NEURAL ODE PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Ordinary Differential Equations (NODEs) use a neural network to model the instantaneous rate of change in the state of a system. However, despite their apparent suitability for dynamics-governed time-series, NODEs present a few disadvantages. First, they are unable to adapt to incoming data-points, a fundamental requirement for real-time applications imposed by the natural direction of time. Second, time-series are often composed of a sparse set of measurements, which could be explained by many possible underlying dynamics. NODEs do not capture this uncertainty. To this end, we introduce Neural ODE Processes (NDPs), a new class of stochastic processes determined by a distribution over Neural ODEs. By maintaining an adaptive data-dependent distribution over the underlying ODE, we show that our model can successfully capture the dynamics of low-dimensional systems from just a few data-points. At the same time, we demonstrate that NDPs scale up to challenging high-dimensional time-series with unknown latent dynamics such as rotating MNIST digits and bouncing ball videos.

1 INTRODUCTION

Many time-series that arise in the natural world, such as the state of a harmonic oscillator, the populations in an ecological network or the spread of a disease, are the product of some underlying dynamics. Sometimes, as in the case of a video of a swinging pendulum, these dynamics are latent, and do not manifest directly in the observation space (pixel space, in this case). Neural Ordinary Differential Equations (NODEs) (Chen et al., 2018), which use a neural network to parametrise the velocity of an ODE, have become a natural choice for capturing the dynamics of such time-series (Çağatay Yıldız et al., 2019; Rubanova et al., 2019; Norcliffe et al., 2020; Kidger et al., 2020).

However, despite their fundamental connection to dynamics-governed time-series, NODEs present certain limitations that hinder their adoption in these settings. Firstly, NODEs cannot adjust predictions as more data is collected without retraining the model. This ability is particularly important for real-time applications, where it is desirable that models adapt to incoming data points as time passes and more data is collected. Secondly, without a larger number of regularly spaced measurements, there is usually a range of plausible underlying dynamics that can explain the data. NODEs, however, only offer a single maximum likelihood prediction and are unable to learn stochastic dynamics. As many real world time-series are comprised of sparse sets of measurements, often irregularly sampled, the model can fail to represent the diversity of suitable solutions.

To address these limitations, we introduce Neural ODE Processes (NDP), a new class of stochastic processes governed by stochastic data-adaptive dynamics. Our probabilistic Neural ODE formulation relies on and extends the framework provided by Neural Processes (NPs), and runs parallel to other attempts to incorporate application-specific inductive biases in this class of models such as Attentive NPs (Kim et al., 2019), ConvCNPs (Gordon et al., 2019), and MPNPs (Day et al., 2020). We demonstrate that NDPs can adaptively capture many potential dynamics of low-dimensional systems when faced with limited amounts of data. Additionally, we show that our approach scales to high-dimensional time series with latent dynamics such as videos of bouncing balls (Sutskever et al., 2009) and rotating MNIST digits (Casale et al., 2018).

2 BACKGROUND AND FORMAL PROBLEM STATEMENT

Problem Statement. We consider modelling random functions $F : \mathcal{T} \to \mathcal{Y}$, where $\mathcal{T} = [t_0, \infty)$ represents time and $\mathcal{Y} \subset \mathbb{R}^d$ is a compact subset of $\mathbb{R}^d$. We assume F has a distribution $\mathcal{D}$, induced by another distribution $\mathcal{D}'$ over some underlying dynamics that govern the time-series. Given a specific instantation $\mathcal{F}$ of F, let $X_{\mathcal{F}} = \{(t_i^C, \boldsymbol{y}_i^C)\}_{i \in I}$ be a set of samples from $\mathcal{F}$ with some indexing set I. We refer to $X_{\mathcal{F}}$ as the context points, as denoted by the superscript C. For a context $X_{\mathcal{F}}$, the task is to predict $\{\boldsymbol{y}_j^T\}_{j \in J}$ at a set of target times $\{t_j^T\}_{j \in J}$, where J is another index set. Conventionally, as in the NP paper, the target times form a superset of the context times and we have $\{t_i^C\} \subseteq \{t_j^T\}$.

During training we assume access to a dataset of such sequences $\{X_{\mathcal{F}}\}$ with $\mathcal{F}$ sampled from F. We are interested in learning the underlying distribution over the dynamics as well as the induced distribution over functions. We note that when the dynamics are not latent, and manifest directly in the observation space $\mathcal{Y}$, the distribution over ODE trajectories and the distribution over functions coincide.

