Deep Kernel Learning for Online Prediction of Ocean Current Vector Fields

Matthew Dim
Virginia Tech
Blacksburg, VA, United States
matthewdim@vt.edu

Abstract

Real-time navigation and mapping of spatio-temporal spaces present computational challenges for Autonomous Underwater Vehicles (AUV) operating in dynamic marine environments. This paper addresses these challenges by applying Deep Kernel Learning (DKL) with multi-output Gaussian processes (MOGP). We propose a novel algorithmic approach that leverages deep network-based feature extraction to transform complex oceanic vector fields from a wide-range of stations from the United States into interpretable probabilistic representations, enabling more robust and adaptive prediction mechanisms for the physics of ocean data. Specifically we utilize feed-forward neural network (FNN) and long short-term memory (LSTM) architectures to learn latent representations for the radial basis function (RBF) kernel. Compared to conventional Gaussian Process (GP) Regression forecasting algorithms, our proposed methodology demonstrates substantially improved accuracy, stability, and predictive certainty for ocean current vectors measured by NOAA stations around US coasts. Utilizing a comprehensive dataset from the National Oceanic and Atmospheric Administration (NOAA), we split the data into distinct training and testing sets to validate our approach. The LSTM-based multi-output Gaussian process (LSTM + MOGP) model achieves an averaged Normalized Root Mean Squared Error (RMSE) of 0.0703, with a standard deviation of 0.069 across all test sites, significantly outperforming existing methodologies. Comparative analysis with state-of-the-art techniques demonstrates the models' high-level accuracy, particularly in generalization scenarios where models are deployed in previously unseen oceanic regions.

Keywords

Deep Kernel Learning, Gaussian Processes, Vector Fields, Kernel Methods, Machine Learning

ACM Reference Format:
Matthew Dim. 2024. Deep Kernel Learning for Online Prediction of Ocean Current Vector Fields. In *Proceedings of XXXX (XXXX'24).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/xxxxxx.xxxxxx

XXXX'24, X, 2024, XXXX Conference

ACM ISBN 978-1-4503-XXXX-X/24/03
https://doi.org/10.1145/xxxxxx.xxxxxx

1 Introduction

Autonomous Underwater Vehicles (AUVs) operate in complex, data-rich marine environments, where accurate real-time predictions of ocean currents are essential for efficient path planning and safe navigation. While regional physics-based models, such as Harmonic Method predictions, provide robust forecasts, their high computational cost and dependence on extensive domain knowledge hinder their use onboard AUVs for real-time applications [10]. Data-driven methods, including deep learning architectures such as Long Short-Term Memory (LSTM) networks, have demonstrated the ability to learn temporal patterns of ocean currents from in-situ measurements[7]. However, their predictive capabilities have largely been validated in regions used for training, leaving open questions about their ability to generalize to new, unseen domains[12, 13].

Gaussian Processes (GPs) offer a heuristic based, Bayesian approach for modeling spatial data and quantifying uncertainty [11]. Their flexibility and nonparametric nature make them attractive for capturing the variability of ocean currents; however, the cubic complexity in data size and the need for domain-specific kernel engineering limit their use in large-scale, real-time scenarios. To address these challenges, others have developed methods such as sparse GP approximations, employing pseudo-inputs or subsets of the training data to reduce computational cost [5, 14]. Complementary efforts such as local ensemble GP approximations and mixture-of-experts models attempt to partition the spatial domain into manageable subsets, alleviating computational burdens while maintaining accuracy [2, 4]. Hybrid methods suffer from complex adaption of local kernel hyperparameters relative to each induced set increasing computational cost. It is our goal to address these issues of learning in an online setting while maintaining accurate predictions.

Deep Kernel Learning (DKL), integrates neural networks with Gaussian Processes to learn data-driven kernel representations [17, 18]. By employing deep architectures, DKL enables models to capture complex, high-dimensional input relationships beyond the scope of conventional kernels. Early applications of LSTM networks for current prediction [7] suggest that incorporating recurrent structures to extract temporal features into a DKL setting could lead to significant performance gains, improved stability, and enhanced generalization across diverse marine environments. Further efforts have been made to learn a temporal kernel structure using deep learning methods by optimizing the hyperparameters using trained data[19, 20]. These efforts relate more towards the objective loss function, but serve as alternative directions that could be made for time-series forcasting. Efforts towards using Variational Auto-encoders (VAE) towards learning latent representations of the space

[15]; however, these generative models are not suited for reliable physics representations. Lastly, efforts have been made by [1] to learn recurrent kernels by forming a latent space from an LSTM and optimizing using marginal likelihood for a closed-form solution. This approach justifies the usefulness of a baysian LSTM model used in context of a GP. We seek to build upon this frame-work to develop an LSTM to learn the recurrent latent space of a vector-valued GP or Multi-output GP (MOGP).

