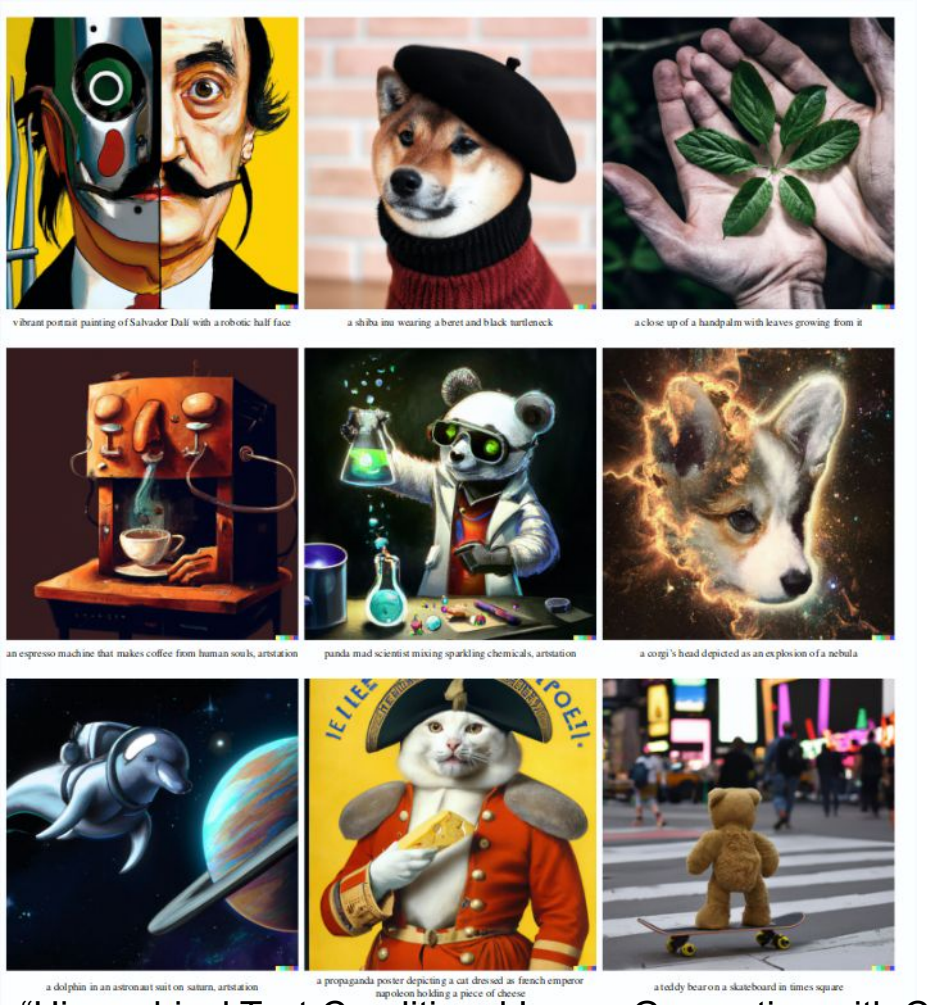
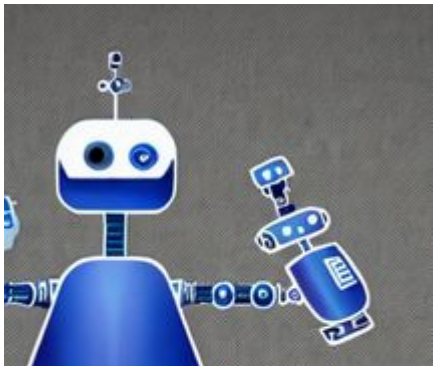


Ramesh et al., “Hierarchical Text-Conditional Image Generation with CLIP Latents.”

Dall-E 2

Ramesh et al., “Hierarchical Text-Conditional Image Generation with CLIP Latents.”

Presenter: Adam Schmidt
adamschmidt@ece.ubc.ca



Overview

-Examples (Images)

-Diffusion Tutorial (how to generate random samples that look like your data!)

-Conditional Diffusion (generate data given common features/classes)

-Conditional Diffusion with CLIP (Dall-E 2–images from text)

-Latent Diffusion (Stable Diffusion–more images from text, but built different)

-Cold Diffusion (do we need noise?)

-Discussion

Alexa is Stealing Your Job, Rhonda

Careers in an AI world

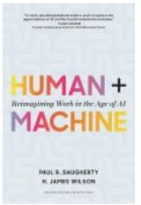
How to Become an Artificial Intelligence Engineer?

3 Best Career Choices for AI Professionals

[PDF] [EPUB] Alexa is Stealing Your Job: The Impac...

