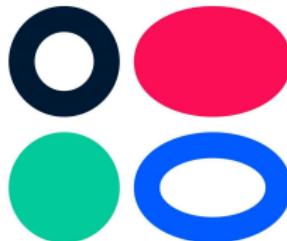


Deep Generative Models

Lecture supp

Roman Isachenko

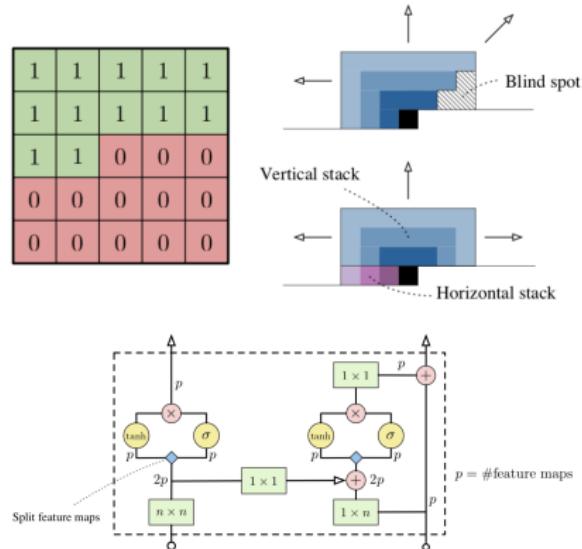


Ozon Masters

Spring, 2021

GatedPixelCNN

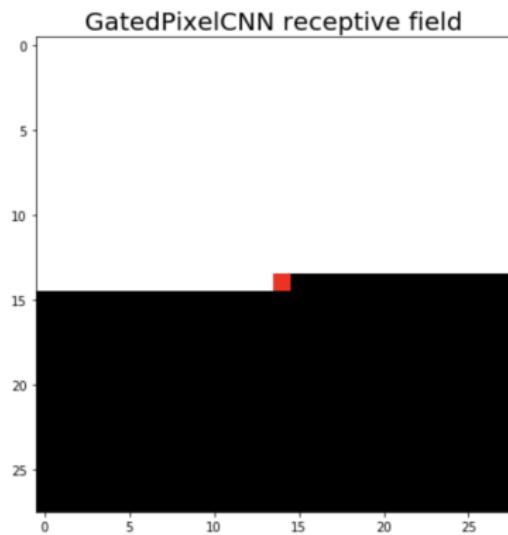
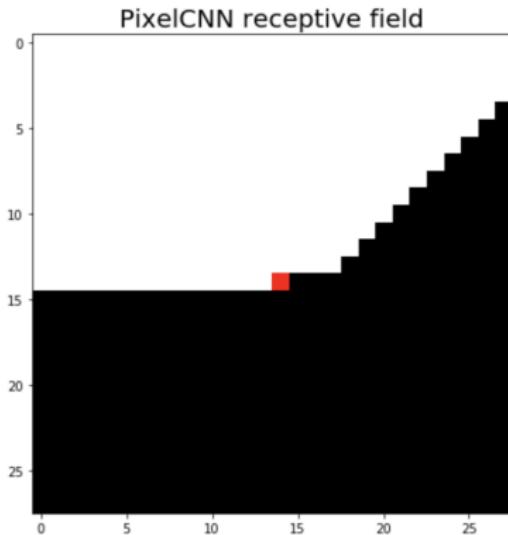
GatedPixelCNN (2016)



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, $\nabla_{\phi}\mathcal{L}(\phi, \theta)$)

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

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:

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

Log-derivative trick

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

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

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

$$\begin{aligned}\nabla_{\phi}\mathcal{L}(\phi, \theta) &= \int \nabla_{\phi}q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \theta) d\mathbf{Z} - \nabla_{\phi}KL = \\ &= \int q(\mathbf{Z}|\mathbf{X}, \phi) [\nabla_{\phi} \log q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \theta)] d\mathbf{Z} - \nabla_{\phi}KL\end{aligned}$$

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

$$\begin{aligned}\nabla_{\phi}\mathcal{L}(\phi, \theta) &\approx n \nabla_{\phi} \log q(\mathbf{z}_i^* | \mathbf{x}_i, \phi) \log p(\mathbf{x}_i | \mathbf{z}_i^*, \theta) - \nabla_{\phi}KL, \\ \mathbf{z}_i^* &\sim q(\mathbf{z}_i | \mathbf{x}_i, \phi).\end{aligned}$$

Problem

Unstable solution with huge variance.

