Nichol and Dhariwal on DDPM (2021)

Sunday, November 6, 2022 2:53 PM

- This note is about two papers published in 2021 by Alex Nichol and Prafulla Dhariwal.
  - Improved Denosing Diffusion Probabilistic Models (ICML 2021)
  - Diffusion Models Beat GANs on Image Synthesis (NeurIPS 2021)

Improved DDPM Paper

- DDPM
  - Reintroduced and popularized by Ho et al. in 2020 [LINK].
  - Shown to produce high quality images.
    - Measured by the FID score.
    - Competitive to GANs and autoregressive models.
  - However, does not do so well with the log-likelihood metrics.
    - Not competitive with autoregressive models yet.

- Log-likelihood metrics

  * We have a data distribution p_{data} that generates the data.
  * We also have a generative model that, given a data item x, can compute the probability $p_\theta(x)$ of generating x
  * The average log-likelihood metric is simply

$$E_{x \sim p_{data}}[-\log p_\theta(x)]$$

  * Note that it's actually negative log-likelihood
  * Commonly abbreviated as NLL
  * The common unit is "bits per dimension".
    - Seems like people like to take logarithm with base 2.
    - Also, the unit implies we must divide the log-likelihood with the number of dimensions of the data item.
  * For a DDPM, the log-likelihood cannot be computed directly.
  * However, we know that the log-likelihood is greater than its variational lower bound derived from Jensen's inequality.

$$E_{x^{(0)} \sim q}[-\log p_\theta(x^{(0)})]$$

$$\leq E_{x^{(0:T)} \sim q}\left[\log q(x^{(1:T)}|x^{(0)}) - \log p_\theta(x^{(0:T)})\right] \quad (1)$$

Here, q is the probability of the forward process
↳ so $q(x^{(0)}) \approx p_{data}(x^{(0)})$

p_θ is the probability of the backward process

T is the number of time steps.

See the PDF write-up on DDPM for other definitions and notations

* Now, (1) does not really tell us how to compute it.

We need another rewrite:

$$E_{x^{(0:T)} \sim q}\left[\log q(x^{(1:T)} | x^{(0)}) - \log p_\theta(x^{(0:T)})\right]$$

$$= E_{x^{(0:T)} \sim q}\Big[D_{KL}\left(q(x^{(T)} | x^{(0)}) \,\|\, p_\theta(x^{(T)})\right) \quad \text{(We know how to compute these.)}$$

$$- \log p_\theta(x^{(0)} | x^{(1)})$$

$$+ \sum_{t=2}^{T} D_{KL}\left(q(x^{(t-1)} | x^{(0)}, x^{(t)}) \,\|\, p_\theta(x^{(t-1)} | x^{(t)})\right)\Big]$$

(2)

* So, to evaluate the variational lower bound of data sample $x^{(0)}$

① Run the forward process to get $x^{(1)}, x^{(2)}, \ldots, x^{(T)}$

② Use the data items in the forward process to compute the above 3-term expression.

- The paper proposes ways to improve log-likelihood of DDPMs

① Use learned noise schedules rather than fixed ones in [Ho et al. 2020].

② Propose new algorithm to sample time step to optimize to reduce gradient noise.

③ Propose a new noise schedule that does not destroy info too quickly.

- Background

* The variational lower bound (2) is denoted by L_{vlb} and the terms are numbered.

$$L_{vlb} := L_0 + L_1 + L_2 + \cdots + L_T$$

$$L_0 := -\log p_\theta(x^{(0)} \mid x^{(1)})$$

$$L_{t-1} := D_{KL}\left(q(x^{(t-1)} \mid x^{(t)}, x^{(0)}) \,\|\, p(x^{(t-1)} \mid x^{(t)})\right)$$

$$L_T := D_{KL}\left(q(x^{(T)} \mid x^{(0)}) \mid p_\theta(x^{(T)})\right)$$

* Evaluating L_0 needs care
  - Pixel values are discrete, not continuous.
  - The paper divide the real line into 256 bins
  - It then compute the probability of $x^{(0)}$ landing into the right bin given $x^{(1)}$.
  - $p(x^{(0)} \mid x^{(1)})$ is a Gaussian, so $\int_a^b p(x^{(0)} \mid x^{(1)})\, dx^{(0)}$ can be computed with Gaussian CDF.

