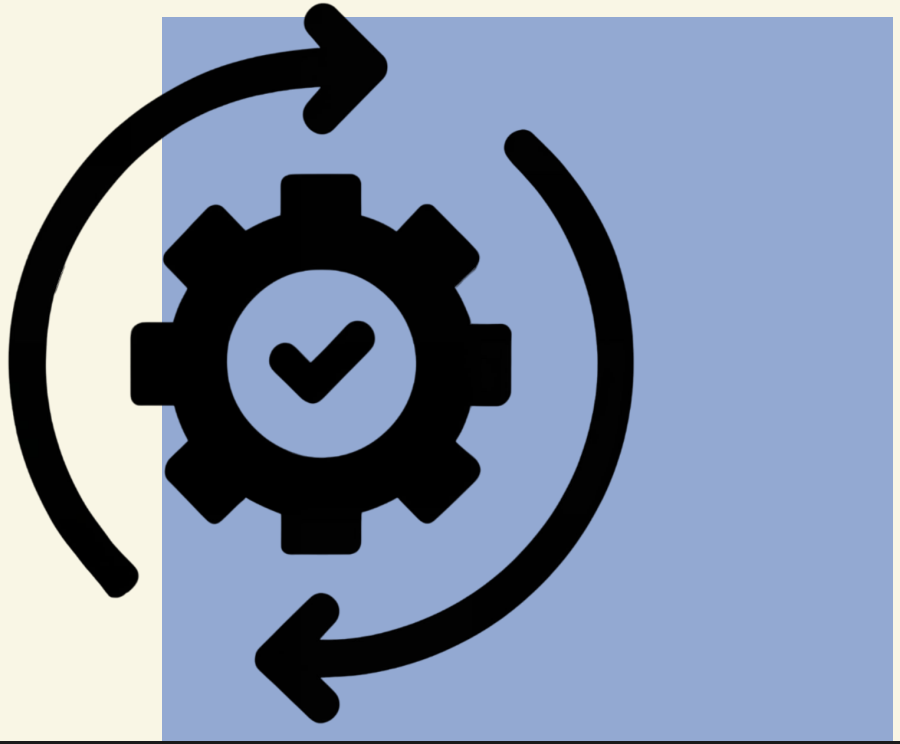
Эффективные системы машинного обучения



Лекция 11

Stable diffusion. Диффузия для решения задач

Преподаватель

Оганов Александр

6 декабря 2024

ВМК МГУ

Напоминание

Что мы знаем:

- Как задавать диффузионные модели в дискретном и непрерывном времени
- Как задать единое СДУ и обратное к нему
- Как перевести модель из одного СДУ в другое
- Как задать условную генерацию (classifier-free guidence)
- Как измерить качество (FID)
- Как решать ОДУ/СДУ и быстро генерировать (в 10-50 шагов)

Эффективные системы машинного обучения, ВМК МГУ
Занятие 11: Stable diffusion
Оганов Александр
6 декабря 2024

Напоминание

Что мы знаем:

- Как задавать диффузионные модели в дискретном и непрерывном времени
- Как задать единое СДУ и обратное к нему
- Как перевести модель из одного СДУ в другое
- Как задать условную генерацию (classifier-free guidence)
- Как измерить качество (FID)
- Как решать ОДУ/СДУ и быстро генерировать (в 10-50 шагов)

Что мы не знаем:

- Зачем это может потребоваться?

О чем поговорим?

- Как обучали большие диффузионные модели, в частности, Stable diffusion 1 (SD1)? Мы не будем рассматривать SD2, SD3, SDXL, SVD и другие крутые open-source модели
- Как мы можем контролировать генерацию: задать форму, глубину, контуры, референсы?
- Как можно использовать диффузионные модели в качестве функции потерь и какие новые задачи благодаря этому решаются

Оганов Александр

Stable diffusion

Задача генерации изображения по тексту называется text-to-image. Одной из самых первых и главных моделей можно назвать Stable diffusion (SD) из статьи High-Resolution Image Synthesis with Latent Diffusion Models (https://arxiv.org/abs/2112.10752)

Figure 13. Combining classifier free diffusion guidance with the convolutional sampling strategy from Sec. 4.3.2, our 1.45B parameter text-to-image model can be used for rendering images larger than the native 256^2 resolution the model was trained on.

CFG (напоминание)

Будем учить нейросеть, подавая метки класса для объекта который пытаемся зашумить (если зашумили 1 из MNIST, то подаем 1). Для обучения безусловной score функции создадим dummy переменную и будем использовать любые объекты.

$$\nabla_x \log p(x \mid y) + \gamma(\nabla_x \log p(x \mid y) - \nabla_x \log p(x)) =$$
$$= \nabla_x \log p(x \mid y) + \gamma(\nabla_x \log p(x \mid y) - \nabla_x \log p(x \mid \emptyset))$$

Именно CFG используют сейчас везде, особенно для text2img генерации, y – текстовый промт, а dummy класс – пустая строка

Что есть лучше?

