Vector Quantised Variational Autoencoder

Prof. Suyash Awate

Shikhar Agrawal
Mayank Jain

Introduction to VQ-VAE

- Our model, the Vector Quantised Variational AutoEncoder (VQ-VAE), differs from VAEs in two key ways: the encoder network outputs discrete, rather than continuous codes; and the prior is learnt rather than static.
- Using the VQ method allows the model to circumvent issues of “posterior collapse”, where the latents are ignored when they are paired with a powerful autoregressive decoder typically observed in the VAE framework.
- Pairing these representations with an autoregressive prior, the model can generate high quality images.

A Note on Posterior Collapse...

When posterior is not collapsed, z_d (d-th dimension of latent variable z) is sampled from $q_\phi(z_d|x) = \mathcal{N}(\mu_d, \sigma_d^2)$, where μ_d and σ_d are stable functions of input x. In other words, encoder distills useful information from x into μ_d and σ_d.

We say a posterior is collapsing, when signal from input x to posterior parameters is either **too weak** or **too noisy**, and as a result, decoder starts ignoring z samples drawn from the posterior $q_\phi(z|x)$.

The too weak signal translates to

$$q_\phi(z|x) \simeq q_\phi(z) = \mathcal{N}(a, b)$$

which means μ and σ of posterior become almost disconnected from input x. In other words, μ and σ collapse to constant values a, and b channeling a weak (constant) signal from *different* inputs to decoder. As a result, decoder tries to reconstruct x by ignoring useless z's which are sampled from $\mathcal{N}(a, b)$.

Discrete Latent Variables

We define a latent embedding space $e \in R^{K \times D}$ where K is the size of the discrete latent space (i.e., a K-way categorical), and D is the dimensionality of each latent embedding vector e_i. Note that there are K embedding vectors $e_i \in R^D$, $i \in 1, 2, ..., K$. As shown in Figure 1, the model takes an input x, that is passed through an encoder producing output $z_e(x)$. The discrete latent variables z are then calculated by a nearest neighbour look-up using the shared embedding space e as shown in equation 1. The input to the decoder is the corresponding embedding vector e_k as given in equation 2. One can see this forward computation pipeline as a regular autoencoder with a particular non-linearity that maps the latents to 1-of-K embedding vectors. The complete set of parameters for the model are union of parameters of the encoder, decoder, and the embedding space e. For sake of simplicity we use a single random variable z to represent the discrete latent variables in this Section, however for speech, image and videos we actually extract a 1D, 2D and 3D latent feature spaces respectively.

Discrete Latent Encoding

The posterior categorical distribution $q(z|x)$ probabilities are defined as one-hot as follows:

$$q(z = k|x) = \begin{cases} 1 & \text{for } \mathrm{k} = \mathrm{argmin}_j \|z_e(x) - e_j\|_2, \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where $z_e(x)$ is the output of the encoder network. We view this model as a VAE in which we can bound $\log p(x)$ with the ELBO. Our proposal distribution $q(z = k|x)$ is deterministic, and by defining a simple uniform prior over z we obtain a KL divergence constant and equal to $\log K$.

The representation $z_e(x)$ is passed through the discretisation bottleneck followed by mapping onto the nearest element of embedding e as given in equations 1 and 2.

$$z_q(x) = e_k, \quad \text{where} \quad k = \mathrm{argmin}_j \|z_e(x) - e_j\|_2 \tag{2}$$

