Evaluating Hyperparameter Selection Techniques for the Ratio-Coupled Loss Function

Austin B. Schmidt
GulfSCEI
University of New Orleans
New Orleans, United States
email: sbaustin@uno.edu

Pujan Pokhrel
GulfSCEI
University of New Orleans
New Orleans, United States
email: ppokhre1@uno.edu

Md Meftahul Ferdaus
GulfSCEI
University of New Orleans
New Orleans, United States
email: mferdaus@uno.edu

Elias Ioup
Center for Geospatial Sciences
Naval Research Laboratory
John C. Stennis Space Center, United States
email: elias.ioup@nrlssc.navy.mil

Mahdi Abdelguerfi
GulfSCEI
University of New Orleans
New Orleans, United States
email: gulfsceidirector@uno.edu

Julian Simeonov
Marine Geosciences Division
Naval Research Laboratory
John C. Stennis Space Center, United States
email: julian.a.simeonov.civ@us.navy.mil

***Abstract*—When forecasting fixed-location observation nodes with statistical surrogate models, combining datasets during training loss calculation has been shown to improve model accuracy. However, traditional methods for tuning the data ratio, such as grid search or random search, are computationally prohibitive. An alternative online methodology for optimizing the data during training has been previously investigated. While both approaches have been independently validated, they have never been directly compared to each other or to other search techniques. This paper presents a direct comparison to evaluate whether the online approach can serve as a viable replacement for conventional search methods. The Cahn-Hilliard physical equation provides a controlled testing environment for this analysis. The results show that the optimization algorithm may require additional improvements before an out-of-the-box approach is appropriate. However, using the derived optimal hyperparameter in an offline setup provides an improvement in accuracy, which implies the methodology is worthwhile when under time constraints.**

Keywords-Ratio-Coupled Loss; Surrogate Model; Hyperparameter Tuning; Cahn-Hilliard.

I. Introduction

Fixed-location forecasting helps fill data gaps in ocean buoys, improve weather station predictions, and reduce uncertainty in tsunami detection [1][2][3]. Expanding the capability of machine learning models to forecast fixed-location time series is therefore an interesting and challenging problem. In the case of Partial Differential Equations (PDEs) and numerical models, machine learning surrogates provide an efficient alternative to direct numerical simulations, particularly for complex oceanographic tasks, such as fluid flow modeling [4]. Surrogate models serve as computationally efficient approximators, allowing for rapid inference without the high cost of solving PDEs from first principles. This is especially valuable in scenarios where real-time forecasting is required or when computational resources are constrained. The Cahn-Hilliard equation is investigated in this study as a controlled test case for evaluating surrogate modeling approaches. This equation describes phase separation processes and serves as a benchmark for studying non-linear PDE behavior, making it a suitable candidate for testing the efficacy of the proposed methodologies. Improvements to surrogate modeling of the Cahn-Hilliard system have potential implications for broader applications in oceanographic and geophysical flow models [5].

Combining data sources in machine learning often improves model performance across various contexts. From a data perspective, representations of physical phenomena are inherently flawed as they are only approximations of underlying behaviors. Sensors are known to be noisy and have measurable errors [6]. Similarly, numerically modeled data can include discretization errors or miscalculations from nonlinear interactions [7]. Therefore, finding ways to combine multiple sources of training data improves model stability and robustness to outlier data. Combining data within the loss function of a model through a ratio of error is shown to be particularly effective [8]. An importance-weighting hyperparameter controls error flow, allowing models to adapt to either data source. Selecting the best hyperparameter has been repeatedly shown to be a principal challenge with this methodology [8][9][10]. Therefore, the main novelty in this work comes from the comparison of multiple hyperparameter techniques to the convex ratio-loss function identified in [11]. In that work, an online algorithm to select the best value of a ratio-inducing hyperparameter was proposed. The paper focused on optimization mathematics and the impact of noise levels on model convergence. To establish its usefulness in modeling PDEs and real-world data, a benchmarking study must be conducted to compare against similar ratio-coupled loss. Accordingly, this work makes the following contributions.

- Hyperparameter search techniques are evaluated to identify the most effective method for error reduction and computational efficiency.
- The convex ratio-coupled loss function is compared to its non-convex counterpart to assess its impact on model accuracy.

- A surrogate model for the Cahn-Hilliard equation is developed to demonstrate the feasibility of surrogate modeling for nonlinear PDEs.

