[KAIST 산학협동 공개강좌]

3D 데이터 기반 딥러닝 기술 동향

2024.02.21

나니아랩스

Manager | 권용민 010.9221.6311 goya.kwon@narnia.ai

NARNIA LABS

Table of Contents

3D Deep Learning 기술 동향

코드 실습

Image based Generative Model

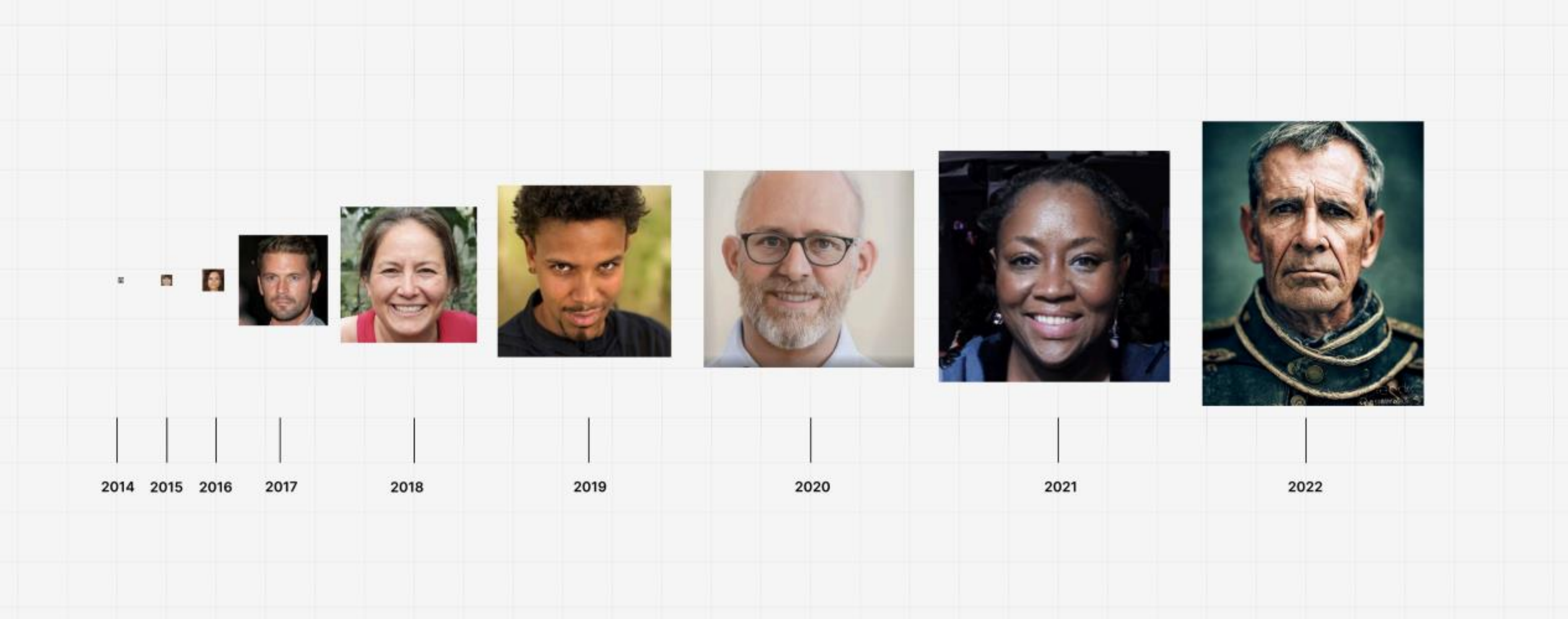
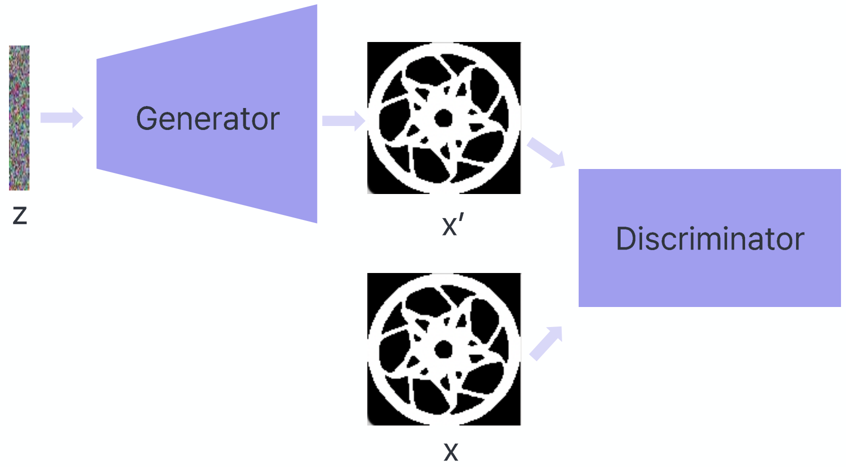
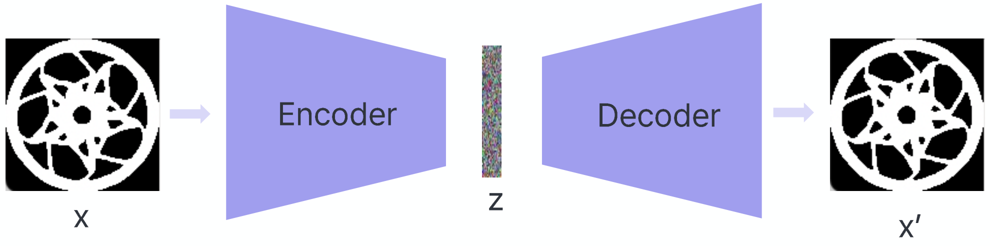


Image based Generative Model

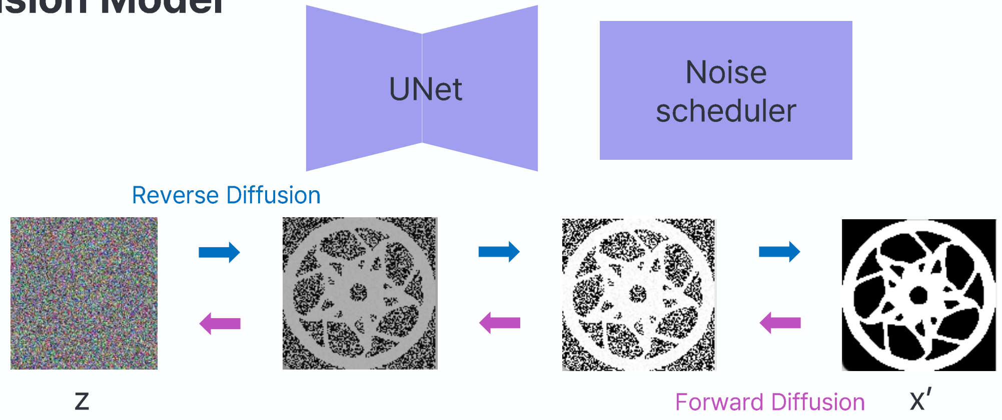
GAN(Generative Adversarial Network)



VAE(Variational Auto-Encoder)



Diffusion Model



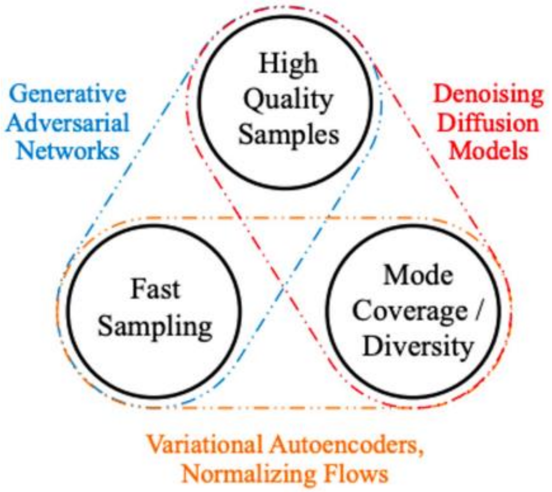
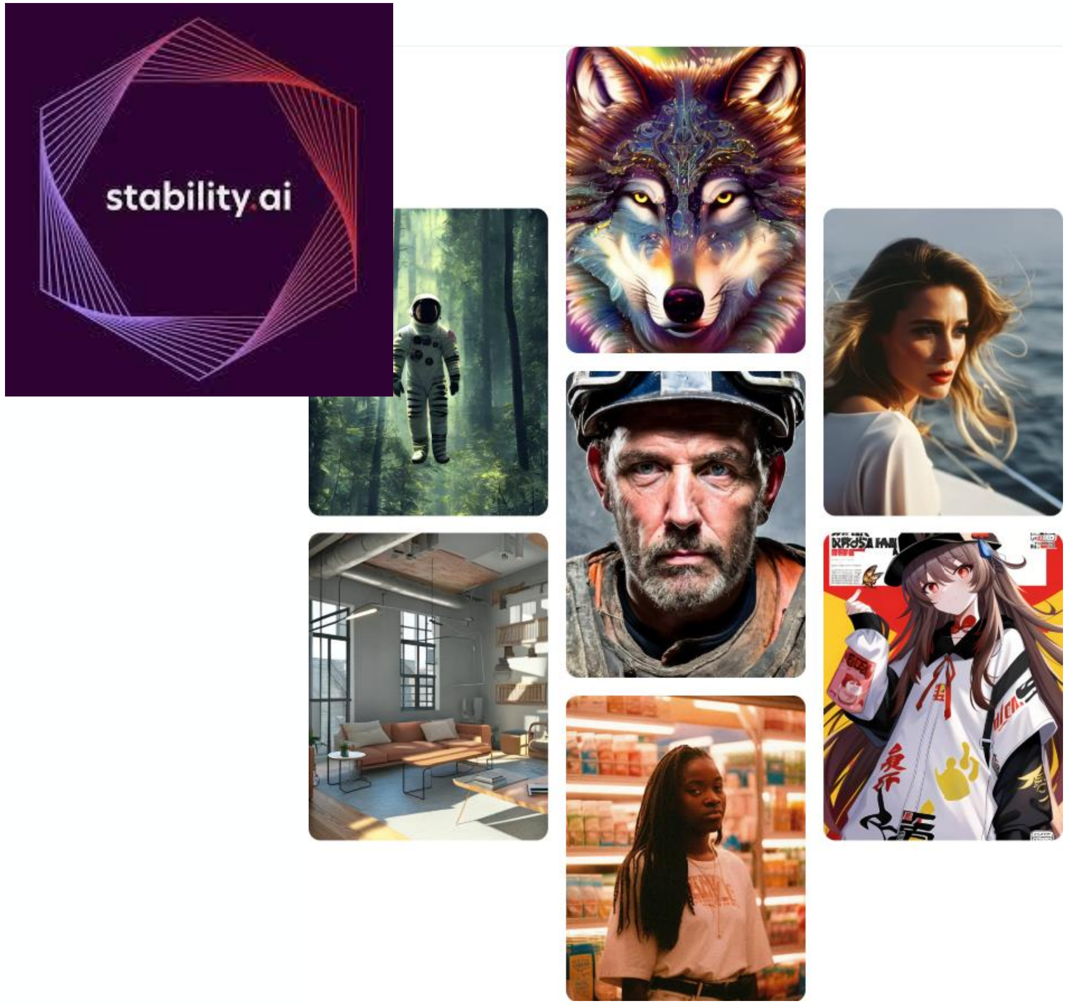