Solution

Reparametrization trick

IWAE, active units, posterior collapse

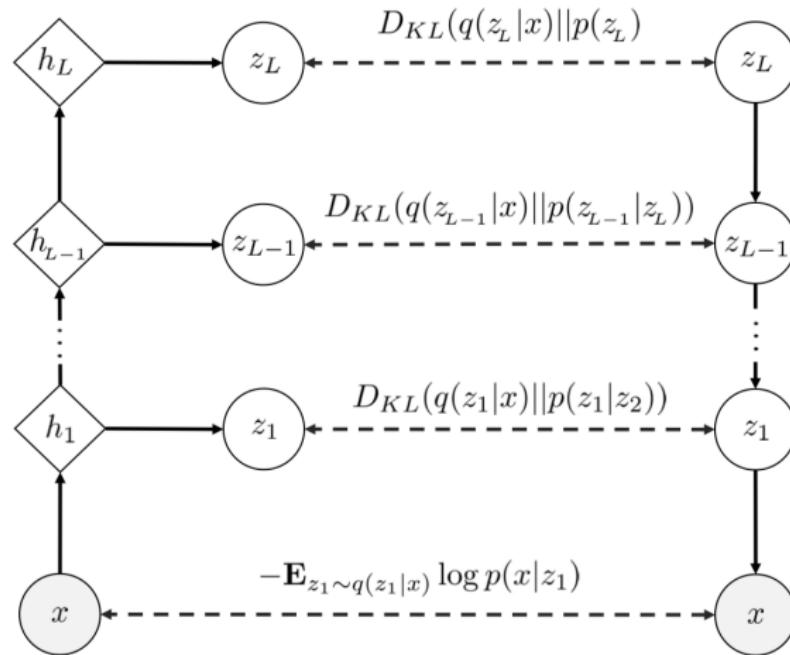
How to determine whether all VAE latent variables are informative?

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

# stoch. layers	k	MNIST				OMNIGLOT			
		VAE		IWAE		VAE		IWAE	
		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

PixelVAE, Hierarchical VAE

Hierarchical VAE



Hierarchical decomposition

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

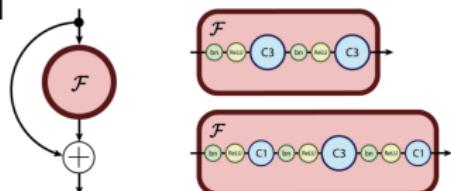
$$q(\mathbf{z}_1, \dots, \mathbf{z}_L|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x})\dots q(\mathbf{z}_L|\mathbf{x}).$$

ELBO

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

RevNet, i-RevNet

- ▶ 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.

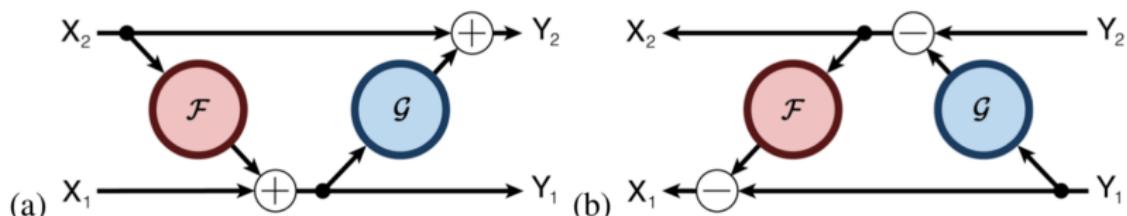
GPUs have limited memory capacity, leading to constraints often exceeded by state-of-the-art architectures (with thousand layers).

NICE

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

RevNet

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



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.

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.

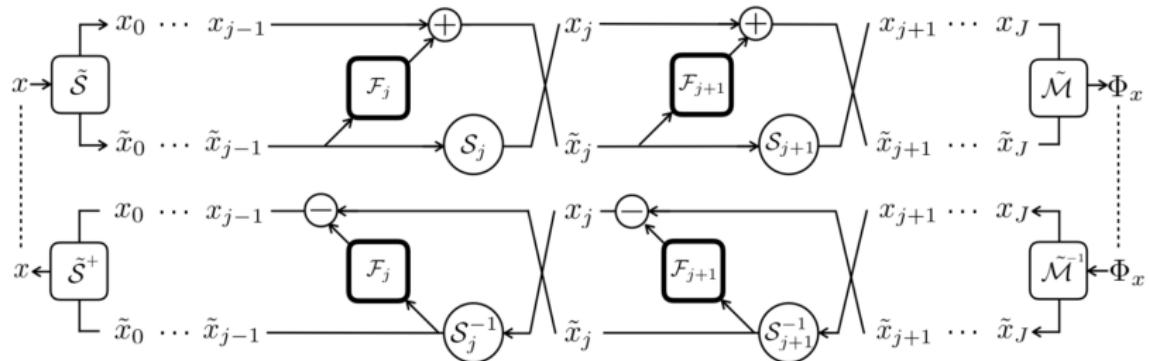
Hypothesis

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.

