A Robust Normalizing Flow using Bernstein-type Polynomials

Sameera Ramasinghe[1]
sameera.ramasinghe@anu.edu.au
Kasun Fernando[2]
kasun.akurugodage@utoronto.ca
Salman Khan[3]
salman.khan@mbzuai.ac.ae
Nick Barnes[1]
nick.barnes@anu.edu.au

[1] Australian National University
[2] University of Toronto
[3] Mohamed bin Zayed University of AI

Abstract

Modeling real-world distributions can often be challenging due to sample data that are subjected to perturbations, *e.g.*, instrumentation errors, or added random noise. Since flow models are typically nonlinear algorithms, they amplify these initial errors, leading to poor generalizations. This paper proposes a framework to construct Normalizing Flows (NFs) which demonstrate higher robustness against such initial errors. To this end, we utilize Bernstein-type polynomials inspired by the optimal stability of the Bernstein basis. Further, compared to the existing NF frameworks, our method provides compelling advantages like theoretical upper bounds for the approximation error, better suitability for compactly supported densities, and the ability to employ higher degree polynomials without training instability. We conduct a theoretical analysis and empirically demonstrate the efficacy of the proposed technique using experiments on both real-world and synthetic datasets.

1 Introduction

Modeling the probability distribution of a set of observations, *i.e.*, generative modeling, is a crucial task in machine learning. It enables the generation of synthetic samples using the learned model and allows the estimation of the likelihood of a data sample. This field has met with great success in many problem domains including image generation [20, 24, 30], audio synthesis [13, 33], reinforcement learning [31, 39], noise modeling [1], and simulating physics experiments [40, 41]. In the recent past, deep neural networks such as generative adversarial networks (GANs) and variational autoencoders (VAEs) have been widely adopted in generative modeling due to their success in modeling high dimensional distributions. However, they entail several limitations: 1) exact density estimation of arbitrary data points is not possible, and 2) training can be cumbersome due to aspects such as mode collapse, posterior collapse and high sensitivity to architectural design of the model [25].

In contrast, normalizing flows (NFs) are a category of generative models that enable exact density computation and efficient sampling (for theoretical foundations see Appendix 1.1 and

references therein). As a result, NFs have been gaining increasing attention from the machine learning community since the seminal work of Rezende and Mohamed [35]. In essence, NFs consist of a series diffeomorphisms that transforms a simple distribution into a more complex one, and must be designed so that the Jacobian determinants of these diffeomorphisms can be efficiently calculated (This is, in fact, an essential part of the implementation). To this end, two popular approaches have been proposed so far: 1) efficient determinant calculation methods such as Berg et al. [4], Grathwohl et al. [18], Lu and Huang [30], and 2) *triangular maps* [9, 10, 23]. The key benefit of triangular maps is that their Jacobian matrices are triangular, and hence, the calculation of Jacobian determinants takes only $O(n)$ steps as opposed to the $O(n^3)$ complexity of the computation of a determinant of an unconstrained $n \times n-$matrix. In this paper, we focus only on triangular maps.

On the one hand, it is not a priori clear whether such a constrained class of maps is expressive enough to produce sufficiently complex transformations. Interestingly, [6] showed that, given two probability densities, there exists a unique increasing triangular map that transforms one density to the other. Consequently, the constructed NFs should be *universal*, *i.e.*, dense in the class of increasing triangular maps, in order to approximate those density transformations with arbitrary precision. But it is observed in Jaini et al. [23] that, despite many NFs being triangular, they are not universal. To remedy this, most have reverted to the empirical approach of stacking several transformations together, thereby increasing the expressiveness of the model. Alternatively, there are NFs that use genuinely universal transformations. Many such methods employ coupling functions based on polynomials, *e.g.*, sum-of-squares (SOS) polynomials in Jaini et al. [23], cubic splines in Durkan et al. [11] or rational quadratic splines in Durkan et al. [12]. Here, we employ another class of polynomials called Bernstein-type polynomials to construct a universal triangular flow which henceforth is called Bernstein-type NF. Our universality proof is a consequence of [6], and unlike the proofs in the previous literature, is constructive, and hence, yields analytic expressions for approximations of known density transformations; see Section 2.4.

On the other hand, noise is omnipresent in data. Sample data can be subjected to perturbations due to experimental uncertainty (instrumentation errors or added random noise). It is well-known that nonlinear systems amplify these initial errors and produce drastically different outcomes even for small changes in the input data; see Higham [19], Lanckriet et al. [28], Taylor [37]. In terms of (deep) classifiers, robustness is often studied in the context of adversarial attacks, where the performance of the classifier should be robust against specific perturbations of the inputs. Similarly, for generative models, this translates to robustly modeling a distribution in the presence of perturbed data. Recently, [8] analyzed the robustness of deep generative models against random perturbations of the inputs, where they designed a VAE variant that is robust to random perturbations. Similarly, [27] also proposed a heuristic-based method to make deep generative models robust against perturbations of the inputs.