Image based Generative Model



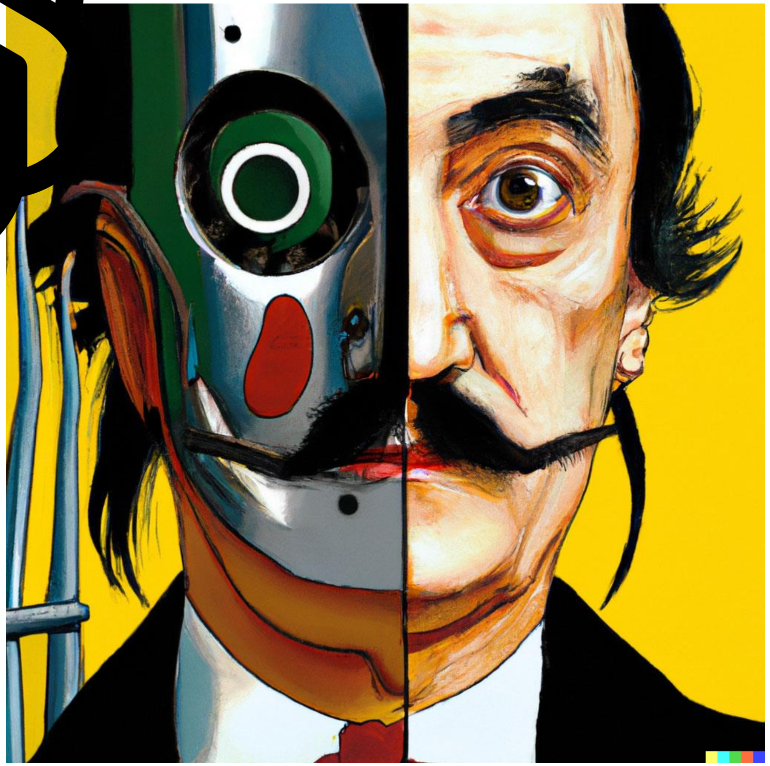
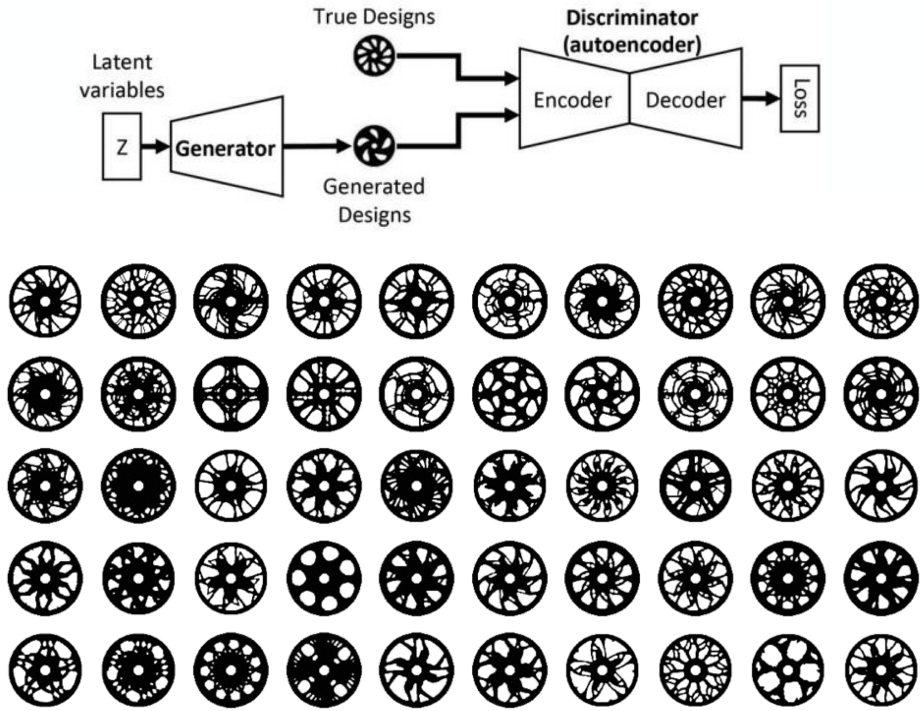
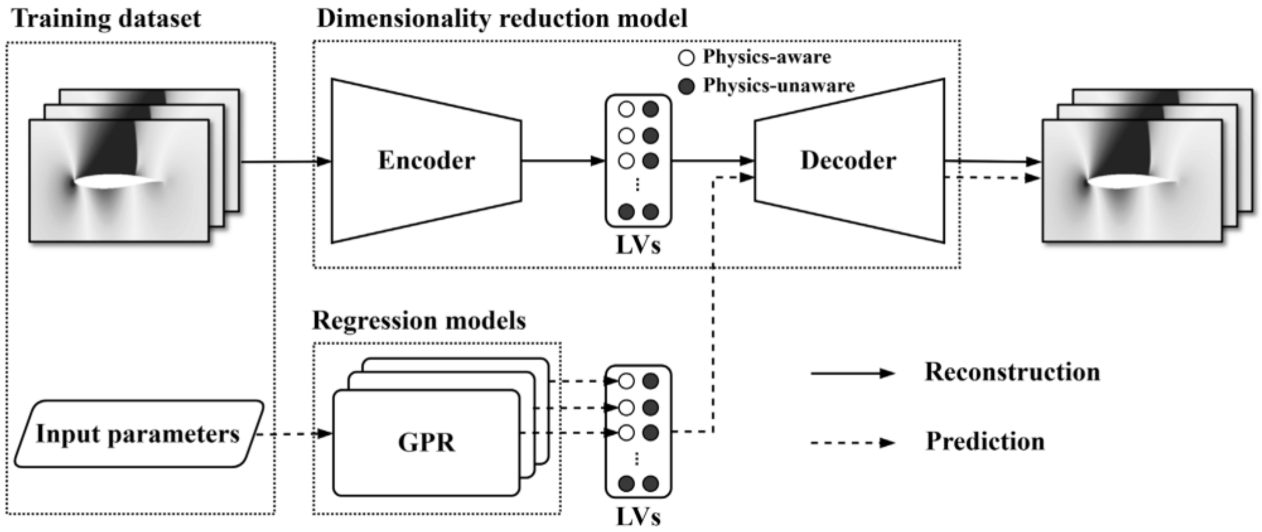


Image based Generative Model

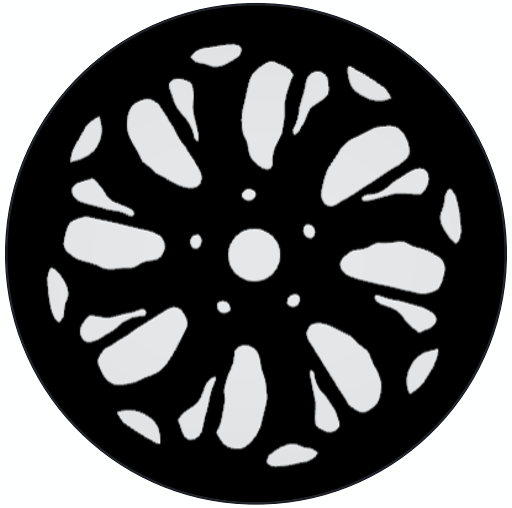




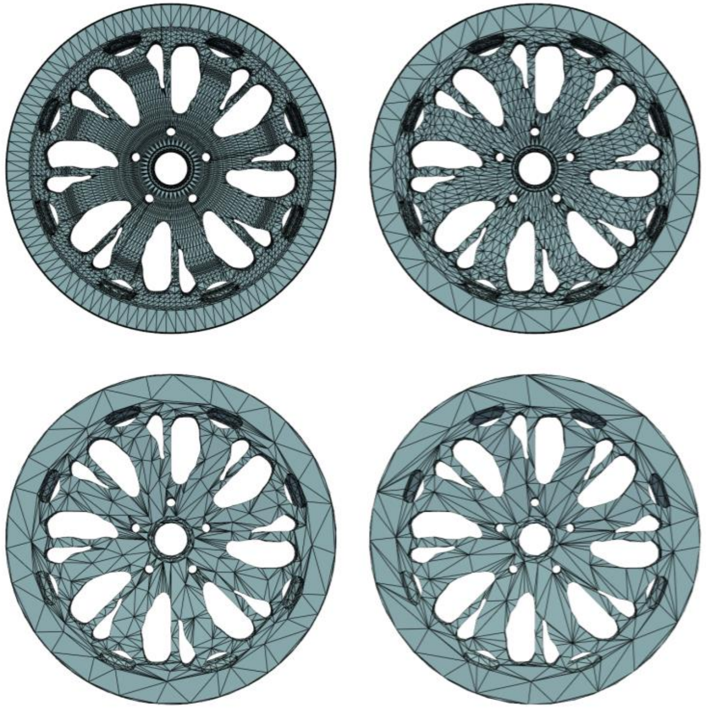
[1] Oh, S., Jung, Y., Kim, S., Lee, I., & Kang, N. (2019). Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design*, *141*(11), 111405.
[2] Kang, Y. E., Yang, S., & Yee, K. (2022). Physics-aware reduced-order modeling of transonic flow via β-variational autoencoder. *Physics of Fluids*, *34*(7).

