WORKSHOP, AMLD 2020

Generative Modelling for Computer Vision

Sandro Barnabishvili
AI/ML Engineer @ MaxinAI

Anzor Gozalishvili
AI/ML Engineer @ MaxinAI

MaxinAI

About us

Sandro Barnabishvili
sandro.barnabishvili@maxinai.com
https://www.linkedin.com/in/sandrobarna/

- AI/ML Engineer @ MaxinAI
- Master in Computer Science, EPFL
- Experience in Deep Learning, CV, NLP and Big Data

Anzor Gozalishvili
anzor.gozalishvili@maxinai.com
https://www.linkedin.com/in/anzor-gozalishvili/

- AI/ML Engineer @ MaxinAI
- Bachelor in Computer Science TSU, UPV
- Experience in NLP, Tabular Data Analysis, Deep Learning, CV

Agenda (~3 hours, 30 min break)

- Intro (theoretical background)
  - Auto-Encoders
  - Variational Auto-Encoders
- Workshop *(Google Colab and PyTorch, laptops optional)*
  - Notebook 1: Comparison of AEs and VAEs on MNIST data, illustrative examples
  - Notebook 2: Learning faces with Convolutional VAEs, generating new faces

Discriminate VS Generative ML

Discriminative ML – Learn to predict something from the data

- Predict which animal is shown on the picture
- Detect a person’s face on the image
- Estimate temperature and humidity given satellite data over past week.
- Predict the helpfulness of Amazon review

…

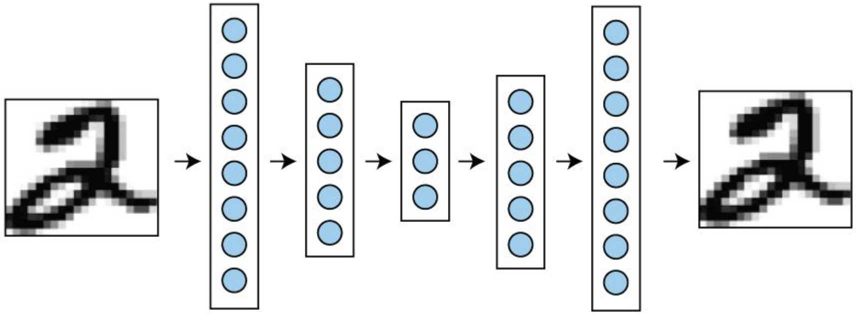
Discriminate VS Generative ML

Generative ML – Learn to generate new data from given data

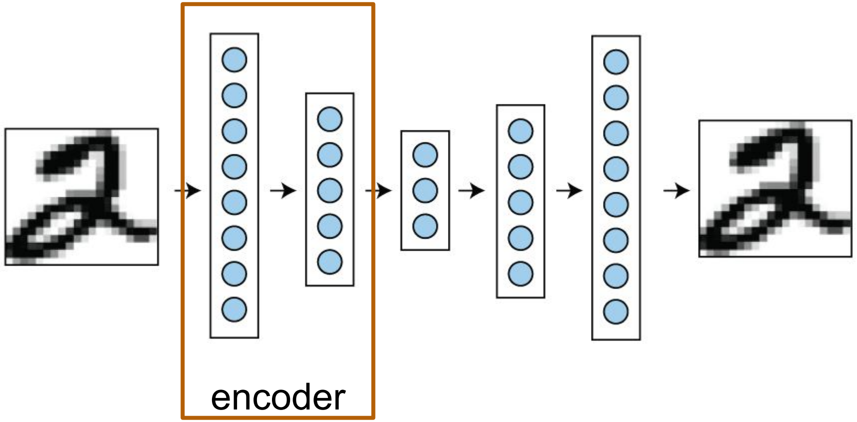
- Given a written text, generate a voice.
- Given images of bedrooms, learn to generate new (unseen) bedrooms
- Given a low quality image, output HD version of it (super resolution)
- Restore a missing parts of a corrupted Image/Audio/Text