Прошло уже 2 года с появления SD1 и мы уже умеем генерировать изображения лучше и быстрее. Например, одни из лучших моделей представлены стартапом Black Forest Labs (https://blackforestlabs.ai/announcing-black-forest-labs/) и доступны в открытом доступе

Проблемы высокого разрешения

Если наша цель сгенерировать изображение с разрешением в 1024 пикселя, то возникают проблемы. Как минимум, мы знаем, что трансформеры плохо масштабируются. По итогу, обучение диффузионной модели становится крайне дорогой и долгой задаче, но основная проблема во времени генерации.

Проблемы высокого разрешения

Если разрешение изображения слишком большое, то стоило бы его уменьшить, например с помощью VAE с маленьким штрафом KL (зачем?)

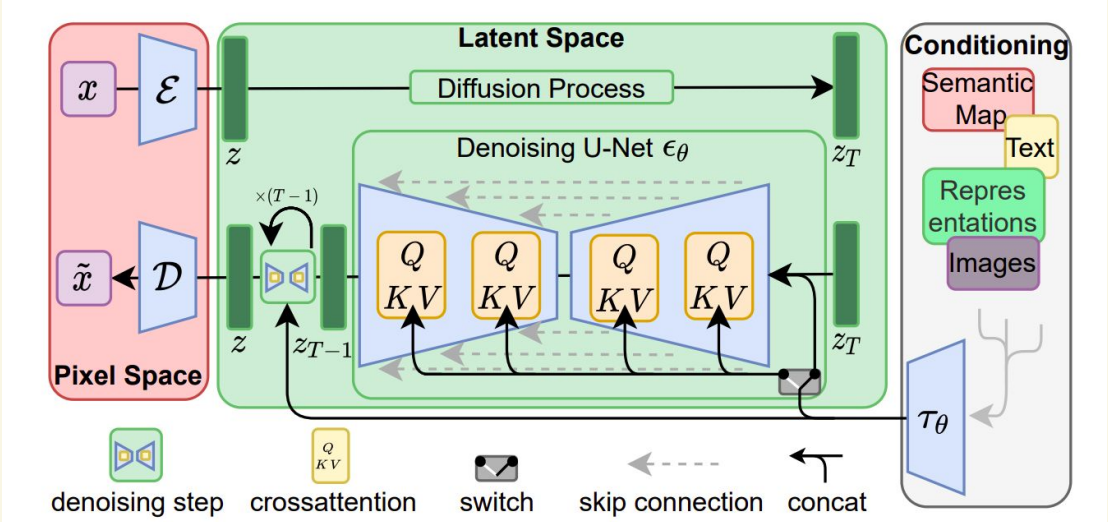


Figure 3. We condition LDMs either via concatenation or by a more general cross-attention mechanism. See Sec. 3.3

Как учить?

Если у нас есть обученный энкодер, то идейно обучение не поменяется

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

Стандартный лосс

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right]$$

Лосс в латетном представлении

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right]$$

Лосс для условной

Каким должен быть VAE

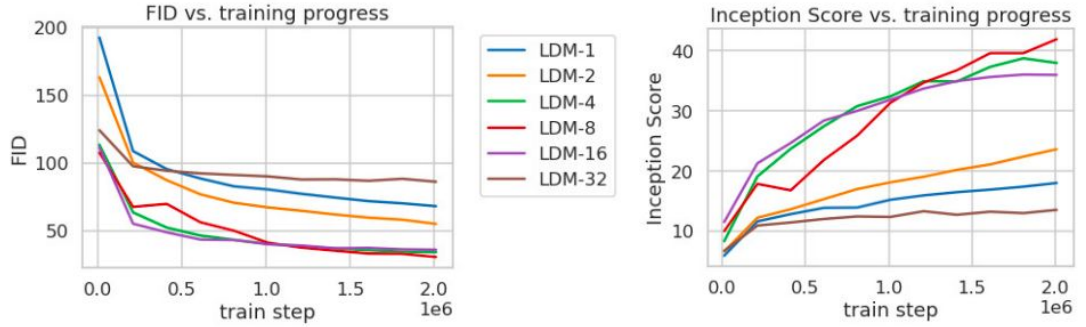


Figure 6. Analyzing the training of class-conditional *LDMs* with different downsampling factors f over 2M train steps on the ImageNet dataset. Pixel-based *LDM-1* requires substantially larger train times compared to models with larger downsampling factors (*LDM-*$\{$*4-16*$\}$). Too much perceptual compression as in *LDM-32* limits the overall sample quality. All models are trained on a single NVIDIA A100 with the same computational budget. Results obtained with 100 DDIM steps [84] and $\kappa = 0$.

