Group Meeting - September 11, 2020

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Sun Tzu

The supreme art of war is to subdue the enemy **without** fighting.

The Great Chinggis Khaan (Genghis Khan)

Violence **never** settles anything.

Buddha

1. To **understand** everything is to **forgive** everything.
2. We can never obtain peace in the outer world until we make **peace with ourselves**.

The Art Institute of Chicago

Papers

1. **Learning Hierarchical Features from Generative Models** (ICML 2017)
   https://arxiv.org/pdf/1702.08396.pdf
2. **Improved Variational Inference with Inverse Autoregressive Flow** (NIPS 2016)
   https://arxiv.org/pdf/1606.04934.pdf
3. **NVAE: A Deep Hierarchical Variational Autoencoder**
   https://arxiv.org/pdf/2007.03898.pdf
4. **Hierarchical Implicit Models and Likelihood-Free Variational Inference** (NIPS 2017)
   https://arxiv.org/pdf/1702.08896.pdf
5. **Deep Hierarchical Implicit Models**
   https://arxiv.org/pdf/1702.08896v1.pdf
6. **Discrete Object Generation with Reversible Inductive Construction**
   https://arxiv.org/pdf/1907.08268.pdf

Note: Paper 1 criticizes paper 2. But both papers claim valid points, and have some frauds. Paper 3 is tunning of paper 2. Paper 5 is a trivial extension of paper 4.

Paper 1

Learning Hierarchical Features from Generative Models (ICML 2017)

Shengjia Zhao, Jiaming Song, Stefano Ermon (Stanford)

`https://arxiv.org/pdf/1702.08396.pdf`

Motivations:

- A key property of deep feed-forward networks is that they tend to learn increasingly abstract and invariant representations at higher levels in the hierarchy.

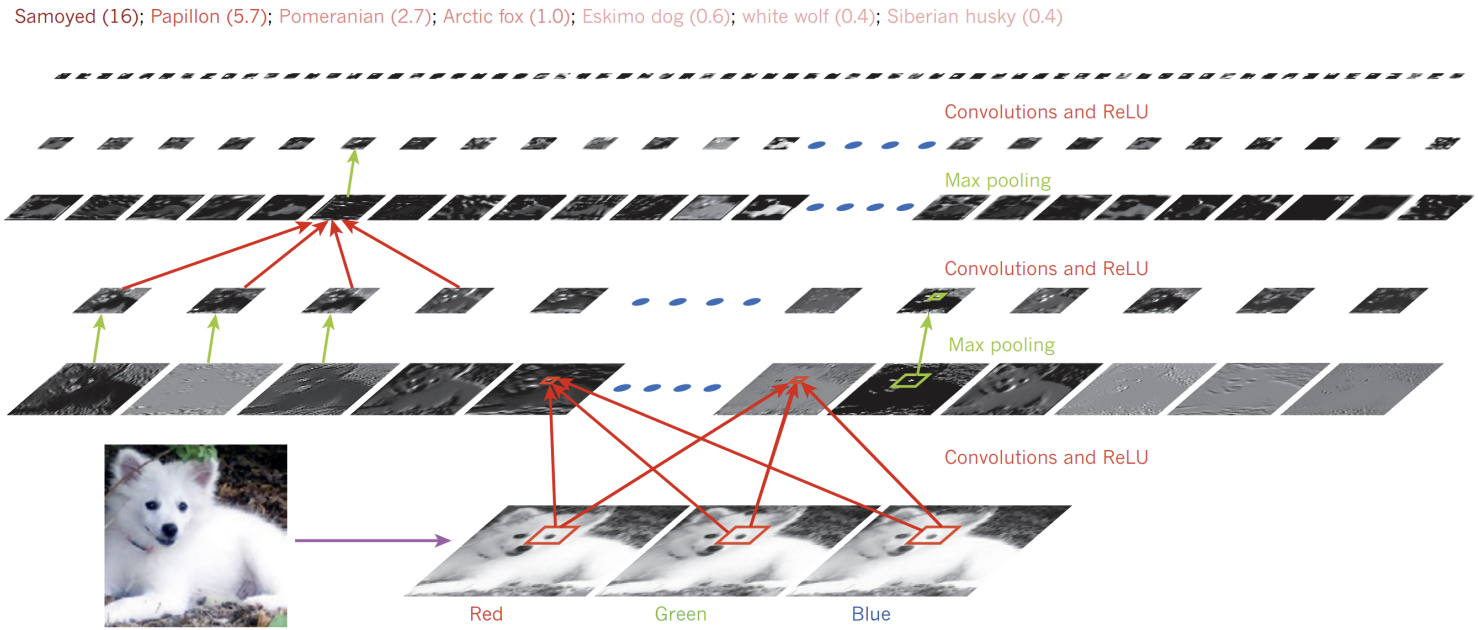


Figure 2 | Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map corresponding to the output for one of the learned features, detected at each of the image positions. Information flows bottom up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output. ReLU, rectified linear unit.

Deep Learning (Nature), Yann LeCun, Yoshua Bengio, Geoffrey Hinton

Motivations:

- Generative models with a hierarchical structure, where there are multiple layers of latent variables, have been **less successful** compared to their supervised counter-parts.
- Why:
  - **For classification:** Local features are often sufficient.
  - **For generation (of images):** Preservation of details are needed. The size and location of a sub-part often has to be dependent on the other sub-parts ← For example, an eye should only be generated with the same size as the other eye, at symmetric locations with respect to the center of the face, with appropriate distance between them.

Learning Hierarchical Features from Generative Models (3)

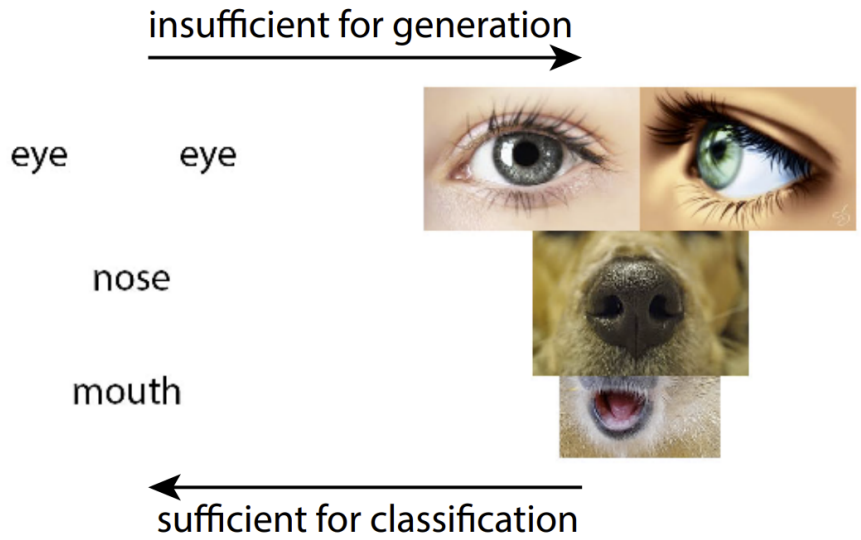


Figure 1. Left: Body parts feature detectors only carry a small amount of information about an underlying image, yet, it is sufficient for a confident classification as a face. Right: if a hierarchical generative model attempts to reconstruct an image based on these high-level features, it could generate inconsistent images, even when each part can be perfectly generated. Even though this "face" is clearly absurd, Google cloud platform classification API can identify with 93% confidence that this is a face.

Proposals

1. **Stacked hierarchy:** Recursively stacking generative models on top of each other. If these models can be trained to optimality, then the bottom layer alone contains enough information to reconstruct the data distribution already, and the layers above can be ignored.
2. **Architectural hierarchy:** Single layer latent variable models that prefers to place high-level features on certain part of the latent code, and low-level features in others (**Variational Ladder Autoencoder**).

