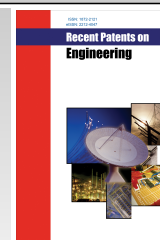



## RESEARCH ARTICLE



# SMART-Flow: Enhancing Fluid Dynamics Simulations with Machine Learning for Scalable Model Reduction and Accurate Prediction



Nishit Kaul<sup>1</sup>, Sameer Kaul<sup>2</sup>, Bharti Bhat<sup>3</sup>, Sheikh Amir Fayaz<sup>4,\*</sup> , Majid Zaman<sup>4,\*</sup>, Waseem Jeelani Bakshi<sup>5</sup> and Bilal Maqbool Beigh<sup>6</sup>

<sup>1</sup>Department of Computer Science, National Society of Professional Engineers, 1420 King Street Alexandria, VA 22314-2794, Virginia, United States; <sup>2</sup>Department of Computer Science, University of Kashmir, Hazratbal, Srinagar, Jammu and Kashmir, India; <sup>3</sup>Department of School Education, Jammu and Kashmir, India; <sup>4</sup>Directorate of IT&SS, University of Kashmir, Hazratbal, Srinagar, Jammu and Kashmir, India; <sup>5</sup>Department of Computer Science & Engineering, University of Kashmir, Hazratbal, Srinagar, Jammu and Kashmir, India; <sup>6</sup>Department of Computer Science and Engineering Cluster University Srinagar, Srinagar, India

**Abstract:** *Aims:* To improve the efficiency of fluid dynamics simulations by applying model reduction and prediction techniques through machine learning, ultimately reducing computational costs while maintaining near-perfect accuracy.

**Background/Introduction:** Fluid dynamics simulations are critical in various engineering and physical science applications. However, their computational complexity often leads to significant time and resource consumption. To address this issue, integrating computational physics with machine learning-based methods offers promising opportunities for performance enhancements.

**Objective:** To develop a machine learning-driven model reduction approach that significantly reduces computational costs while preserving simulation accuracy for fluid dynamics studies.

## ARTICLE HISTORY

Received: December 14, 2024  
Revised: January 30, 2025  
Accepted: February 04, 2025

DOI:  
10.2174/0118722121371511250322051853

**Methods:** The approach combines Principal Component Analysis (PCA) and Proper Orthogonal Decomposition (POD) for initial model reduction. Physics-informed Convolutional Neural Networks (CNNs) output signals are compressed using deep learning-based autoencoders, extracting the key features of fluid dynamics. Machine learning models, including regression methods and neural networks, are trained on this reduced dataset to predict fluid behavior under various conditions. The framework was implemented using Python, with TensorFlow and PyTorch for machine learning and OpenFOAM and ANSYS Fluent for conventional computational fluid dynamics (CFD). Parallel computing was employed to maximize resource utilization during large-scale simulations.

**Results and Discussion:** The integrated framework demonstrated significant improvements in computational efficiency and maintained accuracy when validated against high-fidelity CFD simulations and experimental data.

**Conclusion:** The proposed machine learning-based model reduction approach effectively enhances the accessibility and practicality of fluid dynamics modeling, offering substantial computational gains and maintaining accuracy, thus broadening its application and appeal. The proposed framework also holds potential for patentability, given its innovative integration of machine learning and fluid dynamics modeling.

**Keywords:** Principal component analysis, openFOAM, ANSYS fluent, CFD simulation, proper orthogonal decomposition, neural networks, fluid dynamics, simulation, machine learning enhancement.

\* Address correspondence to this author at the Directorate of IT&SS, University of Kashmir, Hazratbal, Srinagar, Jammum and Kashmir, India; E-mail: [skh.amir88@gmail.com](mailto:skh.amir88@gmail.com)

## 1. INTRODUCTION

The applications of fluid dynamics are in, the aerospace, automobile industry, environment, and biomedical industry respectively. This simulation enables appreciation of the behavior of fluids in specific conditions hence being useful in getting background information when dealing with fluid system designs, safety, and performance. Despite this significance, accurate fluid dynamics simulations are notoriously computationally expensive and time-consuming with a typical requirement of extensive computational resources and long processing times; thus, resulting in bottlenecks in this case especially when real-time analysis and iterative design processes are involved.

The introduction of machine learning (ML) holds much promise for tackling these challenges. Through ML methodologies, it is possible to develop models that vastly simplify fluid dynamics simulations, hence speeding up computations considerably. This study therefore seeks to examine and implement model reduction and prediction based on machine learning to amplify fluid dynamics simulations for easier reaching out to people intending to employ them.

Cutting down on complexity in fluid dynamics simulations is key, and model reduction plays a big part in this. Over the years, scientists have relied on methods like Principal Component Analysis (PCA) and Proper Orthogonal Decomposition (POD) to simplify data and zero in on the main features of fluid flow. These approaches lay the groundwork to compress simulation data, but they have their limits. Because they're linear, they struggle to capture the ins and outs of complex nonlinear dynamics. To get around this problem, we're bringing in deep learning-based autoencoders. These tools can grasp complicated nonlinear patterns in the data, which helps to make model reduction even more effective.

