Generative Models

Computer Vision - Lecture 14

Further Reading

- Slides from J Johnson
- Slides from R Gao
- Slides from B Wang
- CVPR 2022 Tutorial
- Course from P Holderrieth and E Erives

Basics: Generative Models

Dataset $D = \{x_i | 1 \leq i \leq N\}$

Inputs x_i

~~Outputs y_i~~

Learn a generator that generates samples from the same distribution as the dataset:

Training data: $p_{\text{data}}(\mathrm{x})$

Generated samples: $p_{\text{model}}(\mathrm{x})$

Learn generator such that $p_{\text{model}}(\mathrm{x})$ similar to $p_{\text{data}}(\mathrm{x})$

Discriminative vs. Generative

- Discriminative model: learn $p(y|x)$
- Generative model: learn $p(x)$
- Conditional generative model: learn $p(x|y)$

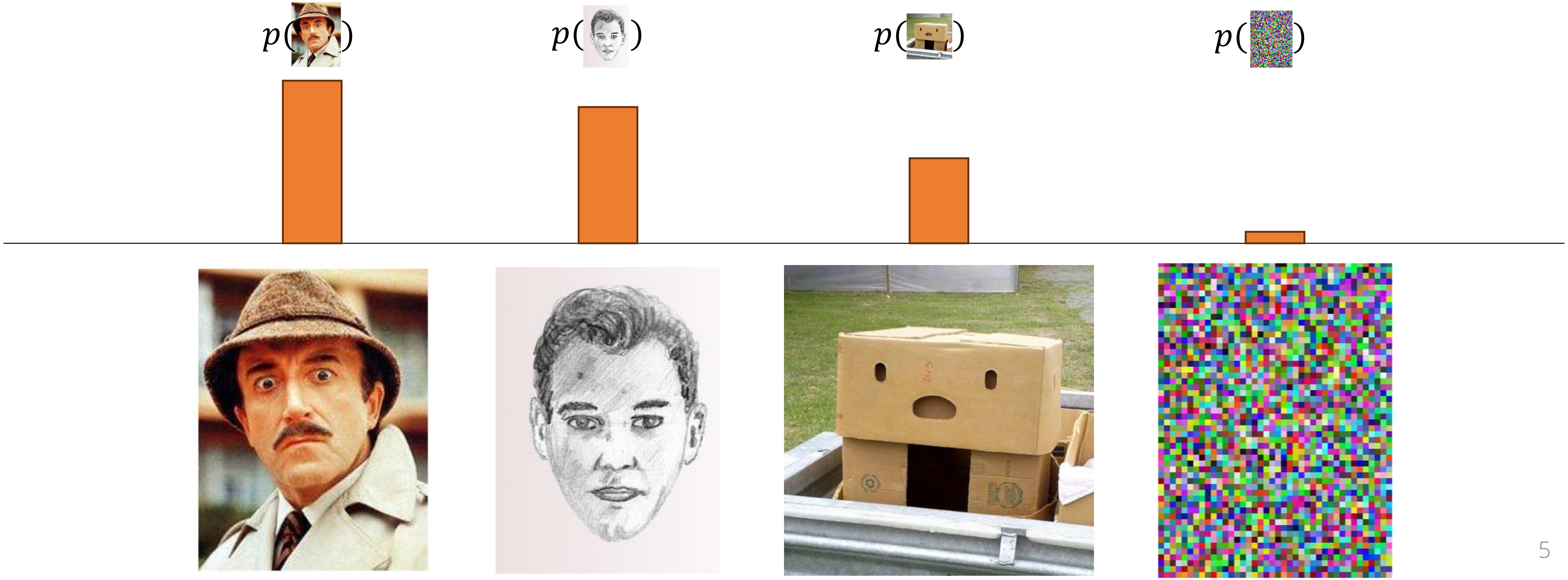
Density function: $p(x) \geq 0, \quad \int_X \, p(x)dx = 1$

Different values of x compete for density.

Generative Models

Learn a probability distribution $p(x)$ over the domain $x \in X$.

“How likely will we find this image in the data?”



Recall: Bayes' Rule

Bayes' Rule lets us build generative models from other components.

Discriminative model

Conditional generative model

$$p(x|y) = \frac{p(y|x)}{p(y)} p(x)$$

Generative model

Prior over labels

Basics: Generative Models

- Explicit: define and solve for the density $p(x)$

- Implicit: sample from $p(x)$ without estimating the density for samples

Generative Models

Explicit Models
can compute $p(x)$

- Tractable Density
  - Autoregressive models (MADE, NADE, PixelRNN, etc.)
- Approximate Density
  - Variational Autoencoders
  - Markov Chain

Implicit Models
can only sample from $p(x)$

- Direct
  - GANs
  - Diffusion Models
- Markov Chain
  - GSN

Updated from Ian Goodfellow, Tutorial on GANs 2017

Autoregressive Models

- Explicit model: fully visible belief network
- Chain rule decomposes the likelihood of an image into distributions of pixel intensities

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

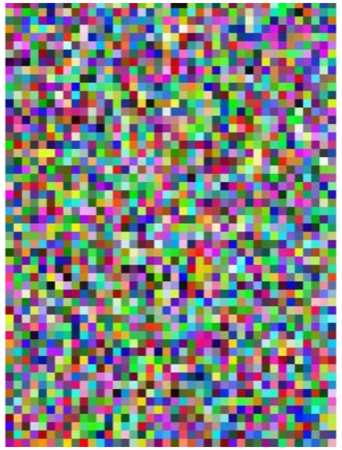
- Train by maximizing likelihood of training data.
- Main generative model in NLP.

Updated from Ian Goodfellow, Tutorial on GANs 2017

Goal: Learning a distribution

- Potentially very complex and high dimensional!
- What is the probability that $x \in \mathbb{R}^{64\times64\times3}$ is an image of a face?

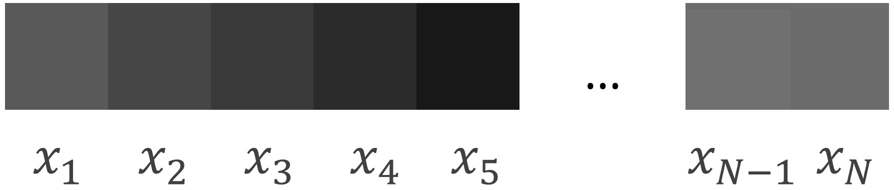


Autoregressive Distribution Estimation

- rearrange an image into a sequence
- now, task can be seen a sequence prediction problem
- What is the colour distribution of the next pixel?

Autoregressive Distribution Estimation



x

$$p(x) = p(x_1)\, p(x_2|x_1) \cdot \ldots \cdot p(x_N|x_1, \ldots, x_{N-1})$$

$$p(x) = \prod_{i=1}^{N} p(x_i|x_1, \ldots, x_{i-1})$$

Connectionist learning of belief networks
Neal RM,Artificial intelligence 1992

The neural autoregressive distribution estimator
Larochelle H, Murray I, AISTATS 2011

Pixel Recurrent Neural Networks
Van Oord A, Kalchbrenner N, Kavukcuoglu K, ICLR 2016

Autoregressive Distribution Estimations

- In general: $p(x_i|x_1, \dots, x_{i-1})$ might still be very complicated.

- But images are easy:

- we store them 8 bit per channel (RGB)

- 256 element softmax per channel and pixel

- models the exact distribution

Sampling

$p(x_1)$

$p(x_2|x_1)$

$p(x_3|x_1, x_2)$

$\vdots$

$p(x_N|x_1, \dots, x_{N-1})$

Inference

$$p(x) = p(x_1)\, p(x_2|x_1)\, p(x_3|x_1,x_2) \cdot \ldots \cdot p(x_N|x_1,\ldots,x_{N-1})$$

