Neural Discrete Representation Learning

Radek Bartyzal

GLAMI AI

2. 2. 2021

Motivation

Paper is by Google DeepMind from 2018.

Goal: Learn useful representations without supervision.

Contribution: Generative model that learns **discrete** representations.

Why discrete? Discrete representations are a natural fit for complex reasoning, planning and predictive learning.

Variational AutoEncoders = VAE

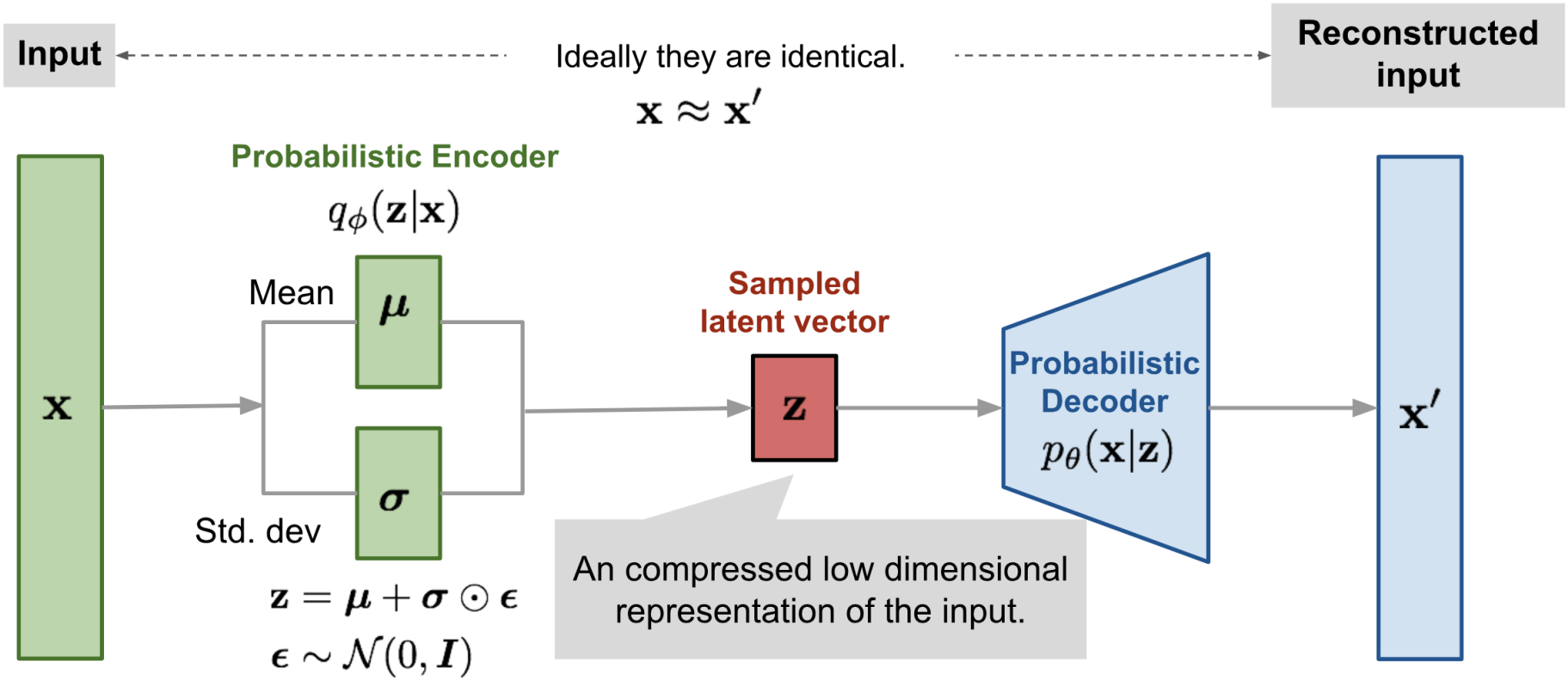

Figure: VAE model with the multivariate Gaussian assumption. [2]