* For L_T
  - It does not depend on θ, but we need to compute it for NLL computation.
  - Will be close to 0 if the forward process sufficiently destroys the information in the input.

* Recall that the Ho et. al. paper uses the backward process of the following form:

$$p(x^{(t-1)} \mid x^{(t)}) = N(x^{(t-1)}; \mu_\theta(x^{(t)}, t), \Sigma_\theta(x^{(t)}, t)).$$

Here, $\mu_\theta(x^{(t)}, t)$ is a neural network.

Actually, $\Sigma_\theta(x^{(t)}, t)$ can be a neural networte too, but Ho et. al. use predetermined constants.

* Ho et al. experiment with two choices of $\Sigma_\theta(x^{(t)}, t)$.

① $\Sigma_\theta(x^{(t)}, t) = \beta_t I$ (the noise schedule of the forward process)

② $\Sigma_\theta(x^{(t)}, t) = \tilde{\beta}_t I = \left(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\right) I$

Here, $\tilde{\beta}_t$ is the variance that show up in $q(x^{(t-1)} \mid x^{(t)}, x^{(0)})$.

* Ho et al. mention that:

① β_t is optimal when $x^{(0)} \sim N(0, I)$

② $\tilde{\beta}_t$ is optimal when $x^{(0)}$ is a single sample.

They cite the 2015 paper by Sohl-Dickstein et al. as the source of the two statements above [LINK]. However, I could not find the proofs in the paper.

* They found that either choice did not yield difference in FID score. So, they just set $\Sigma_\theta(x^{(t)}, t)$ to β_t for simplicity.

* Instead of predicting the mean of $p(x^{(t-1)} | x^{(t)})$ directly, the Ho et al. paper trains a network $\xi_\theta(x^{(t)}, t)$ such that it would predict the noise ξ that is used to turn $x^{(0)}$ to $x^{(t)}$ according to

$$x^{(t)} = \sqrt{\bar{\alpha}_t}\, x^{(0)} + \sqrt{1-\bar{\alpha}_t}\, \xi$$

In this way,

$$\mu_\theta(x^{(t)}, t) = \frac{1}{\sqrt{\alpha_t}} \left(x^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \xi_\theta(x^{(t)}, t) \right).$$

* To train $\xi_\theta(\cdot, \cdot)$, the Ho et al. paper does not optimize L_{vlb} directly. Instead, it optimizes

$$L_{simple} = E_{t \sim \{0,1,\ldots T\},\, x^{(0)} \sim q,\, \xi \sim N(0,I)} \left[\| \xi - \xi_\theta(x^{(t)}, t) \|^2 \right]$$
$$\text{where } x^{(t)} = \sqrt{\bar{\alpha}_t}\, x^{(0)} + \sqrt{1-\bar{\alpha}_t}\, \xi$$

* Notice that L_{simple} does not contain any element that can be used to adjust the variance of $p(x^{(t-1)} | x^{(t)})$.

- Improving the log-likelihood ①: Learning $\Sigma_\theta(x^{(t)}, t)$

* The setup of Ho et. al. achieves good FID, but poor NLL.

* Nichol and Prafulla found that NLL can be improved if T is increased from 1000 to 4000.

* They make the following observations.

- β_t and $\tilde{\beta}_t$ are extreme values that one can set for the variance of $p(x^{(t-1)} | x^{(t)})$.

- β_t and $\tilde{\beta}_t$ are extreme values that one can set for the variance of $p(x^{(t-1)} | x^{(t)})$
  ↳ I actually don't believe this.
- In the Ho et al paper, the choices between β_t and $\tilde{\beta}_t$ do not yield differences in FID score because β_t is close to $\tilde{\beta}_t$ except for when t is near 0.
- They also observe that the first few terms of $L_0, L_1, \ldots, L_T$ contribute the most to NLL.
- Hence, while the choice of variance of $p(x^{(t-1)} | x^{(t)})$ might not matter for FID, it might matter for NLL
  ↳ So, we might be able to improve NLL by optimizing the variance of $p(x^{(t-1)} | x^{(t)})$ when t is near 0.