The paper is organized as follows: Section II reviews related work, highlighting comparable research and contrasting it with this study's objectives. Section III details the methodology. Section IV presents experimental results and their implications. Finally, Section V summarizes key contributions and outlines future directions.

II. Related Work

The Cahn-Hilliard equation originally modeled the phase separation process of binary alloys [12]. In modern research, Cahn-Hilliard equations have uses spanning from problems in material sciences to fluid dynamics [13]. In the context of oceanographic modeling, Cahn-Hilliard formulations have been coupled with the Navier-Stokes equations [5]. In those cases, Navier-Stokes governs fluid velocities while Cahn-Hilliard handles the relative density of fluid atoms. The work conducted in this paper proposes a methodology for forecasting fixed-location Cahn-Hilliard observations points. Future extensions of the work can eventually lead to a coupled environment to better model fluid mixtures.

The combination of varying data sources when modeling is seen in various contexts. Data assimilation improves analysis of physical systems and is recently combined with deep learning models [14][15][16]. Physics-Informed Neural Networks (PINNs) integrate training data with governing equations to enhance convergence and model robustness [17]. They have also been applied to solving Cahn-Hilliard equations with backward-compatible PINNs and adaptive-sampling PINNs [18][19]. Data can also be directly combined as input data from multiple observed and numerical sources [20][21]. The ratio-coupled loss function in this work combines data in the loss function during the training phase, like PINN models. However, the data is collected ahead of time and a ratio of loss from each source is used to regularize the training process [8]. This method is simpler than implementing PINN models as it does not require direct physical knowledge. It is also more flexible than using multiple input variables since additional data is only needed at training time.

Statistical models are frequently used as surrogate models for numerically derived and observed data. For example, Transformers are used to model significant wave heights observed by free floating buoys [22]. Recurrent neural networks are also a valid choice as they can manage long-term dependencies. The Long-Short Term Memory (LSTM) unit is the recurrent unit type featured in this work. LSTM units are used when forecasting many observed ocean parameters, such as sea surface temperature, salinity, significant wave heights, and others [8][10][23]. LSTMs are used in numerical surrogate contexts for modeling epidemic spread, turbulent flows derived from Navier-Stokes, and fluid-particle systems [24][25][26]. The LSTM unit is also used to model PDEs directly [11]. For example, mixed LSTM and convolution layers were used for pattern discovery to model the Cahn-Hilliard equation, achieving good agreement [27].

The investigation of hyperparameter search methodologies is inspired by common problems identified in other ratio-coupled loss research. Similar research uses a single λ parameter with grid search to find the most performant hyperparameter [8][10]. However, both works mention that long training times make the λ selection process difficult. This problem is exacerbated when using the multiple-λ ratio-coupled loss function [9]. As the number of hyperparameters combinations intractably increased, a bounded random search was used to explore the search space. A method for selecting the optimal hyperparameter with an online algorithm was finally proposed in a setting like the one investigated in this work [11]. However, this method was never validated against other search methodologies, focusing instead on the optimization problem itself. This paper extends the ideas in those works by comparing the optimized method with other search techniques.

III. Methodology

The following section describes all major techniques used to support the major claims. Within the section, the Cahn-Hilliard dataset details are outlined, the ratio-coupled loss function is detailed, the hyperparameter search techniques are compared, and the deep learning architecture used is described. The validation parameters and all experimental details are provided for reproduction.

A. Cahn-Hilliard Equation

In this work, the Cahn-Hilliard equation is used to model the concentration of binary fluids as they separate over time. The spontaneous phase separation of each fluid is demonstrated over 100 evolution steps and used to train surrogate models. The mathematical definition of the simple Cahn-Hilliard equation used to generate the training and testing data follows as,

$$\partial_t c = \nabla^2 \left(c^3 - c - \gamma \nabla^2 c \right), \tag{1}$$

where c is a scalar field taking values on the interval $[-1, 1]$ and γ sets the squared interfacial width. In the simulations used for this work, $\gamma = 1.0$. The implementation of this equation is given by the Python package py-pde [28]. The equation is evolved on a 20×20 sized grid with a random initialization of values on the interval $[-1, 1]$ for 100 time steps. The separation of fluid concentrations over the entire evolution period is seen in Figure 1. The figure displays how the random initialization dynamically splits into pure domains over the 100 epochs.

