Meta-Learning Deep Kernels for Latent Force Inference

Jacob D. Moss [1] Felix L. Opolka [1] Jeremy England [2] Pietro Lió [1]

Abstract

Latent force models offer an interpretable alternative to purely data driven inference in dynamical systems. Uncertainty in the output variables is treated by deriving the kernel function of the low-dimensional latent forces directly from the dynamics. However, exact computation of posterior kernel terms is rarely tractable, requiring approximations for complex scenarios such as nonlinear dynamics. In this paper, we overcome these issues by posing the problem as meta-learning a general class of latent force models. By employing a deep kernel and a sensible embedding, we achieve extrapolation from a synthetic dataset to real experimental datasets. Moreover, our model is the first of its kind to scale up to large datasets.

1. Introduction

Differential equations are mathematical models that describe the change of a function with respect to variables such as time. They play a central role in the sciences, providing a grounded method of making historic and future predictions of complex systems. In a machine learning context, this predictive power makes them excellent inductive biases. Latent force models (LFMs) were introduced when Lawrence et al. (2006) modelled a network of genes interacting with a common protein in the biological process of *transcriptional regulation* with ordinary differential equations (ODEs). LFMs are probabilistic models that involve low-dimensional latent forces in the underlying dynamics of a system. This enables handling high-dimensional and nonlinear dynamics in the presence of noise.

LFMs assume a joint Gaussian process prior over the latent forces and observed outputs, whose covariance is determined by the model dynamics as described by the differential equations. Inferring latent forces then requires computing their posterior distribution, which is analytically tractable only for a small set of scenarios. In the remaining cases, which tend to be nonlinear or nonstationary dynamical systems, the posterior requires some approximation. Whilst existing approximations, such as using an ODE solver (Moss et al., 2021), variational inference (Ward et al., 2020), and deep GPs (McDonald & Álvarez, 2021), resolve inference in nonlinear settings, they all suffer from intractabilities for a range of larger scale, real-world scenarios. Indeed, it is often desired to train many LFMs simultaneously in a multi-task setting. For example, in the case of genomics, we often wish to make inferences over thousands of genes or interaction subnetworks.

In this work, we introduce the Deep Kernel Latent Force Model, DKLFM, a novel meta-learning method for multi-task learning under physically-inspired inductive biases. We avoid numerical solving and any variational approximations by instead learning the ODE dynamics in a deep kernel (Wilson et al., 2016). Our framework exploits the representative power of a deep Fourier neural operator (Li et al., 2020) to produce a function embedding for each task. Given an instance consisting only the input mesh, for example time, and the observed functions' embedding, DKLFM infers the associated latent forces with standard Gaussian process conditioning. This makes our approach much faster than training an LFM on individual tasks. DKLFM can model complex nonlinear dynamics and even solves multivariate problems such as partial differential equations which were previously computationally infeasible for large datasets.

2. Preliminaries

Gaussian processes Gaussian processes are stochastic processes commonly used as priors for latent functions in Bayesian machine learning that map inputs $\mathbf{x} \in \mathbb{R}^D$ to predictions $f(\mathbf{x}) \in \mathbb{R}$. A GP prior, $f \sim \mathcal{GP}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$, is described by its mean function $m(\mathbf{x})$ and kernel function $\kappa(\mathbf{x}, \mathbf{x}')$. If the model specifies a Gaussian likelihood for observations y, meaning $y \sim \mathcal{N}(f, \sigma^2)$, the posterior distribution for training data $\mathbf{X}, \mathbf{y}$ is analytically tractable. The marginal likelihood has a closed form expression, enabling gradient-based optimisation over the hyperparameters.

[1]Computer Lab, University of Cambridge Cambridge, UK [2]GSK, 25 Basel St. Petach, Israel. Correspondence to: Jacob Moss <jm2311@cam.ac.uk>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023. Copyright 2023 by the author(s).