As any other nonlinear model, NFs are also susceptible of numerical instabilities. Unless robust, trained NFs may amplify initial errors, and demonstrate out-of-distribution sample generation and poor generalization to unseen data. For instance, consider an NF modeling measured or generated velocities (energies) of molecular movements; see [26]. Similarly, consider a scenario where we intend to flag certain samples of a random process as out-of-distribution data. If the training data is susceptible to noise, the measured log-likelihoods of the test samples may significantly deviate from the true values unless the NF model is robust.

Therefore, it is imperative that the robustness of NFs is investigated during their construction and implementation. Despite this obvious importance, robustness of NFs has not been theoretically or even experimentally studied in the previous literature, unlike other deep

generative models. One of the key motivations behind this work is to fill this void. Accordingly, we show that Bernstein-type polynomials are ideal candidates for the construction of NFs that are not only universal but also robust. Robustness of Bernstein-type NFs follow from the *optimal stability* of the Bernstein basis [15, 16]; see also, Section 2.5.

Recently, Bernstein polynomials have also been used in conditional transformation models to due to their versatility; see for example, Hothorn et al. [22], Hothorn and Zeileis [21], [3] and references therein. In contrast, here, we introduce a novel approach of building NFs using Bernstein polynomials.

In summary, apart from collecting, organizing and summarising in a coherent fashion the appropriate theoretical results which were scattered around the mathematical literature, we **1)** deduce, in Theorem 3, the universality of Bernstein flows, **2)** state and prove, in Theorem 2, a *strict* monotonicity result of Bernstein polynomials which has been mentioned without proof and used in Farouki [14], **3)** prove, in Theorem 1, that, in *any* NF, it is enough to consider compactly supported targets (in the previous literature was implicitly assumed without proper justification), **4)** theoretically establish that, compared to other polynomial-based flow models, Bernstein-type NFs demonstrate superior robustness to perturbations in data. To our knowledge, ours is the first work to discuss robustness in NFs, **5)** discuss a theoretical bound for the rate of convergence of Bernstein-type NFs, which, to our knowledge, has not been discussed before in the context of NFs, and **6)** propose a practical framework to construct normalizing flows using Bernstein-type polynomials and empirically demonstrate that theoretically discussed properties hold in practice.

Moreover, compared to previous NF models, our method has several additional advantages such as suitability for approximating compactly supported target densities; see Section 2.2, the ability to increase the expressiveness by increasing the polynomial degree at no cost to the training stability; see Section 2.2, and being able to invert easily and accurately due to the availability of efficient root finding algorithms; see Section 2.3.

2 Theoretical foundations of the Bernstein-type NF

Here, we elaborate on the desirable properties of Berstein-type polynomials and their implication to our NF model. The mathematical results taken directly from existing literature are stated as facts with appropriate references. The proofs of Theorems 1, 2 and 3, appear in Appendix 2. Also, in each subsection, we point out the advantages of our model (over existing models) based on the properties discussed. We point the reader to Appendix 1.1 for a brief discussion on triangular maps and other preliminaries.

2.1 Bernstein-type polynomials

The degree n Bernstein polynomials, $\binom{n}{k}x^k(1-x)^{n-k}$, $0 \leq k \leq n$, $n \in \mathbb{N}$, were first introduced by Bernstein in his constructive proof of the Weierstrass theorem in Bernstein [5]. In fact, given a continuous function $f : [0,1] \to \mathbb{R}$, its degree n Bernstein approximation, $B_n(f) : [0,1] \to \mathbb{R}$, given by

$$B_n(f)(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right)\binom{n}{k}x^k(1-x)^{n-k}, \tag{1}$$

is such that $B_n(f) \to f$ uniformly in $[0,1]$ as $n \to \infty$. Moreover, Bernstein polynomials form a basis for degree n polynomials on $[0,1]$. More generally, polynomials of Bernstein-type can be defined as follows.

Definition 1. *A degree n polynomial of Bernstein-type is a polynomial of the form*

$$B_n(x) = \sum_{k=0}^{n} \alpha_k \binom{n}{k} x^k (1-x)^{n-k},\ x \in [0,1], \tag{2}$$

where α_k, $0 \leq k \leq n$ are some real constants.

