# CMSC740
# Advanced Computer Graphics

Matthias Zwicker

Fall 2024

# Generative shape models without 3D data

- Given: database of images
- Goal: mechanism that produces random new 3D objects that, when rendered to 2D images, “look just like images from the database”
- Approach: differentiable rendering in combination with generative model for images

Random noise vector → Neural 3D shape representation → Differentiable rendering → Output image

Training objective: generative model (GAN, diffusion model) to make output images look like images in database

# GAN-based geometry synthesis without 3D data

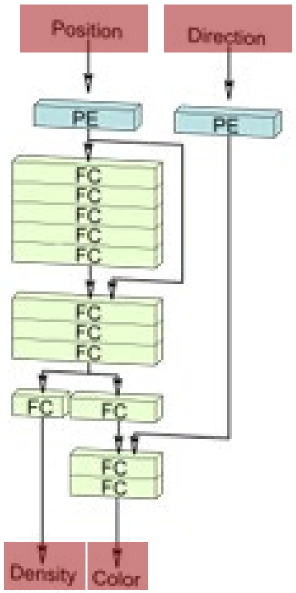
- "Efficient Geometry-aware 3D Generative Adversarial Networks", CVPR 2022
  https://github.com/NVlabs/eg3d
- Contributions
  - Hybrid voxel grid-implicit 3D representation
  - GAN-based training of generative shape model without 3D supervision

Face images rendered via generative 3D model
https://nvlabs.github.io/eg3d/

# Hybrid representation

- Goal: predict density, radiance at 3D locations as in NeRF

Original NeRF
"implicit" network (PE: positional encoding, FC: fully connected layer)



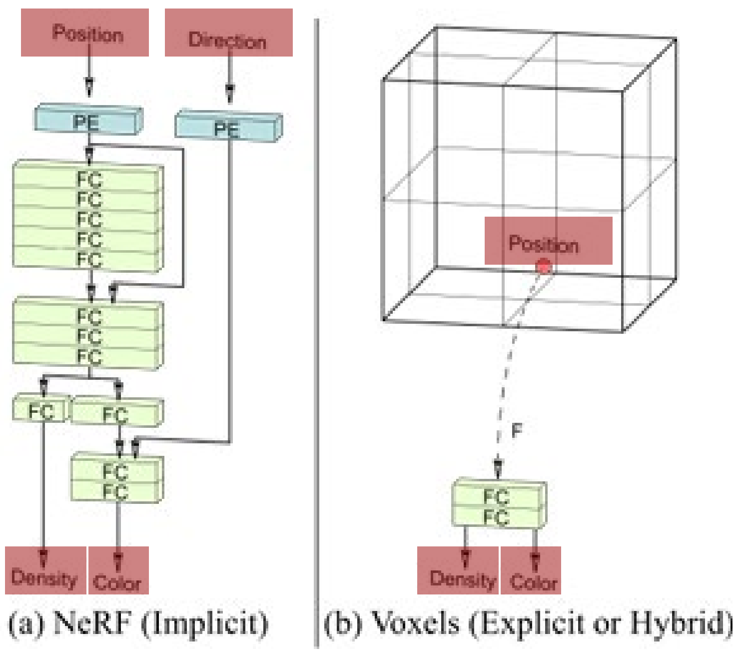

(a) NeRF (Implicit)

# Hybrid representation

- Goal: predict density, radiance at 3D locations as in NeRF

Original NeRF
"implicit" network (PE: positional encoding, FC: fully connected layer)

3D grid with learned features



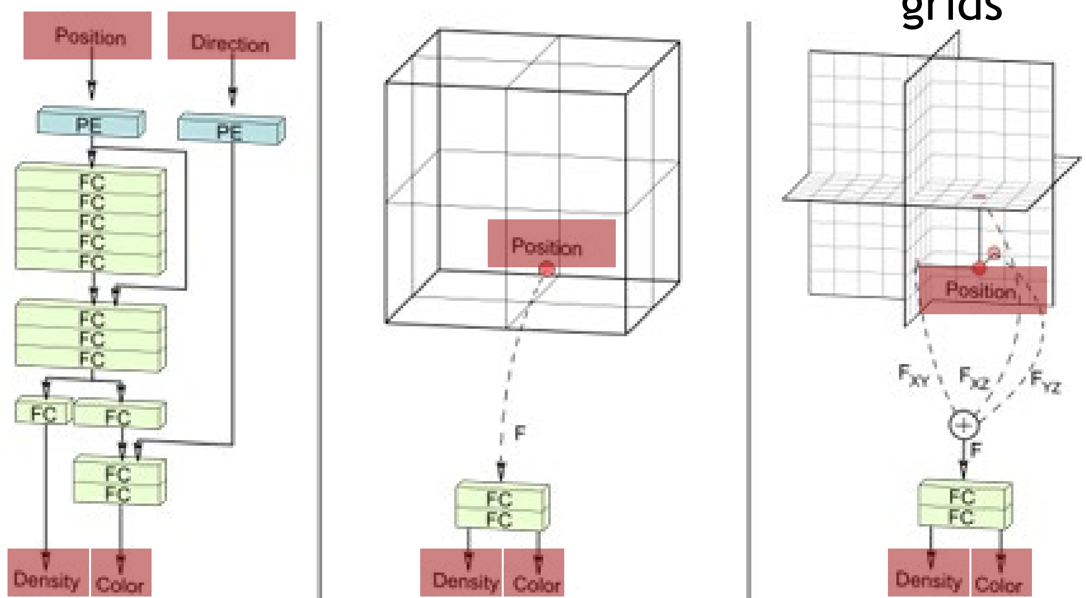
# Hybrid representation

- Goal: predict density, radiance at 3D locations as in NeRF
- Benefits of hybrid voxel grid-implicit representation
  - Can use small network, faster training
  - Avoid storage overhead of full 3D grid, enables higher resolutions

Original NeRF
“implicit” network (PE: positional encoding, FC: fully connected layer)

3D grid with learned features

Tri-planes with learned feature grids




(a) NeRF (Implicit) (b) Voxels (Explicit or Hybrid) (c) Ours (Hybrid)

Hybrid approach: Project 3D location onto tri-planes; look up and average three feature vectors; use as input for small neural network to predict density, color

# Comparison on scene reconstruction

- Comparison using same scene reconstruction problem as in original NeRF

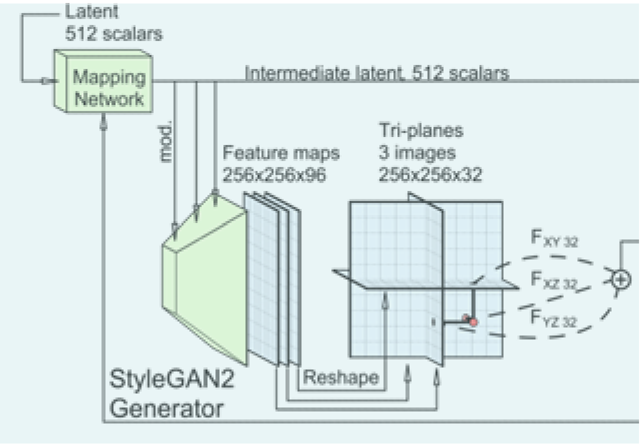

Scene reconstruction on Tanks & Temples dataset
https://www.tanksandtemples.org/

| | MLP | Rel. Speed ↑ | Rel. Mem. ↓ |
|---|---|---|---|
| Mip-NeRF [2] | $8 \times 256$ | $1\times$ | $1\times$ |
| Voxels (hybrid) | $4 \times 128$ | $3.5\times$ | $0.33\times$ |
| Tri-plane (SSO) | $4 \times 128$ | $2.9\times$ | $0.32\times$ |

Approx. 3x speedup and memory savings compared to original NeRF
MLP: nr of layers x nr of neurons per layer

