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

Nielsen, D.; Jaini, P.; Hoogeboom, E.; Winther, O.; Welling, M.

Publication date
2021

Document Version
Author accepted manuscript

Published in

Citation for published version (APA):
https://papers.nips.cc/paper/2020/hash/9578a63fbe545bd82cc5bfe749636af1-Abstract.html
Abstract

Normalizing flows and variational autoencoders are powerful generative models that can represent complicated density functions. However, they both impose constraints on the models: Normalizing flows use bijective transformations to model densities whereas VAEs learn stochastic transformations that are non-invertible and thus typically do not provide tractable estimates of the marginal likelihood. In this paper, we introduce SurV AE Flows: A modular framework of composable transformations that encompasses VAEs and normalizing flows. SurV AE Flows bridge the gap between normalizing flows and VAEs with surjective transformations, wherein the transformations are deterministic in one direction — thereby allowing exact likelihood computation, and stochastic in the reverse direction — hence providing a lower bound on the corresponding likelihood. We show that several recently proposed methods, including dequantization and augmented normalizing flows, can be expressed as SurV AE Flows. Finally, we introduce common operations such as the max value, the absolute value, sorting and stochastic permutation as composable layers in SurV AE Flows.

1 Introduction

Normalizing flows (Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rezende and Mohamed, 2015) provide a powerful modular and composable framework for representing expressive probability densities via differentiable bijections (with a differentiable inverse). These composable bijective transformations accord significant advantages due to their ability to be implemented using a modular software framework with a general interface consisting of three important components: (i) a forward transform, (ii) an inverse transform, and (iii) a log-likelihood contribution through the Jacobian determinant. Thus, significant advances have been made in recent years to develop novel flow modules that are easily invertible, expressive and computationally cheap (Dinh et al., 2015, 2017; Kingma et al., 2016; Papamakarios et al., 2017; Huang et al., 2018; Jaini et al., 2019a,b; Kingma and Dhariwal, 2018; Hoogeboom et al., 2019b, 2020; Durkan et al., 2019; van den Berg et al., 2018).

However, the bijective nature of the transformations used for building normalizing flows limit their ability to alter dimensionality, model discrete data and distributions with discrete structure or disconnected components. Specialized solutions have been developed to address these limitations independently. Uria et al. (2013); Ho et al. (2019) use dequantization to model discrete distributions using continuous densities, while Tran et al. (2019); Hoogeboom et al. (2019a) propose a discrete analog of normalizing flows. Cornish et al. (2019) use an augmented space to model an infinite mixtures of normalizing flows to address the problem of disconnected components whereas Huang et al. (2020); Chen et al., 2020 use a similar idea of augmentation of the observation space to model expressive distributions. VAEs (Kingma and Welling, 2014; Rezende et al., 2014), on the other hand, have no such limitations, but only provide lower bound estimates of the tractable estimates for the
We demonstrate the efficacy of SurV AE Flows with experiments on synthetic datasets, point cloud data, and images. Code to implement SurV AE Flows and reproduce results is publicly available.\footnote{The code is available at https://github.com/didriknielsen/survae_flows}

In this paper, we answer this affirmatively by introducing SurV AE Flows that use surjections to provide a unified, composable, and modular framework for probabilistic modeling. We introduce our unifying framework in §2 by identifying the components necessary to build composable architectures with modular software implementation for probabilistic modeling. We then introduce surjections for probabilistic modeling in §3 and show that these transformations lie at the interface between VAEs (stochastics maps) and normalizing flows (bijective maps). We unify these transformations (bijections, surjections, and stochastic transformations) in a composable and modular framework that we call SurV AE Flows. Subsequently, in §3.1, we propose novel SurV AE Flow layers like V AEs (stochastics maps) and normalizing flows (bijective maps). We unify these transformations (bijective, surjective, and stochastic transformations) in a composable and modular framework that we call SurV AE Flows. Subsequently, in §3.1, we propose novel SurV AE Flow layers like max value used for max pooling layers, absolute value for modeling symmetries in the data, and sorting and stochastic permutations that can be used for modeling exchangeable data and order statistics. Finally, in §3.2 we connect SurV AE Flows to several aforementioned specialised models by expressing them using SurV AE Flow layers which can now be implemented easily using our modular implementation. We demonstrate the efficacy of SurVAE Flows with experiments on synthetic datasets, point cloud data, and images. Code to implement SurVAE Flows and reproduce results is publicly available.\footnote{We note that Wu et al. (2020) also considered stochastic maps in flows using MCMC transition kernels.}

2 Preliminaries and Setup

In this section, we set up our main problem, provide key notations and definitions, and formulate a unifying framework for using different kinds of transformations to model distributions.

