CS 598: Deep Generative and Dynamical Models

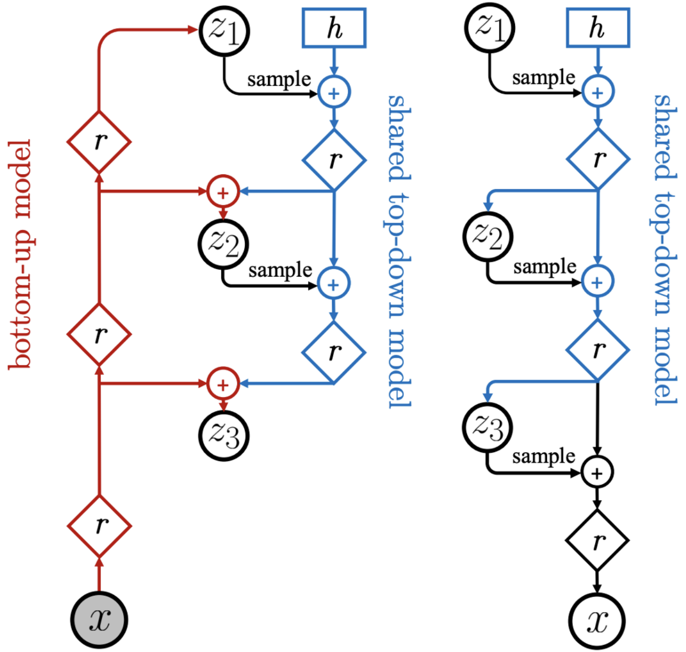
VAE3

Presented by Xiaoyang Bai

NVAE: A Deep Hierarchical Variational Autoencoder

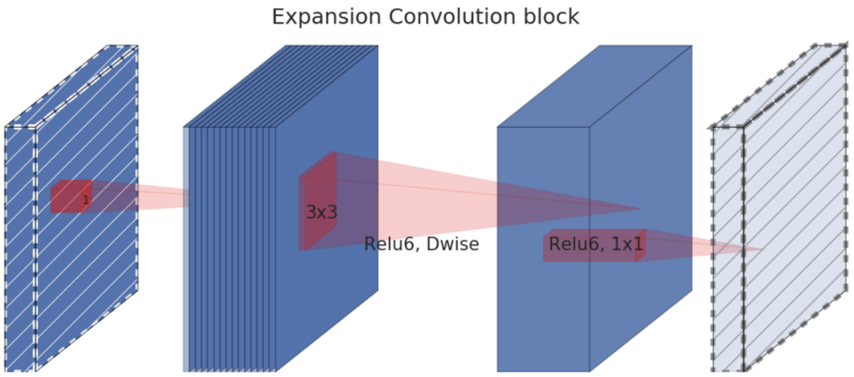
Method: Increasing Long-range Correlation

- Hierarchical multi-scale model
  - $\mathbf{z_1}$ is small-scale
  - Double the spatial size gradually



Method: Increasing Long-range Correlation

- Larger receptive fields
  - Increase the kernel size
  - Depthwise (per-channel) convolution to reduce computation
  - 1x1 convolution layers before and after to scale up number of channels

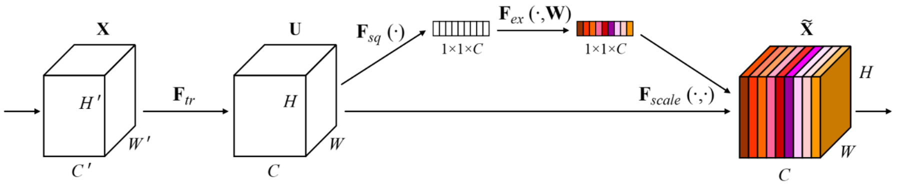


Method: Improving Residual Cells

- Batch normalization (BN) instead of weight normalization (WN)
  - Adjust the momentum hyperparameter
  - Regularization on the norm of scaling parameters
- Swish activation