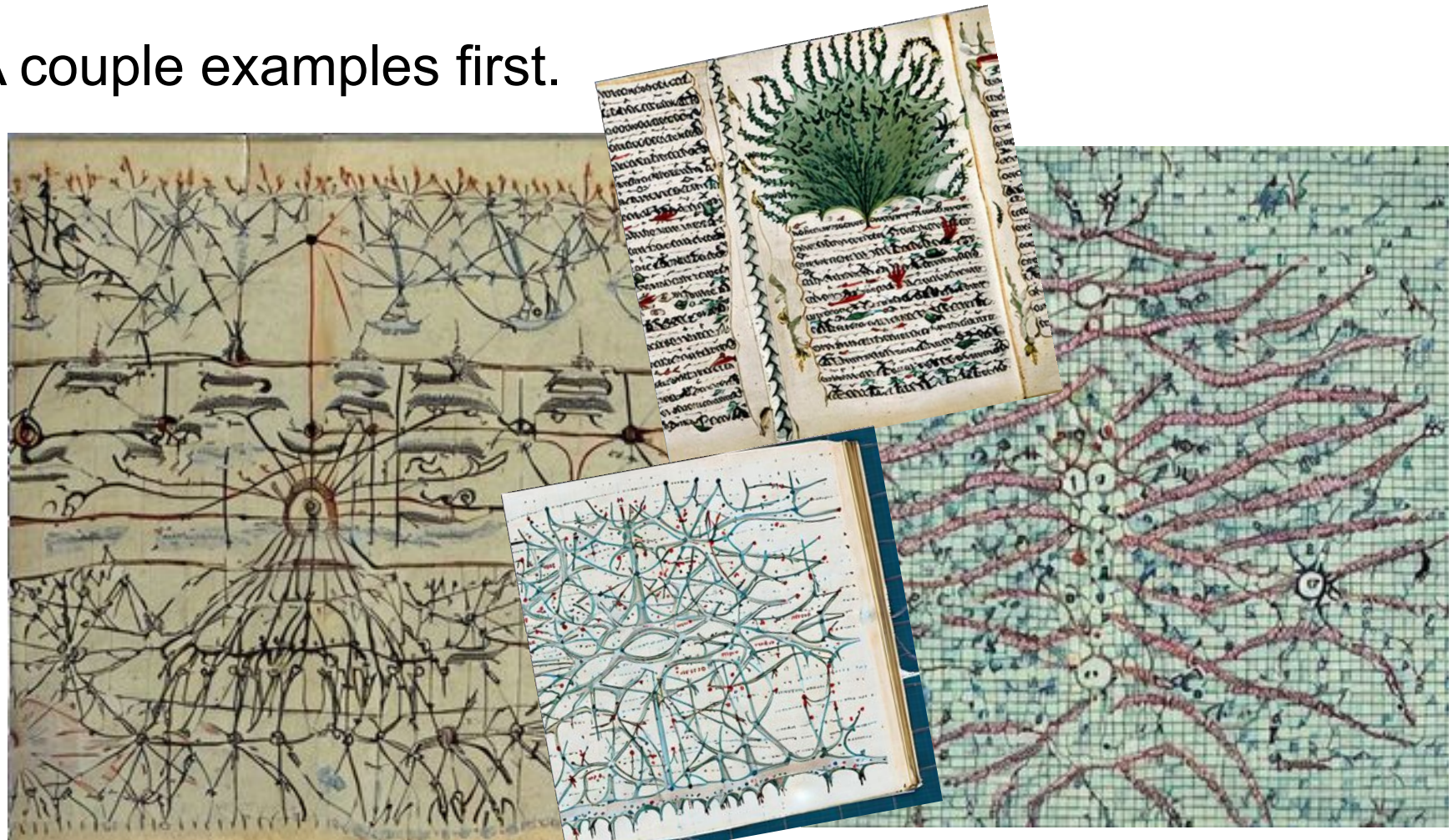

attractive businesswoman holding digital tablet wh...


artificial intellegence



Human + Machine:

A couple examples first.



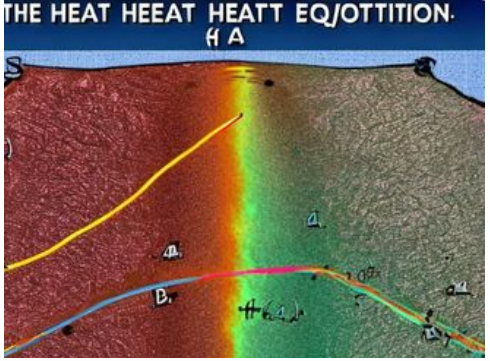
First: Diffusion

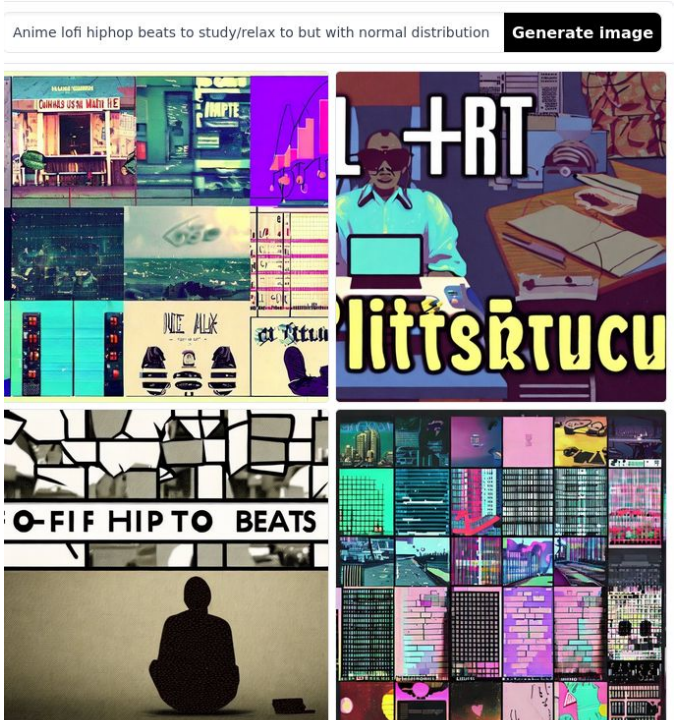
Ways to generate samples from a distribution of noise

Slides from CVPR 2022 tutorial follow.

https://cvpr2022-tutorial-diffusion-models.github.io/

https://drive.google.com/file/d/1DYHDbt1tSl9oqm3O333biRYzSCOtdtmn/view (Arash Vahdat)