Along the evolution of the grid, Figure 1 also displays the target testing nodes. These nodes indicate the 80 fixed-location observation nodes reserved for final testing. The entire 20×20 grid yields 400 available observation points. Then, 350 of these points are randomly reserved yielding an 87.5% data coverage. Subsequently, 100 of those points are reserved as observation nodes for testing and validation data, at an 80/20 split. When modeling with machine learning algorithms, the amount of data is responsible for how well the surrogate model

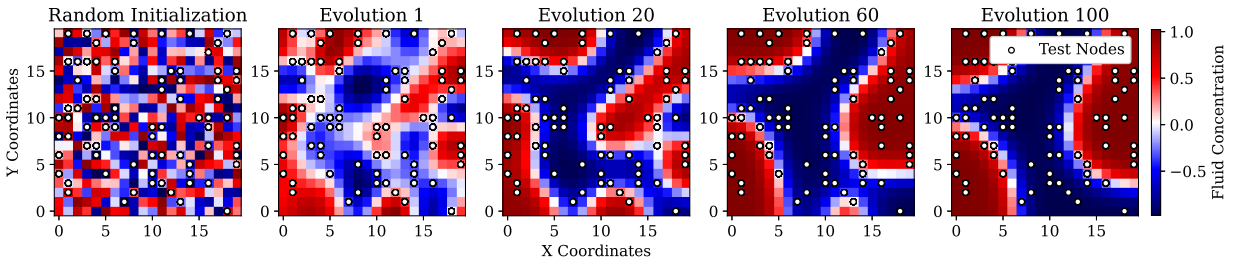

Figure 1. Evolution of the Cahn-Hilliard equation over 100 evolutions. Displays the locations of the 80 reserved observation testing nodes.

can generalize learned behaviors. A lower percentage of data coverage would negatively affect overall model accuracy.

To represent multiple sources of imperfect data, one final processing step is applied to the time series. First, the data is split into three separate instances. One instance represents noisy observations of local conditions and Gaussian noise is applied to the data at a fixed standard deviation rate of 0.01. This subset of data is used as the input to the model and when calculating error in the training loop. A second instance of the data represents larger inaccuracies from an additional, secondary source of data and has an applied Gaussian standard deviation of 0.1. This second source of data is used as a source of regularization when using the ratio-coupled loss function described in the upcoming section. The final instance of the data has no Gaussian noise applied to it and is used to calculate the final metrics of each model. This methodology follows the experimental setup in [11], which first introduced the optimized hyperparameter search explored in this work.

B. Ratio-Coupled Loss Function

A ratio-coupled loss function allows for two sources of data to be used for each feature when training a model. The hyperparameter λ controls the ratio of error generated when compared to either data source. The set of coupled features is defined as $S \subseteq \{1, \ldots, d\}$ where $d \in \mathbb{N}$. The cost function is formally introduced in (2)-(4) as,

$$\Delta_1 = g(\hat{y}, y_o), \tag{2}$$

$$\Delta_2 = g(\hat{y}, y_m), \tag{3}$$

$$\Omega_{\text{ratio-coupled loss}} = \lambda * \Delta_1 + (1 - \lambda) * \Delta_2. \tag{4}$$

In (2), the predicted value and the reserved values y_o are used to generate the first error term. Similarly, the same score is calculated for (3) by comparing the prediction and the secondary data source y_m. In the context of this work, y_o represents the reserved data with Gaussian noise of 0.01, y_m represents the reserved data with Gaussian noise of 0.1, and $\hat{y}$ represents the predicted output from the surrogate model. Therefore, the two Δ terms defined in (2) and (3) represent the error between the prediction and each of the dual sources as calculated with g. In this case, the error formulation g is Huber loss,

$$g_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{if } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta) & \text{if } |a| > \delta, \end{cases},$$

where a is the residual between predicted values and $\delta = 1.35$. Huber loss is used in this context to prevent model underfitting by penalizing inferences close to zero, which is the mean value across the Cahn-Hilliard formulation. The error terms are finally weighted by the hyperparameter λ as outlined in (4). The coupled feature loss is characterized by the ratio of the two Δ values that measure predicted error across multiple sources of truth. So, the selected λ value represents a ratio to determine the importance provided by either source. The hyperparameter is constrained to a ratio such that $\lambda \in [0.0, 1.0]$. The ratio of data that produces the most performant model is unknown and must be tuned by finding an optimal value of λ.

C. Hyperparameter Search Techniques

Four hyperparameter search techniques are compared in this work to validate whether the optimized λ technique is viable for improving the modeling of nonlinear fluid dynamic equations. To this end, a simple grid search, a random search, a Bayesian search, and the highlighted online optimized search are implemented for validation.