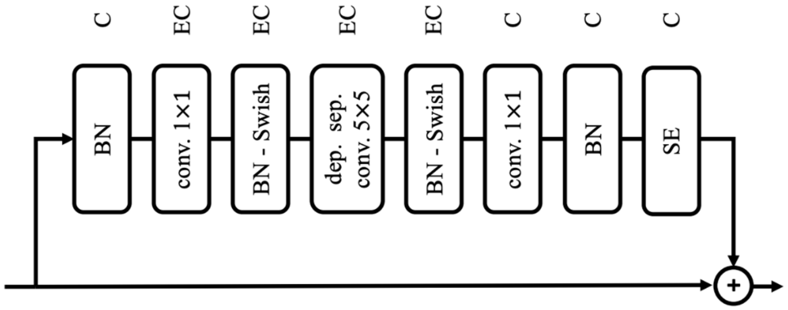
$$f(u) = \frac{u}{1+e^{-u}}$$

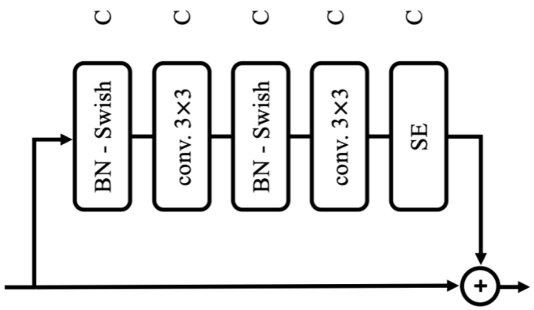
- Squeeze and Excitation (SE) layer
  - Basically a channel-wise attention module



Method: Improving Residual Cells

- Final residual cell architecture (left: decoder, right: encoder):





Method: Stabilizing Training

- The original KL divergence is unstable when two distributions are far away
  - Encoder outputs **log(σ^2)**, and in KL loss there is a term **σ^2 = exp[log(σ^2)]**
- Use residual Normal distribution instead:

$$p(z_l^i|\boldsymbol{z}_{<l}) := \mathcal{N}\left(\mu_i(\boldsymbol{z}_{<l}), \sigma_i(\boldsymbol{z}_{<l})\right)$$

$$q(z_l^i|\boldsymbol{z}_{<l}, \boldsymbol{x}) := \mathcal{N}\left(\mu_i(\boldsymbol{z}_{<l}) + \Delta\mu_i(\boldsymbol{z}_{<l}, \boldsymbol{x}), \sigma_i(\boldsymbol{z}_{<l}) \cdot \Delta\sigma_i(\boldsymbol{z}_{<l}, \boldsymbol{x})\right),$$

- Therefore the KL term becomes:

$$\mathrm{KL}\big(q(z^i|\boldsymbol{x})||p(z^i)\big) = \frac{1}{2}\left(\frac{\Delta\mu_i^2}{\sigma_i^2} + \Delta\sigma_i^2 - \log\Delta\sigma_i^2 - 1\right)$$

- Dropping the exponential term!

Method: Stabilizing Training

- Spectral Regularization (SR):
  - We want the encoder to be Lipschitz
  - So we regularize the largest singular value $\mathbf{s}^{(i)}$ of the **i**-th layer

$$\mathcal{L}_{SR} = \lambda \sum_i s^{(i)}$$

- Additional normalizing flow (NF) layers after encoder output
  - This makes the posterior distribution more expressive

Experiments

- SOTA results among all VAE models

Method	MNIST 28×28	CIFAR-10 32×32	ImageNet 32×32	CelebA 64×64	CelebA HQ 256×256	FFHQ 256×256
NVAE w/o flow	**78.01**	2.93	-	2.04	-	0.71
NVAE w/ flow	78.19	**2.91**	3.92	**2.03**	**0.70**	**0.69**
VAE Models with an Unconditional Decoder						
BIVA [36]	78.41	3.08	3.96	2.48	-	-
IAF-VAE [4]	79.10	3.11	-	-	-	-
DVAE++ [20]	78.49	3.38	-	-	-	-
Conv Draw [42]	-	3.58	4.40	-	-	-
Flow Models without any Autoregressive Components in the Generative Model						
VFlow [59]	-	2.98	-	-	-	-
ANF [60]	-	3.05	3.92	-	0.72	-
Flow++ [61]	-	3.08	**3.86**	-	-	-
Residual flow [50]	-	3.28	4.01	-	0.99	-
GLOW [62]	-	3.35	4.09	-	1.03	-
Real NVP [63]	-	3.49	4.28	3.02	-	-
VAE and Flow Models with Autoregressive Components in the Generative Model						
δ-VAE [25]	-	2.83	3.77	-	-	-
PixelVAE++ [35]	78.00	2.90	-	-	-	-
VampPrior [64]	78.45	-	-	-	-	-
MAE [65]	77.98	2.95	-	-	-	-
Lossy VAE [66]	78.53	2.95	-	-	-	-
MaCow [67]	-	3.16	-	-	0.67	-

Experiments

- Not as good as autoregressive models
  - Will try to solve this problem in the next paper!

