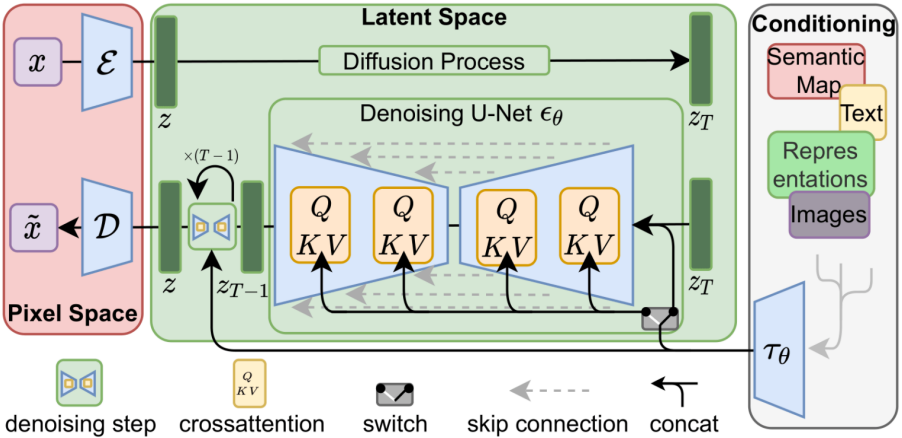
Advanced Image Generation Models

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

June 23, 2025

Table of Contents


Latent Space
Diffusion Process
Denoising U-Net ϵ_θ
Conditioning
Semantic Map
Text
Representations
Images
Pixel Space
x
$\mathcal{E}$
z
z_T
$\tilde{x}$
$\mathcal{D}$
$\times(T-1)$
z_{T-1}
Q K V
τ_θ
denoising step
crossattention
switch
skip connection
concat

Motivation

- Generative Adversarial Networks (GANs) have been the dominant approach for image generation.
- However, GANs can be difficult to train and often suffer from issues like mode collapse.
- Diffusion models offer a new approach that avoids these pitfalls by modeling the data distribution through a gradual denoising process.

Advanced Image Generation Models

- *Why it matters:*
  - Diffusion models are reshaping creativity, art, and design.
  - From AI avatars to realistic landscapes, they have endless applications.
  - You'll gain insight into how these models work — and how to use them yourself!

Learning Outcomes

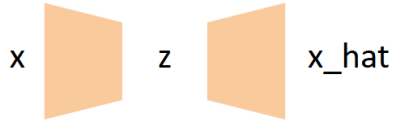
After this session, you will be able to:

- Describe the principles behind latent space diffusion (e.g., Stable Diffusion).
- Summarize score-based diffusion models and their significance.
- Explain PDE-based approaches to diffusion modeling.
- Outline methods for fine-tuning diffusion models for specific tasks.
- Utilize text-conditioned diffusion models for controlled image generation.
- Apply classifier-free guidance to steer diffusion model outputs.

Advanced Diffusion Models: **Latent Space Diffusion (Stable Diffusion)**

Latent Diffusion Models

- Pixel-space diffusion models are computationally expensive.
- Can we learn a compressed latent space, similar to approaches in autoregressive models (e.g., VQGAN)?
- Solution: Train a Variational Autoencoder (VAE) with a very low weighting on the KL divergence loss (e.g., 10^{-6}) to obtain a compact latent representation.



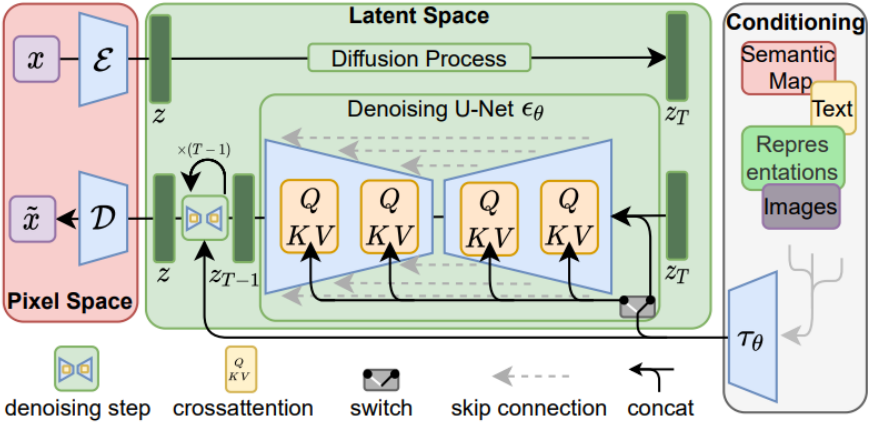
Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

Latent Diffusion Models (cont.)

Key Concepts

- **Latent Diffusion Models (LDMs):** Operate in a compressed latent space for efficiency.
- **Diffusion Process:** Gradually adds noise to data, then learns to reverse this process.
- **Text Conditioning:** Uses CLIP text encoders to guide image generation from prompts.

Latent Diffusion Models (cont.)



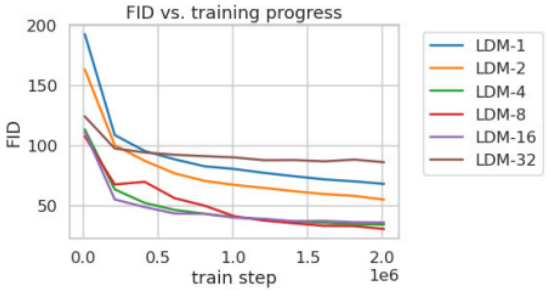
Latent Diffusion Models (cont.)