# 3D GAN with differentiable rendering

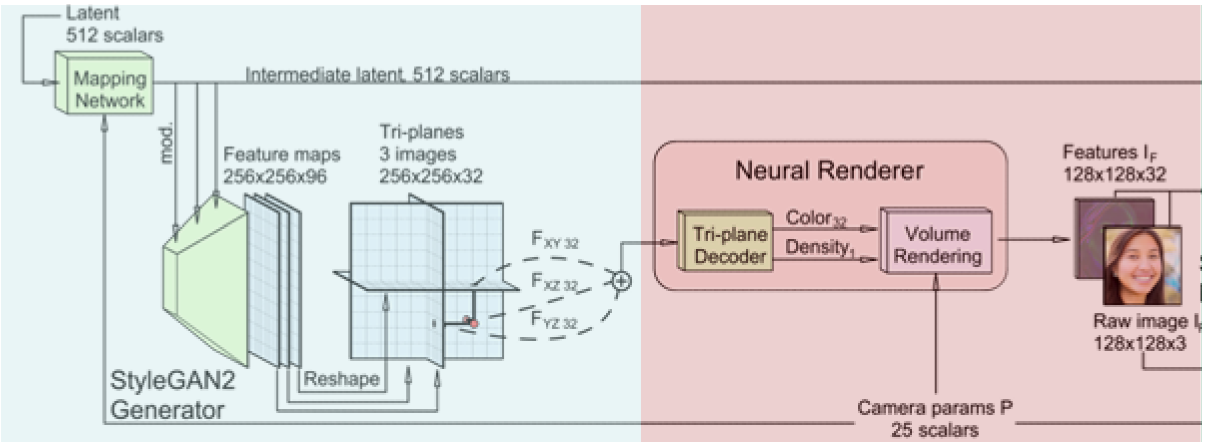
3D shape generator
input: noise (latent space)
output: tri-plane features



# 3D GAN with differentiable rendering

3D shape generator
input: noise (latent space)
output: tri-plane features

Differentiable NeRF rendering

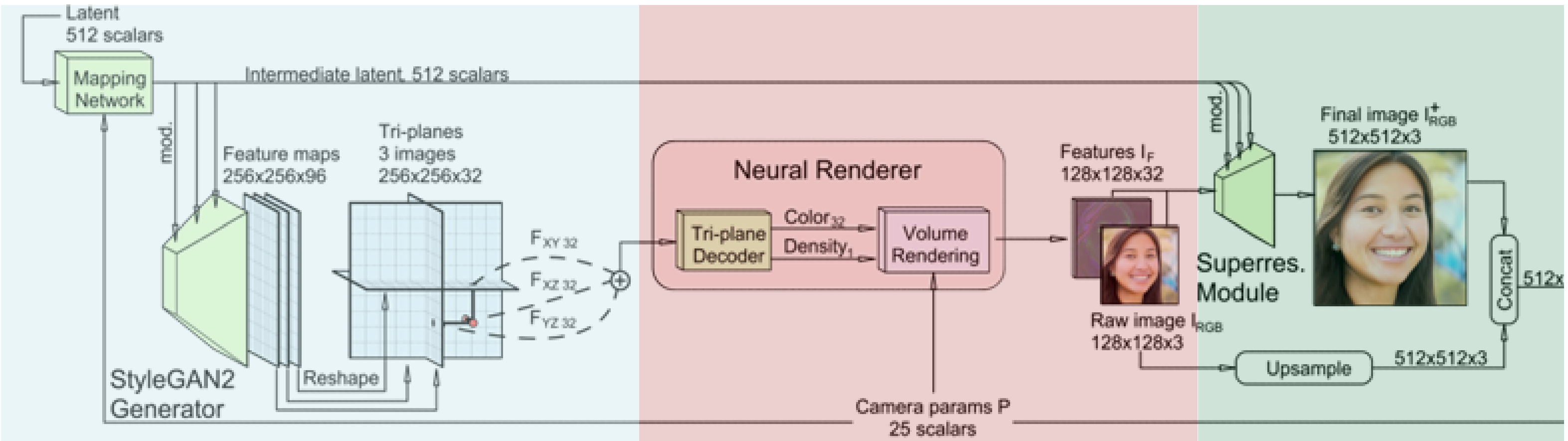


# 3D GAN with differentiable rendering

3D shape generator
input: noise (latent space)
output: tri-plane features

Differentiable NeRF rendering

Image superresolution



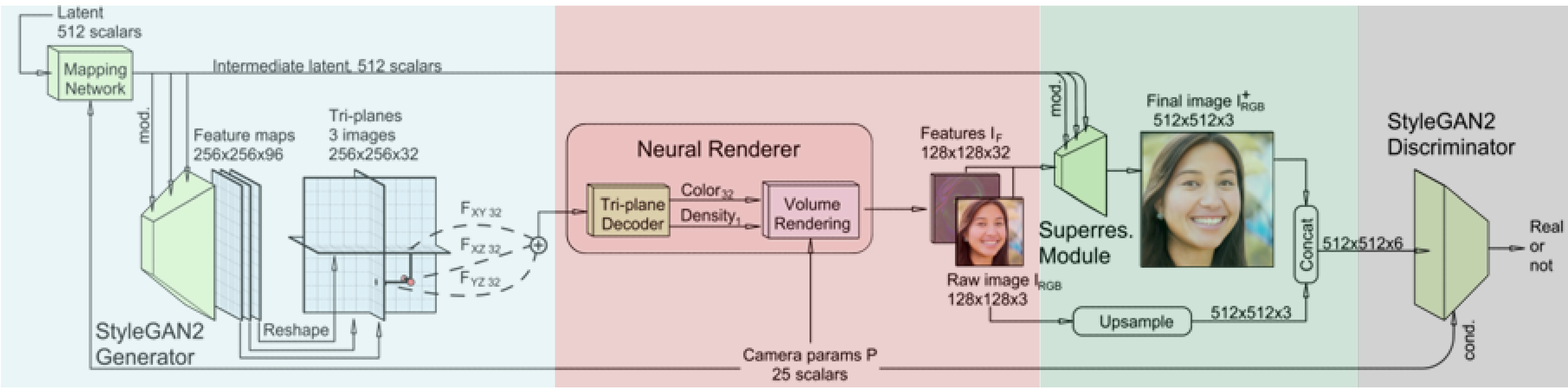
# 3D GAN with differentiable rendering

3D shape generator
input: noise (latent space)
output: tri-plane features

Differentiable NeRF rendering

Image superresolution

GAN discriminator

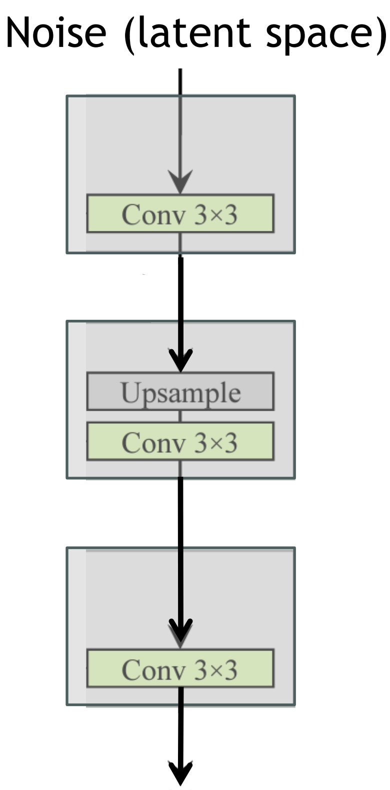



- Image-based GAN discriminator, no 3D training data necessary
  - Camera parameters for real images estimated with standard methods
  - Camera parameters used as input to conditional generator and discriminator, and for volume rendering

# Stylegan 2 generator

- Well-engineered architecture for GAN generators and discriminators to avoid empirically observed artifacts in previous methods
https://github.com/NVlabs/stylegan2

