SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows

Didrik Nielsen[1], Priyank Jaini[2], Emiel Hoogeboom[2], Ole Winther[1], Max Welling[2]

Technical University of Denmark[1] UvA-Bosch Delta Lab, University of Amsterdam[2]

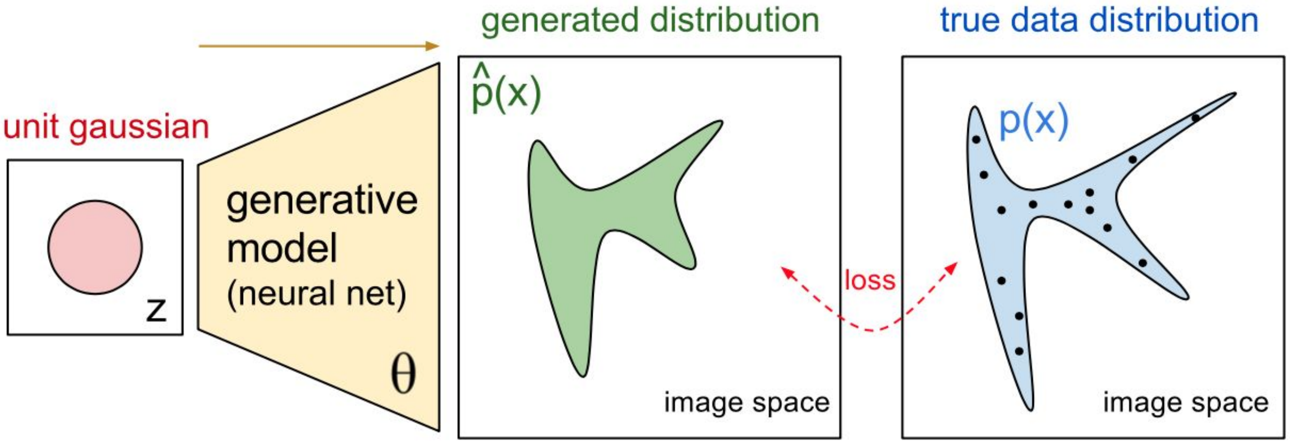
NeurIPS 2020

Presenters: **Keiran Paster, Andrew Li**

SurVAE: Motivation

Variational Autoencoders (VAEs) [Kingma & Welling, 2013] and **Normalizing Flows** [Rezende & Mohamed, 2015] are two distinct approaches to generative modelling.

- Transform a simple prior distribution $p(z)$ to a complex data distribution $p(x)$



https://openai.com/blog/generative-models/

SurVAE: Motivation

Main Ideas:

- *Surjective* transformations to “bridge the gap” between VAEs and Normalizing Flows

- A framework in which VAEs, Normalizing Flows, and surjections are composable layers

Variational Autoencoders

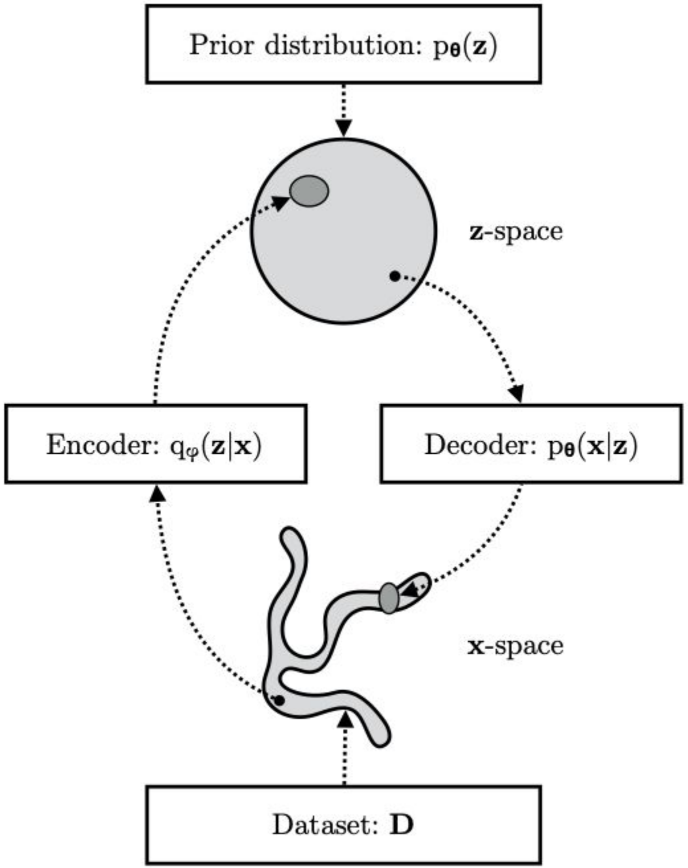
- Models a stochastic generative process:

$$\underbrace{z \sim p_\theta(z)}_{\textbf{Prior}},\ \underbrace{x \sim p_\theta(x|z)}_{\textbf{Decoder}}$$

- The posterior $p_\theta(z|x)$ involves an intractable integral and is approximated via a neural network:

$$\underbrace{q_\phi(z|x)}_{\textbf{Encoder}} \approx p_\theta(z|x)$$

Useful for optimization (to be shown)



[Kingma & Welling, 2019]

VAE Objective

- The goal is to maximize the likelihood of the data $\log p_\theta(x)$ but this is also intractable!
- Instead, a surrogate objective **(ELBO)** is used:

$$\log p_\theta(x) = \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]\right]}_{\textbf{ELBO}} + \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{q_\phi(z|x)}{p_\theta(z|x)}\right]\right]}_{D_{\mathrm{KL}}\left[q_\phi(z|x)\|p_\theta(z|x)\right]}$$

ELBO

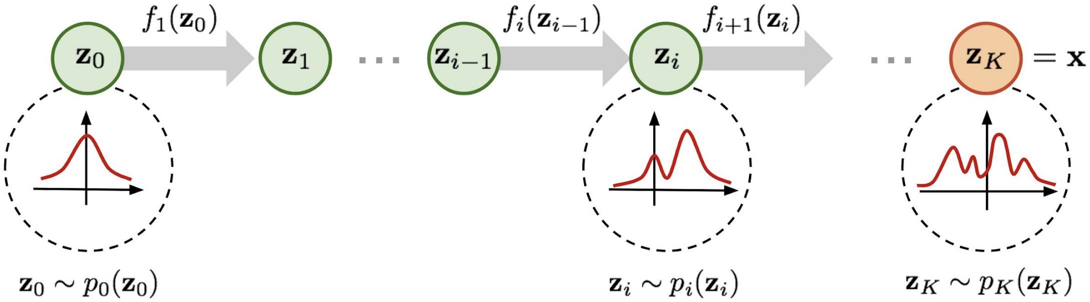
- Note that $$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]\right]$$ **(ELBO)**

  and the better the approximation $q_\phi(z|x) \approx p_\theta(z|x)$, the tighter the bound!

- **Main issue:** Many desirable quantities are intractable in the VAE framework, e.g. $p_\theta(x)$, $p_\theta(z|x)$

Normalizing Flows

- Transform a simple distribution $p(z)$ into a more complicated distribution by composing deterministic, invertible transformations (*bijections*)
- Obtain the exact log-probability of any x:
  - Use **change-of-variables** formula: $p(x) = p(z)|\det \nabla_x f^{-1}(x)|$
  - Optimize the model to maximize likelihood of the data
- Flow layers ideally are expressive, invertible, and have an easily computable Jacobian determinant.



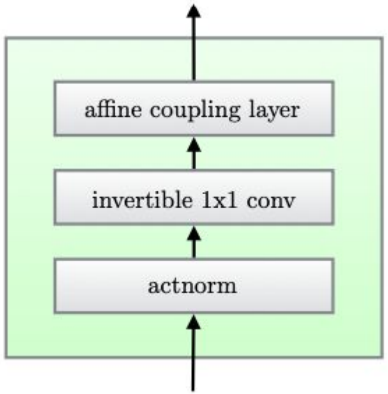
https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Normalizing Flow Layers

- Affine Coupling Layer (RealNVP - Dihn et al. 2017)
  - Input dimensions are split into two parts.

