DDPM analysis

Pirogov Slava

15 декабря 2022 г.

1 Анализ

В статье представляется первая работающая, с точки зрения адекватности сгенерированных изображений, диффузионная модель. Это генерационная модель, которая в первую очередь нацелена на синтезирование изображений, однако после появились и многие изменения, позволяющие работать с почти любым доменом

Соответственно мы решаем задачу синтеза изображения, которые будут похожи на наш тренировочный датасет. Как мы это делаем? Идейно у нас будут 2 процесса: процесс зашумления и разшумления, в первом мы будем итерационно добавлять шум из нормального распределения, тем самым зашумляя изображение; на обратном же будем пытаться расшумлять изображение, обученной моделью. Тем самым мы неявно выучим распределение датасета и латентов и сможем расшумлять случайный шум, получая новые изображения, которых не было в исходном датасете.

Скажем, что наша исходная картинка - x_0, а $x_1, \dots, x_T$ - картинки, получившееся из зашумления (такого же размера как и изначальная), т.е. x_t - изначальная картинка, которую мы зашумили t раз. Считаем, что $q(x_0)$ - распределение тренировочного датасета, которое мы и хотим выучить. Оценивать это распределение будем с помощью $p_\theta(x_0) := \int p_\theta(x_{0:T}) dx_{1:T}$. Прямым процессом назовем процесс зашумления: $q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta t I)$., где ковариации $\beta_1, \dots, \beta_T \in [0, 1]$ заданы по какому-то возрастающему расписанию, в этой статье линейно, но вообще это не лучший вариант.

Почему семплирование из нормального распределения вообще можно считать за постепенное зашумление? Давайте посмотрим на него внимательнее: мы в точке x_t считаем нормальное распределение со средним $\sqrt{1-\beta_t}x_{t-1}$, то есть берем предыдущее изображение и домножаем его на коэффицент, который уменьшается при росте t, т.е. это среднее с течением времени будет все сильнее отличаться от предыдущего, пока окончательно не перетечет в 0. Ковариация при этом будет расти от роста t. Тем самым в конце мы сойдемся в нормальное распределение со средним 0 и ковариацией 1, это важно! Кстати, изображения у нас изначально рескейлятся в [-1; 1]

Обратным процессом назовем как раз ту Марковскую цепочку, которую мы захотим оценивать и выучивать: $p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. Т.е. мы будем выучивать матрицу средних и ковариаций (на самом деле в этой и во многих других не выучивают матрицу ковариаций, а берут пропорциональную единичной) и семплируя из этого распределения будем расшумлять картинку. Почему мы можем расшумлять семпилруя из какого-то распределения - отдельный вопрос, на который ответим позже. В прямом и обратных процессах мы пользуемся тем, что они Марковские: мы можем представить то что зашумление на шаге t зависит только от картинки на $t-1$ шаге; с обратным процессом наоборот, расшумление на шаге $t-1$ зависит только от картинки на шаге t.

Важным моментом явлвяется тот факт, что мы можем зашумлять картинку детерминированно не по 1 шагу, а сразу прыгнуть на любое t (детерминировано с точки зрения совпадения распределений). Тут просто приведу формулы, позволяющие семплировать любой шаг t: пусть $a_t := 1 - \beta_t$ и $\overline{a}_t := \prod_{i=1}^{t} a_i$

$$
\begin{aligned}
&x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\overline{\epsilon}_{t-2} = \dots = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, \\
&\quad \text{где } \epsilon_{t-1}, \epsilon_{t-2}, \dots \sim \mathcal{N}(0, I); \overline{\epsilon}_{t-2} - \text{ смешанные Гауссианы} \\
&\quad q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I)
\end{aligned}
\tag{1}
$$

Насколько я понимаю, из-за того что у нас Марковский процесс, то мы можем сеплировать с помощью динамики Ланжевина, которая требует лишь знать градиент логарифма распределения

Также важным момент является, что при заданном x_0 точный обратный процесс может выражен с помощью теоремы Байеса через нормальное распределение:

$$
\begin{aligned}
&q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I), \\
&\text{where } \overline{\mu}_t(x_t, x_0) := \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}x_t, \text{ and } \overline{\beta}_t := \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}}\beta_t
\end{aligned}
\tag{2}
$$

Но $q(x_{t-1}|x_t)$ уже не получится раписать также, так как оно зависит от x_0, которого мы не знаем

Теперь пришло поговорить про функционал ошибки. Учим обычным методом максимального правдоподобия, т.е. мы хотим максимизировать $logp_\theta(x_\theta)$ - хотим найти такие параметры, при которых вероятность пронаблюдать нашу выборку максимальна. Так как большинство методов оптимизации нацелено на минимазацию функционала - берем минус. Отсюда и берется negative log likelihood.

