META-LEARNING USING PRIVILEGED INFORMATION FOR DYNAMICS

Ben Day, Alexander Norcliffe, Jacob Moss & Pietro Liò
Department of Computer Science
University of Cambridge
Cambridge, United Kingdom
{bjd39, alin2, jm2311, pl219}@cam.ac.uk

ABSTRACT

Neural ODE Processes approach the problem of meta-learning for dynamics using a latent variable model, which permits a flexible aggregation of contextual information. This flexibility is inherited from the Neural Process framework and allows the model to aggregate sets of context observations of arbitrary size into a fixed-length representation. In the physical sciences, we often have access to structured knowledge in addition to raw observations of a system, such as the value of a conserved quantity or a description of an understood component. Taking advantage of the aggregation flexibility, we extend the Neural ODE Process model to use additional information within the Learning Using Privileged Information setting, and we validate our extension with experiments showing improved accuracy and calibration on simulated dynamics tasks.

1 INTRODUCTION & BACKGROUND

Learning using privileged information (LUPI) is a machine learning paradigm where we have access to additional information during training that may not be available at test time (Vapnik & Vashist, 2009; Vapnik & Izmailov, 2015). Typically this information is higher quality in some way, often it conveys some expert understanding we have about the system being modelled. To paraphrase an example of Vapnik & Vashist (2009), we might have access to extensive patient records associated with biopsy scans in a training set but wish to deploy our trained model 'on-the-front-line' where records are incomplete or unavailable. We would like to improve the model by using this information without coming to rely on it; to provide an accurate prognosis without human help. An analogy can be made to the role of a teacher. Besides providing corrections, skilled teachers accelerate the understanding of their students through explanations and insights. This is important in the context of learning about physics, where a new perspective often provides traction with difficult problems.

In this work we propose a method for conveying such insight in the case of modelling dynamics. **Our main contribution** is a new training-mode architecture that allows privileged information to guide learning, that results in more accurate predictions and better calibrated uncertainty estimation.

Neural Processes. Deep neural networks are excellent function approximators that are cheap to evaluate and straightforward to train, but typically only provide point estimates and require retraining to make use of information gained at test time. However, meta-learning (occasionally, *'learning-to-learn'*) with neural networks is enjoying a resurgence in popularity as an answer to the serialised learning question, and has been applied in a range of domains (Hospedales et al., 2020). Gaussian Processes have markedly different advantages, handling uncertainty in a principled way and adapting to new data at test time (Rasmussen, 2003), at the cost of computational expense. Neural Processes (NP) aim to combine the best of both by learning to model a distribution over functions, framing the meta-learning problem as a latent variable model (Garnelo et al., 2018a;b).

Neural ODEs. Dynamical systems are a fundamental object of study in physics and are often most elegantly described using ordinary differential equations (ODE). Neural ODEs (NODE) (Chen et al., 2018) combine the representation learning capabilities of neural networks with an ODE structure to allow ODEs to be learned directly from observational data. Follow-up works have made improvements to expressivity (Dupont et al., 2019), investigated time series modelling (Kidger et al., 2020;

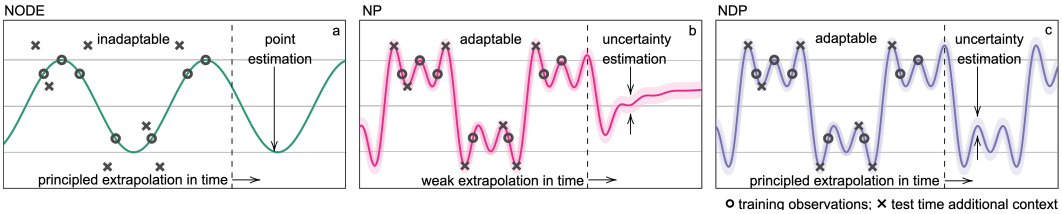


Figure 1: Comparing relevant abilities of NODEs, NPs, and NDPs. (a) NODEs explicitly learn dynamics, and can extrapolate well in time as a result, but cannot adapt to new information without retraining. (b) NPs offer uncertainty estimation and can adapt to new information but do not have a principled way to extrapolate in time. (c) NDPs combine these abilities in a single model.