* Nichol and Dhariwal propose learning $\Sigma_\theta(x^{(t)}, t)$ instead of using fixed values.
- Their $\Sigma_\theta(x^{(t)}, t)$ is a diagonal matrix.
  Each dimension of the data item has its own variance.
- They parameterize $\Sigma_\theta(x^{(t)}, t)$ so that it interpolates between β_t and $\tilde{\beta}_t$ in log space

$$\Sigma_\theta(x^{(t)}, t) = \exp\left(v \log \beta_t + (1-v) \log \tilde{\beta}_t\right)$$

- They modify the noise network $\xi_\theta(x^{(t)}, t)$ so that it has another head that outputs a v for each dimension.
- They did not constrain v to be in the range $[0, 1]$ as they found they did not need to do so in practice.
- To train $\Sigma_\theta(x^{(t)}, t)$, they use the loss

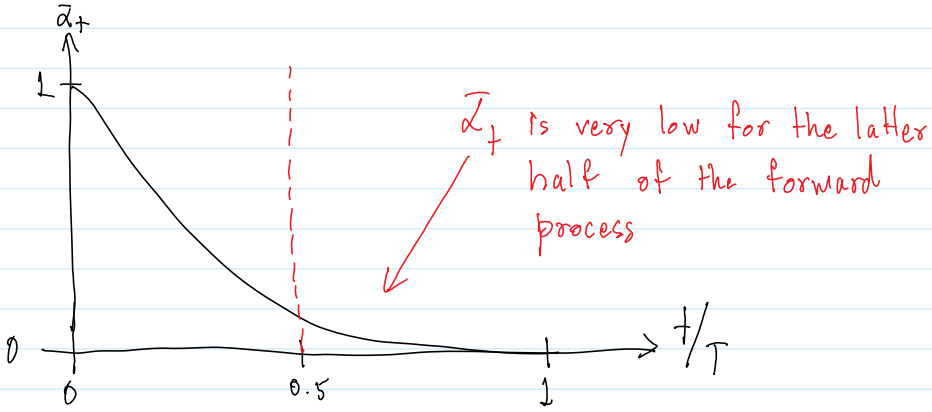
$$L_{hybrid} = L_{simple} + 0.001 L_{vlb}$$

$$L_{hybrid} = L_{simple} + 0.001 L_{vlb}$$

- When evaluating L_{vlb}, they freeze the gradient on $\xi_\theta(x^{(t)}, t)$ so that L_{vlb} does not affect the noise calculation.
- In other words, $\xi_\theta(x^{(t)}, t)$ is mainly guided by L_{simple}
  $\Sigma_\theta(x^{(t)}, t)$ is exclusively guided by L_{vlb}.

- Improving log-likelihood (2): Better noise schedule

* If you graph $\bar{\alpha}_t$ as a function of t, you will get



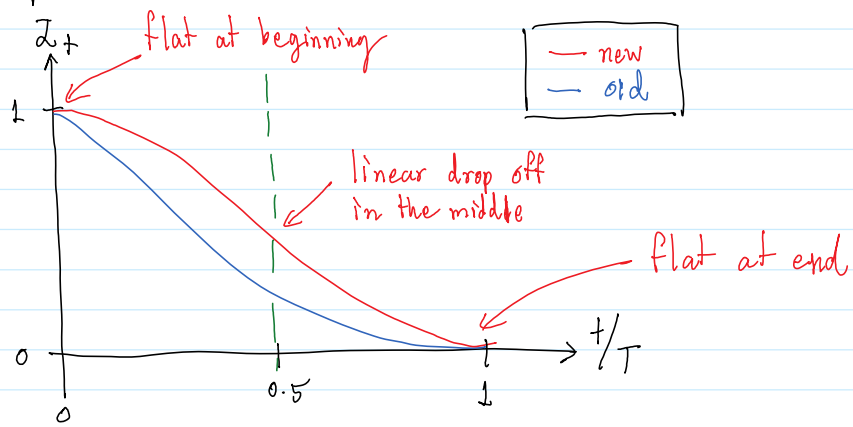
In other words, the noise schedule in the Ho et al. paper seems to destroy information too quickly. As we reach the latter half of the forward process, the distribution already looks so much like $N(0, I)$.

