Inferring Initial Conditions in the 3-body Problem

— Project Report —
Simulation Based Inference

GROUP 23

Ömercan Misirlioglu - 267644

Tobias Rumpe - 229467

Yordan Saputra - 270054

August 2025

TU Dortmund University

Contents

1 Introduction

In 1687, Newton published *Philosophiæ Naturalis Principia Mathematica*, in which he formulated a general solution to the two-body problem. Naturally, this led him to consider the three-body problem. At first glance, the three-body problem does not appear significantly more complicated than its two-body counterpart. Yet, it was here that Newton famously concluded that no analytical solution exists. So much so that he attributed the stability of the solar system to divine intervention, suggesting that God periodically adjusted planetary orbits.

Although an analytical solution remains impossible, numerical methods can accurately predict the motion of three-body systems. However, another challenge arises: when three bodies orbit one another, the system becomes chaotic. This means that even a minuscule change in initial conditions can lead to vastly different trajectories over time. Given this sensitivity, we aim to infer the initial positions of three-body systems based on one or more observations at a later time. Furthermore, because the system is chaotic, the high dimensionality of the parameter space and the inherent uncertainty in the posterior distribution pose a significant computational challenge.

As for modeling, we first simulate the motion of the three-body system using a high-accuracy ODE solver, such as the Runge–Kutta (RK4) method, while adding approximately 1% Gaussian noise to simulate observational uncertainty. The results of these simulations are then used as training data for our BayesFlow model, which can generate neural networks for parameter estimation, model comparison, and model validation when working with intractable simulators whose overall behavior is too complex to be described analytically (Radev et al. (2023)). The estimator then learns to map observed trajectories to a posterior distribution over the initial positions and velocities. Finally, we iterate over all promising model configurations to identify the one that delivers the best performance.

2 Data

The dataset used in this study consists entirely of artificially generated trajectories of a three-body gravitational system. Since the governing equations of motion are fully specified by gravitational force (also known as Newton's law of universal gravitation), the system can be simulated with high accuracy using numerical approximation. We use the Runge–Kutta 4th order method, chosen for its stability and accuracy in solving ordinary differential equations, to produce time series of positions and velocities for all three bodies. To simulate realistic measurement uncertainty, Gaussian noise with a standard deviation approximately 1% of the magnitude is added to the simulated trajectories.

3 Statistical Model (Simulator)

For simplicity, there are a few assumptions that we have to make about the three-body system:

- **Planar Motion**: The system is confined in a two-dimensional plane.
- **Constant Masses**: All three objects have fixed (constant) masses.
- **Point Masses**: The objects are treated as point masses, meaning collisions are ignored.
- **Normalized Gravitational Constant**: The gravitational constant G is set to 1, simplifying the model without loss of generality.

The proper priors for the initial positions and velocities of each of the three bodies are defined as $\mathbf{r_1}(x_1, y_1)$, $\mathbf{r_2}(x_2, y_2)$, $\mathbf{r_3}(x_3, y_3)$, $\mathbf{v_1}(vx_1, vy_1)$, $\mathbf{v_2}(vx_2, vy_2)$, $\mathbf{v_3}(vx_3, vy_3)$, $\in \mathbb{R}^2$. Since the system is chaotic, the prior variance is kept relatively low to encourage plausible interactions and avoid unstable configurations such as immediate collapse or escape trajectories. Accordingly, the initial positions and velocities are drawn from either a multivariate uniform distribution $\mathcal{U}_{[-1,1]^2}$ or a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I_2})$.

As mentioned earlier, for each iteration of the model setting, we simulate the initial positions and velocities independently from either a uniform or a normal distribution, with the choice of distribution specified as a hyperparameter. The trajectories are computed using the RK4 method with a time step of $dt = 0.001$ to ensure numerical accuracy, after which 1% Gaussian noise is added. The trajectories are snapshotted every 10 time steps, and the total simulation length is determined by another hyperparameter, the number of snapshots T, which takes values from the set $\{10, 20, 50, 100\}$.

Each simulated sample consists of a sequence of 2D positions for the three bodies over a fixed number of snapshots. Two-dimensional motion is chosen instead of three-dimensional motion to reduce computational complexity while still capturing the system's chaotic behavior. The resulting datasets are stored as NumPy arrays for easier integration with the BayesFlow framework. This approach ensures that we can generate as much training and evaluation data as needed.