Basic convolutional generator

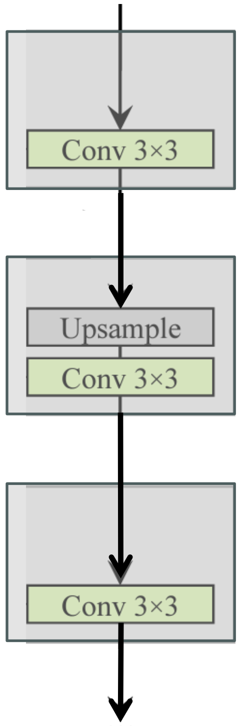


# Stylegan 2 generator

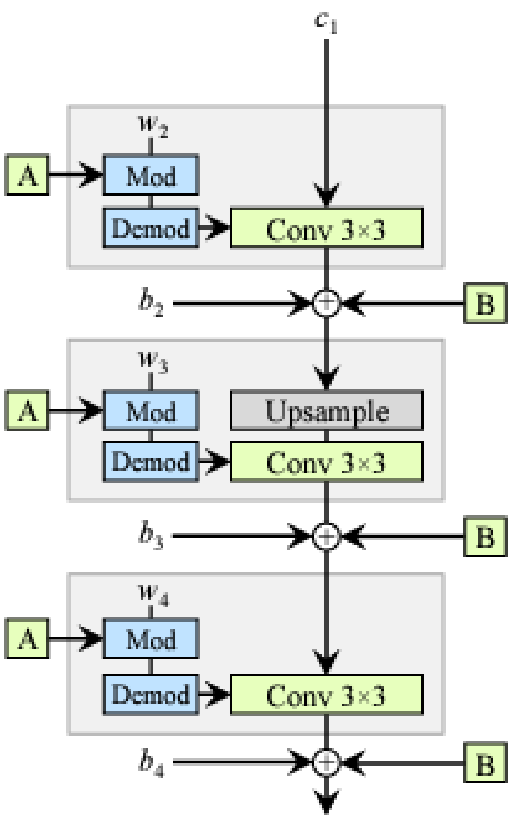
- Well-engineered architecture for GAN generators and discriminators to avoid empirically observed artifacts in previous methods
https://github.com/NVlabs/stylegan2

Basic convolutional generator

Noise (latent space)




Stylegan 2 generator



# 3D shape generator

- Stylegan 2 architecture ("works well")
  - Learned convolution weights $w_{ijk}$ (input channel $i$, output channel $j$, spatial location in convolution kernel $k$); learned biases $b$
- Latent noise vector (512-dim.) fed to convolutional layers via mapping network $A$
  - Provides scaling factors $s_i$ for input feature channel $i$
- Modulation

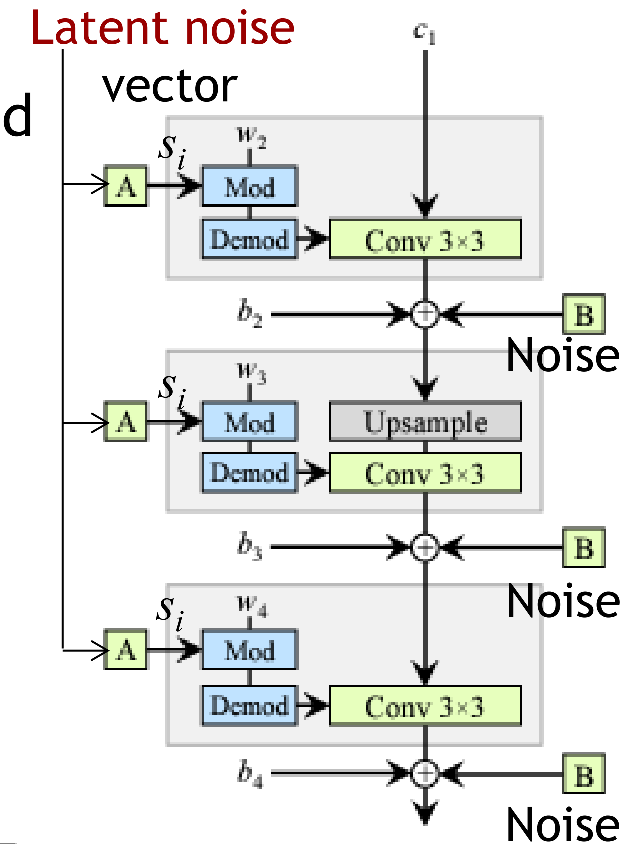
$$w'_{ijk} = s_i \cdot w_{ijk}$$

- Demodulation (small constant $\varepsilon$ to avoid division by zero)

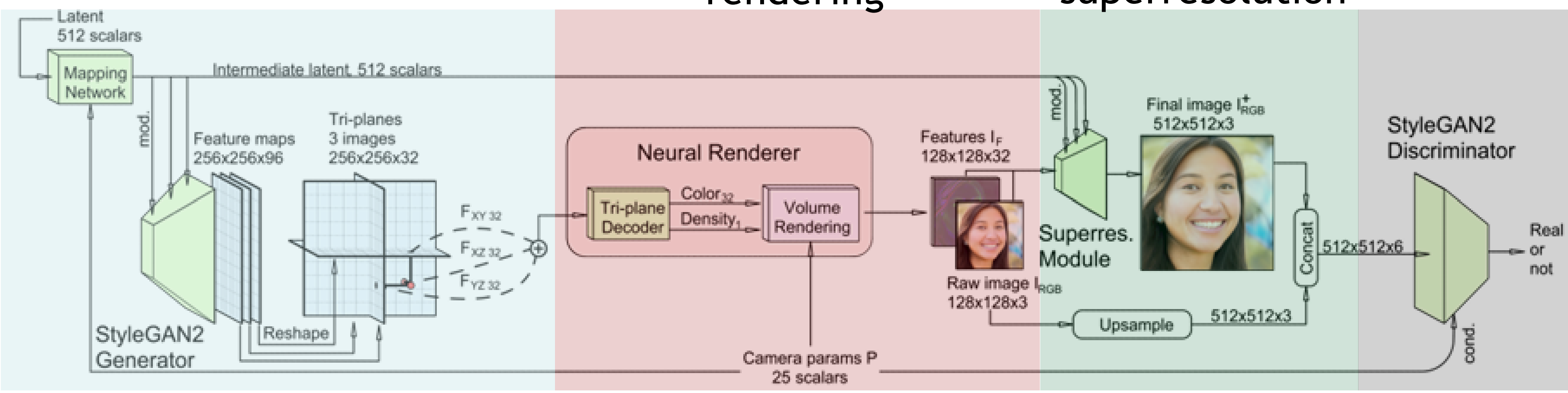
$$w''_{ijk} = w'_{ijk} \Big/ \sqrt{\sum_{i,k} {w'_{ijk}}^2 + \epsilon}$$

