ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 21: NeRF; Gaussian Splatting;
Generative Models; Diffusion Model

04/02/2025

https://deeprob.org/w25/

M | ROBOTICS

Today

- Feedback and Recap (5min)
- Lightning Talk Review (5min)
- Nerf (20min)
- Gaussian Splatting (15min)
- Generative Models (10min)
- Diffusion Models (15min)
- Summary and Takeaways (5min)

Some feedback on the “Lightning Talk”

- 5-min: long or short?
- Connection to lecture content - Thanks!

  (LORA, pose, RL, ResNet, surface norm, cnn/ViT/nerf/splatting, etc.....)
- Dataset
- Be mindful of training time/timeline in general
- Collaboration - divide and conquer
- Encourage explore more recent SOTA methods
- *Bonus for live robot demo!*
- *Encourage further extension of the work*

*Comments and feedback will be released via **Canvas**

April 22nd, final project showcase @FRB atrium
April 28th, final project (report, code, video/website) DUE

M | ROBOTICS

Neural Radiance Field (NeRF) For View Synthesis

https://arxiv.org/pdf/2003.08934

View Synthesis

https://www.matthewtancik.com/nerf

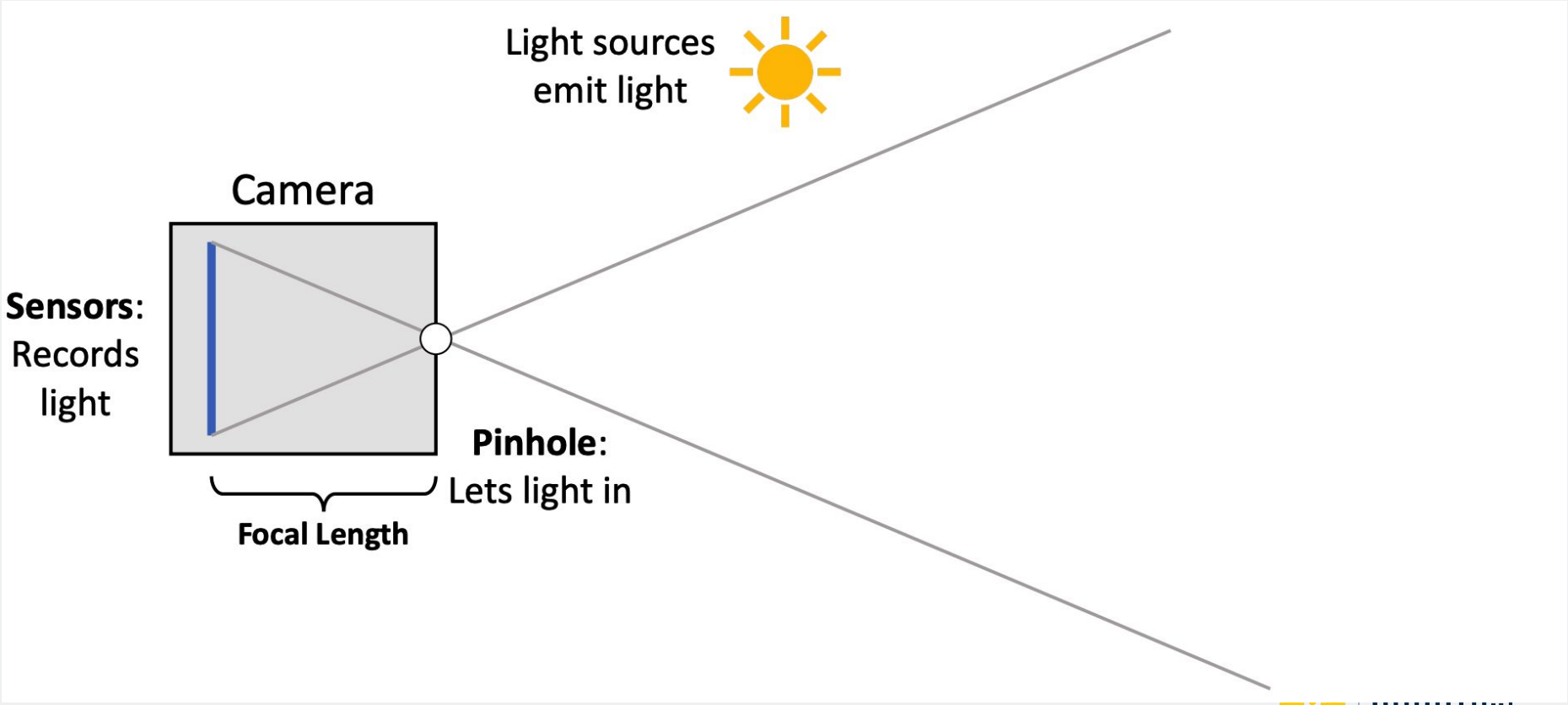
Input: Many images of the same scene (with known camera parameters)