0.12 0.94 0.86 0.99

$p(x_1)$ 0.12

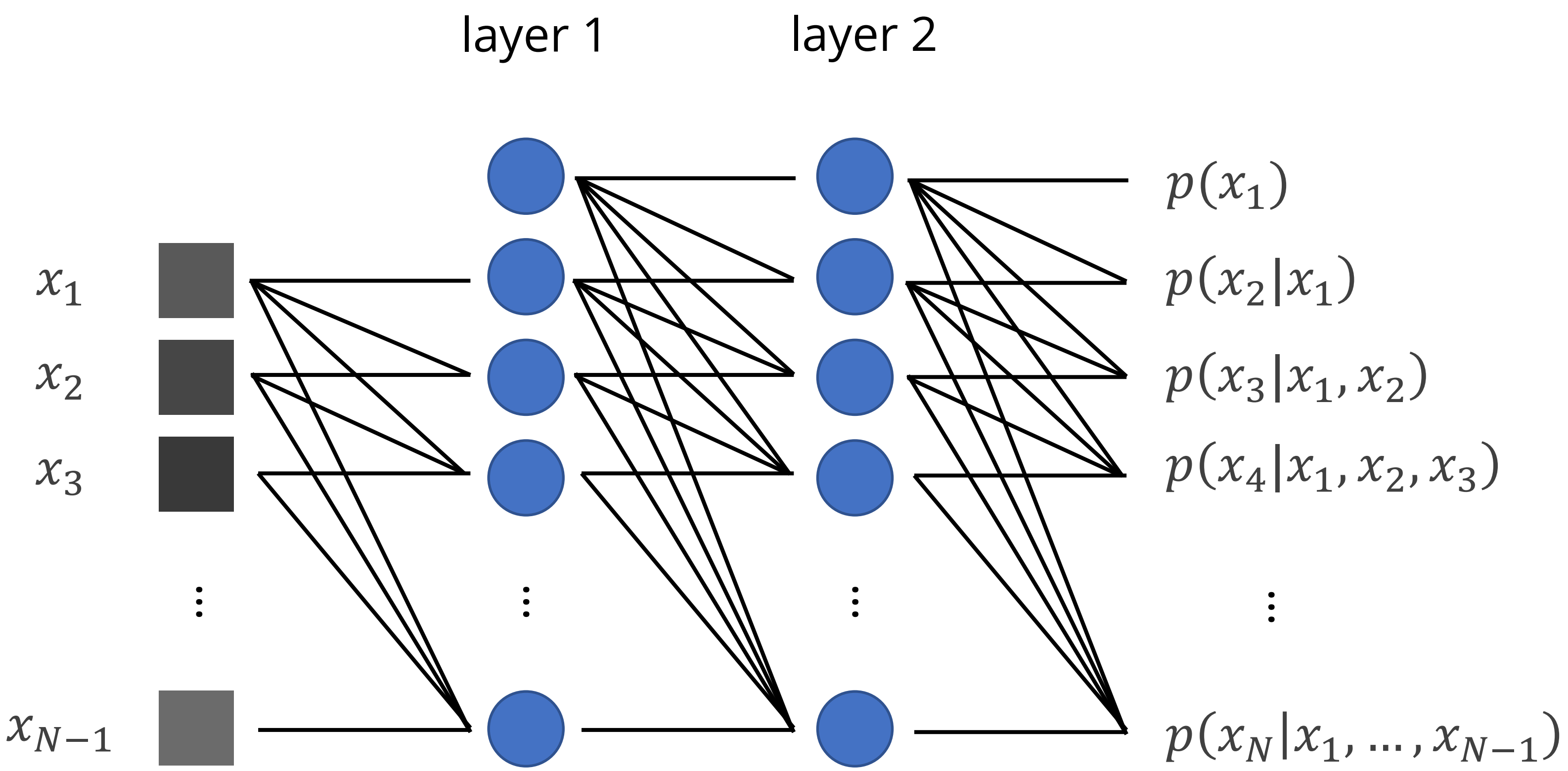
$p(x_2|x_1)$ 0.94

$p(x_3|x_1,x_2)$ 0.86

$\vdots$

$p(x_N|x_1,\ldots,x_{N-1})$ 0.99

Learning - NN



The neural autoregressive distribution estimator
Larochelle H, Murray I
AISTATS 2011

Learning - CNN

- Idea: mask the weights of the convolutions

1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Mask A

for the first layer

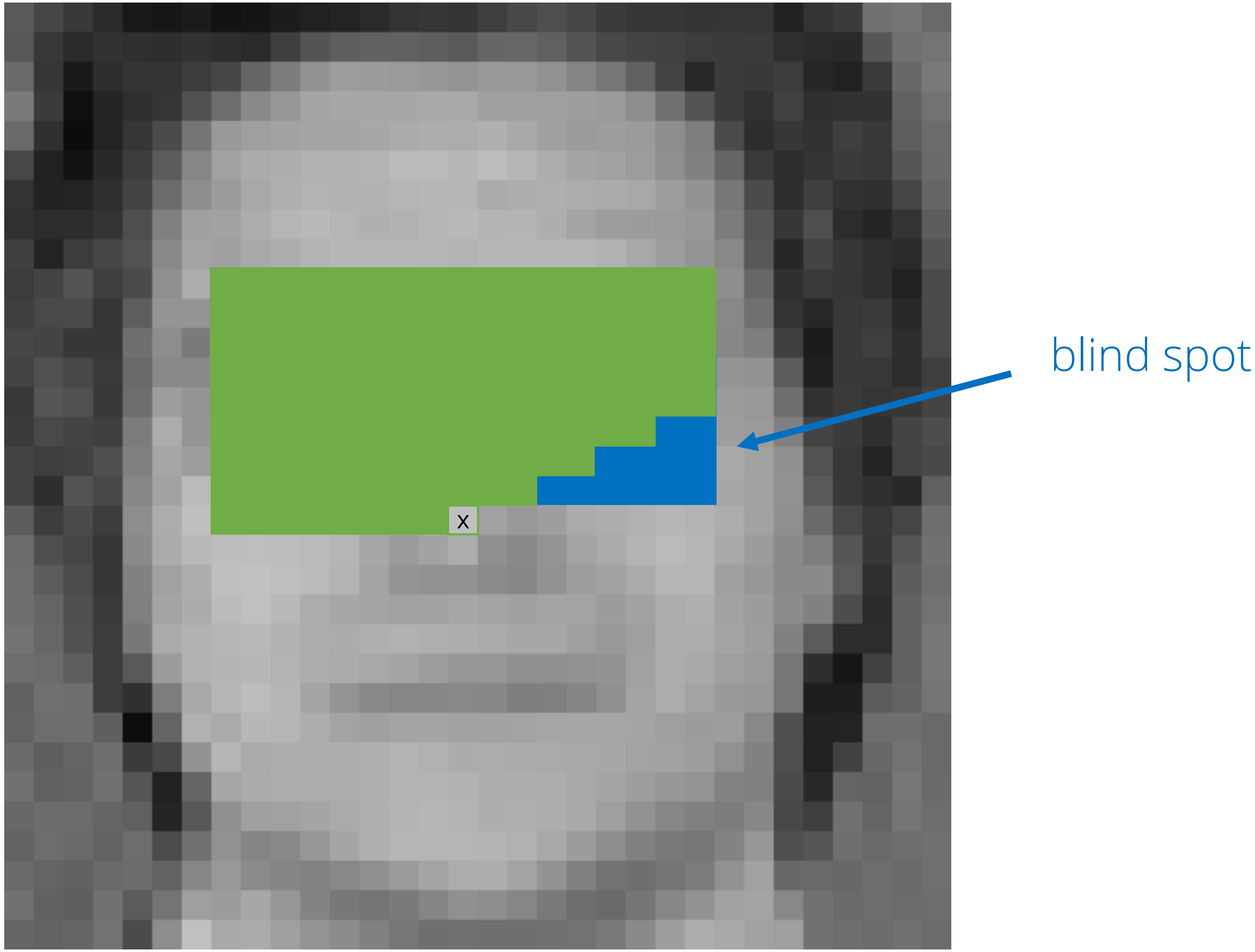
1	1	1	1	1
1	1	1	1	1
1	1	1	0	0
0	0	0	0	0
0	0	0	0	0

Mask B

for all other layers

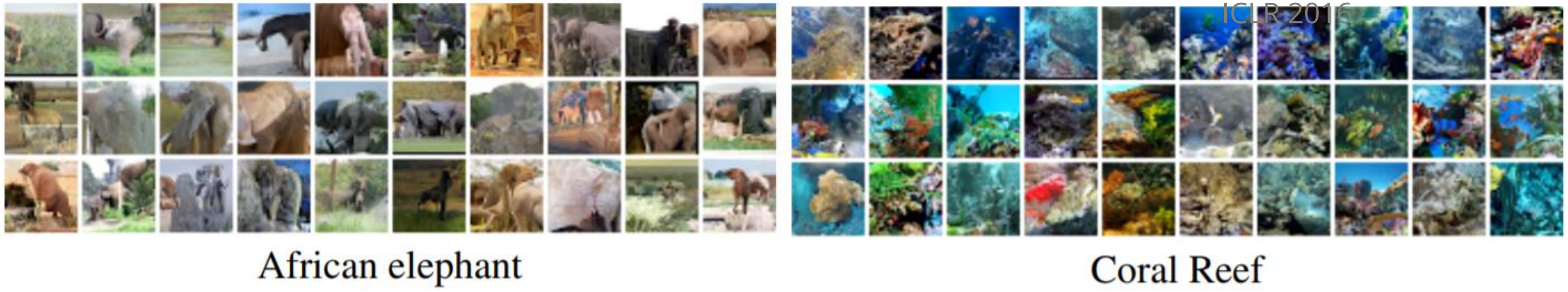
Pixel Recurrent Neural Networks
Van Oord A, Kalchbrenner N, Kavukcuoglu K
ICLR 2016

Learning – CNN Receptive Field



Results

Pixel Recurrent Neural Networks
Van Oord A, Kalchbrenner N, Kavukcuoglu K
ICLR 2016



Conditional Image Generation with PixelCNN Decoders
Van den Oord A, Kalchbrenner N, Espeholt L, Vinyals O, Graves A
NeurIPS 2016