Результаты (uncoditional)

LSUN-Churches 256 × 256				LSUN-Bedrooms 256 × 256			
Method	FID ↓	Prec. ↑	Recall ↑	**Method**	FID ↓	Prec. ↑	Recall ↑
DDPM [30]	7.89	-	-	ImageBART [21]	5.51	-	-
ImageBART [21]	7.32	-	-	DDPM [30]	4.9	-	-
PGGAN [39]	6.42	-	-	UDM [43]	4.57	-	-
StyleGAN [41]	4.21	-	-	StyleGAN [41]	2.35	0.59	0.48
StyleGAN2 [42]	3.86	-	-	ADM [15]	1.90	**0.66**	**0.51**
ProjectedGAN [76]	**1.59**	0.61	0.44	ProjectedGAN [76]	**1.52**	0.61	0.34
*LDM-8** (ours, 200-s)	4.02	**0.64**	**0.52**	*LDM-4* (ours, 200-s)	2.95	**0.66**	0.48

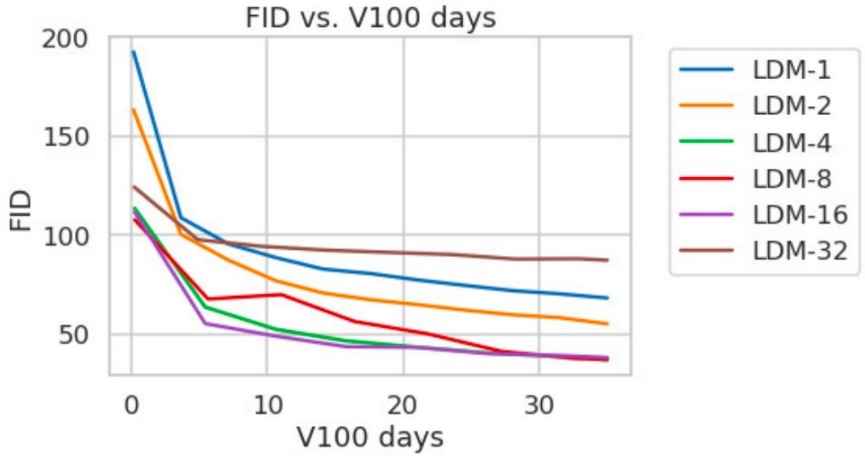
Table 1. Evaluation metrics for unconditional image synthesis. CelebA-HQ results reproduced from [43, 63, 100], FFHQ from [42, 43]. †: N-s refers to N sampling steps with the DDIM [84] sampler. *: trained in KL-regularized latent space. Additional results can be found in the supplementary.

Результаты (conditional)

Text-Conditional Image Synthesis				
Method	FID ↓	IS↑	Nparams	
CogView† [17]	27.10	18.20	4B	self-ranking, rejection rate 0.017
LAFITE† [109]	26.94	<u>26.02</u>	75M	
GLIDE* [59]	<u>12.24</u>	-	6B	277 DDIM steps, c.f.g. [32] $s = 3$
Make-A-Scene* [26]	**11.84**	-	4B	c.f.g for AR models [98] $s = 5$
LDM-KL-8	23.31	$20.03_{\pm 0.33}$	1.45B	250 DDIM steps
*LDM-KL-8-G**	12.63	$\mathbf{30.29}_{\pm \mathbf{0.42}}$	1.45B	250 DDIM steps, c.f.g. [32] $s = 1.5$

Table 2. Evaluation of text-conditional image synthesis on the 256 × 256-sized MS-COCO [51] dataset: with 250 DDIM [84] steps our model is on par with the most recent diffusion [59] and autoregressive [26] methods despite using significantly less parameters. †/*:Numbers from [109]/ [26]