Output: Images showing the scene from novel viewpoints

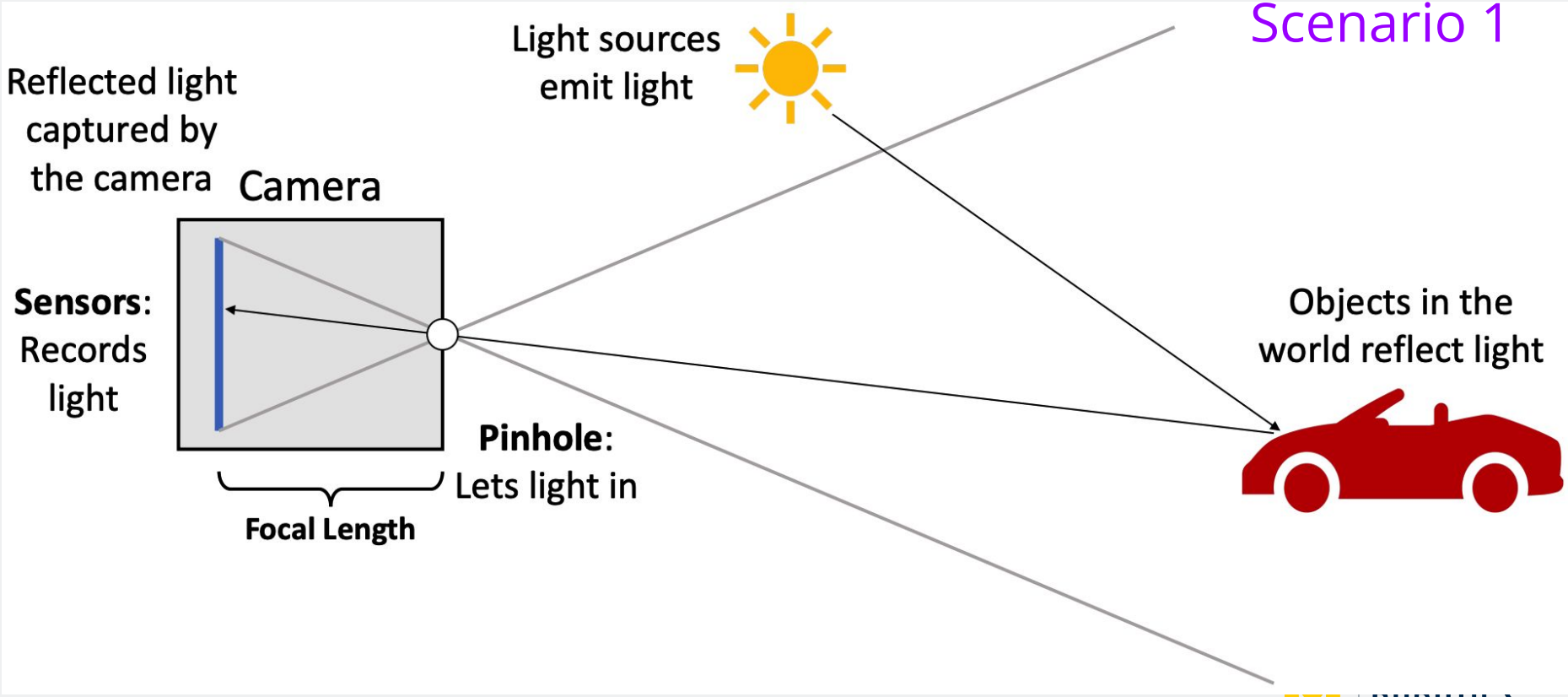
Scene rendering

M | ROBOTICS

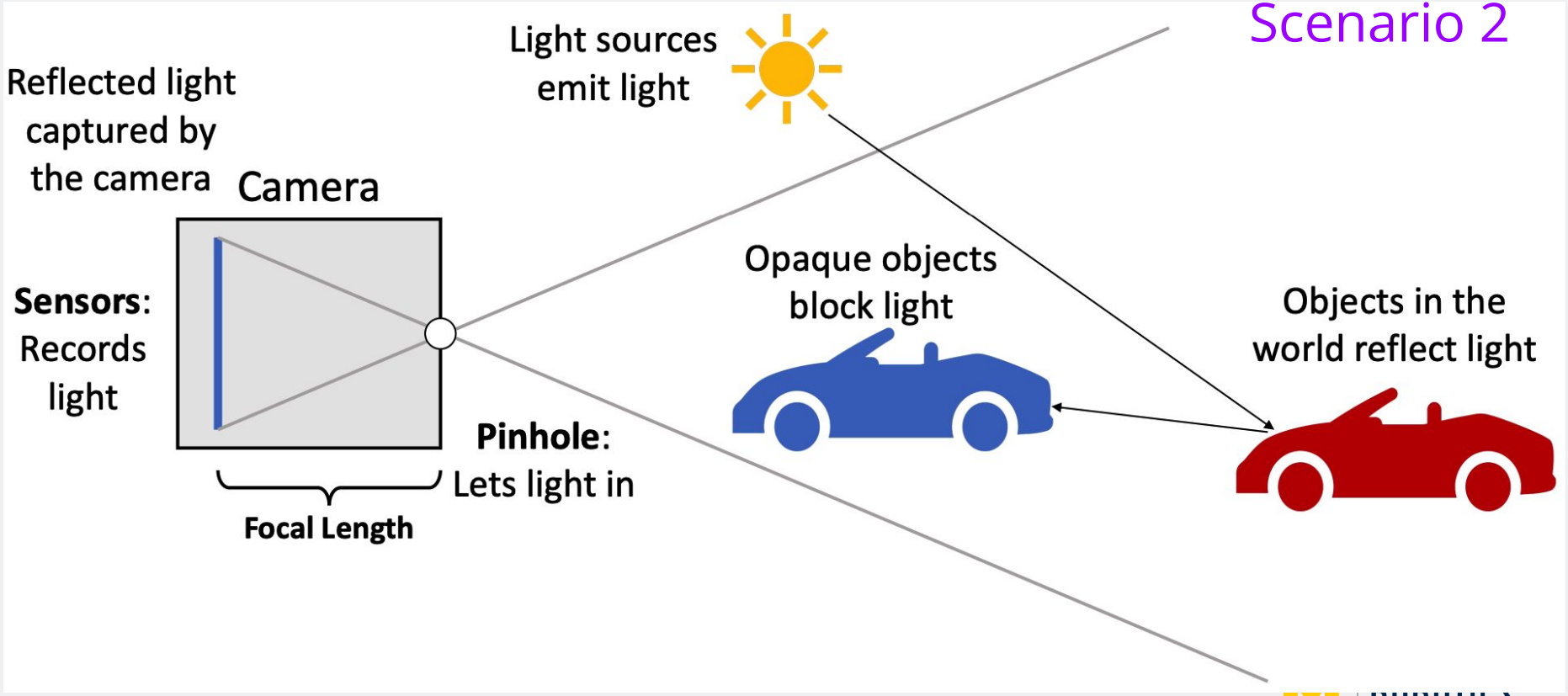
Stepping Back: Pinhole Camera Model



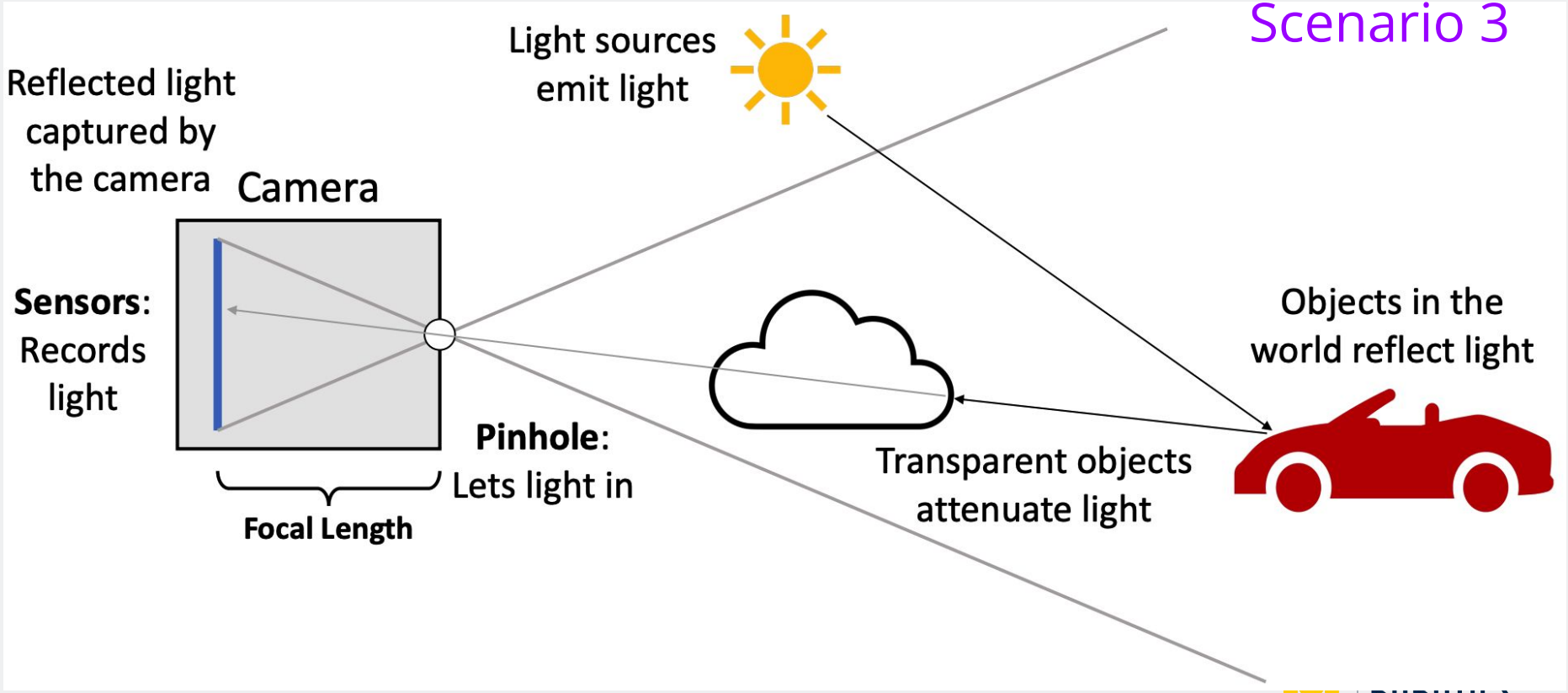
Stepping Back: Pinhole Camera Model



Stepping Back: Pinhole Camera Model



Stepping Back: Pinhole Camera Model



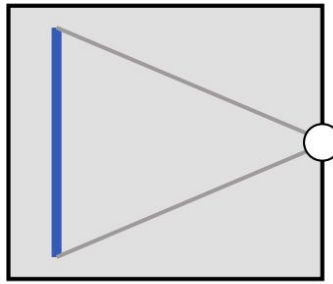
Volume Rendering

Abstract away light sources, objects.

For each point in space, need to know:

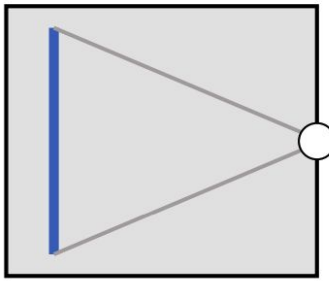
(1) How much light does it emit?

(2) How opaque is it? $\sigma \in [0,1]$



Stepping Back: Pinhole Camera Model

Abstract away light sources, objects.
For each point in space, need to know:
(1) How much light does it emit?
(2) How opaque is it? $\sigma \in [0,1]$

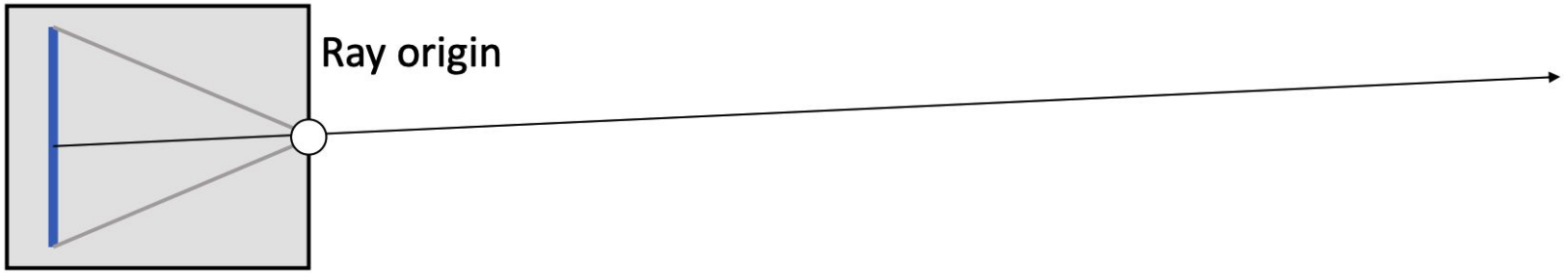


Point in empty space:
(1) Emits no light (black)
(2) Completely transparent $\sigma = 0$

Point on car:
(1) Emits red light in hemisphere
(2) Complete opaque $\sigma = 1$

Q: When $\sigma \in (0,1)$?

Volume Rendering



Parameterize each ray as origin plus direction: $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

Color that a point **p** emits in direction **d** is $c(\boldsymbol{p}, \boldsymbol{d}) \in [0,1]^3$

Q: why volume density is designed to be independent of the viewing direction, but color depends on viewing direction?

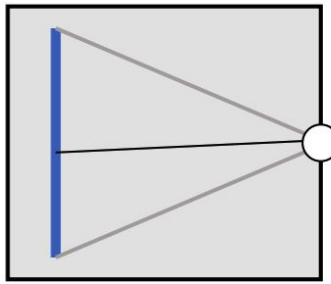
M | ROBOTICS

Volume Rendering

Kajiya, J.T., Herzen, B.P.V.: Ray tracing volume densities. Computer Graphics (SIGGRAPH) (1984)

Color observed by the camera given by **volume rendering equation**:

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t), \boldsymbol{d})dt$$



Ray origin

Near point: t_n

Current point: t

Far point: t_f

Parameterize each ray as origin plus direction: $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

Color that a point **p** emits in direction **d** is $c(\boldsymbol{p}, \boldsymbol{d}) \in [0,1]^3$

Transmittance: How much light from the current point will reach the camera?

Opacity: How opaque is the current point? $\sigma \in [0,1]$

Color: What color does the current point emit along the direction toward the camera?

M | ROBOTICS

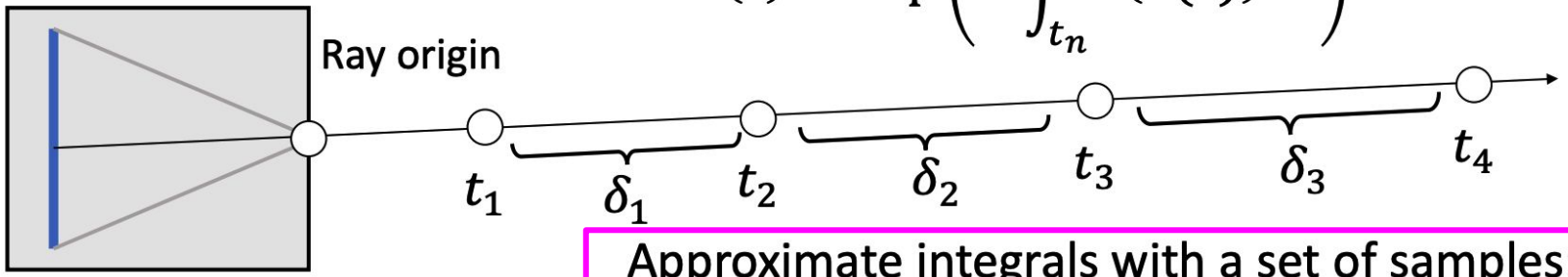
Transmittance

Compute transmittance by **accumulating volume density** up to current point

Color observed by the camera given by **volume rendering equation**:

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t), \boldsymbol{d})dt$$

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\boldsymbol{r}(s))ds\right)$$



Parameterize each ray as origin plus direction: $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

Color that a point **p** emits in direction **d** is $c(\boldsymbol{p}, \boldsymbol{d}) \in [0,1]^3$

Approximate integrals with a set of samples:

$$C(\boldsymbol{r}) \approx \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\,\boldsymbol{c}_i$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$$

Mildenhall et al, "Representing Scenes as Neural Radiance Fields for View Synthesis", ECCV 2020

NeRF

spatial location (x, y, z)

Train a neural network to input position **p** and direction **d**, output $\sigma(\boldsymbol{p})$ and $c(\boldsymbol{p}, \boldsymbol{d})$

viewing direction (θ, ϕ)

Training loss: Estimated pixel colors $C(\boldsymbol{r})$ should match actual pixel colors from images

Volume density

View-dependent RGB color

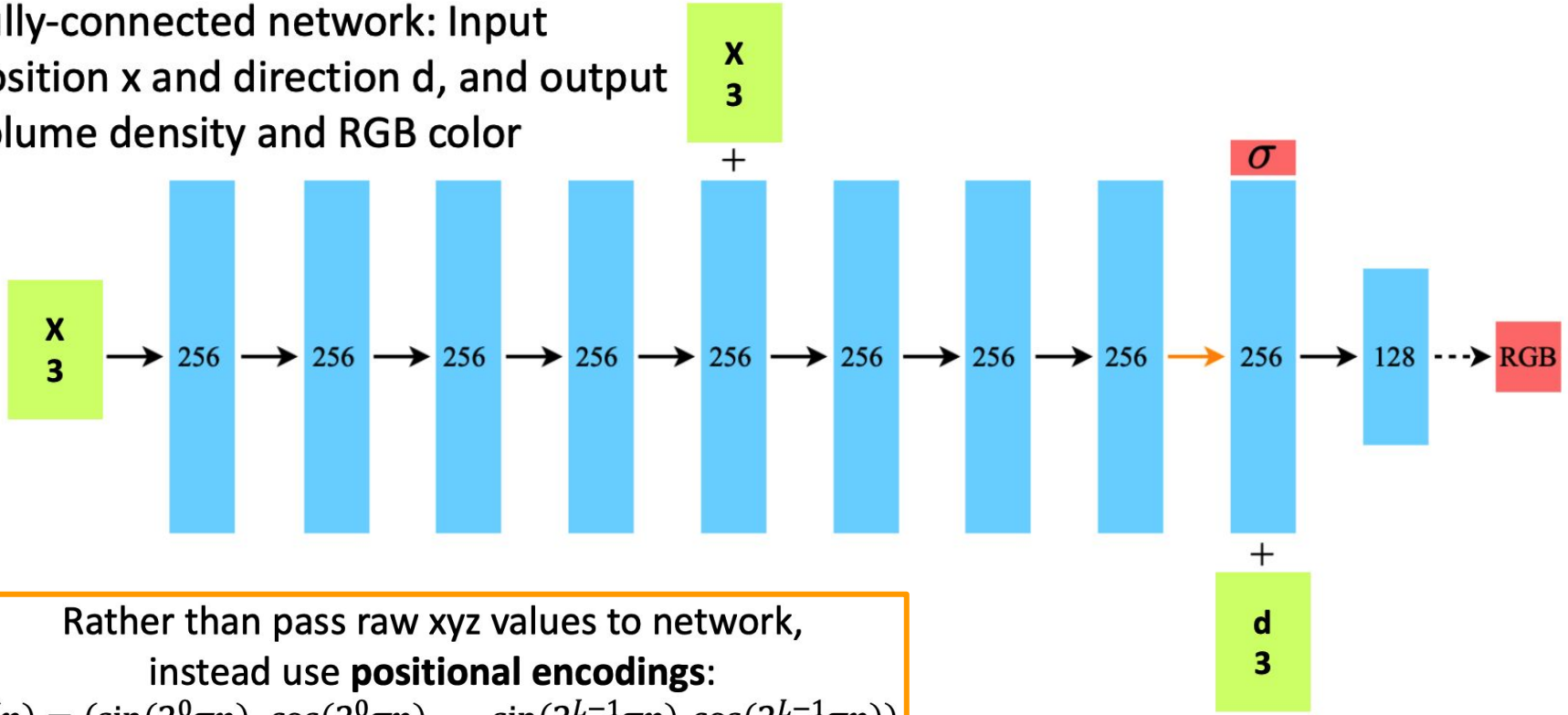
Squared error

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

M | ROBOTICS

NeRF: Network Architecture

Fully-connected network: Input position x and direction d, and output volume density and RGB color



