Neural Markov Jump Processes

Patrick Seifner [1 2] **Ramsés J. Sánchez** [1 2]

Abstract

Markov jump processes are continuous-time stochastic processes with a wide range of applications in both natural and social sciences. Despite their widespread use, inference in these models is highly non-trivial and typically proceeds via either Monte Carlo or expectation-maximization methods. In this work we introduce an alternative, variational inference algorithm for Markov jump processes which relies on neural ordinary differential equations, and is trainable via back-propagation. Our methodology learns neural, continuous-time representations of the observed data, that are used to approximate the initial distribution and *time-dependent* transition probability rates of the posterior Markov jump process. The *time-independent* rates of the prior process are in contrast trained akin to generative adversarial networks. We test our approach on synthetic data sampled from ground-truth Markov jump processes, experimental switching ion channel data and molecular dynamics simulations. Source code to reproduce our experiments is available online.[1]

1. Introduction

Markov Jump Processes (MJP) model time-dependent, discrete probability distributions whose future is entirely determined by knowledge of their present. Given an initial distribution, the time evolution in these processes is completely characterized by the instantaneous transition probability rates between their states, in terms of which one can express quantities of interest that can be probed experimentally. Mean first-passage times, which provide information regarding the short-term dynamics of the process, relaxation times to stationarity and the stationary distribution itself, which concern the long-time asymptotics, or thermodynamic quantities like (stochastic) entropy production, can all be computed via the MJP transition rates.

Such a family of mathematical models is simple enough to have a wide range of approximate validity, which has led to its application in many fields of science. Examples abound, and range from physics (Griffiths et al., 1966), chemistry (Gillespie, 1977) and biology (Kimura, 1980) all the way up to meteorology (Madsen et al., 1985), finance (Turner et al., 1989) and sociology (Singer & Spilerman, 1976). Indeed, there is a large class of systems whose evolution can be phenomenologically understood as the result of individual encounters between members of some population – think of reacting chemical species or population systems, which die, mate and give birth. Researchers can model such systems by assigning probability rates to the transitions caused by those encounters. Likewise many large, complex systems often give rise to long-lived collective modes, trapped in a set of metastable states at some macroscopic scale of observation, whose dynamics can also naturally be modelled in terms of transition rates between their metastable states. In practice, however, a judicious choice of the transition probability rates from first principles, in the large majority of these systems, is seldom analytically tractable, which leads researchers to resort to statistical methods in order to infer them directly from raw data.

When a continuous record of the empirical process under investigation exists, maximum likelihood estimation of the transition rates is simple and well understood (Billingsley, 1961). When, on the contrary, the process is observed at discrete time points only, as it is often the case, inference becomes highly non-trivial and typically proceeds via either Markov Chain Monte Carlo (MCMC) or Expectation Maximization (EM) methods. Yet MCMC, while accurate, does not normally scale well with data (Zhang et al., 2017), whereas EM is frequently formulated to yield only point estimates of its unknown parameters (Opper & Sanguinetti, 2007; Köhs et al., 2021) and can also struggle when faced with large datasets.

In this work we introduce an alternative, variational inference algorithm for MJP – the Neural Markov Jump Process model (NeuralMJP) – which relies on neural variational inference (NVI) (Kingma & Welling, 2014; Rezende et al.,

[1]Lamarr Institute, Bonn, Germany. [2]University of Bonn, Bonn, Germany. Correspondence to: Patrick Seifner <seifner@cs.uni-bonn.de>, Ramsés J. Sánchez <sanchez@bit.uni-bonn.de>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]https://github.com/pseifner/NeuralMJP

2014) and Neural Ordinary Differential Equations (NeuralODE) (Chen et al., 2018). In short, our model encodes neural representations of (noisy) data into the parameters of a master equation describing the time-evolution of the posterior process. Numerically solving this equation yields the instantaneous posterior distribution encoding the data. At the same time, the model learns a global, implicit distributions over the prior MJP, through the minimization of its Kullback-Leibler divergence wrt. the posterior process. In what follows we first review previous work on the inference of MJP (Section 2), and revisit some of the basics of the theory of MJP (Section 3). Section 4 introduces our model, the inference algorithm and some general details about training. We test our methodology on four different datasets in Section 5. Finally, Section 6 closes the paper with some concluding remarks about future work, while Section 7 comments on the main limitations of our approach.

2. Related Work

The question of whether observations on a given empirical process are consistent with an underlying MJP is an old one (Singer & Spilerman, 1976), and goes all the way back to Elfving (1937) who formulated it as an embedding problem. Since then, the problem of parameter estimation for MJP has been approached from many different angles. For example, maximum likelihood estimators for MJP transition rates were first obtained via EM by Asmussen et al. (1996). Their approach was later rediscovered by Bladt & Sørensen (2005) and optimized further by Metzner et al. (2007). Other (maximum likelihood) formulations estimate the transition rates indirectly, by first fitting discrete-time Markov Chains to the data (Crommelin & Vanden-Eijnden, 2006; McGibbon & Pande, 2015). Likewise, Bayesian solutions to the problem of inferring MJP from noisy observations exist in many flavours. Boys et al. (2008), for example, proposed a MCMC algorithm for parameter inference of certain MJP classes. Fearnhead & Sherlock (2006) introduced an exact sampler for Markov-modulated Poisson processes, which was later extended and generalized, using an auxiliary-variable trick, by Rao & Teg (2013). The latter can however experience slow-mixing problems, and its convergence is often difficult to quantify. Small-variance asymptotic approaches (Huggins et al., 2015) and collapsed variational MCMC algorithms (Zhang et al., 2017) were recently introduced to tackle (some of) these issues.

Parallel to these efforts is a line of deterministic, variational models, the first of which was introduced by Opper & Sanguinetti (2007). Their model uses a mean-field approximation to estimate the posterior distribution of coupled MJP and is optimized via EM. Later Cohn et al. (2010) applied Opper's approach to continuous-time Bayesian networks, while Wildner & Koeppl (2019) partitioned the set of transitions of the MJP, to express Opper's variational objective in terms of natural moment functions. More recently, Köhs et al. (2021) proposed a variational EM scheme to infer switching diffusion processes modulated by (hidden) MJPs. Nevertheless, all these models are formulated to yield only point estimates of the MJP parameters they infer.

Finally, most neural-based approaches tackle inference of MJP mainly in specific representations. Point processes, for example, have been modelled with recurrent (Du et al., 2016; Mei & Eisner, 2017) and NeuralODE (Jia & Benson, 2019; Chen et al., 2021) networks, both trained via maximum likelihood. Ojeda et al. (2021a), in contrast, leveraged recurrent, generative adversarial networks to infer queuing processes. To the best of our knowledge, our methodology is the first, neural-based variational solution to the problem of posterior inference and parameter estimation of hidden Markov jump processes.

3. Background

Let us consider a system S that can be described in terms of a stochastic process $Z(t)$ taking values on some countable set $\mathcal{Z}$, over a finite time interval $[0, T]$. We define the instantaneous probability rate of transiting from state z' to another state $z \in \mathcal{Z}$ different from z' as

$$f(z|z', t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} p(z, t + \Delta t | z', t), \tag{1}$$

where $p(z, t|z', t')$ is the probability to find the system in state z at time t, given that it was in state z' at time $t' < t$. If all changes in S are only due to transitions of the form of Eq. 1, then S is Markovian, and one can readily show that the probability distribution describing the state of the system evolves according to the *master equation*

$$\dot{p}(z, t) = \sum_{z' \neq z} \Big(f(z|z', t) p(z', t) - f(z'|z, t) p(z, t) \Big), \tag{2}$$

where we use $\dot{p}(z, t)$ to denote the time derivative of $p(z, t)$. See Appendix A for a brief derivation. The stochastic process $Z(t)$ associated with (the solution of) this equation is made up of right-continuous, piecewise-constant paths (see Appendix G), which explains why one usually refers to it as a Markov jump process.

In this work we concentrate on (the inference of) *homogeneous* MJP, whose transition probability rates f are time-independent, taking values on finite state spaces of size $|\mathcal{Z}| = K$. Note that in this case we can gather all transition rates into a $K \times K$ *rate matrix* $\mathbf{F}$, and rewrite the master equation as $\dot{\mathbf{p}}(t) = \mathbf{p}(t) \cdot \mathbf{F}$, where $\mathbf{p}(t) \in [0, 1]^K$ is the vector containing the time-dependent probabilities over the K states in $\mathcal{Z}$. We can now formally write the solution to the master equation in terms of the matrix exponential $\mathbf{p}(t) = \mathbf{p}(0) \cdot \exp(\mathbf{F}t)$.

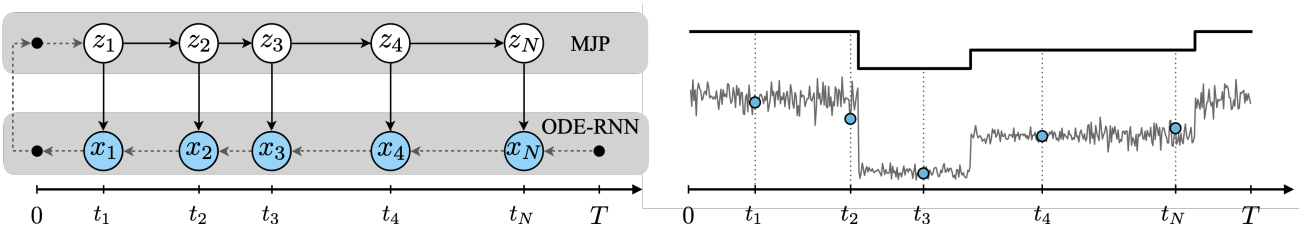

Figure 1. Neural Markov Jump Process model. Left: Graphical model and encoding-decoding procedure. The observations $\mathbf{x}_1, \dots, \mathbf{x}_N$ are encoded backwards in time with a ODE-RNN model (bottom block). The output ODE-RNN representations are used to condition the posterior master equation (Eq. 4), which is solved forward in time (top block). The instantaneous posterior distribution is then sampled at $t_1, \dots, t_N$. The resulting states $z_1, \dots, z_N$ are decoded with the emission model. Right: Illustration of a one-dimension noisy signal (bottom) and its underlying MJP (top). The discrete observations are marked with circles at the observation times $t_1, \dots, t_N$.

The matrix exponential representation naturally leads to the notion of *relaxation times* to stationarity. Indeed, let $\lambda_1, \lambda_2, \dots, \lambda_K$ denote the eigenvalues of $\mathbf{F}$. If all K states in S are connected (i.e. if the Markov chain associated to the process is *irreducible*), the Perron-Frobenious theorem states that $0 = \lambda_1 > \text{Re}(\lambda_2) \geq \cdots \geq \text{Re}(\lambda_K)$, where Re denotes real part (Berman & Plemmons, 1994). It follows that the class probabilities $\mathbf{p}(t)$ are nothing but linear combinations of the terms $1, \exp(-\text{Re}(\lambda_2)t), \dots, \exp(-\text{Re}(\lambda_K)t)$, maybe multiplied by some oscillatory functions. The relaxation time to stationarity is then simply given by the longest time scale in the system, namely $\text{Re}(\lambda_2)^{-1}$. In the long-time limit, all those exponential terms vanish away and one is only left with a distribution $\mathbf{p}^*(t)$ which, at most, oscillates among its states with time. This limit distribution is known as the *stationary distribution* of the process. Appendix B introduces it (properly).

The transition rates can also be used to obtain information regarding the short-TERM behaviour of the MJP. An important example is the *mean-first passage time* $\tau_{zz'}$, which measures how long the system would take to reach some state of interest $z' \in \mathcal{Z}$, given that it started in some other state z, at the beginning of the process. Appendix B defines this time in terms of the rate matrix $\mathbf{F}$.

4. Neural Markov Jump Processes

We address the general problem of inferring homogeneous MJP taking values on some countable set of size K, from a set of (noisy) observations on a given empirical process. The empirical process in question can itself lie on some countable, discrete space, like when one records the population of individuals in a given system. It can lie on a low-dimensional, continuous space, like when one analyses one-dimensional noisy signals jumping between K different mean values. It can also lie on a high-dimensional space, like when one studies the equilibrium dynamics of the atoms in a molecule, which spend most of their time in a set of K metastable, spatial configurations. Although the posterior process associated to such noisy signals is too a MJP (Bishop & Nasrabadi, 2006), the initial distribution and rate probabilities characterizing it are complex, nonlinear functions of the entire set of observations. This is specially true in the high-dimensional, continuous cases of conformational dynamics, typical of the molecular systems we examine below.

In what follows we use NVI methods (Kingma & Welling, 2014; Rezende et al., 2014) to (i) learn neural functions encoding the discrete-time, noisy signal into the parameters of a master equation describing the time-evolution of the posterior process; (ii) solve, in the spirit of NeuralODE (Chen et al., 2018), the master equation to infer the instantaneous posterior distribution encoding the data; and (iii) learn an implicit distribution on the prior homogeneous MJP that best fits the posterior.

4.1. Generative Model

Let us suppose we are given a set of N observations $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in \mathcal{X}$ on some empirical process, recorded at (non-equidistant) times $0 < t_1 < t_2 < \cdots < t_N < T$, where $\mathcal{X}$ denotes the space of observations. We assume the observations are generated conditioned on a hidden, homogeneous MJP, characterized by a master equation like Eq. 2, but where the time-independent transition rates $f_\theta(z|z')$ are trainable functions, parametrized by some learnable parameter θ. The joint probability distribution over the prior MJP process, together with the set of N observations, can then be written as

$$\prod_{i=1}^{N} p_\theta(\mathbf{x}_i|z) p_\theta(z, t = t_i), \tag{3}$$

where $p_\theta(z, t = t_i)$ is the (instantaneous) solution of the master equation specifying the prior process at time t_i, and $p_\theta(\mathbf{x}_i|z)$ is some emission function modelling the noise in the data, with learnable parameters θ. Let us specify each of the components in Eq. 3.

Algorithm 1. Training NeuralMJP

Requires: Dataset $\mathcal{D}$ and time horizon T
for each training iteration **do**
  Sample: $(t_{0:N}, \mathbf{x}_{1:N}) \sim \mathcal{D}$ and $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, 0.1)$
  (1) Train encoder-decoder pair:
  Freeze prior parameters in θ and free the rest
  Compute: $\mathbf{h}_T$ = ODE-RNN($\mathbf{x}_{1:N}, T$)
  Run: ODESOLVE[$q_\phi(z, 0)$, $\dot{q}_\phi(z, t, \mathbf{g}_\phi)$]
  Sample: $\{z_i \sim q_\phi(z, t_i|\mathbf{h}_T)\}_{i=0}^N$
  Backprob: $\theta, \phi \leftarrow$ Adam($\nabla \log p_\theta(\mathbf{x}_{1:N}|z_{0:N})$)
  (2) Train prior parameters:
  Freeze decoder parameters in θ and free the rest
  Compute: $\mathbf{f}_\theta = \boldsymbol{\Phi}_\theta(\boldsymbol{\varepsilon})$
  Backprob: $\theta \leftarrow$ Adam($\nabla \mathcal{L}_{\text{KL}}(\mathbf{f}_\theta, \mathbf{g}_\phi)$)
end for

Emission Model. The choice of the emission functions depends on the problem at hand. Below we shall deal with different noisy signals on $\mathbb{R}^D$, for some fixed dimension D. We therefore choose Gaussian emission models, whose mean and covariance matrices are non-linear functions of the hidden process state, modelled by neural networks.