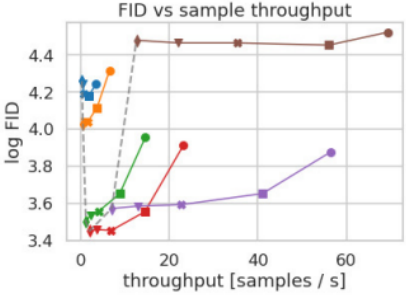
Architecture Overview

- **Encoder:** Maps images to latent space.
- **UNet:** Core denoising network operating in latent space.
- **Decoder:** Reconstructs images from latent representations.
- **Text Encoder:** CLIP-based, encodes prompts for conditioning.

Latent Diffusion Models (cont.)

Speed - Quality Trade-off





What is Stable Diffusion?

- A state-of-the-art text-to-image generative model.
- Developed by Stability AI and collaborators (2022).
- Open-source, enabling wide adoption and customization.

Stable Diffusion (cont.)

- VAE that downsamples images by 8x spatially (e.g., 256 $\rightarrow$ 32, 512 $\rightarrow$ 64).
- UNet architecture with self-attention layers.
- Text conditioning via cross-attention on text features.
- Text features extracted using a CLIP text encoder.

Stable Diffusion (cont.)

Advantages

- High-quality, diverse image generation.
- Efficient training and inference due to latent space operations.
- Open-source and extensible for research and applications.

Applications

- Art and creative design.
- Data augmentation.
- Prototyping and visualization.

Stable Diffusion (cont.)

Recent Advances and Ecosystem

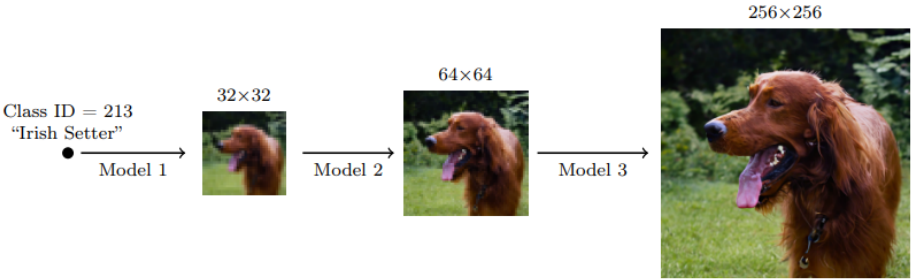
- **Stable Diffusion XL (SDXL):** Improved fidelity and prompt adherence.
- **ControlNet:** Fine-grained control over generation (e.g., pose, edges).
- **Community Models:** Custom fine-tuned models for specific styles/domains.

Stable Video Diffusion

- Initialize from Stable Diffusion.
- Add temporal layers (e.g., 1D Conv, temporal attention).
- Finetune on video data.
- Data pipeline:
  - Scrape **unlabeled** videos from the internet.
  - Use image and video captioners to generate synthetic text labels.
  - Apply aggressive data filtering using several heuristics (CLIP score, optical flow score, aesthetic scores, OCR, etc.).

Stable Video Diffusion (cont.)

Cascaded Diffusion Models



Cascaded Diffusion Models (cont.)

$$p_\theta(x_0) = \int p_\theta(z_1) p_\theta(z_0 \mid z_1) p_\theta(x_0 \mid z_0) dz$$

Ho, Jonathan, et al. "Cascaded diffusion models for high fidelity image generation." The Journal of Machine Learning Research 23.1 (2022): 2249-2281.

Cascaded Diffusion Models (cont.)

- Train one unconditional model; the remaining models are low-resolution conditional super-resolution models.
- Condition on the low-resolution image, perform bilinear or bicubic upsampling, and concatenate the result to the noised input.
- Models can be trained independently.
- Hyperparameters can be tuned specifically for each resolution.

Cascaded Diffusion Models (cont.)

Conditioning Augmentation: Augment the low-res input z

- Method 1: Blurring
  - Apply a Gaussian blurring filter
  - Effective for models at 128, 256 resolution
- Method 2: Non-truncated Conditioning Augmentation
  - Train the super-resolution $p(x|z_0)$ model conditioned on samples from the prior $p(z_0)$ with added corruption from the forward process $q(z_s|z_0)$
  - S (corruption / noise strength) is a hyperparameter
  - Effective for resolutions < 128

Cascaded Diffusion Models (cont.)

(a) 16×16 base

(b) 16×16→64×64 super-resolution, $s = 0$

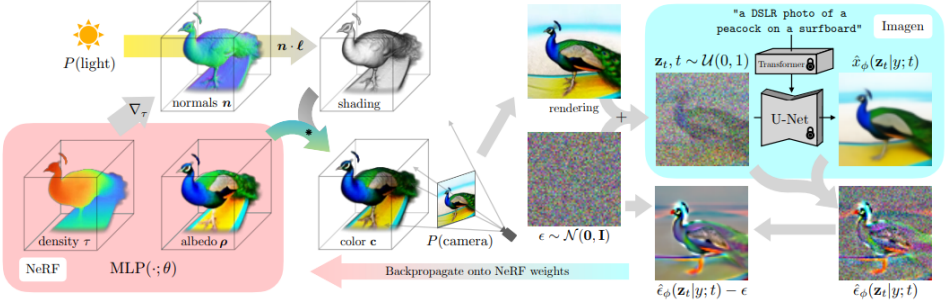
Cascaded Diffusion Models (cont.)