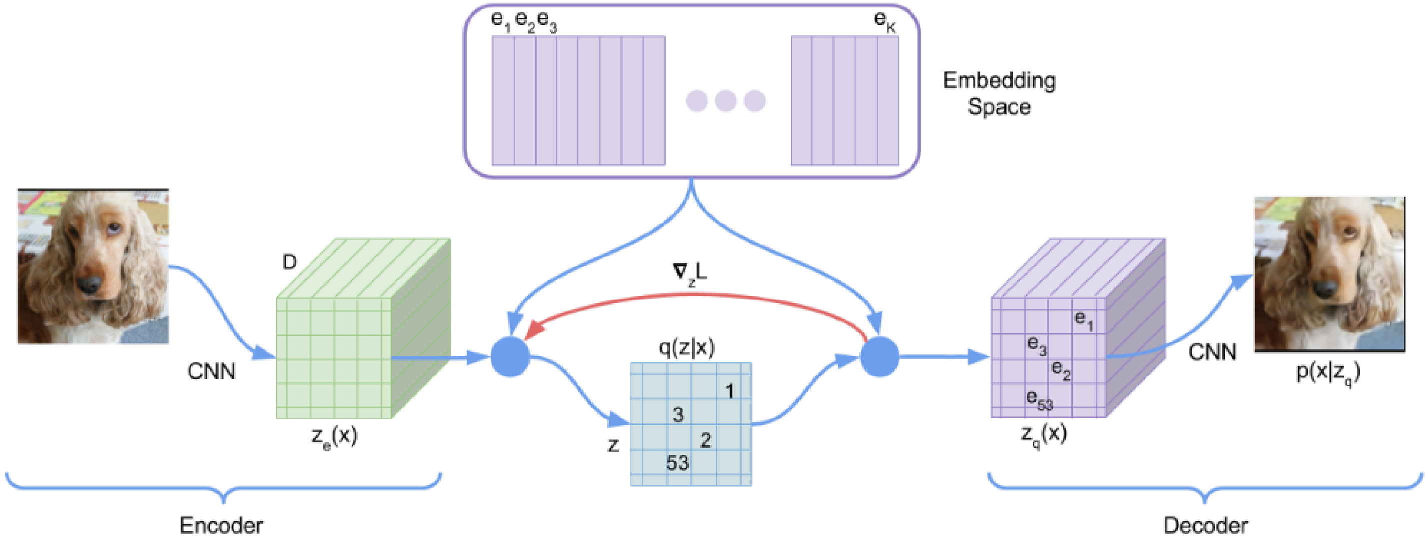
Dall-E

- Autoregressive model made fast by first learning a compressed discrete image representation
- Generation in "token-space"
- Conditioned on text prompts
- Large-scale training 400M image-text pairs

Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. "Zero-shot text-to-image generation." In International Conference on Machine Learning, pp. 8821-8831. PMLR, 2021.

VQ-VAE

- Down-sample and compress the image into a lower-dimensional, discrete representation.
- Bottleneck: replace activations with closest vector from a learned codebook.

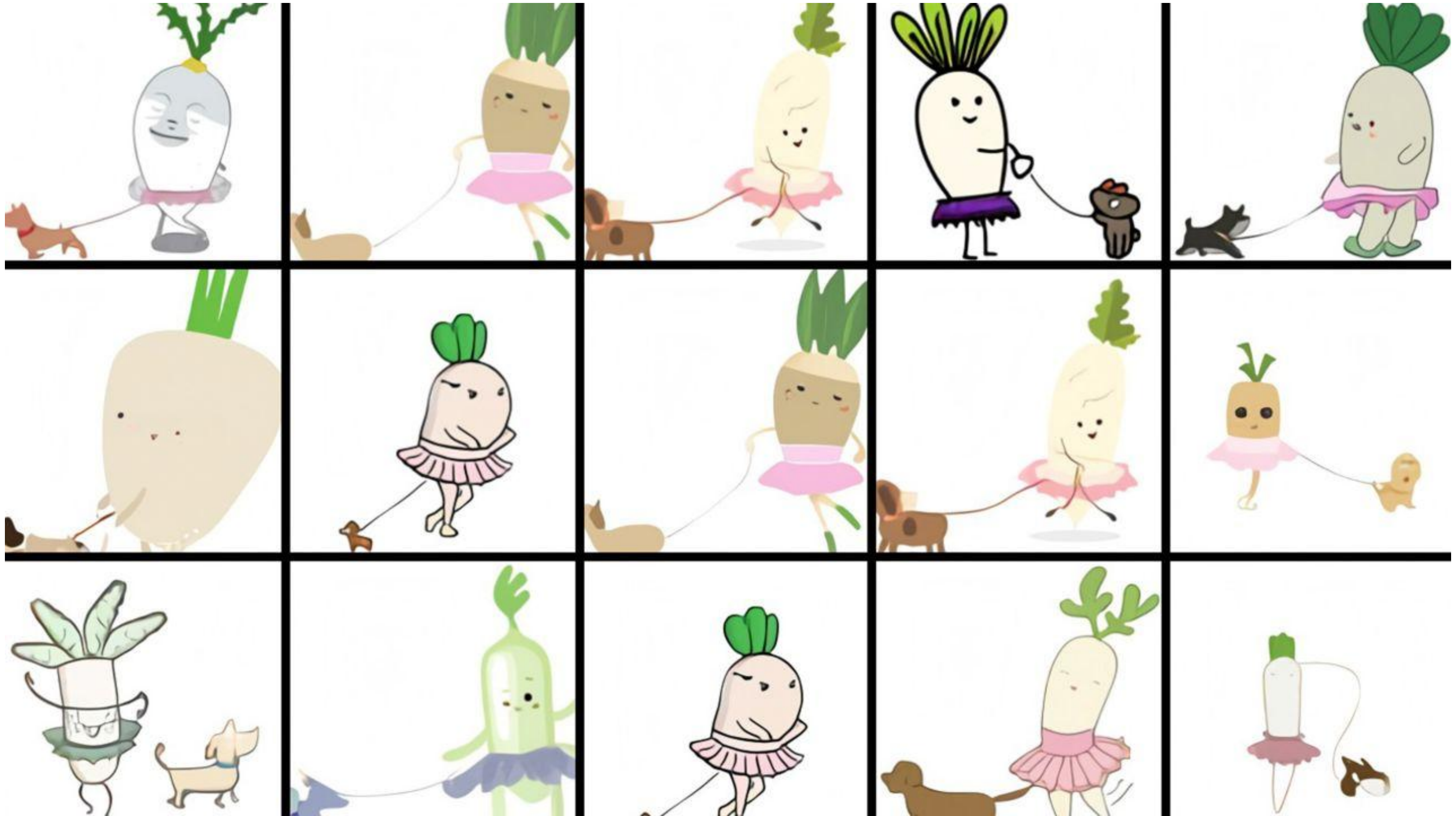


Neural Discrete Representation Learning
Aaron van den Oord, Oriol Vinyals, Koray Kavukcuoglu, 2017

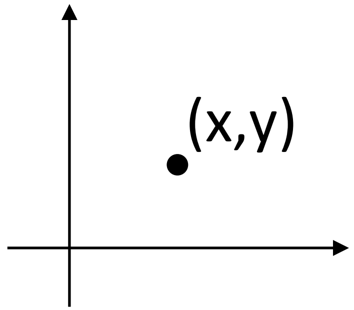
Dall-E

- One of the first examples of very good generalisation:

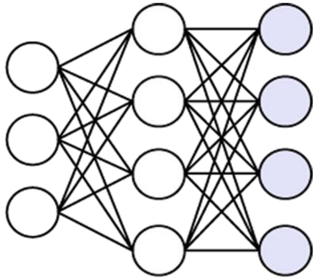
“*an illustration of a baby daikon radish in a tutu walking a dog*”



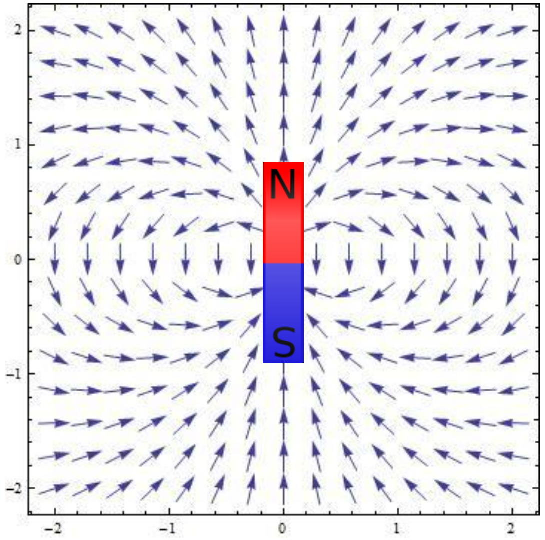
Recall: Neural (Flow) Fields



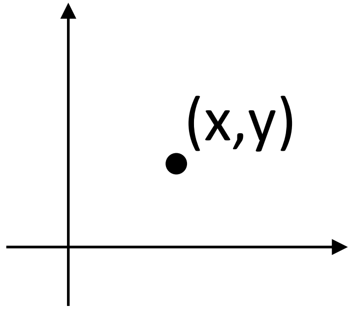
$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$