1) Grid Search: The grid search technique uses a range of λ hyperparameter values at fixed intervals. Each of these values is statically used to train a model. After each selection of λ is used, the results are compared and the best λ value is determined. When using a ratio-coupled loss function, $\lambda \in [0, 1]$, so the grid search is constrained to this interval. Given a fixed step size of 0.1, the grid search evaluates the hyperparameter values $\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. A smaller step size can be selected, although this may be computationally expensive in cases where the amount of data or model size is large.

2) Random Search: A random search explores the hyperparameter space by randomly selecting hyperparameters at a desired resolution for a fixed number of values. This technique is not guaranteed to select an optimal hyperparameter but can give good coverage on a variety of λ values, given enough iterations. In the case of this experiment, 20 random λ values are selected from a uniform distribution with a precision of 0.01.

3) Bayesian Search: Bayesian optimization is a popular hyperparameter search technique in deep learning. The optimization is used to select the minimizing hyperparameter

based on a specified criterion [29]. Therefore, the formulation is set up as,

$$\lambda = \arg \min_{\lambda \in [0,1]} f(\lambda). \quad (5)$$

The search space is defined such that $\lambda \in [0.0, 1.0]$. The criterion function $f(\lambda)$ is defined as the evaluation of a partially trained surrogate when making predictions on the validation dataset. That is, the error value on the validation data should be minimized by the selected λ. Continuing, Bayesian optimization is derived from Bayes' theorem. Given evidence data E, the posterior probability $P(M|E)$ of a model M is proportional to the likelihood $P(E|M)$ of observing E given model M, multiplied by the prior probability $P(M)$. This is expressed as,

$$P(M|E) \propto P(E|M)P(M). \quad (6)$$

The evidence and models terms expressed in (6) are the loss calculation and surrogate trained using a specific λ, respectively. Using this formulation, Bayes optimization iteratively searches the λ space for an optimal value. The specific implementation of the Bayesian optimization used in this work comes from the Python library GPyOpt [30]. The optimization algorithm selects the most performant λ over multiple trial iterations. Each candidate model is trained for 50 epochs, to save computational resources. There is an initial selection of eight random λ values and subsequent values are selected based on the expected improvement that a new λ value will provide. The optimization algorithm is run for a total of 30 iterations before selecting the λ that minimizes the objective function, $f(\lambda)$. Upon the selection of a minimizing λ value, a final model is trained using the full number of epochs for final evaluation.

4) Optimized Search: The final technique compared in this case study is the online optimized λ search that was first explored in [11]. The essential idea is that while the model weights are being trained, the λ hyperparameter is slowly optimized to the value that minimizes the loss function. To allow for direct optimization of the λ value, a modification must be made to the loss formulation described in (4). To make the function differentiable with respect to λ, square terms are added around each ratio term. Therefore, the convex ratio-coupled loss is defined as,

$$\Omega_{\text{convex ratio-coupled loss}} = (\lambda * \Delta_1)^2 + ((1-\lambda) * \Delta_2)^2. \quad (7)$$

The optimized value of the convex loss function is constrained such that the minimal value is guaranteed to exist when $\lambda \in [0.0, 1.0]$, which is one benefit of this method. Also, optimizing the λ simultaneously with model weights means that lengthy tuning times are reduced to the training of a single model. Any optimization scheme may be used, but the Adam optimizer and TensorFlow's gradient tape implementation allows λ to be easily optimized as the model is trained. However, it should be noted that by changing the loss function formulation, the model weights may not converge as they would with the basic coupled loss function. This crucial point is the motivation to compare the optimized search technique with other methodologies.

D. Machine Learning Model Architecture

Machine learning architecture is static among all experimental hyperparameter search types. The full architecture is displayed in Table I and is made up of five main layers and an input layer. The input to the model is a vector of four features, which include the current fluid concentration, the X and Y coordinates, and the timestep being modeled. Following, the LSTM unit is used in the next four layers. LSTM units are recurrent layers that contain input, forget, and update gates, which aid in learning time series dependencies [23]. Although the time horizon for a single forecast step will be one, the LSTM layers use additional parameters, making them a viable choice for surrogate modeling. Each of the LSTM layers uses the hyperbolic tangent activation function, allowing layer outputs in the range $[-1, 1]$. The final layer is a simple densely connected layer with a linear activation, to output the predicted fluid concentration in the next timestep. Dropout layers are placed between each LSTM layer with a dropout value of 0.2. These layers randomly drop weights during the training phase to help prevent overfitting of the model. Finally, the Adam optimization function is used to optimize the model weights.