Consider a family of latent variable models specified by a joint probability distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ over a set of observed variables $\boldsymbol{x}$ and latent variables $\boldsymbol{z}$, parameterized by θ. Let $p_\theta(\boldsymbol{x})$ denote the marginal distribution of $\boldsymbol{x}$.

The objective is to maximize the marginal log-likelihood $p(\boldsymbol{x})$ over a dataset $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, .., \boldsymbol{x}^{(N)}\}$ drawn from some unknown underlying distribution $p_{\text{data}}(\boldsymbol{x})$:

$$\log p_\theta(\boldsymbol{X}) = \sum_{n=1}^{N} \log p_\theta(\boldsymbol{x}^{(i)})$$

which is non-convex and often intractable, as it involves marginalization over the latent variables $\boldsymbol{z}$.

Variational Autoencoders: For optimizing the intractable marginal likelihood, we optimize the evidence lower bound (ELBO) by introducing an inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ parameterized by ϕ:

$$\log p((x)) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{x})]$$

$$= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathcal{D}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) = \mathcal{L}(\boldsymbol{x}; \theta, \phi)$$

Inference Suboptimality in Variational Autoencoders (ICML 2018)

https://arxiv.org/pdf/1801.03558.pdf

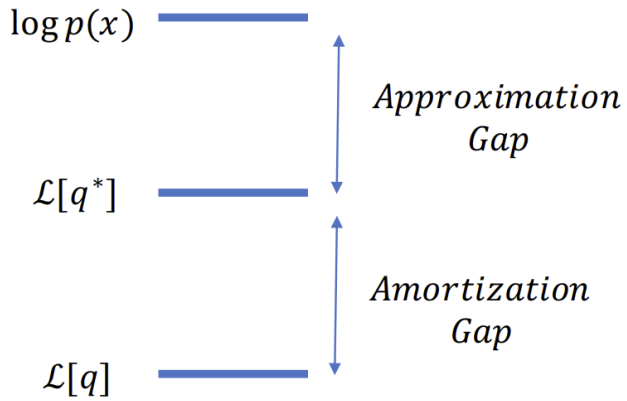


Figure 1. Gaps in Inference

Term	Definition	VAE Formulation
Inference	$\log p(x) - \mathcal{L}[q]$	$\mathrm{KL}\left(q(z\|x)\|\|p(z\|x)\right)$
Approximation	$\log p(x) - \mathcal{L}[q^*]$	$\mathrm{KL}\left(q^*(z\|x)\|\|p(z\|x)\right)$
Amortization	$\mathcal{L}[q^*] - \mathcal{L}[q]$	$\mathrm{KL}\left(q(z\|x)\|\|p(z\|x)\right) - \mathrm{KL}\left(q^*(z\|x)\|\|p(z\|x)\right)$

Table 1. Summary of Gap Terms. The middle column refers to the general case where our variational objective is a lower bound on the marginal log-likelihood. The right most column demonstrates the specific case in VAEs. $q^*(z|x)$ refers to the optimal approximation within a family $\mathcal{Q}$, i.e. $q^*(z|x) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(z|x)||p(z|x)\right)$.

Inference Suboptimality in Variational Autoencoders (ICML 2018)

https://arxiv.org/pdf/1801.03558.pdf

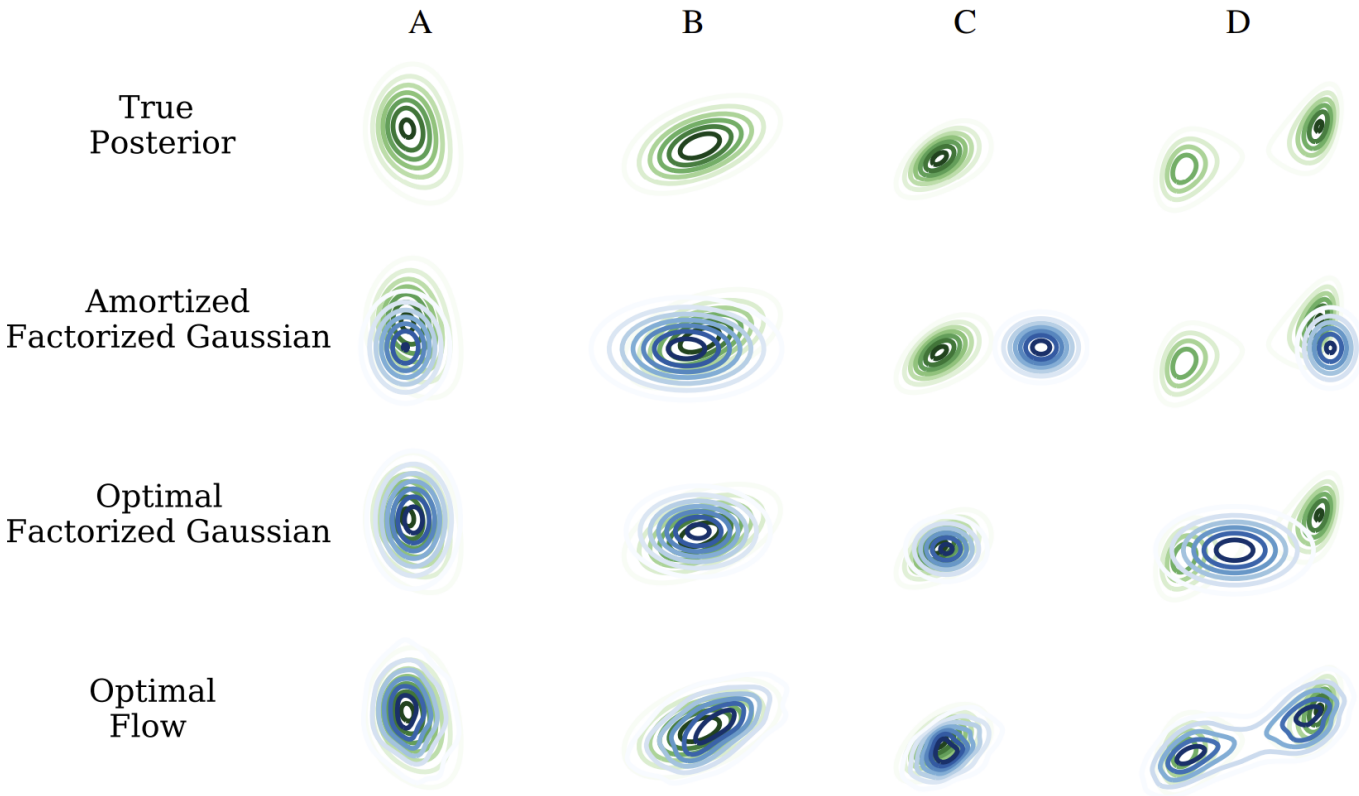


Figure 2. True Posterior and Approximate Distributions of a VAE with 2D latent space. The columns represent four different datapoints. The green distributions are the true posterior distributions, highlighting the mismatch with the blue approximations. Amortized: Variational parameters learned over the entire dataset. Optimal: Variational parameters optimized for each individual datapoint. Flow: Using a flexible approximate distribution.