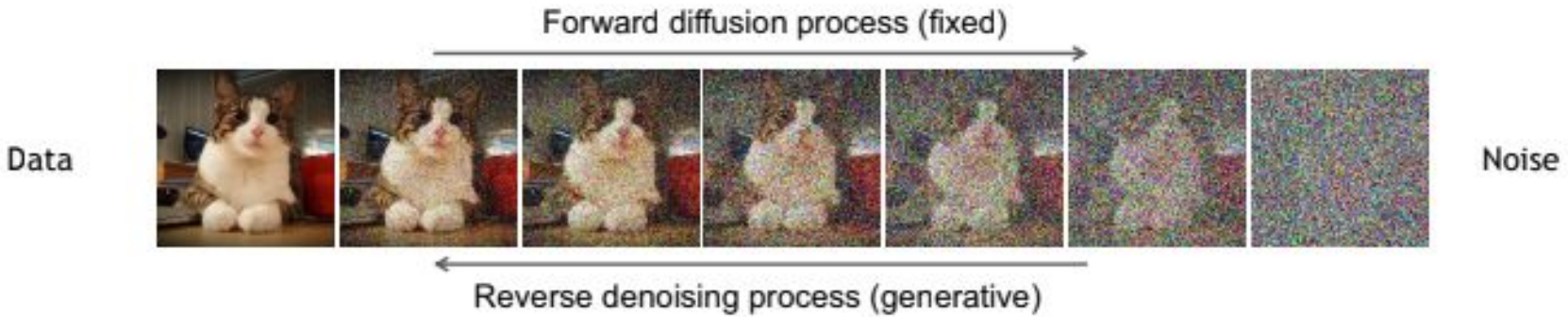


Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising

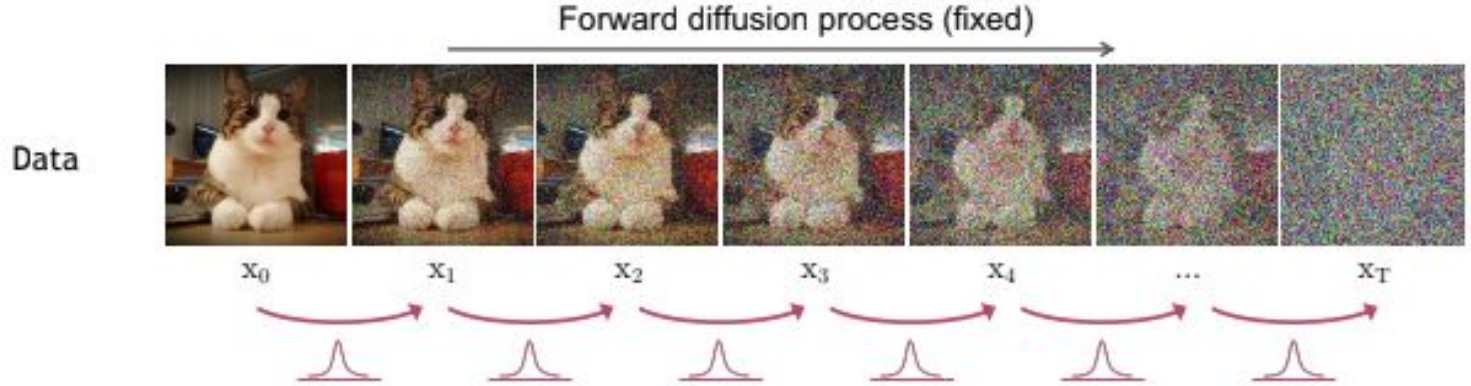


Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process

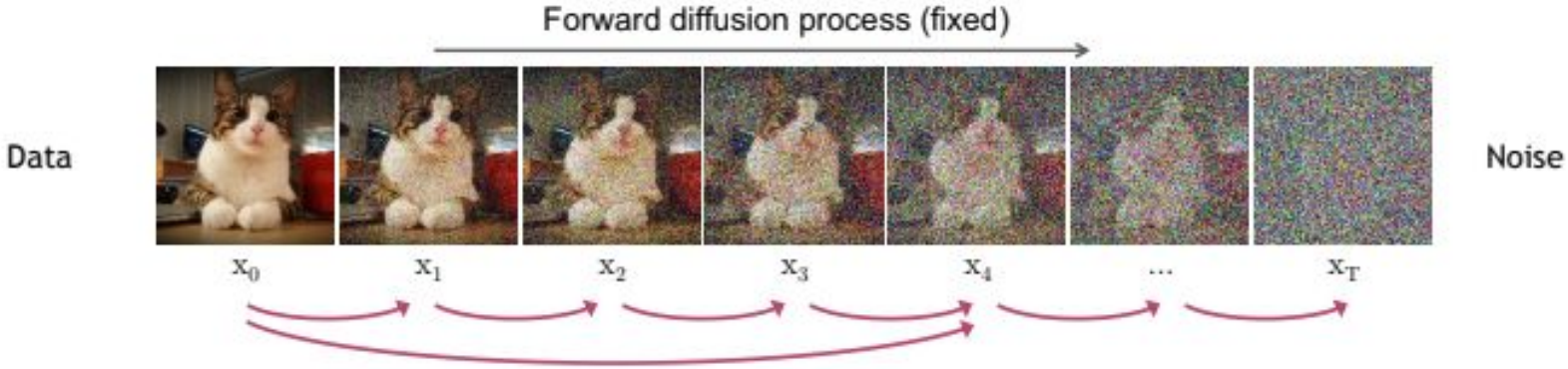
The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