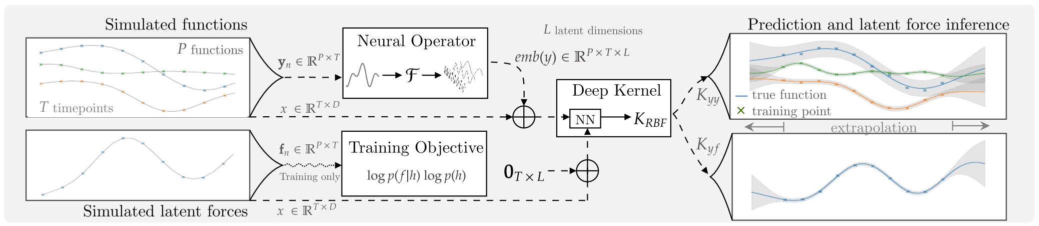

Simulated functions
P functions
T timepoints
$\mathbf{y}_n \in \mathbb{R}^{P \times T}$
Neural Operator
$\mathcal{F}$
L latent dimensions
$emb(y) \in \mathbb{R}^{P \times T \times L}$
$x \in \mathbb{R}^{T \times D}$
Deep Kernel
NN
K_{RBF}
Prediction and latent force inference
K_{yy}
true function
training point
extrapolation
K_{yf}
$\mathbf{f}_n \in \mathbb{R}^{P \times T}$
Training only
Training Objective
$\log p(f|h) \log p(h)$
$\mathbf{0}_{T \times L}$
Simulated latent forces
$x \in \mathbb{R}^{T \times D}$

Figure 1. Schematic of DKLFM. First, a dataset of latent force instances is created by sampling the latent forces and differential equation parameters before solving the forward solution. The simulated functions are embedded by aggregating the output state of a neural operator. A deep kernel is learned to represent the convolution operator of an arbitrary LFM. For training tasks, the model minimises the loss in Equation 1 with access to simulated latent force data. For test tasks, the latent forces are unobserved and inferred via the cross-covariance only, as in a typical LFM scenario. The diagram shows one task, where in reality we would train over batches of tasks.

Deep Kernel Learning Deep kernel learning as presented by Wilson et al. (2016) constitutes an attempt to combine the representation learning capabilities of deep neural networks with the non-parametric nature of Gaussian processes. A neural network is used to map an input $\mathbf{x}$ into a latent space yielding a vector $\mathrm{NN}(\mathbf{x}) \in \mathbb{R}^D$. This representation is then fed into a base kernel $\kappa(\cdot,\cdot)$ (such as an RBF kernel) to yield the covariance between inputs $\kappa(\mathrm{NN}(\mathbf{x}), \mathrm{NN}(\mathbf{x}'))$.

Neural Operators Neural Operators (Kovachki et al., 2021) are neural representations of mathematical operators. They are therefore mesh-invariant and capable of representing function spaces. Our research employs the Fourier variant of these models (Li et al., 2020), which involves learning point-wise transformations in the Fourier domain in order to learn a global convolution in the physical domain.

3. Deep Kernel Latent Force Models

We assume a dataset of N tasks $\{\mathbf{x}_n, \mathbf{y}_n(\mathbf{x}), \mathbf{f}_n(\mathbf{x})\}_{n=0}^N$, where $\mathbf{x}_n \in \mathbb{R}^{T \times D}$ denotes collectively T observed D-dimensional input points and which may be temporal ($D = 1$) or spatio-temporal ($D > 1$). Our output observations, $\mathbf{y}_n \in \mathbb{R}^{P \times T}$ is the set of P function outputs, and $\mathbf{f}_n \in \mathbb{R}^T$ is the latent force at the observed input points. We split the dataset into train and test tasks, where train tasks have access to both the latent forces data and observed outputs, while test tasks only have access to the observed outputs. A model overview is illustrated in Figure 1.

