Physics Guided Deep Learning for Drag Force Prediction in Dense Fluid-Particulate Systems

Nikhil Muralidhar [*†] Jie Bu [*†] Ze Cao [‡] Long He [‡] Naren Ramakrishnan [*]
Danesh Tafti [‡] Anuj Karpatne [*†]

Abstract

Physics-based simulations are often used to model and understand complex physical systems in domains like fluid dynamics. Such simulations although used frequently, often suffer from inaccurate or incomplete representations either due to their high computational costs or due to lack of complete physical knowledge of the system. In such situations, it is useful to employ machine learning to fill the gap by learning a model of the complex physical process directly from simulation data. However, as data generation through simulations is costly, we need to develop models being cognizant of data paucity issues. In such scenarios it is helpful if the rich physical knowledge of the application domain is incorporated in the architectural design of machine learning models. We can also use information from physics-based simulations to guide the learning process using *aggregate supervision* to favorably constrain the learning process. In this paper, we propose PhyNet , a deep learning model using *physics-guided structural priors* and *physics-guided aggregate supervision* for modeling the drag forces acting on each particle in a *Computational Fluid Dynamics-Discrete Element Method* (CFD-DEM). We conduct extensive experiments in the context of drag force prediction and showcase the usefulness of including physics knowledge in our deep learning formulation. PhyNet has been compared to several state-of-the-art models and achieves a significant performance improvement of **7.09%** on average . The source code has been made available[*] and the dataset used is detailed in [1, 2].

1 Introduction

Machine learning (ML) is ubiquitous in several disciplines today and with its growing reach, learning models are continuously exposed to new challenges and paradigms. In many applications, ML models are treated as black-boxes. In such contexts, the learning model is trained in a manner completely agnostic to the rich corpus of physical knowledge underlying the process being modeled. This domain-agnostic training might lead to many unintended consequences like the model learning spurious relationships between input variables, or models learning representations that are not easily verifiable as being consistent with the accepted physical understanding of the process being modeled. Moreover, in many scientific disciplines, generating training data might be extremely costly due to the nature of the data generation or collection process. To be used across many scientific applications, it is important for machine learning models to leverage the rich physical knowledge in scientific disciplines to fill the void due to data paucity and be able to learn good process representations in the context of limited data. This makes the model less expensive to train as well as more interpretable due to the ability to verify whether the learned representation is consistent with existing domain knowledge.

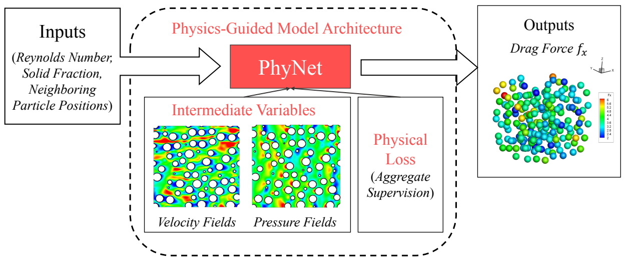


Figure 1: Our proposed PhyNet Model.

In this paper, we present PhyNet and attempt to bridge the gap between physics-based models and machine learning models by incorporating domain knowledge in the design and learning of machine learning models. Specifically, we present three ways for incorporating domain knowledge in neural networks: (1) Physics-guided design of neural network architectures, (2) Learning with auxiliary tasks involving physical intermediate variables, and (3) Physics-guided aggregate supervision of neural network training. Our PhyNet model leverages prior physics theory to learn better representations of the drag forces affecting different particles in assemblies. Specifically, PhyNet has

[*]Dept. of Computer Science, Virginia Tech, VA, USA
[†]Discovery Analytics Center, Virginia Tech, VA, USA
[‡]Dept. of Mechanical Engineering, Virginia Tech, VA, USA
[*]https://github.com/nmuralid1/PhyNet.git

a physics-aware neural network architecture, designed to allow each layer in the network to learn one of the various physical properties that interact to produce the drag force on a particle. This physics-informed architecture design follows a sequential pattern wherein representations learned in earlier layers can be considered to correspond to physical phenomena which have a direct effect on physical phenomena learned in the deeper layers. Such a sequential nature allows the system to learn physically consistent representations. In addition to the novel architecture design, we also introduce aggregate supervision, i.e., we introduce physics-aware statistical constraints during model training, to encourage the learning of more physically consistent representations of complex multi-modal distributions like pressure and velocity field values in the vicinity of each particle in the assembly.

Simulations in computational fluid dynamics (CFD) are expensive to perform and hence generating a large amount of data for training machine learning models is impractical. Hence, one of our primary goals in the paper has been to develop a physics-guided machine learning model that is able to perform effectively under data paucity. The physics-informed nature of the model also helps improve the explainability of the result and allows physics domain experts to verify the consistency model predictions with prior physics knowledge. We showcase this improved explainability of PhyNet through extensive experiments in the paper. This article is an extension of our previous work accepted at SDM 2020 [3] where we introduced the idea of PhyNet and presented some preliminary results showing its efficacy. In this work, we build upon our previous work and introduce several improvements in the technical description of the problem statement and our proposed approach, conceptual modifications in PhyNet to improve its generalization performance, and extensive addition of experimental results to analyze the importance of various components of PhyNet aimed at incorporating physics in machine learning. Here is a summary of the main contributions of our paper:

1. We extend the novel state-of-the-art PhyNet model and improve its representative capacity to model more granular pressure and velocity fields. This is described in Sections 4 and 5.

2. We perform novel experiments to demonstrate the ability of PhyNet to interpolate (see results in Sections 6.1 to 6.4) and to extrapolate to unseen particle assemblies (see results in Section 6.9) and compare model performance of PhyNet to state-of-the-art baselines.

3. We characterize the model performance of PhyNet with increase in granularity of sampled pressure and velocity fields and also the effect of change in particle neighborhood size on the model performance of PhyNet , as described in Sections 6.8 and 6.9.

4. We have developed a sampling procedure for pressure and velocity field sampling around the vicinity of a particle (see Section 5.1). This procedure obeys the periodic boundary conditions that is an inherent property of the simulation domain. This updated sampling procedure allows sampling with increased granularity of the sampled fields used for PhyNet model training.

5. We have also included a detailed description of the Particle Resolved Simulation process in Section 3.

6. Finally, we conduct extensive experimentation to uncover several useful properties of our model in settings with limited data and showcase that PhyNet is consistent with existing physics knowledge about factors influencing drag force over a particle, thus yielding greater model interpretability (see Section 6.5).

The remainder of this paper is organized as follows. Section 2 describes related work at the intersection of physics and machine learning. Section 3 provides the relevant background on the target application of multiphase fluid-particle systems. Section 4 presents our problem formulation and our proposed PhyNet model. Section 5 describes details of the approach used for data generation. Section 6 presents our experimental results while Section 7 presents concluding remarks.

2 Related Work

There have been multiple efforts to leverage domain knowledge in the context of increasing the performance of data-driven or statistical models With the help of physically based priors in probabilistic frameworks [4–6], regularization terms in statistical models [7, 8], constraints in optimization methods [9, 10], and rules in expert systems [11, 12]. In a recent line of research, new types of deep learning models have been proposed (e.g., ODEnet [13] and RKnet [14]) by treating sequential deep learning models such as residual networks and recurrent neural networks as discrete approximations of ordinary differential equations (ODEs).

In [15] the authors explored the idea of incorporating domain knowledge directly as a regularizer in neural networks to influence training and showed better generalization performance. Yaser et al. show hints, i.e., prior knowledge can be incorporated into learning-from-example paradigm [12]. In [16, 17] domain knowledge was incorporated into a customized loss function for weak supervision that relies on no training labels. In a related line of work, physics-informed neural network(PINN) [18, 19] provide a neat idea of how we can train a neural network that follows given PDE constraints. The use of physics-based loss functions to capture monotonic constraints were explored in [20, 21], while [22] including physics-based loss terms to incorporate the principle of energy conservation.

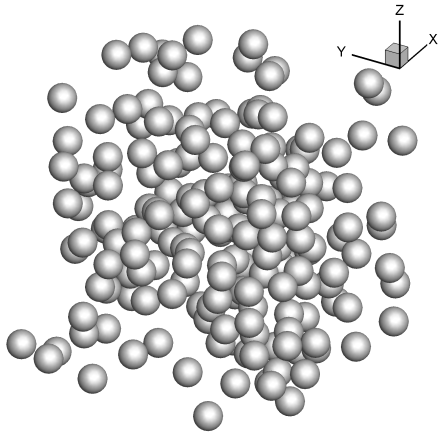
In addition to manipulating loss function, there have been efforts to incorporate prior knowledge into model architecture design, e.g. a low rank structure as *structural prior* was used to designed the convolutional filters in [23]. In [24] the authors propose a neural network model where each neuron learns "laws" similar to physics laws applied to learn the behavior of complex many-body physical systems. In [25], the authors propose a theory that details how to design neural network architectures for data with non-trivial symmetries. The most direct way of using physics priors is explicitly incorporating knowledge as constraints [26]. However, in real-world settings where the physics of the problem is not available as closed-form equations, like the problem discussed in this work, it is necessary to incorporate implicit physical rules [27] to enable learning representations consistent with physics laws, e.g., feature invariance [28]. However, none of these efforts are directly applicable to encode the physical relationships we are interested in modeling in our target problem of drag force prediction, where the relationship between the input variables (neighborhood of particles around a target particle) and the output variable (drag force experienced by the particle) is not explicitly available in the form of a closed-form physical equation.