https://cvpr2022-tutorial-diffusion-models.github.io/

Diffusion Kernel



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

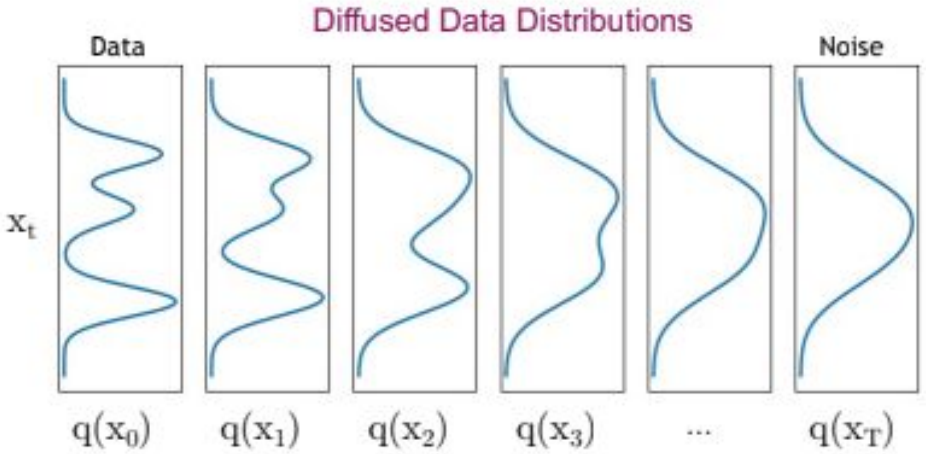
https://cvpr2022-tutorial-diffusion-models.github.io/

What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

https://cvpr2022-tutorial-diffusion-models.github.io/

Generative Learning by Denoising

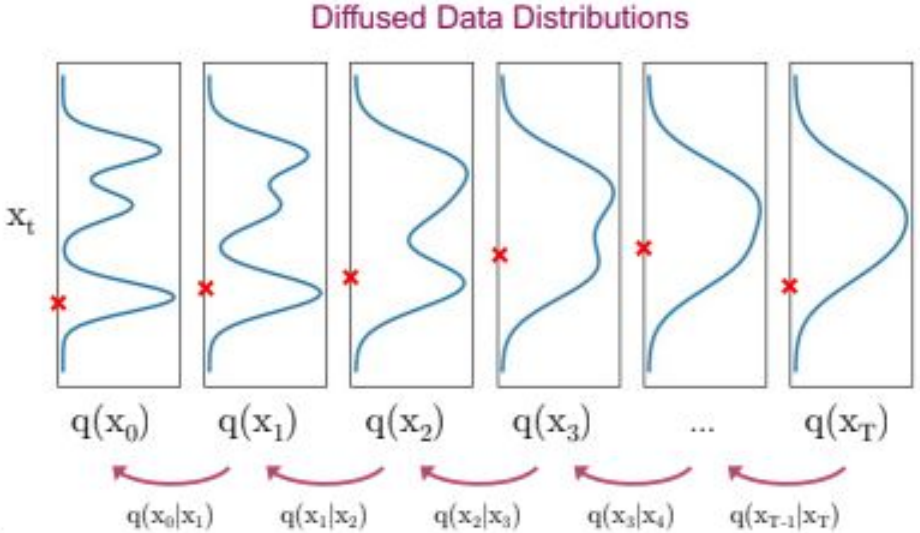
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}$

True Denoising Dist.

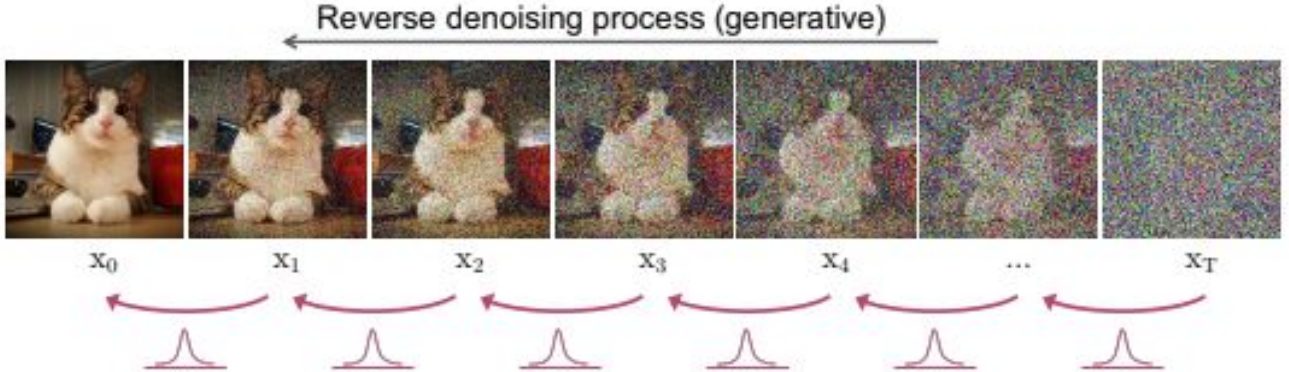


In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

https://cvpr2022-tutorial-diffusion-models.github.io/

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

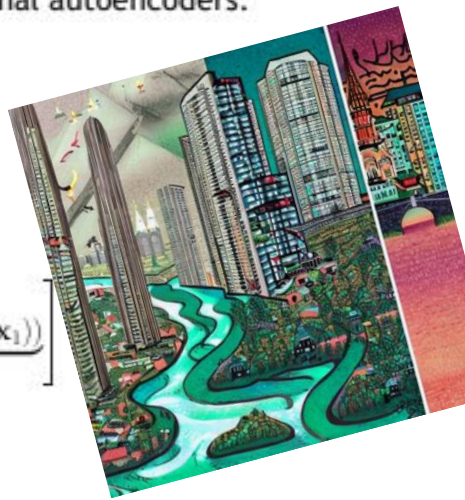
Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is the tractable posterior distribution:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1};\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0),\tilde{\beta}_t\mathbf{I}),$$

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$



https://cvpr2022-tutorial-diffusion-models.github.io/

Sohl-Dickstein et al., “Deep Unsupervised Learning Using Nonequilibrium Thermodynamics.”