Rather than pass raw xyz values to network, instead use **positional encodings**:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$$

NeRF: Network Architecture

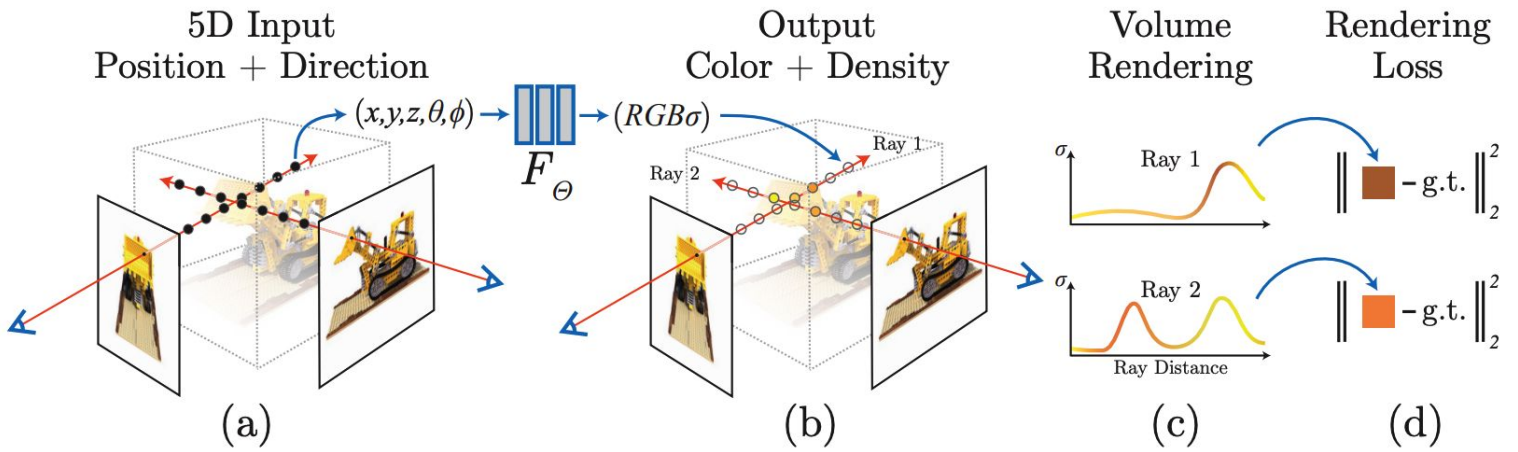


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

ROBOTICS

NeRF

https://www.matthewtancik.com/nerf

ROBOTICS

NeRF (2020)

Main Problem: Very slow!

Also, not generalizable

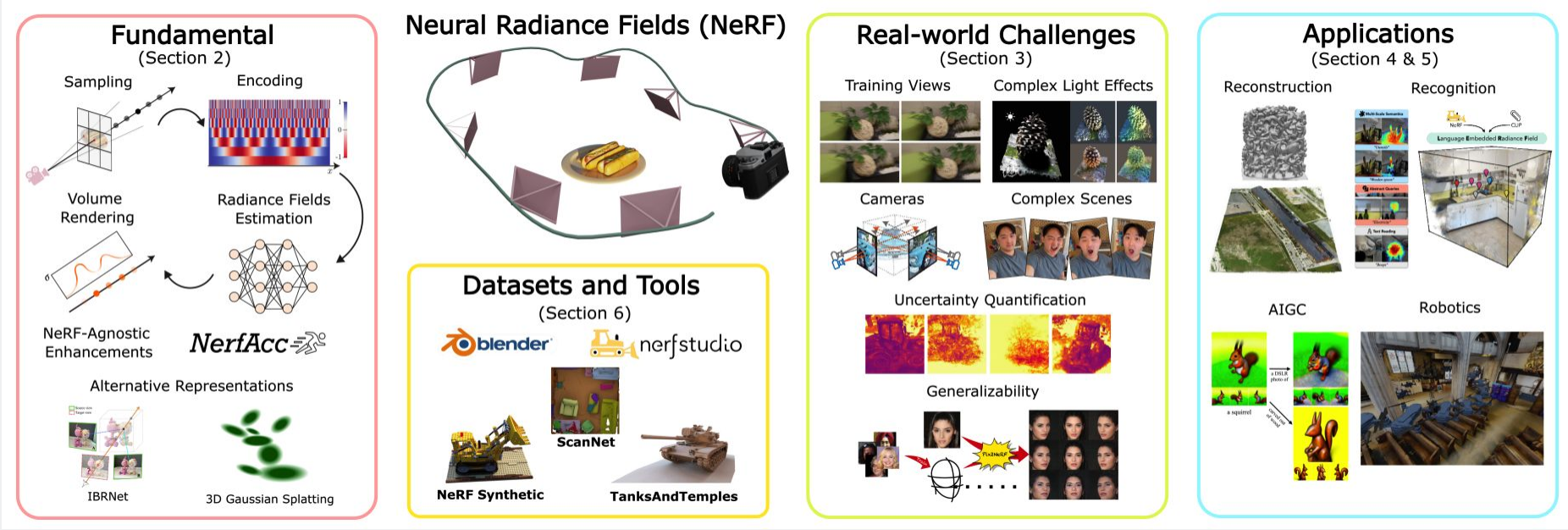
Training: 1-2 days on a V100 GPU, for just a single scene!

Inference: Sampling an image from a trained model:
(256 x 256 pixels) x (224 samples per pixel)
= 14.6M forward passes through MLP

Tons of follow-up work!

Cited by 11521

More on NeRF

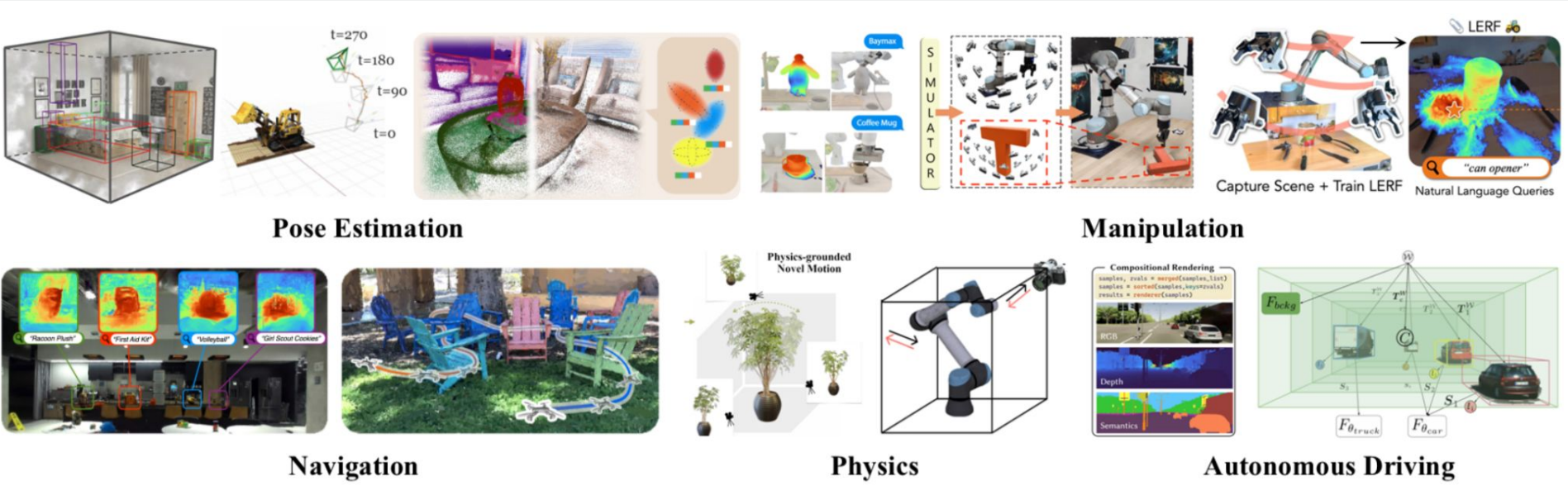


Neural radiance field for the real world: a survey https://arxiv.org/pdf/2501.13104v1
Radiance Fields https://radiancefields.com/
Nerf and 3DGS at CVPR 2024 https://github.com/Yubel426/NeRF-3DGS-at-CVPR-2024
RS-Nerf (rolling shutter), ECCV 2024 https://arxiv.org/pdf/2407.10267

M | ROBOTICS

More on NeRF

Mip-NeRF360 https://jonbarron.info/mipnerf360/ (unbounded scenes)
Block-NeRF https://waymo.com/research/block-nerf/
NeRF in Robotics https://robonerf.github.io/survey/index.html
LeRF https://www.lerf.io/

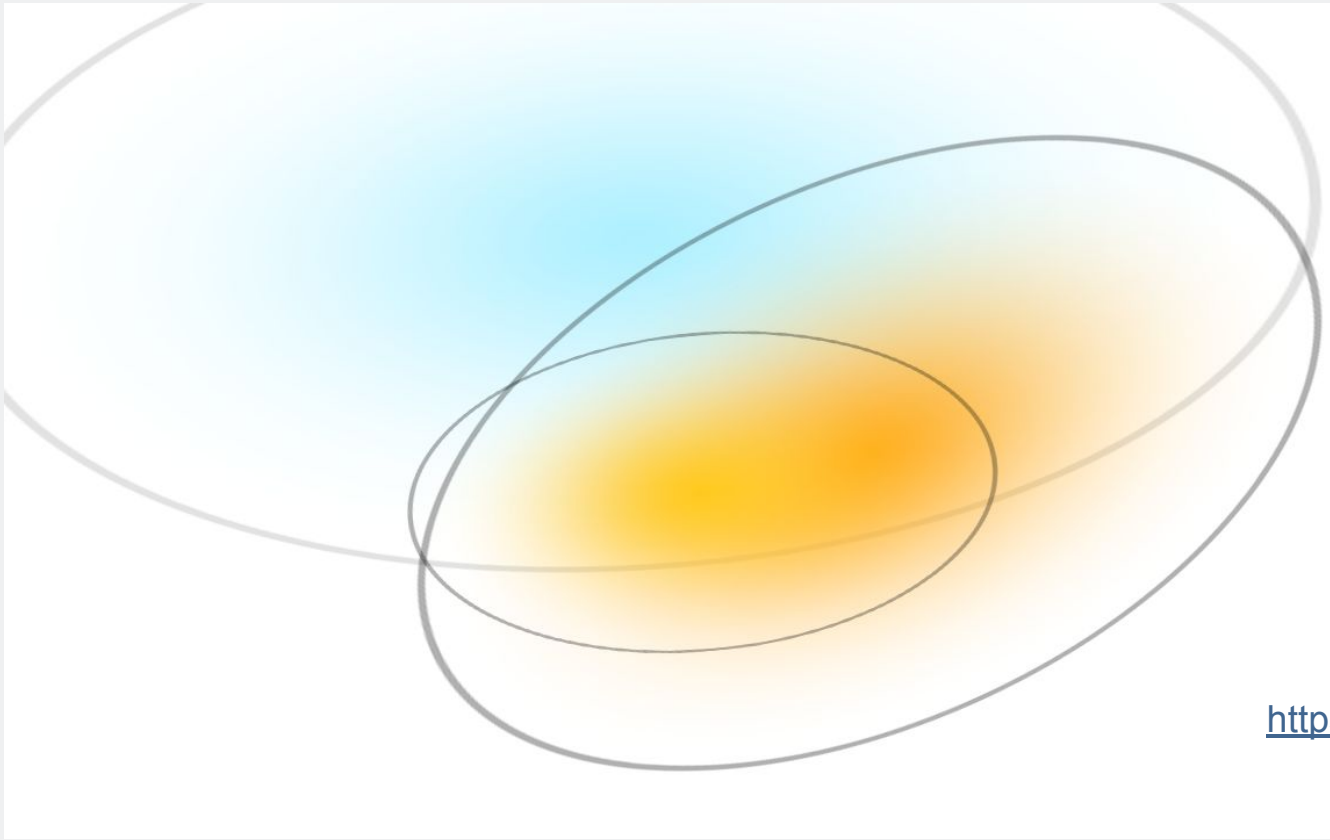


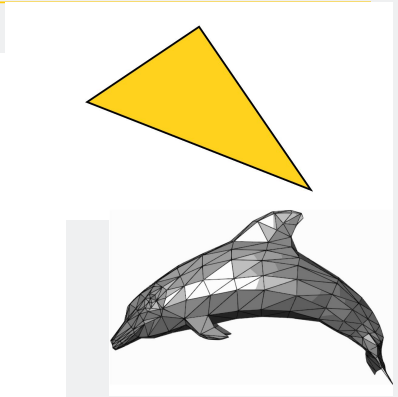
Gaussian Splatting

https://arxiv.org/pdf/2308.04079

Gaussian Splatting (2023)