3 Multiphase Fluid-Particle Systems

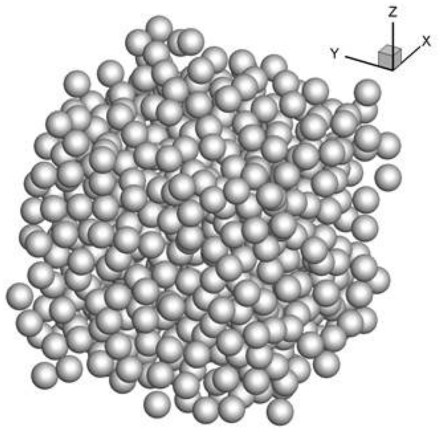
Multiphase fluid-particle systems play a critical role in propulsion, energy, pharmaceutical, food processing, and environmental applications. Particles take the form of solid or liquid fuel droplets in combustion systems, biomass particles in fluidized bed reactors, catalytic agents or ore particles in chemical processing, pill processing in pharmaceuticals, sediment in river beds, and dust, toxins and pollutants in the atmosphere, to give a few examples. Methods for simulating dense fluid-particle mixtures range from extreme high-fidelity fine-grained simulations where only a few thousand particles [29] can be realistically simulated to coarse-grained methods where billions of particles are simulated in the system [30], but with an accompanying loss in accuracy. In high-fidelity Particle Resolved Simulations (PRS), each particle defined by its shape is resolved in the calculation as an independent entity. As a result, the flow and pressure fields resulting from the presence of the particle are directly available from the simulation. However, PRS is quite expensive and only a few 100s or at most 1000s of particles can be resolved in a calculation utilizing grids of O(10^8) degrees of freedom and utilizing O(10^2) processors or cores. In coarse grained simulations such as the Discrete Element Method (DEM), the particles are treated as point masses and the fluid velocity and pressure fields are not resolved around each individual particle but are only available on a spatially-averaged scale larger than the characteristic size of the particle. With still further coarse graining in the Two Fluid Model (TFM), the particles are not treated as separate entities, but instead are treated as a continuum just like the fluid. In the hierarchy from high-fidelity PRS to low-fidelity TFM, orders of magnitude more particles can be simulated but with progressively more dependence on models which typically result in loss of prediction accuracy. The challenge then is to increase the prediction accuracy of large particle systems without incurring the prohibitive cost of using high-fidelity PRS. Considering the thousand particle extreme fine-grain PRS to be a microcosm of the million or billion particle system of DEM or TFM, respectively, we can use fine-grained knowledge from PRS to inform and build models of sub-scale phenomena in the large system for increased prediction accuracy.

One of the critical interaction forces in fluid-particulate systems that has a large bearing on the dynamics of the system is the drag force applied by the fluid on the particles and vice-versa [31]. The drag force, which results from fluid forces acting on the surface of the particle, can be calculated from PRS with high accuracy. Since the velocity and pressure field surrounding each particle is available in PRS, the resulting drag force on each particle in the suspension can be calculated directly without any approximations. However, this is not the case in coarse-grained models such as DEM and TFM, in which the drag force has to be approximated via models. This is because the particle is not resolved but represented by a point mass proxy in DEM and a continuous medium in TFM. As a consequence, the fluid pressure and velocity field is only resolved on a scale that is much larger than the particle diameter.

For an isolated single spherical particle placed in a flow, the drag force acting on the particle is a function of the approach velocity - (U), the diameter of the particle - (D), density - (ρ) and viscosity - (μ) of the fluid, which are combined to define the Reynolds number ($Re = \rho UD/\mu$). When another particle is placed in the close vicinity, it will influence the flow around the first particle and change the drag force on it. Thus when many particles are present as in a suspension



(a) Particle Assembly Solid Fraction (ϕ) = 0.1



(b) Particle Assembly Solid Fraction (ϕ) = 0.35

Figure 2: Immersed Surfaces of Spherical Particle Assemblies.

of particles, the drag on each particle is influenced by all the other particles. Thus, at the least, the mean drag force acting on a particle in suspension is influenced not only by the Reynolds number but also by the number density of particles in the suspension, which is represented by the solid fraction (ϕ) which is the ratio of the volume occupied by the particles to the total volume. A typical application of the drag model in a DEM or TFM calculation would calculate the single particle drag based on local Reynolds number and then modify the value based on the local solid fraction to estimate the mean drag on a particle in suspension [32–34]. Using the mean drag force based on the local Reynolds number and solid fraction is the current state-of-the-art. However, the mean drag is only a zeroth order approximation of the actual drag acting on a particle in suspension.

Given the variability of drag force on individual particles in suspension, this paper explores techniques in physics-guided machine learning to advance the current state-of-the-art for drag force prediction in CFD-DEM by learning from a small amount of PRS data. The PRS simulations are performed using the Immersed Boundary Method (IBM) [35] implemented in a multiblock parallel framework of an in-house computational fluid dynamics software [36,37]. In the IBM instead of having the volume grid conform to each resolved particle, the grid is non-conformal with the surfaces of the particles. Instead a volume Cartesian grid is used at a fine resolution ($\Delta = 1/40$ of particle diameter) and the randomly distributed particles are immersed in the volume grid. The surface of each sphere is defined by 4168 triangular elements. The number of spherical particles in the domain range from 191 to 669 for solid fractions ranging from 0.1 to 0.35 (0.1,0.2,0.3 and 0.35), respectively. For each solid fraction, Re = 10, 50, 100, and 200 are calculated which are in the intermediate regime between Stokes flow and inertial flow. Three different random arrangements are simulated for each solid fraction and Reynolds number with each particle arrangement consisting of 7260 spherical particles.

The PRS calculations are conducted in a fully-periodic cubic domain simulating an unbounded or infinite suspension with flow in the x-direction. A representative particle suspension is shown in Figure 2. The incompressible constant property mass and momentum conservation (Navier-Stokes) equations are solved using a finite volume procedure. Since the volume grid and particle surface grid are completely independent of each other a special procedure is developed for the flow to sense the presence of the particles, which is the essence of the IBM. Using the surface elements of the particle, the background grid cells are divided into fluid cells and solid cells and the grid cells which make up the first layer of fluid cells outside the solid particle are designated as the fluid IB nodes. The IB nodes act as de-facto boundary nodes for the fluid flow calculation such that the no-slip no-penetration fluid boundary condition is satisfied on the particle surface.

After obtaining the flow solution through the interstitial spaces between the spheres in the suspension, the drag force (force applied by fluid on particle in the flow direction) is calculated by direct integration over the particle surface. The forces on the particle surface are made up of viscous shear forces and pressure forces. These are calculated for each surface element and then integrating over all the elements to obtain the viscous and pressure contribution to total drag for each particle.

Collectively, 21,780 unique particle drag force data entries (7260 entries for each of 3 particle assemblies) are obtained from the calculations. For training the neural net, the Reynolds number (Re), solid fraction (ϕ), the locations of the fifteen nearest neighbors of each particle, and the three-dimensional velocity and pressure field through the suspension for each calculation (48) are used.

4 Proposed PhyNet Framework

4.1 Problem Background: Given a collection of N 3D particles suspended in a fluid moving along the X direction, we are interested in predicting the drag force experienced by the i^{th} particle, F_i, along the X direction due to the moving fluid. This can be treated as a supervised regression problem where the output variable is F_i, and the input variables include features capturing the spatial arrangement of particles neighboring particle i, as well as other attributes of the system such as Reynolds Number, Re, and Solid Fraction (fraction of unit volume occupied by particles), ϕ. Specifically, we consider the list of 3D coordinates of 15-nearest neighbors around particle i, appended with (Re, ϕ) as the set of input features, represented as a flat 47-length vector, $\mathbf{A_i}$.

A simple way to learn the mapping from $\mathbf{A_i}$ to F_i is by training feed-forward deep neural network (DNN) models, that can express highly non-linear relationships between inputs and outputs in terms of a hierarchy of complex features learned at the hidden layers of the network. However, black-box architectures of DNNs with arbitrary design considerations (e.g., layout of the hidden layers) can fail to learn generalizable patterns from data, especially when training size is small. To address the limitations of black-box models in our target application of drag force prediction, we present a novel physics-guided DNN model, termed PhyNet , that uses physical knowledge in the design and learning of the neural network, as described in the following.

