Auto-differentiable Transfer Mapping Architecture for Physics-infused Learning of Acoustic Field

Rayhaan Iqbal*, Amir Behjat†, Revant Adlakha‡, Jesse Callanan§, Mostafa Nouh¶, Souma Chowdhury||
Department of Mechanical & Aerospace Engineering
University at Buffalo (SUNY), Buffalo, NY, 14260-4400
Email: *rayhaani@buffalo.edu, †amirbehj@buffalo.edu, ‡revantad@buffalo.edu, §jessecal@buffalo.edu, ¶mnouh@buffalo.edu, ||soumacho@buffalo.edu

***Abstract*—Opportunistic Physics-mining Transfer Mapping Architecture (OPTMA) is a hybrid architecture that combines fast simplified physics models with neural networks in order to provide significantly improved generalizability and explainability compared to pure data-driven machine learning (ML) models. However, training OPTMA remains computationally inefficient due to its dependence on gradient-free solvers or back-propagation with supervised learning over expensively pre-generated labels. This paper presents two extensions of OPTMA that are not only more efficient to train through standard back-propagation but are readily deployable through the state-of-the-art library, PyTorch. The first extension, OPTMA-Net, presents novel manual reprogramming of the simplified physics model, expressing it in Torch tensor compatible form, thus naturally enabling PyTorch's in-built Auto-Differentiation to be used for training. Since manual reprogramming can be tedious for some physics models, a second extension called OPTMA-Dual is presented, where a highly accurate internal neural net is trained apriori on the fast simplified physics model (which can be generously sampled), and integrated with the transfer model. Both new architectures are tested on analytical test problems and the problem of predicting the acoustic field of an unmanned aerial vehicle. The interference of the acoustic pressure waves produced by multiple monopoles form the basis of the simplified physics for this problem statement. An indoor noise monitoring setup in motion capture environment provided the ground truth for target data. Compared to sequential hybrid and pure ML models, OPTMA-Net/Dual demonstrate several fold improvement in performing extrapolation, while providing orders of magnitude faster training times compared to the original OPTMA.**

***Impact Statement*—The new physics-informed machine learning (PIML) architecture presented in this paper provides in-situ transformation of input or estimation of latent parameters via a neural network, allowing a fast interpretable physics model to make accurate predictions. More specifically, this paper presents two extensions of the original underlying PIML architecture that significantly improves its training efficiency and scope of applicability. The latter is enabled by making OPTMA deployable as an end-to-end neural architecture within PyTorch. Classes of applications where OPTMA is expected to be particularly beneficial include problems where simplified (computationally efficient) physics model(s) are available, and they involve tunable parameters (which are otherwise user-prescribed). Such applications are abound in engineering problems, such as flow/aerodynamic analysis, materials characterization, robot dynamics and so on. Broadly speaking, the OPTMA architecture can thus be applied to a wide range of prediction problems in engineering, with end-uses being systems analysis, design and control.**

***Index Terms*—Acoustics, Auto-Differentiation, Extrapolation, Physics-infused machine learning, unmanned aerial vehicle**

I. Introduction

A. Data Driven Modeling of Complex Systems

Data driven machine learning (ML) models are utilized regularly for predicting the behavior of complex systems in different fields such as biological [1], engineering [2], [3], robotics [4]–[6], and energy forecasting [7] systems. While data driven models are seen to generate predictions of a competitive nature [8], [9], they under-perform in generalizing [10], [11] when trained with small or sparse datasets [12] (which is often the case in engineering analyses, design and controls problems), and usually fail at extrapolating [13]. Further, they exhibit sensitivity to noisy inputs [14] and are difficult to interpret [15] due to their black-box nature. For instances where a high fidelity (more complete) physics model is computationally expensive to evaluate and /or data collection by physical experiments is expensive and tedious, lower fidelity simplified physics or "partial physics" models (e.g., kinematics vs. full-dynamics, or a vortex lattice model vs. full CFD simulation) are also used as another alternative. However, these partial physics models, although interpretable and physics-conforming, are often inaccurate, fail to capture critical phenomena underlying the system behavior, and cannot be directly improved with available high-fidelity data. Hybrid surrogate models, a principal class of Physics-Informed Machine Learning (PIML) architectures that mix computationally efficient partial physics models with purely data-driven ML models in some form have been reported as one of the answers to these challenges [16]–[19]. This paper presents important extensions of one such recent type of hybrid model, resulting in the advancement of its training efficiency, testing performance and adaptability. For performance demonstration, we present one of the first known applications of hybrid models to successfully generalize and extrapolate acoustic fields generated by a quadcopter unmanned aerial vehicle (UAV). The remainder of this section briefly reviews related PIML work and outlines the research objectives of this paper.

B. Physics-Infused Machine Learning (PIML)

While hybrid ML architectures exist in various forms, Javed [20] provides a basic classification of them, namely as serial [21]–[24] and parallel [20], [25], [26] architectures. The data-driven model is either set in sequence with the partial physics model or used to tune the partial physics model parameters in serial architectures, while parallel architectures usually present additive or multiplicative ensembles of partial physics and data-driven ML models; the latter typically being artificial neural networks (ANN) or Gaussian Processes (GP) [27]. Several such hybrid PIML architectures have been reported in the literature in the past few years [24], [28]–[45], spanning over a wide range of applications such as in modeling dynamic systems, cyber-physical systems, robotic systems, flow systems and materials behavior, among others. A more comprehensive review of reported work on hybrid PIML models can be readily found in some of this above-cited literature, as well as in the review article by Rai and Sahu [19].

Two other well-known strategies constructing physics-informed ML models for prediction are by introduction of observational and inductive biases. Observational bias can be readily introduced into an ML model by gathering and utilizing data that encompass the implicit physics of the problem [46]–[49]. The drawback of this approach is the need for a fairly large amount of data to cover the entire input domain of a learning task [50]. Hence application of this strategy is challenging when training data comes from expensive simulations or limited physical experimental measurements or observations. Introducing inductive bias on the other hand enables imposing inherent physical restrictions on the prediction process [50]. Any prior information and inductive biases connected with a specific predicting objective are inherently incorporated within the network architecture in this case [51]–[53]. However, implicitly encoding physical laws in a neural network architecture [50] is often challenging, especially under the constrained expressibility of an assumed neural network structure and underlying activation functions.

Our PIML architectures are structurally closer to a serial architecture, with implementations mostly using ANN as the ML component. More specifically, we identify and present critical solutions to limitations of a recent hybrid PIML architecture that otherwise has already shown enhanced generalizability performance over pure data-driven models, with applications to dynamic, robotic, and flow systems [54], [55]. This original hybrid architecture from our earlier work, known as the "*Opportunistic Physics-mining Transfer Mapping Architecture*" (OPTMA) [54], leverages partial physics to process transformed inputs or estimated latent parameters given by a transfer ANN or GP model acting on the original input vector, and produce final output quantity of interest. The goal is to match this output value as closely as possible to the ground truth given by high-fidelity experimental or simulation data. An illustration of the original OPTMA architecture is shown in Fig. 1(I) (top left).

However, the integration of partial physics models that are not readily differentiable limited the choice of training approaches in the original OPTMA. This issue remains an Achilles heel in similar other hybrid PIML architectures that uses the constituent partial physics model in such a manner where differentiating it is necessary to propagate the training error and compute the updates to the constituent ML model parameters (e.g., ANN weights and biases) for back-propagation to be viable. To go around this issue, the original OPTMA was trained using a gradient-free Particle Swarm Optimizer (PSO) [56] for dynamics problems, and using precomputed intermediate labels directly at the output node of the constituent transfer ML model for the flow problem. Both of these training approaches are expensive compared to direct (one-step) back-propagation based training. Moreover, this form of the original architecture limits its deployment through state-of-the-art open-source ML libraries, due to dependencies on the form and language used for the partial physics implementation (which is application-dependent), thereby limiting its adoptability by a wider community of end-users who could potentially benefit from its generalization performance.

C. Research Objectives of this Paper

To solve the above-stated issues with the original OPTMA architecture while preserving – or potentially further enhancing its generalizability and even extrapolability performance by virtue of improved training – in this paper, we contribute two new auto-differentiable variations of OPTMA that are readily deployed using one of the most widely used state-of-the-art open source ML library, aka PyTorch. The new variations are: i) OPTMA-**Net**[1]; and ii) OPTMA-**Dual**.

OPTMA-**Net** presents the ability to represent the partial physics program (when available as a set of algebraic equations) as tensorial expressions that can be directly incorporated within PyTorch or similar Torch-tensor based ML libraries [13]. Thus the partial physics becomes a direct part of the end-to-end network, as shown in Fig. 1(II) (bottom left). This allows standard back-propagation to be used for training the OPTMA model through the in-built auto-differentiation capabilities of the concerned ML library, here PyTorch. The caveat here is the need for re-programming the partial physics model in the specific tensorial form expressed in Python language, which could be tedious.

To alleviate the above-described need in OPTMA-**Net**, we present OPTMA-**Dual** that completely substitutes the partial physics model with an internal neural network, as shown in Fig. 1(III) (right). It separately trains a PyTorch ANN model on the partial physics, where the latter could occur in any programmatic or even closed-source form. The premise here is that, as the partial physics model is inexpensive, it can be generously sampled to train a highly accurate ANN-based surrogate of the partial physics. Once trained, this internal network acts as the second network that now processes the output (e.g., transformed inputs) given by the transfer network in OPTMA, resulting in a unified end-to-end composite neural structure. Note that, since the trained model resulting from OPTMA-**Dual** no more contains the partial physics model, it does lead to a slight reduction in explainability compared to

[1]The prior version of this OPTMA-**Net** was presented and published in the proceedings of the AIAA Scitech 2022 conference [13]. This journal manuscript both summarizes that work and further extends it with another new architecture and comprehensive results analyses and comparisons.

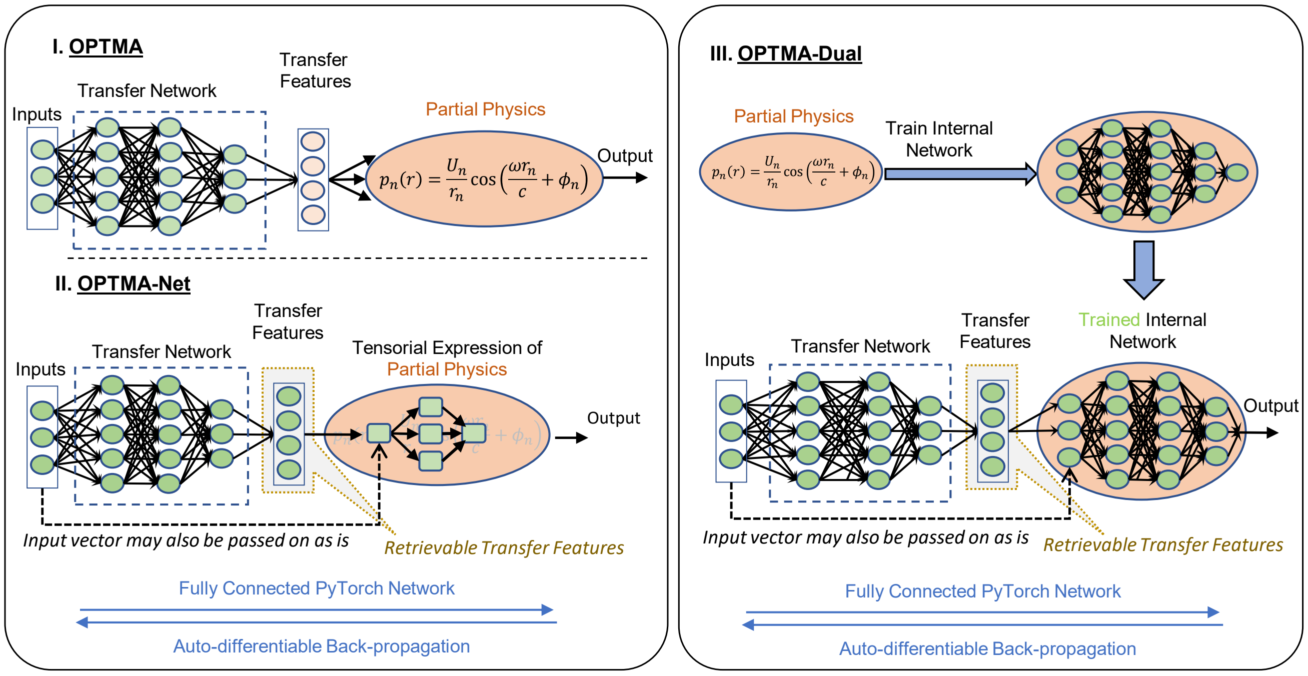
I. OPTMA
Transfer Network
Transfer Features
Inputs
Partial Physics
$p_n(r) = \frac{U_n}{r_n}\cos\left(\frac{\omega r_n}{c} + \phi_n\right)$
Output
II. OPTMA-Net
Transfer Features
Inputs
Transfer Network
Tensorial Expression of Partial Physics
Output
Input vector may also be passed on as is
Retrievable Transfer Features
Fully Connected PyTorch Network
Auto-differentiable Back-propagation
III. OPTMA-Dual
Partial Physics
Train Internal Network
Transfer Features
Trained Internal Network
Inputs
Transfer Network
Output
Input vector may also be passed on as is
Retrievable Transfer Features
Fully Connected PyTorch Network
Auto-differentiable Back-propagation

Figure 1 Model architectures: (I) OPTMA model. (II) OPTMA-**Net** with transfer model and tensorially expressed partial physics. (III) OPTMA-**Dual** with a transfer model and an internal physics based surrogate network.

OPTMA-**Net**, especially if one intends to track the sensitivity from transformed inputs to the final output when the hybrid model is tested (or used for model-based analyses, design or controls). Having said that, since the output nodes of the transfer network, which serve as input nodes of the following internal network, represent meaningful physical parameters (shifted inputs and/or intermediate parameters such as physical coefficients or conditions), OPTMA-**Dual** still provides increased interpretability compared to a standard pure data-driven end-to-end ANN model.

