Latent Normalizing Flows for Discrete Sequences

Anonymous Authors[1]

Abstract

Normalizing flows have been shown to be a powerful class of generative models for continuous random variables, giving both strong performance and the potential for non-autoregressive generation. These benefits are also desired when modeling discrete random variables such as text, but directly applying normalizing flows to discrete sequences poses significant additional challenges. We propose a generative model which jointly learns a normalizing flow-based distribution in the latent space and a stochastic mapping to an observed discrete space. In this setting, we find that it is crucial for the flow-based distribution to be highly multimodal. To capture this property, we propose several normalizing flow architectures to maximize model flexibility. Experiments consider common discrete sequence tasks of character-level language modeling and polyphonic music generation. Our results indicate that an autoregressive flow-based model can match the performance of a comparable autoregressive baseline, and a non-autoregressive flow-based model can improve generation speed with a penalty to performance.

1. Introduction

The goal of generative modeling is to learn the joint distribution of a high-dimensional random variable. One class of models that has shown particularly strong performance are autoregressive models, which parameterize the joint density such that each variable depends on all previous assignments. These models give state-of-the-art performance across many tasks (van den Oord et al., 2016; Salimans et al., 2017; Vaswani et al., 2017; Al-Rfou et al., 2018), and are particularly dominant in natural language processing (NLP) (Vaswani et al., 2017; Al-Rfou et al., 2018). One downside of autoregressive models, however, is that their sampling procedure requires sampling tokens one-by-one and is therefore serial in the length of the sequence, which can pose problems in real-world applications.

Normalizing flows are a class of generative model that implicitly represent the joint distribution of a high-dimensional random variable via an invertible deterministic transformation from a base density (Rezende & Mohamed, 2015; Kingma et al., 2016). Normalizing flows have been explored both to increase the flexibility of the variational posterior distribution in the context of variational autoencoders (Rezende & Mohamed, 2015; Kingma et al., 2016), and to model observed space, which is the focus of this work. Normalizing flows provide two key advantages: model flexibility and control over computational tradeoffs. Flows generalize standard autoregressive models (Papamakarios et al., 2017) and give more distributional flexibility. Furthermore, normalizing flows can be designed that are non-autoregressive during sampling (van den Oord et al., 2018; Kingma & Dhariwal, 2018), enabling parallel generation. Recent work around images has demonstrated accuracy for non-autoregressive models approaching that of autoregressive models in the continuous setting (Kingma & Dhariwal, 2018).

Both properties are desirable in the discrete domain, where autoregressive models are the dominant paradigm. Unfortunately, normalizing flows rely on parameterized applications of the change-of-variables formula. Applying related methods, e.g. via the discrete change of variables or a relaxation, to discrete random variables leads to significant additional challenges. A method for applying flows to discrete data and creating flows flexible enough to model highly multimodal discrete data has not yet been demonstrated.

In this work, we propose an alternative approach for discrete sequence modeling with normalizing flows. We develop a generative model that jointly learns a flow-based density in the latent space and a simple mapping to discrete observations. Specifically we propose (1) a latent variable model that learns all dynamics of the observed discrete space in the latent continuous space, and (2) three specific normalizing flow architectures designed to capture these dynamics, in particular the extreme multimodality inherent in discrete data.

Experiments consider discrete latent generative models for

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

character-level language modeling and polyphonic music modeling. We find that the latent flow model is able to describe the character-level dataset as well as a comparable baseline LSTM-based model, and is able to describe the polyphonic music datasets comparably to other autoregressive latent variable models. We further find that the parallel-generation version of the model is able to generate sentences faster than the baseline model, with a penalty to modeling performance. Finally, we analyze the functionality of the model and demonstrate how it induces the high degree of multimodality needed to map between continuous and discrete spaces.

2. Related Work

Latent Variable Models for Sequences In the context of language modeling, Bowman et al. (2016) experiment with a variational autoencoder (VAE) of fixed size continuous latent space and an autoregressive RNN decoder. In practice, the VAE encodes little information about the sentence in the latent space because the decoder is powerful enough to model the data well, reducing the VAE to a standard autoregressive model. Recent work has focused on increasing the amount of information the model places in the latent space, either by modifying the prior density (Xu & Durrett, 2018), the decoder structure (Yang et al., 2017), or the variational inference procedure (Kim et al., 2018), though in all cases the model still relies heavily on the discrete decoder. Our proposed model removes the discrete autoregressive decoder entirely.

Other methods construct VAEs for sequence data with a variable size latent variable composed of one latent vector per input token (Bayer & Osendorfer, 2015; Chung et al., 2015; Gu et al., 2015). While similar to the model proposed in this work in the layout of the latent dimensions, these models also include an autoregressive discrete decoder.

Chen et al. (2017) propose a VAE model for images with a learned normalizing-flow based prior and a weaker decoder. The latent size is fixed, the model is applied to continuous random variables, and the decoder still allows for dependence between the random variables. This differs from our latent sequence model. To the best of our knowledge, no previous works explore the setting of a latent continuous sequence model with a weak discrete decoder.

Non-Autoregressive Generation In the domain of natural images, Dinh et al. (2017) and Kingma & Dhariwal (2018) propose flow-based models using affine "coupling layers" to allow for non-autoregressive generation. Compared to state-of-the-art autoregressive models, their non-autoregressive model performs both training and generation in parallel but suffers a penalty to model accuracy.

