EE-559

Deep Learning

What's on today?

- Deterministic autoencoders: on discovering structure within the data
- Variational autoencoders: on using a continuous latent space
- Vector quantised-variational autoencoder: on using discrete latents
- Diffusion probabilistic models: on learning noise removal
- Instruction-following diffusion models: on editing with language
- Exercises: autoencoders and variational autoencoders

Deterministic autoencoders

Auto-associative neural network

encoder

maps the
input data $\boldsymbol{x}$
to a hidden representation $\boldsymbol{z}$

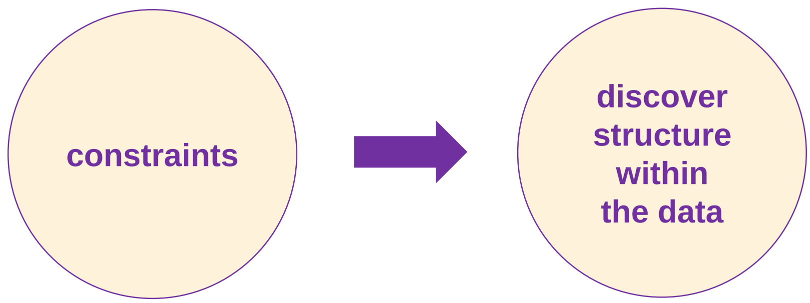
decoder

maps the
hidden representation $\boldsymbol{z}$
to the output $\boldsymbol{y}$

trained to generate an output $\boldsymbol{y}$ that is as close as possible to $\boldsymbol{x}$

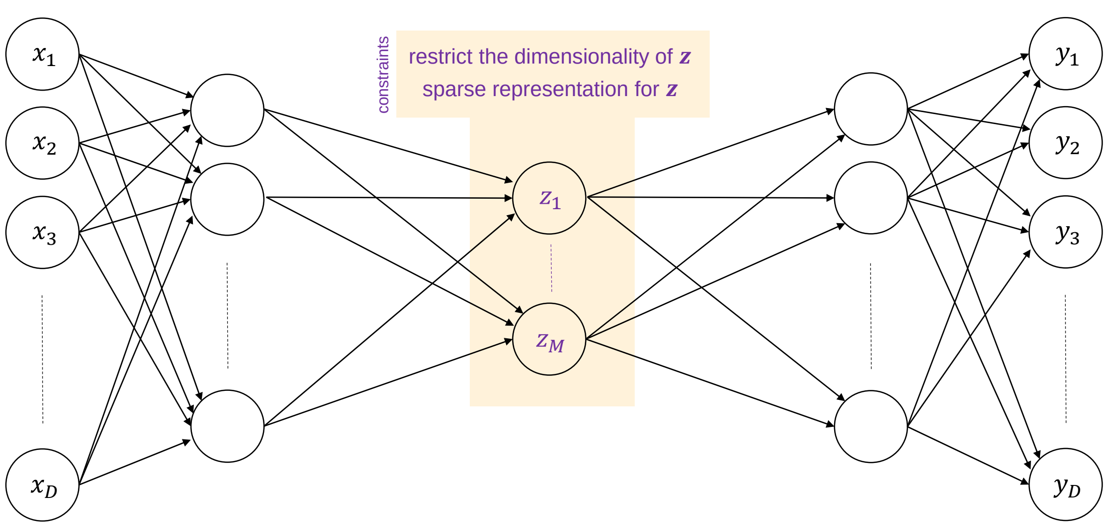
Concepts:
Same number of input and output units; internal representation $\boldsymbol{z}(\boldsymbol{x})$ of each *new* input.