Large improvements in using conditioning augmentation

Conditioning	FID vs train	FID vs validation	IS
No cascading	2.35	2.91	52.72 ± 1.15
$16\times16 \rightarrow 64\times64$ cascading			
$s = 0$	6.02	5.84	35.59 ± 1.19
$s = 101$	3.41	3.67	44.72 ± 1.12
$s = 1001$	**2.13**	**2.79**	$\mathbf{54.47 \pm 1.05}$

Advanced Diffusion Models: **Score-Based Diffusion Models**

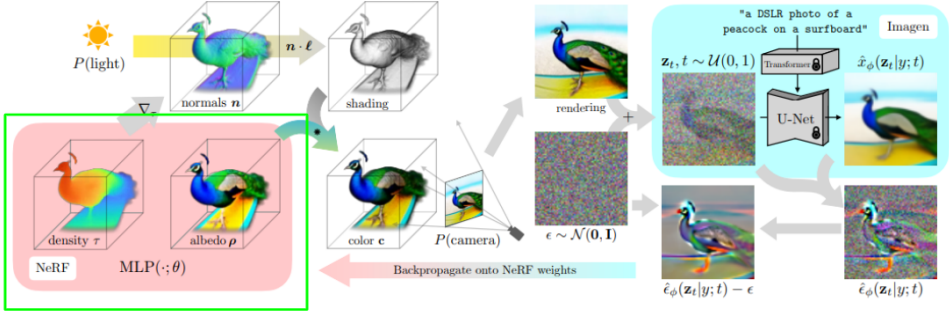
How can you use a pretrained text-image model to generate 3D objects?



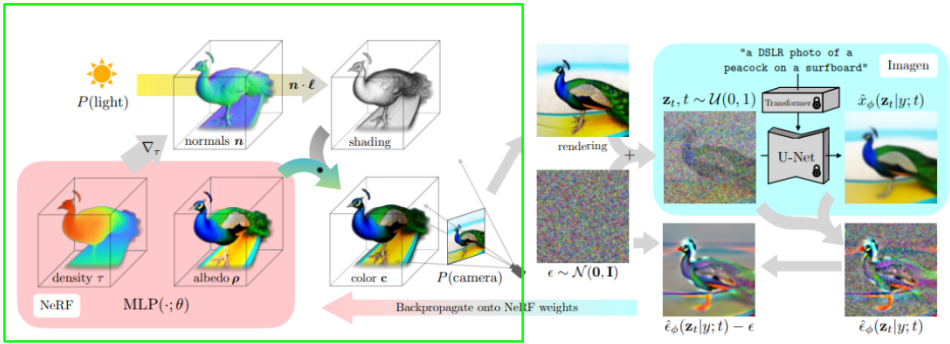
DreamFusion: Text-to-3D Generation (cont.)

Poole, Ben, et al. "Dreamfusion: Text-to-3d using 2d diffusion." arXiv preprint arXiv:2209.14988 (2022).

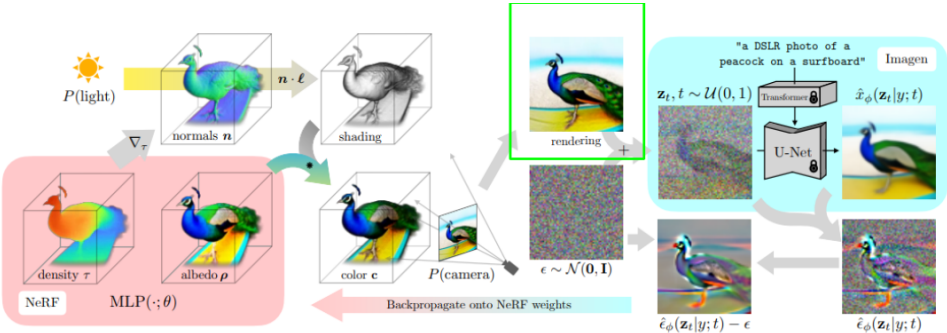
Start with a randomly initialized parameterized 3D representation (NeRF)



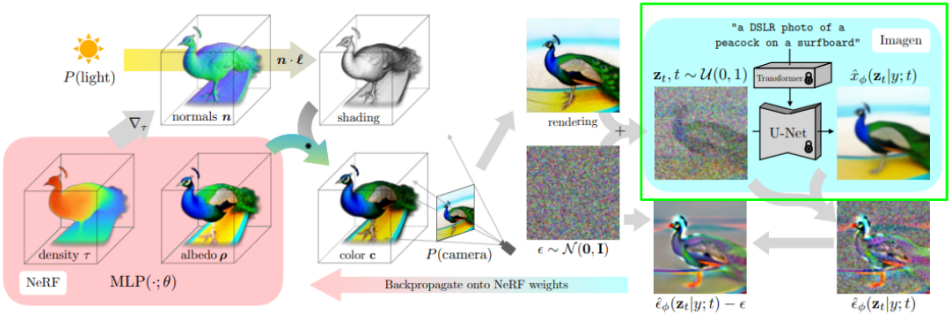
Randomly sample camera angle and lighting



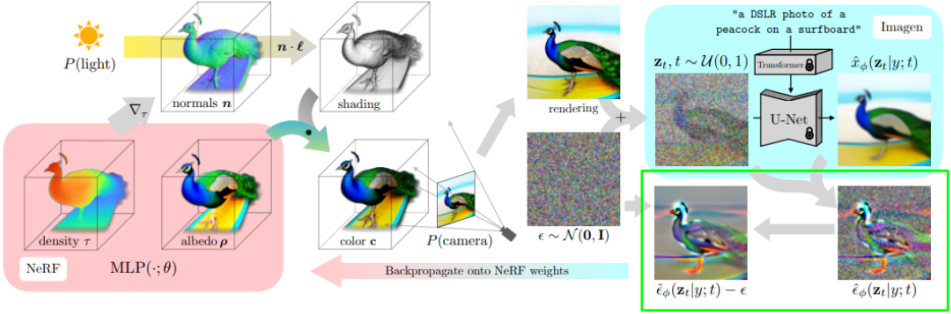
Render the image (64x64)