In the domain of text Gu et al. (2018) propose a model which uses fertility scores as a latent variable, approaching the performance of autoregressive models. While this works for translation due to the aligned nature of the sentences, the fertility framework and required pre-trained autoregressive model preclude the technique from more general application. Lee et al. (2018) propose a deterministic model based on a denoising process to iteratively improve the quality of a non-autoregresively generated sentence. The authors demonstrate strong performance at neural machine translation, but the technique does not model the full distribution and requires a task-specific predetermined denoising process.

In an alternative approach for faster neural machine translation, Kaiser et al. (2018) propose to use a discrete latent space of variable but reduced size (e.g. 8x fewer tokens than the length of the sentence). While this technique speeds up the translation process, it is still serial. Furthermore, the method makes no claims about fully modeling the distribution of the data.

3. Background: Normalizing Flows

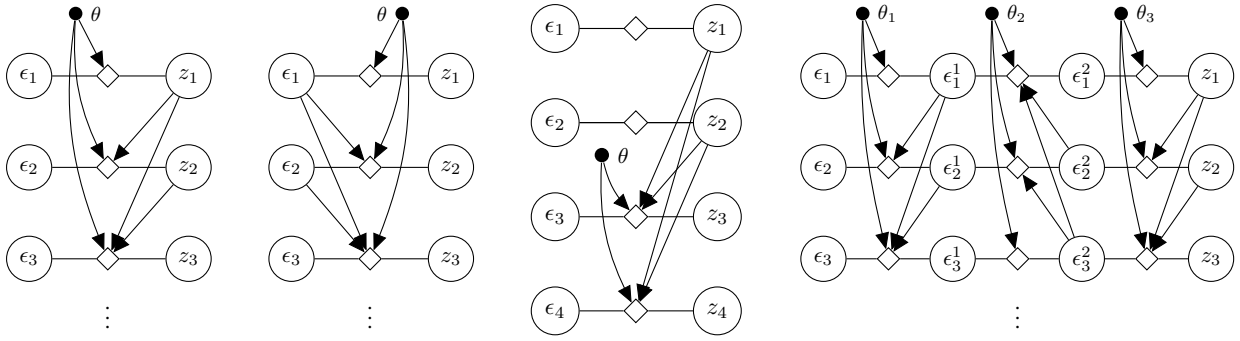
Normalizing flows are a class of model that define a density through a parameterized invertible deterministic transformation from a base density, such as a standard Gaussian (Tabak & Vanden-Eijnden, 2010). Define an invertible transformation $f_\theta : \epsilon \to \mathcal{Z}$ and base density $p_\epsilon(\boldsymbol{\epsilon})$. These specify density $p_Z(\boldsymbol{z})$ via the change-of-variables formula:

$$p_Z(\boldsymbol{z}) = p_\epsilon(f_\theta^{-1}(\boldsymbol{z})) \left| \det \frac{\partial f_\theta^{-1}(\boldsymbol{z})}{\partial \boldsymbol{z}} \right|$$

Consider two core operations defined with flows: (a) Sampling, $\boldsymbol{z} \sim p_Z$, is performed by first sampling from the base distribution, $\boldsymbol{\epsilon} \sim p_\epsilon$, and then applying the forward transformation $\boldsymbol{z} = f_\theta(\boldsymbol{\epsilon})$; (b) density evaluation, $p_Z(\boldsymbol{z})$ for a known $\boldsymbol{z}$, is computed by inverting the transformation, $\boldsymbol{\epsilon} = f_\theta^{-1}(\boldsymbol{z})$, and computing the base density $p_\epsilon(\boldsymbol{\epsilon})$. If f_θ is chosen to have an easily computable Jacobian determinant and inverse, both of these can be computed efficiently.

One method for satisfying these criteria is to compose invertible components, such as scalar affine transformations, and arrange them to ensure a triangular Jacobian matrix and therefore a linear determinant calculation. We consider three different variants on this theme, and discuss the computational tradeoffs for sampling and density evaluation. For this section we assume without loss of generality that $\mathcal{Z} = \mathbb{R}^D$ with ordered dimensions $1, \ldots, D$.

Autoregressive Flow (AF) Autoregressive flows, originally proposed in Papamakarios et al. (2017), ensure an invertible transformation and triangular Jacobian matrix by



(a) AF (←) (b) IAF (→) (c) SCF (↔) (d) 3-layer AF (←)

Figure 1. Flow diagrams for normalizing flows acting on sequences of scalars. Circles represent random variables ϵ_d or z_d. Diamonds represent a parameterized invertible scalar transformation, f_θ, in this case an affine transformation. Diagrams show the sampling process ($\boldsymbol{\epsilon} \rightarrow \boldsymbol{z}$, read left to right) and density evaluation ($\boldsymbol{\epsilon} \leftarrow \boldsymbol{z}$, read right to left). While all models can be used in both directions, they differ in terms of whether the calculation is serial or parallel, i.e. AF is parallel in evaluation but serial in sampling (←) because z_1 is needed to sample z_2, whereas SCF is parallel for both (↔).

conditioning each scalar affine transformation on all previously observed variables $\boldsymbol{z}_{<d}$,

$$f_\theta(\boldsymbol{\epsilon})_d = z_d = a(z_{<d}; \theta) + b(z_{<d}; \theta) \cdot \epsilon_d$$
$$f_\theta^{-1}(\boldsymbol{z})_d = \epsilon_d = \frac{z_d - a(z_{<d}; \theta)}{b(z_{<d}; \theta)}$$

where a and b are the shift and scale functions with shared parameters θ. The Jacobian matrix is triangular because $\frac{\partial z_i}{\partial \epsilon_j}$ is non-zero only for $j \leq i$, with determinant $\prod b(z_{<d}; \theta)$.