3D Generative Model

2D Image



3D Shape

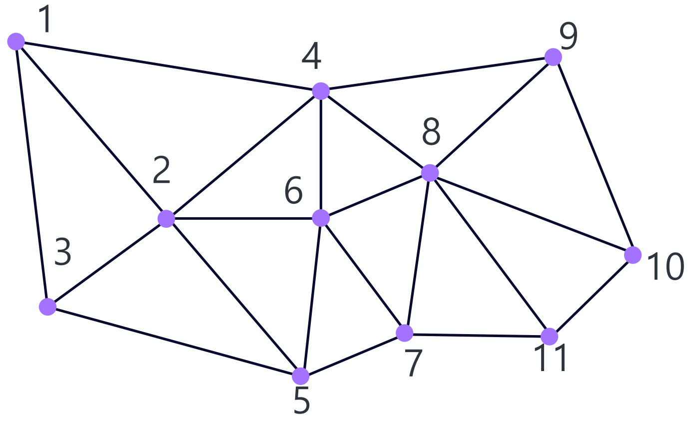


3D Generative Model

2D Image

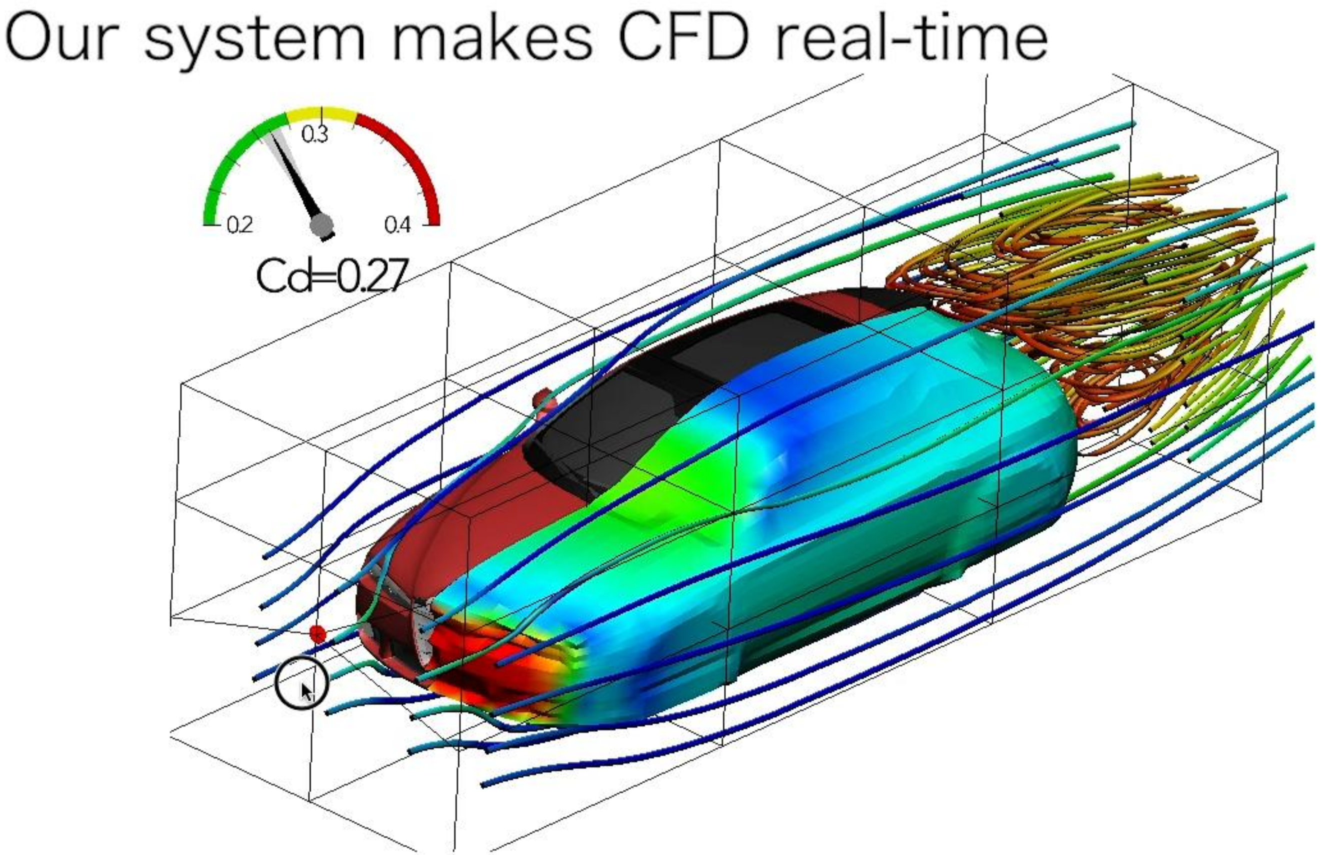
Structured Grid

3D Shape



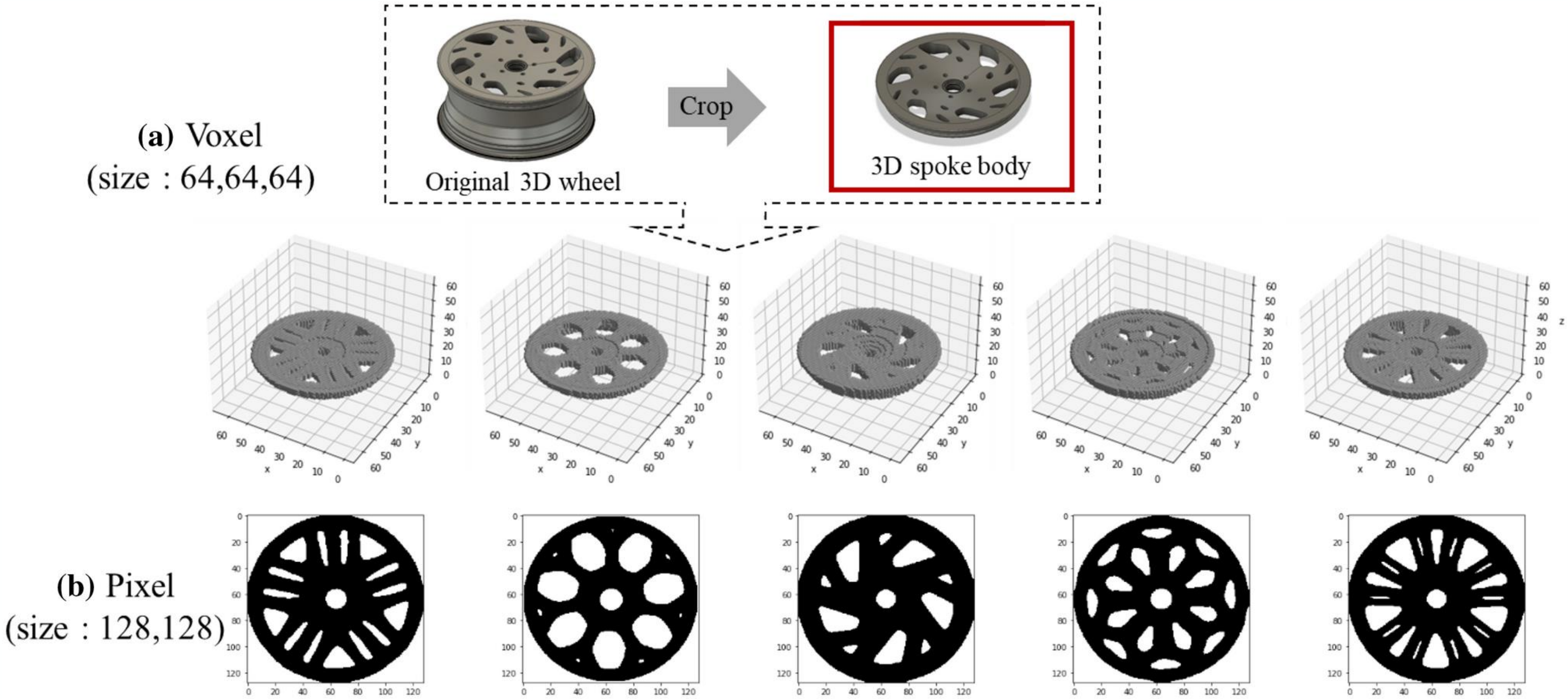
Unstructured Grid

3D Generative Model



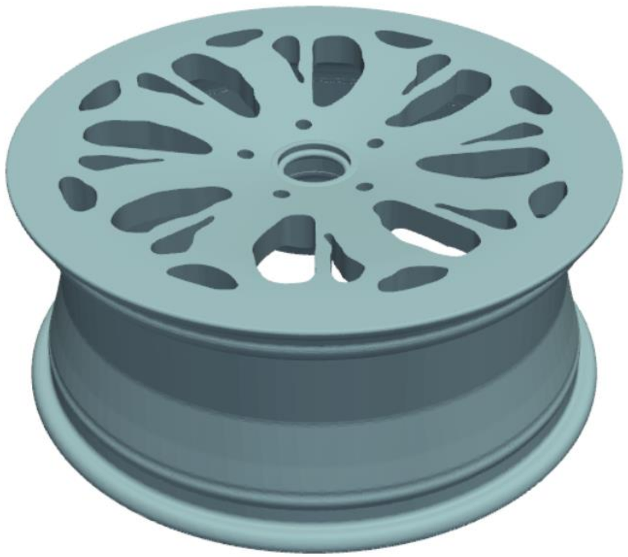
[3] Umetani, N., & Bickel, B. (2018). Learning three-dimensional flow for interactive aerodynamic design. ACM Transactions on Graphics (TOG), 37(4), 1-10.

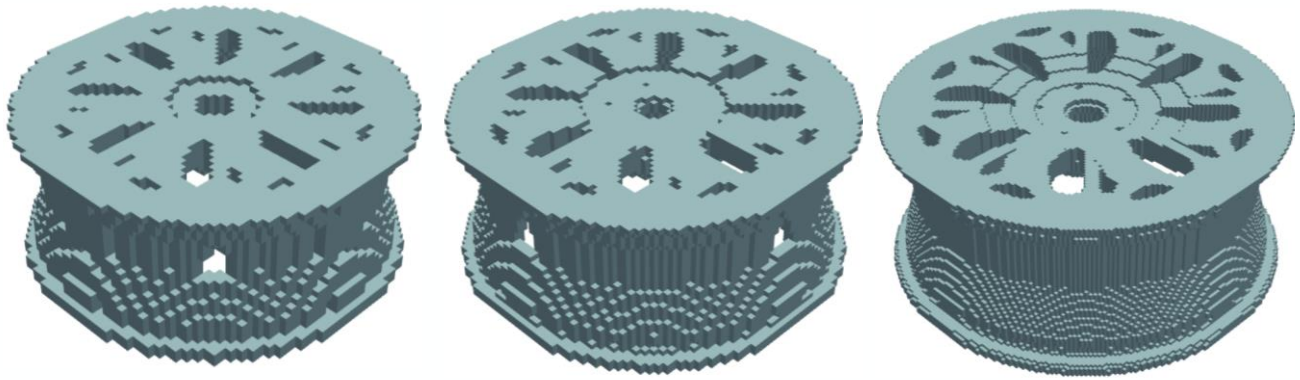
3D Generative Model



[4] Shin, S., Jin, A. H., Yoo, S., Lee, S., Kim, C., Heo, S., & Kang, N. (2023). Wheel impact test by deep learning: prediction of location and magnitude of maximum stress. *Structural and Multidisciplinary Optimization*, *66*(1), 24.