- 3D GAN: 256x256x96 output split into tri-planes at 256x256x32

Stylegan 2 convolution layers

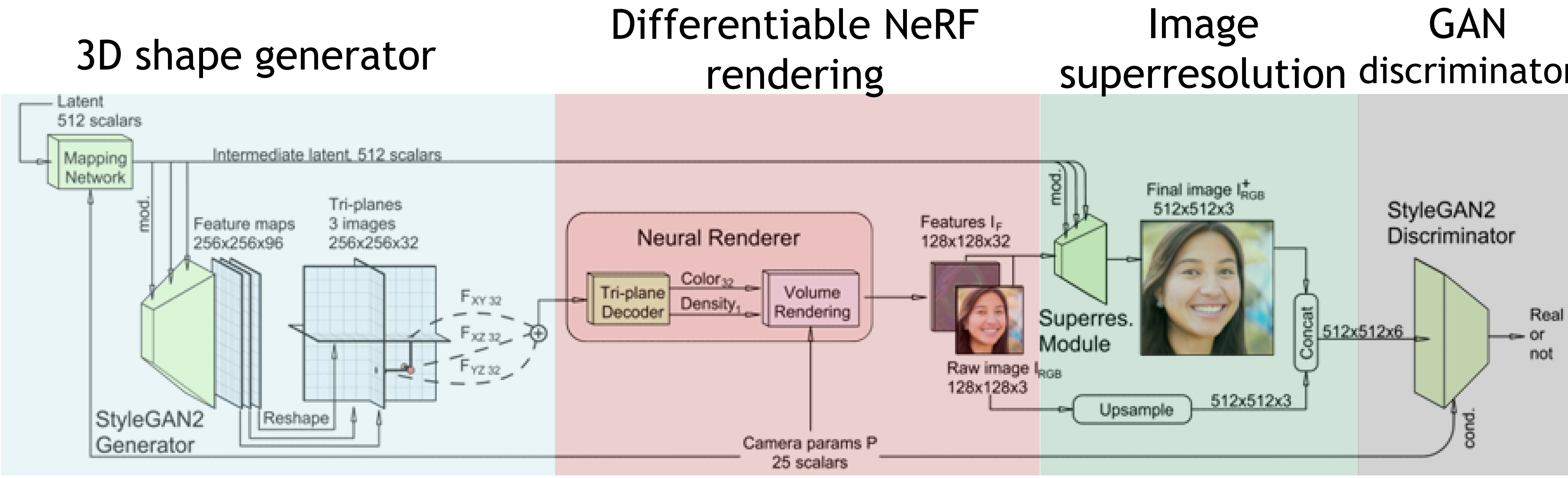


# Differentiable NeRF rendering




- Produces 32 feature channels, 3 interpreted as RGB colors and fed to discriminator

# Superresolution network




- Upsampling from 128x128x32 to 512x512x3
- Using Stylegan 2 convolutional layers, same mapping network for modulation

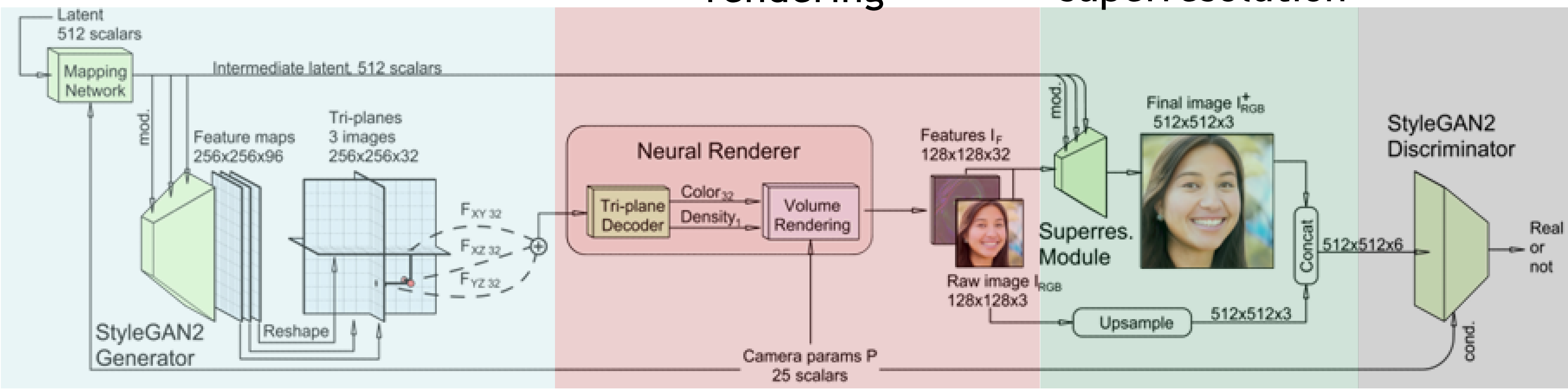
# Discriminator

3D shape generator

Differentiable NeRF rendering

Image superresolution

GAN discriminator




- Convolutional layers
- "Dual discriminator" using final image $I^*_{RGB}$ and upscaled 128x128 image, both at 512x512x3, concatenated to 512x512x6
- Conditional discriminator using camera parameters

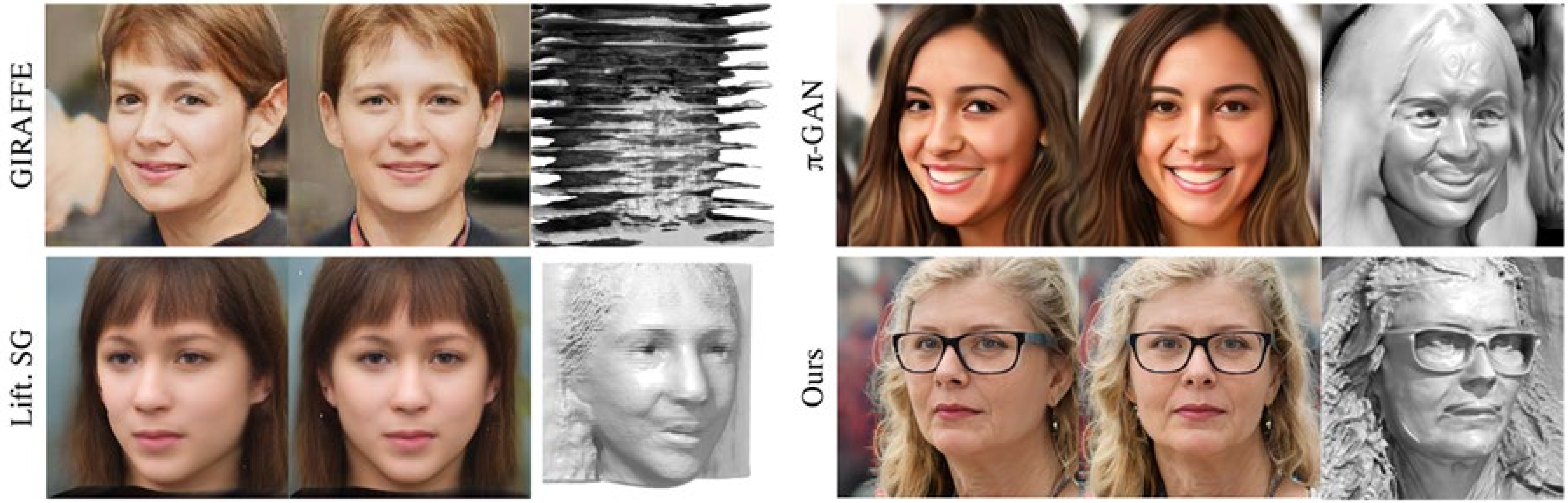
# Results

Trained with FFHQ and AFHQv2 Cats

https://github.com/NVlabs/ffhq-dataset

https://www.v7labs.com/open-datasets/afhq

# Comparisons




GIRAFFE, CVPR 2021 best paper https://m-niemeyer.github.io/project-pages/giraffe/index.html

pi-GAN, CVPR 2021 https://marcoamonteiro.github.io/pi-GAN-website/

Lifting Stylegan https://github.com/seasonSH/LiftedGAN

# Quantitative evaluation

- Challenge: quantitative evaluation of generative models requires comparison of generated and true densities
  - Difficult because of high dimensionality of data (images, video, etc.)
- Idea
  - Estimate densities in lower dimensional space that is relevant to human perception
  - Compare densities by fitting simple model (Gaussian distribution) to lower dimensional data points
- FID: Fréchet Inception Distance (FID)
  https://en.wikipedia.org/wiki/Fr%C3%A9chet_inception_distance
  - Use deepest feature layer of pre-trained Inception V3 network as lower dimensional space (2048 dimensions)
  - Fit Gaussians (mean $\mu$, variance $\Sigma$) to Inception V3 feature vectors of true, generated data
  - Compare Gaussians of true and generated data using Fréchet distance $d_F$

$$d_F(\mathcal{N}(\mu,\Sigma),\mathcal{N}(\mu',\Sigma'))^2 = \|\mu-\mu'\|_2^2 + \operatorname{tr}\left(\Sigma+\Sigma'-2\left(\Sigma^{\frac{1}{2}}\cdot\Sigma'\cdot\Sigma^{\frac{1}{2}}\right)^{\frac{1}{2}}\right)$$