Let \( x \in \mathcal{X} \) and \( z \in \mathcal{Z} \) be two variables with distributions \( p(x) \) and \( p(z) \). We call a deterministic mapping \( f : \mathcal{Z} \to \mathcal{X} \) bijective if it is both surjective and injective. A mapping is surjective if \( \forall x \in \mathcal{X}, \exists z \in \mathcal{Z} \) such that \( x = f(z) \). A mapping is injective if \( \forall z_1, z_2 \in \mathcal{Z}, f(z_1) = f(z_2) \implies z_1 = z_2 \).

If the mapping is not deterministic, we refer to it as a stochastic mapping, and denote it as \( z \sim p(x|z) \).

Normalizing flows (Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rezende and Mohamed, 2015) make use of bijective transformations \( f \) to transform a simple base density \( p(z) \) to a more expressive density \( p(x) \), making use of the change-of-variables formula \( p(x) = p(z) | \det \nabla_{x} f^{-1}(x) | \). VAEs (Kingma and Welling, 2014; Rezende et al., 2014), on the other hand, define a probabilistic graphical model where each observed variable \( x \) has an associated latent variable \( z \) with the generative process as \( z \sim p(z), x \sim p(x|z) \), where \( p(x|z) \) may be viewed as a stochastic transformation. VAEs use variational inference with an amortized variational distribution \( q(z|x) \) to approximate the intractable posterior \( p(z|x) \) which facilitates computation of a lower bound of \( p(x) \) known as the evidence lower bound (ELBO) i.e., \( \mathcal{L} := \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathbb{D}_{KL}[q(z|x)||p(z)] \).

In the following, we introduce a framework to connect flows and VAEs\footnote{We note that Wu et al. (2020) also considered stochastic maps in flows using MCMC transition kernels.} by showing that bijective and stochastic transformations are composable and require three important components for use in probabilistic modeling: (i) a forward transformation, \( f : \mathcal{Z} \to \mathcal{X} \) with an associated conditional probability \( p(x|z) \), (ii) an inverse transformation, \( f^{-1} : \mathcal{X} \to \mathcal{Z} \) with an associated distribution \( q(z|x) \), and (iii) a likelihood contribution term used for log-likelihood computation.

![Figure 1: Classes of SurV AE transformations \( \mathcal{Z} \to \mathcal{X} \) and their inverses \( \mathcal{X} \to \mathcal{Z} \). Solid lines indicate deterministic transformations, while dashed lines indicate stochastic transformations.](image)
Table 1: Composable building blocks of SurVAE Flows.

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Forward $x \leftarrow z$</th>
<th>Inverse $z \leftarrow x$</th>
<th>Likelihood Contribution $\mathcal{V}(x, z)$</th>
<th>Bound Gap $\mathcal{E}(x, z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bijective</td>
<td>$x = f(z)$</td>
<td>$z = f^{-1}(x)$</td>
<td>$\log</td>
<td>\det \nabla_z x</td>
</tr>
<tr>
<td>Stochastic</td>
<td>$x \sim p(x</td>
<td>z)$</td>
<td>$z \sim q(z</td>
<td>x)$</td>
</tr>
<tr>
<td>Surjective (Gen.)</td>
<td>$x = f(z)$</td>
<td>$z \sim q(z</td>
<td>x)$</td>
<td>$\log \frac{p(x</td>
</tr>
<tr>
<td>Surjective (Inf.)</td>
<td>$x \sim p(x</td>
<td>z)$</td>
<td>$z = f^{-1}(x)$</td>
<td>$\log \frac{p(x</td>
</tr>
</tbody>
</table>

**Forward Transformation:** For a stochastic transformation, the forward transformation is the conditional distribution $p(x|z)$. For a bijective transformation, the forward transformation is deterministic and therefore, $p(x|z) = \delta(x - f(z))$ or simply $x = f(z)$.