Не повторять в домашних условиях



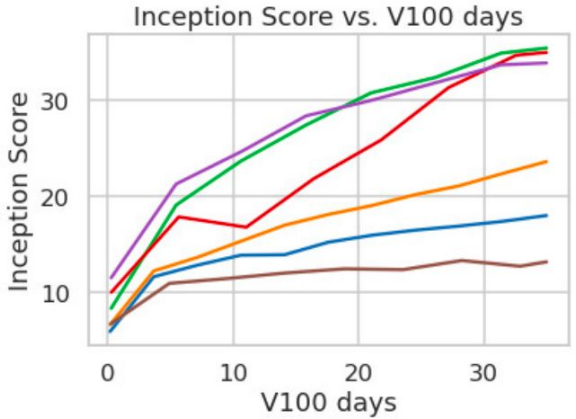
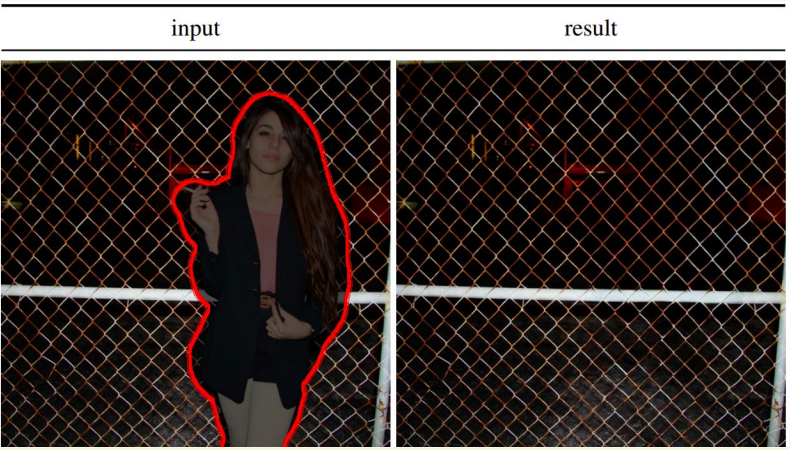
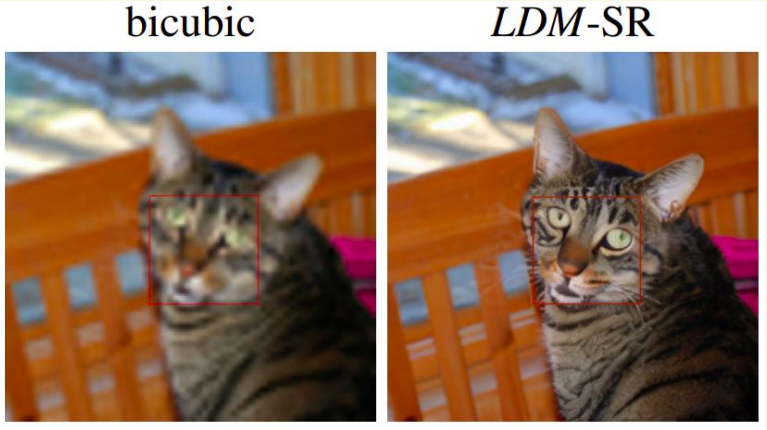
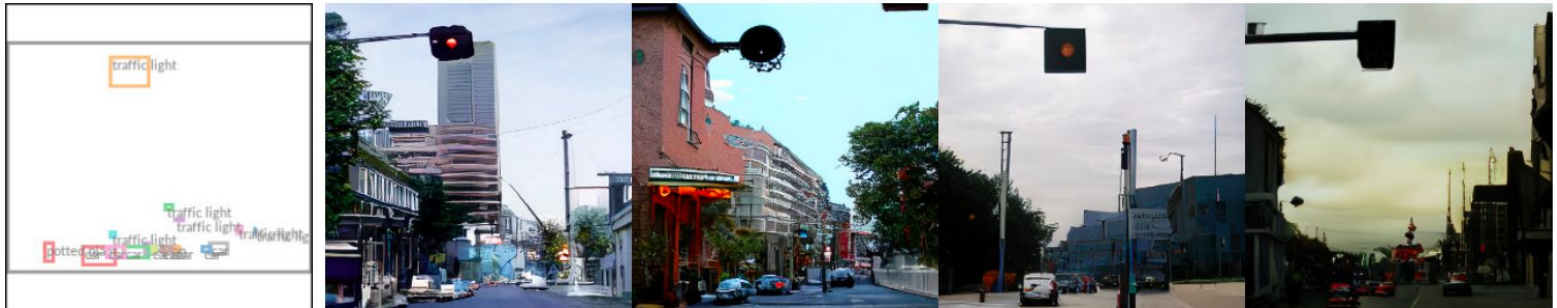


Figure 17. For completeness we also report the training progress of class-conditional *LDMs* on the ImageNet dataset for a fixed number of 35 V100 days. Results obtained with 100 DDIM steps [84] and $\kappa = 0$. FIDs computed on 5000 samples for efficiency reasons.

Что умеем еще?



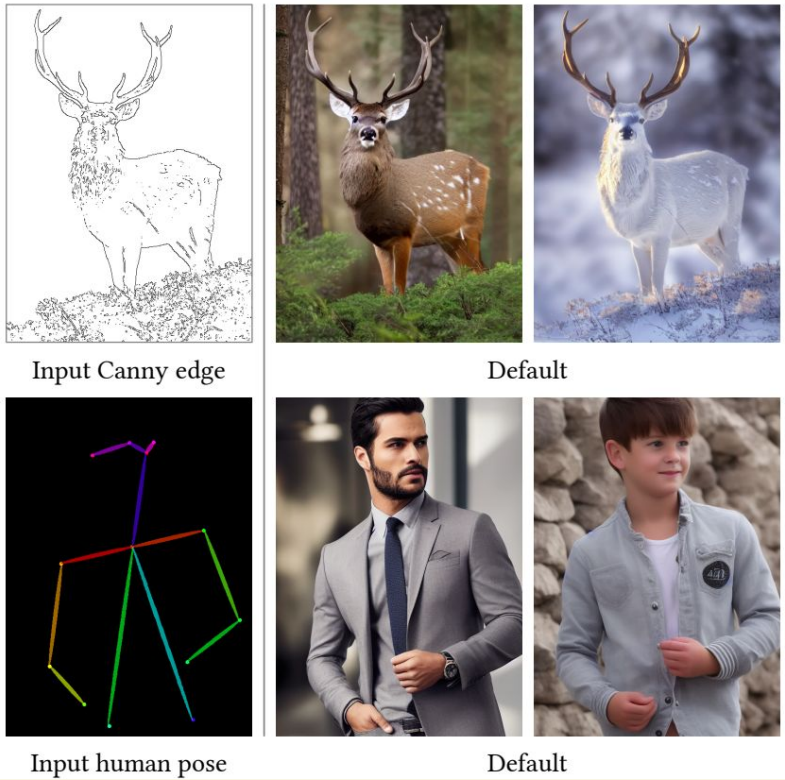




ControlNet

Если мы уже умеем пользоваться условной генерацией, то почему бы не расширить эти знания?