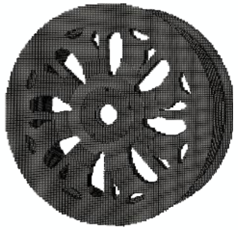
3D Generative Model





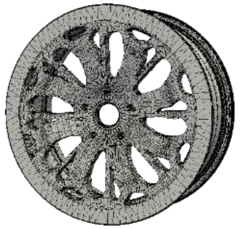
3D Generative Model

Voxel



- Discretization of 3D space into grid
- Easy to process with neural network
- **Memory issue**

Mesh

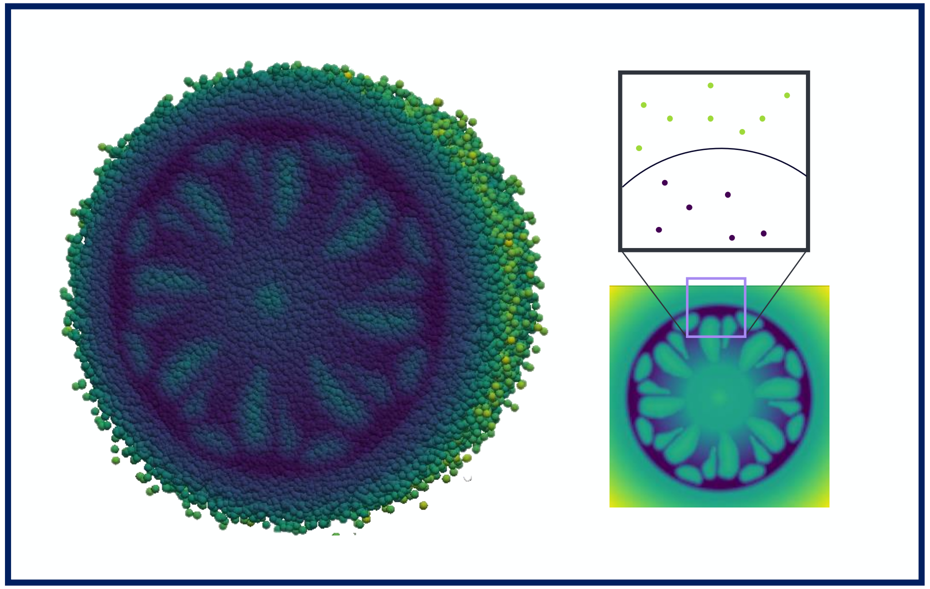


- Discretization into vertices and faces
- Compact representation
- Leads to **self-intersections**

Point Cloud

- Discretization of surface into 3D points
- Doesn’t model **topology**
- Hard to convert to mesh

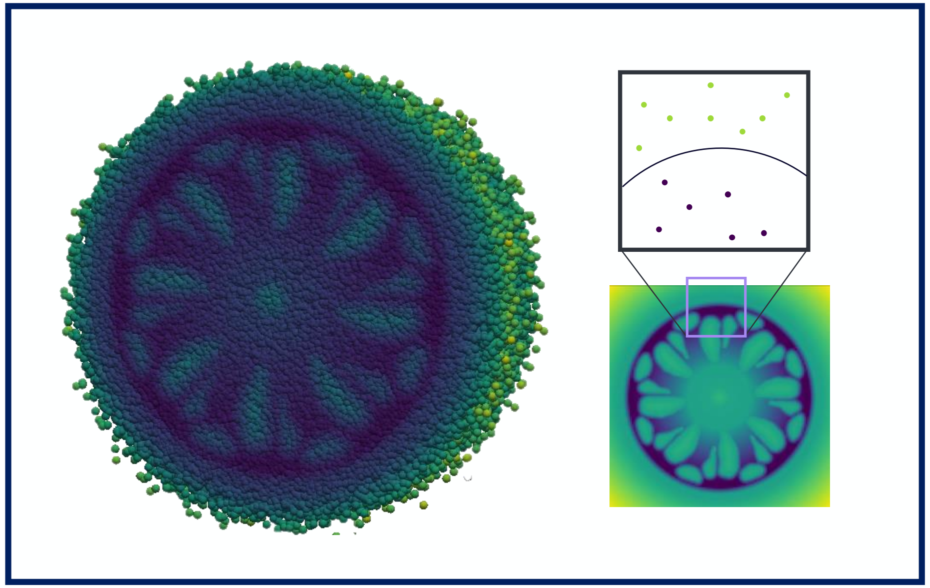
Signed Distance Function(SDF)



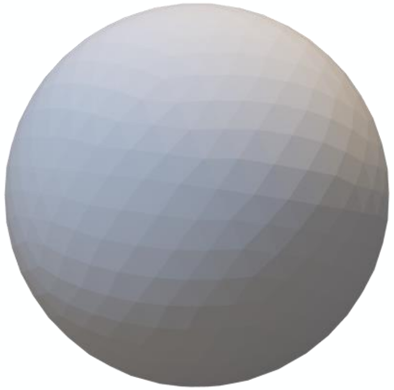
- **Implicit Representation**(w/o discretization)
- **Arbitrary topology** & resolution
- Memory efficient

3D Generative Model

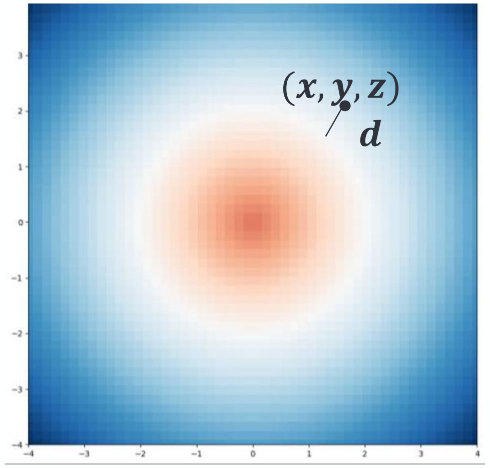
Signed Distance Function(SDF)



SDF란,
특정 공간상의 지점(point)의 좌표와 특정 표면(surface)사이 가장 가까운 거리를 반환하는 음함수로 각 지점이 형상의 내부/외부에 있는지를 부호로 표현



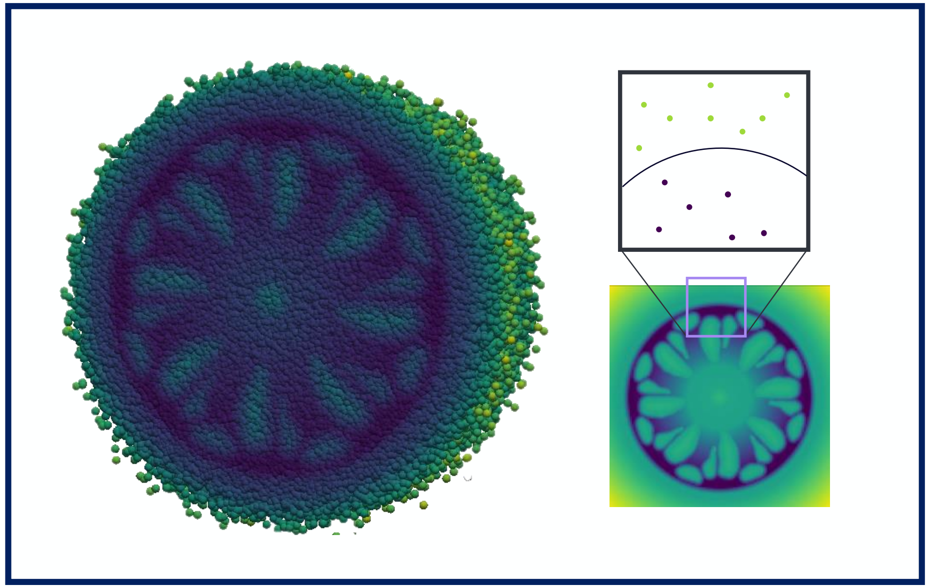
$$x^2 + y^2 + z^2 = 1$$



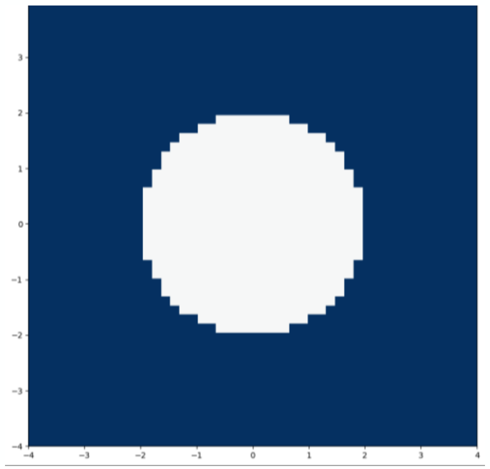
$$f(x, y, z) = d$$

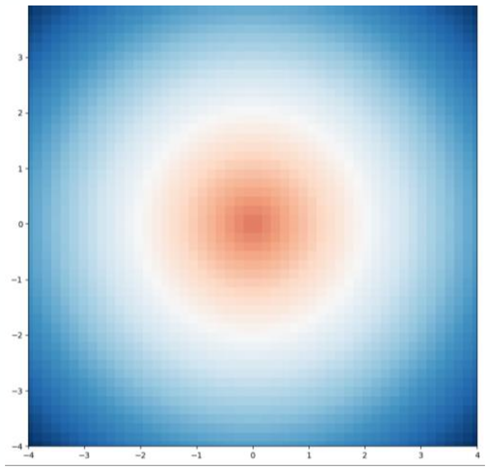
3D Generative Model

Signed Distance Function(SDF)



SDF란,
특정 공간상의 지점(point)의 좌표와 특정 표면(surface)사이 가장 가까운 거리를 반환하는 음함수로 각 지점이 형상의 내부/외부에 있는지를 부호로 표현

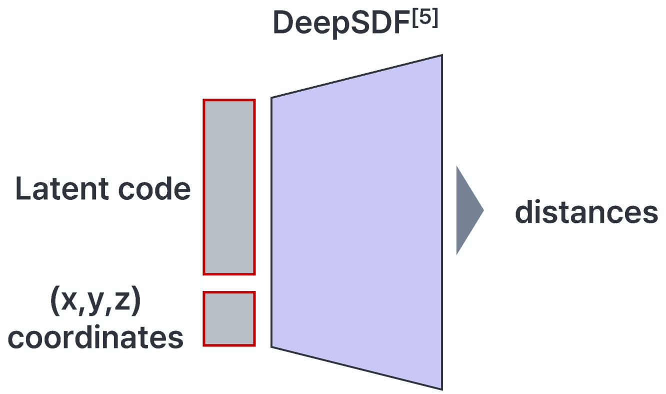




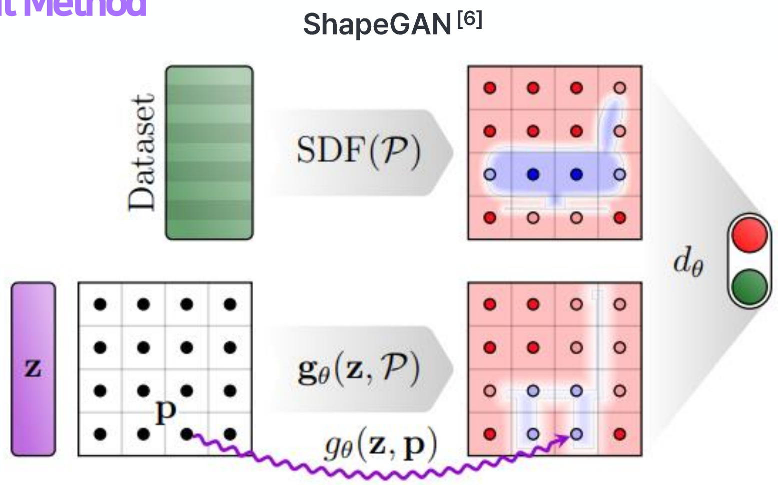
SDF	Interpolation	Detail Description	Arbitrary Topology

3D Generative Model

- Implicit Method



- Explicit Method



NN method	Data type	Memory	Resolution	Performance	Conditional Generation
Implicit	**SDF**, point cloud	**Efficient**	**No limit**	**Good**	Train is needed
Explicit	**SDF(grid type)**, voxel, multi-view, …	Inefficient	limit	Bad	**w pretrained model**

[5] Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 165-174).
[6] Kleineberg, M., Fey, M., & Weichert, F. (2020). Adversarial generation of continuous implicit shape representations. *arXiv preprint arXiv:2002.00349*.