Умели бы мы минимизировать правдоподобие - жизнь была сказкой, однако все не так радужно и мы применяем дефолтную вариационную нижнюю оценку, позволяющую избавиться от подсчета $p_\theta(x_0)$.

$$
\begin{aligned}
-\log p_\theta\left(\mathbf{x}_0\right) &\leq -\log p_\theta\left(\mathbf{x}_0\right)+D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)\right) \\
&=-\log p_\theta\left(\mathbf{x}_0\right)+\mathbb{E}_{\mathbf{x}_{1: T} \sim q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)}\left[\log \frac{q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{0: T}\right) / p_\theta\left(\mathbf{x}_0\right)}\right] \\
&=-\log p_\theta\left(\mathbf{x}_0\right)+\mathbb{E}_q\left[\log \frac{q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{0: T}\right)}+\log p_\theta\left(\mathbf{x}_0\right)\right] \\
&=\mathbb{E}_q\left[\log \frac{q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{0: T}\right)}\right]=\mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0: T}\right)}{q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)}\right]=: L
\end{aligned}
$$

Эту же оценку можно еще более подробно раписать и разделить на несколько KL дивергенций, чтобы мы могли ее посчитать:

$L=\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T}+\sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}]$

Давайте посмотрим, что у нас получилось: кучу KL дивергенций между нормальным распределениями, которые мы можем спокойно посчитать, а также L_T - которая будет всегда константой, т.к. распределение датасета не меняется, а на последнем шаге у нас нормальное фиксированное распределение

Как я уже говорил выше – вместо предсказания шума будем предсказывать параметры нормального распределения для каждого L_t. Получится громоздкая формула, которую авторы предлагаю упростить до обычного MSE, получив новый фунционал:

$$
\begin{aligned}
L_t^{\text{simple}} &= \mathbb{E}_{t \sim[1, T], \mathbf{x_0}, \boldsymbol{\epsilon_t}[}\left[\left\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] \\
&= \mathbb{E}_{t \sim[1, T], \mathbf{0}_0, \epsilon_t}\left[\left\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0+\sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}_t, t\right)\right\|^2\right]
\end{aligned}
$$

Утверждается, что эмпирически проверив мы не теряем в качестве, так что легче забить

Наконец получается следующий алгориитм с функционалом L^{simple}:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
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** $t=T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t>1$, else $\mathbf{z}=\mathbf{0}$
4: $\mathbf{x}_{t-1}=\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right)+\sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Для экспериментов авторы выбрали везде $T = 1000$ (судя по следующим статьям - удачный выбор). В качестве модели авторы используют распространенный U-Net, но с приколами типа self-attention. Авторы сразу же отмечают, что получили на CIFAR FID=3.17, что на тот момент было сотой (вроде). В любом случае это очень качественные картинки. В целом почти вся экспериментальная часть нацелена на поиск лучшего паттерна обучения: почему тренируем на L^{simple}, предсказываем расшумление через среднее. Также делаем вывод о том, что модель не переобучается

Наверное, самая интересная часть это интерполяция картинок. Авторы предлагаю взять 2 картинки, зашумить их полностью, интерполировать латенты с каким-то коэффицентov и расшулять уже его – должно получится что-то среднее между картинками, но при этом из нашего изначального распределения!

Логичнее всего сравнивать диффузионные модели в первую очередь с ГАНами. Какие мы получили преимущества? В этой статье авторы уже показали, что мы умеем генерировать картинки вполне сопоставимые по качеству с ГАНами, при этом у нас скорее нет их главных проблем: почти некотролируемое и нестабильное обучение –

что происходит внутри чаще всего непонятно, лоссы не особо интерпретируемы, очень сильно зависим от выбора гиперпараметров. Как в следующих статьях будет видно – "контролируемость"обучения диффузионок, понимание что происходит на каждой стадии сыграло важнейшую роль для многих задач и стало применимым уже, наверное, на всех доменах. При этом из очевидных минусов это очень долгий процесс семлирования: чтобы расшумить изначальный шум и создать картинку нам нужно честно проделать все 1000 шагов. В будущих работах ученые научились пропускать шаги и ускорять инференс, однако семплинг все еще сильно медленнее чем у ГАНов.

С VAE и потоками я знаком значительно меньше, однако насколько я знаю, они тоже вполне контролитруемы в отличии от ГАНов, но при этом сильно зависят от гиперпараметров и в среднем работают плохо, их очень тяжело завести