# Robust normalizing flows using Bernstein-type polynomials

Sameera Ramasinghe, Kasun Fernando, Salman Khan, Nick Barnes

## Motivation

- *Generative modeling* is modeling probability distributions of data set
  Examples: images, audio signals, observations from physical experiments.
- It is an important aspect in Data Science & Machine Learning.
- Why? Because it can be used
  - To generate synthetic samples.
  - To estimate the likelihood of a sample.
- Three key methods used:
  - GANs
  - VAEs
  - Normalizing Flows
- GANs and VAEs have the following limitations:
  - Exact point-wise density estimation is not possible.
  - Mode and posterior collapse.
  - High sensitivity to the NN architecture.
- Normalizing Flows (NFs) were introduced by Rezende and Mohamed (2016) as a way to overcome these issues.

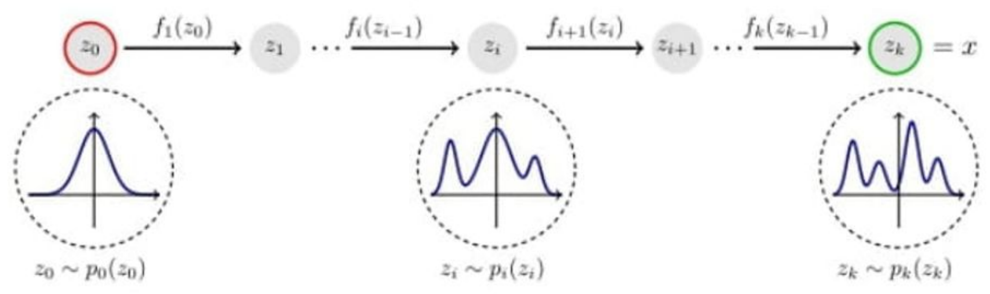


Figure: A 1-D normalizing flow; source: flowtorch.ai

- A normalizing flow is a series of invertible mappings that transform a simple distribution (known prior) to a complex distribution (unknown posterior).
- It is known that between any two probability distributions on $\mathbb{R}^n$ there is a unique (up to null sets) increasing map whose Jacobian is an upper triangular matrix.
- When implementing as an NN, we have to make sure that
  - coupling functions are dense in the class of increasing triangular maps (universality).
  - the NF does not amplify initial errors (robustness).

- Like any other nonlinear model, NFs are susceptible to numerical instabilities.
- In fact, this can be seen from one of our experiments:
  - We train known NF models from scratch on five widely used (noise-free) datasets.
  - Then, we test the models on the noise-free test set to obtain the standard deviation $\sigma$ and mean $\mu$ of the test log-likelihood.
  - Finally, we add i.i.d. noise, sampled from a Uniform$[0, 10^{-2}]$, to those datasets, retrain the models, and obtain the test log-likelihood $y$ on the noise-free test set.
  - The change in the test log-likelihood as a fraction of the standard deviation $\frac{y-\mu}{\sigma}$ is given in the following table.

Table: Test log-likelihood drop for random initial errors relative to $\sigma$.

| Model | Power | Gas | Hepmass | MiniBoone | BSDS300 |
|---|---|---|---|---|---|
| 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 |

- These NFs are *not* robust!!!
- Small initial errors consistently created changes larger than $1.645\,\sigma$.
  Errors in the 5% tails of the distribution of errors (unacceptably large).

