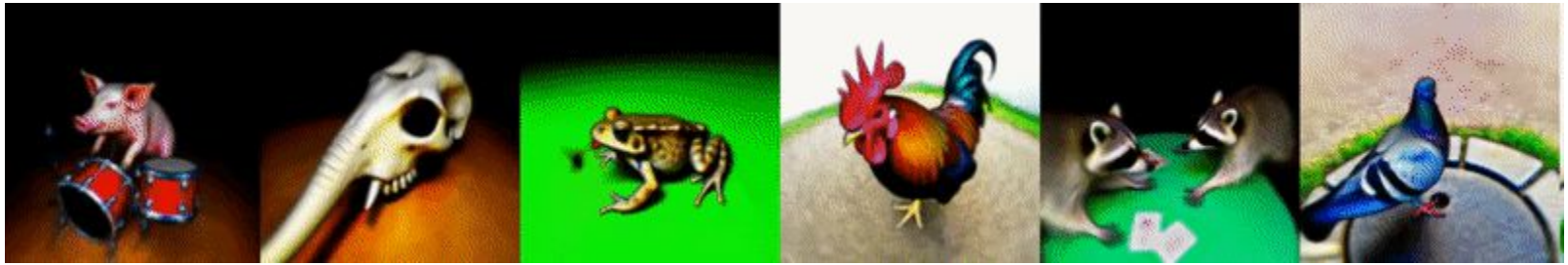
DreamFusion: Text-to-3D using 2D Diffusion

Докладчик: Казанцев Даниил
Рецензент: Бакланов Алексей
Хакер: Фролова Анна

Проблематика

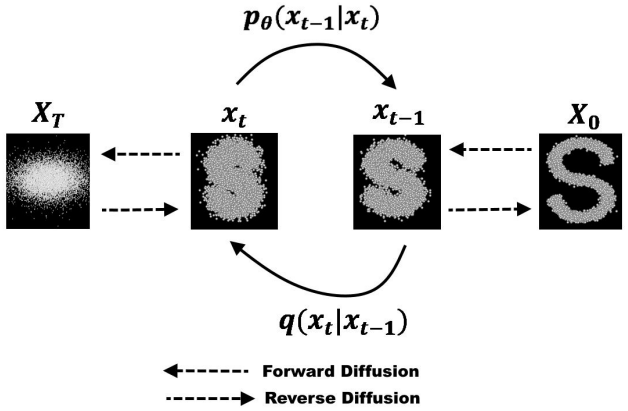
Хотим создавать полноценные 3D-модели по различным текстовым запросам, но:

- Данных достаточно мало. Собрать датасет из 3D-моделей с текстовым описанием значительно труднее, чем сделать то же самое с 2D
- Существующие архитектуры плохо работают с 3D



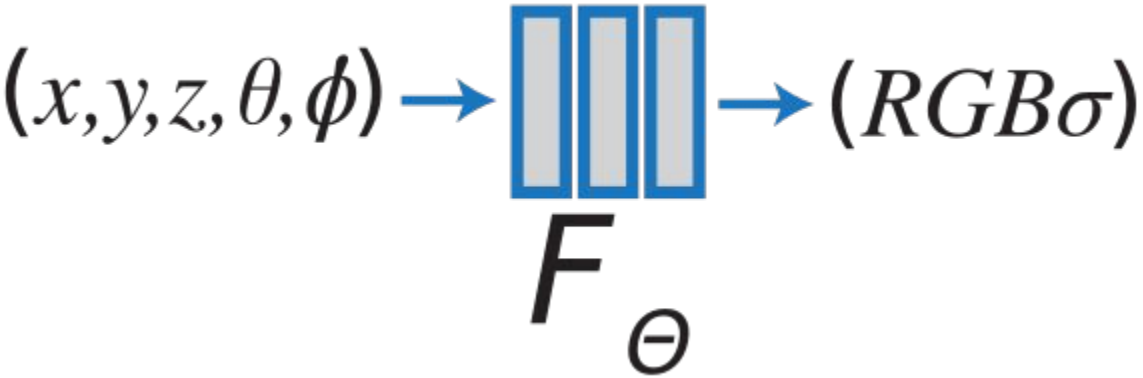
Диффузионные модели

- Хорошо справляются с генерацией 2D-изображений по текстовому описанию
- Обучаются на парах изображение-текст, собрать датасет проще
- Уже есть обученные модели
- Обучаются предсказывать шум, добавленный на определенном шаге к исходному изображению
- Для генерации изображения расшумляют случайный шум с поправкой на текстовое описание



