Classifer-Free Diffusion Guidance

Pramook Khungurn

November 12, 2022

This note is written as I read the paper "Classifer-Free Diffusion Guidance" by Ho and Salimans [HS22].

1 Introduction

- Classifier guidance.
  - First derived by Sohl-Dicksteing et al. in 2015 [SWMG15].
  - Popularized by Dhariwal and Nichol in 2021 [DN21].
  - The technique allows us to force a DDPM to generate images of a certain class.
  - It also allows a DDPM to trade diversity of generated data items for their fidelity.
    - This helps improve metrics such as FID, sFID, and precision.
    - It is similar to the way GANs can improve their metrics by utilizing the "trucation trick."
  - We train a DDPM as usual. However, we also need to train a classifier $p_{\boldsymbol{\phi}}(y|\boldsymbol{x}^{(t)})$ on degraded samples $\mathbf{x}^{(t)}$.
  - In the backward process, when sampling $\mathbf{x}^{(t-1)}$ from $\mathbf{x}^{(t)}$, we use the distribution

    $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, y) = \mathcal{N}\big(\mathbf{x}^{(t-1)}; \boldsymbol{\mu} + s\Sigma\nabla \log p_{\boldsymbol{\theta}}(y|\boldsymbol{\mu})\big)$$

    where
    - $\boldsymbol{\mu} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}, t)$ is the mean of the backward transition,
    - $\Sigma = \Sigma_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}, t)$ is the covariance matrix of the backward transition, and
    - s is a scaling factor.
  - See more details in another note of mine [Khu22].
- Research question: can guidance be performed without a classifier?
  - To use a classifier guidance, we need to train a classifier from scratch.
    - Existing ones cannot be used because we need to train them on degraded samples.
  - Their use in DDPM sampling is also cumbersome.
    - Need to evaluate the gradient of the log of the probability.
    - The most convenient way to do this is to use automatic differentiation in deep learning packages. Still, this is still a hassle.
  - Ho and Salimans also worry that classifier guidance makes DDPMs similar to GANs.
    - Classifier guidance is an "attack" on a certain classifier because it tries to fool that classifier to think that the generated samples are from a certain class.
    - So, in a sense, classifier guidance is similar to adversarial training.
    - Do we have a non-GAN-like way to trade diversity for fidelity?

2 Background

- Ho and Salimans use the formalism developed in Kingma et al.'s paper [KSPH21] with some modifications.
- $\mathbf{x}$ denoted a data item that is sampled from a data distribution $p(\mathbf{x})$.
- The time parameter t is dropped in favor of a new parameter λ. A degraded data item is denoted by $\mathbf{z}_\lambda$, which is indexed by $\lambda \in [\lambda_{\min}, \lambda_{\max}] \subseteq \mathbb{R}$ like this $\mathbf{z}_\lambda$. Here, λ can be negative.
- We define

$$\begin{aligned}
\alpha_\lambda^2 &= \frac{1}{1+e^{-\lambda}} = \text{sigmoid}(\lambda), \\
\sigma_\lambda^2 &= 1 - \alpha_\lambda^2 = \frac{e^{-\lambda}}{1+e^{-\lambda}} = \frac{1}{1+e^{\lambda}} = \text{sigmoid}(-\lambda), \\
\alpha_{\lambda_0|\lambda_1}^2 &= \frac{\alpha_{\lambda_0}^2}{\alpha_{\lambda_1}^2} = \frac{1+e^{-\lambda_1}}{1+e^{-\lambda_0}}, \\
\sigma_{\lambda_0|\lambda_1}^2 &= \sigma_{\lambda_0}^2 - \alpha_{\lambda_0|\lambda_1}^2 \sigma_{\lambda_1}^2 = (1 - e^{\lambda_0 - \lambda_1})\sigma_{\lambda_0}^2. \qquad \text{(Proposition 1)}
\end{aligned}$$