Parameterizing the Denoising Model

Since both $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are Normal distributions, the KL divergence has a simple form:

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$. Ho et al. NeurIPS 2020 observe that:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

They propose to represent the mean of the denoising model using a *noise-prediction* network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

With this parameterization

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), \epsilon\sim\mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2\right] + C$$

Training Objective Weighting

Trading likelihood for perceptual quality

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Big[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2 \Big]$$

The time dependent λ_t ensures that the training objective is weighted properly for the maximum data likelihood training.

However, this weight is often very large for small t's.

Ho et al. NeurIPS 2020 observe that simply setting $\lambda_t = 1$ improves sample quality. So, they propose to use:

MAIN TAKEAWAY:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \Big[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2 \Big]$$

For more advanced weighting see Choi et al., Perception Prioritized Training of Diffusion Models, CVPR 2022.

Ho, Jain, and Abbeel, "Denoising Diffusion Probabilistic Models."

Summary

Training and Sample Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

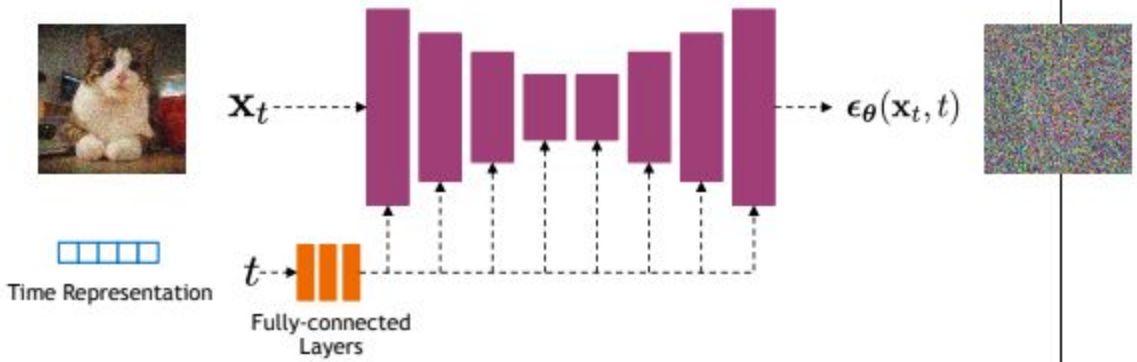
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Implementation Considerations

Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

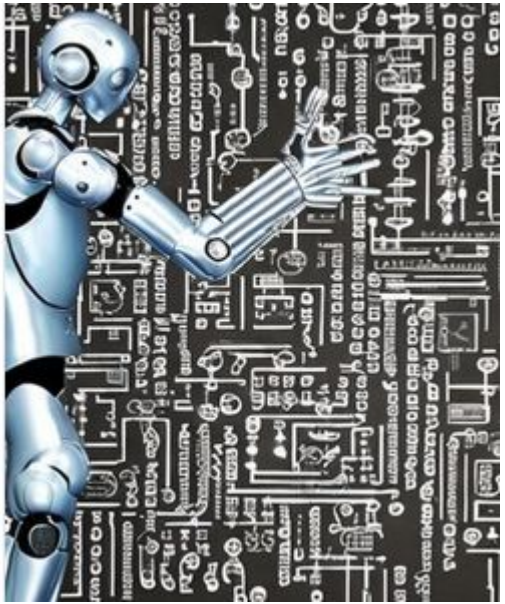
Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dhariwal and Nichol NeurIPS 2021)

Back to scheduled material

We want images from text; how do we make it conditional?

$$\epsilon_\theta(x_t, t, c)$$

We already have time as an input… just tack on a class!



DALL-E 2:

Have:

Large dataset of image/text pairs.

Image,text pairs x, y

z_i, z_t image and text embeddings from CLIP

Want:

Prior $P(z_i|y)$ to get embedding from text

Decoder $P(x|z_i, y)$ to create images given embedding and text caption

Ramesh et al., “Hierarchical Text-Conditional Image Generation with CLIP Latents.”