Hierarchical Variational Autoencoders (HVAE): A series of VAEs stacked on top of each other. Hierarchy of latent variables $\boldsymbol{z} = \{\boldsymbol{z}_1, .., \boldsymbol{z}_L\}$, $\boldsymbol{z}_1$ represents the lowest layer closest to $\boldsymbol{x}$, and $\boldsymbol{z}_L$ the top layer:

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

where $\boldsymbol{z}_{>\ell}$ indicates $(\boldsymbol{z}_{\ell+1}, .., \boldsymbol{z}_L)$. Assume a Markov independence structure on the hidden variables, leading to the following simpler factorization:

$$p(\boldsymbol{x}, \boldsymbol{z}_1, .., \boldsymbol{z}_L) = p(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{\ell=1}^{L-1} p(\boldsymbol{z}_\ell|\boldsymbol{z}_{\ell+1}) p(\boldsymbol{z}_L)$$

Hierarchical Variational Autoencoders (HVAE): For the inference distribution $q(\boldsymbol{z}|\boldsymbol{x})$, we do not assume any factorized structure. Denote $q(\boldsymbol{x}, \boldsymbol{z}) = p_{\text{data}}(\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})$ Both $p(\boldsymbol{x}|\boldsymbol{z})$ and $q(\boldsymbol{z}|\boldsymbol{x})$ are jointly optimized as maximizing the ELBO objective:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathcal{D}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))\right]$$

$$= \sum_{\ell=0}^{L} \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}[\log p(\boldsymbol{z}_\ell|\boldsymbol{z}_{>\ell})] + H(q(\boldsymbol{z}|\boldsymbol{x}))$$

where we define $\boldsymbol{z}_0 \triangleq \boldsymbol{x}$, and H is the entropy of a distribution.

Representational efficiency

Proposition 1. $\mathcal{L}_{\text{ELBO}}$ is globally maximized as a function of $q(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{z})$ when $\mathcal{L}_{\text{ELBO}} = -H(p_{\text{data}}(x))$. If $\mathcal{L}_{\text{ELBO}}$ is globally maximized for a Markov HVAE, the following Gibbs sampling chain converges to $p_{\text{data}}(x)$ if it is ergodic:

$$\boldsymbol{z}_1^{(t)} \sim q(\boldsymbol{z}_1|\boldsymbol{x}^{(t)})$$

$$\boldsymbol{x}^{(t+1)} \sim p(\boldsymbol{x}|\boldsymbol{z}_1^{(t)})$$

What does that mean?

One can sample from $p_{\text{data}}(\boldsymbol{x})$ without using $p(\boldsymbol{z}_\ell|\boldsymbol{z}_{>\ell})$, $1 \leq \ell < L$ at all.
The Gibbs chain only used the bottom layer of the model.

We **hope**:

- $p(\boldsymbol{z}|\boldsymbol{x})$ is a probabilistic feature detector, and $q(\boldsymbol{z}|\boldsymbol{x})$ as an approximation to it.
- q might learn hierarchical features similarly to a feed-forward network $\boldsymbol{x} \to \boldsymbol{z}_1 \to \ldots \to \boldsymbol{z}_L$, where higher layers correspond to higher level features (increasingly abstract).

But:

- If $q(\boldsymbol{z}_{>\ell}|\boldsymbol{z}_\ell)$ maps low-level features to high-level features, then the reverse mapping $q(\boldsymbol{z}_\ell|\boldsymbol{z}_{>\ell})$ maps high-level features into likely low-level sub-features.
- However, the only type of feature hierarchy we can hope to learn is one under which $q(\boldsymbol{z}_\ell|\boldsymbol{z}_{>\ell})$ is also Gaussian. While the hierarchies we observe for feed-forward models require complex multimodal distributions to be captured.

Variational Ladder Autoencoders (VLAE):

- Assume that if $\boldsymbol{z}_i$ is more abstract than $\boldsymbol{z}_j$, then the inference mapping $q(\boldsymbol{z}_i|\boldsymbol{x})$, and generative mapping when other layers are fixed $p(\boldsymbol{x}|\boldsymbol{z}_i, \boldsymbol{z}_{\neg i} = z^0_{\neg i})$ requires a more expressive network to capture.
- **No** hierarchical structure over the latent variables. VLAE encourages the latent code $\boldsymbol{z}_1, .., \boldsymbol{z}_L$ to learn features with different levels of abstraction by carefully choosing the mapping $p(\boldsymbol{x}|\boldsymbol{z})$ and $q(\boldsymbol{z}|\boldsymbol{x})$.

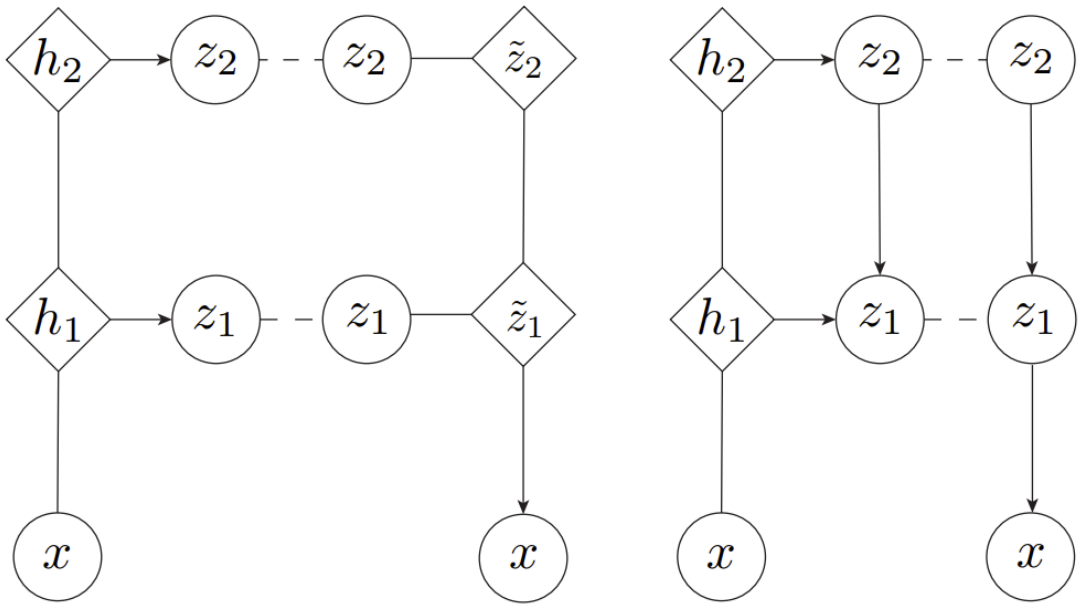


Figure 4. Inference and generative models for VLAE (left) and LVAE (right). Circles indicate stochastic nodes, and squares are deterministically computed nodes. Solid lines with arrows denote conditional probabilities; solid lines without arrows denote deterministic mappings; dash lines indicates regularization to match the prior $p(\mathbf{z})$. Note that in VLAE, we do not attempt to regularize the distance between $\mathbf{h}$ and $\tilde{\mathbf{z}}$.

VLAE's Generative Network: $p(\boldsymbol{z}) = p(\boldsymbol{z}_1, .., \boldsymbol{z}_L)$ is a simple prior on all latent variables, chosen as a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{1})$. The conditional distribution $p(\boldsymbol{x}|\boldsymbol{z}_1, .., \boldsymbol{z}_L)$ is defined implicitly as:

$$\hat{\boldsymbol{z}}_L = \boldsymbol{f}_L(\boldsymbol{z}_L)$$

$$\hat{\boldsymbol{z}}_\ell = \boldsymbol{f}_\ell(\hat{\boldsymbol{z}}_{\ell+1}, \boldsymbol{z}_\ell) \qquad \ell = 1, .., L-1$$

$$\boldsymbol{x} \sim \boldsymbol{r}(\boldsymbol{x}; \boldsymbol{f}_0(\hat{\boldsymbol{z}}_1)) \triangleq \mathcal{N}(\boldsymbol{f}_0(\hat{\boldsymbol{z}}_1), \mathbf{1})$$

where $\boldsymbol{f}_\ell$ is a parameterized neural network:

$$\hat{\boldsymbol{z}}_\ell = \boldsymbol{f}_\ell(\hat{\boldsymbol{z}}_{\ell+1}, \boldsymbol{z}_\ell) = \boldsymbol{u}_\ell([\hat{\boldsymbol{z}}_{\ell+1}; \boldsymbol{v}_\ell(\boldsymbol{z}_\ell)])$$

where $\boldsymbol{u}_\ell$ and $\boldsymbol{v}_\ell$ are neural networks, and $[.,.]$ denotes the concatenation operation of two vectors.