We model the latent force via a latent function h mapping from the inputs x and a task representation $\mathrm{emb}(\mathbf{y}_n)$ to the latent force f. We assign a Gaussian process prior to h and use a Gaussian likelihood for the latent force, i.e.

$$h \sim \mathcal{GP}(m_h(\cdot), \kappa(\cdot,\cdot)) \qquad \text{(prior)}$$
$$f \sim \mathcal{N}(h, \sigma^2) \qquad \text{(likelihood)}$$

where m_h is the mean function of the GP prior and κ is its kernel. Under the LFM assumptions, the distribution of y is implicitly determined via the joint distribution of the latent function and the outputs. More specifically, the latent function outputs at the observed inputs $\mathbf{x}$ and the outputs $\mathbf{y}$ are jointly Gaussian distributed as

$$\mathbf{h}, \mathbf{y} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_h \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{hh} & \mathbf{K}_{hy} \\ \mathbf{K}_{yh} & \mathbf{K}_{yy}\end{bmatrix}\right), \qquad \text{(joint)}$$

where the mean vectors and covariance matrices are obtained by evaluating the mean and kernel function of h as well as a dedicated mean function m_y for the outputs.

As described above, the latent function h operates on both the inputs x and a task representation $\mathrm{emb}(\mathbf{y}_n)$. Its mean and kernel function thus receive a concatenation $\mathbf{z} = \mathrm{emb}(\mathbf{y}_n) \oplus \mathbf{x}$ as input. The mean vectors $\boldsymbol{\mu}_h, \boldsymbol{\mu}_y$ are evaluations of the mean function m_h of the GP prior on h and a dedicated mean function m_y for the outputs, as in

$$\boldsymbol{\mu}_y = m_y\left(\mathrm{emb}(\mathbf{y}_n) \oplus \mathbf{x}\right)$$
$$\mathbf{K}_{yy} = \kappa\left(\mathrm{emb}(\mathbf{y}_n) \oplus \mathbf{x}, \mathrm{emb}(\mathbf{y}_n) \oplus \mathbf{x}\right).$$

The cross-covariance $\mathbf{K}_{yh}$ and covariance $\mathbf{K}_{yy}$ are determined similarly using kernel κ, however the task representation $\mathrm{emb}(\mathbf{y}_n)$ is replaced by zeros in the inputs corresponding to the index set of $\mathbf{f}$. Similarly, the mean $\boldsymbol{\mu}_h$ is the evaluation of a dedicated mean function $\boldsymbol{\mu}_h = m_h(\mathbf{0} \oplus \mathbf{x})$.

In an LFM, the kernel function is derived from a set of differential equations. Here, in order to achieve sufficient expressivity in the general case where the equations cannot be solved, the kernel κ is defined to be a deep kernel (Wilson et al., 2016) with a neural network mapping the inputs to latent representation vectors before feeding them into a base kernel. We use a simple MLP $\mathrm{NN}: \mathbb{R}^{L_1} \to \mathbb{R}^{L_2}$, and an RBF for the base kernel. The mean functions m_h, m_y are constant functions with learned output c.

The trainable parameters in our model are optimised by maximising the marginal likelihood of the observed outputs and latent forces for training tasks. The marginal likelihood

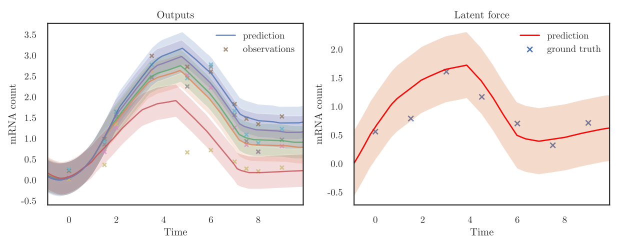


Figure 2. DKLFM infers the protein concentration of transcription factor p53. The ground truth was published by Barenco et al. (2006). The model had been trained on a simulated dataset.

has the closed form:

$$
\begin{aligned}
p(\mathbf{y}, \mathbf{f}) &= \int p(\mathbf{y}, \mathbf{f}, \mathbf{h})\, \mathrm{d}\mathbf{h} \\
&= \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}_{h|y}, \mathbf{K}_{h|y} + \sigma^2 I)\mathcal{N}(\mathbf{y} \mid \boldsymbol{\mu}_y, \mathbf{K}_{yy}). \quad (1)
\end{aligned}
$$