Neural ODEs NODEs are a class of models which parametrize the velocity $\dot{\boldsymbol{z}}$ of a state $\boldsymbol{z}$ with the help of a neural network $\dot{\boldsymbol{z}} = f_\theta(\boldsymbol{z}, t)$. Given the initial time t_0 and target time t_i^T, NODEs predict the corresponding state $\hat{\boldsymbol{y}}_{t_i^T}$ by performing the following integration and decoding operations:

$$\boldsymbol{z}(t_0) = h_1(\boldsymbol{y}_{t_0}), \qquad \boldsymbol{z}(t_i^T) = \boldsymbol{z}(t_0) + \int_{t_0}^{t_i^T} f_\theta(\boldsymbol{z}(t), t)dt, \qquad \hat{\boldsymbol{y}}(t_i^T) = h_2(\boldsymbol{z}(t_i^T)), \tag{1}$$

where h_1 is a neural network and h_2 is a linear transformation. When the dimensionality of $\boldsymbol{z}$ is greater than that of $\boldsymbol{y}$, the resulting model is a generalised Augmented Neural ODE (Dupont et al., 2019) with input layer augmentation (Massaroli et al., 2020). The extra dimensions offer the model additional flexibility as well as the ability to learn higher-order dynamics (Norcliffe et al., 2020).

NODEs are trained by minimising the mean squared error between $\hat{\boldsymbol{y}}_{t_i^C}$ and $\boldsymbol{y}_{t_i^C}$ over all time-steps $\{t_i^C\}$ with $i \in I$. They cannot be trained on multiple time-series in parallel, provide only a maximum-likelihood prediction, and accounting for additional data requires further training or retraining entirely.

Neural Processes (NPs). NPs model a random function $F : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^{d_1}$ and $\mathcal{Y} \subseteq \mathbb{R}^{d_2}$. The NP represents a given instantiation $\mathcal{F}$ of F through the global latent variable $\boldsymbol{z}$, which parametrises the variation in F. Thus, by having $\mathcal{F}(\boldsymbol{x}_i) = g(\boldsymbol{x}_i, \boldsymbol{z})$, the marginal predictive distribution is given by:

$$p(\boldsymbol{y}_{1:n}|\boldsymbol{x}_{1:n}) = \int p(\boldsymbol{z}) \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{y}_i | g(\boldsymbol{x}_i, \boldsymbol{z}), \sigma^2) d\boldsymbol{z}, \tag{2}$$

where $p(\boldsymbol{z})$ is chosen to be a multivariate standard normal distribution and $\boldsymbol{y}_{1:n}$ is a shorthand for the sequence $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$. The model can be trained using an amortised variational inference procedure that naturally gives rise to a *permutation-invariant encoder* $q_\theta(\boldsymbol{z}|\boldsymbol{x}_{1:m}, \boldsymbol{y}_{1:m})$, which stores the information about the context points. Conditioned on this information, the *decoder* $g(\boldsymbol{x}, \boldsymbol{z})$ can make predictions at any input location $\boldsymbol{x}$. We note that while the domain $\mathcal{X}$ of the random function F is arbitrary, in this work we are interested only in stochastic functions with domain on the real line (time-series). Therefore, from here our notation will reflect that, using t as the input instead of $\boldsymbol{x}$. The output $\boldsymbol{y}$ remains the same.

3 NEURAL ODE PROCESSES

In this section, we introduce Neural ODE Processes (NDPs), which learn to approximate a random function defined over time. To do so, we consider an NP whose context z is used to determine a distribution over ODEs. Concretely, z gives a distribution over the initial position (and optionally – the initial velocity) and, at the same time, stochastically conditions its instantaneous velocity. The positions given by the ODE trajectories at any time t are then decoded to give the predictions at t. In what follows, we offer a detailed description of each component of the NDPs. A schematic of the model can be seen in Figure 1.