Norcliffe et al., 2020), and tackled adaptability and uncertainty estimation using Bayesian neural networks (Çağatay Yıldız et al., 2019).

Neural ODE Processes. An alternative to the work of Çağatay Yıldız et al. (2019) for modelling uncertainty in dynamics is the Neural ODE Process (NDP) which uses a neural process derived formulation to learn a distribution over dynamics (Norcliffe et al., 2021). These models are able to fast-adapt to new data points at test time, unlike vanilla-NODEs, whilst inheriting superior time-series modelling as compared to vanilla-NPs. The key components of the NDP are an observation encoder, representation aggregator, latent ODE, and decoder.

Learning using privileged information. The LUPI framework, introduced by Vapnik & Vashist (2009) and expanded by Vapnik & Izmailov (2015), formalises a learning setup in which a teacher is able to provide the student learner with structured explanations, comments, comparisons, etc. beyond direct supervision. Hernández-Lobato et al. (2014) show that privileged information can be used effectively by GP classifiers, whilst Lambert et al. (2018) apply LUPI to deep neural networks by setting the dropout rate to be a function of the privileged information.

2 Learning using privileged information with NDPs

Problem statement. Formally, our setting closely matches that of the NDP: we consider modelling random functions over time, $F : \mathcal{T} \to \mathcal{Y}$, where F has distribution D induced by a second distribution, D', over some underlying dynamics[1]. We are provided with a set of labelled samples from an instantiation $\mathcal{F}$ of F referred to as the context set, indexed by I_{C} and denoted $\text{C} = \{(t_i^{\text{C}}, \boldsymbol{y}_i^{\text{C}})\}_{i \in I_{\text{C}}}$. The task is to predict the values $\{\boldsymbol{y}_j^{\text{T}}\}_{j \in I_{\text{T}}}$ taken by $\mathcal{F}$ at a set of target times $\{t_j^{\text{T}}\}_{j \in I_{\text{T}}}$, indexed by I_{T}, which together form the target set, $\text{T} = \{(t_j^{\text{T}}, \boldsymbol{y}_j^{\text{T}})\}_{j \in I_{\text{T}}}$.

To ensure the model is able to learn the underlying distribution over dynamics, and how this manifests as a distribution over functions, training assumes access to a set of time-series sampled from F. Diverging from the NDP setting, during training the model also has access to privileged information relating to each instantiation $\mathcal{F}$ of F, $\boldsymbol{\pi}_{\mathcal{F}}$. The privileged information could be some physical property or conserved quantity of the system, such as the spring stiffness, as in Figure 2. At test time no privileged information is provided and there is no difference from the vanilla-NDP setting.

Model overview. The differences introduced by the LUPI framework affect the training procedure, introducing a second source of information with which to form the global latent variable $\boldsymbol{z}$. To incorporate the privileged information within the LUPI-NDP we introduce a second encoder to produce a representation $\boldsymbol{r}_\pi = f_e^\pi(\boldsymbol{\pi})$, where f_e^π is parametrised as a fully-connected neural network. During training, an additional aggregation step is introduced to combine the observations derived representation, $\boldsymbol{r}_o$, with that of the privileged information to form a global representation, $\boldsymbol{r}_{\text{train}} = g(\boldsymbol{r}_o, \boldsymbol{r}_\pi)$. Practically, as the test-time global representation is that formed using only the observations, i.e. $\boldsymbol{r}_{\text{test}} = \boldsymbol{r}_o$, we choose to parametrise g as a residual network (He et al., 2016), that is, $g(\boldsymbol{r}_o, \boldsymbol{r}_\pi) = \boldsymbol{r}_o + g'(\boldsymbol{r}_o, \boldsymbol{r}_\pi)$, such that the privileged information is explicitly used as a correction term. At test time, the model we propose is equivalent to a vanilla-NDP.

Learning and Inference. The true posterior is intractable and, as is the NP custom, the model is trained using amortised variational inference. We select an objective that reflects the intended test

[1] If the dynamics manifest directly in observation space (i.e. they are not latent), D and D' coincide.