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})$$

  - Easy to invert and the Jacobian is convenient.
- Invertible 1x1 conv (Glow - Kingma and Dhariwal, 2018)

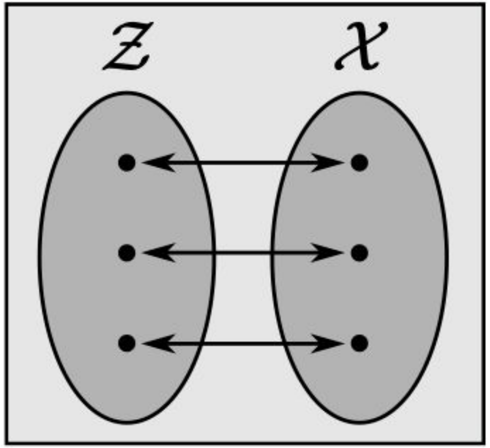


https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Issues with Normalizing Flows

- Transformations must be bijective.
  - Difficult to alter dimensionality
  - Issues mapping continuous latents to discrete data.

- Flow models still fall behind in image quality / log likelihood compared to other model types.

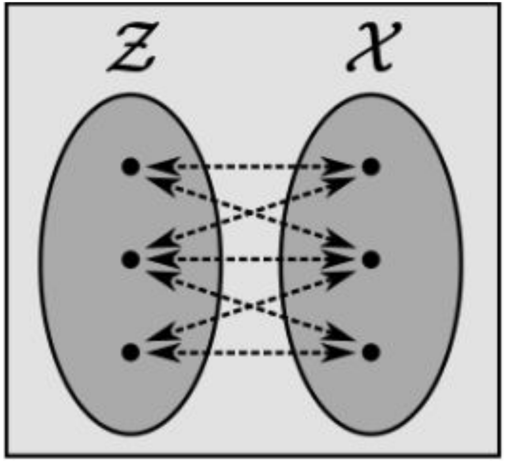


A bijective transformation. Figure from Nielsen et al 2020.

Comparison: VAEs and Flows

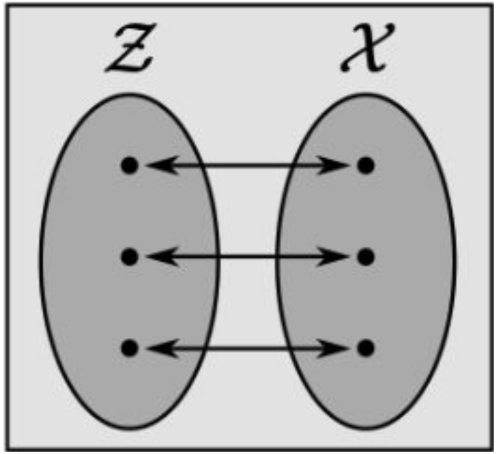
VAEs learn *stochastic* transformations
$\mathcal{Z} \to \mathcal{X}$ and $\mathcal{X} \to \mathcal{Z}$.

- Intractable likelihood $p_\theta(x)$.



Flows learn *deterministic bijections*
$\mathcal{Z} \to \mathcal{X}$ (and through inverting, $\mathcal{X} \to \mathcal{Z}$).

- Difficult to alter dimensionality



VAE
Very Deep VAE (Child, 2020)

Normalizing Flow
Glow (Kingma and Dhariwal, 2018)

SurVAE Flows

- Both VAEs and Flow models optimize the log likelihood of the data $\log p(x)$
  - Flow models optimize this likelihood exactly
  - VAEs optimize a lower bound (ELBO)
- Can we frame VAEs as a layer of a Flow model?