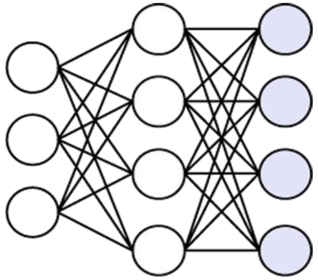
Neural Network (Φ)



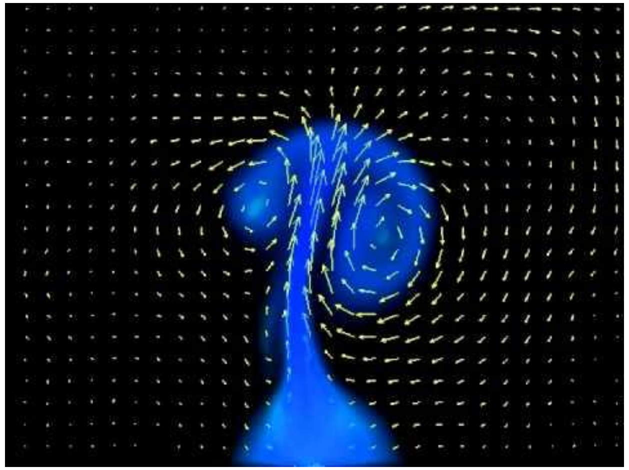
Magnetic Field



$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$



Neural Network (Φ)



Eulerian Flow Field

[Koldora CC]

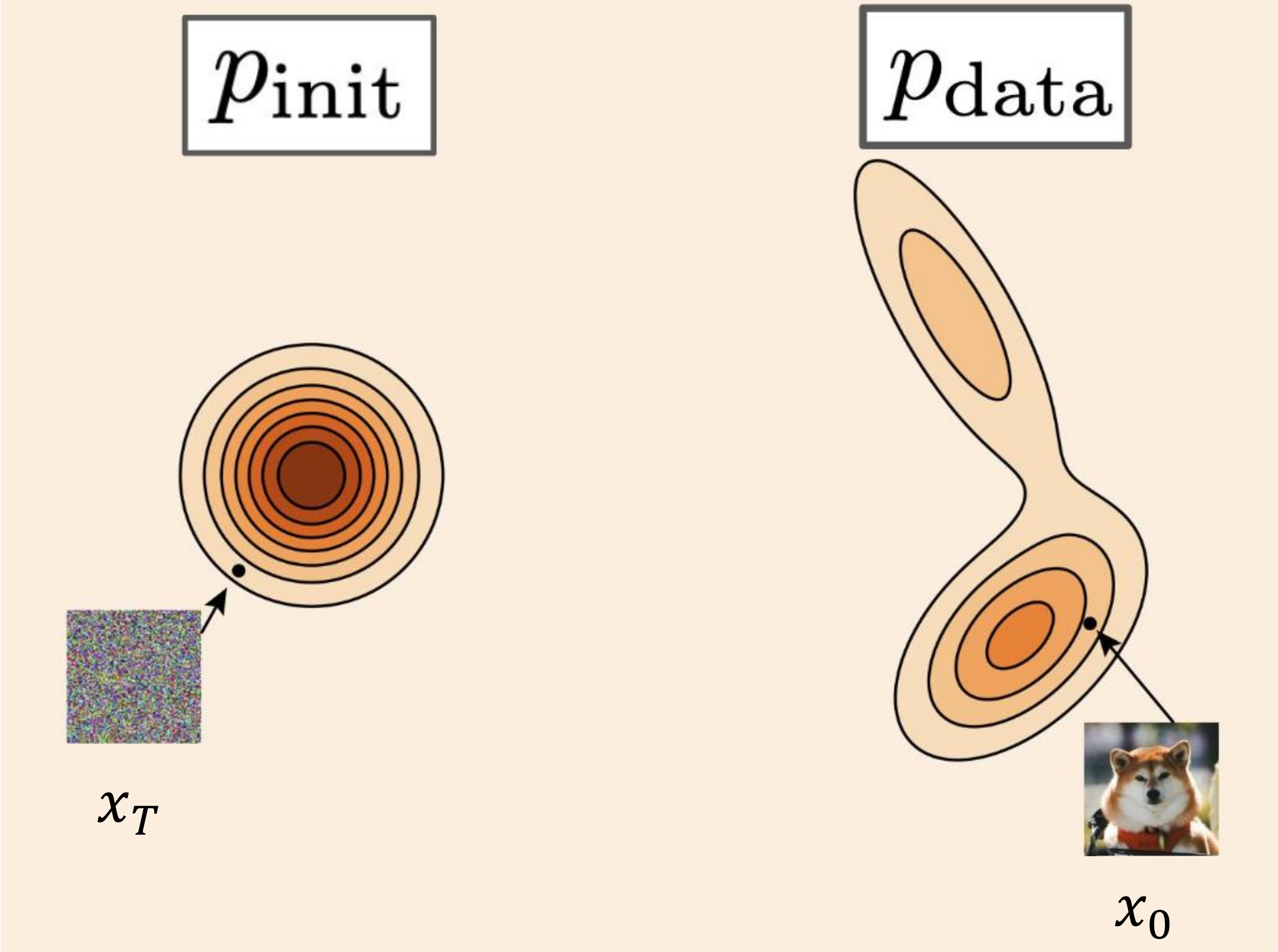
[Slide: Srinath Sridhar, Towaki Takikawa at CVPR ‘22 Tutorial on Neural Fields in Computer Vision]

Converting Samples

- Idea: convert samples from a simple distribution into samples from the data distribution

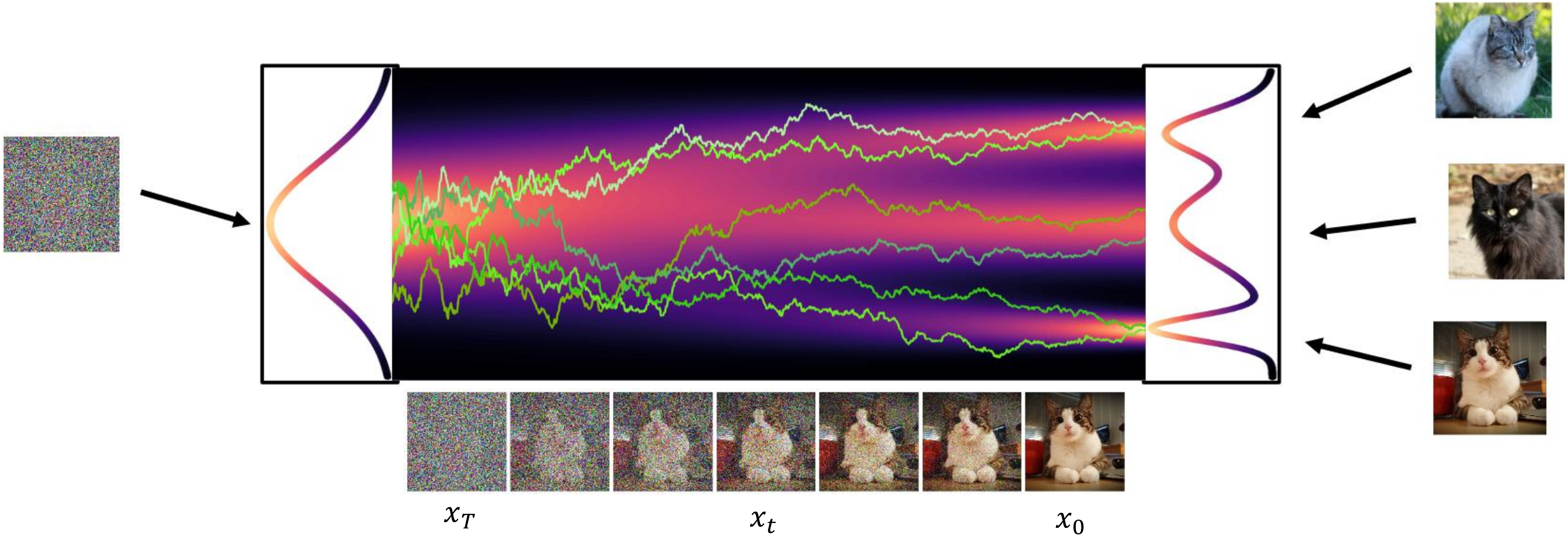
- Learn a neural field to represent the flow from x_T to x_0

- $x_{t-1} = f(x_t) + x_t$



[Figure: Yaron Lipman]