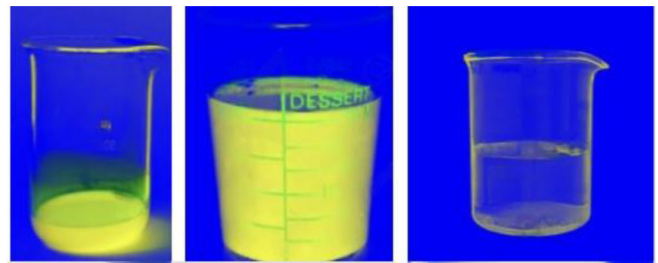
The integration of computer vision with the onset of CNN (Convolutional Neural Networks), has deeply stratified the robust employability of the presented approach, with proper classification of flow structures and stroke patterns.

To implement and validate this hybrid approach, we have used a combination of Python for machine learning and MATLAB or FORTRAN for classical CFD simulations. Machine Learning Models are developed using Tensorflow and Pytorch. The high-fidelity CFD Simulation Data was taken from OpenFoam community use-case dataset. Experimental Fluid Dynamics has been taken from Flow Science Archives, NASA Turbulence modeling resource, and supplementary features from UCI Machine Learning Repository. Image Data has been imported from IMAGENet archive, and large scale data for parallel computation has been developed from recurrent simulations throughout research and also has been derived from XSEDE (Extreme Science and Engineering Discovery Environment) and PRACE (Partnership for Advanced Computing in Europe).

### 1.1. Diverse Nature of Fluids

Many behavior characteristics and different properties make the fluids unique depending on the composition of the

mixture, the state of the flow, and the environment (Fig. 1). This ranges from fluids that can be described by Newton's law of viscosity such as water and air that have linear stress/strain relationships to complicated fluids like blood, gels, and polymers that have time-dependent viscosity and elasticity (Fig. 1). The study applies to both Newtonian and non-Newtonian fluids, the processing of non-Newtonian fluids requires an extra step of calculation of variable viscosity and velocity, using PINN, which is key to observing physics-integrated factors under variable circumstances. It is important to comprehend these various properties to better simulate and forecast the behavior of fluids used in various techniques and for multiple purposes in industries and nature; hence, the a need to advance simulation methods that can capture the variabilities. This also includes turbulent flows, laminar flows multiphase flows, etc.



**Fig. (1).** Shows the behavioral pattern of multiple fluids [Petrol, Blood, Water]. (A higher resolution / colour version of this figure is available in the electronic copy of the article).

### 1.2. Objectives

The purpose of the presented study is to advance the information prediction accuracy and timeliness in fluid dynamic applications and run type simulations *via* machine learning for accurate analysis.

#### 1.2.1. Develop Advanced Model Reduction Techniques

Apply Principal Component Analysis (PCA) and mode Proper Orthogonal Decomposition (POD) to solve the problem of data reduction of high-quality simulation results. Include the deep learning-based autoencoders to capture the complex relations of the process and reduce the simulation data size even more.

#### 1.2.2. Enhance Predictive Modeling

Using regression techniques and neural networks, train machine learning models on these compressed sets of data to predict desired behaviors of fluid dynamics with high levels of precision. Consider the utilization of combined models that augment straightforward and efficient computational physics techniques with machine learning performance estimation. Non-Linear fluid analysis, those with high fidelity, are dealt robustly with development of internal analytic architecture under the proposed neural network architecture. Therefore, the study applies to both linear and non-linear fluid simulations.

### 1.2.3. Integrate Computer Vision, to Interpret Behavior Flow

Apply CNNs to detect the flow patterns in the simulation and experimental data to increase the modality and enlarge the applicability range of the prediction tool. Equip the training framework with real-time analysis so that you can get the knowledge as and when it happens to generate instant insights and decisions.

### 1.2.4. Framework Validation

For the machine learning part, we need to create a completely integrated framework in Python, while for the common other CFD simulations, they can be in MATLAB or FORTRAN as long as they are well integrated with the above machine learning part and well optimized. Perform CFD analysis with the help of tools such as OpenFOAM and ANSYS Fluent that use such methods as FEM, FVM, LBM, etc. To manage such simulations and at the same time ensure that the usage of resources is kept to the lowest level possible then the principles of parallel computing should be used.

### 1.2.5. Validate Against High-Fidelity Simulations and Experimental Data

Rigorously validate the proposed models against high-fidelity CFD simulations and experimental data to ensure accuracy, reliability, and generalizability. Perform comparative analysis to demonstrate the improvements in computational efficiency and predictive accuracy.

## 2. LITERATURE REVIEW

Computational solutions of complicated fluid flow problems involve numerical analysis based on the Navier-Stokes equations. Detailed CFD simulations are high-fidelity simulation analyses that can be resource demanding thus take a lot of time. FEM, FVM, and LBM have been previously applied in CFD; however, they have problems in dealing with extensive and complicated simulations [1]. Model reduction finds its application where it is desired to study the behavior of certain systems by keeping the other secondary effects negligible. The two traditional linear methods of dimensionality reduction in CFD data include Principal Component Analysis (PCA) and Proper Orthogonal Decomposition (POD) [2, 3]. These methods help locate coherent structures in the flow field and in so doing find a lower-dimensional representation of the data. However, they are not very effective in the description of unsteady phenomena that occur in turbulent and multiphase flows.