Noise the image and feed it through the diffusion loss to predict the noise



Compute the SDS loss and backpropagate



Intuitively, we want the NeRF parameters to produce images that minimize the diffusion loss

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

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

But the objective is generally brittle / expensive (backprop through the UNet)

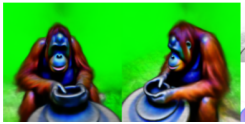
It works better to just remove the U-Net jacobian

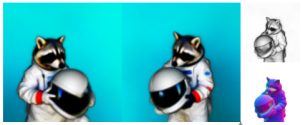
$$\nabla_\theta \mathcal{L}_{\text{Diff}}(\phi, \mathbf{x} = g(\theta)) = \mathbb{E}_{t,\epsilon}\left[w(t) \underbrace{(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon)}_{\text{Noise Residual}} \underbrace{\frac{\partial \hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\mathbf{z}_t}}_{\text{U-Net Jacobian}}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

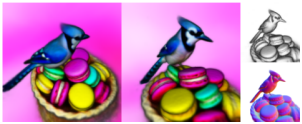
$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon}\left[w(t) (\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \right]$$

There is a theoretical motivation (see Appendix in paper)

DreamFusion: Text-to-3D Generation (cont.)


an orangutan making a clay bowl on a throwing wheel*

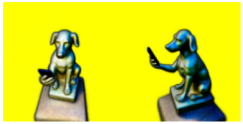

a raccoon astronaut holding his helmet†


a blue jay standing on a large basket of rainbow macarons*


a corgi taking a selfie*


a table with dim sum on it†


a lion reading the newspaper*


Michelangelo style statue of dog reading news on a cellphone


a tiger dressed as a doctor*


a steam engine train, high resolution*


a frog wearing a sweater*


a humanoid robot playing the cello*

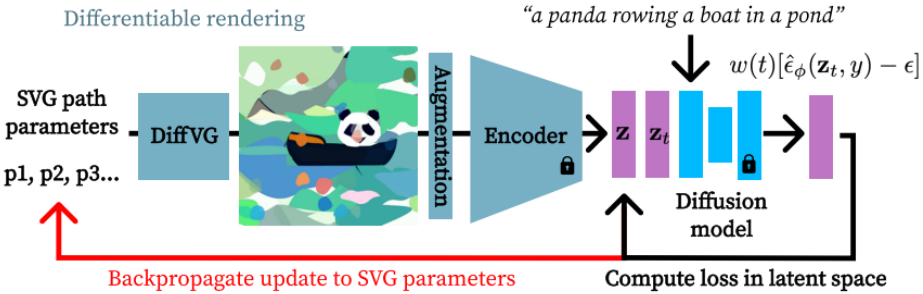

Sydney opera house, aerial view†

Text-to-SVG

- Text-to-SVG generation is a novel application of diffusion models, enabling the creation of vector graphics from textual descriptions.
- The process involves training a diffusion model to learn the distribution of SVG images conditioned on text prompts.
- The model generates SVG paths, which are inherently vector-based and scalable without loss of quality.

Text-to-SVG (cont.)

- Using SDS loss is not limited to text-to-3D.
- SDS provides a general way to backpropagate and learn through any differentiable "rendering" pipeline.



Text-to-SVG (cont.)

Jain, Ajay, Amber Xie, and Pieter Abbeel. "Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

Text-to-SVG (cont.)

a train*

an owl standing on a wire*

Underwater Submarine*

a boat*

A photo of a Ming Dynasty vase on a leather topped table.*

A smiling sloth wearing a leather jacket, a cowboy hat and a kilt.*

a tuba with red flowers protruding from its bell*

a blue poison dart frog sitting on a water lily*

DDPMs as Energy-Based Models (EBMs)

$$p_\theta(\tau) \propto e^{-E_\theta(\tau)}$$

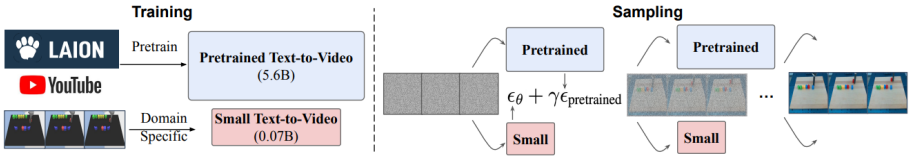
Since the denoising function (ϵ) models the score, we have

$$\epsilon_\theta(\tau^t, t) = \nabla_\tau E_\theta(\tau^t)$$

Learn a distribution as a product of a pretrained (large) model and domain specific (small) model

$$p_{\text{product}}(\tau|\text{text}) \propto \underbrace{p_{\text{pretrained}}(\tau|\text{text})}_{\text{Pretrained Prior}} \underbrace{p_\theta(\tau|\text{text})}_{\text{Video Model}}.$$

Product Distribution



Under the product assumption, we can use the EBM interpretation to combine the scores of the diffusion models

$$p_{product}(\tau|\text{text}) \propto e^{-E_1(\tau)} e^{-E_2(\tau)} = e^{-(E_1(\tau)+E_2(\tau))}$$

$$\nabla_\tau E_\theta(\tau) = -(\nabla E_1(\tau) + \nabla E_2(\tau))$$

VideoAdapter (cont.)