TABLE I. LSTM MODEL ARCHITECTURE BY LAYER. THE TOTAL NUMBER OF TRAINABLE PARAMETERS IS 526,241. N REPRESENTS THE BATCH SIZE.

Layer Type	Shape	Parameters	Activation
Input Layer	(N, 4, 1)	0	None
Reshape	(N, 1, 4)	0	None
LSTM	(N, 1, 256)	267,264	Tanh
Dropout	(N, 1, 256)	0	None
LSTM	(N, 1, 128)	197,120	Tanh
Dropout	(N, 1, 128)	0	None
LSTM	(N, 1, 64)	49,408	Tanh
Dropout	(N, 1, 64)	0	None
LSTM	(N, 1, 32)	12,416	Tanh
Dense	(N, 1)	33	Linear

Before training, all input data is normalized with respect to the training data, including the testing data. To analyze the results, all data is transformed back to the original scale. When using the model to make predictions over multiple time horizons, a rolling forecast methodology is used. That is, only the first forecast uses fresh sensor values. The consecutive forecasts use the predicted fluid concentration as an input to the next prediction. This continues until the entire horizon has been predicted. Only then are fresh observation node values provided again. To improve model stability over multiple horizons, the same methodology is used when training the model. The loss function is summed over multiple predictions in a rolling-style forecast and the accumulated error is used to back propagate the model weights. To improve the model stability for rolling forecasts, the model accumulates training loss in the same way. That is, given a horizon size of eight, the training loop accumulates loss over eight steps while using the model's prediction as input values. This method of training supports long term predictions in the model by improving forecast stability.

E. Validation Metrics

There are two main metrics considered in this work. The first is the average Root Mean Square Error (RMSE) across all observational nodes reserved for testing. RMSE is preferred over other error scores because it highly penalizes large deviations. The RMSE formulation is defined as $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$. The parameter N is the number of test samples, y is the ground truth, and $\hat{y}$ is the prediction vector. Another method for analyzing the results is the time taken in seconds for each hyperparameter search to produce its most optimal result. Therefore, the best method should balance between minimizing the RMSE as well as requiring the least amount of time to produce those results.

IV. Results

After running each of the hyperparameter searches, 34 test cases were considered. By training each model with the same initial seed, only differences caused by varying λ values affect model convergence. Therefore, the effectiveness of each search methodology is considered by ranking the RMSE values over the full 12-step prediction horizon. The top ten results discovered are compared in Table II. In the table, it is noted that the top result uses the optimized λ search, with one major caveat. The online algorithm did not produce an optimal model by itself. Using the λ value derived from the online algorithm as a static λ value worked very well. In that case, the top result highlights an optimized λ in a static training environment. This is significant as it shows the optimized algorithm may not be suitable for directly training a model. However, this suggests it may be suitable for estimating the most performant λ overall. The grid and random searches both found values of λ that would be considered suitable. That is, they reduced the error beyond that found when no ratio-coupled regularization is used ($\lambda = 1.0$). Finally, the Bayesian search yielded interesting results by estimating a λ value that produced the lowest single horizon step RMSE overall. It seems reasonable to suggest that a longer training time per iteration of the Bayes search would result in an optimal λ value, like those found in the top four results.

TABLE II. Top 10 results sorted by the calculated RMSE over the full forecast horizon.

Rank	λ Search	λ Value	Full Horizon RMSE	Single Step RMSE
1	Optimized*	0.95961833	0.045182	0.013879
2	Grid	0.9	0.045754	0.014290
3	Random	0.89	0.045981	0.014298
4	Random	0.95	0.046696	0.014519
5	Grid	1.0	0.047465	0.014367
6	Bayes	0.7319939	0.047566	0.013782
7	Random	0.79	0.048094	0.014380
8	Grid	0.7	0.048827	0.014427
9	Random	0.68	0.048832	0.014167
10	Grid	0.8	0.048988	0.014345

Following, consider the top performing result for each of the λ search techniques in Table III. In this table, the total time taken for each search algorithm to provide the most performant result is given. It is notable that grid search and random search both linearly increase with each test case considered. Given a high number of tested λ values, increasingly better results can be found. However, this becomes prohibitively expensive as the resolution of the hyperparameter increases. Comparatively, the Bayesian search takes little time to explore the hyperparameter space. This is because only 50 epochs are used when searching for candidate values, which would require approximately 5% of the training time. Since the results are worse, it is likely that a higher number of epochs could balance time taken and the RMSE score. In the case of the optimized search, it takes exactly one training cycle of about 1,500 seconds to train the model and λ value. Given the inferior performance, the trade-off of time taken to RMSE is not very impressive. Encouragingly, using the optimized λ with the traditional ratio-coupled loss and a static training setup gives the best overall results, and it only requires enough time to train two models.