To cater to this problem, deep learning-based autoencoders have emerged as key tools for nonlinear model reduction. Researchers have triumphantly maneuvered autoencoders to fluid dynamics problems, portraying a high degree of reduced data-size while preserving flow features [4, 5]. Autoencoders are effective when employed for modular learning of dimensional illustration of high-dimensional data, under unsupervised learning processes. Regression models and neural networks have been employed in various fluid

dynamics simulations over the recent decade, progressing over CFD simulations and including turbulence balancing and modeling along with linear/nonlinear flow prediction [6]. RNN (Recurrent Neural Network) has shown exceptional ability to capture data-features related to the flow behavior and even auto-learn trends under the flow pattern of these fluids. It is grave to note, that these models, however, have shown a significant drop in performance while dealing with unlabeled Newtonian / non-Newtonian fluids. RNN, however, continues to be a widely used model, due to its almost monopoly ability to grasp temporal patterns in fluid flows [7, 8].

Following the recent developments, hybrid machine learning techniques have rightly catered to all needs- balancing procedures of fluid dynamics while improving upon the computational requirements from traditional CFD methods, and at the same time eliminating accuracy blind spots in sole machine learning architectures, reducing the burden on CFD solver [9].

Computer Vision has not been the ideal discussion spot when it comes to fluid behavior in recent times. However, the role that it plays is never less than the core research processes involved. Architectures under CNN have continuously facilitated the action validating the predicted fluid behavior against natural circumstances, which not only adds to the theoretical validation but also unlocks a different horizon of evaluation, aside of just comparing the tallying the predictions with the results of fluid mechanics equations. CNNs are adept at recognizing spatial features in pictures, making them a readily available real-time identifier of flow structures such as vortices and wave-zones in fluid dynamics [10]. While the results and processes are claimed to be time-efficient, the process scheduling of the tasks and data-load balancing features along with flow-analysis and domain decomposition are still an underexplored frame of reference, wherein the computational load of CFD simulations goes below the investigation eye. This course of research also accounts for the use of parallel computing and HPC (High-Performance Computing), essentially for managing the computational burden of the processes used, which shall enable the efficient execution of simulations on HPC systems [11]. Thus, it is safe to say that our course of research not only provides an efficient way of dealing with fluid simulations but at the same time analyzes and improves the process maintenance involved. While significant progress has been made in applying ML to fluid dynamics or Computational Physics as a whole, several issues still prevail, which inhibit the growth of these novel approaches into the industrial use case. Existing studies focus on specific aspects of fluid dynamics, such as turbulence, or simpler wave geometries, under exploring frameworks and usability of the same over a wider range of scenarios. Expandingly, the pipeline integration of Computer Vision with ML-driven aspects still is not studied keenly, which highly handicaps the real-time validation of such approaches. The use case of these machine learning models is also limited to the high-fidelity simulations which directly indicate the absence of an evaluation metric of the entropy of disturbance across the fluid (Table 1).

Table 1. Use-Cases of various techniques used in fluid dynamics.

Technique	Use Case	Key-Findings
Proper Orthogonal Decomposition (POD)	Model reduction in turbulent flows	Noise-reduction, dimensionality reduction, capturing crucial flow features
Convolutional Neural Networks (CNN)	Steady flow estimation	Accurately estimates steady flow, and time elapsed of non-wave pattern
Physics Informed ML	Gaps in Reynolds stress modeling	Accurate reconstruction and trace-mapping of DNS (Direct Numerical Data) data
Deep Learning Models	Unsteady flow estimation	Geometrical Estimation, Learning pattern, physical dynamics, sub-level entropy
Deep Neural Networks with Invariance (DNNia)	Wave-approximation/Turbulence modeling	Enhances Wave Models
Autoencoders	Nonlinear model reduction	High-Compression of simulation data
High-performance Computing (HPC)	Big-Data analysis of fluid-interaction	Large-scale simulations of parallel computing
Hybrid ML-CFD Models	Large-scale fluid structure interaction simulations	ML with traditional CFD to reduce computational load

The scalability of these approaches for large-scale, complex simulations also requires a further course of investigation and significant alterations to the interpretation of theoretical mechanics and real-time metrics of the fluid, which also demands the evaluation of combinatorics of results, flow pattern, and temporal analysis.

### 3. APPLICABLE EXISTING PATENTS - REVIEW

Recent years have seen a notable increase in the application of machine learning to computational fluid dynamics (CFD), which has improved real-time prediction capabilities, computational efficiency, and simulation accuracy. There are numerous existing technologies and patents, that are directly applicable to our research. These patents and technologies are broadly encompassed under four different patents, majorly being Computational fluid dynamics (CFD) coprocessor-enhanced system, and method [12], Systems and methods for the analysis of blood flow state [13], Method and system for assessing vessel obstruction based on machine learning [14], Systems and methods for utilizing a 3D CAD point-cloud to automatically create a fluid model [15]. The Coprocessor-enhanced system was one of the key revolutionizing incorporations, under CFD simulations, however, it is resource-exhausting and lacks periodic efficiency (time-complexity-based efficiency), compared to newer robust architecture. Our research devises neural network-based acceleration, which offers a better alternative to coprocessor-based simulations, by chipping in data-oriented surrogate models at the right time of simulation. The analysis of blood flow is a patent that represents every complex fluid observation, and Brownian motion. The proposed patent is an efficient way to analyze flow-related behavioral patterns, however, the increasing complexity of fluid nature proposes a challenge to the same. The paper proposes a mesh-constructed network, which enhances generalization to disguised flow conditions, random behavioral patterns, and most importantly, better integration of physics-informed constraints, under the usage of (PINN). Furthermore, the use of transformer-based neural networks has helped in adapting to turbulence and mini-flows (flows within a larger gross flow-vector). These transformer-based neural networks involve