Here, $\boldsymbol{\mu}_{h|y}$ and $\mathbf{K}_{h|y}$ are defined as

$$\boldsymbol{\mu}_{h|y} = \boldsymbol{\mu}_h + \mathbf{K}_{hy}\mathbf{K}_{yy}^{-1}\mathbf{y} \quad (2)$$

$$\mathbf{K}_{h|y} = \mathbf{K}_{hh} - \mathbf{K}_{hy}\mathbf{K}_{yy}^{-1}\mathbf{K}_{yh} \quad (3)$$

At test time, we infer the unobserved latent forces using the conditional GP defined by Equations 2 and 3. The posterior predictive distribution gives the distribution over the outputs, which is the conditional GP of the same form.

We seek an output-specific instance embedding in the input space of the GP, otherwise it cannot distinguish between the P outputs. In keeping with the LFM paradigm, this embedding is over the outputs only, with the expectation that latent forces can be completely determined by the learned dynamics. To maintain the flexibility of GPs, our embedding must be input resolution-invariant. We therefore apply the Fourier neural operator on our observations, taking the mean over all input dimensions to yield our instance embedding. The mesh-invariance of our method enables super-resolution inference; test cases can be at an arbitrary resolution higher than the training data, as we demonstrate in Section 4.2.

4. Experiments

In this section we investigate the performance of DKLFM on ODE- and PDE-based LFMs.

4.1. Transcriptional Regulation

We can model the time derivative of mRNA, $y_j(t)$, of gene j with respect to its latent regulating transcription factor

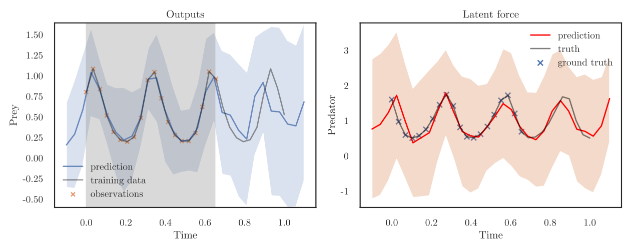


Figure 3. DKLFM infers the predator-prey relationship in a Lotka-Volterra setup. The model has only been trained within a time range denoted by the grey shaded region.

protein $f_i(t)$ (Barenco et al., 2006; Lawrence et al., 2006):

$$\frac{\mathrm{d}y_j(t)}{\mathrm{d}t} = \overbrace{b_j}^{\text{basal rate}} + s_j \overbrace{G(f(t))}^{\text{response}} - \overbrace{d_j y_j(t)}^{\text{decay term}}, \quad (4)$$

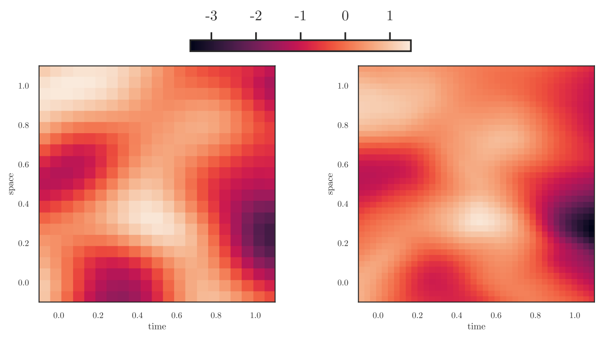
where b_j is the base transcription rate of gene j, s_j is the sensitivity, or a response factor to the transcription factors. In this case, we set G to the softplus function to enforce positivity: $G(f(t)) = \log(1+\exp(f(t)))$. This nonlinearity renders an exact solution intractable.

We generate a synthetic dataset of 500 instances by sampling parameters for Equation 4 from an empirical distribution of parameters, which were learnt by running *Alfi* (Moss et al., 2021) on the p53 network of genes experimentally measured by Barenco et al. (2006). Next, the latent force is sampled from a GP prior with RBF kernel, and the ODE is solved yielding a single instance. Gaussian-distributed random noise is added to the latent forces.