A flow diagram of AF is shown in Figure 1a. To sample $\boldsymbol{z}$, we sample each ϵ_d on the left. The first z_1 is computed through an affine transformation, and then each subsequent z_d is sampled in serial based on ϵ_d and $z_{<d}$. To evaluate the density, we simply apply individual scalar affine transformations in parallel, each depending on all previous observed $z_{<d}$, and compute the base density.

Inverse Autoregressive Flow (IAF) Inverse autoregressive flows, proposed in Kingma et al. (2016), use affine transformations that depend on previous $\epsilon_{<d}$ instead of $z_{<d}$. The transformation f_θ for IAF has the form:

$$f_\theta(\boldsymbol{\epsilon})_d = z_d = a(\epsilon_{<d}; \theta) + b(\boldsymbol{\epsilon}_{<d}; \theta) \cdot \epsilon_d$$
$$f_\theta^{-1}(\boldsymbol{z})_d = \epsilon_d = \frac{z_d - a(\boldsymbol{\epsilon}_{<d}; \theta)}{b(\epsilon_{<d}; \theta)}$$

A flow diagram for IAF is shown in Figure 1b. For the sampling process all z_d can be computed given $\boldsymbol{\epsilon}$ in parallel; conversely, density evaluation requires computing each ϵ_d serially since $\epsilon_{<d}$ is needed for the transformation. In practice AF and IAF encode different inductive biases which can hinder the ability of IAF to generalize as well as AF (van den Oord et al., 2018).

Split Coupling Flow (SCF) Split coupling flows, initially proposed in Dinh et al. (2017) and followed up on in Kingma & Dhariwal (2018), utilize "coupling layers" that keep a subset $\mathcal{S} \subset \{1, 2, ..., D\}$ of the random variables unchanged, i.e. $\boldsymbol{z}_\mathcal{S} = \boldsymbol{\epsilon}_\mathcal{S}$, and use these to condition the transformation for the rest of the random variables $\overline{\mathcal{S}}$. The transformation f_θ for SCF and $d \in \overline{\mathcal{S}}$ can be written:

$$f_\theta(\boldsymbol{\epsilon})_d = z_d = a(\boldsymbol{z}_\mathcal{S}; \theta) + b(\boldsymbol{z}_\mathcal{S}; \theta) \cdot \epsilon_d$$
$$f_\theta^{-1}(\boldsymbol{z})_d = \epsilon_d = \frac{z_d - a(\boldsymbol{z}_\mathcal{S}; \theta)}{b(\boldsymbol{z}_\mathcal{S}; \theta)}$$

A flow diagram for SCF is shown in Figure 1c, where $S = \{1, 2\}$ for visualization. Because only the first two variables are used to condition the rest of the affine transformations, both sampling and density evaluation are parallel. As SCF is a special case of AF it has a strictly reduced modeling flexibility in exchange for improved computational efficiency (Papamakarios et al., 2017).

Layered Flows Each flow encodes an invertible function with a linearly computable Jacobian determinant. Because invertibility is closed under function composition, and the Jacobian determinant of composed functions is the product of the individual Jacobian determinants, more flexible distributions can be created by layering flows and changing the ordering of the dependencies at each layer (Salimans et al., 2017). Changing the ordering between layers allows all z_ds or ϵ_ds to interact with each other, and is usually implemented by reversing or shuffling the ordering of dependencies (Kingma & Dhariwal, 2018).

Figure 1d shows an example with three layers of AF, with reversed dependency ordering between layers. Stacking multiple layers of flow has been shown to significantly increase the modeling flexibility of this class of normalizing flows (Kingma & Dhariwal, 2018; van den Oord et al., 2018).

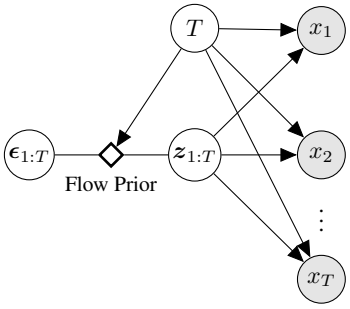


Figure 2. Proposed generative model of discrete sequences. The model first samples a sequence length T and then a latent continuous sequence $\boldsymbol{z}_{1:T}$. Each x_t is shown separately to highlight their conditional independence given $\boldsymbol{z}_{1:T}$. Normalizing flow specifics are abstracted by $p(\boldsymbol{z})$ are described in Section 4.3.

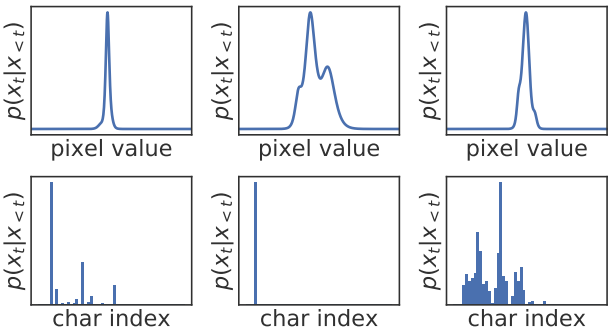


Figure 3. Example conditional distributions $p(x_t|x_{<t})$ from continuous (PixelCNN++, 10 mixture components, trained on CIFAR-10) and discrete (LSTM char-level LM trained on PTB) autoregressive models.