i-RevNet, 2018



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

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 $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x}))$ (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

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

ELBO interpretations

- ▶ Evidence minus posterior KL

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

- ▶ Average negative energy plus entropy

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

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

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

ELBO surgery, 2016

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

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:

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

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

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

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^n KL(q(\mathbf{z}_i | \mathbf{x}_i) || p(\mathbf{z}_i)) &= \sum_{i=1}^n \int q(i) q(\mathbf{z}|i) \log \frac{q(\mathbf{z}|i)}{p(\mathbf{z})} d\mathbf{z} = \\&= \sum_{i=1}^n \int q(i, \mathbf{z}) \log \frac{q(i, \mathbf{z})}{p(\mathbf{z})p(i)} d\mathbf{z} = \int \sum_{i=1}^n q(i, \mathbf{z}) \log \frac{q(\mathbf{z})q(i|\mathbf{z})}{p(\mathbf{z})p(i)} d\mathbf{z} = \\&= \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} + \int \sum_{i=1}^n q(i|\mathbf{z})q(\mathbf{z}) \log \frac{q(i|\mathbf{z})}{p(i)} d\mathbf{z} = \\&= KL(q(\mathbf{z}) || p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z})} \mathbb{H}[q(i|\mathbf{z})] + \log n.\end{aligned}$$

ELBO surgery, 2016

Theorem

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

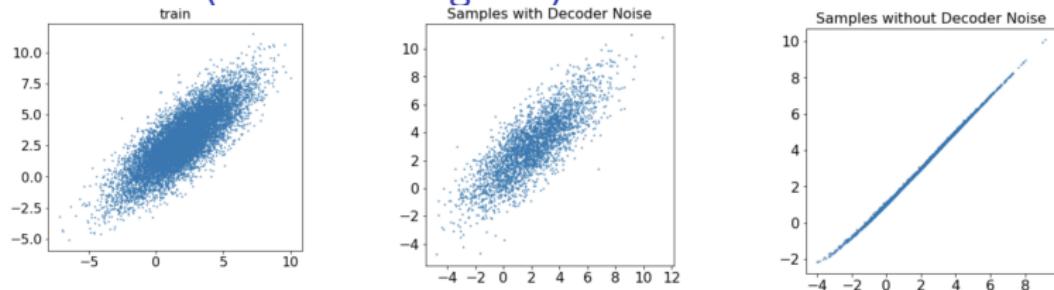
Proof (continued)

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

$$\begin{aligned}\mathbb{I}_{q(i, \mathbf{z})}[i, \mathbf{z}] &= \mathbb{E}_{q(i, \mathbf{z})} \log \frac{q(i, \mathbf{z})}{q(i)q(\mathbf{z})} = \mathbb{E}_{q(\mathbf{z})} \mathbb{E}_{q(i|\mathbf{z})} \log \frac{q(i|\mathbf{z})q(\mathbf{z})}{q(i)q(\mathbf{z})} = \\ &= \mathbb{E}_{q(\mathbf{z})} \mathbb{E}_{q(i|\mathbf{z})} \log \frac{q(i|\mathbf{z})}{q(i)} = -\mathbb{E}_{q(\mathbf{z})} \mathbb{H}[q(i|\mathbf{z})] + \log n.\end{aligned}$$

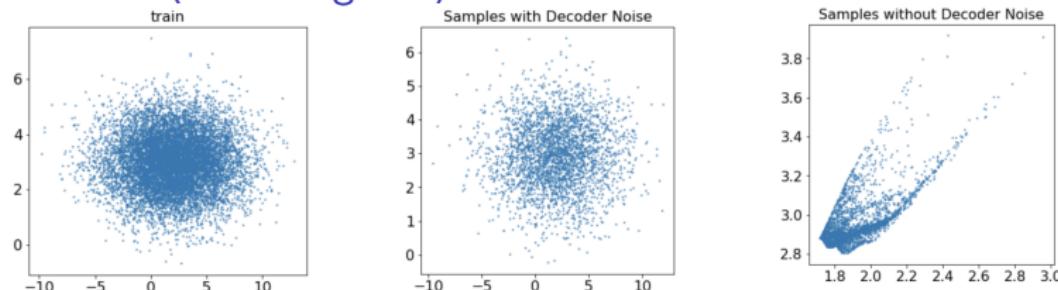
Posterior collapse: toy example

Multivariate (Σ is non-diagonal)



The encoder uses latent variables to model data.