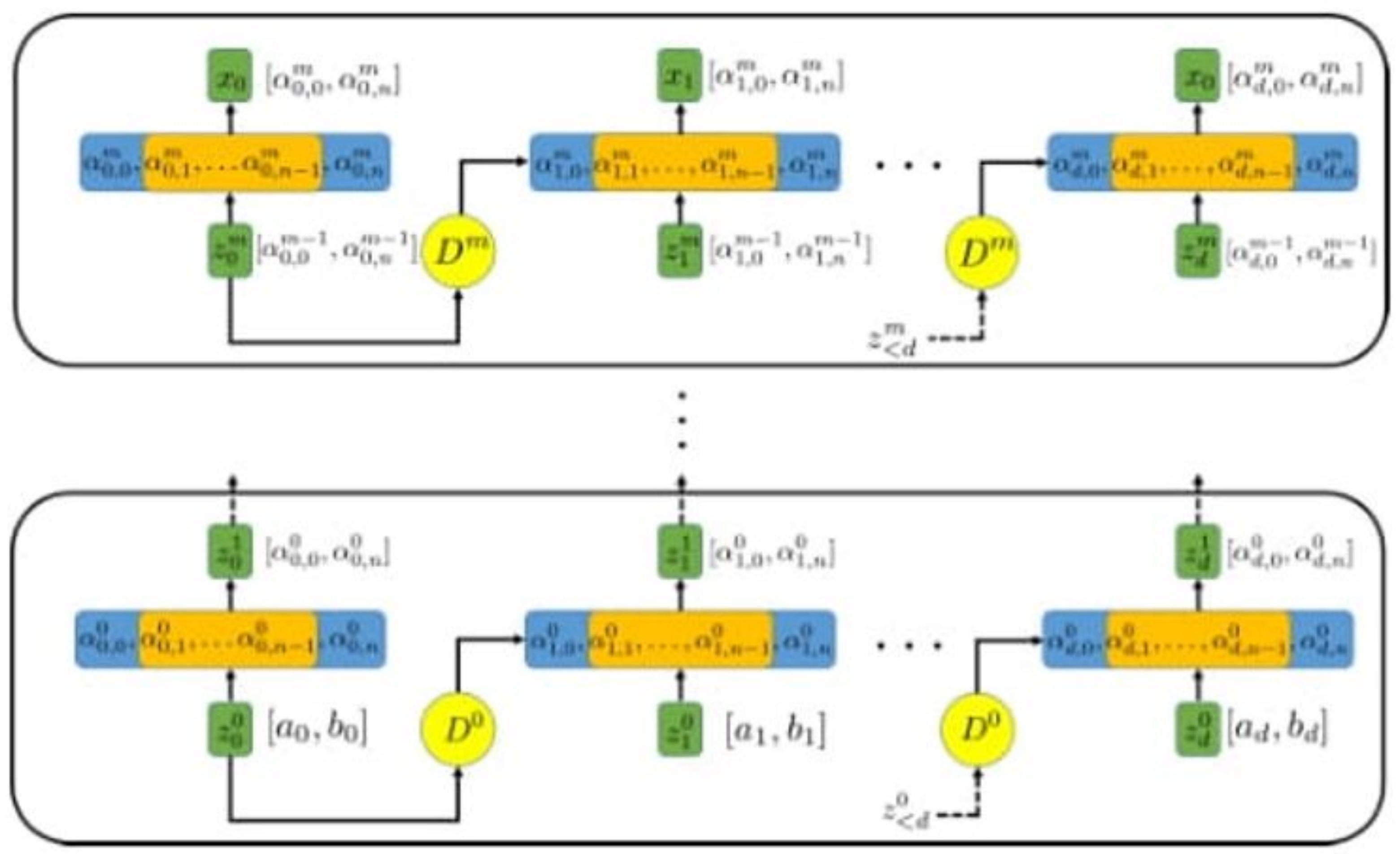
## Method

- We consider a dense class of polynomials called Bernstein polynomials:

$$B_n(z) = \sum_{k=0}^{n} \alpha_k \binom{n}{k} z^k (1-z)^{n-k}, \quad z \in [0,1].$$

- We use $B_n$s as the coupling functions in our normalizing flow - the Bernstein NF.
- It is known that among "positive" polynomial bases the Bernstein basis, i.e., $\binom{n}{k} z^k (1-z)^{n-k}, k = 0, \ldots, n$ is *optimally stable.*
- So, the change in the value of a polynomial caused by the perturbations of coefficients is always smaller in Bernstein basis than in bases such as the power basis.
- So, when polynomials are used to construct NFs:
  - Q-NSF based on quadratic or cubic splines
  - SOS based on some of square polynomials
  - Berntein NF based on Bernstein-type polynomials

  ours yields the most numerically stable NF!!!
- Our experiments, while confirming this, demonstrate that our NF definitively outperforms even the NFs that are *not* based on polynomials.