4 Approximator

Our Bayesflow neural approximator can be split into summary network and inference network.

4.1 Summary Network

Since our simulator outputs three-dimensional tensors of shape `(batch size, num of variables = 12, num of snapshots = T)`, we need to reduce this into two-dimensional tensors of shape `(batch size, summary dim)`. Because our data is a time series, we consider several potential summary networks:

- **Deep Sets**: implements a permutation-invariant neural network for learning structured representations of exchangeable (IID) set-based data, using stacked equivariant layers followed by invariant pooling (Zaheer et al. (2017)). This network is not a great fit for time series but we included it anyway to use it as a baseline.

- **Time Series Network**: implements a hybrid convolutional-recurrent architecture for summarizing multivariate time series, combining local feature extraction (CNN) with long-term dependency modeling (RNN) (Zhang and Mikelsons (2023)).
- **Time Series Transformer**: combines a standard transformer with Time2Vec embeddings to process time series data, optionally incorporating a time vector for handling irregular intervals (Radev et al. (2023)).
- **LSTM Network**: implements a recurrent neural network based on Long Short-Term Memory (LSTM) units, which are designed to capture long-range temporal dependencies in sequential data by mitigating the vanishing gradient problem (Hochreiter and Schmidhuber (1997)). The architecture consists of a LSTM layer for sequence encoding followed by a dense layer to produce a fixed-length summary vector.

For the network configuration, we use a summary dimension of 32 to ensure it is not smaller than the number of parameters in the models. For Deep Sets, the time series network, and the time series transformer, we use the default layers provided in BayesFlow with a summary dimension of 32. For LSTM, which is not provided in BayesFlow, we implement a custom architecture consisting of an LSTM layer with 64 units followed by a dense layer with 32 units to match the summary dimension of the other summary networks.

4.2 Inference Network

For inference network, we consider several options:

- **Coupling Flow**: implements an invertible coupling flow combining ActNorm, learned permutations, and flexible coupling layers for density estimation, supporting various base distributions, such as standard normal (Kingma and Dhariwal (2018), Durkan et al. (2019), Ardizzone et al. (2021), Radev et al. (2020), Alexanderson and Henter (2020)).
- **Flow Matching**: implements Optimal Transport Flow Matching for learning distribution transformations via configurable subnets, supporting both standard and optimal transport enhanced flow matching for improved efficiency. (Liu et al. (2022), Lipman et al. (2023), Tong et al. (2024)).
- **Consistency Model**: implements a Consistency Model with Consistency Training (Song et al. (2023), Song and Dhariwal (2023), Schmitt et al. (2024)).

5 Training

To provide further detail on the training configuration of our model, we employ a dataset consisting of 6,000 training samples and 300 validation samples. The training is carried out over 50 epochs with a batch size of 1,024. Given this configuration, one can approximate the batches per epoch with the following equation: $\lceil 6000/1024 \rceil = 6$. Futhermore, the training regime is executed in the offline way.

For optimization, the default settings are employed. In this case that is an AdamW optimizer which is an improved version of Adam, by decoupling the weight decay from the gradient update step (Zhuang et al. (2022)).

Then, for the learning rate we select the initial value of 5e-4. The learning rate follows a cosine decay schedule, where the final learning rate is scaled proportionally to the square of the initial learning rate. Additionally, the schedule incorporates a warm-up phase with 5% of the total epochs. During this phase the learning rate is gradually increased from 10% of the initial value to the full initial learning rate provided to the workflow.

6 Diagnostics and Inference

The following diagnostics below are handpicked from the top-performing models:

- **For number of snapshots** $T = 10$
  - **Normal Prior, Coupling Flow with LSTM Summary Network**:
    Calibration histograms and rank ECDFs are not within the desirable interval, with significant deviations for the parameters vx_1, vy_1, vx_2, vy_2, vx_3, vy_3. Parameter recoveries appear to be unbiased, but the variance of the posterior z-scores is higher than desired. Posterior contraction values are close to 1, which is a positive outcome. These results may be explained by the insufficient number of snapshots used ($T = 10$).
  - **Uniform Prior, Coupling Flow with LSTM Summary Network**:
    Calibration histograms and rank ECDFs are mostly within the desirable interval, though some deviations occur for the parameters vx_1, vy_1, vx_2, vy_2, vx_3, vy_3. Recoveries are largely unbiased, and posterior contraction values remain close to 1. The variance of posterior z-scores is lower compared to the normal prior at $T = 10$. Remaining deviations may again be attributed to the limited number of snapshots.
- **For number of snapshots** $T = 20$
  - **Normal Prior, Coupling Flow with LSTM Summary Network**:
    Calibration histograms (2) and rank ECDFs (3, 4) improve compared to $T = 10$, with fewer and smaller deviations across the parameters vx_1, vy_1, vx_2, vy_2, vx_3, vy_3. Recoveries (5) remain unbiased, posterior contraction values are close to 1, and the variance of z-scores is reduced relative to $T = 10$ (6).
  - **Uniform Prior, Coupling Flow with LSTM Summary Network**:
    Calibration histograms and rank ECDFs are largely in order, with no major deviations. However, recoveries for some parameters begin to show bias, and

