
Jožef Stefan Institute

# Generating Highdimensional Data using Flow Models

***Author :***
Fedja Močnik

***Mentor :***
prof. dr. Jernej F. Kamenik

September 26, 2024

# Contents

# 1 Abstract

Flow models are powerful generative techniques that transform simple distributions into complex ones via invertible mappings, enabling efficient sampling and density estimation. In this project, I explored the application of flow models to generate high-dimensional data, focusing on the Real-valued Non-Volume Preserving (RealNVP) model. Starting with simpler 2D and 4D datasets, I scaled the model to handle a 12-dimensional dataset from the Pierre Auger Observatory, which captures cosmic ray events. Despite challenges in implementing Bayesian uncertainty quantification techniques in higher dimensions, the RealNVP model achieved moderate success, with a 40% overlap between generated and original data. Future work will address the integration of advanced Bayesian methods to improve performance.

# 2 Introduction

Flow models have emerged as a powerful class of generative models, capable of learning complex data distributions through invertible transformations. At their core, flow models utilize a sequence of bijective mappings, allowing for efficient sampling and density estimation. By transforming simple distributions into more intricate ones, flow models can effectively model high-dimensional data.

Initially, I explored the planar flow model, which, while conceptually straightforward, proved inadequate for capturing the complexities of high-dimensional distributions. Its limitations prompted me to adopt a more sophisticated approach: the Real-valued Non-Volume Preserving (RealNVP) model. RealNVP operates by applying a series of coupling layers, which split the input into two parts. It then transforms one part conditionally based on the other, ensuring that the overall transformation remains invertible. This method allows RealNVP to model intricate distributions with greater accuracy, making it a suitable choice for generating high-dimensional data.

In this project, I aimed to generate 12-dimensional data from Auger Pierre observatory, where they use water Cherenkov detectors to capture and analyze cosmic rays, providing valuable insights into high-energy astrophysical phenomena.

The structure of this article is as follows: Section 2 outlines the methods and implementation details of my project, including the procedures undertaken and code availability on Github. Section 3 presents the results obtained, followed by a discussion on potential improvements. Finally, Section 4 summarizes the findings and concludes the report.

# 3 Methodology and Implementation

## 3.1 Overview of Approach

This project applied RealNVP flow-based models to increasingly complex datasets, starting with 2D examples like "Moons" and "Circles". After this initial exploration, I extended the model to 4D datasets, such as the IRIS dataset, while also incorporating

Bayesian Variational Inference (VI) for uncertainty quantification. Finally, the model was scaled to handle a 12-dimensional dataset from the Pierre Auger Observatory. Challenges arose with implementing Stochastic Variational Inference (SVI) and Bayesian weights in the 12D setting, which remain unresolved.

## 3.2 Implementation

### 3.2.1 RealNVP Model Architecture

The RealNVP model is designed to learn transformations between simple and complex distributions using a series of invertible coupling layers. In each coupling layer, half of the input remains unchanged while the other half is transformed, conditioned on the static part. The transformations are governed by scale ($s$) and translation ($t$) networks.

**The architecture of the model included:**

- **Scale Network (s)**: A fully connected neural network that predicts scaling factors for part of the input data.
- **Translation Network (t)**: A neural network responsible for predicting the translation offsets applied to the input.
- **Coupling Mechanism**: The input data is split into two parts, with one part transformed by the $s$ and $t$ networks, while the other part remains static. The input is flipped between blocks to ensure all dimensions are transformed.

Key architectural details:

- **Number of blocks**: 8
- **Hidden layer size**: 256
- **Activation function**: ReLU
- **Prior distribution**: Multivariate Gaussian

### 3.2.2 RealNVP Transformations

The coupling layers in RealNVP perform a transformation on the input data $x = (x_1, x_2)$ where $x_1$ remains static, and $x_2$ is transformed. Given the input partitioned into $x_1$ and $x_2$, the transformations are as follows:

$$\begin{aligned} y_1 &= x_1, \\ y_2 &= x_2 \odot \exp(s(x_1)) + t(x_1), \end{aligned}$$

where $s(x_1)$ and $t(x_1)$ are the scale and translation networks respectively, and $\odot$ represents the element-wise multiplication.