# Comparisons

- FID: Fréchet Inception Distance using 50k generated images, all real data
- ID: consistency of face identity over different viewpoints using Arcface cosine similarity score
  https://arxiv.org/abs/1801.07698
- Depth/pose: consistency of NeRF geometry/poses with "pseudo ground truth" geometry/poses reconstructed from rendered multiview images

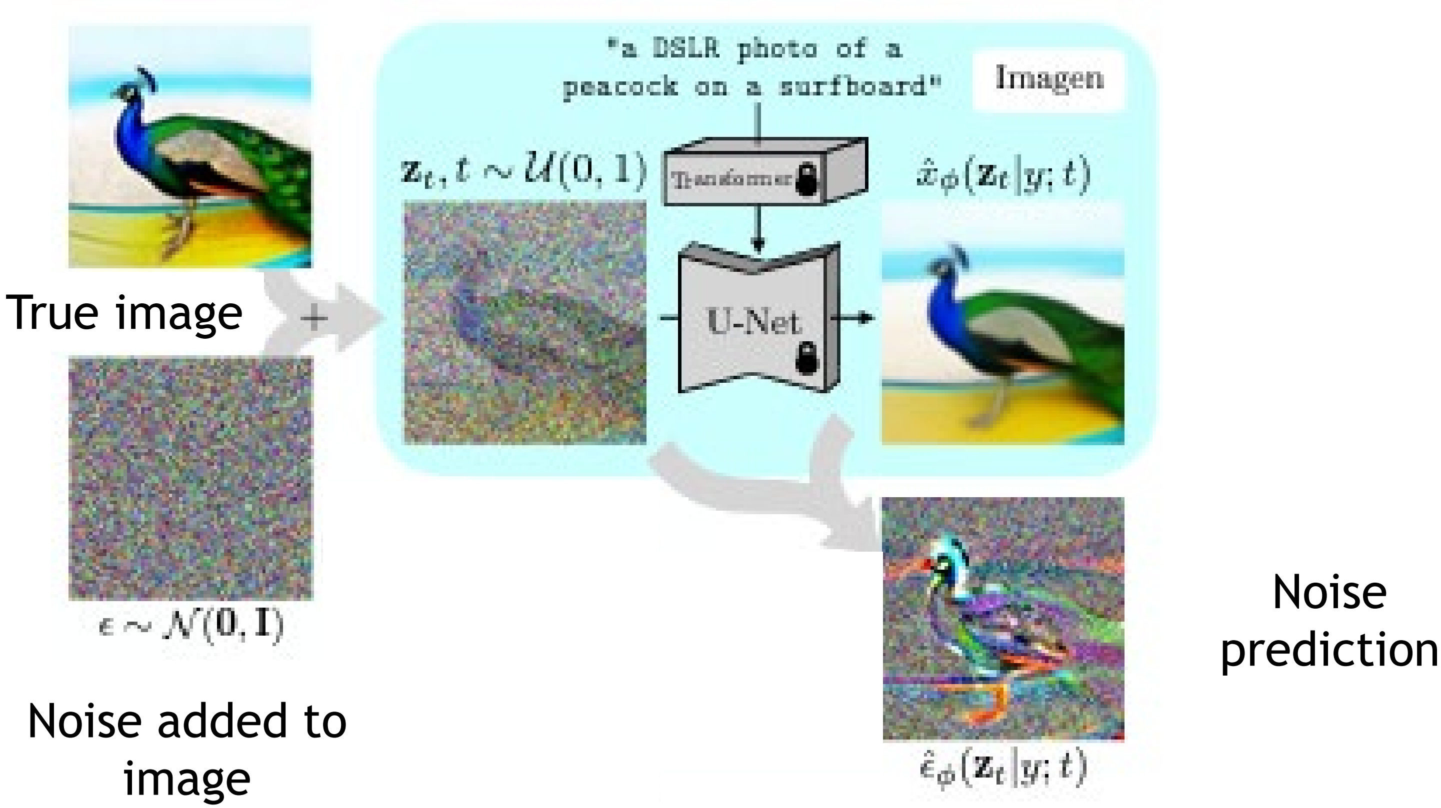
| | FFHQ | | | | Cats |
|---|---|---|---|---|---|
| | FID↓ | ID↑ | Depth↓ | Pose↓ | FID↓ |
| GIRAFFE $256^2$ | 31.5 | 0.64 | 0.94 | .089 | 16.1 |
| $\pi$-GAN $128^2$ | 29.9 | 0.67 | 0.44 | .021 | 16.0 |
| Lift. SG $256^2$ | 29.8 | 0.58 | 0.40 | .023 | — |
| Ours $256^2$ | **4.8** | 0.76 | **0.31** | **.005** | **3.88** |
| Ours $512^2$ | **4.7** | **0.77** | 0.39 | **.005** | **2.77**$^\dagger$ |

# Text-based 3D synthesis

- "DreamFusion: Text-to-3D using 2D Diffusion", https://dreamfusion3d.github.io/
- Goal: conditional generation of 3D objects based on text inputs
  - "Create random 3D objects that are consistent with given text description"
- Challenge
  - Requires large collection of text-3D object pairs, which don't exist, to train conditional generative model
- Approach: leverage existing, diffusion-based text-to-image conditional generative models in combination with NeRF-based differentiable rendering