…

Auto-Encoders

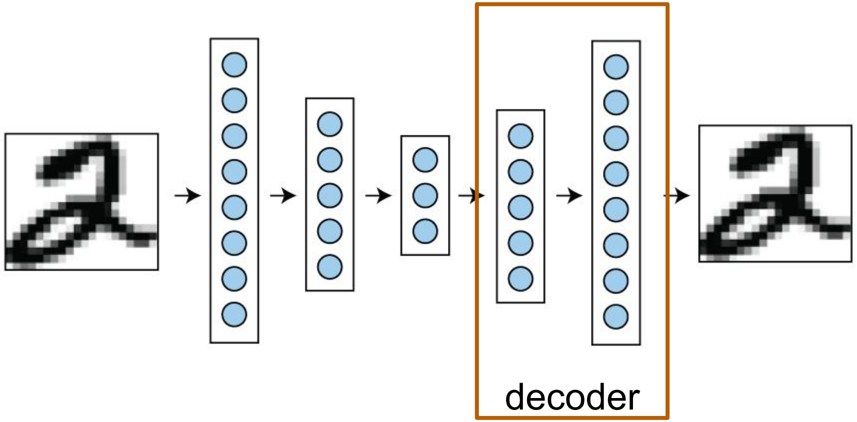


Auto-Encoders



Encoder Network: Transform input into **latent representation**

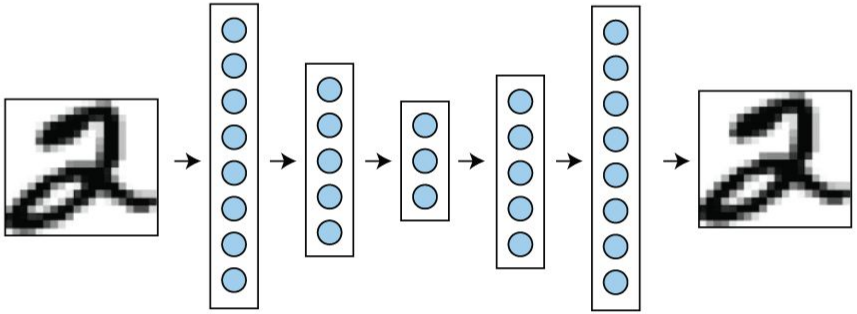
Auto-Encoders



Encoder Network: Transform input into **latent representation**

Decoder Network: Reconstruct original input from **latent representation**

Auto-Encoders

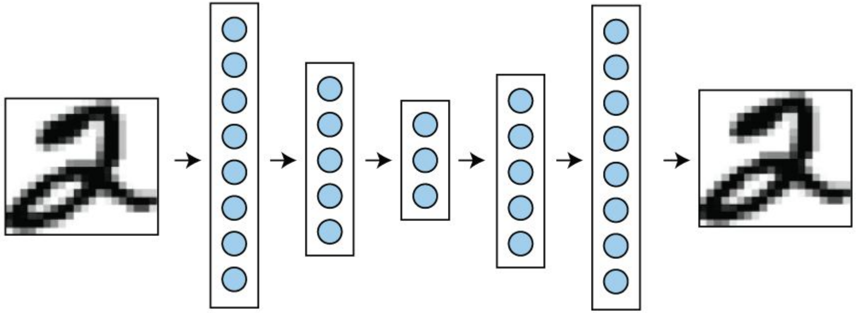


Encoder Network: Transform input into **latent representation**

Decoder Network: Reconstruct original input from **latent representation**

Objective: Minimize Mean Squared Error between input and reconstruction pixels

Auto-Encoders



Depending on the task, encoder and decoder can be MLP CNN, RNN…

We are going to use MLP and CNN in this workshop.

Variational Auto-Encoders

Motivation: We want to learn data distribution P(X) and effectively sample from it, i.e. generate new data points.