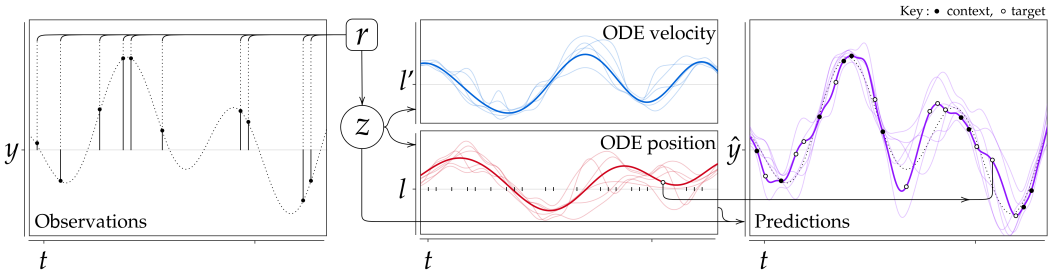

Figure 1: Schematic diagram of Neural ODE Processes. *Left:* Observations from a time series, the context set •, are encoded and aggregated to form r which parametrises the global latent variable $\mathbf{z}$. *Middle:* Samples from the global latent variable, $\boldsymbol{z}$, are used to initialise and condition the latent Neural ODE, with each sample producing a plausible, coherent trajectory. *Right:* Predictions at a target time, t_i^T, are made by decoding the state of the ODE, $l(t_i^T)$, together with the sample $\boldsymbol{z}$ used by that trajectory. An example is represented by the ∘ connected from the ODE position plot to the Predictions plot. *Middle, right:* dark lines relate to the same sample, fainter lines to other samples.

Encoder and Aggregator The encoder produces a representation $\boldsymbol{r}_i = f_e([t_i^C, \boldsymbol{y}_i^C])$ for each context pair $[t_i^C, \boldsymbol{y}_i^C]$. The aggregator combines all of the representations $\boldsymbol{r}_i$ to form a global representation, $\boldsymbol{r}$, that parametrises the distribution of the global latent context, $\boldsymbol{z} \sim \mathcal{N}(\mu(\boldsymbol{r}), \mathrm{diag}(\sigma(\boldsymbol{r})))$. The aggregator must preserve order invariance, e.g. taking the mean.

Latent ODE The context $\boldsymbol{z}$ is split into an initial latent position and the sub-context such that $\boldsymbol{z} = [\boldsymbol{l}(t_0), \boldsymbol{z}']$. Then, using the target times, $t^T = [t_1^T, t_2^T, t_3^T ..., t_N^T]$, the latent state at a given time is found by evolving a Neural ODE:

$$\boldsymbol{l}(t_i^T) = \boldsymbol{l}(t_0) + \int_{t_0}^{t_i^T} f_\theta(\boldsymbol{l}(t), t, \boldsymbol{z}')dt, \tag{3}$$

where f_θ is a neural network that models the velocity of $\boldsymbol{l}$. The addition of a latent state that evolves according to an ODE (Rubanova et al., 2019) allows for a richer encoding if the function is generated by an ODE.

Decoder Uses the sub-context, target time and latent state to make a prediction for $\boldsymbol{y}$ at a target time: $\hat{\boldsymbol{y}}_i^T = g(\boldsymbol{l}(t_i^T), t_i^T, \boldsymbol{z}')$.

Finally, the marginal distribution of the NDP is given by:

$$p(\boldsymbol{y}_{1:n}|t_{1:n}) = \int p(z) \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{y}_i | g(\boldsymbol{l}_i, t_i, \boldsymbol{z}'), \sigma^2) dz. \tag{4}$$

3.1 Model Variations

Here we present the different ways to implement the model. The majority of the variation is in the architecture of the decoder. However, it is possible to vary the encoder such that $f_e([t_i^C, \boldsymbol{y}_i^C])$ can be a multi-layer-perceptron, or contain convolutions.

Neural ODE Process (NDP) In this setup the the decoder is a linear layer, which takes a concatenation of the latent state and a function of the latent state, time and sub-context as input:

$$\hat{\boldsymbol{y}}_i^T = \boldsymbol{W}(\boldsymbol{l}(t_i^T) \parallel h(\boldsymbol{l}(t_i^T), t_i^T, \boldsymbol{z}')), \tag{5}$$

where h can be a multi-layer-perceptron or contain convolutional layers. This type of model is particularly suitable for high-dimensional time-series where the dynamics are fundamentally latent. However, its additional representation power given by h make it a good default choice for most settings.

