Deep Generative Models

Lecture supp

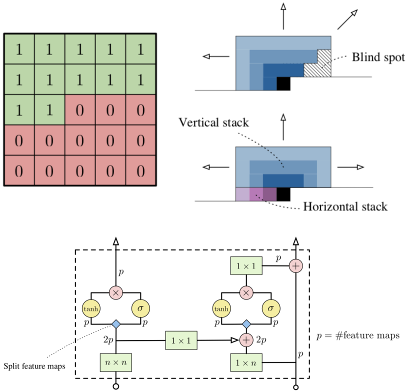
Roman Isachenko

Moscow Institute of Physics and Technology

Autumn, 2021

GatedPixelCNN

GatedPixelCNN (2016)

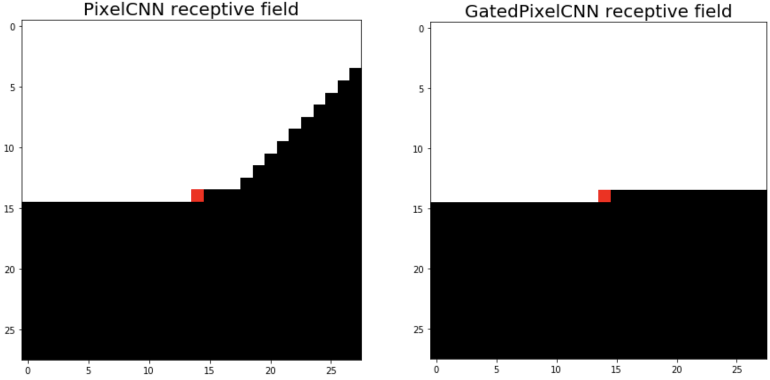


1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Van den Oord A. et al. Conditional image generation with pixelcnn decoders

https://arxiv.org/pdf/1606.05328.pdf

GatedPixelCNN (2016)



Van den Oord A. et al. Conditional image generation with pixelcnn decoders
https://arxiv.org/pdf/1606.05328.pdf

Extensions

- **PixelCNN++**: *Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications* https://arxiv.org/pdf/1701.05517.pdf (mixture of logistics instead of softmax);
- **PixelSNAIL**: *An Improved Autoregressive Generative Model* https://arxiv.org/pdf/1712.09763.pdf (self-attention to learn optimal autoregression ordering).

ELBO gradient, Log derivative trick

ELBO gradient (E-step, $\nabla_{\phi}\mathcal{L}(\phi, \boldsymbol{\theta})$)

$$\mathcal{L}(\phi, \boldsymbol{\theta}) = \mathbb{E}_q \log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) - KL(q(\mathbf{Z}|\mathbf{X}, \phi)||p(\mathbf{Z})) \to \max_{\boldsymbol{\phi}, \boldsymbol{\theta}}.$$

Difference from M-step: density function $q(\mathbf{z}|\mathbf{x}, \phi)$ depends on the parameters ϕ, it is impossible to use Monte-Carlo estimation:

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}(\phi, \boldsymbol{\theta}) = \int \nabla_{\boldsymbol{\phi}} q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) d\mathbf{Z} - \nabla_{\boldsymbol{\phi}} KL$$

Log-derivative trick

$$\nabla_{\xi} q(\eta|\xi) = q(\eta|\xi) \left(\frac{\nabla_{\xi} q(\eta|\xi)}{q(\eta|\xi)} \right) = q(\eta|\xi) \nabla_{\xi} \log q(\eta|\xi).$$

$$\nabla_{\boldsymbol{\phi}} q(\mathbf{Z}|\mathbf{X}, \phi) = q(\mathbf{Z}|\mathbf{X}, \phi) \nabla_{\boldsymbol{\phi}} \log q(\mathbf{Z}|\mathbf{X}, \phi).$$

ELBO gradient (E-step, $\nabla_{\phi}\mathcal{L}(\phi, \boldsymbol{\theta})$)

$$\nabla_{\phi}\mathcal{L}(\phi, \boldsymbol{\theta}) = \int \nabla_{\phi} q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) d\mathbf{Z} - \nabla_{\phi} KL =$$

$$= \int q(\mathbf{Z}|\mathbf{X}, \phi) \big[\nabla_{\phi} \log q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta})\big] d\mathbf{Z} - \nabla_{\phi} KL$$

After applying log-reparametrization trick, we are able to use Monte-Carlo estimation:

$$\nabla_{\phi}\mathcal{L}(\phi, \boldsymbol{\theta}) \approx n \nabla_{\phi} \log q(\mathbf{z}_i^*|\mathbf{x}_i, \phi) \log p(\mathbf{x}_i|\mathbf{z}_i^*, \boldsymbol{\theta}) - \nabla_{\phi} KL,$$

$$\mathbf{z}_i^* \sim q(\mathbf{z}_i|\mathbf{x}_i, \phi).$$

Problem

Unstable solution with huge variance.

Solution

Reparametrization trick

Mean field approximation

Bayesian framework

Bayes theorem

$$p(\mathbf{t}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{t})p(\mathbf{t})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathbf{t})p(\mathbf{t})}{\int p(\mathbf{x}|\mathbf{t})p(\mathbf{t})d\mathbf{t}}$$

- $\mathbf{x}$ – observed variables, $\mathbf{t}$ – unobserved variables (latent variables/parameters);
- $p(\mathbf{x}|\mathbf{t})$ – likelihood;
- $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{t})p(\mathbf{t})d\mathbf{t}$ – evidence;
- $p(\mathbf{t})$ – prior distribution, $p(\mathbf{t}|\mathbf{x})$ – posterior distribution.

Meaning

We have unobserved variables $\mathbf{t}$ and some prior knowledge about them $p(\mathbf{t})$. Then, the data $\mathbf{x}$ has been observed. Posterior distribution $p(\mathbf{t}|\mathbf{x})$ summarizes the knowledge after the observations.

Variational Lower Bound

We have set of objects $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. The goal is to perform Bayesian inference on the unobserved variables $\mathbf{T} = \{\mathbf{t}_i\}_{i=1}^n$.

Evidence Lower Bound (ELBO)

$$\begin{aligned}
\log p(\mathbf{X}) &= \log \frac{p(\mathbf{X}, \mathbf{T})}{p(\mathbf{T}|\mathbf{X})} = \\
&= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{p(\mathbf{T}|\mathbf{X})} d\mathbf{T} = \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T}) q(\mathbf{T})}{p(\mathbf{T}|\mathbf{X}) q(\mathbf{T})} d\mathbf{T} = \\
&= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{q(\mathbf{T})} d\mathbf{T} + \int q(\mathbf{T}) \log \frac{q(\mathbf{T})}{p(\mathbf{T}|\mathbf{X})} d\mathbf{T} = \\
&= \mathcal{L}(q) + KL(q(\mathbf{T}) || p(\mathbf{T}|\mathbf{X})) \geq \mathcal{L}(q).
\end{aligned}$$

We would like to maximize lower bound $\mathcal{L}(q)$.

Mean field approximation

Independence assumption

$$q(\mathbf{T}) = \prod_{i=1}^{k} q_i(\mathbf{T}_i), \quad \mathbf{T} = [\mathbf{T}_1, \dots, \mathbf{T}_k],\, \mathbf{T}_j = \{\mathbf{t}_{ij}\}_{i=1}^{n},\, \mathbf{t}_i = \{\mathbf{T}_{ij}\}_{j=1}^{k}.$$

Block coordinate optimization of ELBO for $q_j(\mathbf{T}_j)$

$$\begin{aligned}
\mathcal{L}(q) = \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{q(\mathbf{T})} d\mathbf{T} &= \int \left[\prod_{i=1}^{k} q_i(\mathbf{T}_i)\right] \log \frac{p(\mathbf{X}, \mathbf{T})}{\left[\prod_{i=1}^{k} q_i(\mathbf{T}_i)\right]} \prod_{i=1}^{k} d\mathbf{T}_i = \\
&= \int \left[\prod_{i=1}^{k} q_i\right] \log p(\mathbf{X}, \mathbf{T}) \prod_{i=1}^{k} d\mathbf{T}_i - \sum_{i=1}^{k} \int \left[\prod_{j=1}^{k} q_j\right] \log q_i \prod_{j=1}^{k} d\mathbf{T}_j = \\
&= \int q_j \left[\int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i \, d\mathbf{T}_i\right] d\mathbf{T}_j - \\
&\quad - \int q_j \log q_j \, d\mathbf{T}_j + \mathrm{const}(q_j) \to \max_{q_j}
\end{aligned}$$

Mean field approximation

Block coordinate optimization of ELBO for $q_j(\mathbf{T}_j)$

$$\mathcal{L}(q) = \int q_j \left[\int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{T}_i \right] d\mathbf{T}_j - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) =$$

$$= \int q_j \log \hat{p}(\mathbf{X}, \mathbf{T}_j) d\mathbf{T}_j - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) \to \max_{q_j}.$$

Here we introduce

$$\log \hat{p}(\mathbf{X}, \mathbf{T}_j) = \int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{T}_i = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}(q_j)$$

