CIS 365 Applied AI

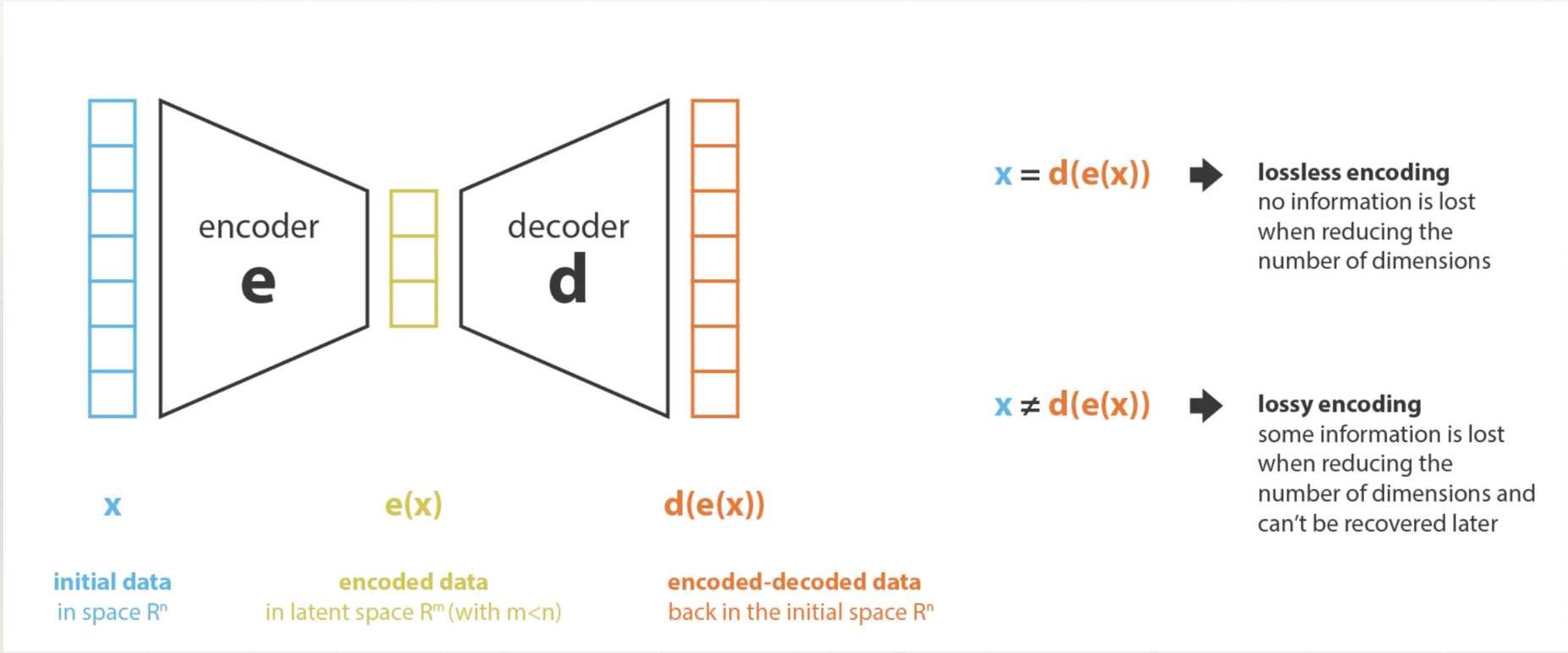
Generative Networks

Dr. Denton Bobeldyk

Topics

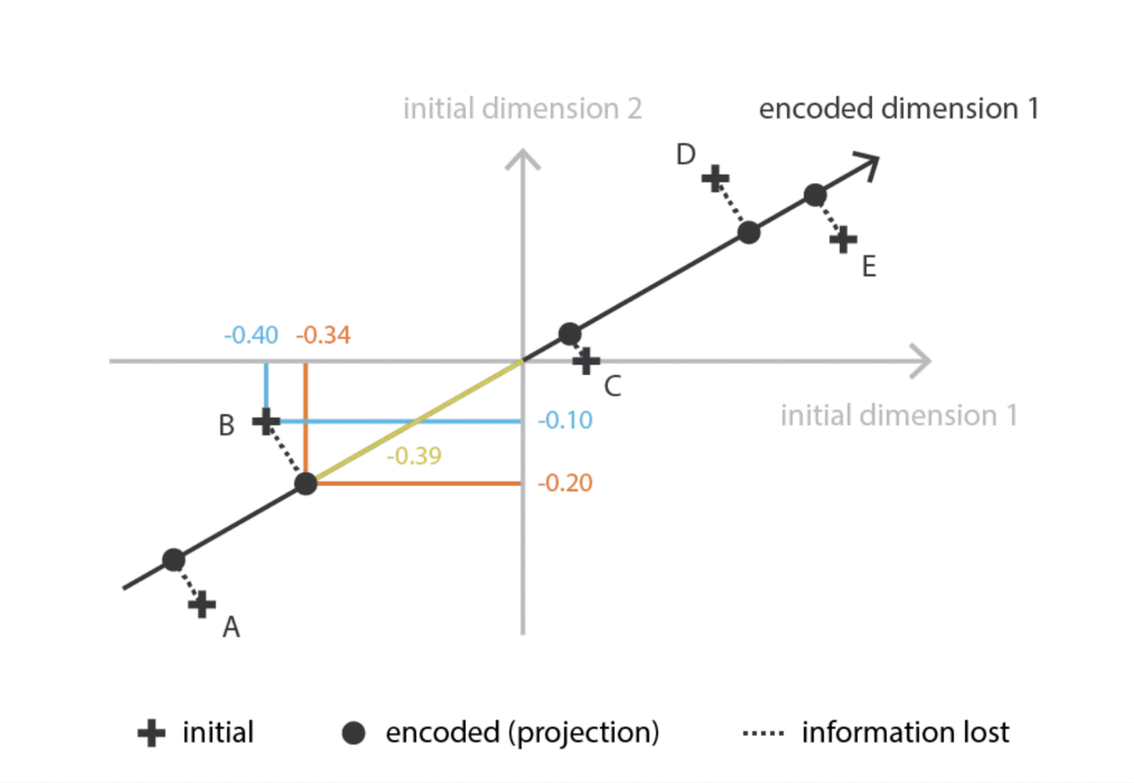
- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models

Variational Autoencoders (VAEs)



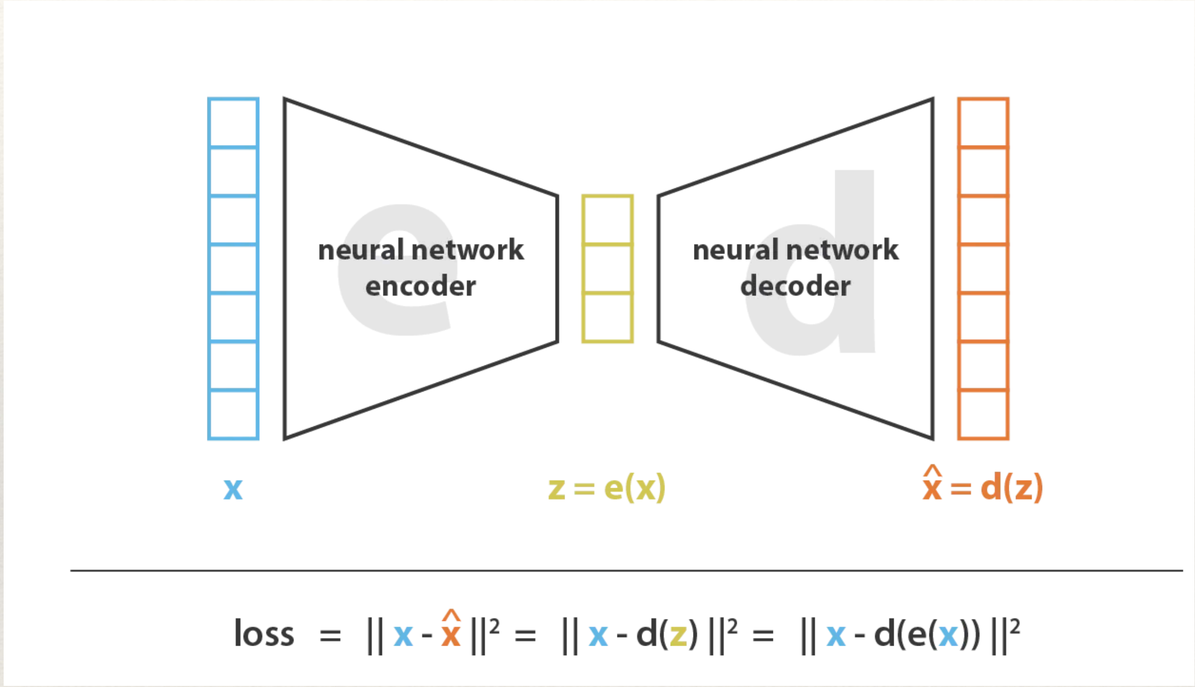
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)