TABLE III. Comparison of the best results for each search technique.

Rank	λ Search	λ Value	Full Horizon RMSE	Single Step RMSE	Total Time Taken ($\approx$ s)
1	Optimized*	0.95961833	0.045182	0.013879	3,109
2	Grid	0.9	0.045754	0.014290	18,701
3	Random	0.89	0.045981	0.014298	30,644
7	Bayes	0.7319939	0.047566	0.013782	5,344
34	Optimized	0.95961833	0.055039	0.014998	1,582

An example observational node forecast is given in Figure 2. This example displays the inferences generated by the best performing models outlined in Table III. Every 12 forecast steps, the model is provided with new observation values, as denoted by the refresh points in the figure. The model does well at matching the fluid concentration over time, especially after the first 20 evolutions. This is in part because of the high coverage of training data as well as the rolling forecast implemented in the training algorithm. The best performing models mostly agree on forecasts. Observed differences are mainly in how close they fit to the actual curve. It is notable that there are 80 different observation nodes and each of them behaves differently, depending on how the Cahn-Hilliard equation evolves over time, as seen in Figure 1. Observation nodes that monitor unstable regions are notoriously more difficult to model in the long term.

Lastly, consider how the average error changes over the Cahn-Hilliard evolution in Figure 3. Comparing the single step error (left) and the 12-step error (right) shows two main behaviors. First, the instability early has the highest rate of error. It is difficult for the models to generalize how the evolution of a chaotic and randomly initialized system will begin to separate. Following, Cahn-Hilliard has two main phases, which are simpler to model. Either the fluid concentration has already separated into a stable group or is transitioning into a stable group. Both behaviors are more easily modeled. Finally, error seemingly rises by the end of the evolution period. This is attributed to the fact that some locations complete the phase separation process, resulting in some binary regions disappearing. Consider evolutions 60 through 100 in Figure

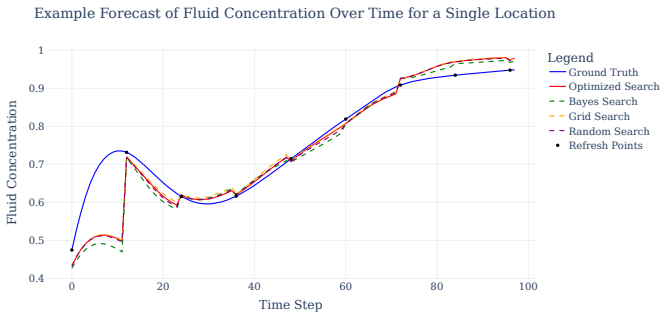

Figure 2. Example plot of a single observation node with a 12-step forecast horizon. Fresh initial values are seen every twelfth step.

1 as an example. The most performant λ values consistently seem to come from the Bayesian search λ and the optimized static λ. Surprisingly, Bayesian search performed well in general but suffered the most when initial conditions were chaotic.

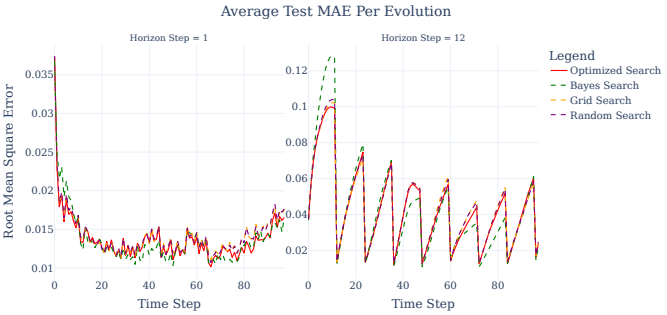

Figure 3. Evolution of average error over the evolution period. The single horizon error (left) and the full horizon error (right) highlight when the forecast problem is most difficult for each selected model.