In Figure 2, we demonstrate how DKLFM is able to extrapolate from a purely simulated dataset to a real microarray dataset from (Barenco et al., 2006). The data pertains to transcript counts for five targets of the transcription factor p53 over seven timepoints.We show our inferred transcription factor concentration alongside the unobserved ground truth. We also show intra-task extrapolation in Figure 3 for Lotka-Volterra instances, where predator-prey dynamics are governed by $\frac{\mathrm{d}u(t)}{\mathrm{d}t} = \alpha u(t) - \beta u(t) f(t)$, with growth rate α, decay rate β, and predator count (latent force) $f(t)$. Using a periodic base kernel, we infer on an expanded input domain compared to the training data.

4.2. Reaction Diffusion Equations

PDE-based LFMs are more complex; solving these numerically involves a mesh–therefore suffering the curse of dimensionality. Here, we demonstrate that DKLFM can be used to fit reaction-diffusion equations with a moderately-sized dataset of 500 low-resolution instances. We use the example of *embryogenesis*, where spatiotemporal RNA expression,



True Latent Force

Predicted Latent Force (super-resolution)

Figure 4. DKLFM infers the unobserved latent force for the reaction diffusion experiment. We trained on a 21×21 spatiotemporal grid and tested on a 40×40 grid to illustrate super-resolution.

Table 1. Comparison to baseline models. For the *DKLFM*, we train on a dataset of 256 and 500 instances for the ODE and PDE settings respectively. We average the MSEs over 20 instances. Models were run on an NVIDIA GeForce RTX 4090 GPU. The times corresponds to the inference time per-instance.

Model	**Latent MSE**	**Output MSE**	**Time (s)**
	Transcriptional Regulation ODE		
Alfi	0.187	**0.117**	3.27
DeepLFM	-	0.332	12.6
DKLFM	**0.116**	0.201	**0.0118**
	Reaction Diffusion PDE		
Alfi	**0.08859**	**0.0215**	> 10m
DeepLFM	-	0.356	96.7
DKLFM	0.633	0.720	**0.0523**

$y(x,t)$, is modeled by the PDE (López-Lopera et al., 2019):

$$\frac{\partial y(x,t)}{\partial t} = Su(x,t) - \lambda y(x,t) + D\frac{\partial^2 y(x,t)}{\partial x^2}, \quad (5)$$

where S is the production rate, $u(x,t)$ is the 2D latent force, λ is the decay rate and D is the diffusion rate.

We simulate a dataset from Equation 5 by implementing the Green's function approximation (López-Lopera et al., 2019). We then sampled parameters empirically from the Drosophila gap gene dataset from Becker et al. (2013), and latent forces from a GP prior with varying lengthscales.

In Figure 4, we demonstrate the super-resolution performance of inferring latent forces for a test task. Moreover, DKLFM is orders of magnitude faster than comparable models at inference time, as shown in Table 1. Since DKLFM solves many LFMs simultaneously rather than a single instance, we plot the mean squared error as a function of dataset size in Figure 5. This demonstrates that with 256

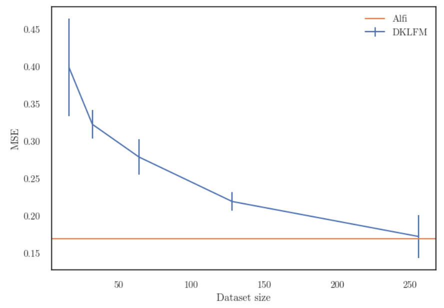


Figure 5. MSE for 20 test ODE tasks plotted against training dataset size, illustrating the point at which it becomes economical to use DKLFM.

instances, we achieve similar performance to training individual LFMs with *Alfi*, a numerical approximation. This dataset size is considerably less than a typical experimental scenario involving thousands of genes.

