SVGFusion: Generating SVGs with Diffusion Models 🖼️

Hassan Jbara - Uni Leipzig

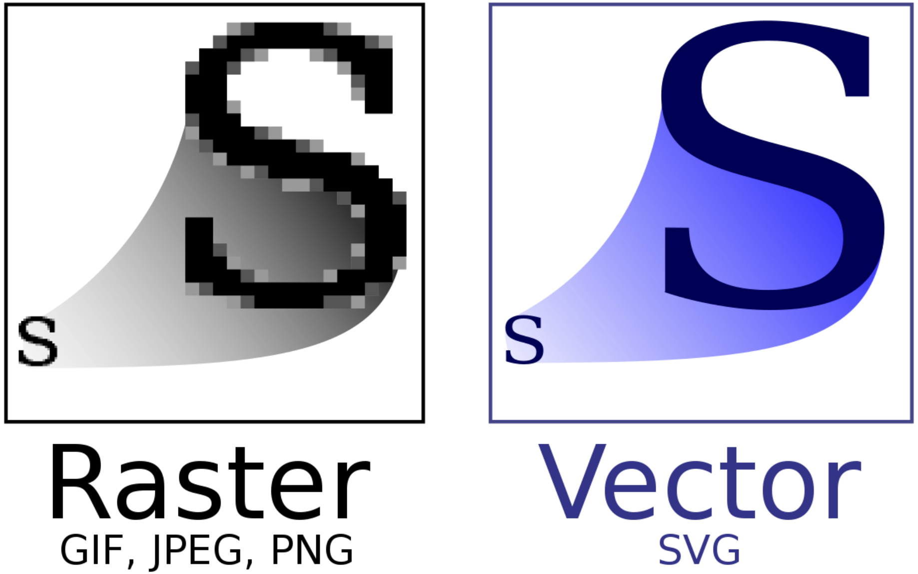
Uses of SVGs

SVGs are an image format used in:

- Websites and Web Development.
- Designs
- Illustrations
- Logos
- Infographics
- ...

Motivation

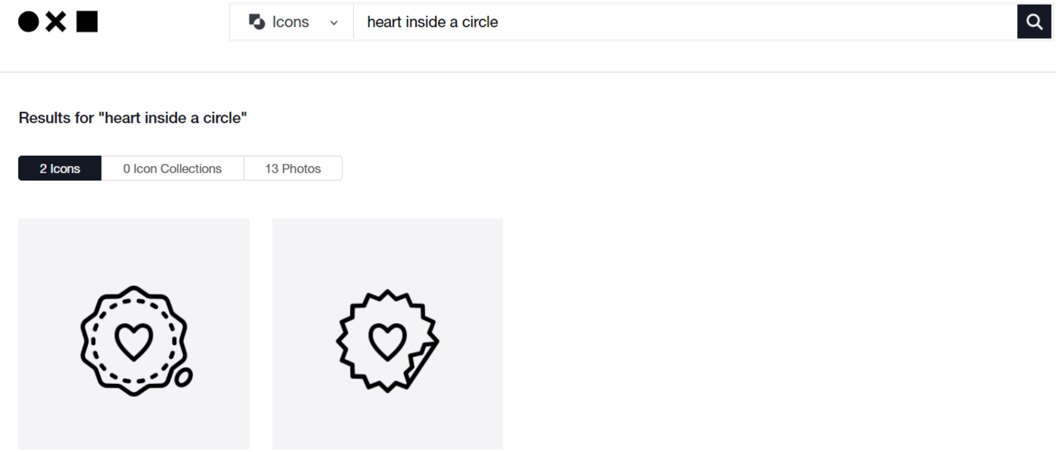
SVGs are infinitely scalable and very useful.



Source: https://en.wikipedia.org/wiki/Vector_graphics

Motivation

The internet doesn't always have what you need.



https://thenounproject.com

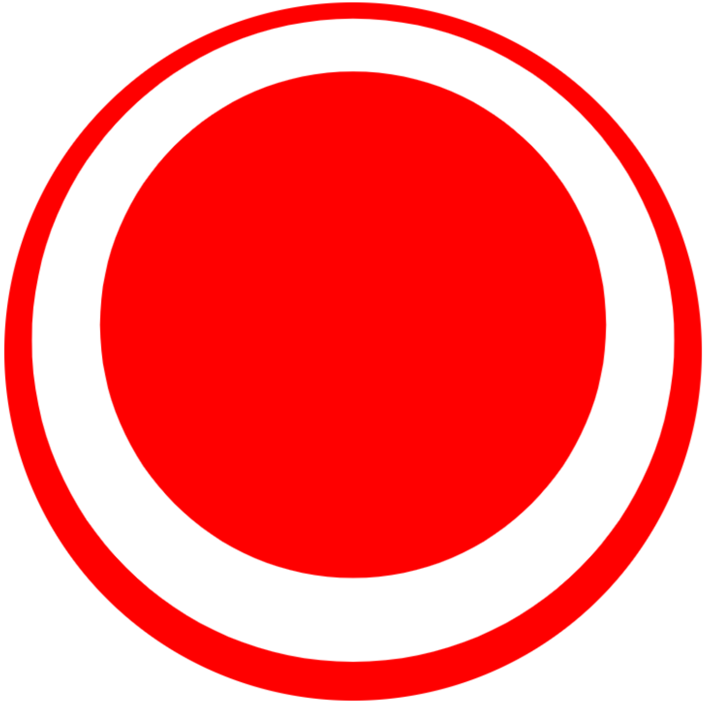
Motivation

Popular image generation models so far only generate raster images.