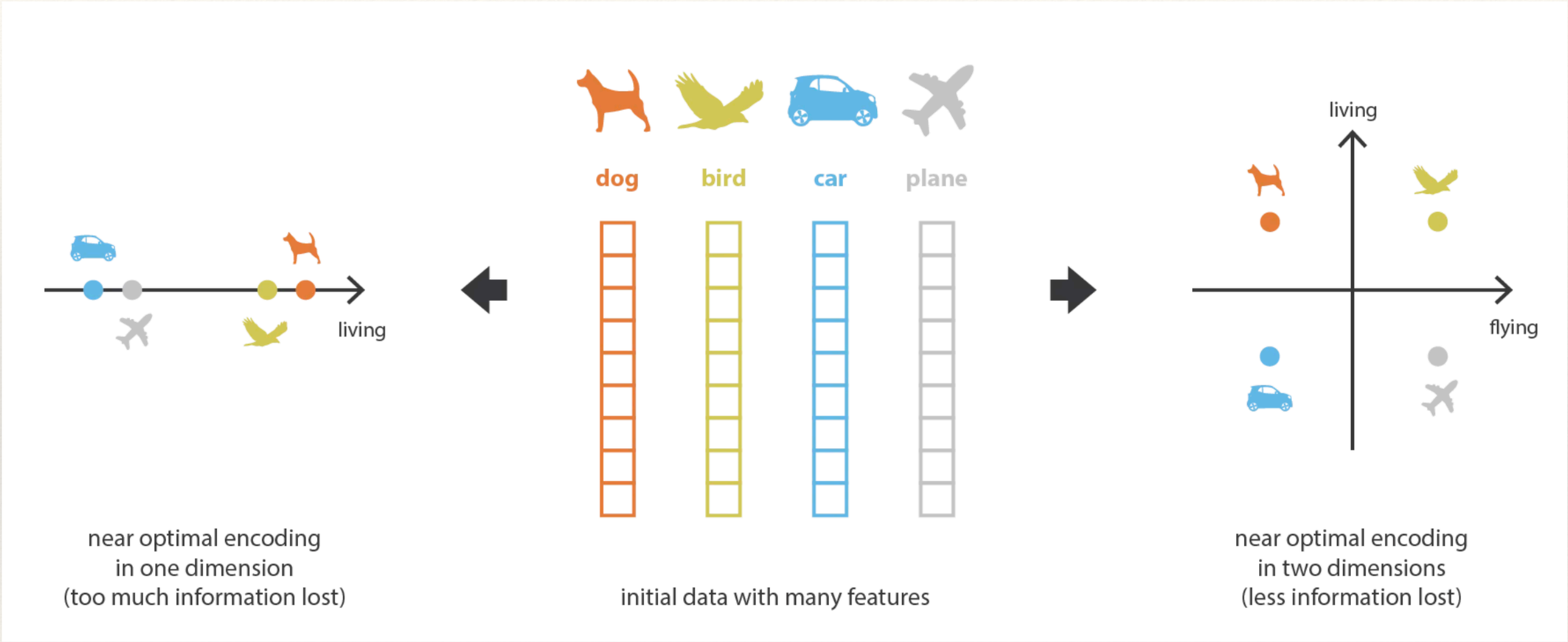


$$\|x - \hat{x}\|^2 = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2$$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

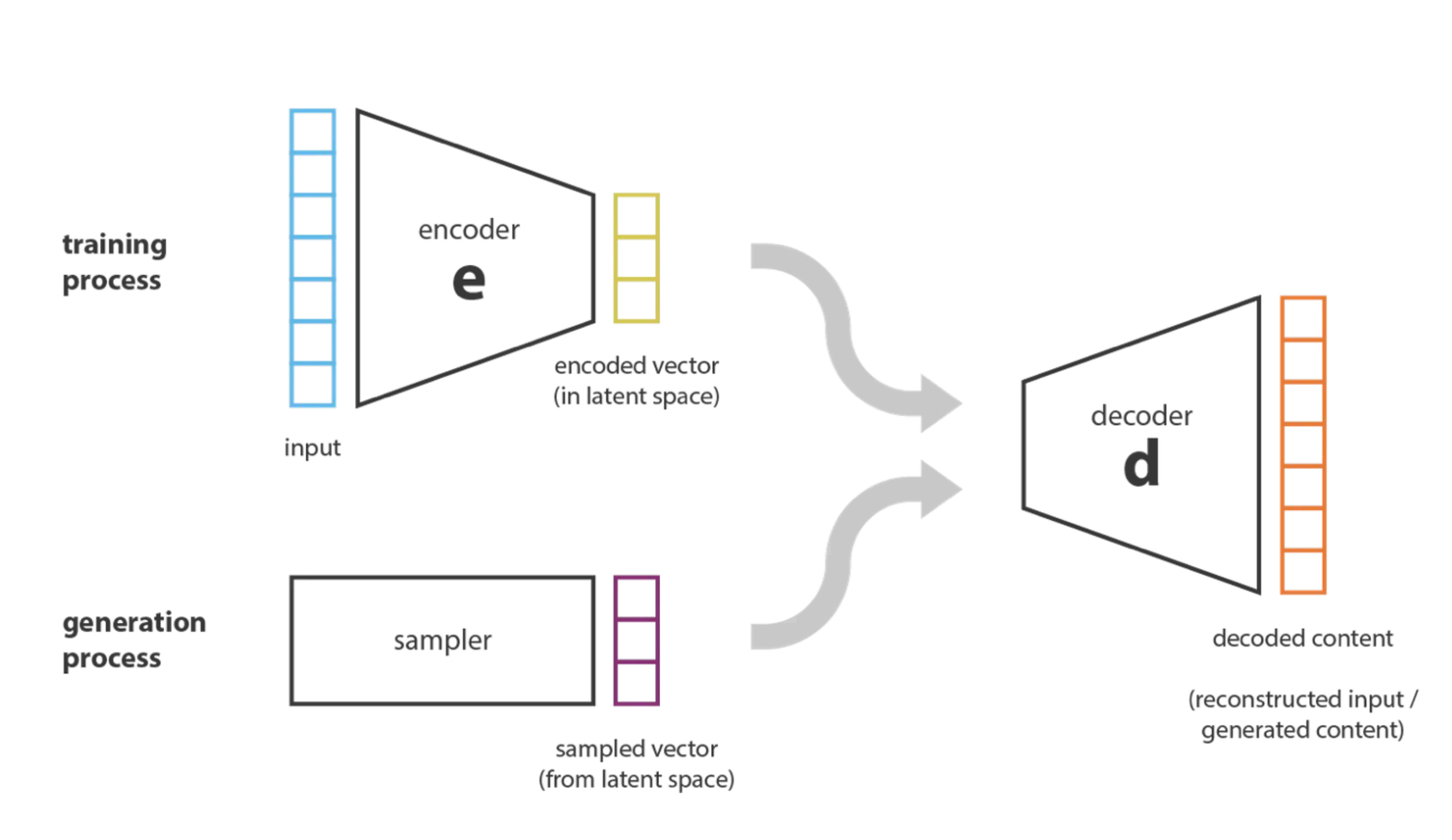
Slides created by Dr. Denton Bobeldyk

Variational Autoencoders (VAEs)