Remark 1. *Polynomials of Bernstein-type on an arbitrary closed interval $[a,b]$ are defined by composing B_n with the linear map that sends $[a,b]$ to $[0,1]$, $L_{a,b}(x) = \frac{x-a}{b-a}$. So, Bernstein-type polynomials on $[a,b]$ take the form $B_n \circ L_{a,b}$. Hereafter, we denote degree n Bernstein-type polynomials by B_n regardless of the domain.*

As we shall see below, one can control various properties of Bernstein-type polynomials like strict monotonicity, range and universality by specifying conditions on the coefficients, and the error of approximation depends on the degree of the polynomials used.

2.2 Easier control of the range and suitability for compact targets

The supports of distributions of samples used when training and applying NFs are not fixed. So, it is important to be able to easily control the range of the coupling functions. In the case of Bernstein-type polynomials, B_ns, this is very straightforward. Note that if B_n is defined on $[a,b]$, then $B_n(a) = \alpha_0$ and $B_n(b) = \alpha_n$. Therefore, one can fix the values of a Bernstein-type polynomial at the end points of $[a,b]$ by fixing α_0 and α_n. So, if B_n is increasing (which will be the case in our model; see Section 2.3), then its range is $[\alpha_0, \alpha_n]$. This translates to a significant advantage when training for compactly supported targets because we can achieve any desired range $[c,d]$ (the support of the target) by fixing $\alpha_0 = c$ and $\alpha_n = d$ and letting only α_k, $0 < k < n$ vary. So, B_ns are ideal for modeling compactly supported targets. In fact, in most other methods except splines in Durkan et al. [11, 12], either there is no obvious way to control the range or the range is infinite. We present the following theorem to establish that for the purpose of training, we can assume the target has compact support (up to a known diffeomorphism).

Theorem 1. *Let $I_j, j = 1,2,3$ be measurable subsets of $\mathbb{R}^d$. Suppose I_1 is the support of the target P_x, I_2 is the support of the prior P_z, $\mathfrak{F}$ is the class of coupling functions with ranges contained in I_3 and $h : I_3 \to I_1$ is a diffeomorphism. If P_y is the distribution on I_3 such that $h_* P_y = P_x$, then*

$$\arg\min_{f \in \mathfrak{F}} \mathrm{KL}(P_x \| (h \circ f)_* P_z) = \arg\min_{f \in \mathfrak{F}} \mathrm{KL}(P_y \| f_* P_z). \tag{3}$$

In the previous literature that uses transformations with compact range, this fact was implicitly assumed without proper justification. As a consequence of the above theorem, our coupling functions having finite ranges is not a restriction, and in *any* NF model, even if the target density is not compactly supported, the learning procedure can be implemented by first converting the target density to a density with a suitable compact support via a diffeomorphism, and then training on the transformed data. Since we deal with compactly supported targets, in practice, we do not need to construct deep architectures (with a higher number of layers), as we can increase the degree of the polynomials to get a better approximation. In other polynomial based methods, a practical problem arises because the higher order polynomials could predict extremely high values initially leading to unstable gradients (*e.g.*, [23]). In contrast, we can avoid that problem as the range of our transformations can be explicitly controlled from the beginning by fixing α_0 and α_n .

2.3 Strict monotonicity and efficient inversion

In triangular flows, the coupling maps are expected to be invertible. Since strict monotonocity implies invertibility, it is sufficient that the B_ns we use are strictly monotone.

Theorem 2. *Consider the Bernstein-type polynomial B_n in (2). Suppose $\alpha_0 < \alpha_1 < \cdots < \alpha_n$. Then, B_n is strictly increasing on $[0,1]$.*

This result was mentioned as folklore without proof and used in Farouki [14]. On the other hand, in Lindvall [29], the conclusion is stated with monotonicity and not *strict* monotonicity (which is absolutely necessary for invertibility). Hence, we added a complete proof of the statement in Appendix 2. According to this result, the strict monotonicity of B_ns depends entirely on the strict monotonicity of the coefficients α_ks. It is easy to see that the assumption of strict monotonicity of the coefficients is not a further restriction on the optimization problem. For example, if the required range is $[c,d]$, we can take $\alpha_{n-k} = c + (d-c)(1+v_0^2+\cdots+v_k^2)^{-1}$ where v_ks are real valued. This converts the constrained problem of finding α_ks to an unconstrained one of finding v_ks. Alternatively, we can take $\alpha_0 = c$ and $\alpha_k = |v_1| + \cdots + |v_k|$, and after each iteration, linearly scale α_ks in such a way that $\alpha_n = d$. After guaranteeing invertibility, we focus on computing the inverse, *i.e.*, at each iteration, given x we solve for $z \in [0,1]$,