4.2 Physics-guided Model Architecture: In order to design the architecture of PhyNet , we derive inspiration from the known physical pathway from the input features $\mathbf{A_i}$ to drag force F_i, which is at the basis of physics-based model simulations such as Particle Resolved Simulations (PRS). Essentially, the drag force on a particle i can be easily determined if we know two key physical intermediate variables: the pressure field ($\mathbf{P_i}$) and the velocity field ($\mathbf{V_i}$) around the surface of the particle. It is further known that $\mathbf{P_i}$ directly affects the *pressure component* of the drag force, F_i^P, and $\mathbf{V_i}$ directly affects the *shear component* of the drag force, F_i^S. Together, F_i^P and F_i^S add up to the total drag force that we want to estimate, i.e., $F_i = F_i^P + F_i^S$.

Using this physical knowledge, we design our PhyNet model so as to express physically meaningful intermediate variables such as the pressure field, velocity field, pressure component, and shear component in the neural pathway from $\mathbf{A_i}$ to F_i. Figure 3 shows the complete architecture of our proposed PhyNet model with details on the number of layers, choice of activation function, and input and output dimensions of every block of layers. In this architecture, the input layer passes on the 47-length feature vectors $\mathbf{A_i}$ to a collection of four *Shared Layers* that produce a common set of hidden features to be used in subsequent branches of the neural network. These features are transmitted to two separate branches: the *Pressure Field Layer* and the *Velocity Field Layer*, that express $\mathbf{P_i}$ and $\mathbf{V_i}$, respectively, as 10-dimensional vectors. Note that $\mathbf{P_i}$ and $\mathbf{V_i}$ represent physically meaningful intermediate variables observed on a sequence of 10 equally spaced points on the surface of the particle along the X direction.

The outputs of pressure field and velocity field layers are combined and fed into a 1D Convolutional layer that extracts the sequential information contained in the 10-dimensional $\mathbf{P_i}$ and $\mathbf{V_i}$ vectors, followed by a Pooling layer to produce 4-dimensional hidden features. These features are then fed into two new branches, the *Shear Component Layer* and the *Pressure Component Layer*, expressing 3-dimensional $\mathbf{F_i^S}$ and $\mathbf{F_i^P}$, respectively. These physically meaningful intermediate variables are passed on into the final output layer that computes our target variable of interest: drag force along the X direction, F_i. Note that we only make use of linear activation functions in all of the layers of our PhyNet model following the Pressure Field and Velocity Field layers. This is because of the domain information that once we have extracted the pressure and velocity fields around the surface of the particle, computing F_i is relatively straightforward. Hence, we have designed our PhyNet model in such a way that most of the complexity in the relationship from $\mathbf{A_i}$ to F_i is captured in the first few layers of the neural network. The layout of hidden layers and the connections among the layers in our PhyNet model is thus physics-guided. Further, the physics-guided design of PhyNet ensures that we hinge some of the hidden layers of the network to express physically meaningful quantities rather than arbitrarily complex compositions of input features, thus adding to the interpretability of the hidden layers.

4.3 Learning with Physical Intermediates: It is worth mentioning that all of the intermediate variables involved in our PhyNet model, namely the pressure field $\mathbf{P_i}$, velocity field $\mathbf{V_i}$, pressure component $\mathbf{F_i^P}$, and shear

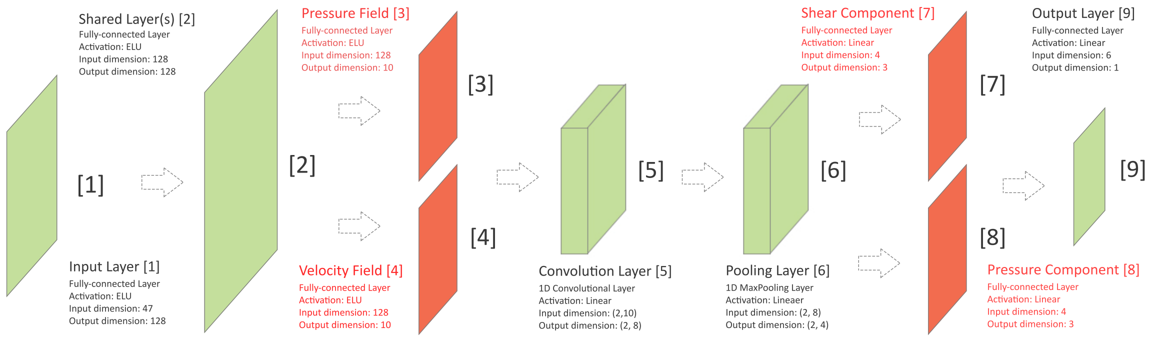


Figure 3: PhyNet Architecture

component $\mathbf{F_i^S}$, are produced as by-products of the PRS simulations that we have access to during training. Hence, rather than simply learning on paired examples of inputs and outputs, $(\mathbf{A_i}, F_i)$, we consider learning our PhyNet model over a richer representation of training examples involving all intermediate variables along with inputs and outputs. Specifically, for a given input $\mathbf{A_i}$, we not only focus on accurately predicting the output variable F_i at the output layer, but doing so while also accurately expressing every one of the intermediate variables $(\mathbf{P_i}, \mathbf{V_i}, \mathbf{F_i^P}, \mathbf{F_i^S})$ at their corresponding hidden layers. This can be achieved by minimizing the following empirical loss during training:

$$Loss_{MSE} = \lambda_P\, MSE(\mathbf{P}, \widehat{\mathbf{P}}) + \lambda_V MSE(\mathbf{V}, \widehat{\mathbf{V}}) +$$
$$\lambda_{FP} MSE(\mathbf{F^P}, \widehat{\mathbf{F^P}}) + \lambda_{FS} MSE(\mathbf{F^S}, \widehat{\mathbf{F^S}}) + MSE(F, \widehat{F})$$

where MSE represents the mean squared error, $\hat{x}$ represents the estimate of x, and λ_P, λ_V, λ_{FP}, and λ_{FS} represent the trade-off parameters in miniming the errors on the intermediate variables. Minimizing the above equation will help in constraining our PhyNet model with loss terms observed not only on the output layer but also on the hidden layers, grounding our neural network to a physically consistent (and hence, generalizable) solution. Note that this formulation can be viewed as a multi-task learning problem, where the prediction of the output variable can be considered as the primary task, and the prediction of intermediate variables can be viewed as auxiliary tasks that are related to the primary task through physics-informed connections, as captured in the design of our PhyNet model.

The generalizability of this architecture is evident by extrapolating the concept of intermediaries to other physically meaningful variables or principles to enhance learning. For instance the product of every CFD simulation is the three-dimensional velocity and pressure field, which are solely responsible for all derived quantities of practical interest. Thus other physically relevant quantities derived from these fields such as velocity and pressure gradients can also be formulated as intermediaries. Additionally in more elaborate settings, principles of mass, momentum, and energy conservation can be included in the loss function to minimize errors in the intermediate variables.

4.4 Using Physics-guided Loss: Along with learning our PhyNet using the empirical loss observed on training samples, $Loss_{MSE}$, we also consider adding an additional loss term that captures our physical knowledge of the problem and ensures that the predictions of our PhyNet model do not violate known physical constraints. In particular, we know that the distribution of pressure and velocity fields over different combinations of Reynolds number (Re) and solid fraction (ϕ) show varying aggregate properties (e.g., different means), thus exhibiting a multi-modal distribution. If we train our PhyNet model on data instances belonging to all (Re,ϕ) combinations using $Loss_{MSE}$, we will observe that the trained model will under-perform on some of the modes of the distribution that are under-represented in the training set. To address this, we make use of a simple form of *physics-guided aggregate supervision*, where we enforce the predictions $\widehat{\mathbf{P}}_{(\mathbf{Re},\phi)}$ and $\widehat{\mathbf{V}}_{(\mathbf{Re},\phi)}$ of the pressure and velocity fields around a particle respectively, at a given combination of (Re,ϕ) to be close to the mean of the actual values of $\mathbf{P}$ and $\mathbf{V}$ produced by the PRS simulations at that combination. If $\overline{P}_{(Re,\phi)}$ and $\overline{V}_{(Re,\phi)}$ represent the mean of the pressure and velocity field respectively for the combination (Re, ϕ), we consider minimizing the following physics-guided loss:

$$Loss_{PHY} = \sum_{Re}\sum_{\phi} MSE(\mu(\widehat{\mathbf{P}}_{(\mathbf{Re},\phi)}), \overline{P}_{(Re,\phi)})$$
$$+MSE(\mu(\widehat{\mathbf{V}}_{(\mathbf{Re},\phi)}), \overline{V}_{(Re,\phi)})$$

The function $\mu(\cdot) : R \to R$ is a *mean* function. We finally consider the combined loss $Loss_{MSE} + Loss_{PHY}$ for learning our PhyNet model.