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

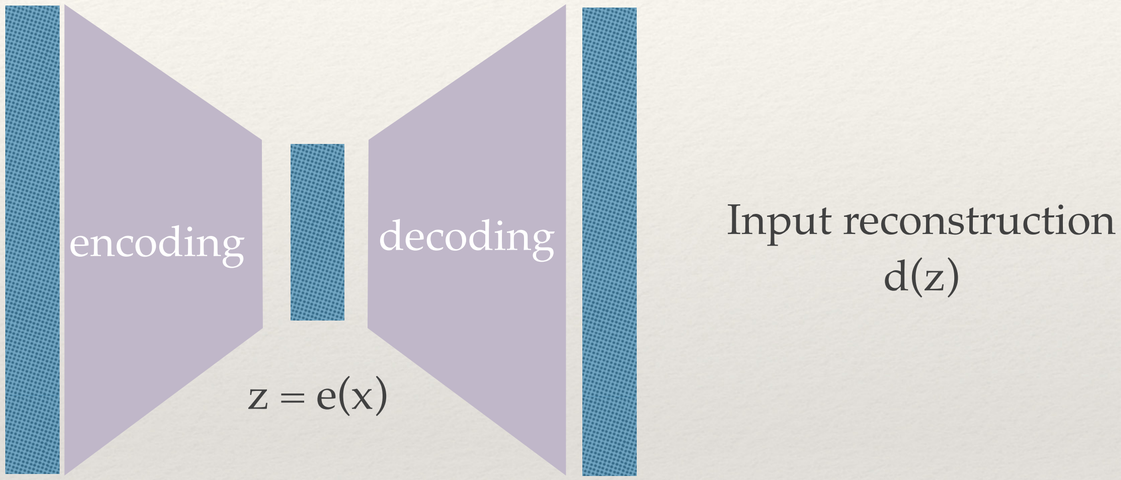
Variational Autoencoders (VAEs)



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)

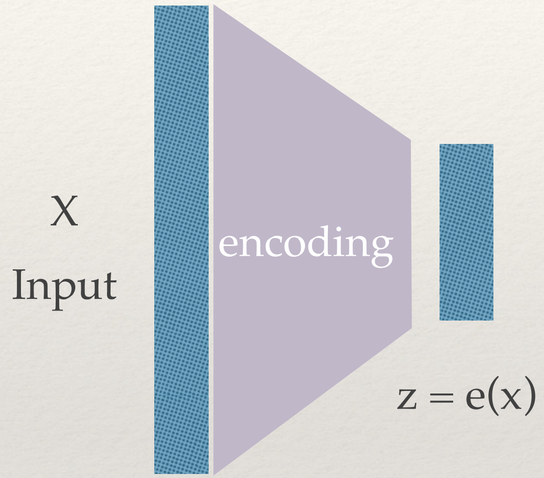
Simple Autoencoders



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)

Simple Autoencoders

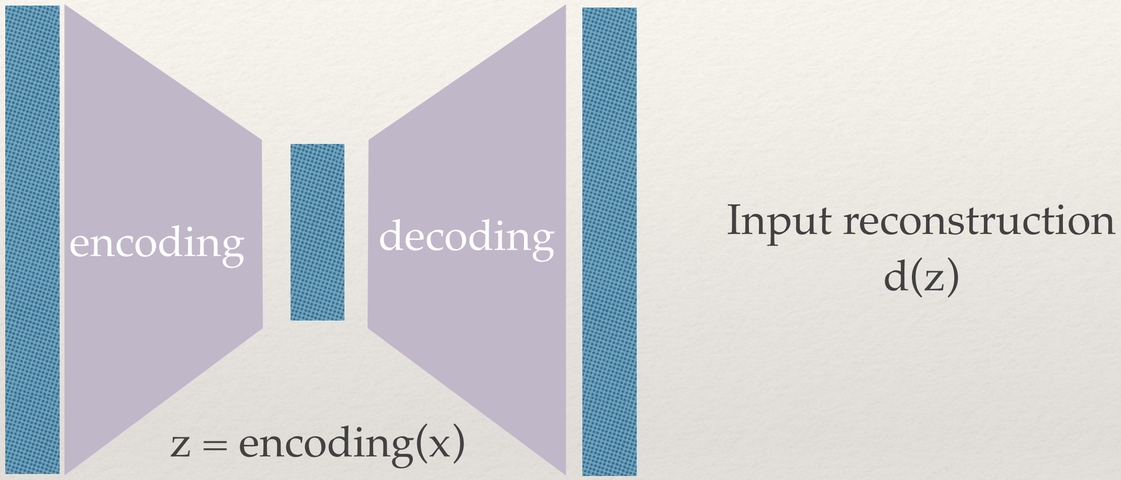


The encoder approximates the posterior distribution p(z|x) using a simpler distribution q(z|x), typically parameterized by a neural network.

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)

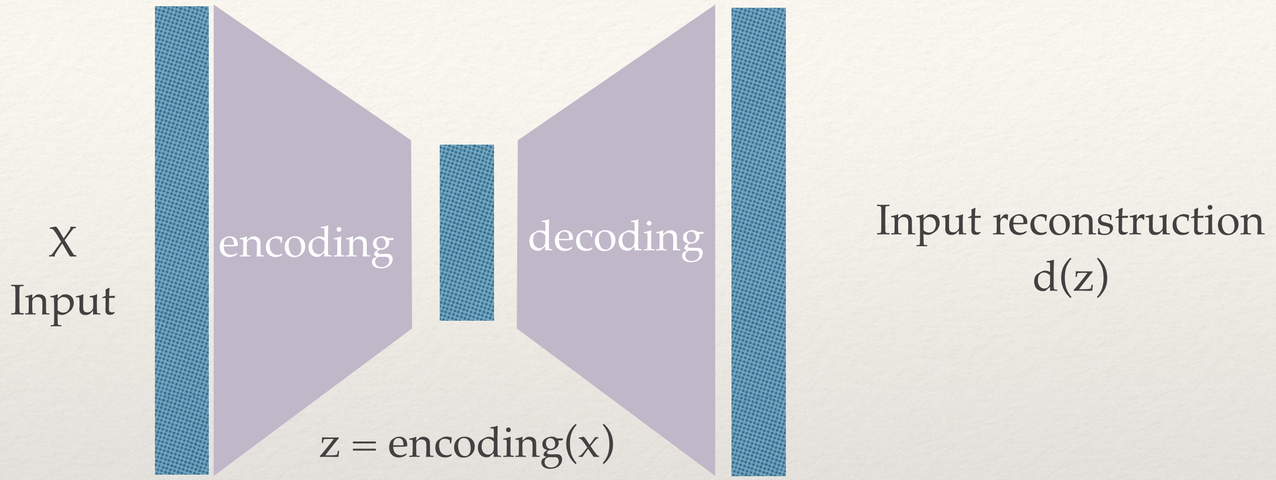
Simple Autoencoders



The goal of a VAE is to model the distribution of the observed data p(x) using the latent variable z

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Autoencoder



$p(x) = \int p(x \mid z)p(z)\,dz$ is often intractable so the autoencoder architecture can be used to approximate

p(x) = The marginal likelihood or evidence of the observed data x
p(x|z) = The likelihood of the data given the latent variable z
p(z) = The prior distribution over the latent variable z
$\int \ldots z$ the integral over all possible values of z

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Autoencoder

$p(x) = \int p(x \mid z)p(z)\, dz$ is often intractable so the autoencoder architecture can be used to approximate

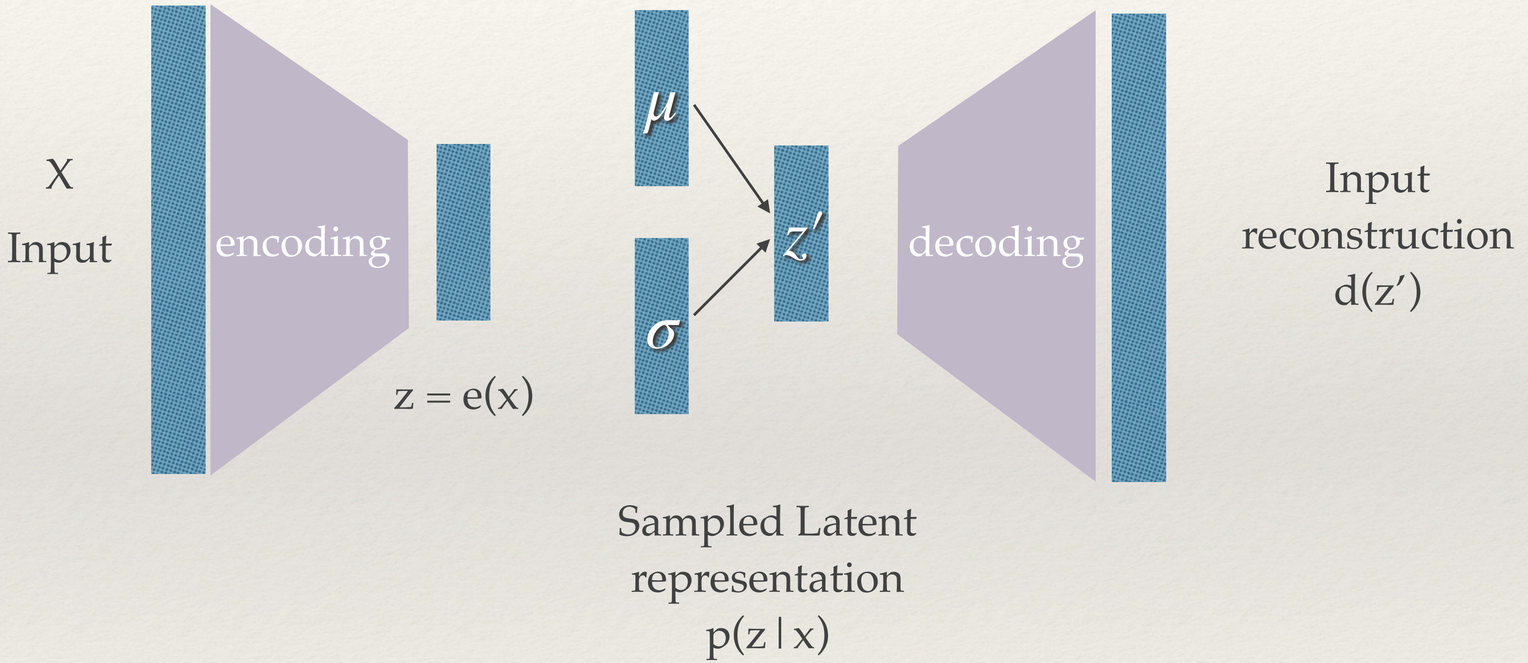
Why is it intractable?