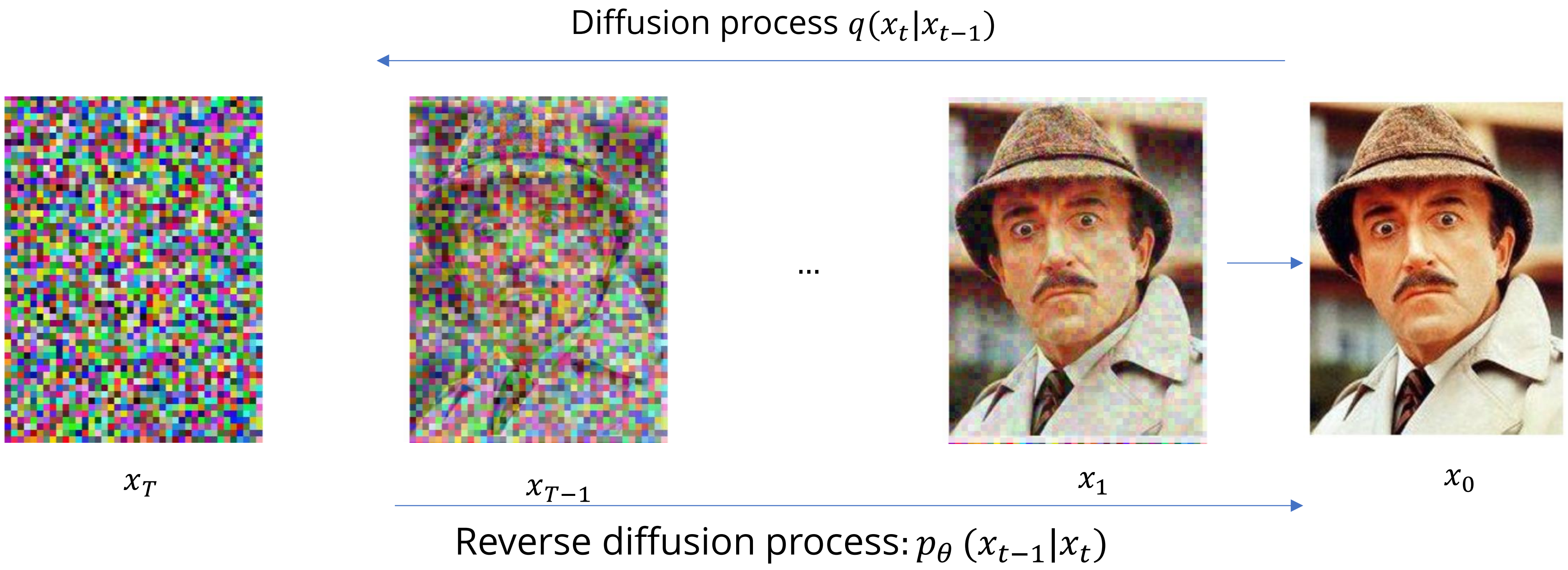
Flow between distributions



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

Diffusion Models

- Generate an image in small steps from (Gaussian) noise ϵ
- Instead of directly learning a model for $p_\theta(x|\epsilon)$, learn small steps along a Markov chain



Noise Schedule

- Learn a model to generate $p_\theta(x_{t-1}|x_t)$
- Construct the diffusion process:

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$$

- β_t is a variance schedule – often fixed

- There is a closed-form solution to sample x_t from x_0

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

Denoising Diffusion Probabilistic Models
Jonathan Ho, Ajay Jain, Pieter Abbeel, 2020

Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." NeurIPS'21.

Noise Schedule

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$$

Random noise image $\epsilon_t \sim \mathcal{N}(0, I)$

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

$\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \sum_{i=1}^{t} \alpha_i$

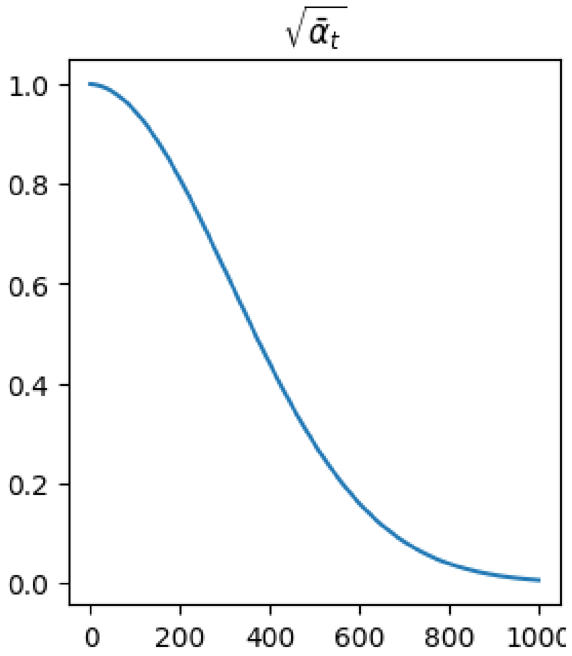
Applying iteratively yields:

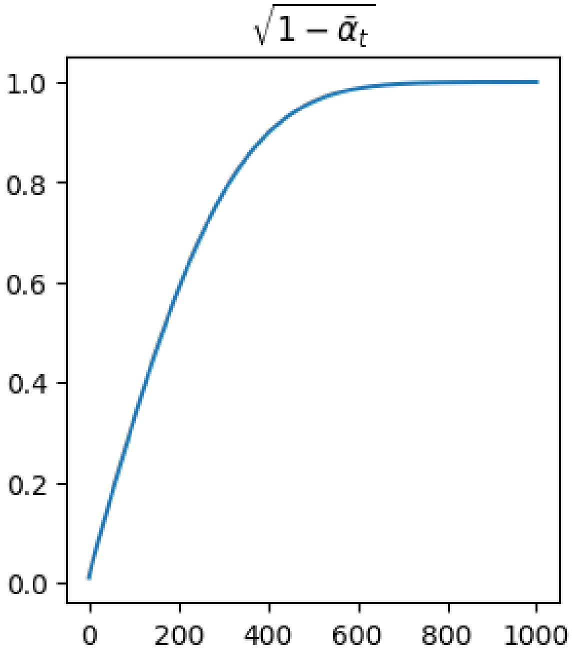
$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

Noise Schedule

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

Not exactly linear interpolation.





Noise Schedule

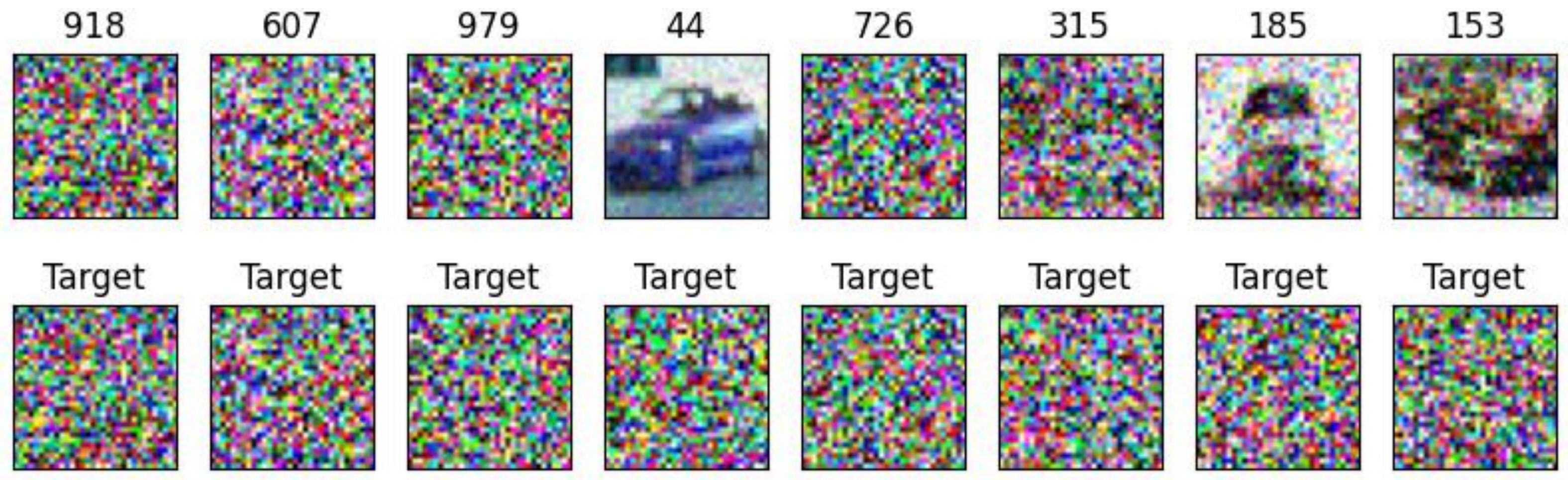
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

It *looks* like steps > 500 are pure noise, but they are not.

These are very important for the model to learn.

Training a diffusion model

- Generate training examples (x_t, ϵ_t)



- Loss $|f(x_t, t) - \epsilon_t|_2^2$ (simple L2 loss)

Sampling from a diffusion model

Sample a noise image: $x_T \sim \mathcal{N}(0, I)$

for $t = T, \dots, 1$:

$$z \sim \mathcal{N}(0, I) \textbf{ if } z > 1 \textbf{ else } 0$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} f(x_t, t)\right) + \sqrt{\beta_t}\, z$$

Evaluate diffusion model T times to generate one sample.