Final ELBO derivation for $q_j(\mathbf{T}_j)$

$$\mathcal{L}(q) = \int q_j(\mathbf{T}_j) \log \hat{p}(\mathbf{X}, \mathbf{T}_j) d\mathbf{T}_j - \int q_j(\mathbf{T}_j) \log q_j(\mathbf{T}_j) d\mathbf{T}_j + \text{const}(q_j) =$$

$$\int q_j(\mathbf{T}_j) \log \frac{\hat{p}(\mathbf{X}, \mathbf{T}_j)}{q_j(\mathbf{T}_j)} d\mathbf{T}_j + \text{const}(q_j) =$$

$$= -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \to \max_{q_j}.$$

Mean field approximation

Independence assumption

$$q(\mathbf{T}) = \prod_{i=1}^{k} q_i(\mathbf{T}_i), \quad \mathbf{T} = [\mathbf{T}_1, \ldots, \mathbf{T}_k], \quad \mathbf{T}_j = \{\mathbf{t}_{ij}\}_{i=1}^{n}.$$

ELBO

$$\mathcal{L}(q) = -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \to \max_{q_j}.$$

Solution

$$q_j(\mathbf{T}_j) = \text{const} \cdot \hat{p}(\mathbf{X}, \mathbf{T}_j)$$

$$\log \hat{p}(\mathbf{X}, \mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

Mean field approximation

ELBO

$$\mathcal{L}(q) = -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \to \max_{q_j}.$$

Solution

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

Assumptions:

- $\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2] = [\mathbf{Z}, \boldsymbol{\theta}]$, $q(\mathbf{T}) = q(\mathbf{T}_1) \cdot q(\mathbf{T}_2) = q(\mathbf{Z}) \cdot q(\boldsymbol{\theta})$.
- restrict a class of probability distributions for $\boldsymbol{\theta}$ to Dirac delta functions:
  $$q_2 = q(\mathbf{T}_2) = q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*).$$
  Under the restrictions the exact solution for q_2 is not reached (KL can be greater than 0).

Mean field approximation

General solution

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

Solution for $q_1 = q(\mathbf{Z})$

$$\log q(\mathbf{Z}) = \int q(\boldsymbol{\theta}) \log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const} =$$

$$= \int \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const} =$$

$$= \log p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^*) + \text{const}.$$

EM-algorithm (E-step)

$$q(\mathbf{Z}) = \arg\max_q \mathcal{L}(q, \boldsymbol{\theta}^*) = \arg\min_q KL(q || p) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^*).$$

Mean field approximation

ELBO

$$\mathcal{L}(q) = -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \to \max_{q_j}.$$

ELBO maximization w.r.t. $q_2 = q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$

$$\begin{aligned}
\mathcal{L}(q_1, q_2) &= -KL(q(\boldsymbol{\theta}) || \hat{p}(\mathbf{X}, \boldsymbol{\theta})) + \text{const}(\boldsymbol{\theta}^*) \\
&= \int q(\boldsymbol{\theta}) \log \frac{\hat{p}(\mathbf{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} + \text{const}(\boldsymbol{\theta}^*) \\
&= \int q(\boldsymbol{\theta}) \log \hat{p}(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const}(\boldsymbol{\theta}^*) \\
&= \int \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \log \hat{p}(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const}(\boldsymbol{\theta}^*) \to \max_{\boldsymbol{\theta}^*}
\end{aligned}$$

Mean field approximation

ELBO maximization w.r.t. $q_2 = q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$

$$\mathcal{L}(q_1, q_2) = \int \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \log \hat{p}(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const} = \log \hat{p}(\mathbf{X}, \boldsymbol{\theta}^*) + \text{const}$$
$$= \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const} = \mathbb{E}_{q_1} \log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}^*) + \text{const}$$
$$= \int q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}^*) d\mathbf{Z} + \log p(\boldsymbol{\theta}^*) + \text{const} \to \max_{\boldsymbol{\theta}^*}$$

EM-algorithm (M-step)

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} d\mathbf{Z}$$
$$= \int q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) d\mathbf{Z} + \text{const} \to \max_{\boldsymbol{\theta}}$$

Mean field approximation

Solution

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

EM algorithm (special case)

- Initialize $\boldsymbol{\theta}^*$;
- E-step

$$q(\mathbf{Z}) = \arg\max_q \mathcal{L}(q, \boldsymbol{\theta}^*) = \arg\min_q KL(q||p) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*);$$

- M-step

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta});$$

- Repeat E-step and M-step until convergence.

Flows intuition

Flows intuition

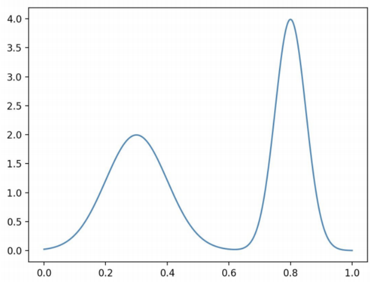
Let ξ be a random variable with density $p(\xi)$. Then

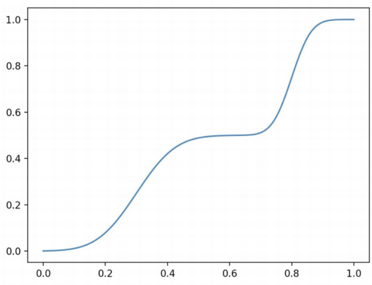
$$\eta = F(\xi) = \int_{-\infty}^{\xi} p(t)dt \sim U[0, 1].$$

$$P(\eta < y) = P(F(\xi) < y) = P(\xi < F^{-1}(y)) = F(F^{-1}(y)) = y$$

Hence

$$\eta \sim U[0, 1]; \quad \xi = F^{-1}(\eta) \quad \Rightarrow \quad \xi \sim p(\xi).$$





---

image credit: https://sites.google.com/view/berkeley-cs294-158-sp19/home

Flows intuition

- Let $z \sim p(z)$ is a random variable with base distribution $p(z) = U[0, 1]$.
- Let $x \sim p(x)$ is a random variable with complex distribution $p(x)$ and cdf $F(x)$.
- Then noise variable z can be transformed to x using inverse cdf F^{-1} ($x = F^{-1}(z)$).

How to transform random variable z which has a distribution different from uniform to x?

- Let $z \sim p(z)$ is a random variable with base distribution $p(z)$ and cdf $G(z)$.
- Then $z_0 = G(z)$ has base distribution $p(z_0) = U[0, 1]$.
- Let $x \sim p(x)$ is a random variable with complex distribution $p(x)$ and cdf $F(x)$.
- Then noise variable z can be transformed to x using cdf G and inverse cdf F^{-1} ($x = F^{-1}(z_0) = F^{-1}(G(z))$).

ELBO interpretations

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})).$$

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int q(\mathbf{z}|\mathbf{x}, \phi) \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \phi)} d\mathbf{z}.$$

- Evidence minus posterior KL

$$\mathcal{L}(q, \boldsymbol{\theta}) = \log p(\mathbf{x}|\boldsymbol{\theta}) - KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})).$$

- Average negative energy plus entropy

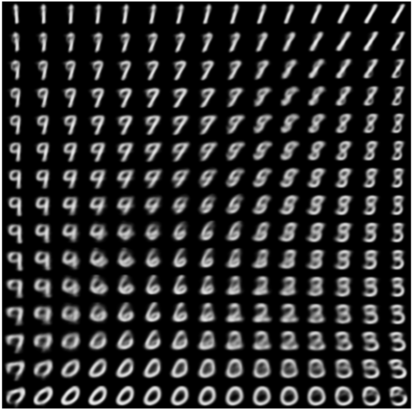
$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)} \left[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) - \log q(\mathbf{z}|\mathbf{x}, \phi)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)} \log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) + \mathbb{H}\left[q(\mathbf{z}|\mathbf{x}, \phi)\right].\end{aligned}$$

- Average reconstruction minus KL to prior

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)} \left[\log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}, \phi)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z})).\end{aligned}$$

Variational Autoencoder

Generated images for latent objects $\mathbf{z}$ sampled from prior $\mathcal{N}(0, \mathbf{I})$



---

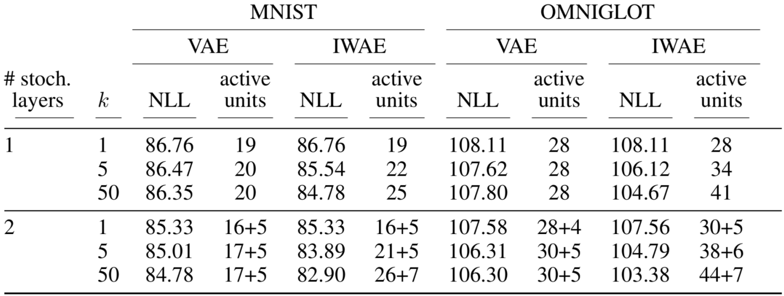
image credit: https://habr.com/ru/post/331664

IWAE, active units, posterior collapse

IWAE, 2015

How to determine whether all VAE latent variables are informative?

$$A_i = \mathrm{cov}_{\mathbf{x}}\left(\mathbb{E}_{q(z_i|\mathbf{x})}[z_i]\right) > 0.01 \quad \Leftrightarrow \quad z_i \text{ is active}$$