Prior MJP Model. The prior MJP is characterized by its initial condition and transition probability rates. We are interested in learning the latter. Let $\mathbf{f}_\theta \in (\mathbb{R}^+)^{(K-1)K}$ denote the set of $(K-1)K$ independent, global parameters of the prior rate matrix. A natural solution to the problem of learning $\mathbf{f}_\theta$ would be to define it as a vector of randomly initialized, trainable parameters. However, we have found empirically that such an approach yields fairly unstable training, which is both prone to get stuck in local minima and highly dependent on the random initialization (see Appendix C for details). The Bayesian alternative would be to impose and learn prior distributions on the entries of $\mathbf{f}_\theta$ itself. Instead we take a more pragmatic approach. We define a generative neural model $\boldsymbol{\Phi}_\theta : \mathbb{R}^p \to (\mathbb{R}^+)^{(K-1)K}$ that, akin to the generator in Generative Adversarial Networks (Goodfellow et al., 2017), maps random p-dimensional vectors, sampled from a Gaussian distribution, into the transition rates $\mathbf{f}_\theta$. That is

$$\mathbf{f}_\theta = \boldsymbol{\Phi}_\theta(\boldsymbol{\varepsilon}), \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma),$$

with σ a hyperparameter of the model. Similar to earlier approaches (Mohamed & Lakshminarayanan, 2016; Huszár, 2017; Yin & Zhou, 2018), this (prior) model implicitly defines a distribution over the rate matrices of homogeneous MJP. Below we demonstrate that the empirical mean of this distribution is sharply peak around the parameters of ground-truth rate matrices.

4.2. Inference Model

Exact inference of the generative model above is clearly intractable. In this section we approximate the true posterior MJP with a variational, inhomogeneous MJP solving the master equation

$$\dot{q}_\phi(z, t|\mathbf{x}_{1:N}) = \sum_{z' \neq z} \Big\{ g_\phi(z|z', t, \mathbf{x}_{1:N}) q_\phi(z', t|\mathbf{x}_{1:N}) - g_\phi(z'|z, t, \mathbf{x}_{1:N}) q_\phi(z, t|\mathbf{x}_{1:N}) \Big\}, \quad (4)$$

with data-dependent initial condition $q_\phi(z, 0|\mathbf{x}_{1:N})$. Here ϕ labels the trainable parameters of the inference model, and $\mathbf{x}_{1:N}$ denotes the sequence of empirical observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Also note that we allow the variational approximation to capture inhomogeneous MJP, simply by making the posterior rates g_ϕ explicitly time dependent.

As can be seen already from Eq. 4, our strategy consists in encoding the empirical process $\mathbf{x}_{1:N}$ into the posterior MJP indirectly, through the initial condition and transition rates defining its master equation. We define the former as

$$q_\phi(z, 0|\mathbf{x}_{1:N}) = \boldsymbol{\Lambda}_\phi(\mathbf{h}_T), \quad \boldsymbol{\Lambda}_\phi : \mathbb{R}^H \to [0, 1]^K,$$

where $\mathbf{h}_T \in \mathbb{R}^H$ is a neural representation encoding $\mathbf{x}_{1:N}$, and $\boldsymbol{\Lambda}_\phi$ is implemented with neural networks. Accordingly, we define the $(K-1)K$ time-dependent transition rates of the posterior MJP, which we denote by $\mathbf{g}_\phi$, as follows

$$\mathbf{g}_\phi(t, \mathbf{x}_{1:N}) = \boldsymbol{\Psi}_\phi(\mathbf{h}_T, t), \;\; \boldsymbol{\Psi}_\phi : \mathbb{R}^H \times \mathbb{R} \to (\mathbb{R}^+)^{(K-1)K}$$

where $\boldsymbol{\Psi}_\phi$ is implemented via neural networks too.

The only missing piece is the neural data representation $\mathbf{h}_T$. This representation can in general be computed using any sequence-processing neural network, like e.g. a LSTM (Hochreiter & Schmidhuber, 1997) or Transformer (Vaswani et al., 2017) network. We experimented somewhat with different architectures, but in the end settled on the ODE-RNN encoder of Rubanova et al. (2019), which we found to be robust in all experiments without too much overhead. For the sake of completeness, let us revisit it briefly.

ODE-RNN Encoder. Consider the time interval $[0, T]$ for some time horizon T, so that $0 < t_1 < \cdots < t_N < T$. Given some initial state $\mathbf{h}_0$, ODE-RNN encodes the sequence $\mathbf{x}_{1:N}$ *backwards in time*, starting from the time horizon $t = T$ and reaching the end of the interval at $t = 0$. ODE-RNN leverages two neural networks, namely (i) a NeuralODE to update the data representation $\mathbf{h}(t)$ in between observations, and (ii) a recurrent neural network to model instantaneous updates at the observation points. With this procedure in mind, we define the representation $\mathbf{h}_T$ encoding the complete data sequence as the output of *the*

Table 1. Inference of (ground-truth) Discrete Flashing Ratchet process. Mean and standard deviation computed with 1000 samples of the generative prior MJP model.

	V	r	b
GROUND TRUTH	1.00	1.00	1.00
IRREGULAR GRID	1.06 ± 0.02	1.17 ± 0.01	1.14 ± 0.02
SHARED GRID	0.97 ± 0.02	1.17 ± 0.02	1.17 ± 0.01
REGULAR GRID	0.98 ± 0.05	1.37 ± 0.04	1.39 ± 0.04

last NeuralODE update, starting at $t = t_1$ and ending at $t = 0$. See Rubanova et al. (2019) for details.

Given $\mathbf{h}_T$, we can now numerically solve the Eq. 4 above to obtain the instantaneous posterior distribution encoding the data at the observation times. Thus NeuralMJP solves one (posterior) master equation for each realization of the empirical process (that is, for each time series in the dataset). Figure 1 summarizes and illustrates this procedure.

4.3. Objective Functions

To optimize the parameter set $\{\theta, \phi\}$ of our latent variable model we maximize a variational lower bound on the logarithm of the marginal likelihood of the data. Following Opper & Sanguinetti (2007), we derive the latter using a discrete-time representation of our MJP, and taking the continuous-time limit at the end of the calculation. See Appendix D for details. Our result reads

$$\begin{aligned}\mathcal{L} \quad = \quad & \int_0^T dt\, \mathbb{E}_{q_\phi(z,t)} \sum_{z' \neq z} \Big\{ g_\phi(z'|z,t,\mathbf{x}_{1:N}) - f_\theta(z'|z) \\ & - g_\phi(z'|z,t,\mathbf{x}_{1:N}) \log \frac{g_\phi(z'|z,t,\mathbf{x}_{1:N})}{f_\theta(z'|z)} \Big\} \\ & + \sum_{i=0}^{N} \mathbb{E}_{q_\phi(z,t_i)} \left[\log p_\theta(\mathbf{x}_i|z(t_i)) \right], \end{aligned} \tag{5}$$

where the integral is nothing but the Kullback-Leibler divergence (Kullback & Leibler, 1951) between prior and posterior MJP, whereas the last term is the reconstruction cost. Let us note that the expectation values in the expression above are computed wrt. the instantaneous posterior distribution $q_\phi(z, t|\mathbf{x}_{1:N})$. We omitted its dependence on the data in the equation above for clarity.

The training algorithm for our model is given in Algorithm 1. Note that we train the prior model separately from the posterior (encoder) and emission (decoder) models. See Appendix E for our reasoning behind this. See also Appendix F for additional details and tricks we employ during training.

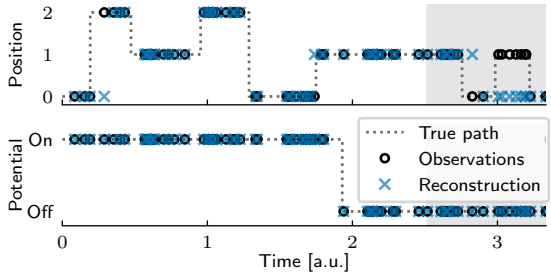

Figure 2. Sample Discrete Flashing Ratchet trajectory. The white region shows greedy samples from the posterior within the observation window. Gray region shows predictions of the model.

4.4. Prediction Process

NeuralMJP naturally allows for prediction of future events of the empirical process, that is, events beyond the time horizon T the model was trained on. Indeed, in order to predict the behaviour of the empirical process at some time $T + l$, for any $l > 0$, one must rely on the generative process of the model, Eq. 3, albeit conditioned on the past.

Essentially, one must generate Monte Carlo samples from the posterior distribution at the end of the observation window (i.e. at T) and propagate the latent representations into the future, either by solving the *prior master equation*, or by simulating the *prior process* (Gillespie, 1977) up to the time of interest $T + l$. Finally, one decodes the samples from the instantaneous prior distribution back onto the space of observation via the emission model. For completeness, we briefly revisit the classical simulation algorithm for MJP in Appendix G.

5. Experiments

In this section we test our model on four different datasets of varying complexity, each characterized by different dimensionality, and corrupted by noise signals of very different nature.

We begin by studying how NeuralMJP handles irregularly sampled data, which we extract from a noiseless simulation of a simple MJP named Discrete Flashing Ratchet process. Next, we train the model on experimental, switching ion channel data, that corresponds to a one-dimensional noisy signal switching among a set of mean values. We infer the equilibrium distribution, and relaxation and mean-first passage times of the hidden MJP, and compare against the switching diffusion model of Köhs et al. (2021). We then use NeuralMJP to infer birth-death processes, a class of coupled (i.e. interacting) MJPs, and compare against the classical solution of Opper & Sanguinetti (2007). Finally, we infer long- and short-time properties of (one all-atom

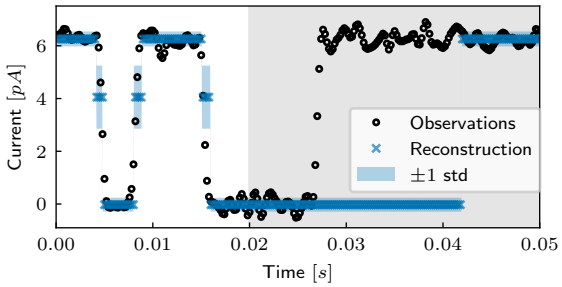

Figure 3. Sample trajectory from Switching Ion Channel dataset. The white region shows mean reconstruction values, conditioned on posterior samples within the observation window. Gray region shows predictions of the model.

simulation of) a small molecule named alanine dipeptide, and compare against the neural, discrete-time variational Markov models of Mardt et al. (2017) and Varolgünes et al. (2019). Appendix N and O contain two more experiments, in which we further compare NeuralMJP against the models of Mardt et al. (2017) and Köhs et al. (2021), respectively.

Before diving in, let us provide some details regarding the training of NeuralMJP, which are common to all the experiments below.

Experimental Details. First of all, we normalize all observation times to lie within the interval $[0, 1]$ and set the time horizon T to 1.1.

To specify the posterior model, we fix the dimension H of $\mathbf{h}_T$ to 256 in all experiments. Its ODE-RNN encoder uses a GRU network (Cho et al., 2014), with a hidden dimension of 256, for the instantaneous updates, and solves a NeuralODE network, parametrized by an MLP with $[256, 256]$ layers, backwards in time with the Runge-Kutta method, starting from T with initial condition $\mathbf{h}_0 = 0$. The functions $\mathbf{\Lambda}_\psi$ and $\mathbf{\Psi}_\phi$ are both MLPs with $[128, 128]$ and $[256, 256, 128]$ internal layers, respectively. The posterior master equation (Eq. 4) is solved with the adaptive step Dormand–Prince method. The Kullback-Leibler divergence in Eq. 5 is approximated via Gaussian quadrature with 200 points. Next, and regarding the generative prior, we set the dimension p of the random input vector $\boldsymbol{\varepsilon}$ to 64, and model $\mathbf{\Phi}_\theta$ with a single hidden layer of 64 units. The emission model, when used, is set to a Gaussian model, whose mean and variance are modelled with and MLP of $[128, 128]$ layers.

Finally, all parameters are optimized using Adam (Kingma & Ba, 2015), with learning rates varying from 1×10^{-3} to 1×10^{-4}, and possible learning rate annealing. We provide additional details on datasets, architectures and training in Appendix H. We also report on training times and resource consumption of NeuralMJP in Appendix I.

Table 2. Stationary distribution for the switching ion channel process. We report the distributions inferred by NeuralMJP when trained on both the one-second window (1 SEC) and the complete dataset (FULL).

	BOTTOM	MIDDLE	TOP
KÖHS ET AL. (2021)	0.17961	0.14987	0.67052
NEURALMJP (1 SEC)	0.17672	0.09472	0.72856
NEURALMJP (FULL)	0.19631	0.07153	0.73217

5.1. Discrete Flashing Ratchet

Before testing our methodology on real-world inference problems, we evaluate the ability of the model to infer ground-truth MJPs in a controlled experiment. To this end, we consider the Discrete Flashing Ratchet (DFR) process, introduced by Ajdari & Prost (1992).

The DFR process is a six-state MJP, which models the position of a particle that is subject to a periodic, asymmetric potential, and is in contact with a thermal bath (here, of unit temperature). The particle has three available states and we denote them with $i \in \{0, 1, 2\}$. The potential has two states, and we denote them with ON or OFF. The transition rates of the DFR process are defined as follows (Roldán & Parrondo, 2010)

$$\begin{aligned} f(j, \text{ON} \mid i, \text{ON}) &= \exp\left(-\frac{V}{2}(j-i)\right), \quad \text{for } i \neq j, \\ f(j, \text{OFF} \mid i, \text{OFF}) &= b, \quad \text{for } i \neq j, \\ f(i, \text{OFF} \mid i, \text{ON}) &= f(i, \text{ON} \mid i, \text{OFF}) = r, \end{aligned}$$

where V, r and b are parameters of the model. We set them all to 1 in what follows and simulate a set of trajectories from the DFR process thus defined.