$$B_n(z) = \sum_{k=0}^{n} \alpha_k \binom{n}{k} z^k (1-z)^{n-k} = x \iff \sum_{k=0}^{n} (\alpha_k - x) \binom{n}{k} z^k (1-z)^{n-k} = 0 \tag{4}$$

because Bernstein polynomials form a partition of unity on $[0,1]$. So, finding inverse images, *i.e.*, solving the former is equivalent to finding solutions to the latter. Due to our assumption of increasing α_ks, B_n is increasing, and has at most one root on $[0,1]$. The condition $(\alpha_0 - x)(\alpha_n - x) < 0$ (which can be easily checked) guarantees the existence of a unique solution, and hence, the invertibility of the original transformation.

Due to the extensive use of Bernstein-type polynomials in computer-aided geometric design, there are several well-established efficient root finding algorithms at our disposal [36]. For example, the parabolic hull approximation method in Rajan et al. [34] is ideal for higher degree polynomials with fewer roots (in our case, just one) and has cubic convergence for simple roots (better than both the bijection method and Newton's method). Further, because of the numerical stability described in Section 2.5 below, the use of Bernstein-type polynomials in our model minimizes the errors in such root solvers based on floating–point arithmetic. Even though inverting splines are easier due to the availability of analytic expressions for roots, compared to all other other NF models, we have more efficient and more numerically stable algorithms that allow us to reduce the cost of numerical inversion in our setting.

2.4 Universality and the explicit rate of convergence

In order to guarantee universality of triangular flows, we need to use a class of coupling functions that well-approximates increasing continuous functions. This is, in fact, the case for B_ns, and hence, we have the following theorem whose proof we postpone to Appendix 2.

Theorem 3. *Bernstein-type normalizing flows are universal.*

The basis of all the universality proofs of NFs in the existing literature is that the learnable class of functions is dense in the class of increasing continuous functions. In contrast, the argument we present here is constructive. As a result, we can write down sequences of approximations for (known) transformations between densities explicitly; see Appendix 4.

In the case of cubic-spline NFs of Durkan et al. [11], it is known that for $k = 1,2,3$ and 4, when the transformation is k times continuously differentiable and the bin size is h, the error is $O(h^k)$ [2, Chapter 2]. However, we are not aware of any other instance where an error bound is available. Fortunately for us, the error of approximation of a function f by its Bernstein polynomials has been extensively studied. We recall from Voronovskaya [38] the following error bound: for $f : [0,1] \to \mathbb{R}$ twice continuously differentiable

$$B_n(f) - f = \frac{x(1-x)}{2n} f''(x) + o(n^{-1}) \tag{5}$$

and this holds for an arbitrary interval $[a,b]$ with $x(1-x)$ replaced by $(x-a)(b-x)$. Since the error estimate is given in terms of the degree of the polynomials used, we can improve the optimality of our NF by avoiding unnecessarily high degree polynomials. This allows us to keep the number of trainable parameters under control in our NF model. It can be shown that the error $O(n^{-1})$ above does not necessarily improve when SOS polynomials are used instead; see Appendix 3. In our NF, at each step, the estimation is done using a univariate polynomial, and hence, the overall convergence rate is, in fact, the minimal univariate convergence rate of $O(n^{-1})$ (equivalently, the error upper bound is the maximum of univariate upper bounds), and in general, cannot be improved further regardless of how regular the density transformation is. However, our experiments (in Section 4.3) show that our model on average has a significantly smaller error than the given theoretical upper-bound.

2.5 Robustness of Bernstein-type normalizing flows

In this section, we recall some known results in Farouki and Goodman [15], Farouki and Rajan [16] about the optimal stability of the Bernstein basis. The two key ideas are that smaller *condition numbers* lead to smaller numerical errors and that the Bernstein basis has the optimal condition numbers compared to other polynomial bases.

To illustrate this, let $p(x)$ be a polynomial on $[a,b]$ of degree n expressed in terms of a basis $\{\phi_k\}_{k=0}^n$, *i.e.*,

$$p(x) = \sum_{k=0}^{n} c_k \phi_k(x), x \in [a,b]. \tag{6}$$

Let c_k be randomly perturbed, with perturbations δ_k where the relative error $\delta_k/c_k \in (-\varepsilon, \varepsilon)$. Then the total pointwise perturbation is

$$\delta(x) = \sum_{k=0}^{n} \delta_k \phi_k(x) \implies |\delta(x)| \leq \sum_{k=0}^{n} |\delta_k \phi_k(x)| \leq \varepsilon \sum_{k=0}^{n} |c_k \phi_k(x)| \leq \varepsilon C_\phi(p(x)), \tag{7}$$

where $C_\phi(p(x)) := \sum_{k=0}^n |c_k \phi_k(x)|$ is the condition number for the total perturbation with respect to the basis ϕ_k. It is clear that $C_\phi(p(x))$ controls the magnitude of the total perturbation.

