Cubic-Spline Flows

Conor Durkan [*1] **Artur Bekasov** [*1] **Iain Murray** [1] **George Papamakarios** [1]

Abstract

A normalizing flow models a complex probability density as an invertible transformation of a simple density. The invertibility means that we can evaluate densities and generate samples from a flow. In practice, autoregressive flow-based models are slow to invert, making either density estimation or sample generation slow. Flows based on coupling transforms are fast for both tasks, but have previously performed less well at density estimation than autoregressive flows. We stack a new coupling transform, based on monotonic cubic splines, with LU-decomposed linear layers. The resulting *cubic-spline flow* retains an exact one-pass inverse, can be used to generate high-quality images, and closes the gap with autoregressive flows on a suite of density-estimation tasks.

1. Introduction

Normalizing flows are flexible probabilistic models of continuous data. A *normalizing flow* models data $\mathbf{x}$ as the output of an invertible smooth transformation $\mathbf{f}$ of noise $\mathbf{u}$:

$$\mathbf{x} = \mathbf{f}(\mathbf{u}) \quad \text{where} \quad \mathbf{u} \sim \pi(\mathbf{u}). \tag{1}$$

Typically, the noise distribution $\pi(\mathbf{u})$ is taken to be simple (a standard normal is a common choice), whereas the transformation $\mathbf{f}$ and its inverse $\mathbf{f}^{-1}$ are implemented by an invertible neural network. The probability density of $\mathbf{x}$ under the flow is obtained by a change of variables:

$$p(\mathbf{x}) = \pi\big(\mathbf{f}^{-1}(\mathbf{x})\big)\left|\det\left(\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{x}}\right)\right|. \tag{2}$$

Given training data $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, the flow is trained with backpropagation, by maximizing the total log likelihood $\sum_{n=1}^{N} \log p(\mathbf{x}_n)$ with respect to the parameters of the transformation $\mathbf{f}$. Normalizing flows have been successfully used for density estimation [5, 23], variational inference [17, 27, 29], image, audio and video generation [14, 16, 18, 26], and likelihood-free inference [24].

In practice, the Jacobian determinant of $\mathbf{f}$ must be tractable, so that the density $p(\mathbf{x})$ can be computed efficiently. Several invertible transformations with a tractable Jacobian determinant have been proposed, including coupling transforms [4, 5], autoregressive transforms [3, 13, 17, 23], invertible convolutions [11, 16], transformations based on the matrix-determinant lemma [27, 29], and continuous transformations based on differential equations [2, 8].

Even though the above transformations are invertible in theory, they are not always efficiently invertible in practice. For instance, the affine autoregressive transforms used by MAF [23] and IAF [17] are D times slower to invert than to evaluate, where D is the dimensionality of $\mathbf{x}$. Even worse, inverting the non-affine transformations used by NAF [13] and block-NAF [3] would require numerical optimization. In practice, the forward transformation $\mathbf{f}$ is needed for generating data, whereas the inverse transformation $\mathbf{f}^{-1}$ is needed for computing $p(\mathbf{x})$; for a flow to be applicable in both situations, both directions need to be fast.

Transformations that are equally fast to invert and evaluate do exist. Affine coupling transforms, used by Real NVP [5] and Glow [16], only require a single neural-network pass in either direction; however, more flexible autoregressive transforms have obtained better density-estimation results. More flexible coupling transforms have been proposed, based on piecewise-quadratic splines [20], although these have not previously been evaluated as density estimators. Continuous flows such as Neural ODEs [2] and FFJORD [8] are also equally fast in both directions, but they require numerically integrating a differential equation in each direction, which can be slower than a single neural-network pass.

In this paper we introduce *cubic-spline flows*, which, like Real NVP and Glow, only require a single neural-network pass in either the forward or the inverse direction, but in practice are as flexible as state-of-the-art autoregressive flows. Cubic-spline flows alternate between non-affine coupling transforms that use flexible monotonic cubic splines to transform their input, and LU-decomposed linear layers that combine their input across dimensions.

[*]Equal contribution [1]School of Informatics, University of Edinburgh, United Kingdom. Correspondence to: Conor Durkan <conor.durkan@ed.ac.uk>, Artur Bekasov <artur.bekasov@ed.ac.uk>.