DALL-E 2 Enables

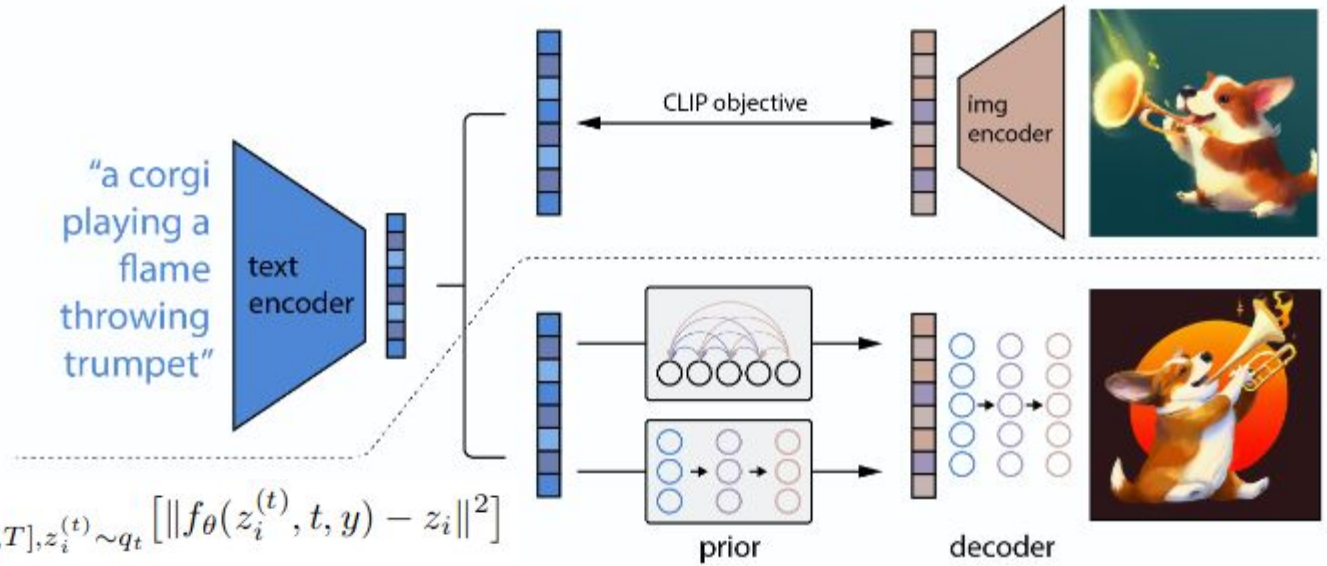
Interpolation of text. z_t

Generating samples with same caption, y

Interpolate images. z_i

Generating similar samples given an image:

Figure 3: Variations of an input image by encoding with CLIP and then decoding with a diffusion model. The variations preserve both semantic information like presence of a clock in the painting and the overlapping strokes in the logo, as well as stylistic elements like the surrealism in the painting and the color gradients in the logo, while varying the non-essential details.



$$L_{\text{prior}} = \mathbb{E}_{t\sim[1,T], z_i^{(t)}\sim q_t}\left[\|f_\theta(z_i^{(t)}, t, y) - z_i\|^2\right]$$

Figure 2: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

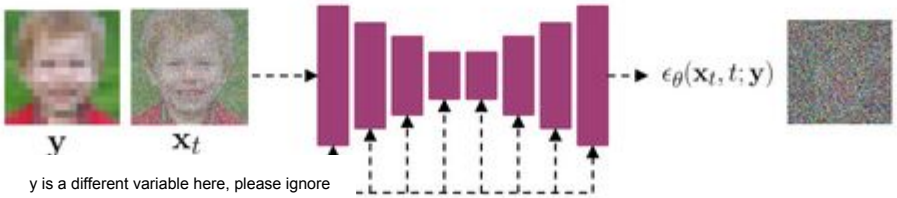
Decoder

-one layer of low resolution image generation 64,64 $P(x|z_i, y)$

Random drop embedding to 0 z_i 10% and text y 50%

-multiple layers of conditional superresolution diffusion models ->256->1024

Superresolution:



Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

https://cvpr2022-tutorial-diffusion-models.github.io/

Interpolating Images, z_i

Figure 4: Variations between two images by interpolating their CLIP image embedding and then decoding with a diffusion model. We fix the decoder seed across each row. The intermediate variations naturally blend the content and style from both input images.

Interpolating Text, z_t

Figure 5: Text diffs applied to images by interpolating between their CLIP image embeddings and a normalised difference of the CLIP text embeddings produced from the two descriptions. We also perform DDIM inversion to perfectly reconstruct the input image in the first column, and fix the decoder DDIM noise across each row.

Some ablation

GLIDE model: just uses text caption

Text embedding (instead of image embedding)