the use of both mesh and non-mesh networks at certain points of comparison as described later in the paper, *e.g.* (Graph-Neural Network). Since meshing strategies have been an inevitable part of this course of research, the paper proposes a strong advantage over any present aerodynamic-analyzing CFD simulation in the industry, as it readily adapts to early-meshing and delayed-meshing, or even point-based CFD solvers, which necessarily require different network-architectures at any point of time. In conclusion, it is safe to say that the proposed system has the highest extent of adaptability, and sustainable simulation, can perform focused physics-integrated calculations, and generalize random behavior easily. The system is super-generic and can perform to a similar degree, with minute scale adjustments, with any kind of fluid or physical complexity.

### 4. DATA

The High-Fidelity CFD Simulation dataset consists of simulation data generated using high-fidelity computational fluid dynamics CFD tools like OpenFOAM and ANSYS Fluent. The dataset consists of a plethora of fluid dynamics scenarios over the diversity of use-cases and system-settings, such as turbulent and laminar flows, multiphase flow, *etc.* This also consists of unclassified fluids under different geometries. The data reads various features including stroke pressure, velocity, pressure-fields, matrix-direction, Wind Tunnel, Hires Tunnel, functional constraints, and temperature distributions across a range of Reynolds numbers and boundary situations. The OpenFOAM community archive of diverse use-cases also facilitated the entropical simulation proposed in the research.

The Experimental Fluid Dynamics (EFD) dataset comprises lab data obtained from fluid dynamics archives and laboratories and published research [16]. The set includes velocity profiles, pressure metrics, and flow-image negatives, which facilitate image segmentation and localization, under techniques such as Particle Image Velocimetry (PIV), and Schlieren imaging. This data is centric for matching the real-time framework applicability and use-case scenarios, rather than limiting validation to mathematical equations only. The high-fidelity dataset has also been used to facilitate dimen-



sionality reduction techniques alongside principal component analysis (PCA), and perform deep-seated Proper Orthogonal Decomposition (POD), and autoencoders. This part of the research focuses on high-dimensional data, which mainly structures upon reduction techniques.

The training data includes subsets of high-fidelity CFD simulation data and lab data specifically classified for training and testing machine learning models; it has been clustered from the rigorous simulation during research and used along the main component dataset from the Lab Archive. The computational match-dataset has been derived from UCI Machine Learning Repository. The synthetic data generation techniques are used to ensure the wholesome inclusion of diverse fluid dynamics scenarios