Autoencoder

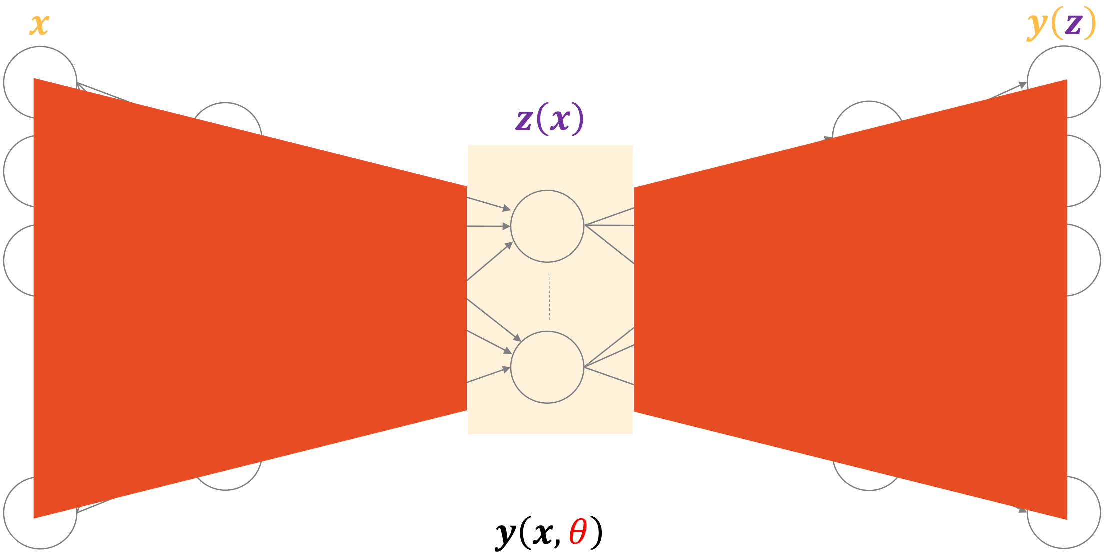


Concept:
Non-linear form of Principal Component Analysis (PCA)

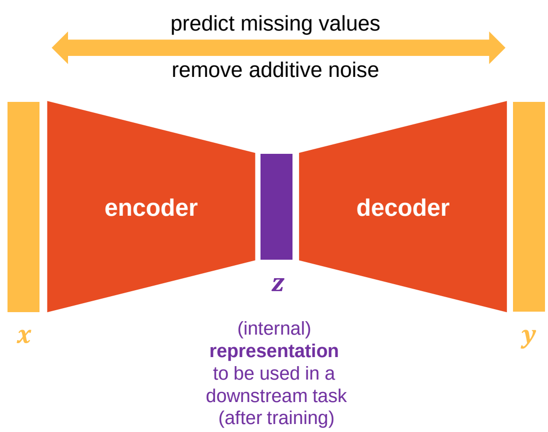
Autoencoder: network diagram



Autoencoder



Autoencoder: training



Concepts: Self-supervision; modify the training process to *undo* corruptions of the input vector $\boldsymbol{x}$; dimensionality of subspace to be defined *before* training the network.

Denoising autoencoders

$$\boldsymbol{x}_n \xrightarrow{\text{noise}} \widetilde{\boldsymbol{x}}_n$$

examples

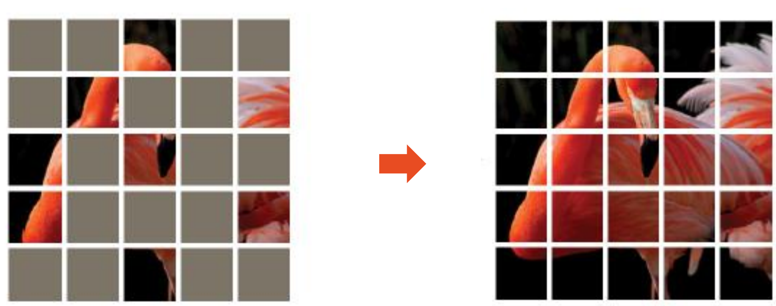
set a random amount ρ of input variables to 0 $\quad 0 \le \rho \le 1$

add *independent* zero-mean Gaussian noise to each input variable

$$E(\theta) = \sum_{n=1}^{N} \left|\left|\boldsymbol{y}(\widetilde{\boldsymbol{x}}_n, \theta) - \boldsymbol{x}_n\right|\right|^2$$

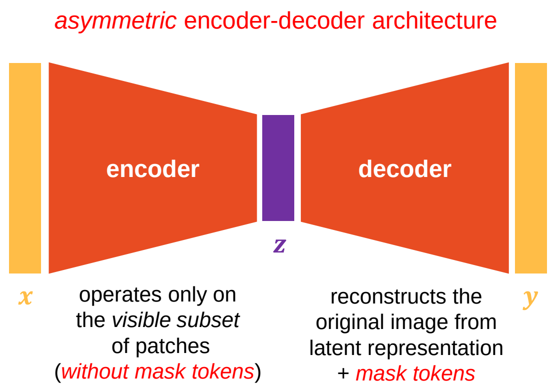
Concept:
Network encouraged to learn (aspects of) the structure of the data by learning to denoise the input data.

