Notes on DDPMs

Dates	
Status	

Introduction

- Diffusion models are a class of generative models
- Particularly good for image generation, rivalling/surpassing GANs, VAEs, Normalising Flows
- State-of-the-art for unconditional generation, text-conditioned generation, inpainting and manipulation

Diffusion Probabilistic Models

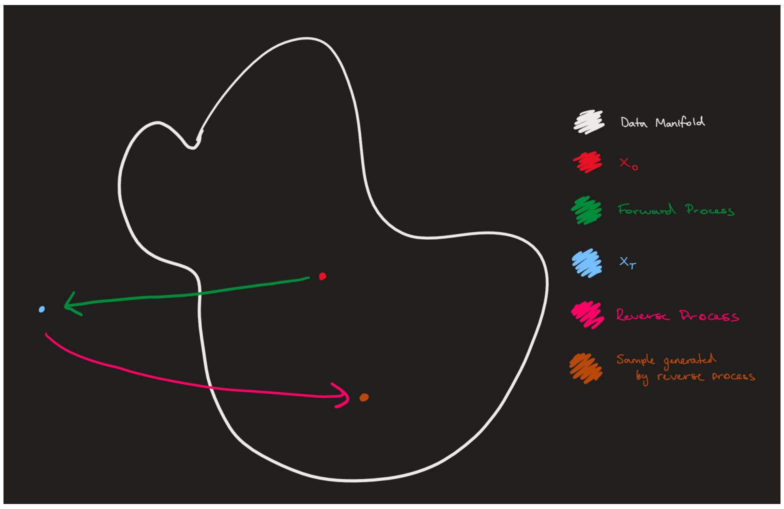
The Forward Process

- Diffusion models start with a sample from a target distribution called x_0
- The forward process gradually adds noise to x_0 over T timesteps
- This process is a Markov chain determined by $q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$
    - $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$ - a diagonal Gaussian
- β_t is the variance at time t and is usually fixed to a predetermined variance schedule
    - $\beta_t \in (0, 1)$ generally increases with time
    - Therefore, $\sqrt{1-\beta_t}$ will be non-zero but < 1
    - Therefore, the mean of each new Gaussian will be closer to 0
- In the limit, as $t \to \infty$, $q(x_T|x_0) \to \mathcal{N}(0, \mathbf{I})$, losing all information about the original sample
    - In practice, $T \approx 1000$
    - Using a large, but finite number of steps allows us to set the individual variances β_t to very small values, while still approximately maintaining the same limiting distribution
    - This small step size means that learning to undo the steps of the forward process is feasible
    - In the limit of infinitesimal step sizes, the true reverse process has the same functional form as the forward process - Feller 1949 "On the Theory of Stochastic Processes, with Particular Reference to Applications"

The Reverse Process

- Our model, the reverse process, is tasked with starting at x_T and undoing the noise
- The reverse process leverages the observation of Feller 1949, parameterising each learned reverse step as a unimodal, diagonal Gaussian - $p_\theta(x_{t-1}, x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$
    - As well as the sample at time t, the model also takes the timestep t as an input to account for the forward process variance schedule
    - i.e., the reverse process accounts for the fact that different timesteps are associated with different noise levels and the model learns to undo these individually
- The process is a Markov chain determined by $p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$
    - $p_\theta(x_{t-1}, x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$

    - $p(x_T) = q(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$



The Training Objective (aka Loss)

- Do we optimise for maximum likelihood?
    - This would be maximising the density assigned to x_0 by the model
    - No
    - This would involve calculating $p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T}$
    - This requires marginalising over all possible ways of arriving at x_0 starting from a noise sample
    - This is intractable
- We instead maximise a lower bound
    - Latent variables: $x_1, \ldots, x_T$
    - Observed variable: x_0
    - This framing allows us to view diffusion models as a latent variable generative model
        - So we can use a similar training objective to variational auto-encoders
        - In VAEs, we have an encoder (forward process) and decoder (reverse process)
        - Unlike a VAE, the forward process is fixed, so only a single network needs to be trained
        - In a model with observations x and latents z, we can derive the variational lower bound (aka evidence lower bound)
            - This is a lower bound on the marginal log likelihood
            - i.e., $\log p_\theta(x) \geq ELBO$
            - The ELBO is given by a likelihood term subtracted by a KL-divergence term - $\mathbb{E}_{q(z|x)}[\log p_\theta(x|z)] - D_{KL}(q(z|x) \| p_\theta(z))$
            - The likelihood term, $\mathbb{E}_{q(z|x)}[\log p_\theta(x|z)]$, encourages the model to maximise the expected density assigned to the data
            - The KL-Divergence, $D_{KL}(q(z|x) \| p_\theta(x))$, encourages the approximate posterior, $q(z|x)$, to be similar to the prior on the latent variable, $p_\theta(z)$
        - We can substitute in our actual observations and latents
            - $\mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(x_0|x_{1:T})] - D_{KL}(q(x_{1:T}|x_0) \| p_\theta(x_{1:T}))$

