All About Autoencoders: HW Commentary

Michael Huang, Chancharik Mitra, Joshua You, Jason Zhang

CS 182 - Spring 2023

For this project, our main goal was to bring a student through various flavors of autoencoders, starting from a basic autoencoder architecture and ending with more recent developments (in particular, the VQ-VAE).

1 Motivation

One of the things we tried to do was to motivate each step in the progression from autoencoders to VAEs to VQ-VAEs. In particular, we wanted to show the students what issues existed with each implementation through various visualizations, and why a change to the architecture could improve on these issues. We address these issues both in the analytical questions and in the notebooks, with questions in the written portion asking the student to analyze the results they saw in the notebook, similar to what is done on other CS182 homeworks.

2 Latent Sampling Methods

One of the most important parts of the VQ-VAE architecture were the sampling methods they used to create a discrete distribution. We wanted to motivate this in the written homework by asking students questions about how we can keep the encoder representations close to the discrete sampled representations. We first introduced the idea of vector quantitization, then motivated the idea of nearest neighbors sampling to output discrete encodings.

In the implementation of the VQ-VAE, we kept a vector embedding table that can be updated through backpropagation. In the nearest neighbors sampling, we first calculated the encoding from the image, then found the closest vector embedding. The formal equation is:

$$\arg\min_{e_j} ||z_e(x_i) - e_j||_2^2$$

We guided students towards discovering this, and the corresponding (and unintuitive) method for getting gradients through this process. From there, for each pixel of the image, we had a sampled pixel we could then send into the decoder for image reconstruction, similar to the VAE architecture when sampling the normal distribution.

3 The VQ-VAE Loss Function and Training

Another one of the primary learning objectives was a deeper understanding of the loss functions and theoretical underpinnings of the VQ-VAE. In the written portion of the homework, we cover the motivation for the loss function of the VAE as a special case of a general variational model, giving students the opportunity to practice and exercise this high-level, abstract understanding in the notebook. In particular, we asked them to implement the forward pass (including the reparameterization trick) and the loss function for the base VAE. After this, the VQ-VAE section in the written section goes through a very practical, "do what you can"-style motivation for the design choices of the VQ-VAE, and again we give the students the opportunity to turn theory into practice through implementing the VQ-VAE in the demo.

4 Image Generation

Another learning objective we wanted students to learn is how to sample images using the VQ-VAE architecture. We wanted to create a clear motivation between the VAE architecture and the VQ-VAE architecture. We emphasised this through the loss function of the VAE, which was:

$$\arg\max_{\theta,\phi} \mathbb{E}[log(p_\theta(x_i|z))] - \lambda D_{KL}(q_\phi(z|x_i)||p(z))$$

We had students derive this optimization function in the written homework to outline that we wanted to minmize reconstruction loss while also keeping the sampling function close to what we actually want to sample from (i.e $N(0, I)$). We had students do this in code by sampling the normal and sending the result to the decoder to generate MNIST digits.

From there, we highlighted the disconnection between VQ-VAE sampling and VAE sampling, as VQ-VAE sampling relied on discrete distributions. We did this by having students engineer the loss function for VQ-VAEs using the motivations of the VAE architecture. From there, we highlighted one problem with sampling from the VQ-VAE, which was that we don't know the exact distribution to sample from when trying to generate images, just the discrete patterns we see after training. Thus, one problem in the written assignment has students engineering a way to generate these discrete encodings so they can sample from the VQ-VAE itself.

We wanted students to figrue out there really wasn't any mathematical way to create these encodings (mathematical meaning any type of sampling function). We motivated this by creating a uniform sampling function that samples each encoding linearly, which should output garbage. We wanted students to figure out that they can fit a VAE on the encodings themselves so we can start sampling from the latent distribution. The VQ-VAE architecture uses the PixelCNN architecture in their paper, which was what we also used.

5 Results

We wanted students to have a tangible sense of what the latent space and reconstructions of the different architectures looked like so they could build a more intuitive sense of the strengths and weaknesses of the different architectures. Specifically, we added many spots in the code which visualized the outputs of the autoencoder model and asked the student to stop and ponder, critically considering what they were seeing. Lastly, we wanted students to walk away with additional resources and links to more state-of-the-art architectures that they could explore on their own if they wanted. To this end, we made sure to keep a list of references to all papers and guides that we referred to while building the demos, as well as a small survey of the most recent SoTA architectures building off of the idea of variational inference.