In this paper, we introduce an online Gaussian Process regression algorithm that leverages Deep Kernel Learning for vector-valued ocean current prediction. The method (1) integrates feed-forward neural networks (FNN) and LSTMs in a DKL context combining deep learning with gaussian process regression, (2) learn latent representations that inform a scalable, nonstationary kernel for the GP capturing a physics informed latent space for from ocean data, (3) extends DKL to a vector-valued prediction for environmental mapping providing an easy to understand bridge between conventional machine learning and Bayesian statistics. This work establishes a novel pathway for seamless integration of physics-informed learning compared with existing methods. By leveraging deep architectures and nonparametric Bayesian inference we show that ocean current mapping at local stations can be interpolated with high-accuracy.

2 Methodology

2.1 Gaussian Processes

Gaussian Processes (GPs) are a nonparametric, Bayesian framework for modeling distributions over functions, providing not only predictions but also uncertainties[11]. Given a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{X}$ represents input features and $\mathbf{y}$ represents corresponding outputs, a GP assumes that any finite set of function values $f(\mathbf{x})$ follows a joint Gaussian distribution:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot)) \tag{1}$$

Where, $\mu(\mathbf{x})$ is the mean function, often set to zero for simplicity, and $k(\mathbf{x}, \mathbf{x}')$ is the covariance function (or kernel), which encodes prior assumptions about the smoothness and structure of the function.

The GP posterior, given training data $\mathcal{D}$ and new test points X_* the posterior distribution of the function value at x_* is given by:

$$f_*|\mathbf{X}, \mathbf{y}, X_* \sim \mathcal{N}(\mu_*, \Sigma_*) \tag{2}$$

and is computed as:

$$\mu(\mathbf{x}_*) = \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{3}$$

$$\mathrm{Var}(f(\mathbf{x}_*)) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_* \tag{4}$$

where:

$\mathbf{K}$ is the covariance matrix between training points.
$\mathbf{k}_*$ is the covariance vector between the test point $\mathbf{x}_*$ and the training points.
σ_n^2 represents Gaussian noise variance.

2.2 Kernel Functions

The kernel function $k(\mathbf{x}, \mathbf{x}')$ is a critical component of Gaussian Processes, defining the covariance between function values at different input points $\mathbf{x}$ and $\mathbf{x}'$. There are many different kernel functions that can represent infinite latent spaces often referred to as Reproducing Kernel Hilber Space (RKHS), where $\mathcal{H}$ is defined as a hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, such that there exists a reproducing kernel $k : X \times X \to \mathbb{R}$. In this case, we choose to use the RBF kernel as a baseline for it's properties of smoothing and capturing complex functions using minimal hyper-parameters. It encodes assumptions about the smoothness, periodicity, and other properties of the target function. A commonly used kernel is the Radial Basis Function (RBF) kernel, also known as the squared exponential kernel, given by:

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right) \tag{5}$$

where:

- σ_f^2 is the signal variance, controlling the vertical variation of the function.
- l is the length scale, determining how quickly the function varies with input changes.

The hyperparameters $\theta = \{\sigma_f^2, l\}$ are learned by maximizing the log marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K} + \sigma_n^2 \mathbf{I}| - \frac{n}{2} \log 2\pi \tag{6}$$

where $\mathbf{K}$ is the covariance matrix formed using $k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}')$ defined in equation 5.

The RBF kernel is widely used due to its smoothness properties, making it ideal for modeling functions that vary smoothly over the input space. However, its expressive power can be limited in high-dimensional or complex data spaces, motivating extensions such as Deep Kernel Learning, discussed in later sections.

2.3 Multi-output Gaussian Processes: Intrinsic Coregionalization Model (ICM)

Multi-output Gaussian Processes (MOGPs) extend standard GPs to model vector-valued outputs $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_D(\mathbf{x})]^\top$, where D is the number of output dimensions, mapping our input $\mathbb{R}^M \to \mathbb{R}^D$ [2, 3, 16]. Unlike independent GPs, which model each output independently, MOGPs exploit correlations between outputs to improve predictive performance and computational efficiency.