Generated by DALL-E for “heart inside a circle svg”

Motivation

Large Language Models are also no good.



Generated by GPT-3 for “svg of a heart inside a circle”

Understanding SVGs

SVGs are markup, very similar to HTML.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <path
    fill="none"
    stroke="red"
    d="M 10,30
       A 20,20 0,0,1 50,30
       A 20,20 0,0,1 90,30
       Q 90,60 50,90
       Q 10,60 10,30 z
       " />
</svg>
```

Source: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/d

Understanding SVGs

The SVG code contains:

- Commands for drawing the SVG.
- Coordinates (Arguments) for the commands.
- Other descriptors such aus fill, stroke etc.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <path
    fill="none"
    stroke="red"
    d="M 10,30
       A 20,20 0,0,1 50,30
       A 20,20 0,0,1 90,30
       Q 90,60 50,90
       Q 10,60 10,30 z
       " />
</svg>
```

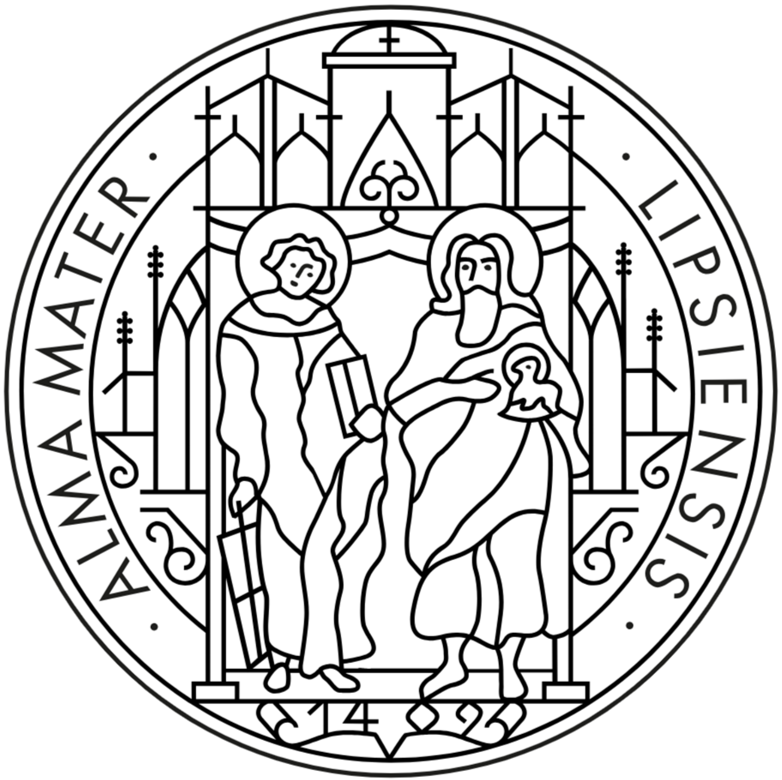
Source: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/d

Understanding SVGs

Most common tag is *<path>*, and with its commands:

- MoveTo: M, m
- LineTo: L, l, H, h, V, v
- Cubic Bézier Curve: C, c, S, s
- Quadratic Bézier Curve: Q, q, T, t
- Elliptical Arc Curve: A, a
- ClosePath: Z, z

You can draw almost anything!



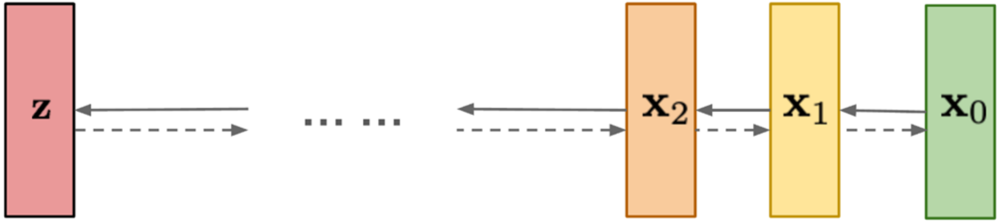
Source: https://de.wikipedia.org/wiki/Universität_Leipzig

Understanding Diffusion Models

Two main parts:

1. Forward Process (Noising).
2. Backward Process (Denoising).

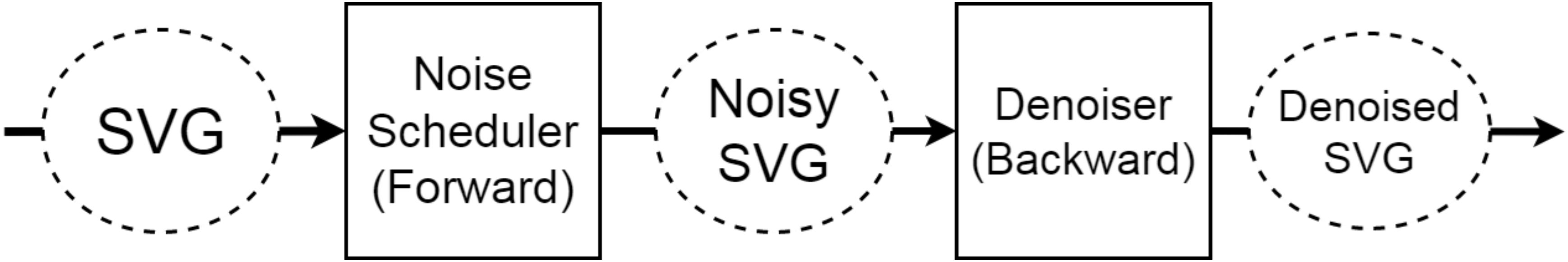
Diffusion models:
Gradually add Gaussian noise and then reverse



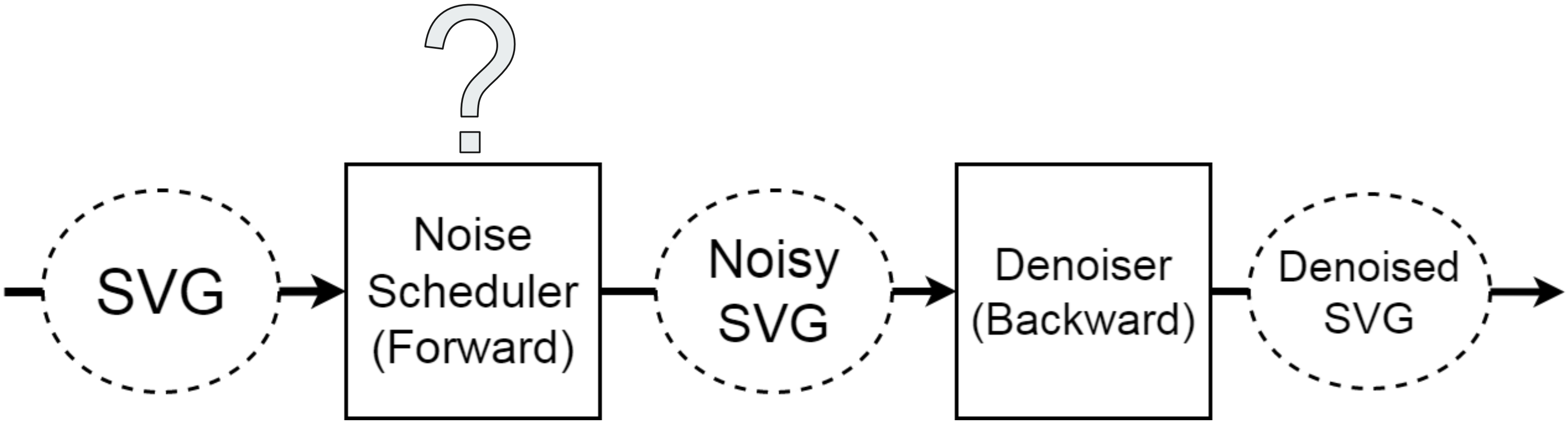
Source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Initial Idea

Our goal is **“Denoised SVG” ≈ “SVG”**



Initial Idea

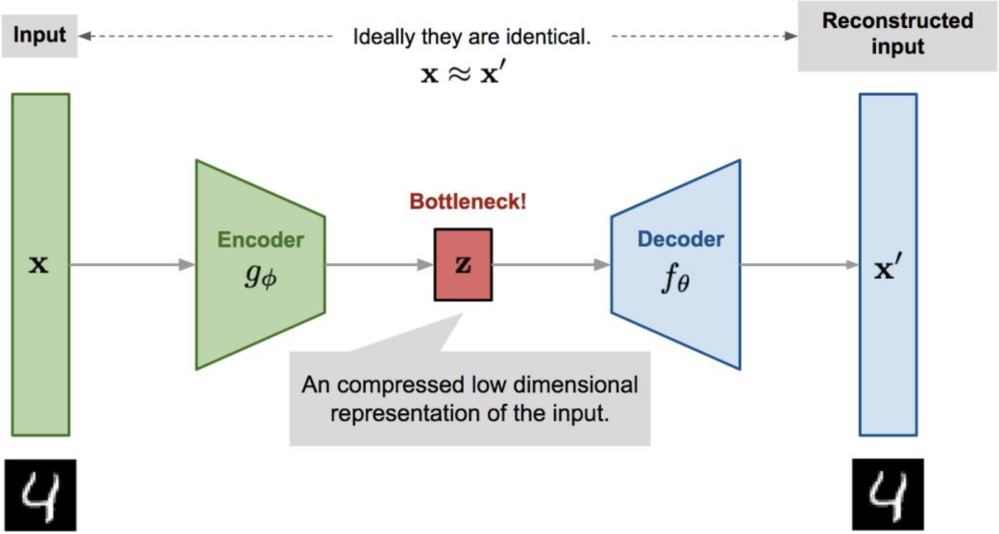


Numerical Representations

Problem: We can't work with SVGs in their original form.