A Connection Between VAEs and Flows

VAE:

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{z})\right] + \underbrace{\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}|\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\right]}_{\text{Lik. contrib. } \mathcal{V}(\boldsymbol{x},\boldsymbol{z})} + \underbrace{\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right]}_{\text{Bound looseness } \mathcal{E}(\boldsymbol{x},\boldsymbol{z})}$$

Normalizing Flow:

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}) + \underbrace{\log \left|\det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right|}_{\text{Lik. contrib. } \mathcal{V}(\boldsymbol{x},\boldsymbol{z})}, \quad \boldsymbol{z} = f^{-1}(\boldsymbol{x})$$

(change-of-variables)

[Nielsen et al 2020]

A Connection Between VAEs and Flows

- VAEs learn stochastic mappings while Flows learn deterministic mappings $\mathcal{Z} \rightarrow \mathcal{X}$.

- Dirac delta-function:

$$\delta(x) = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$

- Can write a deterministic function $x = f(z)$ as a probability distribution with $p(x|z) = \delta(x - f(z))$

A Connection Between VAEs and Flows

- Let $p(x|z) = \delta(x - f(z))$

  $$p(z|x) = \delta(z - f^{-1}(x))$$

  $$q(z|x) = p(z|x)$$

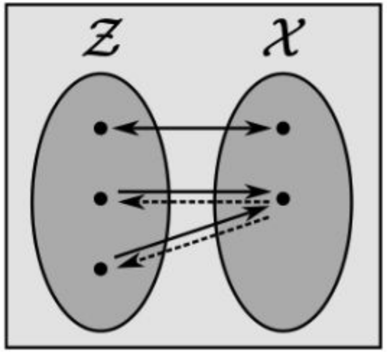
- Then $$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\overbrace{\log p(\boldsymbol{z}) + \log \frac{p(\boldsymbol{x}|\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}}^{\textbf{(ELBO)}} + \overbrace{\log \frac{q(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}}^{0}\right]$$

  $$= \log p(\boldsymbol{z}) + \log|\det \boldsymbol{J}|, \quad \text{for } \boldsymbol{z} = f^{-1}(\boldsymbol{x}),$$ **(change-of-variables)**

  where $\boldsymbol{J}^{-1} = \frac{\partial f(z)}{\partial z}\Big|_{z=f^{-1}(x)}$

Surjective Transformations

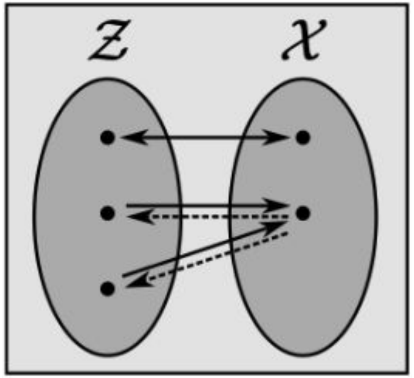
- Next we consider *surjective* transformations with properties of both VAEs and Flows.
- $f : \mathcal{Z} \to \mathcal{X}$ is surjective if every $x \in \mathcal{X}$ has a pre-image $z \in \mathcal{Z}$ such that $f(z) = x$
- Multiple inputs can map to the same output.



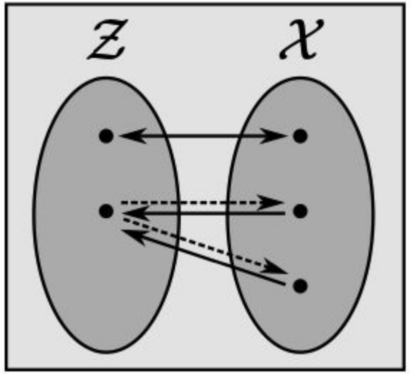
[Nielsen et al 2020]

Surjective Transformations

- Surjective transformations are *deterministic* forwards and can have *stochastic* inverses.
- Can also have surjections in the "inference" direction $\mathcal{X} \rightarrow \mathcal{Z}$.