Implicit Neural Representation

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

Jeong Joon Park[1,3†] Peter Florence [2,3†] Julian Straub[3] Richard Newcombe[3] Steven Lovegrove[3]

[1]University of Washington [2]Massachusetts Institute of Technology [3]Facebook Reality Labs

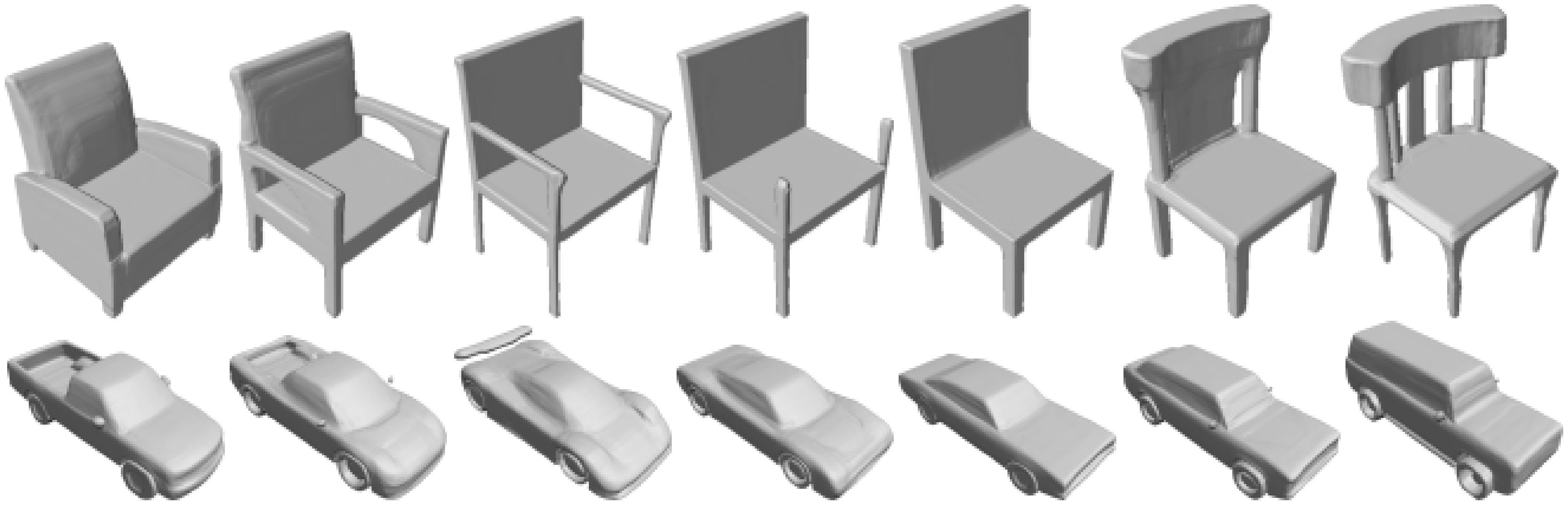
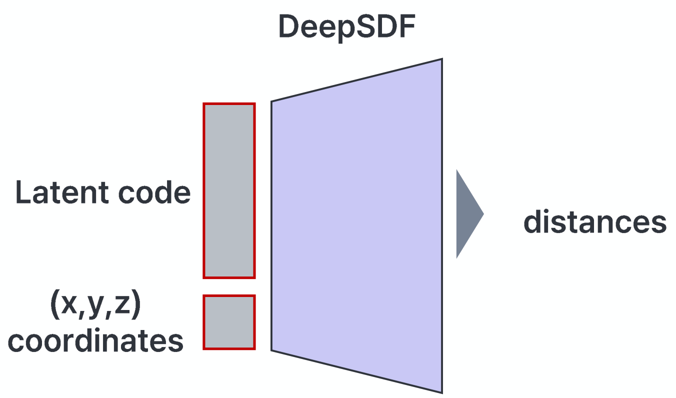
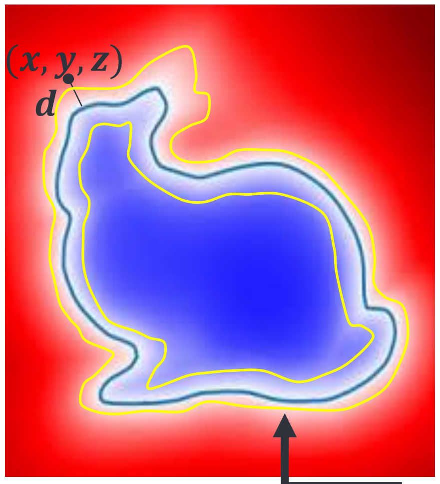


Figure 1: DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

Implicit Neural Representation

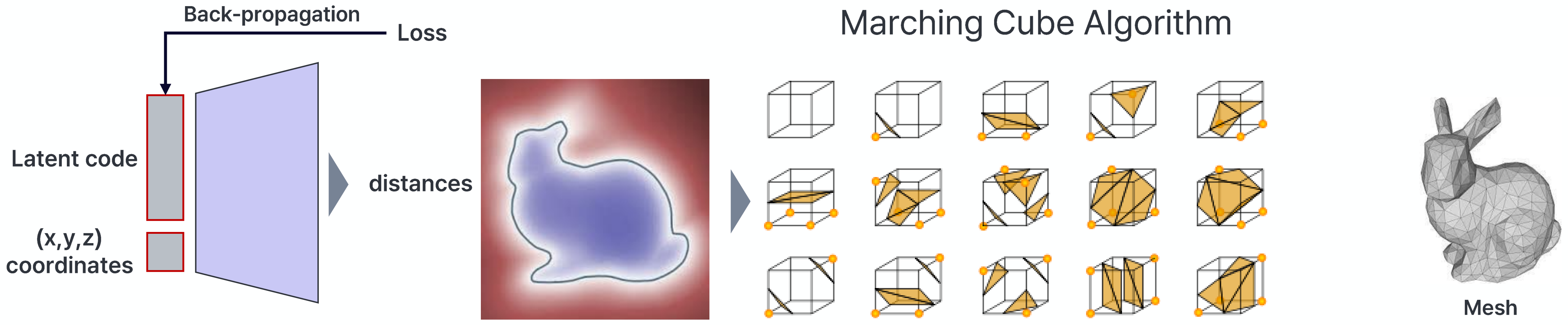




$$\mathcal{L}(f_\theta(\boldsymbol{x}), s) = \left| \boxed{\mathrm{clamp}}(f_\theta(\boldsymbol{x}), \delta) - \mathrm{clamp}(s, \delta) \right|,$$

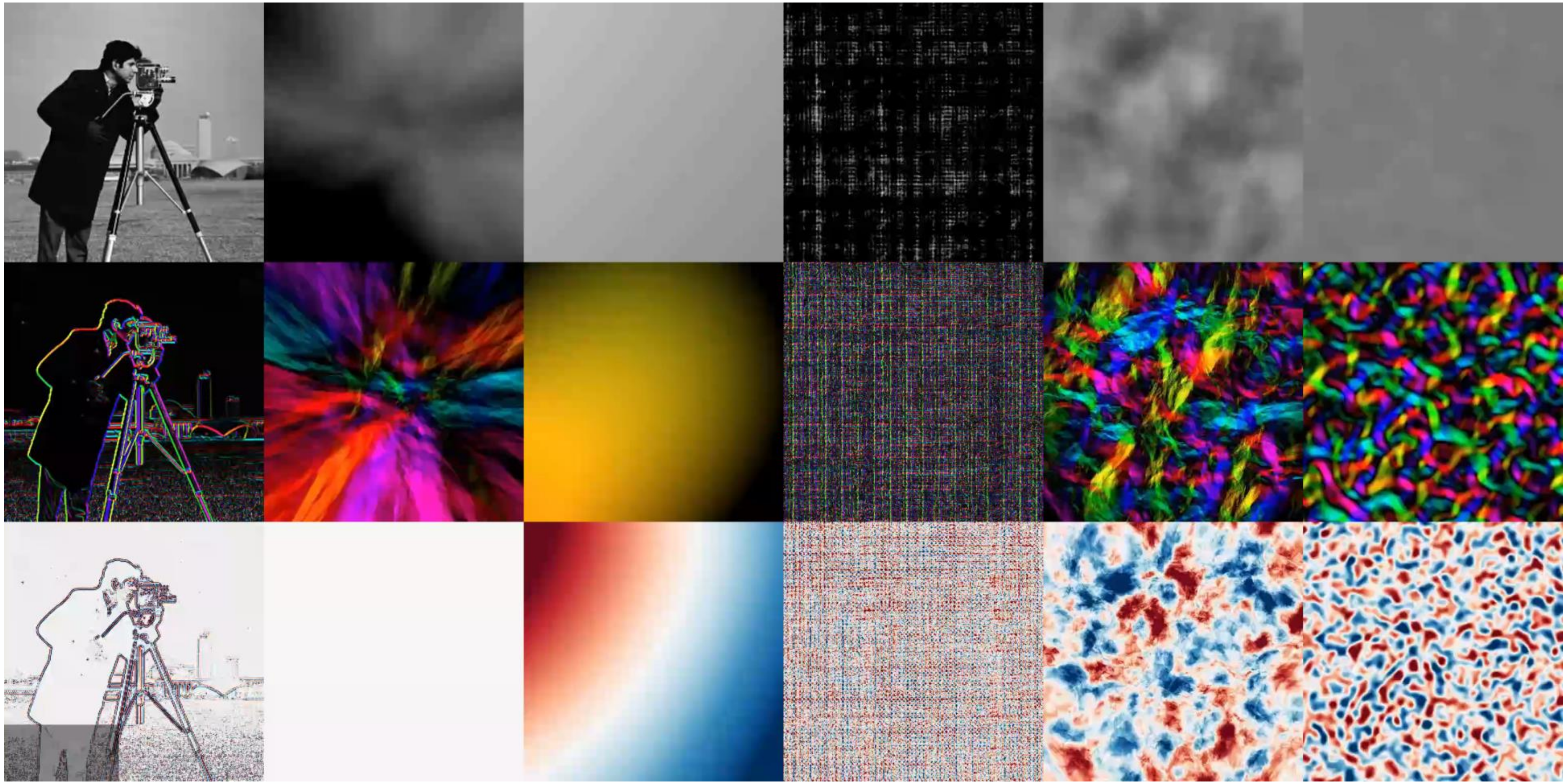
$$\hat{\boldsymbol{z}} = \overbrace{\boxed{\arg\min_{\boldsymbol{z}} \sum_{(\boldsymbol{x}_j, \boldsymbol{s}_j) \in X} \mathcal{L}(f_\theta(\boldsymbol{z}, \boldsymbol{x}_j), s_j)}}^{\text{Reconstruction}} + \overbrace{\boxed{\frac{1}{\sigma^2} ||\boldsymbol{z}||_2^2}}^{\text{Regularization}}$$

Implicit Neural Representation



Implicit Neural Representation

SIREN

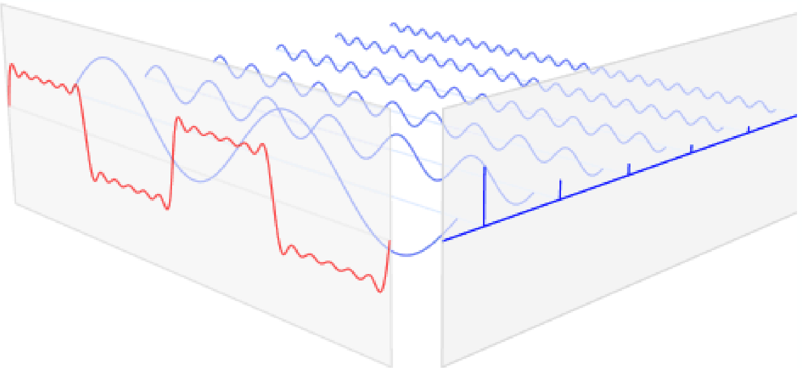


[7] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, *33*, 7462-7473.