→ Use Autoencoders!

We will run the diffusion in the latent (compressed) space.



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

Related Work

- **DeepSVG (Autoencoder for SVG)**
  - Embeds SVGs into a continuous latent space.
  - A bijection from SVG-space to a 256-dimensional latent space.
- **DiT (Diffusion with Transformers)**
  - Is our denoiser.
  - State-of-the-art and is more flexible than U-Nets.

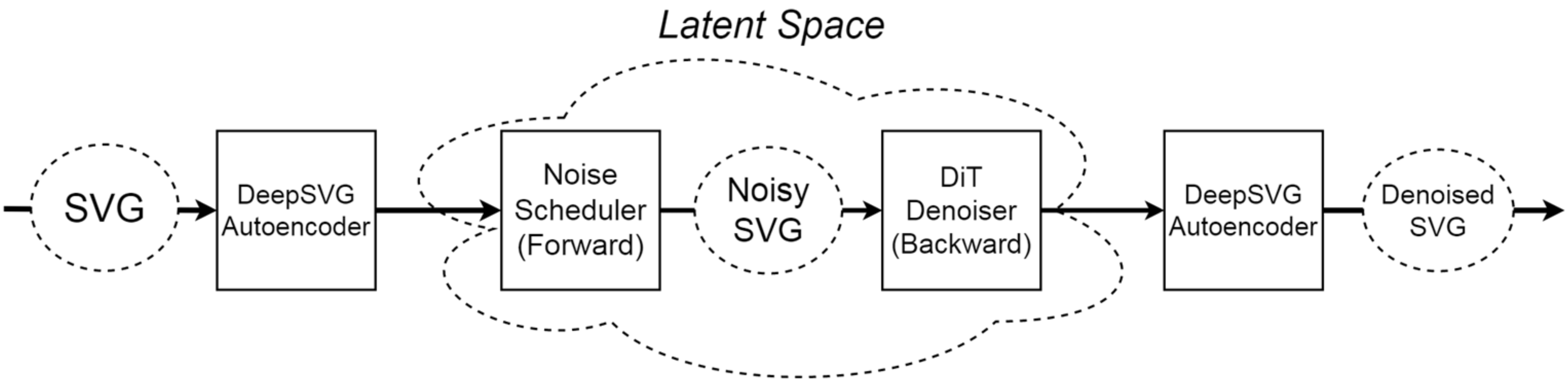
DiT vs U-Nets

U-Nets have been the goto standard for diffusion models, but:

- U-Nets require certain depth in the data.
- U-Nets contain convolution that try to extract semantic meaning from images, which might not be present in SVGs.

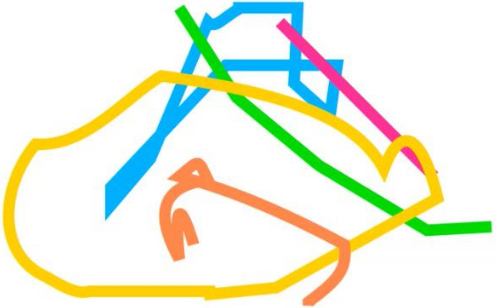
→ **transformers are more flexible and suit our needs better.**