Second Order Neural ODE Process (ND2P) This variation has the same decoder architecture as NDP, however the latent ode evolves according to a second order ODE. The latent state, $\boldsymbol{l}$, is split in half into a 'position', $\boldsymbol{l}_1$ and 'momentum', $\boldsymbol{l}_2$:

$$\dot{\boldsymbol{l}_1} = \boldsymbol{l}_2, \qquad \dot{\boldsymbol{l}_2} = f_\theta(\boldsymbol{l}_1, \boldsymbol{l}_2, t, \boldsymbol{z}'). \tag{6}$$

This model is designed for time-series where the dynamics are second-order. This is often the case given the use of Newton's 2nd Law to describe physical laws (Norcliffe et al., 2020).

NDP Latent-Only (NDP-L) Here the decoder is a simple linear layer of the latent state:

$$\hat{\boldsymbol{y}}_i^T = \boldsymbol{W}(\boldsymbol{l}(t_i^T)). \tag{7}$$

This is based on the assumption that the latent state, which evolves according to an ODE, contains all the required information without any non-linearities. This would be suitable for simple functions generated by ODEs, for example sines and exponentials. This decoder implicitly contains the time and context, because the latent state was evolved according to an ODE that depended on time and context given by Equation 3.

ND2P Latent-Only (ND2P-L) This model combines the assumption of second order underlying dynamics, and the idea of the latent state containing all of the information. The decoder is a linear layer of the latent state as in Equation 7, whose phase space dynamics are constrained as in Equation 6.

3.2 Neural ODE Processes as Stochastic Processes

The Kolmogorov Extension Theorem states that *exchangeability* and *consistency* are necessary and sufficient conditions for a collection of joint marginal distributions to define a stochastic process Øksendal (2003); Garnelo et al. (2018). We define these conditions and show that the NDP model satisfies them.

Definition 3.1 (Exchangeability)**.** Exchangeability refers to the invariance of the joint distribution $\rho_{t_{1:n}}(\boldsymbol{y}_{1:n})$ under permutations of $\boldsymbol{y}_{1:n}$. That is, for a permutation π of $\{1, 2, ..., n\}$, $\pi(t_{1:n}) = (t_{\pi(1)}, ..., t_{\pi(2)})$ and $\pi(\boldsymbol{y}_{1:n}) = (\boldsymbol{y}_{\pi(1)}, ..., \boldsymbol{y}_{\pi(2)})$, the joint probability distribution $\rho_{t_{1:n}}(\boldsymbol{y}_{1:n})$ is invariant if $\rho_{t_{1:n}}(\boldsymbol{y}_{1:n}) = \rho_{\pi(t_{1:n})}(\pi(\boldsymbol{y}_{1:n}))$

Proposition 3.1. *NDPs satisfy the exchangeability condition.*

Proof. Neural ODE Processes use an additional latent state, $\boldsymbol{l}_i$, where $\boldsymbol{l}_i = \boldsymbol{l}(t_i)$, in the decoder (the latent state is also a function of $\boldsymbol{z}'$, however the sub-context stays constant in time so is excluded for notation):

$$\hat{\boldsymbol{y}}_i = \mathcal{F}(t_i) = g(\boldsymbol{l}_i, t_i, \boldsymbol{z}'). \tag{8}$$

The exchangeability is satisfied because any permutation acting on t also acts on $\boldsymbol{l}$, and there are no permutations of $\boldsymbol{z}'$. □

Definition 3.2 (Consistency)**.** Consistency says if a part of a sequence is marginalised out, then the joint probability distribution is the same as if it was only originally taken from the smaller sequence:

$$\rho_{t_{1:m}}(\boldsymbol{y}_{1:m}) = \int \rho_{t_{1:n}}(\boldsymbol{y}_{1:n}) d\boldsymbol{y}_{m+1:n}. \tag{9}$$

Proposition 3.2. *NDPs satisfy the consistency condition.*

Proof. By using the same formulation from NPs:

$$\rho_{t_{1:n}}(\boldsymbol{y}_{1:n}) = \int p(\mathcal{F}) \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{y}_i | \mathcal{F}(t_i), \sigma^2) d\mathcal{F}. \tag{10}$$

Consistency is guaranteed if the prediction of $\boldsymbol{y}_i$ can be written as a single function of t_i, and using Equation 8, it is shown that the fixed and learnable function used to predict $\boldsymbol{y}_i$ is $g(\boldsymbol{l}_i, t_i, \boldsymbol{z}')$, however it can be rewritten to be just a function of t, $\mathcal{F}(t)$. Therefore, consistency is also satisfied so NDPs are stochastic processes. □

It is important to note that the stochasticity comes from sampling the full context $\boldsymbol{z}$. There is no stochasticity in the ODE (Liu et al., 2019; Li et al., 2020; Tzen & Raginsky, 2019; Jia & Benson, 2020), such as Brownian motion. For any given context, both the latent state trajectory and the observation space trajectory are fully determined.