Variational Auto-Encoders

Motivation: We want to learn data distribution P(X) and effectively sample from it, i.e. generate new data points.

How do we sample?

Learn a function (neural network) that transforms some known distribution (e.g. Gaussian) into our desired data distribution.

(lot of models use this technique: VAEs, GANs, Normalizing Flows)

Variational Auto-Encoders

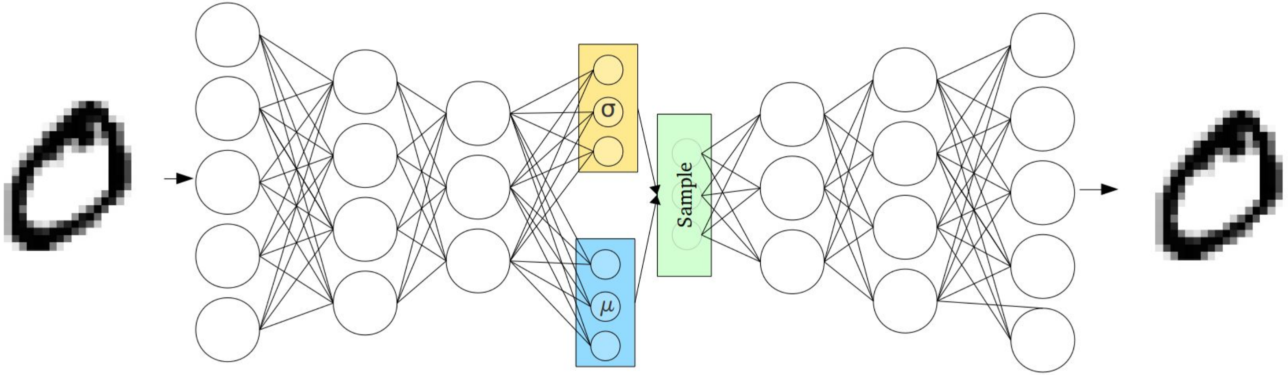


Image taken from towardsdatascience.com

<u>Encoder Network:</u> Maps input to some **mean** and **variance** of the Gaussian.

Variational Auto-Encoders

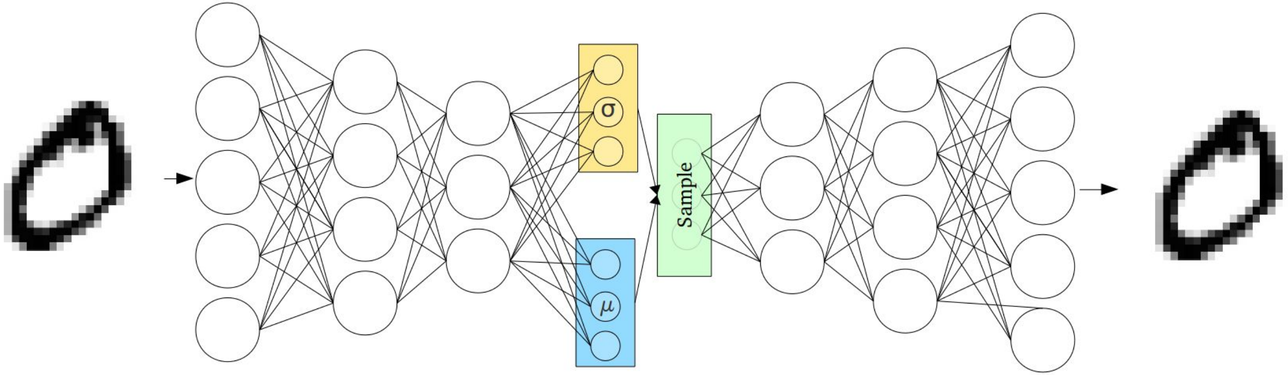


Image taken from towardsdatascience.com

Encoder Network: Maps input to some **mean** and **variance** of the Gaussian.
Decoder Network: Takes a **sample** from this Gaussian and transforms it to output image.