5 Dataset Description

The dataset used has 7260 particles. Each particle has 47 input features including three-dimensional coordinates for fifteen nearest neighbors relative to the target particle's position, the Reynolds number (Re) and solid fraction (ϕ) of the specific experimental setting (there are a total of 16 experimental settings with different (Re, ϕ) combinations). Labels include the drag force in the X-direction $F_i \in \mathbb{R}^{1\times1}$ as well as variables for auxiliary training, i.e., pressure fields ($\mathbf{P_i} \in \mathbb{R}^{k\times1}$), velocity fields ($\mathbf{V_i} \in \mathbb{R}^{k\times1}$), pressure components ($\mathbf{F_i^P} \in \mathbb{R}^{3\times1}$) and shear components of the drag force ($\mathbf{F_i^S} \in \mathbb{R}^{3\times1}$). The dimension k of the pressure fields ($\mathbf{P_i}$) and velocity fields ($\mathbf{V_i}$) is a hyperparameter indicating the number of samples around the particle vicinity at which to record the pressure and velocity field. Hence, k governs the granularity of the field representation used to train the PhyNet models. We use a pressure and velocity field with $k = 100$ to train our models unless stated otherwise.

Features	Range of Data
$\mathbf{X} \in \mathbb{R}^{15\times1}$	$-5 \sim 5$
$\mathbf{Y} \in \mathbb{R}^{15\times1}$	$-5 \sim 5$
$\mathbf{Z} \in \mathbb{R}^{15\times1}$	$-5 \sim 5$
$Re \in \mathbb{R}^{1\times1}$	{10, 50, 100, 200}
$\phi \in \mathbb{R}^{1\times1}$	{0.1, 0.2, 0.3, 0.35}

Table 1: The 47 input features of the dataset. $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ correspond to the x, y, z coordinates respectively of the nearest neighboring particles of a particular particle of interest. Re is the Reynolds numbers. ϕ is the global solid fraction for the particular experimental setting.

5.1 Pressure & Velocity Field Sampling Methodology We now outline the procedure for sampling the pressure field ($\mathbf{P_i}$) and velocity field ($\mathbf{V_i}$) around a particle p_i. We capture a representation of the pressure and velocity field in the vicinity of the particle through a discrete sampling approach using the equations described in Eq. 5.1.

$$\begin{aligned} \mathbf{q_j^x} &= p_i.x + \epsilon \cdot cos(t) \\ \mathbf{q_j^y} &= p_i.y + \epsilon \cdot sin(t) \\ \mathbf{q_j^z} &= p_i.z \end{aligned} \tag{5.1}$$

The sampling field locations of a particle p_i can be represented by $\mathbf{Q_i} \in R^{k\times3}$. Let $p_i.x$, $p_i.y$, $p_i.z$ represent the x,y and z coordinate respectively, of the center of particle p_i. Then, $\mathbf{Q_i} = \{\mathbf{q_1}^T, ..., \mathbf{q_k}^T\}$ where $\mathbf{q_j} \in \mathbb{R}^{3\times1}$ and $\mathbf{q_j} = \{q_j^x, q_j^y, q_j^z\}$. For each location $\mathbf{q_j} \in \mathbf{Q_i}$, we record the pressure field value and the velocity field value at that point. ϵ is a distance 0.15 units away from the particle surface and is maintained constant throughout our experiments. It must be noted that unlike the pressure field, the velocity field yields a 3-dimensional vector value and we calculate the magnitude of the velocity field vector at each point q_j and use that as the sampled, discrete representation of the velocity field in the vicinity of a particle p_i.

5.2 Experimental Setup All deep learning models used have 5 hidden layers, a hidden size of 128 and were trained for 500 epochs with a batch size of 100. Unless otherwise stated, 55% of the dataset was used for training while the remaining data was used for testing and evaluation. We applied standardization to all the input features and labels in the data preprocessing step.
Baselines: We compare the performance of PhyNet with several state-of-the-art regression baselines and a few close variants of PhyNet .

1. Linear Regression (Linear Reg.), Random Forest Regression (RF Reg.), Gradient Boosting Regression (GB Reg.) [38]: We employed an ensemble of 100 estimators for RF, GB Reg. models and left all other parameters unchanged.

2. DNN: A standard feed-forward neural network model for predicting the scalar valued particle drag force F_i.

3. DNN-MT-Pres: A DNN model which predicts the pressure field around a particle ($\mathbf{P_i}$) in addition to F_i. The pressure and drag force tasks are modeled in a multi-task manner with a set of disjoint layers for each of the two tasks and a separate set of shared layers.

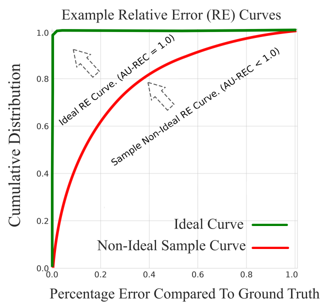
4. DNN-MT-Vel: Similar to DNN-MT-Pres except in this case the auxiliary task models the velocity field around the particle ($\mathbf{V_i}$) in addition to drag force (F_i).

We employ three metrics for model evaluation:

MSE & MRE: We employ the mean squared error (MSE) and mean relative error (MRE) [2] metrics to evaluate model performance. Though MSE can capture the absolute deviation of model prediction from the ground truth values, it can vary a lot for different scales of the label values, e.g., for higher drag force values, MSE is prone to be higher, vice versa. Thus, the need for a metric that is invariant to the scale of the label values brings in the MRE as an important supplemental metric in addition to MSE.

$$MRE = \frac{1}{m}\sum_{i=1}^{m} \frac{|\widehat{F}_i - F_i|}{\overline{F}_{(Re,\phi)}}$$

$\overline{F}_{(Re,\phi)}$ is the mean drag force for (Re, ϕ) setting and $\widehat{F}_i$ the predicted drag force for particle i.



AU-REC: The third metric we employ is the *area under the relative error curve* (AU-REC). The relative error curve represents the cumulative distribution of relative error between the predicted drag force values and the ground truth PRS drag force data. AU-REC calculates the area under this curve. The AU-REC metric ranges between [0,1] and higher AU-REC values indicate superior performance.

Model	**MSE**	**MRE** (%IMP)	**AU-REC** (%IMP)
Linear Reg.	49.80	38.48 (-68.58%)	0.731 (-19.9%)
RF Reg.	32.58	19.38 (-37.62%)	0.819 (-8.08%)
GB Reg.	28.70	18.04 (-32.98%)	0.832 (-6.62%)
DNN	20.77	13.91 (-13.1%)	0.874 (-2.0%)
DNN-MT-Pres	20.83	15.01 (-19.45%)	0.864 (-3.03%)
DNN-MT-Vel	21.02	14.79 (-18.26%)	0.865 (-2.92%)
PhyNet-$F_x^P F_x^S$	**15.01**	12.46 (-2.96%)	0.888 (-0.34%)
PhyNet	15.78	**12.09** (–)	**0.891** (–)

Table 2: We compare the performance of PhyNet and its variant PhyNet-$F_x^P F_x^S$ (only x-components of pressure and shear drag are modeled) with many state-of-the-art regression baselines and show that the PhyNet model yields significant performance improvement over all other models for the particle drag force prediction task. We evaluate model performance in the context of three specific metrics described in Section 5.2. The last column of the table reports the AU-REC metric while the center column reports the mean relative error metric. Both these columns also quantify the percentage improvement of the best performing model i.e PhyNet w.r.t all other models in the context of the specific metric (AU-REC & MRE). We notice that PhyNet models are able to achieve lower errors across all metrics relative to other models.

6 Experimental Results

We conducted multiple experiments to characterize and evaluate the model performance of PhyNet with *physics-guided architecture* and *physics-guided aggregate supervision*. Cognizant of the cost of generating drag force data, we aim to evaluate models in settings where there is a paucity of labeled training data. Our main goals are to generate effective predictions of drag force under data paucity and show consistency of the trained prediction model with known prior domain knowledge. We conduct several experiments to verify the consistency of the intermediate predictions with known physics phenomena thereby ensuring explainability of the model predictions. Finally, we also tackle the challenging problem of extrapolation and characterize the ability of the proposed PhyNet model to extrapolate to unseen settings.

6.1 Physics-Guided Auxiliary Task Selection

When data about the target task is limited, we may employ exogenous inputs of processes that have an indirect influence over the target process to alleviate the effects of data paucity on model training. An effective way to achieve this is through multi-task learning. Table 2 shows the results of several multi-task and single task architectures that we tested in the context of the particle drag force prediction task. It is widely known and accepted in physics that the drag force on each particle in fluid-particle systems such as the one being considered in this paper, is influenced strongly by the pressure and velocity fields acting on the particles [2]. Hence, we wish to explicitly model the pressure and velocity fields around a particle, in addition to the main problem of predicting its drag force. To this end, we design two multi-task models, DNN-MT-Pres, DNN-MT-Vel, as described in section 5.2. We notice that the two multi-task models DNN-MT-Pres and DNN-MT-Vel show inferior performance to the DNN model, however the PhyNet model which is a combination of both the auxiliary tasks is able to outperform the DNN and all other models as shown in Table 2. This improvement in performance may be attributed to the carefully selected auxiliary task and model architecture to aid in learning the representation of the main task.