To ensure the transformation is invertible, the inverse is given by:

$$\begin{aligned} x_1 &= y_1, \\ x_2 &= (y_2 - t(y_1)) \odot \exp(-s(y_1)). \end{aligned}$$

The Jacobian of this transformation, crucial for calculating the likelihood during training, is easily computable and has a triangular structure. The log-determinant of the Jacobian is:

$$\log \left| \det \frac{\partial y}{\partial x} \right| = \sum_{i=1}^{d_2} s(x_1)_i,$$

where $d_2$ is the dimensionality of $x_2$. This allows efficient computation during training.

## 3.3 4-dimensional and Bayesian Extension for 2-dimensional

In the 2-dimensional experiments, which included the Moons dataset, I introduced Bayesian elements through Variational Inference (VI) to add uncertainty quantification to the model. The goal was to evaluate how uncertainty estimation could improve model robustness, especially in tasks like outlier detection. While I encountered challenges incorporating full Bayesian techniques in higher-dimensional settings, the 2-dimensional model with Bayesian layers achieved strong performance, particularly on the Moons dataset, where I achieved 90% accuracy.
For the 4-dimensional dataset, I utilized the IRIS dataset, achieving a notable classification accuracy of 91%. The model was based on a RealNVP architecture, which closely parallels the structure I subsequently employed for the 12-dimensional data.

## 3.4 Scaling to 12 dimensions

The final phase of the project involved scaling the RealNVP model to a 12-dimensional dataset, utilizing data from the Pierre Auger Observatory on cosmic ray detection. This dataset contained physical measurements such as energy, totalEnergy, angles (theta, phi), data at the highest energy event (dEdXmax, distXmax, heightXmax, xmax), galactic coordinates (l, b) and equatorial coordinates (ra, dec). The model was inputted a total of 1565 vectors of shape `[energy, theta, phi, l, b, ra, dec, dedXmax, distXmax, heightXmax, xmax, totalEnergy]`.

### 3.4.1 Data Preprocessing

The raw data was stored in JSON format and needed to be preprocessed before it could be used by the model. I applied the following steps:

- **Standardization**: The dataset was normalized using `StandardScaler` to ensure each feature had zero mean and unit variance.
- **Trimming**: To ensure consistency across dimensions, features were trimmed to the smallest length in the dataset.
- **Train/Validation/Test Split**: The data was split into training (70%), validation (15%), and test (15%) sets using PyTorch's `random_split` function.

The preprocessing pipeline and datasets are provided in the project's GitHub repository for full reproducibility.

### 3.4.2 Data Loading

To facilitate efficient data loading during training, I created a custom PyTorch `Dataset` class. The `DataLoader` was used for batching and shuffling data. A batch size of 128 was applied across all experiments.

## 3.5 Training

### 3.5.1 Hyperparameters

For the 12-dimensional model, the following training hyperparameters were applied:

- **Number of blocks**: 8
- **Hidden layer size**: 256
- **Learning rate**: $1 \cdot 10^{-4}$
- **Number of epochs**: 200, the number is arbitrary due to using early stopping
- **Batch size**: 128

I used the Adam optimizer, known for its adaptive learning rate, along with a scheduler that decreased the learning rate by a factor of 0.1 after every 10 epochs.

### 3.5.2 Early Stopping

Since the model was prone to overfitting, early stopping was employed with a patience threshold of 12 epochs. If the validation loss did not improve by at least 1e-4 over this period, training was terminated early.

### 3.5.3 Loss Function

The model was trained to maximize the likelihood of the data under the learned distribution using the negative log-likelihood (NLL) loss function. This involved computing the log probability of the transformed data under the Gaussian prior and adjusting for the Jacobian's log-determinant to account for the transformations applied by the coupling layers.