**Inverse Transformation:** For a bijective transformation, the inverse is also deterministic and given by $z = f^{-1}(x)$. For a stochastic transformation, the inverse is also stochastic and is defined by Bayes theorem $p(z|x) = p(x|z)p(z)/p(x)$. Computing $p(z|x)$ is typically intractable and we thus resort to a variational approximation $q(z|x)$.

**Likelihood Contribution:** For bijections, the density $p(x)$ can be computed from $p(z)$ and the mapping $f$ using the change-of-variables formula as:

$$\log p(x) = \log p(z) + \log |\det J|, \quad z = f^{-1}(x)$$  \hspace{1cm} (1)

where $|\det J| = |\det \nabla_x f^{-1}(x)|$ is the absolute value of the determinant of the Jacobian matrix of $f^{-1}$ which defines the likelihood contribution term for a bijective transformation $f$. For stochastic transformations, we can rewrite the marginal density $p(x)$ as:

$$\log p(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathbb{D}_{KL}[q(z|x)||p(z|x)] + \mathbb{D}_{KL}[q(z|x)||p(z|x)]$$  \hspace{1cm} (2)

The ELBO $\mathcal{L}$ in Eq. 2 can then be evaluated using a single Monte Carlo sample: $\mathcal{L} \approx \log p(z) + \log \frac{p(x|z)}{q(z|x)}$, $z \sim q(z|x)$. Therefore, the likelihood contribution term for a stochastic transformation is defined as $\log \frac{p(x|z)}{q(z|x)}$. Furthermore, we show in App. A that Eq. 2 allows us to recover the change-of-variables formula given in Eq. 1 by using Dirac delta functions, thereby drawing a precise connection between VAEs and normalizing flows. Crucially, Eq. 2 helps us to reveal a unified modular framework to model a density $p(x)$ under any transformation by restating it as:

$$\log p(x) \simeq \log p(z) + \mathcal{V}(x, z) + \mathcal{E}(x, z), \quad z \sim q(z|x)$$  \hspace{1cm} (3)

where $\mathcal{V}(x, z)$ and $\mathcal{E}(x, z)$ are the likelihood contribution and bound looseness terms, respectively. The likelihood contribution is $\mathcal{V}(x, z) = \log |\det J|$ for bijections and $\log \frac{p(x|z)}{q(z|x)}$ for stochastic transformations. For bijections, likelihood evaluation is deterministic and exact with $\mathcal{E}(x, z) = 0$, while for stochastic maps it is stochastic and unbiased with a bound looseness of $\mathcal{E}(x, z) = \log \frac{q(z|x)}{p(z|x)}$. This is summarized in Table 1. The first term in Eq. 3, $\log p(z)$, reveals the compositional nature of the transformations, since it can be modeled by further transformations. While the compositional structure has been used widely for bijective transformations, Eq. 3 demonstrates its viability for stochastic maps as well. We demonstrate this unified compositional structure in Alg. 1.

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

Data: $x, p(z) \& \{f_t\}_{t=1}^T$

Result: $\mathcal{L}(x)$

for $t$ in range($T$), do

if $f_t$ is bijective then

$z = f_t^{-1}(x)$;
$V_t = \log |\det \frac{\partial z}{\partial x}|$;
else if $f_t$ is stochastic then

$z \sim q_t(z|x)$;
$V_t = \log \frac{p_t(z|x)}{q_t(z|x)}$;
end

return $\log p(z) + \sum_{t=1}^T V_t$
3 SurVAE Flows

As explained in Section 2, bijective and stochastic transformations provide a modular framework for constructing expressive generative models. However, they both impose constraints on the model: bijective transformations are deterministic and allow exact likelihood computation, but they are required to preserve dimensionality. On the other hand, stochastic transformations are capable of altering the dimensionality of the random variables but only provide a stochastic lower bound estimate of the likelihood. Is it possible to have composable transformations that can alter dimensionality and allow exact likelihood evaluation? In this section, we answer this question affirmatively by introducing surjective transformations as SurVAE Flows that bridge the gap between bijective and stochastic transformations.

In the following, we will define composable deterministic transformations that are surjective and non-injective. For brevity, we will refer to them as surjections or surjective transformations. Note that for surjections, multiple inputs can map to the same output, resulting in a loss of information since the input is not guaranteed to be recovered through inversion. Similar to bijective and stochastic transformations, the three important components of composable surjective transformations are:

**Forward Transformation**: Like bijections, surjective transformations have a deterministic forward transformation \( p(x|z) = \delta(x - f(z)) \) or \( x = f(z) \).