Algorithm 1 Sampling algorithm of Video Adapter

Input: Pretrained Text-to-Video Model $\epsilon_{\text{pretrained}}(\tau, t|\text{text})$, Inverse Temperature α, Prior strength γ.
Initialize sample $\tau_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, \ldots, 1$ **do**
  $\tilde{\epsilon}_{\text{text}} \leftarrow \epsilon_\theta(\tau_t, t|\text{text}) + \gamma\epsilon_{\text{pretrained}}(\tau_t, t|\text{text})$ // compute score using text-conditioned prior
  $\epsilon \leftarrow \epsilon_\theta(\tau_t, t)$ // compute unconditional score
  $\tilde{\epsilon}_{\text{cfg}} \leftarrow \epsilon + \alpha(\tilde{\epsilon}_{\text{text}} - \epsilon)$ // compute classifier guidance score score
  $\tau_{t-1} = \texttt{ddpm\_sample}(\tau_t, \tilde{\epsilon}_{\text{cfg}})$ // run diffusion sampling (can use other samplers)
end for

VideoAdapter (cont.)

	Bridge			Ego4D		
Model	FVD ↓	FID ↓	Param (B)↓	FVD ↓	IS ↑	Param (B) ↓
Small (S)	186.8	38.8	0.07	228.3	2.28	0.07
Small (S) + Pretrained	**177.4**	**37.6**	0.07	156.3	2.82	0.07
Small (L)	152.5	30.1	0.14	65.1	3.31	2.8
Small (L) + Pretrained	**148.1**	**29.5**	0.14	**52.5**	**3.53**	2.8
Pretrained	350.1	42.6	5.6	91.7	3.12	5.6

VideoAdapter (cont.)

Can adapt to different domains (robot trajectories, ego-centric camera)

Link: GIFF

Link: GIFF

Advanced Diffusion Models: **PDE-based Diffusion Models**

PDE-based Diffusion Models

- Leverage diffusion models, originally for image generation, to solve and analyze PDEs.
- Learn complex PDE relationships by adding noise to solutions and reversing the process.
- Effective for handling incomplete or uncertain data.
- Valuable in scientific and engineering applications.

Core Idea

- Diffusion models transform data into a simple distribution (e.g., Gaussian noise) via a forward process.
- A reverse process is learned to reconstruct the original data from noise.
- For PDEs, the model maps input or boundary conditions to PDE solutions, capturing the equation's dynamics.

Why Use Diffusion Models for PDEs?

- **Generative Solutions:** Can produce multiple plausible PDE solutions, useful under uncertainty or incomplete information.
- **Uncertainty Modeling:** Capture a distribution of possible solutions, not just a single answer.
- **Partial Data Learning:** Train effectively on datasets with missing or incomplete data, suitable for real-world scenarios.
- **Flexibility:** Adaptable to both forward (solving PDEs) and inverse (parameter estimation) problems.

The diffusion equation is a fundamental PDE describing the distribution of a quantity (such as heat or particles) diffusing through a medium:

$$\frac{\partial u}{\partial t} = D\nabla^2 u \tag{1}$$

where:

- $u = u(\mathbf{x}, t)$ is the quantity of interest (e.g., concentration, temperature)
- D is the diffusion coefficient
- ∇^2 is the Laplacian operator

DiffusionPDE (cont.)

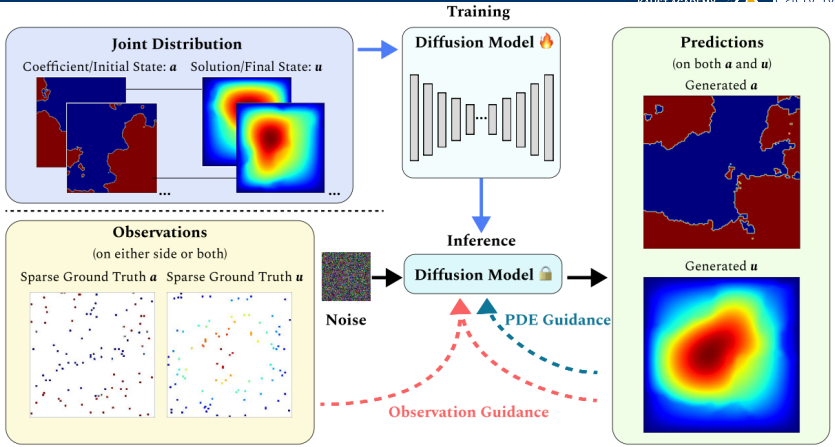


Figure 1: We propose DiffusionPDE, a generative PDE solver under partial observations. Given a family of PDE with coefficient (initial state) a and solution (final state) u, we train the diffusion model on the joint distribution of a and u. During inference, we gradually denoise a Gaussian noise, guided by sparse observation and known PDE function, to recover the full prediction of both a and u that align well with the sparse observations and the given equation.

Advanced Diffusion Models: **Fine-Tuning**

Fine-Tuning Diffusion Models

- Fine-tuning lets you teach a diffusion model new things using your own images.
- You can make the model generate pictures in your style or for your special needs.
- Just use a few of your own photos—like products, anime, or anything unique to you.
- This way, the model becomes more useful and personal for your projects.

Why Fine-Tune?

- **Customization:** Generate images that reflect your brand, style, or unique requirements.
- **Personalization:** Learn your own visual style, such as custom portraits, branding elements, or new characters.
- **Domain Adaptation:** Improve performance on niche or underrepresented domains.

Popular Fine-Tuning Methods

- **DreamBooth:**
  - Enables the model to learn new concepts from a small set of images.
  - Useful for injecting specific subjects or styles into the model.
  - Lightweight and efficient—requires only a few images and limited compute.
- **LoRA (Low-Rank Adaptation):**
  - Introduces trainable low-rank matrices into the model.
  - Allows efficient fine-tuning with minimal changes to the original weights.
  - Reduces memory and compute requirements.