posterior contraction values decrease to around 0.6-0.8, with a subset clustering directly at 0. For longer snapshots ($T = 50$ and $T = 100$), calibration remains nearly perfect, but recoveries become drastically biased and contraction decreases significantly for about half of the parameters.

- **For number of snapshots** $T = 50$
  - **Normal Prior, Coupling Flow with LSTM Summary Network**: Calibration histograms and rank ECDFs are in order with only negligible deviations. Parameter recoveries are nearly unbiased, aside from slight, non-significant biases across all velocity parameters. Posterior contraction values are generally close to 1, with a few exceptions near 0. Overall, performance improves noticeably at $T = 50$.
  - **Normal Prior, Coupling Flow with DeepSet Summary Network**: Calibration histograms and rank ECDFs are in order except for a slight deviation in one parameter. Parameter recoveries are unbiased, with the exception of two parameters (vy_2, vy_3) that fail to recover correctly. Posterior contraction values are close to 1, except for these same two parameters, where contraction values are significantly lower, consistent with the failed recoveries.
  - **Uniform Prior, Coupling Flow with LSTM Summary Network**: As noted above, performance deteriorates further with increasing T, showing more biased recoveries and lower contraction values.
- **For number of snapshots** $T = 100$
  - **Normal Prior, Coupling Flow with LSTM Summary Network**: Calibration histograms and rank ECDFs are well-aligned, with little to no deviation. Parameter recoveries show slight bias. Posterior contraction values are close to 1 for all location parameters, but cluster around 0.6-0.8 for the velocity parameters. Some other model variants were tested for $T = 100$, but none provided satisfactory results.

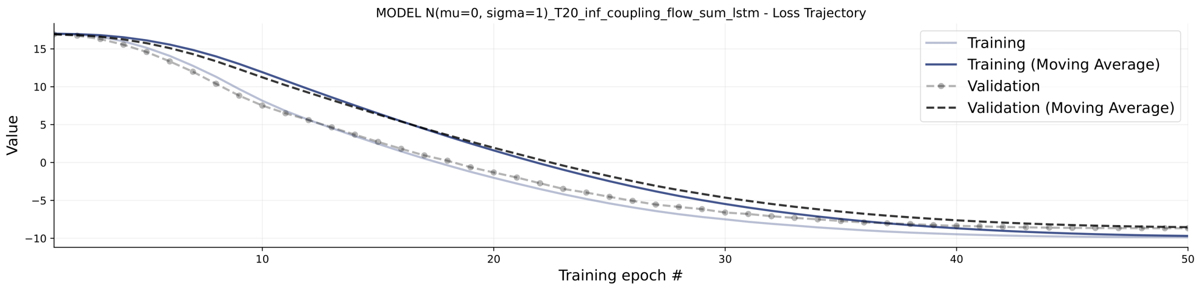

Figure 1: Training and Validation Losses of Normal Prior, Coupling Flow with LSTM Network at $T = 20$