VLAE's Inference Network: For the inference newtork, we choose $q(\boldsymbol{z}|\boldsymbol{x})$ as

$$\boldsymbol{h}_\ell = \boldsymbol{g}_\ell(\boldsymbol{h}_{\ell-1})$$

$$\boldsymbol{z}_\ell \sim \mathcal{N}(\boldsymbol{\mu}_\ell(\boldsymbol{h}_\ell), \boldsymbol{\sigma}_\ell(\boldsymbol{h}_\ell))$$

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

VLAE's Learning:

$$\log p((x)) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{x})]$$

$$= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathcal{D}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$$

Paper 2

Improved Variational Inference with Inverse Autoregressive Flow

(NIPS 2016)

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, Max Welling

`https://arxiv.org/pdf/1606.04934.pdf`

Let $p(\boldsymbol{x}, \boldsymbol{z})$ be the parametric **generative model**. Perform maximum marginal likelihood learning of its parameters, i.e. to maximize:

$$\log p(\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, .., \boldsymbol{x}^{(N)}\}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)})$$

Introduce $q(\boldsymbol{z}|\boldsymbol{x})$ a parametric **inference model** and optimize the **variational lower bound** on the marginal log-likelihood of each observation $\boldsymbol{x}$:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{x})] = \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta})$$

$\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta})$ is clearly a lower bound of $\log p(\boldsymbol{x})$:

$$\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}) = \log p(\boldsymbol{x}) - \mathcal{D}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x}))$$

Normalizing Flow (NF), introduced by Rezende and Mohamed (2015):

- Start with an initial random variable with a relatively simple distribution with known (and computationally cheap) probability density function: $\boldsymbol{z}_0 \sim q(\boldsymbol{z}_0|\boldsymbol{x})$
- Apply a chain of invertible parameterized transformation $\boldsymbol{f}_t$, such that the last iterate $\boldsymbol{z}_T$ has a more flexible distribution: $\boldsymbol{z}_t = \boldsymbol{f}_t(\boldsymbol{z}_{t-1}, \boldsymbol{x})$

Probability density function of the last iterate:

$$\log q(\boldsymbol{z}_T|\boldsymbol{x}) = \log q(\boldsymbol{z}_0|\boldsymbol{x}) - \sum_{t=1}^{T} \log \det \left| \frac{d\boldsymbol{z}_t}{d\boldsymbol{z}_{t-1}} \right|$$

The authors proposed a limited family of **invertible** (why invertible?) transformations with known Jacobian determinant ($\boldsymbol{u}$, $\boldsymbol{w}$ vectors and h non-linearity):

$$\boldsymbol{f}_t(\boldsymbol{z}_{t-1}) = \boldsymbol{z}_{t-1} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{z}_{t-1} + b)$$

Inverse Autoregressive Flow (IAF):

1. An initial encoder neural network output $\boldsymbol{\mu}_0$ and $\boldsymbol{\sigma}_0$, in addition to an extra output $\boldsymbol{h} \leftarrow$ inspired by LSTM.
2. We draw a random sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and initialize the chain

$$\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \odot \boldsymbol{\epsilon}$$

3. The flow consists of a chain of T of the following transformations:

$$\boldsymbol{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \boldsymbol{z}_{t-1}$$

where at the t-th step of the flow, we use a different autoregressive neural network with inputs $\boldsymbol{z}_{t-1}$ and $\boldsymbol{h}$, and outputs $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$.

Improved Variational Inference with Inverse Autoregressive Flow (4)

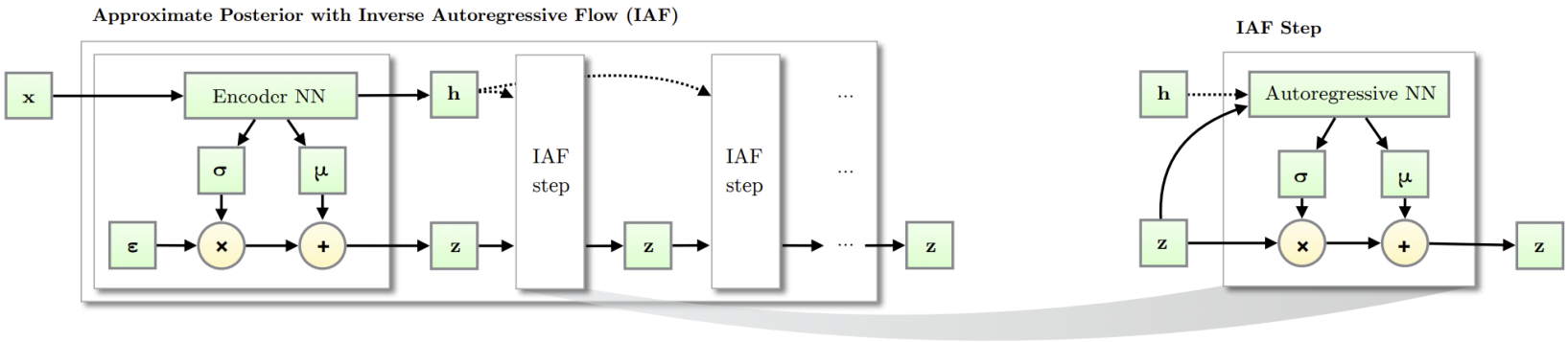


Figure 2: Like other normalizing flows, drawing samples from an approximate posterior with Inverse Autoregressive Flow (IAF) consists of an initial sample $\mathbf{z}$ drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of $\mathbf{z}$, each with a simple Jacobian determinants.

Improved Variational Inference with Inverse Autoregressive Flow (5)

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

Data:

$\mathbf{x}$: a datapoint, and optionally other conditioning information
$\boldsymbol{\theta}$: neural network parameters
$\texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
$\texttt{AutoregressiveNN}[*](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
$\texttt{sum}(.)$: sum over vector elements
$\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:

$\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample '$\mathbf{z}$'

$[\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\epsilon}^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m}, \mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1 - \boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end

Improved Variational Inference with Inverse Autoregressive Flow (6)