Masked autoencoders



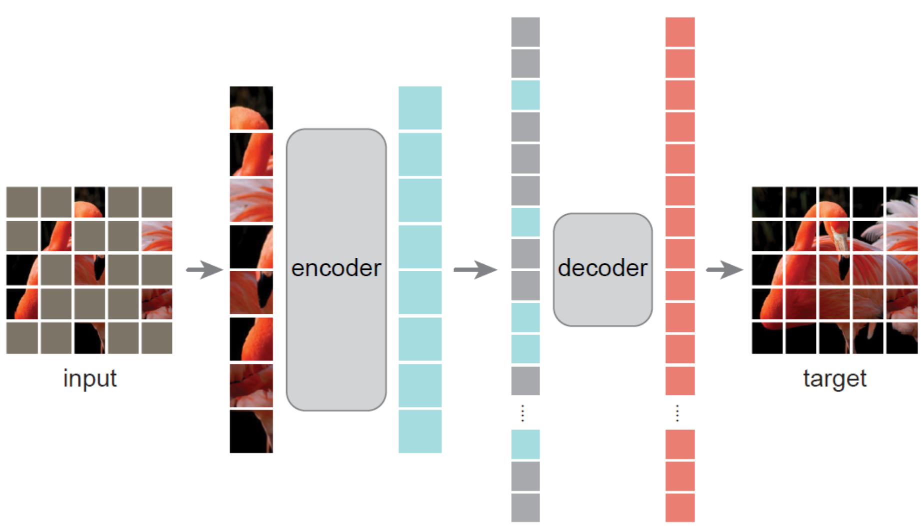
arXiv:2111.06377

Masked autoencoders



Concepts: Pass only randomly selected input patches (as low as 25% of an image); each mask token is augmented with positional encoding.