		MNIST				OMNIGLOT			
		VAE		IWAE		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19	108.11	28	108.11	28
	5	86.47	20	85.54	22	107.62	28	106.12	34
	50	86.35	20	84.78	25	107.80	28	104.67	41
2	1	85.33	16+5	85.33	16+5	107.58	28+4	107.56	30+5
	5	85.01	17+5	83.89	21+5	106.31	30+5	104.79	38+6
	50	84.78	17+5	82.90	26+7	106.30	30+5	103.38	44+7

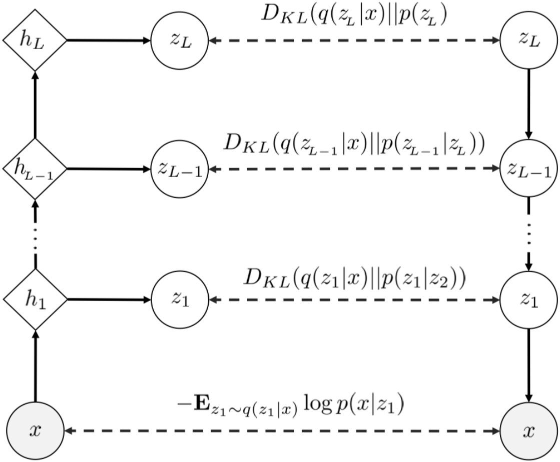
---

https://arxiv.org/pdf/1509.00519.pdf

PixelVAE, Hierarchical VAE

PixelVAE, 2016

Hierarchical VAE



---

https://arxiv.org/pdf/1611.05013.pdf

PixelVAE, 2016

Hierarchical decomposition

$$p(\mathbf{z}_1, \dots, \mathbf{z}_L) = p(\mathbf{z}_L) p(\mathbf{z}_{L-1}|\mathbf{z}_L) \dots p(\mathbf{z}_1, \mathbf{z}_2);$$
$$q(\mathbf{z}_1, \dots, \mathbf{z}_L|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \dots q(\mathbf{z}_L|\mathbf{x}).$$

ELBO

$$\begin{aligned}
\mathcal{L}(q, \boldsymbol{\theta}) &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\theta}) - KL(q(\mathbf{z}_1, \dots, \mathbf{z}_L|\mathbf{x}) || p(\mathbf{z}_1, \dots, \mathbf{z}_L)) \\
&= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\theta}) - \int \prod_{j=1}^{L} q(\mathbf{z}_j|\mathbf{x}) \sum_{i=1}^{L} \log \frac{q(\mathbf{z}_i|\mathbf{x})}{p(\mathbf{z}_i|\mathbf{z}_{i+1})} d\mathbf{z}_1 \dots d\mathbf{z}_L \\
&= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\theta}) - \sum_{i=1}^{L} \int \prod_{j=1}^{L} q(\mathbf{z}_j|\mathbf{x}) \log \frac{q(\mathbf{z}_i|\mathbf{x})}{p(\mathbf{z}_i|\mathbf{z}_{i+1})} d\mathbf{z}_1 \dots d\mathbf{z}_L \\
&= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\theta}) - \sum_{i=1}^{L} \int q(\mathbf{z}_{i+1}|\mathbf{x}) q(\mathbf{z}_i|\mathbf{x}) \log \frac{q(\mathbf{z}_i|\mathbf{x})}{p(\mathbf{z}_i|\mathbf{z}_{i+1})} d\mathbf{z}_i d\mathbf{z}_{i+1} \\
&= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\theta}) - \sum_{i=1}^{L} \mathbb{E}_{q(\mathbf{z}_{i+1}|\mathbf{x})} \left[KL(q(\mathbf{z}_i|\mathbf{x}) || p(\mathbf{z}_i|\mathbf{z}_{i+1})) \right]
\end{aligned}$$

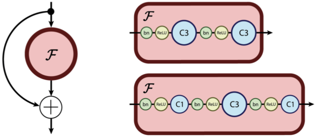
---

https://arxiv.org/pdf/1611.05013.pdf

RevNet, i-RevNet

RevNets, 2017

- Modern neural networks are trained via backpropagation.
- Residual networks are state of the art in image classification.
- Backpropagation requires storing the network activations.



Problem

Storing the activations imposes an increasing memory burden. GPUs have limited memory capacity, leading to constraints often exceeded by state-of-the-art architectures (with thousand layers).

Gomez A. N. et al. The Reversible Residual Network: Backpropagation Without Storing Activations, 2017

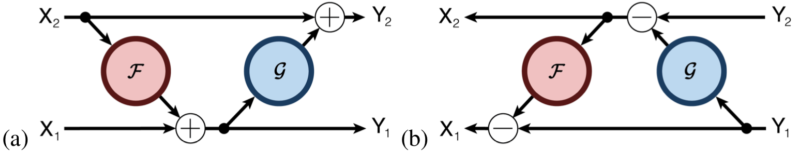
RevNets, 2017

NICE

$$\begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = \mathbf{x}_2 + \mathcal{F}(\mathbf{x}_1, \boldsymbol{\theta}); \end{cases} \quad \Leftrightarrow \quad \begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 - \mathcal{F}(\mathbf{z}_1, \boldsymbol{\theta}). \end{cases}$$

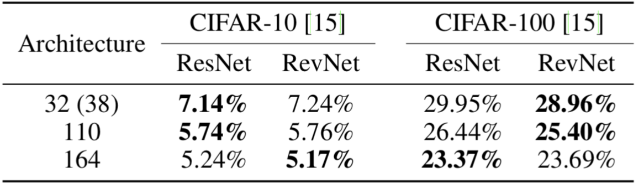
RevNet

$$\begin{cases} \mathbf{y}_1 = \mathbf{x}_1 + \mathcal{F}(\mathbf{x}_2, \boldsymbol{\theta}); \\ \mathbf{y}_2 = \mathbf{x}_2 + \mathcal{G}(\mathbf{y}_1, \boldsymbol{\theta}); \end{cases} \quad \Leftrightarrow \quad \begin{cases} \mathbf{x}_2 = \mathbf{y}_2 - \mathcal{F}(\mathbf{y}_1, \boldsymbol{\theta}); \\ \mathbf{x}_1 = \mathbf{y}_1 - \mathcal{G}(\mathbf{x}_2, \boldsymbol{\theta}). \end{cases}$$



Gomez A. N. et al. The Reversible Residual Network: Backpropagation Without Storing Activations, 2017

RevNets, 2017



Architecture	CIFAR-10 [15]		CIFAR-100 [15]	
	ResNet	RevNet	ResNet	RevNet
32 (38)	**7.14%**	7.24%	29.95%	**28.96%**
110	**5.74%**	5.76%	26.44%	**25.40%**
164	5.24%	**5.17%**	**23.37%**	23.69%

- If the network contains non-reversible blocks (poolings, strides), activations for these blocks should be stored.
- To avoid storing activations in the modern frameworks, the backward pass should be manually redefined.

Gomez A. N. et al. The Reversible Residual Network: Backpropagation Without Storing Activations, 2017

i-RevNet, 2018

Hypothesis

The success of deep convolutional networks is based on progressively discarding uninformative variability about the input with respect to the problem at hand.

- It is difficult to recover images from their hidden representations.
- Information bottleneck principle: an optimal representation must reduce the MI between an input and its representation to reduce uninformative variability + maximize the MI between the output and its representation to preserve each class from collapsing onto other classes.

Jacobsen J. H., Smeulders A., Oyallon E. i-RevNet: Deep Invertible Networks, 2018

i-RevNet, 2018

Hypothesis

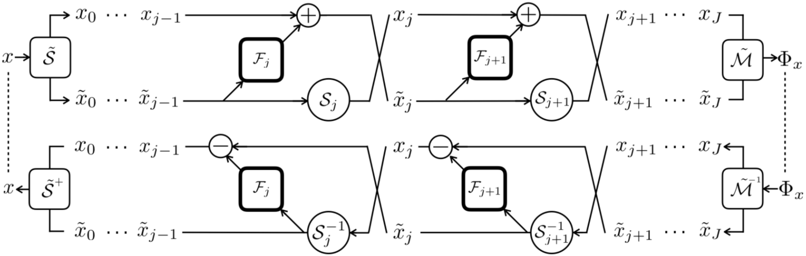
The success of deep convolutional networks is based on progressively discarding uninformative variability about the input with respect to the problem at hand.

Idea

Build a cascade of homeomorphic layers (i-RevNet), a network that can be fully inverted up to the final projection onto the classes, i.e. no information is discarded.

Jacobsen J. H., Smeulders A., Oyallon E. i-RevNet: Deep Invertible Networks, 2018

i-RevNet, 2018



Architecture	**Injective**	**Bijective**	**Top-1 error**	**Parameters**
ResNet	-	-	24.7	26M
RevNet	-	-	25.2	28M
i-RevNet (a)	yes	-	24.7	181M
i-RevNet (b)	yes	yes	26.7	29M

Jacobsen J. H., Smeulders A., Oyallon E. i-RevNet: Deep Invertible Networks, 2018

Posterior collapse, toy example

Posterior collapse: toy example

Let define latent variable model in the following way:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}) d\mathbf{z}$$

- prior distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$;
- probabilistic model $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\sigma}_{\boldsymbol{\theta}}(\mathbf{z}))$ (diagonal covariance);
- variational posterior $q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}_{\phi}(\mathbf{x}))$ (diagonal covariance).