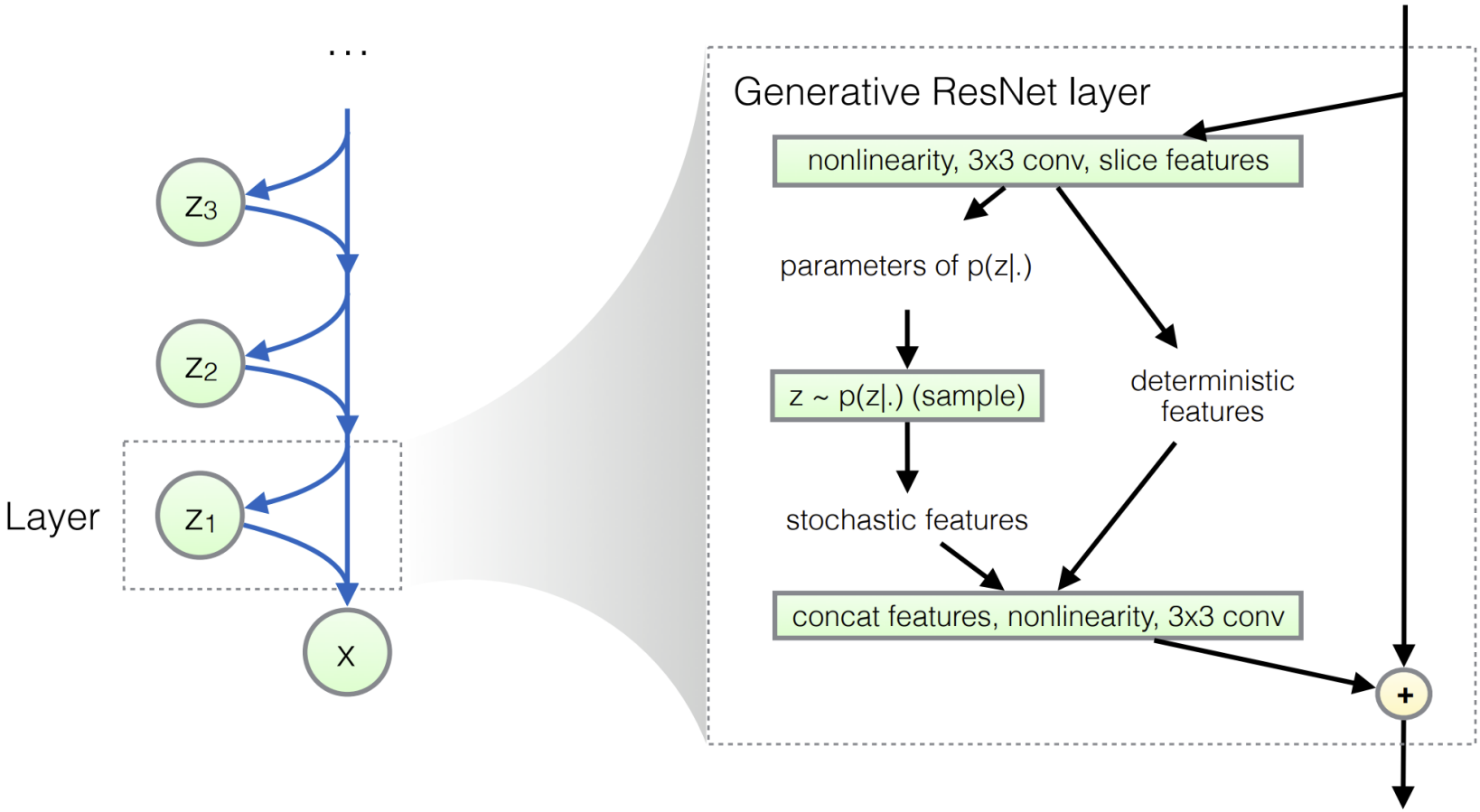


Figure 4: Generative ResNet and detail of layer. This is the generative component of our ResNet VAE.

Paper 3

NVAE: A Deep Hierarchical Variational Autoencoder

Arash Vahdat, Jan Kautz (NVIDIA)

`https://arxiv.org/pdf/2007.03898.pdf`

NVAE: A Deep Hierarchical Variational Autoencoder (1)

I believe there must be someones who look (ever looked) like these generated pictures!

Figure 1: 256×256-pixel samples generated by NVAE, trained on CelebA HQ [28].

The first row is not based on any scientific evidence, but the Bible (!?)

Year	Population	Births per 1,000	Births Between Benchmarks	Number Ever Born	Percent of Those Ever Born
50,000 B.C.E.	2	–	–	–	–
8000 B.C.E.	5,000,000	80	1,137,789,769	1,137,789,769	0.4
1 C.E.	300,000,000	80	46,025,332,354	47,163,122,125	0.6
1200	450,000,000	60	26,591,343,000	73,754,465,125	0.6
1650	500,000,000	60	12,782,002,453	86,536,467,578	0.6
1750	795,000,000	50	3,171,931,513	89,708,399,091	0.9
1850	1,265,000,000	40	4,046,240,009	93,754,639,100	1.3
1900	1,656,000,000	40	2,900,237,856	96,654,876,956	1.7
1950	2,516,000,000	31-38	3,390,198,215	100,045,075,171	2.5
1995	5,760,000,000	31	5,427,305,000	105,472,380,171	5.5
2011	6,987,000,000	23	2,130,327,622	107,602,707,793	6.5
2019	**7,692,000,000**	**19**	**1,157,835, 998**	**108,760,543,791**	**7.1**
2030	**8,932,000,000**	**16**	**2,226,104,131**	**110,986,647,922**	**8.0**
2050	**9,854,000,000**	**14**	**2,120,539,877**	**113,107,187,799**	**8.7**

`https://www.prb.org/howmanypeoplehaveeverlivedonearth/`

(Data from United Nations)

Summary:

- This paper does **not** propose new model, but a carefully designed neural architecture for hierarchical VAEs (named Nouveau VAE or NVAE):
  1. Depth-wise separable convolutions and batch normalization.
  2. Residual parameterization of Normal distributions.
  3. Spectral regularization to stabilized the training.
- Hierarchical VAEs (Max Welling's work):
  1. Normalizing flows
  2. Autoregressive models
  3. Deep energy-based models

Insights:

1. VAEs often respond differently to the over-parameterization in neural networks. Over-parameterization the decoder network may hurt the test log-likelihood, whereas powerful encoders can yield better models because of reducing the amortization gap.
   Inference Suboptimality in Variational Autoencoders (ICML 2018)
   `https://arxiv.org/pdf/1801.03558.pdf`
2. VAEs should model long-range correlations in data, requiring the networks to have large receptive fields.

Inspired by **Inverse Autoregressive Flow** or IAF (Paper 2), but not exactly:

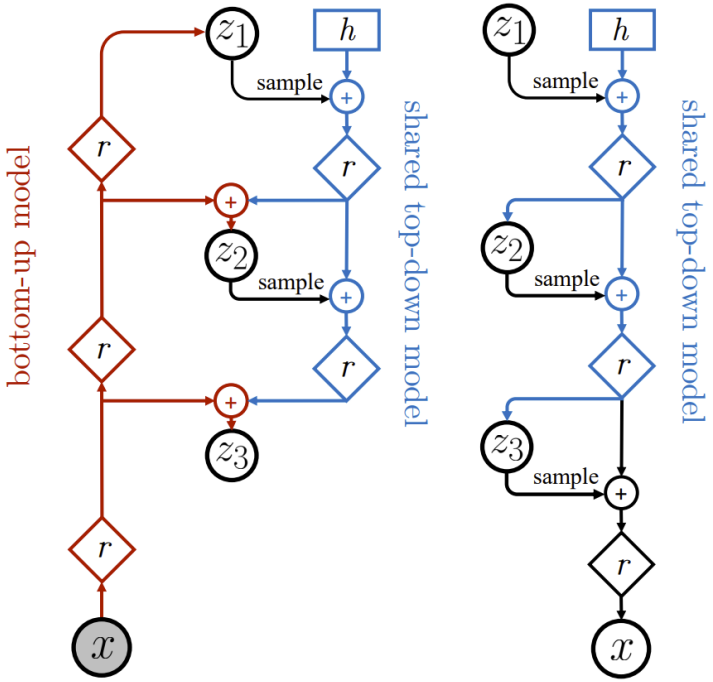
- To increase the expressiveness of both the approximate posterior and prior, the latent variables are partitioned into disjoint groups $\boldsymbol{z} = \{\boldsymbol{z}_1, .., \boldsymbol{z}_T\}$.
- The prior is represented by $p(\boldsymbol{z}) = \prod_t p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$
- The approximate posterior is represented by $p(\boldsymbol{z}|\boldsymbol{x}) = \prod_t q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x})$
- Each conditional $p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$ and $q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x})$ are represented by factorial Normal distributions.