The Intrinsic Coregionalization Model assumes that the outputs are linearly correlated through a shared kernel function and a coregionalization matrix $\mathbf{B}$:

$$k_{\mathrm{ICM}}(\mathbf{x}, \mathbf{x}'; d, d') = k_{\mathrm{shared}}(\mathbf{x}, \mathbf{x}') \cdot B_{d,d'} \tag{7}$$

where:

$k_{\mathrm{shared}}(\mathbf{x}, \mathbf{x}')$ is a shared kernel function (e.g., RBF or Matern).
$B_{d,d'}$ is an element of the positive semi-definite coregionalization matrix $\mathbf{B}$, capturing the correlations between outputs d and d'.

For an input set $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, the joint covariance matrix for all outputs can be expressed as:

$$\mathbf{K}_{\mathrm{MOGP}} = \mathbf{B} \otimes \mathbf{K}_{\mathrm{shared}} \tag{8}$$

where $\otimes$ denotes the Kronecker product, and $\mathbf{K}_{\mathrm{shared}}$ is the covariance matrix formed using $k_{\mathrm{shared}}(\mathbf{x}, \mathbf{x}')$. By encoding these relationships in the coregionalization matrix $\mathbf{B}$, the MOGP efficiently

learns dependencies between these outputs, improving predictions for both quantities, even in regions with limited training data.

Given training data $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, where $\mathbf{Y}$ contains multi-dimensional outputs, the predictive posterior mean and variance for a test point $\mathbf{x}_*$ are the exact same from equations 3 and 4.

2.4 LSTM Models

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to capture temporal dependencies in sequential data[6]. Unlike traditional RNNs, LSTMs mitigate the vanishing gradient problem by introducing a gating mechanism that enables the model to retain long-term dependencies. This property makes LSTMs particularly well-suited for time-series data, such as ocean current predictions. An LSTM cell consists of three primary gates: the input gate, forget gate, and output gate. These gates control the flow of information through the network:

- The *input gate* determines how much of the current input to incorporate into the cell state.
- The *forget gate* decides how much of the previous cell state to retain.
- The *output gate* regulates the information passed to the next layer or output.

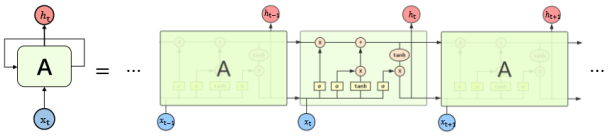


Figure 1: LSTM Model Architecture

In this paper, LSTM networks are employed to capture the temporal dynamics of ocean currents. Each station's time-series data (e.g., speed and direction) is treated as a sequence, where each timestep corresponds to a 5-6 minute interval. The sequence was chosen to be 5 total timesteps from the past $(x_1^*, ..., x_k^*)$ to motivate better prediction accuracy for 30 minutes into the future. The LSTM processes this data to learn patterns over time where outputs are used as feature representations for downstream tasks, including providing input to the Deep Kernel Learning (DKL) model for learning a nonstationary kernel.

2.5 Deep Kernel Learning

Deep Kernel Learning (DKL) integrates the representational power of deep neural networks with the probabilistic framework of Gaussian Processes (GPs)[17]. By learning data-driven feature representations, DKL enables the construction of flexible and scalable kernels that capture complex relationships in the input space. The core idea is to use a neural network $\phi(\mathbf{x}; \Theta)$ to map input data $\mathbf{x}$ into a latent feature space, where a Gaussian Process operates with a kernel defined as:

$$k_{\text{DKL}}(\mathbf{x}, \mathbf{x}') = k_{\text{GP}}(\phi(\mathbf{x}; \Theta), \phi(\mathbf{x}'; \Theta)) \tag{9}$$

Here, k_{GP} can be any standard GP kernel, such as the Radial Basis Function (RBF), applied to the learned feature representations.

Our implementation of DKL builds on [17], incorporating an LSTM network as the feature extractor. The architecture consists of:

- An LSTM network $\phi_{\text{LSTM}}(\mathbf{x}; \Theta_{\text{LSTM}})$ to capture temporal dependencies in the input time-series data learning a feature space represented by $\phi(\mathbf{x}) \in \mathbb{R}^5$.
- A feed-forward neural network (FNN) $\phi_{\text{FNN}}(\cdot; \Theta_{\text{FNN}})$ to further transform the LSTM output into a compact latent feature space.
- A Multi-output Gaussian Process (MOGP) that operates on the learned feature representations to model vector-valued outputs (e.g., current speed and direction).