    - We denote $\mathbb{E}_{q(x_{1:T}|x_0)}$ as $\mathbb{E}_q$
    - $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T})\right] - D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}))$
    - We expand the KL-divergence, giving an expectation, $\mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T})}\right]$
    - $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T})\right] - \mathbb{E}_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T})}\right]$
    - We can combine the expectations
    - $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T}) - \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T})}\right]$
    - We use log rules
    - $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T}) + \log \frac{p_\theta(x_{1:T})}{q(x_{1:T}|x_0)}\right]$
    - More log rules - addition of logs can be replaced by a multiplication - $p_\theta(x_0|x_{1:T}) \times p_\theta(x_{1:T}) = p_\theta(x_{0:T})$
    - $\mathbb{E}_q\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$
    - We can refactor the chain probabilities into their individual steps and use log rules to make it a sum
        - $q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$
        - $p_\theta(x_{0:T}) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$
        - $\mathbb{E}_q\left[\log p(x_T) + \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$
- As the sum of independent Gaussian steps is also a Gaussian, we can sample any arbitrary step of the forward process in closed form
    - $q(x_t|x_0) = (x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$
    - $\alpha_t := 1 - \beta_t$
    - $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
    - This means, at training time, we can obtain any term of the objective without having to simulate the entire chain
    - We can optimise the objective by randomly sampling pairs of x_{t-1} and x_t, and maximising the conditional density assigned by the reverse step $x_t \to x_{t-1}$
- As different trajectories may visit different samples at time $t-1$ on the way to hitting x_t, this setup can have high variance, which limits training efficiency
- To help with this, we can rearrange the objective as follows
    - We start with $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T}) - D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}))\right]$, as before
    - $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T}) - D_{KL}(q(x_T|x_0)||p_\theta(x_T)) - \sum_{t\geq 1} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right]$
    - We note that $D_{KL}(q(x_T|x_0)||p_\theta(x_T))$ is fixed, so we denote it as constant C
    - $\mathbb{E}_q\left[\log p_\theta(x_0|x_{1:T}) - \sum_{t\geq 1} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right] + C$
    - We can move the $\log$ term into the sum too
    - $\mathbb{E}_q\left[\sum_{t\geq 1} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) - \log p_\theta(x_0|x_1)\right] + C$
    - We have a sum of KL-divergences, each between a forward step posterior conditioned on x_0, and a reverse step
    - When we treat the original sample x_0 as known, as it is during training, we can show using Bayes' Rules that the $q(x_{t-1}|x_t, x_0)$ terms are just Gaussians

    - $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}\mathbf{I})$
    - $\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$
    - $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}$
  - The reverse step is also parameterised as a Gaussian, so each KL-divergence is comparing two gaussians, so can be evaluated in closed form
  - This helps reduce variance in the training process as, instead of aiming to reconstruct Monte Carlo samples, the targets for the reverse step become the true posteriors of the forward process, given x_0
  - The are multiple ways we could implement the reverse step
  - We examine the method using by Ho et al. 2020