While these newer variations of OPTMA have a structural analogy to architectures that adopt an observation bias and induction bias strategy respectively, the direct torch-tensorial implementation makes it easier and more efficient to train. However, as in the induction bias strategy, one would still be somewhat limited in what physical model descriptions can be expressed in the torch tensorial form for an assumed neural structure; further research is needed to break down this barrier.

The objectives of this paper are thus: **1)** describe the two newly contributed variations of the OPTMA architecture; **2)** test these new architectures on analytical problems with comparisons to the original OPTMA and other baselines (pure ML and sequential hybrid models); and **3)** develop implementations of OPTMA-**Net** and OPTMA-**Dual** for predicting the acoustic field of a UAV where the partial physics model is given by a monopole based acoustic model and ground-truth is collected from physical experiments reported in Callanan et al. [57]. The outcomes of OPTMA-**Net** and OPTMA-**Dual**'s performance on this complex real-world problem are expected to inform the design of quieter UAVs in the future [58].

The remainder of this paper is structured as follows: the next section describes the architecture and training of OPTMA-**Net** and OPTMA-**Dual**. Section III reports the performance of OPTMA-**Net** and OPTMA-**Dual** on an analytical test problem. Section IV describes the UAV acoustic modeling problem, with results on this problem and further analyses presented in Section V. Conclusions are given in Section VI.

II. OPTMA-Net and OPTMA-Dual Architecture

A. *OPTMA as a PIML model*

Figure 1(I) shows the concept of OPTMA [59]. The transfer model takes the original inputs. It outputs the optimal transferred features that the computationally inexpensive partial physics model requires as input. This dynamically tuned partial physics model outputs the final quantities of interest. The transferred features could be shifted input values and/or intermediate parameters that serve as key coefficients and constants in simplified physics models. In implementations where the transferred features solely represent intermediate (coefficient) parameters of the partial physics model (and not shifted inputs), then the original input vector is also fed into the partial physics model as additional input features. The resulting physics-aware hybrid model is expected to provide improved generalization [54] and interpretability. The overall hybrid model loss is computed on how the output of the second portion, namely the partial physics model, deviates from the observed ground truth (high-fidelity output). Hence, in order to back-propagate the model loss, both the partial physics model and the ANN-based transfer model must be auto-differentiable. Very few partial physics models provide an auto-differentiable implementation, and even when it is available, it may not be amenable to interfacing with standard

ML libraries. Hence, a more straight-forward optimization based training was originally used, where the overall hybrid model loss is simply treated as an objective function to be minimized by a gradient-free solver. A Particle Swarm Optimization algorithm [56] was adopted for this purpose, which also helped in dealing with the observed multi-modality of the loss function (further explained in [59]).

*B. OPTMA-**Net***

OPMTA-Net [13] is proposed to address the training process limitations of the original OPTMA architecture. Fig 1(II) shows the architecture of OPTMA-**Net**. It utilizes a PyTorch structure. Specifically, the partial physics model – a set of algebraic equations in our target problems – is represented as PyTorch compatible tensorial expressions. This model thus expressed is connected at the output end of the transfer model, where the latter is a feedforward ANN for our case studies. However, this transfer model could be any other standard ANN such as recurrent networks (RNN), Long Short-Term Memory networks (LSTM), convolutional networks (CNN), graph neural networks (GNN) and so on, as suited to the nature of the input space for the concerned application. The resulting hybrid model now occurs as a fully connected PyTorch network, allowing usage of PyTorch's automatic differentiation for back-propagation over the entire network.

Additionally, the transfer features in the network have physical relevance with respect to the partial physics equations. This is because the tensorial physics layers in OPTMA-**Net** are an exact representation of the physics equations being used. A summary description of the implementation of OPTMA-**Net** is provided in the Supplementary Material. Preferably, OPTMA-**Net** must be used in problem statements where the partial physics is simple to express in Torch-tensorial form.

*C. OPTMA-**Dual**: Framework*

In OPTMA-**Dual**, the manually programmed partial physics layers are replaced by a separate neural network, which we call the *internal network* here onward. This internal surrogate network is trained apriori on a large dataset generated using the inexpensive partial physics model. Thus the internal network is considered to very closely embody the partial physics model. Once trained, the internal network is added at the output end of the transfer network in the forward propagation loop of OPTMA-**Dual**, as shown in Figure 1(III). The first part of OPTMA-**Dual** represents the transfer model and outputs the transfer features which are inferred as the parametric inputs to the partial physics model. But, in this case, these parametric inputs or transferred features are fed directly to the internal network. The output of the internal network is the desired output of interest to be compared with high-fidelity ground truth for computing the model loss of OPTMA-**Dual**.

OPTMA-Dual: Internal network: The goal of introducing a separately trained neural network inside OPTMA architecture is twofold: **1)** enable an end-to-end neural network whose model complexity is less dependent on that of the partial physics; and **2)** alleviate the need for end-users to manually re-program partial physics models into Pytorch amenable tensorial forms. Note that such re-programming might not be practically viable when the partial physics model involves non-differentiable evaluations. A basic example would be any partial physics function involving sorting or ranking evaluations, which are known to be problematic for end-to-end automatically differentiable pipelines [60]. While there might be differentiable approximations available (e.g., differentiable approximate sorting [61]), they are not guaranteed to be available for all such scenarios. Implementation of differentiable approximation modules in an existing workflow is also challenging and time-consuming if at all viable.

OPTMA-**Dual**'s substituted internal neural network offers a flexible yet familiar alternative to such partial physics functions, which is easy to incorporate. Again note that, any standard type of networks (DNN, LSTM, RNN, CNN and GNN) can be used as the internal network depending on the input space and numerical nature of the partial physics model.

Therefore, the prediction of the OPTMA-**Dual** model on any given input x can be expressed as:

$$\begin{aligned} Y_{\texttt{OPTMA-D}}(x) &= F_{\texttt{IN}}([x, x_{TF}]), \\ \text{where} \quad x_{TF} &= F_{\texttt{TN}}(x) \end{aligned} \tag{1}$$

where $F_{\texttt{IN}}$ and $F_{\texttt{TN}}$ respectively represent the output of the internal ($\texttt{IN}$) and transfer ($\texttt{TN}$) networks. Note that in application scenarios where the transferred features (x_{TF}) are simply shifted values of the input vector x as opposed to model coefficients/constants in the partial physics, the input to the internal network does not need to separately include x.

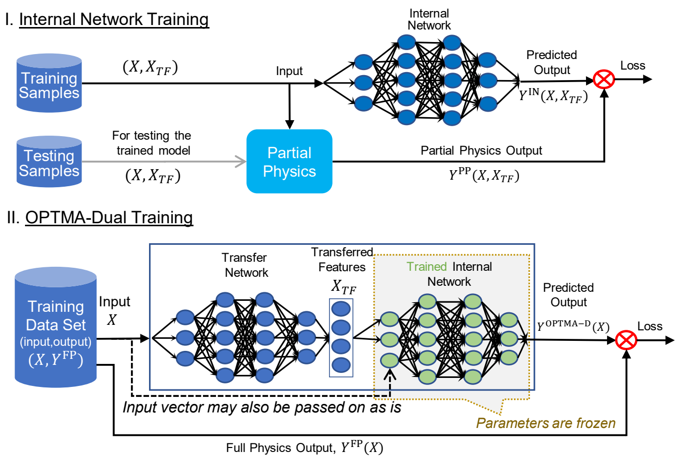

Figure 2 OPTMA-**Dual** training and application architecture. **I)** Internal network training architecture, **II)** OPTMA-**Dual** (mainly transfer network) training architecture.

*D. OPTMA-**Dual**: Training and Application*

OPTMA-**Dual** is trained in a 2-stage process, shown in Fig. 2. In the first stage (Fig. 2(I)) we train the internal network over partial physics samples. Here, sampling is done over the entire input feature space of the partial physics as in its assumed role within the original OPTMA, which includes the transferred features (X_{TF}), as well as the original input vector (X) if they have not been shifted. Labels for training the internal network are generated by passing these input samples through the partial physics model at this stage, i.e., $F_{\texttt{PP}}([X, X_{TF}])$. The internal network obtained thereof is then tested over unseen partial physics samples, as a sanity check, to ensure reasonable accuracy in substituting the partial physics. Both the training and testing data set are separately generated using Latin Hypercube Sampling or LHS [62] by default, unless a problem-specific design of experiments (DOE) is available or required. The criteria used in LHS is centering of the points within the sampling intervals. The internal network is then infused in the forward propagation of OPTMA-**Dual**, shown in Figure 1(II). Now, we freeze the (trained) internal network parameters using *`model_params.requires_grad=False`*. Hence, in the second training stage, shown in Fig. 2(II), only the transfer network parameters get updated. At this second stage, the loss function is computed based on the samples of the full physics function (i.e., $X, Y_{\texttt{FP}}(X)$). The input portion of these training samples is also generated using LHS by default, unless a problem-specific DOE is required. In some applications these samples might simply be given to us, resulting from a physical or simulation experiment that we did not necessarily have the liberty to design.

The hybrid model in OPTMA-**Dual** is essentially a composite of two networks that is readily trainable with in-built Auto-differentiation based solvers (e.g., SGD and Adam) in PyTorch. This structure makes OPTMA-**Dual** agnostic to the nature of the partial physics model, which along with the programmatic ease of deployment significantly opens up the applicability and adoptability of OPTMA. While OPTMA-**Dual** contains pure neural network components, it still retains a degree of interpretability, since the intermediate information, i.e., the output of the transfer network (which is also the input to the internal network) depicts meaningful physical parameters, and can be readily retrieved for analyses and interpretation of how the transfer network is helping correct the substituted partial physics for accurate predictions. There are several ways of retrieving these transferred parameter values, one of which is by placing a hook on the output of the transfer model in OPTMA-**Dual**.

III. Demonstrative Test Problems

A. Analytical Test Problem: Gramacy & Lee Function

For ease of analysis of OPTMA-**Net** and OPTMA-**Dual**, and comparisons with baselines, we first use a simple analytical test problem, namely the Gramacy & Lee Problem from [54]. Here, the goal is to predict the output of two different full physics functions. To this end, OPTMA and its proposed new variations here will use a common partial physics function. Based on the described framework in Section II-C the trained model using OPTMA-**Dual** will transfer the input features which will be used with the partial physics model to make the full physics predictions. The equations of the partial physics ($F_{\texttt{PP}}$) and the two full physics ($F_{\texttt{FP1}}$ and $F_{\texttt{FP2}}$) functions are given in Section S-I-A in the Supplementary Material.

Three baseline models are used to compare the performance of the OPTMA architectures over this analytical study: i) A Pure Data-Driven ML Model, to highlight the benefits of partial physics infusion; ii) A Sequential Hybrid Physics Infused Model adopted from [57] and iii) The original OPTMA model, here named as OPTMA-PSO, and illustrated in Fig. 1(I). This is to specifically point out the advantages of the newer variations presented in this paper, namely OPTMA-**Net** and OPTMA-**Dual**. A more detailed description about the baseline models is provided in Section S-I-B in the Supplementary Material.

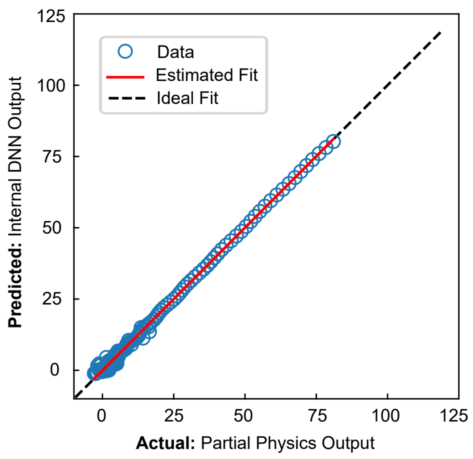

Figure 3 OPTMA-**Dual** internal network prediction VS Gramacy & Lee partial physics function regression plot for the testing dataset

B. Analytical Test Problem: Numerical Settings & Results

The structure of the internal neural network in OPTMA-**Dual**, the data set size and settings used to train it over `PP` samples are given in Table I. To ensure an accurate capture of the partial physics function (`PP`), the trained internal network is tested on 200 unseen samples, yielding a small mean-squared error of 0.009. The testing performance on the internal network is shown as a comparison between the actual and predicted in Fig. 3, illustrating the accuracy of the internal network.

The settings and model structure used for training the transfer model in OPTMA-PSO, OPTMA-**Net** and OPTMA-**Dual** are given in Table II.

With each of the two full physics Gramacy & Lee functions, we set up four cases, defined by the size of the training data set. The OPTMA models and the baseline models are trained for each of these cases, and tested on 200 unseen samples. The performance of the trained models on unseen samples is termed as generalizability in this case [54], [63]. The testing performance is reported in terms of the root mean squared error or RMSE. Section S-I-C in the Supplementary Material shows how the RMSE is calculated in this paper.

The testing RMSE results are provided in Table III. As seen from Table III, OPTMA-PSO performs the best with smaller training samples of `FP1`, with OPTMA-**Net** performing the best in the last case of `FP1`. However, the testing RMSE for

TABLE I Gramacy & Lee: OPTMA-**Dual**'s *internal network's* training settings and testing performance on the `PP` function

# Inputs/Outputs	Train Size	Test Size	# Layers	# Nodes/layer	Dropout	Learning Rate	Tr. Time	Test MSE
1/1	1000	200	5	50	0.6	0.00035	24.3 sec	0.009

TABLE II Gramacy & Lee: Training settings of OPTMA transfer models and of networks in sequential hybrid and pure data-driven models for test type FP2

Model	# Layers	# Nodes/ layer	Dropout	Learning Rate
OPTMA-PSO*	4	11	–	–
OPTMA-**Net***	2	50	0.43	2×10^{-4}
OPTMA-**Dual***	2	50	0.43	2×10^{-4}
Sequential Hybrid	3	50	0.1	1×10^{-5}
Pure Data-Driven	3	50	0.1	1×10^{-5}

*Transfer network.

the OPTMA models are within the same order of magnitude. For `FP2`, OPTMA-**Dual** outperforms all the other models, except in the last case where OPTMA-**Net** performs the best. Overall, both OPTMA-PSO and the newer OPTMA-**Net** and OPTMA-**Dual** perform better than the pure data-driven and sequential hybrid baselines. Specifically, compared to the pure data-driven model, the RMSE of OPTMA-**Dual** is more than an order of magnitude smaller for `FP1`, and about 2 to 4-fold smaller for `FP2`. The performance gains are more notable when the training set is sparse, demonstrating the increased usefulness of partial physics infusion in such scenarios.