Implicit Neural Representation

SIREN

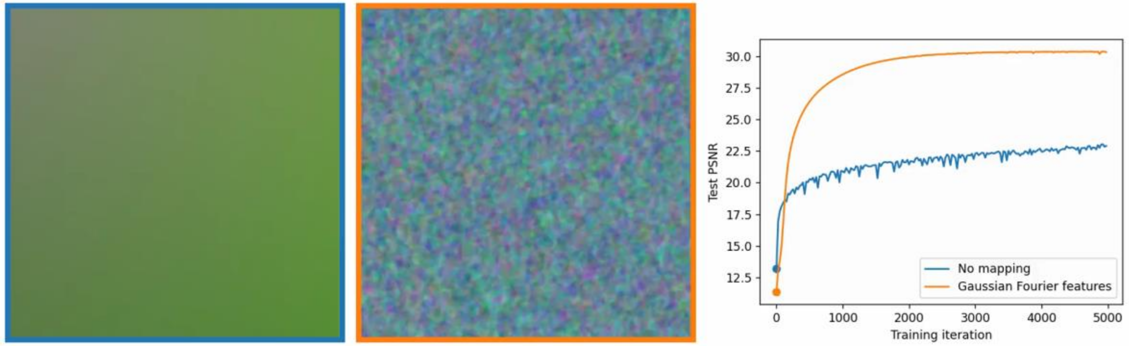
- **Positional Encoding**



Fourier Transform

시간이나 공간에 대한 함수를 시간 또는 공간 주파수 성분으로 분해하는 변환

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



Implicit Neural Representation

SIREN

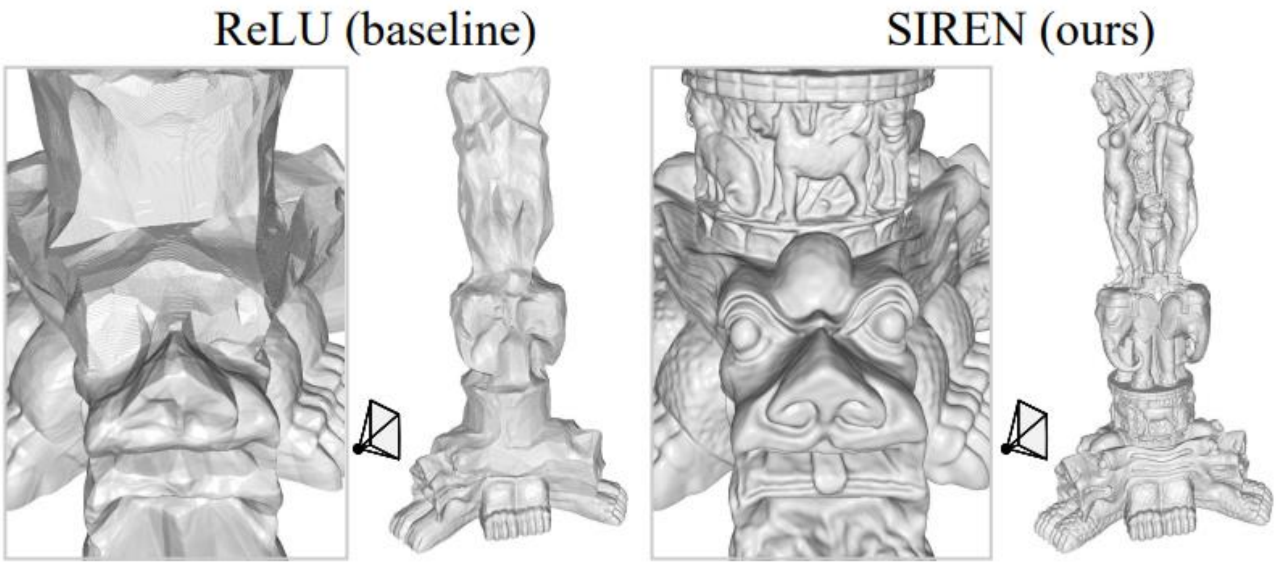
- **SIREN(Sinusoidal Representation Network)**

ReLU

$$y = max(0, w^T x + b)$$

SIREN

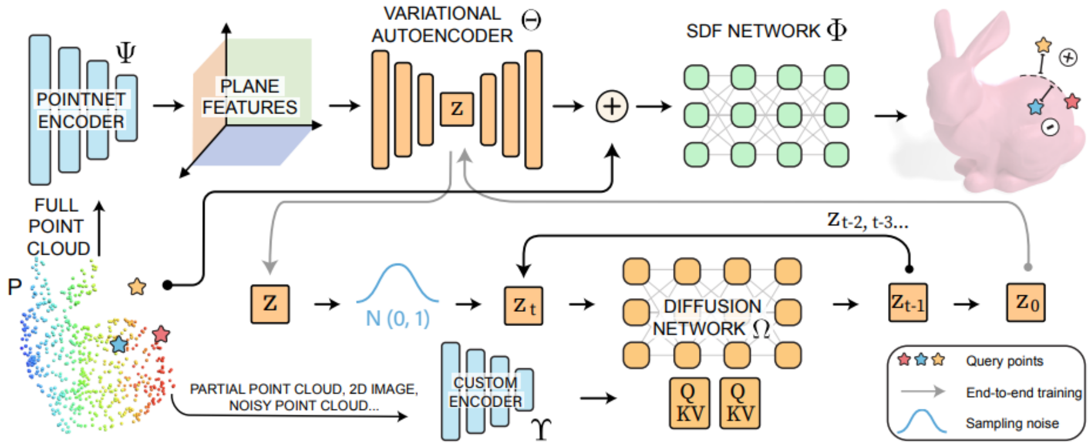
$$y = \sin(w^T x + b)$$



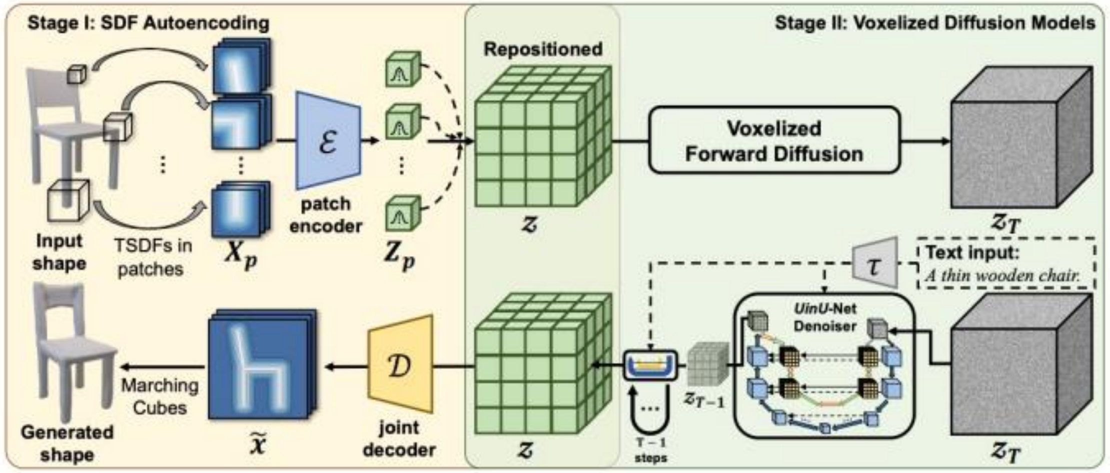
Implicit Neural Representation

- Implicit Method

Explicit Method



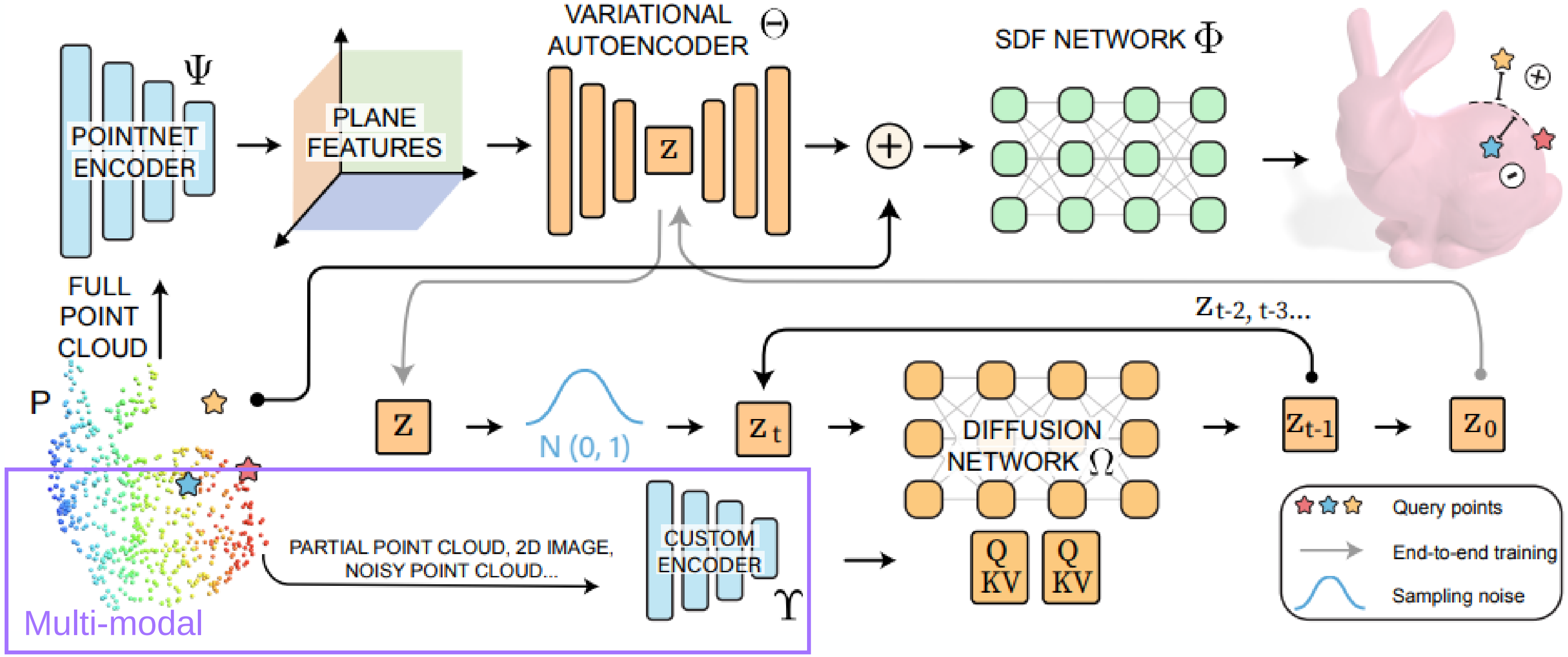
DiffusionSDF, Chou et al., 2022



Diffusion-SDF, Li et al., 2023

[8] Chou, G., Bahat, Y., & Heide, F. (2022). Diffusionsdf: Conditional generative modeling of signed distance functions. *arXiv preprint arXiv:2211.13757*.
[9] Li, M., Duan, Y., Zhou, J., & Lu, J. (2023). Diffusion-sdf: Text-to-shape via voxelized diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12642-12651).