Denoising Diffusion Probabilistic Models

- Recall the loss, $L := \mathbb{E}_q\left[D_{KL}(q(x_T|x_0)||p_\theta(x_T)) + \sum_{t\geq 1} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) - \log p_\theta(x_0|x_1)\right] + C$
  - We define $L_0 := -\log p_\theta(x_0|x_1)$
  - We define $L_{t-1} := D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$
  - We define $L_T := D_{KL}(q(x_T|x_0)||p_\theta(x_T))$
  - Therefore, $L := \mathbb{E}_q\left[L_T + \sum_{t\geq 1}(L_{t-1} - L_0)\right]$
  - As stated previously, L_T is a constant which we denote as C
  - Therefore, $L := \mathbb{E}_q\left[\sum_{t\geq 1}(L_{t-1} - L_0)\right] + C$
- We examine one approach to implementing the reverse step from Ho et al.
- The reverse process is $p_\theta(x_{t-1}, x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$
- In DDPMs, we set the reverse process variances to time specific constants, i.e., we fix Σ_θ a priori, as Ho et al. found that learning them led to unstable training and lower quality samples
  - We fix $\Sigma_\theta(x_t, t) = \sigma_t^2\mathbf{I}$
  - So, $p_\theta(x_{t-1}, x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2\mathbf{I})$
- The reverse process network is therefore solely tasked with learning the means
- We can now rewrite $L_{t-1} := D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$
  - We use the fact that we are finding the KL-divergence between two Gaussians to calculate the divergences in a Rao-Blackwellised fashion
  - $p_\theta(x_{t-1}, x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2\mathbf{I})$
  - $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}\mathbf{I})$
  - $L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t))||^2\right] + C$
- Ho et al. then suggests that we reparameterise, learning the noise added rather than the means of the Gaussians, μ_θ
- We previously stated that we could sample any arbitrary step of the forward process in closed form using the following notation
  - $q(x_t|x_0) = (x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$
  - $\alpha_t := 1 - \beta_t$

  - $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- We introduce an auxiliary noise variable ϵ, and rewrite the closed form above
  - $x_t(x_0, \epsilon) = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
  - $\epsilon \sim \mathcal{N}(0, I)$ - ϵ has a constant distribution, independent of t
  - The reverse step model simply has to predict this epsilon
  - So, we can rewrite $L_{t-1} - C = \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t\left(x_t(x_0,\epsilon), \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t(x_0,\epsilon) - \sqrt{1-\bar{\alpha}_t}\epsilon)\right) - \mu_\theta(x_t(x_0,\epsilon), t)\right\|^2\right]$
  - $L_{t-1} - C = \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\alpha_t}\left(x_t(x_0,\epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(x_t(x_0,\epsilon), t)\right\|^2\right]$
- Our new equation for L_{t-1} reveals that μ_θ must predict $\frac{1}{\alpha_t}\left(x_t(x_0,\epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$, given x_t
  - We can use the following parameterisation, as x_t is available as input to the model
  - $\mu_\theta(x_t, t) = \tilde{\mu}_t\left(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t)\right)\right) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$
  - ϵ_θ is a function approximator intended to predict ϵ from x_t
- To sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ is to compute the following
  - $x_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z$
  - $z \sim \mathcal{N}(0, \mathbf{I})$
  - This resembles Langevin dynamics with ϵ_θ as a learned gradient of the data density
- We can simplify the loss using our parameterisation
  - $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$
  - $L_{t-1} - C = \mathbb{E}_{x_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right]$
- We simplify the notation
  - $w_t = \frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}$
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
  - $L_{t-1} - C = \mathbb{E}_{x_0,\epsilon}\left[w_t\left\|\epsilon - \epsilon_\theta(x_t, t)\right\|^2\right]$
- The authors found that a simpler version of the lower bound that discards the term weights led to better sample quality
  - $L_{t-1} - C = \mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(x_t, t)\right\|^2\right]$
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
  - This effectively gives less weight to earlier timesteps where there is less noise
  - This allows training to focus on more challenging timesteps where there is more noise to remove

Conditional Generation

- Diffusion models can be made to sample conditionally on some other variable of interest - $p_\theta(x_0|y)$
- There are a few approaches
  - Feed the conditioning variable y as an additional input during training - $\epsilon_\theta(x_t, t, y)$

    - In theory, the model should learn to use y as a hint to learn what it should be constructing
  - In practice, literature has shown that further guiding the process with a separate classifier can help
    - We take a trained classifier and push the reverse diffusion process in the direction of the gradient of the target label probability wrt. the current noisy image
  - Alternatively, works have shown that good results can be achieved using special training of the diffusion model instead of using an additional classifier
- There are also multiple approaches to take for inpainting as well
  - The naïve approach is to take a model trained in the standard way, and then at inference time, replace known regions of the image with a sample from the forward process after each reverse step
    - This method can lead to edge artifacts as the model is not made aware of the full surrounding context, only a hazy version
  - A better approach is to randomly remove sections of training images and have the model attempt to fill them in, conditioned on the full, clear context - Palette 2022