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

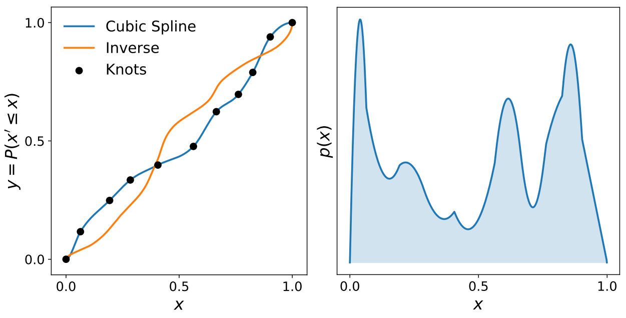

Figure 1: **Left**: Illustration of a random monotonic cubic spline with $K = 10$ bins which maps $[0, 1]$ to $[0, 1]$. Monotonic splines act as drop-in replacements for affine or additive transformations in coupling layers. **Right**: Since the cubic spline can be interpreted as a cumulative distribution function, its derivative defines a probability density function which is piecewise quadratic.

Experimentally, we show that cubic-spline flows match or exceed state-of-the-art performance in density estimation, and that they are capable of generating high-quality images with only a single neural-network pass.

2. Method

2.1. Coupling transforms

A *coupling transform* ϕ maps an input $\mathbf{x}$ to an output $\mathbf{y}$ in the following way:

1. Split the input $\mathbf{x}$ into two parts, $\mathbf{x} = [\mathbf{x}_{1:d-1}, \mathbf{x}_{d:D}]$.
2. Compute parameters $\boldsymbol{\theta} = \mathrm{NN}(\mathbf{x}_{1:d-1})$, where NN is an arbitrary neural network.
3. Compute $y_i = g_{\boldsymbol{\theta}_i}(x_i)$ for $i = d : D$, where $g_{\boldsymbol{\theta}_i}$ is an invertible function parameterized by $\boldsymbol{\theta}_i$.
4. Set $\mathbf{y}_{1:d-1} = \mathbf{x}_{1:d-1}$, and return $\mathbf{y} = [\mathbf{y}_{1:d-1}, \mathbf{y}_{d:D}]$.

The Jacobian matrix of a coupling transform is lower triangular, since $\mathbf{y}_{d:D}$ is given by transforming $\mathbf{x}_{d:D}$ elementwise as a function of $\mathbf{x}_{1:d-1}$, and $\mathbf{y}_{1:d-1}$ is equal to $\mathbf{x}_{1:d-1}$. Thus, the Jacobian determinant of the coupling transform ϕ is

$$\det\left(\frac{\partial \phi}{\partial \mathbf{x}}\right) = \prod_{i=d}^{D} \frac{\partial g_{\boldsymbol{\theta}_i}}{\partial x_i}. \tag{3}$$

Coupling transforms solve two important problems for normalizing flows: first, they have a tractable Jacobian determinant; second, they can be inverted exactly in a single pass. The inverse of a coupling transform can be easily computed by running steps 1–4 above, this time inputting $\mathbf{y}$, and using $g^{-1}_{\boldsymbol{\theta}_i}$ to compute $\mathbf{x}_{d:D}$ in step 3.

2.2. Monotonic-spline coupling transforms

Typically, the function $g_{\boldsymbol{\theta}_i}$ takes the form of an *affine* or *additive* transformation for computational ease. The affine transformation is given by:

$$g_{\boldsymbol{\theta}_i}(x_i) = \alpha_i x_i + \beta_i \quad \text{where} \quad \boldsymbol{\theta}_i = \{\alpha_i, \beta_i\}. \tag{4}$$

The additive transformation corresponds to the special case $\alpha_i = 1$. Both the affine and the additive transformation are easy to invert, but they lack flexibility.

Nonetheless, we are free to choose any function $g_{\boldsymbol{\theta}_i}$ as long as it is differentiable and we can easily invert it. Recently, Müller et al. [20] proposed a powerful generalization of the above forms, based on monotonic piecewise polynomials. The idea is to restrict the input domain of $g_{\boldsymbol{\theta}_i}$ between 0 and 1, partition the input domain into K bins, and define $g_{\boldsymbol{\theta}_i}$ to be a simple polynomial segment within each bin (see fig. 1). Müller et al. [20] restrict themselves to monotonically-increasing linear and quadratic polynomial segments, whose coefficients are parameterized by $\boldsymbol{\theta}_i$. Moreover, the polynomial segments are restricted to match at the boundaries so that $g_{\boldsymbol{\theta}_i}$ is continuous. Functions of this form are known as *polynomial splines*.