This integral is typically intractable because it requires summing over a high-dimensional latent space. VAEs approximate this using Evidence Lower Bound (ELBO) and the reparameterization trick

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)

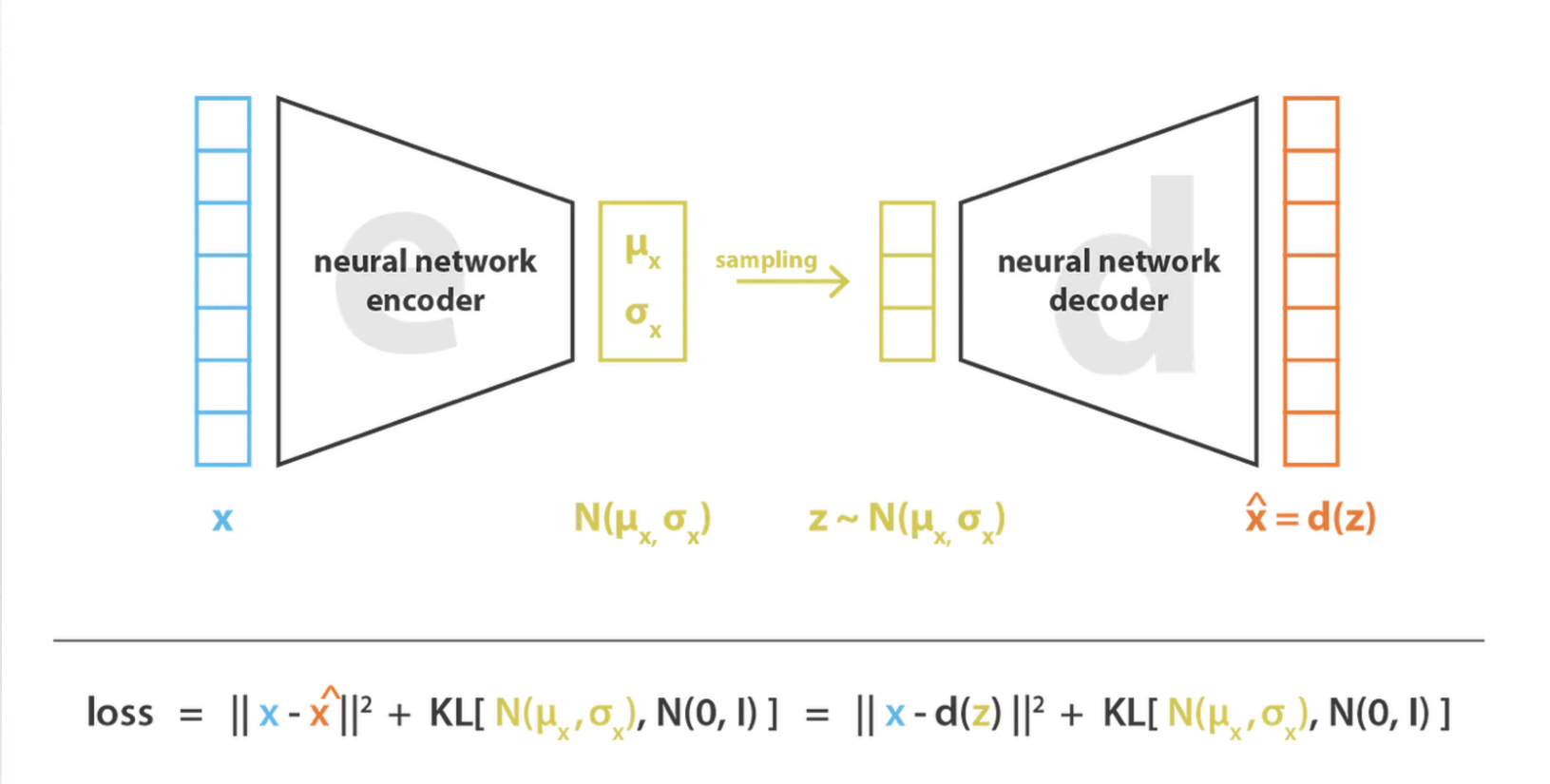
Simple Autoencoders



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

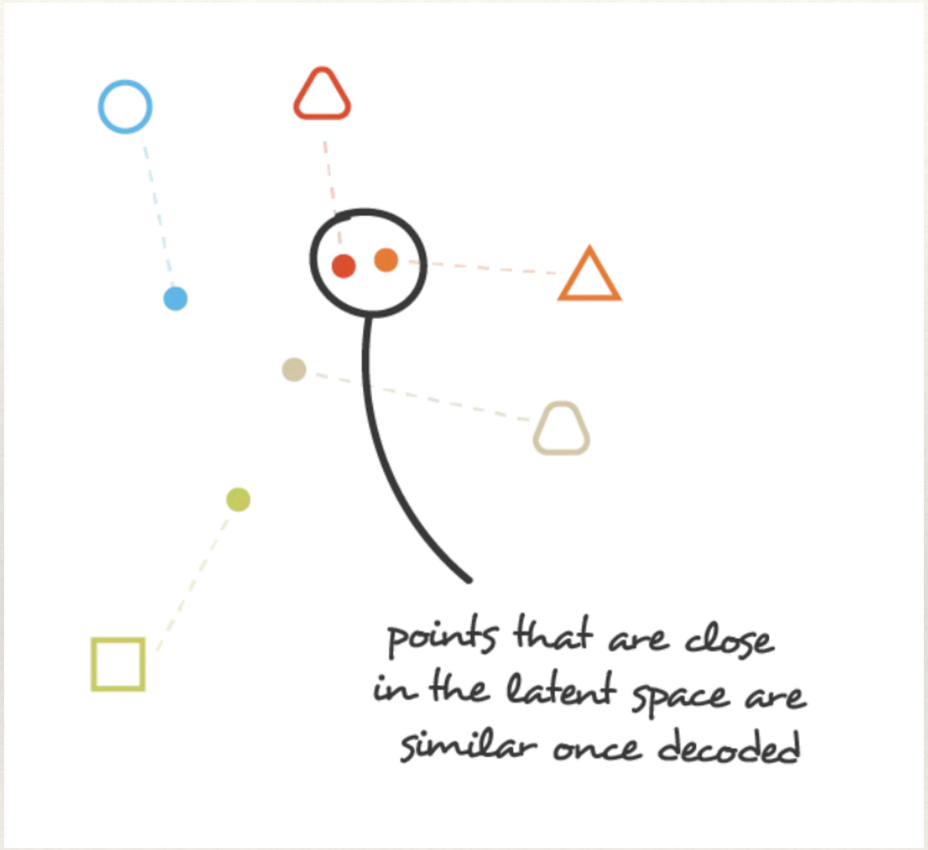
Slides created by Dr. Denton Bobeldyk

Variational Autoencoders (VAEs)