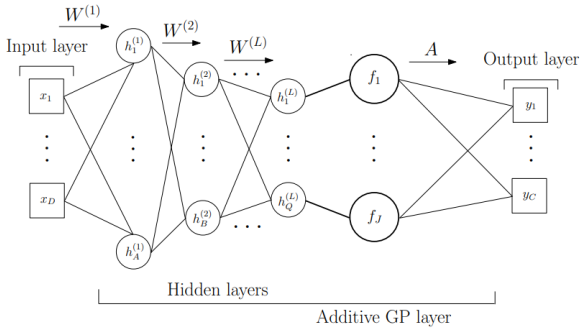


Figure 2: Deep Kernel Learning (DKL) architecture for vector-valued functions. A convolutional neural network (CNN) extracts features from the input space, transforming vector field data into a lower-dimensional feature representation. These features are passed into a Gaussian Process (GP) to compute the kernel for multi-output predictions

The overall kernel for the MOGP in our DKL framework is defined as:

$$k_{\text{DKL-MOGP}}(\mathbf{x}, \mathbf{x}') = k_{\text{shared}}(\phi_{\text{LSTM}}(\mathbf{x}), \phi_{\text{LSTM}}(\mathbf{x}')) \cdot B_{d,d'} \tag{10}$$

where k_{shared} is the shared kernel function (e.g., RBF), and $B_{d,d'}$ is the coregionalization matrix for vector-valued outputs.

The DKL model is trained end-to-end by maximizing the log marginal likelihood (LML) of the MOGP, while the LSTM and FNN parameters are optimized jointly to minimize the LML loss defined in equation 6.

Optimization is performed using ADAM method for stochastic optimization [8], which scales to large datasets by approximating the posterior with a set of inducing points. The LSTM and FNN parameters (Θ_{LSTM} and Θ_{FNN}) are updated jointly with the GP hyperparameters to maximize the LML.

By integrating LSTM networks into the DKL framework, our model offers (1) temporal feature extraction though multi-scale temporal dependencies from the input data captured by the LSTM, such as diurnal and semi-diurnal tidal patterns, (2) vector-valued output modeling using the MOGP kernel leveraging shared structure between outputs (speed and direction).

2.6 Dataset

The dataset for this paper is sourced from the historic stations dataset maintained by the National Oceanic and Atmospheric Administration (NOAA) [9]. Ocean current data from 260 stations were initially downloaded; however, only 76 stations were selected after data quality checks, and the models wer ultimately trained on data from 65 stations. Additionally, the location of stations were taken into account to train global representations of ocean data along the coast of the entire United States. These stations span depths ranging from 10 to 150 meters, with Acoustic Doppler Current Profilers (ADCPs) used to measure ocean current speeds and directions.

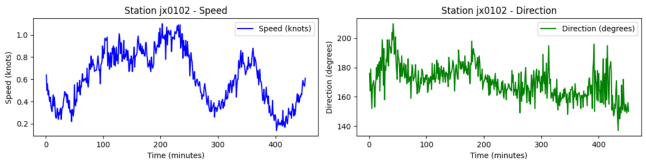


Figure 3: Florida coast station jx0102 current data

Given the task of online prediction, only one month of data was used for training at each station; our goal is for local-time predictions not far into the future which cater to the nature of GP regression and AUV navigation requirements. Time series data was processed to represent each timestep as an index corresponding to 5–6 minutes of real time. Any missing data points within gaps of less than one hour were linearly interpolated, while stations with significant data gaps or anomalies in current direction were excluded. This ensured a high-quality dataset free from artifacts that could compromise model performance.

The ocean current data consist of two key outputs: speed and direction. Both variables were normalized to a scale of 0–1 during preprocessing to ensure balanced contributions to the multi-output Gaussian process (GP) model. Normalization also prevented the GP model from overemphasizing one variable, such as current direction, due to differing magnitudes. By limiting the training set to one month per station, we capture sufficient temporal variability while maintaining computational efficiency. Test data is split from the final 10 percent of the dataset for prediction in local spaces. For context, an AUV looking to predict a path should only be concerned with accurate information within a few minutes as opposed to highly variable information the next day. The dataset is designed to learn complex representations in an online setting to predict and assign probability to time-steps hours in the future.

3 Evaluation

We evaluate the performance of DKL-MOGP over the dataset defined in section 2.6 over 5 test stations. Data is pruned to 200 time-steps or 1000 minute representation for prediction in a setting where we try to predict 20 time-steps or 120 minutes ahead. Additionally, we prune the data to 50 time-steps for focus on learning in an online setting where we are only trying to predict 30 minutes ahead. The second application may be more relevant to AUV navigation given sparse data and need for high accuracy mapping. The loss function is defined as Marginal Log-likelihood defined in equation 6, the adam optimizer is used for traversing the gradient over 1000 and 500 epochs respectively for each test set, and learning rate is set to $\alpha = 0.01$. We use GPyTorch to implement GP and DKL functions for testing and organization. Models are trained using A100 GPUs from Google Cloud services.