* Nichol and Dhariwal propose using

$$\bar{\alpha}_t = \frac{f(t)}{f(0)} \quad \text{where} \quad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

$s = 0.008$

The graph of $\bar{\alpha}_t$ now looks like this:

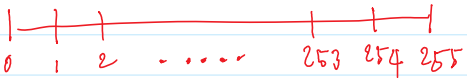


* Rationale for $\cos^2$

- Does not destroy info too fast.
- Linear drop off near middle → should be where most work is done
- Flat at beginning and end → easier to match distro there
- $\cos^2$ has the right shape, but other functions can also be used.

* Rationale for the s constant
  - Prevent β_t from being too small near $t = 0$
    ↳ Recall that $x^{(t)} = \sqrt{\bar{\alpha}_t}\, x^{(0)} + \sqrt{1-\bar{\alpha}_t}\,\xi$.
    ↳ So, if β_t is too small near $t=0$, $\sqrt{1-\bar{\alpha}_t}\,\xi$ will be small too.
    ↳ As a result, it becomes hard to predict ξ
  - s is chosen so that $\sqrt{\beta_0}$ is slightly smaller than the bin size $\frac{1}{127.5}$.

seems like pixel is normalized to $[-1, 1]$ range. Pixel value is 0 to 255, and so 255 bins



* Moreover, the paper also clip $\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}$ to be no larger than 0.999 so that there would be no singularity near $t = T$.
  ↳ Wait, isn't β_t supposed to be low so that each step would be easily reversible?

- Improving log-likelihood ③ : Reducing gradient noise
  * During training, if we graph L_{simple} and L_{vlb} against number of training iterations, the graph would look jaggy.
  * Nichol and Dhariwal hypothesized that sampling t uniformly causes the loss values to be noisy, espicially L_{vlb}.
  * They propose importance sampling based on $\mathbb{E}[L_t^2]$.
    So, $L_{vlb} := \mathbb{E}_{t\sim p_t}\left[L_t / p_t \right]$ where $p_t \propto \sqrt{\mathbb{E}[L_t^2]}$ and $\sum p_t = 1$.
  * $\mathbb{E}[L_t^2]$ is estimated by maintaining a history of the last 10 evaluations of L_t.
    ↳ Initially, t is sampled uniformly until we have 10 samples for each

T.

↳ Initially, t is sampled uniformly until we have 10 samples for each L_t.

* After the new sampling strategy is introduced, the graph of L_{vlb} becomes very smooth.

↳ It's almost too good to be true. Did they cheat?

- Ablation

* $\cos^2$ noise schedule improves both FID and NLL.

* Training with L_{simple} → good FID, bad NLL

Training with L_{vlb} → bad FID, good NLL

Training with L_{hybrid} → good FID, good NLL

- Additional benefit: Faster sampling

* The paper's DDPM is trained with $T = 4000$, so it is 4 times slower than that in Ho et al.'s paper.

* However, Nichol and Dhariwal also experimented with sampling using subsequences of $(1, 2, 3, \ldots, T)$.

* Say, the subsequence is $(s_1, s_2, \ldots, s_K)$. We need to modify the values that $\Sigma_\theta(\cdot, \cdot)$ interpolate to

$$\beta_{s_k} = 1 - \frac{\bar{\alpha}_{s_k}}{\bar{\alpha}_{s_{k-1}}}, \quad \tilde{\beta}_{s_k} = \frac{1 - \bar{\alpha}_{s_{k-1}}}{1 - \bar{\alpha}_{s_k}} \beta_{s_k}$$

* The paper experimented with using subsequences of length $K = 25, 50, 100, 200, 400, 1000$

* They found that DDPMs trained with L_{simple} suffers hugh loss in FID scores when K becomes smaller.

However, for models trained with L_{hybrid}, the FID scores did not deteriorate so dramatically.

* They also compared their results to DDIM [LINK]

- When $K < 50$, DDIM achieved better FID scores.

- When $K \geq 50$, their DDPM achieved better FID scores.

- Additional result: impact of model size on performance

* The paper changes model size by changing the number of channels in the first layer.
  ↳ 64, 96, 128, 192

* They found that FID and NLL improve as model size increases.

* FID improvement follow a power law (i.e. log FID is asymptotically linear).

* NLL improvement does not follow a power law.

DDPMs beats GANs paper

- What the paper does.

  ① Improve DDPM architecture.

  ② Experiment on classifier guidance.

  This allow DDPM to trade diversity for fidelity, like GANs can.

- Result: Their DDPMs achieved better metric values than GANs on multiple datasets.