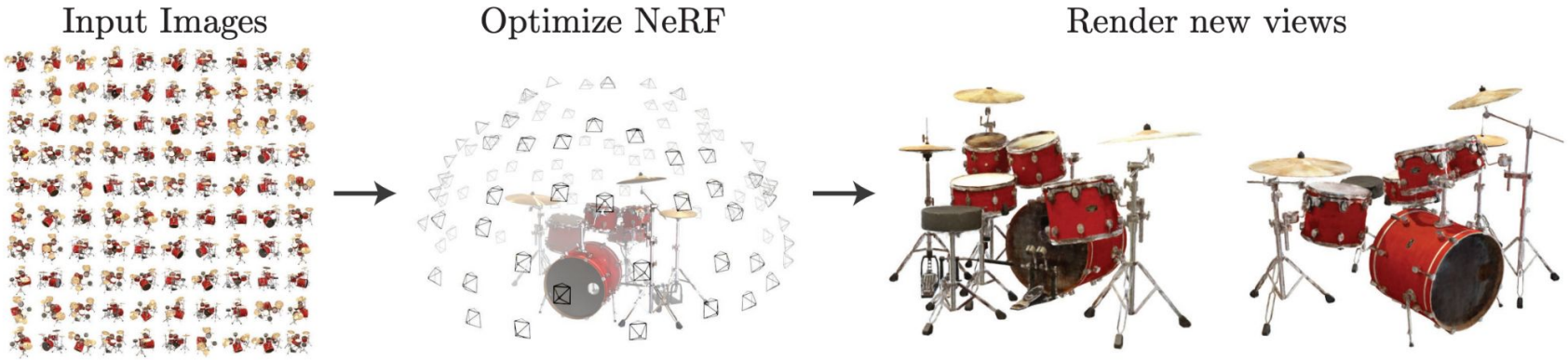
Neural Radiance Fields (NeRF)

- Полносвязная нейронная сеть
- Принимает на вход координаты точки в пространстве и углы положения камеры
- Возвращает цвет и прозрачность вокселя



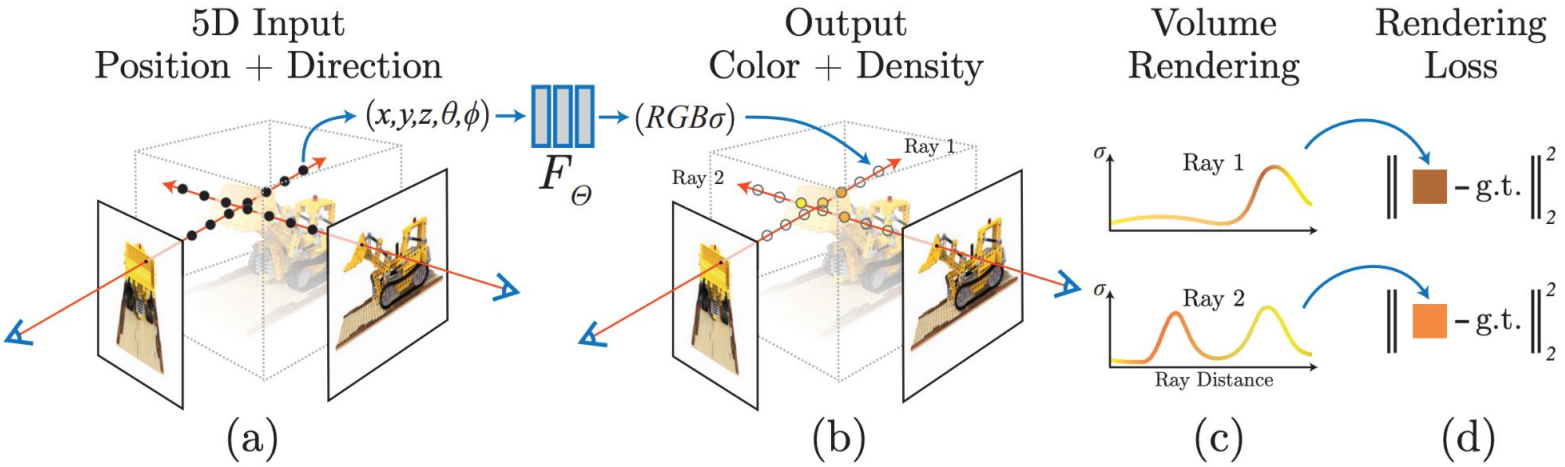
Neural Radiance Fields (NeRF)

- Один NeRF - одна 3D-модель
- Для обучения необходимо множество изображений одного и того же объекта под разными углами



Neural Radiance Fields (NeRF)

