Learning Hierarchical Features from Generative Models

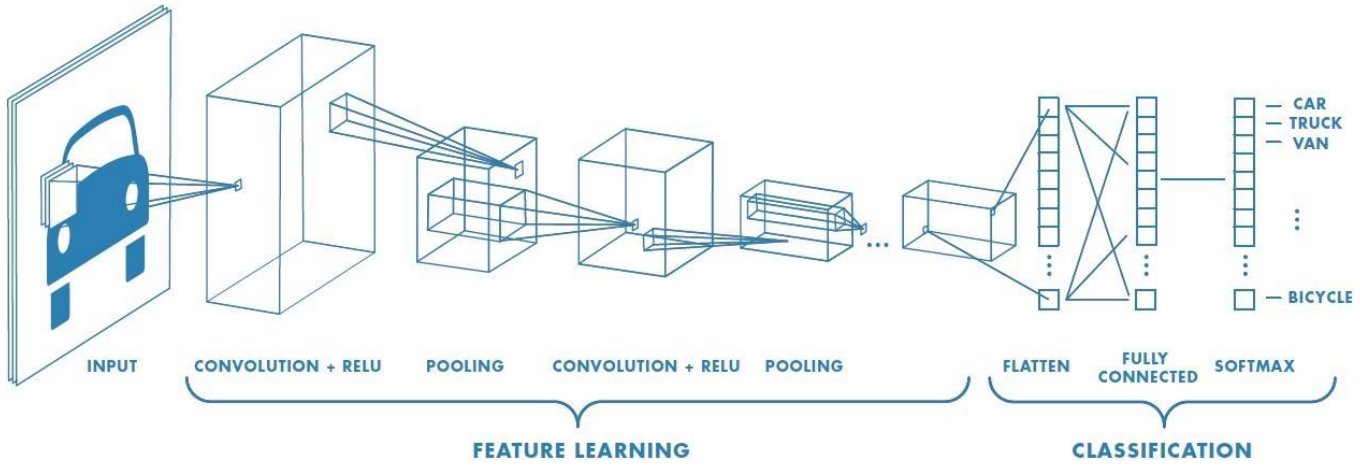
Shengjia Zhao, Jiaming Song, Stefano Ermon

ICML 2017

Presented by Eungyeup Kim

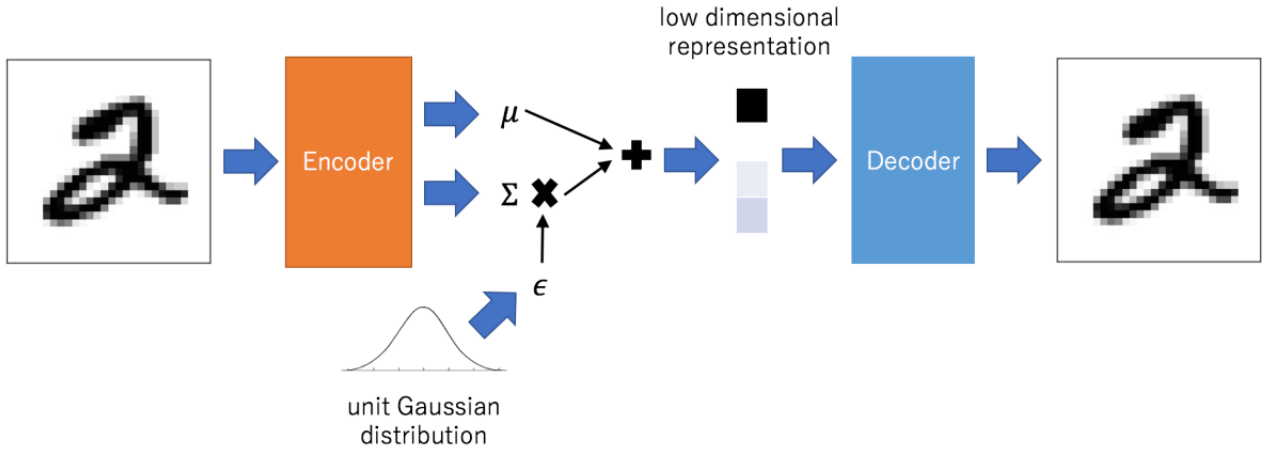
Learning hierarchical features

In CNN



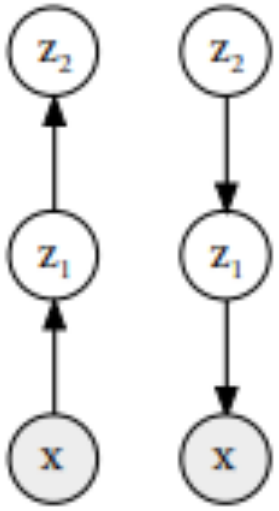
Learn hierarchical features very well

In VAE..?