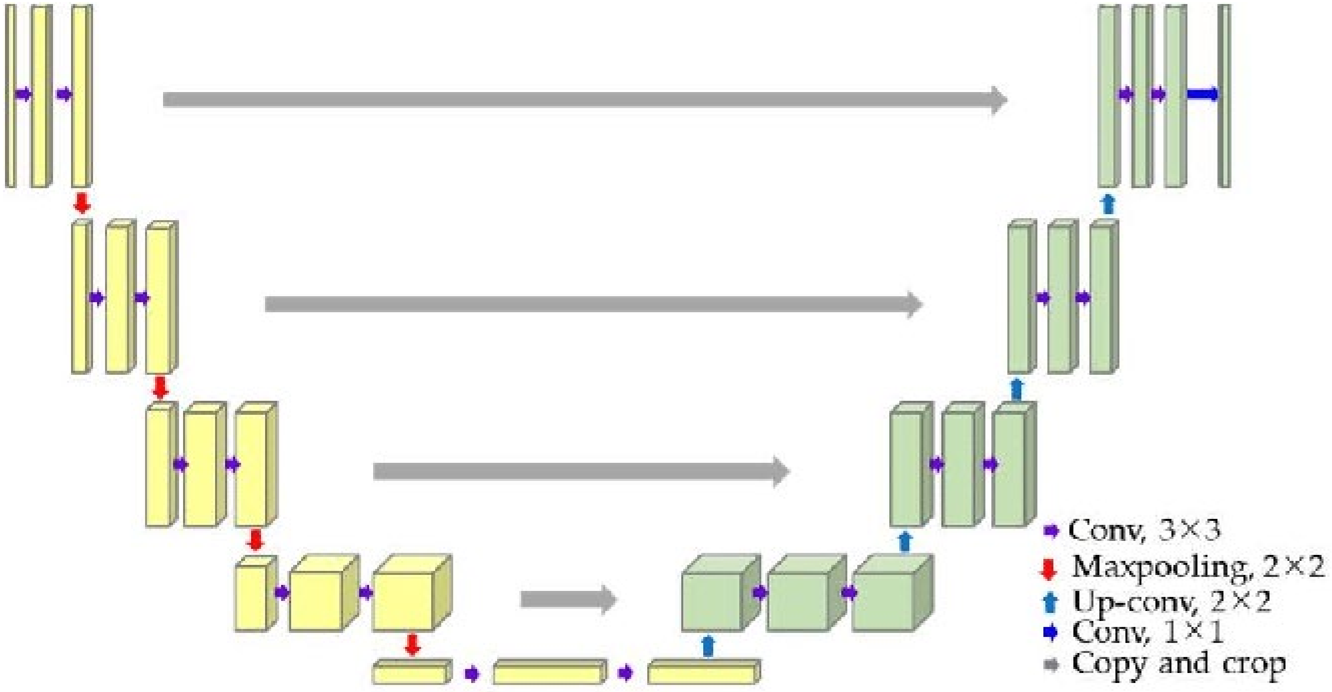
# Diffusion-based text-to-image

- Training: given text-image pairs, train U-net conditioned on text embedding trained to predict (synthetic) noise added to image
  - Here imagen model https://imagen.research.google/ https://arxiv.org/pdf/2205.11487.pdf
  - Transformer-based text embedding from pre-trained large language model (here T5-XXL, see also https://github.com/google-research/t5x)



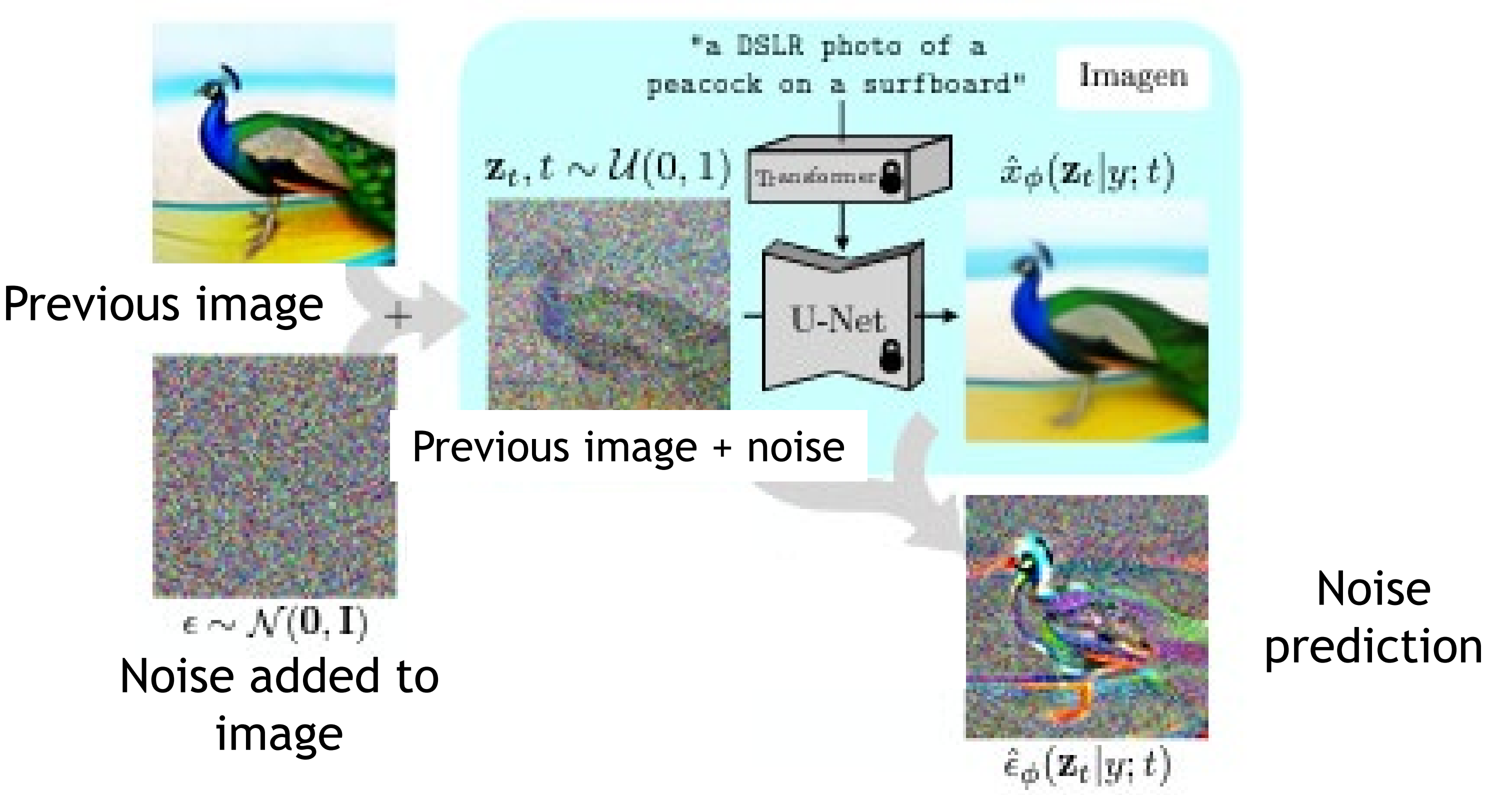
# U-net architecture

https://en.wikipedia.org/wiki/U-Net



# Diffusion-based text-to-image

- Sampling (image synthesis): in each step, subtract (noise prediction – noise) from previous image (equivalent to subtracting noise prediction from (previous image + noise), see pseudocode from last time); add new noise; iterate



# Diffusion-based text-to-image

- Sampling (image synthesis): in each step, subtract (noise prediction – noise) from previous image (equivalent to subtracting noise prediction from (previous image + noise), see pseudocode from last time); add new noise; iterate