Masked autoencoders

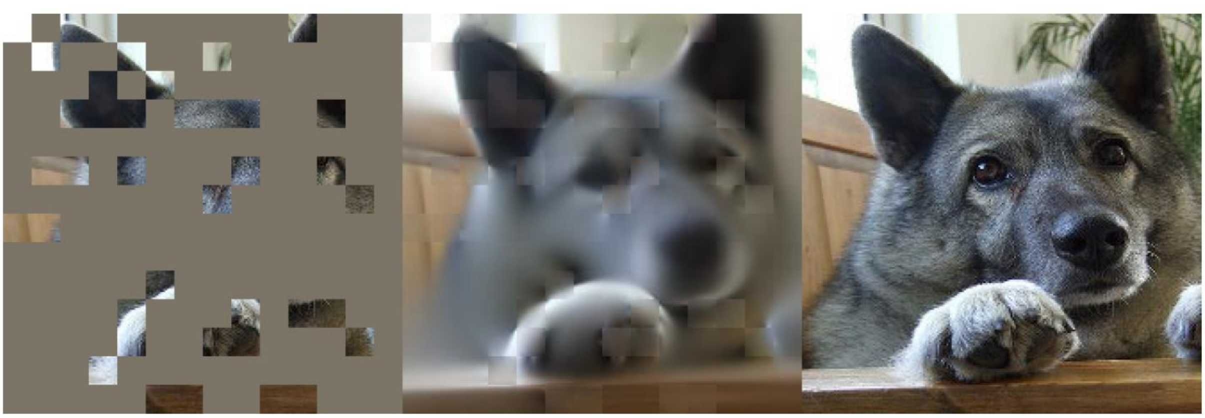


arxiv:2111.06377

Masked autoencoder: example

arXiv:2111.06377

Masked autoencoder: example



arXiv:2111.06377

Sparse autoencoders

regularizer to encourage sparse representation

$$\tilde{E}(\theta) = E(\theta) + \lambda \sum_{k=1}^{K} |z_k|$$

un-regularized error

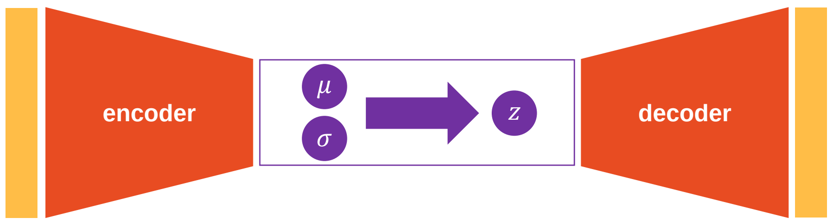
regularizer applied to the unit activations of a hidden layer

Concept:
To constrain the internal representation with a regularizer.

Variational autoencoders

VAE

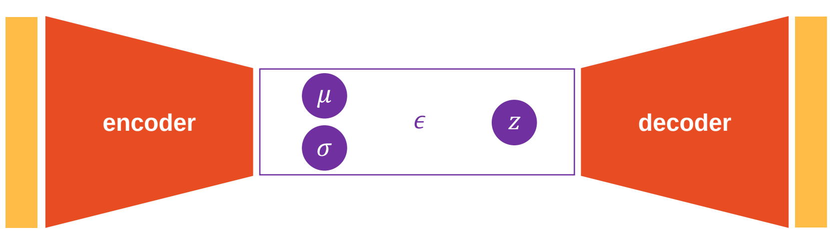
continuous, structured latent space



mapping input data into the *parameters* of a
probability distribution (e.g. *mean* & *variance* of a Gaussian)

arXiv:1312.6114

VAE: flow of gradients during training



ϵ Gaussian **random variable** with *zero mean* and *unit variance*

$z = \sigma\epsilon + \mu$ reparametrization trick (**replaces** direct sample of z)

z Gaussian distribution with *mean* μ and *variance* σ^2

arXiv:1312.6114

Vector Quantised-Variational Autoencoder

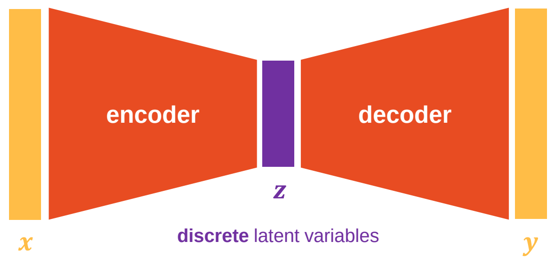
Neural discrete representation learning

discrete latent representation

prior is learnt
(rather than static)

Concepts: Parameterization of the posterior distribution of (discrete) latent variables given an observation; vector quantisation is used to learn a discrete latent.

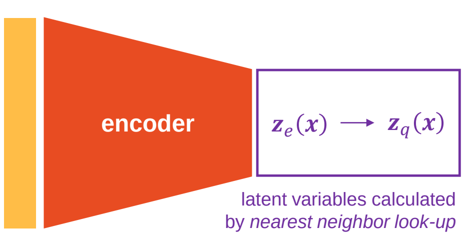
VQ-VAE



Concepts: Categorical posterior and prior distributions;
samples (drawn from these distributions) index an embedding.

Discrete latent variables

$\boldsymbol{e} \in \mathbb{R}^{K \times D}$ latent embedding space
K size of the discrete latent space
D dimensionality of each latent embedding vector



posterior categorical distribution (one-hot)

output of the encoder network

$$q(z = k \mid \boldsymbol{x}) = \begin{cases} 1 & \text{for } k = \text{argmin}_j \|\boldsymbol{z}_e(\boldsymbol{x}) - e_j\|_2 \\ 0 & \text{otherwise} \end{cases}$$