**Inverse Transformation**: In contrast with bijections, surjections \( f : \mathcal{Z} \rightarrow \mathcal{X} \) are not invertible since multiple inputs can map to the same output. However, they have right inverses, i.e. functions \( g : \mathcal{X} \rightarrow \mathcal{Z} \) such that \( f \circ g(x) = x \), but not necessarily \( g \circ f(z) = z \). We will use a stochastic right inverse \( q(z|x) \) which can be thought of as passing \( x \) through a random right inverse \( g \). Importantly, \( q(z|x) \) only has support over the preimage of \( x \), i.e. the set of \( z \) that map to \( x \), \( B(x) = \{ z | x = f(z) \} \).

So far, we have described what we will term generative surjections, i.e. transformations that are surjective in the generative direction \( \mathcal{Z} \rightarrow \mathcal{X} \). We will refer to a transformation which is surjective in the inference direction \( \mathcal{X} \rightarrow \mathcal{Z} \) as an inference surjection. These are illustrated in Fig.1. Generative surjections have stochastic inverse transformations \( q(z|x) \), while inference surjections have stochastic forward \( p(x|z) \) transformations.

**Likelihood Contribution**: For continuous surjections, the likelihood contribution term is:

\[
\mathbb{E}_{q(z|x)} \left[ \log \frac{p(x|z)}{q(z|x)} \right] , \quad \text{as} \quad \begin{cases} p(x|z) \rightarrow \delta(x - f(z)), \quad \text{for gen. surjections.} \\ q(z|x) \rightarrow \delta(z - f^{-1}(x)), \quad \text{for inf. surjections.} \end{cases}
\]

While generative surjections generally give rise to stochastic estimates of the likelihood contribution and introduce lower bound likelihood estimates, inference surjections allow exact likelihood computation (see App. B). Before proceeding further, we give a few examples to better understand the construction of a surjective transformation for probabilistic modeling.

**Example 1 (Tensor slicing)** Let \( f \) be a tensor slicing surjection that takes input \( z = (z_1, z_2) \in \mathbb{R}^d \), and returns a subset of the elements, i.e. \( x = f(z) = z_1 \). To develop this operation as a SurVAE layer, we first specify the stochastic forward and inverse transformations as:

\[
p(x|z) = \mathcal{N}(x|z_1, \sigma^2 I), \quad \text{and} \quad q(z|x) = \mathcal{N}(z_1|x, \sigma^2 I)q(z_2|x)
\]

We next compute the likelihood contribution term in the limit that \( p(x|z) \rightarrow \delta(x - f(z)) \). Here, this corresponds to \( \sigma \rightarrow 0 \). Thus,

\[
\mathcal{Y}(x, z) = \lim_{\sigma^2 \rightarrow 0} \mathbb{E}_{q(z|x)} \left[ \log \frac{p(x|z)}{q(z|x)} \right] = \mathbb{E}_{q(z_2|x)} \left[ -\log q(z_2|x) \right] ,
\]

which corresponds to the entropy of \( q(z_2|x) \) that is used to infer the sliced elements \( z_2 \). We illustrate the slicing surjection for both the generative and inference directions in Fig. 2.

**Example 2 (Rounding)** Let \( f \) be a rounding surjection that takes an input \( z \in \mathbb{R}^d \) and returns the rounded \( \hat{x} := \lfloor z \rfloor \). The forward transformation is a discrete surjection \( P(x|z) = \delta(z \in B(x)) \), for \( B(x) = \{ x + u | u \in [0,1]^d \} \). The inverse transformation \( q(z|x) \) is stochastic with support in \( B(x) \). Inserting this in the likelihood contribution term and simplifying, we find

\[
\mathcal{Y}(x, z) = \mathbb{E}_{q(z|x)} \left[ -\log q(z|x) \right] .
\]

This generative rounding surjection gives rise to dequantization (Uria et al., 2013; Ho et al., 2019) which is a method commonly used to train continuous flows on discrete data such as images.
We provide a summary of these in Table 2. Due to space constraints, we defer the derivations and details on each of these surjections to Appendix D-G along with detailed tables on generative and inference surjection layers including the stochastic forward proceeds by (i) sampling an index $k$ and setting $x_k = z$, and (ii) imputing the remaining values $x_{-k}$ of $x$ such that they are all smaller than $x_k$. Abs surjections are useful for modelling data with symmetries which may not be injective. We demonstrate the use of max surjections for probabilistic modelling of images.