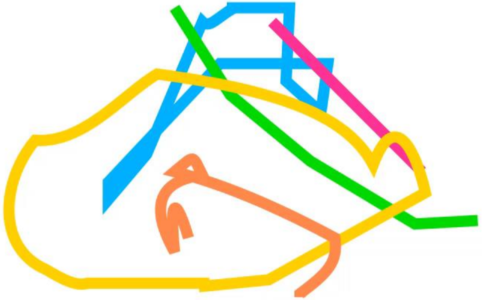
Refined Idea



Results #1

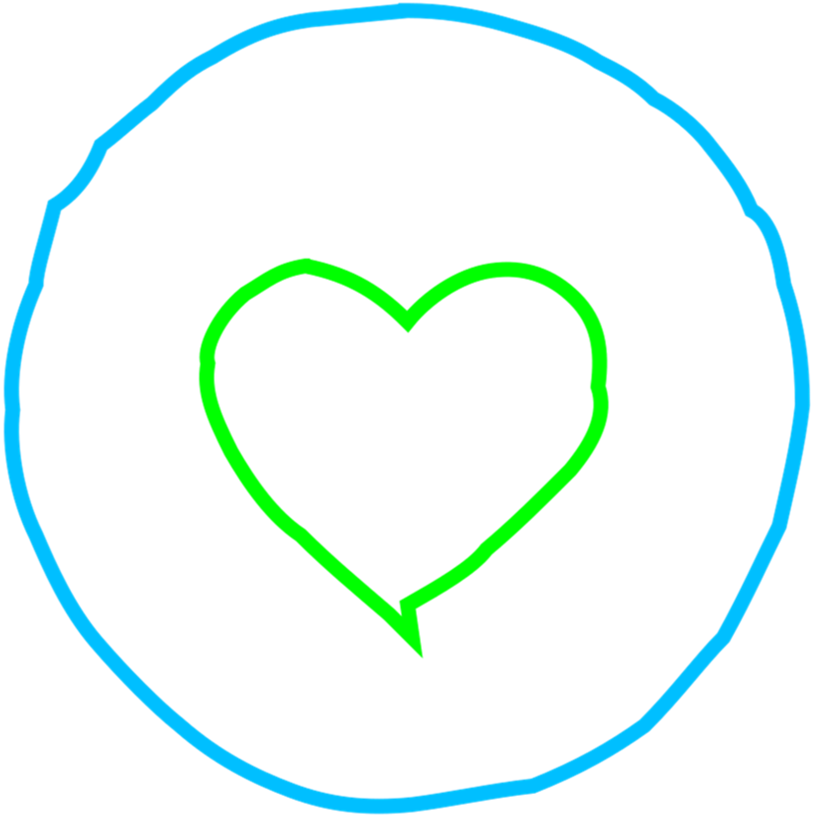
Starting with noise like this





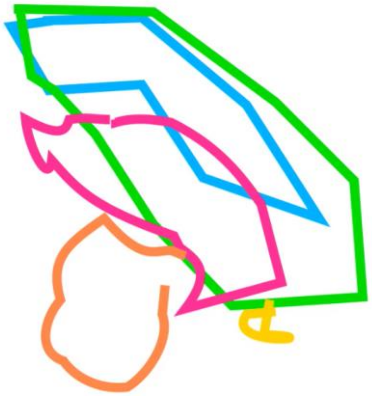
Results #1

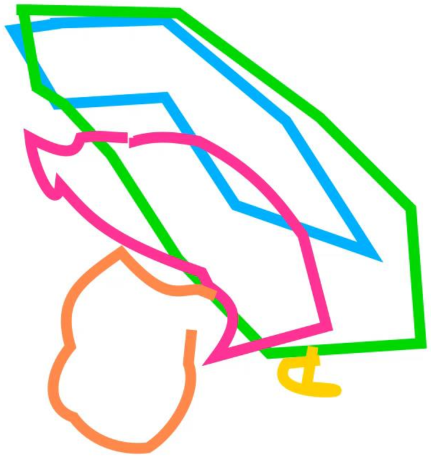
We get this! →



Results #2

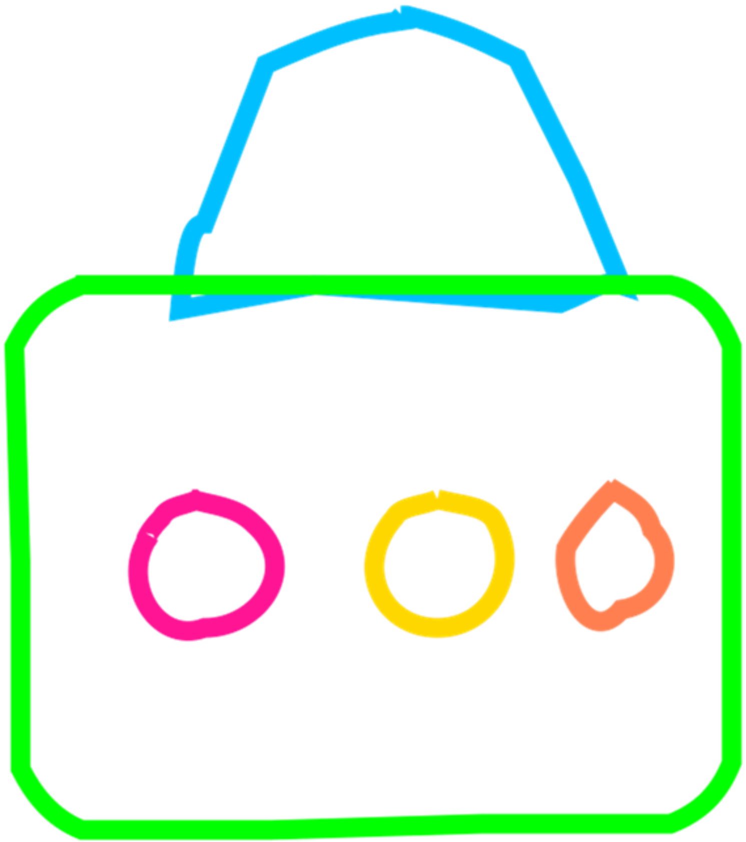
Starting with noise like this





Results #2

We get this! →



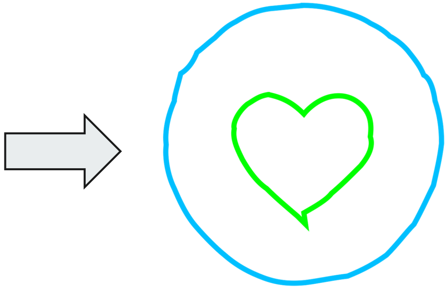
Summary

- **My Work:**
  - Built a system for generating SVGs with diffusion models.
  - Explored the possibilities and limitations of generative AI in regards to SVGs.
- **My Contribution:**
  - Extended the generative diffusion model paradigm to an SVG specific latent space.