3.3 Running Time Complexity

For a model with m context points and n extra target points, an NP has running time complexity $O(m+n)$, since the model only has to encode and decode each point. However, a Neural ODE Process has an added level of complexity due to the integration process. This has runtime complexity $O(\text{NFE})$ where NFE is the number of function evaluations. In turn, the worst-case NFE depends on the minimum step size δ the ODE solver can use and the maximum time we are interested in, which we denote by $t_{\max}$. Therefore, the worst-case complexity of NDPs is $O(m+n+t_{\max}/\delta)$. We note that for very complex solutions δ could be extremely small, which could result in a very large constant dramatically affecting practical performance. However, modern ODE solvers use adaptive step sizes that adjust to the data that has been supplied and it is unlikely to be a problem in practice. Additionally, the added term is constant with respect to $(m+n)$, and therefore NDP will also scale linearly with the number of context and target points.

In our experiments, we notice the NDP models are between 1 and 1.5 orders of magnitude slower to train than NPs in terms of wall-clock time. At the same time, the additional training time is traded-off by a significantly faster loss decay per epoch and superior final performance. We provide a table of time ratios from our 1D experiments, from section 4.1, in Appendix A.

4 Experiments

To test the proposed advantages of NDPs we carried out various experiments on time series data. Additional details for every task considered can be found in C.

4.1 One Dimensional Regression

We begin with a set of 1D toy problems of increasing complexity—straight lines, sine waves, exponentials and damped oscillators—that can be described by ODEs. We compare the loss with Neural Processes throughout training on a test set. The results can be seen in Figure 2. It can be seen that NDPs train in fewer iterations to a lower test loss due to the additional encoded structure of the latent ode state, despite having approximately 10% fewer parameters.

Further to this we carry out an ablation study, training all model variants on all the 1D datasets. We expect second order models to be superior on the Harmonic Oscillator and the Linear Dataset. The Linear dataset counter-intuitively, is easier with second order, $f_\theta(\boldsymbol{l}_1, \boldsymbol{l}_2, t, \boldsymbol{z}') = 0$, and then the initial $\boldsymbol{l}_2(t_0)$ needs to be the gradient. Whereas for NDP it is harder to learn $f_\theta(\boldsymbol{l}, t, \boldsymbol{z}') = C$. Additionally due to the wide latent state, it is just as easy to learn a first order ODE for a sine curve as the second order ODE, so we do not expect ND2P to outperform NDP on the sine dataset. Final test MSE losses provided in table 1 and the training plots presented in Appendix C.1.

We see that NDP and ND2P outperform NPs, however there is no significant difference between them. We also see in the training plots that all models can reach a lower loss in fewer iterations than NP. Therefore, we continue to use only NDPs. Firstly because they are the model variant with the fewest restrictions, with unrestricted latent phase-space dynamics, unlike the second order counterparts and a more expressive decoder architecture, unlike the latent-only variants. Secondly, as is seen in A, NDPs train faster in a faster wall clock time than other variants.

4.2 Predator-Prey Dynamics

The Lotka-Volterra Equations are used to model the dynamics of a two species system, where one species predates on the other. The populations of the prey, u, and the predator, v, are given by the differential equations:

$$\frac{du}{dt} = \alpha u - \beta uv, \qquad \frac{dv}{dt} = \delta uv - \gamma v. \tag{11}$$