With such a simple process at hand we set out to investigate how NeuralMJP handles observations recorded at non-equidistant times. To do this, we sub-sample three different datasets from the set of simulated DFR trajectories: (I) IRREGULAR GRID, in which each realization (i.e. each trajectory) has a different, irregular grid; (II) SHARED GRID, in which all realizations share a fixed, irregular grid; and (III) REGULAR GRID, in which all realizations have a fixed and regular grid. We do not corrupt these datasets with any noise model. Accordingly, we trained a NeuralMJP without emission model and use a cross-entropy loss as reconstruction cost. We also only allow transition rates in the prior MJP model which are consistent with the ground truth. The posterior is however left unconstrained. Additional details about the data simulation process, data preprocessing, hyperparameters and training procedure can be found in Appendix J.

Table 3. Inference of (ground-truth) Lotka-Volterra process against the baselines. Mean and standard deviation values are computed with 1000 samples of the generative prior MJP model. All values are of the order of 10^{-4}.

	α	β	δ	γ
GROUND TRUTH	5	1	1	5
OPPER & SANGUINETTI (2007)	13.5	2.32	1.78	15.7
NEURALMJP	4.5 ± 0.3	1.00 ± 0.08	0.76 ± 0.08	3.5 ± 0.2

Results. Table 1 shows the values inferred by NeuralMJP from our three artificial datasets. Specifically, we report the empirical mean and standard deviation computed from 1000 samples of the generative prior MJP. Note how all results are remarkably close to (and sharply peak around) the ground-truth values, which indicates that NeuralMJP can successfully infer MJP from sequential data. Interestingly, the model performs (in average) best on the IRREGULAR GRID, arguably because the dataset contains more information about the underlying process. All in all we conclude that NeuralMJP handles irregularly sampled data well.

Figure 2 shows a sample trajectory from the DFR process, together with a NeuralMJP trajectory. The white region corresponds to the observation window and displays greedy samples from the posterior model. The gray region corresponds to future (unseen) data and shows predictions of the (prior) MJP model. The posterior process clearly models the data well. The prediction process captures correctly some of the jumps too.

For completeness, we have also trained 4 additional models, with different initializations, on all three datasets. We report the mean and std. of our results in Table 8 of Appendix J. Our conclusions remain unaltered.

5.2. Switching Ion Channel Data

In this section we study the conformational switching of the viral potassium channel $\text{Kcv}_{\text{MT325}}$, which is known to switch between three different configurations (Gazzarrini et al., 2006). Markov models are a natural choice to describe the switching processes of ion channels (Rauh et al., 2017; 2018) and our methodology provides a scalable alternative to directly infer both, long- and short-term dynamic properties of the switching process, without the need to resort to frequentist approaches or introduce artificial lag time scales.

Let us assume that underlying the empirical switching process is a three-state MJP. We denote its states as TOP, MIDDLE and BOTTOM. Each state is characterized by its ion permeability, which can be experimentally measured by applying a voltage drop across the cell membrane and recording the ion flow. Here we analyse a current signal which was generated by applying a voltage drop of 140 mV, and recorded at a frequency of 5 kHz over a period of about 34 seconds. The dataset was made available to us via private communication (see the acknowledgements below). Figure 3 illustrates a snapshot of the current signal. One can visually identify the TOP and BOTTOM states, with small, noisy oscillations about them.

We train NeuralMJP with a Gaussian emission model on (i) a one-second window of the data (5000 observations) and (ii) the complete dataset (without the first burn-in second, which amounts to about 167000 observations). See Appendix K for details. We compare against the variational, switching diffusion model of Köhs et al. (2021). Note that they trained their model on the one-second window of the data (i.e. the same window we consider).

Results. We report in Table 2 the stationary distribution (of the hidden process) inferred by each model. For NeuralMJP, this is done by sampling the prior rate matrix 1000 times, computing its corresponding stationary distribution and averaging. The results agree well. Note in particular that the MIDDLE state is the least likely, according to both models. Very importantly, the distributions inferred by NeuralMJP in both datasets agree well too.

As regards the short-term dynamics, we report in Table 9 of Appendix K the estimated mean-first passage times. NeuralMJP infers shorter transition times to and from the MIDDLE state, as compared to the baseline, which implies that NeuralMJP resolves transitions between TOP and BOTTOM states via fast transitions to the MIDDLE state. This feature is consistent in both short and long datasets. In contrast, the model of Köhs et al. (2021) resolves TOP $\Leftrightarrow$ BOTTOM transitions that do not pass through the MIDDLE state.

Figure 3 shows the NeuralMJP reconstruction of a test trajectory, both within and without the observation window. The model clearly fits the data well. Figure 6 in the Appendix additionally compares the prediction capabilities of NeuralMJP against the model of Köhs et al. (2021). The performance of both models is similar.

5.3. A Birth-Death Process: Lotka-Volterra

In this section we leverage NeuralMJP to infer coupled MJP. Let us consider the Lotka-Volterra process (LV), the stereo-

typical Birth-Death Process, as our ground-truth, coupled MJP. The LV process takes values on $\mathbb{N} \times \mathbb{N}$ and models the population levels of prey and predator species as they interact over time. The process is defined in terms of four positive parameters (α, β, γ and δ), which control the strength of the species interaction, as well as the birth and death rate of prey and predator, respectively. In our experiments, we fix the tuple (α, β, γ, δ) to $(5, 1, 1, 5) \times 10^{-4}$ and simulate a set of trajectories from the LV process. The task is to infer these four parameters from a dataset of noisy observations on the LV process. See Appendix L for details.

Results. To handle the coupling in the LV process we follow Opper & Sanguinetti (2007) and use their mean-field approximation. Table 3 contains the inferred parameters of the LV process against the point estimates reported by Opper & Sanguinetti (2007). Our inferred parameters are clearly closer to (and sharply peak around) the ground-truth values as compared to the baseline. Nevertheless, let us remark that the latter was trained on a significantly smaller dataset. For the sake of a fairer comparison, we trained NeuralMJP on the same small dataset used by the baseline. Appendix L contains our results, which show our model outperforms the baseline in this case too. These findings indicate our model can successfully implement the mean-field trick and infer coupled MJP from noisy data.

Figures 7 and 8 in the Appendix show the NeuralMJP reconstructions of LV trajectories, for models trained on both large and small datasets.

5.4. Alanine Dipeptide

Alanine dipeptide (ADP) is a well-studied, 22-atom molecule that serves as a benchmark system for molecular dynamics (MD) simulations and computational biology (Rossky & Karplus, 1979). When ADP is in equilibrium, its energy surface consists of a multitude of local minima. However, the structures of the protein associated to these minima depend mainly on the relative configuration of two central backbone atom sequences (Ramachandran et al., 1963). This relative configuration can be represented by two Ramachandran angles (ψ and ϕ), which are dihedral angles of the two atom sequences (see e.g. Fig. 9 in the Appendix). Projection to these angles reveal metastable conformations of the protein, each containing several local minima (Mironov et al., 2019).

Table 4. Stationary distributions for six-state Markov models of ADP. VAMPnets results were extracted from Mardt et al. (2017). The states are ordered such that the protein conformations associated to a given state are comparable in both models.

	PROBABILITY PER STATE					
	I	II	III	IV	V	VI
VAMPNETS	0.30	0.24	0.20	0.15	0.11	0.01
NEURALMJP	0.30	0.31	0.23	0.10	0.05	0.01

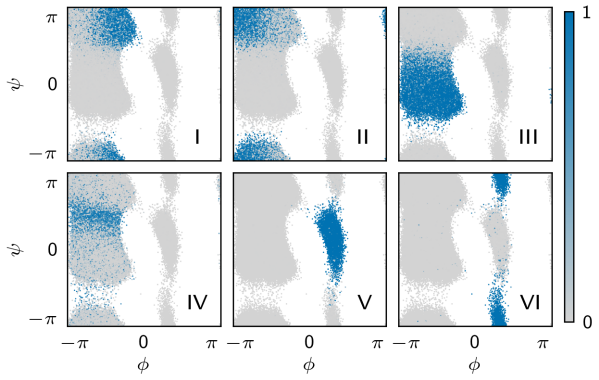

Figure 4. Alaine Dipeptide Protein six-state separation. For each observation, the intensity of blue in subplot $i = \mathrm{I}, \ldots,$ VI indicates the posterior probability mass for state i at that observation.

While the conformations themselves have been studied extensively in the past (Hermans, 2011), their dynamics are still inadequately understood. Trajectory analysis of MD simulations is one typical approach to investigate the slow dynamics, but is generally plagued with sampling problems, specially when dealing with rare events (Chekmarev et al., 2004; Trendelkamp-Schroer & Noé, 2016). In contrast, constructing discrete-time Markov models from MD simulations to describe the slow, conformational dynamics of simple molecules has proven a successful strategy in many scenarios (Husic & Pande, 2018). Yet, discrete-time models of inherently continuous-time dynamical systems necessarily introduce a finite time scale, the so-called lag time (i.e. the amount of time passing in between each step), which is an extra hyperparameter that has to be chosen by hand, and validated via secondary procedures. Such an arbitrary discretization of time is rarely interpretable (McGibbon & Pande, 2015), and the corresponding lag time has to be sufficiently long to make the process Markovian, which lowers the resolution of the model (Köhs et al., 2022).

A natural alternative is to assume that the conformational dynamics of ADP is described by a hidden (continuos-time) MJP. In this section we take this route and train NeuralMJP on an all-atom MD simulation of ADP. This simulation dataset was originally used by Wang et al. (2019) and was made available to us via private communication (see the acknowledgements below). Details about data preprocessing and training procedures are provided in Appendix M.

Let us choose a six-state NeuralMJP to model the ADP conformational dynamics. We compare our findings against

those of three models: (i) the VAMPnets model of Mardt et al. (2017), (ii) the GMVAE model of Varolgünes et al. (2019), and (iii) the estimates provided by the trajectory analysis of Trendelkamp-Schroer & Noé (2016).

Results. NeuralMJP automatically splits the Ramachandran plot into the six regions shown in Figure 4. These states can be identified with known metastable ADP conformations, and are in good agreement with the states found by both VAMPnets and GMVAE (Mardt et al., 2017; Varolgünes et al., 2019).

From Table 4 we find that the asymptotic dynamics inferred by our model, as measured by the stationary distribution, is comparable to the stationary distribution of the hidden Markov chain inferred by VAMPnets. Much of the probability mass is concentrated on states I and II, which correspond to low-energy regions of the Ramachandran plot. Also note that the stationary probabilities for states V and VI are considerably lower than those for the other states. Transitions to these states represent the rare events of the ADP dynamics (Trendelkamp-Schroer & Noé, 2016). Table 5 reports the relaxation time scales characteristic of the ADP as inferred by all models. Ours identifies time scales of three different orders of magnitude. The fast time scales inferred by us agree well with the fast time scales found by the baselines. The longest relaxation time is however short, as compared to what is extracted by the other models.

A similar pattern emerges with our mean first-passage time estimates. These are gathered in Table 6. Our model predicts more frequent transitions to the rare states V and VI, i.e. it predicts faster first-passage times, as compared to e.g. the 60 ns estimate reported in Trendelkamp-Schroer & Noé (2016). And yet for the more recurrent I $\Leftrightarrow$ IV transitions the estimates of both approaches are quite similar. These observations seem to indicate that NeuralMJP is biased towards fitting faster time scales (see also Section 5.2 above for similar findings).

All in all, our results show that NeuralMJP can correctly describe the conformational dynamics of simple molecules, without the need to introduce any artificial lag time scale. We report additional results (as e.g. experiments with a different number of hidden states, error bar estimations and further analysis of the model) in Appendix M.

6. Conclusions

In this work we introduced a novel, neural-based variational inference algorithm for MJP. The model leverages NVI and NeuralODEs to encode, in an end-to-end fashion, noisy time series data into the parameters defining the master equation of the posterior MJP. We empirically demonstrated that the model was able to successfully infer hidden MJP from synthetic, experimental and MD simulation data.

Table 5. Relaxation time scales for six-state Markov models of ADP. The time scales are ordered by size and reported in nanoseconds. VAMPnet results are taken from Mardt et al. (2017), GMVAE from Varolgünes et al. (2019) and MSM from Trendelkamp-Schroer & Noé (2016).

	RELAXATION TIME SCALES (IN ns)				
VAMPNETS	0.008	0.009	0.055	0.065	1.920
GMVAE	0.003	0.003	0.033	0.065	1.430
MSM	-	-	-	-	1.490
NEURALMJP	0.009	0.009	0.043	0.069	0.774

Table 6. Mean first-passage times for six-state Markov models of ADP. The row and column labels are the states identified in Figure 4. The entry of row i and column j represents the mean first-passage time for transitions $i \to j$ reported in nanoseconds.

τ_{ij}	I	II	III	IV	V	VI
I	0.	0.028	0.290	0.107	13.727	15.864
II	0.032	0.	0.287	0.103	13.728	15.866
III	0.132	0.134	0.	0.063	13.729	15.859
IV	0.073	0.074	0.187	0.	13.706	15.844
V	0.917	0.913	1.055	0.932	0.	3.959
VI	0.804	0.800	0.952	0.824	2.563	0.

7. Limitations

The main limitation we find with NeuralMJP is its bias towards better fitting the faster time scales of the empirical data. We observed evidence of these when modelling both, the switching ion channel data and the alanine dipeptide MD simulation. We speculate that this bias might arise from the KL evaluation, which favors larger transition rates. We leave an analysis of this feature for future work.

8. Acknowledgement

This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence (LAMARR22B). We would like to thank Lukas Köhs for sharing the experimental ion channel data with us. The actual experiment was carried out by Kerri Kukovetz and Oliver Rauh, while working in the lab of Gerhard Thiel of TU Darmstadt. Similarly, we would like to thank Nick Charron and Cecilia Clementi, from the Theoretical and Computational Biophysics group of the Freie Universität Berlin, for sharing the all-atom alanine dipeptide simulation data with us. The simulation was carried out by Christoph Wehmeyer while working in the research group of Frank Noé of the Freie Universität Berlin.

References

Ajdari, A. and Prost, J. Mouvement induit par un potentiel périodique de basse symétrie: diélectrophorese pulsée. *Comptes rendus de l'Académie des sciences*, 315(13): 1635–1639, 1992.

Asmussen, S., Nerman, O., and Olsson, M. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, pp. 419–441, 1996.

Bengio, E., Bacon, P.-L., Pineau, J., and Precup, D. Conditional computation in neural networks for faster models. *ArXiv preprint*, abs/1511.06297, 2015.

Berman, A. and Plemmons, R. J. *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.

Billingsley, P. *Statistical Inference for Markov Processes*. University of Chicago Press, 1961.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Bladt, M. and Sørensen, M. Statistical inference for discretely observed Markov jump processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3):395–410, 2005.

Boys, R. J., Wilkinson, D. J., and Kirkwood, T. B. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18(2):125–135, 2008.