Variational Auto-Encoders

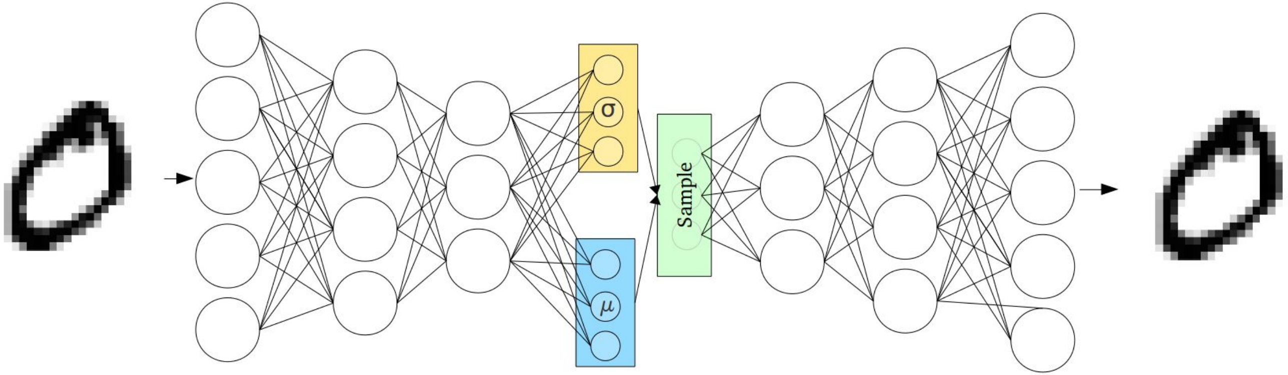


Image taken from towardsdatascience.com

<u>Question:</u> How do we train it end-to-end? How can we propagate gradient through **sampling procedure** that is part of the network?

Variational Auto-Encoders

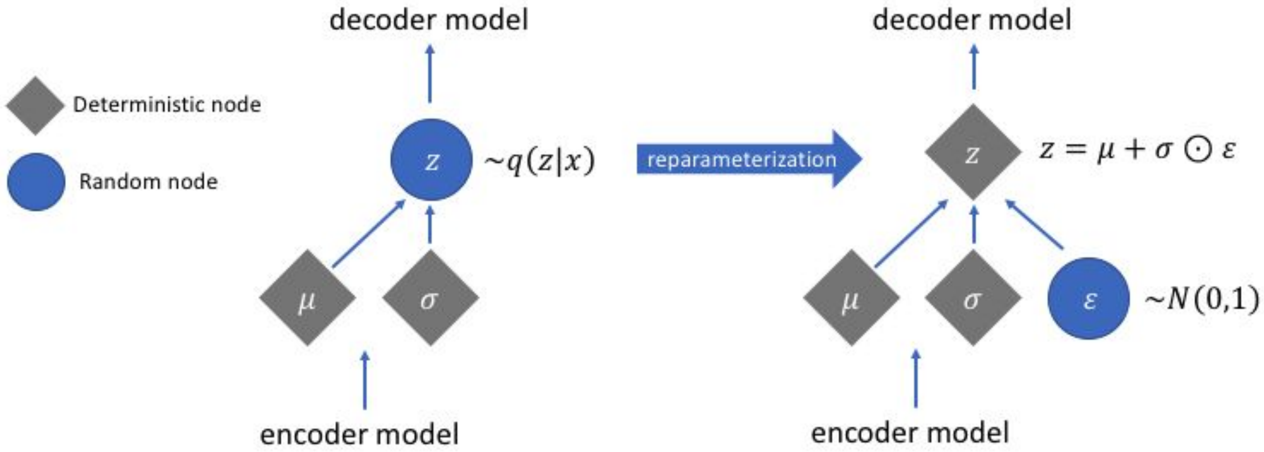


Image taken from https://bit.ly/2NJ6J0V

<u>Reparametrization Trick:</u> Uses the fact that standard gaussian can be scaled and shifted to get any other gaussian.