Since the conceived advantage of the newer versions of OPTMA is their training efficiency (and performance), it's critical to compare the training (CPU) time of OPTMA-**Net** and OPTMA-**Dual**, with that of the original OPTMA-PSO. These computing times of the training process for the Gramacy & Lee problem are reported in Table IV. These computing times correspond to execution on a 6 core Intel i7-9750H CPU and an NVIDIA GeForce GTX 1660 Ti 2.60 workstation (with the models running on the GPU). Table shows that both OPTMA-**Net** and OPTMA-**Dual** trains significantly (39 to 77 times) faster than OPTMA-PSO, by virtue of the auto-differentiability of the newer implementations. Note that in the case of OPTMA-**Dual**, there's an additional cost for training the internal neural network, which along with the cost of training the transfer function in OPTMA-**Dual** made it overall (3 to 4 times) slower-to-train compared to OPTMA-**Net**.

IV. PIML Modeling of UAV Acoustic Field

A. Problem Statement and Experimental Data Collection

The goal here is to develop a model that can predict the noise signature of a hovering quadcopter unmanned aerial vehicle (UAV) in an indoor environment. Noise mitigation has become a very significant area of R&D in UAVs, due to the increasing usage of UAVs in a human-robot collaborative environment such as in an indoor warehouse setting [64], [65] or last-mile delivery [66]. Such noise can both cause annoyance and distraction as well as have harmful cognitive and hearing impact with long term exposure [67]. To develop noise mitigation techniques or design quieter UAVs, there is a critical need for frameworks to model the noise signature of multirotor UAVs. Traditional methods to model UAV sound fields include using Finite Element simulations, creating synthetic data using real recordings [68], and software packages like iNoise [69], [70] for industrial noise predictions leading to the proposal of various frameworks to measure and model the UAV sound field [71]. However, there are very few data-driven approaches to modeling the UAV sound field. Some of the recent efforts include our prior work with a purely data-driven model and a sequential hybrid model [57], which this current work improves upon.

To this end, in an earlier work [57], we developed an experimental setup for measuring UAV noise for a small set of scenarios, and using the data to predict the 3D noise field of the UAV. Such experiments are not only complex but also expensive in terms of time and money, which limits both the distances (from source) at which such measurements can be taken and the number of measurements. Hence, a modeling approach that can use such a small and distance-limited data set, and still provide a generalizable and extensible (in space) prediction of the noise field would be uniquely helpful to the UAV and co-robotics communities. We hypothesize that OPTMA is well-suited to serve in this role of generating generalizable and extrapolatable prediction models.

To test this hypothesis, data presented in our previous study [57] is used in this paper to train and test OPTMA models of the UAV noise field, and compare their performance with that of the baselines. The dataset contains the root mean square (rms) sound pressure level (SPL) measurements of noise generated by an unconstrained hovering UAV (a popular commercial model). This includes data points for a total of 1700 locations in a 3-dimensional space as collected by a movable microphone array. A custom scanning head microphone array called the Large Aperture Scanning Microphone Array (LASMA) was constructed and used for this purpose. The LASMA is capable of scanning a total area of 2.3×1.2 m^2. The LASMA consisted of four microphones (BSWA MPA416 pre-polarized $1/4$" microphones) which generate electronic signals digitally recorded with a USB MC3522 DAQ. A Vicon motion capture systems was utilized to capture the locations of the microphones and the sound source (UAV) with the help of retro-reflective markers. While the location of microphones was precisely controllable, the hovering UAV as usual experienced small drifts during each experiment (with the microphone array at a particular setting). This resulted in a series of data points representing the scalar sound pressure level field at a set of locations distributed irregularly. Thus,

TABLE III Gramacy & Lee (GL): Testing results in terms of normalized RMSE for predicting the `FP1` & `FP2` functions

Train Size	Pure DataDriven [13]	Seq. Hybrid Model [13]	OPTMA-PSO [54]	OPTMA-**Net** [13]	OPTMA-**Dual**
FP1					
20	0.243	0.106	**0.039**	0.094	0.053
50	0.111	0.081	**0.019**	0.074	0.043
100	0.073	0.052	**0.028**	0.049	0.040
200	0.064	0.041	0.036	**0.022**	0.031
FP2					
20	0.172	0.211	0.061	0.072	**0.054**
50	0.123	0.068	0.080	0.055	**0.046**
100	0.086	0.075	0.088	0.048	**0.045**
200	0.079	0.059	0.083	**0.040**	0.045

TABLE IV Computing times to train OPTMA transfer network on the GL problem

Problem statement	OPTMA-PSO	OPTMA-**Net**	OPTMA-**Dual***
FP1	702.97 sec	**9.38** sec	17.86 sec
FP2	734.42 sec	**9.55** sec	13.5 sec

*Excludes time taken to train internal network which is reported in Table I.

the objective of modeling (trained on this data) is to predict the SPL value (L_p) at any location $\mathbf{r} = [x, y, z]$, with the UAV assumed to be located at the origin.

B. Partial Physics Acoustic Model

A simple wave-based acoustics model is used as the partial physics model to characterize the noise field of the hovering UAV. This model could be implemented with an arbitrary number of spherical acoustic sources. The hypothesis is that when the parameters of this model are tuned to the optimal values, the model would be able to predict the experimentally obtained acoustic field with high accuracy, attributed to the constructive and destructive interference of waves generated by different monopoles. The parameters in the acoustic function include the amplitude, frequency, phase, and relative positions of the arbitrary monopoles. The time-independent pressure field generated by an individual monopole can be defined as

$$p_n(\mathbf{r}) = \frac{U_n}{r_n}\cos(\omega_n r_n/c + \phi_n) \tag{2}$$

Here $\mathbf{r}$ is the position vector of the field point relative to the origin; and $r_n = |\mathbf{r}_n|$ is the Euclidean distance of the field point from the n^{th} acoustic source, where $\mathbf{r}_n = \mathbf{r} - \mathbf{q}_n$, with $\mathbf{q}_n$ as the position vector of the n^{th} acoustic source relative to the origin. The n^{th} source has a normalized amplitude U_n, angular frequency ω_n, the speed of sound defined at STP as $c = 343$ m/s, and ϕ_n as the phase angle. The subscript $(\bullet)_n$ is the shorthand notation for the n^{th} acoustic spherical source.

The net acoustic pressure field can be computed by adding the pressure fields of all the acoustic sources as given by

$$P(\mathbf{r}) = \sum_{n=1}^{N} p_n(\mathbf{r}) \tag{3}$$

where N is the total number of acoustic sources. Finally, the sound pressure level (SPL) can be computed as

$$L_p(\mathbf{r}) = 20\log_{10}[|P(\mathbf{r})|/P_{\text{ref}}] \tag{4}$$

where $P_{\text{ref}} = 20\ \mu$Pa and $|\bullet|$ is the absolute value. It is worth noting that the parameters related to the arbitrary number of monopoles when tuned may or may not hold physical significance as the aim is to estimate the sound pressure level with high accuracy for a field point and numerous combinations in the parameter space could possibly produce the same output (a many-to-one mapping).

It is worth noting, that the parameters of each monopole are intrinsic properties of a theoretical sound source that accurately predicts the time-averaged sound pressure levels in an anechoic, i.e. reflectionless, environment and not the physical sound source such as the UAV in this instance. The time-averaged SPL at a given field point is not unique to a specific source parameter configuration and can be reproduced with infinite different combinations of source parameters and source positions. The hybrid modeling approach presented here aims to force the model to learn the optimal distribution of the parameters of the theoretical acoustic model to reproduce the desired SPL at the specific field point of interest. Additionally, predicting the SPL in a reverberant space such as a warehouse creates modeling challenges due to the reflections and interferences caused by room acoustics, thereby motivating the use of multiple asymmetrically located monopoles with locations tuned to cater to predicting accurate SPL at different measurement points.

*C. OPTMA-**Net** Implementation: UAV Acoustic Field*

To implement OPTMA-**Net**, the partial physics model described in Eqs. 2 to 4 is re-programmed in torch tensorial form, details of which are provided in the Supplementary Material. Here, the normalized amplitude of each n^{th} monopole, U_n. is selected to serve as the transferred feature. Hence the size of transferred feature vector is equal to the number of monopoles used to represent the noise source. The other parameters of this partial physics model are kept fixed, which includes c, ω, ϕ, and $\mathbf{q}$ for each of the four monopoles. Hence the OPTMA-**Net** model for this problem, which predicts the SPL ($L_p(\mathbf{r})$) given the location ($\mathbf{r} = [x, y, z]$) from the source (UAV), can be mathematically expressed as:

$$\begin{aligned} L_p(\mathbf{r}) &= F_{\texttt{PP-Net}}(\mathbf{r}, \mathbf{U}, [c, \omega, \phi, \mathbf{q}]) \\ \text{where} \quad \mathbf{U} &= F_{\texttt{TN}}(\mathbf{r}) \end{aligned} \tag{5}$$

where $F_{\texttt{PP-Net}}(\bullet)$ represents exact network tensorial implementation of the acoustic partial physics model with N monopoles. Here, the fixed values of the other parameters, c, ω (set to be the same for all 4 monopoles), $\phi = [\phi_1, \ldots, \phi_N]$ and $\mathbf{q} = [\mathbf{q}_1, \ldots, \mathbf{q}_N]$, used in computing $F_{\texttt{PP-Net}}$ are given in Table - S - I in the Supplementary Material. Note that each $\mathbf{q}_n$ is a vector representing the 3D location of each monopole source in the partial physics model. The PyTorch implementation of the transfer and partial physics network layers in OPTMA-**Net** is provided as Listings and 1 and 2 in the Supplement.

When implementing the acoustics partial physics within Pytorch in OPTMA-**Net**, the choice of physics model parameters to be transferred must be decided apriori. If this choice is changed, the torch tensorial implementation needs to be partly reprogrammed, which somewhat limits the flexibility of OPTMA-**Net** in problems where there are various parameter options that could be fixed or allowed to be predicted by the transfer network. Here, we consider four monopoles in the partial physics implementation with their amplitude (U) treated as the transferred feature in OPTMA-**Net**. Hence the number of transferred features, i.e., the size of the output of the transfer network, is four in this case.

*D. OPTMA-**Dual** Implementation: UAV Acoustic Field*

Since OPTMA-**Dual** substitutes the partial physics by a neural network, unlike OPTMA-**Net**, here it is easy to change the choice of which physics model parameters to pre-fix and which to predict via the transfer network. When implementing OPTMA-**Dual** on this UAV acoustics problem, we readily increase the transferred feature space to include all the four parameters, namely amplitude (U_n), phase (ϕ_n), frequency (ω_n), and the 3D position ($\mathbf{q}_n \in \mathbb{R}^3$), for each n^{th} monopole. We train the internal network on the acoustics partial physics model implemented with four monopoles, to allow ready comparison with OPTMA-**Net**. Thus in this case, for OPTMA-**Dual**, the size of the transferred feature space is 24. While the increased transferred feature space might lead to an increase in the training effort for the transfer network, with more knobs to tune it is expected to allow greater flexibility in matching the ground truth.

The steps to setup OPTMA-**Dual** for the acoustic problem is illustrated in Fig. - S - 1 in the Supplementary Material. Step I: We first train the internal network on a dataset generated by the acoustic partial physics model. By training the internal network with the global set of partial physics parameters treated as inputs (here 24 parameters), and the LHS sampling covering this entire set, one can readily use the internal network (later on in step II) with any subset of parameters treated as the transferred features, with others fixed at user-prescribed values if desired, thereby providing greater modeling flexibility. Step II: This internal network is infused at the output end of the transfer network in the forward propagation of OPTMA-**Dual**. The code to incorporate a pre-trained network has been shown in Listing 3 in Supplementary Material. Note that this pre-trained internal network is kept frozen during the second training step, i.e., when the transfer network is trained.

E. Internal (Acoustic) Network: Training and Validation

Along with the global set of 24 tunable parameters in the UAV acoustics' partial physics model, the inputs to the internal network also include the original problem inputs, namely the 3D location ($\mathbf{r} \in \mathbb{R}^3$) at which the SPL is being measured. This brings the total size of the input vector to 27 for the internal network. LHS is used to sample this 27 dimensional space, with the range of each physical input parameter given in Table II in the Supplementary Material. The corresponding SPL value for each sample is computed from the acoustic partial physics model, which form the labels to train the internal network. We generate 10000 samples using LHS [62] in the input space and use 80% of the data for training the model. The inputs and outputs are normalized using the range of data. Since the inputs to the internal surrogate network are in the normalized range, the transfer feature analysis of OPTMA-**Dual** would correlate to the normalized latent space of the transfer features; analysis of the output transfer features is explained in Section VI-C. Note that the same SPL normalization limits must be used in both the internal network training as well as the complete OPTMA-**Dual** training for meaningful application.

Prior to use in Step II, we first test the quality of the trained internal network on the remaining 20% partial physics samples. A normalized mean square error of 0.009 is observed in testing, showing that the internal network provides an acceptable substitute of the acoustic partial physics model. The training settings, network structure and testing performance of the internal (acoustics) network are summarized in Table V.