Overall, each search methodology was compared based on the RMSE and time taken to produce its most optimal result. In the case of the optimized search, the resulting model performed poorly. However, the λ value produced from the algorithm yielded best results overall. This shows potential for estimating data-specific hyperparameters in a two-stage process. Although, if this methodology is to be useful in a wide range of tasks, formal investigations into methods for improving the online training approach are necessary. Through the investigation of a dataset distorted with Gaussian data, an obvious question arises. It is currently unknown whether other distributions of noise are optimizable in this way. Further investigations on real-world data and data distorted with other noise distributions should be validated against.

When considering the other search methodologies, Bayesian search also showed promise, given enough training epochs in the investigative phase. Grid search and random search are both valid methods for λ discovery, given enough time to explore enough model iterations. Overall, the best models performed very similarly, with minor variations in performance. This is to be expected when using the exact same model architecture. However, the main differences are seen when comparing the computational and time resources required to find the best result.

The Cahn-Hilliard equation is a good test bed for understanding how non-linear PDEs can be solved using surrogate modeling methods. Given the interesting initial results, a more generalized surrogate that can model Cahn-Hilliard under varying initial and forcing conditions is an interesting challenge. Future work that focuses on the coupling of these forecasts with Navier-Stokes to improve wave modeling, is a natural goal.

V. Conclusion and Future Work

In this work, surrogate models were investigated to forecast observation nodes of a Cahn-Hilliard equation with a random initialization. The models were trained using the ratio-coupled loss function and a rolling forecast-style training procedure. A selection of common hyperparameter search methodologies were compared to an online hyperparameter tuning algorithm. This algorithm adds quadratic terms to the ratio-coupled loss, to make the hyperparameter optimizable. Given varying amounts of noise added to two datasets, the search algorithms identified values for λ that were most performant. In all variations of the experiments, the average RMSE of all test cases and time taken to find performant hyperparameters were used to determine the best results. The comparison of search methodologies revealed that while the optimized search performed poorly overall, the λ value it produced led to the best results, highlighting the potential for a two-stage hyperparameter estimation process. Bayesian search also showed promise with sufficient training epochs, while grid and random searches remained valid given enough computational resources.

Future work will focus on three main identified weak points. Most importantly, investigations into improving the convex ratio-coupled loss function should be explored. Early stopping mechanisms to provide a stable training environment can be considered. Also, using the ratio-coupled loss function for model weights while the convex function for online tuning of the λ values might work in some situations. Next, the optimization algorithm should be validated on other datasets. In this case the models were not trained on different initializations of Cahn-Hilliard, which leads to poor generalization. Noise across varying data sources is not necessarily Gaussian, so future work would benefit from implementing different distributions of noise. Similarly, the methodology should be validating using real world data. Lastly, more advanced model architecture should be considered to understand how well the ratio-coupled loss function reacts when using more sophisticated models. Overall, there is a wide range of research directions to consider in the future.

References

[1] S. S. Kolukula and P. Murty, “Enhancing observations data: A machine-learning approach to fill gaps in the moored buoy data,” *Results in Engineering*, p. 104 708, 2025.

[2] D. S. Roy, "Forecasting the air temperature at a weather station using deep neural networks," *Procedia computer science*, vol. 178, pp. 38–46, 2020.

[3] M. Angove *et al.*, "Ocean observations required to minimize uncertainty in global tsunami forecasts, warnings, and emergency response," *Frontiers in Marine Science*, vol. 6, p. 350, 2019.

[4] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112 732, 2020.

[5] F. Boyer, C. Lapuerta, S. Minjeaud, B. Piar, and M. Quintard, "Cahn–hilliard/navier–stokes model for the simulation of three-phase flows," *Transport in Porous Media*, vol. 82, pp. 463–483, 2010.

[6] W. L. Oberkampf, S. M. DeLand, B. M. Rutherford, K. V. Diegert, and K. F. Alvin, "Error and uncertainty in modeling and simulation," *Reliability Engineering & System Safety*, vol. 75, no. 3, pp. 333–357, 2002.

[7] H. Y. Teh, A. W. Kempa-Liehr, and K. I.-K. Wang, "Sensor data quality: A systematic review," *Journal of Big Data*, vol. 7, no. 1, p. 11, 2020.

[8] A. B. Schmidt, P. Pokhrel, M. Abdelguerfi, E. Ioup, and D. Dobson, "Forecasting buoy observations using physics-informed neural networks," *IEEE Journal of Oceanic Engineering*, pp. 1–20, 2024. DOI: 10.1109/JOE.2024.3378408.