Vector Quantised VAE = VQ-VAE

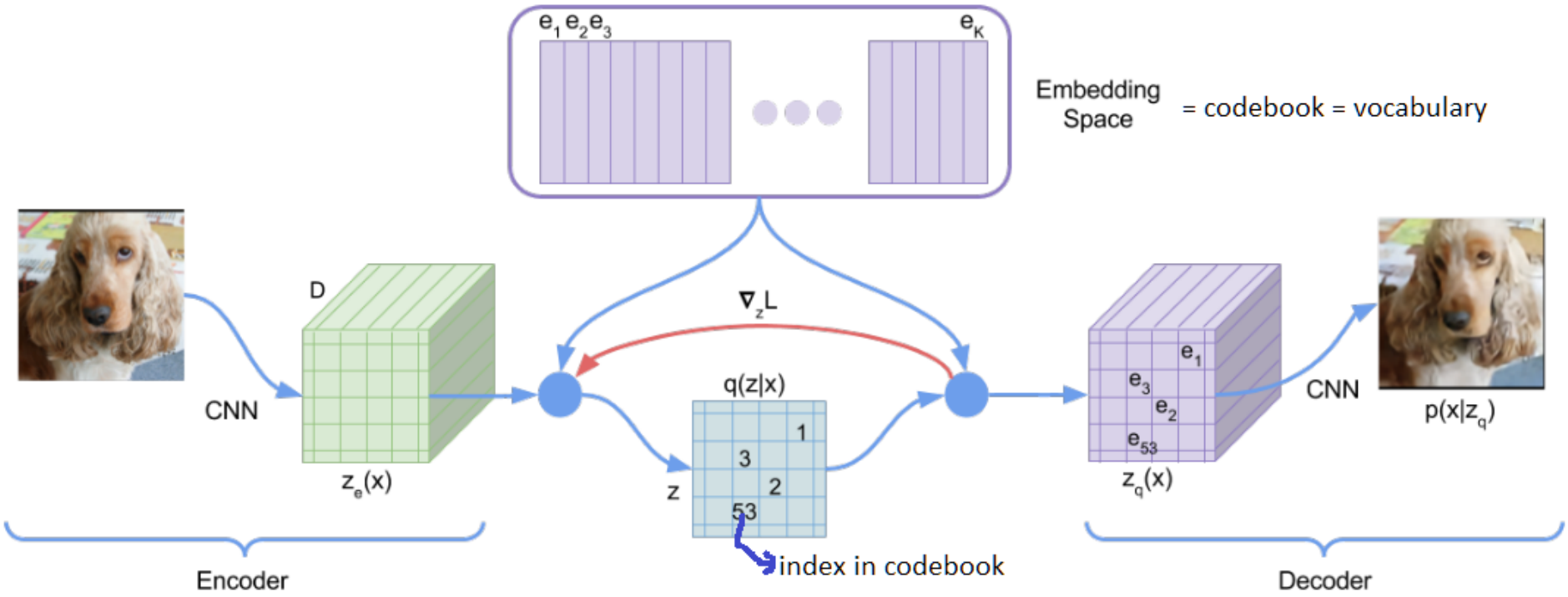

Figure: The output of the encoder $z(x)$ is mapped to the nearest point e_i in codebook. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

VQ-VAE Training

Discrete representation = 1024 of 8096 codebook (vocabulary) embeddings for each image.

Forward pass:

- Split image into a grid of 32x32 = 1024 squares
- encoder predicts real-valued vector $z_e(x) = E(x)$ for each square
- $E(x)$ is replaced by an **index** of nearest neighbor from codebook
- index = integer = discrete
- this (32x32x1) map of indices is passed to decoder
- decoder uses the same codebook to translate indices to vectors and reconstructs the image

VQ-VAE Training

$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = ||\mathbf{x} - D(\mathbf{e})||_2^2 + ||sg[E(\mathbf{x})] - \mathbf{e}||_2^2 + \beta||sg[\mathbf{e}] - E(\mathbf{x})||_2^2$$

Figure: Loss = reconstruction + codebook + commitment losses.