F. Sequential Hybrid Model Baseline: UAV Acoustic Field

The sequential hybrid model is adopted from [57]. Here, along with the spatial inputs $\mathbf{r}$ (i.e., location vector w.r.t. noise source), SPL outputs of 5 different instances of the partial physics model (operating on $\mathbf{r}$) are passed as additional input features into a neural network. The five different instances of the acoustics partial physics model corresponds to five different combinations of physical parameters (with variations in $\mathbf{U}$, keeping $\mathbf{q}$, ϕ and ω fixed) in the partial physics model. Here the partial physics model was implemented with four monopoles, i.e., $n = 1, 2, 3, 4$. The neural network then predicts the final output of interest, the SPL value L_p at location $\mathbf{r}$.

TABLE V UAV Acoustic Field: OPTMA-**Dual**'s *internal network's* training settings and testing performance on the PP function

# Inputs/Outputs	Train Size	Test Size	# Layers	# Nodes/layer	Dropout	Learning Rate	Tr. Time	Test MSE*
27/1	8000	2000	6	600	0.01	8×10^{-4}	25.1 sec	**0.0097**

*The MSE is reported on the range normalization of the dataset.

V. Model Testing Cases: UAV Acoustic Field

For this UAV acoustics problem, we will compare OPTMA-**Net** and OPTMA-**Dual** with the sequential hybrid PIML model (described in Section IV-F) and a pure data driven network network model. The last one simply uses a single neural network that takes as input a 3D location and outputs the SPL value at that location. In order to test the generalization and extrapolation performance of the models generated by the stated methods, we set up three case studies: **i)** Percentage split testing; **ii)** Quadrant split testing; and **iii)** Radial split testing. These case studies follow different approaches to split the overall data from the experiment [57] into training and test sets, which are further described below. Mean Square Error (MSE) and Relative Error (RE) are the testing metrics used (Equations S-V and S-VI in the Supplementary Material).

A. Percentage Split Testing

This approach randomly splits the whole experimental (ground truth) data set into training and test sets, such that the distribution of data is similar between these two sets. Four sub-cases are created, which vary the ratio of train vs. test points as follows: 90%/10%, 70%/30%, 50%,50%, 30%/70% and 10%/90%. The progressive reduction in training samples used is designed to analyze model performance under growing sparsity of data. Figure 4 (a) gives an illustration of the percentage splitting for 10% training dataset sub-case. To have a fair comparison, the same dataset samples, created by the apriori splits, are used for training and testing each method.

B. Quadrant Split Testing

The quadrant split testing is devised to analyze model performance when the testing samples come from a region that did not contribute any training samples, although both regions have overlapping range in terms of their input vector (albeit very different data distribution by virtue of the split). More specifically, here we use points only from the first quadrant of the y-z plane for training, whereas points from the three other unseen quadrants are used for testing. This data split is expected to be challenging to address with standard regression modeling due to the induced asymmetry in the acoustic field being used to train the models. For instance, due to the geometry of the quad-rotor and its asymmetry along the z-axis caused by the spinning propellers, it is expected to have higher SPL values in the lower two quadrants in 2D space, or the four octants in 3D space with negative z-value. Conversely the symmetry of the UAV in the x-y plane will probably lead to relatively symmetric distribution of SPL values in the x-y space. Note that radial distance between the measurement point and UAV will decrease the SPL value, leading to a relatively symmetric radial distribution. Due to these effects, successful prediction of the output in unseen testing quadrants requires more meaningful capture of the physics of the problem. Figure 4 (b) gives an illustration of the quadrant splitting case study, where the models are trained on the first quadrant and tested on the remaining three quadrants.

C. Radial Split Testing

Figure 4 (c) illustrates the radial splitting case study. This case is designed to particularly test the (radial) extrapolation capability of each model. Here the data is split into training and testing sets purely based on the radial distance of the point in space w.r.t. the noise source (UAV) that is located at the origin. To implement this split, the entire experimental data set is sorted based on their Euclidean distance from the UAV. The most distant point is identified and its distance from the UAV is used as reference. The sample points within 50% of this reference distance are used for training; thus the training envelope represents a sphere centered at the UAV with a radius equal to the 50% of the reference distance. The points outside of this sphere are used for testing.

TABLE VI UAV Acoustic Field: Training settings of OPTMA transfer models and of networks in sequential hybrid and pure data-driven models for percentage split testing with 70% (1209) of the data used for training

Model	# Layers	# Nodes/ layer	Dropout	Learning Rate
OPTMA-PSO*	4	10	–	–
OPTMA-**Net***	3	50	0.1	1×10^{-4}
OPTMA-**Dual***	3	50	0.1	1×10^{-4}
Sequential Hybrid	3	100	0.1	1×10^{-4}
Pure Data-Driven	3	100	0.1	1×10^{-4}

*Transfer network.

VI. Results: UAV Acoustics Modeling

A. Training settings

The settings of the transfer network in the OPTMA architectures and those for the networks in the sequential hybrid and pure data-driven models are summarized in Table VI, using the example of the 70%/30% percentage split case. With each method, we train a total of seven models, one each for the five percentage split cases, the quadrant split case and the radial split case. Note that the original OPTMA-PSO architecture proved to be too inefficient to be trained for this problem (taking multiple hours even for small improvement in the loss function), and hence only one representative training

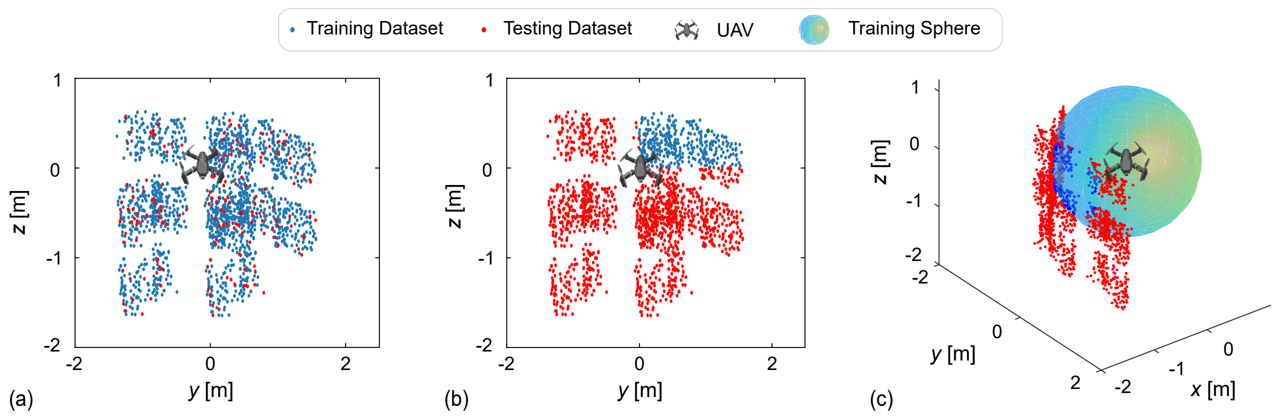

Figure 4 UAV Acoustic Field Train-Test Splitting Case Studies: (a) Scatter plot for Percentage splitting in the y-z plane. This case is for 10% training data. (b) Scatter plot for Quadrant splitting in the y-z plane. The data points in the first quadrant are used to train the model. (c) Radial splitting case, where the data points inside the sphere are used for training, and those outside are used for testing.

TABLE VII UAV Acoustic Field: Testing results in terms of normalized MSE

Test Type	Train Size	Pure DataDriven [13]	Seq Hybrid [13]	OPTMA-**Net** [13]	OPTMA-**Dual**	Partial Phys
Generalization						
Percent	1555 (90%)	0.0083	0.0095	0.0083	0.0088	4.981
Percent	1209 (70%)	**0.0083**	0.0089	0.0086	0.0089	4.766
Percent	864 (50%)	**0.0083**	0.0085	0.0083	0.0085	4.701
Percent	518 (30%)	**0.0087**	0.0091	0.0090	0.0088	4.632
Percent	172 (10%)	0.0095	0.0239	0.0097	0.0095	4.643
Extrapolation						
Quadrant*	347	0.129	0.089	0.034	**0.026**	5.472
Radial**	238	0.055	0.041	**0.013**	0.017	5.067

*Quadrant uses training data only from the first quadrant in the extruded $y-z$ plane. Tests on data from the other three quadrants.

**Radial uses training data only within a 114cm sphere centered on the UAV location. Tests on data outside this sphere (extending radially till ZZZ cm)

instance of OPTMA-PSO is included to specifically allow the comparison of training costs later in this paper.

TABLE VIII Computation times for training OPTMA transfer network models on the UAV Acoustic Field problem for percentage split testing with 70% (1209) of the data used for training.

Model	Computation Time	Error Reduction
OPTMA-PSO*	$5,047$ sec	13%
OPTMA-**Net**	19.3 sec	82.4%
OPTMA-**Dual****	**9.4** sec	84.1%

*Due to very slow training, OPTMA-PSO was not run until convergence.

**Excludes time to train internal network which is reported in Table V.

B. Testing Performance

The testing results of the pure data-driven, the sequential hybrid, OPTMA-**Net** and OPTMA-**Dual** models for the UAV acoustics problem is given in Table VII, in the form of range-normalized MSE. This includes results for the percentage split, quadrant split and radial split case studies. The prediction performance of the partial physics model, with fixed user-prescribed parameter settings, over the corresponding testing data sets is also included. This is to illustrate that the superior performance of the PIML models is not simply an artifact of the (included) partial physics model's prediction capacity.

***Percentage split cases*:** Table VII show that the testing performance of OPTMA-**Net** and OPTMA-**Dual** is close to that of the pure data-driven model for the percentage split testing cases, which essentially encapsulate generalizability. This is attributed to the training set being sufficient to allow a pure data-driven neural network model to accurately capture the acoustic field, as long as data points (even if sampled sparsely) from the entire input space is used for training – i.e., when the training and testing data distributions are similar. Interestingly, the sequential hybrid model is found to offer the poorest generalizability performance, which is likely due to it dependence on the fixed (untuned) partial physics implementation, where the latter is quite inaccurate when not adaptively tuned (as seen from the last column of Table VII).

***Quadrant split case*:** The quadrant test case is expected to be more challenging to predict compared to percentage testing, since training occurs solely on data points in one of the quadrants, and the trained model is required to predict the SPL values in other quadrants. In this test, both OPTMA-**Net** and OPTMA-**Dual** perform substantially better than the baseline models. More specifically, OPTMA-**Dual** provides the smallest testing errors, which is about 3.4 times and 5 times smaller than the testing errors given by the pure data-driven and sequential hybrid models, respectively. For physical clarity, the relative errors in the predicted SPL values for this case are also plotted in Fig. 5, where both the color and circle size represent the magnitude of error. Figure 5 shows that compared to OPTMA-**Dual**, the baseline models suffer from high prediction error for locations in the second and third quadrant visualized in the y-z projection plane. This drop in performance of the baseline models can be attributed to the non identical distributions of the training and testing datasets. These datasets were generated using measurements from different parts of the room at varying distances. For instance, the first quadrant represents the areas closer to the ceiling of the room and the third quadrant represents the areas closer to the floor of the room creating unique acoustic signatures in the various quadrants attributed to the geometric characteristics of the room. On the other hand, spatial adaptation of the partial physics parameters in OPTMA-**Net** and OPTMA-**Dual** allows capturing these differences, lending them a higher prediction accuracy in the quadrant split test.

***Radial split case*:** This test directly assesses each model's extrapolation ability. Table VII shows that OPTMA-**Net** and OPTMA-**Dual** again provide remarkably better extrapolation performance compared to the baselines. Here OPTMA-**Net** gives the smallest testing errors (slightly smaller than that of OPTMA-**Dual**), which is found to be $\sim$ 4 times and 3 times smaller than the testing errors given by the pure data-driven and sequential hybrid models, respectively. The relative error (RE) for OPTMA-**Dual** and the baselines in the radial split case is also visualized vs. increasing distance of the testing points from the source (UAV) in Fig. 6. This figure shows that OPTMA-**Dual**'s extrapolation performance in this case is agnostic to the distance from the source (UAV), while both baselines suffer from higher errors at points farther away from source. The pure data-driven model even exhibits high error regions closer to the source.

Both the quadrant split and radial split testing cases clearly demonstrate the advantage of not only using partial physics, but also adaptively tuning it, as unqiuely performed in OPTMA-**Net** and OPTMA-**Dual**. This advantage is particularly apparent when predicting test data that is from a distribution or spatial region different from those used in training, or is completely outside of the regions used for training (true extrapolation). To provide further insights into how the adaptive tuning of the acoustic partial physics parameters enable this several fold improvement in prediction performance, next we provide further statistical analyses of the transferred features in OPTMA-**Dual**.

C. Acoustic Transfer Feature Analysis

As previously mentioned, the space of tunable parameters includes six parameters per spherical acoustic source (monopole) in the OPTMA-**Dual** implementation on this problem. Thus the design space of tunable parameters become twenty four dimensional $\mathbb{D} \in \mathbb{R}^{24}$, where $\mathbb{D}$ is the design space. Moreover, by virtue of its architecture, OPTMA-**Dual** allows us to retrieve the intermediately predicted values of these parameters for any given spatial location $\mathbf{r}$. Leveraging this capability, we retrieve these parameter values for the 10%/90% percentage split case, and analyze their statistical distribution over the testing data set. These distributions are also illustrated as boxplots in Fig. - S - 2 in the Supplementary Material. From this figure, we notice a variation of the median value of each monopole property across the four monopoles, with the third monopole being the closest and the fourth one being the farthest from the source. This is likely attributed to the need to capture the asymmetric nature of the acoustic field generated by the hovering UAV.