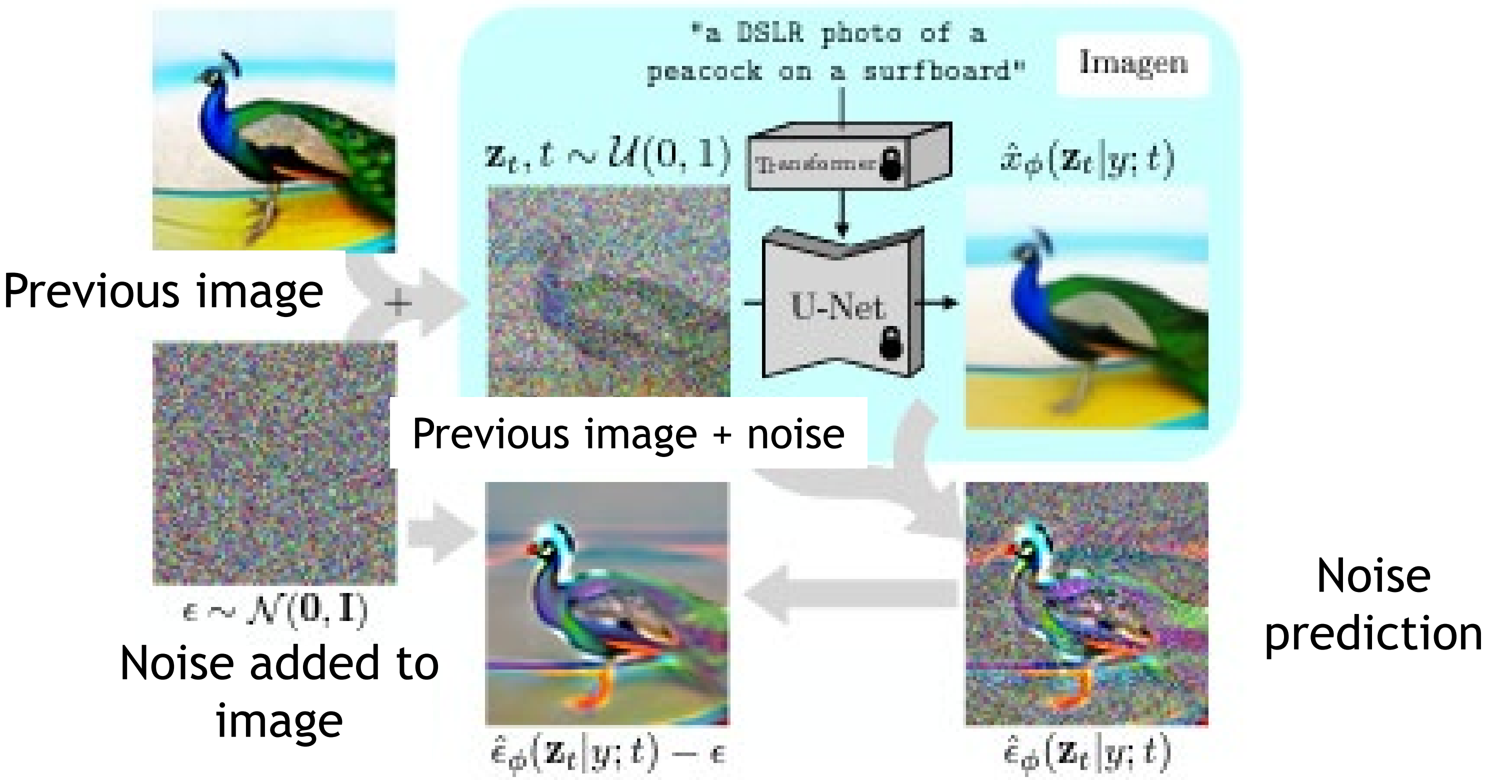
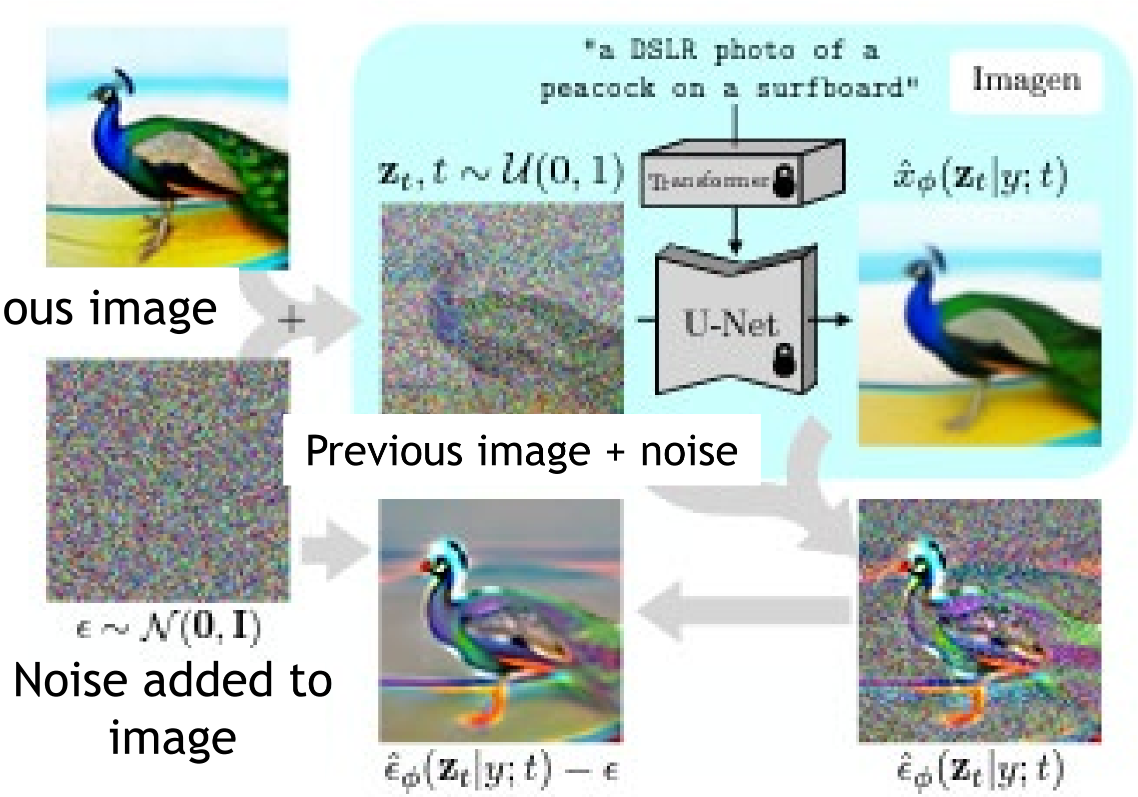



Image update given by noise prediction - noise

# Diffusion-based text-to-image

- Sampling (image synthesis): in each step, subtract (noise prediction – noise) from previous image (equivalent to subtracting noise prediction from (previous image + noise), see pseudocode from last time); add new noise; iterate