- What has been used so far.

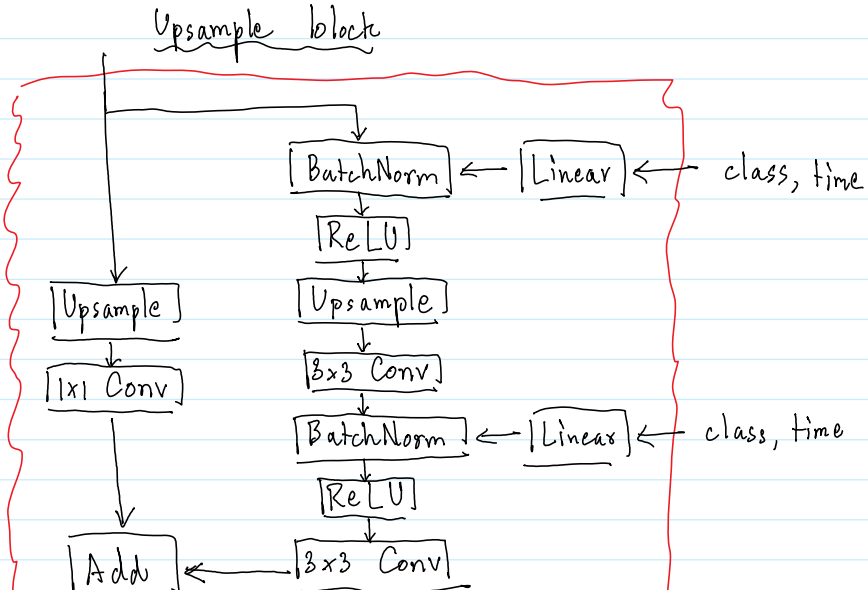
  * Ho et al. 2020
    - UNet architecture
    - Global attention layer at 16×16 resolution (single head)
    - Timestep embedding is projected and added into each residual block.

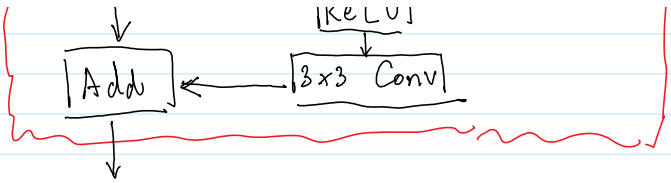
  * Nichol and Dhariwal 2020 (previous section)
    - Learned $\Sigma_\theta(x^{(t)}, t)$ and its parameterization
    - $\cos^2$ noise schedule
    - L_{hybrid} loss function
    - Sampling of t based on $\mathbb{E}[L_t^2]$ during training.

  * Song et al. 2020 (the DDIM one)
    - Deterministic sampling procedure.

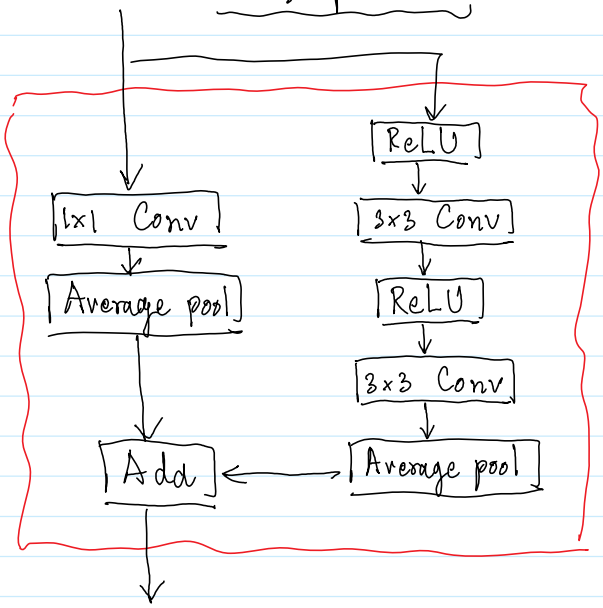
- Metrics used to evaluate generative models.

