Model Inversion for Spatio-temporal Processes using the Fourier Neural Operator

Dan MacKinlay
CSIRO Data61
dan.mackinlay@data61.csiro.au

Dan Pagendam
CSIRO Data61
dan.pagendam@data61.csiro.au

Petra M. Kuhnert
CSIRO Data61
petra.kuhnert@data61.csiro.au

Tao Cui
Office of Groundwater Impact Assessment
Queensland Government
taocuisunny@gmail.com

David Robertson
CSIRO Land and Water
david.robertson@csiro.au

Sreekanth Janardhanan
CSIRO Land and Water
sreekanth.janardhanan@csiro.au

Abstract

We explore black-box *model inversion* using the Fourier Neural Operator (FNO) of Li et al [4]. The approach learns an emulator of a partial differential equation forward operator from simulated realisations and then infers unobserved system parameters by minimising emulator predictive loss with respect to the observations of the system outputs. Our results suggest that this underdetermined inverse problem is significantly harder than the forward problems or initial condition inference of [4], but by careful regularisation we are able to improve our inference substantially.

1 Introduction

Many environmental problems (e.g. predicting and forecasting the dynamics of ocean currents, weather events, and dynamics of water and solute movement described by surface and groundwater hydrological models) are challenging to model due to complex non-linearities and dynamical processes that characterise the system. Models of these systems can be described by *partial differential equations* (PDEs). In many industrial application, standard solvers are black-box systems which do not provide gradient information. For such systems, inference of high-dimensional unobserved parameters is challenging because gradients must be estimated by finite difference, which rapidly becomes prohibitively expensive.

The challenges have sparked interest in neural network approximation to the PDE solution operators [5, 6, 3]. Classical solution methods for a PDE might involve approximating some continuous domain process into a discrete approximation with complex dependency structure, e.g. stylized in Figure 1a. Naive inference over the graphical model [1] in such systems is forbiddingly complex. A PDE *emulator* can conceptually collapse the complex dependency graph into a smaller more tractable one, e.g. Figure 1b at the cost of requiring us to accept entire functions as inputs and output of the network, i.e. to learn operators on functions. Recent work by Li et al. [4] shows the potential of the FNO for precisely this kind of purpose. The FNO is lauded for a number of features; notably, it is rapid to train, achieves high accuracy with little tuning, and may be interpreted as a resolution-independent mapping between functions on continuous domains without presuming a fixed quantisation of the

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

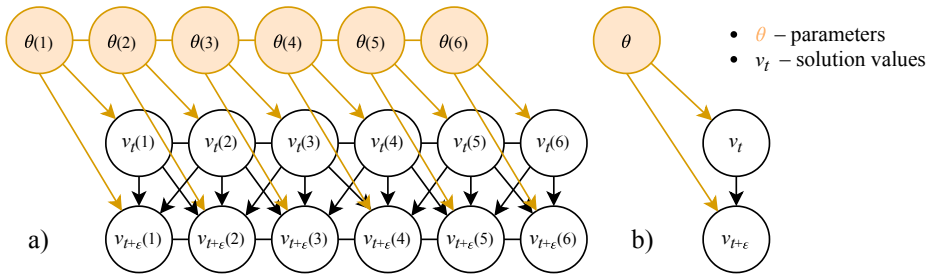

Figure 1: The conditional independence graph of a PDE model with with one spatially-varying parameter, represented as a) as the solution of a discretized PDE solver, evaluated at 6 spatial locations, and b) in terms of continuous operators.

domain. Furthermore, the FNO has recently been shown to satisfy a universal approximation property for such operators [2]. Once learned, the emulator can be used to predict a solution surface with significant speed up over direct PDE solvers, which is differentiable with respect to all the input. Li et al. [4] emphasise the potential application of such operators in downstream tasks such as Bayesian model inversion (i.e., estimating physical system parameters from observations). A further useful property of the FNO to this end is that differentiation of the operator is trivial in all inputs using off-the-shelf backpropagation.

We also follow this thread of inquiry, learning a FNO for the forward operator of the PDE and then computing derivatives with respect to the latent parameters within an optimisation routine to perform inversion, i.e. estimation of the value of an unobserved parameter.