Univariate (Σ is diagonal)



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

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

- ▶ $\text{WindowAround}(i)$ restricts the receptive field (it forbids to represent arbitrarily complex distribution over \mathbf{x} without dependence on \mathbf{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 \mathbf{z} .

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?

CIFAR10

MNIST

Model	NLL Test
Normalizing flows (Rezende & Mohamed, 2015)	85.10
DRAW (Gregor et al., 2015)	< 80.97
Discrete VAE (Rollef, 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

Method	bits/dim \leq
<i>Results with tractable likelihood models:</i>	
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
<i>Results with variationally trained latent-variable models:</i>	
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

Disentanglement, InfoGAN

InfoGAN

GAN objective

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

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

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

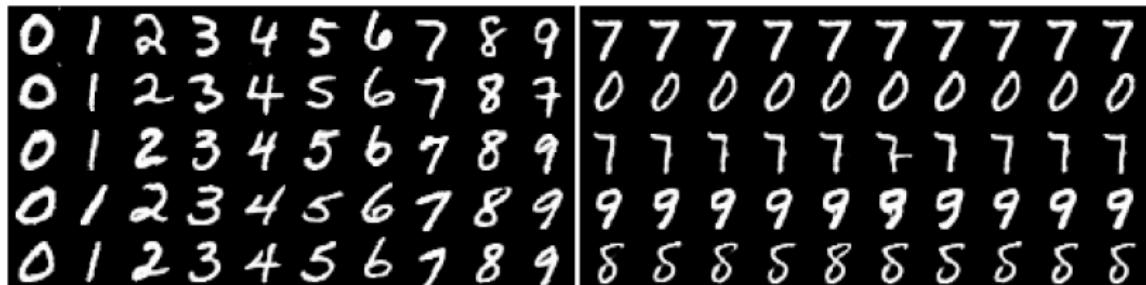
$$\min_G \max_D V(G, D) - \lambda I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$

Variational Information Maximization

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

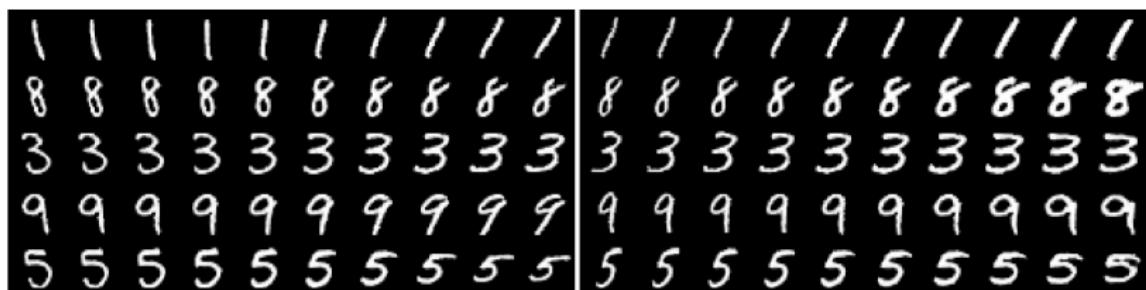
InfoGAN

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)

InfoGAN

Latent codes on 3D Faces



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow

β -VAE, Disentanglement metric

β -VAE

Disentangling metric

1. Generate two sets of objects

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

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

2. Find representations

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

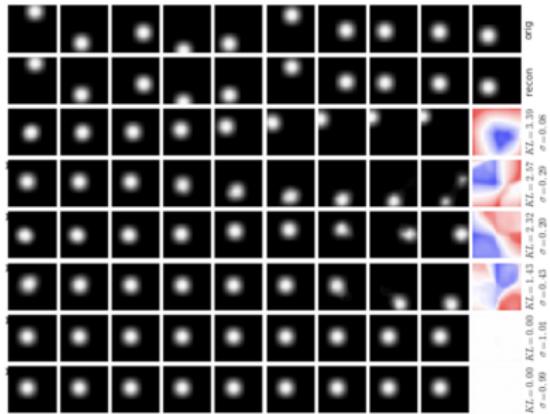
3. Use accuracy of classifier $p(y|\mathbf{z}_{\text{diff}})$ with a low VC-dimension as metric of disentanglement

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

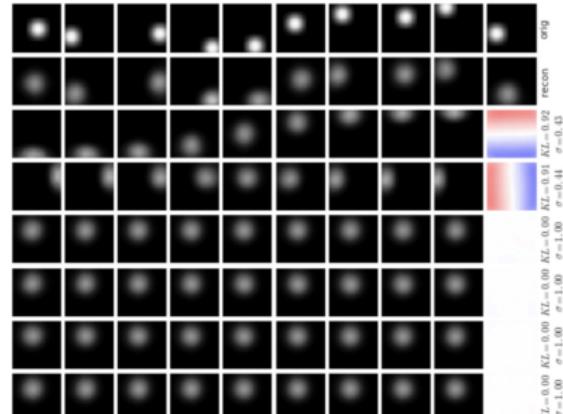
β -VAE

- ▶ **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.