Outlook

- **Current Objectives:**
  - Text guidance.
  - Limited user study.
- **Future Objectives (beyond thesis):**
  - User interfaces.
  - Advance methods to convert user expectations into SVGs (RLHF).
  - Generated symbolic images research.

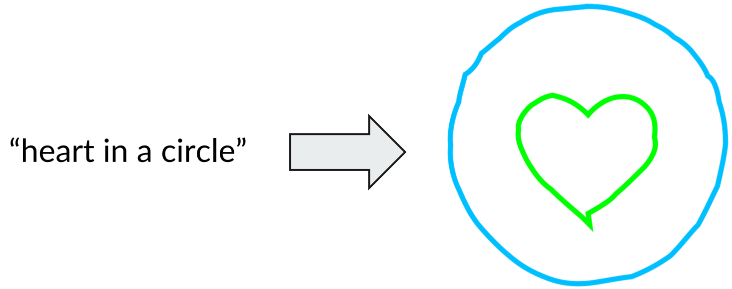
“heart in a circle”



Text guidance example

Outlook - Thanks for Listening!

- **Current Objectives:**
  - Text guidance.
  - Limited user study.
- **Future Objectives (beyond thesis):**
  - User interfaces.
  - Advance methods to convert user expectations into SVGs (RLHF).
  - Generated symbolic images research.



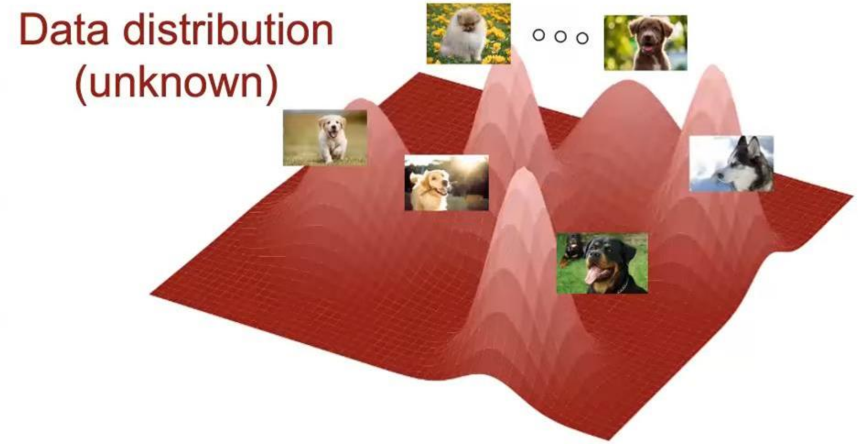
Text guidance example

Extra Slides

Basic Principle of Diffusion Models

given the probability density function f_X(i.e. data distribution) of our data, we can sample new points $\boldsymbol{x} \sim f_X$.

Problem: f_X is unknown and very complicated for high dimensional data.

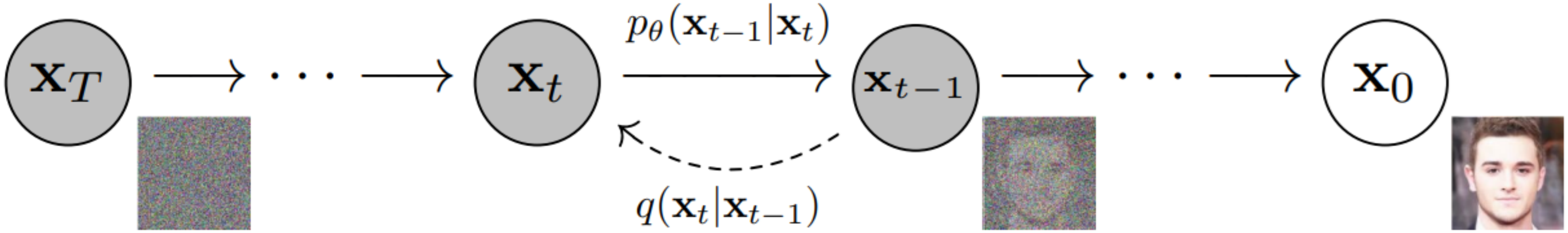


Source: https://www.youtube.com/watch?v=nv-WTeKRLl0

Understanding Diffusion Models - Forward

Given sample x_0 from our distribution $x_0 \sim q$, we produce $x_1, x_2, \ldots, x_T$ noisy samples using a **noise scheduler**:

- **$q(x_t|x_{t-1})$**: probability of x_t given x_{t-1}.