Statistical Significance Comparison: In order to further verify the validity of model performance, we evaluate the statistical significance of PhyNet predictions relative to the other deep learning architectures mentioned in Table 2. We conducted a two-sided Mann-Whitney-Wilcox rank-sum test [39] which is a non-parametric hypothesis test, in our case indicating whether the difference in performance of a pair of regression models is statistically significant. We notice from Table 3 that the PhyNet model yields statistically significant performance improvements over all the other deep learning architectures, further corroborating our earlier findings in Table 2.

Model	**p-Value**
DNN	0.00039
DNN-MT-Pres	4.74e-8
DNN-MT-Vel	4.519e-8
PhyNet-$F_x^P F_x^S$	0.003769
PhyNet	–

Table 3: Results of Mann-Whitney-Wilcox rank-sum test for statistical significance. Each p-value represents a result of the test performed to compare the statistical significance of PhyNet with every other model. We notice that based on the p-values obtained, we can comfortably conclude that the performance improvement obtained with PhyNet are statistically significant.

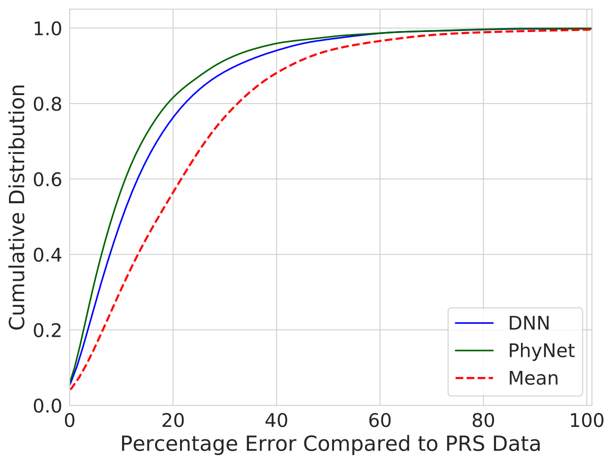


Figure 4: The cumulative distribution function of relative error for all (Re, ϕ) combinations. Overall, the PhyNet model comfortably outperforms the DNN model and the Mean baseline (dotted red line).

6.2 Physics-Guided Learning Architecture Section 6.1 showcases the effectiveness of multi-task learning and of *physics-guided auxiliary task selection* in the context of PhyNet models, for learning improved representations of particle drag force.

We now delve deeper and inspect the effects of expanding the realm of auxiliary tasks. In addition to this, we also use our domain knowledge regarding the physics of entities affecting the drag force acting on each particle, to influence model architecture through *physics-guided structural priors*. As mentioned in Section 4, PhyNet has four carefully and deliberately chosen auxiliary tasks (pressure field prediction, velocity field prediction, predicting the pressure component(s) of drag, predicting the shear components of drag) aiding the main task of particle drag force prediction. In addition to this, the auxiliary tasks are arranged in a sequential manner to incorporate physical inter-dependencies among them leading up to the main task of particle drag force prediction. The effect of this carefully chosen physics-guided architecture and auxiliary tasks can be observed in Table 2. We now inspect the different facets of this physics-guided architecture of the PhyNet model.

We first characterize the performance of our PhyNet models with respect to the DNN and mean baseline. Fig. 4 represents the cumulative distribution of relative error of the predicted drag forces and the PRS ground truth drag force data. We notice that both DNN and PhyNet outperform the mean baseline which essentially predicts the mean value per (Re,ϕ) combination. The PhyNet model significantly outperforms the DNN (current state-of-the-art [2]) model to yield the best performance overall. We also tested DNN variants with dropout and L_2 regularization but found that performance deteriorated. Another important takeaway from Fig. 4 is, we notice that over 80% of the predictions of the PhyNet model have lower than a 20% error with respect to PRS based drag force estimates. The percentage of predictions with less than 20% error is significantly lower in the case of the DNN and *Mean* models.

6.3 Performance With Limited Data Bearing in mind the high data generation cost of the PRS simulation, we wish to characterize an important facet of the PhyNet model, namely, its ability to learn effective representations when faced with a paucity of training data. Hence, we evaluate the performance of the PhyNet model as well as the other single task and multi-task DNN models, on different experimental settings obtained by continually reducing the fraction of data available for training the models. In our experiments, the training fraction was reduced from 0.85 (i.e 85% of the data used for training) to 0.35 (i.e 35% of the data used for training).

Fig. 5 showcases the model performance in settings with limited data. We see that PhyNet model significantly outperforms all other models in most settings (sparse and dense). We note that even for the setting with highest data paucity i.e training fraction 0.35, PhyNet outperforms all other models. The gradient

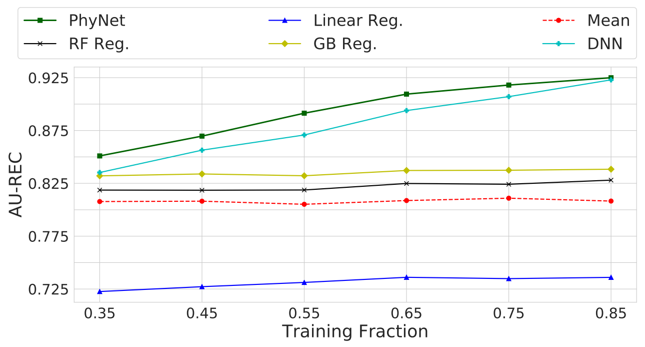


Figure 5: Model performance comparison for different levels of data paucity. We can see that PhyNet outperforms all other models for all training fractions.

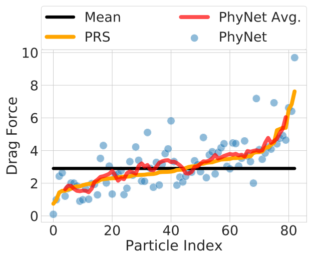
boosting (and all the other regression models except DNN) fail to learn useful information as more data is provided for training. We also notice that the DNN model fails to outperform the PhyNet model for all settings although the performance of the DNN and the PhyNet models is quite comparable for the setting with the highest volume of training data i.e 0.85 training fraction.

6.4 Characterizing PhyNet Performance For Different (Re, ϕ) Settings. In addition to quantitative evaluation, qualitative inspection is necessary for a deeper, holistic understanding of model behavior. Hence, we showcase the particle drag force predictions by the PhyNet model for different (Re,ϕ) combinations in Fig. 6. We notice that the PhyNet model yields accurate predictions (i.e yellow and red curves are aligned). This indicates that the PhyNet model is able to effectively capture sophisticated particle interactions and the consequent effect of said interactions on the drag forces of the interacting particles. We notice that for high (Re,ϕ) as in Fig. 6p, the drag force i.e PRS curve (yellow) is nonlinear in nature and that the magnitude of drag forces is also higher at higher (Re,ϕ) settings. Such differing scales of drag force values can also complicate the drag force prediction problem as it is non-trivial for a single model to effectively learn such multi-modal target distributions. However, we find that the PhyNet model is effective in this setting.
Thus far, we characterized the performance of the PhyNet model in isolation for different (Re,ϕ) contexts. In order to gain a deeper understanding of the performance of PhyNet models for different (Re,ϕ) combinations, we show percentage improvement for the AU-REC metric of PhyNet model and three other models in Fig. 7a - Fig. 7c. We choose DNN, DNN-MT-Pres, DNN-MT-Vel as these are the closest by design to PhyNet among all the baselines we consider in this paper. In Fig. 7, we see that PhyNet outperforms the other models in most of the (Re,ϕ) settings. PhyNet when compared with the DNN model achieves especially good performance for low solid fraction settings which may be attributed to the inability of the DNN model to learn effectively with low data volumes as lower solid fractions have fewer training instances. In the case of the DNN-MT models, the PhyNet model achieves significant performance improvement for low and high solid fraction and Reynolds number cases indicating that PhyNet is able to perform well in the most complicated scenarios (high Re, high ϕ) as well as under data paucity (low ϕ). PhyNet is able to achieve superior performance in 14 out of the 16 (Re, ϕ) settings across all three models.

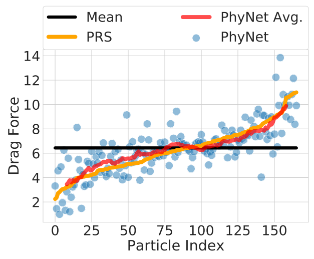
6.5 Verifying Consistency With Domain Knowledge A significant advantage of *physics-guided multi-task architecture design* is the increased interpretability provided by the resulting architecture. Since each component of the PhyNet model has been designed and included based on sound domain theory, we may employ this theoretical understanding to verify through experimentation that the resulting behavior of each auxiliary component is indeed consistent with known theory. We first verify the performance of the pressure and shear drag component prediction task in the PhyNet model. It is well accepted in theory that for high Reynolds numbers, the proportion of the shear components of drag ($\mathbf{F^S}$) decreases [2]. In order to evaluate this, we consider the ratio of the magnitude of the predicted pressure components in the x-direction ($F_x^P \in \mathbf{F^P}$) to the magnitude of the predicted shear components in the x-direction ($F_x^S \in \mathbf{F^S}$) for every (Re, ϕ) setting[†]. The heatmap in Fig. 8 depicts the comparison of this ratio of predicted pressure components to predicted shear components to a similar ratio derived from the ground truth pressure and shear components. We notice that there is good agreement between the predicted and ground truth ratios for each (Re, ϕ) setting and also that the behavior of the predicted setting is indeed consistent with known domain theory as there is a noticeable decrease in the contribution of the shear components as we move toward high Re and high solid fraction ϕ settings.