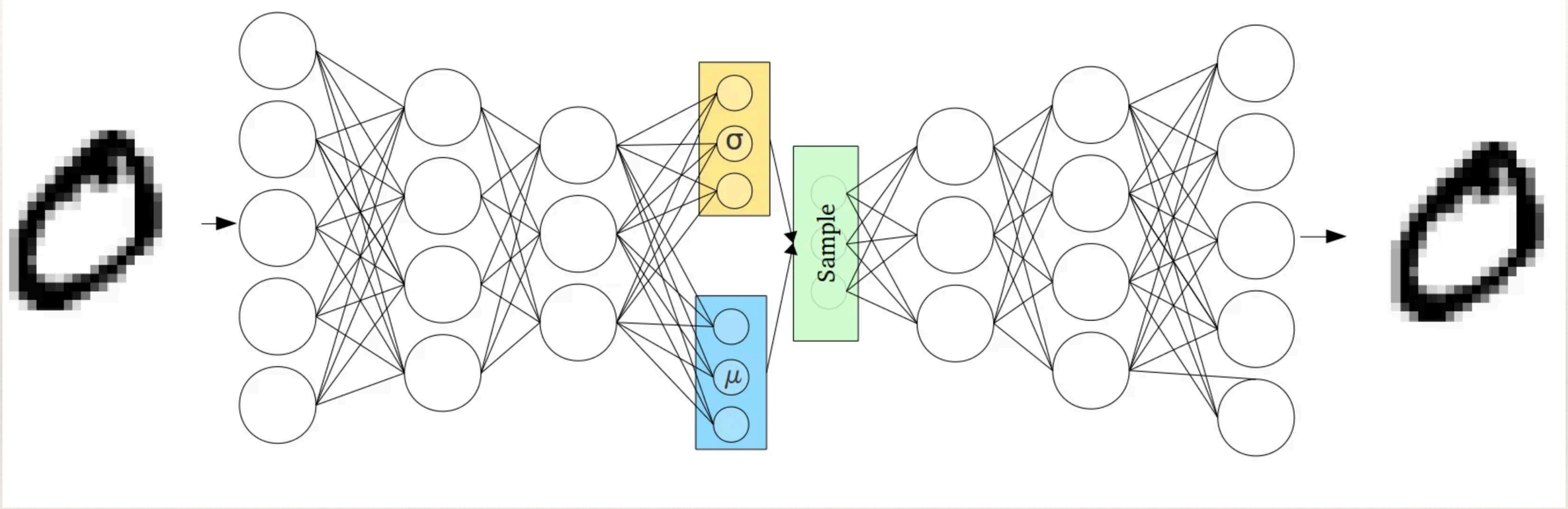
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs)



https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Reparameterization Trick

- Offers a solution to overcome the problem of non-differentiability that arises when sampling random variables like in the VAE
- Back-propagation is typically used for training and gradients can’t flow through random variables.

Reparameterization Trick

- Instead of directly sampling from a distribution, the reparameterization trick allows us to break the sampling process into two steps:
  - Generate a random noise ϵ from a simple, fixed distribution (like a standard normal distribution $\mathcal{N}(0,1)$)
  - Transform this noise using learned parameters (mean μ and standard deviation σ) to create the variable z:

$$z = \mu + \sigma \cdot \epsilon$$

Reparameterization Trick

- During the training, the VAE optimizes:
  - The reconstruction loss (ensuring the decoder can reconstruct the input x from z)
  - The KL divergence term, which ensures $q(z \mid x)$ stays close to a standard normal distribution $p(z) = N(0,1)$
  - The encoder learns the optimal μ and σ that minimizes this combined loss, making them the learned parameters

Reparameterization Exercise

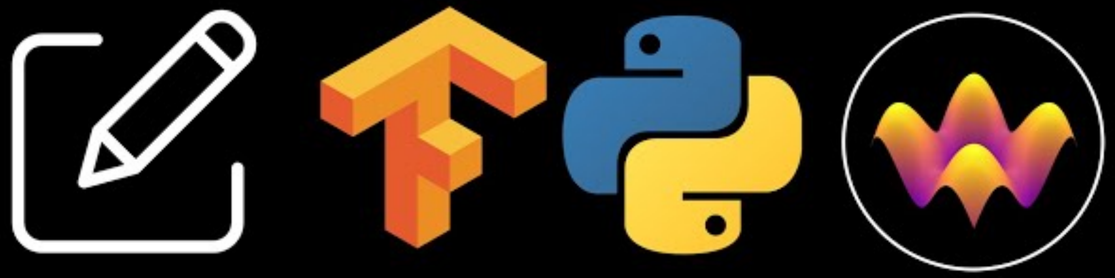
Dice Exercise

Kullback Leibler (KL) Divergence

Intuitively Understanding the KL divergence

$$D_{KL}(P||Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$$

$$p_1\log\frac{p_1}{q_1} + p_2\log\frac{p_2}{q_2}$$



[english]

Kullback-Leibler (KL) Divergence

Introduction & Example

w\ example in TensorFlow Probability

$$KL(p||q) = \mathbb{E}_{x\sim p(x)}\left[\log \frac{p(x)}{q(x)}\right] = \begin{cases} \sum_x p(x)\log\frac{p(x)}{q(x)}, & \text{if } X \text{ discrete} \\ \int_x p(x)\log\frac{p(x)}{q(x)}\,dx, & \text{if } X \text{ continuous} \end{cases}$$

Conditional VAE's

- Generating data based on labels

Heirarchical VAEs

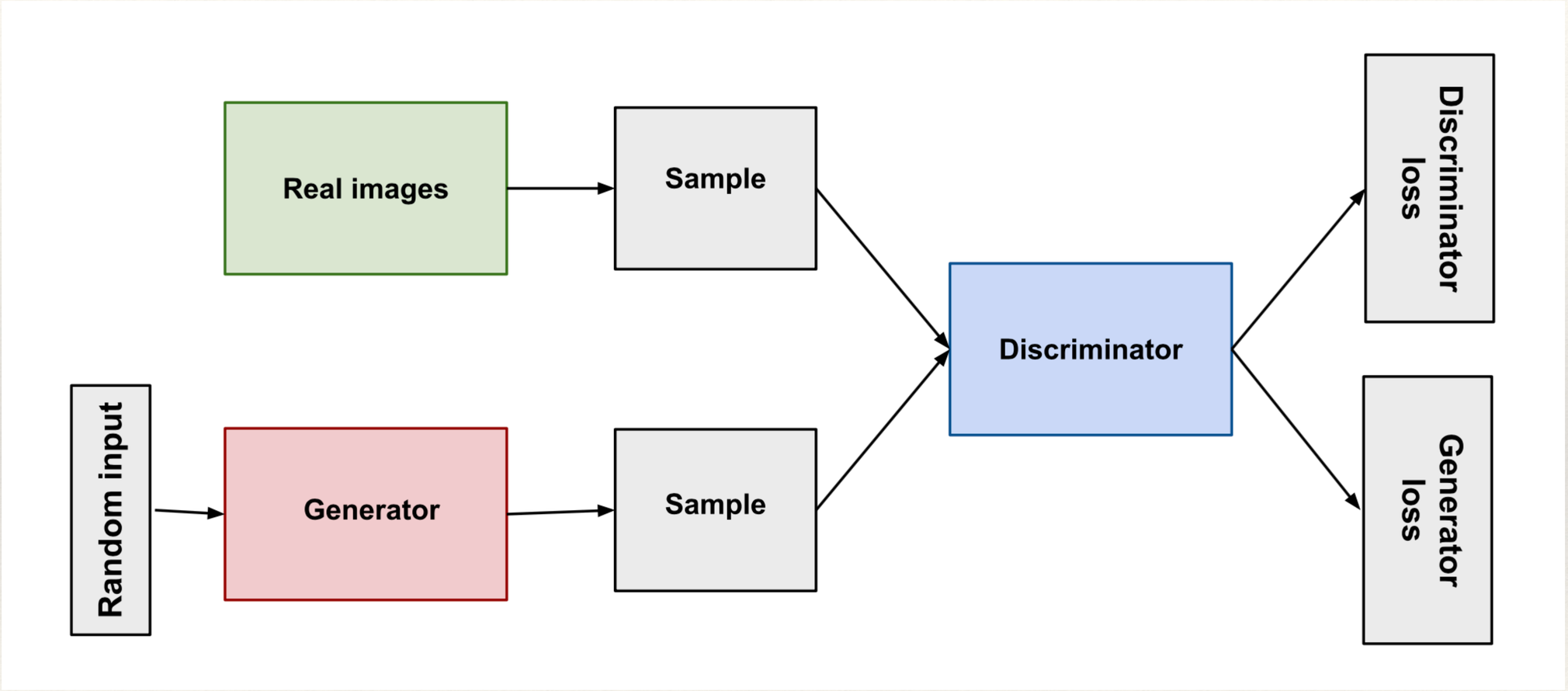
- Generating data based on multiple layers of latent spaces