Recall: Triangle Meshes





https://arxiv.org/pdf/2308.04079

M | ROBOTICS

Gaussian Splatting

M | ROBOTICS

Review: SfM

Input: a set of images
Output: image projections for each point (**camera parameters**) + structure/scene
("geometrically verified images")

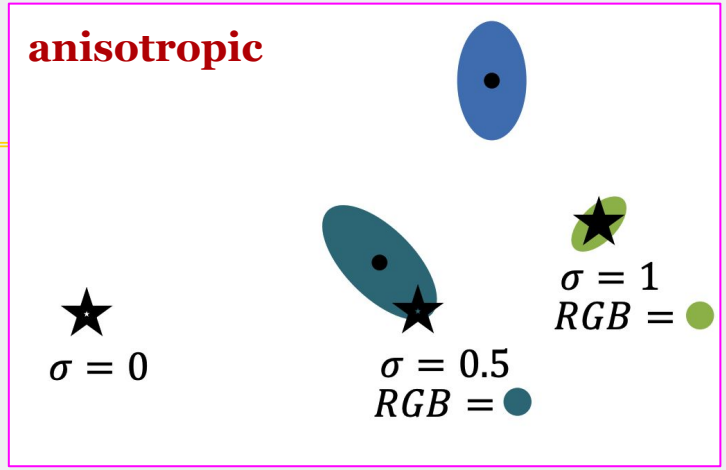
Structure from Motion

Figure 1. Result of Rome with 21K registered out of 75K images.

https://openaccess.thecvf.com/content_cvpr_2016/papers/Schonberger_Structure-From-Motion_Revisited_CVPR_2016_paper.pdf

M | ROBOTICS

Gaussian Splatting



Input:

- a set of images of static scenes
- camera parameters (calibrated by SfM)

➡ 3D Gaussians (sparse)

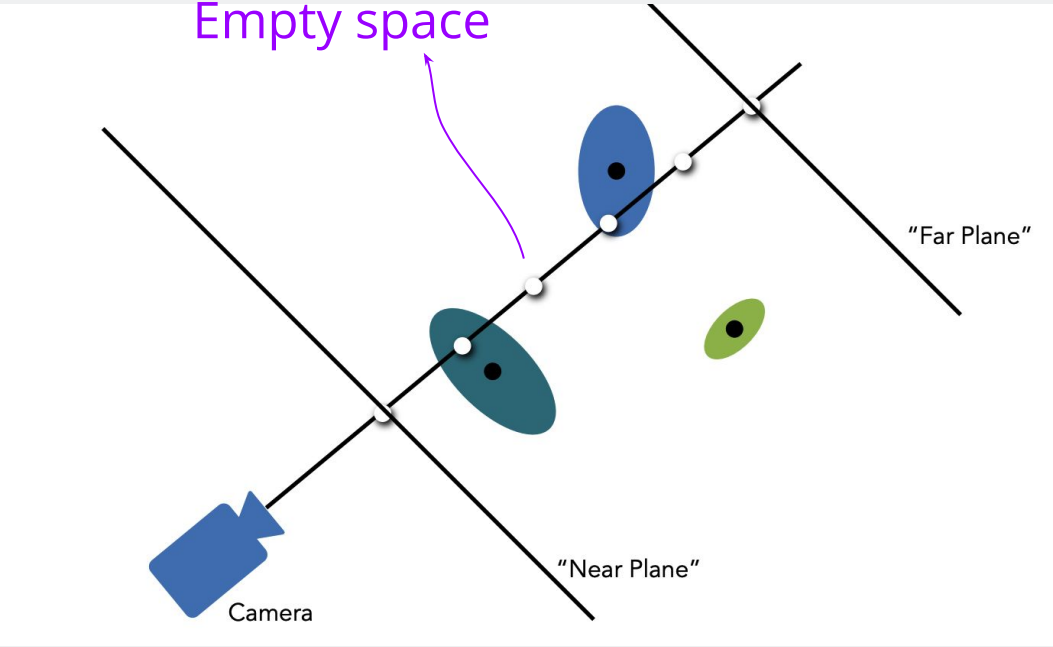
a position (mean), covariance matrix and opacity α

spherical harmonic (SH) coefficients (color)

https://www.scenerepresentations.org/courses/2023/fall/inverse-graphics/

M | ROBOTICS

Gaussian Splatting

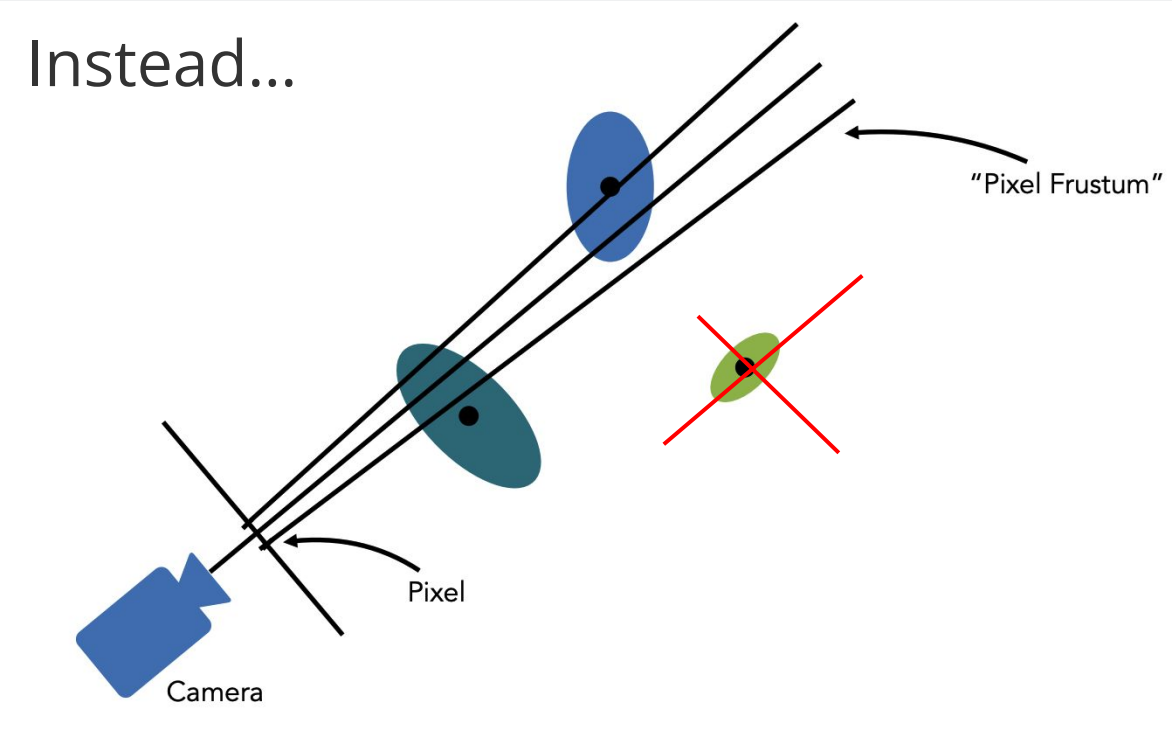


Similar to NeRF, Same volume rendering integral

BUT - let's NOT sample a lot of empty spaces...

https://www.scenerepresentations.org/courses/2023/fall/inverse-graphics/

Gaussian Splatting



- Splitting the screen into 16×16 tiles
- "Cull" Gaussians against the view frustum and each tile
- Only keep Gaussians with a 99% confidence interval intersecting the view frustum

Additionally, adaptive Gaussian densification

https://www.scenerepresentations.org/courses/2023/fall/inverse-graphics/
https://web.stanford.edu/class/cs231a/lectures/lecture17_gaussian_splatting.pdf

M | ROBOTICS

3DGS

https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

high-quality real-time (≥ 100 fps) novel-view synthesis at 1080p resolution.

ROBOTICS

More on Splat

3DGS tutorial https://3dgstutorial.github.io/

3D reconstruction https://arxiv.org/pdf/2503.17316 (Pow3R, Dust3R)

4D Gaussian Splatting (March 31, 2025)
https://arxiv.org/pdf/2503.22159

Segmentation (March 28, 2025)
https://vulab-ai.github.io/Segment-then-Splat/
https://arxiv.org/pdf/2503.22268 Segment any Motion (CVPR 2025)

Generative Models

Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Data: x

Label: y

Cat

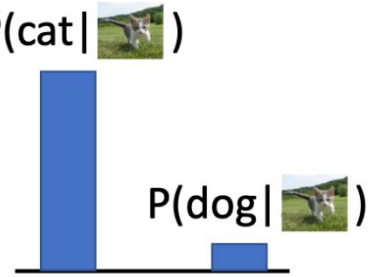
M | ROBOTICS

Discriminative vs Generative Models

classifiers

Discriminative Model: Learn a probability distribution p(y|x)

Data: x



Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Density Function
p(x) assigns a positive number to each possible x; higher numbers mean x is more likely

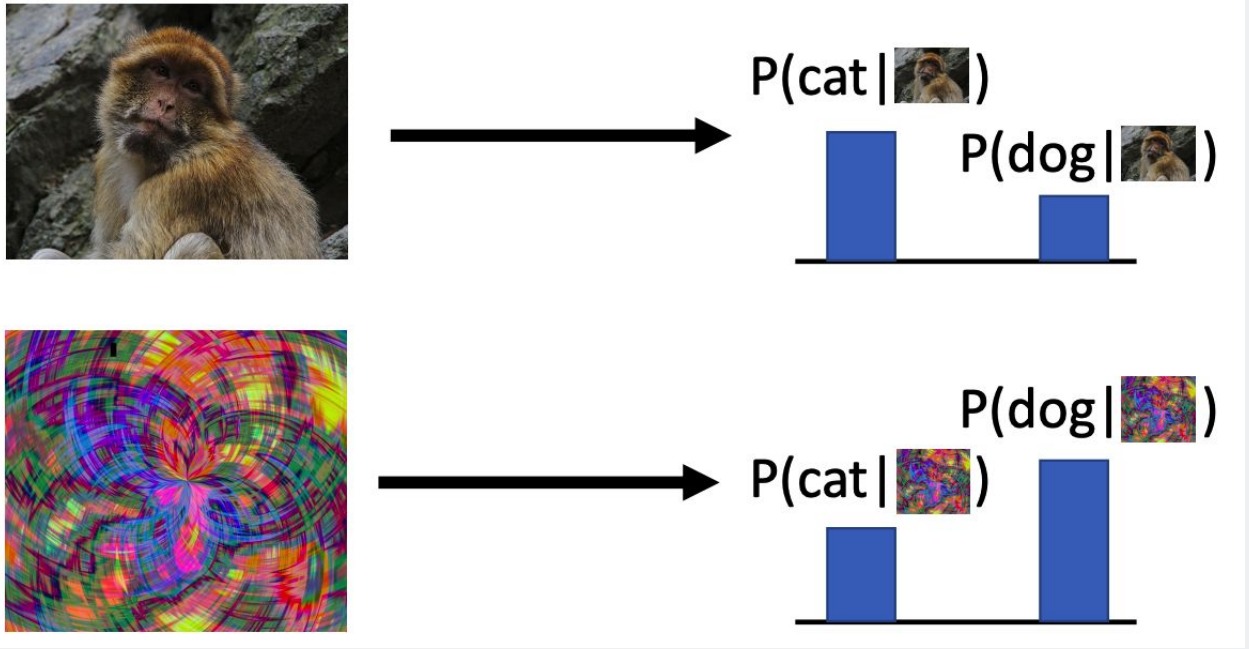
Density functions are **normalized**:

$$\int_X p(x)dx = 1$$

Different values of x **compete** for density

Discriminative vs Generative Models

Note about discriminative model:



Usually,

- No way for the model to handle unreasonable inputs;
- it must give label distributions for all images
- No “competition” between images

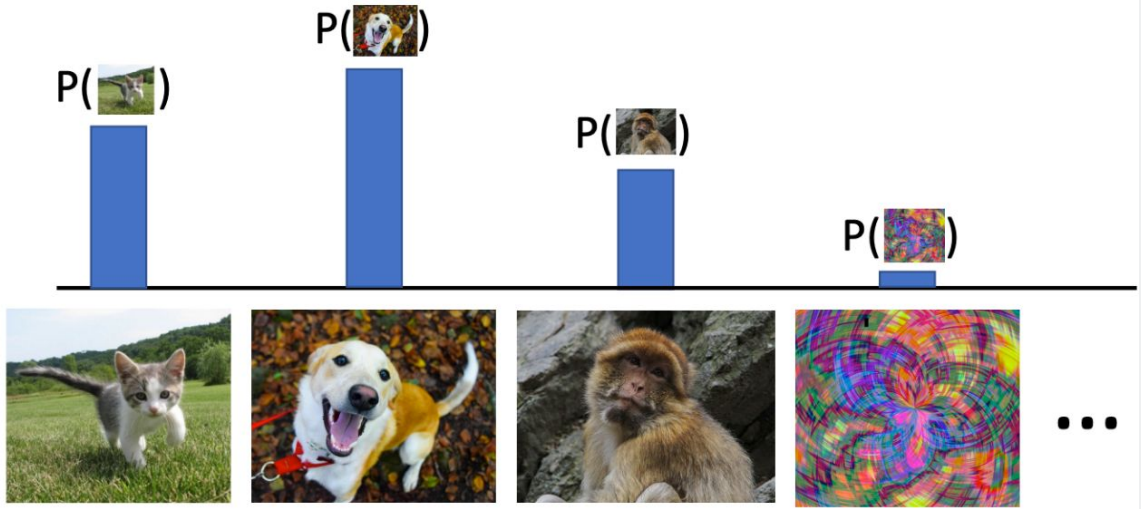
M | ROBOTICS

Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



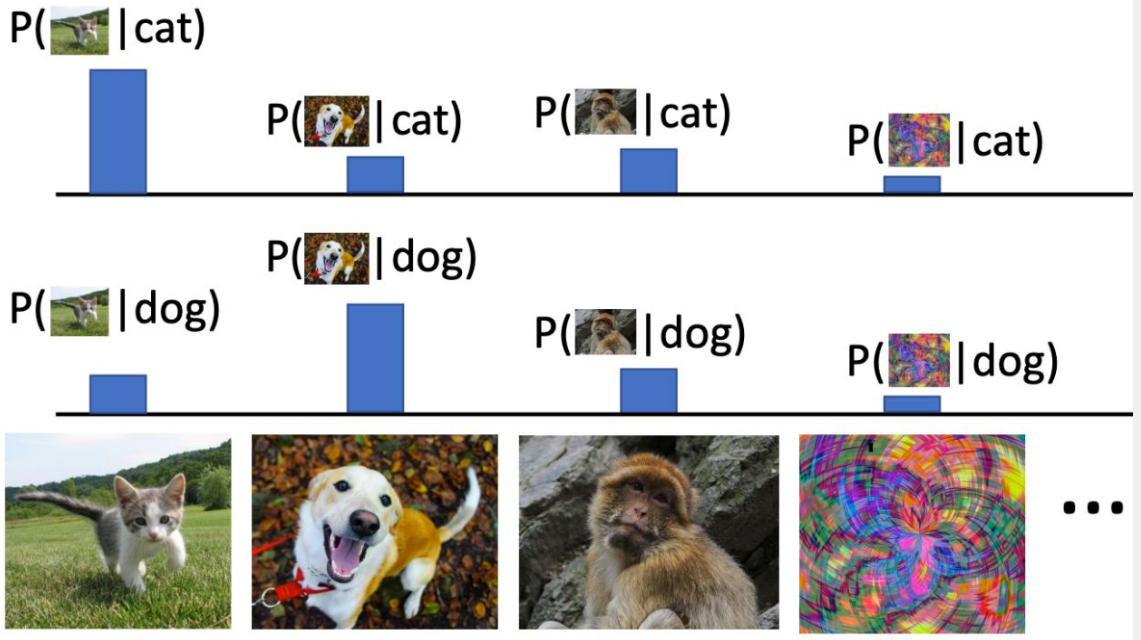
Generative model: All possible images compete with each other for probability mass

Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

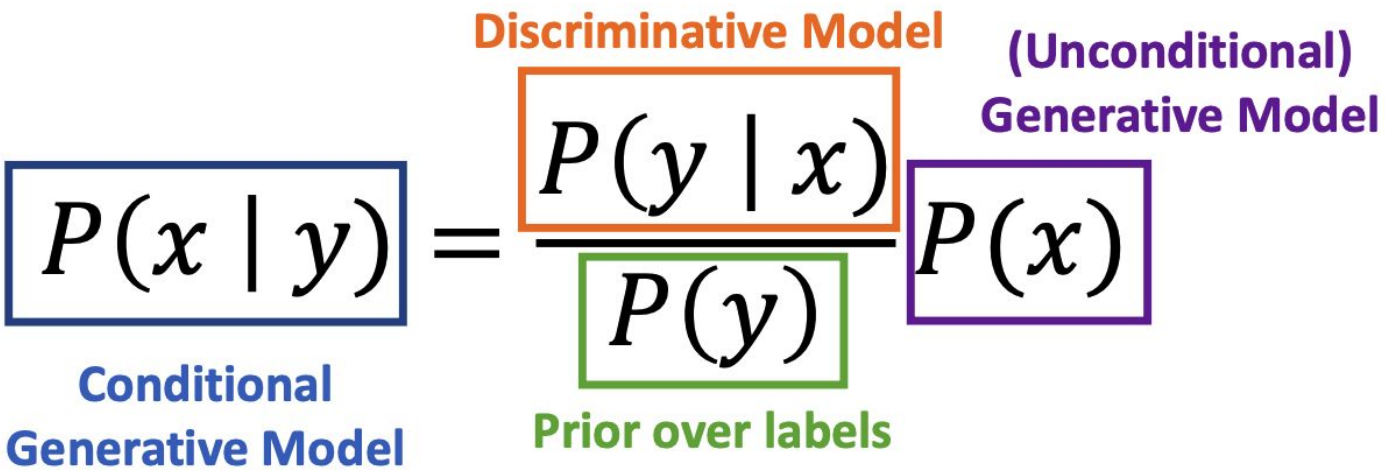
Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



Conditional Generative Model: Each possible label induces a competition among all images

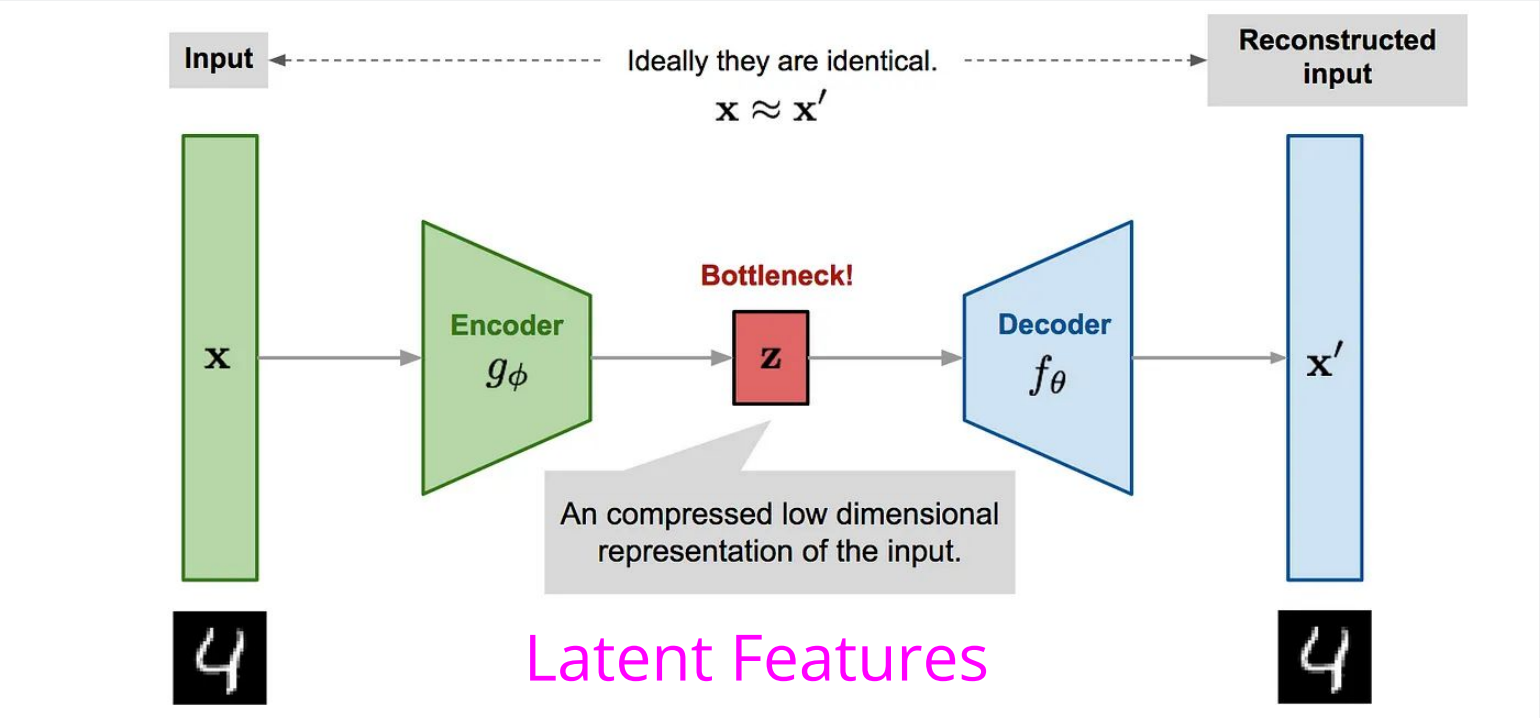
Recall: Bayes' Rule



We can build a conditional generative model from other components!