- Для генерации новых изображений и процесса обучения используется трейсинг лучами с суммированием полученных вдоль луча цветов согласно их удаленности и прозрачности

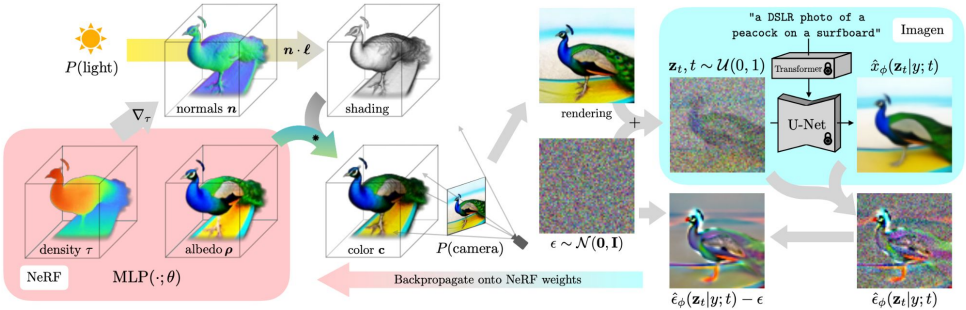


Как все это использовать?

Основная идея: обучить случайно инициализированный NeRF, используя вместо 3D-моделей предобученную диффузионную модель

Алгоритм:

- Random camera and light sampling
- Rendering
- Diffusion loss with view-dependent conditioning
- Optimization



Random camera and light sampling

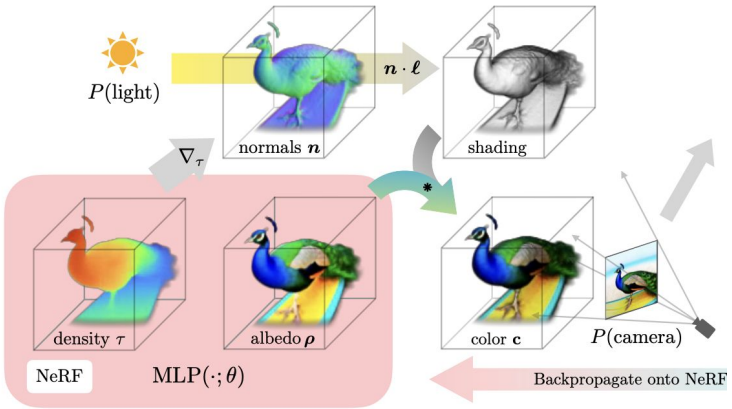
- Сэмплируем положение камеры в сферических координатах:
  - зенитный угол [-10º, 90º]
  - азимутальный угол [0º, 360º]
  - расстояние до центра [1, 1.5]
- Мультипликатор фокусного расстояния
- Источник света:
  - положение вокруг камеры
  - интенсивность
  - интенсивность окружающего света

Rendering

- Изменили NeRF на модифицированную версию. Теперь он предсказывает volumetric density и albedo
- Запускаем лучи из камеры в каждый пиксель изображения, каждый луч - множество точек, в которых модель дает ответ
- На основе полученных от модели данных считается карта нормалей и добавляется освещение

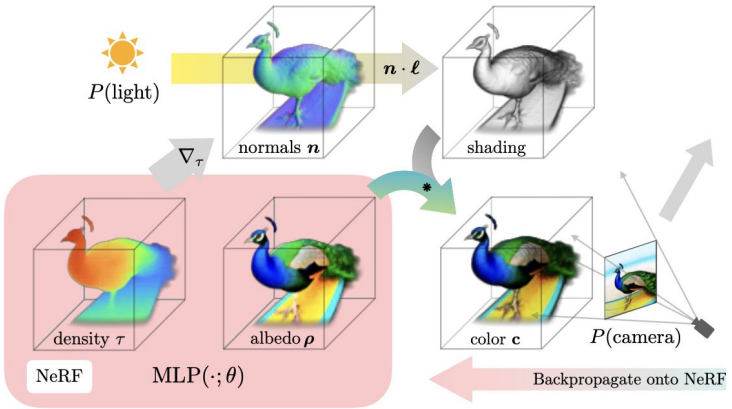
$$\mathbf{c} = \boldsymbol{\rho} \circ (\boldsymbol{\ell}_\rho \circ \max(0, \boldsymbol{n} \cdot (\boldsymbol{\ell} - \boldsymbol{\mu}) / \|\boldsymbol{\ell} - \boldsymbol{\mu}\|) + \boldsymbol{\ell}_a)$$