2 Methods

2.1 The Physical System

Consider a PDE, with parameters , that describes the evolution of some vector-valued field, $v(s; tj\ ;\ )\ 2\ V\quad \mathrm{R}^{d_v}$ over two-dimensional space ($s\ =\ (x; y)^{>}; s\ 2\ S\quad \mathrm{R}^2$) and time ($t\ 2\ T\quad \mathrm{R}$). Such a PDE might be used to model an environmental process (e.g. groundwater pressure head across an aquifer ($d_v\ =\ 1$)). Typically, the PDE will exhibit dynamics that are governed by: (i) function spaces of parameters, for example a spatial field $(s);\quad : S\ !\ G$ where $G\quad \mathrm{R}^{d_p}$ encodes physical properties of the spatial environment; and (ii) boundary conditions and/or initial conditions that define constraints on the state of the system at particular locations and/or times. For our purposes, we may think of the boundary conditions as implicit constraints on the solutions which are encoded within the PDE simulations (i.e. the data) used in this study. Both and may vary over time and/or space.

We handle the observations of the PDE in discrete time. For each time t, we assume the instantaneous "slices" of solutions to the PDE are functions v_t belonging to the Banach space $\mathscr{F}$, with $v_t : D\ !\ R$ where $D\ =\ S$ is also a compact set of positive Lebesgue measure and $R\ =\ V$.

A fundamental property of a PDE is the forward operator $M\ :\mathscr{F}\ !\ \mathscr{F}$ which produces the entire solution surface at some future time $t\ +$, given the current state and boundary conditions and parameters: (see also Figure 1 b.)

$$v_{t+}\ j\ ;\ ;v_t\ =\ M\ [v_t j\ ;\ ] \tag{1}$$

The forward operator of a PDE is dependent not only on the current state, but also upon temporal derivatives of the state field. We augment $v_t(s)$ with additional components in $\mathrm{R}^{d_v+d_\partial}$, where $d_@$ is the number of temporal derivatives sufficient to define the PDE. In practice, these derivatives could be approximately encoded by augmenting the state vector to be $(v_t(s)^{>}; v_t\quad (s)^{>}; : : : ; v_{t\ m}\ (s)^{>})^{>}$ for sufficiently small values of and m sufficiently large. Herafter we will hold the boundaries contraints fixed, and they are suppressed.

2.2 The Fourier Neural Operator as a Model Emulator

Figure 2: Samples of the vorticity field w_t from Navier-stokes simulation. Viscosity= 10^{-4}.

We use a simple 2d model of flow, the well-known Navier-Stokes equations as a case study[1], and investigate the behaviour of optimisation-based inference for this problem. In two dimensions the Navier-Stokes system is described by a set of equations that characterise the flow vorticity field (Figure 2). We closely follow the methods of Li et al [4][in fitting a discrete-time FNO to approximate the forward solution of a 2d Navier-Stokes equation on a toroidal domain, by training it to minimise prediction error data generated by a classic PDE solver. However, unlike the original paper, we generate simulations subject to a time-invariant, spatially-varying random forcing parameter $a(s)$ simulated from a Gaussian random field. Our goal is to infer the value of a for new data by numerically solving 3 by gradient descent. Using a classic PDE solver, we generate simulations from this system and use those to train the FNO network. For details of that network we refer the reader to [3].

2.3 Inverse Inference of Parameter Fields by Forward Prediction Error Minimisation

A pragmatic form of inference for a is to choose an estimate $\hat{a}$ to minimises discrepancy d between observations $v_{t+\tau}$ and predictions of those same observations from through $\mathcal{M}$;

$$\hat{a} := \operatorname{argmin}_a \; d\left(\mathcal{M}[v_t|a], v_{t+\tau}|a\right). \tag{2}$$

As a computationally intensive, infinite dimensional problem, this is typically infeasible in a naive implementation, but by emulation we may efficiently approach some approximation of this if we use the emulation $\mathcal{M}_\theta$ in place of the true PDE operator. A realisable inference procedure involves additional approximations; we represent a by some finite parametrisation, $\theta : \mathbb{R}^d \to \mathcal{A}$. We use an approximate mean-squared discrepancy d between predicted solution surfaces at a finite set of sites $\{s_j\}$. If we have chosen a flexible parametrisation for our candidates, this problem is potentially under-specified, so we allow for a regularization term $p(\theta) \geq 0$ and a penalty weight $\lambda \geq 0$ which allow us to bias the solution towards parameter ranges.