- This is a MADE style network for $d$-dimensional sources $P_z(\mathbf{z})$ and targets $P_x(\mathbf{x})$.
- The element-wise mapping between the components $x_j$ and $z_j$ is approximated using a Bernstein-type polynomial as $x_j = B_n^j(z_j)$.
- We obtain the parameters of $B_n^j(z_j)$ using an NN which is conditioned on $z_{<j}$.
- Fixed coefficients are in blue (fixing the range of each transformation) and trainable coefficients are in orange boxes.
- 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.

## Results

Table: Test log-likelihood drop for random initial errors, relative to $\sigma$.

| 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** |

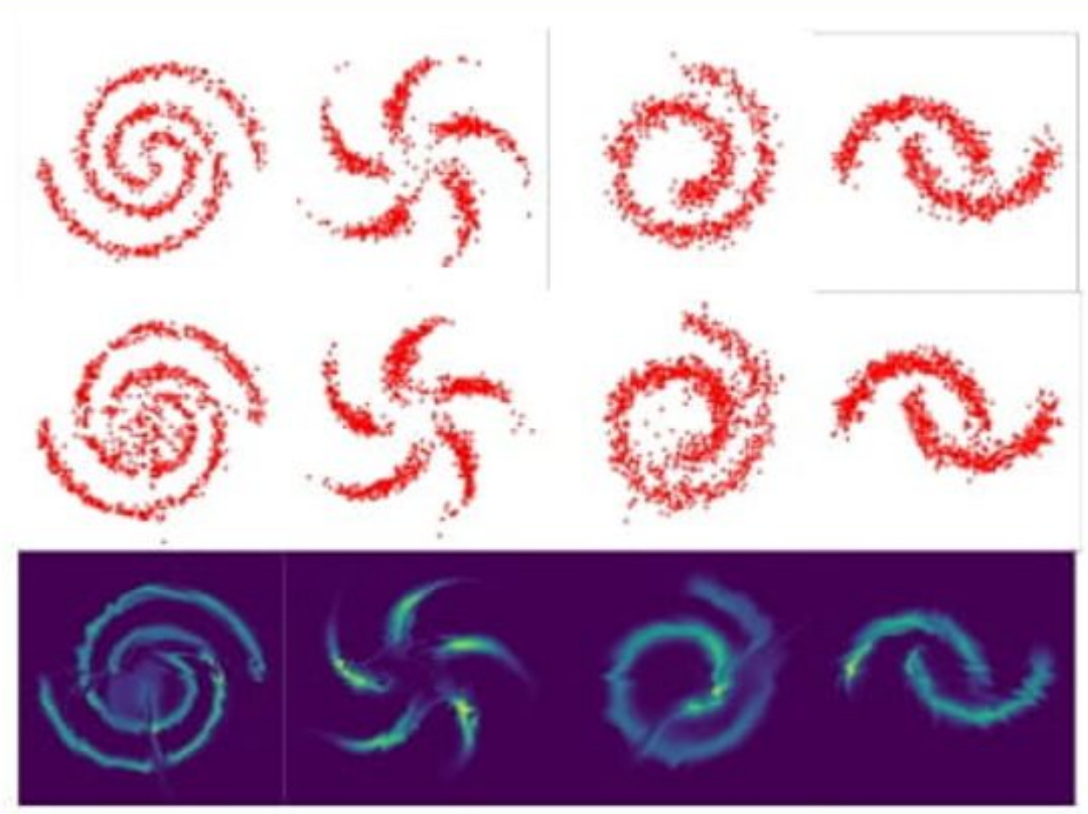


Figure: Qualitative results for modeling the toy distributions. *From the top row:* ground truth, prediction, and predicted density.

Figure: Samples generated by the Bernstein NF on MNIST and CIFAR10.