## 3.6 Challenges with SVI

In the 12-dimensional experiment, I attempted to incorporate Stochastic Variational Inference (SVI) to learn the posterior distributions of the model's weights and better capture uncertainty. However, I faced several difficulties in its implementation, which ultimately limited my ability to successfully add Bayesian layers to the high-dimensional RealNVP model. Further investigation into PyTorch's `torch.distributions` module or alternative Bayesian deep learning frameworks will be necessary to overcome this limitation.

## 3.7 Experiments and Evaluation

### 3.7.1 Evaluation Metrics

For the evaluation of the 12-dimensional dataset, I utilized several key metrics to assess model performance:

- **Loss Curves**: I monitored the training and validation loss over time to ensure the model was converging appropriately.
- **KDE Overlap**: I used Kernel Density Estimation (KDE) to measure the overlap between the distribution of generated samples and the original data, which helped evaluate how well the model learned the underlying data distribution.

No significant results were obtained by using SVI in the final 12-dimensional experiment due to challenges with incorporating SVI and Bayesian weights.

# 4 Results and Discussion

In this section I will only discuss the results of the 12-dimensional data, since it is the main part of my project. The results are from `RealNVP.ipynb`, so the non-Bayesian approach with the RealNVP architecture.

## 4.1 Results

The following figures illustrate the training and validation loss, as well as the overlap between the original and generated test set data. The Kernel Density Estimation (KDE) overlap was approximately 40%, indicating a moderate similarity between the distributions of the generated and original datasets.