AutoEncoders

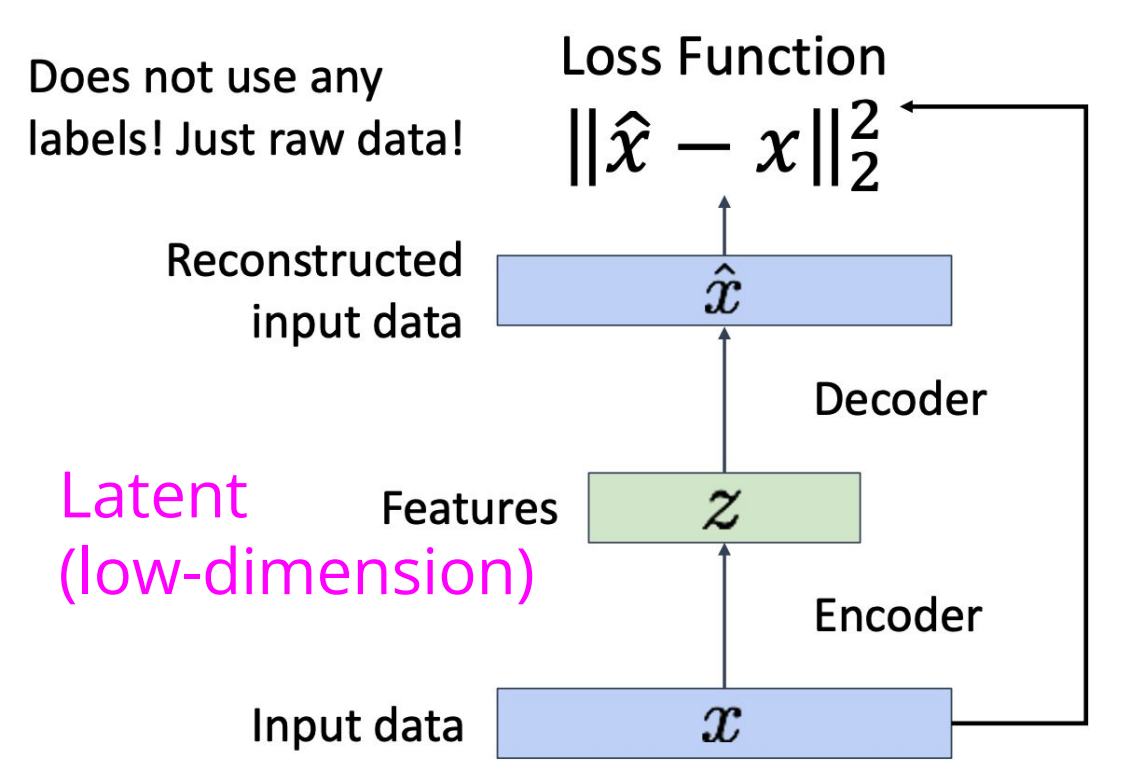
Learn from data **without** labels

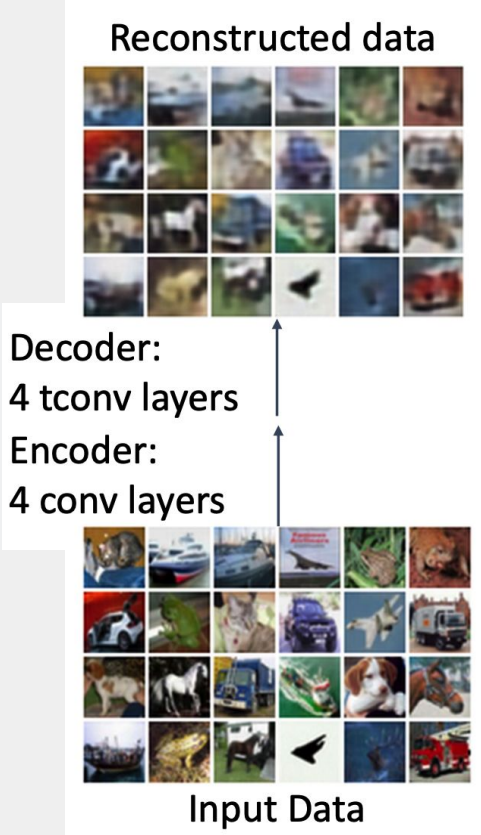


https://lilianweng.github.io/posts/2018-08-12-vae/

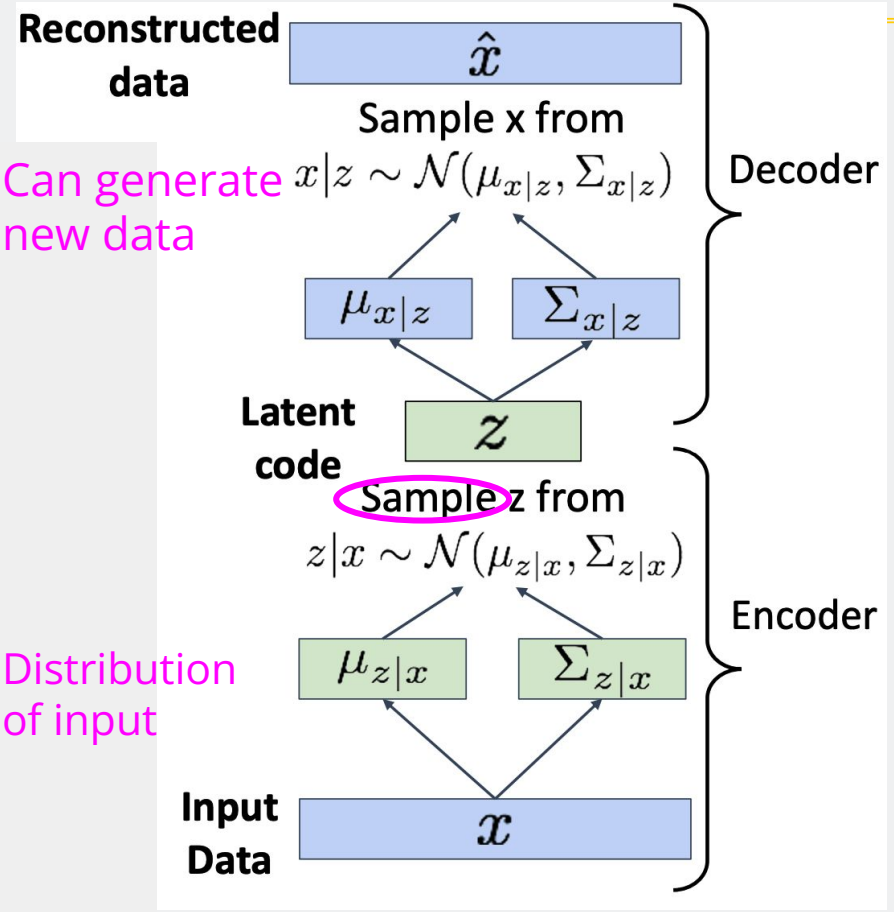
AutoEncoders

Learn from data **without** labels





VAE (Variational Autoencoders)



Can generate new data

Distribution of input

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

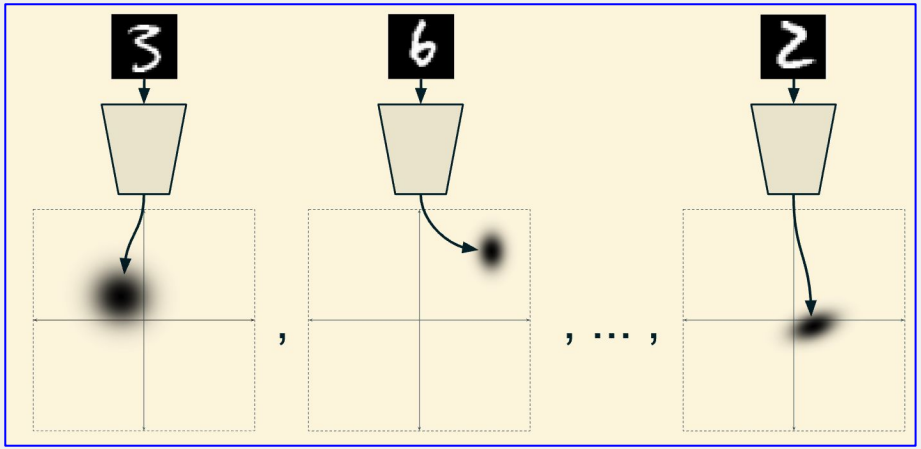
1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output
4. Run sampled code through **decoder** to get a distribution over data samples
5. **Original input data should be likely under the distribution output from (4)!**
6. Can sample a reconstruction from (4)

https://ijdykeman.github.io/ml/2016/12/21/cvae.html

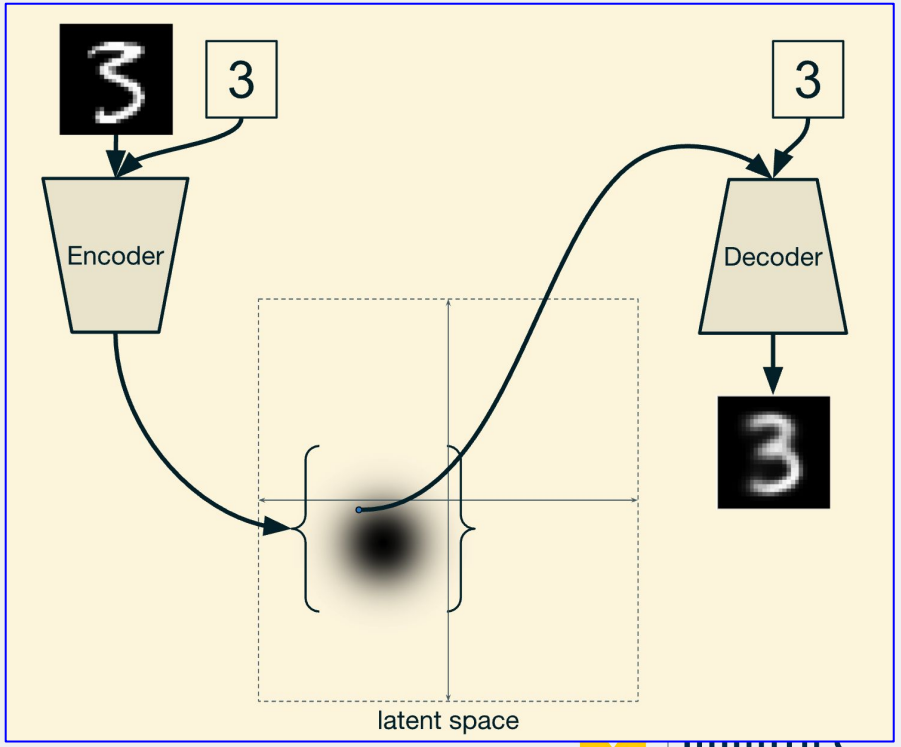
CVAE - labeled inputs

CVAE: Conditional VAE

VAE



CVAE - labeled inputs



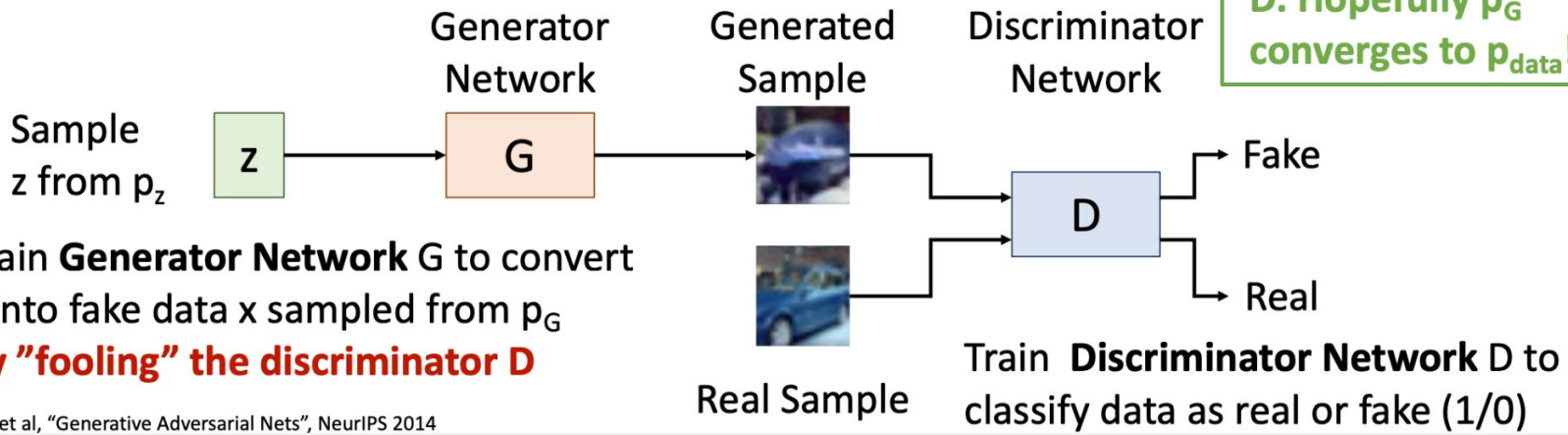
https://ijdykeman.github.io/ml/2016/12/21/cvae.html

GANs (Generative Adversarial Networks)

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

Idea: Introduce a latent variable z with simple prior p(z).
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)
Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!