- Получаем итоговое изображение



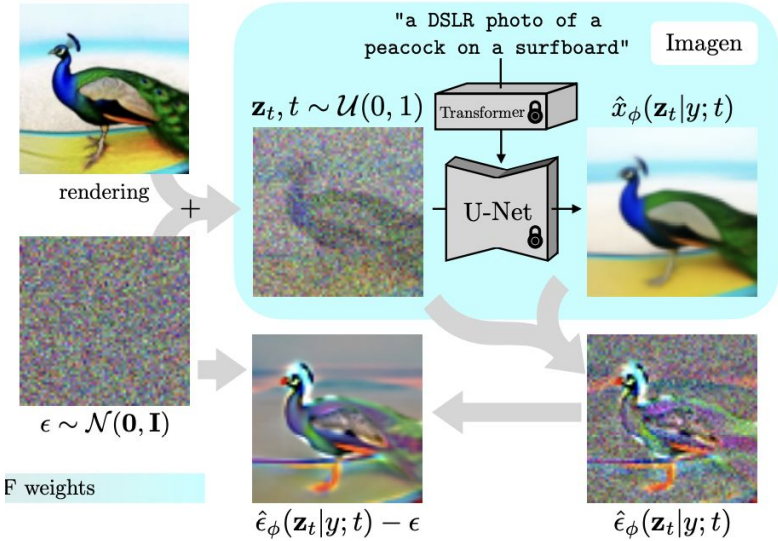
Rendering. Особенности

- Изображение 64x64
- При обучении случайно выбирается рендеринг из:
  - обычный полноценный рендеринг
  - рендеринг без текстуры (замена albedo на белый цвет)
  - рендеринг без теней
- Фон генерируется отдельной моделью



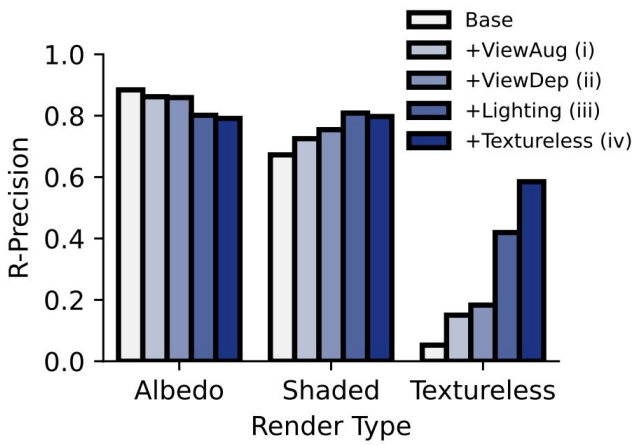
Diffusion loss with view-dependent conditioning

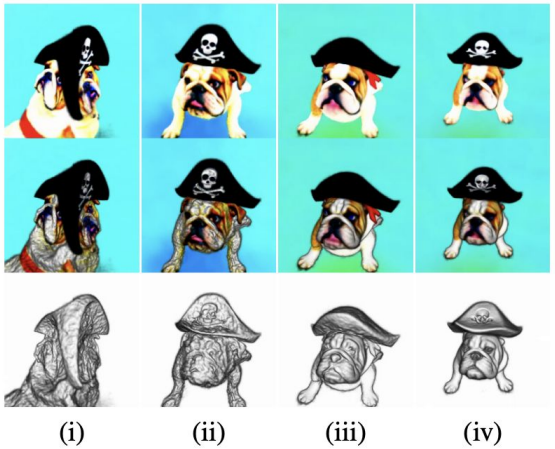
- Сэмплируем t и шум
- Добавляем шум к изображению из NeRF
- Подаем на вход диффузионной модели зашумленное изображение и t вместе с целевым текстовым запросом
- Считаем градиент по функции потерь, обновляем NeRF



Diffusion loss with view-dependent conditioning

Для того, чтобы диффузионная модель могла генерировать изображения под определенным углом к эмбеддингу текста добавляются взвешенные согласно углу камеры эмбеддинги ключевых слов: “front view”, “side view”, “back view” и т. п.