Chekmarev, D., Ishida, T., and Levy, R. M. Long-time conformational transitions of alanine dipeptide in aqueous solution: Continuous and discrete-state kinetic models. *Journal of Physical Chemistry B*, 108:19487–19495, 2004.

Chen, R. T. Q. torchdiffeq, 2018.

Chen, R. T. Q., Amos, B., and Nickel, M. Neural Spatio-Temporal Point Processes. In *9th International Conference on Learning Representations*. OpenReview.net, 2021.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, pp. 6572–6583, 2018.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. pp. 103–111. Association for Computational Linguistics, 2014. doi: 10.3115/v1/W14-4012.

Cohn, I. Mean Field Variational Approximations in Continuous-Time Markov Processes. 2009.

Cohn, I., El-Hay, T., Friedman, N., and Kupferman, R. Mean field variational approximation for continuous-time Bayesian networks. *The Journal of Machine Learning Research*, 11:2745–2783, 2010.

Crommelin, D. and Vanden-Eijnden, E. Fitting timeseries by continuous-time Markov chains: A quadratic programming approach. *Journal of Computational Physics*, 217 (2):782–805, 2006.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proceedings of the 22nd ACM SIGKDD*, pp. 1555–1564. ACM, 2016. doi: 10.1145/2939672.2939875.

Durrett, R. Essentials of Stochastic Processes. 1999.

Elfving, G. Zur theorie der Markoffschen ketten. *Acta Soc. Finn. A*, 2, 1937.

Fearnhead, P. and Sherlock, C. An exact Gibbs sampler for the Markov-modulated Poisson process. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(5):767–784, 2006.

Gardiner, C. W. Stochastic Methods: A Handbook for the Natural and Social Sciences. 2009.

Gazzarrini, S., Kang, M., Epimashko, S., Van Etten, J. L., Dainty, J., Thiel, G., and Moroni, A. Chlorella virus MT325 encodes water and potassium channels that interact synergistically. *Proceedings of the National Academy of Sciences*, 103(14):5355–5360, 2006.

Ghosh, A., Behl, H. S., Dupont, E., Torr, P. H. S., and Namboodiri, V. STEER : Simple Temporal Regularization For Neural ODE. In *Advances in Neural Information Processing Systems 33*, 2020.

Gillespie, D. T. Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, 81:2340–2361, 1977.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017.

Griffiths, R. B., Weng, C.-Y., and Langer, J. S. Relaxation Times for Metastable States in the Mean-Field Model of a Ferromagnet. *Phys. Rev.*, 149:301–305, 1966. doi: 10.1103/PhysRev.149.301.

Hermans, J. The amino acid dipeptide: Small but still influential after 50 years. *Proceedings of the National Academy of Sciences*, 108:3095 – 3096, 2011.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

Huggins, J. H., Narasimhan, K., Saeedi, A., and Mansinghka, V. K. JUMP-Means: Small-Variance Asymptotics for Markov Jump Processes. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 693–701. JMLR.org, 2015.

Husic, B. E. and Pande, V. S. Markov state models: From an art to a science. *Journal of the American Chemical Society*, 140(7):2386–2396, 2018.

Huszár, F. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations*. OpenReview.net, 2017.

Jia, J. and Benson, A. R. Neural Jump Stochastic Differential Equations. In *Advances in Neural Information Processing Systems 32*, pp. 9843–9854, 2019.

Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., and Shazeer, N. Fast Decoding in Sequence Models Using Discrete Latent Variables. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2395–2404. PMLR, 2018.

Kidger, P. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Kimura, M. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16 (2):111–120, 1980.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

Köhs, L., Alt, B., and Koeppl, H. Variational Inference for Continuous-Time Switching Dynamical Systems. In *Advances in Neural Information Processing Systems 34*, pp. 20545–20557, 2021.

Köhs, L., Kukovetz, K., Rauh, O., and Koeppl, H. Nonparametric Bayesian inference for meta-stable conformational dynamics. *Physical Biology*, 19, 2022.

Kullback, S. and Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1): 79–86, 1951.

Madsen, H., Spliid, H., and Thyregod, P. Markov Models in Discrete and Continuous Time for Hourly Observations of Cloud Cover. *Journal of Applied Meteorology and Climatology*, 24(7):629 – 639, 1985. doi: 10.1175/1520-0450(1985)024⟨0629:MMIDAC⟩2.0.CO;2.

Mardt, A., Pasquali, L., Wu, H., and Noé, F. VAMPnets for deep learning of molecular kinetics. *Nature Communications*, 9, 2017.

McGibbon, R. T. and Pande, V. S. Efficient maximum likelihood parameterization of continuous-time Markov processes. *The Journal of chemical physics*, 143(3):034109, 2015.

Mei, H. and Eisner, J. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *Advances in Neural Information Processing Systems 30*, pp. 6754–6764, 2017.

Metzner, P., Horenko, I., and Schütte, C. Generator estimation of Markov jump processes based on incomplete observations nonequidistant in time. *Phys. Rev. E*, 76: 066702, 2007.

Mironov, V., Alexeev, Y., Mulligan, V. K., and Fedorov, D. G. A systematic study of minima in alanine dipeptide. *Journal of Computational Chemistry*, 40(2):297–309, 2019.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Ojeda, C., Cvejoski, K., Georgiev, B., Bauckhage, C., Schuecker, J., and Sánchez, R. J. Learning Deep Generative Models for Queuing Systems. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pp. 9214–9222. AAAI Press, 2021a.

Ojeda, C., Georgiev, B., Cvejoski, K., Schucker, J., Bauckhage, C., and Sánchez, R. J. Switching dynamical systems with deep neural networks. In *25th International Conference on Pattern Recognition (ICPR)*, pp. 6305–6312. IEEE, 2021b.

Opper, M. and Sanguinetti, G. Variational inference for Markov jump processes. In *Advances in Neural Information Processing Systems 20*, pp. 1105–1112. Curran Associates, Inc., 2007.

Ramachandran, G. N., Ramakrishnan, C., and Sasisekharan, V. Stereochemistry of polypeptide chain configurations. *Journal of molecular biology*, 7:95–9, 1963.

Rao, V. and Teg, Y. W. Fast MCMC Sampling for Markov Jump Processes and Extensions. *Journal of Machine Learning Research*, 14(11), 2013.

Rauh, O., Urban, M., Henkes, L. M., Winterstein, T., Greiner, T., Van Etten, J. L., Moroni, A., Kast, S. M., Thiel, G., and Schroeder, I. Identification of intrahelical bifurcated h-bonds as a new type of gate in k+ channels. *Journal of the American Chemical Society*, 139(22):7494–7503, 2017.

Rauh, O., Hansen, U., Scheub, D., Thiel, G., and Schroeder, I. Site-specific ion occupation in the selectivity filter causes voltage-dependent gating in a viral k+ channel. *Scientific Reports*, 8(1):10406, 2018.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286. JMLR.org, 2014.

Roldán, É. and Parrondo, J. M. R. Estimating dissipation from single stationary trajectories. *Physical review letters*, 105 15:150607, 2010.

Rossky, P. J. and Karplus, M. Solvation. a molecular dynamics study of a dipeptide in water. *Journal of the American Chemical Society*, 101(8):1913–1937, 1979.

Rubanova, Y., Chen, T. Q., and Duvenaud, D. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems 32*, pp. 5321–5331, 2019.

Schwab, P., Miladinovic, D., and Karlen, W. Granger-Causal Attentive Mixtures of Experts: Learning Important Features with Neural Networks. In *Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 4846–4853. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014846.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *5th International Conference on Learning Representations*. OpenReview.net, 2017.

Singer, B. and Spilerman, S. The representation of social processes by Markov models. *American journal of sociology*, 82(1):1–54, 1976.

Tolver, A. An Introduction to Markov Chains, 2016.

Trendelkamp-Schroer, B. and Noé, F. Efficient estimation of rare-event kinetics. *Physical Review X*, 6(1):011009, 2016.

Turner, C. M., Startz, R., and Nelson, C. R. A Markov model of heteroskedasticity, risk, and learning in the stock market. *Journal of Financial Economics*, 25(1):3–22, 1989. doi: https://doi.org/10.1016/0304-405X(89)90094-9.

Varolgünes, Y. B., Bereau, T., and Rudzinski, J. F. Interpretable embeddings from molecular simulations using Gaussian mixture variational autoencoders. *Machine Learning: Science and Technology*, 1, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, J., Olsson, S., Wehmeyer, C., Pérez, A., Charron, N. E., De Fabritiis, G., Noé, F., and Clementi, C. Machine learning of coarse-grained molecular dynamics force fields. *ACS central science*, 5(5):755–767, 2019.

Wildner, C. and Koeppl, H. Moment-Based Variational Inference for Markov Jump Processes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6766–6775. PMLR, 2019.

Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., and Achan, K. Self-attention with Functional Time Representation Learning. In *Advances in Neural Information Processing Systems 32*, pp. 15889–15899, 2019.

Yin, M. and Zhou, M. Semi-implicit variational inference. In *International Conference on Machine Learning*, pp. 5660–5669. PMLR, 2018.

Zhang, B., Pan, J., and Rao, V. A. Collapsed variational Bayes for Markov jump processes. In *Advances in Neural Information Processing Systems 30*, pp. 3749–3757, 2017.

A. Derivation of Master Equation

We roughly follow Gardiner (2009) for the derivation of the master equation.

Given some stochastic process $\{Z(t) : t \in \mathbb{R}^+\}$ taking values on some countable (and possibly unbounded) set $\mathcal{Z}$, the Markov property states that the conditional probability of observing any future state of the process depends entirely on its current state.

Thus the probability of observing $Z(t) = x$, given that $Z(t') = y$ and $Z(t'') = z$, with $t \geq t' \geq t''$, is given by

$$p(x, t|y, t'; z, t'') = p(x, t|y, t').$$

The **instantaneous probability rate** of transitioning from state $z' \in \mathcal{Z}$ to any other state $z \neq z' \in \mathcal{Z}$ at $t \in \mathbb{R}^+$ is defined as

$$f(z \mid z', t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} p(z, t + \Delta t|z', t). \tag{6}$$

The **forward master equation** describes how the conditional probabilities of Z evolve with time. This evolution is completely described by the instantaneous probability rates:

$$\begin{aligned}
\partial_t p(x, t) &= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left[p(x, t + \Delta t) - p(x, t)\right] \\
&= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left[\sum_z p(x, t + \Delta t|z, t)p(z, t) - p(x, t)\right] \\
&= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left[\sum_{z \neq x} p(x, t + \Delta t|z, t)p(z, t) - \left(1 - p(x, t + \Delta t|x, t)\right) p(x, t)\right] \\
&= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left[\sum_{z \neq x} p(x, t + \Delta t|z, t)p(z, t) - \left(\sum_{z \neq x} p(z, t + \Delta t|x, t)\right) p(x, t)\right] \\
&= \sum_{z \neq x} \left[f(x|z, t)p(z, t) - f(z|x, t)p(x, t)\right]
\end{aligned} \tag{7}$$

Note that we used the law of total probability in the second and fourth line above.

Moreover, if $\mathcal{Z}$ is finite, the master equation 7 can be written in matrix form. Let $\mathbf{p}(t) = (p(x, t))_{x \in \mathcal{Z}}$ be the vector containing the time-dependent probabilities over the states in $\mathcal{Z}$ and $\dot{\mathbf{p}}(t)$ its time derivative. Then we can define the **rate matrix** or **$\mathcal{Q}$-matrix** $\mathbf{F}(t) \in \mathbb{R}^{|\mathcal{Z}| \times |\mathcal{Z}|}$ by

$$\mathbf{F}_{zx}(t) = \begin{cases} -\sum_{z' \neq x} f(z'|x, t) & \text{if } x = z \\ f(x|z, t), & \text{otherwise.} \end{cases}$$

and express the forward master equation by

$$\dot{\mathbf{p}}(t) = \mathbf{p}(t) \cdot \mathbf{F}(t).$$

B. On Stationary Distribution, Relaxation Times and Mean First-Passage Times of MJPs

A MJP with *time-independent* transition rates is called **homogeneous**. Let $\mathcal{Z}$ be the finite state space of a homogeneous MJP Z with transition rates f and let $\mathbf{F}$ be the associated (time-independent) rate matrix.

Stationary Distribution: A probability distribution $\mathbf{p}^*$ on $\mathcal{Z}$ is called a **stationary distribution** of Z, if $\mathbf{p}^* \cdot \mathbf{F} = \mathbf{0}$. Put differently, a stationary distribution is a left eigenvector of $\mathbf{F}$ with eigenvalue 0, that is also a probability distribution. If Z is **irreducible**, roughly meaning all states are connected, there exists a unique stationary distribution $\mathbf{p}^*$. Moreover, Z will converge to $\mathbf{p}^*$ from any initial distribution. See (Tolver, 2016) for a detailed derivation of these results.

We find that the learned prior MJPs in our experiments are irreducible and will therefore converge to their stationary distribution in the long-term. However, short-term predictions with the prior MJP still significantly depend on the observed time series.

Relaxation Time: Let $\lambda_2, \lambda_3, \ldots, \lambda_{|\mathcal{Z}|}$ be the non-zero eigenvalues of $\mathbf{F}$. The **time scales** of the MJP are defined as $|\mathrm{Re}\,\lambda_2|^{-1}, |\mathrm{Re}\,\lambda_3|^{-1}, \ldots, \left|\mathrm{Re}\,\lambda_{|\mathcal{Z}|}\right|^{-1}$. They are exponential rates of decay that determine convergence from any state of the process to its stationary distribution.

The long-term convergence behaviour is dominated by the largest time scale, which is known as the **relaxation time** of the process.

Mean First-Passage Times: Suppose $Z(0) = i \in \mathcal{Z}$, then the first-passage time to state $j \in \mathcal{Z}$ is defined as

$$T_{ij} = \inf\{t \geq 0 : Z(t) = j \mid Z(0) = i\}$$

and the associated **mean first-passage time** as

$$\tau_{ij} = \mathbb{E}[T_j \mid Z(0) = i] \quad .$$

One can show that for finite state, time-homogeneous MJP, mean first-passage times are the solutions of $|\mathcal{Z}|$ systems of equations:

$$\begin{cases} \tau_{ii} = 0 \\ 1 + \sum_k \mathbf{F}_{ik}\tau_{kj} = 0 \quad , j \neq i \end{cases}$$

See Section 4.4 of (Durrett, 1999) for a derivation of this result.

C. Implicit vs Explicit Parametrization of The Prior Transition Rates

During early development of NeuralMJP, we treated the entries of the prior transition matrix as trainable parameters. One of the main empirical observations we made was that, during training, their values changed very slowly, and eventually got stuck in different regions of parameter space. We provide an example of this behaviour in Figure 5, for the α parameters of the Lotka-Volterra process (see Appendix L for details).