According to Farouki and Goodman [15], if $\phi = \{\phi_k\}_{k=0}^n$ and $\psi = \{\psi_k\}_{k=0}^n$ are non-negative bases for polynomials of degree n on $[a,b]$, and for all j, latter is a non-negative linear combination of the former, that is, $\psi_j = \sum_{k=0}^n M_{jk} \phi_k$ with $M_{jk} \geq 0$. Then, for any polynomial $p(x)$,

$$C_\phi(p(x)) \leq C_\psi(p(x)). \tag{8}$$

For $0 \leq a < b$, the Bernstein polynomials and the power monomials, $\{1, x, x^2, \ldots, x^n\}$, are non-negative bases on $[a,b]$. It is true that the latter is a positive linear combination of the former but *not* vice-versa; see Farouki and Goodman [15]. Therefore, Bernstein polynomial basis has the lowest condition number out of the two. This means that the change in the value of a polynomial caused by a perturbation of coefficients is always smaller in the Bernstein basis than in the power basis. A more involved computation gives

$$\tilde{C}_{\phi}(x_0) = \left(\frac{m!}{|p^{(m)}(x_0)|} \sum_{k=0}^{n} |c_k \phi_k(x_0)| \right)^{1/m} \tag{9}$$

as the condition number that controls the computational error for a $m-$fold root x_0 of $p(x)$ in $[a,b]$; see Farouki and Rajan [16]. There, it is proved that if $\tilde{C}_{\psi}(x_0)$ and $\tilde{C}_{\phi}(x_0)$ denote the condition numbers for finding roots of *any* polynomial on $[0,1]$ in the power and the Bernstein bases on $[0,1]$, respectively, then $\tilde{C}_{\phi}(x_0) < \tilde{C}_{\psi}(x_0)$ for $x_0 \in (0,1]$ and $\tilde{C}_{\phi}(0) = \tilde{C}_{\psi}(0)$. This means that the change in the value of a root of a polynomial caused by a perturbation of coefficients is always smaller in the Bernstein basis than in the power basis.

In fact, a universal statement is true: Among all non-negative basis on a given interval, the Bernstein polynomial basis is *optimally stable* in the sense that no other non-negative basis gives smaller condition numbers for the values of polynomials (see [32, Theorem 2.3]) and no other basis expressible as non-negative combinations of the Bernstein basis gives smaller condition numbers for the roots of polynomials (see [15, Section 5.3]).

In particular, B_ns are systematically more stable than the polynomials in the power form when determining roots (for example, when inverting) and evaluation (for example, when finding image points). As a result, when polynomials are used to construct NFs (for example, Q-NSF based on quadratic or cubic splines, SOS based on some of square polynomials and our NF based on Bernstein-type polynomials, ours yields the most numerically stable NF, i.e., it is theoretically impossible for them to be more robust. Our experiments in Section 4.2, while confirming this, demonstrate that our method outperforms even the NFs that are *not* based on polynomials.

3 Construction of the Bernstein-type normalizing flow

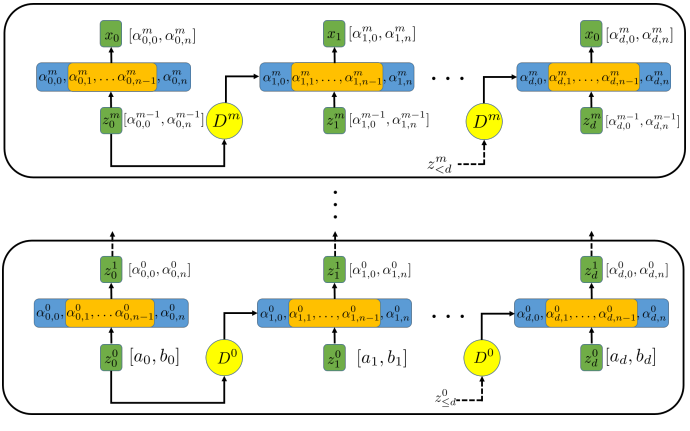

Figure 1: Overall Bernstein NF architecture with $m+1$ layers for d-dimensional distributions. The ranges of transformations are within brackets and trainable coefficients are in orange boxes.