Learning hierarchical features in VAE

Stacked VAE



$$p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_L) = p(\mathbf{x}|\mathbf{z}_{>0}) \prod_{\ell-1}^{L-1} p(\mathbf{z}_\ell|\mathbf{z}_{>\ell})p(\mathbf{z}_L)$$

where $\mathbf{z}_{>\ell}$ indicates $(\mathbf{z}_{\ell+1}, \cdots, \mathbf{z}_L)$, and $\mathbf{z}_{>0} = (\mathbf{z}_1, \ldots, \mathbf{z}_L)$

Problems :

- Low representational efficiency: A gibbs chain on the bottom layer is used to recover p_{data} exactly.
- Weak feature learning: Gaussian Distribution limits the hierarchical representation we can learn.

Learning hierarchical features in VAE

- A Gibbs chain on the bottom layer is used to recover p_{data} exactly.

$$\mathcal{L}_{ELBO} = E_{p_{data}(\mathbf{x})} E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - E_{p_{data}(\mathbf{x})}[\mathbb{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))]$$

$$= E_{p_{data}(\mathbf{x})q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right]$$

$$= E_{p_{data}(\mathbf{x})q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})}\right] + E_{p_{data}(\mathbf{x})}[\log p(\mathbf{x})]$$

$$= -E_{p_{data}(\mathbf{x})}[\mathbb{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))] - \mathbb{KL}(p_{data}(\mathbf{x})||p(\mathbf{x})) - H(p_{data}(\mathbf{x}))$$

For optimality,

1. $q(z|x) = p(z|x)$
2. $p_{data}(x) = p(x)$

Because the following Gibbs chain converges to $p_{data}(\mathbf{x})$ when it is ergodic

$$\mathbf{z}_1^{(t)} \sim q(\mathbf{z}_1|\mathbf{x}^{(t)})$$
$$\mathbf{x}^{(t+1)} \sim q(\mathbf{x}|\mathbf{z}_1^{(t)})$$

We can replace $q(\mathbf{x}|\mathbf{z}_1^{(t)})$ with $p(\mathbf{x}|\mathbf{z}_1^{(t)})$ using (7) and the chain still converges to $p_{data}(\mathbf{x})$. □

Learning hierarchical features in VAE

- A Gibbs chain on the bottom layer is used to recover p_{data} exactly.

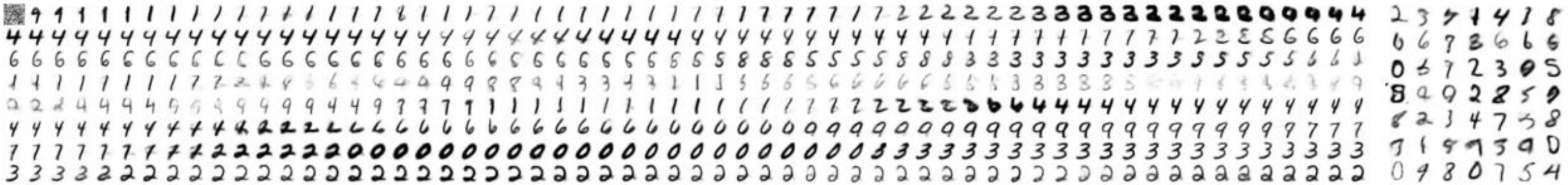


Figure 2. **Left:** Samples obtained by running the Gibbs sampling chain in Proposition 1, using only the bottom layer of a 3-layer recursive hierarchical VAE. **Right:** samples generated by ancestral sampling from the same model. The quality of the samples is comparable, indicating that the bottom layer contains enough information to reconstruct the data distribution.

Learning hierarchical features in VAE

- Gaussian Distribution limits the hierarchical representation we can learn.