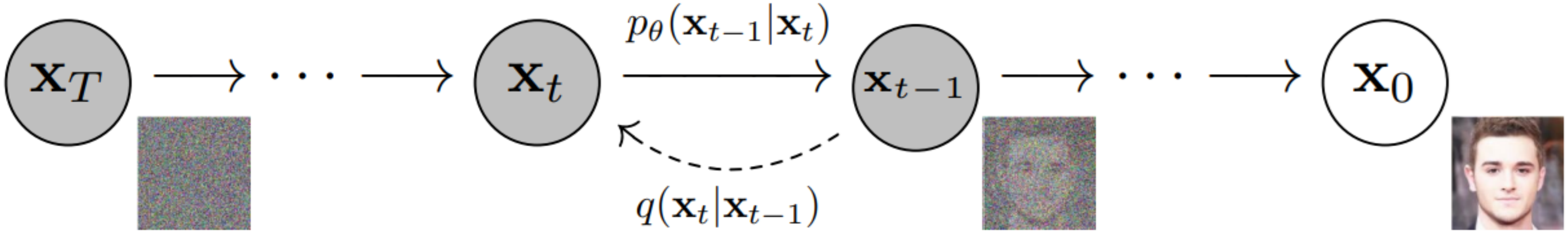


Source: https://arxiv.org/pdf/2006.11239.pdf

Understanding Diffusion Models - Forward

Noise Scheduler: $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$

Where $\beta_t \in (0,1)$; $\beta_1 < \beta_2 < \ldots < \beta_T$, is a variance schedule of our choosing.

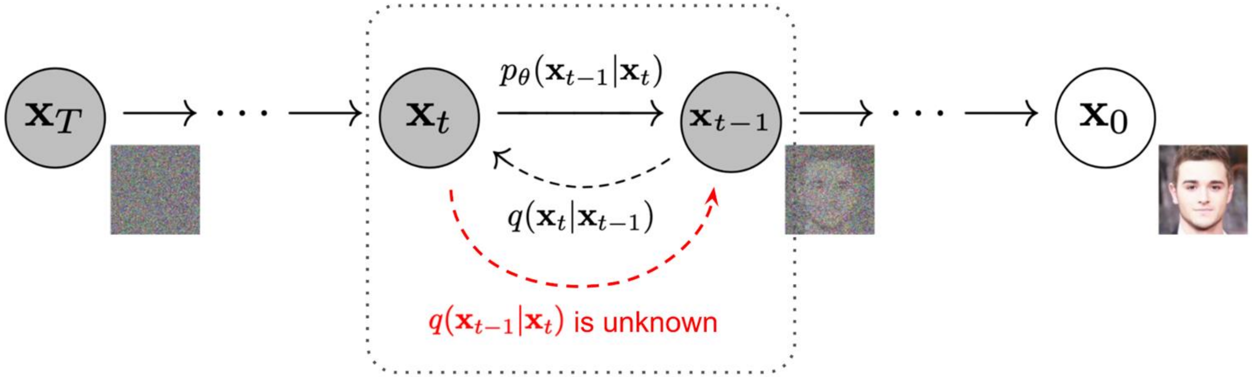


Source: https://arxiv.org/pdf/2006.11239.pdf

Understanding Diffusion Models - Backward

$q(x_{t-1}|x_t)$ is unknown, and **$p_\theta(x_{t-1}|x_t)$** is our estimation (our model) of **$q(x_{t-1}|x_t)$**:

- **$p_\theta(x_{t-1}|x_t)$**: probability of x_{t-1} given x_t, or denoising.



Source: https://arxiv.org/pdf/2006.11239.pdf

Understanding Diffusion Models - Backward

Remember that $\boldsymbol{\beta}$ is a constant of our choosing, which means the only unknown here is $\boldsymbol{\epsilon}$, or the random noise.

Think of it like this: Noisy_image = original_image + noise. If we know the amount of noise added, removing it to get the original image should be easy.

Understanding Diffusion Models

Training to predict the noise with a simple loss function:

$$L_{simple}(\theta) = ||\epsilon_\theta(x_t) - \epsilon_t||_2^2$$

Where:

- $\epsilon_\theta(x_t)$ the model prediction for x at timestep t
- ϵ_t the added noise (ground truth) for timestep t.

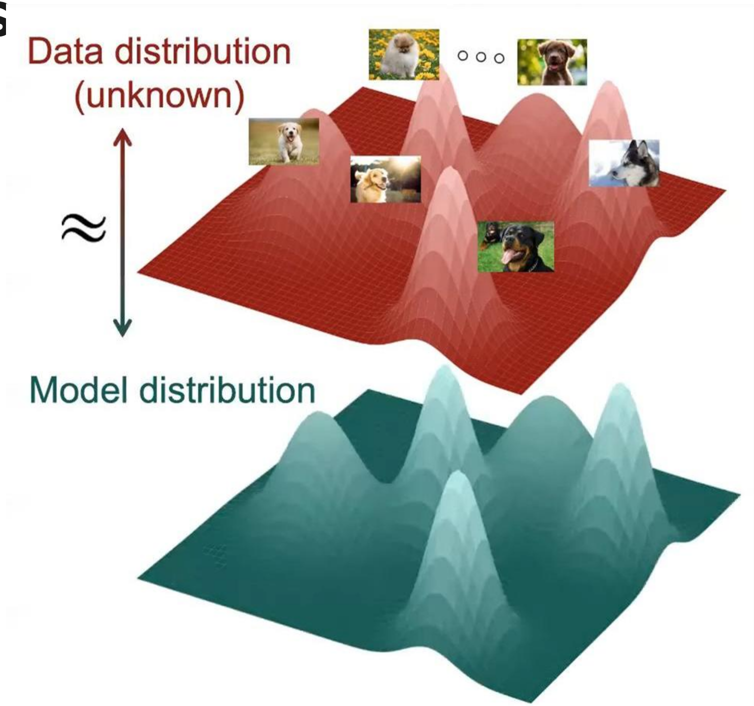
In other word: mean square error of predicted noise against actual noise for each t.