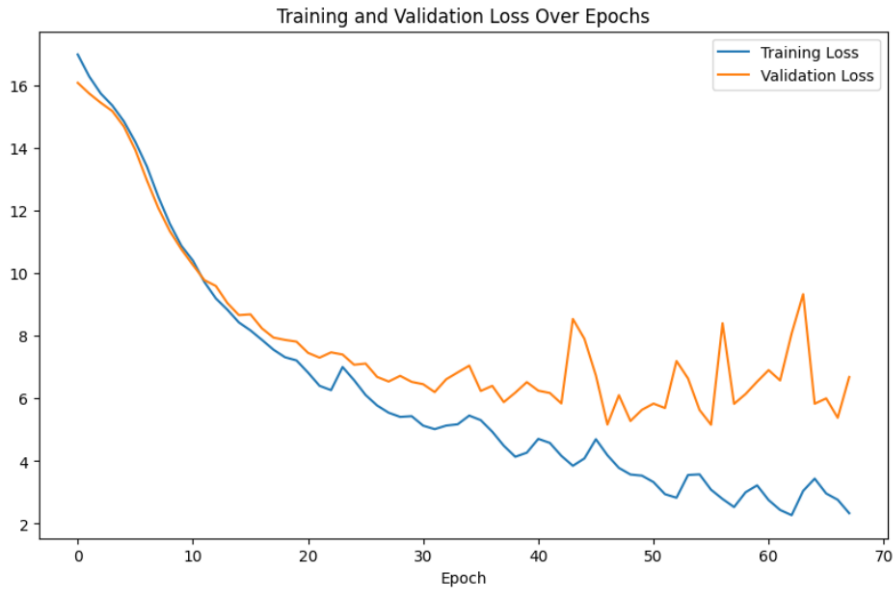


Figure 1: Training and validation Loss Over Epochs

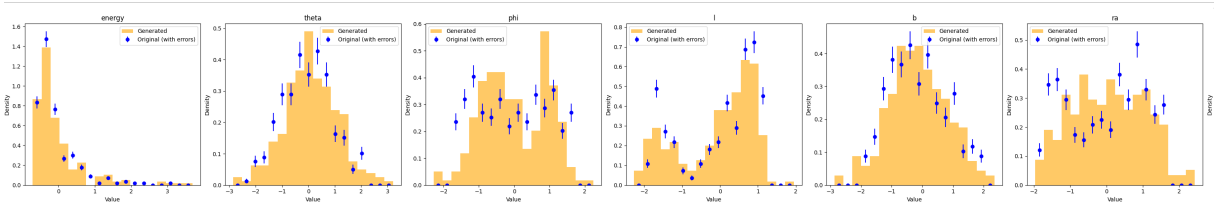


Figure 2: Overlap: energy, theta, phi, l, b and ra

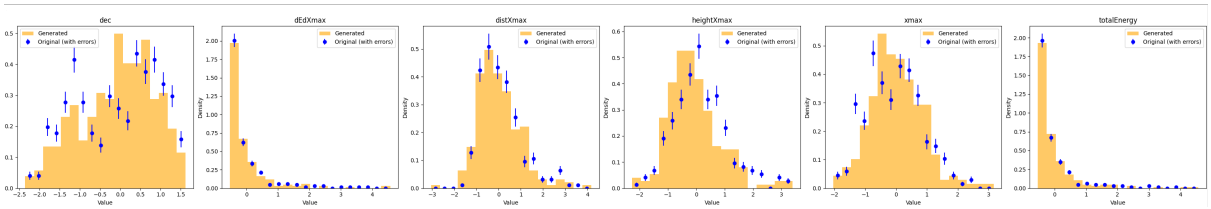


Figure 3: Overlap: dec, dEdXmax, distXmax, heightXmax, xmax and totalEnergy

## 4.2 Discussion

The relatively simple structure of the distributions suggests that affine transformations could be sufficient for accurate results. The small dataset size, however, likely benefits from Bayesian techniques, as evidenced by the success of Bayesian approaches in the 2D case.

Future work will address the SVI implementation challenges and explore more advanced Bayesian techniques like Bayesian Neural Networks or Stochastic Gradient Langevin Dynamics (SGLD). Further optimization of the RealNVP architecture may improve results on complex datasets like the 12D cosmic ray data.

# 5 Summary and Conclusion

This project explored the use of RealNVP flow models for generating high-dimensional data, starting with 2D and 4D experiments and scaling up to a 12-dimensional dataset from the Pierre Auger Observatory. The RealNVP models successfully captured the data distribution in 2-dimensional, 4-dimensional, and 12-dimensional settings. However, while Bayesian techniques were effective for uncertainty quantification in lower dimensions, I faced implementation challenges when applying them to the 12-dimensional data. In the 12D case, the model achieved a moderate performance, with the Kernel Density Estimation (KDE) overlap around 40%.

In conclusion, my time at the Jožef Stefan Institute has been a valuable learning experience. I am deeply grateful to Prof. Dr. Jernej F. Kamenik for his guidance. This project has expanded my knowledge of generative models and inspired further exploration of machine learning techniques.

# 6 References

## 6.1 Code and Availability

To ensure retracability all of the code and the zip of the used data is available on this Github repository [Generating highdimensional data using Flow modes].

## 6.2 References

The report was written by me and the grammer was corrected by ChatGPT.
[1] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," arXiv preprint arXiv:1605.08803, 2016.

[2] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked Autoregressive Flow for Density Estimation," arXiv preprint arXiv:1705.07057, 2017. [Online]. Available: https://doi.org/10.48550/a

[3] D. J. Rezende, and S. Mohamed, "Variational Inference with Normalizing Flows," in Proceedings of the 32nd International Conference on Machine Learning, arXiv preprint arXiv:1505.05770, 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1505.05770.

[4] P. F. Bystrom, D. Krefl, and J. Rojo, "End-to-end simulation of particle physics events with Flow Matching and generator Oversampling," arXiv preprint arXiv:2402.13684, 2024. [Online]. Available: https://arxiv.org/html/2402.13684v2.

[5] E. Jang, "Normalizing Flows Tutorial: Introduction to Invertible Generative Models," Eric Jang's Blog, Jan. 2018. Available: `https://blog.evjang.com/2018/01/nf1.html`.

[6] L. Weng, "Flow-based Models: Normalizing Flows," Lil'Log, Oct. 2018. Available: `https://lilianweng.github.io/posts/2018-10-13-flow-models/`.