6.6 Auxiliary Representation Learning With Physics-Guided Statistical Constraints Two of the auxiliary prediction tasks involve predicting the pressure and velocity field samples around each par-

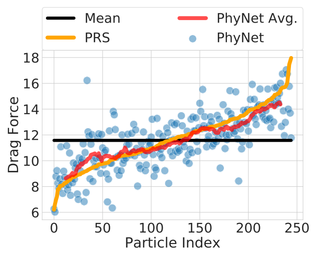
[†]Similar behavior was recorded even when ratios were taken for all three pressure and shear drag components.



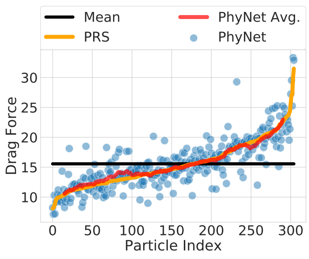
(a) $Re = 10, \phi = 0.1$



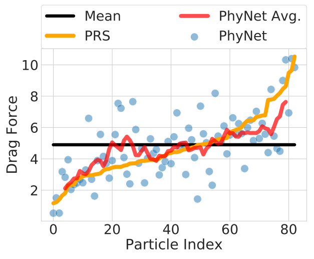
(b) $Re = 10, \phi = 0.2$



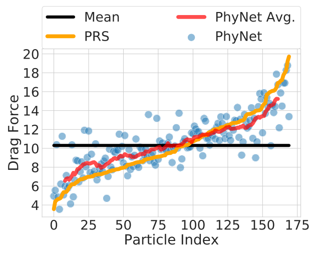
(c) $Re = 10, \phi = 0.3$



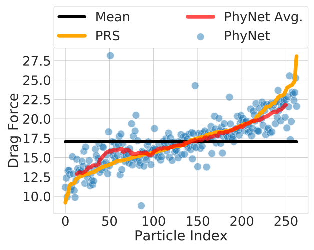
(d) $Re = 10, \phi = 0.35$



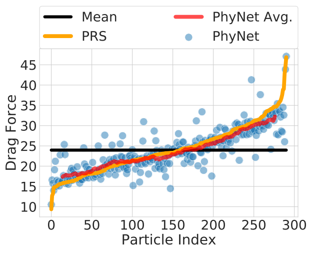
(e) $Re = 50, \phi = 0.1$



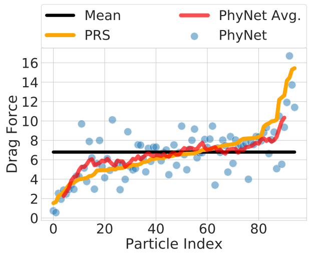
(f) $Re = 50, \phi = 0.2$



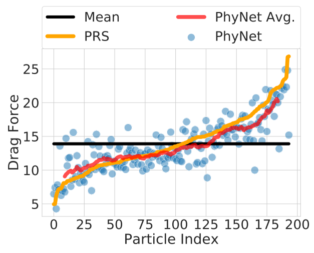
(g) $Re = 50, \phi = 0.3$



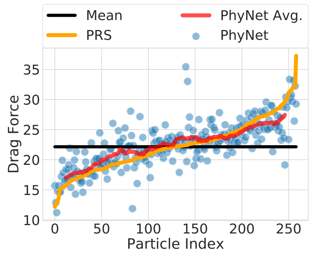
(h) $Re = 50, \phi = 0.35$



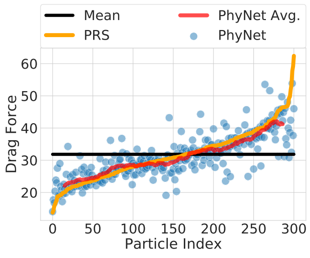
(i) $Re = 100, \phi = 0.1$



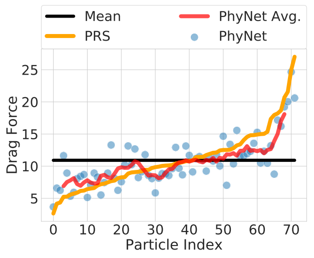
(j) $Re = 100, \phi = 0.2$



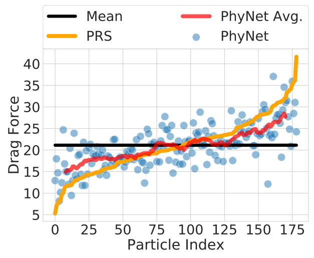
(k) $Re = 100, \phi = 0.3$



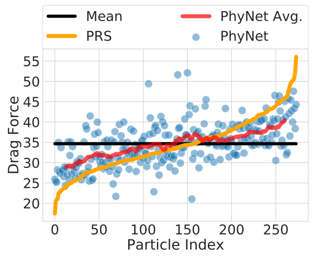
(l) $Re = 100, \phi = 0.35$



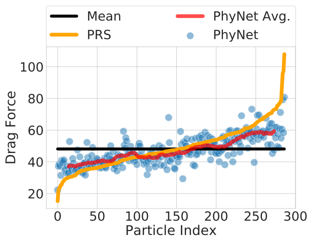
(m) $Re = 200, \phi = 0.1$



(n) $Re = 200, \phi = 0.2$

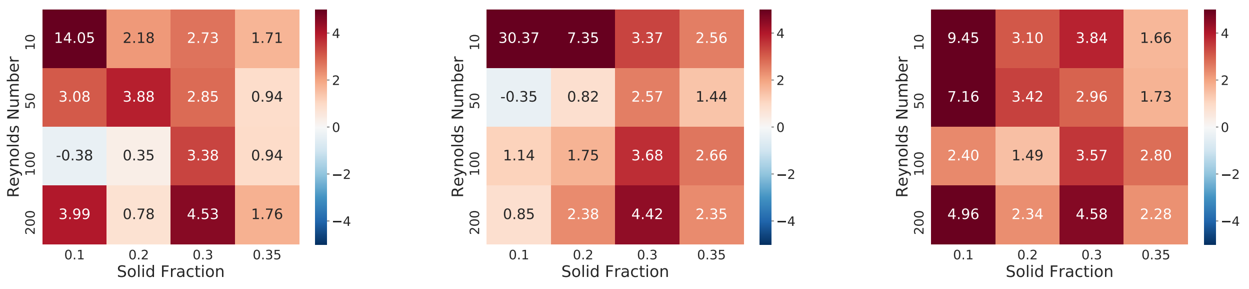


(o) $Re = 200, \phi = 0.3$



(p) $Re = 200, \phi = 0.35$

Figure 6: Each figure shows a comparison between avg. PhyNet predictions (red curve) and ground truth drag force data (yellow curve), for different (Re,ϕ) cases. In each figure, the particle index on the x-axis indicates unique particle IDs assigned in increasing order of predicted drag force per particle. PhyNet Avg. is a rolling average (window size 10) over the individual PhyNet predictions (blue dots) ordered by particle index. We also showcase the mean drag force value for each (Re,ϕ) case (black). The top row of figures indicates experiments conducted with low Re i.e Re=10 and different ϕ values. Notice that as ϕ increases, the number of samples is higher and hence the model is able to achieve a better representation of the corresponding PRS data curve (yellow). We also notice that as Re and ϕ increase, the degree of non-linearity of the system increases due to the increase in complexity of the interactions between the particles. The magnitude of drag forces is also higher at higher Re and ϕ values.



(a) PhyNet vs. DNN (b) PhyNet vs. DNN-MT-Pres. (c) PhyNet vs. DNN-MT-Vel

Figure 7: Each figure indicates the percentage improvement in the context of the AU-REC metric of the PhyNet model over the DNN (Fig. 7a), DNN-MT-Pres (Fig. 7b) and DNN-MT-Vel (Fig. 7c). Red squares show that PhyNet does better and blue squares indicate that PhyNet is outperformed by other models. The figures show that PhyNet yields significant performance improvement over other models. In settings corresponding to low solid fractions, (i.e low number of particles), we notice significant performance improvement of PhyNet over all other models. It is to be noted that the percentage improvement is at least 1.76% over other models even in the most complex modeling setting of Re=200 and $\phi = 0.35$.

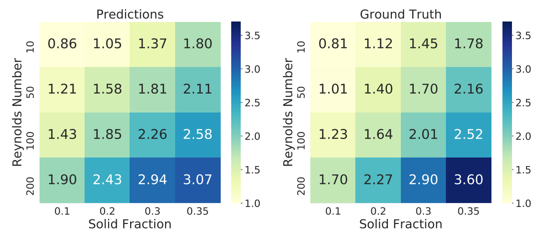


Figure 8: Heatmap with ratio of absolute value of pressure drag (F_x^P) x-component to shear drag (F_x^S) x-component i.e $\left(\frac{|F_x^P|}{|F_x^S|}\right)$. Left figure shows ratio for PhyNet predictions and right figure shows the same ratio for ground truth data. Distribution of ratios in both figures is almost identical.