Generative surjection



Inference surjection

SurVAE Flows

- We just saw how Flow models are a special case of VAEs with a certain distribution.
- SurVAE Flows give a general framework for computing likelihood for different choices of forward and inverse transformations.

$$\log p(x) \simeq \log p(z) + \mathcal{V}(x, z) + \mathcal{E}(x, z)$$

likelihood contribution

$$\mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)}{q(z|x)}\right]$$

bound looseness

$$\geq 0$$

Algorithm 1: $\text{log} - \text{likelihood}(\boldsymbol{x})$

Data: $\boldsymbol{x}$, $p(\boldsymbol{z})$ & $\{f_t\}_{t=1}^T$
Result: $\mathcal{L}(\boldsymbol{x})$
for t *in* range(T), **do**
  if f_t *is bijective* **then**
    $\boldsymbol{z} = f_t^{-1}(\boldsymbol{x})$;
    $\mathcal{V}_t = \log \left|\det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right|$;
  else if f_t *is stochastic* **then**
    $\boldsymbol{z} \sim q_t(\boldsymbol{z}|\boldsymbol{x})$;
    $\mathcal{V}_t = \log \frac{p_t(\boldsymbol{x}|\boldsymbol{z})}{q_t(\boldsymbol{z}|\boldsymbol{x})}$;
  $\boldsymbol{x} = \boldsymbol{z}$;
end
return $\log p(\boldsymbol{z}) + \sum_{t=1}^T \mathcal{V}_t$

Example: Rounding Surjection

- Consider the surjective transformation $x = \lfloor z \rfloor$
- The forward transformation is given by $P(x|z) = \mathbb{I}(x = \lfloor z \rfloor)$
- The backward transformation $q(z|x)$ is stochastic with support over $\{x + u | u \in [0, 1)^d\}$

Example: Rounding Surjection

- To find $\mathcal{V}(x, z)$, use $p(x|z) = \delta(x = \lfloor z \rfloor)$
- $$\mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)}{q(z|x)}\right] = \mathbb{E}_{q(z|x)}[\log p(x|z)] + \mathbb{E}_{q(z|x)}[-\log q(z|x)]$$

Expectation of log delta fn = 0

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

Related Work

Many well-known methods in the literature can be expressed as SurVAE Flows.

- ***Dequantization*** (Uria et al, 2013; Ho et al, 2019) is used to train continuous flows on *discrete* data.
  - Can be obtained via the *Rounding Surjection.*

Table 3: SurVAE Flows as a unifying framework.

Model	SurVAE Flow architecture
Probabilistic PCA (Tipping and Bishop, 1999) VAE (Kingma and Welling, 2014; Rezende et al., 2014) Diffusion Models (Sohl-Dickstein et al., 2015; Ho et al., 2020)	$\mathcal{Z} \xrightarrow{\text{stochastic}} \mathcal{X}$
Dequantization (Uria et al., 2013; Ho et al., 2019)	$\mathcal{Z} \xrightarrow{\text{round}} \mathcal{X}$
ANFs, VFlow (Huang et al., 2020; Chen et al., 2020)	$\mathcal{X} \xrightarrow{\text{augment}} \mathcal{X} \times \mathcal{E} \xrightarrow{\text{bijection}} \mathcal{Z}$
Multi-scale Architectures (Dinh et al., 2017)	$\mathcal{X} \xrightarrow{\text{bijection}} \mathcal{Y} \times \mathcal{E} \xrightarrow{\text{slice}} \mathcal{Y} \xrightarrow{\text{bijection}} \mathcal{Z}$
CIFs, Discretely Indexed Flows, DeepGMMs (Cornish et al., 2019; Duan, 2019; Oord and Dambre, 2015)	$\mathcal{X} \xrightarrow{\text{augment}} \mathcal{X} \times \mathcal{E} \xrightarrow{\text{bijection}} \mathcal{Z} \times \mathcal{E} \xrightarrow{\text{slice}} \mathcal{Z}$
RAD Flows (Dinh et al., 2019)	$\mathcal{X} \xrightarrow{\text{partition}} \mathcal{X}_{\mathcal{E}} \times \mathcal{E} \xrightarrow{\text{bijection}} \mathcal{Z} \times \mathcal{E} \xrightarrow{\text{slice}} \mathcal{Z}$