### 3.1 Novel SurVAE Layers

We developed the tensor slicing and rounding surjections in Examples 1 and 2. In this section, we introduce additional novel SurVAE layers including the absolute value, the maximum value and sorting as surjective layers and stochastic permutation as a stochastic layer. We provide a summary of these in Table 2. Due to space constraints, we defer the derivations and details on each of these surjections to Appendix D-G along with detailed tables on generative and inference surjections in Table 6 and 7.

#### Abs Surjection (App. D)

The abs surjection returns the magnitude of its input, $z = |x|$. As a SurVAE layer, we can represent the abs surjection with the forward and inverse transformations as:

$$
p(x|z) = \sum_{s \in \{-1, 1\}} p(x, s)p(s|z) = \sum_{s \in \{-1, 1\}} \delta(x - sz)p(s|z),
$$

$$
q(z|x) = \sum_{s \in \{-1, 1\}} q(z|x, s)p(s|x) = \sum_{s \in \{-1, 1\}} \delta(z - sz)\delta_{s \text{sign}(x)}
$$

where $q(z|x)$ is deterministic corresponding to $z = |x|$. The forward transformation $p(x|z)$ involves the following steps: (i) sample the sign $s$, conditioned on $z$, and (ii) apply the sign to $z$ to obtain $x = sz$. Abs surjections are useful for modelling data with symmetries which we demonstrate in our experiments.

#### Max Surjection (App. E, Fig. 2)

The max operator returns the largest element of an input vector, $z = \max x$. We can represent this transformation as

$$
p(x|z) = \sum_{k=1}^{K} p(x|z, k)p(k|z) = \sum_{k=1}^{K} \delta(x_k - z)p(x_{-k}|z, k)p(k|z),
$$

$$
q(z|x) = \sum_{k=1}^{K} q(z|x, k)q(k|x) = \sum_{k=1}^{K} \delta(z - x_k)\delta_{k \text{arg max}(x)},
$$

where $q(z|x)$ is deterministic and corresponds to $z = \max x$. While the inverse is deterministic, the stochastic forward proceeds by (i) sampling an index $k$ and setting $x_k = z$, and (ii) imputing the remaining values $x_{-k}$ of $x$ such that they are all smaller than $x_k$. Max surjections are useful in implementing the max pooling layer commonly used in convolutional architectures for downsampling. In our experiments, we demonstrate the use of max surjections for probabilistic modelling of images.

#### Sort Surjection (App. F)

Sorting, $z = \text{sort}(x)$ returns a vector in sorted order. It is a surjective (and non-injective) transformation since the original order of the vector is lost in the operation even

<table>
<thead>
<tr>
<th>Surjection</th>
<th>Forward</th>
<th>Inverse</th>
<th>$V(x, z)$</th>
</tr>
</thead>
</table>
| Abs        | $s \sim \text{Bern}(\pi(z))$
  $x = s \cdot z, \ s \in \{-1, 1\}$ | $s = \text{sign } x$
  $z = |x|$ | log $p(s|z)$ |
| Max        | $k \sim \text{Cat}(\pi(z))$
  $x_k = z, x_{-k} \sim p(x_{-k}|z, k)$ | $k = \text{arg max } x$
  $z = \max x$ | log $p(k|z) + p(x_{-k}|z, k)$ |
| Sort       | $\mathcal{I} \sim \text{Cat}(\pi(z))$
  $x = z_{\mathcal{I}}$ | $\mathcal{I} = \text{argsort } x$
  $z = \text{sort } x$ | log $p(\mathcal{I}|z)$ |
though the dimensions remain the same. Sort surjections are useful in modelling naturally sorted data, learning order statistics, and learning an exchangeable model using flows.

**Stochastic Permutation** (App. G). A stochastic permutation transforms the input vector by shuffling the elements randomly. The inverse pass for a permutation is the same as the forward pass with the likelihood contribution term equal to zero, \( V = 0 \). Stochastic permutations help to enforce permutation invariance i.e. any flow can be made permutation invariant by adding a final permutation SurVAE layer. In our experiments, we compare sorting surjections and stochastic permutations to enforce permutation invariance for modelling exchangeable data.