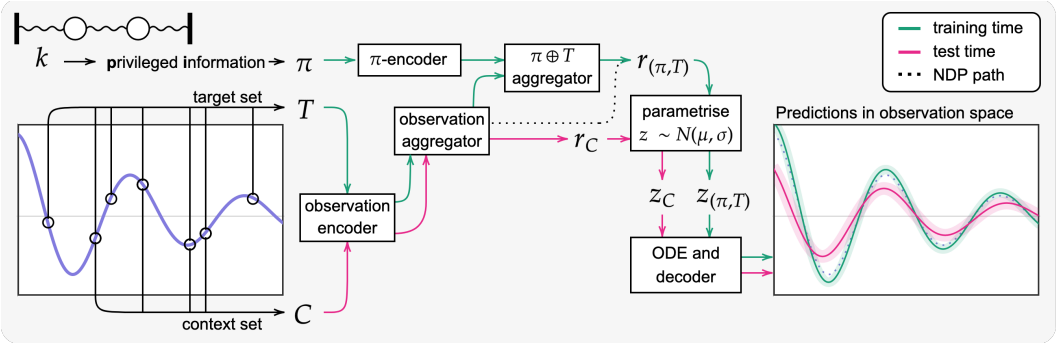


Figure 2: LUPI-NDP computational diagram. The LUPI training-mode is shown in green, with a dotted line for the vanilla-NDP path (no privileged information). Pink shows the evaluation-mode path, that matches the NDP evaluation-mode exactly. Observations are first encoded then aggregated. Privileged information, in this case the spring stiffness, is separately encoded and, during training, combined with the aggregated observations representation. As in NPs and NDPs, the representations parametrise the global latent variable $\boldsymbol{z}$, which is then used to condition the decoder. Typically, as in this cartoon, the privileged information enables highly accurate predictions.

time behaviour of making predictions based solely on the context set, given by

$$\log p(\boldsymbol{y}_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \mathrm{C}) \geq \mathbb{E}_{q(z|\mathrm{T},\boldsymbol{\pi})}\left[\log \frac{q'(\boldsymbol{z}|\mathrm{C})}{q(\boldsymbol{z}|\mathrm{T},\boldsymbol{\pi})} + \sum_{i\in I_\mathrm{T}} \log p(\boldsymbol{y}_i|\boldsymbol{z}, t_i)\right], \tag{1}$$

with variational posterior q (a derivation is provided in Appendix A). Note that during training C is a subset of T. The original NDP model is recovered by setting $\boldsymbol{\pi}$ to null (dropping it).

3 Experiments

We compare our proposed LUPI-NDP model with a vanilla-NDP that does not make use of the privileged information and, as there are no architectural changes, at test time the models are distinguishable only by the value of their weights. As explained in Section 2, both models are NDPs, so we refer to them as LUPI and NoPI (no-privileged-information) for clarity. Full model and training details are provided in Appendix C.

Metrics. Besides providing high quality predictions, NDPs, and NPs generally, are interesting because the way they approach the meta-learning problem (as a latent variable model) produces uncertainty estimates. We measure the quality of these estimates by the calibration error and sharpness. *Calibration* measures the degree to which uncertainty estimates are commensurate with residuals: if the model estimates an outcome to happen one time in every ten, does it actually occur that frequently? *Sharpness* is simply how low the uncertainty estimates are: between equally well-calibrated models we should favour the sharper model as being more informative. Practically speaking, we value accuracy over calibration over sharpness. We follow the definitions of Kuleshov et al. (2018) for the calibration error and sharpness, detailed in Appendix B.

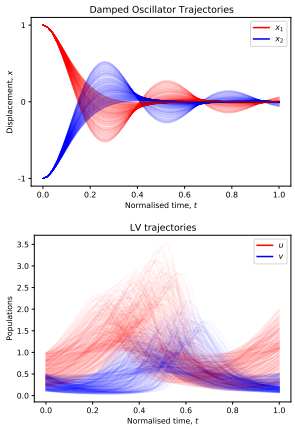
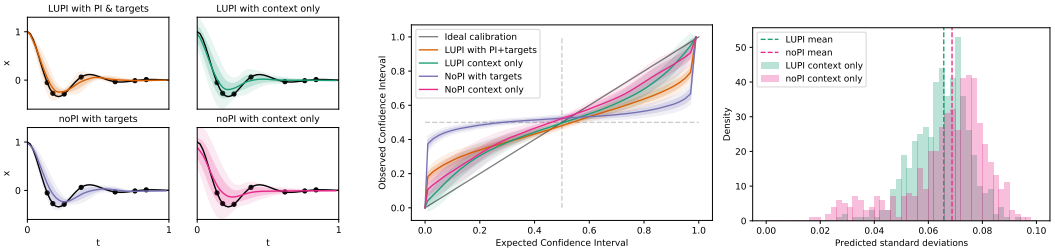


Figure 3: Induced dynamics.