Let data distribution is $\pi(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Possible cases:

- covariance matrix $\boldsymbol{\Sigma}$ is diagonal (univariate case);
- covariance matrix $\boldsymbol{\Sigma}$ is **not** diagonal (multivariate case).

What is the difference?

ELBO surgery, 2016

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int q(\mathbf{Z}|\mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z}|\mathbf{X})} d\mathbf{Z}.$$

ELBO interpretations

- Evidence minus posterior KL

$$\mathcal{L}(q, \boldsymbol{\theta}) = \log p(\mathbf{X}|\boldsymbol{\theta}) - KL(q(\mathbf{Z}|\mathbf{X})||p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})).$$

- Average negative energy plus entropy

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X})} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) + \mathbb{H}\left[q(\mathbf{Z}|\mathbf{X})\right].$$

- Average term-by-term reconstruction minus KL to prior

$$\mathcal{L}(q, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left[\mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i)} \log p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta}) - KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i))\right].$$

---

http://approximateinference.org/accepted/HoffmanJohnson2016.pdf

ELBO surgery, 2016

$$\mathcal{L}(q, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left[\mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i)} \log p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta}) - KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) \right].$$

Theorem

$$\frac{1}{n} \sum_{i=1}^{n} KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) = KL(q(\mathbf{z})||p(\mathbf{z})) + \mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}],$$

where i is treated as random variable:

$$q(i, \mathbf{z}) = q(i)q(\mathbf{z}|i); \quad p(i, \mathbf{z}) = p(i)p(\mathbf{z}); \quad q(i) = p(i) = \frac{1}{n}; \quad q(\mathbf{z}|i) = q(\mathbf{z}|\mathbf{x}_i).$$

$$q(\mathbf{z}) = \sum_{i=1}^{n} q(i, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} q(\mathbf{z}|\mathbf{x}_i); \quad \mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}] = \mathbb{E}_{q(i,\mathbf{z})} \log \frac{q(i, \mathbf{z})}{q(i)q(\mathbf{z})}.$$

---

http://approximateinference.org/accepted/HoffmanJohnson2016.pdf

ELBO surgery, 2016

Theorem

$$\frac{1}{n}\sum_{i=1}^{n} KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) = KL(q(\mathbf{z})||p(\mathbf{z})) + \mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}].$$

Proof

$$\frac{1}{n}\sum_{i=1}^{n} KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) = \sum_{i=1}^{n} \int q(i)q(\mathbf{z}|i)\log\frac{q(\mathbf{z}|i)}{p(\mathbf{z})}d\mathbf{z} =$$

$$= \sum_{i=1}^{n} \int q(i,\mathbf{z})\log\frac{q(i,\mathbf{z})}{p(\mathbf{z})p(i)}d\mathbf{z} = \int \sum_{i=1}^{n} q(i,\mathbf{z})\log\frac{q(\mathbf{z})q(i|\mathbf{z})}{p(\mathbf{z})p(i)}d\mathbf{z} =$$

$$= \int q(\mathbf{z})\log\frac{q(\mathbf{z})}{p(\mathbf{z})}d\mathbf{z} + \int \sum_{i=1}^{n} q(i|\mathbf{z})q(\mathbf{z})\log\frac{q(i|\mathbf{z})}{p(i)}d\mathbf{z} =$$

$$= KL(q(\mathbf{z})||p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z})}\mathbb{H}\left[q(i|\mathbf{z})\right] + \log n.$$

---

http://approximateinference.org/accepted/HoffmanJohnson2016.pdf

ELBO surgery, 2016

Theorem

$$\frac{1}{n}\sum_{i=1}^{n} KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) = KL(q(\mathbf{z})||p(\mathbf{z})) + \mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}].$$

Proof (continued)

$$\frac{1}{n}\sum_{i=1}^{n} KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) = KL(q(\mathbf{z})||p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z})}\mathbb{H}\left[q(i|\mathbf{z})\right] + \log n$$

$$\mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}] = \mathbb{E}_{q(i,\mathbf{z})} \log \frac{q(i,\mathbf{z})}{q(i)q(\mathbf{z})} = \mathbb{E}_{q(\mathbf{z})}\mathbb{E}_{q(i|\mathbf{z})} \log \frac{q(i|\mathbf{z})q(\mathbf{z})}{q(i)q(\mathbf{z})} =$$

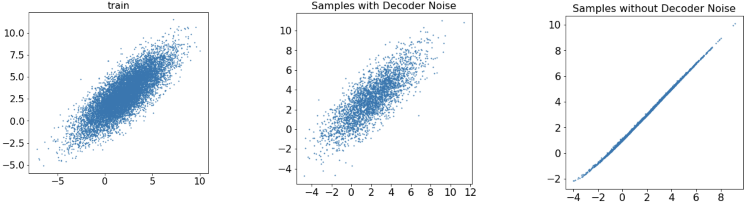
$$= \mathbb{E}_{q(\mathbf{z})}\mathbb{E}_{q(i|\mathbf{z})} \log \frac{q(i|\mathbf{z})}{q(i)} = -\mathbb{E}_{q(\mathbf{z})}\mathbb{H}\left[q(i|\mathbf{z})\right] + \log n.$$

---

http://approximateinference.org/accepted/HoffmanJohnson2016.pdf

Posterior collapse: toy example

Multivariate ($\boldsymbol{\Sigma}$ is non-diagonal)

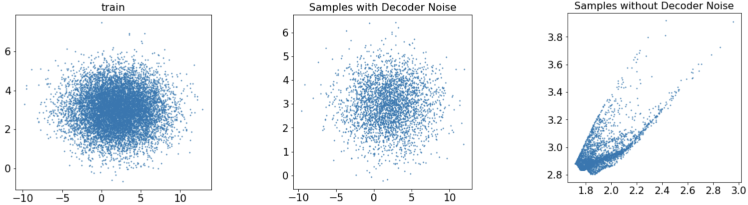

train

Samples with Decoder Noise

Samples without Decoder Noise

The encoder uses latent variables to model data.

Univariate ($\boldsymbol{\Sigma}$ is diagonal)


train

Samples with Decoder Noise

Samples without Decoder Noise

Latent variables are not used, since the decoder could model the data without the encoder.

Posterior collapse, VLAE

Variational Lossy AutoEncoder

Lossy code via explicit information placement

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \prod_{i=1}^{m} p(x_i|\mathbf{z}, \mathbf{x}_{\text{WindowAround}(i)}, \boldsymbol{\theta}).$$

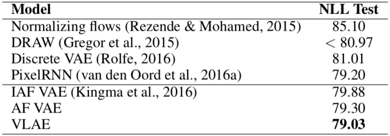
- WindowAround(i) restricts the receptive field (it forbids to represent arbitrarily complex distribution over **x** without dependence on **z**).
- Local statistics of 2D images (texture) will be modeled by a small local window.
- Global structural information (shapes) is long-range dependency that can only be communicated through latent code **z**.

Chen X. et al. Variational Lossy Autoencoder, 2016

Variational Lossy AutoEncoder

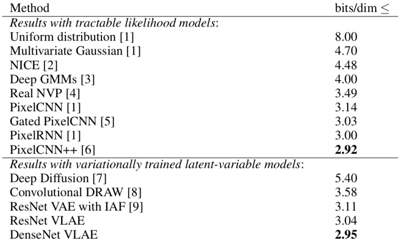
- Can VLAE learn lossy codes that encode global statistics?
- Does using AF priors improves upon using IAF posteriors as predicted by theory?
- Does using autoregressive decoding distributions improve density estimation performance?

MNIST



Model	NLL Test
Normalizing flows (Rezende & Mohamed, 2015)	85.10
DRAW (Gregor et al., 2015)	< 80.97
Discrete VAE (Rolfe, 2016)	81.01
PixelRNN (van den Oord et al., 2016a)	79.20
IAF VAE (Kingma et al., 2016)	79.88
AF VAE	79.30
VLAE	**79.03**

CIFAR10



Method	bits/dim $\leq$
Results with tractable likelihood models:	
Uniform distribution [1]	8.00
Multivariate Gaussian [1]	4.70
NICE [2]	4.48
Deep GMMs [3]	4.00
Real NVP [4]	3.49
PixelCNN [1]	3.14
Gated PixelCNN [5]	3.03
PixelRNN [1]	3.00
PixelCNN++ [6]	**2.92**
Results with variationally trained latent-variable models:	
Deep Diffusion [7]	5.40
Convolutional DRAW [8]	3.58
ResNet VAE with IAF [9]	3.11
ResNet VLAE	3.04
DenseNet VLAE	**2.95**

Chen X. et al. Variational Lossy Autoencoder, 2016

Disentanglement, InfoGAN

InfoGAN

GAN objective

$$\min_G \max_D V(G, D)$$

$$V(G, D) = \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

Latent vector $\mathbf{z}$ is not imposed to be disentangled.
InfoGAN decomposes input vector:

- $\mathbf{z}$ – incompressible noise;
- $\mathbf{c}$ – structured latent code $p(\mathbf{c}) = \prod_{j=1}^{d} p(c_j)$.