Denoising Diffusion Probabilistic Models
Jonathan Ho, Ajay Jain, Pieter Abbeel, 2020

Predicting images instead of noise

- An equivalent model can be trained by learning to predict x_0 instead: $|f(x_t, t) - x_0|_2^2$

- We can use $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ to convert between sampling steps and noise.

- In practice predicting noise works often a bit better: more diversity during training.
- Predict x_0 model has the same target for every timestep: easier overfitting/memorisation.

Large Scale Diffusion Models

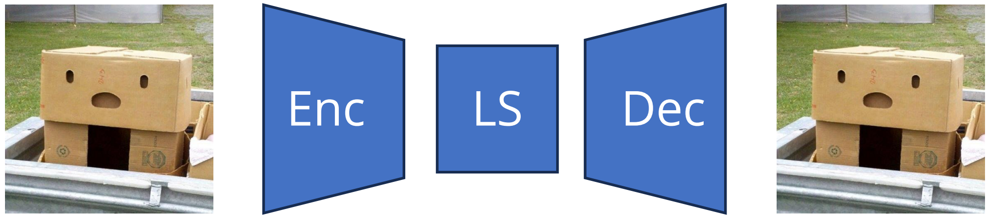
- Conditional diffusion: trained conditional on text input.

- Large scale training: >1B images (&text)

- Several details:
  - Often more stable to predict noise instead of the clean sample. One can always compute one from the other.
  - Noise schedule is important.
  - Latent diffusion.

Latent Diffusion

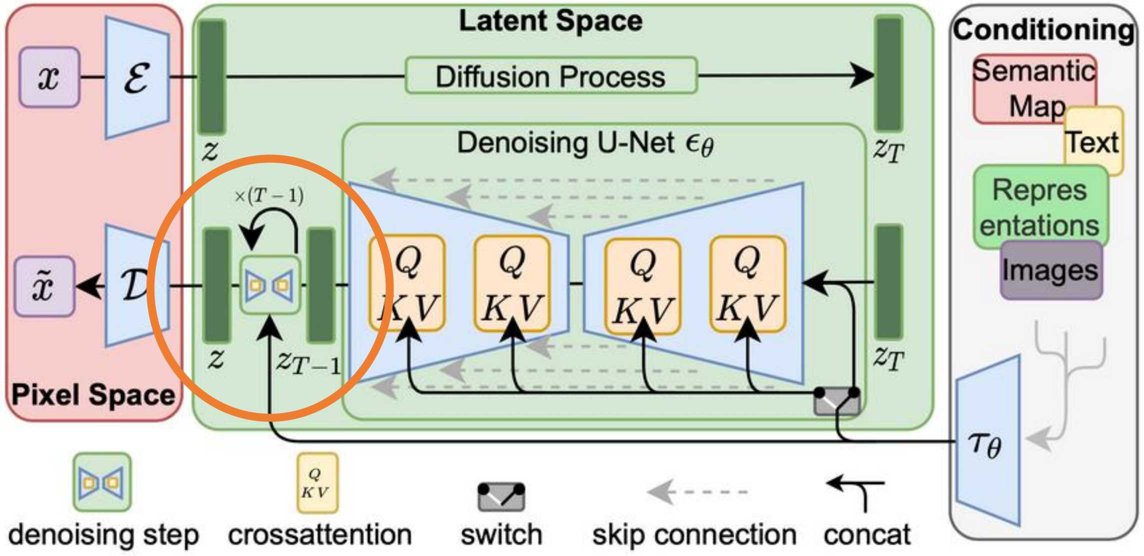
- Diffusion needs many evaluations of the noise estimator.
- We can make it cheaper but first compressing the image into a latent representation.
- Train an encoder decoder architecture: VQ-VAE.
- E.g.: D=4, H’ = H/4, W’ = W/4



3 x H x W | latent space D x H’ x W’ | 3 x H x W

Stable Diffusion 1

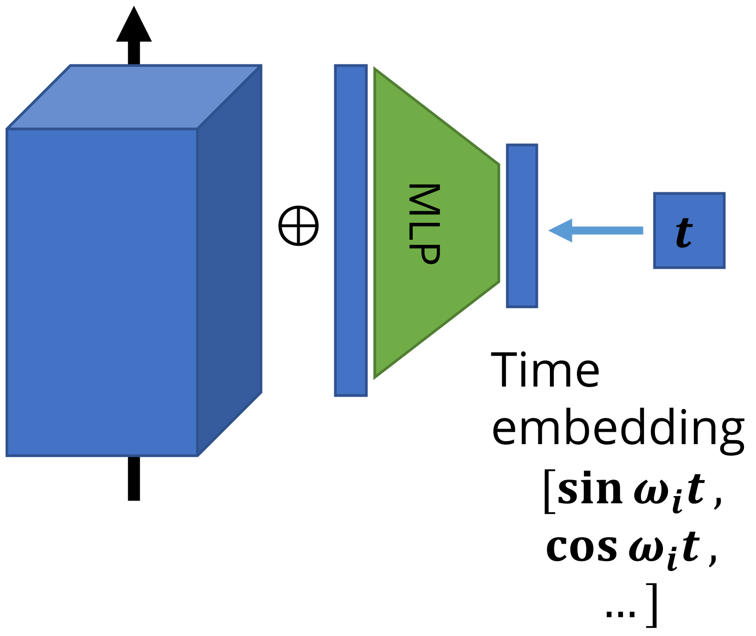
- Conditional diffusion model.
- Latent diffusion.
- U-Net architecture + cross attention layers to text & timestep



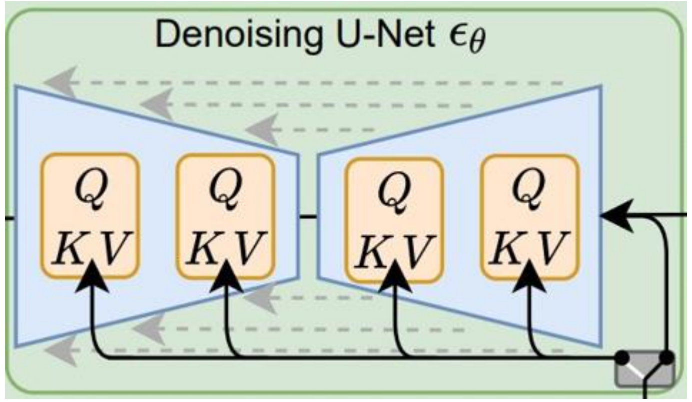
Note: Add Time Dependency

- The score function is *timestep-dependent*.
  - $f(x, t)$

- Add time dependency
  - Assume time dependency is spatially homogeneous.
  - Add one scalar value per channel $f(t)$
  - Parametrize $f(t)$ by MLP / linear of Fourier basis.