**Stochastic Inverse Parameterization.** For surjections, the stochastic inverses have to be defined so that they form a distribution over the possible right-inverses. Different right-inverse distributions do not have to align over subsets of the space. Consequently, for more sophisticated choices of right-inverses, the log-likelihood may be discontinuous across boundaries of these subsets. Since these points have measure zero, this does not influence the validity of the log-likelihood. However, it may impede optimization using gradient-based methods. In our experiments, we did not encounter any specific issues, but for a more thorough discussion see (Dinh et al., 2019).

### 3.2 Connection to Previous Work

The results above provide a unified framework based on SurVAE Flows for estimating probability densities. We now connect this general approach to several recent works on generative modelling.

The differentiable and bijective nature of transformations used in normalizing flows limit their ability to alter dimensionality, model discrete data, and distributions with disconnected components. Specialized solutions have been proposed in recent years to address these individually. We now show that these works can be expressed using SurVAE Flow layers, as summarized in Table 3.

#### 3.2.1 Using Stochastic Transformations

As discussed in Sec. 2, VAEs (Kingma and Welling, 2014; Rezende et al., 2014) may be formulated as composable stochastic transformations. Probabilistic PCA Tipping and Bishop (1999) can be considered a simple special case of VAEs wherein the forward transformation is linear-Gaussian, i.e. \( p(x|z) = \mathcal{N}(x|Wz, \sigma^2 I) \). Due to the linear-Gaussian formulation, the posterior \( p(z|x) \) is tractable and we can thus perform exact stochastic inversion for this model. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) are another class of models closely related to VAEs. For diffusion models, the inverse \( q(z|x) \) implements a diffusion step, while the forward transformation \( p(x|z) \) learns to reverse the diffusion process. Wu et al. (2020) propose an extended flow framework consisting of bijective and stochastic transformations using MCMC transition kernels. Their method utilizes the same computation as in the general formulation in Algorithm 1, but does not consider surjective maps or an explicit connection to VAEs. Their work shows that MCMC kernels may also be implemented as stochastic transformations in SurVAE Flows.

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

Dequantization (Uria et al., 2013; Ho et al., 2019) is used for training continuous flow models on ordinal discrete data such as images and audio. Dequantization fits into the SurVAE Flow framework as a composable generative rounding surjection (cf. Example 2) and thus simplifies implementation. When the inverse \( q(z|x) \) is a standard uniform distribution, \textit{uniform dequantization} is obtained, while a more flexible flow-based distribution \( q(z|x) \) yields \textit{variational dequantization} (Ho et al., 2019).

VFlow (Chen et al., 2020) and ANFs (Huang et al., 2020) aim to build expressive generative models by augmenting the data space and jointly learning a normalizing flow for the augmented data space as well as the distribution of augmented dimensions. This strategy was also adopted by Dupont et al. (2019) for continuous-time flows. VFlow and ANFs can be obtained as SurVAE Flows by composing a bijection with a generative tensor slicing surjection (cf. Example 1 and Figure 3a). The reverse transformation, i.e. inference slicing, results in the \textit{multi-scale architecture} of Dinh et al. (2017).

CIFs (Cornish et al., 2019) use an indexed family of bijective transformations \( g(\cdot; \epsilon) : \mathcal{Z} \rightarrow \mathcal{X} \) where \( \mathcal{Z} = \mathcal{X} \subseteq \mathbb{R}^d \), and \( \epsilon \in \mathcal{E} \subseteq \mathbb{R}^{d_\epsilon} \) with the generative process as: \( z \sim p(z), \epsilon \sim p(\epsilon|z) \) and \( x = g(z; \epsilon) \). CIFs are akin to modeling densities using an infinite mixture of normalizing flows since \( g \) is a surjection from an augmented space \( \mathcal{Z} \times \mathcal{E} \) to the data space \( \mathcal{X} \). Consequently, CIFs can be expressed as a SurVAE flow using a augument surjection composed with a bijection and tensor slicing (cf. Figure 3b). Similarly, Duan (2019) used a \textit{finite} mixture of normalizing flows to model densities by using a discrete index set \( \mathcal{E} = \{1, 2, 3, \cdots, K\} \) with bijections \( g(\cdot; \epsilon) \). Deep Gaussian mixture models (Oord and Dambre, 2015) form special case wherein the bijections \( g(\cdot; \epsilon) \) are linear transformations.