2.3. Monotonic cubic-spline coupling transforms

Inspired by this direction of research, we introduce *cubic-spline flows*, a natural extension to the framework of Müller et al. [20]. We propose to implement $g_{\boldsymbol{\theta}_i}$ as a *monotonic cubic spline* [7], where each segment is defined by a monotonically-increasing cubic polynomial (fig. 1). Cubic splines are well-known across science, finding use in many applications, and their theory is well-explored [6, 12].

We employ *Steffen's method* [28] to parameterize a monotonically-increasing cubic spline that maps $[0, 1]$ to $[0, 1]$. Given a fixed number of bins K, Steffen's method takes $K + 1$ coordinates $\{(x_k, y_k)\}_{k=0}^{K}$, known as *knots*, and two values for the derivatives at inputs 0 and 1. Then, the method constructs a continuously-differentiable cubic spline that passes through the knots, has the given boundary derivatives, and is monotonic on each segment (for details, see appendix A.1). To ensure the spline is monotonically-increasing and maps $[0, 1]$ to $[0, 1]$, we require:

$$(x_0, y_0) = (0, 0), \quad (x_K, y_K) = (1, 1), \tag{5}$$
$$x_k < x_{k+1} \text{ and } y_k < y_{k+1} \quad \text{for } k = 0 : K - 1. \tag{6}$$

Our implementation of the *cubic-spline coupling transform* is as follows:

1. A neural network NN takes $\mathbf{x}_{1:d-1}$ and outputs an unconstrained parameter vector $\boldsymbol{\theta}_i$ of length $2K + 2$ for each $i = d : D$.
2. Vector $\boldsymbol{\theta}_i$ is partitioned as $\boldsymbol{\theta}_i = \left[\boldsymbol{\theta}_i^w, \boldsymbol{\theta}_i^h, \boldsymbol{\theta}_i^d\right]$, where $\boldsymbol{\theta}_i^w$ and $\boldsymbol{\theta}_i^h$ have length K, and $\boldsymbol{\theta}_i^d$ has length 2.
3. Vectors $\boldsymbol{\theta}_i^w$ and $\boldsymbol{\theta}_i^h$ are each passed through a softmax; the outputs are interpreted as the widths and heights of the K bins, which must be positive and sum to 1.

4. A cumulative sum of the K bin widths and heights yields the $K+1$ knots $\{(x_k, y_k)\}_{k=0}^{K}$. Vector $\boldsymbol{\theta}_i^d$ is interpreted as the two boundary derivatives.

Evaluating a cubic spline at location x requires finding the bin in which x lies, which can be done efficiently with binary search, since the bins are sorted. The Jacobian determinant of the cubic-spline coupling transform can be computed in closed form as a product of quadratic-polynomial derivatives (see appendix A.2). For the inverse, we need to compute the roots of a cubic polynomial whose constant term depends on the desired value to invert, which can be done analytically. However, in early experiments, we found that naïve root finding using trigonometric and hyperbolic methods was unstable for certain values of the coefficients. To address this issue, we follow a modified version, provided by Peters [25], of the scheme outlined by Blinn [1], which stabilizes many of the potentially difficult cases (see appendix A.3).

Unlike the affine and additive transformations which have limited flexibility, a monotonic spline can approximate any continuous monotonic function arbitrarily well, given sufficiently many bins. Yet, cubic-spline coupling transforms have a closed-form Jacobian determinant and can be inverted in one pass. Finally, our parameterization of cubic splines using Steffen's method is fully differentiable, so the transform can be trained by gradient methods.