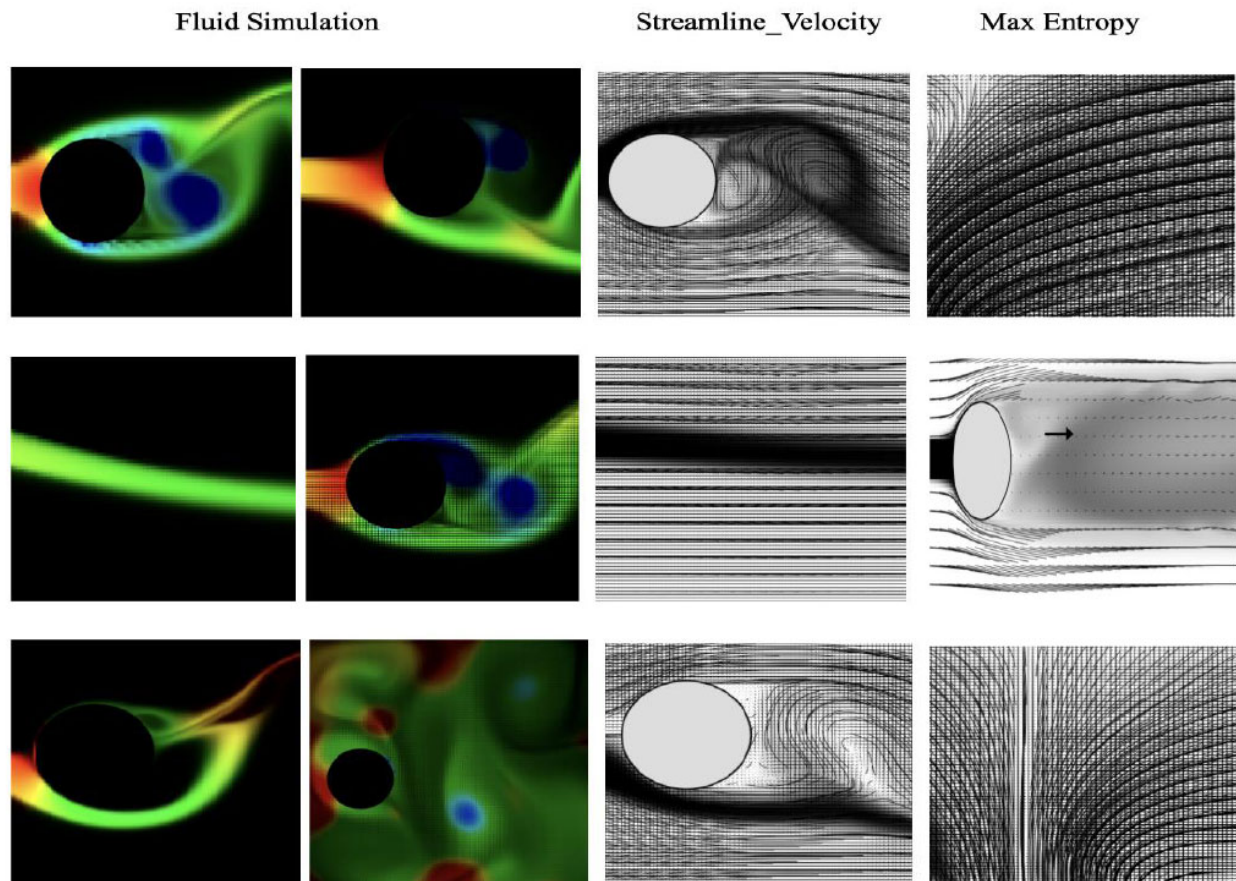
#### 4.1. Data Preprocessing

The preprocessing of data encompasses numerous steps to prepare it for use in machine learning models and simulations. Dimensionality Reduction Techniques such as principal component analysis (PCA), Proper Orthogonal Decomposition (POD), and autoencoders. The objective is to trace

down key features like pressure fields, and temperature distributions, along with velocity components, which can be illustrated in a reduced-dimensional space. For the PCA, the data is normalized to ensure each feature is accounted for in the analysis. The process mainly encompasses scaling the data, resulting in a mean of 0, and a standard deviation of 1.

The above figure (Fig. 2) signifies the randomness in each component of the fluid dynamics and also depicts pressure fields under the colored region. The rightmost column signifies the maximum entropy plots, which in turn signifies regions of high random component flow complexity.

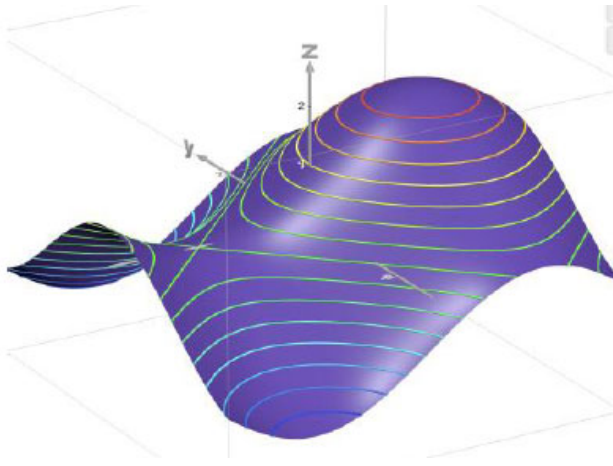
The illustrations below show the simulations and different velocity vectors along the fluid flow. It is grave to note here that Max Entropy signifies the complete random vector flow to the maximum degree in the fluid. Ultimately signifying the bound of opposed vector flow in the fluid. The irrelevant columns of data compiled such as time of record and other factors that are not of scientific bounds were dropped, these attributes accounted for 3 columns. showing a matrix of 3 x 642 grid.



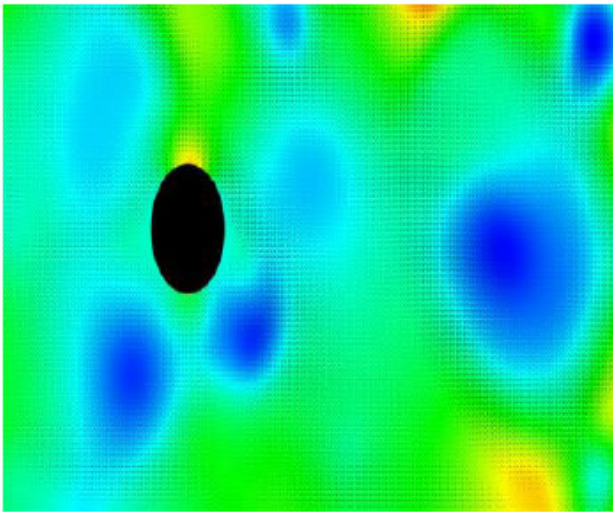
**Fig. (2).** CFD Simulation, developed in ParaView. The entropy corresponds to small random flow patterns, within a big flow direction. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

The CFD was done in Paraview, and the Streamline and Entropical images were generated using Fortran, MATLAB, and CFD Python visualizing kit. The distribution contour was designed in seaborn and includes vector algebra.

The temperature distribution graph (Fig. 3) comprises various crests and troughs throughout the complete contour of the fluid. The crests signify fluid regions of high temperature, it is important to note here that the graph is built across median observations of Newtonian and non-Newtonian fluid. The viscosity is expected to decrease with an increase in higher temperature regions across the fluid (Fig. 4). The graph has been constructed using Seaborn, a popular data visualization framework in Python. To the right is the actual visualization of high-temperature and low-viscosity regions marked in blue, visualized under an in-house HTML5 visualizer.



**Fig. (3).** Temperature distribution in the fluid. (A higher resolution / colour version of this figure is available in the electronic copy of the article).



**Fig. (4).** High-Temperature Zones [Blue Zones]. (A higher resolution / colour version of this figure is available in the electronic copy of the article).