Variational Auto-Encoders

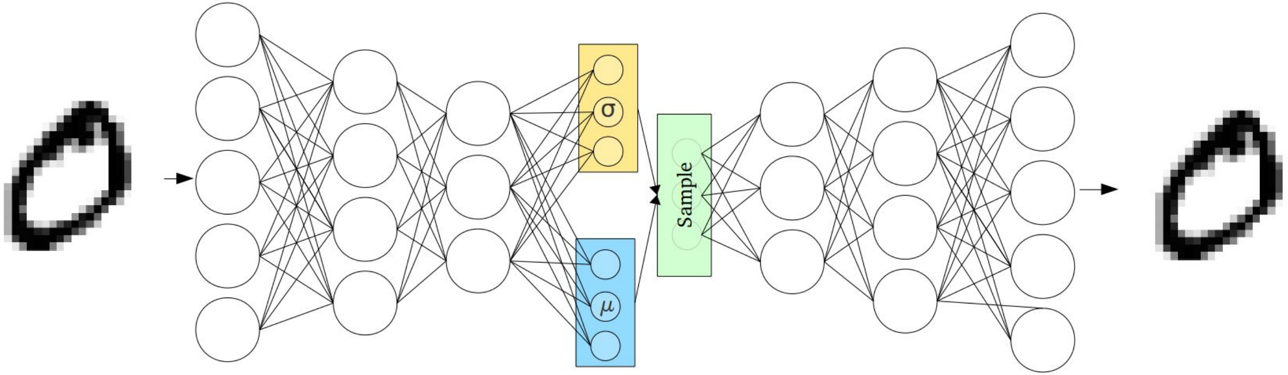


Image taken from towardsdatascience.com

<u>Loss function:</u> Reconstruction Loss + KL Divergence between standard Gaussian and the Gaussian with learned mean and variance.

Detailed math is beyond this presentation

Preparing Workspace

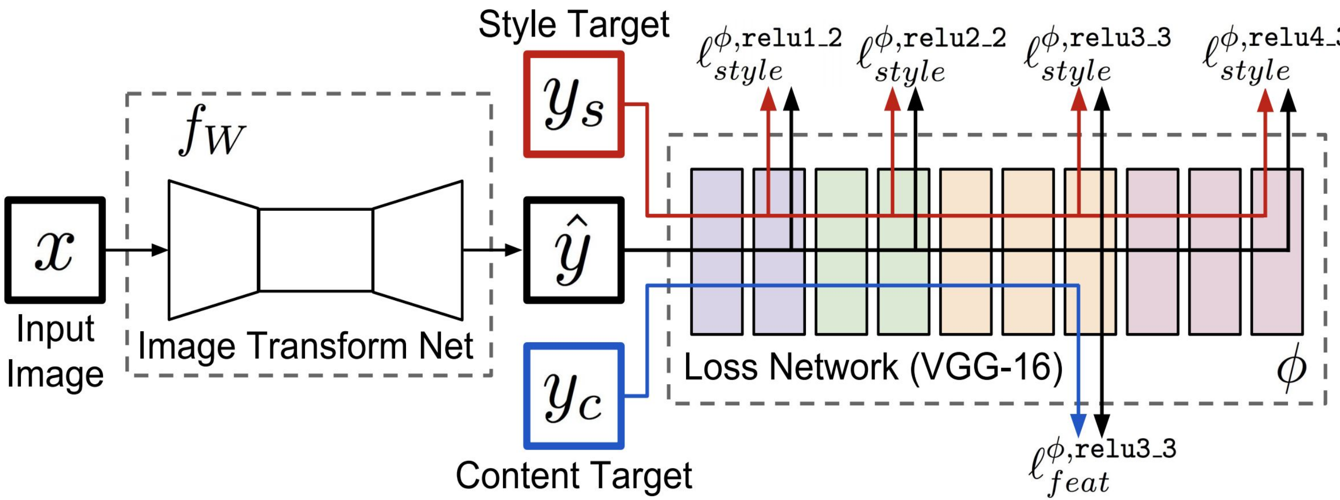
1. Open Google Colab
2. File -> Open Notebook -> Github
3. Type https://github.com/MaxinAI/amld2020-workshop
4. Change Runtime to GPU