A multilayer flow represents a true invertible vector transformation $f_\theta(\boldsymbol{\epsilon})$ with a dense Jacobian matrix. Forming the building blocks for the discrete flow models, we denote a multilayer AF as $f_{\text{AF}}(\boldsymbol{\epsilon};\theta)$, a multilayer IAF as $f_{\text{IAF}}(\boldsymbol{\epsilon};\theta)$, and a multilayer SCF $f_{\text{SCF}}(\boldsymbol{\epsilon};\theta)$.

4. Latent Flows for Discrete Sequences

Using these building blocks, we aim to develop flexible flow-based models for discrete sequences. The first difficulty is that any deterministic non-trivial mapping between a discrete space and a continuous space or between two discrete spaces is not invertible. Instead we explore using a latent-variable model, with a continuous latent sequence modeled through normalizing flows. We begin by describing the full generative process and then focus on the flow-based prior.

4.1. Generating Discrete Sequences

Our central process will be a latent-variable model for a discrete sequence. However, unlike standard discrete autoregressive models, we aim to lift the main dynamics of the system into continuous space, i.e. into the prior. In particular, we make the strong assumption that each discrete symbol is *conditionally independent* given the latent.

Concretely, we model the generation of a discrete sequence $\boldsymbol{x}_{1:T} = \{x_1, ..., x_T\}$ conditioned on a latent sequence $\boldsymbol{z}_{1:T}$ made up of continuous random vectors $\{\boldsymbol{z}_1, ..., \boldsymbol{z}_T\}$ with $\boldsymbol{z}_t \in \mathbb{R}^H$ and H is a hidden dimension. Define $p(\boldsymbol{z}_{1:T}|T)$ as our prior distribution, and generate from the conditional distribution over discrete observed variables $p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}, T)$. The conditional likelihood generates each x_t conditionally independently: $p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}, T) = \prod_{t=1}^{T} p(x_t|\boldsymbol{z}_{1:T}, T)$, where the emission distribution depends on the dataset.

To allow for non-autoregressive generation, the length of the sequence T is explicitly modeled as a latent variable and all parts of the model are conditioned on it. Length conditioning is elided in the following discussion (see the Supplementary Materials for details). The complete graphical model is shown in Figure 2.

4.2. Criteria for Effective Flow Parameterization

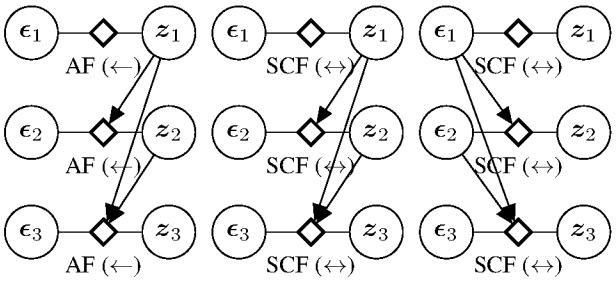
The prior $p(\boldsymbol{z}_{1:T})$ in this process needs to capture the dynamics of the discrete system in a continuous space. Unlike common continuous spaces such as images, in which conditional distributions $p(x_t|x_{<t})$ are often modeled well by unimodal or few-modal distributions, discrete spaces with fixed generation order are highly multimodal.

Figure 3 illustrates this difficulty. First consider the continuous distributions generated by an AF model (PixelCNN++ (Salimans et al., 2017)) with 10 mixture components. Despite its flexibility, the resulting distributions have a limited modality indicating that increasing flexibility does not better model the data. Further corroborating this hypothesis, (Salimans et al., 2017) report that using more than 5 mixture components does not improve performance.

In contrast, Figure 3b shows a similar experiment on discrete data. Here the first and third distributions are highly multimodal (given previous characters there are multiple different possibilities for the next character). Furthermore, the degree of multimodality can vary significantly, as in the second example, requiring models to be able to adjust the number of indicated modes in addition to their locations. In the proposed model, because the conditional likelihood models each x_t as independent, this multimodality at each time step needs to exist almost exclusively in the latent space with each likelihood $p(x_t|\boldsymbol{z})$ being highly constrained in its conditioning.

4.3. Flow Architectures for Sequence Dynamics

We consider three flow architectures that describe relations across the time and hidden dimensions that aim to maximize



(a) AF / AF (←) (b) AF / SCF (←) (c) IAF / SCF (→)

Figure 4. Normalizing flows acting on TxH random variables proposed in this work. Circles with variables represent random *vectors* of size H. Bold diamonds each represent a multilayer AF (←) or a multilayer SCF (↔), as in Figure 1d. Arrows to a bold diamond represent additional dependencies to all affine transformations within the indicated AF or SCF. As above the (arrows) point to the parallel direction, i.e. (a) is parallel in density evaluation whereas (c) is parallel in sampling.

the potential for multimodal distributions. These differ in their inductive biases as well as the sampling and density evaluation processes. Note, that throughout this section $\boldsymbol{z}_t \in \mathbb{R}^H$ represents a random vector, and so the model is over $D = T$xH random variables. The main concern is the interactions between time T and hidden H dimensions.