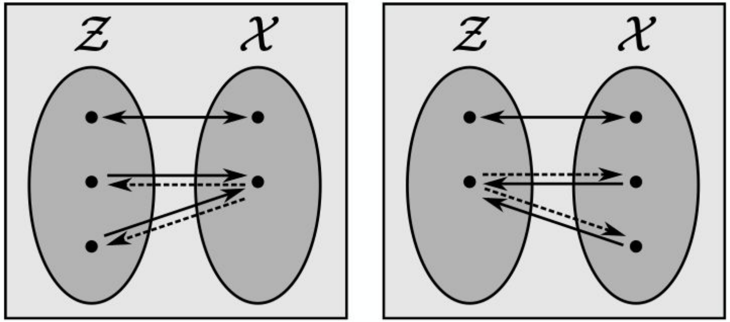
SurVAE Layers

Table 6: Summary of some generative surjection layers.

Surjection	**Forward**	**Inverse**	$\mathcal{V}(\boldsymbol{x}, \boldsymbol{z})$
Rounding	$x = \lfloor z \rfloor$	$z \sim q(z\|x)$ where $z \in [x, x+1)$	$-\log q(z\|x)$
Slicing	$\boldsymbol{x} = \boldsymbol{z}_1$	$\boldsymbol{z}_1 = \boldsymbol{x}, \boldsymbol{z}_2 \sim q(\boldsymbol{z}_2\|\boldsymbol{x})$	$-\log q(\boldsymbol{z}_2\|\boldsymbol{x})$
Abs	$s = \operatorname{sign} z$ $x = \|z\|$	$s \sim \operatorname{Bern}(\pi(x))$ $z = s \cdot x, \quad s \in \{1, -1\}$	$-\log q(s\|x)$
Max	$k = \arg\max \boldsymbol{z}$ $x = \max \boldsymbol{z}$	$k \sim \operatorname{Cat}(\boldsymbol{\pi}(x))$ $z_k = x, \boldsymbol{z}_{-k} \sim q(\boldsymbol{z}_{-k}\|x, k)$	$-\log q(k\|x) - \log q(\boldsymbol{z}_{-k}\|x, k)$
Sort	$\mathcal{I} = \operatorname{argsort} \boldsymbol{z}$ $\boldsymbol{x} = \operatorname{sort} \boldsymbol{z}$	$\mathcal{I} \sim \operatorname{Cat}(\boldsymbol{\pi}(\boldsymbol{x}))$ $\boldsymbol{z} = \boldsymbol{x}_{\mathcal{I}}$	$-\log q(\mathcal{I}\|\boldsymbol{x})$
ReLU	$x = \max(z, 0)$	if $x = 0 : z \sim q(z)$, else : $z = x$	$\mathbb{I}(x = 0)[-\log q(z)]$

Table 7: Summary of some inference surjection layers.

Surjection	**Forward**	**Inverse**	$\mathcal{V}(\boldsymbol{x}, \boldsymbol{z})$
Rounding	$x \sim p(x\|z)$ where $x \in [z, z+1)$	$z = \lfloor x \rfloor$	$\log p(z\|x)$
Slicing	$\boldsymbol{x}_1 = \boldsymbol{z}, \boldsymbol{x}_2 \sim p(\boldsymbol{x}_2\|\boldsymbol{z})$	$\boldsymbol{z} = \boldsymbol{x}_1$	$\log p(\boldsymbol{x}_2\|\boldsymbol{z})$
Abs	$s \sim \operatorname{Bern}(\pi(z))$ $x = s \cdot z, \quad s \in \{-1, 1\}$	$s = \operatorname{sign} x$ $z = \|x\|$	$\log p(s\|z)$
Max	$k \sim \operatorname{Cat}(\boldsymbol{\pi}(z))$ $x_k = z, \boldsymbol{x}_{-k} \sim p(\boldsymbol{x}_{-k}\|z, k)$	$k = \arg\max \boldsymbol{x}$ $z = \max \boldsymbol{x}$	$\log p(k\|z) + \log p(\boldsymbol{x}_{-k}\|z, k)$
Sort	$\mathcal{I} \sim \operatorname{Cat}(\boldsymbol{\pi}(\boldsymbol{z}))$ $\boldsymbol{x} = \boldsymbol{z}_{\mathcal{I}}$	$\mathcal{I} = \operatorname{argsort} \boldsymbol{x}$ $\boldsymbol{z} = \operatorname{sort} \boldsymbol{x}$	$\log p(\mathcal{I}\|\boldsymbol{z})$
ReLU	if $z = 0 : x \sim p(x)$, else : $x = z$	$z = \max(x, 0)$	$\mathbb{I}(z = 0) \log p(x)$