  Note that the formula for $\sigma_{\lambda_0|\lambda_1}^2$ works only when $\lambda_0 \le \lambda_1$. Otherwise, we would have $\sigma_{\lambda_0|\lambda_1}^2 < 0$.
- The signal-to-noise ratio (SNR) is given by

$$\text{SNR}(\lambda) = \frac{\alpha_\lambda^2}{\sigma_\lambda^2} = \frac{1+e^{\lambda}}{1+e^{-\lambda}} = \frac{e^{\lambda}(1+e^{-\lambda})}{1+e^{-\lambda}} = e^{\lambda}.$$

  So,

$$\lambda = \log \frac{\alpha_\lambda^2}{\sigma_\lambda^2} = \log \text{SNR}(\lambda).$$

- The forward process is denoted by q, so we have probabilities like $q(\mathbf{z}_\lambda|\mathbf{x})$ and $q(\mathbf{z}_\lambda|\mathbf{z}_{\lambda'})$.
  - The forward process starts from $\lambda_{\max}$ (highest SNR, least noise) and goes to $\lambda_{\min}$ (lower SNR, most noise). The backward process does the opposite.
  - As usual, we set

$$\begin{aligned}
q(\mathbf{z}_\lambda|\mathbf{x}) &= \mathcal{N}(\alpha_\lambda \mathbf{x}, \sigma_\lambda^2 I), \\
q(\mathbf{z}_{\lambda_0}|\mathbf{z}_{\lambda_1}) &= \mathcal{N}(\alpha_{\lambda_0|\lambda_1}\mathbf{x}, \sigma_{\lambda_0|\lambda_1}^2 I).
\end{aligned}$$

    for any λ, λ_0, and λ_1 with the condition that $\lambda_0 < \lambda_1$.
  - Given $\lambda_0 < \lambda_1$, we can show that

$$q(\mathbf{z}_{\lambda_1}|\mathbf{z}_{\lambda_0}, \mathbf{x}) = \mathcal{N}(\mathbf{z}_{\lambda_1}; \boldsymbol{\mu}_{\lambda_1|\lambda_0}(\mathbf{z}_{\lambda_0}, \mathbf{x}), \widetilde{\sigma}_{\lambda_1|\lambda_0}^2 I)$$

    where

$$\begin{aligned}
\boldsymbol{\mu}_{\lambda_1|\lambda_0}(\mathbf{z}_{\lambda_0}, \mathbf{x}) &= \frac{\alpha_{\lambda_0|\lambda_1}\sigma_{\lambda_1}^2}{\sigma_{\lambda_0}^2}\mathbf{z}_{\lambda_0} + \frac{\alpha_{\lambda_1}\sigma_{\lambda_0|\lambda_1}^2}{\sigma_{\lambda_0}^2}\mathbf{x} = e^{\lambda_0-\lambda_1}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}}\mathbf{z}_{\lambda_0} + (1-e^{\lambda_0-\lambda_1})\alpha_{\lambda_1}\mathbf{x}, \\
\widetilde{\sigma}_{\lambda_1|\lambda_0}^2 &= \frac{\sigma_{\lambda_0|\lambda_1}^2\sigma_{\lambda_1}^2}{\sigma_{\lambda_0}^2} = (1-e^{\lambda_0-\lambda_1})\sigma_{\lambda_1}^2.
\end{aligned}$$

    See Proposition 2 for the proof.

- The backward process is denoted by $p_{\boldsymbol{\theta}}$.
  - With $\lambda_{\min}$ low enough, we may set $p_{\boldsymbol{\theta}}(\mathbf{z}_{\lambda_{\min}}) = \mathcal{N}(\mathbf{z}_{\lambda_{\min}}; \mathbf{0}, I)$.
  - The transition probability from $\mathbf{z}_{\lambda_0}$ to $\mathbf{z}_{\lambda_1}$ is given by

    $$p_{\boldsymbol{\theta}}(\mathbf{z}_{\lambda_1}|\mathbf{z}_{\lambda_0}) = \mathcal{N}(\mathbf{z}_{\lambda_1}; \mu_{\lambda_1|\lambda_0}(\mathbf{z}_{\lambda_0}, \mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_{\lambda_0}, \lambda_0)), (\widetilde{\sigma}^2_{\lambda_0|\lambda_1})^{1-v}(\sigma^2_{\lambda_1|\lambda_0})^v I)$$