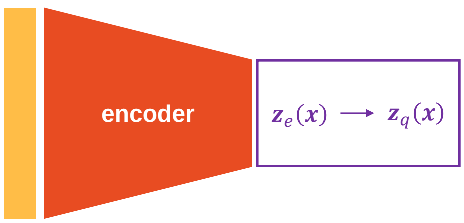
embedding vector
$\boldsymbol{e}_i \in \mathbb{R}^D \; i = 1, 2, \dots, K$

$$\boldsymbol{z}_q(\boldsymbol{x}) = e_k \qquad \text{where } k = \text{argmin}_j \|\boldsymbol{z}_e(\boldsymbol{x}) - e_j\|_2$$

arXiv:1711.00937

Learning

$\boldsymbol{e} \in \mathbb{R}^{K \times D}$ latent embedding space
K size of the discrete latent space
D dimensionality of each latent embedding vector



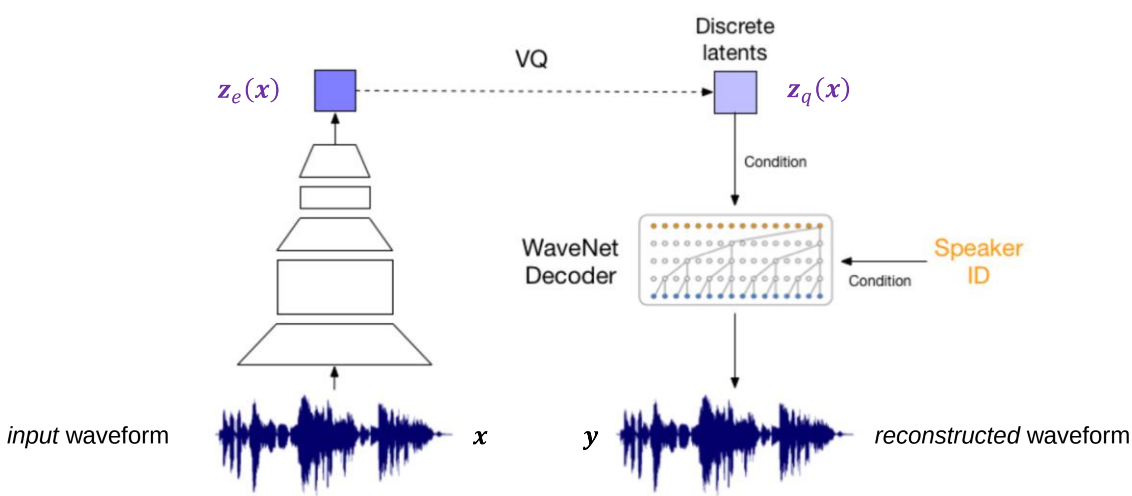
$\boldsymbol{z}_q(\boldsymbol{x}) = \boldsymbol{e}_k$ where $k = \mathrm{argmin}_j \|\boldsymbol{z}_e(\boldsymbol{x}) - \boldsymbol{e}_j\|_2$

no gradient defined for this mapping!

copy gradients from decoder input $\boldsymbol{z}_q(\boldsymbol{x})$
to encoder output $\boldsymbol{z}_e(\boldsymbol{x})$

arXiv:1711.00937

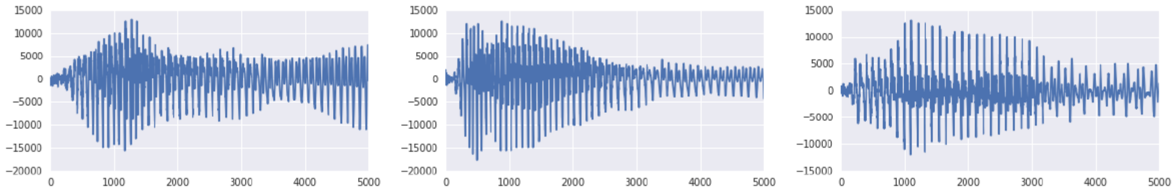
VQ-VAE: example



arXiv:1711.00937

VQ-VAE: example

original waveform | reconstructed with *same speaker-id* | reconstructed with *different speaker-id*