References & Reading Material

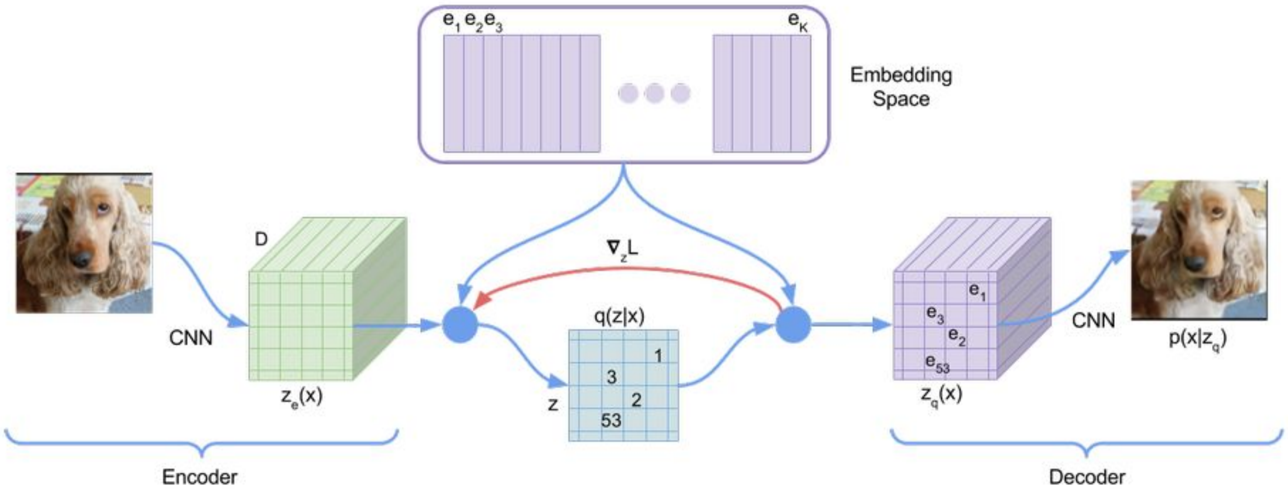
- Original VAE paper https://arxiv.org/abs/1312.6114
- CelebA dataset http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
- https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html#sparse-autoencoder
- Weight initialization https://arxiv.org/abs/1502.01852
- More upsampling techniques https://arxiv.org/abs/1609.05158
- Vector-Quantized Auto-Encoders https://arxiv.org/abs/1711.00937
- Perceptual Loss https://arxiv.org/abs/1603.08155