Unet in Stable Diffusion 1



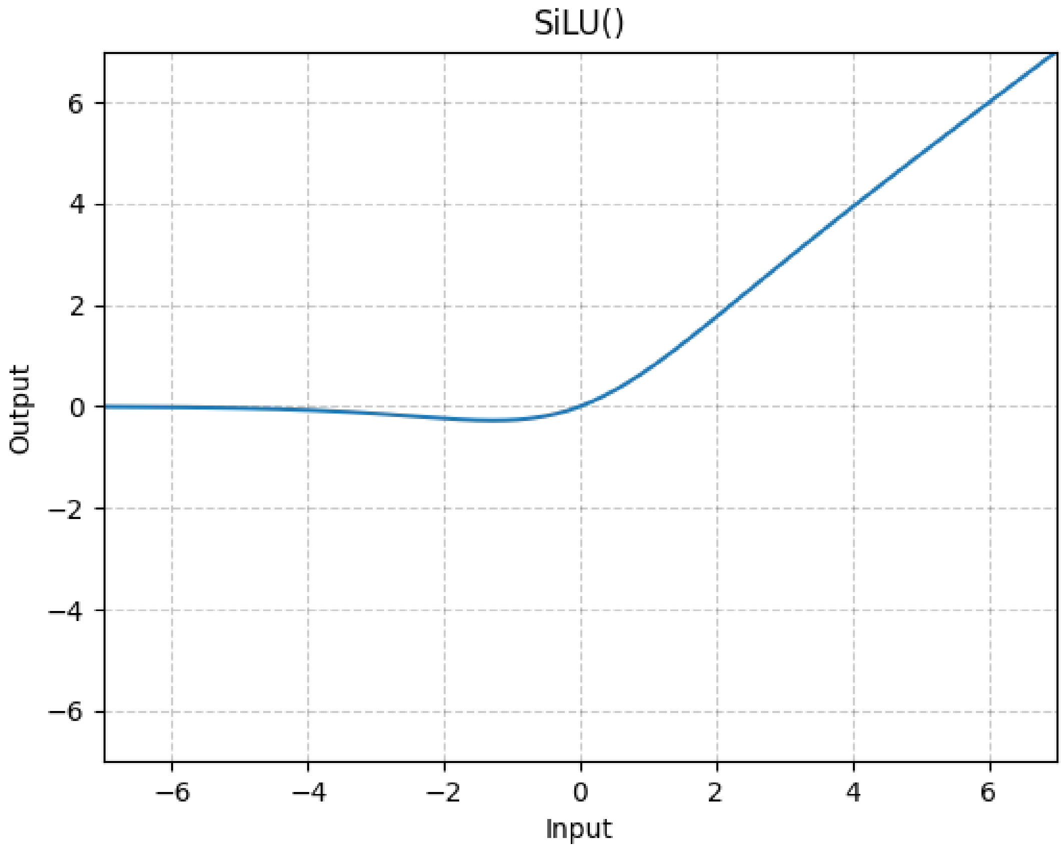

```
(conv_in): Conv2d(4, 320, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(time_proj): Timesteps()
(time_embedding): TimestepEmbedding
        (linear_1): Linear(in_features=320, out_features=1280, bias=True)
        (act): SiLU()
        (linear_2): Linear(in_features=1280, out_features=1280, bias=True)
(down_blocks):
        (0): CrossAttnDownBlock2D
        (1): CrossAttnDownBlock2D
        (2): CrossAttnDownBlock2D
        (3): DownBlock2D
(up_blocks):
        (0): UpBlock2D
        (1): CrossAttnUpBlock2D
        (2): CrossAttnUpBlock2D
        (3): CrossAttnUpBlock2D
(mid_block): UNetMidBlock2DCrossAttn
        (attentions):
        (resnets):
(conv_norm_out): GroupNorm(32, 320, eps=1e-05, affine=True)
(conv_act): SiLU()
(conv_out): Conv2d(320, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

SiLU: Sigmoid Linear Unit

- $\mathrm{silu}(x) = x\sigma(x)$
- $\sigma(x) = \frac{e^x}{1+e^x}$

- Gradients are not cut off before 0

- Can be more stable than ReLU.
- Higher computational cost.



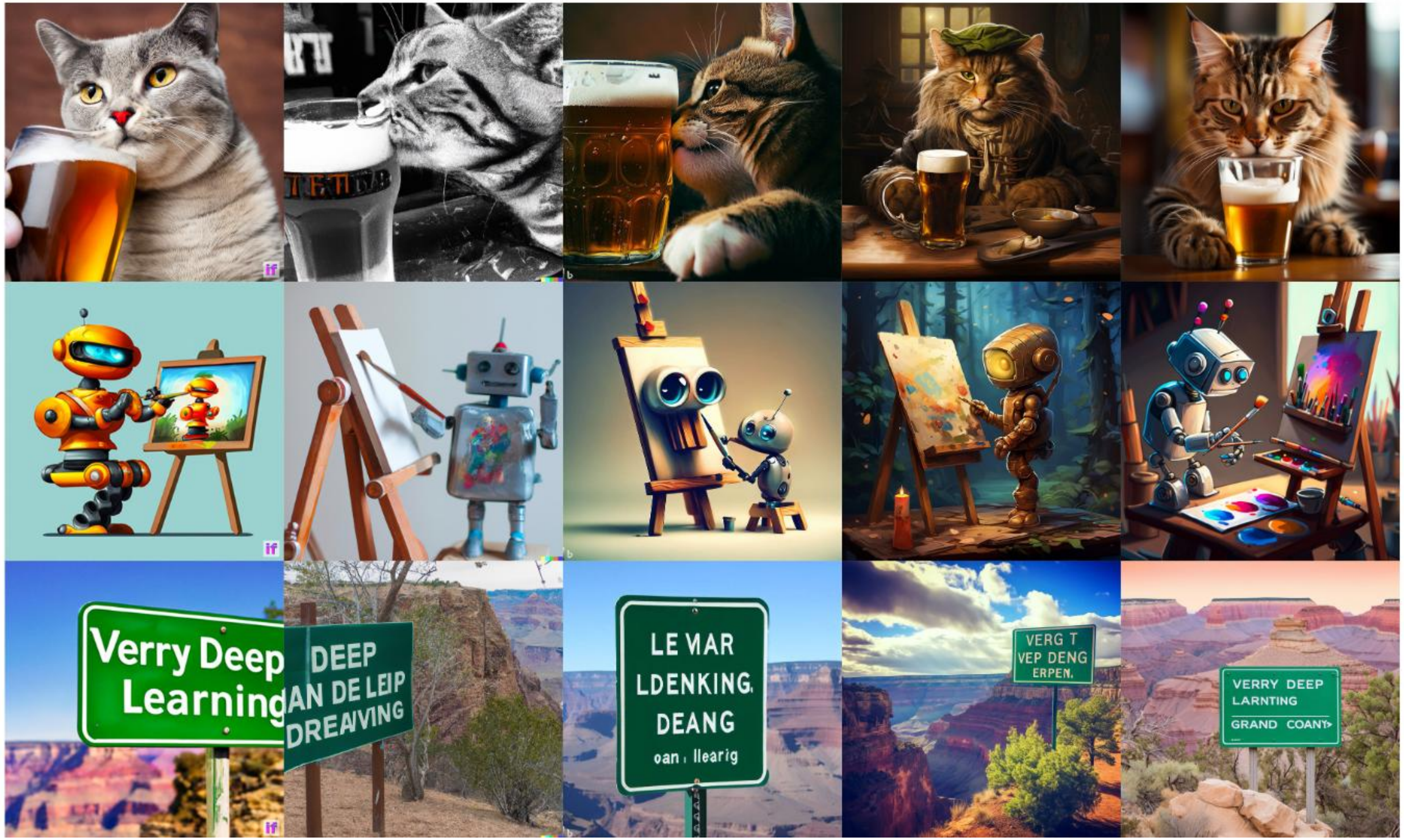
Elfwing et al., Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning, 2017

Large Scale Diffusion Models

a cat drinking a pint of beer

a cute robot artist painting on an easel concept art

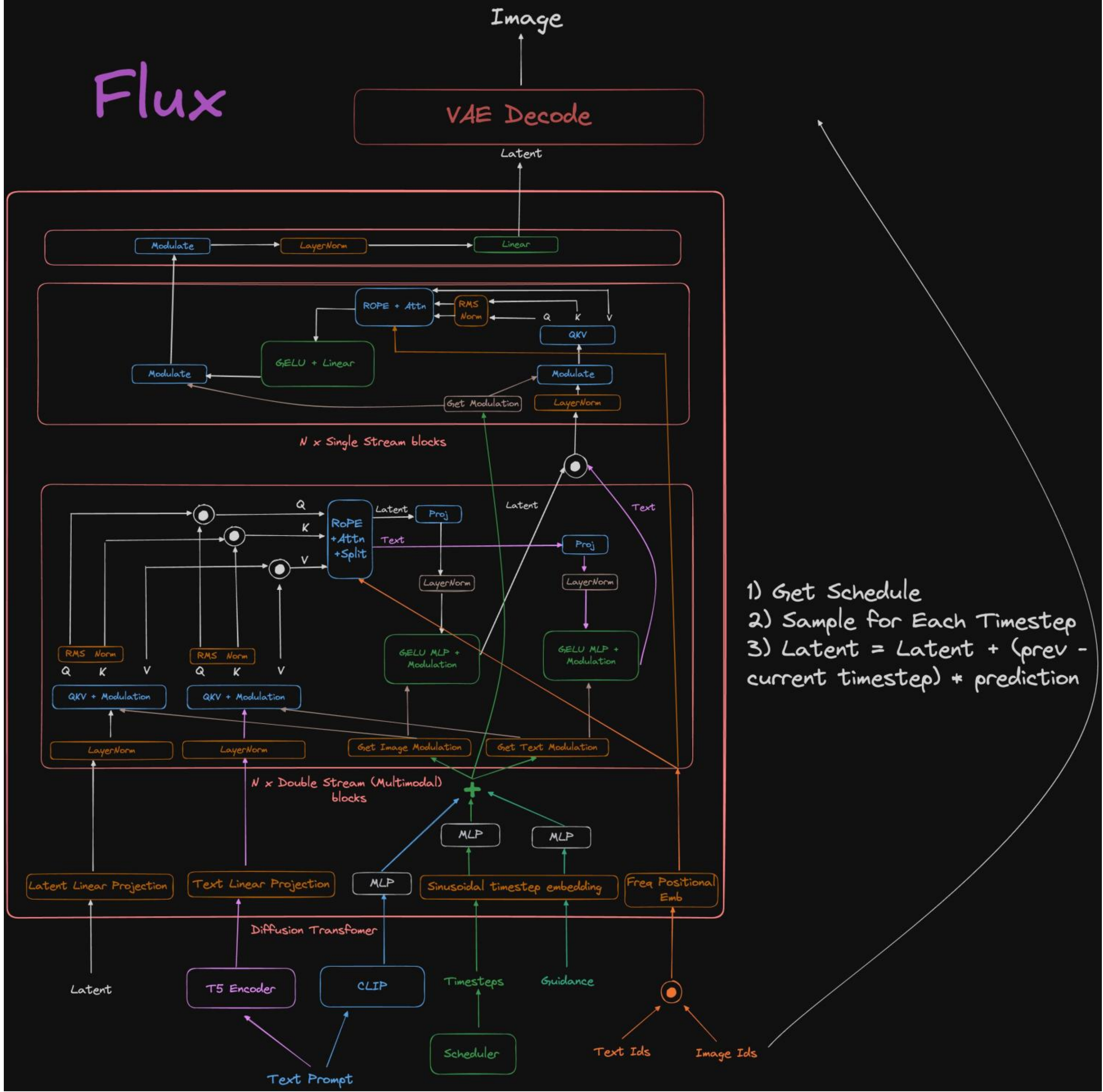
a green sign that says "Very Deep Learning" and is at the edge of the Grand Canyon



DeepFloyd IF | Dalle-2 | Bing | Midjourney | SDXL

Flux

- [Code/models]()
- Transformer-based architecture
- Latent diffusion model
- Flow-based model



[Diagram credit]() @nrehiew_

Learning better prompts

- Original: a cat drinking a pint of beer
- Enhanced: A whimsical feline sipping a frothy pint of golden ale, the condensation on the glass glistening in the warm light of a cozy pub, the cat's whiskers twitching as it savors the rich flavor and aroma of the beer, its paws curled around the glass as it sits on a worn wooden stool, surrounded by the rustic charm of a classic British pub.

flux-ai.org, Feb'2025

Evaluation

- Evaluation of explicit generative models is easy:
  - Measure $\log(p(x))$ on a test set.
  - Whichever model has higher probability wins.

- Evaluation of implicit models is very difficult.
  - We can only sample from the model.
  - We do not know how to measure the quality/probability of a sample.

Human Evaluation

- Ask people which model they prefer.

	win (%)	tie (%)	lose (%)
Example: Emu vs. SDXL	68.4	2.1	29.5

- Preference is vague.
- Also include faithfulness to text prompt, realism, etc.