$\beta = 1$



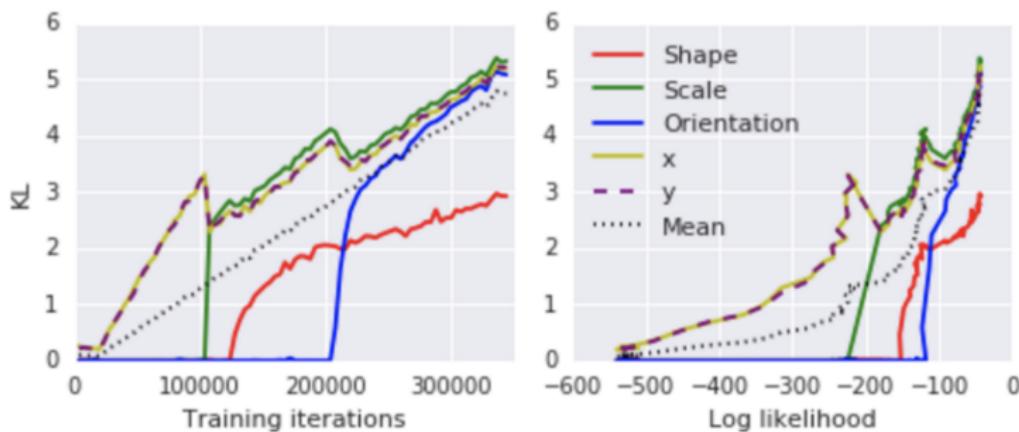
$\beta = 150$



β -VAE

Controlled encoding capacity

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

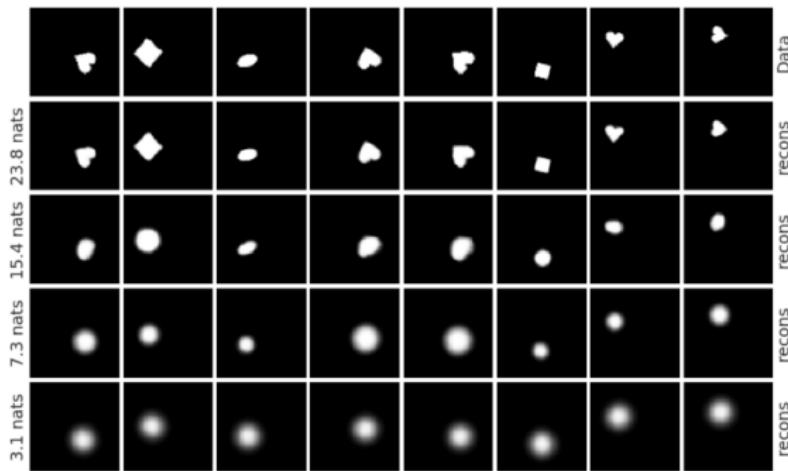


The early capacity is allocated to positional latents only, followed by a scale latent, then shape and orientation latents.