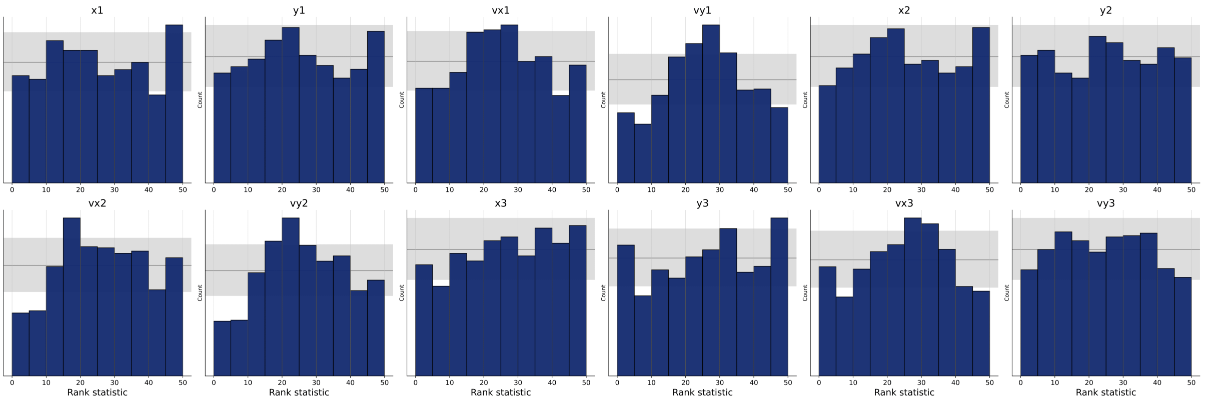

Figure 2: Calibration Histogram of Normal Prior, Coupling Flow with LSTM Network at $T = 20$

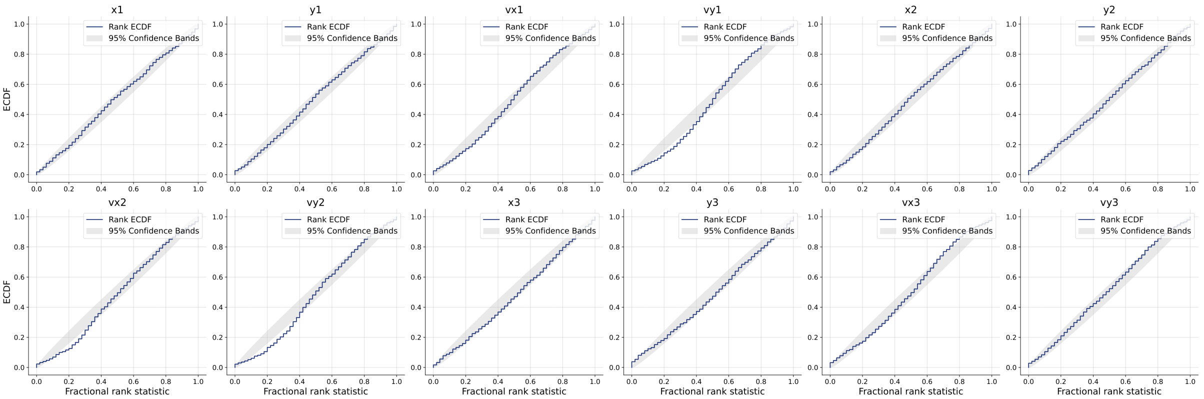

Figure 3: ECDF Plot of Normal Prior, Coupling Flow with LSTM Network at $T = 20$

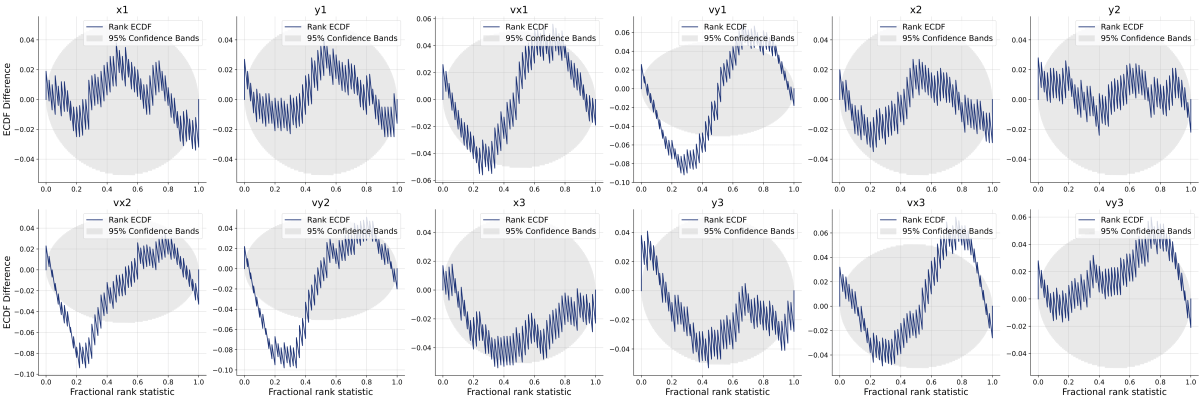

Figure 4: ECDF Difference Plot of Normal Prior, Coupling Flow with LSTM Network at $T = 20$