In this section, we describe the construction of normalizing flows using the theoretical framework established in Section 2 for compactly supported targets. For this, we employ a MADE style network [17]. Consider a d-dimensional source $P_z(\mathbf{z})$ and a d-dimensional target $P_x(\mathbf{x})$. Then, the element-wise mapping between the components x_j and z_j is approximated using a Bernstein-type polynomial as $x_j = B^j_n(z_j)$. We obtain the parameters of $B^j_n(z_j)$ using a neural network which is conditioned on $z_{<j}$. This ensures a triangular mapping between the distributions. We fix α_0 and α_n to be constants, and thus, define the range of each

transformation; see Section 2.2. Moreover, as per Theorem 2, α_ks need to be strictly increasing for a transformation to be strictly increasing. However, when we convert this constrained problem to an unconstrained one as proposed in Section 2.3, we obtain v_ks using the neural network and then calculate α_ks as described.

For each B_n^j, we employ a fully-connected neural net with three layers to obtain the parameters, except in the case of B_n^0 in which we directly optimize the parameters. Figure 1 illustrates a model architecture with $m+1$ layers and degree n polynomials for d-dimensional distributions. Here, there are $(n-1)(m+1)$ variable coefficients altogether. We use maximum likelihood to train the model with a learning rate 0.01 with a decay factor of 10% per 50 iterations. All the weights are initialized randomly using a standard normal distribution.

4 Experiments

In this section, we summarise our empirical evaluations of the proposed model based on both real-world and synthetic datasets and compare our results with other NF methods.

4.1 Modeling sample distributions

Table 1: Test log-likelihood comparison against the state-of-the-art on real-world datasets (higher is better). Results for competing methods are extracted from Durkan et al. [12] where error bars correspond to two standard deviations. Log-likelihoods are averaged over 5 trials for Bernstein.

MODEL	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
FFJORD	0.46 ± 0.01	8.59 ± 0.12	-14.92 ± 0.08	-19.43 ± 0.04	157.40 ± 0.19
GLOW	0.42 ± 0.01	12.24 ± 0.03	-16.99 ± 0.02	-10.55 ± 0.45	156.95 ± 0.28
MAF	0.45 ± 0.01	12.35 ± 0.02	-17.03 ± 0.02	-10.92 ± 0.46	156.95 ± 0.28
NAF	0.62 ± 0.01	11.96 ± 0.33	-15.09 ± 0.04	-8.86 ± 0.15	157.73 ± 0.04
BLOCK-NAF	0.61 ± 0.01	12.06 ± 0.09	-14.71 ± 0.38	-8.95 ± 0.07	157.36 ± 0.03
RQ-NSF (AR)	0.66 ± 0.01	13.09 ± 0.02	-14.01 ± 0.03	-9.22 ± 0.48	157.31 ± 0.28
Q-NSF (AR)	0.66 ± 0.01	13.09 ± 0.02	-14.01 ± 0.03	-9.22 ± 0.48	157.31 ± 0.28
SOS	0.60 ± 0.01	11.99 ± 0.41	-15.15 ± 0.10	-8.90 ± 0.11	157.48 ± 0.41
BERNSTEIN	0.63 ± 0.01	12.81 ± 0.01	-15.11 ± 0.02	-8.93 ± 0.08	157.13 ± 0.11

We conducted experiments on four datasets from the UCI machine-learning repository and BSDS300 dataset. Table 1 compares the obtained test log-likelihood against recent flow-based models. As illustrated, our model achieves competitive results on all of the five datasets. We observe that our model consistently reported a lower standard deviation which may be attributed to the robustness of our model.

Table 2: Test log-likelihood comparison against the state-of-the-art on image datasets (higher is better). Results for competing methods are extracted from Jaini et al. [13]. Note that the first three models use multi-scale convolutional architectures.

MODEL	MNIST	CIFAR10
REAL-NVP	-1.06	-3.49
FFJORD	-0.99	-3.40
GLOW	-1.05	-3.35
MAF	-1.89	-4.31
MADE	-2.04	-5.67
SOS	-1.81	-4.18
BERNSTEIN	-1.54	-4.04

We also applied our method to two low-dimensional image datasets, CIFAR10 & MNIST. The results are reported in Table 2. Among the methods that do not use multi-scale convolutional architectures, we obtain the optimal results. In addition, we tested our model on several toy datasets (shown in Figure 3). Note that these 2D datasets contain multiple modes, sharp jumps and are not fully supported. So, the densities are not that obvious to learn. Despite the difficulties, our model is able to estimate the given distributions in a satisfactory manner.