Related work Ward et al. (2020) employ a state-space model for approximating the posterior of a non-linear LFM. The authors use autoregressive flows to construct a joint density of the state using variational inference. This approach suffers from an over-confidence prevalent in such black-box variational approaches. Moss et al. (2021) avoided the complex derivations of kernel functions by sampling from the GP prior and numerically solving with an ODE or PDE solver. The use of a solver renders this approach computationally intensive and inappropriate for a multi-task setting. McDonald & Álvarez (2021) tackles a similar problem of handling highly non-linear and non-stationary dynamics. This is achieved by constructing a deep GP (Damianou & Lawrence, 2013) with a convolution of an RBF kernel with the Green's function of the ODE at each layer. This deep representation makes the model applicable to nonlinear dynamics, but not directly to PDEs or multi-task settings.

5. Conclusion

We have introduced DKLFM, a novel meta-learning framework for LFMs. Combining the deep kernel's expressivity with the neural operator's ability to embed functions, we enable fast inference of latent forces by conditioning only on observed output data. DKLFM is the first *exact inference* framework for learning the solution operator for any nonlinear LFM. We achieve this by learning the cross-covariance between latent forces and observations corresponding to the differential equation from a simulated set of instances. DKLFM explicitly treats the uncertainty in the latent forces and output functions. Active learning is therefore possible by query input points where the output function or latent forces have high uncertainty.

The performance and speed is limited by generation of simulated dataset–easily parallelised–and the inversion of an $T \times T$ matrix which has computational complexity of $O(T^2)$ or $O(TM^2)$ if an inducing points are used. Once trained on a simulated dataset, inference for unseen instances is immediate compared to a standard LFM, which requires an optimisation loop for determining kernel parameters.

Broader impact

We hope to see these models being used and distributed like large language models (Shanahan, 2022), where a user can use a pretrained DKLFM to make latent force inferences on their dataset, with or without fine-tuning. This would be extremely useful in the biological sciences, where the experimental datasets are quite small and noisy. However, the creation of a dataset–especially where it involves solving a PDE–can be computationally time-consuming and therefore consume substantial computational resources. As with all machine learning models of this kind, care must be taken to ensure that bias in the training data is considered. There could be sensitive applications (e.g. medical time-series) where blind faith in the inferred forces can lead to potentially detrimental effects on the patient.

References

Barenco, M., Tomescu, D., Brewer, D., Callard, R., Stark, J., and Hubank, M. Ranked prediction of p53 targets using hidden variable dynamic modeling. *Genome biology*, 7: 1–18, 2006.

Becker, K., Balsa-Canto, E., Cicin-Sain, D., Hoermann, A., Janssens, H., Banga, J. R., and Jaeger, J. Reverse-engineering post-transcriptional regulation of gap genes in drosophila melanogaster. *PLoS computational biology*, 9(10):e1003281, 2013.

Damianou, A. and Lawrence, N. D. Deep gaussian processes. In *Artificial intelligence and statistics*, pp. 207–215. PMLR, 2013.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Lawrence, N., Sanguinetti, G., and Rattray, M. Modelling transcriptional regulation using gaussian processes. *Advances in Neural Information Processing Systems*, 19, 2006.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

López-Lopera, A. F., Durrande, N., and Alvarez, M. A. Physically-inspired gaussian process models for post-transcriptional regulation in drosophila. *IEEE/ACM transactions on computational biology and bioinformatics*, 18 (2):656–666, 2019.

McDonald, T. and Álvarez, M. Compositional modeling of nonlinear dynamical systems with ode-based random features. *Advances in Neural Information Processing Systems*, 34:13809–13819, 2021.

Moss, J. D., Opolka, F. L., Dumitrascu, B., and Lió, P. Approximate latent force model inference. *arXiv preprint arXiv:2109.11851*, 2021.

Shanahan, M. Talking about large language models. *arXiv preprint arXiv:2212.03551*, 2022.

Ward, W., Ryder, T., Prangle, D., and Alvarez, M. Black-box inference for non-linear latent force models. In *International Conference on Artificial Intelligence and Statistics*, pp. 3088–3098. PMLR, 2020.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, 2016.