Programming for AI[1]

Kartik Srinivas , Ojjas Tyagi

May 6, 2024

[1] AI1104 - Dr PN. Karthik

Note: To see the animations in detail, please see the `README.md` file in the Github repository or the code submission.

Idea

The Idea was to program both auto-encoders and variational autoencoders from scratch. Furthermore, we wanted to see what their latent spaces looked like and wanted to visualize them. To visualize them we wanted to create animations of how the generated images would change when we move in the latent spaces of these two types of models. Furthermore, we provide a PCA embeddings of examples in the VAE latent, to notice how the latent manifold looks like in 2D.

Autoencoder

The Autoencoder is trained using a simple $L-2$ reconstruction loss, The encoder (ϕ) and decoder (ρ) both contain Linear layers with RELU activation functions. The bottleneck dimension is around 20 dimensions. The images are flattened, and then passed through the encoder to a bottleneck after which they are expanded back to the original dimensions. The autoencoder is much easier to train [2]

$$\min_{\rho,\phi} \mathbb{E}_{x\sim p_x} \|\rho(\phi(x)) - x\|^2 \tag{1}$$

[2] Checkout the Loss evolution per 300 iterations in the *A2.ipynb* file

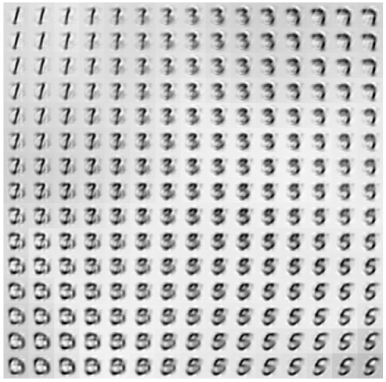


Figure 1: Moving in VAE latent space (see *A2.ipynb*)

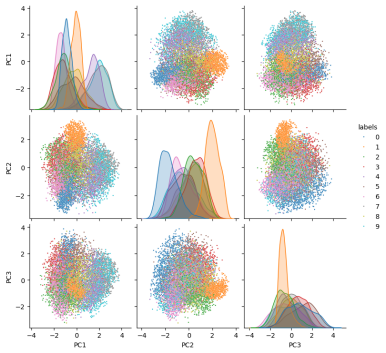
Variational Autoencoder

In this type of model, in-order to "smooth" the latent space, the encoder predicts a distribution of images rather than a single point in the latent dimension. The decoder then samples from this distribution and then decodes the latent to an image. Since the sampling is not a differentiable operation, we can use a reparametrization trick to backpropagate gradients into the predicted mean and covariance of the latent distribution.

$$\min_{\rho,\phi} \mathbb{E}_{x\sim p_x} \underbrace{\mathbb{E}_{z\sim q_\phi(z|x)} \log p_\rho(x|z)}_{\text{Sample from Latent and pass through } \rho \text{ to get "similar image"}} + \underbrace{\mathbb{KL}(q_\phi||\mathcal{N}_I)}_{\text{Calibrate to Normal}} \tag{2}$$

Visualization of the distributions of latents can be achieved through:-

Doing PCA in 2 or 3 principal axises which allows us to observe the patterns among the various digits such as 3,5 and 8 being correlated to each other and the pairs (2,7),(0,1) and (4,6) being far apart



GAN

Although we tried training a Generative Adversarial Network, the training was a bit too hard to get on with. With a learning rate of approximately $1e-6$, both the generator and discriminator losses were decrementing, however this requires atleast 10-50 more epochs for convergence. GAN's tend to be quite finnicky with Hyper-parameters. Check out the loss evolution in the figure for around 2-3 epochs