Suppose we train $\mathcal{L}_{ELBO}$ in Equation (5) to optimality, we would have

$$p(\mathbf{x}) = p_{data}(\mathbf{x}), q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$$

Recall that

$$q(\mathbf{x}, \mathbf{z}) := p_{data}(\mathbf{x})q(\mathbf{z}|\mathbf{x})$$
$$p(\mathbf{x}, \mathbf{z}) := p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = p(\mathbf{x})p(\mathbf{z}|\mathbf{x})$$

Comparing the two we see that

$$p(\mathbf{x}, \mathbf{z}) = q(\mathbf{x}, \mathbf{z})$$

if the joint distributions are identical, then any conditional distribution would also be identical, which implies that for any $\mathbf{z}_{>\ell}$, $q(\mathbf{z}_\ell|\mathbf{z}_{>\ell}) = p(\mathbf{z}_\ell|\mathbf{z}_{>\ell})$.

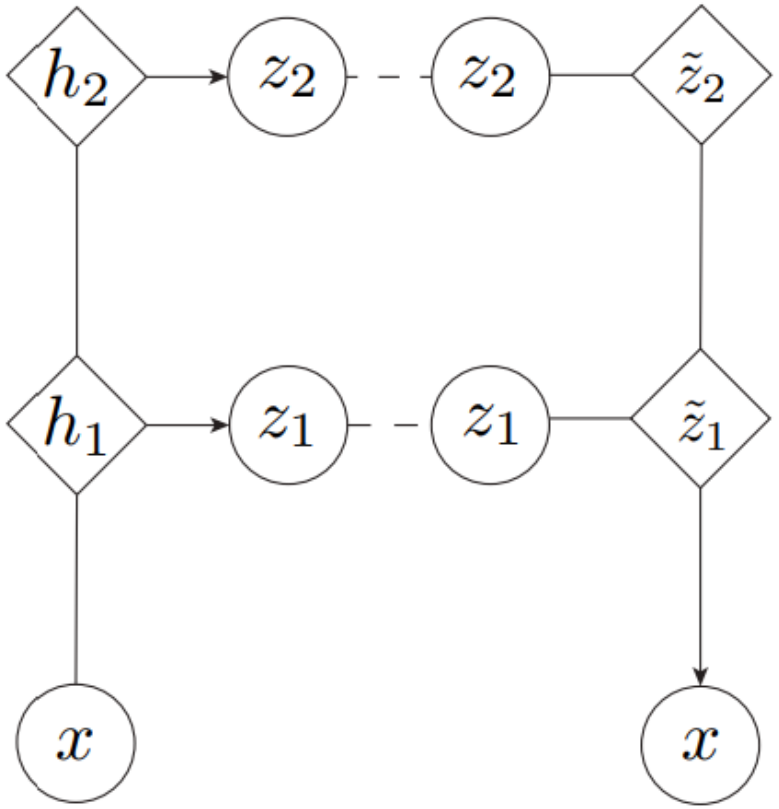
→ This limits the hierarchical representation we learn!!

Learning hierarchical features in VAE

- Gaussian Distribution limits the hierarchical representation we can learn.

Figure 3. A hierarchical three layer VAE with Gaussian conditional distributions $p(\mathbf{z}_l|\mathbf{z}_{l+1})$ does not learn a meaningful feature hierarchy on MNIST and SVHN when trained with the ELBO objective. **Left panel:** Samples generated by sampling noise $\boldsymbol{\epsilon}_1$ at the bottom layer, while holding $\boldsymbol{\epsilon}_2$ and $\boldsymbol{\epsilon}_3$ constant. **Center panel:** Samples generated by sampling noise $\boldsymbol{\epsilon}_2$ at the middle layer, while holding $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_3$ constant. **Right panel:** Samples generated by sampling noise $\boldsymbol{\epsilon}_3$ at the top layer, while holding $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ constant. For both MNIST and SVHN we observe that the top layer represents essentially all the variation in the data (right panel), leaving only very minor local variations for the lower layers (left and center panels). Compare this with the rich hierarchy learned by our VLAE model, shown in Figures 5 and 6.