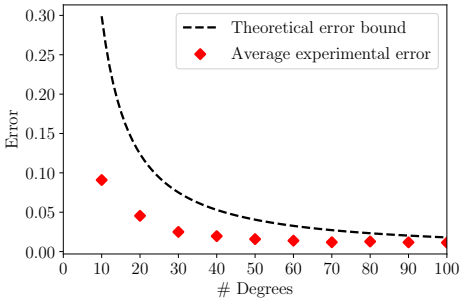

Figure 2: Error bound vs average experimental error.

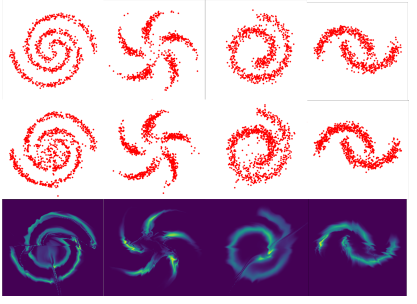

Figure 3: Qualitative results for modeling the toy distributions. *From the top row*: ground truth, prediction, and predicted density.

4.2 Robustness

In order to experimentally verify that Bernstein-type NFs are more numerically stable than other polynomial based NFs (as claimed in Section 2.5) we use a standard idea in the literature; see Taylor [37].

We add i.i.d. noise, sampled from a Uniform$[0, 10^{-2}]$, to the five datasets included in Table 1, and measure the change in the test log-likelihood as a fraction of the standard deviation (so that the change is in terms of standard deviations). In practice, values of the experimented datasets are rescaled to a magnitude around unity. In signal processing, a good SNR is considered to be above 40DB which is used in most real-world cases. Here, we have chosen a noise order of 10^{-2} because our intention is to demonstrate that a SNR level below or even around that range can affect the performance of NFs.

For a fair comparison, we train all the models from scratch on the noise-free train set using the codes provided by the authors, strictly following the instructions in the original works to the best of our ability. Then, we test the models on the noise-free test set. We run the above experiment 5 times to obtain the standard deviation σ and mean μ of the test log-likelihood. Next, we add noise to the training set, retrain the model, and obtain the test log-likelihood y on the noise-free test set. Finally, we obtain the metric $\frac{|y-\mu|}{\sigma}$ which we report in Table 3.

Table 3: Test log-likelihood drop for random initial errors, relative to the original standard deviations.

MODEL	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
FFJORD	2.7	4.4	3.2	1.7	6.6
REAL-NVP	2.4	4.2	3.6	1.4	7.4
GLOW	2.1	4.1	2.3	0.8	6.9
NAF	2.2	3.7	3.3	0.7	6.6
MAF	2.4	4.4	3.9	0.8	7.1
MADE	2.1	4.6	3.6	2.4	8.1
RQ-NSF	2.3	5.4	4.1	0.9	7.8
SOS	2.1	1.7	1.9	1.6	6.1
BERNSTEIN	1.1	1.3	1.1	0.6	2.3

As expected, Bernstein NF demonstrate the lowest relative change in performance, implying the robustness against random initial errors. In fact, other models are not robust: even small initial errors consistently (at least in 4 out of 5 datasets) created changes *larger* than $1.645\,\sigma$ (corresponding to the two 5% tails of the distribution of errors) where σ is the original standard deviation of error. In comparison, the change in our NF consistently (4 out of 5) was well-within the acceptable range and is at most $1.3\,\sigma$. In the remaining dataset, change

in our model is $2.3\,\sigma$ while in all other models it's *more* than $6\,\sigma$.

4.3 Validation of the theoretical error upper-bound

The degree $n\,(\geq 5)$ Bernstein approximation of $f \in C^3[0,1]$ has an error upper-bound

$$E_n = n^{-1}\|\rho^2 f^{(2)}\|_\infty + n^{-3/2}\|\rho^3 f^{(3)}\|_\infty \tag{10}$$

where $\rho(x) = \sqrt{x(1-x)}$ [7, Chapter 4]. Now, we verify this using a Kumarswamy$(2,5)$ distribution as the prior and Uniform$[0,1]$ as the target. Let $f(x) = 1-(1-x^2)^5$ and B_n be the learned degree n Bernstein-type polynomial. The average error, $\int_0^1 |f(x)-B_n(x)|\,dx$, obtained using the learned B_ns and E_n satisfying

$$1.25n^{-1} + 5n^{-3/2} < E_n < 1.25n^{-1} + 5.5n^{-3/2} \tag{11}$$

are plotted in Figure 2. It shows that the observed (average) error is smaller than this theoretical upper-bound. In the NF, we have used a single layer and increased the degree of the polynomial from 10 to 100. The NF model was stable even when the degree 100 polynomial was used. So, this experiment also demonstrates that our model is, in fact, stable even when higher degree polynomials are used (as claimed in Section 2.2).