In addition to transforming $\mathbf{x}_{d:D}$ by monotonic cubic splines whose parameters are a function of $\mathbf{x}_{1:d-1}$, we introduce a set of monotonic cubic splines which act elementwise on $\mathbf{x}_{1:d-1}$, whose parameters are optimized directly with stochastic gradient descent. This means that our coupling layer transforms all input variables at once, rather than being restricted to a subset, as follows:

$$\boldsymbol{\theta}_{1:d-1} = \text{Trainable parameters} \tag{7}$$

$$\boldsymbol{\theta}_{d:D} = \text{NN}(\mathbf{x}_{1:d-1}) \tag{8}$$

$$y_i = g_{\boldsymbol{\theta}_i}(x_i) \quad \text{for each } i = 1:D. \tag{9}$$

2.4. Linear transforms with an LU decomposition

To ensure all input variables can interact with each other, it is common to permute the elements of intermediate layers in a normalizing flow. Permutation is an invertible linear transformation, with absolute determinant equal to 1. Kingma & Dhariwal [16] generalize this approach using an efficient parameterization of a linear transformation, demonstrating improvements on a range of image tasks. In particular, they parameterize the *lower-upper* or *LU decomposition* of a square matrix as $\mathbf{W} = \mathbf{PLU}$, where $\mathbf{P}$ is a fixed permutation matrix, $\mathbf{L}$ is lower-triangular with ones on the diagonal, and $\mathbf{U}$ is upper triangular. Written in this form, the determinant of $\mathbf{W}$ can be calculated in $\mathcal{O}(D)$ time as the product of the diagonal elements of $\mathbf{U}$. Also, by parameterizing the diagonal elements of $\mathbf{U}$ to be positive, $\mathbf{W}$ is guaranteed to be invertible. Given that we have the LU factorization of $\mathbf{W}$, we can efficiently apply the inverse transformation, and sample from the model in one pass.

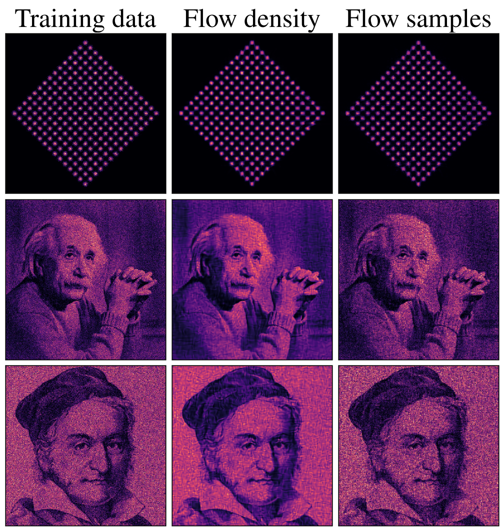

Figure 2: Qualitative results for two-dimensional synthetic datasets using cubic-spline flows with two coupling layers. Some previous flows struggle to model such fine details, as demonstrated by, e.g., Nash & Durkan [21].

2.5. Cubic-spline flows

We propose a general-purpose flow based on alternating LU-decomposed linear layers and monotonic cubic-spline coupling transforms. Since the splines map $[0,1]$ to $[0,1]$, we apply a sigmoid, and its inverse, the logit, before and after each coupling layer, so the overall transform is compatible with unconstrained data and linear layers. We clip the inputs of the logit to $[10^{-6}, 1-10^{-6}]$ to prevent saturation with 32-bit floating-point precision. The function is no longer strictly reversible for all inputs due to this clipping.

Like Real NVP or Glow, cubic-spline flows can represent either a transformation from data to noise, or from noise to data. In both cases, the transformation requires only a single pass of the neural network defining the flow. Overall, our proposed cubic-spline flow resembles a traditional feed-forward neural network architecture, alternating between linear transformations and piecewise nonlinearities, while retaining exact invertibility.

3. Experiments

For our experiments, NN in eq. (8) is a fully-connected or convolutional neural network with two pre-activation residual blocks [9, 10]. We use the Adam optimizer [15], and a cosine schedule for annealing the learning rate [19]. All experiments use a standard normal for the noise distribution $\pi(\mathbf{u})$, except for the two-dimensional experiments, which use a uniform distribution.

Table 1: Test log likelihood (in nats) for UCI datasets and BSDS300; higher is better. Error bars correspond to two standard deviations (FFJORD do not report error bars). Apart from quadratic and cubic splines, all results are taken from existing literature. NAF† report error bars across five repeated runs rather than across the test set.

	Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
One-pass flows	FFJORD [8]	0.46	8.59	-14.92	-10.43	157.40
	Quadratic-spline	0.65 ± 0.01	13.13 ± 0.02	-14.95 ± 0.02	-9.18 ± 0.43	157.49 ± 0.28
	Cubic-spline	0.65 ± 0.01	13.14 ± 0.02	-14.59 ± 0.02	-9.06 ± 0.44	157.24 ± 0.28
Auto-regressive flows	MAF [23]	0.30 ± 0.01	10.08 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
	NAF† [13]	0.62 ± 0.01	11.96 ± 0.33	-15.09 ± 0.40	-8.86 ± 0.15	157.73 ± 0.04
	Block-NAF [3]	0.61 ± 0.01	12.06 ± 0.09	-14.71 ± 0.38	-8.95 ± 0.07	157.36 ± 0.03
	TAN various [22]	0.60 ± 0.01	12.06 ± 0.02	-13.78 ± 0.02	-11.01 ± 0.48	159.80 ± 0.07

3.1. Synthetic datasets

We first demonstrate the flexibility of cubic-spline flows on synthetic two-dimensional datasets (fig. 2). The mixture of 225 Gaussians and Einstein are taken from [21], and we generate another dataset using an image of Gauss. For each task, we train on one-million data points, use a cubic-spline flow with two coupling layers, no linear layers, and $K = 128$ bins. As we can see, the model is flexible enough to fit complex, multimodal densities that other flows would struggle with. In each case, the model consists of fewer than 30% of the parameters than the 512×512 histograms which are used to display the data.

3.2. Density estimation on tabular data

We next consider the UCI and BSDS300 datasets, a standard suite of benchmarks for density estimation of tabular data. Our spline flows alternate coupling layers with LU-decomposed linear layers, and we use $K = 10$ bins in all experiments. For all datasets, we stack ten of these composite linear and coupling layers, apart from MINIBOONE, where we use five. Hyperparameters such as the number of training steps and the hidden dimension used by NN in coupling layers are tuned separately for each cubic model. As an ablation study, we compare with a flow using quadratic instead of cubic splines, which extends [20] with LU layers and elementwise transformations as in section 2.

Table 1 shows our results. The cubic and quadratic models are close in performance, with the error bars on test log likelihood overlapping. However, a paired comparison computing the mean and standard error of the difference between models on individual examples shows that the cubic-spline flow is statistically significantly better on three of the five datasets, while being indistinguishable on POWER and slightly worse than the quadratic-spline flow on BSDS300. On the other hand, the quadratic-spline flow is slightly faster and less prone to numerical issues. Appendix A.4 further compares the cubic and quadratic models.

Spline flows achieve state-of-the-art performance on all tasks against FFJORD [8], the previously strongest flow-based model with a one-pass inverse. Moreover, the spline flows are competitive against the best-performing autoregressive flows, achieving state of the art for any flow model on POWER and GAS. These results demonstrate that it is possible for neural density estimators based on coupling layers to challenge autoregressive models, and that it may not be necessary to sacrifice one-pass sampling for density-estimation performance.

3.3. Image generation

Finally, we demonstrate that cubic-spline flows scale to high-dimensional data using the Fashion-MNIST dataset [30]. We use a Glow-like model [16] with cubic-spline coupling transforms, which includes a multi-scale architecture, actnorm layers with data-dependent initialization, and invertible 1×1 convolutions. In line with Kingma & Dhariwal [16], we illustrate the effect of scaling the standard deviation of the noise distribution $\pi(\mathbf{u})$ by a temperature $T \in [0, 1]$. Qualitative results are shown in fig. 3. We leave the comparison with state-of-the-art flow-based image models on higher-dimensional image datasets for future work.

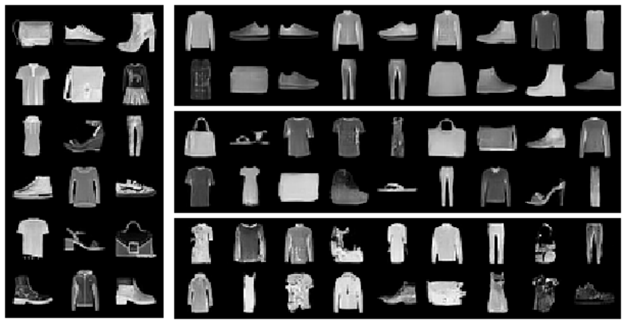

Figure 3: Qualitative results for Fashion-MNIST dataset. **Left**: Training data. **Right**: Unconditional samples from the cubic-spline flow. Samples for three temperatures are shown in separate blocks, top-to-bottom: 0.5, 0.75, 1.0.