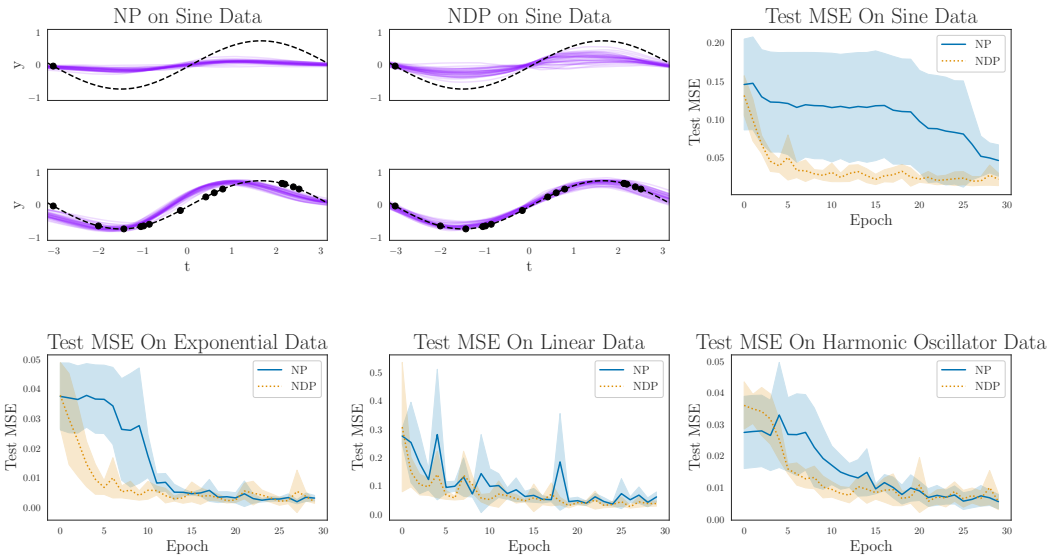

Figure 2: NPs and NDPs on the 1D datasets, we present the loss plots during training and example posteriors on the sine data. On the sine data we see that NDPs are able to give a larger range of functions when only one point is available, and a smaller, more accurate range as more points in the time series are observed. On the right and below we see that quantitatively NDPs are also able to train to a higher accuracy in fewer iterations. This is expected for functions that are generated by ODEs. Both models were trained for 30 epochs.

Table 1: Final MSE Losses on 1D regression. The latent-only model variants perform roughly equally and tend to have an equal or slightly higher MSE than NP. The more expressive variants also have similar final losses, but they are consistently lower than NP and the latent-only variants. We see ND2P outperforming NDP on the datasets that benefit from second order dynamics, Harmonic Oscillators and Linear.

Final MSE	**Sine**	**Exponential**	**Linear**	**Oscillators**
NP	0.046 ± 0.021	0.003 ± 0.001	0.061 ± 0.018	0.006 ± 0.002
NDP	0.022 ± 0.008	0.002 ± 0.001	0.037 ± 0.009	0.006 ± 0.003
ND2P	0.030 ± 0.009	0.002 ± 0.001	0.023 ± 0.009	0.006 ± 0.001
NDP-L	0.029 ± 0.016	0.005 ± 0.002	0.056 ± 0.029	0.007 ± 0.001
ND2P-L	0.022 ± 0.005	0.005 ± 0.003	0.045 ± 0.027	0.007 ± 0.002

for positive real parameters, $\alpha, \beta, \delta, \gamma$. Intuitively, when prey is plentiful, the predator population increases ($+\delta uv$), and when there are many predators, the prey population falls ($-\beta uv$). The populations exhibit periodic behaviour, with the phase-space orbit determined by the conserved quantity $V = \delta u - \gamma \ln(u) + \beta v - \alpha \ln(v)$. Thus for any predator-prey system there exists a range of stable functions describing the dynamics, with any particular realisation being determined by the initial conditions, (u_0, v_0). We consider the system $(\alpha, \beta, \gamma, \delta) = (2/3, 4/3, 1, 1)$

We see that due to the additional ODE in the encoding, NDPs are able to train in fewer iterations with lower loss (see C.2). This is despite having a representation $\boldsymbol{r}$ and context $\boldsymbol{z}$ with lower dimensionality, leading to fewer model parameters (10% fewer than the NP model). We visualise these advantages in Figure 3.