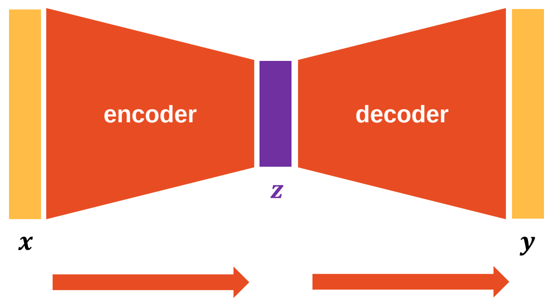


the **contents** of the three waveforms are the **same**

arXiv:1711.00937

Diffusion probabilistic models

Denoising diffusion probabilistic models

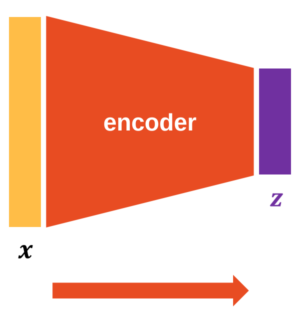


multi-step *noise process* that transforms input into a sample from a Gaussian distribution

generate new data from samples of a Gaussian distribution

Concepts: Hierarchical variational autoencoder; encoder distribution is fixed (*defined by the noise process*); only the generative distribution of the decoder is learned.

Diffusion process



Markov chain that
gradually *adds noise* to the data
until the signal/information is destroyed

multi-step *noise process* that transforms input into a sample from a Gaussian distribution

Diffusion (forward) process

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

posterior

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

fixed Markov chain that gradually adds Gaussian noise

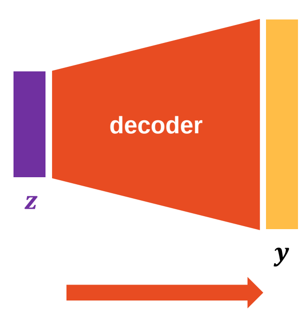
$\beta_1, \ldots, \beta_T$ variance schedule (the β_i are fixed)

arXiv:2006.11239

Reverse process

if diffusion consists of small amounts of Gaussian noise, then it suffices to set the *sampling chain transitions* to conditional Gaussians too

simple neural network parameterization!



generate new data from samples of a Gaussian distribution

arXiv:2006.11239

Reverse process

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$$ **reverse process** (joint distribution)

$$p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$$ start of the Markov chain

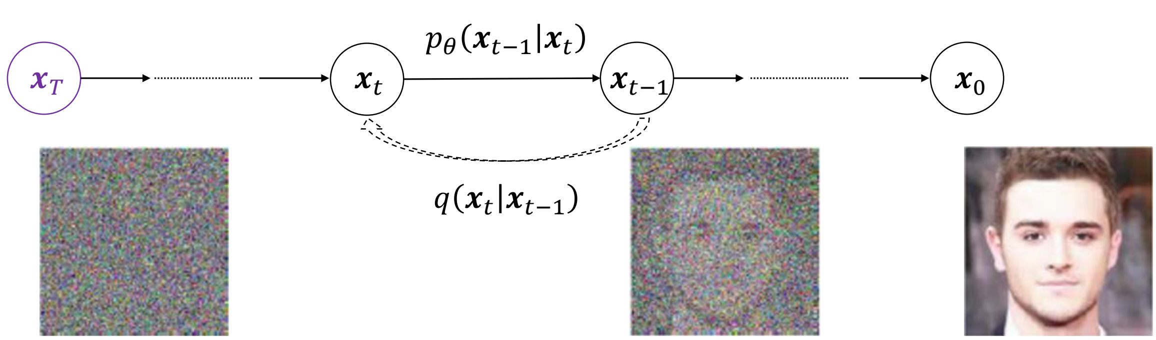
$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t))$$ learned Gaussian transitions

$\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T$ latents of the same dimension as the data $\boldsymbol{x}_0$

arXiv:2006.11239

Diffusion probabilistic models

Reverse process:
parameterized Markov chain
trained using variational inference



arXiv:2006.11239

Reverse process: examples

arXiv:2006.11239

Instruction-following diffusion models

Text-to-image diffusion

"Teddy bears swimming at the Olympics 400m Butterfly event"