In summary, across all model configurations, both training and validation losses (1) decrease smoothly without instability, indicating stable optimization . The parallel tra-

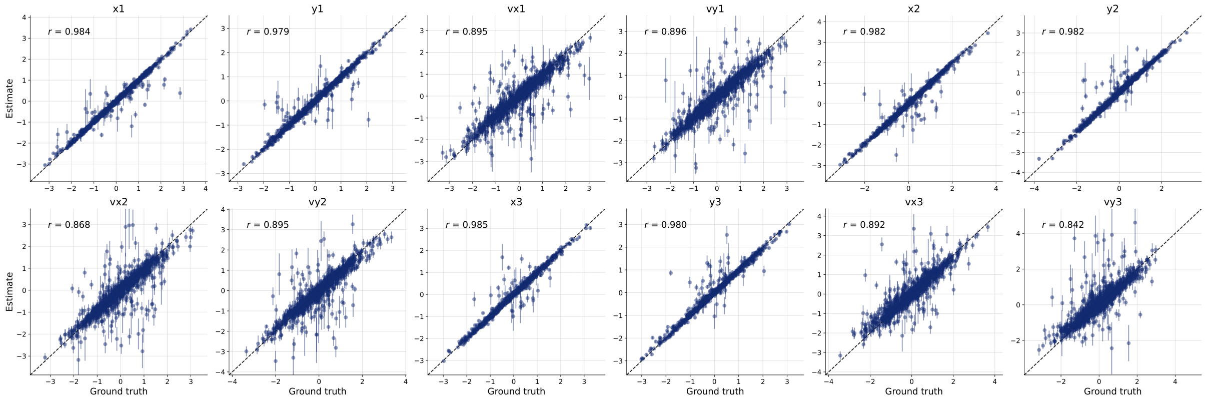

Figure 5: Parameters Recovery Plot of Normal Prior, Coupling Flow with LSTM Network at $T = 20$

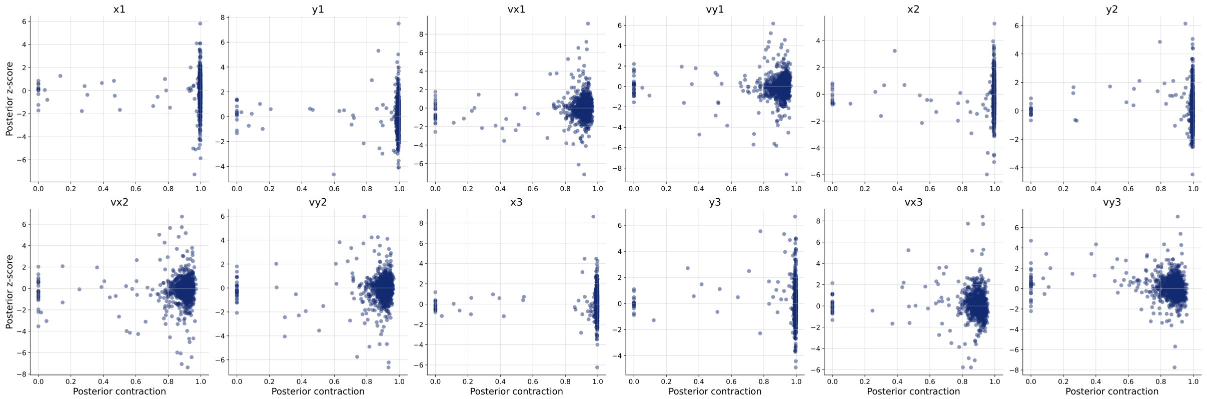

Figure 6: Posterior Contraction and z-score Plot of Normal Prior, Coupling Flow with LSTM Network at $T = 20$

jectories of these curves demonstrate strong generalization with no signs of overfitting. In later epochs, the losses approach a steady plateau, confirming that all models reached proper convergence under the specified experimental conditions.

The diagnostics indicate that the inference procedure has converged reliably and the simulation-based calibration results show that the posterior distributions are well-calibrated with respect to the true parameters. Additional checks such as posterior contraction confirm that the posterior mass concentrates around the ground truth as the amount of information increases, which provides further evidence for the validity of the model.