- Metrics used to evaluate generative models.
  * FID is used to assess image quality (diversity + fidelity)
  * IS is used to assess fidelity
  * Improved precision and recall (Kynkäänniemi et al. 2019 [LINK])
    - Precision is used to measure fidelity.
    - Recall is used to measure diversity.
- Architecture improvements
  * The changes
    - Increasing depth versus width, holding model size relatively constant.
      Width of what? Is this the kernel size?
    - Increasing number of attention heads
    - Using attention at 32×32, 16×16, and 8×8 instead of just 16×16
    - Using BigGAN residual block for upsampling and downsampling activations.
    - Rescaling residual connections with $1/\sqrt{2}$.
    - Adaptive Group Normalization layer (AdaGN)
  * BigGAN residual block
    - See the BigGAN paper [LINK] for more details.
    - Two types of blocks: upsample and downsample blocks





Downsample block



- The BigGAN paper doesn't seem to specify the normal residual block. However, I think it's just removing the [Upsample] operator from the upsampling block above.

* Scaling residual connection by $1/\sqrt{2}$
  - Used in StyleGAN and StyleGAN 2 papers
  - Residual block typically computes $x \longmapsto x + h(x)$.
  - StyleGANs computes $x \longmapsto \frac{x + h(x)}{\sqrt{2}}$

* Adaptive Group Normalization (AdaGN)
  - BigGAN residual block uses batch norm.
  - However, the DDPM in this paper uses AdaGN

$$\text{AdaGN}(x, y_s, y_b) = y_s \,\text{GroupNorm}(x) + y_b$$

(y_s, y_b: linear projections of class and time embedding.)

* Best configuration.
  - Base num channels = 128

- 2 residual blocks per resolution
- multi-resolution attention (32×32, 16×16, 8×8)
- 64 channels per attention head (# heads determined by # channels?)

- Classifier guidance
  * Used to force a DDPM to generate images from a certain class.
  * Train a classifier $p_\phi(y \mid x^{(t)}, t)$ on noisy image $x^{(t)}$.
  * Use gradient $\nabla \log p_\phi(y \mid x^{(t)}, t)$ during the sampling process.
    ↳ DDPM training procedure does not change.
    ↳ Classifier can be trained separately from DDPM.
  * Derivation
    - To condition on class y, it is enough to sample $x^{(t-1)}$ according to

$$p_{\theta,\phi}(x^{(t-1)} \mid x^{(t)}, y) = Z\, p_\theta(x^{(t-1)} \mid x^{(t)})\, p_\phi(y \mid x^{(t-1)})$$

      where Z is a normalizing constant.
    - In general, this is intractable, but it can be approximated.
    - First, recall the form of $p_\theta(x^{(t-1)} \mid x^{(t)})$.

$$p_\theta(x^{(t-1)} \mid x^{(t)}) = N(x^{(t-1)}; \mu, \Sigma)$$

$$\log p_\theta(x^{(t-1)} \mid x^{(t)}) = -\frac{1}{2}(x^{(t-1)} - \mu)^T \Sigma^{-1} (x^{(t-1)} - \mu) + C$$

    - We may assume that $\log p_\phi(y \mid x^{(t-1)})$ is not as peaky as $\log p_\theta(x^{(t-1)} \mid x^{(t)})$
      ↳ This assumption is reasonable because as $T \to \infty$, Σ will become smaller and smaller.
    - Given the assumption, we can approximate $\log p_\phi(y \mid x^{(t-1)})$ with a first order Taylor series expansion around μ.

$$\log p_\phi(y \mid x^{(t-1)}) \approx \log p_\phi(y \mid \mu) + (x^{(t-1)} - \mu)^T \nabla \log p_\phi(y \mid x^{(t-1)})\Big|_{x^{(t-1)}=\mu}$$
$$= (x^{(t-1)} - \mu) g + C_1$$

    - This gives

- This gives

$$\log\left(p_\theta(x^{(t-1)}|x^{(t)})\, p_\phi(y|x^{(t-1)})\right)$$

$$\approx -\frac{1}{2}(x^{(t-1)}-\mu)^T\Sigma^{-1}(x^{(t-1)}-\mu) + (x^{(t-1)}-\mu)g + C_2$$

Since $x^{(t-1)}$ is handful to write, let us abbreviate it with just x for now.

