Deep Generative Models

Chapter 5: Variational Inference and VAEs

Ali Bereyhi

`ali.bereyhi@utoronto.ca`

Department of Electrical and Computer Engineering

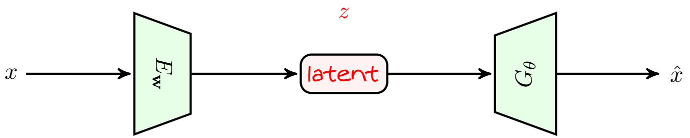
University of Toronto

Summer 2025

Early Work on VAE

For some time: *people considered using decoder of AEs to generate data*

1. *train an encoder-decoder architecture*
2. *sample latent at random and pass it through decoder*



The idea however does not work in this basic form, because

$$\boxed{\textit{prior distribution of the latent is unknown}}$$

Early Work: *Kingma and Welling*

Kingma and Welling used variational inference *to control the latent prior*

this somehow gave birth to VAEs

In their basic experiments, *they considered MLP encoder and decoder*

- *decoder computes feature* $h = \tanh\left(\texttt{Linear}\left(z\right)\right)$ *and then*

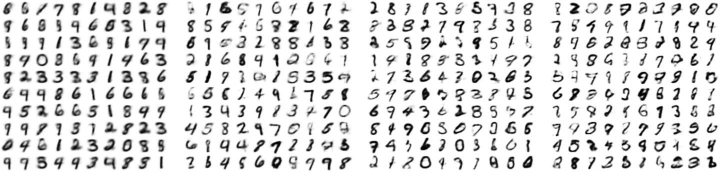
$$\mu = \texttt{Linear}\left(h\right)$$
$$\log \sigma = \texttt{Linear}\left(h\right)$$

- *encoder works exactly the same with* x *and* z *swapped*

Early Work: *Kingma and Welling*

The results on Frey Face dataset is acceptable

and the impact of latent size can be easily seen via MNIST

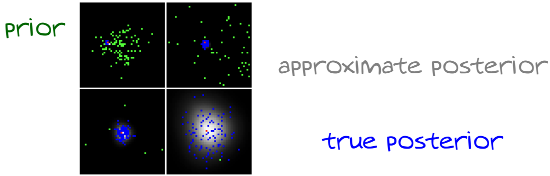


(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Early Work: *Rezende, Mohamed, and Wierstra*

Kingma and Welling motivated the others to dive deeper

- *Rezende et al. provided solid base for VAE*
- *They looked into the stochastic computation graph of VAEs*
- *They visualized the true and approximate posterior*

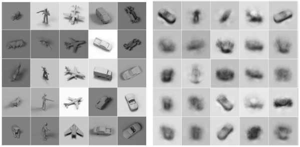


In their experiment, they considered simple MLP encoder and decoder

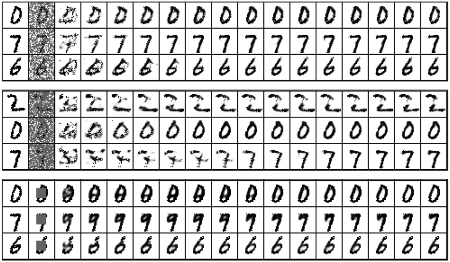
- *both modules were three-layer MLPs*
- *they also investigated other applications, e.g., image inpainting*