Для контролирования генерации есть архитектура ControlNet (https://arxiv.org/abs/2302.05543)



ControlNet (базовый блок)

Основная идея: будем учить смещение к уже обученным весам нейронной сети. Тем самым мы, скорее всего, не сильно испортим уже обученную диффузионную модель

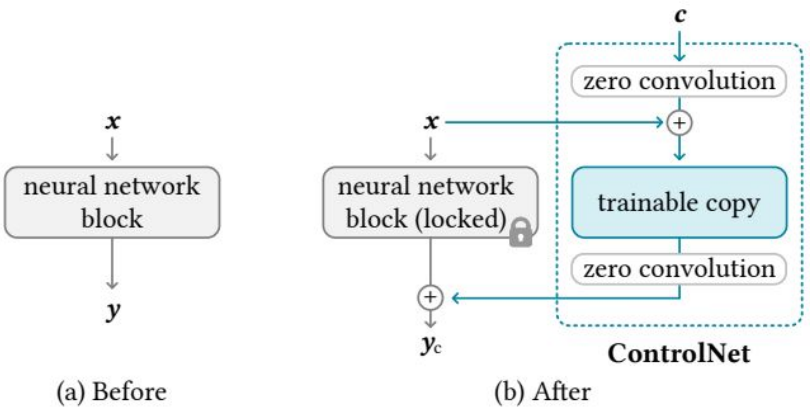
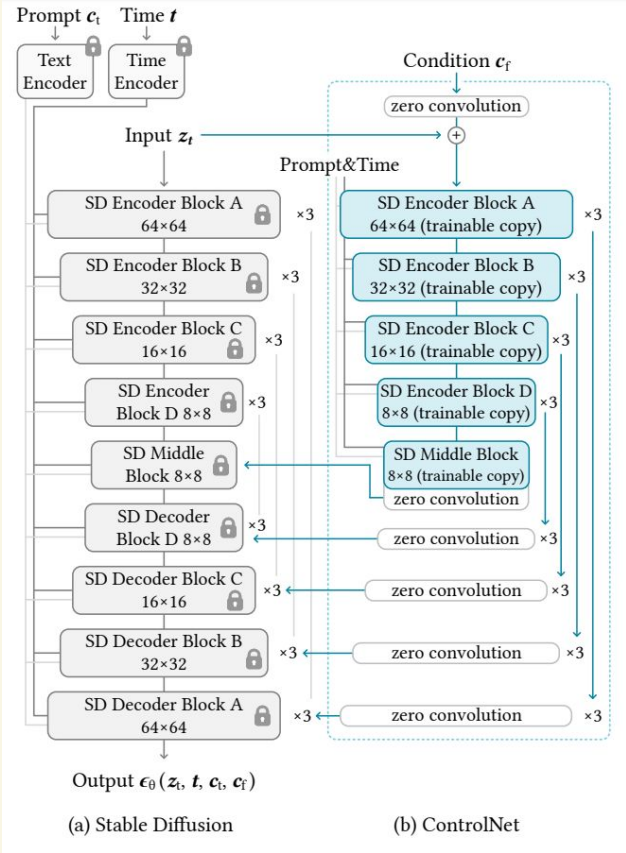


Figure 2: A neural block takes a feature map x as input and outputs another feature map y, as shown in (a). To add a ControlNet to such a block we lock the original block and create a trainable copy and connect them together using zero convolution layers, *i.e.*, 1×1 convolution with both weight and bias initialized to zero. Here c is a conditioning vector that we wish to add to the network, as shown in (b).

ControlNet (SD)

Для примера “SD Encoder Block A” содержит 4 resnet блока и 2 ViT блока, а “×3” показывает, что блоки повторяются 3 раза



ControlNet (обучение)

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{z}_0, \boldsymbol{t}, \boldsymbol{c}_t, \boldsymbol{c}_\mathrm{f}, \epsilon \sim \mathcal{N}(0,1)} \left[\| \epsilon - \epsilon_\theta(\boldsymbol{z}_t, \boldsymbol{t}, \boldsymbol{c}_t, \boldsymbol{c}_\mathrm{f}) \|_2^2 \right]$$

Figure 12: Transfer pretrained ControlNets to community models [16, 61] without training the neural networks again.