Damped coupled oscillators. We first consider modelling a system of two masses attached by identical springs in series between parallel walls, described by the second-order differential equations

$$m_1\ddot{x}_1 = (x_2 - 2x_1)k - c\dot{x}_1 \quad , \quad m_2\ddot{x}_2 = (x_1 - 2x_2)k - c\dot{x}_2, \tag{2}$$

with spring and damping constants, k and c, respectively. A distribution over dynamics is induced by sampling some parameters of the system, and this parameter forms the privileged information. Examples of the trajectories induced by sampling over the drag coefficient as $c \sim \mathrm{U}(0.5, 2)$ are presented in Figure 3.



(a) Example predictions (b) Calibration (c) Sharpness

Figure 4: Various comparisons of trained LUPI and NoPI models on the damped coupled oscillators task with varying stiffness, k. The LUPI model is more accurate, better calibrated and sharper. All the models are overconfident to some degree, but the training-mode NoPI is by far the least calibrated. Though the LUPI model is sharper, the NoPI model has greater dispersion (spread in uncertainty estimates) which would be preferable if the model were better calibrated.

Table 1: Mean squared error (MSE) and measures of uncertainty quality for the varying-stiffness, varying-damping, and L-V tasks. Models labelled * were evaluated in the training setting, i.e. using the full target set for context and, in the case of the LUPI models, the privileged information, and are included for reference. Lower is better; the better performance in each bracket is indicated in **bold**; units are arbitrary and results should not be compared between the tasks.

	Varying stiffness, $k \sim \mathrm{U}(0.2, 1)$			Varying damping, $c \sim \mathrm{U}(0.5, 2)$		
Model	MSE ↓	Calib. error ↓	Sharp. ↓	MSE ↓	Calib. error ↓	Sharp. ↓
NoPI	1.05 ± 0.05	0.51 ± 0.02	6.88	2.82 ± 0.29	0.84 ± 0.04	2.15
LUPI	$\mathbf{0.93 \pm 0.04}$	$\mathbf{0.47 \pm 0.02}$	6.57	$\mathbf{2.39 \pm 0.09}$	$\mathbf{0.37 \pm 0.02}$	4.71
NoPI*	0.16 ± 0.02	2.69 ± 0.02	1.00	0.56 ± 0.02	1.56 ± 0.03	0.93
LUPI*	$\mathbf{0.06 \pm 0.01}$	$\mathbf{0.91 \pm 0.02}$	1.10	$\mathbf{0.25 \pm 0.01}$	$\mathbf{0.73 \pm 0.03}$	1.18

	L-V, $u_0 \sim \mathrm{U}(0.2, 1), v_0 \sim \mathrm{U}(0.1, 0.5)$		
Model	MSE ↓	Calib. error ↓	Sharp. ↓
NoPI	6.44 ± 0.44	2.19 ± 0.05	2.23
LUPI	$\mathbf{1.82 \pm 0.13}$	$\mathbf{0.90 \pm 0.04}$	3.44
NoPI*	5.24 ± 0.30	2.89 ± 0.04	1.37
LUPI*	$\mathbf{0.73 \pm 0.02}$	$\mathbf{1.23 \pm 0.04}$	1.48

Lotka-Volterra. We investigate modelling a second two dimensional system, the Lotka-Volterra equations (L-V). The populations of 'predator' and 'prey' species, v and u, are governed by

$$\dot{u} = \alpha u - \beta uv \quad ; \quad \dot{v} = \delta uv - \gamma v. \tag{3}$$

In these examples we use the values $\alpha = 2/3, \beta = 4/3, \gamma = 1, \delta = 1$. To produce a range of dynamics we uniformly sample different initial populations $u_0 \sim \mathrm{U}(0.2, 1), v_0 \sim \mathrm{U}(0.1, 0.5)$ as shown in Figure 3. For these equations $V = \delta u - \gamma \ln(u) + \beta v - \alpha \ln(v)$ is a conserved quantity, and is provided as privileged information.