VAE Challenges

- Blurred output - can be difficult to generate sharp images
- Mode collapse
- Scalability - computational challenges

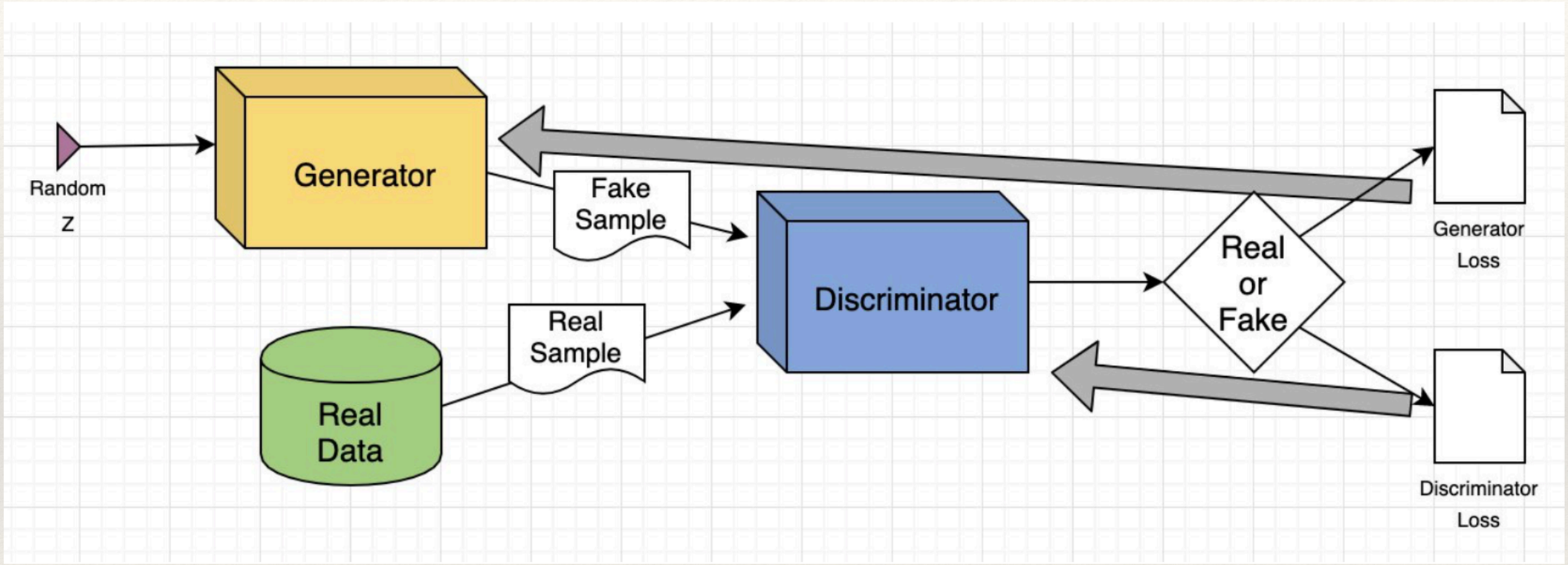
Generative Adversarial Networks (GANs)

Generative Adversarial Network



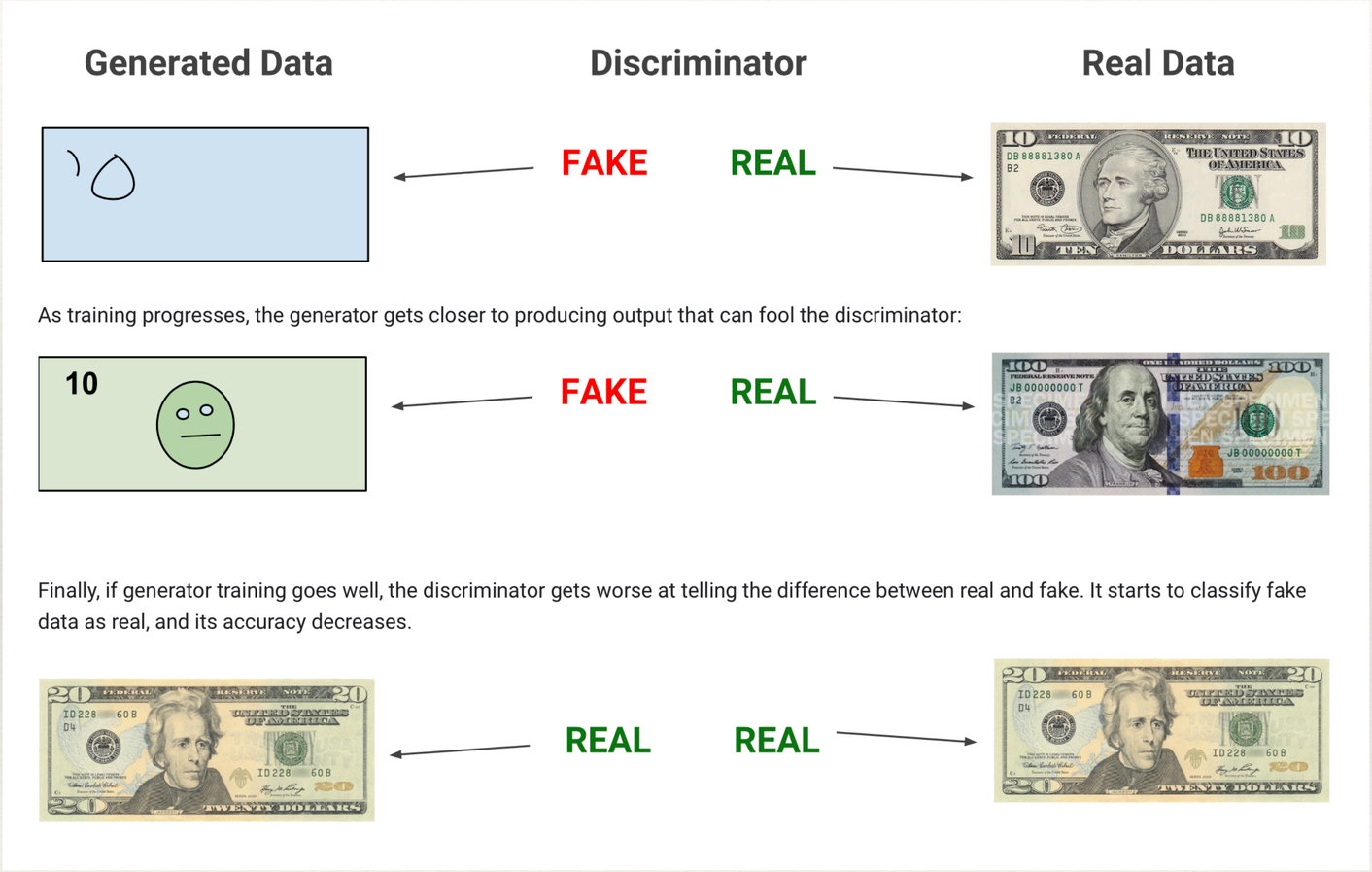
https://developers.google.com/machine-learning/gan/gan_structure