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

Nataniel Ruiz Yuanzhen Li Varun Jampani Yael Pritch Michael Rubinstein Kfir Aberman

Google Research

It's like a photo booth, but once the subject is captured, it can be synthesized wherever your dreams take you…

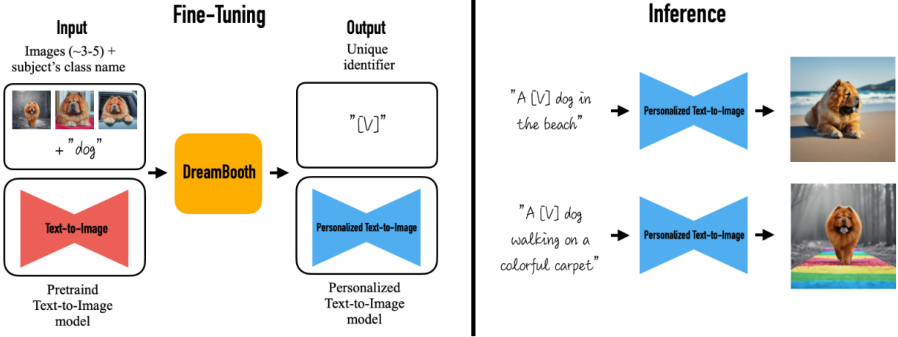
What is DreamBooth?

- A method for fine-tuning large diffusion models on a small number of images.
- Developed by Google Research (2022).
- Allows personalization of generative models with minimal data.

Key Concepts

- **Personalization:** Adapts a pre-trained model to generate images of specific subjects (e.g., pets, people).
- **Fine-tuning:** Uses a small set of images to adjust the model's parameters.
- **Text Conditioning:** Leverages text prompts to guide image generation.

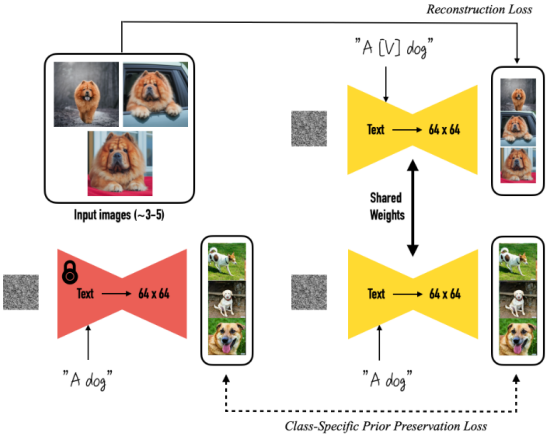
DreamBooth (cont.)

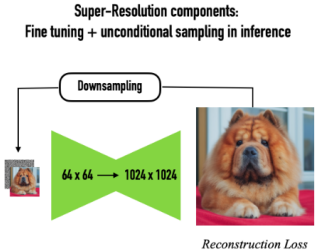


How Does DreamBooth Personalize Models?

- DreamBooth teaches a model to recognize a new subject using just a few photos.
- After training, the model can create new images of that subject in different places or situations.
- This is called “few-shot” learning—using only a few examples!

DreamBooth (cont.)





How Do We Tell the Model About the New Subject?

- We use a special made-up word (called a *rare-token*) to represent the new subject.
- For example: "a photo of `sks` dog at the beach".
- The model learns to link this rare-token to the subject in your photos.

Why Use Rare-tokens?

- Rare-tokens are unique and not found in the model's original training data.
- This avoids mixing up your subject with things the model already knows.

How Does DreamBooth Avoid Forgetting?

- If we only train on your subject, the model might forget how to make other images.
- To prevent this, we also show it regular class prompts (like "a photo of a dog").
- We use a special loss to balance learning the new subject and keeping old knowledge:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{subject}} + \lambda \mathcal{L}_{\text{prior}}$$

where:

- $\mathcal{L}_{\text{subject}}$ = loss for your subject (rare-token prompt)
- $\mathcal{L}_{\text{prior}}$ = loss for general class (class prompt)
- λ = weight to balance the two

DreamBooth: Results

[Ruiz et al., 2023](https://arxiv.org/abs/2208.12242)

Art Rendition

Original artistic renditions of our subject dog in the style of famous painters. We remark that many of the generated poses were not seen in the training set, such as the Van Gogh and Warhol rendition. We also note that some renditions seem to have novel composition and faithfully imitate the style of the painter - even suggesting some sort of creativity (extrapolation given previous knowledge).

Input images

Vincent Van Gogh | Michelangelo | Rembrandt

Johannes Vermeer | Pierre-Auguste Renoir | Leonardo da Vinci

[Ruiz et al., 2023](https://arxiv.org/abs/2208.12242)

LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu

LoRA Overview

Low-Rank Adaptation (LoRA) is a technique for efficient fine-tuning of large neural networks. Instead of updating all parameters, LoRA injects trainable low-rank matrices into each layer, significantly reducing the number of trainable parameters.

Key Idea: Decompose the weight update ΔW as a product of two low-rank matrices:

$$\Delta W = AB \tag{2}$$

where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$, with $r \ll \min(d, k)$.

Low-Rank Adaptation (LoRA) (cont.)

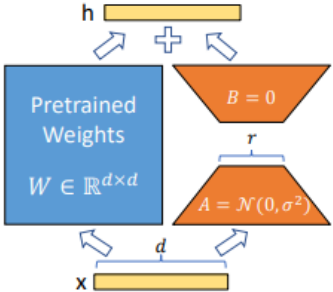


Figure 1: Our reparametrization. We only train A and B.

[Hu et al., 2021](https://arxiv.org/abs/2106.09685)

LoRA in Linear Layers

Consider a linear layer with weight $W_0 \in \mathbb{R}^{d \times k}$. In LoRA, the forward pass is modified as:

$$y = W_0 x + \alpha A B x \tag{3}$$

where α is a scaling factor, and only A and B are updated during fine-tuning.