5 Conclusion

We propose a novel method to construct a universal autoregressive NFs with Bernstein-type polynomials as the coupling functions, and is the first instance of robustness of NFs being discussed. We show that Bernstein-type NFs posses advantages like true universality, robustness against initial and round-off errors, efficient inversion, having an explicit convergence rate, and better training stability for higher degree polynomials.

References

[1] Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. Noise flow: Noise modeling with conditional normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3165–3173, 2019.

[2] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. *The Theory of Splines and their Applications*. Academic Press, 1967.

[3] Philipp F. M. Baumann, Torsten Hothorn, and David Rügamer. Deep conditional transformation models. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 3–18. Springer International Publishing, 2021.

[4] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

[5] S. Bernstein. Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. *Communications of the Kharkov Mathematical Society*, pages 1–2, 1912.

[6] Vladimir Igorevich Bogachev, Aleksandr Viktorovich Kolesnikov, and Kirill Vladimirovich Medvedev. Triangular transformations of measures. *Sbornik: Mathematics*, 196(3):309, 2005.

[7] J. Bustamante. *Bernstein Operators and Their Properties*. Birkhauser, 2017.

[8] Filipe Condessa and Zico Kolter. Provably robust deep generative models. *arXiv preprint arXiv:2004.10608*, 2020.

[9] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[10] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[11] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-spline flows. *arXiv preprint arXiv:1906.02145*, 2019.

[12] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *arXiv preprint arXiv:1906.04032*, 2019.

[13] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, et al. Universal audio synthesizer control with normalizing flows. *arXiv preprint arXiv:1907.00971*, 2019.

[14] R. T. Farouki. Convergent inversion approximations for polynomials in bernstein form. *Comput. Aided Geom. Des.*, 17(2):179–196, 2000.

[15] R. T. Farouki and T. N. T. Goodman. On the optimal stability of the Bernstein basis. *Mathematics of Computation*, 65(216):1553–1566, 1996.

[16] R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design 4*, 4(3):191–216, 1987.

[17] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

[18] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[19] N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 1996.

[20] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[21] Torsten Hothorn and Achim Zeileis. Predictive distribution modelling using transformation forests. *Journal of Computational and Graphical Statistics*, 2021.

[22] Torsten Hothorn, Lisa Möst, and Peter Bühlmann. Most likely transformations. *Scandinavian Journal of Statistics*, 45(1):110–134, 2018.

[23] Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. *arXiv preprint arXiv:1905.02325*, 2019.

[24] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[25] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1, 2020.

[26] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pages 5361–5370. PMLR, 2020.

[27] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *2018 ieee security and privacy workshops (spw)*, pages 36–42. IEEE, 2018.

[28] G. R. G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

[29] T. Lindvall. *Lectures on the coupling method*. Dover Publications, 2002.

[30] You Lu and Bert Huang. Woodbury transformations for deep generative flows. *Advances in Neural Information Processing Systems*, 33, 2020.

[31] Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pages 430–444. PMLR, 2020.

[32] J. M. Peña. B-splines and optimal stability. *Mathematics of Computation*, 66(220): 1555–1560, 1997.

[33] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[34] V. T. Rajan, S. R. Klinkner, and R. T. Farouki. Root isolation and root approximation for polynomials in Bernstein form. Technical Report RC14224, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, N.Y. 10598, 1988.

[35] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[36] M. R. Spencer. *Polynomial Real Root Finding in Bernstein Form*. PhD thesis, Brigham Young University, 1994.

[37] J. R. Taylor. *An introduction to error analysis: the study of uncertainties in physical measurements*. Mill Valley, Calif, University Science Books, 1982.

[38] E. Voronovskaya. Détermination de la forme asymptotique d'approximation des fonctions par les polynômes de M. Bernstein. *Doklady Akademii Nauk SSSR*, pages 79–85, 1932.

[39] Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. Improving exploration in soft-actor-critic with normalizing flows policies. *arXiv preprint arXiv:1906.02771*, 2019.

[40] Peter Wirnsberger, Andrew J Ballard, George Papamakarios, Stuart Abercrombie, Sébastien Racanière, Alexander Pritzel, Danilo Jimenez Rezende, and Charles Blundell. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14):144–112, 2020.

[41] Kaze WK Wong, Gabriella Contardo, and Shirley Ho. Gravitational-wave population inference with deep flow-based generative network. *Physical Review D*, 101(12): 123005, 2020.