Implicit Neural Representation



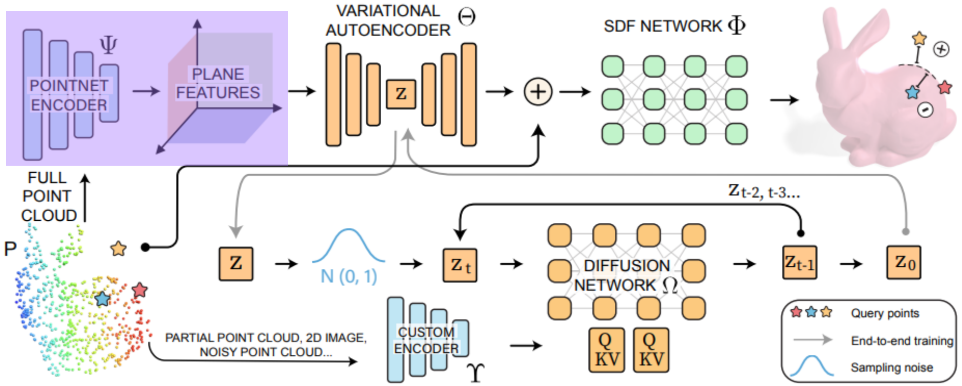
Architecture

Pointnet | Beta-VAE

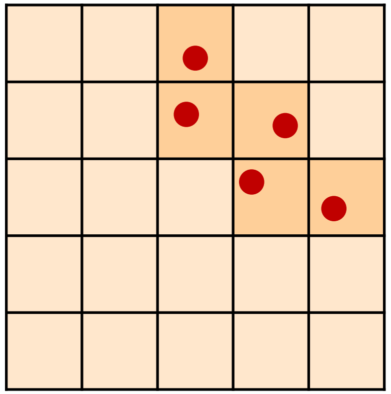
Diffusion | Decoder

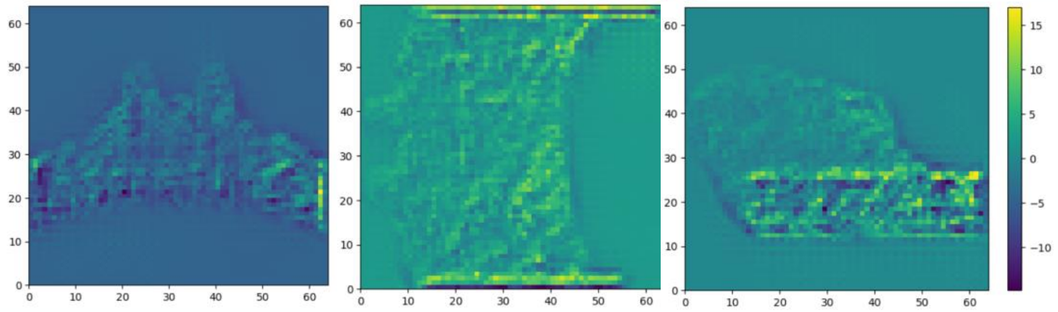
Implicit Neural Representation

Pointnet



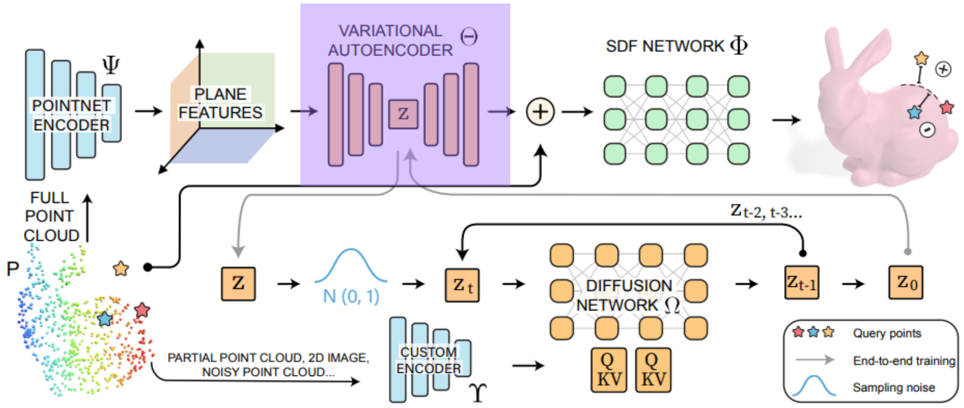
point → ResNet → Point Cloud Features1 → U-Net → Point Cloud Features2





(B x 512 x reso x reso)

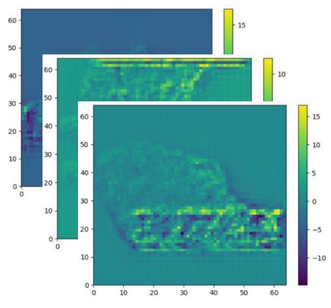
Implicit Neural Representation Beta-VAE

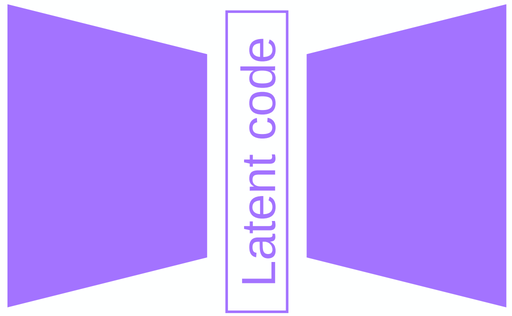


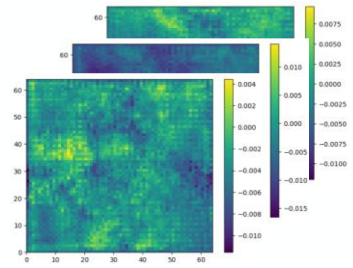

```python
# only using VAE loss
def loss_function(self,
                  *args,
                  **kwargs) -> dict:
    self.num_iter += 1
    recons = args[0]
    data = args[1]
    mu = args[2]
    log_var = args[3]
    kld_weight = kwargs['M_N']  # Account for the minibatch samples from the dataset
    #print("recon, data shape: ", recons.shape, data.shape)
    #recons_loss = F.mse_loss(recons, data)

    # kld_loss = torch.mean(-0.5 * torch.sum(1 + log_var - mu ** 2 - log_var.exp(), dim = 1), dim = 0)


    if self.kl_std == 'zero_mean':
        latent = self.reparameterize(mu, log_var)
        #print("latent shape: ", latent.shape) # (B, dim)
        l2_size_loss = torch.sum(torch.norm(latent, dim=-1))
        kl_loss = l2_size_loss / latent.shape[0]

    else:
        std = torch.exp(0.5 * log_var)
        gt_dist = torch.distributions.normal.Normal( torch.zeros_like(mu), torch.ones_like(std)*self.kl_std )
        sampled_dist = torch.distributions.normal.Normal( mu, std )
        #gt_dist = normal_dist.sample(log_var.shape)
        #print("gt dist shape: ", gt_dist.shape)

        kl = torch.distributions.kl.kl_divergence(sampled_dist, gt_dist) # reversed KL
        kl_loss = reduce(kl, 'b ... -> b (...)', 'mean').mean()

    return kld_weight * kl_loss
```

Only KLD Loss

Point Cloud Features2





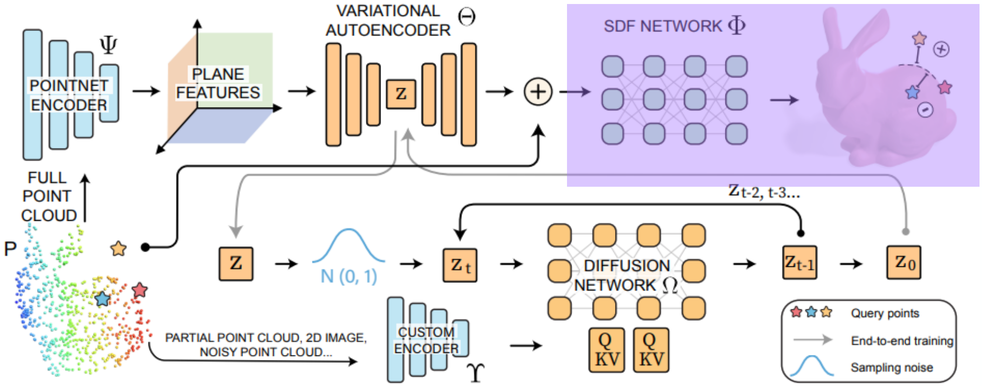


(B x 256 x reso x reso)

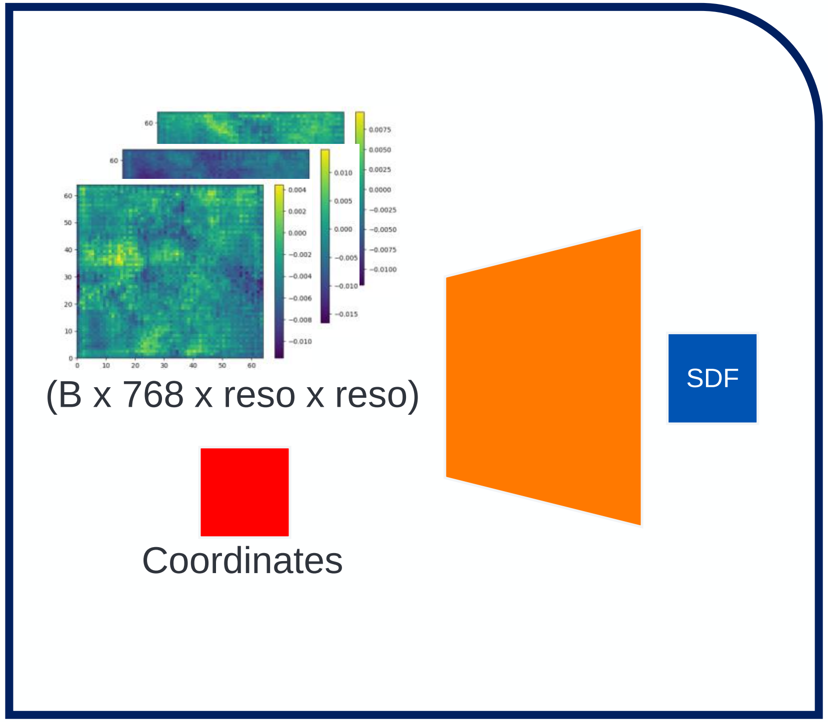
(B x 768 x reso x reso)