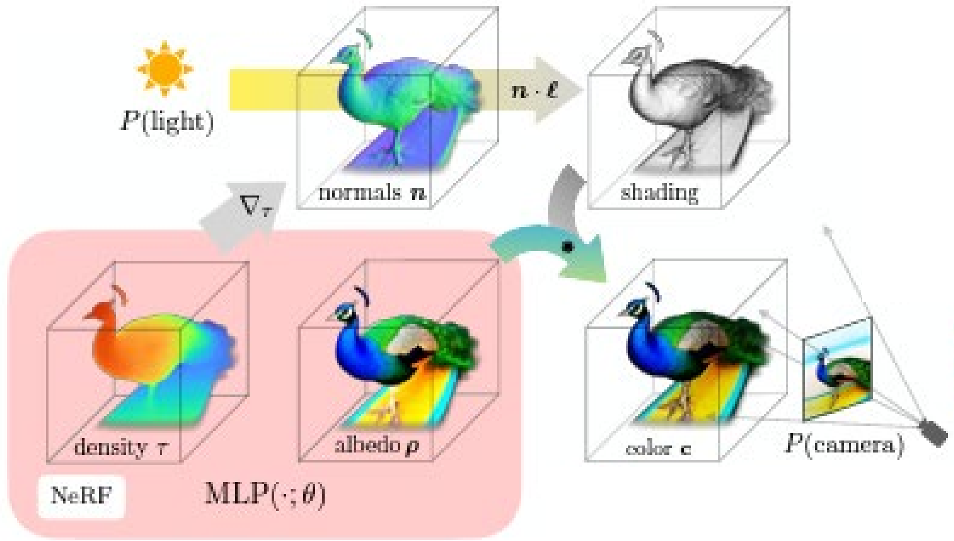

Noise prediction

Next iteration

Image update given by noise prediction - noise

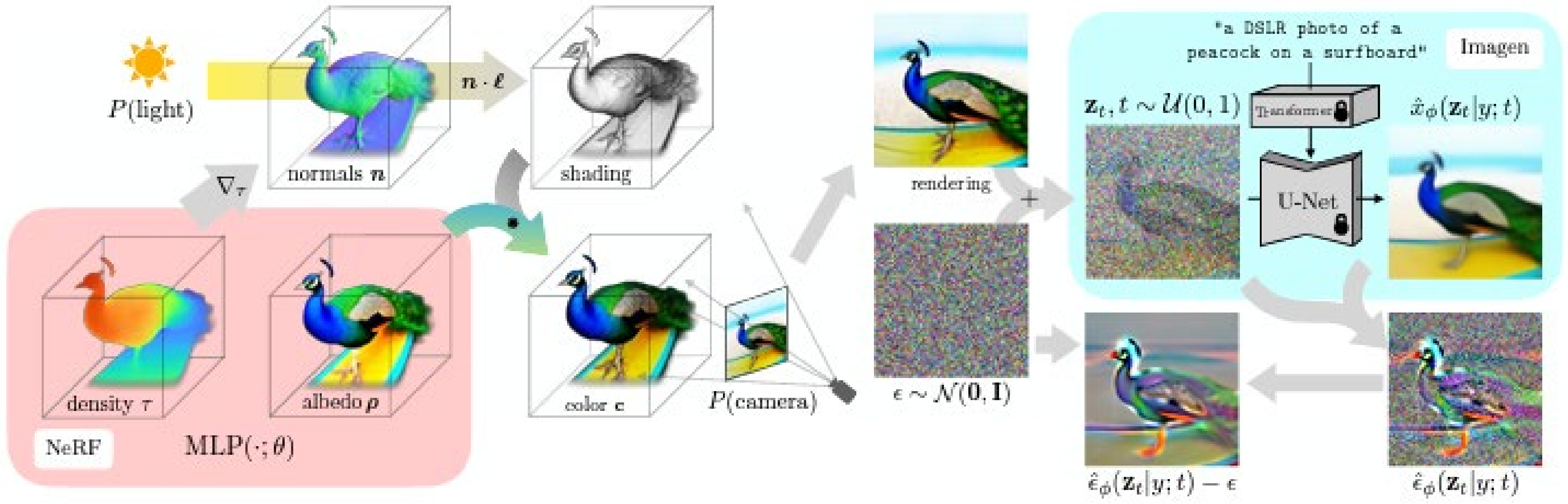
# 3D synthesis

- Image rendered using NeRF, starting from random initialization



# 3D synthesis

- Image rendered using NeRF, starting from random initialization
- Update to rendered image computed using pretrained diffusion model



# 3D synthesis

- Image rendered using NeRF, starting from random initialization
- Update to rendered image computed using pretrained diffusion model
- Desired update backpropagated to NeRF parameters

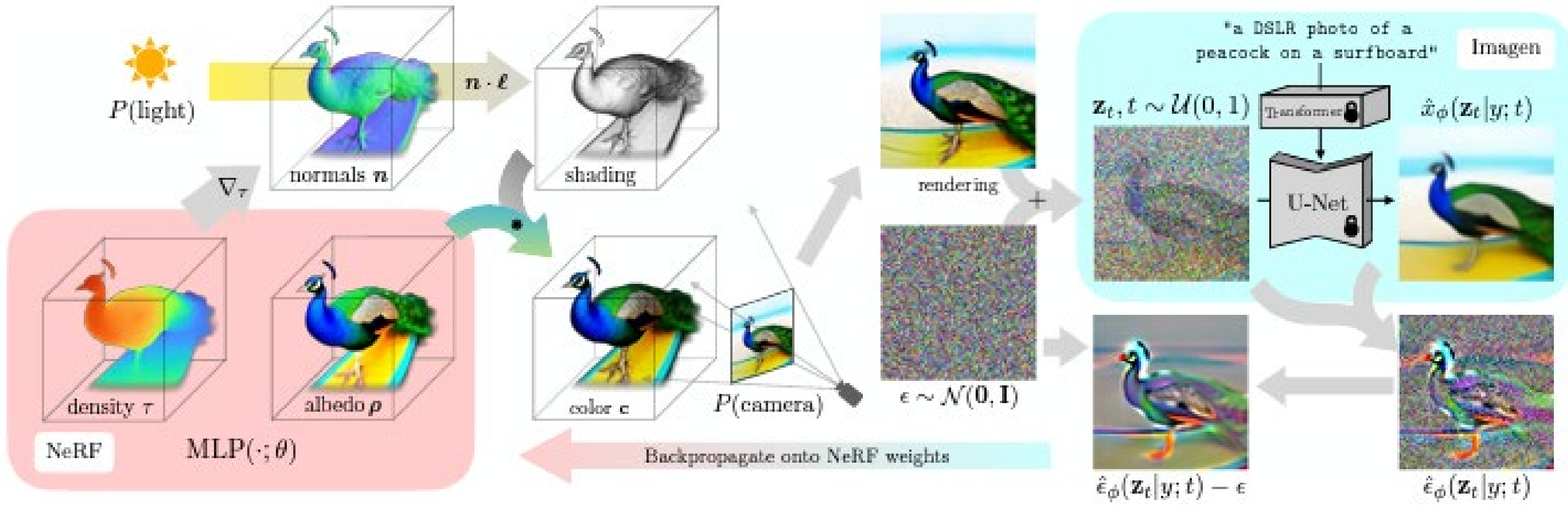



Image update given by
noise prediction - noise

# Note

- NeRF parameters optimized for each 3D object that is generated ("sampling the generative model via optimization")
  - Random viewpoints for rendering; text prompts augmented with "front", "side", "back" based on viewpoint location
  - 1.5 hours (15,000 iterations) per scene using TPUv4 machine with 4 chips; each chip rendering separate view and evaluating diffusion U-Net with per-device batch size of 1
  - More theory in DreamFusion paper https://arxiv.org/abs/2209.14988
- Text-to-image diffusion model is frozen during 3D generation; not involved in gradient computation for NeRF parameters
  - DreamFusion uses ImaGen model https://imagen.research.google/

# NeRF rendering

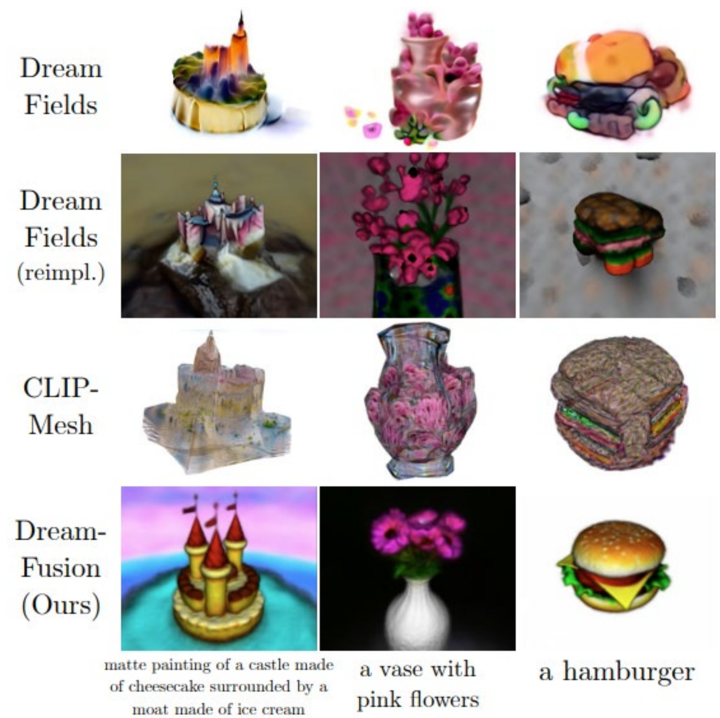
- Predict diffuse BRDF (albedo), instead of radiance
- Compute surface normals using gradient of density field
- Shading using randomly positioned point light, ambient light
  - Using shading model helps reconstruct geometry (empirically shown in ablation studies)
- NeRF only evaluated within bounding box; background color outside bounding box predicted using separate MLP

# Evaluation

- "R-Precision": accuracy with which CLIP retrieves correct image caption among a set of distractors given scene rendering
  - CLIP ("contrastive Language-Image Pre-training") trained to predict which images were paired with which texts in large image/text training dataset https://openai.com/blog/clip/

| 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 | 68.3 | – | 74.2 | – | – | – |
| (reimpl.) | **78.6** | 1.3 | (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** |

R-precision using different CLIP models



# Follow up work

- Better ways to backpropagate image update to rendering model (NeRF, or something else depending on application)
- Variational score distillation (NeurIPS 2023)
  https://github.com/thu-ml/prolificdreamer
- Classifier score distillation (ICLR 2024)
  https://github.com/CVMI-Lab/Classifier-Score-Distillation
- Other applications
  - Scene texturing (CVPR 2024)
    https://daveredrum.github.io/SceneTex/