Jointly train G and D. Hopefully p_G converges to p_{data}!



Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

https://papers.nips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf

M | ROBOTICS

GANs (Generative Adversarial Networks)

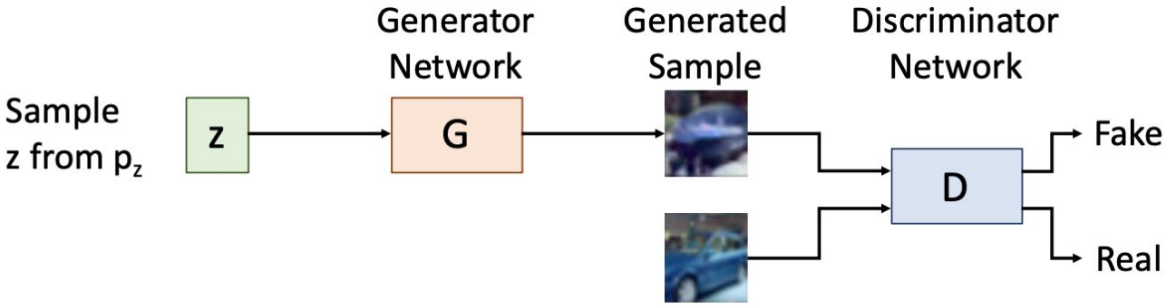
Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants
D(x) = 1 for real data

Discriminator wants
D(x) = 0 for fake data

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

Generator wants
D(x) = 1 for fake data



GANs (Generative Adversarial Networks)

Noise

Image

https://www.youtube.com/watch?v=IUi0REAWj2c

ROBOTICS

Stable Diffusion

Noise ➡ Data

v3.5, released Oct. 22, 2024
https://stability.ai/news/introducing-stable-diffusion-3-5

Stable diffusion v3 research paper
https://arxiv.org/pdf/2403.03206
https://stability.ai/news/stable-diffusion-3-research-paper

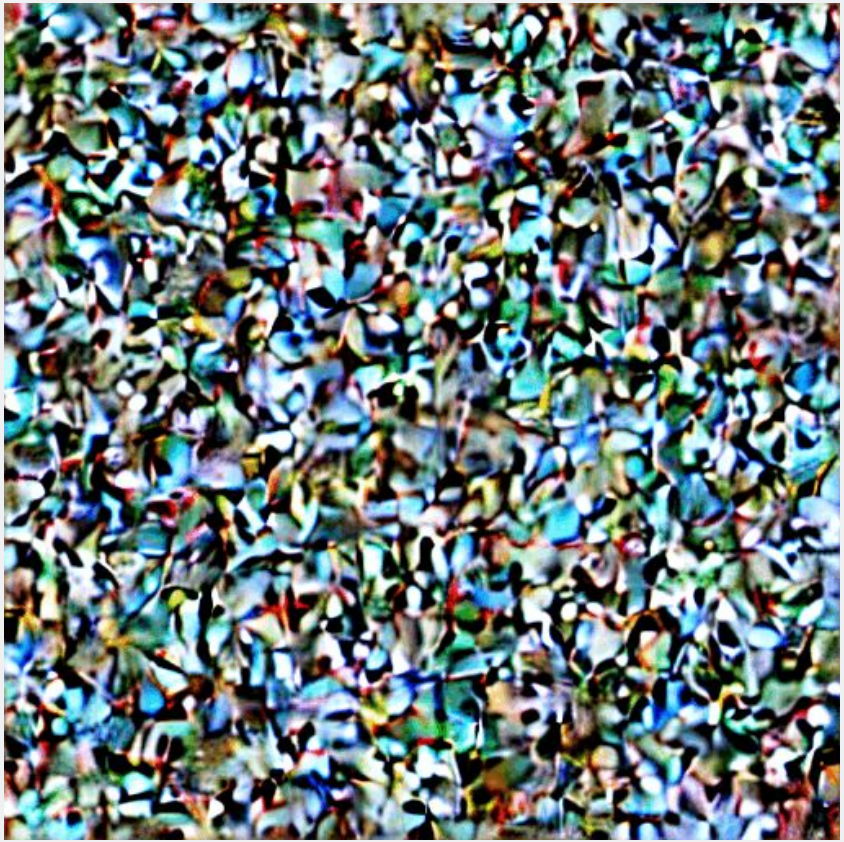
> “outperforms state-of-the-art text-to-image generation systems such as DALL·E 3, Midjourney v6, and Ideogram v1 in typography and prompt adherence, based on human preference evaluations.”

M | ROBOTICS

Stable Diffusion

Noise ➡ Data

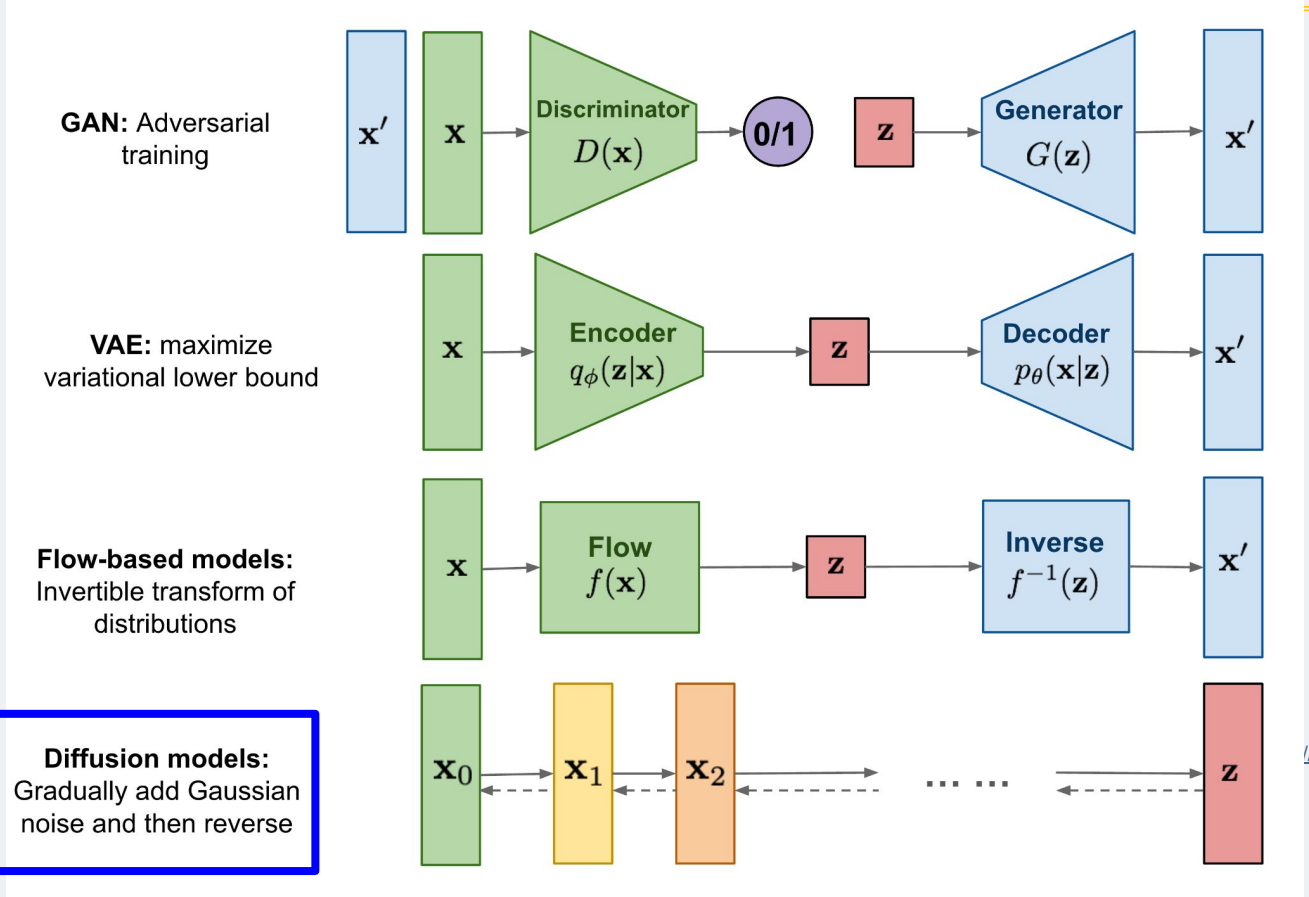
Also, text prompts -> images



https://scholar.harvard.edu/binxuw/classes/machine-learning-scratch/materials/stable-diffusion-scratch

M | ROBOTICS

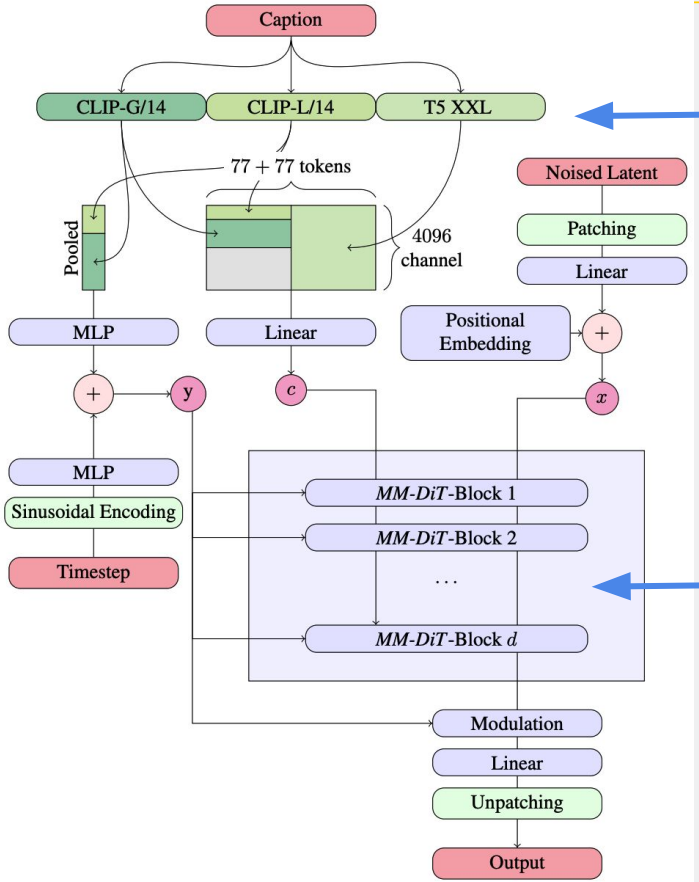
Diffusion Model



//lilianweng.github.io/posts/2021-07-11-diffusion-models/

M | ROBOTICS

Stable Diffusion



- Text/word representation
  e.g., CLIP (more on this)
- Understanding prompts

Transformer attention modules
Text ⟷ Image

https://arxiv.org/pdf/2403.03206

M | ROBOTICS

Diffusion Model

1. Forward diffusion: noising $x_0 \rightarrow x_1 \rightarrow \ldots x_T$

Take a data distribution $x_0 \sim p(x)$

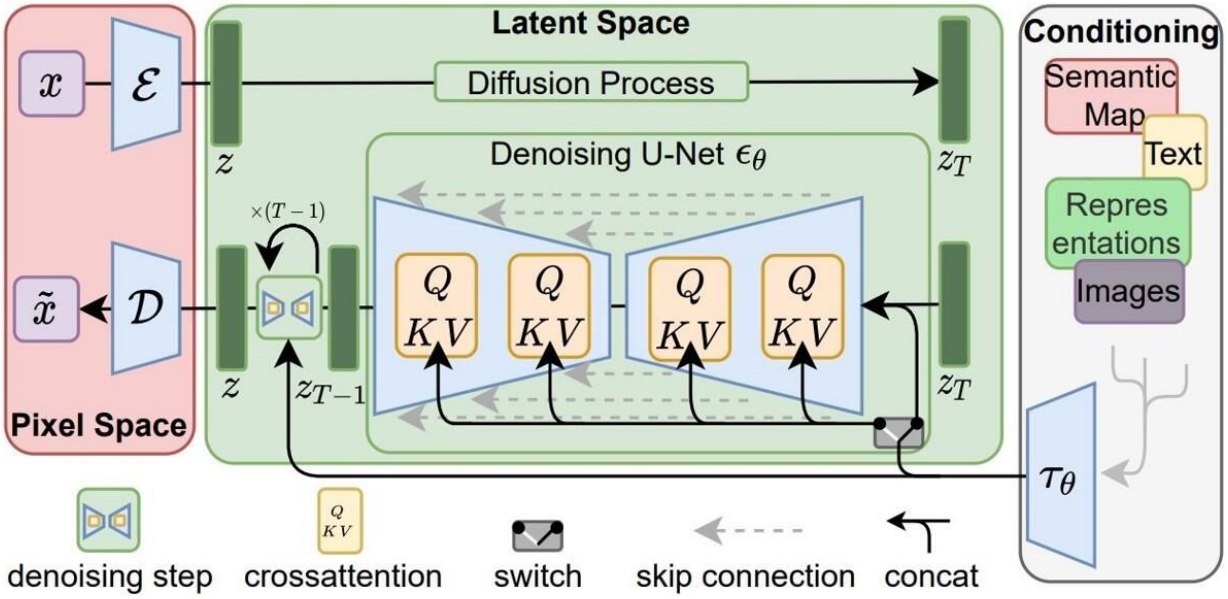
Turin it into noise by $x_T \sim N(0, \sigma^2 I)$