Parameter Efficiency: The number of trainable parameters is reduced from $d \times k$ to $r \times (d + k)$.

LoRA Training Objective

The training objective remains the same as standard fine-tuning, e.g., minimizing a loss function $\mathcal{L}$:

$$\min_{A,B} \mathcal{L}(y, \hat{y}) \tag{4}$$

where y is the model output and $\hat{y}$ is the ground truth.

Regularization: Optionally, regularization terms can be added to encourage low-rank structure or prevent overfitting.

LoRA in Attention Mechanisms

LoRA is commonly applied to the query and value projection matrices in attention layers:

$$Q' = Q + \Delta Q = Q + A_q B_q \tag{5}$$

$$V' = V + \Delta V = V + A_v B_v \tag{6}$$

where A_q, B_q, A_v, B_v are low-rank matrices for the query and value projections, respectively.

Low-Rank Adaptation (LoRA) (cont.)

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	**95.1**$_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	**93.3**$_{\pm.3}$	$90.8_{\pm.1}$	**86.6**$_{\pm.7}$	**91.5**$_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	**90.6**$_{\pm.2}$	$96.2_{\pm.5}$	**90.9**$_{\pm1.2}$	**68.2**$_{\pm1.9}$	**94.9**$_{\pm.3}$	$91.6_{\pm.1}$	**87.4**$_{\pm2.5}$	**92.6**$_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	**68.3**$_{\pm1.0}$	**94.8**$_{\pm.2}$	**91.9**$_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	**90.5**$_{\pm.3}$	**96.6**$_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	**94.8**$_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	**90.6**$_{\pm.2}$	$96.2_{\pm.5}$	**90.2**$_{\pm1.0}$	$68.2_{\pm1.9}$	**94.8**$_{\pm.3}$	$91.6_{\pm.2}$	**85.2**$_{\pm1.1}$	**92.3**$_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	**91.9**$_{\pm.2}$	$96.9_{\pm.2}$	**92.6**$_{\pm.6}$	**72.4**$_{\pm1.1}$	**96.0**$_{\pm.1}$	**92.9**$_{\pm.1}$	**94.9**$_{\pm.4}$	**93.0**$_{\pm.2}$	**91.3**

Table 2: $\text{RoBERTa}_{\text{base}}$, $\text{RoBERTa}_{\text{large}}$, and $\text{DeBERTa}_{\text{XXL}}$ with different adaptation methods on the GLUE benchmark. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics. * indicates numbers published in prior works. † indicates runs configured in a setup similar to Houlsby et al. (2019) for a fair comparison.

[Hu et al., 2021](https://arxiv.org/abs/2106.09685)

Model & Method	# Trainable Parameters	E2E NLG Challenge BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter^{L})*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter^{L})*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter^{H})	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
GPT-2 L (Adapter^{L})	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

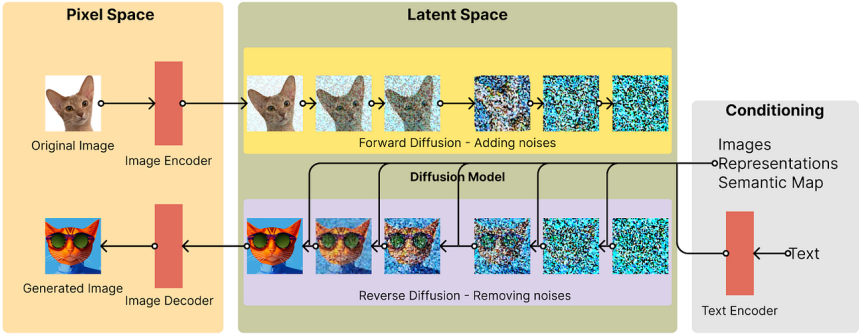
Table 3: GPT-2 medium (M) and large (L) with different adaptation methods on the E2E NLG Challenge. For all metrics, higher is better. LoRA outperforms several baselines with comparable or fewer trainable parameters. Confidence intervals are shown for experiments we ran. * indicates numbers published in prior works.

Applications of Fine-Tuning

- **Portrait Generation:** Create personalized avatars or stylized portraits.
- **Branding:** Generate images consistent with your brand's visual identity.
- **Character Design:** Develop new characters for games, comics, or animation.
- **Product Imagery:** Produce custom product images for marketing or e-commerce.

Advanced Diffusion Models:
Text-Conditioned

Text-Conditioned Diffusion Models (cont.)



Application: GLIDE - OpenAI

"a man wearing a white hat"

Image Inpainting with GLIDE

[0] GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

Application: DALL.E 2 - OpenAI

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

Text to image generation eith DAll.E 2

[0] Hierarchical Text-Conditional Image Generation with CLIP Latents

Fix the CLIP embedding $\mathbf{z}$.
Decode using different decoder latents $\mathbf{x}_T$.

Image Variations

[0]Hierarchical Text-Conditional Image Generation with CLIP Latents

Interpolate image CLIP embeddings $\mathbf{z}_i$

Use different $\mathbf{x}_T$ to get different interpolation trajectories.

Image interpolation

[0]Hierarchical Text-Conditional Image Generation with CLIP Latents

Change the image CLIP embedding towards the difference of the text CLIP embeddings of two prompts.

Decoder latent is kept as a constant.

Text Difference Image interpolation

[0] Hierarchical Text-Conditional Image Generation with CLIP Latents

Imagen - Google

A brain riding a rocketship heading towards the moon.

[0]Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Imagen - Google

A brain riding a rocketship heading towards the moon.