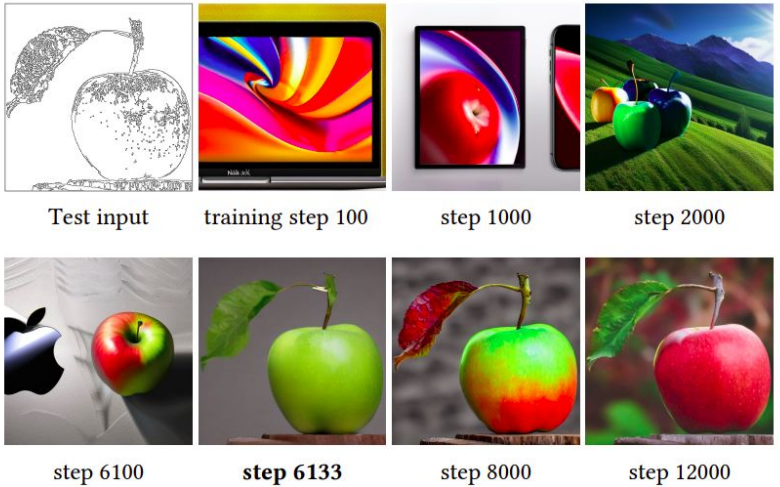


Figure 4: The sudden convergence phenomenon. Due to the zero convolutions, ControlNet always predicts high-quality images during the entire training. At a certain step in the training process (*e.g.*, the 6133 steps marked in bold), the model suddenly learns to follow the input condition.

ControlNet (итоги)

Важно, что архитектуру и топологию нейронной сети меняют крайне редко, поэтому ControlNet обученный на SD1 часто применим к другим чекпоинтам и моделям. Со временем возможности контролируемой генерации увеличиваются (https://github.com/lllyasviel/ControlNet-v1-1-nightly)

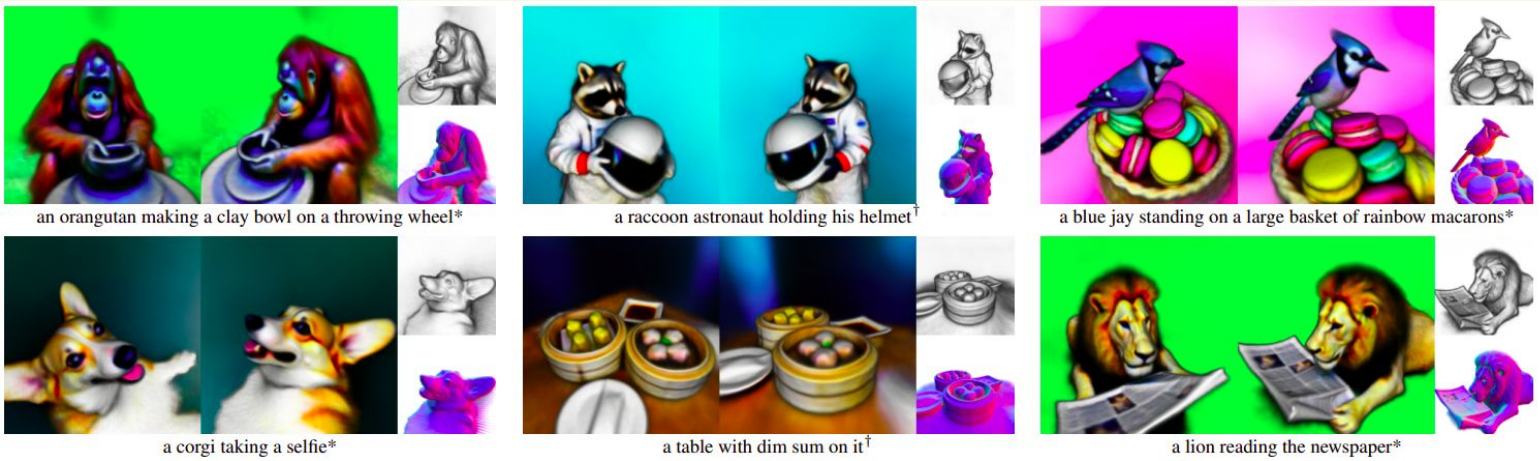
Dream fusion

Допустим, мы умеем решать text-to-image, что еще можно решить с помощью диффузионок?

Dream fusion

Допустим, мы умеем решать text-to-image, что еще можно решить с помощью диффузионок?

В работе DreamFusion: Text-to-3D using 2D Diffusion (https://arxiv.org/abs/2209.14988) предлагают решать задачу text-to-3D



Dream fusion (обучение)

Допустим, мы умеем дифференцировано рендерить изображения с разных углов (например с помощью NERF)

θ – параметры сцены

z – угол

$x = g(\theta, z)$

$$\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(t)\|\epsilon_\phi(\alpha_t\mathbf{x} + \sigma_t\epsilon; t) - \epsilon\|_2^2\right]$$

Все готово! Фиксируем диффузию и учим параметры сцены, правда же?....