Variational lower bound $\mathcal{L}_{VAE}(\boldsymbol{x})$ on $\log p(\boldsymbol{x})$ as:

$$\mathcal{L}_{VAE}(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathcal{D}(q(\boldsymbol{z}_1|\boldsymbol{x})||p(\boldsymbol{z}_1))$$

$$- \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{z}_{<t}|\boldsymbol{x})}[\mathcal{D}(q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}))||p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})]$$

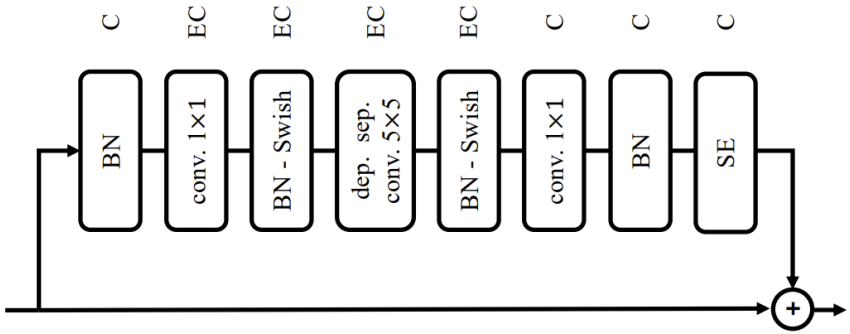
NVAE: A Deep Hierarchical Variational Autoencoder (6)



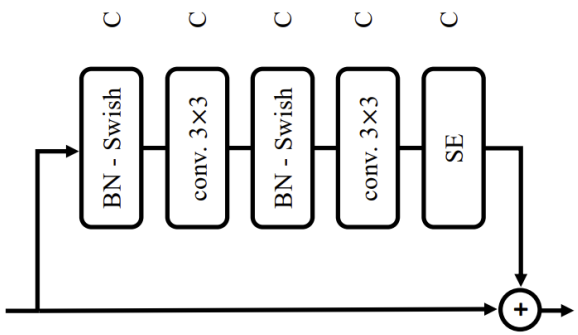
(a) Bidirectional Encoder (b) Generative Model

Figure 2: The neural networks implementing an encoder $q(\boldsymbol{z}|\boldsymbol{x})$ and generative model $p(\boldsymbol{x}, \boldsymbol{z})$ for a 3-group hierarchical VAE. ◇ denotes residual neural networks, ⊕ denotes feature combination (e.g., concatenation), and □ is a trainable parameter.

NVAE: A Deep Hierarchical Variational Autoencoder (7)



(a) Residual Cell for NVAE Generative Model



(b) Residual Cell for NVAE Encoder

Figure 3: The NVAE residual cells for generative and encoder models are shown in (a) and (b). The number of output channels is shown above. The residual cell in (a) expands the number of channels E times before applying the depthwise separable convolution, and then maps it back to C channels. The cell in (b) applies two series of BN-Swish-Conv without changing the number of channels.

Paper 4

Hierarchical Implicit Models and Likelihood-Free Variational Inference (NIPS 2017)

Dustin Tran, Rajesh Ranganath, David M. Blei

`https://arxiv.org/pdf/1702.08896.pdf`

What are implicit probabilistic models?

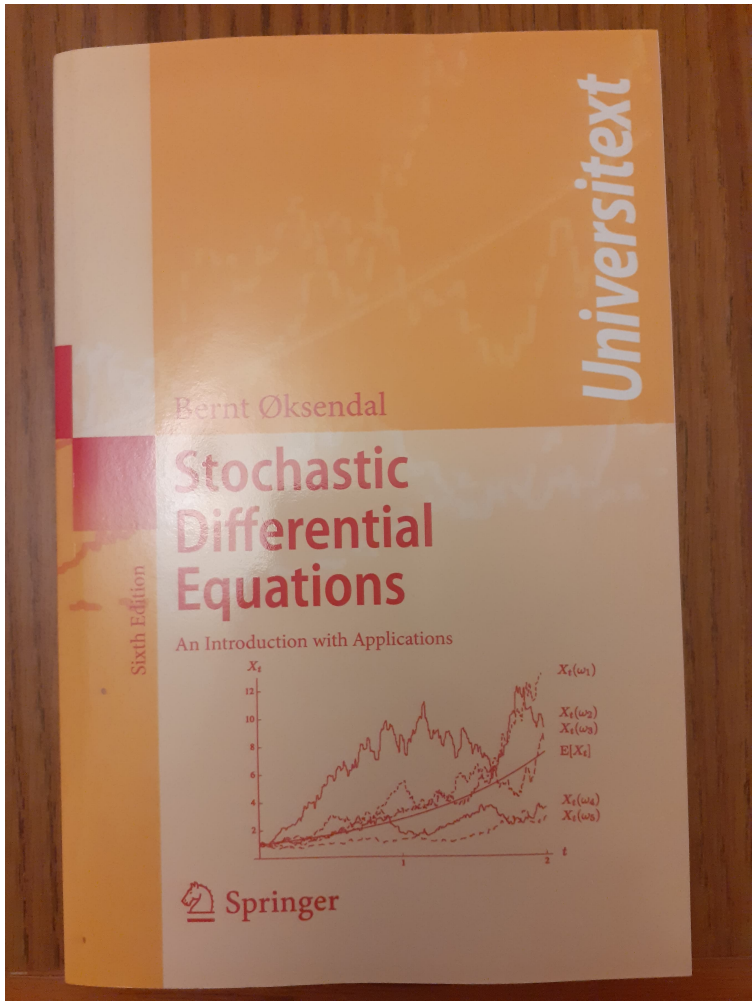
Implicit probabilistic models are a flexible class of models defined by a simulation process of data.

Example: Given initial conditions, physical simulators describe a stochastic process that generates data. In population ecology, the Lotka-Volterra model simulates predator-prey populations ($x_1, x_2 \in \mathbb{R}^+$) over time via a **stochastic differential equation**:

$$\frac{dx_1}{dt} = \beta_1 x_1 - \beta_2 x_1 x_2 + \epsilon_1, \qquad \epsilon_1 \sim \mathcal{N}(0, 10)$$

$$\frac{dx_2}{dt} = -\beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon_2, \qquad \epsilon_2 \sim \mathcal{N}(0, 10)$$

where Gaussian noises ϵ_1, ϵ_2 are added at each full time step.



Just bought this book

Proposals

1. **Hierarchical implicit models** (HIMs): combines the idea of implicit densities with hierarchical Bayesian modeling, and defines models via simulators of data.
2. **Likelihood-free variational inference** (LFVI): a scalable variational inference algorithm for HIMs.

Figure 1: (**left**) Hierarchical model, with local variables $\mathbf{z}$ and global variables β. (**right**) **Hierarchical implicit model**. It is a hierarchical model where $\mathbf{x}$ is a deterministic function (denoted with a square) of noise ϵ (denoted with a triangle).

Hierarchical models:

$$p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\beta}) = p(\boldsymbol{\beta}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\beta}) p(\boldsymbol{z}_n|\boldsymbol{\beta})$$

where $\boldsymbol{x}_n$ is an observation, $\boldsymbol{z}_n$ is the latent variable associated to that observation (local variables), and $\boldsymbol{\beta}$ is the latent variable shared across observations (global variable).