β -VAE

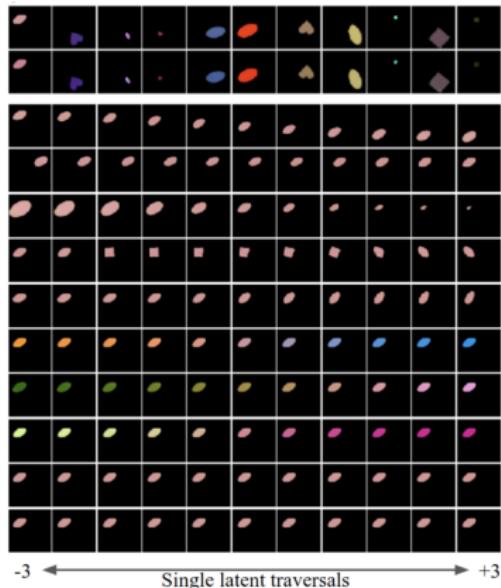
Controlled encoding capacity

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

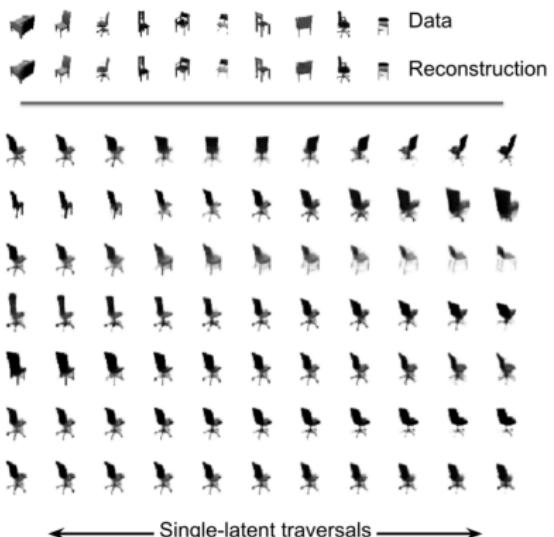


β -VAE

(a) Coloured dSprites



(b) 3D Chairs



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

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

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

FactorVAE

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

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

Density ratio trick

$$\begin{aligned} \frac{q_1(\mathbf{x})}{q_2(\mathbf{x})} &= \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})}{p(y=1)} \Big/ \frac{p(y=0|\mathbf{x})p(\mathbf{x})}{p(y=0)} = \\ &= \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{1 - p(y=1|\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})} \end{aligned}$$

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 $q_2(\mathbf{x})$.

FactorVAE

FactorVAE objective

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

Total correlation regularizer

$$\begin{aligned} KL(q(\mathbf{z}) || \prod_{j=1}^d q(z_j)) &= KL(q(\mathbf{z}) || \bar{q}(\mathbf{z})) = \\ &= \mathbb{E}_{q(\mathbf{z})} \log \frac{q(\mathbf{z})}{\bar{q}(\mathbf{z})} \approx \mathbb{E}_{q(\mathbf{z})} \log \frac{D(\mathbf{z})}{1 - D(\mathbf{z})} \end{aligned}$$

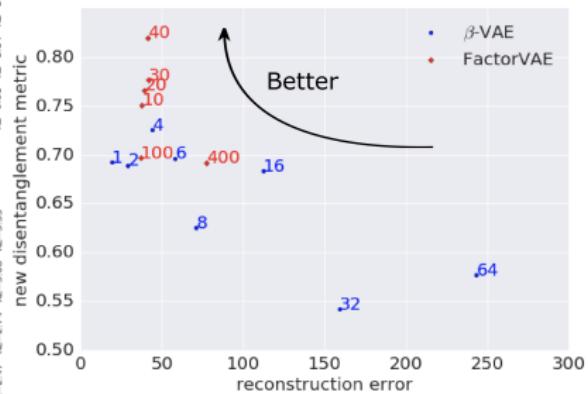
VAE and GAN are trained simultaneously.

FactorVAE

β -VAE ($\beta = 8$)



FactorVAE ($\gamma = 10$)



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.

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).

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

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



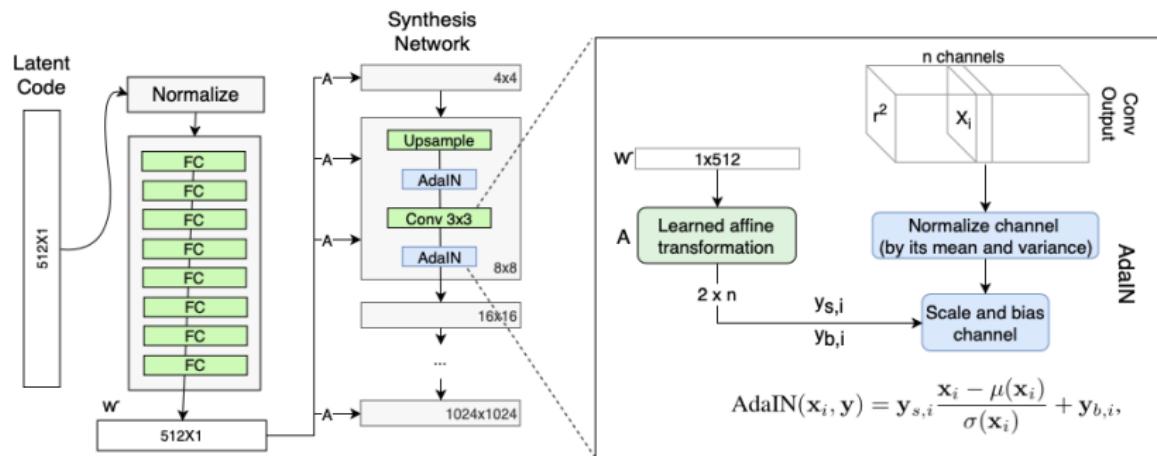
Use reference fixed batch to compute the normalization statistics. To avoid overfitting construct batch with the reference batch and the current sample.