Key here is that the location of these regions was strongly dependent on the parameters' initialization values. Assuming no prior knowledge about the scale of the entries of the (prior) transition matrix, as is the case in both the experimental ion channel data and the molecular dynamics simulation, one cannot but choose the initialization values randomly, which led to inconsistencies in the inferred rates.

We speculate that the reason behind the slow dynamics of the prior parameters was the magnitude of the gradient signals, which directly modified their values. Opposite to the simple trainable parameters, a generator network can leverage weak gradient signals to produce large changes in its output values, as to better fit the posterior rates. We found in practice that this was indeed the case.

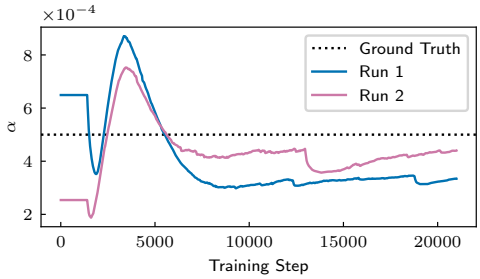

Figure 5. Evolution of the Lotka-Volterra trainable parameter α during training of NeuralMJP for two random initializations. Given that the prior parameters are learned separately from the rest (see Algorithm 1), α is learned only later in training and thus constant at the beginning. The gap in α of the two runs is not consistently closed and remains significant throughout training.

D. Derivation of Variational Bound

In this section we derive Eq. 5, a variational lower bound on the logarithm of the marginal likelihood of the data and the objective of NeuralMJP. This bound was already used by Opper & Sanguinetti (2007) for their expectation-maximization approach. See (Cohn, 2009) for a similar derivation of this bound.

We drop the dependence on parameters ψ, ϕ from the subscripts and the dependency of posterior transition rates g on the observations for a less cluttered derivation.

Consider observations $\mathbf{x}_0, \ldots, \mathbf{x}_N$ recorded at observation times $0 \leq \tilde{t}_0 < \cdots < \tilde{t}_N \leq T$. Let $t_{0:K}$ be a discretization $0 = t_0 < t_1 < \cdots < t_K = T$ of the interval $[0, T]$ such that $\Delta t_k = t_{k+1} - t_k$ is small and bounded from above by some $\overline{\Delta t} > 0$. Moreover, assume that, without loss of generality, observation times $\tilde{t}_i$ are part of the discretization, so there exists a map $m : \{0, \ldots, N\} \longrightarrow \{0, \ldots, K\}$ such that $\tilde{t}_i = t_{m(i)}$. Let $z_{0:K}$ denote a trajectory of latent states recorded at the discretization grid and $\sum_{z_{0:K}}$ the sum over all such trajectories.

By the assumptions on the generative model and Jensen's inequality, we have

$$\begin{aligned}\log p(\mathbf{x}_0,\ \mathbf{x}_1, \ldots, \mathbf{x}_N) &= \log \left(\sum_{z_{0:K}} \frac{q(z_{0:K}, t_{0:K}) p(z_{0:K}, t_{0:K}) p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N \mid z_{0:K}, t_{0:K})}{q(z_{0:K}, t_{0:K})} \right) \\ &\geq \sum_{i=0}^{N} \mathbb{E}_{q(z, t_{m(i)})}[\log p(\mathbf{x}_i \mid z_{m(i)})] - \mathrm{KL}_{\text{discr}}[q(z_{0:K}, t_{0:K}) \| p(z_{0:K}, t_{0:K})] \end{aligned} \tag{8}$$

with

$$\mathrm{KL}_{\text{discr}}[q(z_{0:K}, t_{0:K}) \| p(z_{0:K}, t_{0:K})] = \sum_{z_{0:K}} q(z_{0:K}, t_{0:K}) \log \frac{q(z_{0:K}, t_{0:K})}{p(z_{0:K}, t_{0:K})} \quad .$$

Note that the first summand of Eq. 8, referred to as the reconstruction cost, already appears in Eq. 5, using the notation $z(\tilde{t}_i) = z_{m(i)}$.

We now focus on the second summand. Because of the Markov property, we can factorize prior and posterior probabilities in $\mathrm{KL}_{\text{discr}}[q(z_{0:K}, t_{0:K}) \| p(z_{0:K}, t_{0:K})]$. After rearranging some terms, each summand is independent of future trajectories, simplifying the expression.

$$\begin{aligned} &\mathrm{KL}_{\text{discr}}[q(z_{0:K}, t_{0:K}) \| p(z_{0:K}, t_{0:K})] \\ &= \sum_{z_{0:K}} q(z_0, t_0) q(z_1, t_1 \mid z_0, t_0) \ldots q(z_K, t_K \mid z_{K-1}, t_{K-1}) \log \frac{q(z_0, t_0) q(z_1, t_1 \mid z_0, t_0) \ldots q(z_K, t_K \mid z_{K-1}, t_{K-1})}{p(z_0, t_0) p(z_1, t_1 \mid z_0, t_0) \ldots p(z_K, t_K \mid z_{K-1}, t_{K-1})} \\ &= \sum_{z_0} q(z_0) \log \frac{p(z_0)}{q(z_0)} + \sum_{k=0}^{K-1} \sum_{z_k} q(z_k, t_k) \sum_{z_{k+1}} q(z_{k+1}, t_{k+1} \mid z_k, t_k) \log \frac{q(z_{k+1}, t_{k+1} \mid z_k, t_k)}{p(z_{k+1}, t_{k+1} \mid z_k, t_k)} \\ &= \sum_{z_0} q(z_0) \log \frac{p(z_0)}{q(z_0)} + \sum_{k=0}^{K-1} \Delta t_k \sum_{z} \sum_{z'} \frac{q(z, t_k) q(z', t_{k+1} \mid z, t_k)}{\Delta t_k} \log \frac{q(z', t_{k+1} \mid z, t_k)}{p(z', t_{k+1} \mid z, t_k)} \end{aligned} \tag{9}$$

In the last equation, we changed the notation in preparation of the limit $\Delta t_k \to 0$ and the Riemann integral below.

We consider the summands of Eq. 9 individually.

Following Opper & Sanguinetti (2007), our model does not include a prior for the initial condition, hence the first summand vanishes.

For the other summand, we first consider the case of $z' \neq z$. Recall that by Eq. 6 and the time-homogeneous prior, we have

$$\begin{aligned} q(z', t_{k+1} \mid z, t_k) &= g(z' \mid z, t_k) \Delta t_k + o(\Delta t_k) \\ p(z', t_{k+1} \mid z, t_k) &= f(z' \mid z) \Delta t_k + o(\Delta t_k) \end{aligned}$$

for small Δt_k. Therefore:

$$\begin{aligned}
&\frac{q(z,t_k)q(z',t_{k+1} \mid z,t_k)}{\Delta t_k} \log \frac{q(z',t_{k+1} \mid z,t_k)}{p(z',t_{k+1} \mid z,t_k)} \\
=&q(z,t_k)\left[g(z' \mid z,t_k) + \frac{o(\Delta t_k)}{\Delta t_k}\right] \log \frac{g(z' \mid z,t_k) + \frac{o(\Delta t_k)}{\Delta t_k}}{f(z' \mid z) + \frac{o(\Delta t_k)}{\Delta t_k}} \\
&\overset{\Delta t_k \to 0}{\longrightarrow} q(z,t_k) g(z' \mid z,t_k) \log \frac{g(z' \mid z,t_k)}{f(z' \mid z)}
\end{aligned} \tag{10}$$

We now consider the case of $z' = z$, where we use the approximation $\log(1+y) = y + o(y)$ for small y. By the law of total probability we can express $q(z, t_{k+1} \mid z, t_k)$ as

$$\begin{aligned}
q(z,t_{k+1} \mid z,t_k) &= 1 - \sum_{z' \neq z} q(z',t_{k+1} \mid z,t_k) \\
&= 1 - \Delta t_k \sum_{z' \neq z} g(z' \mid z,t_k) - o(\Delta t_k)
\end{aligned}$$

and similarly $p(z, t_{k+1} \mid z, t_k)$. Therefore:

$$\begin{aligned}
&\frac{q(z,t_k)q(z,t_{k+1} \mid z,t_k)}{\Delta t_k} \log \frac{q(z,t_{k+1} \mid z,t_k)}{p(z,t_{k+1} \mid z,t_k)} \\
=&q(z,t_k)\left[1 - \Delta t_k \sum_{z' \neq z} g(z' \mid z,t_k) - o(\Delta t_k)\right]\left[-\sum_{z' \neq z} g(z' \mid z,t) + \sum_{z' \neq z} f(z' \mid z) + \frac{o(\Delta t_k)}{\Delta t_k}\right] \\
&\overset{\Delta t_k \to 0}{\longrightarrow} q(z,t_k)\left[\sum_{z' \neq z} f(z' \mid z) - g(z' \mid z,t_k)\right]
\end{aligned} \tag{11}$$

Finally, taking the limit $\overline{\Delta t} \to 0$ and combining Eq. 8 with 10 and 11 yields the variational bound

$$\begin{aligned}
&\log p(\mathbf{x}_0,\ \mathbf{x}_1, \ldots, \mathbf{x}_N) \geq \\
&\sum_{i=0}^{N} \mathbb{E}_{q(z,\tilde{t}_i)}[\log p(\mathbf{x}_i \mid z(\tilde{t}_i))] - \int_0^T dt \mathbb{E}_{q(z,t)} \sum_{z' \neq z} \left\{ f(z' \mid z) - g(z' \mid z,t) + g(z' \mid z,t) \log \frac{g(z' \mid z,t)}{f(z' \mid z)} \right\}.
\end{aligned}$$

E. Two-Step Optimization Scheme

We describe the training scheme for our implementation of NeuralMJP in Algorithm 1. Notably, we employ a two-step update approach that trains the encoder-decoder pair and prior parameters separately.

In principle, one can train the encoder-decoder pair together with the prior parameters in a single back-propagation step. However, we have found empirically that the KL regularization had a smoothing effect on the inferred, time-dependent (i.e. inhomogeneous) posterior rates. This effect generally led the model to underestimate the time-independent (i.e. homogeneous) transition rates in the ground-truth experiments and, in some cases, it even led to mode collapse.

By construction, the posterior process is restricted to be an inhomogeneous MJP, even without the KL regularization. Simultaneously updating the encoder-decoder pair while keeping the prior components fixed, and only updating the latter at a second step, allowed us to find the best inhomogeneous MJPs fitting the data. Thus the main purpose of the isolated KL term is to drive the prior towards homogeneous MJPs that best fit the optimal, inhomogeneous posterior process.

In practice we found this two-step update approach to yield excellent results with respect to the ground-truth processes, as compared to the baselines.

F. Training Instabilities And Index Collapse

In experiments with gaussian emission models, NeuralMJP can exhibit behaviour comparable to index collapse (Bengio et al., 2015; Shazeer et al., 2017; Kaiser et al., 2018; Schwab et al., 2019; Ojeda et al., 2021b). The posterior MJP of such models never assigns any probability mass to some states.

This defect affects the time series reconstructions of these models, which gets partially compensated by larger learned covariances in the emission model. However, its effect on the posterior rates extends to the prior rates, which can not be compensated.

In our experiments, we chose the number of latent states intentionally, either to recover some ground-truth or to compare with the results of other papers. Therefore, we want to avoid index collapse in our experiments and use two schedulers to counteract this behaviour.

The first scheduler is directly connected to the gaussian emission models. At the start of training, we fix the covariance to a constant, diagonal covariance matrix and only learn the gaussian mean. Only after the model has arranged the latent states and their corresponding mean reconstruction, we start learning the covariance matrix.

The second scheduler is connected to the latent dynamics and is inspired by a remark from Kidger (2021). At the beginning of training, randomly initialized posterior rates do not represent the observed dynamics well. This can lead the model towards local minima, where the latent space is not suitably arranged.

Therefore, we only train on the first 10 observations of each time series for the first 3000 batches. Afterwards, the number of observations is annealed to the full time series over a period of 5000 batches. For the Alanine Dipeptide Protein dataset, we increased this period to 10000 batches.

In our experiments, the combination of these schedulers reduced the occurrence of index collapse noticeably.

G. How to Sample Markov Jump Processes Trajectories?

The algorithm for sampling trajectories of homogeneous MJPs is due to Gillespie (1977). Here we recall a derivation of the algorithm for convenience.

Let Z be a homogeneous MJP with finite state space $\mathcal{Z}$ and time-independent transition rates f. Assume Z is at state $y \in \mathcal{Z}$ at time t, so $Z(t) = y$, and $f(z \mid y) \neq 0$ for at least one $z \in \mathcal{Z}$.

To sample a trajectory of Z starting at t, it is enough to repeatedly sample a single transition, because of the Markov property. Moreover, to sample the transition of Z away from y, it is enough to sample the arrival time of the next transition and the corresponding state transition.

Let $T(y, t) \geq t$ be the time of the next transition of Z, given $Z(t) = y$. To find the distribution of $T(y, t)$, consider $Q(y, t, t + \tau) = \mathbb{P}[T(y, t) \geq t + \tau]$ for some $\tau \geq 0$.

By construction of the transition rates f (Eq. 6), we have

$$p(z, \tau + \Delta t \mid y, \tau) = f(z \mid y)\Delta t + o(\Delta t)$$

for $z \neq y$ and small Δt. Hence, the following transformations are valid:

$$\begin{aligned}
Q(y, t + \tau, t + \tau + \Delta t) &= \prod_{z \neq y} \mathbb{P}[\text{No transition from } y \text{ to } z \text{ in } [t + \tau, t + \tau + \Delta t]] \\
&= \prod_{z \neq y} (1 - p(z, t + \tau + \Delta t \mid y, t + \tau)) \\
&= \prod_{z \neq y} (1 - f(z \mid y)\Delta t + o(\Delta t)) \\
&= 1 - \sum_{z \neq y} f(z \mid y)\Delta t + o(\Delta t)
\end{aligned}$$

Therefore,

$$\begin{aligned} Q(y, t, t+\tau+\Delta t) &= Q(y, t, t+\tau) Q(y, \tau, t+\tau+\Delta t) \\ &= Q(y, t, \tau)(1 - \sum_{z \neq y} f(z \mid y)\Delta t + o(\Delta t)) \end{aligned}$$

and after some rearranging and $\Delta t \to 0$,

$$\partial_\tau Q(y, t, t+\tau) = -\sum_{z \neq y} f(z \mid y) Q(y, t, t+\tau)$$

With the assumptions from above we have $Q(y, t, t) = 1$. Solving the initial value problem yields

$$Q(y, t, t+\tau) = e^{-\sum_{z \neq y} f(z|y)\tau}$$

and hence

$$\begin{aligned} \mathbb{P}[T(y, t) = \tau] &= \partial_\tau [1 - \mathbb{P}[T(y, t) \geq \tau]] \\ &= \sum_{z \neq y} f(z \mid y) e^{-\sum_{z \neq y} f(z|y)\tau} \end{aligned}$$

So $T(y, t)$ is exponentially distributed on values $[t, \infty)$ with rate $\lambda = \sum_{z \neq y} f(z \mid y)$ and can be sampled efficiently.