Generative Adversarial Network



https://aws.amazon.com/what-is/gan/

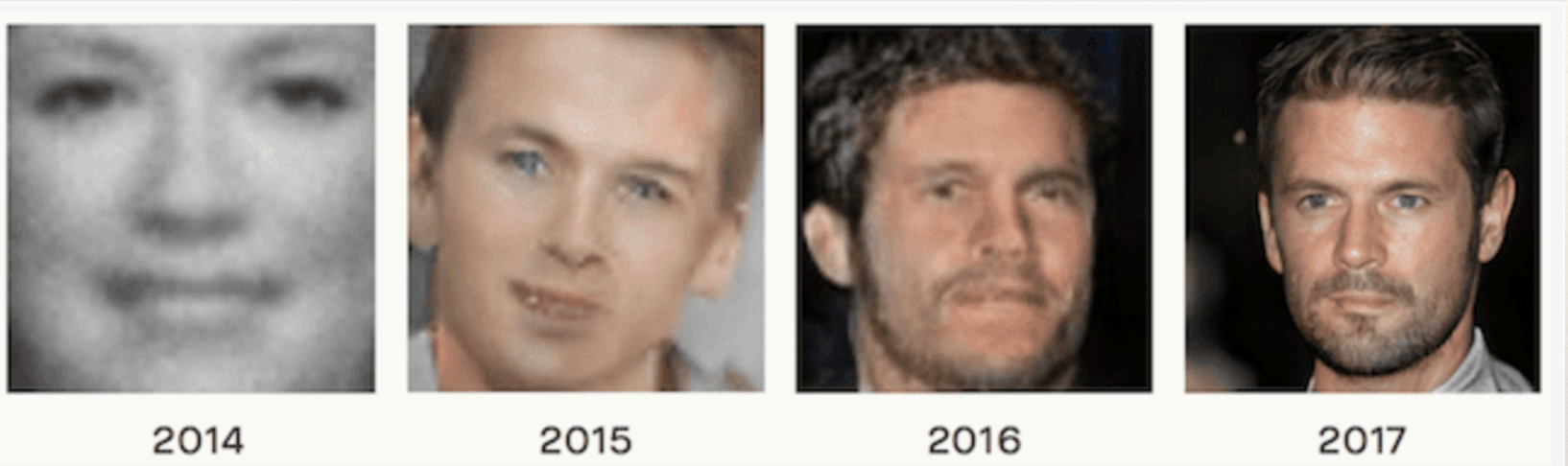
Generative Adversarial Network



https://developers.google.com/machine-learning/gan/gan_structure

Slides created by Dr. Denton Bobeldyk

Generative Adversarial Network



Example of the Progression in the Capabilities of GANs From 2014 to 2017. Taken from The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation, 2018.

https://developers.google.com/machine-learning/gan/gan_structure

Original GAN paper

https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

GAN Applications

- Image generation and enhancement
  - Super-resolution: improving the quality of low-resolution images (e.g., enhancing satellite images or restoring old photos)
  - Style transfer: applying artistic styles to photos or creating hybrid images that combine multiple styles
  - Synthetic image generation: creating realistic images of people, objects, or scenes (e.g. generating faces with tools like thispersondoesnotexist.com)

GAN Applications

- Video applications
  - Video prediction: predicting future frames in a video sequence for motion analysis
  - Frame interpolation: creating smoother transitions by generating intermediate frames between two existing frames
  - Deepfake generation: generating realistic face swaps in videos (used in both entertainment and malicious applications)

GAN Applications

- Audio applications
  - Music generation: composing music by learning patterns from existing tracks
  - Speech synthesis: generating human-like voices for AI assistants (e.g., generating specific accents or notations)
  - Speech enhancement: denoising or restoring degraded audio signals

GAN Applications

- Text-to-image conversion
  - GANs are used in combination with other models (e.g., transformers) to generate images from textual descriptions (e.g., “a cat sitting on a beach with a sunset background”)

GAN Applications

- Healthcare and biomedical imaging
  - Medical image synthesis: generating synethic MRI or CT images for training diagnostic systems
  - Data augmentation: expanding datasets by creating realistic medical images to improve machine learning models
  - Disease detection: enhancing medical image quality for more accurate diagnoses

GAN Applications

- Gaming and Virtual Reality
  - Character and scene design: generating realistic characters, environments, and textures in games
  - Dynamic content generation: creating procedurally generated content that adapts to gameplay

GAN Applications

- Art and Creativity
  - Digital Art: assisting artists by generating creative artworks or suggesting design ideas
  - Interactive art tools: allowing users to draw simple sketches that GANS turn into detailed images

GAN Applications

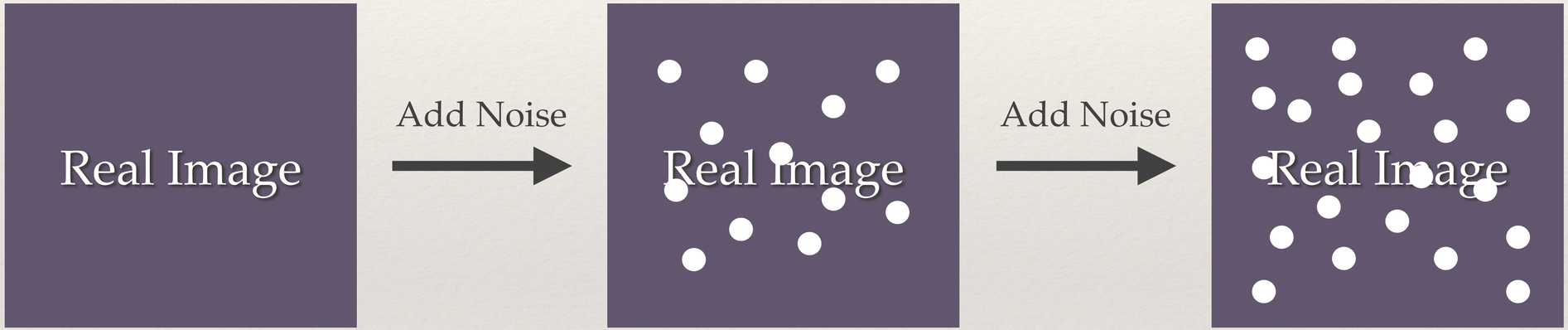
- Research and education
  - Simulation of physics: generating simulated data for scientific research, such as fluid dynamics or particle physics
  - Training data generation: creating synthetic datasets for machine learning when real data is scarce or expensive to collect

GAN Applications

- Security and privacy
  - Anonymization: replacing faces in videos with synthetic ones to maintain privacy while preserving context
  - Forgery detection: developing adversarial systems to detect GAN generated fake media