RAD flows (Dinh et al., 2019) are also “similar” to CIFs but it partitions the data space into finite disjoint subsets \( \{\mathcal{X}_i\}_{i=1}^K \subseteq \mathcal{X} \) and defines bijections \( g_i : \mathcal{Z} \rightarrow \mathcal{X}_i, \forall \ i \in \{1, 2, ..., K\} \) with the generative process as \( z \sim p(z), i \sim p(i|z) \) and \( x = g_i(z) \). Interestingly, RAD can be seen to implement a class of inference surjections that rely on partitioning of the data space. The partitioning is learned during training, thus allowing learning of expressive inference surjections. However, careful parameterization is required for stable gradient-based training. We note that the abs and sort inference surjections introduced earlier may be expressed using a static (non-learned) partitioning of the data space \( \mathcal{X} \) and thus have close ties to RAD. However, RAD does not express generative surjections or more general inference surjections that do not rely on partitioning, such as dimensional changes.

Finally, we note that apart from providing a general method for modelling densities, SurVAE Flows provide a modular framework for easy implementation of the methods described here. We discuss these important software perspectives using code snippets in App. H.

\[ \text{Bijection} \]

\[ \text{Augmentation} \]

\[ \text{Factor out} \]

\[ \text{(Inf. Tensor Slicing)} \]

(a) Augmented Flow.

\[ \text{Bijection} \]

\[ \text{Augmentation} \]

\[ \text{(Gen. Tensor Slicing)} \]

(b) Infinite Mixture of Flows.

Figure 3: Flow architectures making use of tensor slicing.
4 Experiments

We investigate the ability of SurVAE flows to model data that is difficult to model with normalizing flows. We show that the absolute value surjection is useful in modelling data where certain symmetries are known to exist. Next, we demonstrate that SurVAE flows allow straightforward modelling of exchangeable data by simply composing any flow together with either a sorting surjection or a stochastic permutation layer. Furthermore, we investigate the use of max pooling – which is commonly used for downsampling in convolutional neural networks – as a surjective downsampling layer in SurVAE flows for image data.

Synthetic Data. We first consider modelling data where certain symmetries are known to exist. We make use of 3 symmetric and 1 anti-symmetric synthetic 2D datasets. The absolute value inference surjection can be seen to fold the input space across the origin and can thus be useful in modelling such data. The baseline uses 4 coupling bijections, while our AbsFlow adds an extra abs surjection. For the anti-symmetric data, AbsFlow uses only a single abs surjection with a classifier (i.e. for $P(s|z)$) which learns the unfolding. For further details, see App. I.1. The results are shown in Fig. 4.

Point Cloud Data. We now consider modelling exchangeable data. We use the SpatialMNIST dataset (Edwards and Storkey, 2017), where each MNIST digit is represented as a 2D point cloud of 50 points. A point cloud is a set, i.e. it is permutation invariant. Using SurVAE flows, we can enforce permutation invariance on any flow using either 1) a sorting surjection – forcing a canonical order on the inputs, or 2) a stochastic permutation – forcing a random order on the inputs.

We compare 2 SurVAE flows, SortFlow and PermuteFlow, both using 64 layers of coupling flows parameterized by Transformer networks (Vaswani et al., 2017). Transformers are – when not using positional encoding – permutation equivariant. PermuteFlow uses stochastic permutation in-between the coupling layers. SortFlow, on the other hand, uses an initial sorting surjection, which introduces an ordering, and fixed permutations after. The Transformers thus make use of learned positional encodings for SortFlow, but not for PermuteFlow. See App. I.2 for further details, and Fig. 5 for

Figure 4: Comparison of flows with and without absolute value surjections modelling anti-symmetric (top row) and symmetric (3 bottom rows) 2-dimensional distributions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Flow</th>
<th>AbsFlow (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkerboard</td>
<td>3.65</td>
<td>3.49</td>
</tr>
<tr>
<td>Corners</td>
<td>3.19</td>
<td>3.03</td>
</tr>
<tr>
<td>Gaussians</td>
<td>3.01</td>
<td>2.86</td>
</tr>
<tr>
<td>Circles</td>
<td>3.44</td>
<td>2.99</td>
</tr>
</tbody>
</table>

Figure 5: Point cloud samples from permutation-invariant SurVAE flows trained on SpatialMNIST.
Table 4: Unconditional image modeling results in bits/dim.