    where $\mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_{\lambda_0}, \lambda_0)$ is the denoising model the tries to predict $\mathbf{x}$ from $\mathbf{z}_{\lambda_0}$.
  - The variance of the backward transition is taken from Dhariwal and Nichol [DN21], which specifies that it is an interpolated value in log space between $\widetilde{\sigma}^2_{\lambda_0|\lambda_1}$ and $\sigma^2_{\lambda_1|\lambda_0}$.
    - The paper found it effective to use a constant hyperparameter v instance of a learned v that depends on $\mathbf{z}_{\lambda_0}$.
  - During sampling, we apply the transition along an increasing sequence

    $$\lambda_{\min} = \lambda_1 < \lambda_2 < \cdots < \lambda_T = \lambda_{\max}.$$

    This is the standard ancestral sampling used in [HJA20].
  - The denosing model $\mathbf{x}_{\boldsymbol{\theta}}$ is parameterized in the terms of the noise prediction model $\boldsymbol{\xi}_{\boldsymbol{\theta}}$ such that there relationship is given by

    $$\mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda) = \frac{\mathbf{z}_\lambda - \sigma_\lambda \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda)}{\alpha_\lambda}.$$

  - The noise prediction model is trained to minimize the loss

    $$\mathcal{L} = E_{\mathbf{x}\sim p(\mathbf{x}), \lambda\sim p(\lambda), \boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0}, I)}\left[\|\boldsymbol{\xi} - \boldsymbol{\xi}_{\boldsymbol{\theta}}(\alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\xi}, \lambda)\|^2\right]$$

    where $\mathbf{x}$ is drawn from the data distribution $p(\mathbf{x})$.
  - λ is drawn from the distribution $p(\lambda)$ where λ is sampled through the expression

    $$\lambda := -2 \log \tan(au + b(1-u))$$

    where u is sampled unformly from interval $[0, 1]$, $a = \arctan(e^{-\lambda_{\min}/2})$, and $b = \arctan(e^{-\lambda_{\max}/2})$.
  - If the noise prediction model is optimized well enough, it would predict the scaled version of the score of p_λ. In other words,

    $$\boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda) \approx -\sigma_\lambda \nabla \log q(\mathbf{z}_\lambda).$$

- For conditional generation, we are given conditioning information $\mathbf{c}$. We are required to sample from $p(\mathbf{x}|\mathbf{c})$. The derivation above remains the same except for the fact that we need to add the conditioning information to everything. In particular,

  $$\begin{aligned} p(\mathbf{x}) &\mapsto p(\mathbf{x}|\mathbf{c}), \\ q(\mathbf{z}_\lambda) &\mapsto q(\mathbf{z}_\lambda|\mathbf{c}), \\ \xi_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda) &\mapsto \xi_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}). \end{aligned}$$

- Classifier guidance, when written using the notation of the present paper is equivalent to predicting the noise with

  $$\widetilde{\boldsymbol{\xi}}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) = \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda) - w\sigma_\lambda \nabla \log p(\mathbf{c}|\mathbf{z}_\lambda)$$

  where $p(\mathbf{c}|\mathbf{z}_\lambda, \lambda)$ denote the classifier, and w is a scaling factor.

  - We are not using the mean $\mu_{\lambda+\Delta\lambda|\lambda}(\mathbf{z}_\lambda, \mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda))$ to compute the classifier gradient here because (1) it's what people do even if the derivation tell them other words, and (2) it makes sense when you take the limit as the step size approaches 0, making the model a continuous-time model.