Функция потерь

Изначально хотели использовать функцию потерь диффузионок:

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

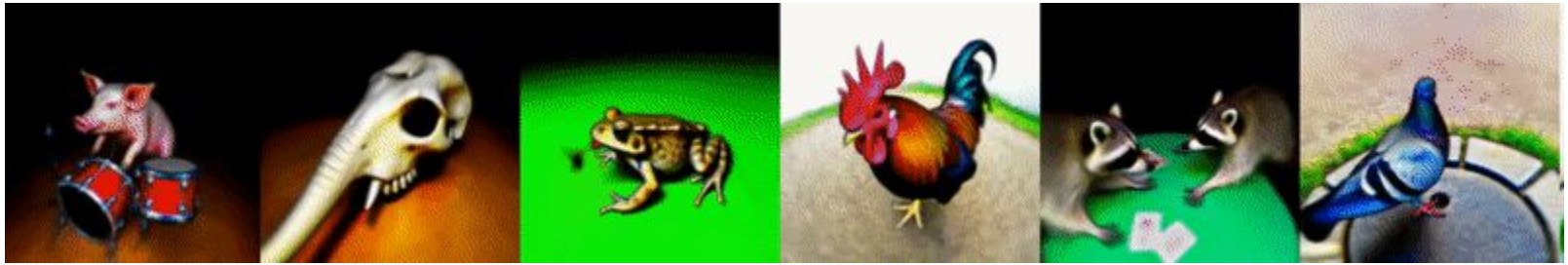
$$\nabla_\theta\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x}=g(\theta)) = \mathbb{E}_{t,\epsilon}\left[w(t)\underbrace{(\hat{\epsilon}_\phi(\mathbf{z}_t;y,t)-\epsilon)}_{\text{Noise Residual}}\underbrace{\frac{\partial\hat{\epsilon}_\phi(\mathbf{z}_t;y,t)}{\mathbf{z}_t}}_{\text{U-Net Jacobian}}\underbrace{\frac{\partial\mathbf{x}}{\partial\theta}}_{\text{Generator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

Но якобиан U-Net долго считается и плохо обусловлен при маленьких t, поэтому его выкинули из градиента. После долгих преобразований получилось:

$$\nabla_\theta\mathcal{L}_{\text{SDS}}(\phi, \mathbf{x}=g(\theta)) = \nabla_\theta\mathbb{E}_t\left[\sigma_t/\alpha_t w(t)\mathbf{KL}(q(\mathbf{z}_t|g(\theta);y,t)\|p_\phi(\mathbf{z}_t;y,t))\right]$$

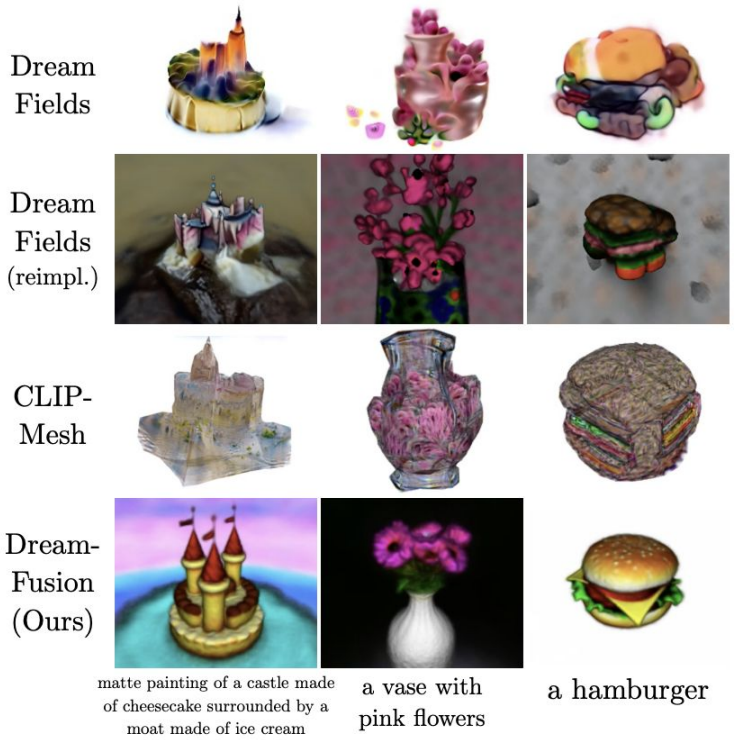
Генерация модели

- Обучаем на текстовом запросе NeRF по предложенному алгоритму
- Генерируем необходимые точки обзора и получаем изображения с них от NeRF
- Собираем из полученных значений 3D модель



Эксперименты

Method	R-Precision ↑					
	CLIP B/32		CLIP B/16		CLIP L/14	
	Color	Geo	Color	Geo	Color	Geo
GT Images	77.1	–	79.1	–	–	–
Dream Fields (reimpl.)	68.3 **78.6**	– 1.3	74.2 (99.9)	– (0.8)	– **82.9**	– 1.4
CLIP-Mesh	67.8	–	75.8	–	74.5†	–
DreamFusion	75.1	**42.5**	**77.5**	46.6	79.7	**58.5**



Все!