Backup slides

Perceptual Loss

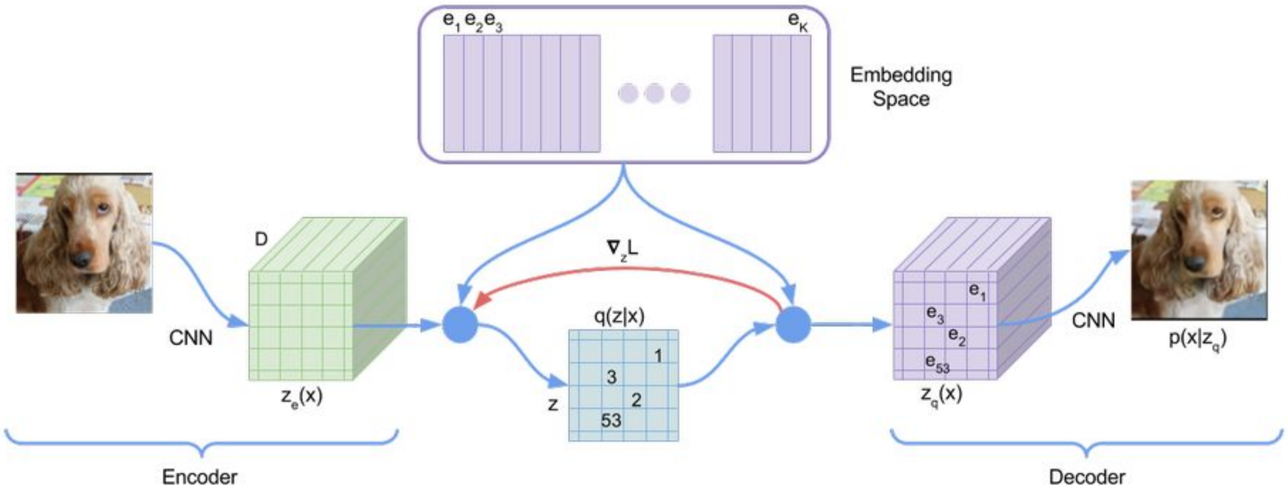


Vector Quantized Variational Auto-Encoders



Encoder Network: Maps input to some **latent space** vectors

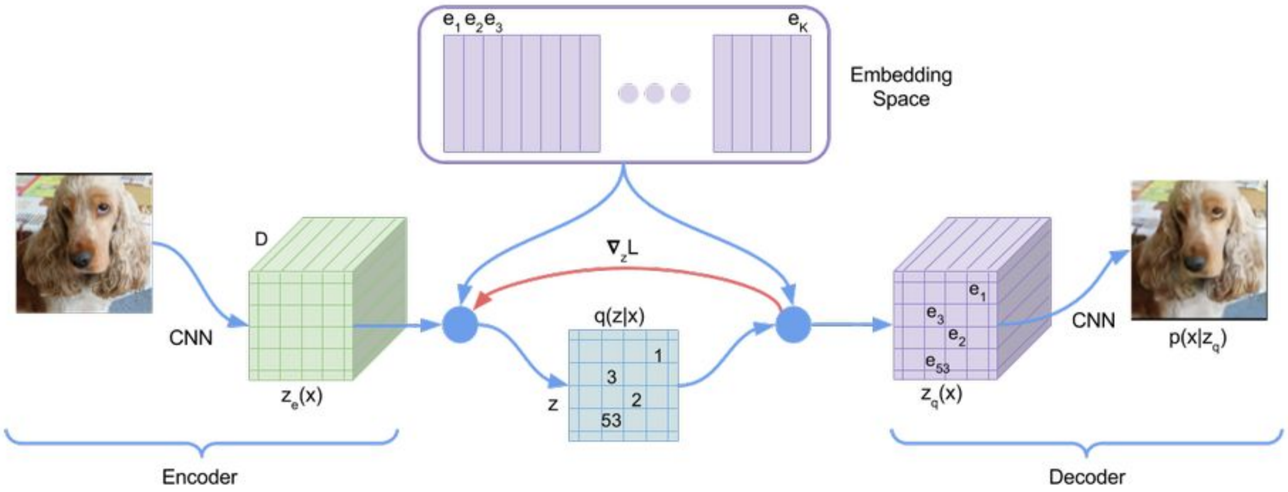
Vector Quantized Variational Auto-Encoders