Score Distillation Sampling

К сожалению, когда вы работаете с зашумлением и генерацией все не может быть стабильно и хорошо... Для начала посмотрим на градиент функции потерь:

$$\nabla_\theta \mathcal{L}_{\text{Diff}}(\phi, \mathbf{x} = g(\theta)) = \mathbb{E}_{t,\epsilon}\left[w(t) \underbrace{(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon)}_{\text{Noise Residual}} \underbrace{\frac{\partial \hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\mathbf{z}_t}}_{\text{U-Net Jacobian}} \underbrace{\frac{\partial \mathbf{x}}{\partial \theta}}_{\text{Generator Jacobian}} \right]$$

На практике такой лосс не работает (обучение слишком шумное), поэтому появился Score Distillation Sampling лосс:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon}\left[w(t) (\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \right]$$

Score Distillation Sampling

Единственное отличие – мы убрали якобиан U-Net по зашумленному изображению. Почему такая замена имеет смысл и вообще работает?

Ответ (практический): так эффективней учится и так проще
Ответ (теоретический): надо рассмотреть градиент и понять для какой функции это верно (подробнее можно прочитать на курсе 3D Computer Vision)

$$\mathcal{L}_{Diff} \approx -\log p_0(x_0)$$

$$\mathcal{L}_{SDS} \approx -\mathbb{E}_t w(t) \mathbb{E}_{p_t(x_t|x_0)} \log p_t(x_t)$$

Dream fusion (итог)

С помощью обученной диффузионной модели и правильного лосса, можно научиться решать задачи 3D генерации, то есть генерировать изображения одной сцены с разных ракурсов

На практике все еще необходимо добавить немного эвристик и магии, но это вопросы для совсем другого курса)

Distribution Matching Distillation

Что если мы хотим ускорить диффузию, например, сделать ее одношаговой?

Допустим, у нас уже есть обученная диффузионная модель и наша цель обучить генератор (одношаговую модель) с помощью дистилляции (перенести знания из диффузионки в генератор). Для этой цели есть много разных подходов, например, на основе Consistency Models (https://arxiv.org/abs/2303.01469)

Мы же посмотрим на статью One-step Diffusion with Distribution Matching Distillation

Эффективные системы машинного обучения, ВМК МГУ
Занятие 11: Диффузия для решения задач
Оганов Александр
6 декабря 2024

DMD (цель)

Формально нашу цель можно записать как минимизацию KL-дивергенции между реальным распределением (генерация из диффузионки) и фейковым распределением (генерация из генератора):

$$
\begin{aligned}
D_{KL}\left(p_{\text{fake}} \parallel p_{\text{real}}\right) &= \mathop{\mathbb{E}}_{x \sim p_{\text{fake}}} \left(\log \left(\frac{p_{\text{fake}}(x)}{p_{\text{real}}(x)} \right) \right) \\
&= \mathop{\mathbb{E}}_{\substack{z \sim \mathcal{N}(0;\mathbf{I}) \\ x = G_\theta(z)}} -\left(\log\ p_{\text{real}}(x) - \log\ p_{\text{fake}}(x) \right)
\end{aligned}
$$

DMD (градиенты)

Вопрос: Почему не начать просто оптимизировать? В чем проблема посчитать скор функцию?

$$D_{KL}\left(p_{\text{fake}} \parallel p_{\text{real}}\right) = \mathop{\mathbb{E}}_{x \sim p_{\text{fake}}} \left(\log \left(\frac{p_{\text{fake}}(x)}{p_{\text{real}}(x)} \right) \right)$$

$$= \mathop{\mathbb{E}}_{\substack{z \sim \mathcal{N}(0;\mathbf{I}) \\ x = G_\theta(z)}} -\left(\log\, p_{\text{real}}(x) - \log\, p_{\text{fake}}(x) \right)$$

$$\nabla_\theta D_{KL} = \mathop{\mathbb{E}}_{\substack{z \sim \mathcal{N}(0;\mathbf{I}) \\ x = G_\theta(z)}} \left[-\left(s_{\text{real}}(x) - s_{\text{fake}}(x) \right) \frac{dG}{d\theta} \right]$$

DMD (градиенты)

Вопрос: Почему не начать просто оптимизировать? В чем проблема посчитать скор функцию?

$$\nabla_\theta D_{KL} = \mathop{\mathbb{E}}_{\substack{z\sim\mathcal{N}(0;\mathbf{I}) \\ x=G_\theta(z)}} \left[-\left(s_{\text{real}}(x) - s_{\text{fake}}(x)\right) \frac{dG}{d\theta} \right]$$

DMD (градиенты)

Вопрос: Почему не начать просто оптимизировать? В чем проблема посчитать скор функцию?