Autoregressive Models						
SPN [68]	-	-	3.85	-	0.61	-
PixelSNAIL [34]	-	2.85	3.80	-	-	-
Image Transformer [69]	-	2.90	3.77	-	-	-
PixelCNN++ [70]	-	2.92	-	-	-	-
PixelRNN [41]	-	3.00	3.86	-	-	-
Gated PixelCNN [71]	-	3.03	3.83	-	-	-

Experiments

- Some qualitative results…

(a) MNIST ($t = 1.0$) (b) CIFAR-10 ($t = 0.7$) (c) CelebA 64 ($t = 0.6$)

(d) CelebA HQ ($t = 0.6$) (e) FFHQ ($t = 0.5$)

Experiments

- And ablation study on all aforementioned components:

Table 2: Normalization & activation

Functions	$L=10$	$L=20$	$L=40$
WN + ELU	3.36	3.27	3.31
BN + ELU	3.36	3.26	3.22
BN + Swish	**3.34**	**3.23**	**3.16**

Table 3: Residual cells in NVAE

Bottom-up model	Top-down model	Test (bpd)	Train time (h)	Mem. (GB)
Regular	Regular	3.11	43.3	6.3
Separable	Regular	3.12	49.0	10.6
Regular	Separable	**3.07**	48.0	10.7
Separable	Separable	**3.07**	50.4	14.9

Table 4: The impact of residual dist.

Model	# Act. $\boldsymbol{z}$	Training KL	Training Rec.	Training $\mathcal{L}_{\text{VAE}}$	Test LL
w/ Res. Dist.	53	**1.32**	1.80	**3.12**	**3.16**
w/o Res. Dist.	54	1.36	1.80	3.16	3.19

Table 5: SR & SE

Model	Test NLL
NVAE	**3.16**
NVAE w/o SR	3.18
NVAE w/o SE	3.22

Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images

Motivation: Autoregressive and Latent Variable Models

- Autoregressive Models (e.g. PixelCNN):
    - Learn dependencies within observed variables
- Latent Variable Models (e.g. VAE):
    - Learn dependency between latent & observed variables
- The latter should theoretically be better
    - Faster inference
    - Scalable to higher-dimensional data
    - Potentially functional with a smaller architecture
- However, Gated PixelCNN still outperforms VAE models…

Motivation: Autoregressive and Latent Variable Models

- Autoregressive Models (e.g. PixelCNN):
  - Learn dependencies within observed variables
- Latent Variable Models (e.g. VAE):
  - Learn dependency between latent & observed variables
- The latter should theoretically be better
  - Faster inference
  - Scalable to higher-dimensional data
  - Potentially functional with a smaller architecture
- However, Gated PixelCNN still outperforms VAE models…

Hierarchical VAE

- Use Ladder VAE (LVAE) as base architecture
- The network learns the following probabilities:

$$p_\theta(\boldsymbol{z}) = p_\theta(\boldsymbol{z}_0)p_\theta(\boldsymbol{z}_1|\boldsymbol{z}_0)...p_\theta(\boldsymbol{z}_N|\boldsymbol{z}_{<N}) \tag{2}$$

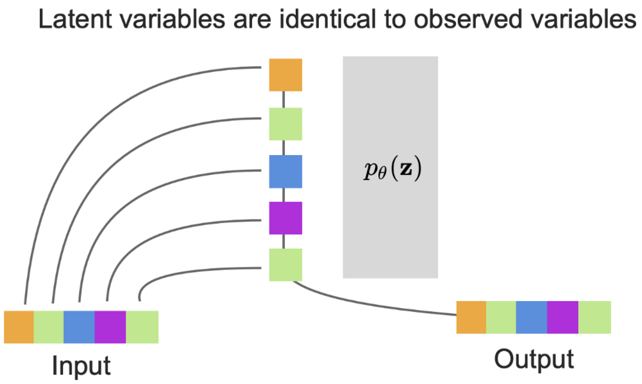
$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}_0|\boldsymbol{x})q_\phi(\boldsymbol{z}_1|\boldsymbol{z}_0, \boldsymbol{x})...q_\phi(\boldsymbol{z}_N|\boldsymbol{z}_{<N}, \boldsymbol{x}) \tag{3}$$