Backwards pass:

- reconstruction loss = classic L2 loss
- codebook loss = pushes codebook vectors closer to encoder output
- commitment loss = pushes encoder output closer to the nearest codebook vector
- sg = stop gradient = gradient does not flow into that element
- nearest neighbor op is not differentiable = copy gradients from decoder input to encoder output = red arrow

VQ-VAE Reconstruction examples

Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with K=512.

- only slightly blurry images = very good reconstruction

VQ-VAE Generation of images

- after training the VQ-VAE
- train a per-pixel auto-regressive generative model on the discrete 32x32x1 space of the trained VQ-VAE
- then sample from this auto-regressive model conditioned on a class label
- sample $=$ 32x32x1
- input the sample into decoder $\Longrightarrow$ generated image

Why train this extra model?
Latent variables sampled from the learned prior at test time are close to what the decoder network has observed during training which results in more coherent outputs.

VQ-VAE Generation examples

Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images. From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup.

Auto-regressive models:

- PixelCNN = used in this paper
- PixelSNAIL [3] = used in VQ-VAE-2 = hierarchical version

VQ-VAE-2 = Hierarchical version

- just like VQ-VAE but done in 3 steps
- each step increases the resolution and is conditioned on te previous step
- standard hierarchical stuff to generate higher resolution images

VQ-VAE-2 Training

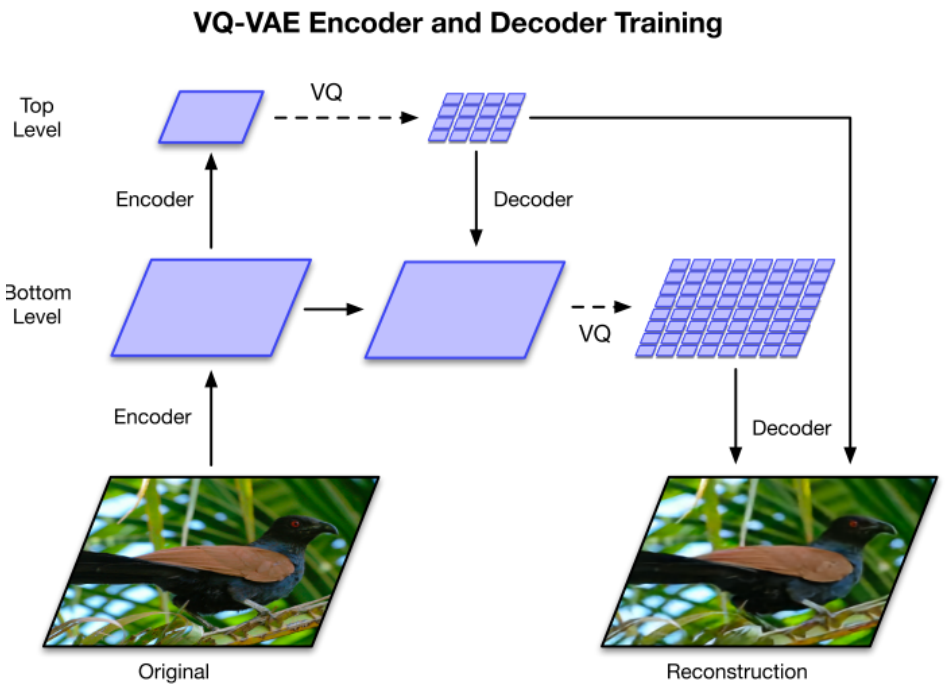

Figure: The encoders and decoders consist of deep neural networks. The input to the model is a 256 x 256 image that is compressed to quantized latent maps of size 64 x 64 and 32 x 32 for the bottom and top levels, respectively. The decoder reconstructs the image from the two latent maps.

VQ-VAE-2 Reconstruction examples

h_{top} $h_{\text{top}}, h_{\text{middle}}$ $h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$ Original

Figure 3: Reconstructions from a hierarchical VQ-VAE with three latent maps (top, middle, bottom). The rightmost image is the original. Each latent map adds extra detail to the reconstruction. These latent maps are approximately 3072x, 768x, 192x times smaller than the original image (respectively).