Information-theoretic regularization

$$\max I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$

Information in the latent code $\mathbf{c}$ should not be lost in the generation process.

Chen X. et al. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016

InfoGAN

Objective

$$\min_G \max_D V(G, D) - \lambda I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$

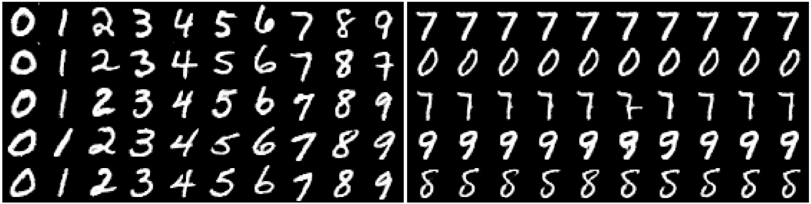
Variational Information Maximization

$$\begin{aligned} I(\mathbf{c}, G(\mathbf{z}, \mathbf{c})) &= H(\mathbf{c}) - H(\mathbf{c}|G(\mathbf{z}, \mathbf{c})) = \\ &= H(\mathbf{c}) + \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\left[\mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c}|\mathbf{x})} \log p(\mathbf{c}'|\mathbf{x})\right] = \\ &= H(\mathbf{c}) + \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})} KL(p(\mathbf{c}'|\mathbf{x})||q(\mathbf{z}'|\mathbf{x})) + \\ &+ \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c}|\mathbf{x})} \log q(\mathbf{c}'|\mathbf{x}) \geq \\ &\geq H(\mathbf{c}) + \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c}|\mathbf{x})} \log q(\mathbf{c}'|\mathbf{x}) = \\ &\qquad H(\mathbf{c}) + \mathbb{E}_{\mathbf{c}\sim p(\mathbf{c})}\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})} \log q(\mathbf{c}|\mathbf{x}) \end{aligned}$$

Chen X. et al. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016

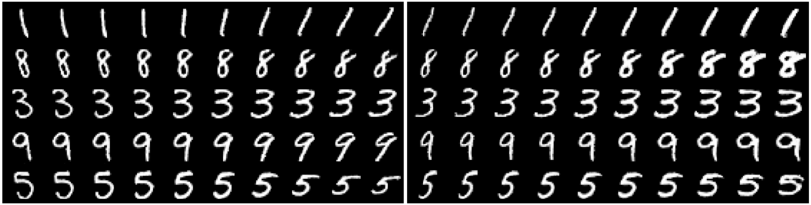
InfoGAN

Latent codes on MNIST



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



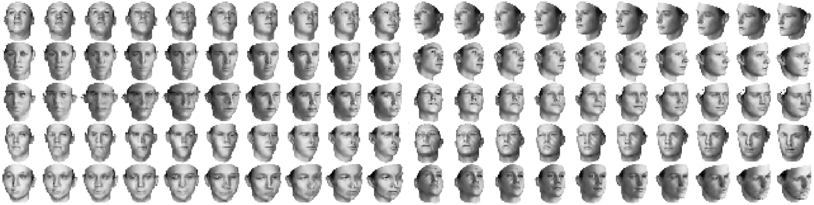
(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Chen X. et al. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016

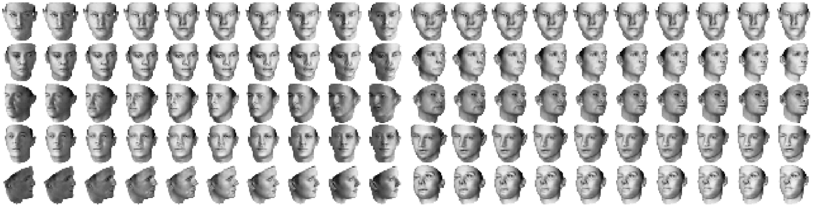
InfoGAN

Latent codes on 3D Faces



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow

Chen X. et al. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016

β-VAE, Disentanglement metric

β-VAE

Disentangling metric

1. Generate two sets of objects

$$\mathbf{x}_{li} \sim \mathrm{Sim}(\mathbf{v}_{li}, \mathbf{w}_{li}); \quad \mathbf{x}_{lj} \sim \mathrm{Sim}(\mathbf{v}_{lj}, \mathbf{w}_{lj}); \quad y_{ij} \sim U[1, d].$$

$$\mathbf{v}_{li} \sim p(\mathbf{v}); \quad \mathbf{v}_{lj} \sim p(\mathbf{v})\left([v_{li}]_y = [v_{lj}]_y\right); \quad \mathbf{w}_{li}, \mathbf{w}_{lj} \sim p(\mathbf{w}).$$

2. Find representations

$$q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mu(\mathbf{x})|\sigma^2(\mathbf{x})\right); \quad \mathbf{z}_{li} = \mu(\mathbf{x}_{li}); \quad \mathbf{z}_{lj} = \mu(\mathbf{x}_{lj}).$$

3. Use accuracy of classifier $p(y|\mathbf{z}_{\text{diff}})$ with a low VC-dimension as metric of disentanglement

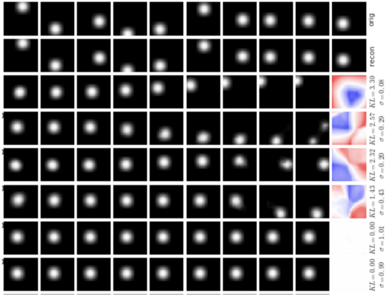
$$\mathbf{z}_{\text{diff}} = \frac{1}{L} \sum_{l=1}^{L} |\mathbf{z}_{li} - \mathbf{z}_{lj}|.$$

Higgins I. et al. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, 2017

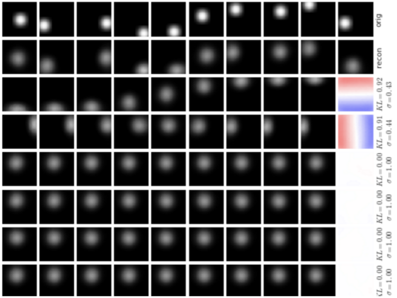
β-VAE

- **Top row:** original images.
- **Second row:** the corresponding reconstructions.
- **Remaining rows:** latent traversals ordered by KL divergence with the prior.
- **Heatmaps:** latent activations for each 2D position.

β = 1



β = 150

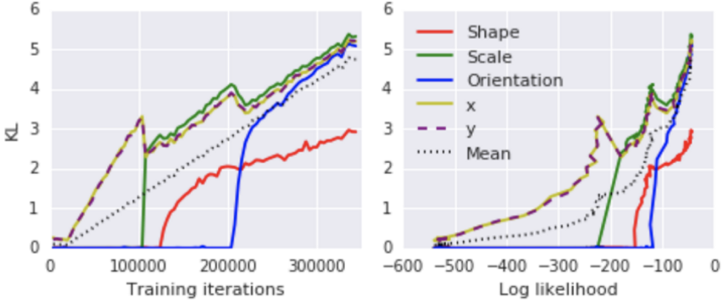


Burgess C. P. et al. Understanding disentangling in β-VAE, 2018

β-VAE

Controlled encoding capacity

$$\mathcal{L}(q, \boldsymbol{\theta}, \beta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - |KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - C|.$$



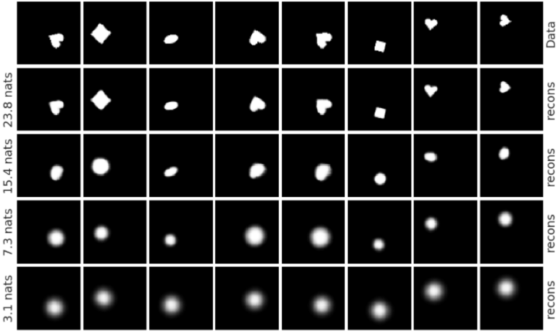
The early capacity is allocated to positional latents only, followed by a scale latent, then shape and orientation latents.

Burgess C. P. et al. Understanding disentangling in β-VAE, 2018

β-VAE

Controlled encoding capacity

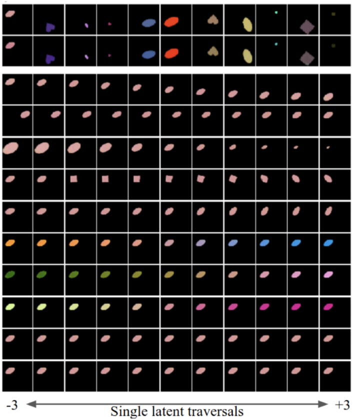
$$\mathcal{L}(q, \boldsymbol{\theta}, \beta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - |KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - C|.$$



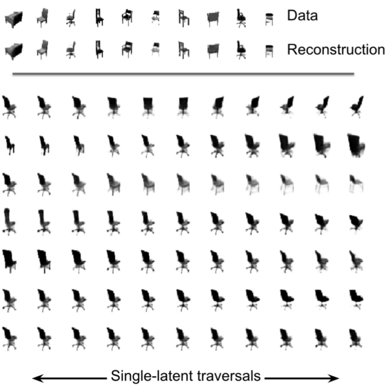
Burgess C. P. et al. Understanding disentangling in β-VAE, 2018

β-VAE

(a) Coloured dSprites



(b) 3D Chairs



Burgess C. P. et al. Understanding disentangling in β-VAE, 2018

FactorVAE

FactorVAE

Disentangled aggregated variational posterior

$$q(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} q(\mathbf{z}|\mathbf{x}) = \prod_{j=1}^{d} q(z_j)$$

Total correlation regularizer

$$\min KL(q(\mathbf{z}) || \prod_{j=1}^{d} q(z_j))$$

FactorVAE objective

$$\min_{\boldsymbol{\phi}, \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) - \gamma \cdot KL(q(\mathbf{z}) || \prod_{j=1}^{d} q(z_j))$$

- The last term is intractable.
- FactorVAE uses density ratio trick for estimation.