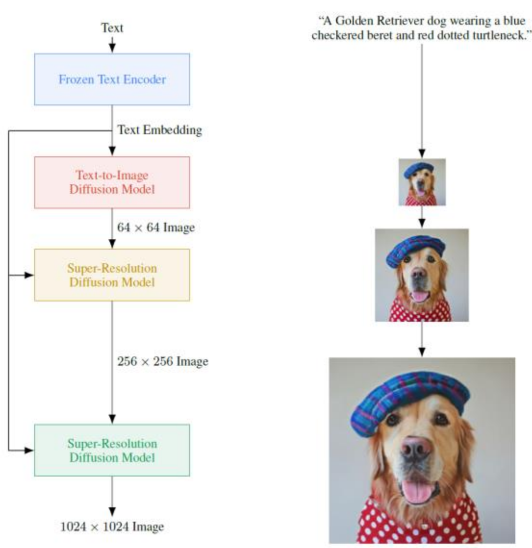
encoding text for image synthesis

image-text alignment

capturing complexity & compositionality of arbitrary natural language text inputs

arXiv:2205.11487

Text-to-image diffusion

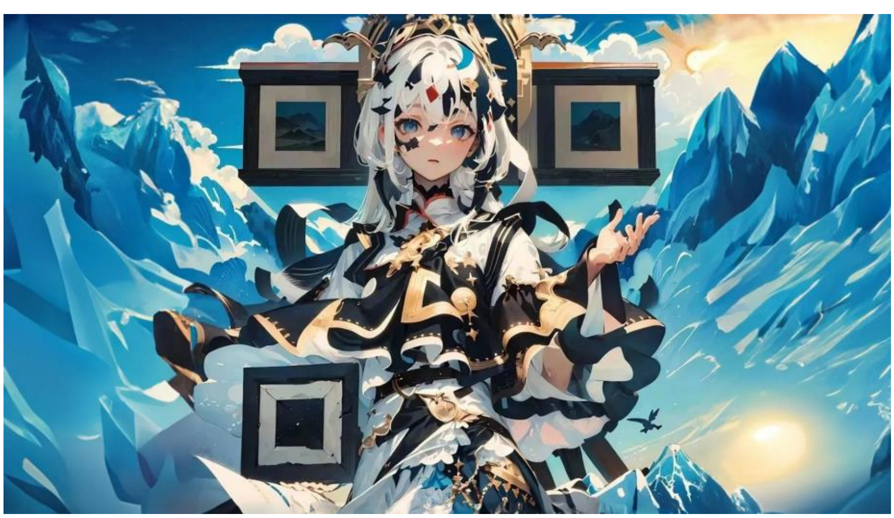


arXiv:2205.11487

Anime quick response codes

https://arstechnica.com/information-technology/2023/06/redditor-creates-working-anime-qr-codes-using-stable-diffusion

Anime quick response codes



https://arstechnica.com/information-technology/2023/06/redditor-creates-working-anime-qr-codes-using-stable-diffusion

Editing images from human instructions

conditional diffusion model

Large Language Model

GPT-3

Text-to-image Model

Stable Diffusion

combined to create synthetic paired training data for a task spanning both language & images

arXiv:2211.09800

Synthetic multi-modal training data

arXiv:2211.09800

Instruction-following diffusion model

"turn her into a snake lady"

arXiv:2211.09800

Cross-attention control for editing

cross-attention maps **bind pixels** & **tokens** extracted from the prompt text

inject the cross-attention maps during the *diffusion process*
controlling which **pixels** attend to which **tokens** of the prompt text
during which *diffusion steps*

change a single token's value in the prompt
(e.g. *cat* to *dog*)
fix the cross-attention maps
to preserve the scene composition

globally edit an image
(e.g. change the style)
add **new words** to the prompt
freeze the attention on previous tokens
allow **new attention** to flow to the **new tokens**

arXiv:2208.01626

Cross-attention control for editing: example

"Photo of a cat riding on a bicycle."

source image | *cat* > dog | *cat* > chicken | *cat* > squirrel | *cat* > elephant

no need for model training, fine-tuning, extra data, or optimization

arXiv:2208.01626

Practice exercises

Today’s practice exercises

- Autoencoders for anomaly detection

- Variational autoencoders for image generation

What did we learn today?

- Deterministic autoencoders
- Variational autoencoders
- Vector quantised-variational autoencoder
- Diffusion probabilistic models
- Instruction-following diffusion models
- Exercises

EE-559

Deep Learning

andrea.cavallaro@epfl.ch