- The paper notes that the best result in [DN21] is obtained by applying classifier-guidance on top of an already conditional model. In other words, the noise prediction model can be written as:

$$\widetilde{\boldsymbol{\xi}}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) = \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - w\sigma_\lambda \nabla \log p(\mathbf{c}|\mathbf{z}_\lambda). \tag{1}$$

3 Classifier-Free Guidance

- Instead of training a separate classifier model, we train an unconditional model $p_{\boldsymbol{\theta}}(\mathbf{x})$ togother with a conditional model $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{c})$.

- We use a single neural network $\xi_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c})$ to serve as our noise prediction model. When we want the network to predict unconditional noise, we simply set the conditioning information to $\mathbf{c} = \emptyset$ where $\emptyset$ is a special token.

- During training, $\mathbf{c}$ is set to $\emptyset$ with some probability p_{uncond}, which is a hyperparameter.

- Sampling is performed using the following linear combination of noise estimate:

$$\widetilde{\boldsymbol{\xi}}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) = (1+w)\boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - w\boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda).$$

- The above noise estimate is derived from the fact that

$$q(\mathbf{c}|\mathbf{z}_\lambda) \approx \frac{q(\mathbf{z}_\lambda|\mathbf{c})}{q(\mathbf{z}_\lambda)}$$

  As a result,

$$\nabla \log q(\mathbf{c}|\mathbf{z}_\lambda) = \nabla \log q(\mathbf{z}_\lambda|\mathbf{c}) - \nabla \log q(\mathbf{z}_\lambda).$$

  Hence, using the classifier guidance equation (1), we have that

$$\begin{aligned}
\widetilde{\boldsymbol{\xi}}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) &= \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - w\sigma_\lambda \nabla \log p(\mathbf{c}|\mathbf{z}_\lambda) \\
&= \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - w\sigma_\lambda(\nabla \log q(\mathbf{z}_\lambda|\mathbf{c}) - \nabla \log q(\mathbf{z}_\lambda)) \\
&= \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) + w(\boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - \boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda)) \\
&= (1+w)\boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - w\boldsymbol{\xi}_{\boldsymbol{\theta}}(\mathbf{z}_\lambda, \lambda).
\end{aligned}$$

4 Experiments

- The experiments were performed on the scaled-down (64×64 and 128×128) versions of ImageNet.

- The paper uses the same architecture as that of Dhariwal and Nichol [DN21] with some modifications.
  - The interpolating parameter v is not learned, but fixed.
    - For the 64×64 model, the paper fixes $v = 0.3$.
    - For the 128×128 model, the paper fixes $v = 0.2$.
  - $\lambda_{\text{min}} = -20$ and $\lambda_{\text{max}} = 20$.
  - The models were trained for 2.7 million steps.

- The paper experimented with $w \in \{0, 0.1, 0.2, \ldots, 4\}$ and $p_{\text{uncond}} = \{0.1, 0.2, 0.5\}$.

  - For FID, it dropped until $w = 0.1$ where the best score was atttained with $p_{\text{uncond}} = 0.1$. Then, FID increased as w increase.
  - IS increases with w. The best IS was attained at $w = 4$ and $p_{\text{uncond}} = 0.1$.
  - p_{uncond} of 0.1 and 0.2 performed about equally well, but 0.5 was the worst.

- The author draws two conclusion.
  - Classifier-free guidance can trade diversity for fidelity like classifier guidance can.
    * This is evident from the fact that IS scores (which depends much on fidelity) improved as w increases.
    * FID is more complicated as it depends on both fidelity and diversity.
  - Only a small portion of the model capacity is needed for unconditional generation.

A Derivations

- **Proposition 1.** *For $\lambda_0 < \lambda_1$, it holds that $\sigma^2_{\lambda_0|\lambda_1} = (1 - e^{\lambda_0 - \lambda_1})\sigma^2_{\lambda_0}$.*