(b) Surjective (Gen.) (c) Surjective (Inf.)

Generative Surjection: stochastic $q(z|x)$

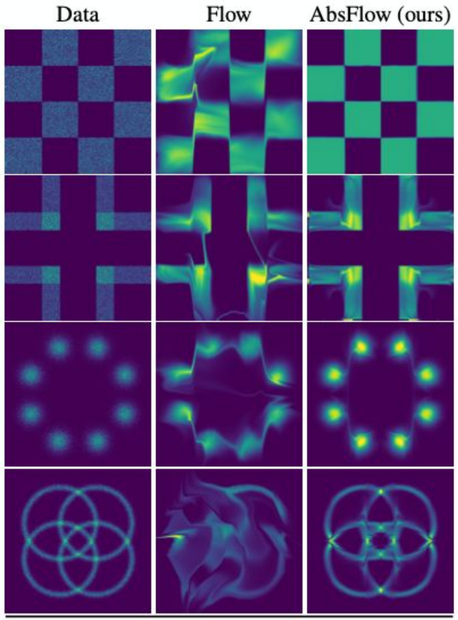
Inference Surjection: stochastic $p(x|z)$

Note: inference surjections have $\mathcal{E}(x, z) = 0$ i.e. exact likelihood.

Experiments - Symmetrical Synthetic Data

- Using an *absolute value inference surjection* improves the modeling of data with symmetries.

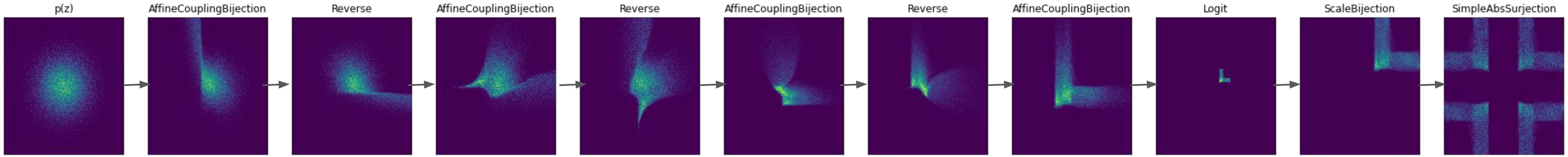
- Normalizing flows have trouble modelling disconnected structure in the data.



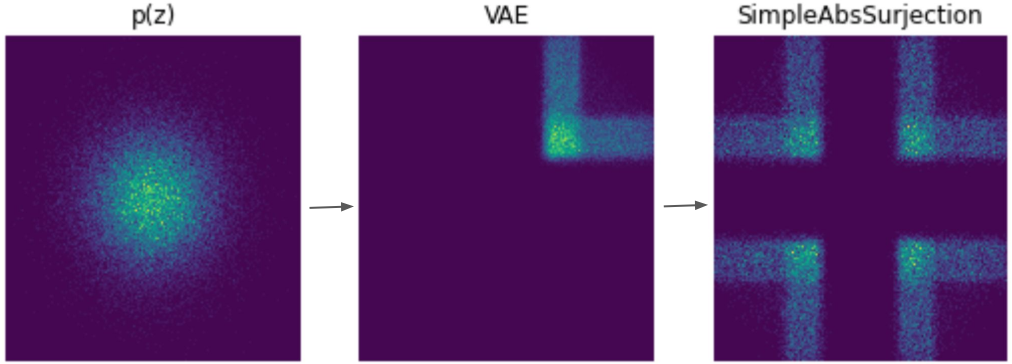
Dataset	Flow	AbsFlow (ours)
Checkerboard	3.65	**3.49**
Corners	3.19	**3.03**
Gaussians	3.01	**2.86**
Circles	3.44	**2.99**

2D Visualization (Code Notebook)

- Normalizing flow combined with an absolute value surjection (last layer).