<u>Encoder Network:</u> Maps input to some **latent space** vectors

<u>Quantizer:</u> Maps **latent space** vectors to closest **Codebook** vectors

Vector Quantized Variational Auto-Encoders

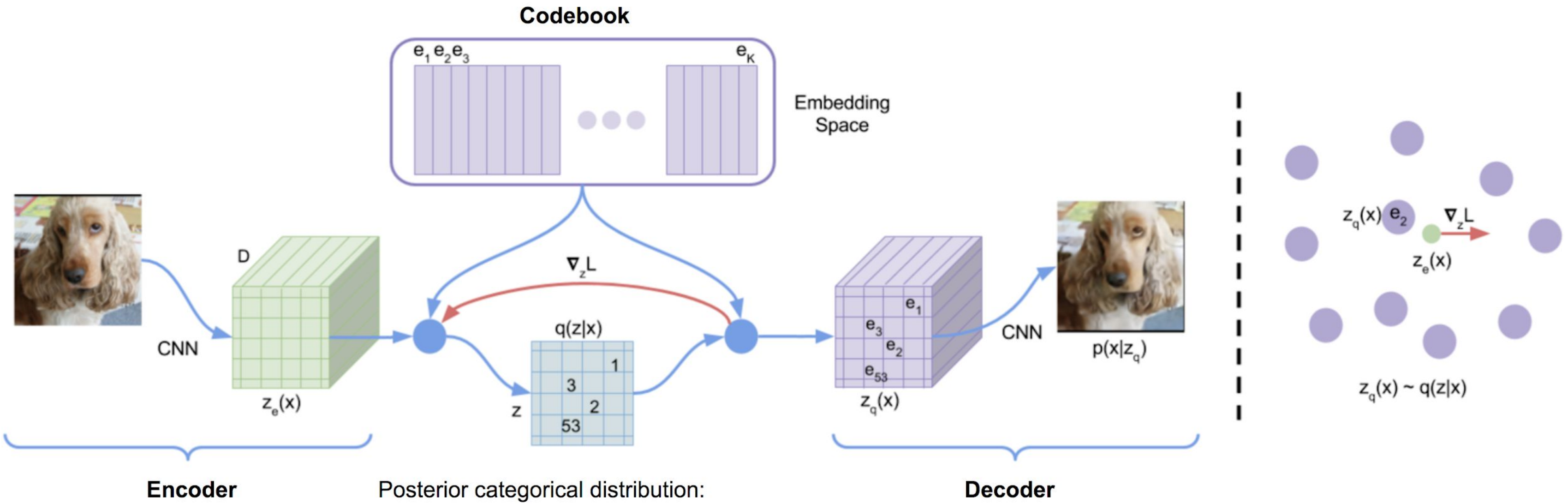


<u>Encoder Network:</u> Maps input to some **latent space** vectors

<u>Quantizer:</u> Maps **latent space** vectors to closest **Codebook** vectors

<u>Decoder Network:</u> Takes **quantized latent vectors** and transforms it to output image.

Vector Quantized Variational Auto-Encoders



Posterior categorical distribution:

$$q(\mathbf{z} = \mathbf{e}_k|\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

Because **argmin()** is non-differentiable on a discrete space, the gradients $\nabla_z L$ from decoder input $\mathbf{z}_q$ is copied to the encoder output $\mathbf{z}_e$

Vector Quantized Variational Auto-Encoders

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{reconstruction loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ loss}} + \underbrace{\beta\|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{commitment loss}}$$

<u>Reconstruction Loss:</u> To keep **input** and **output** close

Vector Quantized Variational Auto-Encoders

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{reconstruction loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ loss}} + \underbrace{\beta\|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{commitment loss}}$$

Reconstruction Loss: To keep **input** and **output** close

VQ Loss: L2 error between Embeddings in **Codebook** and Encoder outputs

Vector Quantized Variational Auto-Encoders

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{reconstruction loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ loss}} + \underbrace{\beta\|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{commitment loss}}$$

<u>Reconstruction Loss:</u> To keep **input** and **output** close

<u>VQ Loss:</u> L2 error between Embeddings in **Codebook** and Encoder outputs

<u>Commitment Loss:</u> To encourage Encoder to stay close to Embeddings