Kim H., Mnih A. Disentangling by Factorising, 2018

FactorVAE

Consider two distributions $q_1(\mathbf{x})$, $q_2(\mathbf{x})$ and probilistic model

$$p(\mathbf{x}|y) = \begin{cases} q_1(\mathbf{x}), & \text{if } y = 1, \\ q_2(\mathbf{x}), & \text{if } y = 0, \end{cases} \qquad y \sim \text{Bern}(0.5).$$

Density ratio trick

$$\frac{q_1(\mathbf{x})}{q_2(\mathbf{x})} = \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})}{p(y=1)} \Bigg/ \frac{p(y=0|\mathbf{x})p(\mathbf{x})}{p(y=0)} =$$

$$= \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{1 - p(y=1|\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}$$

Here $D(\mathbf{x})$ could be treated as a discriminator a model the output of which is a probability that $\mathbf{x}$ is a sample from $q_1(\mathbf{x})$ rather than from $q_2(\mathbf{x})$.

Kim H., Mnih A. Disentangling by Factorising, 2018

FactorVAE

FactorVAE objective

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathrm{ELBO}(\boldsymbol{\theta}, \phi) - \gamma \cdot KL(q(\mathbf{z}) || \prod_{j=1}^{d} q(z_j))$$

Total correlation regularizer

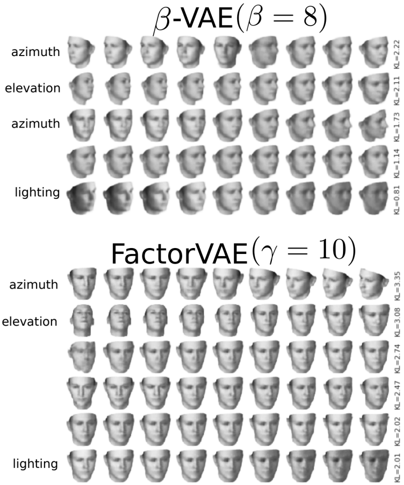
$$KL(q(\mathbf{z}) || \prod_{j=1}^{d} q(z_j)) = KL(q(\mathbf{z}) || \bar{q}(\mathbf{z})) =$$

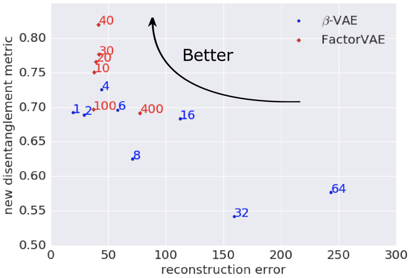
$$= \mathbb{E}_{q(\mathbf{z})} \log \frac{q(\mathbf{z})}{\bar{q}(\mathbf{z})} \approx \mathbb{E}_{q(\mathbf{z})} \log \frac{D(\mathbf{z})}{1 - D(\mathbf{z})}$$

VAE and GAN are trained simultaneously.

Kim H., Mnih A. Disentangling by Factorising, 2018

FactorVAE





Kim H., Mnih A. Disentangling by Factorising, 2018

Challenging Disentanglement Assumptions

Challenging Disentanglement Assumptions

Proof (1)

1. Consider the function $g : \mathrm{supp}(\mathbf{z}) \to [0,1]^d$:

$$g_i(\mathbf{u}) = P(z_i \leq u_i), \quad i = 1, \dots, d.$$

   - g is bijective (since $p(\mathbf{z}) = \prod_{i=1}^{d} p(z_i)$).
   - $\frac{\partial g_i(\mathbf{u})}{\partial u_i} \neq 0$, for all i and $\frac{\partial g_i(\mathbf{u})}{\partial u_j} = 0$ for all $i \neq j$.
   - $g(\mathbf{z})$ is an independent d-dimensional uniform distribution.

2. Consider $h : (0,1]^d \to \mathbb{R}^d$

$$h_i(\mathbf{u}) = \psi^{-1}(u_i), \quad i = 1, \dots, d.$$

Here ψ denotes the CDF of a standard normal distribution.

   - h is bijective.
   - $\frac{\partial h_i(\mathbf{u})}{\partial u_i} \neq 0$, for all i and $\frac{\partial h_i(\mathbf{u})}{\partial u_j} = 0$ for all $i \neq j$.
   - $h(g(\mathbf{z}))$ is a d-dimensional standard normal distribution.

Locatello F. et al. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations, 2018

Challenging Disentanglement Assumptions

Proof (2)

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ be an arbitrary orthogonal matrix with $A_{ij} \neq 0$ for all i, j. The family of such matrices is infinite.

- $\mathbf{A}$ is orthogonal, it is invertible and thus defines a bijective linear operator.
- $\mathbf{A}h(g(\mathbf{z})) \in \mathbb{R}^d$ is hence an independent, multivariate standard normal distribution.
- $h^{-1}(\mathbf{A}h(g(\mathbf{z}))) \in \mathbb{R}^d$ is an independent d-dimensional uniform distribution.

Define $f : \text{supp}(\mathbf{z}) \to \text{supp}(\mathbf{z})$:

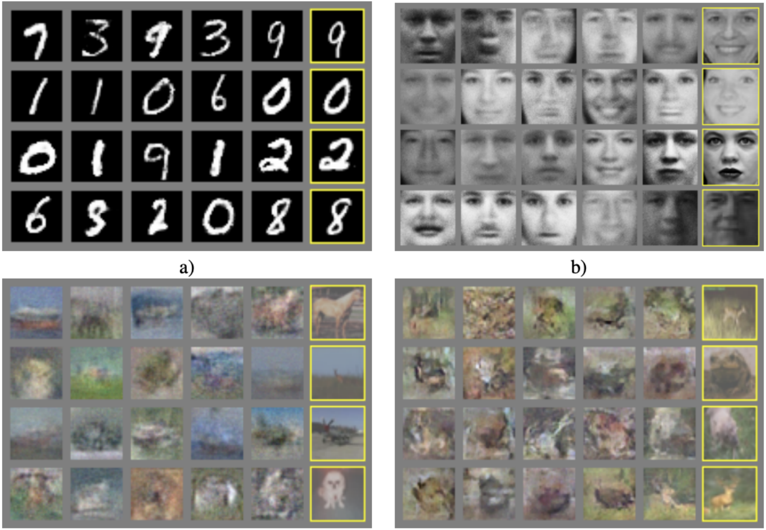
$$f(\mathbf{u}) = g^{-1}(h^{-1}(\mathbf{A}h(g(\mathbf{z})))).$$

By definition $f(\mathbf{z})$ has the same marginal distribution as $\mathbf{z}$: $P(\mathbf{z} \leq \mathbf{u}) = P(f(\mathbf{z}) \leq \mathbf{u})$ and $\frac{\partial f_i(\mathbf{z})}{\partial z_j} \neq 0$.

Locatello F. et al. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations, 2018

DCGAN

Vanilla GAN results

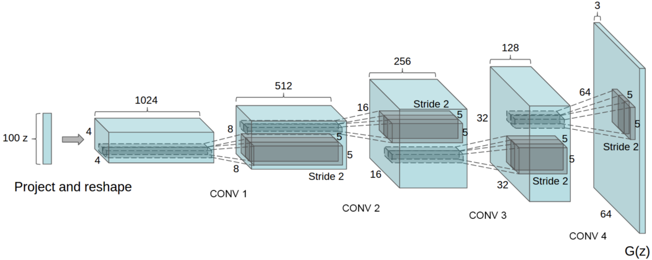


a) b) c) d)

Goodfellow I. J. et al. Generative Adversarial Networks, 2014

Deep Convolutional GAN

Architecture



- Mean-pooling instead of max-pooling.
- Transposed convolutions in the generator for upsampling.
- Downsample with strided convolutions and average pooling.
- ReLU for generator, Leaky-ReLU (0.2) for discriminator.
- Output nonlinearity: tanh for Generator, sigmoid for discriminator.
- Batch Normalization used to prevent mode collapse (not applied at the output of G and input of D).
- Adam: small LR = 2e-4; small momentum: 0.5, batch-size: 128.

Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015

Deep Convolutional GAN

ImageNet samples

Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015

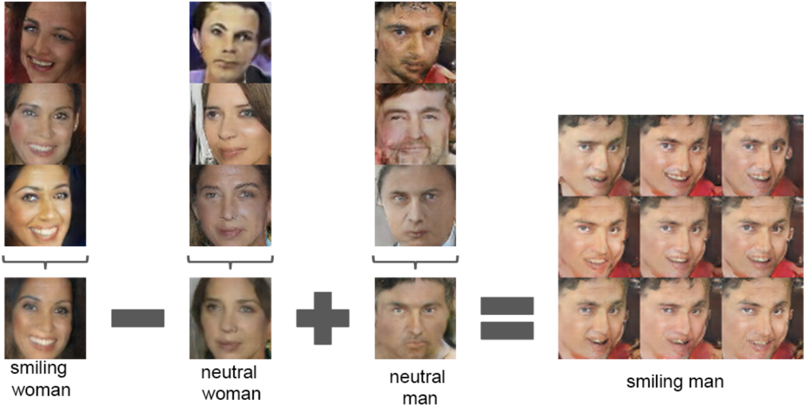
Deep Convolutional GAN

Smooth interpolations

Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015

Deep Convolutional GAN

Vector arithmetic



Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015

Improved techniques for training GANs

Improved techniques for training GANs

- Feature matching

$$\mathcal{L}_G = \|\mathbb{E}_{\pi(\mathbf{x})}\mathbf{d}(\mathbf{x}) - \mathbb{E}_{p(\mathbf{z})}\mathbf{d}(G(\mathbf{z}))\|_2^2$$

  Here $\mathbf{d}(\mathbf{x})$ – intermediate layer of discriminator. Matching the learned discriminator statistics instead of the output of the discriminator. Helps to avoid the vanishing gradients for sufficiently good discriminator.

- Historical averaging adds extra loss term for generator and discriminator losses

$$\|\boldsymbol{\theta} - \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\theta}_t\|_2^2.$$

  Here $\boldsymbol{\theta}_t$ – value of parameters at the previous step t. It allows to stabilize training procedure.

Salimans T. et al. Improved Techniques for Training GANs, 2016

Improved techniques for training GANs

- One-sided label smoothing. Instead of using one-hot labels in classification, use $(1 - \alpha)$ for real data (the generated samples are not smoothed).

$$D^*(\mathbf{x}) = \frac{(1 - \alpha)\pi(\mathbf{x})}{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}$$

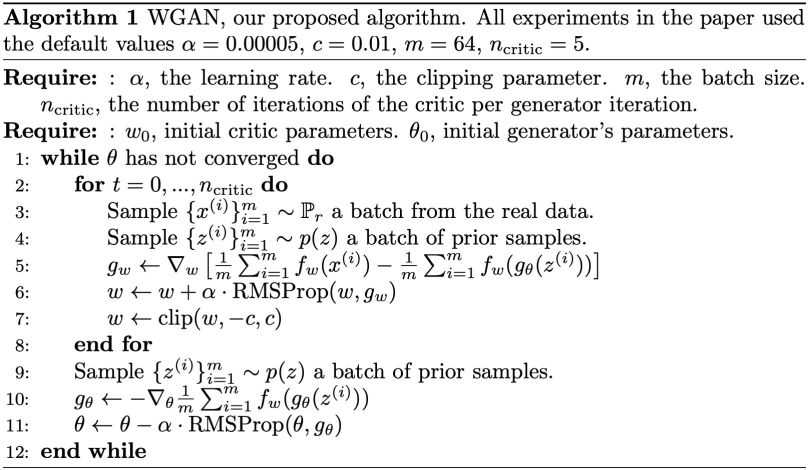
- Virtual batch normalization. BatchNorm makes samples within minibatch are highly correlated.

Use reference fixed batch to compute the normalization statistics. To avoid overfitting construct batch with the reference batch and the current sample.

Salimans T. et al. Improved Techniques for Training GANs, 2016

WGAN

Wasserstein GAN



Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

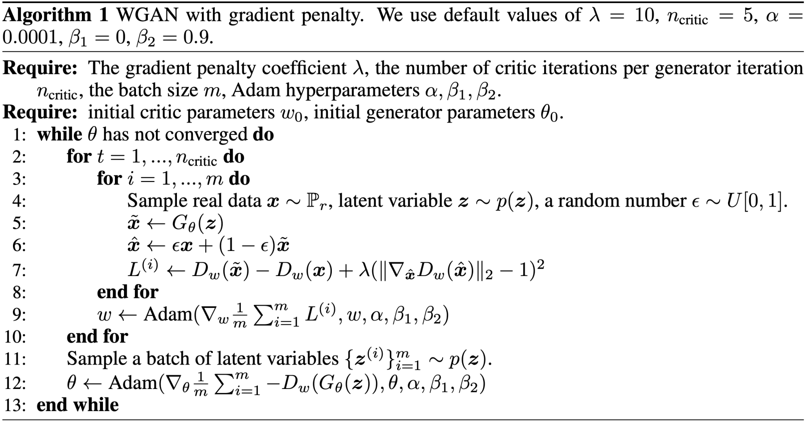
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

```
while $\theta$ has not converged do
    for $t = 0, ..., n_{\text{critic}}$ do
        Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while
```

Arjovsky M., Chintala S., Bottou L. Wasserstein GAN, 2017

Wasserstein GAN with Gradient Penalty



Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
while $\theta$ has not converged do
    for $t = 1, ..., n_{\text{critic}}$ do
        for $i = 1, ..., m$ do
            Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
            $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
            $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
            $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
        end for
        $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while
```

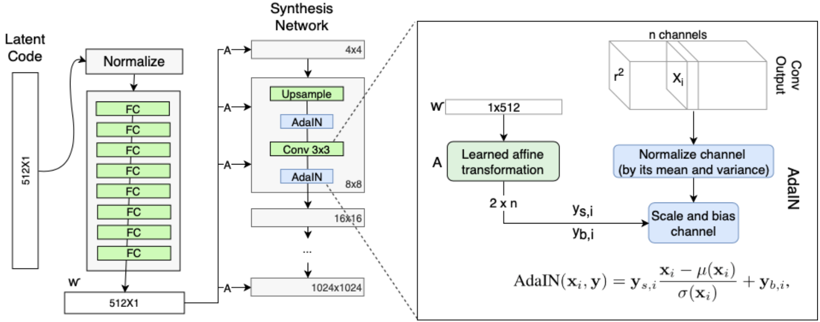
Gulrajani I. et al. Improved Training of Wasserstein GANs, 2017

StyleGAN

StyleGAN

Step 2: Style modulation

- Adaptive Instance Normalization transfers the **w** vector to the synthesis Network.
- The module is added to each resolution to define the visual expression of the features.



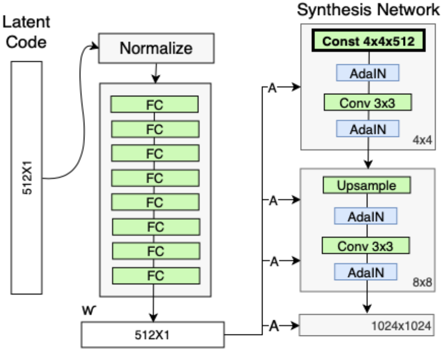
$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i}\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

StyleGAN

Step 3: Remove traditional input

Mapping network provides stochasticity to different stages of the synthesis network. Input of the synthesis network is a trainable vector.

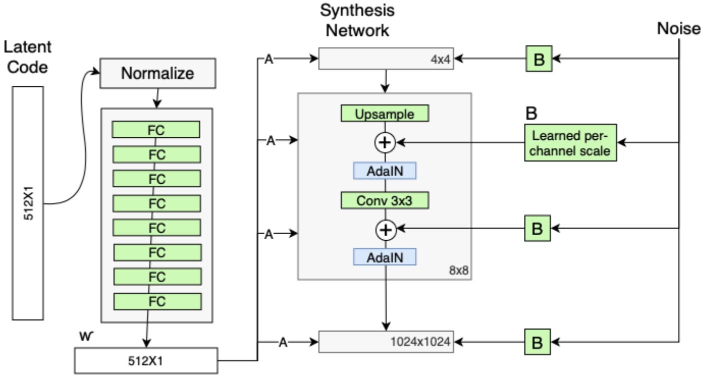


Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

StyleGAN

Step 4: Stochastic variation

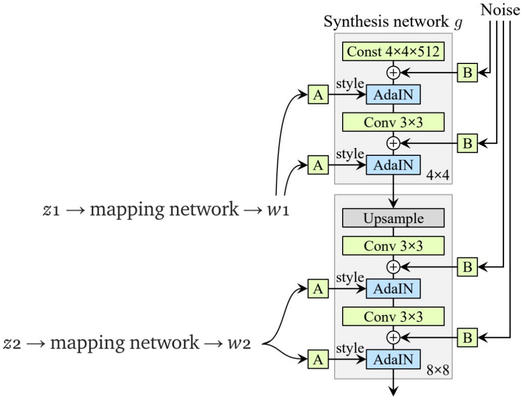
Inject random noise to add small aspects, such as freckles, exact placement of hairs, wrinkles, features which make the image more realistic and increase the variety of outputs.



Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

StyleGAN

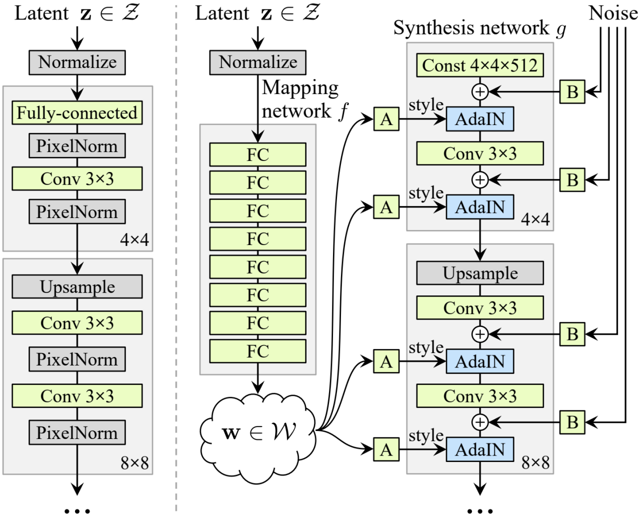
Step 4: Style Mixing



- Makes different levels of synthesis network to be independent.
- Allows to couple diffirent styles.

Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

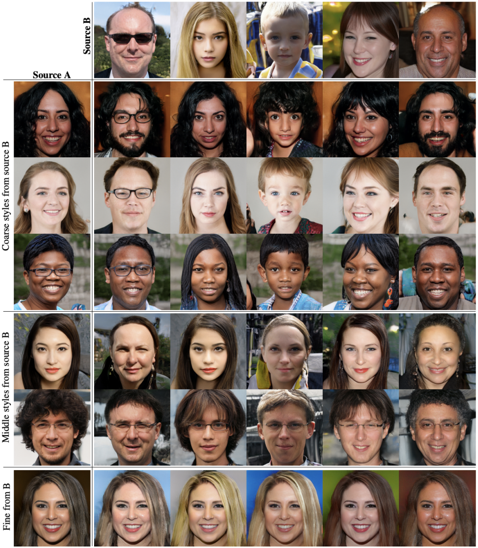
StyleGAN



(a) Traditional (b) Style-based generator

Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

StyleGAN

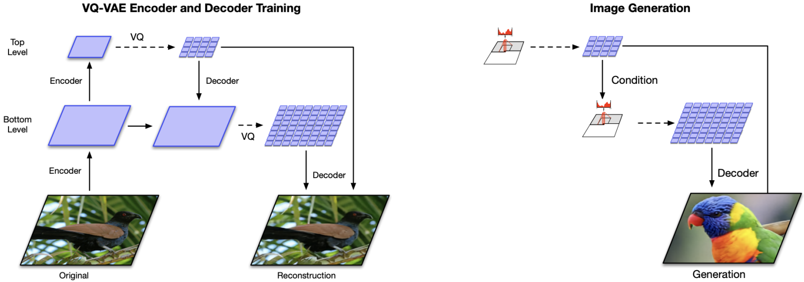


Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

Vector Quantized VAE-2

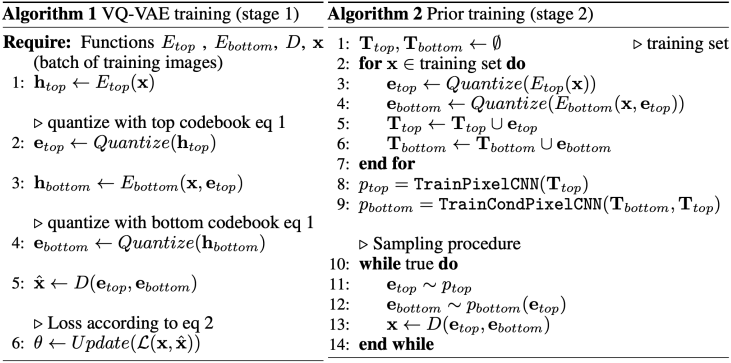
Vector Quantized VAE-2

- Use multi-scale hierarchical model.
- Use autoregressive prior model in each scale of the hierarchy.
- Improve autoregressive prior (PixelSNAIL with self-attention in bottom layer, PixelCNN++ in bottom layer).
- Train the encoder and decoder at the first stage, train the priors at the second stage.



Razavi A., Oord A., Vinyals O. Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019

Vector Quantized VAE-2



Algorithm 1 VQ-VAE training (stage 1)