Training Fraction	No Aggregate Supervision (AUREC)	Aggregate Supervision (AUREC)
0.35	0.83265	**0.85096**
0.45	0.85874	**0.86969**
0.55	0.85635	**0.89138**
0.65	0.9005	**0.91**
0.75	0.86516	**0.918016**
0.85	0.90869	**0.92495**

Table 4: Effect of aggregate supervision on PhyNet for different levels of data paucity. We notice that PhyNet with aggregate supervision outperforms the variant without it in all cases.

ticle. We hypothesized that since the drag force of a particle is influenced by the pressure and velocity fields, modeling them explicitly should help the model learn an improved representation of the main task of particle drag force prediction. In Fig. 9, we notice that ground-truth pressure field PDFs exhibit a grouped structure. Interestingly, the pressure field PDFs can be divided into three distinct groups with all the pressure fields with $\phi = 0.2$ being grouped to the left of the plot, pressure fields with $\phi = 0.1$ being grouped toward the bottom, right of the plot and the rest of the PDFs forming a core (highly dense) group in the center. Hence, we infer that solid fraction has a significant influence on the pressure field. It is non-trivial for models to automatically replicate such multi-modal and grouped behavior and hence we introduce *physics-guided statistical priors* through aggregate supervision during model training of PhyNet . We notice that the learned distribution with aggregate supervision Fig. 9 (center) has a similar grouped structure to the ground truth PDF pressure field. We also obtained the predicted pressure field PDFs of a version of PhyNet trained without aggregate supervision and the result is depicted in Fig. 9 (right). We notice that the PDFs exhibit a kind of *mode collapse* behavior and do not display any similarities to ground truth pressure field PDFs. Similar aggregate supervision was also applied to the velocity field prediction task and we found that incorporating physics-guided aggregate supervision to ensure learning representations consistent with theory, significantly improved model performance. The effect of aggregate supervision is empirically characterized in Table 4 where we compare PhyNet with and without aggregate supervision for different training fractions (0.35 - 0.85) as before. We notice that in all settings PhyNet with aggregate supervision performs better than the variant without aggregate supervision.

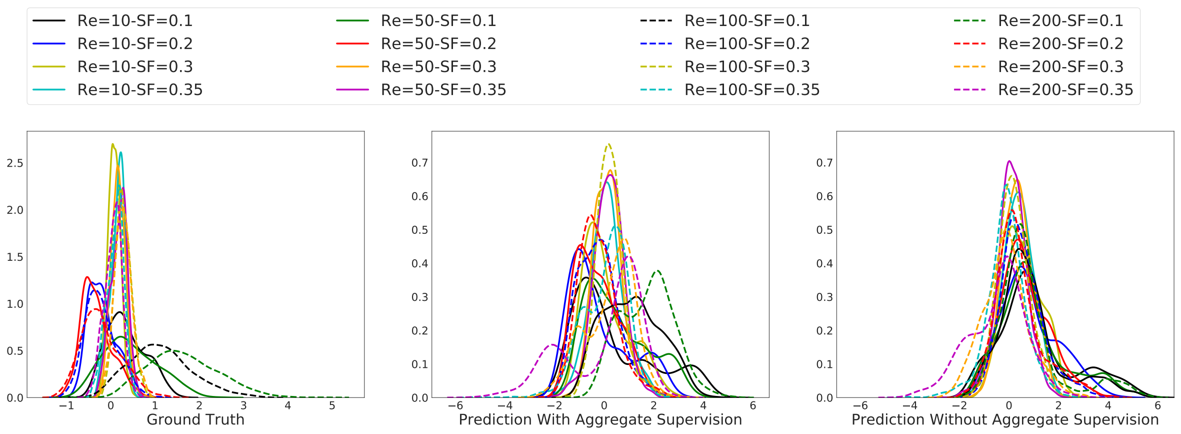


Figure 9: The figure depicts the densities of the ground truth (left) and predicted (center, right) pressure fields of the PhyNet model for each (Re,ϕ). Specifically, we wish to highlight the effect of aggregate supervision (physics-guided statistical prior) on the predicted pressure field. Notice that the PDFs of the pressure fields predicted with aggregate supervision are relatively more distributed similar to the ground truth distribution of pressure field PDFs as opposed to the plot on the right which represents predicted pressure field PDFs in the abscence of aggregate supervision and incorrectly depicts a some what uniform behavior for all the PDFs of different (Re,ϕ) cases.

6.7 Hyperparameter Sensitivity As outlined in Section 4, each of the four auxiliary tasks in the PhyNet model, is governed by a hyperparameter during model training. In our experiments, we only tune the hyperparameters for the pressure field and velocity field prediction tasks leaving all other hyperparameters set to static values for all experiments. We employ a grid search procedure on the validation set to select the optimal hyperparameter values for the pressure and velocity field prediction auxiliary tasks in the PhyNet model. In order to characterize the effect of this hyperparameter selection procedure on the model evaluation, we evaluate the sensitivity of the model to different hyperparameter values.

Training Fraction	λ_P	λ_V
0.35	$1e^{-1}$	$1e^{-4}$
0.45	$1e^{-3}$	$1e^{-3}$
0.55	$1e^{-1}$	$1e^{-4}$
0.65	$1e^{-2}$	$1e^{-4}$
0.75	$1e^{-5}$	$1e^{-3}$
0.85	$1e^{-5}$	$1e^{-2}$

Table 5: The table showcases hyperparameter values of PhyNet , for different levels of training fractions each obtained through gridsearch. It must be noted that only the hyperparameters for the pressure and velocity field prediction auxiliary tasks were tuned and the rest of the values were kept constant for all experiments $\lambda_{FP} = 0.01, \lambda_{FS} = 0.01$.

We design the hyperparameter sensitivity experiment

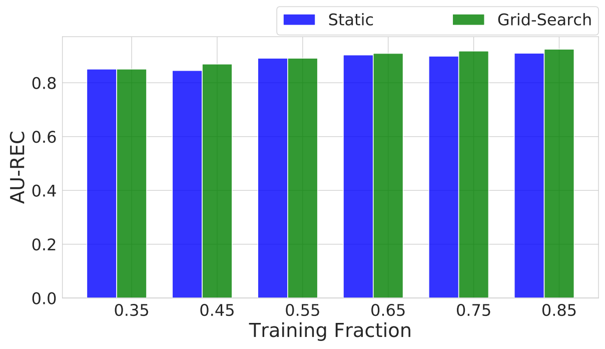


Figure 10: Hyperparameter sensitivity evaluation of the grid search hyperparameter selection procedure for the PhyNet model. We notice that PhyNet is robust to different settings of hyperparameters as we do not see significant changes in the AU-REC between the settings where hyperparameters for the PhyNet were selected through grid search on the validation set (green) and the settings wherein the hyperparameter values were set to a constant value by hand before the experiment (blue) i.e no parameter tuning.

to inspect how model performance varies with different training fractions (i.e different experimental settings). We conduct an experiment by reducing the training fraction from 0.85 to 0.35. Fig. 10 shows the results of our experiment wherein the blue bars indicate the AU-REC values obtained when the PhyNet model was trained with a static (predefined) set of hyperparameters[‡]. The green bars indicate the setting where the

[‡]The optimal hyperparameters for the 0.55 training fraction

optimal hyperparameters for pressure and velocity field prediction for the PhyNet model were obtained through gridsearch on the validation set. We notice that over all the training fractions, there is no significant difference between the two models and hence conclude that the PhyNet model is robust across different hyperparameter settings. Exact hyperparameter values are detailed in Table. 5.

6.8 Effect of Pressure & Velocity Sampling Methodology In Table 6, we characterize the model performance with different sampling frequencies of the pressure and velocity fields around each particle. The sampling plane is the XY plane with the Z axis aligned with the particle center. We notice from the results in Table 6, that the model performance improves with increasing sampling frequency indicating that higher sampling frequencies capture the overall pressure and velocity fields in a more representative manner.

Num. Samples	AUREC	MRE (% IMP)
10	0.88241	13.03 (-7.2%)
20	0.88694	12.56 (-3.8%)
30	0.88575	12.71 (-4.88%)
40	0.88852	12.43 (-2.74%)
100	**0.89138**	**12.09** (–)

Table 6: Effect of pressure & velocity sampling rates on drag force prediction. We can observe that the learned model representation improves with increase in sampling frequency and the model with sampling frequency 100 yields the best performance. We notice that the most granular pressure and velocity field sampling procedure (100 samples) yields an improvement of **7.2%** over the coarse grained pressure and velocity sampling procedure (10 samples).

6.9 Effect of Neighborhood Size & Extrapolation to Unseen Assemblies Extrapolation is a challenging task for machine learning models and is the ultimate test of generalizability of a learned representation. We conducted experiments to evaluate the generalization capability of our PhyNet model by testing the model performance in the context of predicting drag forces of unseen particle assemblies. A particle assembly indicates a certain spatial arrangement of particles for a particular (Re,ϕ) case used to perform a CFD experiment. This is important because the spatial arrangement of neighboring particles around a particle of interest, affects the drag forces acting on the particle.