To understand the spatial adaptation of the monopole properties (tunable parameters) achieved by the transfer network in OPTMA-**Dual**, we analyze their predicted values over 3D (r-vector) space (illustrated in Fig.- S - 3 in the Supplementary Material). Here, each row represents one tunable parameter and each column represents one monopole. The parameters plotted are Amplitude, Phase, Frequency, X, Y, and Z coordinates of the monopole ranging from the first row to the last respectively. We observe that the amplitude of the first monopole stays consistently low compared to the other monopoles for various field points which show more variation in the as the field point location is varied. From this figure, we make the following key physical observations: The amplitude of the second monopole shows higher sensitivity to the field points in the upper octant of the 3D space (denoted in light green) while the third and fourth monopoles show lesser sensitivity in lower octants (denoted in blue). Another example is variation in phase (second row) where the first monopole shows higher sensitivity towards the field points with hotter marker color while the fourth monopole shows an inverted behavior. Similar trends could be identified in the different parameters as shown in Fig. - S - 3 in the Supplementary Material. As explained in section IV-E, the parameters act as the normalized latent space values of the acoustic physical parameters which yield a highly accurate predicted value.

D. Training (Computing) Cost Analysis of OPTMA

Lastly, to again highlight the improved training efficiency of OPTMA-**Net** and OPTMA-**Dual**, compared to the original OPTMA-PSO, we analyze the computing time invested in training the transfer network in each of these architectures, taking the exaple of the 70%/30% percentage split case study of the UAV acoustics problem. These computing times are listed in Table VIII. Here OPTMA-PSO run was stopped prematurely due to its inefficiency, during which it achieved just a 13% reduction in error. On the other hand, OPTMA-**Net** and OPTMA-**Dual** both converge with more than 80% error reduction in two orders of magnitude less computing time compared to OPTMA-PSO. This significantly improved efficiency further strengthens the benefit of the direct PyTorch network implementation used in both OPTMA-**Net** and OPTMA-**Dual**, allowing the use of state-of-the-art auto-differentiation for training the transfer network.

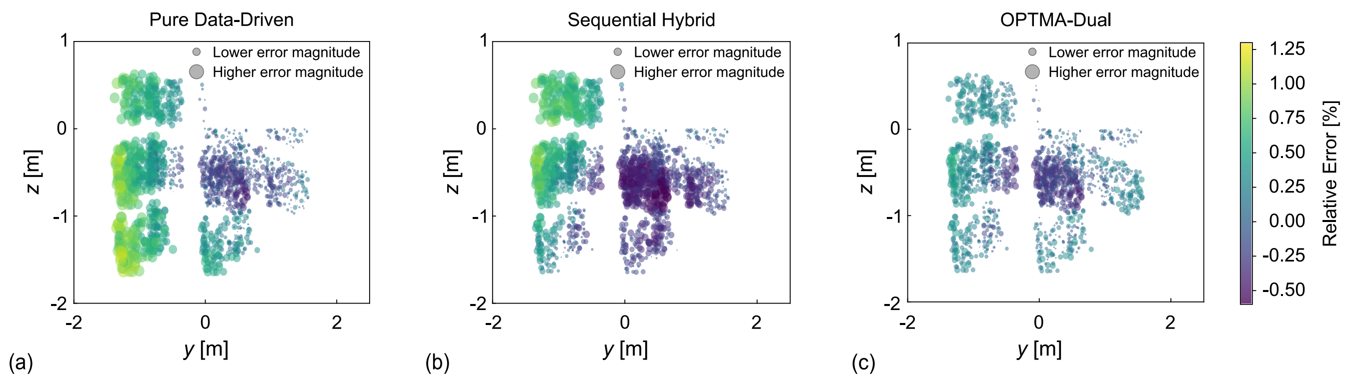

Figure 5 UAV Acoustic Field Testing Results: the relative error (RE) in the **quadrant split** case shown on the *y*-*z* plane for the following models: (a) Pure data-driven model, (b) Sequential hybrid model, and (c) OPTMA-**Dual** model. Size of the circles are directly proportional to the magnitude of error.

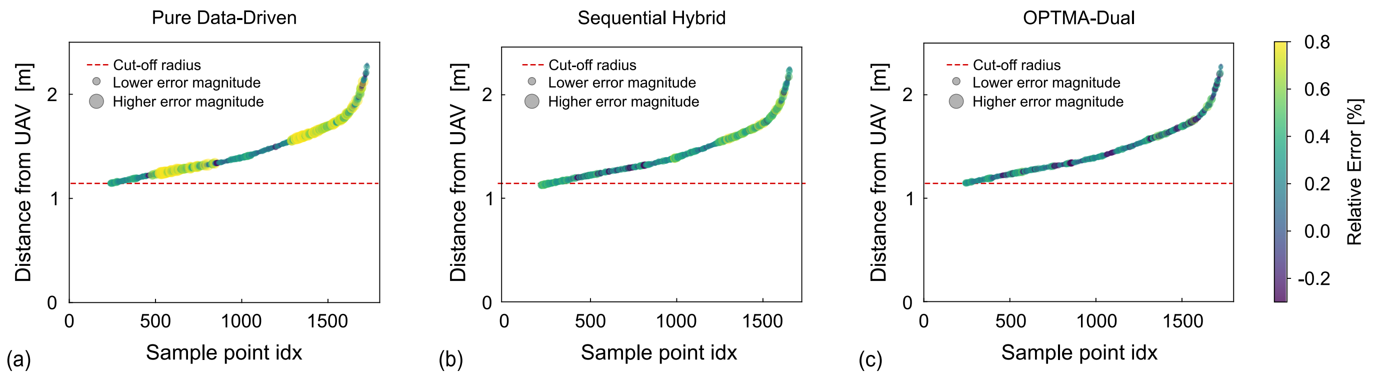

Figure 6 UAV Acoustic Field Testing Results: the relative error (RE) in the **radial split** case for the following models: (a) Pure data-driven model, (b) Sequential hybrid model, and (c) OPTMA-**Dual** model. RE is plotted vs. the index of points sorted in the increasing order of their Euclidean distance from the source (UAV), with color and size of the circles being proportional to the error magnitude. The red dashed cut-off line marks the radius of the sphere within which lies the data points used for training.

VII. Conclusion

OPTMA is a recently reported physics-infused ML (PIML) architecture that transfers the original inputs (via a transfer neural network) into a feature space that informs a simplified partial physics model; this physics model, thus tuned input-adaptively, then makes accurate predictions of the output of interest. This paper presented two advanced variations of OPTMA, called OPTMA-**Net** and OPTMA-**Dual**, both of which are end-to-end PIML architectures, fully implemented in a state-of-the-art ML framework (PyTorch). OPTMA-**Net** incorporates the partial physics model into PyTorch by reprogramming it in the torch tensorial form. OPTMA-**Dual** instead trains a separate internal neural network as a surrogate for the partial physics, which is then connected at the output end of the transfer network. This approach alleviates the need for reprogramming the partial physics which could be both tedious and challenging depending on the nature of the functions embodying the partial physics. Both advancements retain the ability to retrieve physically meaningful transferred features for increased explainability, while offering a substantially faster training process (compared to original OPTMA) by enabling the use of in-built auto-differentiation capabilities in PyTorch. Note that fundamentally OPTMA is a competitive choice of prediction or surrogate modeling architecture in any problem scenario where a fast partial physics model with tunable parameters or terms are available. Now, whether to use OPTMA-**Dual** or OPTMA-**Net** to solve a particular problem depends on the programmability of the partial physics in PyTorch. In situations where the partial physics is viable to reprogram in PyTorch, OPTMA-**Net** should be preferred since an exact mathematical representation of the partial physics can be achieved and greater interpretability of the overall hybrid model is retained. On the other hand, when the partial physics is non-differentiable or challenging to re-program for any other reason (e.g., proprietary model), OPTMA-**Dual** should be used. In cases where reprogramming the partial physics in PyTorch or other Torch-tensorial ML libraries is time-consuming, the model selection is at the user's discretion. To test the performance of OPTMA-**Net** and OPTMA-**Dual**, we compare it with the original OPTMA and two baselines, a pure data-driven model and a sequential hybrid model, over benchmark functions. Results demonstrate that the new versions of OPTMA continue to provide 10-fold improved generalizability performance compared to pure data-driven baselines over sparse to dense sampling scenarios.

The OPTMA architectures and baselines are also applied to a real-world problem of predicting the acoustic field of a quadrotor UAV, by learning from experimental data on sound pressure levels (SPL) at various 3D locations w.r.t. the UAV at the origin. To this end, we specifically implement a multi-monopole based simplified acoustic model, serving as the partial physics, within PyTorch – first reprogrammed in tonsorial form for OPTMA-**Net**, and later surrogated by the internal network in OPTMA-**Dual**. To test the prediction performance of various models on this problem, we create three case studies that respectively assess generalizability, predictability over out-of-distribution testing set, and extrapolability. The latter two capabilities are particularly important in this problem since the acoustic environment around a hovering UAV is asymmetric, and experiments are practically feasible only within smaller spatial envelopes while predictions are needed beyond that envelop for research on design and control of quieter UAVs. In our tests, the generalizability performance of all the models appear to be close to each other. More importantly, for out-of-distribution predictions and extrapolations, both OPTMA-**Net** and OPTMA-**Dual** are found to be around 4-times more accurate than the pure data-driven model and over 2-times more accurate than the sequential hybrid model. Further analyses of the transferred features showed that OPTMA-**Dual** (for example) is able to achieve this significantly improved performance by asymmetrically locating the four monopoles and adapting their properties (e.g., amplitude and frequency) w.r.t. different inputs, i.e., locations where the SPL is being predicted.

Lastly, in both the benchmark problem and the UAV acoustics problem, OPTMA-**Net** and OPTMA-**Dual** presented orders of magnitude faster training processes compared to the original OPTMA-PSO, thereby supporting our primary premise behind developing these newer OPTMA implementations as end-to-end PyTorch based architectures. In their current form, these newer OPTMA architectures still require the specification of which partial physics parameters to be used as transferred features; this prescription could be alleviated in the future by exploring inclusion of classification layers allowing the transfer model to self select the best sub-set of physics parameters to use as transfer features. This capability, along with the ability to use multiple (multi-fidelity) partial physics models, could further expand the use and adoption of the OPTMA PIML architecture.

Acknowledgments

Assistance by Chen Zeng in operating the UAV during testing is gratefully acknowledged. Partial support of this work by the "SMART Start Phase II" program, from the Sustainable Manufacturing & Advanced Robotic Technologies Center of Excellence at the University at Buffalo is also gratefully acknowledged. Support from the NSF Award CMMI-2128578 is gratefully acknowledged. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Nagesh Shukla, Markus Hagenbuchner, Khin Than Win, and Jack Yang. Breast cancer data analysis for survivability studies and prediction. *Computer methods and programs in biomedicine*, 155:199–208, 2018.

[2] Mohammad Bataineh, Timothy Marler, Karim Abdel-Malek, and Jasbir Arora. Neural network for dynamic human motion prediction. *Expert Systems with Applications*, 48:26–34, 2016.

[3] Gowtham Garimella, Joseph Funke, Chuang Wang, and Marin Kobilarov. Neural network modeling for steering control of an autonomous vehicle. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2609–2615. IEEE, 2017.

[4] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017.

[5] Lin Shao, Fabio Ferreira, Mikael Jorda, Varun Nambiar, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Oussama Khatib, and Jeannette Bohg. Unigrasp: Learning a unified model to grasp with n-fingered robotic hands. *arXiv preprint arXiv:1910.10900*, 2019.

[6] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[7] Payam Ghassemi, Kaige Zhu, and Souma Chowdhury. Optimal surrogate and neural network modeling for day-ahead forecasting of the hourly energy consumption of university buildings. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 58134, page V02BT03A026. American Society of Mechanical Engineers, 2017.

[8] Dawn An, Nam H Kim, and Joo-Ho Choi. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. *Reliability Engineering & System Safety*, 133:223–236, 2015.

[9] Emre Artun. Characterizing interwell connectivity in waterflooded reservoirs using data-driven and reduced-physics models: a comparative study. *Neural Computing and Applications*, 28(7):1729–1743, 2017.

[10] Pamela J Haley and DONALD Soloway. Extrapolation limitations of multilayer feedforward neural networks. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 4, pages 25–30. IEEE, 1992.

[11] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[12] D Solomatine, Linda M See, and RJ Abrahart. Data-driven modelling: concepts, approaches and experiences. In *Practical hydroinformatics*, pages 17–30. Springer, 2009.

[13] Rayhaan Iqbal, Amir Behjat, Revant Adlakha, Jesse Callanan, Mostafa Nouh, and Souma Chowdhury. Efficient training of transfer mapping in physics-infused machine learning models of uav acoustic field. In *AIAA SCITECH 2022 Forum*, page 0384, 2022.

[14] Davood Karimi, Haoran Dou, Simon K Warfield, and Ali Gholipour. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical Image Analysis*, 65:101759, 2020.

[15] Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6):741–760, 2021.

[16] Vepa Atamuradov, Kamal Medjaher, Pierre Dersin, Benjamin Lamoureux, and Noureddine Zerhouni. Prognostics and health management for maintenance practitioners-review, implementation and tools evaluation. *International Journal of Prognostics and Health Management*, 8(060):1–31, 2017.

[17] Ali Mehmani, Souma Chowdhury, Weiyang Tong, and Achille Messac. Adaptive switching of variable-fidelity models in population-based optimization. In *Engineering and Applied Sciences Optimization*, pages 175–205. Springer, 2015.

[18] Renato Giorgiani Nascimento and Felipe AC Viana. Fleet prognosis with physics-informed recurrent neural networks. *arXiv preprint arXiv:1901.05512*, 2019.

[19] Rahul Rai and Chandan Sahu. Driven by data or derived throughphysics: Hybrid physics guidedmachine learning approach. *Ieee access*, 2020.

[20] Kamran Javed. *A robust & reliable Data-driven prognostics approach based on extreme learning machine and fuzzy clustering.* PhD thesis, Université de Franche-Comté, 2014.

[21] Kumpati S Narendra and Kannan Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27, 1990.