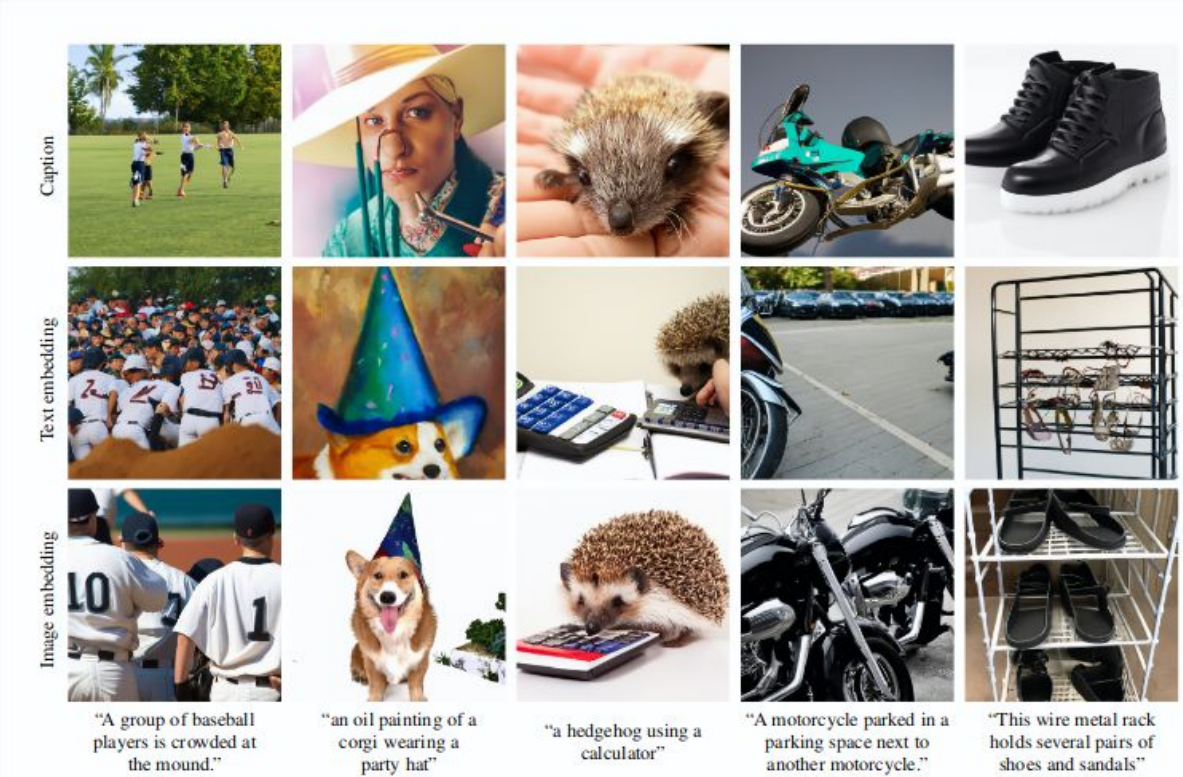


Figure 8: Samples using different conditioning signals for the *same* decoder. In the first row, we pass the te caption to the decoder, and pass a zero vector for the CLIP embedding. In the second row, we pass both th text caption and the CLIP text embedding of the caption. In the third row, we pass the text and a CLIP imag embedding generated by an autoregressive prior for the given caption. Note that this decoder is only traine to do the text-to-image generation task (without the CLIP image representation) 5% of the time.

Reconstruction! Even if probabilities incorrect.

Even if classified as an ipod, it still remembers the apple :’)

Last week–max classification? (there can be both an ipod and an apple)

Granny Smith: 100%
iPod: 0%
Pizza: 0%

Granny Smith: 0.02%
iPod: 99.98%
Pizza: 0%

Granny Smith: 94.33%
iPod: 0%
Pizza: 5.66%

Figure 6: Variations of images featuring typographic attacks [20] paired with the CLIP model’s predicted probabilities across three labels. Surprisingly, the decoder still recovers Granny Smith apples even when the predicted probability for this label is near 0%. We also find that our CLIP model is slightly less susceptible to the “pizza” attack than the models investigated in [20].

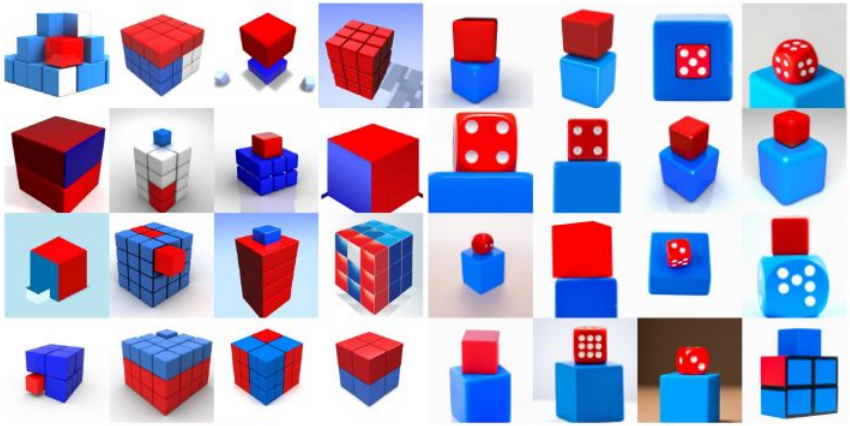
DALL-E 2 Snags

Can mix up colors, context, words.

Figure 15: Reconstructions from the decoder for difficult binding problems. We find that the reconstructions mix up objects and attributes. In the first two examples, the model mixes up the color of two objects. In the rightmost example, the model does not reliably reconstruct the relative size of two objects.

Figure 16: Samples from unCLIP for the prompt, “A sign that says deep learning.”

(a) A high quality photo of a dog playing in a green field next to a lake.



(a) unCLIP (b) GLIDE

Figure 14: Samples from unCLIP and GLIDE for the prompt “a red cube on top of a blue cube”.

Latent Diffusion (Stable Diffusion)