4.3 Rotating MNIST

We consider the rotating MNIST digits task (Casale et al., 2018; Çağatay Yıldız et al., 2019) and introduce a more challenging additional version. Samples in the original start upright and rotate once

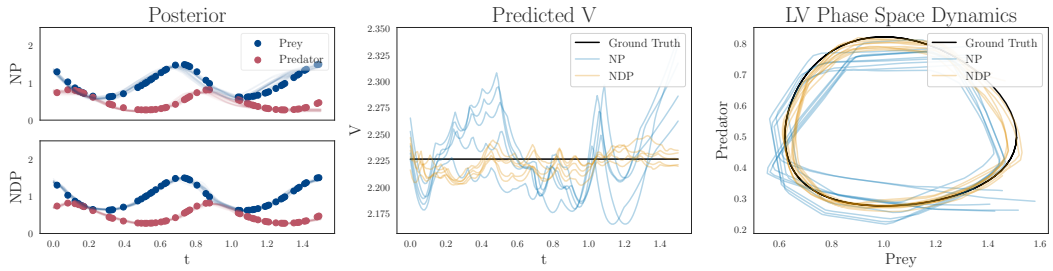

Figure 3: NPs and NDPs on the Lotka-Volterra task. *Left:* NPs are less able to model the dynamics, diverging from the ground truth even in regions with frequent context samples, whereas the variation in the NDP is more appropriately concentrated in the less sampled regions. *Middle:* Plotting the theoretically conserved quantity V better exposes how the models deviate from the ground truth. *Right:* In phase space (u, v) the NDP is more clearly seen to better track the ground truth.

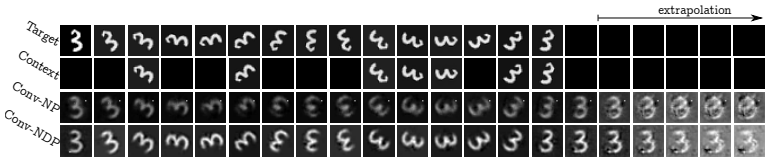

Figure 4: Rotating MNIST dataset. *First row:* The ground truth time-series. *Second row:* Seven context points supplied to the model. *Third row:* NP predictions. *Fourth row:* NDP predictions. NDPs perform better at interpolation and marginally better at extrapolation. This is also confirmed by the MSE results in the second half of Table 2.

Table 2: MSE on the Rotating MNIST test set.

Model	**MSE** $\times 10^{-2}$
GPPVAE	2.88 ± 0.01
ODE^{2}VAE-KL	**1.94** ± **0.03**
Conv-NP-1	3.18 ± 0.56
Conv-NDP-1	**1.94** ± **0.40**
Conv-NP-7	2.80 ± 0.42
Conv-NDP-7	**1.63** ± **0.49**

over 16 frames $(= 360° s^{-1})$ (i.e. constant angular velocity, zero angular shift). In our adaptation, the angular velocity varies between samples in the range $(360° \pm 90°)s^{-1}$ and each sample starts at a random initial rotation. We adapt our model to use a convolutional neural network encoder and a transposed convolutional decoder.

As visible in Figure 5, the Conv-NDP model is able to extrapolate on the variable velocity MNIST dataset, although such extrapolation is not evident with the standard rotating MNIST dataset. We hypothesise that this capability is due to the different angular velocities in the dataset. In a manner similar to dropout (Srivastava et al., 2014), the model is forced to unlearn the relatively trivial interpolation between frames in favour of learning dynamics. Only by doing so is the model able to reduce the loss beyond interpolation. Moreover, both models exhibit good interpolation performance, as seen in Figures 4 and 5. For comparisons to baseline models, we show the mean squared errors (MSE) for the 4th rotated MNIST digit in Table 2. GPPVAE (Casale et al., 2018) is a Gaussian process prior variational autoencoder acting as a baseline for this experiment alongside ODE^{2}VAE. The number after Conv-NDP- indicates the number of context frames used. Finally, we show that different stylistic aberrations are predicted using our model in Figure 10.

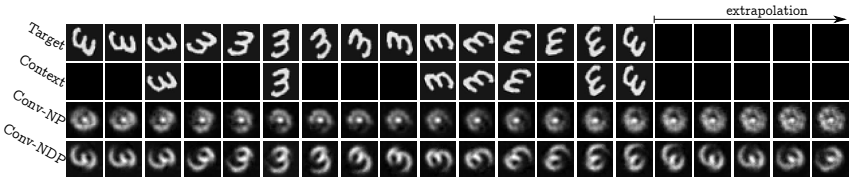

Figure 5: Variable angular velocity and angular shift rotating MNIST extrapolation. Conv-NDP is able to extrapolate beyond the training data range whereas Conv-NP cannot. The top row shows the targets (ground truth).

5 Discussion and Related Work

We now consider two perspectives on how Neural ODE Processes relate to existing work and discuss the model in these contexts.