2D Visualization (Code Notebook)



- Composing a VAE layer and an abs. (inference) surjection.

Experiments - MaxPoolFlow for Image Data

MaxPooling is used to downscale from the image dimension to a smaller latent dimension.

Adding MaxPooling yields worse log-likelihoods but better inception and FID scores.

Model	Inception ↑	FID ↓
DCGAN*	6.4	37.1
WGAN-GP*	6.5	36.4
PixelCNN*	4.60	65.93
PixelIQN*	5.29	49.46
Baseline (Ours)	5.08	49.56
MaxPoolFlow (Ours)	**5.18**	**49.03**

Table 5: Inception score and FID for CIFAR-10. *Results taken from Ostrovski et al. (2018).

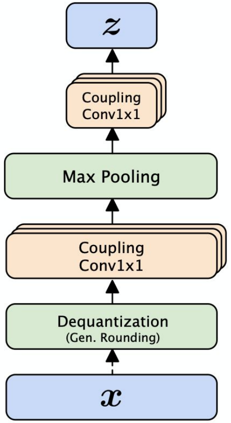


Figure 6: Flow architecture with max pooling. Surjections in green.

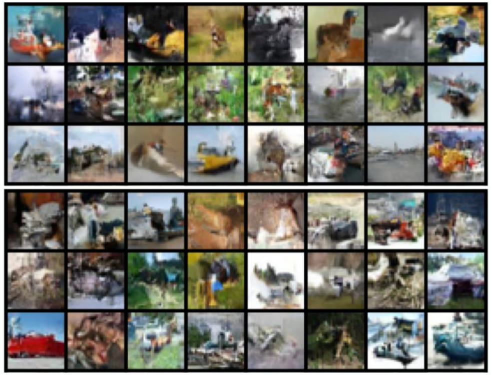


Figure 7: Samples from CIFAR-10 models. Top: MaxPoolFlow, Bottom: Baseline.

Summary

- SurVAE Flows is a framework in which VAEs and Normalizing Flows are modular layers.
    - Encompasses several additional works in the literature, e.g. *Dequantization*
- Surjective layers are introduced as a “bridge” between VAEs and Normalizing Flows
    - Can alter dimensionality
    - (In some cases) can compute exact likelihood
- Several novel, practical surjective layers are derived and introduced.
    - e.g. *MaxPool, AbsoluteValue, Sort*
    - Allows greater flexibility in designing Flow architectures.
    - **Limitation:** most of the introduced surjections are targeted to very specific functions and require extensive domain knowledge to be applied.

References

1. Nielsen, D., Jaini, P., Hoogeboom, E., Winther, O., & Welling, M. (2020). Survae flows: Surjections to bridge the gap between vaes and flows. *In Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS).*
2. Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *In Proceedings of the 2nd International Conference on Learning Representations (ICLR).*
3. Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning* (pp. 1530-1538). PMLR.
4. Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.
5. Child, R. (2020). Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. *arXiv preprint arXiv:2011.10650*.
6. Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real NVP. *In Proceedings of the 5th International Conference on Learning Representations (ICLR)*.
7. Kingma, D. P., & Dhariwal, P. (2018). Glow: generative flow with invertible 1× 1 convolutions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (pp. 10236-10245).
8. Uria, B., Murray, I., & Larochelle, H. (2013). RNADE: The real-valued neural autoregressive density-estimator. In *Proceedings of the 27th Conference on Advances in Neural Information Processing Systems (NeurIPS).*
9. Ho, J., Chen, X., Srinivas, A., Duan, Y. & Abbeel, P.. (2019). Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. *Proceedings of the 36th International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 97:2722-2730.