4 Discussion

Training in the LUPI setting produces significantly more accurate and better calibrated models in each task. Table 1 presents numerical results, Figure 4 provides more detail for the varying stiffness task, and further plots are provided in the Appendix. The models are mostly less sharp ($5/6$) but sharpness is a secondary measure to calibration, especially when the sharper model is less accurate, as is the case here. We would also highlight the subtle difference in the kinds of information being incorporated by the models—stiffness and damping are independent variables, whereas V is a conserved quantity that arises from the dynamical system directly—and suggest this is a promising result for the wider applicability of the model. Future work could explore the effects LUPI has on generalisation, recovering estimates of the privileged information at test-time, and, as the model can be applied as-is to any Neural Process, tasks other than dynamics.

REFERENCES

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems 32*, 2018.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural ODEs. In *Advances in Neural Information Processing Systems 32*. 2019.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Mark Harrower and Cynthia A Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Daniel Hernández-Lobato, Viktoriia Sharmanska, Kristian Kersting, Christoph H Lampert, and Novi Quadrianto. Mind the nuisance: Gaussian process classification using privileged noise. *arXiv preprint arXiv:1407.0179*, 2014.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

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

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pp. 2796–2804. PMLR, 2018.

John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8886–8895, 2018.

Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh. Empirical Evaluation of Neural Process Objectives. In *NeurIPS workshop on Bayesian Deep Learning*, 2018.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Nikola Simidjievski, and Pietro Liò. On second order behaviour in augmented neural ODEs. *arXiv preprint arXiv:2006.07220*, 2020.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. Neural ODE processes. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=27acGyyI1BY`.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16(1):2023–2049, 2015.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

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

A OBJECTIVE DERIVATION

At test time we want to (accurately) predict the targets $y_{j\in I_\mathrm{T}}$ at known times $t_{j\in I_T}$ given the context set C, which means maximising $p(y_{j\in I_\mathrm{T}}|t_{j\in I_T}, \mathrm{C})$ during training. Knowing that we want to end up with something similar to the objectives used by Garnelo et al. (2018b) and Norcliffe et al. (2021), i.e. something like

$$\log p(\boldsymbol{y}_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \mathrm{C}) \geq \mathbb{E}_{q(z|\mathrm{T})}\left[\log \frac{q(\boldsymbol{z}|\mathrm{C})}{q(\boldsymbol{z}|\mathrm{T})} + \sum_{i\in I_\mathrm{T}} p(y_i|t_i, \boldsymbol{z})\right],$$

we start with the marginal

$$p(\boldsymbol{y}_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \mathrm{C}) = \int \mathrm{d}\boldsymbol{z}\; p(\boldsymbol{z}|t_{j\in I_\mathrm{T}}, \mathrm{C}) p(y_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \boldsymbol{z}, \mathrm{C}).$$

Noting $p(y_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \boldsymbol{z}, \mathrm{C}) = p(y_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \boldsymbol{z})$ and $p(\boldsymbol{z}|t_{j\in I_\mathrm{T}}, \mathrm{C}) = p(\boldsymbol{z}|\mathrm{C})$, we multiply by $q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi})/q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi}) = 1$ to get

$$\begin{aligned} p(\boldsymbol{y}_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \mathrm{C}) &= \int \mathrm{d}\boldsymbol{z}\; q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi}) \frac{p(\boldsymbol{z}|\mathrm{C})}{q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi})} p(y_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \boldsymbol{z}) \\ &= \mathbb{E}_{q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi})}\left[\frac{p(\boldsymbol{z}|\mathrm{C})}{q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi})} p(y_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \boldsymbol{z})\right]. \end{aligned}$$

As usual, $p(\boldsymbol{z}|\mathrm{C})$ is intractable, and we approximate it with $q'(\boldsymbol{z}|\mathrm{C}) = q(\boldsymbol{z}|\mathrm{C}, \boldsymbol{\pi} \leftarrow \textbf{null})$. Finally, applying Jensen's inequality produces our objective

$$\log p(\boldsymbol{y}_{j\in I_\mathrm{T}}|t_{j\in I_\mathrm{T}}, \mathrm{C}) \geq \mathbb{E}_{q(z|\mathrm{T}, \boldsymbol{\pi})}\left[\log \frac{q'(\boldsymbol{z}|\mathrm{C})}{q(\boldsymbol{z}|\mathrm{T}, \boldsymbol{\pi})} + \sum_{i\in I_\mathrm{T}} \log p(\boldsymbol{y}_i|\boldsymbol{z}, t_i)\right]. \tag{4}$$

This objective is similar to the evidence lower-bound but better reflects the intended test time model behaviour.