We have replaced the original problem with one which, for appropriately chosen $\mathcal{M}_\theta$ and p, is differentiable in its (finite-dimensional) arguments, allowing incremental updating of solutions using the loss gradient. Putting this together, we define estimate $\hat{\theta} = \hat{a}$ via

$$\hat{\theta} := \operatorname{argmin}_\theta \; d\left(\mathcal{M}_\theta[v_t|\theta], v_{t+\tau}|\theta\right) + \lambda p(\theta). \tag{3}$$

Nothing in this set up guarantees that the solution we find this way is unique, or that the optimum matches the true value.

3 Results

For brevity, we defer description of the basic FNO to Li et al [4] and note only where we diverge. All simulation and model parameters are included in released code[2]. All execution times are on a Tesla P100 GPU. We differ in that we have simulated a new dataset of 2d Navier-Stokes simulation, with both initial conditions and latent forcing parameter generated from the same Gaussian Random field,

[1] The Navier-Stokes model is sufficiently common that non-black-box solvers are available; we do not exploit that here.

[2] https://github.com/csiro-mlai/fno_inversion_ml4ps2021

(a) Unregularised, $\ell(\hat{b}) \approx 0.30$ (45s).

(b) With $\lambda = 0.3$, $\ell(\hat{b}) \approx 0.14$ (1m59s).

Figure 3: Estimated latents $\hat{b}$ at convergence.

in the latter scaled by a factor of $\frac{1}{100}$. There are 1000 training and 200 of each validation and testing time series on a (256×256) grid, the generation of which takes 15 hours. Example realisations are shown in Figure 2. Forward model fit terminates by early stopping after 28m.

In the inversion stage we hold the weights of the forward model fixed. For the prediction error minimisation we choose a simple parametrisation for b; specifying its value on a discrete lattice of $K \times K$ points $\{k_i, j_j\}$ spanning D, i.e. $b(k_i, j_j) \approx b_{jk}$. For this problem $K = 256$; and we choose the same lattice upon which the training PDE simulations are evaluated. Writing $\|\cdot\|_2$ for the empirical 2-norm evaluated on the lattice, we define the target predictive divergence to be the same used in the NN training, $d(f_1, f_2) := \|f_1 - f_2\|_2^2$, and we measure the quality in the inverse problem by relative error $\ell(\hat{b}) = \|\hat{b} - b\|_2 / \|b\|_2$ scaled to the magnitude of the estimand; where $\mathbb{E}\|b\|_2 = 0.0112$. In this toy example we know the latent functions b are smooth by construction, so we impose an empirical roughness penalty in b, specifically, the mean squared empirical first order finite differences $\rho(b, \lambda) = \frac{1}{K^2} \sum_{j,k=1}^{K} (b_{j,k} - b_{j,k-1})^2 + (b_{j,k} - b_{j-1,k})^2$. We draw initial conditions $b_{i,j} \sim \mathcal{N}(0, 0.01)$.

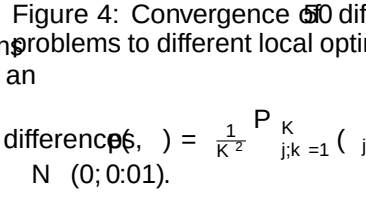

Figure 4: Convergence of 50 different inference problems to different local optima.

Optimisation proceed by first-order gradient descent via a stock Adam optimiser with learning rate 0.0025. For this naïve parameterisation, there are 256^2 parameters so higher order optimisation is not tenable. However, individual optimisation steps here are cheap. In batches of 50 examples, estimand gradient update steps take 2.23 seconds on a Tesla P100 GPU, and convergence is attained with at most 200 Adam iterations. However, the estimation procedure is challenging, requiring careful tuning to achieve convergence. Without regularisation of the latent field, $\lambda = 0$, the results are poor even at optima, with large and systematic errors (see Figure 3a). By contrast, when regularisation is performed, gradient descent may find plausible solutions. Selecting optimisation parameters by grid search to minimise median reconstruction error over a random validation set sample (figure 5), we obtain a improved prediction of the latent field, with estimated relative error 0.28 at $\lambda = 0.3$.