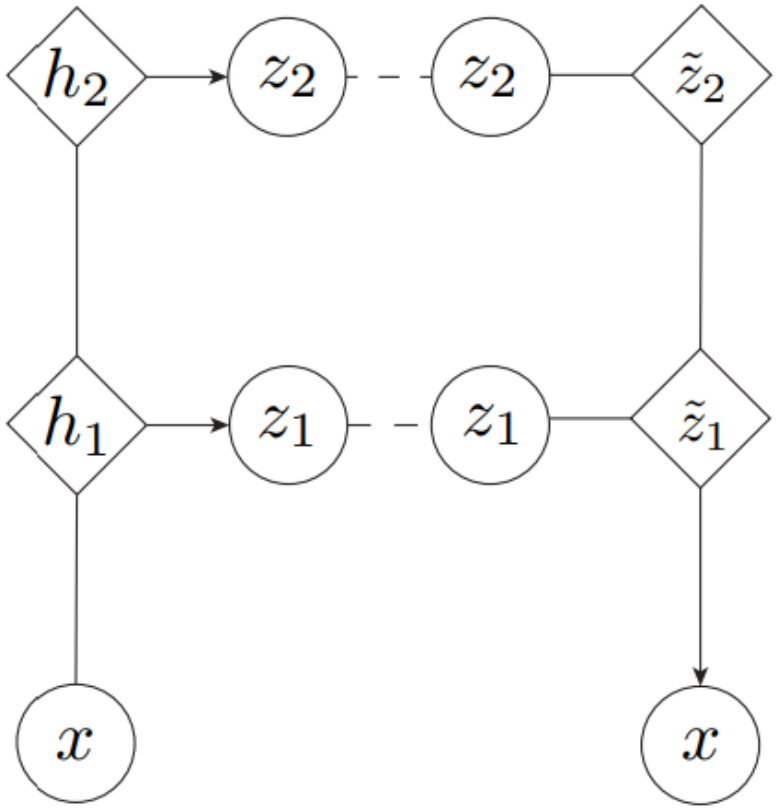
Variational Ladder Autoencoders



- No hierarchical structure over the latent variable $\mathrm{p}(\mathrm{z}) = p(z_1, z_2, \ldots, z_L)$

- Carefully mapping p(x|z) and q(z|x) between z and x
  - If z_i is more abstract than z_j, then z_i requires more expressive networks (depth of network)

Variational Ladder Autoencoders



1) Generative Network:

- We choose $\mathrm{p}(z) = p(z_1, z_2, \ldots, z_L)$ as a standard Gaussian prior.
- Conditional distribution $p(x|z_1, z_2, \ldots, z_L)$ is defined as:

$$\tilde{\mathbf{z}}_L = \mathbf{f}_L(\mathbf{z}_L)$$
$$\tilde{\mathbf{z}}_\ell = \mathbf{f}_\ell(\tilde{\mathbf{z}}_{\ell+1}, \mathbf{z}_\ell) \quad \ell = 1, \cdots, L-1$$
$$\mathbf{x} \sim \mathbf{r}(\mathbf{x}; \mathbf{f}_0(\tilde{\mathbf{z}}_1))$$

2) **Inference Network**:

- We choose $\mathrm{q}(z|x)$ as

$$\mathbf{h}_\ell = \mathbf{g}_\ell(\mathbf{h}_{\ell-1})$$
$$\mathbf{z}_\ell \sim \mathcal{N}(\boldsymbol{\mu}_\ell(\mathbf{h}_\ell), \boldsymbol{\sigma}_\ell(\mathbf{h}_\ell))$$

where $\ell = 1, \cdots, L$, $\mathbf{g}_\ell$, $\boldsymbol{\mu}_\ell$, $\boldsymbol{\sigma}_\ell$ are neural networks, and $\mathbf{h}_0 \equiv \mathbf{x}$.

3) **Objective**:

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \mathbb{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

Experiments

Figure 5. VLAE on MNIST. Generated digits obtained by systematically exploring the 2D latent code from one layer, and randomly sampling from other layers. **Left panel:** The first (bottom) layer encodes stroke width, **Center panel:** the second layer encodes digit width and tilt, **Right panel:** the third layer encodes (mostly) digit identity. Note that the samples are not of state-of-the-art quality only because of the restricted 2-dimensional latent code used to enable visualization.

Figure 6. VLAE on SVHN. Each sub-figure corresponds to images generated when fixing latent code on all layers except for one, which we randomly sample from the prior distribution. From left to right the random sampled layer go from bottom layer to top layer. **Left panel:** The bottom layer represents color schemes; **Center-left panel:** the second layer represents shape variations of the same digit; **Center-right panel:** the third layer represents digit identity (interestingly these digits have similar style although having different identities); **Right panel:** the top layer represents the general structure of the image.

Experiments

Figure 7. VLAE on CelebA. Each sub-figure corresponds to images generated when fixing latent code on all layers except for one, which we randomly sample from the prior distribution. From left to right the random sampled layer go from bottom layer to top layer. **Left panel:** The bottom layer represents ambient color; **Center-left panel:** the second bottom layer represents skin and hair color; **Center-right panel:** the second top layer represents face identity; **Right panel:** the top layer presents pose and general structure.