NDPs as Neural Processes From the perspective of stochastic processes, NDPs can be seen as an NP defined over time and, as such, existing improvements in this family are likely orthogonal to our own. For instance, following the work of Kim et al. (2019), we would expect adding an attention mechanism to NDPs to reduce uncertainty around context points and improve performance generally. Additionally, intrinsic sequential nature of stochastic time-series could be further exploited as in Sequential NPs (Singh et al., 2019). For application domains where the observations evolve on a graph structure, such as traffic networks, relational information could be exploited with message passing operation as in MPNPs (Day et al., 2020).

NDPs as Neural ODEs From a dynamics perspective, NDPs can be thought of as an amortised Bayesian Neural ODE. In this sense, ODE^{2}VAE (Çağatay Yıldız et al., 2019) is the model that is most closely related to our method. While there are many common ideas between the two, significant differences exist. Firstly, NDPs do not use a Bayesian neural network, but are linked to them through the theoretical connections inherited from NPs (Garnelo et al., 2018). Secondly, during training, NDPs use samples from the ODE trajectories, rather than explicitly evolving the probability density of the initial position like a Continuous Normalizing Flow (Chen et al., 2018). Finally, NDPs condition the latent ODE's initial position and velocity throughout the trajectory on a flexible context (of variable size), whereas the ODE^{2}VAE only conditions the initial position and velocity. Therefore, our model can dynamically adapt to changing contexts.

Another recent paper from the neural differential equations literature has proposed Neural Controlled Differential Equations (NCDEs) (Kidger et al., 2020) to address the adaptability problem of NODEs. To that end, NCDEs use cubic spline interpolations of the time series to "modulate" a differential equation. Unlike NDPs, their method offers adaptivity without assuming a given distribution $\mathcal{D}$ over the time-series. However, NCDEs do not capture stochastic dynamics. Ultimately, NCDEs and NDPs are completely orthogonal directions and, in our early experiments of integrating the two, we noticed significantly improved local context adaptivity. We believe this could be a very promising line of future work.

6 Conclusion

We introduce Neural ODE Processes (NDPs), a new class of stochastic processes suitable for modelling data-adaptive stochastic dynamics. NDPs tackle the two main problems faced by Neural ODEs applied to dynamics-governed time series: adaptability to incoming data points and uncertainty in the underlying dynamics when the data is sparse and, potentially, irregularly sampled. To do so, NDPs include a probabilistic ODE as an additional encoded structure, thereby incorporating the inductive bias that the time-series is the direct or latent manifestation of an underlying ODE. Furthermore, NDPs maintain a linear runtime complexity in the size of the inputs, with an added term accounting for the complexity of the ODE solver. This linear runtime makes NDPs scalable to larger datasets, similarly to NPs.

We evaluate our model on synthetic 1D and 2D data, as well as higher dimensional problems such as rotating MNIST digits. Our method exhibits superior training performance when compared with NPs, yielding a lower loss in fewer iterations. Whether or not the underlying ODE of the data is latent, we find that where there is a fundamental ODE governing the dynamics, NDPs perform well.

REFERENCES

Francesco Paolo Casale, Adrian V Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian Process Prior Variational Autoencoders. *arXiv e-prints*, art. arXiv:1810.11738, October 2018.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Ben Day, Cătălina Cangea, Arian R. Jamasb, and Pietro Liò. Message passing neural processes, 2020.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 3140–3150. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8577-augmented-neural-odes.pdf`.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes, 2018.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes, 2019.

Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations, 2020.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*, 2020.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes, 2019.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations, 2020.

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise, 2019.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes, 2020.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Nikola Simidjievski, and Pietro Liò. On second order behaviour in augmented neural odes, 2020.

Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent odes for irregularly-sampled time series, 2019.

Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. Sequential neural processes. In *Advances in Neural Information Processing Systems*, pp. 10254–10264, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in neural information processing systems*, pp. 1601–1608, 2009.

Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit, 2019.

Ben H Williams, Marc Toussaint, and Amos J Storkey. Extracting motion primitives from natural handwriting data. In *International Conference on Artificial Neural Networks*, pp. 634–643. Springer, 2006.

Çağatay Yıldız, Markus Heinonen, and Harri Lähdesmäki. Ode^{2}vae: Deep generative second order odes with bayesian neural networks, 2019.