Understanding Diffusion Models

What does this achieve?

→ We are teaching the model to estimate the probability distribution of the data!

(... or technically the gradients of the probability distribution)



Source: https://www.youtube.com/watch?v=nv-WTeKRLl0

Understanding Diffusion Models

Algorithms for training and sampling are therefore pretty simple, although a lot of other variations exist.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
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: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
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** $\mathbf{x}_0$

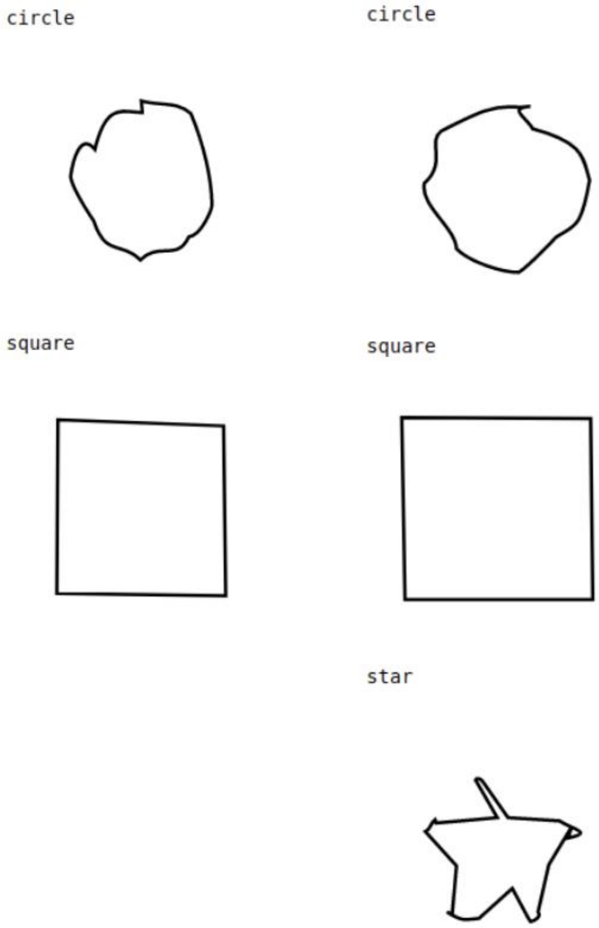
Source: https://arxiv.org/pdf/2006.11239.pdf

Improvement Potential

DeepSVG has a lot of ***shortcomings***:

- Bad at reproducing basic shapes such as circles and squares.
- Not flexible enough for all types of SVGs.

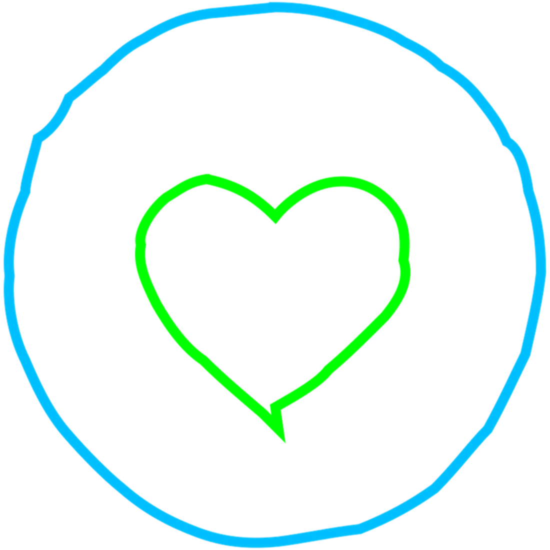
→ it's the main limiting factor for the quality of our model currently, and a better VAE will help a lot.



Source: https://pwichmann.github.io/deep-learning-with-vector-graphics-book/03_related_work/papers/Carlier_et_al_2020.html

Improvement Potential

What DeepSVG originally produces.



What our model learned to produce

Improvement Potential

Two Potential Fixes for DeepSVG:

- A model trained on correcting the inaccuracies that DeepSVG produces (e.g. trained on smoothing squiggly lines).
- Sophisticated algorithms for correcting inaccuracies that DeepSVG produces (i.e. path correction\smoothing algorithms).