Early Work: *Rezende, Mohamed, and Wierstra*

Since model is simple, we get lower quality on more complex datasets like NORB

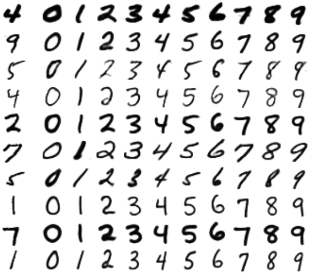


But on simple MNIST, we can do very well

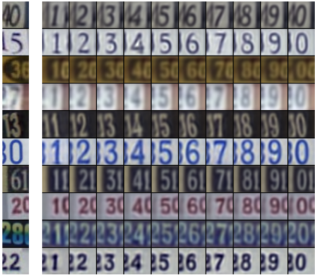


Deep Convolutional VAE

People started to use CNNs to better encode and decode



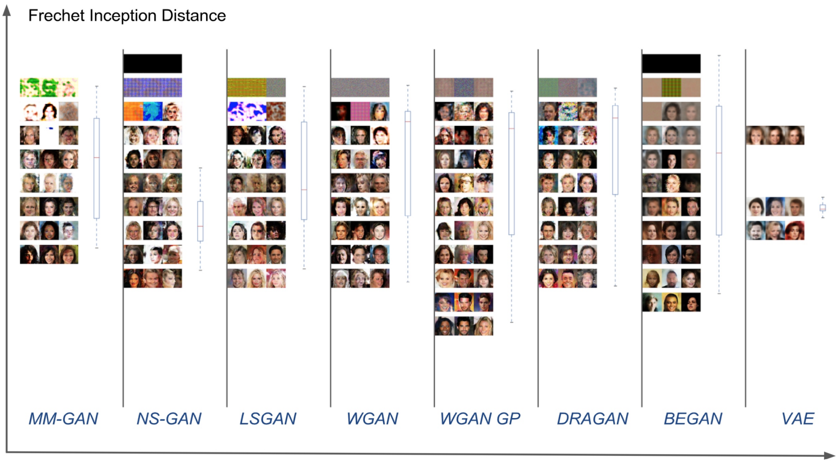
(b) MNIST analogies



(c) SVHN analogies

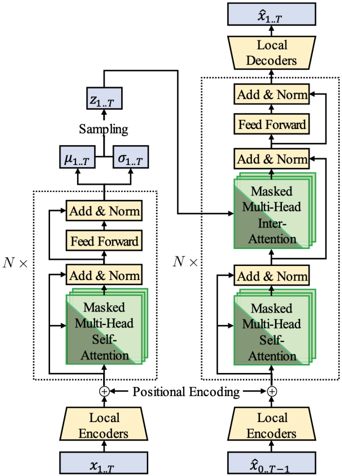
ResNet-VAEs

The results motivated people to further improve encoder and decoder



Transformer-VAEs

The results motivated people to further improve encoder and decoder



VAE Limits: *Sharpness*

Despite computational advances: *there are a few key restrictions to VAEs*

1. *VAE visual output is quite blurry* $\rightsquigarrow$ *for a generic data: not too sharp*
   - ↳ *This is observed simply by comparing VAE and GAN outputs*

   - ↳ *The reason is intuitively understandable*
     - *The output is sampled conditionally from a Gaussian distribution*
     - *This means that after generating* $\mu = G_\theta(z)$ *we compute*

$$x = \mu + \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

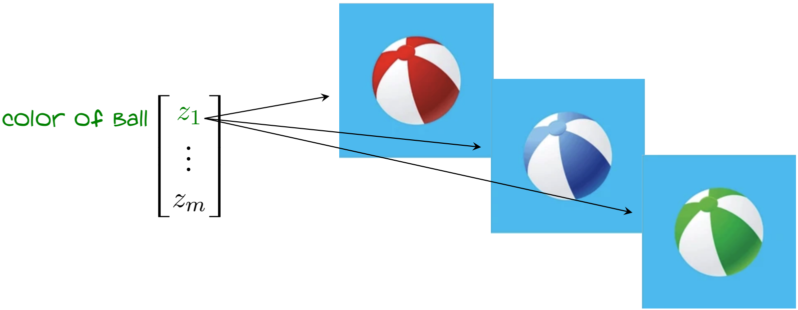
which is always a noisy image!

Disentanglement

Disentanglement in Latent Representation

A latent representation $z \in \mathbb{R}^m$ is bluewell-disentangled if

- *each z_i corresponds to a single generative factor, e.g., position or color*
- *changing z_i only affects the corresponding factor leaving others invariant*



Disentanglement via β-VAE

In VAEs: we compute a latent representation

- *If latent is well-disentangled we can better modify generation*
- *This is an ideal property for the decoder*

---

2. *VAEs have limited control over latent disentanglement*
   ↳ *There is no parameter in encoder or decoder to impose such property*
   ↳ *Intuitively, we can improve disentanglement by increasing latent dimension*
   ↳ *Larger latent size however introduces redundant features as well*

A better approach is the so-called β-VAE modification

$$\beta - \mathrm{ELBO}\left(\mathbf{w}, \theta | x\right) = \hat{\mathbb{E}}_{z \sim Q_{\mathbf{w}}} \left\{ \log P_\theta \left(x|z\right)\right\} - \beta D_{\mathrm{KL}} \left(Q_{\mathbf{w}} \| \mathcal{N}^0\right)$$

↳ *As $\beta \uparrow \rightsquigarrow$ KL penalizes stronger $\rightsquigarrow$ latent becomes statistically i.i.d.*
↳ *Statistically i.i.d. z is better disentangled*

Posterior Collapse

Posterior Collapse

Posterior collapse refers to the case in which decoder learns to ignore latent during training

+ *What does that mean?!*
– *It means that the decoder learns to sample data by its own without caring about encoder!*

Say we are training a VAE: with posterior collapse we could say that

- *loss computed by passing encoder's output through decoder, and*
- *loss computed by passing an independent latent through decoder*

are roughly the same!