Runs on pixels of latent code

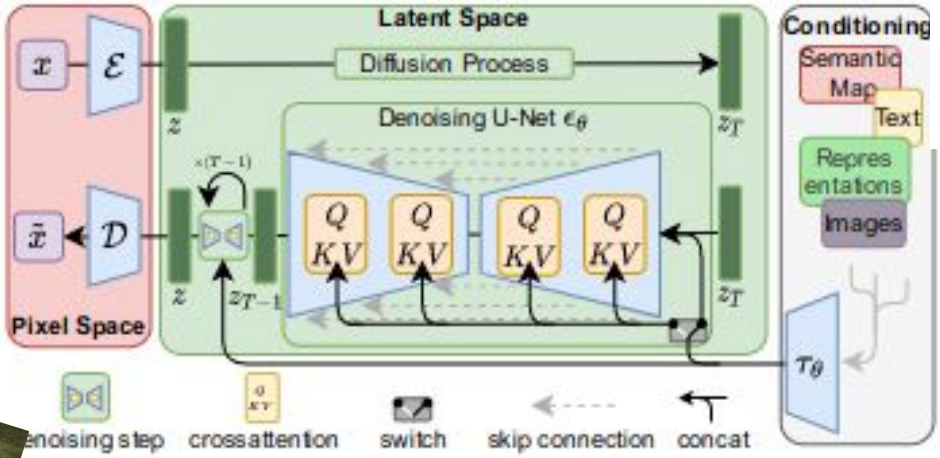
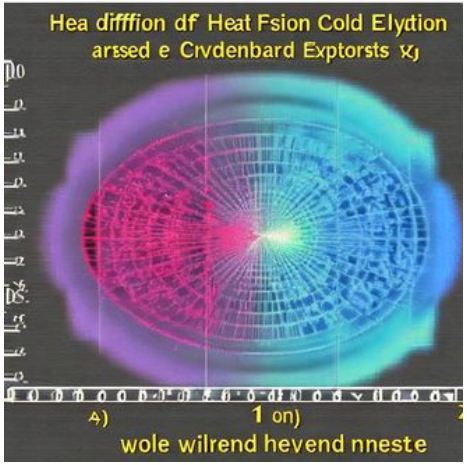
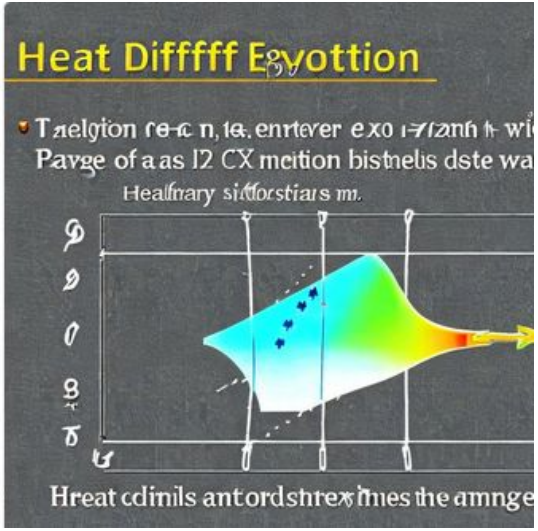


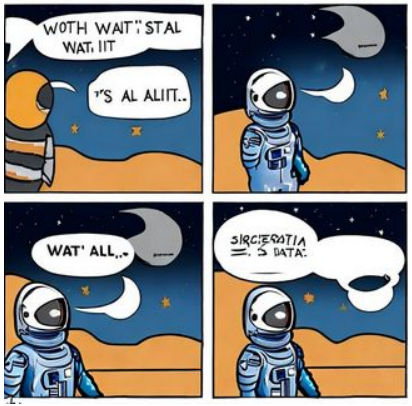
Figure 3. We condition LDMs either via concatenation or by a more general cross-attention mechanism. See Sec. 3.3

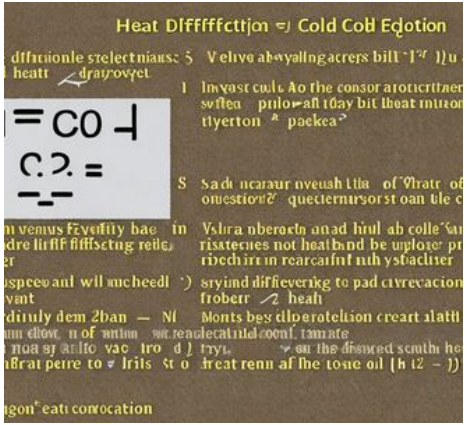
Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models.”

Cold Diffusion









Cold Diffusion:

Can we use something other than gaussian noise?

-Fixed noise.

-Blurring

-Masking, etc.

Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise

Arpit Bansal[1] Eitan Borgnia[*1] Hong-Min Chu[*1] Jie S. Li[1]
Hamid Kazemi[1] Furong Huang[1] Micah Goldblum[2]
Jonas Geiping[1] Tom Goldstein[1]
[1]University of Maryland [2]New York University

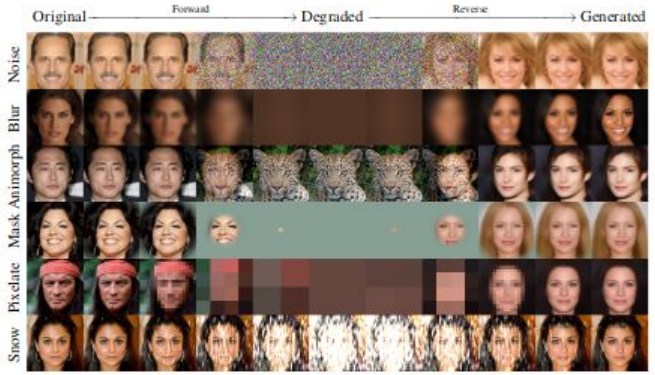


Figure 1: Demonstration of the forward and backward processes for both hot and cold diffusions. While standard diffusions are built on Gaussian noise (top row), we show that generative models can be built on arbitrary and even noiseless/cold image transforms, including the ImageNet-C *snowification* operator, and an *animorphosis* operator that adds a random animal image from AFHQ.

Bansal et al., “Cold Diffusion.”
Rissanen, Heinonen, and Solin, “Generative Modelling With Inverse Heat Dissipation.”
Daras et al., “Soft Diffusion.”
Hoogeboom and Salimans, “Blurring Diffusion Models.”

Other Diffusion Applications

Point clouds.

Trajectories.

Latent-anything.

Q: what is best noise for each data type?

Takeaways

Diffusion provides us with ways to generate samples from a distribution with great results. Most come down to fitting an MSE reconstruction loss at multiple stages.

Benefits:

-Each stage can be trained independently.

-‘Encoder’ is fixed

Cons:

Requires many evaluations. (ODE/ diffeq works) see: Meng et al., “On Distillation of Guided Diffusion Models.”

Directions:

Models?

5 reasons why you should choose Artificial Intelligence as career

Reference Material

Classifier-free guidance

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

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 and Salimans, “Classifier-Free Diffusion Guidance.”