Model 1: AF in time, AF in hidden (AF / AF) First consider an autoregressive flow along the time dimension with each time step applying an autoregressive flow along the hidden dimension. The transformation function can be written as,

$$\boldsymbol{z}_t = f_{\text{AF}}(\boldsymbol{\epsilon}_t; \boldsymbol{z}_{<t}, \theta), \quad \boldsymbol{\epsilon}_t = f_{\text{AF}}^{-1}(\boldsymbol{z}_t; \boldsymbol{z}_{<t}, \theta)$$

where $f_{\text{AF}}(\cdot; \boldsymbol{z}_{<t}, \theta)$ is a layered AF transformation described above with each constituent affine transformation conditioned on $\boldsymbol{z}_{<t}$ in addition to θ. A proof that this represents a valid normalizing flow is given in the Supplementary Materials.

The flow diagram is shown in Figure 4a. At each time step the AF-in-hidden induces dependencies along the hidden dimension (inside f_{AF}) to create a multimodal distribution. The AF-in-time conditions each subsequent f_{AF} on the previous latent vectors $\boldsymbol{z}_{<t}$. For density evaluation, $p(\boldsymbol{z}_{1:T})$, both the dependencies within each f_{AF} and the dependencies across time can be computed in parallel. For sampling, each hidden dimension at each time step must be computed in serial.

Model 2: AF in time, SCF in hidden (AF / SCF) Model 1 can be evaluated efficiently, but the serial sampling procedure may be an issue in applications. As an alternative we consider a flow which replaces AF-in-hidden dimension with a layered SCF. The prior is defined by the forward and inverse transformation functions,

$$\boldsymbol{z}_t = f_{\text{SCF}}(\boldsymbol{\epsilon}_t; \boldsymbol{z}_{<t}, \theta), \quad \boldsymbol{\epsilon}_t = f_{\text{SCF}}^{-1}(\boldsymbol{z}_t; \boldsymbol{z}_{<t}, \theta)$$

The flow diagram is shown in Figure 4b. This model allows for similar parallel density evaluation as Model 1, however it is parallel in sampling along the hidden dimension, which can help efficiency. The downside is that SCF may not be able to induce the flexible multimodality required for the discrete case.

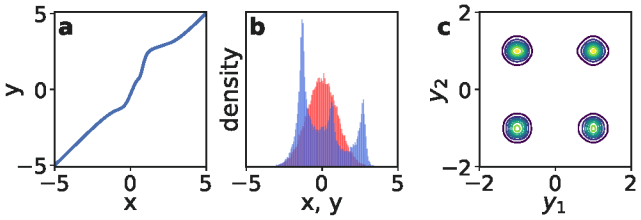


Figure 5. **(a, b)** Transformation defined by hand-selecting 4 layers of flow parameters, demonstrating the ability of the flow to model complicated multimodal distributions: **(a)** composed transformation, **(b)** base density (red), final density (blue). **(c)** Resulting density for learned 2D transformation via 5 layer AF-like using the NLSq flow from a standard Gaussian to a Gaussian mixture distribution.

Model 3: IAF in time, SCF in hidden (IAF / SCF) Finally, the autoregressive sampling behavior can be removed completely. The final model uses an IAF-in-time to remove this serial dependency in sampling. The transformation functions are:

$$\boldsymbol{z}_t = f_{\text{SCF}}(\boldsymbol{\epsilon}_t; \boldsymbol{\epsilon}_{<t}, \theta), \quad \boldsymbol{\epsilon}_t = f_{\text{SCF}}^{-1}(\boldsymbol{z}_t; \boldsymbol{\epsilon}_{<t}, \theta)$$

The flow diagram is shown in Figure 4c. For sampling, given $\boldsymbol{\epsilon}_{1:T}$ the time-wise and depth-wise dependencies can be satisfied in parallel (they all appear on the right side of the forward transformation function). Density evaluation, on the other hand, becomes parallel along hidden and serial in time.[1]

Extension: The Non-Linear Squared Flow We can add further flexibility to the model by modifying the core flows. Building on the observations of (Huang et al., 2018), we propose replacing the affine scalar transformation with an invertible non-linear squared transformation (designated NLSq):

$$f(\epsilon) = z = a + b\epsilon + \frac{c}{1 + (d\epsilon + g)^2}$$

[1] We also considered an IAF / IAF model; however having fully serial operation in density evaluation makes training prohibitively expensive.

This transformation has five pseudo-parameters instead of the two for the affine. It reduces to the affine function in the case where $c = 0$. When $c \neq 0$, the function effectively adds a perturbation with position controlled by g and scale controlled by c and d, which even in 1D can induce multimodality. Under conditions on the scale parameter c the function can be guaranteed to be invertible, and the analytical inverse is the solution to a cubic equation (see Supplementary Materials for details).

Figure 5 illustrates the transformation. Figure 5a, b show an example of four compositions of NLSq functions, and the initial and final density. Whereas the affine transformation would simply scale and shift the Gaussian, the NLSq function induces multimodality. As a final example of the ability of this function to model a multimodal distribution within the flow framework, Figure 5c shows the learned 2D density for a toy dataset consisting of a mixture of four Gaussians. Consistent with Huang et al. (2018), we find that an AF even with many layers fails to learn to model the same distribution.

5. Variational Inference and Training

To train the model, we need to learn both the simple likelihood and the prior models. This requires being able to efficiently perform posterior inference, i.e. compute the posterior distribution $p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$, which is computationally intractable. We instead use the standard approach of amortized variational inference (Kingma & Welling, 2014) by introducing a trained inference network, $q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$. This distribution q models each $\boldsymbol{z}_t$ as a diagonal Gaussian with learned mean and variance:

$$q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t(\boldsymbol{x}_{1:T};\phi), \sigma_t^2(\boldsymbol{x}_{1:T};\phi) I_H).$$

While this mean-field factorization results in a weak inference model, preliminary experiments indicated that increasing the flexibility of the inference model with e.g. IAF (Kingma et al., 2016) did not improve performance.