4 Discussion

In practice, inverse modelling of the latent eld using the FNO emulator presents a number of challenges. The FNO by design does not observe physical constraints, such as conservation of matter or energy. Moreover, the inversion problem is underdetermined and we need to be aware of instability in the optimisation process. Careful regularisation of the inferred eld may be needed to avoid physically implausible solutions to the model inversion. When comparing the unregularised optimisation results (Figure 3) to those obtained in the experiments of Li et al. [4], our results suffer from the presence of high frequency artefacts whereas theirs do not. Notwithstanding the estimands are slightly different (latent parameters versus initial conditions), this is an interesting nding and we speculate that the inferred eld in Li et al. [4] avoids these artifacts by taking spatially-pointwise means over many eld realisations which are individually noisy. One avenue for further enquiry is whether the average of an ensemble of estimates removes the need for regularisation of the inferred eld.

Figure 5: vs estimated relative error( ) on held-out set (2h30). Error bars denote 90% range.

Solutions are for the high regularization values are visually plausible, but may be far from ground truth. That is, an optimum of low predictive error may nonetheless differ from the generating parameters (Figure 4). This is compatible with the hypothesis of nding alternative solutions to an underdetermined problem. Choosing parametrisations, regularisation and hyperparameters to improve the quality of inference in this domain is a topic of ongoing research.

A further practical concern is in real-world inference process we may not know the generating process of the true as we do here and in Li et al. Further research to attain more plausible predictions could involve eliciting domain-expert priors, or selecting a nonparametric functional with parameters chosen by a hierarchical Bayes method.

Broader Impact

The modeling of physical systems of hydrological processes, and processes in general, can directly bene t from the ideas presented in this paper. Groundwater models and models of ood inundation for example represent complex systems with partial knowledge and noisy, observational data. Inference about these systems are usually employs hierarchical model to infer parameters or infer intervention effects. However, such inference is non-trivial as some variables are de ned in terms of PDE solution steps, which have complex implementation and demanding compute requirement, and whose solutions are provided by black-box solvers. The FNO provides one approach for conducting an invertible approximate forward simulation of a PDE cheaply. The inversion method presented here has the potential to solve more general inference problems, where the focus is on learning the conditional inverse mapping of outputs to latent features. This generates a wider range of inference methods, scenario simulations and research questions within computational reach of the hydrological community.

Acknowledgments and Disclosure of Funding

We thank Alasdair Tran for helpful feedback on the methods and code.

References

[1] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models : Principles and Techniques*. MIT Press, Cambridge, MA, 2009.

[2] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for Fourier Neural Operators. arXiv:2107.07562 [cs, math], July 2021.

[3] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces. arXiv:2108.08481 [cs, math], September 2021.

[4] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:2010.08895 [cs, math], October 2020.

[5] Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv:1910.03193 [cs, stat], April 2020.

[6] Maziar Raissi, P. Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, February 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately re ect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identi able information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix: Architecture of the FNO

The architecture of a FNO has three basic steps: (i) projection of the input data into a higher-dimensional space through a shallow, feed-forward neural network; (ii) a series of K layers, each of which simultaneously performs both a non-local integral operation and local linear operation then sums these two tensors and passes the result through a non-linear activation function, ; and (iii) a projection to the outputs using a final feed-forward neural network. The structure of the model is outlined in Figure 6.

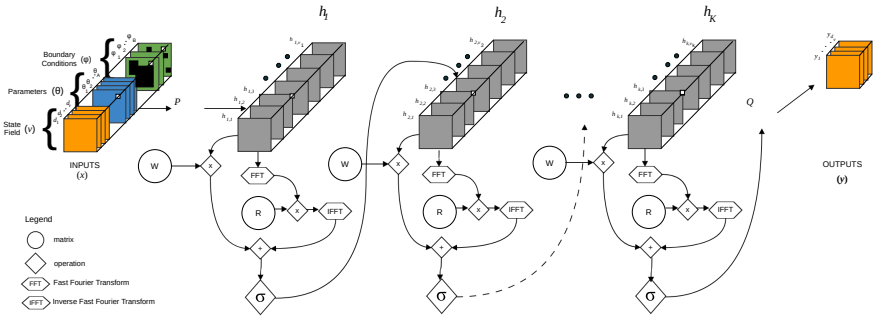

Figure 6: The FNO approximation to operator M .