[0]Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Imagen - Google

A dragon fruit wearing karate belt in the snow.

[0] Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Imagen - Google

A relaxed garlic with a blindfold reading a newspaper while floating in a pool of tomato soup.

[0] Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Advanced Diffusion Models: **Classifier-Free Guidance**

What is Classifier-Free Guidance?

- It's a trick used in image generation models (like diffusion models).
- Lets us control how much the model listens to our prompt (like a text description) without needing a separate classifier.
- Makes the model more flexible and easier to use.

CLASSIFIER-FREE DIFFUSION GUIDANCE

Jonathan Ho & Tim Salimans
Google Research, Brain team
{jonathanho,salimans}@google.com

ABSTRACT

Classifier guidance is a recently introduced method to trade off mode coverage and sample fidelity in conditional diffusion models post training, in the same spirit as low temperature sampling or truncation in other types of generative models. Classifier guidance combines the score estimate of a diffusion model with the gradient of an image classifier and thereby requires training an image classifier separate from the diffusion model. It also raises the question of whether guidance can be performed without a classifier. We show that guidance can be indeed performed by a pure generative model without such a classifier: in what we call classifier-free guidance, we jointly train a conditional and an unconditional diffusion model, and we combine the resulting conditional and unconditional score estimates to attain a trade-off between sample quality and diversity similar to that obtained using classifier guidance.

Classifier-Free Guidance (cont.)

[Ho et al., 2022](https://arxiv.org/abs/2207.12598)

How does it work?

- The model is trained in two ways:
  - **With prompt:** The model gets a text prompt and learns to generate images that match it.
  - **Without prompt:** The model ignores the prompt and just generates images freely.
- At generation time, we ask the model for both versions:
  - One with the prompt (conditional).
  - One without the prompt (unconditional).
- We mix these two outputs to control how much the prompt matters.

The math (don't worry, it's simple!):

$$\epsilon_{\text{guided}} = \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) + w\left[\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon_\theta(\mathbf{x}_t, t, \varnothing)\right]$$

- $\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})$: Output with the prompt.
- $\epsilon_\theta(\mathbf{x}_t, t, \varnothing)$: Output without the prompt.
- w: How strongly we want to follow the prompt.

What does w do?

- $w = 0$: Ignore the prompt (model is creative, less controlled).
- $w = 1$: Use the prompt as normal.
- $w > 1$: Really focus on the prompt (more faithful to what you asked).

Figure 2: The effect of guidance on a mixture of three Gaussians, each mixture component representing data conditioned on a class. The leftmost plot is the non-guided marginal density. Left to right are densities of mixtures of normalized guided conditionals with increasing guidance strength.

[Ho et al., 2022](https://arxiv.org/abs/2207.12598)

Classifier-Free Guidance (cont.)

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

repeat
  $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
  $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
  $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
  Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model
until converged

[Ho et al., 2022](https://arxiv.org/abs/2207.12598)

Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$

$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = 1, \ldots, T$ **do**
  ▷ Form the classifier-free guided score at log SNR λ_t
  $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
  ▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
  $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t}\tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
  $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v}(\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
end for
return $\mathbf{z}_{T+1}$

[Ho et al., 2022](https://arxiv.org/abs/2207.12598)

Classifier-Free Guidance (cont.)

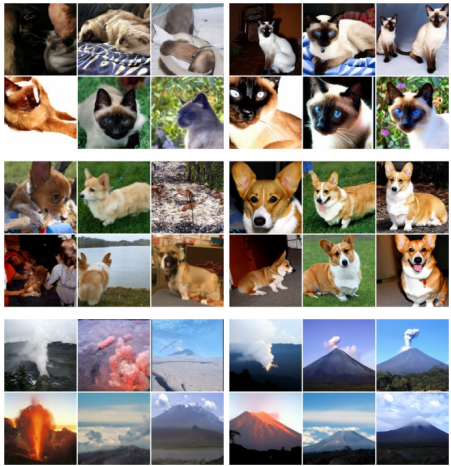


Figure 3: Classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$. Interestingly, strongly guided samples such as these display saturated colors. See Fig. 8 for more.

Classifier-Free Guidance (cont.)

[Ho et al., 2022](https://arxiv.org/abs/2207.12598)

Classifier-Free Guidance (cont.)

Figure 8: More examples of classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$.

[Ho et al., 2022](https://arxiv.org/abs/2207.12598)

Classifier-Free Guidance (cont.)

Advanced Diffusion Models: **Limitations**

Limitations

Challenge	Explanation
Compute cost	Denoising many steps can be slow / costly
Biases & ethics	Models reflect biased training data
Guide artifacts	High guidance can oversharpen or distort
Fine-tune data needs	Requires quality, diverse data
Legal concerns	Generated content may face IP issues

Advanced Diffusion Models: **References**

References

[1] Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684-10695. 2022.

[2] Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-based generative modeling through stochastic differential equations." In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Ruiz, Nataniel, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation." In *arXiv preprint arXiv:2208.12242*, 2022.

References (cont.)

[4] Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models." In *arXiv preprint arXiv:2106.09685*, 2021.

[5] Saharia, Chitwan, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour et al. "Photorealistic text-to-image diffusion models with deep language understanding." In *arXiv preprint arXiv:2205.11487*, 2022.

[6] Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical text-conditional image generation with clip latents." In *arXiv preprint arXiv:2204.06125*, 2022.

[7] Abbeel, Pieter, Wilson Yan, Kevin Frans, and Philipp Wu. "CS294-158 Deep Unsupervised Learning, Lecture 6: Diffusion Models." UC Berkeley, 2022. https://rail.eecs.berkeley.edu/deeprlcourse-fa22/static/notes/lec6.pdf

References (cont.)

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.