Hierarchical models typically use a tractable likelihood $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\beta})$. But many likelihoods of interest, such as simulator-based models, and generative adversarial networks do not admit a tractable likelihood. Thus, we develop **hierarchical implicit models** (HIMs) that define a function g that takes random noise $\boldsymbol{\epsilon}_n \sim s(.)$ and outputs $\boldsymbol{x}_n$, given $\boldsymbol{z}_n$ and $\boldsymbol{\beta}$:

$$\boldsymbol{x}_n = g(\boldsymbol{\epsilon}_n|\boldsymbol{z}_n, \boldsymbol{\beta}), \qquad \boldsymbol{\epsilon}_n \sim s(.)$$

The implicit likelihood of $\boldsymbol{x}_n \in A$ given $\boldsymbol{z}_n$ and $\boldsymbol{\beta}$ is:

$$\mathcal{P}(\boldsymbol{x}_n \in A | \boldsymbol{z}_n, \boldsymbol{\beta}) = \int_{\{g(\boldsymbol{\epsilon}_n | \boldsymbol{z}_n, \boldsymbol{\beta}) = \boldsymbol{x}_n \in A\}} s(\boldsymbol{\epsilon}_n) d\boldsymbol{\epsilon}_n$$

is typically intractable. Therefore, we need **likelihood-free variational inference** (LFVI).

The basic idea is to avoid computing the likelihood explicitly, and instead we compute the unbiased gradients with Monte Carlo. We skip the technical details here.

My observation

Many papers of David Blei share the same idea of local-global decomposition from **Latent Dirichlet Allocation** (LDA).

Paper 5

Deep Hierarchical Implicit Models

Dustin Tran, Rajesh Ranganath, David M. Blei

`https://arxiv.org/pdf/1702.08896v1.pdf`

Stacking layers of implicit latent variables produces a **deep implicit model** (DIM). This is basically a trivial extension of the previous one (same authors).

DIMs share the same local-global decomposition as hierarchical implicit models. The key difference is that instead of a single local variable $\boldsymbol{z}_n$, DIMs define layers of implicit local variables (This is the idea from Hierarchical VAE). Formally, for layers $1, .., T$, the local variables for data point $\boldsymbol{x}_n$ are generated via:

$$\boldsymbol{z}_{n,T} = g_T(\boldsymbol{\epsilon}_{n,T}|\boldsymbol{\beta}_T) \qquad \boldsymbol{\epsilon}_{n,T} \sim s(.)$$

$$\boldsymbol{z}_{n,t} = g_t(\boldsymbol{\epsilon}_{n,t}|\boldsymbol{z}_{n,t+1}, \boldsymbol{\beta}_t) \qquad \boldsymbol{\epsilon}_{n,t} \sim s(.) \qquad t < T$$

where g_t is the transformation for layer t, s is random noise, and $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, .., \boldsymbol{\beta}_T\}$ are parameters of the T transformations.

Deep Hierarchical Implicit Models (2)

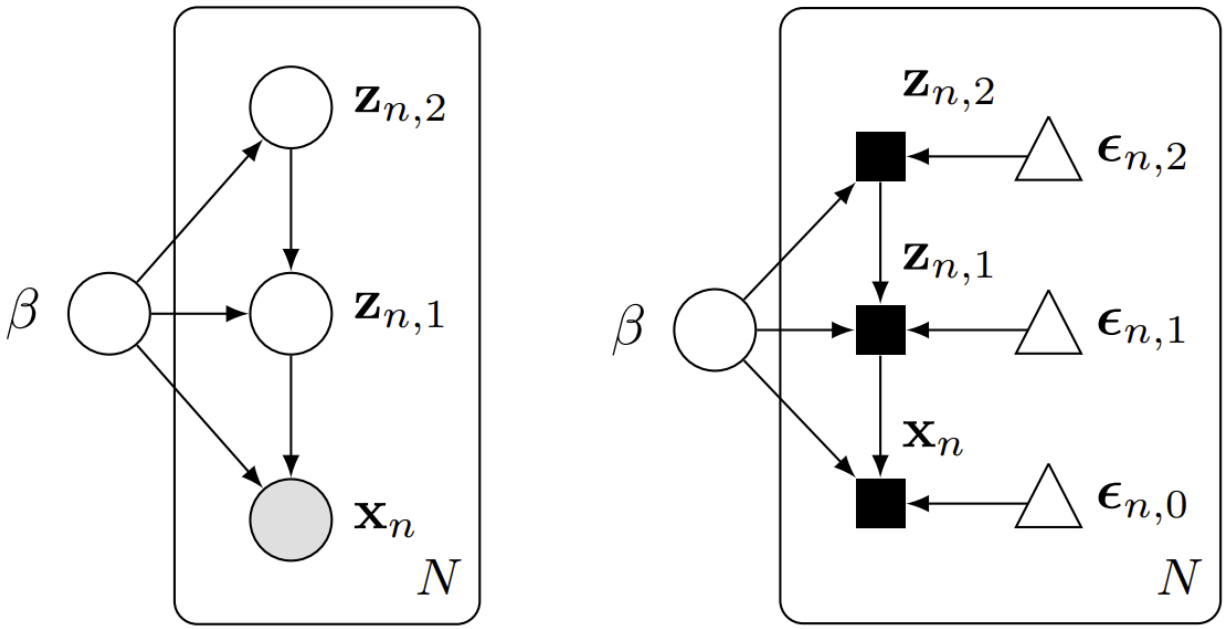


Figure 2. (**left**) Deep generative model with 2 layers $\{\mathbf{z}_1, \mathbf{z}_2\}$ and global variables β. (**right**) **Deep implicit model**. It is a deep generative model with implicit likelihood and implicit priors.

Paper 6

Discrete Object Generation with Reversible Inductive Construction

Ari Seff, Wenda Zhou, Farhan Damani, Abigail Doyle, Ryan P. Adams

`https://arxiv.org/pdf/1907.08268.pdf`

Proposal

Generative model (interpreted from denoising autoencoders) for discrete objects (e.g. molecules, source code, graphs, etc.) employing a **Markov chain** where transitions are restricted to a set of **local operations that preserve validity**:

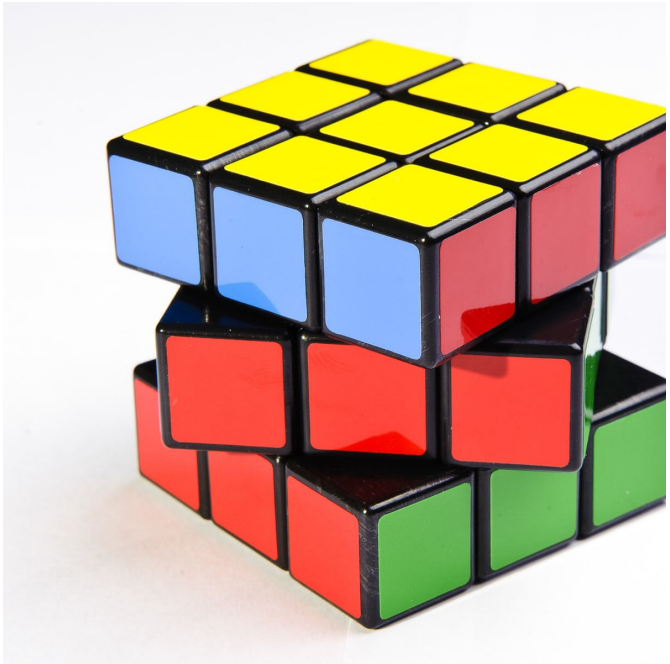
1. a sequence of corrupted objects that are valid but not from the data distribution.
2. a learned reconstruction distribution that attempts to fix the corruptions while also preserving validity.

My opinion

What is the difference between this and the **Reinforcement Learning** approach (Markov decision process) then?

The idea is pretty much similar to generating a solvable training example for **Rubik**:

1. Start from the original state x, we begin to corrupt it iteratively for k steps. Let say operations $s = [s_1, s_2, .., s_k]$ done, and result in state $\bar{x}$.
2. Each operation is reversible. The solution to solve state $\bar{x}$ back into the original state x is a reverse sequence $[s_k^{-1}, .., s_2^{-1}, s_1^{-1}]$.