We have that

$$-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + (x-\mu)^T g + C_2$$

$$= \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + \frac{1}{2}x^T\Sigma^{-1}\Sigma g - \frac{1}{2}\mu^T\Sigma^{-1}\Sigma g - \frac{1}{2}g^T\Sigma\Sigma^{-1}\mu + \frac{1}{2}g^T\Sigma\Sigma^{-1}x - \underbrace{\frac{1}{2}g^T\Sigma^{-1}g}_{\text{constant}} + C_3$$

$$= -\qquad\qquad + C_3$$

$$= \log N(x; \mu+\Sigma g, \Sigma) + C_4$$

- So, conditional transition can be approximated by shifting the mean $\mu_\theta(x^{(t)}, t)$ by $\Sigma\nabla\log p_\phi(y|\mu_\theta(x^{(t)}, t))$
- To make the process more customizable, we will scale $\Sigma\nabla\log p_\phi(y|\mu)$ by a scaling factor s before adding to μ.
- The conditional sampling algorithm is as follows.

$x^{(T)} \leftarrow$ sample from $N(0, I)$

for $t \leftarrow T, T-1, T-2, \ldots 1$ do

$\mu \leftarrow \mu_\theta(x^{(t)}, t)$

$\Sigma \leftarrow \Sigma_\theta(x^{(t)}, t)$

$x^{(t-1)} \leftarrow$ sample from $N(\mu + s\Sigma\nabla\log p_\phi(y|\mu), \Sigma)$

end for

return $x^{(0)}$

* Sampling algorithm for DDIM

- The above conditional algorithm only works for vanilla DDPM. It doesn't work for DDIM.
- In general, for any DDPM, we train a noise prediction network $\epsilon_\theta(x^{(t)}, t)$. It has the property that

In general, ... we train a noise prediction network $\xi_\theta(x^{(t)}, t)$. It has the property that

$$-\frac{\xi_\theta(x^{(t)}, t)}{\sqrt{1-\bar{\alpha}_t}} \text{ is a good approximation of } \nabla \log p_\theta(x^{(t)}).$$

- So, $$\nabla \log\left(p_\theta(x^{(t)}) p_\phi(y|x^{(t)})\right)$$
$$= \nabla \log p_\theta(x^{(t)}) + \nabla \log p_\phi(y|x^{(t)})$$
$$= -\frac{\xi_\theta(x^{(t)}, t)}{\sqrt{1-\bar{\alpha}_t}} + \nabla \log p_\phi(y|x^{(t)})$$

- The new noise prediction $\hat{\xi}_\theta(x^{(t)}, y, t)$ is given by

$$\hat{\xi}_\theta(x^{(t)}, y) = -\sqrt{1-\bar{\alpha}_t}\, \nabla \log\left(p_\theta(x^{(t)}) p_\phi(y|x^{(t)})\right)$$
$$= -\sqrt{1-\bar{\alpha}_t}\, \nabla \log p_\theta(x^{(t)}) - \sqrt{1-\bar{\alpha}_t}\, p_\phi(y|x^{(t)})$$
$$= \xi_\theta(x^{(t)}, t) - \sqrt{1-\bar{\alpha}_t}\, p_\phi(y|x^{(t)})$$

We can then use this new noise prediction to compute $x^{(t-1)}$ from $x^{(t)}$.

* Scaling classifier guidance

- Dhariwal and Nichol found that that had to scale the gradient by a factor of more than 1 (that is $s > 1$).
  - ↳ When $s = 1$, the classifier assigns reasonable probability (around 50%) to the final sample $x^{(0)}$. However, the sample's class does not match the intended class upon visual inspection.
  - ↳ When $s > 1$, the class probability increased to nearly 100%.
- We can understand the effect of s by noting that

$$s \cdot \nabla \log p_\phi(y|x) = \nabla \log \frac{(p_\phi(y|x))^s}{Z}$$

where Z is the normalization constant.

When $s > 1$, the distribution $(p_\phi(y|x))^s$ becomes peakier by exponentiation. So, $(p_\phi(y|x))^s$ focuses more on the modes of

the distribution. This leads to higher fidelity in the generated samples.

* Summary of results
  - Guidance trades recall for precision.
  - Increasing s improves precision at the cost of recall

- Comparison to GANs and other generative models.
  * Dhariwal's and Nichol's DDPM generally yielded higher FID and sFID scores than other models, even without guidance.
  * With guidance, FID and precision improves further.
  * However, their DDPMs consistenly lose to BigGAN-deep in precision and DCTransformer in recall