Also following Eq. 6, the probability of transitioning from y to x must be

$$\frac{f(x \mid y)}{\sum_{z \neq y} f(z \mid y)} \quad . \tag{12}$$

Thus, sampling one transition of a MJP at state y at time t requires sampling a exponentially distributed waiting time from Exp(λ) and distribution 12 over $\mathcal{Z} \setminus \{y\}$.

Note that this algorithm also applies to MJPs with countable state space $\mathcal{Z}$, if for every $z \in \mathcal{Z}$ only finitely many transition rates $f(y \mid z)$ are non-zero, like in the Lotka-Volterra process (see Appendix L for details).

Furthermore, this sampling algorithm implies that MJP trajectories are right-continuous and piecewise-constant.

H. Reproducibility and Experiments

H.1. Dataloading

Because of data generation and pre-processing, all time series in train, test and validation sets of a given dataset are of the same length. Each observation is comprised of a observation time and a observation value. We enrich them by the time difference to the subsequent observation. For the last observation of each time series, his information is missing, so we drop it during dataloading.

H.2. Architectures and Hyperparameters

The network architectures described below are kept constant across all experiments, apart from the LV single time series dataset, where we reduced the MLPs to a single hidden layer of size 64.

We fix the dimension H of $\mathbf{h}_T$ to 256 in all experiments, apart from the LV single time series dataset, where it is set to 64.

MLPs of the ODE-RNN and the posterior process $\mathbf{\Psi}_\theta$ consist of hidden layers $[256, 256]$ and $[256, 256, 128]$ respectively, have Tanh activation functions and are initialized with zero bias and weights sampled from $\mathcal{N}(0, 0.01)$. The MLP $\mathbf{\Lambda}_\phi$ has hidden layers $[128, 128]$ with ReLU activation functions and is initialized with Kaiming initialization. If the emission model contains a MLP, it has hidden layers $[128, 128]$, ReLU activation functions and is initialized with zero bias and weights sampled from $\mathcal{N}(0, 0.01)$.

The MLP for the generative prior Φ_θ consist of one layer of width 64 with ReLU activation function and is initialized with Kaiming initialization. Its input is a 64-dimensional sample from $\mathcal{N}(0, 0.01)$.

We use layer normalization and dropout of 0.2 for all MLPs.

H.3. NeuralODEs: Additional Details

We use the torchdiffeq python package (Chen, 2018) to solve all NeuralODEs in our model. The ODE-RNN is solved with the Runge-Kutta method. The posterior master equation is solved with the adaptive step Dormand-Prince method and tolerances of 0.001. For faster training on the ADP dataset, we set the tolerances to 0.01 for these experiments.

To handle batches of irregular time series, we apply the technique described in Appendix F of (Chen et al., 2021).

We use STEER from (Ghosh et al., 2020) to regularize the ODE-RNN.

H.4. On The Variational Posterior Process

The posterior transition rates $\mathbf{g}_\phi$ are time-dependent. We use Mercer a embedding (Xu et al., 2019) of t with 10 frequencies and 20 Fourier coefficients as the time component of the input of $\mathbf{\Psi}_\phi$, the MLP for $\mathbf{g}_\phi$.

Note that the Mercer embedding is not an integral part of our model. We found anecdotally that their addition leads to marginal improvements in the learned dynamics, but do not change the results fundamentally.

The marginal posterior distribution at observation times are sampled with the Gumbel-Softmax estimator from (Jang et al., 2017) with a sampling temperature of 1.

H.5. Numerical Approximation of the Kullback-Leibler Divergence

We approximate the integration of the Kullback-Leibler divergence in Eq. 5 by Gaussian quadrature with 200 points.

Alternatively, it could be approximated by augmenting the dynamics of the posterior master equation with the integrand of the Kullback-Leibler divergence. In practice, we found this method to be much slower without providing noticeable benefits.

Finally, we tried approximating the KL with Tanh-Sinh quadrature. Its main advantage over Gaussian quadrature is the progressive use of integrand evaluations when lowering the stepsize. We can not take advantage of this property, because any additional integrand evaluation requires solving the posterior master equation, which is computationally expensive.

H.6. Optimizers

We use the Adam optimizer and apply gradient clipping with global norm 1 for all experiments. However, learning rate and possible learning rate annealing depend on the specific experiments.

For the single LV time series and the Two-Mode Hybrid System we use a learning rate of 0.001.

For LV, DFR and ADP data we use a learning rate of 0.001, annealed by a factor of 0.8 every 50 epochs.

For Simple Protein Folding Model and Switching Ion Channel Data we use a learning rate of 0.0001, with an annealing of 0.8 every 100 and 200 epochs respectively.

In experiments with the Switching Ion Channel Data we found that the generative prior does not converge fast enough with this annealing schedule. Hence, we use a separate optimizer only for the corresponding MLP with constant learning rate 0.0005.

H.7. Metrics

To compare RMSE between test and prediction sets, we use the following RMSE for multivariate time series.

Assume there are N time series of length T in D dimensional space. Let x_{ijk} be the observed k-th dimension of observation j of the i-th time series. Let $\hat{x}_{ijk}$ be the prediction of x_{ijk}. Then we define:

$$RMSE = \sqrt{\frac{1}{NT}\sum_{i=1}^{N}\sum_{j=1}^{T}\sum_{k=1}^{D}(\hat{x}_{ijk} - x_{ijk})^2}$$

Next, we consider the RMSE of the prediction at a specific step j of all N time series:

$$RMSE_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{D}(\hat{x}_{ijk} - x_{ijk})^2}$$

Note that this is the above formula with $T = 0$.

To evaluate the capabilities on the first m steps we compute the mean RMSE of the first m predictions:

$$\frac{1}{m}\sum_{j=1}^{m} RMSE_j$$

I. Training times and resource consumption

We summarize the training times for all experiments in Table 7. Some models only only after several days of training, despite loose tolerances. Presumably, more modern NeuralODE implementations like diffrax (Kidger, 2021) speed up training of our model significantly. However, note that these training times are on the same scale as those of other NeuralODE based models.

Each experiment was performed on a single NVIDIA GeForce GTX 1080 Ti.

I.1. Computational Complexity

The structure of NeuralMJP, a combination of a backward in time ODE-RNN encoding and a forward in time NeuralODE solve, is the structure of LatentODE (Rubanova et al., 2019). Thus, the computational complexity of NeuralMJP and LatentODE is equivalent. Like the ODE-Net (Chen et al., 2018), it scales linearly with the number of function evaluations of the ODE solvers in the forward pass of the model.

In our implementation of NeuralMJP, the master equation is solved with an adaptive step solver. The number of steps it takes depends on the complexity of the underlying dynamics.

Table 7. Training times for selected runs of all NeuralMJP experiments, including the number of datapoints in their training data.

Data	Experiment	Number of datapoints $\times 10^3$	Training time (hours)
DFR	Irregular grid	224	41.3
	Shared grid	224	22.5
	Regular grid	224	31.2
	Single Time Series	0.016	36.61
Ion Channel	One-Second Window	5	21.5
	Full Dataset	162.6	22.0
LV	Irregular grid	224	33.1
	Shared grid	224	19.5
	Regular grid	224	19.0
ADP	Six States	896	48.1
	Three States	896	20.6
	Two States	896	19.4
Simple Protein Folding Model		89.6	26.8
Two-Mode Hybrid System		0.067	26.3

J. Discrete Flashing Ratchet

J.1. Data Description and Pre-Processing

The Discrete Flashing Ratchet (DFR) process (Roldán & Parrondo, 2010) is a MJP that models the position of a particle in a termal bath of temperature T subject to a periodic, asymmetric potential. The particle can attain states $0, 1$ or 2. The potential can be ON or OFF, yielding six possible states for the MJP. The transition rates of the DFR process are (Roldán & Parrondo, 2010)

$$\begin{aligned} f(j, \text{ON} \mid i, \text{ON}) &= \exp\left(-\frac{\beta V}{2}(j-i)\right) \quad \text{for } i \neq j, \\ f(j, \text{OFF} \mid i, \text{OFF}) &= b, \quad \text{for } i \neq j, \\ f(i, \text{OFF} \mid i, \text{ON}) &= f(i, \text{ON} \mid i, \text{OFF}) = r, \end{aligned}$$

where $\beta = \frac{1}{T}$ and V, r and b are parameters of the model.

We sample trajectories of DFR with parameters $T = 1$, $V = 1$, $r = 1$ and $b = 1$ and a time window of $[0, 2.5]$. Initial states are sampled from the stationary distribution of the process:

$$\begin{aligned} p(0, \text{ON}) &= 0.3012 & p(1, \text{ON}) &= 0.1365 & p(2, \text{ON}) &= 0.0623 \\ p(0, \text{OFF}) &= 0.2003 & p(1, \text{OFF}) &= 0.1591 & p(2, \text{OFF}) &= 0.1406 \end{aligned}$$

We sample 5000 trajectories of DFR with the algorithm described in Appendix G and observe them at three observation time grids of length 50. For the IRREGULAR GRID dataset, the observation times for every time series are sampled uniformly in a given time window. For the SHARED GRID dataset, observation times are sampled uniformly in a given time window, and then used as a time series for all trajectories. Finally, for the REGULAR GRID dataset, all time series have a fixed, regular grid with equidistant observation times. We do not add noise to the observed states, treat them as categorical observations and load them as one-hot vectors.

Out of the 5000 time series, we use 70 batches of size 64 for training and leave the remaining time series for test and validation. During training, we normalize observation times to lie in $[0, 1]$.

We simulate 100 additional trajectories on a time window twice as long as above for prediction. The first half of each trajectory is treated as another test set and observed on the same grid as the training data. The second half is also observed at 50 observation times on the same type of grid. Thus, each time series in the prediction set is of length 100.

J.2. Modelling and Results

We trained NeuralMJP with a six-state, unconstrained posterior MJP, meaning transitions between all states are possible. The prior is constrained to a DFR process and parametrized as above. On the IRREGULAR GRID dataset, we additionally trained a NeuralMJP, in which the posterior MJP is constrained to transitions which are possible in the prior process.

In the absence of noise, we train these NeuralMJPs without an emission model and use cross-entropy as reconstruction loss. We summarize our results in Table 8. Constraining the posterior MJP reduces the KL loss, but the inferred parameters are slightly more inaccurate.

Table 8. DFR parameters learned by NeuralMJP on multiple datasets with different observation grids. Additionally, KL divergence and reconstruction loss on the test set. Last row contains the results on IRREGULAR GRID DATA with constrained posterior process. Values are reported with mean and standard deviation of 5 runs of NeuralMJP with different initializations.

	KL	NLL	V	r	b
GROUND TRUTH	-	-	1.00	1.00	1.00
IRREGULAR GRID DATA	20.0 ± 1.1	0.21 ± 0.01	0.98 ± 0.05	1.11 ± 0.06	1.13 ± 0.04
SHARED GRID DATA	18.7 ± 0.5	0.14 ± 0.01	0.95 ± 0.03	1.18 ± 0.04	1.14 ± 0.06
REGULAR GRID DATA	21.8 ± 0.7	0.18 ± 0.02	1.00 ± 0.05	1.37 ± 0.05	1.36 ± 0.05
MASKED POSTERIOR RATES	17.1 ± 0.4	0.22 ± 0.01	0.99 ± 0.02	1.18 ± 0.04	1.16 ± 0.03

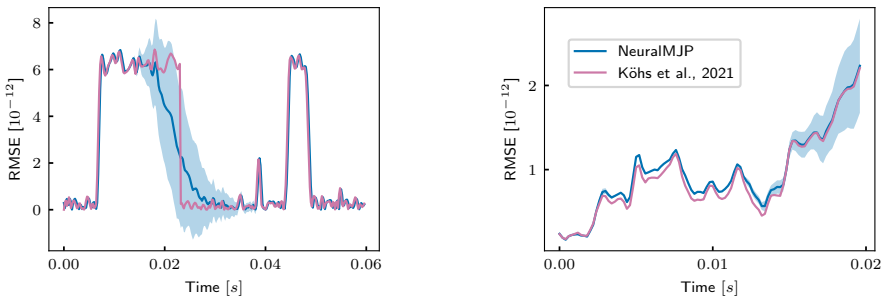

Figure 6. Comparison of prediction capabilities of models trained on Switching Ion Channel data. Left: RMSE on observations after training set. Right: 25 time series extracted from training period. Posterior from (Köhs et al., 2021) was extracted approximately from their Figure 4.

K. Switching Ion Channel Data

K.1. Data Description and Pre-Processing

The ion permeability of the vial potassium channel $\text{Kcv}_{\text{MT325}}$ is experimentally measured by applying a voltage drop across the cell membrane and recording the ion flow. The current has been measured with a frequency of 5 kHz and a voltage of 140 mV for a total of 34.323 seconds or 171615 observations.

Köhs et al. (2021) consider the one-second time window from observation 9200 to 14200. We cut this time window into 50 time series of length 100, create 5 batches of size 10 and use all of them for training. Additional test and validation time series of length 100 are taken from parts of the trajectory outside of this time window.

The observation time for the first observation in each time series is set to 0 and subsequent observation times are adjusted accordingly, retaining the measurement frequency from the original trajectory.

The time series for prediction includes observations from 14100 to 14500, where the first 100 observations are used to infer the posterior distribution at observation 14200.

We also experiment with the full dataset. After removing a burn-in time of 5000 observations from the start of the measurements, we split the trajectory into prediction, train, test and validation sets analogous to Section M.

We use 100 time series of length 200 for prediction and 10 time series of length 100 for test and validation each. This leaves 1626 time series of length 100 for training, from which we create 25 batches of size 64 and a additional batch with the remaining time series.

We normalize observation times and values to lie in $[0, 1]$ when training on both datasets.

Table 9. Mean first-passage times, expressed in seconds, of models trained on Switching Ion Channel one-second window data and the full dataset. Comparing available results from (Köhs et al., 2021) and NeuralMJP. Entry j in row i is mean first-passage time of transition $i \to j$ of the corresponding model.

	One-Second Window						Full Dataset		
	Köhs et al. (2021)			NeuralMJP			NeuralMJP		
τ_{ij}/s	Bottom	Middle	Top	Bottom	Middle	Top	Bottom	Middle	Top
Bottom	0.	0.068	0.054	0.	0.019	0.031	0.	0.008	0.014
Middle	0.133	0.	0.033	0.083	0.	0.014	0.031	0.	0.006
Top	0.181	0.092	0.	0.119	0.038	0.	0.048	0.018	0.

Table 10. Ordered relaxation time scales in seconds of models trained on Switching Ion Channel data.