Let $p(x)$ be an unknown probability mass function. Our goal is to learn a generative model $p_\theta(x)$, approximating $p(x)$. Let $c(\bar{x}|x)$ be a fixed corrupting distribution, and $p_\theta(x|\bar{x})$ be the learned reconstruction distribution:

$$p(x, \bar{x}) = c(\bar{x}|x)p(x)$$

We use supervised learning to train a reconstruction distribution model $p_\theta(x|\bar{x})$ to approximate $p(x|\bar{x})$.

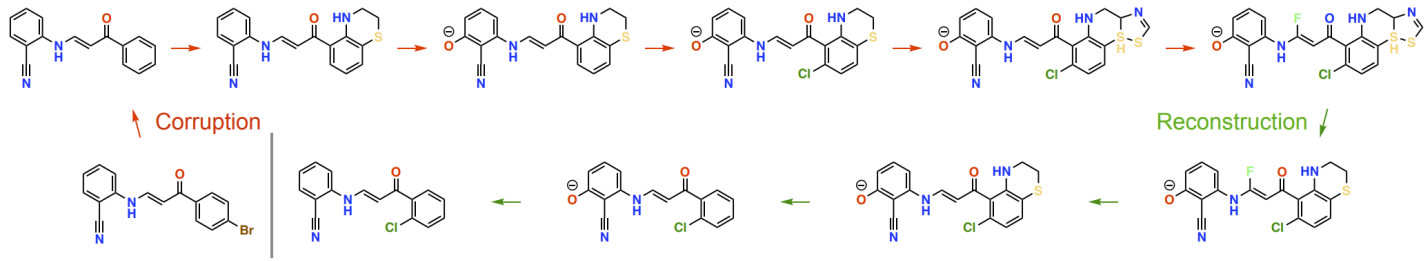


Figure 2: Corruption and subsequent reconstruction of a molecular graph. Our method generates discrete objects by running a Markov chain that alternates between sampling from fixed corruption and learned reconstruction distributions that respect validity constraints.

Discrete Object Generation with Reversible Inductive Construction (4)

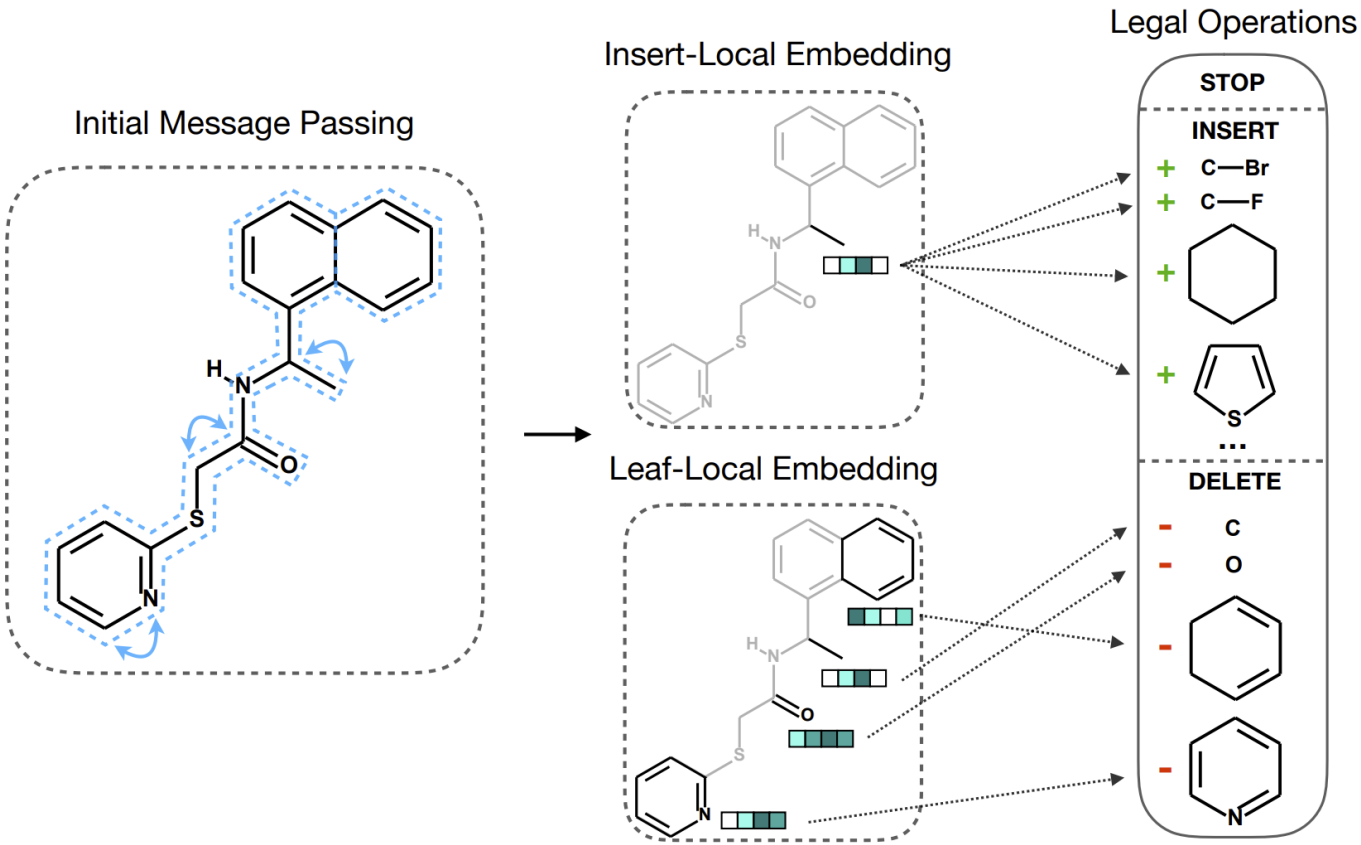


Figure 1: Reconstruction model processing given an input molecule. Location-specific representations computed via message passing are passed through fully-connected layers outputting probabilities for each legal operation.

The probability of arriving at corrupted sample $\bar{x}$ from x is:

$$c(\bar{x}|x) = \sum_{s} c(\bar{x}, s|x) = \sum_{s \in S(x,\bar{x})} c(s|x)$$

where $S(x, \bar{x})$ denotes the set of all corrupting sequences from x to $\bar{x}$, and $c(s|x)$ is a fixed conditional distribution over a sequence of corrupting operations $s = [s_1, .., s_k]$ in which $s_i \in \text{Ind}(\bar{x}_i)$ – the set of legal inductive moves for a given $\bar{x}_i$. The joint data-generating distribution:

$$p(x, s, \bar{x}) = c(\bar{x}, s|x)p(x)$$

where $c(\bar{x}, s|x) = 0$ if $s \notin S(x, \bar{x})$.

Optimization problem:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{p(x,s,\bar{x})} \mathcal{D}_{\mathrm{KL}}(p(s,x|\bar{x})||p_\theta(s,x|\bar{x}))$$

which is to maximize likelihood estimation $p_\theta(s,x|\bar{x})$ and likewise $p_\theta(x|\bar{x})$. For the expectation over the joint data-generating distribution $p(x,s,\bar{x})$, we can sample a corruption sequence as follows:

$$x \sim p(x)$$

$$\bar{x}, s \sim c(\bar{x}, s|x)$$

The learned reconstruction sequence model can be factorized (represented) as a product of memoryless transitions culminating with a stop token:

$$p_\theta(s^{-1}|\bar{x}) = p_\theta(\mathrm{stop}|x) p_\theta(x|\bar{x}_1) \prod_{i=1}^{k-1} p_\theta(\bar{x}_i|\bar{x}_{i+1})$$

where $s^{-1} = [s_k^{-1}, .., s_1^{-1}, \mathrm{stop}]$.