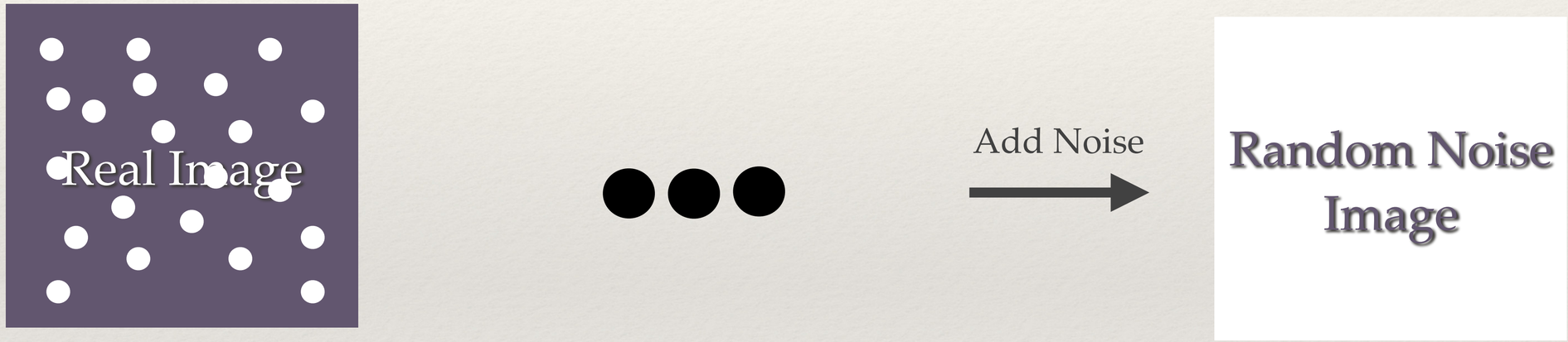
Diffusion Models

Diffusion models

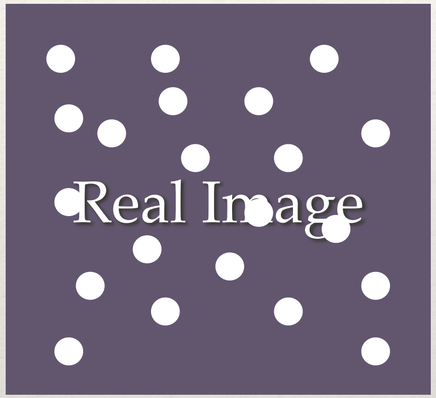


Slides created by Dr. Denton Bobeldyk

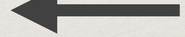
Diffusion models

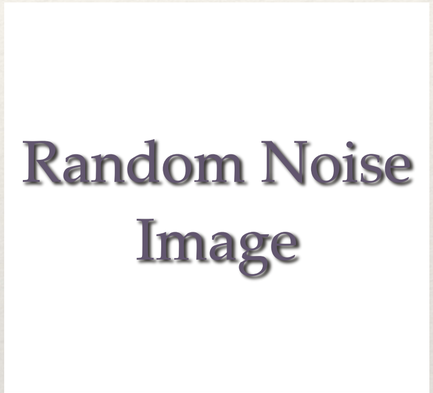


Diffusion models

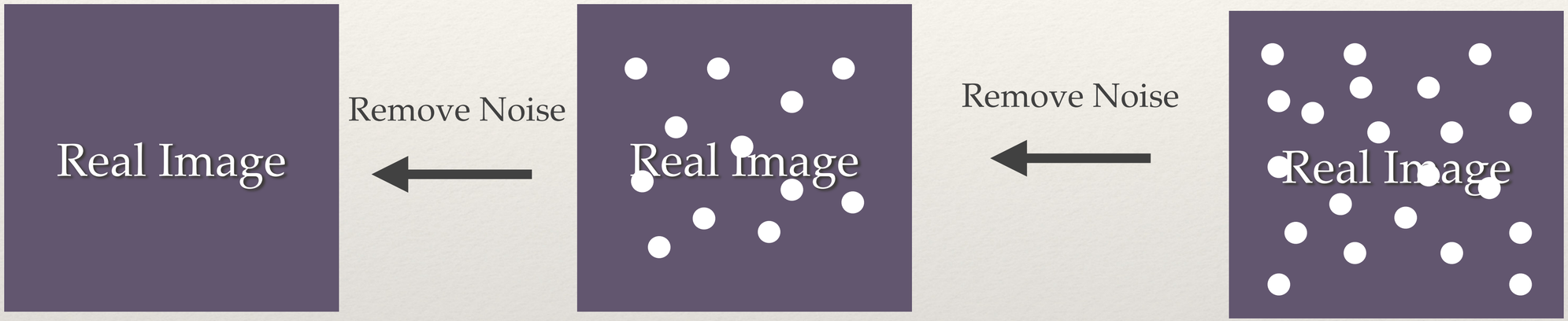


Remove Noise





Diffusion models



Diffusion models

- By learning the reverse process, the model can generate realistic images from scratch by starting with random noise

Diffusion models

- The forward process is designed to corrupt the data with Gaussian noise progressively

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} \cdot x_{t-1}, (1 - \alpha_t) \cdot \mathbf{I})$$

- Starting with Clean data x_0 noise is gradually added over time T time steps to transform the data into pure noise x_T

Diffusion models

- The forward process is designed to corrupt the data with Gaussian noise progressively

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} \cdot x_{t-1}, (1 - \alpha_t) \cdot \mathbf{I})$$

$q(x_t \mid x_{t-1})$ = The probability of the noisy data x_t given the noisy data x_{t-1} from the previous step

Diffusion models

- The forward process is designed to corrupt the data with Gaussian noise progressively

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} \cdot x_{t-1}, (1 - \alpha_t) \cdot \mathbf{I})$$

$\mathcal{N}(x; \mu, \sigma^2)$ = The normal distribution with mean μ and variance σ^2

Diffusion models

- The forward process is designed to corrupt the data with Gaussian noise progressively

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} \cdot x_{t-1}, (1 - \alpha_t) \cdot \mathbf{I})$$

$\sqrt{\alpha_t} \cdot x_{t-1}$ The mean is a scaled version of the previous step's data, x_{t-1}. The scaling factor $\sqrt{\alpha_t}$ controls how much of the original data is retained as noise is added. α_t is typically part of a predefined noise schedule that determines the rate of noise addition over time

Diffusion models

- The forward process is designed to corrupt the data with Gaussian noise progressively

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} \cdot x_{t-1}, (1 - \alpha_t) \cdot \mathbf{I})$$

$(1 - \alpha_t) \cdot \mathbf{I}$ The variance represents the amount of noise added at each step. It increases as t progresses, ensuring the data gets noisier over time

Diffusion models

The forward process defines how noise is systematically added to the data. Understanding this process is critical because the **reverse process**—which denoises x_t to x_{t-1} is what the diffusion model learns to approximate during training. By accurately modeling this reverse process, the model can generate realistic data samples from noise.

Diffusion models

The model learns the reverse transition

$$p_\theta(x_{t-1} \mid x_t)$$

Where θ is parameterized by a neural network

Diffusion models

The model is trained to predict the noise that was added at each step. It uses a loss function:

$$L = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

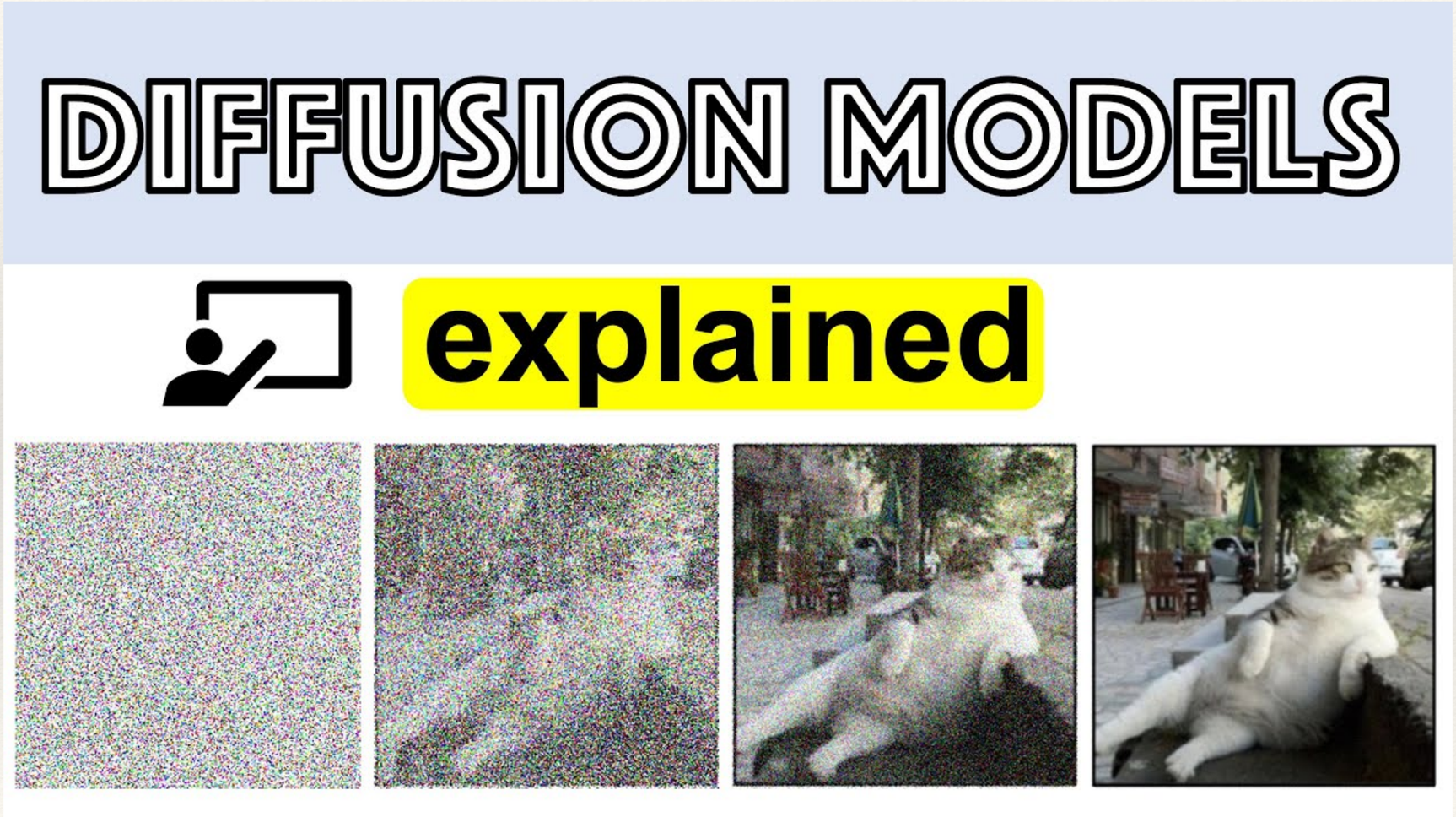
ϵ	The actual noise added during the forward process
$\epsilon_\theta(x_t, t)$	The noise predicted by the model at time step t
$\|\epsilon - \epsilon_\theta(x_t, t)\|^2$	The squared error between the actual noise and the predicted noise
$\mathbb{E}_{t,x_0,\epsilon}$	Expectation over time steps t, the original data x_0, and noise ϵ

Slides created by Dr. Denton Bobeldyk

Diffusion Models

Slides created by Dr. Denton Bobeldyk

Diffusion Models



Slides created by Dr. Denton Bobeldyk

End