of the latent encoders vary over time in such a way that the distribution of the latent at final timestep T is a standard Gaussian.

From the first restriction, with some abuse of notation, we can now represent both true data samples and latent variables as $\mathbf{x}_t$, where $t = 0$ represents true data samples and $t \in [1, T]$ represents a corresponding latent with hierarchy indexed by t. The VDM posterior can now be rewritten as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{13.22}$$

From the second assumption, we know that the distribution of each latent variable in the encoder is a Gaussian centered around its previous hierarchical latent. Unlike a Markovian HVAE, the structure of the encoder at each timestep t is not learned; it is fixed as a linear Gaussian model, where the mean and standard deviation can be set beforehand as hyper-parameters or learned as parameters. We parameterize the Gaussian encoder with mean as $\boldsymbol{\mu}_t\left(\mathbf{x}_t\right) = \sqrt{\alpha_t}\mathbf{x}_{t-1}$ and variance $\boldsymbol{\Sigma}_t\left(\mathbf{x}_t\right) = (1-\alpha_t)\mathbf{I}$. For variance-preserving,

$$\begin{aligned} Var_q\left(\mathbf{x}_t\right) &= \mathbb{E}_{\mathbf{x}_{t-1}} Var_q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right) + Var\mathbb{E}\left[\mathbf{x}_t|\mathbf{x}_{t-1}\right] \\ &= \mathbb{E}_{\mathbf{x}_{t-1}}\left[(1-\alpha_t)\mathbf{I} + Var_q\left(\sqrt{\alpha_t}\mathbf{x}_{t-1}\right)\right] \\ &= \mathbb{E}_{\mathbf{x}_{t-1}}\left[(1-\alpha_t)\mathbf{I}\right] + \alpha_t Var_q\left(\mathbf{x}_{t-1}\right), \end{aligned} \tag{13.23}$$

hence we have $Var_q\left(\mathbf{z}_t\right) = \mathbf{I}, \forall t$. Like any HVAE, the VDM can be optimized by maximizing the ELBO, which can be derived as

$$\begin{aligned} ELBO &= \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})}\left[\log \frac{p\left(\mathbf{x}_T\right) p_{\boldsymbol{\theta}}\left(\mathbf{x}_0|\mathbf{x}_1\right) \prod_{t=1}^{T-1} p_{\boldsymbol{\theta}}\left(\mathbf{x}_t|\mathbf{x}_{t+1}\right)}{q_\phi\left(\mathbf{x}_T|\mathbf{x}_{T-1}\right) \prod_{t=1}^{T-1} q_\phi\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x}_0|\mathbf{x}_1\right)\right] + \mathbb{E}_{q_\phi(\mathbf{x}_T|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q_\phi(\mathbf{x}_t,\mathbf{x}_{t-1}|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x}_0|\mathbf{x}_1\right)\right] - \mathrm{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) - \sum_{t=2}^{T} \mathbb{E}_{q_\phi(\mathbf{x}_t|\mathbf{x}_0)}\left[\mathrm{KL}(p(\mathbf{x}_{t-1}|\mathbf{x}_t)||q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0))\right], \end{aligned} \tag{13.24}$$

which an interpretation for the ELBO that can be estimated with lower variance, as each term is computed as an expectation of at most one random variable at a time. As a side note, one observes that in the process of both ELBO derivations,only the Markov as- sumption is used; as a result, these formulae will hold true for any arbitrary Markovian HVAE.

13.4.3 How to compute the loss

In this derivation of the ELBO, the bulk of the optimization cost once again lies in the summation term, which dominates the reconstruction term. Whereas each KL Divergence term is difficult to minimize for arbitrary posteriors in arbitrarily complex Markovian HVAEs due to the added complexity of simultaneously learning the encoder, in a VDM we can leverage the Gaussian transition assumption to make optimization tractable. By Bayes rule, we have:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}. \tag{13.25}$$

As we alreay know that

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}) \sim \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}), \tag{13.26}$$

what remains is deriving for the forms of $q(\mathbf{x}_t|\mathbf{x}_0)$ and $q(\mathbf{x}_{t-1}|\mathbf{x}_0)$. Fortunately, these are also made tractable by utilizing the fact that the encoder transitions of a VDM are linear Gaussian models. Recall that under the reparameterization trick, samples $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_{t-1})$ can be rewritten as

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \tag{13.27}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$. Then, the form of $q(\mathbf{x}_t|\mathbf{x}_0)$ can be recursively derived through repeated applications of the reparameterization trick. Suppose that we have access to noise variables $\{\boldsymbol{\epsilon}_t^*, \boldsymbol{\epsilon}_t\}_{t=0}^{T} \overset{i.i.d.}{\sim} \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$. Then, for an arbitrary sample $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$, we have

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^* \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\mathbf{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon}_0 \\
&= \sqrt{\bar{\alpha}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}}\boldsymbol{\epsilon}_0 \sim \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}}\mathbf{x}_0, \sqrt{1-\bar{\alpha}}\mathbf{I}\right),
\end{aligned} \tag{13.28}$$

then the KL divergence between $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ could be computed. We need to find an optimal $\boldsymbol{\theta}$ to minimize the KL divergence. Therefore, optimizing a VDM boils down to learning a neural network to predict the original ground truth image from an arbitrarily noisified version of it. Furthermore, minimizing the summation term of our derived ELBO objective across all noise levels can be approximated by minimizing the expectation over all timesteps:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\text{Uniform}[2,T]}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t))\right], \tag{13.29}$$

which can then be optimized using stochastic samples over timesteps.