The training and validation dataset was obtained from UCI ML repository and Kaggle Inc. The training data will be used to develop regression models [17], RNN feeds, and hybrid architectures, to pre-empt the fluid dynamics behavior. The split will help ensure the validity of the models in real-time scenarios (Fig. 5).

Synthetic Image datasets, along with localized videos have been segmented into a separate database for the Convolutional Neural Network, to capitalize upon [18]. CNNs analyze and augment flow patterns, increasing predictive accuracy and developing a robust hold of data features during validation in real-time [19].

Every data factor, along all the sources excluding the image-dataset, has been utilized for large-scale HPC systems, including performance metrics of the machine learning models used, computational usage, and simulation data from scenarios, proving its feasibility for handling large-scale simulation.

## 5. METHODOLOGY

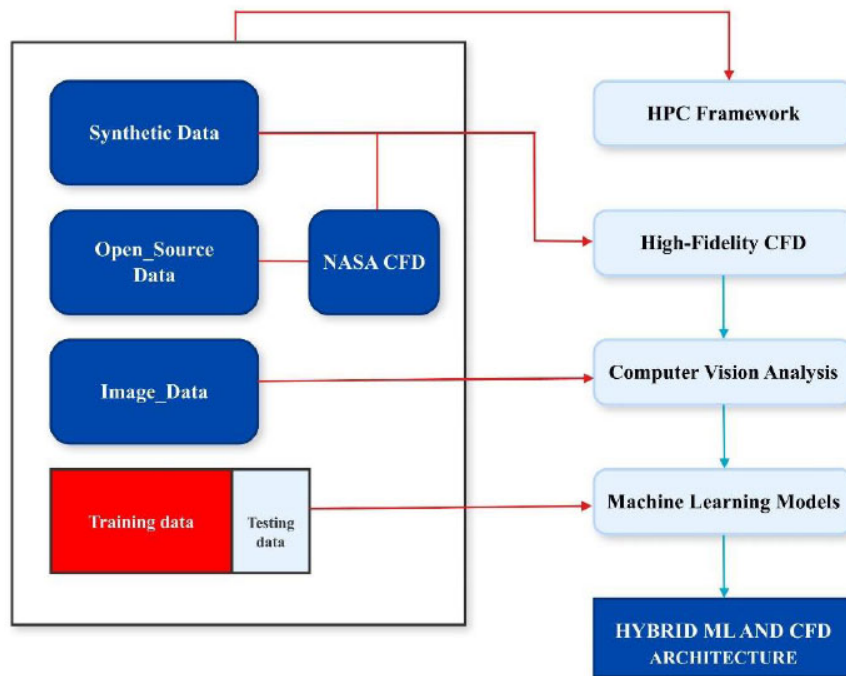
### 5.1. Principal Component Analysis (PCA)

Under fluid dynamics simulations, the incredible diversity and volume of data generated can be certainly arduous to manage and analyze completely. To cater to this issue, we use dimensionality reduction techniques, mainly Principal Component Analysis (PCA) [20] and Proper Orthogonal Decomposition (POD). These methods certainly reduce the complexity of high-dimensional data, and not only retain the essential features that capture the fluid dynamics behavior but also make the process of tracing down these features simplified.

Principal Component Analysis (PCA) is a scientific-statistical method used to convert correlated variables to uncorrelated variables called principal components, being arranged according to the variance in the data [21], the first elements holding the highest degree of variance. The data used under the process has already been scaled to (mean = 0, and s.d= 1). Furthermore, the covariance matrix of the data helps in understanding variance and correlation among various features [22]. For a dataset, with  $n$  features, the covariance matrix will be  $n \times n$  matrix. The diagonal components of the matrix portray the variance of the components [23], which in turn means the feature's deviation from its mean, and the off-diagonal components show covariance. Covariance is crucial to building the problem system map [24], it shows a positive relationship (components increase/decrease together) or a negative relationship (components are inversely proportional).

$$C = \begin{bmatrix} \sigma_{x1^2} & \sigma_{x1x2} \\ \sigma_{x1x2} & \sigma_{x2^2} \end{bmatrix}$$

The equation above signifies that the Covariance Matrix,  $\sigma_{x1^2}$  and  $\sigma_{x2^2}$  are variances over  $x_1$  and  $x_2$  respectively.



**Fig. (5).** Dataset segmentation. (A higher resolution / colour version of this figure is available in the electronic copy of the article).

The eigenvalue decomposition on  $C$ , gives us the eigenvalues and their corresponding eigenvectors, If  $E_1$  and  $E_2$  are eigenvalues, and  $E_1 > E_2$ , then  $V_1$  (eigenvector 1) [25] is the first principal component. The results of the same showed, that our first principal component typically captured the highest variance in data, getting 55% of the total variance. The second principal component added around 25%. By the onset of the 10th component, 95% of the total variance has been captured by the first 9 components only [26].

```
pca = PCA()
```

```
principal_components = pca.fit_transform(data_scaled)
```

To visualize the PCA-explained variance we use the following functions under Matplotlib-Python,