[9] A. B. Schmidt *et al.*, "Physics-regularized buoy forecasts: A multi-hyperparameter approach using bounded random search," in *COCE 2024: The First International Conference on Technologies for Marine and Coastal Ecosystems*, ThinkMind Digital Library, 2024.

[10] E. Sandner *et al.*, "A multiple-location modeling scheme for physics-regularized networks: Recurrent forecasting of fixed-location buoy observations," *COCE 2024: The First International Conference on Technologies for Marine and Coastal Ecosystems*, 2024.

[11] A. B. Schmidt, P. Pokhrel, M. Abdelguerfi, E. Ioup, and D. Dobson, "An algorithm for modelling differential processes utilising a ratio-coupled loss," *TechRxiv*, 2024.

[12] J. Kim, S. Lee, Y. Choi, S.-M. Lee, and D. Jeong, "Basic principles and practical applications of the cahn–hilliard equation," *Mathematical Problems in Engineering*, vol. 2016, no. 1, p. 9 532 608, 2016.

[13] J. Shen and X. Yang, "Numerical approximations of allen-cahn and cahn-hilliard equations," *Discrete Contin. Dyn. Syst*, vol. 28, no. 4, pp. 1669–1691, 2010.

[14] P. Pokhrel, M. Abdelguerfi, and E. Ioup, "A machine-learning and data assimilation forecasting framework for surface waves," *Quarterly Journal of the Royal Meteorological Society*, vol. 150, no. 759, pp. 958–975, 2024.

[15] J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino, "Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model," *Journal of computational science*, vol. 44, p. 101 171, 2020.

[16] M. Adrian, D. Sanz-Alonso, and R. Willett, "Data assimilation with machine learning surrogate models: A case study with fourcastnet," *arXiv preprint arXiv:2405.13180*, 2024.

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[18] C. L. Wight and J. Zhao, "Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks," *arXiv preprint arXiv:2007.04542*, 2020.

[19] R. Mattey and S. Ghosh, "A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 390, p. 114 474, 2022.

[20] E. Jang, Y. J. Kim, J. Im, and Y.-G. Park, "Improvement of smap sea surface salinity in river-dominated oceans using machine learning approaches," *GIScience & Remote Sensing*, vol. 58, no. 1, pp. 138–160, 2021.

[21] G. Ibarra-Berastegi *et al.*, "Wave energy forecasting at three coastal buoys in the bay of biscay," *IEEE Journal of Oceanic Engineering*, vol. 41, no. 4, pp. 923–929, 2016.

[22] P. Pokhrel, E. Ioup, J. Simeonov, M. T. Hoque, and M. Abdelguerfi, "A transformer-based regression scheme for forecasting significant wave heights in oceans," *IEEE Journal of Oceanic Engineering*, vol. 47, no. 4, pp. 1010–1023, 2022. DOI: 10.1109/JOE.2022.3173454.

[23] Q. Zhang, H. Wang, J. Dong, G. Zhong, and X. Sun, "Prediction of sea surface temperature using long short-term memory," *IEEE geoscience and remote sensing letters*, vol. 14, no. 10, pp. 1745–1749, 2017.

[24] D. A. David, L. Street, S. Ramakrishnan, and M. Kumar, "Lstm-based estimation of time-varying parameters in a spatiotemporal pde model for prediction of epidemic spread," *IFAC-PapersOnLine*, vol. 58, no. 28, pp. 468–473, 2024.

[25] A. T. Mohan and D. V. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks," *arXiv preprint arXiv:1804.09269*, 2018.

[26] A. Hajisharifi *et al.*, "An lstm-enhanced surrogate model to simulate the dynamics of particle-laden fluid systems," *Computers & Fluids*, vol. 280, p. 106 361, 2024.

[27] E. Kiyani, S. Silber, M. Kooshkbaghi, and M. Karttunen, "Machine-learning-based data-driven discovery of nonlinear phase-field dynamics," *Physical Review E*, vol. 106, no. 6, p. 065 303, 2022.

[28] D. Zwicker, "Py-pde: A python package for solving partial differential equations," *Journal of Open Source Software*, vol. 5, no. 48, p. 2158, 2020. DOI: 10.21105/joss.02158.

[29] J. Wu *et al.*, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[30] GPyOpt Project, *Gpyopt: A bayesian optimization framework in python*, http://github.com/SheffieldML/GPyOpt, [retrieved: September, 2025], 2016.