4. Conclusion

The cubic-spline flow is the first model that combines the density-estimation performance of state-of-the-art autoregressive models like NAF [13] and TAN [22], while still being able to generate realistic samples in a single pass like Real NVP [5] and Glow [16]. More generally, we have shown that flexible monotonic splines are a useful differentiable module, which could be included in many models that can be trained end-to-end. We intend to implement autoregressive flows with these transformations, which may further improve the state of the art in density estimation.

Acknowledgements

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh. George Papamakarios was also supported by Microsoft Research through its PhD Scholarship Programme.

A. More details on monotonic cubic splines

A.1. Parameterization of the spline

We here include an outline of the method of Steffen [28] which closely matches the original paper. Let $\{(x_k, y_k)\}_{k=0}^{K}$ be a given set of knot points in the plane which satisfy

$$(x_0, y_0) = (0, 0), \quad (x_K, y_K) = (1, 1), \tag{10}$$

$$x_k < x_{k+1} \text{ and } y_k < y_{k+1} \text{ for } k = 0 : K-1. \tag{11}$$

If additionally the derivatives $\{d_k\}_{k=0}^{K}$ at the knot points are known, a unique cubic polynomial

$$f_k(\xi) = \alpha_{k0} + \alpha_{k1}\xi + \alpha_{k2}\xi^2 + \alpha_{k3}\xi^3, \tag{12}$$

where $\xi = x - x_k$, is determined on each bin. The coefficients are given by

$$\alpha_{k0} = y_k \tag{13}$$

$$\alpha_{k1} = d_k \tag{14}$$

$$\alpha_{k2} = \frac{3s_k - 2d_k - d_{k+1}}{w_k} \tag{15}$$

$$\alpha_{k3} = \frac{d_k + d_{k+1} - 2s_k}{(w_k)^2}, \tag{16}$$

where $w_k = x_{k+1} - x_k$ is the width of each bin, and $s_k = (y_{k+1} - y_k)/w_k$ is the slope of the line joining consecutive knots. The resulting cubic spline is continuously differentiable on the interval $[0, 1]$, and it remains only to determine the derivative values $\{d_k\}_{k=0}^{K}$ so that the overall spline is monotonic on this interval.

To this end, Steffen [28] uses the unique quadratic function passing through the points (x_{k-1}, y_{k-1}), (x_k, y_k) and (x_{k+1}, y_{k+1}) to determine the derivative at location x_k. Monotonicity of this quadratic on the interval defined by these points is sufficient to specify an appropriate derivative, which is given by

$$p_k = \frac{s_{k-1}w_k + s_k w_{k-1}}{w_{k-1} + w_k}. \tag{17}$$

However, it is possible that the quadratic in question is not monotonic on the given interval, and in this case the derivative value needs to be modified. Steffen [28] proposes using the smallest value for the derivative at x_k such that the quadratic is monotonic on the given interval, showing that

$$d_k = \begin{cases} 2\min(s_{k-1}, s_k) & \text{if } p_k > 2\min(s_{k-1}, s_k) \\ p_k & \text{otherwise} \end{cases} \tag{18}$$

is sufficient.

A.2. Computing the derivative

The derivative of eq. (12) is straightforwardly given by

$$\frac{\mathrm{d}f_k(\xi)}{\mathrm{d}x} = \alpha_{k1} + 2\alpha_{k2}\xi + 3\alpha_{k3}\xi^2. \tag{19}$$

The logarithm of the absolute value of the determinant of a cubic-spline coupling transform is thus given by a sum of the logarithm of eq. (19) for each transformed x.

A.3. Computing the inverse

Computing the inverse of a cubic polynomial requires solving for the roots of a cubic polynomial whose constant term depends on the value to invert. That is, for a given y, we wish to find x which satisfies

$$(\alpha_{k0} - y) + \alpha_{k1}\xi + \alpha_{k2}\xi^2 + \alpha_{k3}\xi^3 = 0, \tag{20}$$

where ξ is defined as before. Monotonicity means there is at most one root x in each bin, and it is possible to determine this value analytically, using either Cardano's formula, or trigonometric or hyperbolic methods. In practice, careful numerical consideration is necessary to ensure stable root-finding. We found the comprehensive treatment of Blinn [1] particularly useful, ultimately settling on a slightly modified version of a procedure outlined in this latter work [25]. We refer readers to Blinn [1] for full details.