StyleGAN

StyleGAN

Step 2: Style modulation

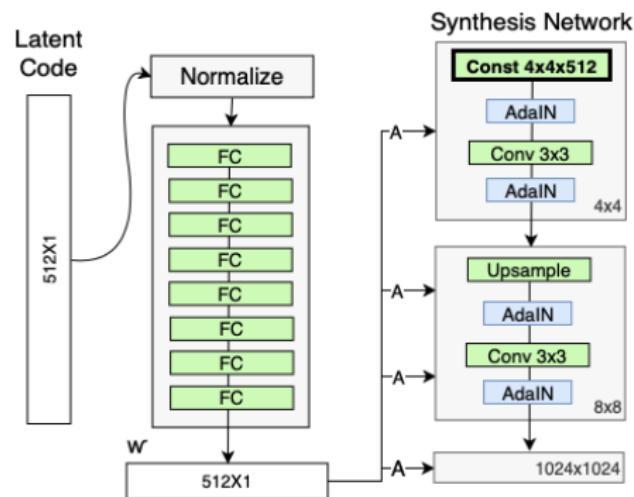
- ▶ Adaptive Instance Normalization transfers the \mathbf{w} vector to the synthesis Network.
- ▶ The module is added to each resolution to define the visual expression of the features.



StyleGAN

Step 3: Remove traditional input

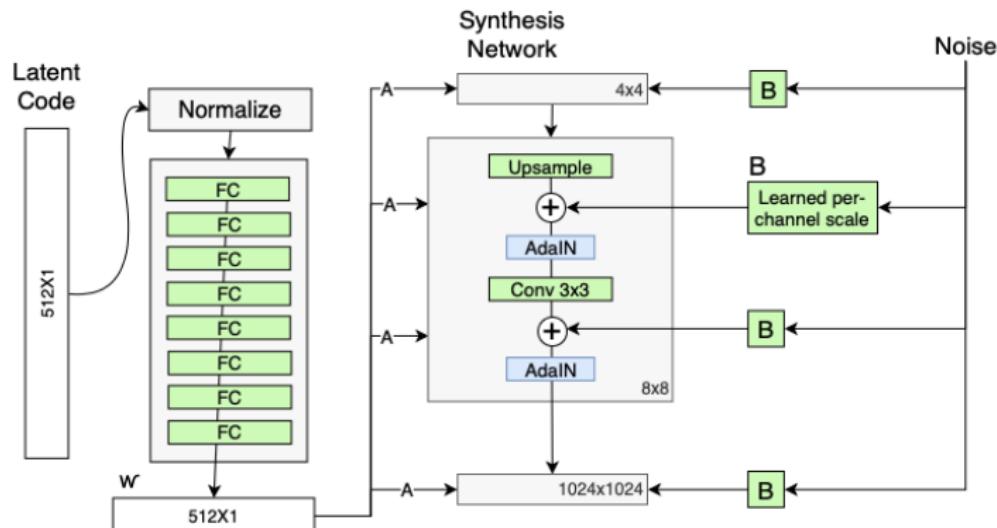
Mapping network provides stochasticity to different stages of the synthesis network. Input of the synthesis network is a trainable vector.



StyleGAN

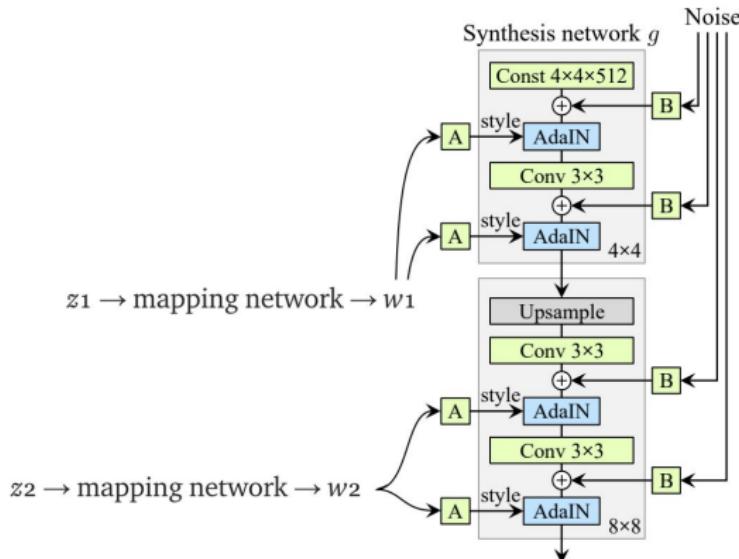
Step 4: Stochastic variation

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.