This inference network is trained jointly with the model to maximize the evidence lower-bound (ELBO),

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q_\phi}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))$$

Training proceeds by estimating the expectation with monte-carlo samples and optimizing the lower bound for both the inference network parameters ϕ as well as the prior $p(\boldsymbol{z})$ and likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ parameters.

6. Methods and Experiments

We consider two standard discrete sequence modeling tasks: character-level language modeling and polyphonic music modeling. For all experiments, we compute the negative log-likelihood (NLL) estimated with importance sampling and evaluate on a held-out test set to evaluate distribution-modeling performance. As a baseline, we use a LSTM-based language model as in Press & Wolf (2017), the standard discrete autoregressive model. For all experiments we use a baseline LSTM of the same size as the flow-based model. For all flow-based models, a BiLSTM is used to compute the likelihood model $p(x_t|\boldsymbol{z})$ and the inference network $q(\boldsymbol{z}_t|\boldsymbol{x})$. All flow-based models use NLSq unless otherwise noted. Optimization and hyperparameter details are given in the Supplementary Materials.

Table 1. Character-level language modeling results on PTB. NLL for generative models is estimated with importance sampling using 50 samples, the reconstruction term and KL term refer to the two components of the ELBO. The LSTM from Cooijmans et al. (2017) uses the standard character-setup which crosses sentence boundaries (see footnote).

Model	Test NLL (bpc)	Reconst. (bpc)	KL (bpc)
LSTM	1.38	-	-
LSTM (sentence-wise)	1.41	-	-
AF-only	2.90	0.15	2.77
AF / AF	1.42	0.10	1.37
AF / SCF	1.46	0.10	1.43
IAF / SCF	1.63	0.21	1.55

6.1. Character-Level Language Modeling

Character-level language modeling tests the ability of a model to capture the full distribution of high entropy data with long-term dependencies. We use the Penn Treebank dataset, with the standard preprocessing as in (Mikolov et al., 2012). The dataset consists of approximately 5M characters, with rare words replaced by "<unk>" and a character-level vocabulary size of $V = 51$.[2]

Table 1 shows results. The LSTM baseline establishes a "gold standard" representing a model trained directly on the observed discrete sequence with the same T conditioning as the proposed model. In terms of absolute NLL score, AF / AF nearly matches the LSTM baseline, whereas AF

[2] Unlike previous works on character-level language modeling which consider the dataset to be a single continuous string of characters, non-autoregressive generation requires the dataset to be split up into finite chunks. Following previous text-based VAE works in the literature (Bowman et al., 2016), the dataset is split into sentences. To avoid extreme outliers, the dataset is limited to sentences of length less than 288 tokens, which accounts for 99.3% of the original dataset. Due to these two modifications the absolute NLL scores are not precisely comparable between this dataset and the one used in previous works, although the difference is small.

Table 2. Ablation experiments. AF / AF is the same result as in Table 1. -NLSq indicates the affine transformation is used instead of the NLSq transformation. -AF hidden indicates no dependencies across hidden (an independent vector affine transformation is used instead).

Model	Test NLL (bpc)	Reconst. (bpc)	KL (bpc)
AF / AF	1.42	0.10	1.37
- NLSq	1.50	0.11	1.51
- AF hidden	1.57	0.14	1.57
- AF hidden and NLSq	1.56	0.29	1.56

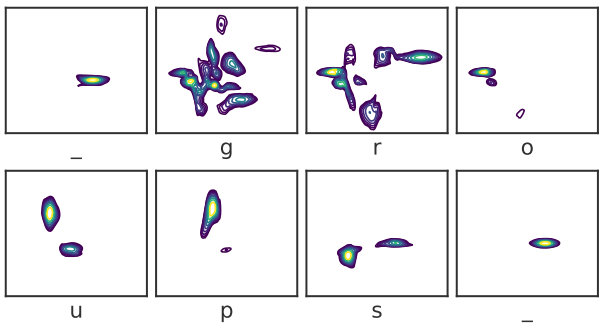


Figure 6. Conditional prior densities corresponding to characters in the string '_groups_' (_ indicates a space), from top left to bottom right. Each figure shows $p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$ for increasing t, where $\boldsymbol{z}_{1:T}$ is sampled from $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$ and $\boldsymbol{x}_{1:T}$ comes from validation.

/ SCF is within 0.05 of the LSTM baseline. These results demonstrate that the combination of AF-in-hidden and the NLSq scalar invertible function induce enough multimodality in the continous distribution to model the discrete data. The AF-only "unigram" model removes the relationships across time in the prior model, effectively dropping the time-dynamics.

The IAF / SCF model performs worse than the other models, which reflects the additional challenges associated with non-autoregressive sampling. The same effect is seen with normalizing flow-based generative models for images (Dinh et al., 2017; Kingma & Dhariwal, 2018), where non-autoregressive models have not reached the state-of-the-art performance. Still, compared to the AF-only baseline the autoregressive model clearly learns important dependencies between characters.

Interestingly, in all models the KL term dominates the ELBO, always accounting for over 90% of the ELBO. This is in stark contrast to previous NLP latent-variable models with strong likelihood models. In these models, the KL term accounts for less than 5% of the ELBO (Bowman et al., 2016; Kim et al., 2018; Xu & Durrett, 2018), or less than 30% of the ELBO when using a specially designed auxiliary loss (Goyal et al., 2017). This indicates that the model 1) is using the latent space to predict each letter, and 2) is rewarded in terms of NLL for accurately encoding the discrete tokens in both the reconstruction term and the KL term.