To analyze the performance of the models, we compute the Root Mean Squared Error (RMSE) which allows for comparison of model suitability to the data.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{11}$$

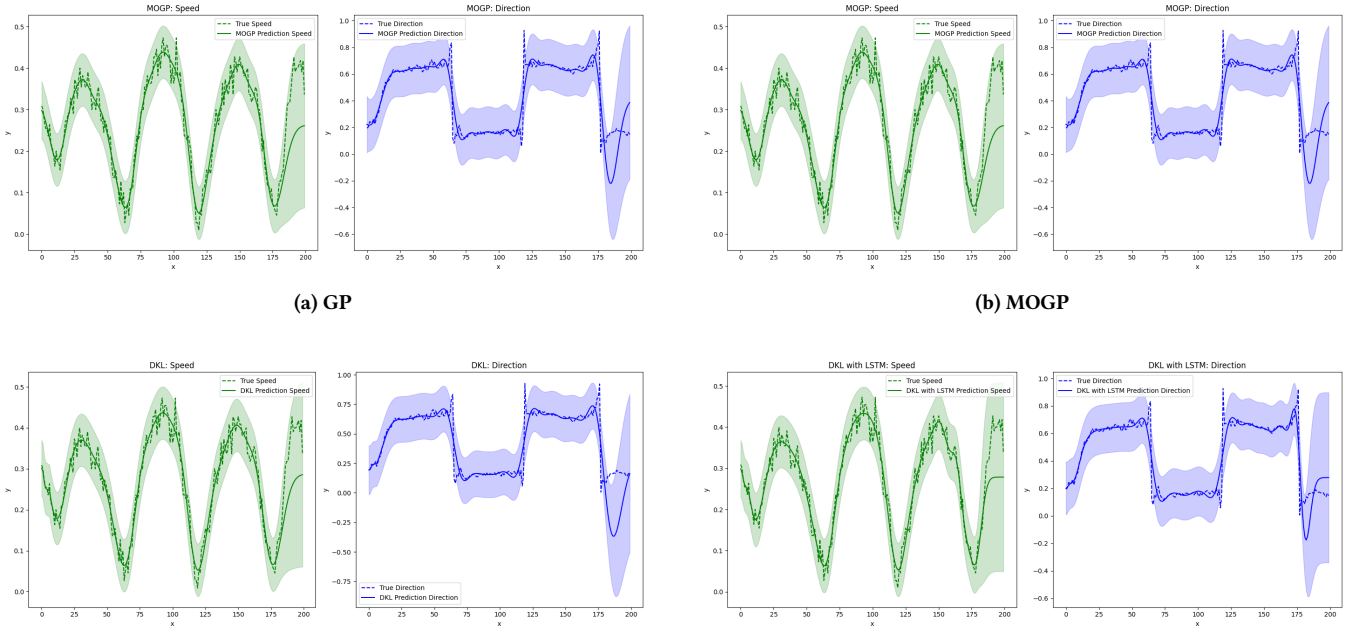
Additionally, we use the inherent probabilistic nature of GP predictions to visualize the variance in our error to evaluate model performance.

Station	Model	RMSE	Std. Dev	Time (s)
CHB0302	GP	0.084324	0.083217	8.180892
	MOGP	0.077274	0.077096	8.140150
	DKL + NN	0.074214	0.073646	9.210708
	DKL + LSTM	**0.068812**	**0.068669**	9.959922
COI0420	GP	0.090870	0.090455	7.999387
	MOGP	0.091506	0.091086	8.396413
	DKL + NN	0.074735	0.074076	9.212672
	DKL + LSTM	**0.073249**	**0.072303**	10.044534
FLK1301	GP	0.064954	0.064128	8.030176
	MOGP	0.060029	0.059495	8.181824
	DKL + NN	0.062793	0.061890	9.062302
	DKL + LSTM	**0.050370**	**0.050135**	9.920693
HAI1102	GP	0.150849	0.148256	7.959627
	MOGP	0.138798	0.137313	8.103623
	DKL + NN	0.146477	0.145015	9.068761
	DKL + LSTM	**0.129532**	**0.128648**	9.906644
JOP1101	GP	0.065782	0.064656	7.966617
	MOGP	0.041820	0.040895	8.159753
	DKL + NN	0.066721	0.065619	9.054097
	DKL + LSTM	**0.039526**	**0.039322**	9.909388

Table 1: Models trained over 20 hours of data for 2 hours of prediction - 1000 epochs and 0.01 learning rate

The first test set was pruned to 1000 minutes, or a length of 20 hours, and trained over 1000 epochs for best model fit. Given these constraints, we found from Table 1 that LSTM + DKL is able to out-perform all other models in RMSE and error measured through standard deviation. The model took the longest to train taking on average 1.8 s longer than the standard GP model which is to be expected.