[22] Chih-Chieh Young, Wen-Cheng Liu, and Ming-Chang Wu. A physically based and machine learning hybrid approach for accurate rainfall-runoff modeling during extreme typhoon events. *Applied Soft Computing*, 53:205–216, 2017.

[23] Vahid Nourani, Mohammad T Alami, and Mohammad H Aminfar. A combined neural-wavelet model for prediction of ligvanchai watershed precipitation. *Engineering Applications of Artificial Intelligence*, 22(3):466–472, 2009.

[24] Shubhendu Kumar Singh, Ruoyu Yang, Amir Behjat, Rahul Rai, Souma Chowdhury, and Ion Matei. Pi-lstm: Physics-infused long short-term memory network. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 34–41. IEEE, 2019.

[25] Shunfeng Cheng and Michael Pecht. A fusion prognostics method for remaining useful life prediction of electronic products. In *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pages 102–107. IEEE, 2009.

[26] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[27] FRANCESCA MANGILI. Development of advanced computational methods for prognostics and health management in energy components and systems. 2013.

[28] Wen Li Zhao, Pierre Gentine, Markus Reichstein, Yao Zhang, Sha Zhou, Yeqiang Wen, Changjie Lin, Xi Li, and Guo Yu Qiu. Physics-constrained machine learning of evapotranspiration. *Geophysical Research Letters*, 46(24):14496–14507, 2019.

[29] SungHo Park, Ki Uhn Ahn, Seungho Hwang, Sunkyu Choi, and Cheol Soo Park. Machine learning vs. hybrid machine learning model for optimal operation of a chiller. *Science and Technology for the Built Environment*, 25(2):209–220, 2019.

[30] Berkcan Kapusuzoglu and Sankaran Mahadevan. Physics-informed and hybrid machine learning in additive manufacturing: application to fused filament fabrication. *Jom*, 72(12):4695–4705, 2020.

[31] Dominic A Rufa, Hannah E Bruce Macdonald, Josh Fass, Marcus Wieder, Patrick B Grinaway, Adrian E Roitberg, Olexandr Isayev, and John D Chodera. Towards chemical accuracy for alchemical free energy calculations with hybrid physics-based machine learning/molecular mechanics potentials. *BioRxiv*, 2020.

[32] Akshay Rai and Mira Mitra. A hybrid physics-assisted machine-learning-based damage detection using lamb wave. *Sādhanā*, 46(2):1–11, 2021.

[33] Ion Matei, Chen Zeng, Souma Chowdhury, Rahul Rai, and Johan de Kleer. Controlling draft interactions between quadcopter unmanned aerial vehicles with physics-aware modeling. *Journal of Intelligent & Robotic Systems*, 101(1):1–21, 2021.

[34] Zhibo Zhang, Rahul Rai, Souma Chowdhury, and David Doermann. Midphynet: Memorized infusion of decomposed physics in neural networks to model dynamic systems. *Neurocomputing*, 428:116–129, 2021.

[35] Xinzhong Chen, Ziheng Yao, Suheng Xu, Alexander S McLeod, Stephanie N Gilbert Corder, Yueqi Zhao, Makoto Tsuneto, Hans A Bechtel, Michael C Martin, G Lawrence Carr, et al. Hybrid machine learning for scanning near-field optical spectroscopy. *ACS Photonics*, 8(10):2987–2996, 2021.

[36] Hong-Cheol Choi, Chuhao Deng, and Inseok Hwang. Hybrid machine learning and estimation-based flight trajectory prediction in terminal airspace. *IEEE Access*, 9:151186–151197, 2021.

[37] Renato G Nascimento, Matteo Corbetta, Chetan S Kulkarni, and Felipe AC Viana. Hybrid physics-informed neural networks for lithium-ion battery modeling and prognosis. *Journal of Power Sources*, 513:230526, 2021.

[38] Agastya P Bhati, Shunzhou Wan, Dario Alfè, Austin R Clyde, Mathis Bode, Li Tan, Mikhail Titov, Andre Merzky, Matteo Turilli, Shantenu Jha, et al. Pandemic drugs at pandemic speed: infrastructure for accelerating covid-19 drug discovery with hybrid machine learning-and physics-based simulations on high-performance computers. *Interface focus*, 11(6):20210018, 2021.

[39] Kajetan Fricke, Renato Giorgiani do Nascimento, and Felipe Viana. Quadcopter soft vertical landing control with hybrid physics-informed machine learning. In *AIAA Scitech 2021 Forum*, page 1018, 2021.

[40] Derek Machalek, Jake Tuttle, Klas Andersson, and Kody M Powell. Dynamic energy system modeling using hybrid physics-based and machine learning encoder-decoder models. *Energy and AI*, page 100172, 2022.

[41] Yangyang Lai, Ke Pan, Yuqiao Cen, Junbo Yang, Chongyang Cai, Pengcheng Yin, and Seungbae Park. An intelligent system for reflow oven temperature settings based on hybrid physics-machine learning model. *Soldering & Surface Mount Technology*, 2022.

[42] Brittny Freeman, Yufei Tang, Yu Huang, and James VanZwieten. Physics-informed turbulence intensity infusion: A new hybrid approach for marine current turbine rotor blade fault detection. *Ocean Engineering*, 254:111299, 2022.

[43] Abhejit Rajagopal, Andrew P Leynes, Nicholas Dwork, Jessica E Scholey, Thomas A Hope, and Peder EZ Larson. Physics-driven deep learning for pet/mri. *arXiv preprint arXiv:2206.06788*, 2022.

[44] King Ankobea-Ansah and Carrie Michele Hall. A hybrid physics-based and stochastic neural network model structure for diesel engine combustion events. *Vehicles*, 4(1):259–296, 2022.

[45] Andreas Maier, Harald Köstler, Marco Heisig, Patrick Krauss, and Seung Hee Yang. Known operator learning and hybrid machine learning in medical imaging—a review of the past, the present, and the future. *Progress in Biomedical Engineering*, 2022.

[46] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[47] Ali Kashefi, Davis Rempe, and Leonidas J Guibas. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Physics of Fluids*, 33(2):027104, 2021.

[48] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[49] Yibo Yang and Paris Perdikaris. Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems. *Computational Mechanics*, 64(2):417–434, 2019.

[50] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[51] Julia Ling, Reese Jones, and Jeremy Templeton. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016.

[52] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[53] Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7):074602, 2018.

[54] Amir Behjat, Chen Zeng, Rahul Rai, Ion Matei, David Doermann, and Souma Chowdhury. A physics-aware learning architecture with input transfer networks for predictive modeling. *Applied Soft Computing*, 96:106665, 2020.

[55] Payam Ghassemi, Amir Behjat, Chen Zeng, Sumeet S Lulekar, Rahul Rai, and Souma Chowdhury. Physics-aware surrogate-based optimization with transfer mapping gaussian processes: for bio-inspired flow tailoring. In *AIAA AVIATION 2020 FORUM*, page 3183, 2020.

[56] Souma Chowdhury, Weiyang Tong, Achille Messac, and Jie Zhang. A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation. *Structural and Multidisciplinary Optimization*, 47(3):367–388, 2013.

[57] Jesse Callanan, Rayhaan Iqbal, Revant Adlakha, Amir Behjat, Souma Chowdhury, and Mostafa Nouh. Large-aperture experimental characterization of the acoustic field generated by a hovering unmanned aerial vehicle. *The Journal of the Acoustical Society of America*, 150(3):2046–2057, 2021.

[58] Beat Schäffer, Reto Pieren, Kurt Heutschi, Jean Marc Wunderli, and Stefan Becker. Drone noise emission characteristics and noise effects on humans—a systematic review. *International Journal of Environmental Research and Public Health*, 18(11):5940, 2021.

[59] Amir Behjat, Steve Paul, and Souma Chowdhury. Learning reciprocal actions for cooperative collision avoidance in quadrotor unmanned aerial vehicles. *Robotics and Autonomous Systems*, 121:103270, 2019.

[60] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *Advances in neural information processing systems*, 32, 2019.
[61] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR, 2020.
[62] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.
[63] Balaji Krishnapuram, Lawrence Carin, Mário AT Figueiredo, and Alexander J Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):957–968, 2005.
[64] Jesse Callanan, Payam Ghassemi, James DiMartino, Maulikkumar Dhameliya, Christina Stocking, Mostafa Nouh, and Souma Chowdhury. Ergonomic impact of multi-rotor unmanned aerial vehicle noise in warehouse environments. *Journal of Intelligent & Robotic Systems*, 100(3):1309–1323, 2020.
[65] Zhu Chen, Xiao Liang, and Minghui Zheng. Including image-based perception in disturbance observer for warehouse drones. *arXiv preprint arXiv:2007.02907*, 2020.
[66] J. Wilke. *A Drone Program Taking Flight*. Amazon: Bellevue, WA, USA, 2019.
[67] Revant Adlakha, Wansong Liu, Souma Chowdhury, Minghui Zheng, and Mostafa Nouh. Integration of acoustic compliance and noise mitigation in path planning for drones in human-robot collaborative environments. *Journal of Vibration and Control*, pages (accepted, in press), 2022.
[68] Kurt Heutschi, Beat Ott, Thomas Nussbaumer, and Peter Wellig. Synthesis of real world drone signals based on lab recordings. *Acta Acustica*, 4(6):24, 2020.
[69] John Kennedy, Simone Garruccio, and Kai Cussen. Modelling and mitigation of drone noise. *Vibroengineering Procedia*, 37:60–65, 2021.
[70] Kai Cussen, Simone Garruccio, and John Kennedy. Uav noise emission—a combined experimental and numerical assessment. In *Acoustics*, volume 4, pages 297–312. MDPI, 2022.
[71] Tommy Langen, Vimala Nunavath, and Ole Henrik Dahle. A conceptual framework proposal for a noise modelling service for drones in u-space architecture. *International Journal of Environmental Research and Public Health*, 19(1):223, 2021.
[72] Bart Van Merriënboer, Olivier Breuleux, Arnaud Bergeron, and Pascal Lamblin. Automatic differentiation in ml: Where we are and where we should be going. *arXiv preprint arXiv:1810.11530*, 2018.
[73] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
[74] Jian Su and John E Renaud. Automatic differentiation in robust optimization. *AIAA journal*, 35(6):1072–1079, 1997.
[75] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.
[76] Andreas Griewank. On automatic differentiation and algorithmic linearization. *Pesquisa Operacional*, 34(3):621–645, 2014.
[77] Christian H Bischof and H Martin Bücker. Computing derivatives of computer programs. Technical report, Argonne National Lab., IL (US), 2000.

Rayhaan Iqbal received his M.S. in Mechanical Engineering from the University at Buffalo (SUNY) in 2021. His contribution to this paper occurred during his M.S. Thesis performed under the supervision of Dr. Souma Chowdhury. His research interests include physics-infused machine learning and design optimization.

Amir Behjat is a Post-Doctoral Research Associate at Purdue University. He received his Ph.D. in Mechanical Engineering from the University at Buffalo (SUNY) in 2021. His contributions to this paper occurred during his Ph.D. He graduated with his BS and MS degrees in Mechanical Engineering from Sharif University of Technology. His research focuses on neuroevolution, physics-Aware machine learning, and UAV collision avoidance.

Revant Adlakha is a Ph.D. student in the Department of Mechanical and Aerospace Engineering at the University at Buffalo (SUNY). He received his B.Tech and M.S. from Symbiosis International University (20xx) and the University at Buffalo (SUNY) (20xx), respectively. His research focuses on structural dynamics, noise control, and wave propagation in acoustic metamaterials.

Jesse Callanan received his Ph.D. from the Department of Mechanical and Aerospace Engineering at the University at Buffalo (SUNY) in 2022, where the research for this paper was conducted. His research focuses on high-precision instrumentation, materials characterization, and advanced manufacturing.

Mostafa Nouh is an Associate Professor of Mechanical and Aerospace Engineering and the Director of the Sound and Vibrations Lab at the University at Buffalo (SUNY). He received his M.S. and Ph.D. in Mechanical Engineering from the University of Maryland. His research focuses on structural dynamics and acoustics with applications in phononic crystals and metamaterials, thermoacoustic energy generation, and non-reciprocal mechanics. He is a recipient of the NSF CAREER award, ASME's Gary Anderson Early Achievement award, and the University at Buffalo's Young Investigator award. He currently serves as the vice-chair of the ASME Noise Control and Acoustics Division.

Souma Chowdhury is an Associate Professor of Mechanical and Aerospace Engineering at the University at Buffalo (SUNY). Dr. Chowdhury received his B.S., M.S. and Ph.D. in Mechanical Engineering, respectively from IIT Kharagpur in India, Florida International University in Miami and Rensselaer Polytechnic Institute in Troy. His research interests lie at the intersection of multi-fidelity optimization and machine learning with applications to the design and control of autonomous systems, swarm robotics and energy systems. He has co-authored 145 articles in leading journals and full-length conference proceedings in related areas. His research has been sponsored by the NSF, DARPA, ONR, NASA and AFOSR.

Supplementary Material

S-I. Analytical Test Problem: Gramacy & Lee Function

A. Partial Physics And Full Physics Functions

Gramacy & Lee function is used in this paper as an analytical test problem to analyze the performance of OPTMA-**Net** and OPTMA-**Dual**. Equations (S-I), (S-II), and (S-III) define the partial physics function ($F_{\mathtt{PP}}$) and the two full physics functions ($F_{\mathtt{FP1}}$ and $F_{\mathtt{FP2}}$), respectively. Note that, the given partial physics model has a more strongly correlated output with that of the first full physics model, while the input space of the partial physics model and that of the second full physics model is relatively more strongly correlated. Further explanation of these characteristics can be found in [59].