Table 2 shows model ablations. Without either the NLSq function or the AF-in-hidden dependencies the performance degrades. Once AF-in-hidden is removed, however, further removing NLSq appears to make only a small difference in terms of NLL. These results provide further evidence to our hypothesis that modeling discrete data requires a high degree of multimodality. Furthermore, standard normalizing flows without these additions do not achieve the required flexibility.

Visualizing learned distributions Figure 6 shows the prior densities of AF /AF with $H = 2$. A continuous sequence of 2-vectors $\boldsymbol{z}$ is sampled from $q(\boldsymbol{z}|\boldsymbol{x})$. The AF / AF model is used to evaluate $p(\boldsymbol{z})$, which gives $p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$ at every timestep. The figure shows the series of 8 distributions $p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$ corresponding to the characters "_groups_". In the first plot we can see that given the previous $\boldsymbol{z}_{<t}$ the prior distribution is unimodal, indicating the model identifies that following the continuous representation for "business" there is only one likely token (a space). At the next timestep, however, corresponding to the token that starts the next word, the distribution is highly multimodal, indicating uncertainty of the new word. As the model sees more of the context in the continuous space corresponding to successive characters in the word "groups", the number of modes decreases. In two cases, corresponding to the token following "gro" and the token following "group" the distribution is bimodal, indicating a clear two-way branching decision.

6.2. Polyphonic Music Modeling

Next we consider the polyphonic music modeling task (Boulanger-Lewandowski et al., 2012). Here each timestep consists of an 88-dimensional binary vector indicating the musical notes played. Unlike character-level language modeling where one token appears at each time step, multiple notes are played simultaneously giving a maximum effective vocabulary size of 2^{88}. For this dataset all models are modified so the emission distributions $p(x_t|\boldsymbol{x}_{<t})$ and $p(x_t|\boldsymbol{z})$ are independent Bernoulli distributions instead of Categorical distributions.

Table 3 presents the results, split into model classes. RNN/LSTM is the weakest class, capturing the temporal dependencies but treating the 88 notes as independent. RNN-NADE is the strongest class, explicitly modeling the joint distribution of notes in addition to the temporal dependencies. The rest are different latent variable approaches to this

Table 3. Polyphonic music likelihood results. RNN and RNN-NADE are separated to highlight the difference in modeling class, and results from this work are at the bottom. All numbers are NLL values in nats per note, importance sampling is used to estimate the NLL for latent-variable models. RNN and RNN-NADE from (Boulanger-Lewandowski et al., 2012), TSBN from (Gan et al., 2015), STORN from (Bayer & Osendorfer, 2015), NASMC from (Gu et al., 2015), SRNN from (Fraccaro et al., 2016), DMM from (Krishnan et al., 2017).

Model	Nottingham	Piano	Musedata	JSB
RNN	4.46	8.37	8.13	8.71
RNN-NADE	2.31	7.05	5.6	5.19
TSBN	3.67	7.89	6.81	7.48
STORN	2.85	**7.13**	**6.16**	6.91
NASMC	2.72	7.61	6.89	**3.99**
SRNN	2.94	8.2	6.28	4.74
DMM	2.77	7.83	6.83	6.39
LSTM	3.43	7.77	7.23	8.17
AF / AF	**2.39**	8.19	6.92	6.53
AF / SCF	2.56	8.26	6.95	6.64
IAF / SCF	2.54	8.25	7.06	6.59

problem. They each treat the 88 notes as conditionally independent given a variable-length latent variable. By storing useful information in the latent space these models should out-perform the RNN baseline. All models make different modeling and inference choices and all except DMM include dependencies between observed random variables x_t.

The AF / AF model outperforms all models on the Nottingham dataset, SRNN on the Piano dataset, and TSBN and STORN on the JSB dataset. The AF / AF model also approaches the RNN-NADE model on the Nottingham dataset. AF / AF performs most poorly on the Piano dataset, which has the longest sequences but only 87 individual sequences. The dataset therefore poorly matches the inductive bias of the discrete flow models, which is designed to ingest whole sequences. The AF / SCF model performs slightly worse than AF / AF on all datasets, which is expected given the loss of modeling power. IAF / SCF performs slightly worse than AF / AF but surprisingly better than AF / SCF on all datasets except Musedata. Given the small amount of training data, IAF / SCF overfits less than AF /SCF, explaining the improved generalization despite being overall a weaker model.

Overall, the performance on the polyphonic music datasets demonstrates that the discrete flow model can work at least as well as models which explicitly include connections between the x_ts, and that the weakness of the inference model is made up for by the flexibility of the prior.

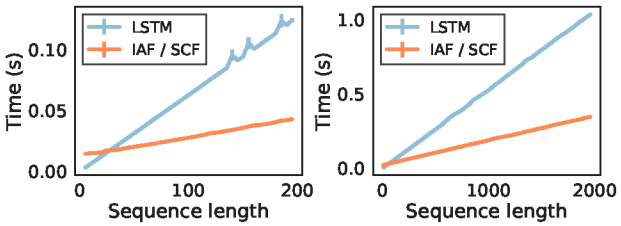


Figure 7. Timing analysis of sentence generation as a function of sequence length, comparing a baseline LSTM model to the IAF / SCF model. Character-level language modeling is shown on the left, the Nottingham polyphonic dataset is on the right. Errorbars show a 95% confidence interval, estimated with 20 generations for each length for both models.