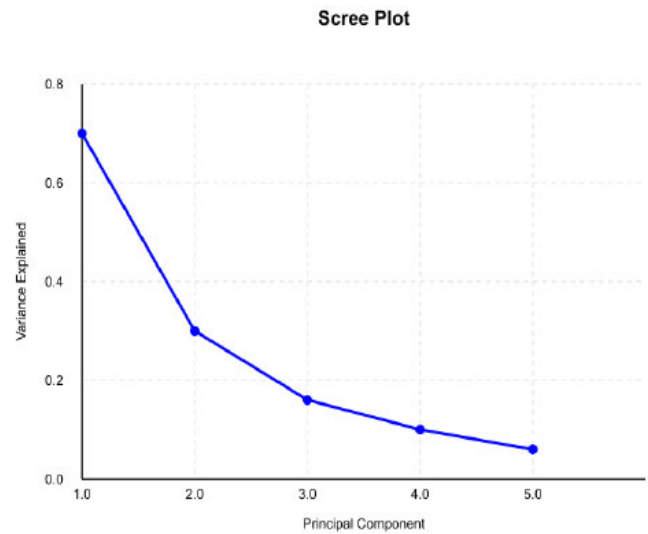
```
plt.bar(range(1, len(explained_variance) + 1), explained_variance, alpha=0.5, align='center', label='Individual Explained Variance')
```

```
plt.step(range(1, len(cumulative_explained_variance)+1), cumulative_explained_variance, where='mid', label='Cumulative Explained Variance')
```

The First Principal Component is crowded by features related to the velocity field, showing the most significant variance in the data comes across flow speed and vector-direction. The Second Principal Component is a keen set of pressure and temperature variations, indicating that these features are a key feature of secondary patterns in data.

The remaining PCs capture intricate shifts between pressure and other features along with some tensor values of

strain, if present in the fluid. The reconstruction using the first 10-15 PCs, with approximately 95% accuracy, compared to the original non-normalized data, indicating minimal loss of important reserves. PCA has reduced the data noise and helped in a cleaner interpretation of data, the same shall help in data compression and feature selection during the ML model-building process, which in turn supports our hybridized research model (Fig. 6).



**Fig. (6).** Variance explained by each PC. (A higher resolution / colour version of this figure is available in the electronic copy of the article).



## 5.2. Incorporating Deep learning-based Autoencoders for Data-reduction

Supporting our hybridized simulation model, we can use deep learning-based autoencoders to capture even intricate complex relationships in the data and progress to even further dimensionality reduction. Autoencoders learn the intricate coding within the input data in an unsupervised manner [27], the process does not no further standardization the dataset, which enables us to reduce the time complexity even on such a large-process scale, the autoencoders function over the normalized data under PCA setup. The Autoencoder we are using is operated on Relu activation on the input layer and sigmoid on the encoded stage, which is generic when it comes to generic neural networks, or even with GANs.

```
input_layer = Input(shape=(input_dim,))
```

```
encoded = Dense(encoding_dim, activation='relu')(input_layer)
```

```
decoded = Dense(input_dim, activation='sigmoid')(encoded)
```

the input\_dim is scaled to shape [1], and the encoding dimension is set to 10, this architecture proved to be the most suitable after repetitive submissions. The autoencoder is trained initially over a modeled batch size of 256 [28] and shuffled with epoch rate at 50.

```
history = autoencoder.fit(data_scaled,
```

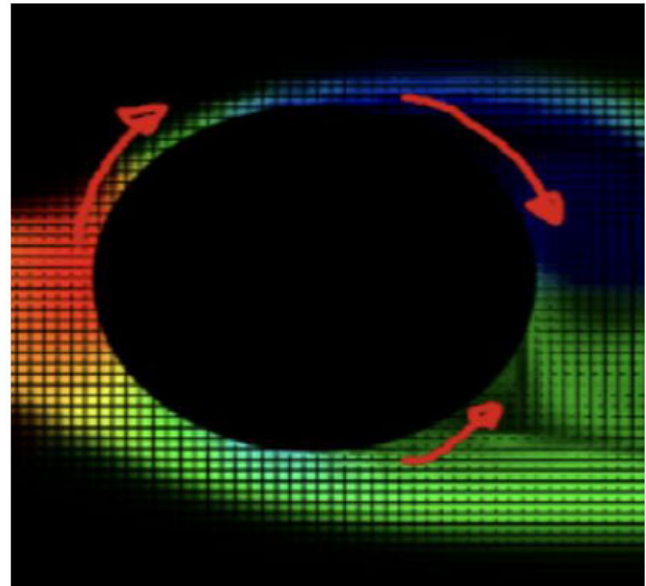
```
data_scaled, epochs=50,
```

```
batch_size=256,
```

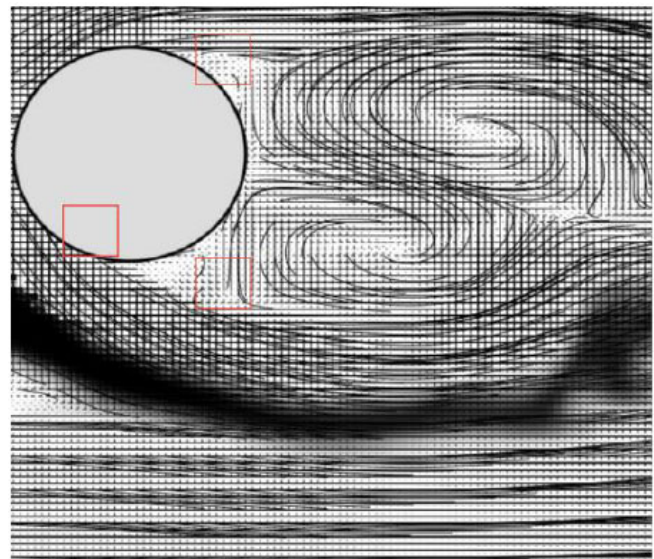
```
shuffle=True, validation_split=0.2, verbose=1)
```