We generated three separate particle assemblies (each with 16 combinations of the same range of (Re,ϕ) settings) and used 55% of one of the particle assemblies for training while the entirety of the other two particle assemblies were held out and used to evaluate model extrapolation performance. Table 7 showcases that both the PhyNet model and the DNN model yield improved performance with larger particle neighborhoods. This indicates that larger neighborhoods enable learning of richer particle interaction information leading to better representation learning. We notice that the PhyNet model is able to outperform the DNN model for higher particle neighborhoods (i.e., cases when 10 ,15 neighboring particles considered as inputs), while the DNN slightly outperforms the PhyNet model for the case with 5 neighbors.

case were used for all other settings.

Neighborhood Size	Model	MSE	MRE(% IMP)
5	DNN	41.64	**22.89**
	PhyNet	**39.563**	22.95
10	DNN	32.613	21.42
	PhyNet	**29.67**	**21.15**
15	DNN	28.447	19.72
	PhyNet	**24.79**	**18.88**

Table 7: Extrapolation to two unseen particle assemblies using different sized particle neighborhoods. The results depict that the PhyNet model outperforms the DNN model in the context of higher neighborhood sizes (10,15 neighbors). PhyNet achieves an average of **2%** improvement over the DNN model in terms of mean relative error, measured across all the extrapolation settings.

7 Conclusion

In this paper, we introduce PhyNet , a physics inspired deep learning model developed to incorporate fluid mechanical theory into the model architecture and propose physics informed auxiliary tasks selection to aid with training under data paucity. We conduct a rigorous analysis to test PhyNet performance in settings with limited training data and find that PhyNet significantly outperforms all state-of-the-art baselines for the task of particle drag force prediction, achieving an average performance improvement of **7.09%** across all models. We verify that each physics informed auxiliary task of PhyNet is consistent with existing physics theory, yielding greater model interpretability. We also introduce a sampling procedure consistent with the periodic boundary condition of the underlying simulation domain, for obtaining a granular sample of the pressure and velocity fields around the particle surface and showcase that the PhyNet model was able to learn higher quality representations of the particle drag force with fine grained pressure and velocity field samples. We also show the effect of augmenting PhyNet with physics-guided aggregate supervision to constrain auxiliary tasks to be consistent with ground truth data. The effect of the size of particle neighborhood on modeling has also been detailed and we notice that larger particle neighborhoods enable better modeling of the drag forces acting on the particle of interest. Finally, we also demonstrate the ability of PhyNet to extrapolate to unseen particle assemblies and wish to conduct additional experiments further characterizing extrapolation ability in yet other settings moving forward. In the future, we also plan to study the effect that upstream and downstream particles have on the pressure, velocity fields and drag force of a particle of interest.

In conclusion, the paper gives a general framework for incorporating physics into machine learning through intermediaries when these intermediaries influence the quantity being modeled but are not available during model deployment. Such situations abound in computational science and engineering when highly resolved simulations are used to develop models to be deployed as "subgrid" models in low resolution calculations. While the PhyNet framework has been demonstrated for finding particle drag in a suspension, the same framework can be deployed for other CFD-based model development efforts in a variety of engineering fields and in fields such as atmospheric and geological sciences.

8 Acknowledgements

This work is supported by the National Science Foundation via grants DGE-1545362 and IIS-1633363.

References

[1] L. He, D. K. Tafti, and K. Nagendra, "Evaluation of drag correlations using particle resolved simulations of spheres and ellipsoids in assembly," *Powder Technology*, vol. 313, 2017.

[2] L. He and D. K. Tafti, "A supervised machine learning approach for predicting variable drag forces on spherical particles in suspension," *Powder technology*, vol. 345, 2019.

[3] N. Muralidhar, J. Bu, Z. Cao, L. He, N. Ramakrishnan, D. Tafti, and A. Karpatne, "Physics-guided design and learning of neural networks for predicting drag force on particle suspensions in moving fluids," *arXiv preprint arXiv:1911.04240*, 2019.

[4] K. C. Wong, L. Wang, and P. Shi, "Active model with orthotropic hyperelastic material for cardiac image analysis," in *Functional Imaging and Modeling of the Heart*. Springer, 2009.

[5] J. Xu *et al.*, "Robust transmural electrophysiological imaging: Integrating sparse and dynamic physiological models into ecg-based inference," in *MICCAI*. Springer, 2015.

[6] H. Denli *et al.*, "Multi-scale graphical models for spatio-temporal processes," in *NeurIPS*, 2014.

[7] S. Chatterjee *et al.*, "Sparse group lasso: Consistency and climate applications." in *SDM12'*. SIAM, 2012.

[8] J. Liu *et al.*, "Accounting for linkage disequilibrium in genome-wide association studies: a penalized regression method," *Statistics and its interface*, vol. 6, no. 1, 2013.

[9] A. J. Majda and J. Harlim, "Physics constrained nonlinear regression models for time series," *Nonlinearity*, vol. 26, no. 1, 2012.

[10] A. J. Majda and Y. Yuan, "Fundamental limitations of ad hoc linear and quadratic multi-level regression models for physical systems," *Discrete and Continuous Dynamical Systems B*, vol. 17, no. 4, 2012.

[11] D. Waterman, "A guide to expert systems," 1986.

[12] Y. S. Abu-Mostafa, "Learning from hints in neural networks," *Journal of complexity*, vol. 6, no. 2, 1990.

[13] T. Q. Chen *et al.*, "Neural ordinary differential equations," in *NeurIPS*, 2018.

[14] M. Zhu, B. Chang, and C. Fu, "Convolutional neural networks combined with runge-kutta methods," *arXiv:1802.08831*, 2018.

[15] A. Karpatne *et al.*, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE TKDE*, vol. 29, no. 10, 2017.

[16] H. Ren *et al.*, "Learning with weak supervision from physics and data-driven constraints." *AI Magazine*, vol. 39, no. 1, 2018.

[17] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge," in *AAAI*, 2017.

[18] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10566*, 2017.

[20] A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," *arXiv preprint arXiv:1710.11431*, 2017.

[21] N. Muralidhar *et al.*, "Incorporating prior domain knowledge into deep neural networks," in *Big Data*. IEEE, 2018.

[22] X. Jia *et al.*, "Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles," in *SDM19'*. SIAM, 2019.

[23] Y. A. Ioannou, "Structural priors in deep neural networks," Ph.D. dissertation, University of Cambridge, 2018.

[24] B. Anderson *et al.*, "Cormorant: Covariant molecular neural networks," *arXiv:1906.04015*, 2019.

[25] R. Kondor and S. Trivedi, "On the generalization of

equivariance and convolution in neural networks to the action of compact groups," *arXiv:1802.03690*, 2018.

[26] J. Z. Leibo *et al.*, "View-tolerant face recognition and hebbian learning imply mirror-symmetric neural tuning to head orientation," *Current Biology*, 2017.

[27] S. Seo and Y. Liu, "Differentiable physics-informed graph networks," *arXiv:1902.02950*, 2019.

[28] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, vol. 807, 2016.

[29] A. Eshghinejadfard, S. A. Hosseini, and D. Thevenin, "Effect of particle density in turbulent channel flows with resolved oblate spheroids," *Computers and Fluids*, vol. 184, p. 2939, 2019.

[30] F. Li, F. Song, S. Benyahia, W. Wang, and J. Li, "Mp-pic simulation of cfb riser with emms-based drag model," *Chem. Eng. Sci.*, vol. 82, no. 12, pp. 104–113, 2012.

[31] J. Li and J. Kuipers, "Gas-particle interactions in dense gas-fluidized beds," *Chemical Engineering Science*, vol. 58, no. 3-6, 2003.

[32] C. Y. Wen, "Mechanics of fluidization," in *Chemical Engineering Progress Symposium Series*, vol. 62, 1966, pp. 100–111.

[33] R. Di Felice, "The voidage function for fluid-particle interaction systems," *International Journal of Multiphase Flow*, vol. 20, no. 1, 1994.

[34] S. Tenneti, R. Garg, and S. Subramaniam, "Drag law for monodisperse gas–solid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres," *International journal of multiphase flow*, vol. 37, no. 9, 2011.

[35] K. Nagendra, D. K. Tafti, and K. Viswanath, "A new approach for conjugate heat transfer problems using immersed boundary method for curvilinear grid based solvers," *J. Comput. Phys.*, vol. 267, pp. 225–246, 2014.

[36] T. D. K, "A scalable parallel computational tool for simulating complex turbulent flows," in *Proc. ASME Fluids Eng. Div.*, 2001.

[37] D. K. Tafti, *Advances in Computational Fluid Dynamics and Heat Transfer: Time-accurate techniques for turbulent heat transfer analysis in complex geometries.* Ed. Amano, R. and Sunden, B., WIT Press, 2011.

[38] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *JMLR*, vol. 12, 2011.

[39] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.