Partial Physics Model (PP):

$$F_{\mathtt{PP}}(x) = \frac{\sin(10\pi x)}{2x} + (x-1)^4;\ x \in [-2, 2.5] \tag{S-I}$$

Full Physics Model 1 (FP1):

$$\begin{aligned} F_{\mathtt{FP1}}(x) =& F_{\mathtt{PP}}(3-x) \\ F_{\mathtt{FP1}}(x) =& \frac{\sin\left(10\pi[3-x]\right)}{2(3-x)} + \left([3-x]-1\right)^4;\ x \in [0.5, 2.5] \end{aligned} \tag{S-II}$$

Full Physics Model 2 (FP2):

$$\begin{aligned} F_{\mathtt{FP2}}(x) =& F_{\mathtt{PP}}\Big\{0.5 + 2\sin\left(\pi[x-2]/2\right)\Big\} \\ F_{\mathtt{FP2}}(x) =& \frac{\sin\Big(10\pi\{0.5 + 2\sin\left(\pi[x-2]/2\right)\}\Big)}{2\{0.5 + 2\sin\left(\pi[x-2]/2\right)\}} \\ &+ \Big\{\left[0.5 + 2\sin\left(\pi\langle x-2\rangle/2\right)\right] - 1\Big\}^4;\ x \in [0.5, 2.5] \end{aligned} \tag{S-III}$$

where $F_{\mathtt{PP}}$ is the partial physics model, $F_{\mathtt{FP1}}$ is the first full physics model and $F_{\mathtt{FP2}}$ is the second full physics model.

B. Baseline models for Comparison

1) Pure Data-Driven ML model: To firstly highlight the benefit of infusing partial physics, the OPTMA architectures are compared with a pure data-driven neural network baseline. As the name suggests, this pure data-driven network simply takes the input and directly predicts the output quantities of interest, and is trained in the standard supervised manner over any set of ground-truth samples.

2) Sequential Hybrid Physics Infused Model: The sequential hybrid model is adopted from [57], and represents a different kind of PIML model that involves a front-end infusion of physics into the ML. More specifically, in this sequential hybrid model, the inputs (X) to the neural network include both the original input vector (x) and the output(s) of the partial physics model acting on this same input vector ($f_{\mathtt{PP}}(x)$), i.e., $X = [x, f_{\mathtt{PP}}(x)]$. Here, any constants of coefficients within the partial physics model are set at user-prescribed values. In this sequential hybrid model, it is thus also possible to include multiple instances of the partial physics model, each with a different set of prescribed values for the coefficients. In that case, the neural network model receives the output of each of those partial physics instances as additional input features. For the analytical problem, only a single instance of $f_{\mathtt{PP}}$ is used for simplicity.

C. RMSE Calcualation

The RMSE calculation for the Gramacy & Lee problem is shown in Equation S-IV:

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{n_S}(Y_i^{\mathtt{FP}} - Y_i^{\mathtt{ML}})^2}{n_S}} \tag{S-IV}$$

Here n_S is the size of the test dataset, $Y_i^{\mathtt{FP}}$ is the full physics output of the i^{th} test sample, and the generic $Y_i^{\mathtt{ML}}$ represents the output predictions by an ML model (OPTMA or a baseline model) in response to the inputs of that test sample. Here all the output values are normalized by a reference value, and hence the RMSE provide a sense of the error as a fraction of this reference value. The observed ranges $[-1, 5]$ and $[-3, 40]$ of the full physics outputs across the training samples were used as this reference for FP1 and FP2 respectively.

S-II. OPTMA-Net

Model Architecture

OPTMA-**Net** is a fully connected PIML architecture, with an ANN based transfer model and tensorially represented physics based expressions. The transfer model outputs the transfer features which are fed directly to the physics layers of the model. The physics tensorial layers are written as the exact representation of the partial physics function. Hence, the transfer features have real importance in relation to the partial physics function. These can be retrieved from the model for physics based analysis of the problem statement.

PyTorch Modelling

Being a PyTorch architecture, OPTMA-**Net** uses automatic differentiation (AD) for model training. Breaking down a code into a set of fundamental actions or primitives, whose derivatives are established and whereby the chain rule may be applied, is a key component of AD [72]. PyTorch uses Operator Overloading (OO) [73], one of two of the well known AD implementations: i) Operator Overloading (OO) and ii) Source Code Transformation (SCT). Although each implementation has its own merits and demerits, OO is typically simpler to implement. Comprehensive explanation on AD is accessible here [74]–[76]. Detailed description on the mechanism involved in computing gradients can be found in [77]. The loss function back-propagates over the fully connected transfer model and physics tensorial expressions.

Model Complexity

Training OPTMA-**Net** would be substantially slower if the partial physics model were to be computationally expensive when compared to a full physics model of the problem. This embodies the trade-off between a more precise and accurate OPTMA-**Net** model and the resulting increase in computation cost due to a sophisticated partial physics function. In this case, OPTMA-**Net**'s model complexity changes with modifications to the physics based tensorial layers. The computational cost grows with the number of training samples times the number of training epochs times the number of new nodes for each node that is added.

Case Study: UAV Noise problem setup

Listing 1 shows the programmatic implementation of the fully connected network of OPTMA-**Net**, including both the transfer network and the downstream partial physics model (of UAV acoustics) that is re-programmed in torch tensorial form. Listing 2 shows how arbitrary values can taken for the constant parameters in this implementation.

Table - S - I Fixed physical parameters of OPTMA-**Net** implementation for UAV acoustic modelling

Monopole	Frequency $\frac{\omega}{2\pi}$ (Hz)	Phase angle ϕ (deg)	Monopole Location (x,y,z) $\mathbf{q}$ (m, m, m)	Speed of Sound c (m/s)
Monopole 1	175	45	(0.176, 0.176, 0)	343
Monopole 2	175	45	(−0.176, 0.176, 0)	343
Monopole 3	175	45	(−0.176, −0.176, 0)	343
Monopole 4	175	45	(0.176, −0.176, 0)	343

Table - S - II Range of physical input parameters used to generate samples to train OPTMA-**Dual**'s internal network for UAV acoustic modelling

Spatial Location (x,y,z) $\mathbf{r}$	Normalized Amplitude U_n	Frequency $\frac{\omega}{2\pi}$ (Hz)	Phase angle ϕ (deg)	Monopole Location (x,y,z) $\mathbf{q}$ (m)
-2 to 2	-3 to 3	0 to 180	0 to 180	-2 to 2

```python
import torch
import config1 as c     #Import Partial Physics configuration

class Fully_connected(torch.nn.Module):
    def __init__(self, D_in, D_out,config):
        super(Fully_connected, self).__init__()
        self.layers = torch.nn.ModuleList()
        H = config['hidden_layer_size']
        self.norm = torch.nn.BatchNorm1d(D_in)
        self.linear_in = torch.nn.Linear(D_in, H)
        self.dropoutp = config['dropout']
        for i in range(c.Num_layers):
            self.layers.append(torch.nn.Linear(H,H))
        self.drop = torch.nn.Dropout(p=self.dropoutp)
        self.linear_out = torch.nn.Linear(H, D_out)
        self.nl1 = torch.nn.ReLU()

    def forward(self, x):
        #Transfer Model Layers: Standard Layers of desired description
        out = self.linear_in(self.norm(x))
        for i in range(len(self.layers)):
            net = self.layers[i]
            out = self.nl1(self.drop(net(out)))
        out = self.linear_out(out)

        #Partial Physics Layers: Custom PyTorch compatible layers
        P = torch.zeros(out.shape[0],1,
                        dtype=torch.cfloat).to(c.device)
        for n in range (0,4):   #Loop over N=4 monopoles
            r=torch.sqrt(torch.pow(x[:,0]-c.mono_loc[0,n],2)+
                         torch.pow(x[:,1]-c.mono_loc[1,n],2)+
                         torch.pow(x[:,2]-c.mono_loc[2,n],2))
            P[:,0]=P[:,0]+(out[:,n]*torch.cos
                         (c.kappa[n]*r+c.phi[0,n]))/r
        spl = (20* torch.log10(torch.abs(P)/c.P_ref))
        return spl  #Normalize before returning

def l2_loss(input, target):
     loss = torch.nn.MSELoss()
     return loss(input,target)

def make_train_step(model,optimizer,scheduler=None):
    #One step in the training loop
    def train_step(x, y,test=False):
        a=model
        if not test:
            yhat = a(x)
            loss = l2_loss(yhat, y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
        else:
            a = model.eval()
            with torch.no_grad():
                yhat = a(x)
                loss = l2_loss(yhat, y)
            if scheduler:
                scheduler.step(loss)
        return loss.item()
    #Returns the function called during training
    return train_step
```

Listing 1 OPTMA-Net UAV Noise problem setup in PyTorch: Implemented transfer network and partial physics layers

```

import torch
#Partial Physics Parameters
mono_loc=torch.cuda.FloatTensor([[0.176,  -0.176,  -0.176,  0.176],
                            [0.176,  0.176,  -0.176,  -0.176],
                            [0,  0,  0,  0]])

comp_1i = torch.tensor([[0.0 + 1j]]).to(device)
phi = torch.cuda.FloatTensor([[45,  45,  45,  45]])
P_ref = torch.cuda.FloatTensor([[20e-6]])
freq=torch.cuda.FloatTensor([[175,  175,  175,  175]])
pi = torch.acos(torch.zeros(1)).item() * 2
ang_freq = 2*pi*freq[0,:]
kappa = ang_freq/343
```

Listing 2 OPTMA-Net UAV Noise problem setup: Implemented fixed partial physics parameters

S-III. OPTMA-**Dual** Case Study: UAV Noise

Architectures

The training setup of OPTMA-Dual for the UAV Noise problem is shown in Fig. - S - 1. The PyTorch implementation for the same is shown in Listing 3

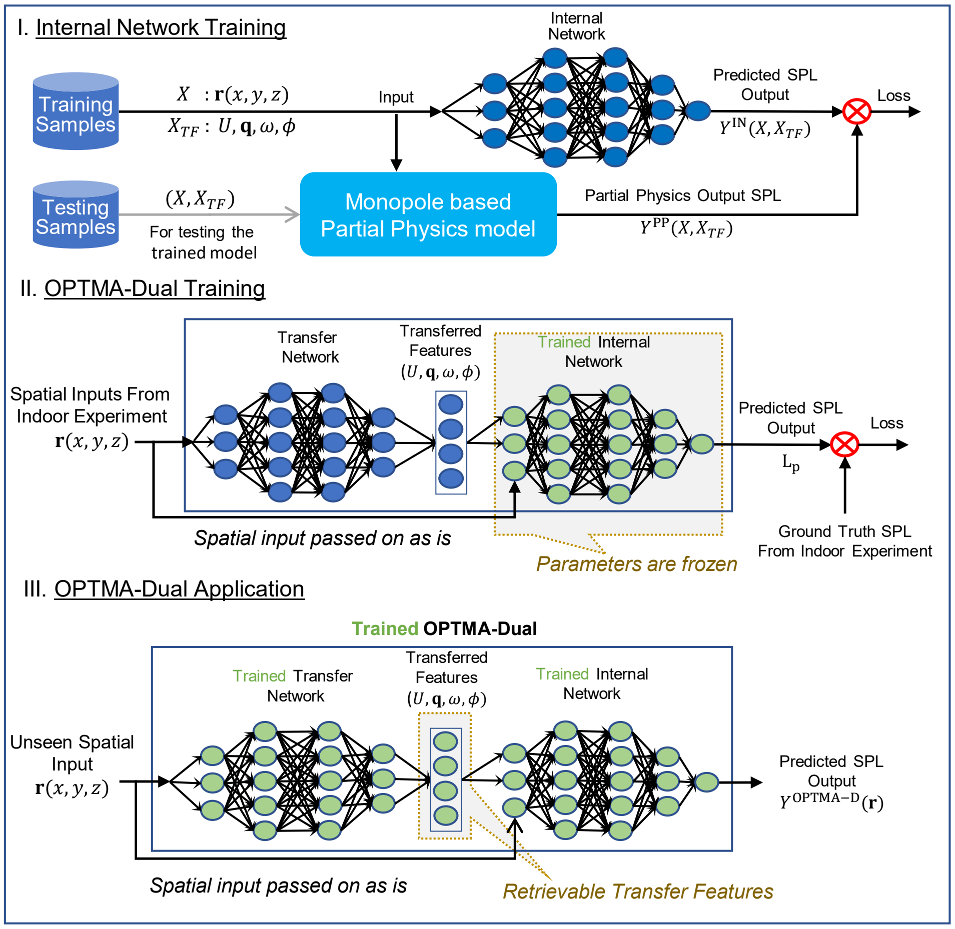
I. Internal Network Training
Training Samples
X : $\mathbf{r}(x,y,z)$
X_{TF} : $U,\mathbf{q},\omega,\phi$
Input
Internal Network
Predicted SPL Output
$Y^{\mathrm{IN}}(X,X_{TF})$
Loss
Testing Samples
(X,X_{TF})
For testing the trained model
Monopole based Partial Physics model
Partial Physics Output SPL
$Y^{\mathrm{PP}}(X,X_{TF})$
II. OPTMA-Dual Training
Spatial Inputs From Indoor Experiment
$\mathbf{r}(x,y,z)$
Transfer Network
Transferred Features
$(U,\mathbf{q},\omega,\phi)$
Trained Internal Network
Predicted SPL Output
L_p
Loss
Ground Truth SPL From Indoor Experiment
Spatial input passed on as is
Parameters are frozen
III. OPTMA-Dual Application
Trained OPTMA-Dual
Unseen Spatial Input
$\mathbf{r}(x,y,z)$
Trained Transfer Network
Transferred Features
$(U,\mathbf{q},\omega,\phi)$
Trained Internal Network
Predicted SPL Output
$Y^{\mathrm{OPTMA-D}}(\mathbf{r})$
Spatial input passed on as is
Retrievable Transfer Features