Posterior Collapse

Though posterior collapse does not directly impact generation, we can say

- *the encoder of such VAE is useless!*
  - ↳ *it gives a random latent representation for data samples*
- *decoder mainly learns to fit to the training samples* $\rightsquigarrow$ *when sampling*
  - ↳ *it can show worse performance as it did not learn any structure*
  - ↳ *it might stick to a few samples* $\rightsquigarrow$ *it shows mode collapse*

\+ *How do we know if posterior collapse is happening?!*

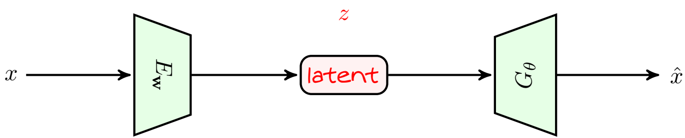
– *If during training* $D_{\mathrm{KL}}\left(Q_{\mathbf{w}} \| \mathcal{N}^0\right)$ *gets very close to zero*

---

3. *In VAEs with powerful decoder, we can observe posterior collapse*
   - ↳ *We can control the KL divergence term as in* β*-VAE*
   - ↳ *We can also reduce the capacity of the decoder*

Treating Sharpness Issue

Looking at sharpness issue: *people thought of old AE-based generation idea*



This old idea says

- ✔ *train deterministic encoder $E_{\mathbf{w}}$ and decoder G_θ for perfect reconstruction*

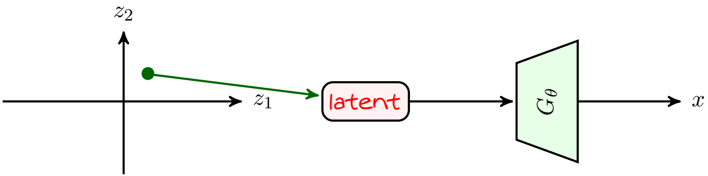
$$R\left(\mathbf{w}, \theta | x\right) = \left\| x - G_\theta\left(E_{\mathbf{w}}\left(x\right)\right)\right\|$$

- ? *learn the distribution of the latent $P\left(z\right)$ using encoder*
- ✔ *sample $z \sim P\left(z\right)$ and generate $x = G_\theta\left(z\right)$*

Back to AEs: *Perfect Reconstruction*

+ *Why should this basic idea resolves sharpness issue?!*
– Well! There is no *Gaussian sampling at the decoder anymore!*

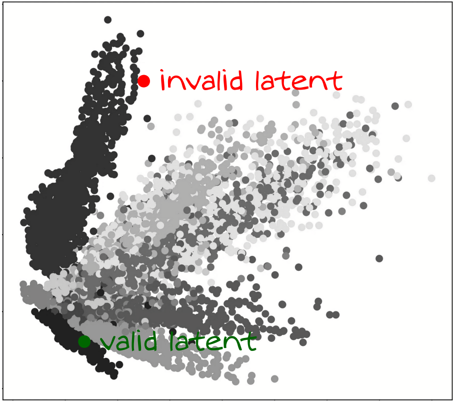
If we are sure z is a valid latent $\rightsquigarrow$ we can compute x perfectly



Challenge: *Unknown Latent Prior*

The key challenge is that *in this case*

prior latent distribution is not controlled and hence is unknown!



Learning Latent Prior

- + *Can't we learn this prior from the data?!*
- – Theoretically, we can!

`AE_General_Latent(`$E_{\mathbf{w}}, G_\theta$`:AE, `P_ϕ`:Prior_Model):`

Initiate the encoder and decoder
for *multiple epochs* **do**
    Train $E_{\mathbf{w}}$ and G_θ for perfect reconstruction
end for
for *all sample x^j in dataset* **do**
    Compute latent representation $z^j = E_{\mathbf{w}}\left(x^j\right)$
    Keep it in the latent dataset $\mathbb{Z} \leftarrow z^j$
end for
for *multiple epochs on* $\mathbb{Z}$ **do**
    Train P_ϕ `#learn latent prior`
end for
return *Latent prior and AE*

Learning Latent Prior

+ *What should be P_ϕ?!*
– It could be any explicit model we had in Chapter 3

From Chapter 3 we know that

it is not really easy to train P_ϕ on a continuous latent

This motivated for using a discrete latent

- *Latent entries belongs to a predefined codebook of k codewords*

$$z_i \in \mathcal{C} = \{c_1, \dots, c_k\}$$

- *Given that latent space is finite, we can learn P_ϕ more efficiently*
    ↳ *For example, we can train an AR model with k-output Softmax at its end*