A.4. Comparison with monotonic quadratic splines

Following Müller et al. [20], we construct a quadratic spline by integrating an unconstrained linear spline. The result is a monotonic piecewise-quadratic function, where the values and derivatives of the function are continuous at the knots. The derivatives of the quadratic spline at the knots, corresponding to the density, are set directly by the linear spline. However, some monotonically-increasing knot locations (x_k, y_k) cannot be expressed through this construction.

In contrast, Steffen's cubic-spline construction [28] can interpolate arbitrary monotonic settings of the knot locations (x_k, y_k), so unlike the quadratic spline can set arbitrary quantiles of the implied distribution at the knot locations. However, the method sets the derivatives of the function based on prior smoothness assumptions, rather than allowing the user to set them directly as in the quadratic spline. As a result, the number of parameters for the two spline constructions are similar. The cubic spline could be made strictly more flexible by adding parameters for the derivatives at the knots, constrained to maintain monotonicity.

As they are, the cubic and quadratic splines give different functions for knots specifying the same quantiles, corresponding to different inductive biases. Steffen [28] discusses several nice properties of the interpolants chosen by his cubic-spline method.

References

[1] Blinn, J. F. How to solve a cubic equation, part 5: Back to numerics. *IEEE Computer Graphics and Applications*, 27(3):78–89, 2007.

[2] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

[3] De Cao, N., Titov, I., and Aziz, W. Block neural autoregressive flow. *Conference on Uncertainty in Artificial Intelligence*, 2019.

[4] Dinh, L., Krueger, D., and Bengio, Y. NICE: Nonlinear independent components estimation. *International Conference on Learning Representations, Workshop track*, 2015.

[5] Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations*, 2017.

[6] Durrleman, S. and Simon, R. Flexible regression models with cubic splines. *Statistics in medicine*, 8(5): 551–561, 1989.

[7] Fritsch, F. N. and Carlson, R. E. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[8] Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. K. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2018.

[9] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. *European Conference on Computer Vision*, 2016.

[11] Hoogeboom, E., van den Berg, R., and Welling, M. Emerging convolutions for generative normalizing flows. *arXiv:1901.11137*, 2019.

[12] Hou, H. and Andrews, H. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on acoustics, speech, and signal processing*, 26(6): 508–517, 1978.

[13] Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *International Conference on Machine Learning*, 2018.

[14] Kim, S., Lee, S.-g., Song, J., and Yoon, S. FloWaveNet: A generative flow for raw audio. *arXiv:1811.02155*, 2018.

[15] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[16] Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1×1 convolutions. *Advances in Neural Information Processing Systems*, 2018.

[17] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 2016.

[18] Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. VideoFlow: A flow-based generative model for video. *arXiv:1903.01434*, 2019.

[19] Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016.

[20] Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural importance sampling. *arXiv:1808.03856*, 2018.

[21] Nash, C. and Durkan, C. Autoregressive energy machines. *International Conference on Machine Learning*, 2019.

[22] Oliva, J., Dubey, A., Zaheer, M., Poczos, B., Salakhutdinov, R., Xing, E., and Schneider, J. Transformation autoregressive networks. *International Conference on Machine Learning*, 2018.

[23] Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 2017.

[24] Papamakarios, G., Sterratt, D. C., and Murray, I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *International Conference on Artificial Intelligence and Statistics*, 2019.

[25] Peters, C. How to solve a cubic equation, revisited. `http://momentsingraphics.de/?p=105`, 2016. Accessed: 2019-04-17.

[26] Prenger, R., Valle, R., and Catanzaro, B. WaveGlow: A flow-based generative network for speech synthesis. *arXiv:1811.00002*, 2018.

[27] Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *International Conference on Machine Learning*, 2015.

[28] Steffen, M. A simple method for monotonic interpolation in one dimension. *Astronomy and Astrophysics*, 239:443, 1990.

[29] van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *Conference on Uncertainty in Artificial Intelligence*, 2018.

[30] Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.