Fig.-S- 1 UAV Acoustic Field Modeling by OPTMA-**Dual**. **I)** Internal network training using data generated by the 4-monopole partial physics model. **II)** OPTMA-**Dual** training architecture using the spatial location as the input ($\mathbf{r}$), and the acoustic partial physics features (U, $\mathbf{q}$, ω, and ϕ) as the output of the transfer network; the internal network (kept frozen during this stage) takes the original input ($\mathbf{r}$) and the output of the transfer network to predict the SPL (L_p) at that location, which is compared with the experimental SPL data (ground truth) to compute the loss function. **III)** Acoustic OPTMA-**Dual** application with the spatial location ($\mathbf{r}$) as input and SPL (L_p) as predicted output.

```python
import torch
import transfer_config as c     #Import transfer model configuration

# Internal pretrained DNN parameters
import config_DNN as c_DNN
from network_DNN import Fully_connected_DNN

# Load the internal pre-trained DNN
cd_DNN = {
    'network_size' : c_DNN.Num_layers,
    'dropout': c_DNN.dropout,
    'hidden_layer_size':c_DNN.Hidden_layer_size
}
model_par =Fully_connected_DNN(c_DNN.D_in, c_DNN.D_out, cd_DNN)
model_par.to(c_DNN.device)
model_par.load_state_dict(torch.load('internal_DNN/output/trained_model_DNN_new.pt'))
model_par.eval()

# fixing internal network parameters
for params in model_par.parameters():
    params.requires_grad = False

class Fully_connected(torch.nn.Module):
    def __init__(self, D_in, D_out,config):
        super(Fully_connected, self).__init__()
        self.layers = torch.nn.ModuleList()
        H = config['hidden_layer_size']
        self.norm = torch.nn.BatchNorm1d(D_in)
        self.linear_in = torch.nn.Linear(D_in, H)
        self.dropoutp = config['dropout']
        for i in range(c.Num_layers):
            self.layers.append(torch.nn.Linear(H,H))
        self.drop = torch.nn.Dropout(p=self.dropoutp)
        self.linear_out = torch.nn.Linear(H, D_out)
        self.nl1 = torch.nn.ReLU()

    def forward(self, x):
        #Transfer Model Layers
        out = self.linear_in(self.norm(x))
        for i in range(len(self.layers)):
            net = self.layers[i]
            out = self.nl1(self.drop(net(out)))
        out = self.linear_out(out)

        #Call physics based pre-trained internal network
        spl_out = model_par(torch.cat((out,x),axis=1))
        return spl_out

def l2_loss(input, target):
     loss = torch.nn.MSELoss()
     return loss(input,target)

def make_train_step(model,optimizer,scheduler=None):
    #One step in the training loop
    def train_step(x, y,test=False):
        a=model
        if not test:
            yhat = a(x)
            loss = l2_loss(yhat, y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
        else:
            a = model.eval()
            with torch.no_grad():
                yhat = a(x)
                loss = l2_loss(yhat, y)
            if scheduler:
                scheduler.step(loss)
        return loss.item()
    #Returns the function called during training
    return train_step
```

Listing 3 OPTMA-Dual: Network implementation in PyTorch

Model analysis

The predicted acoustic transfer features (monopole properties) by OPTMA-**Dual** are further analyzed using two plots: i) boxplot of the acoustic transfer features' statistical distribution over the testing data set shown in Fig. - S - 2 and ii) plotting the acoustic transfer features' color coded values in a 3D space as shown in Fig. -S - 3. Both of these plots report the monopole properties predicted by OPTMA-Dual for the 10%/90% percentage split case.

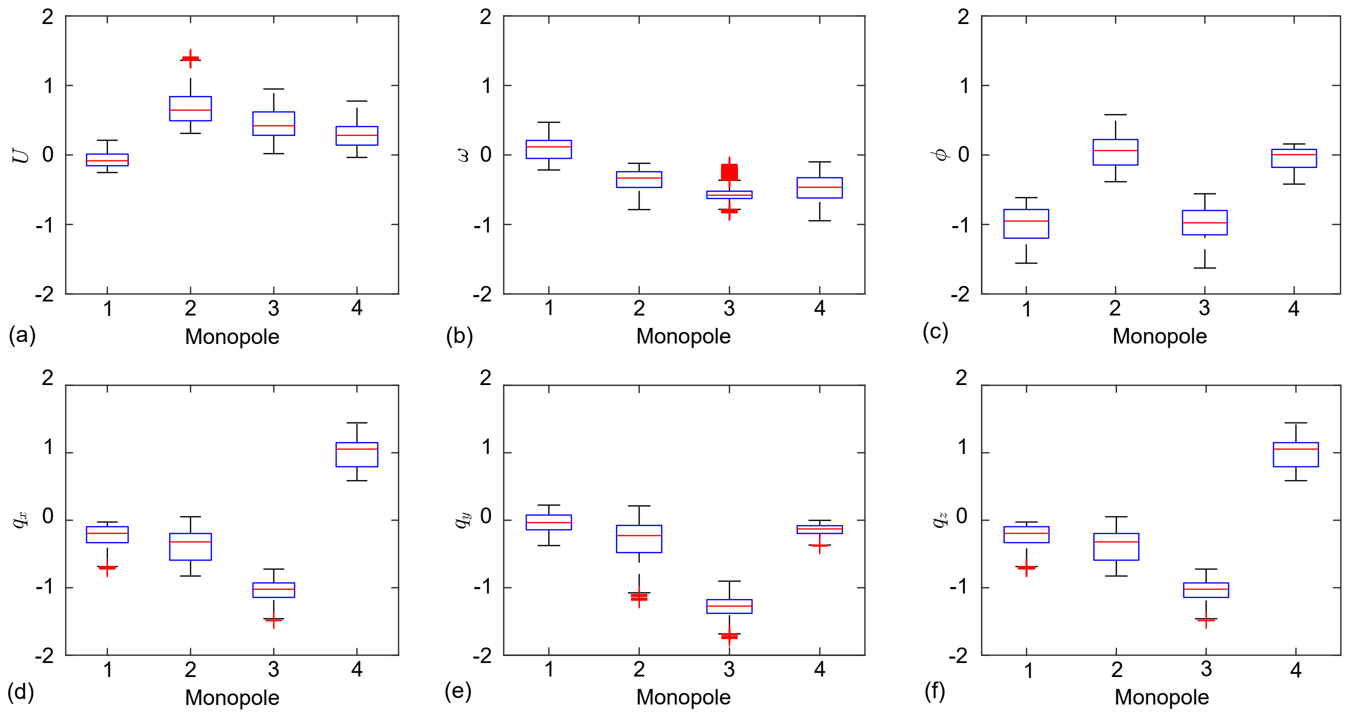

Fig.-S- 2 UAV Acoustic Field – OPTMA-**Dual** for 10%/90% percentage splitting case: Statistical analysis of the various transfer features over the testing data set. Box plot for (a) amplitude (U), (b) frequency (ω), (c) phase (ϕ), (d) x-coordinate (q_x), (e) y-coordinate (q_y), and (f) z-coordinate (q_z), for the four different monopoles. Here, $\mathbf{q} = [q_x, q_y, q_z]^{\mathrm{T}}$. The red line in the box plots indicates the median values, the red markers $'+'$ denote the outliers, the black whiskers extend to the most extreme datapoints which are not regarded as outliers, and the lower and upper limits of the box (blue lines) indicate the 25 and 75 percentile values of the dataset.

Convergence History

The convergence histories of OPTMA-**Dual** and OPTMA-**Net** are plotted in Fig. - S - 4 for the 70%/30% percentage split case. The MSE (Mean Square Error) is reported on the log scale on the y-axis against the epochs on the x-axis.

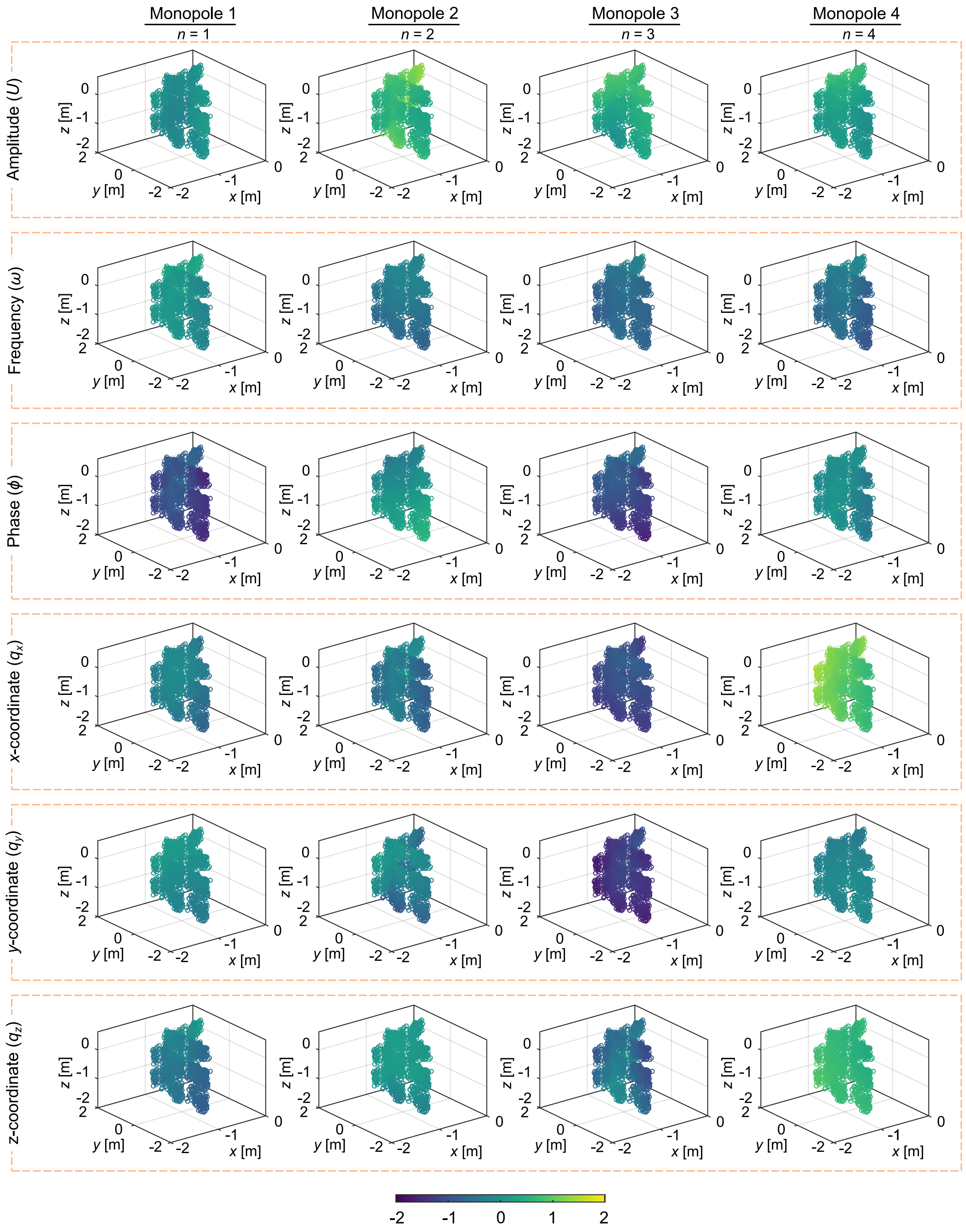

Fig.-S- 3 UAV Acoustic Field – OPTMA-**Dual** for 10%/90% percentage splitting case: Variation of transfer features as a function of spatial location of the testing point. The rows show the various transfer features values for various monopoles which are arranged in a columnar format, with monopole 1 being the far left column and monopole 4 being the far right one. The plots show the values of amplitude (U), frequency (ω), phase (ϕ), x-coordinate (q_x), y-coordinate (q_y) , and z-coordinate (q_z) of the monopole locations, respectively, for all field points $\mathbf{r} \in \mathbb{R}^3$. The color of the data point denotes the value of the parameter as shown in the colorbars.

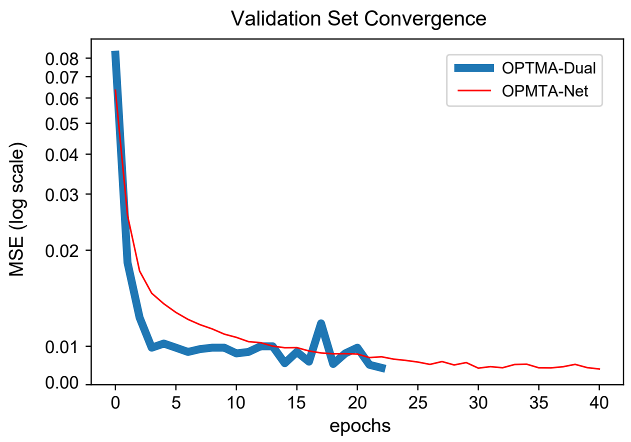

Fig.-S- 4 UAV Acoustic Field: Convergence history of the transfer network training process for OPTMA-**Net** and OPTMA-**Dual**; the history is shown in terms of the validation loss in the 70%/30% percentage split testing case.

S-IV. UAV Acoustic Problem: Testing Metrics

We use the Normalized Mean Square Error (MSE) and the Relative error (RE) as the primary metrics to compare the results of the different ML models for predicting the UAV acoustic field. These error metrics can be expressed as follows:

$$\text{MSE} = \frac{\sum_{n=1}^{n_S}(Y_i^{\text{ExpNrm}} - Y_i^{\text{MLNrm}})^2}{n_S} \tag{S-V}$$

$$\text{RE}_\text{i}(\%) = \frac{(Y_i^{\text{ML}} - Y_i^{\text{Exp}})}{Y_i^{\text{Exp}}} \times 100 \tag{S-VI}$$

where n_S is the total number of samples in the testing dataset, Y_i^{Exp} is the experimentaly recorded ground truth value of a generic i^{th} sample, and Y_i^{ML} is the ML model prediction for this sample. Here, $Y_i^{\text{Exp-Nrm}}$ and Y_i^{MLNrm} respectively represent the normalized values of the ground truth output and the model predicted output. Thus the MSE is calculated over the normalized values of the SPL.