VQ-VAE-2 Generation

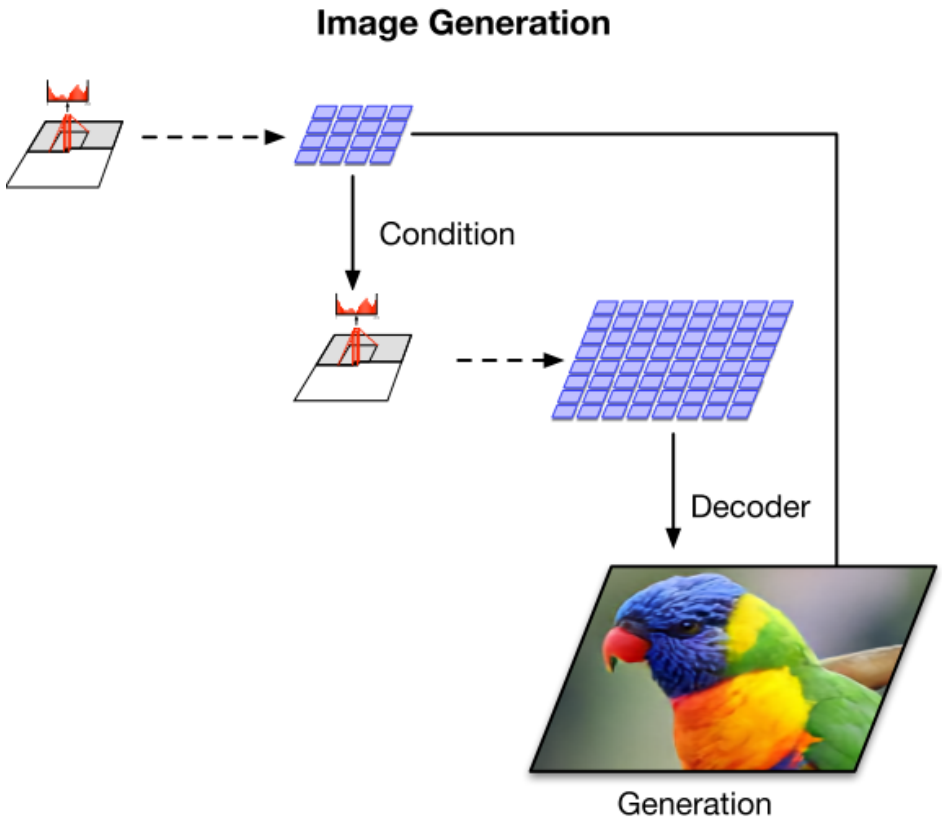

Figure: Multi-stage image generation. The top-level PixelCNN prior is conditioned on the class label, the bottom level PixelCNN is conditioned on the class label as well as the first level code. Thanks to the feed-forward decoder, the mapping between latents to pixels is fast. (The example image with a parrot is generated with this model).

VQ-VAE-2 Generation examples

Figure 4: Class conditional random samples. Classes from the top row are: 108 sea anemone, 109 brain coral, 114 slug, 11 goldfinch, 130 flamingo, 141 redshank, 154 Pekinese, 157 papillon, 97 drake, and 28 spotted salamander.

VQ-VAE-2 Generation examples

Sources

1.Oord, Aaron van den, Oriol Vinyals, and Koray Kavukcuoglu. "Neural discrete representation learning." arXiv preprint arXiv:1711.00937 (2017). https://arxiv.org/abs/1711.00937

2. VAE overview blog post. https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

3. Chen, Xi, et al. "Pixelsnail: An improved autoregressive generative model." International Conference on Machine Learning. PMLR, 2018. https://arxiv.org/abs/1712.09763

4. Razavi, Ali, Aaron van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2." arXiv preprint arXiv:1906.00446 (2019). https://arxiv.org/abs/1906.00446

5. VQ-VAE implementation https://github.com/nadavbh12/VQ-VAE/tree/master/vq_vae