6.3. Non-Autoregressive Generation

While our main goal was to develop a flexible multimodal latent flow model, a secondary goal was to develop a non-autoregressive approach discrete generation. Of the 3 models, IAF / SCF best fits this goal. Therefore we examine the practical speed of this model compared to discrete autoregressive models.

Figure 7 shows generation speed for both tasks. Experiments are run on a single Tesla V100 GPU with a batch size of one, with the IAF / SCF model using an LSTM to implement time-wise conditioning. Compared to the baseline LSTM model, the speedup comes from the fact that in the IAF formulation all of the inputs ϵ are available to the LSTM in parallel and therefore cuDNN can parallelize parts of the computation.

Figure 7 shows that for very short sequences the overhead of the proposed model makes generation slower than the baseline LSTM, whereas after that point the IAF / SCF is faster than the LSTM. This experiment was run with a batch size of 1, for small batch sizes the trend holds while for large batch sizes the additional parallelization afforded by having access to all LSTM inputs becomes less important.

7. Conclusion

This work proposes a latent-variable model for discrete sequences that learns a highly multimodal normalizing flow-based continuous distribution. We show that two flows, AF / AF and AF / SCF, succeed in learning rich multimodal distributions. Furthermore, we show that IAF / SCF, while slightly less accurate, is an efficient approach for non-autoregressive generation. Future work can explore moving to alternate architectures such as those based on self-attention, which give performance and are more parallelizable than LSTMs.

The proposed models can also be adapted for conditional

language modeling for use in e.g. character-level translation. Furthermore,We hope this work encourages further exploration of the interplay between and relative merits of discrete and continuous representations.

References

Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. Character-Level Language Modeling with Deeper Self-Attention. *arXiv preprint arXiv:1808.04444v1*, 2018.

Bayer, J. and Osendorfer, C. Learning Stochastic Recurrent Networks. *arXiv preprint arXiv:1411.7610*, 2015. URL http://arxiv.org/abs/1411.7610.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating Sentences from a Continuous Space. *Proceedings of CoNLL*, 2016. URL http://arxiv.org/abs/1511.06349.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational Lossy Autoencoder. *Proceedings of ICLR*, 2017.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. A Recurrent Latent Variable Model for Sequential Data. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 2980–2988, 2015. ISSN 10495258. doi: 10.3109/02713683.2016.1141962.

Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., and Courville, A. Recurrent Batch Normalization. *Proceedings of ICLR*, 2017.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *Proceedings of ICLR*, 2017.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential Neural Models with Stochastic Layers. *30th Conference on Neural Information Processing Systems*, 2016.

Gan, Z., Li, C., Henao, R., Carlson, D., and Carin, L. Deep Temporal Sigmoid Belief Networks for Sequence Modeling. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 2467–2475, 2015.

Goyal, A., Sordoni, A., Côté, M.-A., Ke, N. R., and Bengio, Y. Z-Forcing: Training Stochastic Recurrent Networks. *31st Conference on Neural Information Processing Systems*, 2017.

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. Non-Autoregressive Neural Machine Translation. *Proceedings of ICLR*, 2018.

Gu, S., Ghahramani, Z., and Turner, R. E. Neural Adaptive Sequential Monte Carlo. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 2629–2637, 2015. URL http://arxiv.org/abs/1506.03338.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural Autoregressive Flows. *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL http://arxiv.org/abs/1804.00779.

Kaiser, Ł., Roy, A., Vaswani, A., Parmar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. Fast Decoding in Sequence Models Using Discrete Latent Variables. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Kim, Y., Wiseman, S., Miller, A. C., Sontag, D., and Rush, A. M. Semi-Amortized Variational Autoencoders. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Kingma, D. P. and Dhariwal, P. Glow : Generative Flow with Invertible 1x1 Convolutions. *arXiv preprint arXiv:1807.03039v1*, 2018.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *Proceedings of ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving Variational Inference with Inverse Autoregressive Flow. *29th Conference on Neural Information Processing Systems*, 2016.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured Inference Networks for Nonlinear State Space Models. *Proceedings of AAAI*, 2017.

Lee, J., Mansimov, E., and Cho, K. Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. *Proceedings of EMNLP*, 2018.

Mikolov, T., Sutskever, I., Deoras, A., Le, H.-S., Kombrink, S., and Cernocky, J. Subword language modeling with neural networks. *Unpublished*, 2012. URL http://www.fit.vutbr.cz/{~}imikolov/rnnlm/char.pdf.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. *Advances in Neural Information Processing Systems*, 2017.

Press, O. and Wolf, L. Using the Output Embedding to Improve Language Models. *Proceedings ofthe 15th Conference ofthe European Chapter ofthe Association for Computational Linguistics*, 2017.

Rezende, D. J. and Mohamed, S. Variational Inference with Normalizing Flows. *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *Proceedings of ICLR*, 2017.

Tabak, E. G. and Vanden-Eijnden, E. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel Recurrent Neural Networks. *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *Proceedings of the 35 th International Conference on Machine Learning*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need. *31st Conference on Neural Information Processing Systems*, pp. 5998–6008, 2017.

Xu, J. and Durrett, G. Spherical Latent Spaces for Stable Variational Autoencoders. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *Proceedings of the 34th International Conference on Machine Learning*, 2017.