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10</th>
<th>ImageNet32</th>
<th>ImageNet64</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealNVP (Dinh et al., 2017)</td>
<td>3.49</td>
<td>4.28</td>
<td>-</td>
</tr>
<tr>
<td>Glow (Kingma and Dhariwal, 2018)</td>
<td>3.35</td>
<td>4.09</td>
<td>3.81</td>
</tr>
<tr>
<td>Flow++ (Ho et al., 2019)</td>
<td>3.08</td>
<td>3.86</td>
<td>3.69</td>
</tr>
<tr>
<td>Baseline (Ours)</td>
<td>3.08</td>
<td>4.00</td>
<td>3.70</td>
</tr>
<tr>
<td>MaxPoolFlow (Ours)</td>
<td>3.09</td>
<td>4.01</td>
<td>3.74</td>
</tr>
</tbody>
</table>

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

Table 5: Inception score and FID for CIFAR-10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Inception ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCGAN*</td>
<td>6.4</td>
<td>37.1</td>
</tr>
<tr>
<td>WGAN-GP*</td>
<td>6.5</td>
<td>36.4</td>
</tr>
<tr>
<td>PixelCNN*</td>
<td>4.60</td>
<td>65.93</td>
</tr>
<tr>
<td>PixelIQN*</td>
<td>5.29</td>
<td>49.46</td>
</tr>
<tr>
<td>Baseline (Ours)</td>
<td>5.08</td>
<td>49.56</td>
</tr>
<tr>
<td>MaxPoolFlow (Ours)</td>
<td>5.18</td>
<td>49.03</td>
</tr>
</tbody>
</table>

*Results taken from Ostrovski et al. (2018).

model samples. Interestingly, PermuteFlow outperforms SortFlow, with -5.30 vs. -5.53 PPLL (per-point log-likelihood), even though it only allows computation of lower bound likelihood estimates. For comparison, BRUNO (Korshunova et al., 2018) and FlowScan (Bender et al., 2020) obtain -5.68 and -5.26 PPLL, but make use of autoregressive components. Neural Statistican (Edwards and Storkey, 2017) utilizes hierarchical latent variables without autoregressive parts and obtains -5.37 PPLL. PermuteFlow thus obtains state-of-the-art performance among non-autoregressive models.

Image Data. Max pooling layers are commonly used for downsampling in convolutional neural networks. We investigate their use as surjective downsampling transformations in flow models for image data here.

We train a flow using 2 scales with 12 steps/scale for CIFAR-10 and ImageNet 32×32 and 3 scales with 8 steps/scale for ImageNet 64×64. Each step consists of an affine coupling bijection and a 1×1 convolution (Kingma and Dhariwal, 2018). We implement a max pooling surjection for downscaling and compare it to a baseline model with tensor slicing which corresponds to a multi-scale architecture (Dinh et al., 2017). We report results for the log-likelihood in Table 4 and the inception and FID scores in Table 5 with bolding indicating best among the baseline and MaxPoolFlow. The results show that compared to slicing surjections, the max pooling surjections yield marginally worse log-likelihoods, but better visual sample quality as measured by the Inception score and FID. We also provide the generated samples from our models in Fig. 7 and App. J. Due to space constrains, we refer the reader to App. I.3 for more details on the experiment.

5 Conclusion

We introduced SurVAE flows, a modular framework for constructing likelihood-based models using composable bijective, surjective and stochastic transformations. We showed how this encompasses normalizing flows, which rely on bijections, as well as VAEs, which rely on stochastic transformations. We further showed that several recently proposed methods such as dequantization and augmented normalizing flows may be obtained as SurVAE flows using surjective transformations. One interesting direction for further research is development of novel non-bijective transformations that might be beneficial as composable layers in SurVAE flows.
Acknowledgements

We thank Laurent Dinh and Giorgio Giannone for helpful feedback.

Funding Disclosure

This research was supported by the NVIDIA Corporation with the donation of TITAN X GPUs.

Broader Impact

This work constitutes foundational research on generative models/unsupervised learning by providing a unified view on several lines of work and further by introducing new modules that expand the generative modelling toolkit. This work further suggests how to build software libraries to that allows more rapid implementation of a wider range of deep unsupervised models. Unsupervised learning has the potential to greatly reduce the need for labeled data and thus improve models in applications such as medical imaging where a lack of data can be a limitation. However, it may also potentially be used to improve deep fakes with potentially malicious applications.

References