Two Statements

- **N**-layer VAEs generalize autoregressive models when **N** is data dimension
  - With the following settings, we only need to learn dependencies among z's
  - That is, dependencies among observed variables

$$q(z_i = x_i | z_{<i}, \boldsymbol{x}) = 1, \text{ and } p(x_i = z_i | \boldsymbol{z}) = 1.$$

- To visualize:



Two Statements

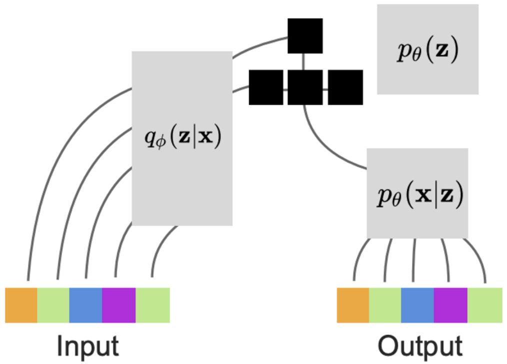
- **N**-layer VAEs can fully represent **N**-dimensional latent densities
  - Proven in *Huang et al. (2017)*
- That is, if the data distribution is on a low-dimensional manifold, we can subsequently reduce the latent dimension and retain full capacity
  - Which is usually the case for image datasets

Moreover…

- Hierarchical VAEs can learn conditional independence of variables
  - Which enables fast parallel computation
  - Formally:

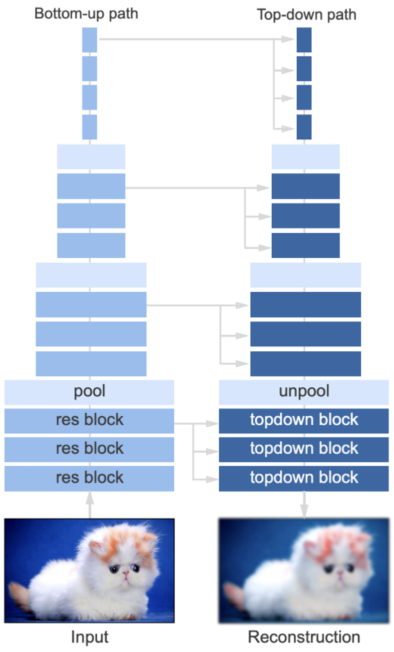
$$q_\phi(\boldsymbol{z}_N | \boldsymbol{z}_{<N}, \boldsymbol{x}) = \prod_d q_\phi(z_N^{(d)} | \boldsymbol{z}_{<N}, \boldsymbol{x})$$

Latent variables allow for parallel generation



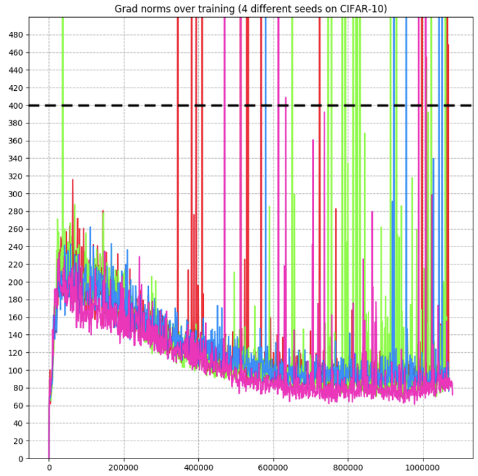
Network Design

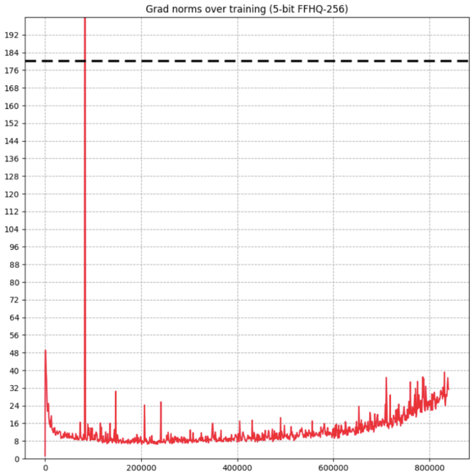
- So theoretically hierarchical VAEs should outperform autoregressive models
    - What is the bottleneck?
- Maybe the depth is not enough!
    - Solution: very deep VAE with ResBlocks



Network Design

- Gradient skipping to stabilize training
  - High threshold so that less than 0.01% of updates are skipped
  - Alternatively: spectral regularization (SR) in NVAE