	ORDERED TIME SCALES	
KÖHS ET AL. (2021)	0.01228	0.03396
NEURALMJP (1 SEC)	0.00295	0.02116
NEURALMJP (FULL)	0.00099	0.0097

K.2. Modelling and Results

We train three-state NeuralMJPs with gaussian emission models on both both datasets. They recover the three known configurations of the ion channel.

We compare our results to the results of Köhs et al. (2021) on the one-second window dataset.

The learned dynamics of both models, here represented by mean first-passage times in Table 9, differ by their resolution of transitions between TOP and BOTTOM. NeuralMJP resolves them by short transitions to the MIDDLE state, indicated by the short mean first-passage times to and from state MIDDLE. These are much longer for Köhs et al. (2021), indicating that transitions between TOP and BOTTOM do not pass through state MIDDLE.

This structural difference is also reflected in the significantly shorter learned time scales (see Table 10).

Figure 6 shows that the prediction capabilities of both models are very similar, despite their structural differences in the learned dynamics.

Moreover, Table 10 reveals that NeuralMJP learns different relaxation time scales for each dataset. It captures a data shift over the complete trajectory, where the MIDDLE configuration is only visited for very short periods and less frequently.

The fact that the relaxation time scales are shorter in the FULL dataset, as opposite to the 1 SEC dataset, reflect the fact that the time-averaged pdf describing the system of the FULL dataset is closer to the stationary distribution, than the corresponding time-averaged pdf of the 1 SEC dataset.

In other words, this result simply reflects the ergodicity of the system.

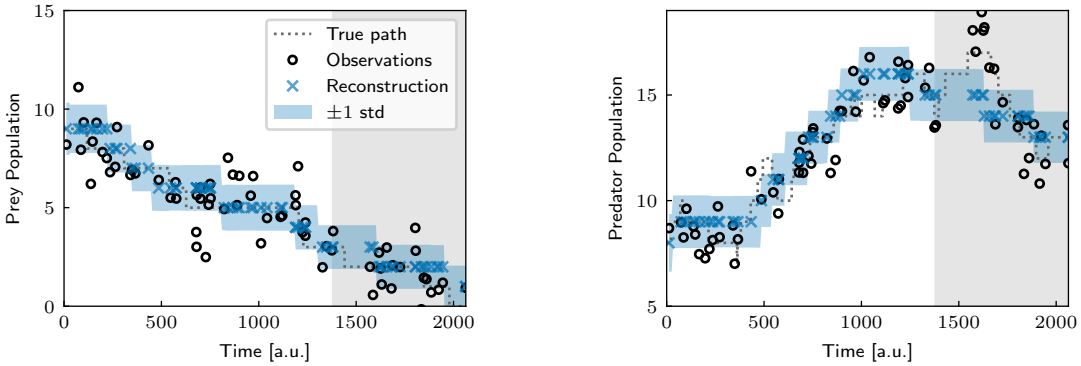

Figure 7. Lotka-Volterra trajectory from the IRREGULAR GRID dataset, with the true underlying path (black dots) and the noisy observations (black circles). The white background represents the test part of the trajectory, the grey shaded background on the right represents the prediction part. Reconstruction of NeuralMJP (blue crosses) with learned standard deviation of the emission model (blue shaded) follows the true path closely on the test and prediction parts of the trajectory.

Table 11. LV parameters learned by NeuralMJP on multiple datasets. Additionally, KL divergence and reconstruction loss on the test set. Values are reported as mean and standard deviations of 5 runs of NeuralMJP. Parameter values are reported in 10^{-4}.

	KL	NLL	α	β	δ	γ
GROUND TRUTH	-	-	5	1	1	5
OPPER & SANGUINETTI (2007)	-	-	13.5	2.32	1.78	15.7
IRREGULAR GRID	31 ± 2	166 ± 2	4.3 ± 0.6	0.94 ± 0.07	0.83 ± 0.09	4.1 ± 0.4
SHARED GRID	36 ± 4	155 ± 2	5.4 ± 0.6	1.10 ± 0.07	1.00 ± 0.06	4.9 ± 0.4
REGULAR GRID	28 ± 5	163 ± 4	4.5 ± 0.6	1.00 ± 0.03	0.90 ± 0.08	4.3 ± 0.5
SINGLE TRAJECTORY	192 ± 114	13 ± 24	2.1 ± 0.6	0.67 ± 0.06	1.05 ± 0.32	9.3 ± 8.1

L. Lotka-Volterra

L.1. Data Description and Pre-Processing

The Lotka-Volterra (LV) process is a MJP on $\mathbb{N} \times \mathbb{N}$ that models the coupled population levels of prey and predator species over time. As a pure birth-death process, only transitions to neighbouring states are possible. Moreover, it is assumed that population levels of both species can not change at the exact same time.

To describe the transition rates, let $x \in \mathbb{N}$ and $y \in \mathbb{N}$ denote the population levels of prey and predator respectively. The potentially non-zero transition rates of the Lotka-Volterra process are

$$f(x+1, y \mid x, y) = \alpha x \qquad f(x-1, y \mid x, y) = \beta xy$$
$$f(x, y+1 \mid x, y) = \delta xy \qquad f(x, y-1 \mid x, y) = \gamma y$$

where $\alpha, \beta, \delta, \gamma \in \mathbb{R}^+$ are parameters of the model.

We sample trajectories of LV with parameters $\alpha = 0.0005$, $\beta = 0.0001$, $\delta = 0.0001$ and $\gamma = 0.0005$ and a time window of $[0, 1500]$. Like Opper & Sanguinetti (2007), we add a small constant 10^{-6} to the transition rates $f(1, y \mid 0, y)$ and $f(x, 1 \mid x, 0)$, which would be zero otherwise. Initial population levels are sampled uniformly random from $[5, 20] \times [5, 20]$.

As in Appendix J, we sample 5000 LV trajectories and observe them at three observation grids of length 50. We use 70 batches of size 64 for training and leave the remaining time series for test and validation. For prediction, we generate 100 additional time series of length 100, just as in Appendix J. The observed population levels are corrupted with samples from $\mathcal{N}(0, 1)$. These observations are loosely bounded from above by 60.

L.2. Modelling and Results

We follow Opper & Sanguinetti (2007) and use a mean-field approach by training NeuralMJP with two posterior MJPs with 60 states each. Each MJP is a birth-death process and transitions out of lower and upper bounds are impossible.

The prior is constrained to a LV process and parametrized as above, but transition rates for transitions out of the boundary are set to 0. We use a gaussian emission model, where the sampled latent state are the gaussian mean and only the diagonal covariance matrix is learned. Thus, each species is linked to one posterior process, which is in accordance with (Opper & Sanguinetti, 2007).

We summarize the losses of our models on test sets and the inferred LV parameters in Table 11 and the RMSE on a prediction set in Table 12. The inferred parameters are close to the ground-truth values and consistent across multiple runs of NeuralMJP with different initializations.

In Figure 7 we see that the posterior MJP follows the underlying dynamics closely and learns a reasonable standard deviation. Moreover, prediction with the prior process captures the trend of the continued sample path.

Table 12. Reconstruction RMSE in test and prediction sets at a single observation for NeuralMJPs trained on LV datasets. For prediction, we consider the RMSE at the 1st, 5th and 10th observation in the prediction set. Values are reported as mean and standard deviation of 5 runs of NeuralMJP. Values for Opper & Sanguinetti (2007) are calculated based on their Figure 1.

	RMSE Test	RMSE Prediction at Observation 1	5	10
Opper & Sanguinetti (2007)	2.672	3.000	9.487	11.402
Irregular Grid	1.87 ± 0.04	2.67 ± 0.17	3.1 ± 0.1	3.3 ± 0.1
Shared Grid	1.67 ± 0.04	2.33 ± 0.08	2.8 ± 0.2	3.1 ± 0.2
Regular Grid	1.80 ± 0.08	1.75 ± 0.14	2.4 ± 0.2	2.7 ± 0.2
Single Trajectory	2.02 ± 0.04	5.69 ± 2.66	9.2 ± 3.6	10.6 ± 4.3

L.3. Single Lotka-Volterra Time Series

We also experiment with the single, short Lotka-Volterra time series from Figure 1 of Opper & Sanguinetti (2007).

They simulated the trajectory with parameters $\alpha, \beta, \delta, \gamma$ from above and chose initial states $x = 19$ and $y = 7$. Observations at 16 equidistant times in $[0, 1500]$ were then corrupted by a two-sided, discrete exponential noise.

They also provide the continuation of the trajectory up to time 3000. We use this continuation to create a prediction set by observing it at 15 equidistant times in $[1600, 3000]$. Note that we do not add any noise these additional observations.

We train NeuralMJP with the setup from above on this single trajectory. The results in Table 11 show that our model predicts parameters closer to the ground truth, compared to Opper & Sanguinetti (2007), although there is a lot of variance in γ. This translates to more accurate predictions, as depicted in Figure 8.

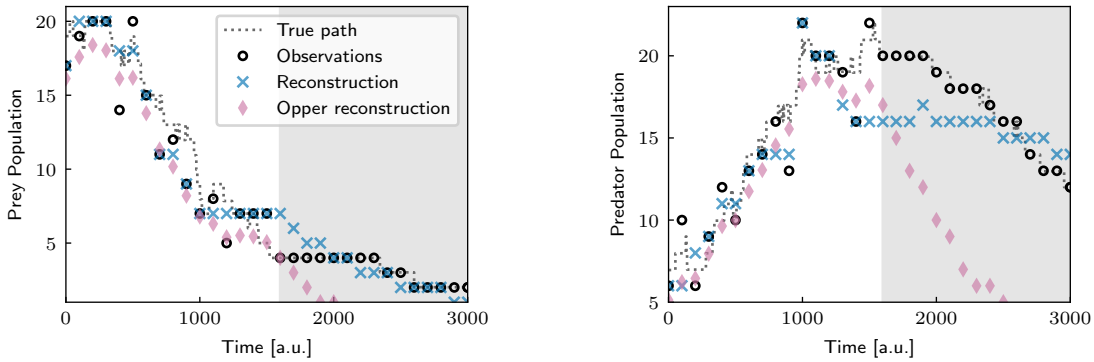

Figure 8. Lotka-Volterra single time series from (Opper & Sanguinetti, 2007). The training part of the trajectory is on the white background to the left, while the prediction part of the trajectory is on the grey background to the right. Reconstruction and prediction with NeuralMJP trained on this single trajectory (blue crosses). Comparison to the results from (Opper & Sanguinetti, 2007) (red diamonds), where their learned master equation was solved for prediction, with initial value approximated from Figure 1 (Opper & Sanguinetti, 2007).

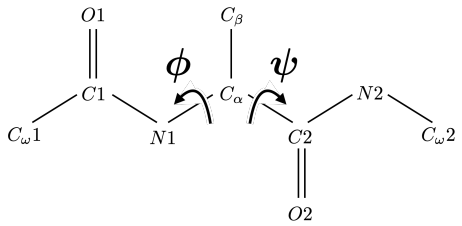

Figure 9. Heavy atoms of the Alanine Dipeptide Protein with locations of Ramachandran angles ψ and ϕ.

M. Alanine Dipeptide Protein

M.1. Data Description and Pre-Processing

The Alanine Dipeptide Protein (ADP) data used in Section 5.4 is a 10^6 step all atoms molecular dynamics simulation with stepsize 1 picosecond (ps). The system is in equilibrium and the data has been aligned to the first frame, removing global translations and rotations of the protein.

We first transform the observations into Ramachandran angles, which are dihedral angles of two backbone atom sequences. The dihedral angles for atom sequences $N1 - C_\alpha - C2 - N2$ and $C1 - N1 - C_\alpha - C2$ are denoted by ψ and ϕ respectively. Their locations in the atom are depicted in Figure 9. Partially following (Varolgünes et al., 2019), we use $\sin\psi$, $\cos\psi$, $\sin\phi$ and $\cos\phi$ as features, which take the periodicity of the angles into consideration.

For prediction, we first extract 100 non-overlapping time series of length 200 from this trajectory. Afterwards, the remaining trajectory is cut up into time series of length 100. We create 140 batches of size 64 for training and leave the other time series for test and validation.

The observation time for the first observation in each time series is set to 0 and subsequent observation times are adjusted accordingly, retaining the stepsize from the original trajectory. We normalize observation times to lie in $[0, 1]$.

Table 13. Mean first-passage times in nanoseconds of the six-state NeuralMJP trained on APD data with standard deviation from sampling the generative prior. These results are depicted in Table 6 without standard deviation. Entry j in row i is the mean first-passage time of transitions $i \to j$.

τ_{ij}/ns	I	II	III	IV	V	VI
I	0.000 ± 0.000	0.028 ± 0.002	0.290 ± 0.030	0.107 ± 0.006	14.000 ± 4.000	16.000 ± 4.000
II	0.032 ± 0.003	0.000 ± 0.000	0.290 ± 0.030	0.103 ± 0.006	14.000 ± 4.000	16.000 ± 4.000
III	0.130 ± 0.010	0.130 ± 0.010	0.000 ± 0.000	0.063 ± 0.005	14.000 ± 4.000	16.000 ± 4.000
IV	0.073 ± 0.008	0.070 ± 0.010	0.190 ± 0.020	0.000 ± 0.000	14.000 ± 4.000	16.000 ± 4.000
V	0.920 ± 0.220	0.910 ± 0.220	1.100 ± 0.240	0.900 ± 0.200	0.000 ± 0.000	4.000 ± 1.000
VI	0.800 ± 0.210	0.800 ± 0.210	0.950 ± 0.210	0.800 ± 0.200	3.000 ± 1.000	0.000 ± 0.000

Table 14. Transition rates from the six-state NeuralMJP trained on ADP data. They are expressed in nanoseconds with standard deviation from sampling the generative prior. Entry j in row i is the rate of transition $i \to j$.

$i \to j$	I	II	III	IV	V	VI
I	0.00 ± 0.00	53.00 ± 4.00	0.19 ± 0.06	7.90 ± 0.90	0.06 ± 0.03	0.02 ± 0.01
II	47.00 ± 3.00	0.00 ± 0.00	0.05 ± 0.03	12.00 ± 0.90	0.04 ± 0.02	0.01 ± 0.01
III	0.28 ± 0.09	0.13 ± 0.05	0.00 ± 0.00	17.00 ± 1.40	0.02 ± 0.01	0.04 ± 0.02
IV	36.00 ± 3.00	26.90 ± 2.70	41.00 ± 4.00	0.00 ± 0.00	0.30 ± 0.10	0.01 ± 0.01
V	0.17 ± 0.06	0.20 ± 0.10	0.40 ± 0.20	0.20 ± 0.20	0.00 ± 0.00	3.00 ± 1.00
VI	1.20 ± 0.60	1.80 ± 0.80	0.50 ± 0.20	0.70 ± 0.40	19.00 ± 7.50	0.00 ± 0.00

M.2. Modelling and Results

We train NeuralMJPs with six, three and two states and unconstrained prior and posterior MJPs. For this dataset, we chose a gaussian emission model and learn the gaussian mean and the complete covariance matrix.