Among the different candidate architectures we evaluated, the Coupling Flow with LSTM proved to be the most robust choice. It consistently delivered acceptable results across all considered values of T, while some alternative models showed less stable performance or systematic deviations. Although slight deviations were still observed for certain T values, these remained within an acceptable range and did not undermine the reliability of the inference.

When applying the selected model to the test data generated by the simulator, the results aligned well with the expected behavior, supporting both the accuracy and robustness of the inference approach. Taken together, these findings justify the choice of the Coupling Flow with LSTM as the final model and demonstrate that it provides reliable posterior estimates within the scope of the study.

7 Conclusion

In this project, we investigate the challenge of inferring the initial conditions of the three-body problem using simulation-based inference techniques. By using a high-accuracy ODE solver such as RK4 together with BayesFlow, we are able to map observed (synthetic) trajectories to posterior distributions of initial positions and velocities. Our experiments demonstrate that the Coupling Flow with an LSTM summary network provides the most robust and consistent results across different snapshot lengths, outperforming alternative models.

Although velocity parameters remain more difficult to estimate accurately, especially at longer snapshot horizons, the overall inference procedure remains stable and converges consistently. The posterior distribution is generally approximated closely to the correct posterior, demonstrating the effectiveness of the chosen model.

Ultimately, this work shows that simulation-based inference can successfully handle the complexity of three-body systems, offering an alternative method to infer initial conditions. However, improvements to the model are still possible, such as implementing online training, varying the masses of the three bodies, allowing for longer training times (currently limited by hardware), adopting more state-of-the-art network architectures, using larger training and evaluation datasets, extending the setup from 2D to 3D, and increasing prior variance to cover a wider range of possibilities.

Reflecting on our learning, this project has shown us that simulation-based inference makes it possible to infer parameters of a complex model without an analytical solution. While the results are not perfectly accurate, they still provide meaningful insights, and the process is relatively straightforward compared to more conventional methods. Initially, we mistakenly attempted to solve the problem with a simple neural network rather than a simulation-based inference approach, but this project helped us to understand the fundamental difference between the two and the strengths of simulation-based methods.

References

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. https://arxiv.org/abs/1807.03039

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2019). Neural spline flows. https://arxiv.org/abs/1906.04032

Alexanderson, S., & Henter, G. E. (2020). Robust model training and generalisation with studentising flows. https://arxiv.org/abs/2006.06599

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., & Köthe, U. (2020). Bayesflow: Learning complex stochastic models with invertible neural networks. https://arxiv.org/abs/2003.06281

Ardizzone, L., Kruse, J., Lüth, C., Bracher, N., Rother, C., & Köthe, U. (2021). Conditional invertible neural networks for diverse image-to-image translation. https://arxiv.org/abs/2105.02104

Liu, X., Gong, C., & Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. https://arxiv.org/abs/2209.03003

Zhuang, Z., Liu, M., Cutkosky, A., & Orabona, F. (2022). Understanding adamw through proximal methods and scale-freeness. https://arxiv.org/abs/2202.00089

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., & Le, M. (2023). Flow matching for generative modeling. https://arxiv.org/abs/2210.02747

Radev, S. T., Schmitt, M., Schumacher, L., Elsemüller, L., Pratz, V., Schälte, Y., Köthe, U., & Bürkner, P.-C. (2023). BayesFlow: Amortized Bayesian workflows with neural networks. *Journal of Open Source Software*, *8*(89), 5702.

Song, Y., & Dhariwal, P. (2023). Improved techniques for training consistency models. https://arxiv.org/abs/2310.14189

Song, Y., Dhariwal, P., Chen, M., & Sutskever, I. (2023). Consistency models. https://arxiv.org/abs/2303.01469

Zhang, Y., & Mikelsons, L. (2023). Solving stochastic inverse problems with stochastic bayesflow. *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 966–972. https://doi.org/10.1109/aim46323.2023.10196190

Schmitt, M., Pratz, V., Köthe, U., Bürkner, P.-C., & Radev, S. T. (2024). Consistency models for scalable and fast simulation-based inference. https://arxiv.org/abs/2312.05440

Tong, A., Fatras, K., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Wolf, G., & Bengio, Y. (2024). Improving and generalizing flow-based generative models with minibatch optimal transport. https://arxiv.org/abs/2302.00482