B CALIBRATION & SHARPNESS FOR REGRESSION

These descriptions and definitions borrow heavily from Kuleshov et al. (2018) and are included for completeness.

B.1 CALIBRATION

Classification. In the classification setting, we say a forecaster (making a large number of predictions) is calibrated if events that are assigned some probability occur about that frequently. Formally, a forecaster H is calibrated if

$$\frac{\sum_{t=1}^{T} y_t \mathbb{1}\{\mathrm{H}(x_t) = p\}}{\sum_{t=1}^{T} \mathbb{1}\{\mathrm{H}(x_t) = p\}} \to p \;\forall\, p \in [0, 1] \text{ as } T \to \infty,$$

that is, in the long run ($T \to \infty$) predictions assigned probability p ($\mathrm{H}(x_t) = p$) occur ($y_t = 1$) with frequency p.

Regression. In the regression setting, the forecaster outputs a cumulative distribution function F_t targeting y_t. Calibration here means that y_t should fall within a 90% confidence interval approximately 90% of the time. We can formalise this using the quantile function, defined as returning the threshold value that random draws from F_t would fall below p of the time. For quantile function $Q_t(p) = \inf\{y : p \leq F_t(y)\}$, we define calibration to mean

$$\frac{\sum_{t=1}^{T} \mathbb{1}\{y_t \leq Q_t(p)\}}{T} \to p \;\forall\, p \in [0, 1] \text{ as } T \to \infty.$$

That is, in the long-run ($T \to \infty$) targets fall below the quantile function at p ($y_t \leq Q_t(p)$) with frequency p, for any p.

To produce a calibration score, we consider how far the empirical calibration deviates from a perfectly calibrated model. This is measured by computing the empirical frequency at a set of confidence levels $0 \leq p_1 < p_2 < ... < p_m \leq 1$ as

$$\hat{p}_j = \frac{|\{y_t | F_t(y_t) \leq p_j, t = 1, \ldots, T\}|}{T} \tag{5}$$

i.e. how often do the targets fall at a confidence level that is less than the threshold p_j, and computing the score

$$\text{cal}(F_1, y_1, ..., F_t, y_t) = \sum_{j=1}^{m} (p_j - \hat{p}_j)^2 \tag{6}$$

(which is the mean-squared-error of the cumulative histogram of confidence over empirical frequency from the identity.)

B.2 Sharpness

A sharp forecast has confidence intervals that are tightly bound, and the sharpness score can be more straightforwardly defined as the mean variance of the cumulative distribution F_t,

$$\text{sha}(F_1, \ldots, F_T) = \frac{1}{T} \sum_{t=1}^{T} \text{var}(F_t).$$

C Architectural and training details

The context/targets distinction is related to training and evaluation and is separate from the architecture of the model. As such, for this explanation we refer to the set of observations as $\text{X} = \{(t_i^{\text{X}}, \boldsymbol{y}_i^{\text{X}})\}_{i \in I_{\text{X}}}$ indexed by I_{X}, for which we can substitute C or T as appropriate. We conceive of the model as consisting of