```
Require: Functions $E_{top}$ , $E_{bottom}$, $D$, $\mathbf{x}$ (batch of training images)
$\mathbf{h}_{top} \leftarrow E_{top}(\mathbf{x})$
    ▷ quantize with top codebook eq 1
$\mathbf{e}_{top} \leftarrow Quantize(\mathbf{h}_{top})$
$\mathbf{h}_{bottom} \leftarrow E_{bottom}(\mathbf{x}, \mathbf{e}_{top})$
    ▷ quantize with bottom codebook eq 1
$\mathbf{e}_{bottom} \leftarrow Quantize(\mathbf{h}_{bottom})$
$\hat{\mathbf{x}} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$
    ▷ Loss according to eq 2
$\theta \leftarrow Update(\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}))$
```

Algorithm 2 Prior training (stage 2)

```
$\mathbf{T}_{top}, \mathbf{T}_{bottom} \leftarrow \emptyset$    ▷ training set
for $\mathbf{x} \in$ training set do
    $\mathbf{e}_{top} \leftarrow Quantize(E_{top}(\mathbf{x}))$
    $\mathbf{e}_{bottom} \leftarrow Quantize(E_{bottom}(\mathbf{x}, \mathbf{e}_{top}))$
    $\mathbf{T}_{top} \leftarrow \mathbf{T}_{top} \cup \mathbf{e}_{top}$
    $\mathbf{T}_{bottom} \leftarrow \mathbf{T}_{bottom} \cup \mathbf{e}_{bottom}$
end for
$p_{top} = \texttt{TrainPixelCNN}(\mathbf{T}_{top})$
$p_{bottom} = \texttt{TrainCondPixelCNN}(\mathbf{T}_{bottom}, \mathbf{T}_{top})$

    ▷ Sampling procedure
while true do
    $\mathbf{e}_{top} \sim p_{top}$
    $\mathbf{e}_{bottom} \sim p_{bottom}(\mathbf{e}_{top})$
    $\mathbf{x} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$
end while
```

h_{top} $h_{\text{top}}, h_{\text{middle}}$ $h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$ Original

---

Razavi A., Oord A., Vinyals O. Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019

Feature Quantized GAN

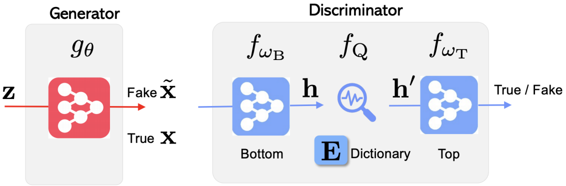
Feature Quantized GAN

- GAN tries to find Nash equilibrium, minibatch training is unstable. GAN relies heavily on the minibatch statistics.
- Lots of feature matching strategies were proposed to stabilize the training.

Feature quantized GAN discriminator

$$D(\mathbf{x}) = f_{\mathbf{w}_T} \circ f_{\mathbf{w}_B}(\mathbf{x}) \quad \Rightarrow \quad D(\mathbf{x}) = f_{\mathbf{w}_T} \circ f_Q \circ f_{\mathbf{w}_B}(\mathbf{x}).$$

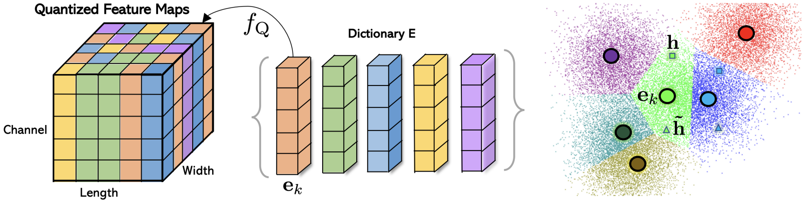
Here f_Q is a vector quantization operation.



Zhao Y. et al. Feature Quantization Improves GAN Training, 2020

Feature Quantized GAN

Quantization procedure



Quantized features

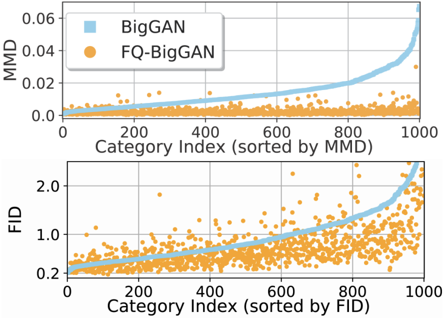
Zhao Y. et al. Feature Quantization Improves GAN Training, 2020

Feature Quantized GAN

ImageNet-1000

Models		64 × 64	128 × 128
		FID* ↓ / IS* ↑	FID* ↓ / IS* ↑
Half	TAC-GAN	-	23.75 / 28.86±0.29‡
	BigGAN	12.75 / 21.84±0.34	22.77 / 38.05±0.79‡
	FQ-BigGAN	**12.62 / 21.99±0.32**	**19.11 / 41.92±1.15**
256K	BigGAN	10.55 / 25.43±0.15	14.88 / 63.03±1.42†
	FQ-BigGAN	**9.67 / 25.96**±0.24	**13.77** / 54.36±1.07

Per class metrics for ImageNet



FFHQ

Resolution	32^2	64^2	128^2	1024^2
StyleGAN	3.28	4.82	6.33	5.24
FQ-StyleGAN	**3.01**	**4.36**	**5.98**	**4.89**

Zhao Y. et al. Feature Quantization Improves GAN Training, 2020