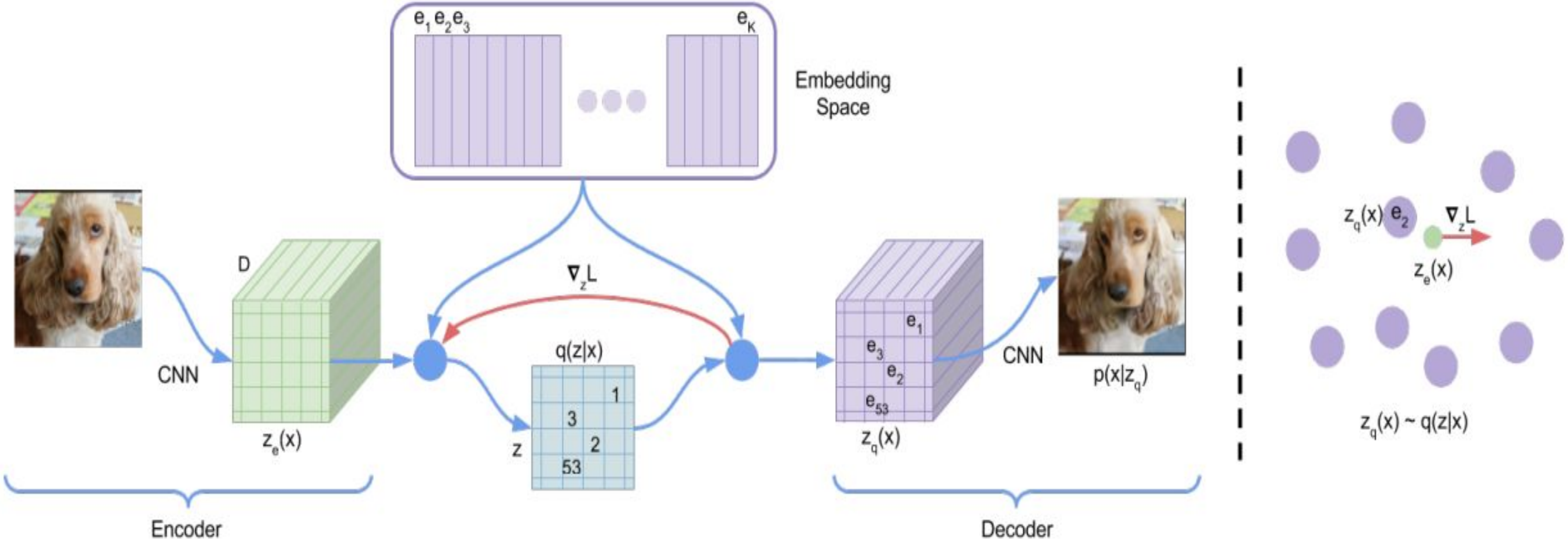

Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

BackPropagation Issues?

- Note that there is no real gradient defined for equation 2, however we approximate the gradient similar to the straight-through estimator and just copy gradients from decoder input z_q(x) to encoder output z_e(x).
- During forward computation the nearest embedding z_q(x) (equation 2) is passed to the decoder, and during the backwards pass the gradient ∇_z(L) is passed unaltered to the encoder.
- Since the output representation of the encoder and the input to the decoder share the same D dimensional space, the gradients contain useful information for how the encoder has to change its output to lower the reconstruction loss.
- As seen on Figure 1 (right), the gradient can push the encoder output to be discretized differently in the next forward pass, because the assignment in equation 1 will be different.

Loss Function

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

- It is has three components that are used to train different parts of VQ-VAE. The first term is the reconstruction loss (or the data term) which optimizes the decoder and the encoder (through the estimator explained above).
- Due to the straight-through gradient estimation of mapping from z_e(x) to z_q(x), the embeddings ei receive no gradients from the reconstruction loss log p(z|z_q(x)).
- Therefore, in order to learn the embedding space, we use one of the simplest dictionary learning algorithms, Vector Quantisation (VQ). The VQ objective uses the L2 error to move the embedding vectors e_i towards the encoder outputs z_e(x) as shown in the second term of above equation.
- Finally, since the volume of the embedding space is dimensionless, it can grow arbitrarily if the embeddings ei do not train as fast as the encoder parameters. To make sure the encoder commits to an embedding and its output does not grow, we add a commitment loss, the third term in above equation.

Loss Function

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

- where sg stands for the stop-gradient operator that is defined as identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant. The decoder optimises the first loss term only, the encoder optimises the first and the last loss terms, and the embeddings are optimised by the middle loss term.
- Since we assume a uniform prior for z, the KL term that usually appears in the ELBO is constant w.r.t. the encoder parameters and can thus be ignored for training.

Log Likelihood

The log-likelihood of the complete model $\log p(x)$ can be evaluated as follows:

$$\log p(x) = \log \sum_k p(x|z_k)p(z_k),$$

Because the decoder $p(x|z)$ is trained with $z = z_q(x)$ from MAP-inference, the decoder should not allocate any probability mass to $p(x|z)$ for $z \neq z_q(x)$ once it has fully converged. Thus, we can write

$$\log p(x) \approx \log p(x|z_q(x))p(z_q(x)).$$

What about the prior?

- The prior distribution over the discrete latents p(z) is a categorical distribution, and can be made autoregressive by depending on other z in the feature map. Whilst training the VQ-VAE, the prior is kept constant and uniform. After training, we fit an autoregressive distribution over z, p(z), so that we can generate x via ancestral sampling. We use a PixelCNN over the discrete latents for images. Training the prior and the VQ-VAE jointly, which could strengthen our results, was left as future research.