768 = 3 * 256

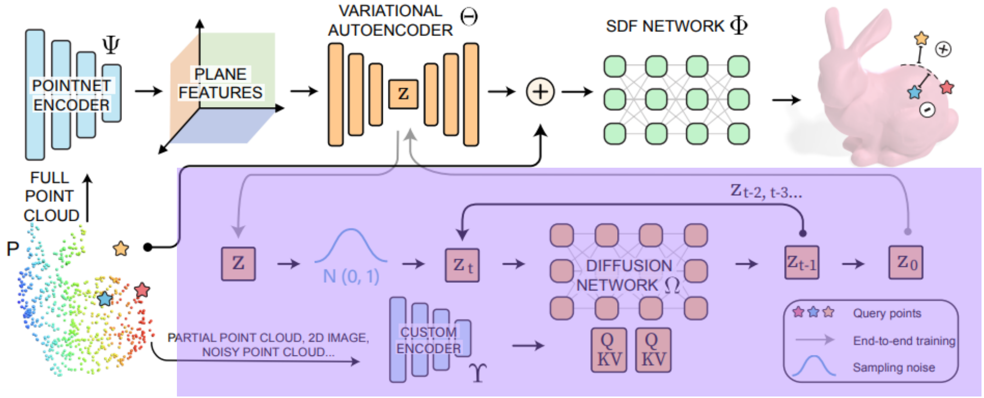
Implicit Neural Representation Decoder




```python
def training_step(self, x, idx):

    xyz = x['xyz'].cuda() # (B, 16000, 3)
    gt = x['gt_sdf'].cuda() # (B, 16000)
    pc = x['point_cloud'].cuda() # (B, 1024, 3)

    modulations = self.pointnet(pc, xyz)

    pred_sdf, new_mod = self.model(xyz, modulations)

    sdf_loss = F.l1_loss(pred_sdf.squeeze(), gt.squeeze(), reduction = 'none')
    sdf_loss = reduce(sdf_loss, 'b ... -> b (...)', 'mean').mean()

    return sdf_loss
```



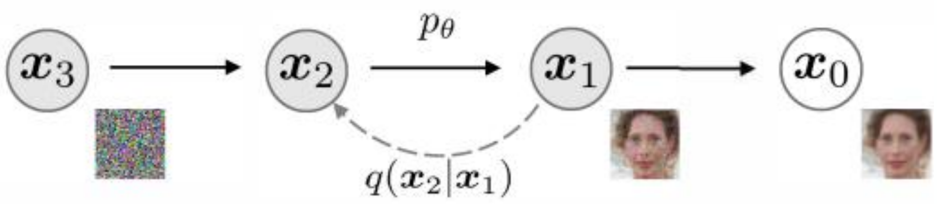
Implicit Neural Representation
Diffusion



$$q(x_1|x_0)\,q(x_2|x_1,x_0)\,q(x_3|x_2,x_0)\cdots q(x_T|x_{T-1},x_0)$$

$$= \cancel{q(x_1|x_0)}\cdot\frac{q(x_1|x_2,x_0)\,\cancel{q(x_2|x_0)}}{\cancel{q(x_1|x_0)}}\cdot\frac{q(x_2|x_3,x_0)\,\cancel{q(x_3|x_0)}}{\cancel{q(x_2|x_0)}}\cdots\frac{q(x_{T-1}|x_T,x_0)\,q(x_T|x_0)}{\cancel{q(x_{T-1}|x_0)}}$$

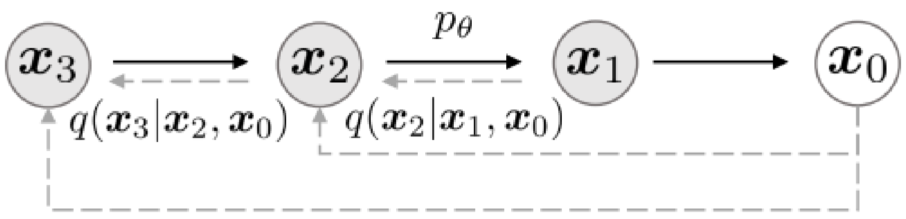
$$= \prod_{t=1}^{T} q(x_{t-1}|x_t,x_0)\cdot q(x_T|x_0)$$



DDPM Forward Process Distribution

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Markov Chain

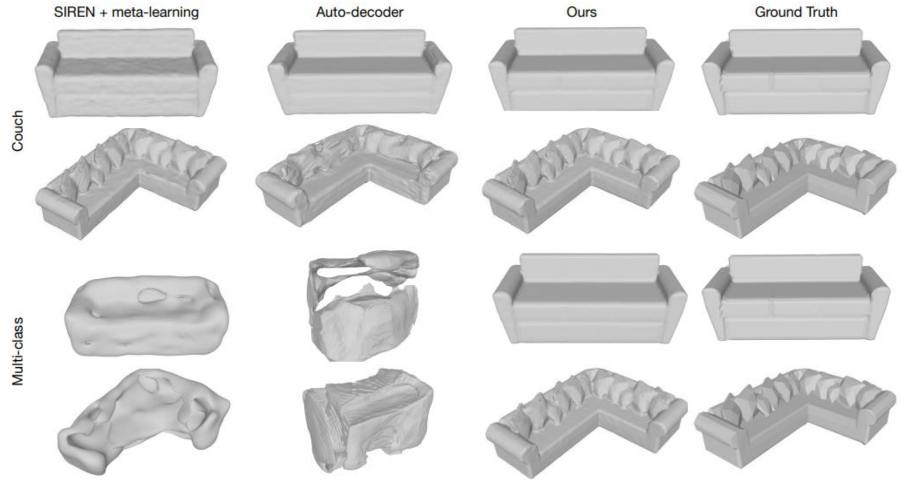


DDIM Forward Process Distribution

$$q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$$

Non-Markov Chain

Implicit Neural Representation



Conditional Generation

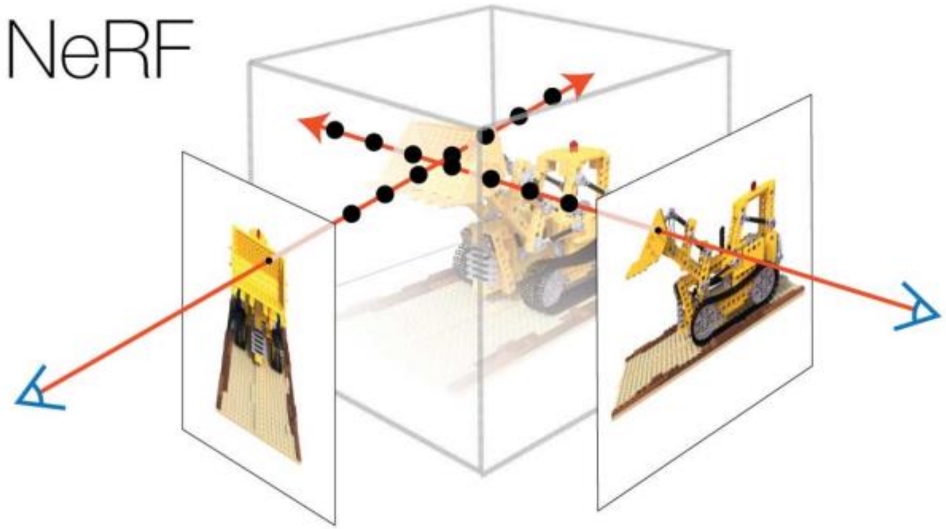
Figure 1. Our method generates clean meshes with diverse geometries. (**Top**) Unconditional generations from training on multiple classes. (**Bottom**) Conditional generation given various visual inputs, such as partial point clouds (same point cloud overlaid on sample), real-

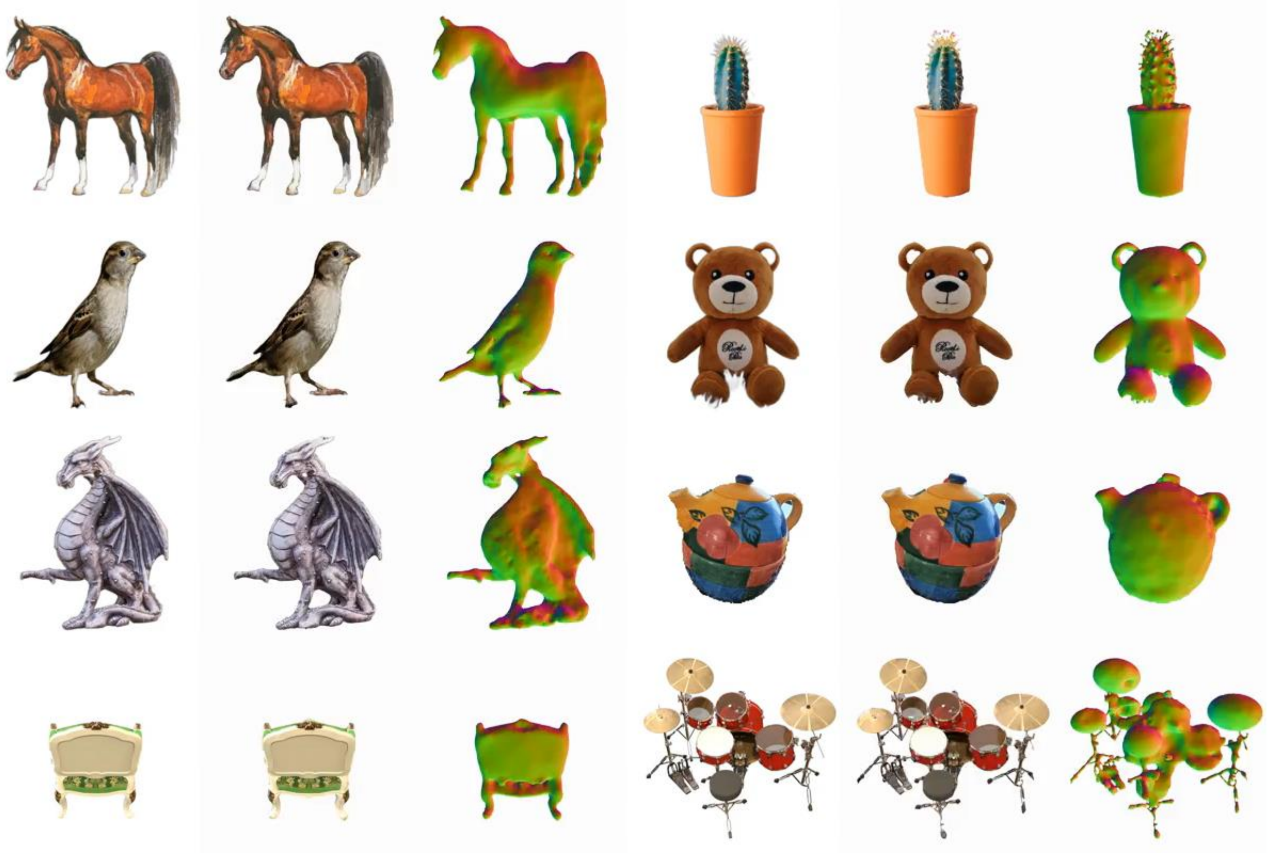
Training Data	# meshes	CD of couches (↓)	CD of all meshes (↓)
Couch	366	1.04	-
All classes	7148	0.87	0.92

A number of data ▶ Good Quality

Implicit Neural Representation

Multi-View





Multi-View

Code Implementation

나니아랩스는 딥러닝 기반 제너레이티브 디자인 기술을 통해
공학 설계 및 디자인 문제를 해결합니다.

다양한 산업 경험과 노하우로 다져진
나니아랩스만의 생성형 AI 기술을 만나 보세요.

감사합니다.

Narnia: goya.kwon@narnia.ai

KAIST: kymin1002@kaist.ac.kr

웹사이트 www.narnia.ai
문의 contact@narnia.ai