When training in context of only 5 hours and prediction of 5 timesteps ahead (or 30 minutes) from Table 2, models performed in a manner that is more expected in an online setting. For example, times for non-stationary interpolation for each model took within the ranges of 4-5 seconds while still maintaining low error. In training, the DKL models, less than 1 second extra was spend extracting latent space features. When training with less data, there are instances where DKL + LSTM is not able to outperform other models, such as stations COI0420 and FKL1310; however, margins



(a) GP **(b) MOGP**

(c) DKL + NN **(d) DKL + LSTM**

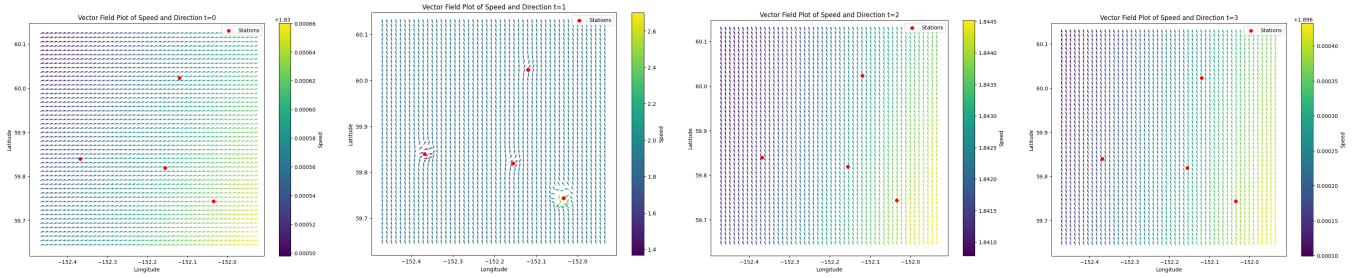
Figure 4: Predictive mean and 95% confidence regions for station CHB0302: GP, MOGP (top), and DKL + NN, DKL + LSTM (bottom).

Station	Model	RMSE	Std. Dev	Time (s)
CHB0302	GP	0.043374	0.043133	4.362759
	MOGP	0.027278	0.027246	4.194437
	DKL + NN	0.042624	0.042614	4.541619
	DKL + LSTM	**0.024672**	**0.024670**	4.797211
COI0420	GP	0.109790	0.109532	3.972780
	MOGP	0.109358	0.109284	4.121579
	DKL + NN	**0.090556**	**0.089548**	4.622804
	DKL + LSTM	0.118907	0.118520	4.796507
FLK1301	GP	0.088723	0.088721	3.944323
	MOGP	**0.086216**	**0.086214**	3.908787
	DKL + NN	0.094722	0.094150	4.427830
	DKL + LSTM	0.098436	0.097509	4.805147
HAI1102	GP	0.114350	0.113642	3.852692
	MOGP	0.107893	0.107503	3.922256
	DKL + NN	0.112007	0.111571	4.471146
	DKL + LSTM	**0.104361**	**0.104360**	4.775656
JOP1101	GP	0.063329	0.061685	3.924130
	MOGP	0.037627	0.036646	3.942790
	DKL + NN	0.105352	0.104013	4.414643
	DKL + LSTM	**0.034197**	**0.034115**	4.790038

Table 2: Models trained over 5 hours of data for 0.5 hours of prediction - 500 epochs and 0.01 learning rate

are close enough that we can favor other instances of prediction. In stations CHB0302 and JOP1101 the RMSE is nearly half of what was trained for the GP and DKL + NN models showing just how powerful the LSTM can be in feature extraction. We also note that the MOGP was also able to achieve low RMSE with training times comparable to the GP, making it a reasonable choice for low-fidelity models that need to be deployed.