  Proof. We have that

$$\begin{aligned}\sigma^2_{\lambda_0|\lambda_1} &= \sigma^2_{\lambda_0} - \alpha^2_{\lambda_0|\lambda_1}\sigma^2_{\lambda_1} = 1 - \alpha^2_{\lambda_0} - \frac{\alpha^2_{\lambda_0}}{\alpha^2_{\lambda_1}}(1 - \alpha^2_{\lambda_1}) = 1 - \alpha^2_{\lambda_0} - \frac{\alpha^2_{\lambda_0}}{\alpha^2_{\lambda_1}} + \alpha^2_{\lambda_0} = 1 - \frac{\alpha^2_{\lambda_0}}{\alpha^2_{\lambda_1}} \\ &= 1 - \frac{1 + e^{-\lambda_1}}{1 + e^{-\lambda_0}} = \frac{e^{-\lambda_0} - e^{-\lambda_1}}{1 + e^{-\lambda_0}} = \frac{e^{\lambda_0}(e^{-\lambda_0} - e^{-\lambda_1})}{e^{\lambda_0}(1 + e^{-\lambda_0})} = \frac{1 - e^{\lambda_0 - \lambda_1}}{1 + e^{\lambda_0}} \\ &= (1 - e^{\lambda_0 - \lambda_1})\sigma^2_{\lambda_0}\end{aligned}$$

  as required. □

- **Proposition 2.** *For $\lambda_0 < \lambda_1$, it holds that*

$$\boldsymbol{\mu}_{\lambda_0|\lambda_1}(\mathbf{z}_{\lambda_0}, \mathbf{x}) = e^{\lambda_0 - \lambda_1}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}}\mathbf{z}_{\lambda_0} + (1 - e^{\lambda_0 - \lambda_1})\alpha_{\lambda_1}\mathbf{x}.$$

  Proof. We have that

$$\boldsymbol{\mu}_{\lambda_0|\lambda_1}(\mathbf{z}_{\lambda_0}, \mathbf{x}) = \frac{\alpha_{\lambda_0|\lambda_1}\sigma^2_{\lambda_1}}{\sigma^2_{\lambda_0}}\mathbf{z}_{\lambda_0} + \frac{\alpha_{\lambda_1}\sigma^2_{\lambda_0|\lambda_1}}{\sigma^2_{\lambda_0}}\mathbf{x}.$$

  Now,

$$\frac{\alpha_{\lambda_0|\lambda_1}\sigma^2_{\lambda_1}}{\sigma^2_{\lambda_0}} = \frac{\alpha_{\lambda_0}\sigma^2_{\lambda_1}}{\alpha_{\lambda_1}\sigma^2_{\lambda_0}} = \frac{\alpha^2_{\lambda_0}\sigma^2_{\lambda_1}}{\alpha^2_{\lambda_1}\sigma^2_{\lambda_0}}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}} = \frac{\alpha^2_{\lambda_0}(1 - \alpha^2_{\lambda_1})}{\alpha^2_{\lambda_1}(1 - \alpha^2_{\lambda_0})}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}} = \frac{\alpha^{-2}_{\lambda_1} - 1}{\alpha^{-2}_{\lambda_0} - 1}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}} = \frac{e^{-\lambda_1}}{e^{-\lambda_0}}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}} = e^{\lambda_0 - \lambda_1}\frac{\alpha_{\lambda_1}}{\alpha_{\lambda_0}},$$

  and

$$\frac{\alpha_{\lambda_1}\sigma^2_{\lambda_0|\lambda_1}}{\sigma^2_{\lambda_0}} = \frac{\alpha_{\lambda_1}(1 - e^{\lambda_0 - \lambda_1})\sigma^2_{\lambda_0}}{\sigma^2_{\lambda_0}} = \alpha_{\lambda_1}(1 - e^{\lambda_0 - \lambda_1}) = (1 - e^{\lambda_0 - \lambda_1})\alpha_{\lambda_1}.$$

  We are done. □

References

[DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[HS22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[Khu22] Pramook Khungurn. Nichol and dhariwal on ddpm (2021). `https://pkhungurn.github.io/notes/notes/ml/ddpm-nichol-dhariwal-2021/ddpm-nichol-dhariwal-2021.pdf`, 2022. Accessed: 2022-11-08.

[KSPH21] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2021.

[SWMG15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015.