VQ-VAE: *Vector Quantized VAE*

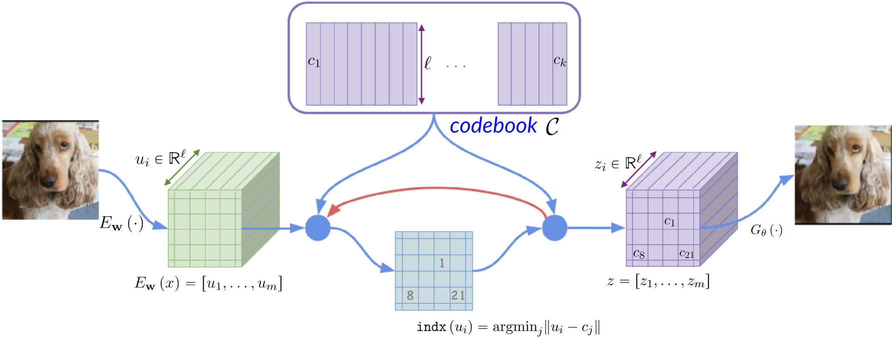
VQ-VAE is an example of generative AEs with discrete latent

- ↳ *It's usually used for visual and audio data*
- ↳ *It quantizes the outputs of the encoder on a codebook*
- ↳ *The codebook can be seen as an embedding*
  - ↳ Its codewords, i.e., $c_j \in \mathcal{C}$, are learnable

! Attention

Despite its name, it is technically not a VAE, since it does not use ELBO anymore

VQ-VAE: *Architecture*



Training VQ-VAE

Note that in this architecture $E_{\mathbf{w}}$, G_θ and codebook are learnable

$$R(\mathbf{w}, \theta, \mathcal{C}|x) = \|x - G_\theta(E_{\mathbf{w}}(x))\|^2 + \beta \|E_{\mathbf{w}}(x) - z\|^2$$

VQ-VAE: *Prior Learning*

To learn the prior, we invoke AR modeling:

- *Define a computational model F_ϕ that extracts context up to entry i*

$$\mathbf{c}_i = F_\phi \left(z_{<i} \right) \in \mathbb{R}^k$$

- *Define a conditional distribution as*

$$P_\phi \left(z_i | z_{<i} \right) = \text{Soft}_{\max} \left(\mathbf{c}_i \right) = \text{Soft}_{\max} \left(F_\phi \left(z_{<i} \right) \right)$$

- *Model the latent prior as*

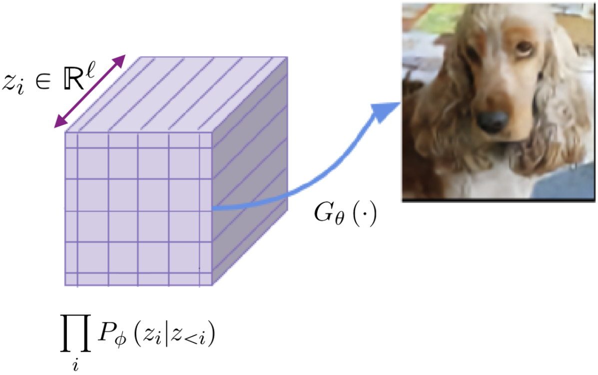
$$P \left(z \right) = \prod_i P_\phi \left(z_i | z_{<i} \right)$$

We train this model on encoded latent samples given by the trained encoder

VQ-VAE: *Sampling*

We can sample data distribution by

1. *sampling the AR latent prior*
2. *passing the latent sample through the decoder*



VQ-VAE: *Sample Output*

In the original VQ-VAE proposal

- *CNN encoder and decoder is deployed*
- *PixelCNN is used to learn latent prior*

Wrap Up

In practice VAEs show *various favorable properties*

- *They are very stable while training*
  - ↳ *ELBO maximization* $\equiv$ *conventional empirical risk minimization*
- *They are easy to sample*
  - ↳ *They are sampled as flow-based models with Gaussian prior*

They could however exhibit

- *blurry output*
  - ↳ *This happens due toGaussian sampling at the decoder*
- *imperfect disentanglement*
  - ↳ *This happens due to limited control on the latent*
- *posterior collapse*
  - ↳ *This happens when the decoder is very strong*

Wrap Up

We can address the key issues of VAEs by various modifications, e.g., by

- *modifying the ELBO term as in β-VAE*
- *quantizing the latent representation as in VQ-VAE*

VAEs are still considered as powerful generative models

- *Like GANs, they are straightforward to implement in simple applications*
- *They can provide us further with latent representation*
  - ↳ *This is in particular very useful when we change modality*

$$\textit{image} \xrightarrow{E_{\mathbf{w}}} \textit{latent} \xrightarrow{\texttt{LM}} \textit{text}$$

Next Stop: Indirect distribution learning by *diffusion process*