An important metric to consider for environmental modeling and mapping, are the uncertainty measurements. When using in-situ measurements, data is assumed to be noisy which must be considered for decision making. GPs are a suitable choice for such decision making as they inherently give an uncertainty measurement through the kernel function. In Figure 4, the predictive mean and variance of station CHB0302 are shown where we can observe the different predictions and confidence of each model. In the DKL + LSTM model we observe a trend that no other model was able to capture, a flat region showing the changing direction of speed and direction. This shows that the LSTM is learning meaningful relationships between the time-series data and the outputs during joint training of the LSTM and MOGP. It should als be noted that in this example, the speed also seems to favor a flat regions indicating that the cross-covariance function $B_{d,d'}$ is capturing too much dependence for the speed and direction; however, the confidence region indicates that the MOGP could still choose an objective function that fits the model correctly. These results also indicate that the hyperparameters of the kernel function can be tuned more, or that a new kernel function may need to be tested to better fit the model. For example, at timestep 180 in direction measurements DKL + LSTM, the model dips before correcting to a constant direction. In this case, the model could favor a smoother transition that must be obtained through kernel function tuning.



(a) LSTM+DKL at t=0 **(b) LSTM+DKL at t=1** **(c) LSTM+DKL at t=2** **(d) LSTM+DKL at t=3**

Figure 5: LSTM + MOGP DKL Model over test stations COI0419, COI0420, COI0511, COI1203 over 100m depth of the coast of Alaska

4 Discussion

The proposed LSTM + MOGP DKL framework has significant potential for advancing Autonomous Underwater Vehicle (AUV) applications for creating a probabilistic map of marine environments. Leveraging deep feature extraction for the LSTM network combined with the MOGP shared kernel structure allowed for both the input (time) and output (speed and direction) to be taken into consideration when choosing a regression function. This is highly suitable for navigation in ocean vector-fields where there will be unseen or data-sparse regions; decisions in this case will be informed by the correlated input and output vectors within the area.

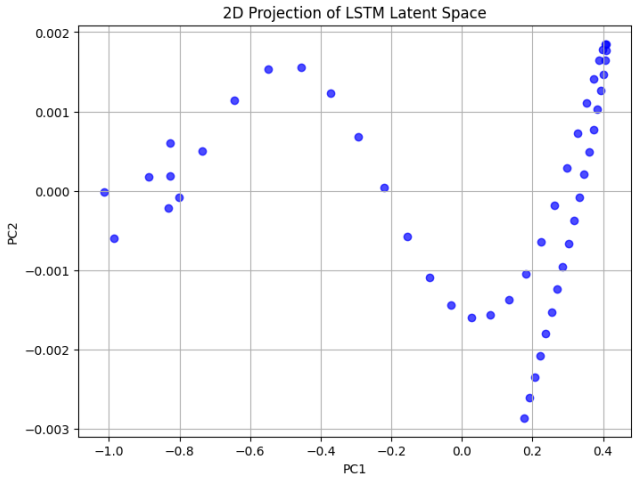


Figure 6: 2D Principle Component Analysis (PCA) of latent features learned by LSTM

In our LSTM model we learn a latent space represented as $\phi(\mathbf{x})$ and in Figure 5 we visualize the learned space for station CHB0302 in 2 components (or 2-dimensions). We can take note of the harmonic nature where the space seeks to capture not only the speed but directional data; which could an explanation for the vertical gradient of the input space towards the end of component one, as it is being projected into an orthogonal direction in a higher space. This input latent space serves as an example of the strengths of our model to capture complex features for the MOGP to learn. These features can then used for the MOGP to construct final vector map.

In figure 5, we extend our results from our LSTM + DKL model for application over spatial data. In this application we show how incorporating coordinates of our station data can also be used to construct a vector-field to demonstrate the change in speed and direction over time. Additionally, we interpolate an objective function represented by arrow directional vector and color-gradient representing speed to be human interpretable. At each time-step we notice the changes to the vector-fields and affects that station measurements have on neighbors. Figure 5 has very sparse data representations miles apart in distance; however, it serves as an example of potential representations that are possible through our model with limited data. With the short-training times limited to less than 5 seconds and sparse data representations, LSTM + DKL has shown that it can be extended to an online setting for AUV applications.

5 Conclusion

In this paper, we introduced a novel Deep Kernel Learning (DKL) approach for vector-valued Intrinsic Coregionalization Gaussian Process (MOGP) regression to predict ocean current vector fields. Specifically we used Long Short-Term Memory (LSTM) models for feature extraction for a MOGP model to find an objective vector-valued function. By incorporating LSTM and FNN architectures, the proposed method significantly improves prediction accuracy and stability while providing robust uncertainty estimates. Comparative results demonstrate that the DKL with LSTM model outperforms traditional GP and MOGP techniques, making it a promising solution for real-time AUV navigation and environmental mapping. Future research developments include kernel tuning, real-time adaptability, and scalable deployment for dynamic underwater applications given spatial representations.