The separation of states in models with two and three states is depicted in Figure 11. They resemble two slower relaxation time scales of the protein.

Table 13 contains the mean first-passage times of the six-state model presented in Section 5.4. Note that it contains the values from Table 6, with additional standard deviation from sampling the prior rates. The learned transition rates of that model are in Table 14. We remark that the results for that model were discussed thoroughly in Section 5.4.

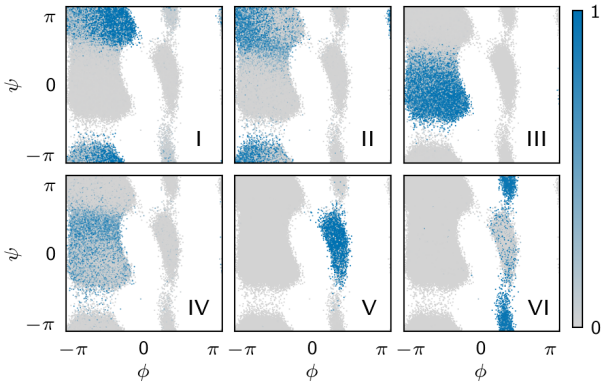

Figure 10. ADP state separation from another six-state NeuralMJP run that differs from t he one presented in Figure 4. For each observation, the intensity of blue in subplot $i = \text{I}, \ldots, \text{VI}$ indicates the posterior probability mass for state i at that observation.

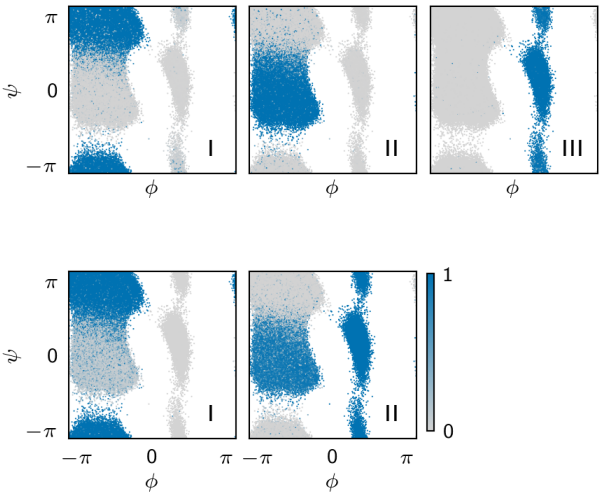

Figure 11. State separation of three and two-state NeuralMJPs trained on ADP data. For each observation, the intensity of blue in subplot $i = \text{I}, \text{II}, \text{III}$ indicates the posterior probability mass for state i at that observation.

M.3. Model Instabilities

Results across multiple runs of six-state NeuralMJPs on the ADP data are not as stable as the results of multiple runs on other experiments.

One source of instability is the failure of separating state VI from state V, which can be attributed to its rare occurrence in the data. The same failure has been reported by Mardt et al. (2017) and Varolgünes et al. (2019), who train end-to-end, unsupervised Markov models on similar ADP trajectories.

Note that this failure cannot be easily detected based on training dynamics or metrics. In Table 15, we display the results of 15 runs of NeuralMJP with six states on the ADP data. Only a few of them resolve state VI accurately. However, they are not immediately discernible from each other based on their reconstruction performance, here represented by RMSE, or learned dynamics, here represented by the learned time scales.

Another source of instability is the size, shape and position of clusters learned by our model, in particular in the $\phi < 0$ region of the Ramachandran plane. By Mironov et al. (2019), this region contains several low energy minima of the protein, which can be grouped by their proximity in the Ramachandran plane and other configuration properties.

However, unsupervised clustering of that region by NeuralMJP does not produce such consistent clusters. Compare Figures 4 and 10 for two runs of six-state NeuralMJPs. Although both runs produce comparable results on the test set, there is a noticeable difference in their clusters. In particular, the edges of clusters in Figure 10 are not as well defined as in Figure 4.

The posterior distribution depends significantly on the particular clustering, so these inconsistencies extend to differences in the learned dynamics. Therefore, comparing models by their transition rates or stationary distributions is only possible, if their clusters are comparable.

Finally, some prior rates exhibit a large sampling standard deviation, which can lead to considerable uncertainty of eigenvalues, relaxation times and mean first-passage times, even for a single run. See Table 14, where some of the smaller rates display considerable standard deviation.

Considering the above, we selected the run presented in Section 5.4 based on its separation of states in the $\phi > 0$ region of the Ramachandran plane and its distinct clusters that we can identify with clusters found, for example, by Mironov et al. (2019) and Varolgünes et al. (2019).

Table 15. RMSE and relaxation time scales of 15 NeuralMJP with six states on ADP data. Standard deviation in time scales is due to sampling the generative prior. Standard deviation in RMSE due to the 100 time series in test set. First 5 experiments were handpicked by their separation of states V and VI. Out of the other 10 non-handpicked experiments, only experiment 7 separated states V and VI. VAMPNets results are taken from Mardt et al. (2017), GMVAE from Varolgünes et al. (2019).

EXPERIMENT	RMSE	ORDERED TIME SCALES IN ns				
VAMPNETS	-	0.008	0.009	0.055	0.065	1.920
GMVAE	-	0.003	0.003	0.033	0.065	1.430
1	0.6 ± 0.1	0.008 ± 0.000	0.011 ± 0.001	0.027 ± 0.005	0.084 ± 0.005	0.800 ± 0.200
2	0.7 ± 0.2	0.021 ± 0.001	0.048 ± 0.004	0.055 ± 0.003	0.500 ± 0.100	1.000 ± 0.200
3	0.7 ± 0.1	0.012 ± 0.000	0.014 ± 0.001	0.040 ± 0.010	0.068 ± 0.004	1.500 ± 0.300
4	0.6 ± 0.1	0.009 ± 0.000	0.013 ± 0.001	0.071 ± 0.005	0.250 ± 0.030	0.410 ± 0.070
5	0.6 ± 0.1	0.009 ± 0.000	0.009 ± 0.000	0.040 ± 0.010	0.069 ± 0.004	0.800 ± 0.200
6	0.6 ± 0.1	0.006 ± 0.000	0.008 ± 0.000	0.010 ± 0.000	0.059 ± 0.001	0.700 ± 0.100
7	0.7 ± 0.1	0.012 ± 0.000	0.012 ± 0.000	0.060 ± 0.001	0.080 ± 0.004	1.300 ± 0.200
8	0.6 ± 0.1	0.008 ± 0.000	0.010 ± 0.000	0.013 ± 0.000	0.073 ± 0.001	0.800 ± 0.100
9	0.7 ± 0.1	0.010 ± 0.000	0.015 ± 0.000	0.017 ± 0.000	0.064 ± 0.001	0.670 ± 0.080
10	0.6 ± 0.1	0.005 ± 0.000	0.009 ± 0.000	0.010 ± 0.000	0.077 ± 0.001	0.500 ± 0.050
11	0.7 ± 0.2	0.010 ± 0.000	0.011 ± 0.000	0.018 ± 0.000	0.053 ± 0.001	0.800 ± 0.100
12	0.7 ± 0.2	0.007 ± 0.000	0.009 ± 0.000	0.011 ± 0.000	0.063 ± 0.001	0.800 ± 0.100
13	0.6 ± 0.1	0.007 ± 0.000	0.008 ± 0.000	0.013 ± 0.000	0.060 ± 0.000	0.700 ± 0.100
14	0.6 ± 0.1	0.007 ± 0.000	0.008 ± 0.000	0.011 ± 0.000	0.063 ± 0.001	0.610 ± 0.060
15	0.6 ± 0.1	0.005 ± 0.000	0.006 ± 0.000	0.008 ± 0.000	0.070 ± 0.000	0.650 ± 0.060

	LOW STD	HIGH STD
MARDT ET AL. (2017)	0.73	0.27
NEURALMJP	0.74 ± 0.02	0.26 ± 0.02

Table 16. Stationary distributions of models trained on the dataset. Values of NeuralMJP are mean and standard deviation over 5 runs.

	NEURALMJP
LOW STD $\rightarrow$ HIGH STD	0.028 ± 0.007
HIGH STD $\rightarrow$ LOW STD	0.085 ± 0.011

Table 17. Transition rates learned by NeuralMJP on the dataset. Values are mean and standard deviation over 5 runs.

N. A Simple Protein Folding Model

N.1. Data Description and Pre-Processing

Mardt et al. (2017) simulate a 10^5 time step trajectory of the 5-dimensional Brownian dynamics

$$dx(t) = -\nabla U(x(t)) + \sqrt{2}dW(t) \quad ,$$

where $U(x)$ only depends on the norm $r(x) = |x|$ of x:

$$U(x) = \begin{cases} -2.5[r(x) - 3]^2 & \text{, if } r(x) < 3 \\ 0.5[r(x) - 3]^3 - [r(x) - 3]^2 & \text{, if } r(x) \geq 3 \end{cases}$$

These dynamics are bistable along the norm $r(x)$ and serve them as a toy protein folding model, representing a folded and unfolded state.

We follow Mardt et al. (2017) and use the Euler method with $\Delta t = 0.1$ to simulate trajectories of these dynamics. Initial states are sampled from $\mathcal{N}(-\mathbf{1}, 4I)$, where $\mathbf{1} = [1, 1, 1, 1, 1] \in \mathbb{R}^5$ and $I \in \mathbb{R}^{5\times5}$ is the identity matrix. The initial distribution was extracted from the code that accompanies their paper. Its samples are likely to be close to one of the bistable regions of the dynamics.

We simulate 1000 trajectories with 100 steps after a burn-in period of 1000 steps, for a total of 10^5 observations. During training, we normalize observation times to lie in $[0, 1]$. For training, we use 28 batches of size 32 and leave the remaining time series for test and validation.

We simulate 100 additional time series for prediction, each with 200 time steps after a burn-in period of 1000 steps.

N.2. Modelling and Results

We trained a NeuralMJP with two-state, unconstrained prior and posterior MJPs to capture the two stable regions along $r(x)$. For this dataset, we chose a gaussian emission model and learn the gaussian mean and diagonal covariance matrix.

Indeed, NeuralMJP discerns the two stable regions. It learns a mean reconstruction close to $\mathbf{0}$ for both states, but two different covariance matrices, one with smaller diagonal entries, representing the folded state, the other with larger diagonal entries, representing the unfolded state.

The long-term dynamics, represented by the stationary distribution, of our continuous time model agrees with the discrete time model of Mardt et al. (2017), see Table 16. We provide the learned transition rates of our model in Table 17.

Table 18. Two-Mode Hybrid System transition rates including standard deviation over 5 runs. Note that one run of NeuralMJP suffered from index collapse (see Appendix F). Learned transition rates from the remaining runs are close to the ground truth.

	BOTTOM $\rightarrow$ TOP	TOP $\rightarrow$ BOTTOM
GROUND TRUTH	0.2	0.2
KÖHS ET AL. (2021)	0.64	0.63
NEURALMJP, 5 RUNS	0.16 ± 0.02	0.90 ± 0.90
NEURALMJP, 4 RUNS	0.19 ± 0.02	0.36 ± 0.06

O. A Toy Two-Mode Switching System

O.1. Data Description and Pre-Processing

Köhs et al. (2021) generate a time series of length 67 from a trajectory of a switching stochastic differential equation with two modes. We generate this time series with the code provided by the authors. Here, we only provide a rough outline of the generation process for convenience and refer to (Köhs et al., 2021) for a detailed description.

They consider a two-state, time-homogeneous MJP with transition rates

$$\begin{pmatrix} 0 & 0.2 \\ 0.2 & 0 \end{pmatrix}$$

as the underlying switching process of their dynamics. A trajectory z of this process defines a switching stochastic differential equation

$$dy(t) = \alpha_{z(t)}(\beta_{z(t)} - y(t)) + 0.5dW(t)$$

with $\alpha_1 = \alpha_2 = 1.5$, $\beta_1 = -1$ and $\beta_2 = 1$. This is understood as a system that switches between two Ornstein-Uhlenbeck processes with different drift terms, based on the state of the underlying switching process z.

Köhs et al. (2021) simulate a single trajectory of these dynamics, observe it at observation times sampled from a Poisson point process with intensity $\frac{1}{\lambda} = 0.35$ and corrupt the observations with samples from $\mathcal{N}(0, 0.1)$.

During training, we normalize observation times and values to lie in $[0, 1]$.

O.2. Modelling and Results

We train a two-state NeuralMJP with a gaussian emission model and learn the gaussian mean and standard deviation.

NeuralMJP reconstructs observations stemming from both processes closely, see Figure 12. Moreover, the posterior process approximates the true switching dynamics of the observed time series almost perfectly. Learned transition rates are closer to the ground-truth switching process than the model of Köhs et al. (2021) (see Table 18).

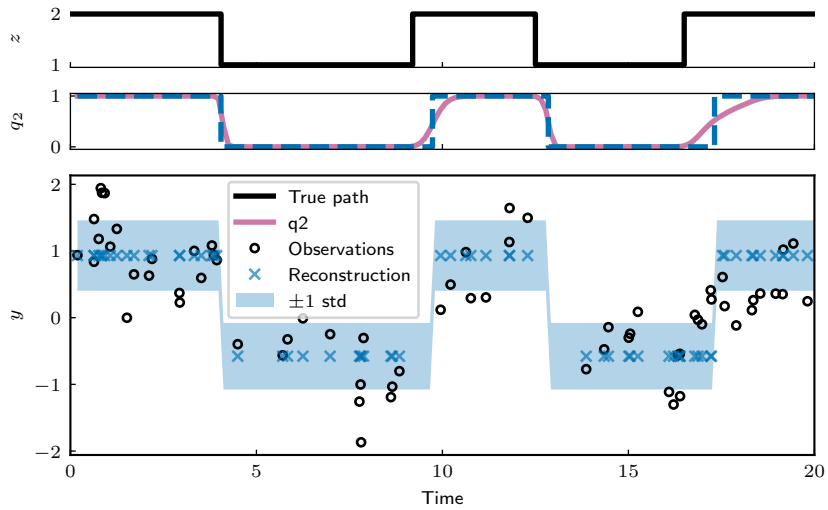

Figure 12. Results on Two-Mode Hybrid System. Bottom: Observations of the Hybrid System and reconstruction by NeuralMJP. Middle: Posterior probability of latent state corresponding to upper Ornstein-Uhlenbeck process (red solid line) and greedy prediction of that state (blue dashed line). Top: True underlying switching dynamics.