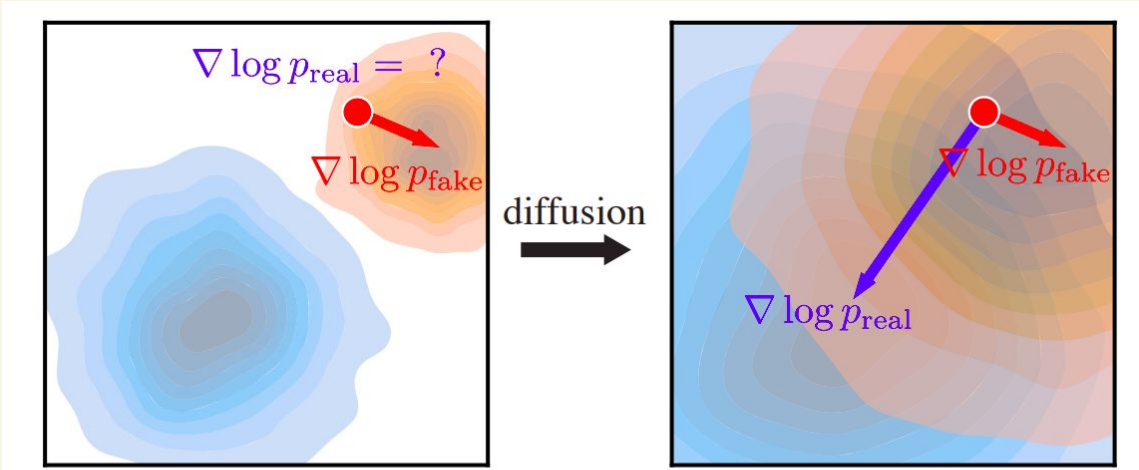
Ответ:

1) Мы не умеем считать скор функцию для чистых данных
2) Даже если бы и умели, то у фейковых изображений была бы низкая вероятность с точки зрения настоящего распределения

$$\nabla_\theta D_{KL} = \mathop{\mathbb{E}}_{\substack{z\sim\mathcal{N}(0;\mathbf{I}) \\ x=G_\theta(z)}} \left[-\left(s_{\text{real}}(x) - s_{\text{fake}}(x)\right) \frac{dG}{d\theta} \right]$$

DMD (решение)

Будем считать скор не для “чистых изображений” а для “зашумленных”, что уже является стандартной задачей. По сути мы размываем распределения, что позволяет легче учить и считать градиенты



(a) for unperturbed distributions, both scores may not be defined simulateneously everywhere

(b) after diffusion, the distributions overlap, making our objective well-defined

DMD

Со всеми хитростями мы получим лосс:

$$\nabla_\theta D_{KL} \simeq \mathop{\mathbb{E}}_{z,t,x,x_t} \left[w_t \alpha_t \left(s_{\text{fake}}(x_t, t) - s_{\text{real}}(x_t, t) \right) \frac{dG}{d\theta} \right]$$

Всегда нужны какие-то эвристики…

DMD

Со всеми хитростями мы получим лосс:

$$\nabla_\theta D_{KL} \simeq \mathop{\mathbb{E}}_{z,t,x,x_t} \left[w_t \alpha_t \left(s_{\text{fake}}(x_t, t) - s_{\text{real}}(x_t, t) \right) \frac{dG}{d\theta} \right]$$

Всегда нужны какие-то эвристики…

$$w_t = \frac{\sigma_t^2}{\alpha_t} \frac{CS}{||\mu_{\text{base}}(x_t, t) - x||_1}$$

Еще немного идей

Предложенный лосс будет хорошо работать при t >> 0 (именно об этом говорит интуиция с размытием распределений), но как он будет работать при t около 0? На практике это решается добавлением регрессионного лосса:

$$\mathcal{L}_{\text{reg}} = \mathop{\mathbb{E}}_{(z,y)\sim\mathcal{D}} \ell(G_\theta(z), y)$$

DMD (итог)

Мы очень кратко посмотрели на статью, немного коснулись обучения и как переносятся знания от учителя к ученику. Более подробно и с деталями можно почитать в оригинальной статье (https://arxiv.org/abs/2311.18828).

Мы получили **метод матчинга двух распределений** с помощью диффузионных моделей! (а они учатся куда лучше, чем GAN)

Method	# Fwd Pass (↓)	FID (↓)
BigGAN-deep [4]	1	4.06
ADM [9]	250	**2.07**
Progressive Distillation [65]	1	15.39
DFNO [92]	1	7.83
BOOT [16]	1	16.30
TRACT [3]	1	7.43
Meng et al. [51]	1	7.54
Diff-Instruct [50]	1	5.57
Consistency Model [75]	1	6.20
DMD (Ours)	1	**2.62**
EDM† (Teacher) [31]	512	2.32

Итоги

Мы очень кратко затронули новые области применения диффузии:

- Stable diffusion и подобное, как генерация изображений по тексту
- ControlNet для контролируемой генерации
- DreamFusion для обобщения диффузии на 3D
- SDS лосс для обучения с учетом правдоподобия объекта на всех уровнях шума
- DMD как один из способов дистилляции знаний из диффузии, но еще и способ матчинга распределений с помощью обучения диффузии

При обучении диффузии мы извлекаем достаточно знаний, чтобы строить новые модели используя скор функции и знания из обученных диффузионных моделей!