References

[1] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P. Xing. [n. d.]. Learning Scalable Deep Kernels with Recurrent Structure. https://doi.org/10.48550/arXiv.1610.08936 arXiv:1610.08936

[2] Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. [n. d.]. Kernels for Vector-Valued Functions: a Review. https://doi.org/10.48550/arXiv.1106.6251 arXiv:1106.6251

[3] Edwin V Bonilla, Kian Chai, and Christopher Williams. [n. d.]. Multi-task Gaussian Process Prediction. In *Advances in Neural Information Processing Systems*

(2007), Vol. 20. Curran Associates, Inc. https://papers.nips.cc/paper/2007/hash/66368270ffd51418ec58bd793f2d9b1b-Abstract.html

[4] Thang D. Bui, Josiah Yan, and Richard E. Turner. [n. d.]. A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation. 18, 104 ([n. d.]), 1–72. http://jmlr.org/papers/v18/16-603.html

[5] Lehel Csató and Manfred Opper. [n. d.]. Sparse on-line gaussian processes. 14, 3 ([n. d.]), 641–668. https://doi.org/10.1162/089976602317250933

[6] Sepp Hochreiter and Jürgen Schmidhuber. [n. d.]. Long Short-Term Memory. 9, 8 ([n. d.]), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[7] Alexandre Immas, Ninh Do, and Mohammad-Reza Alam. [n. d.]. Real-time in situ prediction of ocean currents. 228 ([n. d.]), 108922. https://doi.org/10.1016/j.oceaneng.2021.108922

[8] Diederik P. Kingma and Jimmy Ba. [n. d.]. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980 arXiv:1412.6980

[9] NOAA. [n. d.]. *CO-OPS Current Station Data*. https://tidesandcurrents.noaa.gov/cdata/StationList?type=Current+Data&filter=historic

[10] Rich Pawlowicz, Bob Beardsley, and Steve Lentz. [n. d.]. Classical tidal harmonic analysis including error estimates in MATLAB using T_TIDE. 28, 8 ([n. d.]), 929–937. https://doi.org/10.1016/S0098-3004(02)00013-4

[11] Carl Edward Rasmussen and Christopher K. I. Williams. [n. d.]. *Gaussian Processes for Machine Learning*. The MIT Press. https://doi.org/10.7551/mitpress/3206.001.0001

[12] Dripta Sarkar, Michael A. Osborne, and Thomas A. A. Adcock. [n. d.]. Prediction of tidal currents using Bayesian machine learning. 158 ([n. d.]), 221–231. https://doi.org/10.1016/j.oceaneng.2018.03.007

[13] Dripta Sarkar, Michael A. Osborne, and Thomas A. A. Adcock. [n. d.]. Spatiotemporal Prediction of Tidal Currents Using Gaussian Processes. 124, 4 ([n. d.]), 2697–2715. https://doi.org/10.1029/2018JC014471

[14] Edward Snelson and Zoubin Ghahramani. [n. d.]. Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems* (2005), Vol. 18. MIT Press. https://proceedings.neurips.cc/paper/2005/hash/4491777b1aa8b5b32c2e8666dbe1a495-Abstract.html

[15] Dustin Tran, Rajesh Ranganath, and David M. Blei. [n. d.]. The Variational Gaussian Process. https://doi.org/10.48550/arXiv.1511.06499 arXiv:1511.06499

[16] Mark van der Wilk, Vincent Dutordoir, S. T. John, Artem Artemev, Vincent Adam, and James Hensman. [n. d.]. A Framework for Interdomain and Multioutput Gaussian Processes. arXiv:2003.01115 [cs, stat] http://arxiv.org/abs/2003.01115

[17] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. [n. d.]. Deep Kernel Learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (2016-05-02). PMLR, 370–378. https://proceedings.mlr.press/v51/wilson16.html

[18] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. [n. d.]. Stochastic Variational Deep Kernel Learning. https://doi.org/10.48550/arXiv.1611.00336 arXiv:1611.00336

[19] Boqiang Xu and Chao Liu. [n. d.]. A deep kernel regression-based forecasting framework for temperature-induced strain in large-span bridges. 323 ([n. d.]), 119259. https://doi.org/10.1016/j.engstruct.2024.119259

[20] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. [n. d.]. A Temporal Kernel Approach for Deep Learning with Continuous-time Information. https://doi.org/10.48550/arXiv.2103.15213 arXiv:2103.15213