2. Reverse diffusion: denoising

Sample from the noise distribution, reverse the diffusion process to generate data

https://scholar.harvard.edu/files/binxuw/files/stable_diffusion_a_tutorial.pdf

Diffusion Model



- Principle of Diffusion models (sampling, learning)
- Diffusion for Images – e.g., UNet architecture
- Understanding prompts – Word as vectors, e.g, CLIP
- Let words modulate diffusion – Conditional Diffusion, Cross Attention
- Diffusion in latent space – AutoEncoderKL
- Training on Massive Dataset

https://scholar.harvard.edu/binxuw/classes/machine-learning-scratch/materials/stable-diffusion-scratch

M | ROBOTICS

SORA

https://openai.com/sora/

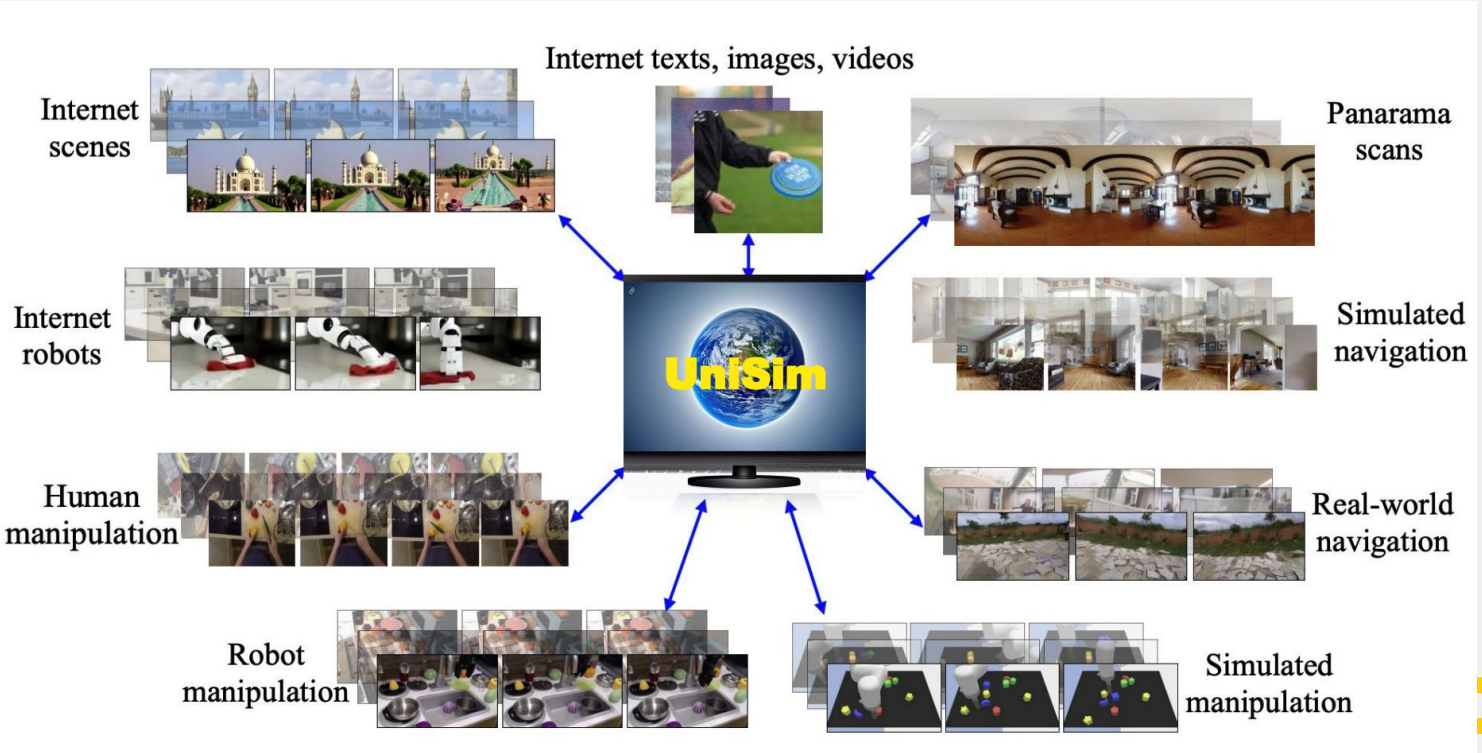
ROBOTICS

“Video generation models as world simulators”

https://openai.com/index/video-generation-models-as-world-simulators/

Not official technical report from openAI https://arxiv.org/pdf/2402.17177

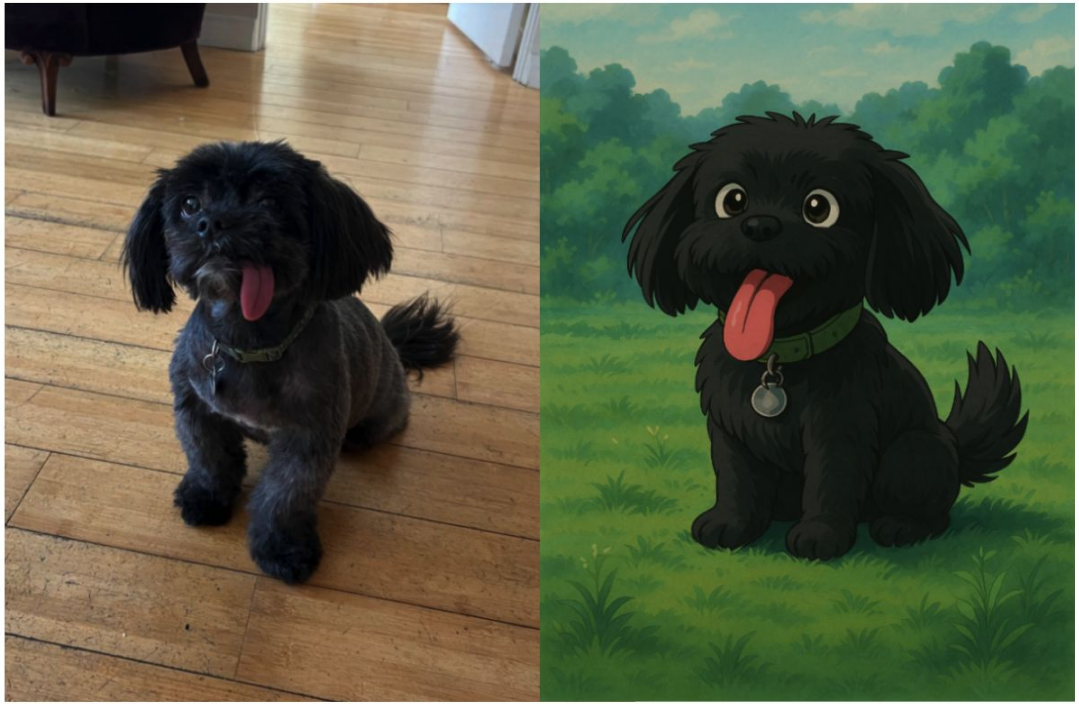
Uni-Sim https://universal-simulator.github.io/unisim/ (ICLR 2024) https://arxiv.org/pdf/2310.06114



“Action-Rich” Simulations

M | ROBOTICS

Image Generation



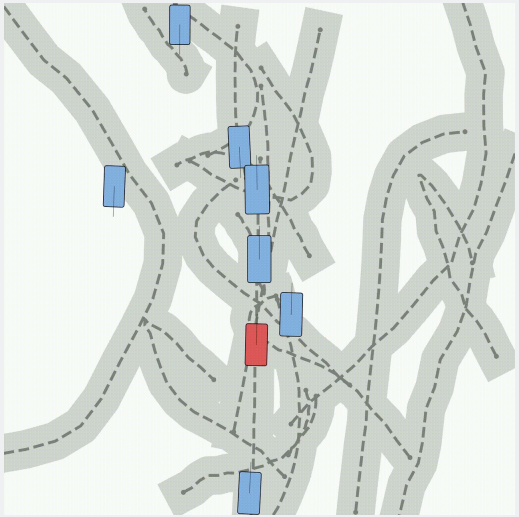
A REAL DOG (LEFT) AND AN AI-GENERATED IMAGE OF A DOG CHATGPT MADE IN STUDIO GHIBLI STYLE (RIGHT).

IMAGE CREDITS:MAXWELL ZEFF/OPENAI

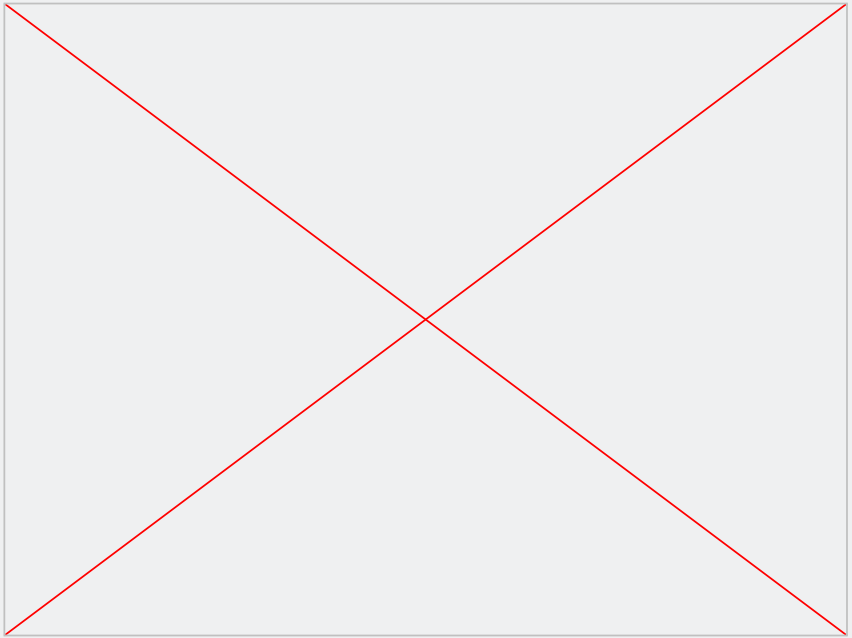
M | ROBOTICS

More on Diffusion

Scenario Dreamer https://princeton-computational-imaging.github.io/scenario-dreamer/ (CVPR 2025)



Stable virtual camera https://stable-virtual-camera.github.io/



Robotic Piano playing https://taco-group.github.io/PANDORA/

M | ROBOTICS

Summary

ROBOTICS