1. an observation encoder mapping observations to representations $\boldsymbol{r}_i = f_{\text{obs}}(t_i, \boldsymbol{y}_i)$
2. an aggregator that combines observation representations into a fixed length representation $\boldsymbol{r}_{\text{X}} = \oplus_{i \in I_{\text{X}}}(\boldsymbol{r}_i)$
3. a privileged information encoder mapping the privileged information to a representation $\boldsymbol{r}_\pi = f_\pi(\boldsymbol{\pi})$
4. a second aggregator that combines $\boldsymbol{r}_{\text{X}}$ and $\boldsymbol{r}_\pi$, $\boldsymbol{r} = g(\boldsymbol{r}_{\text{X}}, \boldsymbol{r}_\pi)$, where $\boldsymbol{r}$ and $\boldsymbol{r}_{\text{X}}$ have the same dimensionality
5. a pair of functions, $\mu(\boldsymbol{r}), \sigma(\boldsymbol{r})$, that parametrise the global latent variable $\boldsymbol{z}$ as a function of either $\boldsymbol{r}$ or $\boldsymbol{r}_{\text{X}}$, $\boldsymbol{z} \sim \text{N}(\mu(\boldsymbol{r}), \sigma(\boldsymbol{r}))$
6. a function to initialise the latent ODE state from a sample from the global latent variable, $L(0) = f_{\text{init}}(z')$
7. the neural ODE derivative as a function of time, the instantaneous latent state and the global latent sample, $\dfrac{dL}{dt} = f_{\text{ODE}}(L(t), z', t)$
8. and a decoder that maps from the latent ODE state back to observation space and also depends on the sample from the global latent, $\hat{y}(t) = f_{\text{dec}}(L(t), z')$

as is the fashion, we chose to use neural networks for every parametric function, that is all but the first aggregation $\oplus$. Now we know what the parts are for, the architecture we used is

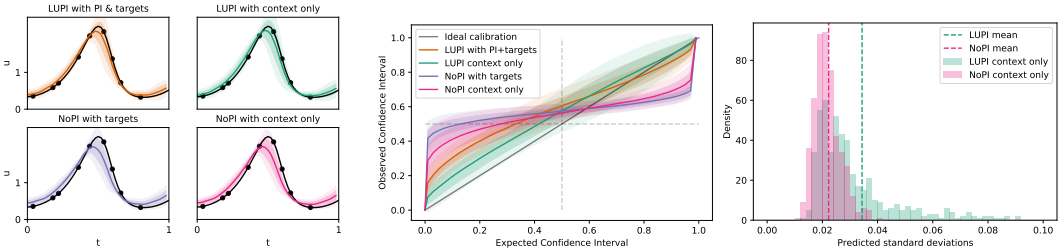
1. f_{obs}: a three-layer MLP with ReLU activations on the hidden layers (not the output) and hidden dimension 16
2. $\oplus$: the mean concatenated with the LogSumExp (a smooth approximation of max)
3. f_π: a three-layer MLP with ReLU activations on the hidden layers (not the output) and hidden dimension 16

4. g: a ResNet with an input-to-output skip connection for $\boldsymbol{r}_{\mathrm{X}}$ and a residual connection consisting of a three-layer MLP with ReLU activations on the hidden layers (not the output) and hidden dimension 16
5. μ, σ: two three-layer MLPs with ReLU activations on the hidden layers (not the output) and hidden dimension 16, with shared weights in the first two hidden layers
6. f_{init}: a three-layer MLP with ReLU activations on the hidden layers (not the output) and hidden dimension 16
7. f_{ODE}: a three-layer MLP with softplus$(x) = \log(1 + \exp(x))$ activations on the hidden layers (not the output) and hidden dimension 16
8. f_{dec}: a three-layer MLP with ReLU activations on the hidden layers (not the output) and hidden dimension 16.

We did not undertake any extensive hyperparameter tuning though we can report that neither model is able to learn if the hidden dimension is set to be small (< 4). We follow the best practices established by Le et al. (2018) for training NPs, and during training use a learned uncertainty estimate rather than resampling. This estimate is produced as an additional output from the decoder.

We use Adam with a learning rate of $0.001 = 10^{-3}$ and otherwise default PyTorch settings, and train for 100 epochs with a training set of 500 examples split 80/20 between train/validation. We do not use early stopping as we found that the models are stable at convergence i.e. they do not diverge when "overtrained". The test sets consist of a further 500 examples (of 100 time steps each) over which the reported evaluation metrics were calculated.

A full model with data generators and Google colab notebook will be released after review.



(a) Example predictions (b) Calibration (c) Sharpness

Figure 5: Various comparisons of trained LUPI and NoPI models on the Lotka-Volterra task with initial conditions, $(u(0), v(0))$. The LUPI model is more accurate and better calibrated, but less sharp. All the models are overconfident to some degree, but the LuPI models are significantly better calibrated.

D LOTKA-VOLTERRA PLOTS

Figure 5 shows additional details for the Lotka-Volterra experiment.