Experiments

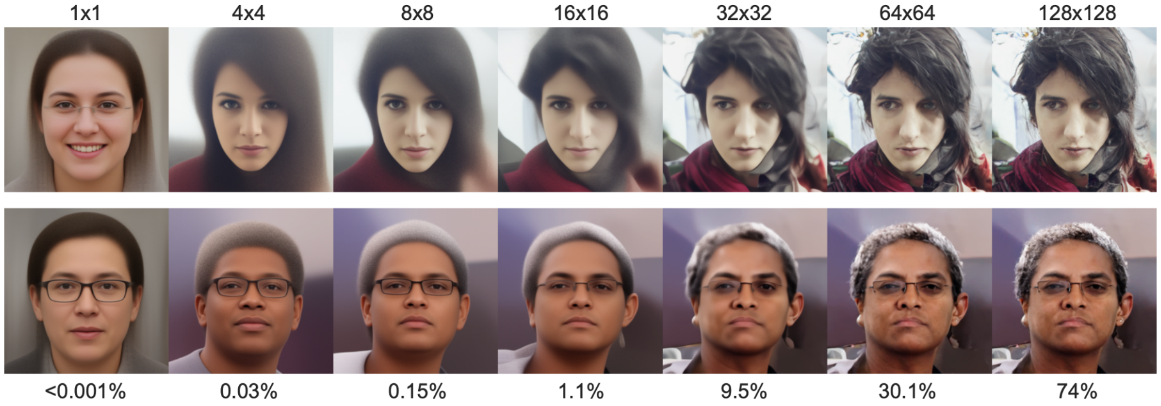
- Group latent variables together to adjust model depth
- Findings:
  - Deeper VAE have larger capacity (left)
  - Higher dimensional latent variables are more powerful (right)

Depth	Params	Test Loss
3	41M	4.30
6	41M	4.18
12	41M	4.06
24	41M	3.98
48	41M	3.95

Distribution of 48 layers					Test Loss
32x32	16x16	8x8	4x4	1x1	
10	10	10	10	8	3.98
12	12	10	8	6	3.97
14	14	10	6	4	3.96
16	16	10	4	2	3.95

Experiments

- Also hierarchical VAEs are more efficient
  - A small number of latent variables encode most of the information
  - Therefore later layers can largely be parallelized
  - We don't need to maintain a latent space as large as the image space



Experiments

- Quantitative evaluation:
  - Comparable performance as autoregressive models and Transformer
  - But less parameters

CIFAR-10					
PixelCNN++ (Salimans et al., 2017)	AR	53M*		D	2.92
PixelSNAIL (Chen et al., 2017)	AR			D	2.85
Sparse Transformer (Child et al., 2019)	AR	59M		D	**2.80**
VLAE (Chen et al., 2016)	VAE			D	≤ 2.95
IAF-VAE (Kingma et al., 2016)	VAE		12	1	≤ 3.11
Flow++ (Ho et al., 2019)	Flow	31M		1	≤ 3.08
BIVA (Maaløe et al., 2019)	VAE	103M	15	1	≤ 3.08
NVAE (Vahdat & Kautz, 2020)	VAE	131M	30	1	≤ 2.91
Very Deep VAE **(ours)**	VAE	39M	45	1	$\leq \mathbf{2.87}$

Experiments

- Quantitative evaluation (cont.)

ImageNet-32					
Gated PixelCNN	AR	177M*	10	D	3.83
Image Transformer (Parmar et al., 2018)	AR			D	**3.77**
BIVA	VAE	103M*	15	1	$\leq$ 3.96
NVAE	VAE	268M	28	1	$\leq$ 3.92
Flow++	Flow	169M		1	$\leq$ 3.86
Very Deep VAE **(ours)**	VAE	119M	78	1	$\leq$ **3.80**
ImageNet-64					
Gated PixelCNN	AR	177M*		D	3.57
SPN (Menick & Kalchbrenner, 2018)	AR	150M		D	3.52
Sparse Transformer	AR	152M		D	**3.44**
Glow (Kingma & Dhariwal, 2018)	Flow			1	3.81
Flow++	Flow	73M		1	$\leq$ 3.69
Very Deep VAE **(ours)**	VAE	125M	75	1	$\leq$ **3.52**

Also…

- VAEs can easily scale to very high-dimensional data
    - For example, 1024x1024 images
    - While PixelCNNs cannot