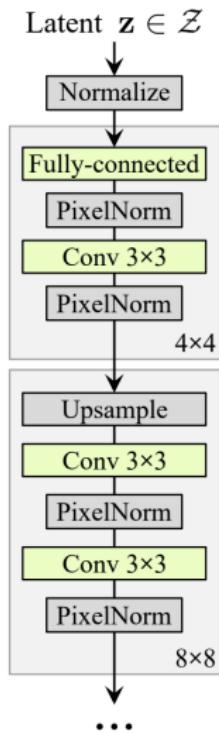
StyleGAN

Step 4: Style Mixing

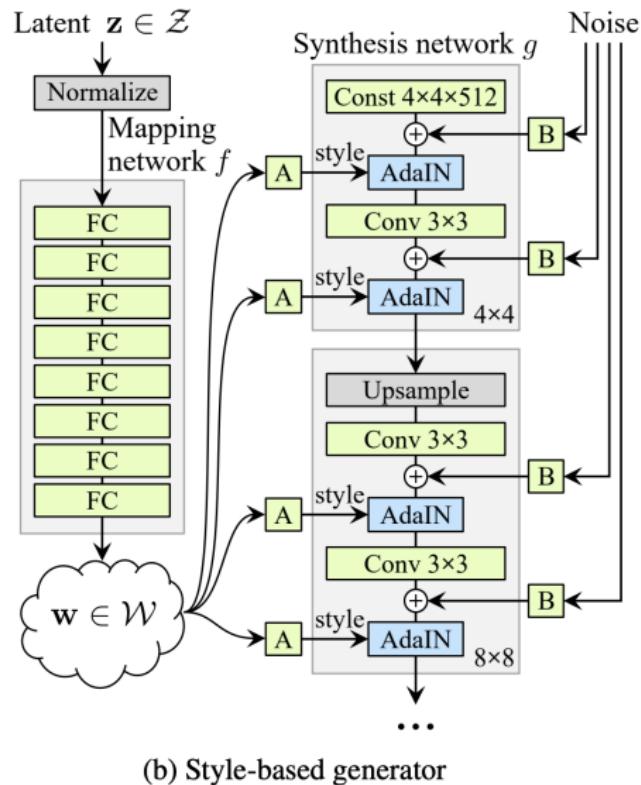


- ▶ Makes different levels of synthesis network to be independent.
- ▶ Allows to couple different styles.

StyleGAN

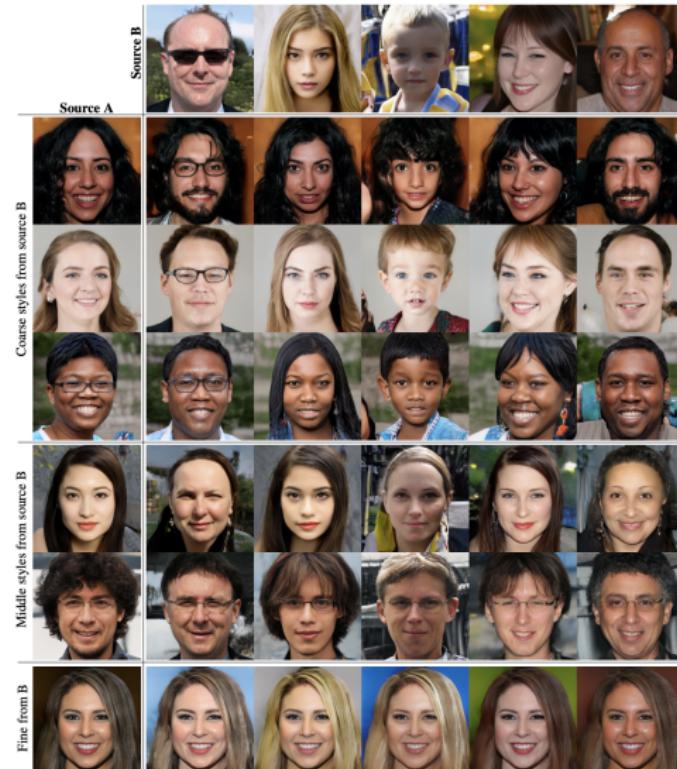


(a) Traditional



(b) Style-based generator

StyleGAN



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