- Difficult to scale/use to improve model.


Utensils, a bottle, and a glass positioned behind a stove


A decadent chocolate treat adorned with decorative sugar art

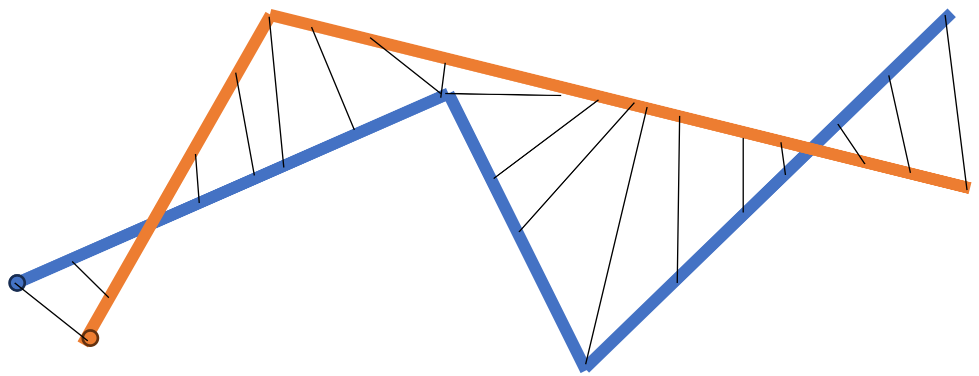

A beaver dressed in a vest, wearing glasses and a vibrant necktie, in a library


a cow eating a green leafy plant

Dai et al., Emu: Enhancing Image Generation Models Using Photogenic Needles in a Haystack, 2023
Podell et al., SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis, 2023

Fréchet Distance

- Dog and owner walk on separate paths.
- Can only go forward.
- FD: the shortest possible leash that allows both to complete the path.



Maurice Fréchet,1878-1973

Heusel et al., GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, 2017

FID: Fréchet Inception Distance

- Measure the *distance* between generated images and real images.
- Measure the distance in feature space (Inception v3 model).

- Input: feature extractor $f(I) \in \mathbb{R}^d$, real imges $\mathcal{I}_r$, samples $\mathcal{I}_g$.
- Compute $f(\mathcal{I}_r)$ and $f\left(\mathcal{I}_g\right)$
- Fit Gaussians to each set: $\mathcal{N}(\mu_r, \Sigma_r)$, $\mathcal{N}(\mu_g, \Sigma_g)$
- $d_F = \left\|\mu_r - \mu_g\right\|_2^2 + \mathrm{tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{\frac{1}{2}}\right)$

Heusel et al., GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, 2017

FID

- Real images: pick a dataset (e.g. ImageNet)
- Aligns to a certain degree with human judgement.
- Is a popular metric and often used.
- We know it is not very good, but we don’t have good alternatives.
- Many others have been proposed but there is no clear winner.

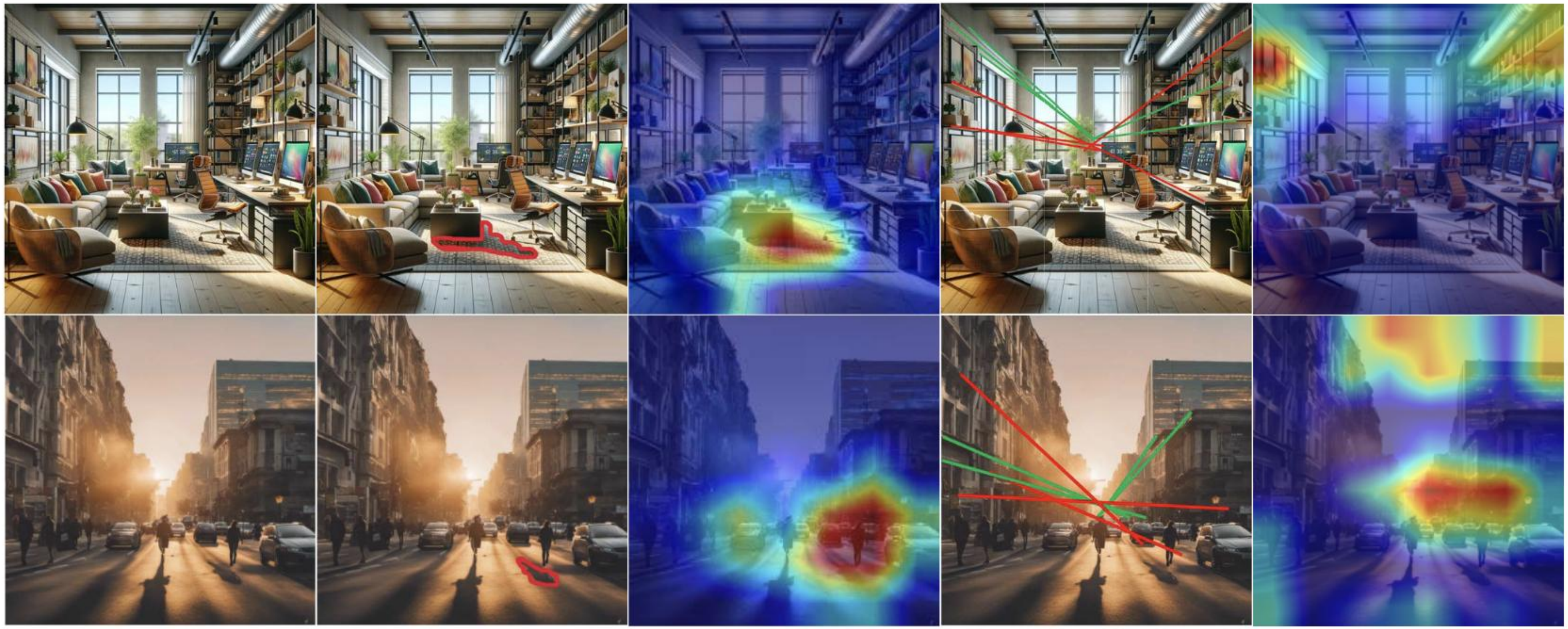
Video Diffusion Models

- Image architecture 2D U-Net.
- Video architecture 3D U-Net.
- Maybe: share 2D U-Net across frames, add time attention.
- Train on 2D, then video.

Blattmann et al., Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets, 2023

Shadows Don't Lie and Lines Can't Bend!
Generative Models don't know Projective Geometry...for now

Ayush Sarkar*[1] Hanlin Mai*[1] Amitabh Mahapatra*[1] Svetlana Lazebnik[1]
D.A. Forsyth[1] Anand Bhattad[2]
[1]University of Illinois Urbana-Champaign [2]Toyota Technological Institute at Chicago
https://projective-geometry.github.io/



Generated Image | Shadow Errors | Detected Shadow Errors | Vanishing Point Errors | Detected Perspective Errors

Generative Models + Geometry

- Instead of reconstructing a scene, can we just generate new views?

- Take camera parameters (extrinsics and intrinsics) as input!