The training loss was minimal, and the results of feature-holding stood out to be excellent, the advantage that this process gives here is, that there is no separate normalization of data needed for intricate investigation [29], which required conjuring another CFD architecture earlier, and it extracts interesting and crucial patterns which often went unnoticed by CFD Architectures. However, it might be resource-consuming if not used after PCA, but that shall not affect the course of research because PCA is an abundant process in every experimental setup.

The interesting patterns we have received are anti-parallel vectors between two bigger vectors in a fluid, not broad temperature zones as marked earlier but temperature gradient, which helps in seeing the overall distribution of energy-entropy-heat in the fluid. the temporal dynamics, such as vortex shedding, turbulence, and flow separation, which were inherently non-linear and not captured by PCA, and POD. The most differentiating result is the detection of airfoil geometries (complex to detect, often unnoticed by PCA) around the fluid, which shows the intricacy in deep learning architecture over the problem, this feature can support modern research work of air-fluid relation going beyond concepts like drag (Figs. 7 and 8).



**Fig. (7).** small airfoil zones, manually. (A higher resolution / colour version of this figure is available in the electronic copy of the article).



**Fig. (8).** airfoil zones, deep learning. (A higher resolution / colour version of this figure is available in the electronic copy of the article).

By integrating autoencoders, we capture complex non-linear relationships that techniques like PCA, with minimal effort of procession and a lesser processing time, presenting ground-breaking observations like flow-type, minute vectors between vectors, and other crucial features like bifurcations and chaos.



### 5.3. Using Machine Learning Models, and Developing Neural Networks for Predictive Fluid Dynamics

Catering to the complex nature of fluid dynamics, we have used physics-informed machine-learning models. These types of models use hybrid physics-principles structured architectures with extreme learning capabilities over deep learning techniques. This technique ensures adherence of advanced machine learning techniques to fluid-mechanics laws, as discussed by [30].

Physics-informed Neural Networks (PINN) are an excellent example of such models that go in adherence to the partial differential equations, Navier Stokes Equation, Bernoulli's hypothesis, and pressure spread over a particular container. The animations for the text data and records have been designed in a blender. The adherence also includes continuous relation with 2-D Laplace Equation, where in 2 is the Laplacian operator, and the cartesian coordinates of all features can be defined as, a Laplacian function given by partial derivative of (x,y,z) [30].

Defining the physical laws of the model under PINN, is subject to velocity field(vector). The Navier-stokes equations describe the motion of fluid substances, consisting of a set of nonlinear partial differential equations.  $\frac{du}{dt} + (u \cdot \nabla)u = -\nabla p + \nu \nabla^2 u + f$ .  $f$  represents external force example gravity.

The data-driven loss curve, used under neural networks, such as mean squared error (MSE), compares actual and predicted values examples. velocity and pressure. Physics-driven loss is an important differentiator here, It is one of the crucial advantages here, the feature-curve, including the evaluation of the residuals of the equation, using automatic differentiation. Both of these functions contribute towards a Combined Loss function.

The combined Loss Function for PINN, is defined as given below:

```
input_layer = Input(shape=(X.shape [1],))
```

```
hidden_layer1 = Dense(64, activation='relu')(input_layer)
```

```
hidden_layer2 = Dense(64, activation='relu')(hidden_layer1)
```

```
output_layer = Dense(y.shape [1])(hidden_layer2)
```

The Naveir-Stokes equation definition is given below

$$\text{continuity} = u_x + v_y + w_z$$

$$\text{momentum}_x = u_t + u * u_x + v * u_y + w * u_z + p_x - (1/Re) * (u_{xx} + u_{yy} + u_{zz})$$

$$\text{momentum}_y = v_t + u * v_x + v * v_y + w * v_z + p_y - (1/Re) * (v_{xx} + v_{yy} + v_{zz})$$

$$\text{momentum}_z = w_t + u * w_x + v * w_y + w * w_z + p_z - (1/Re) * (w_{xx} + w_{yy} + w_{zz})$$

The loss as the mean squared error is given as:

$$\text{loss\_continuity} = \text{tf.reduce\_mean}(\text{tf.square}(\text{continuity}))$$

$$\text{loss\_momentum}_x = \text{tf.reduce\_mean}(\text{tf.square}(\text{momentum}_x))$$

$$\text{loss\_momentum}_y = \text{tf.reduce\_mean}(\text{tf.square}(\text{momentum}_y))$$

$$\text{loss\_momentum}_z = \text{tf.reduce\_mean}(\text{tf.square}(\text{momentum}_z))$$

The PINN architecture (Fig. 9) helps in calculating the residual volumes of Laplacian and Navier Stokes Equation; thus, the inclusion of these features happens in an efficient period of time which gives an advantage over stratified CFD calculations for the same.

### 5.4. Combined Model-Architecture for Machine Learning in Fluid Dynamics

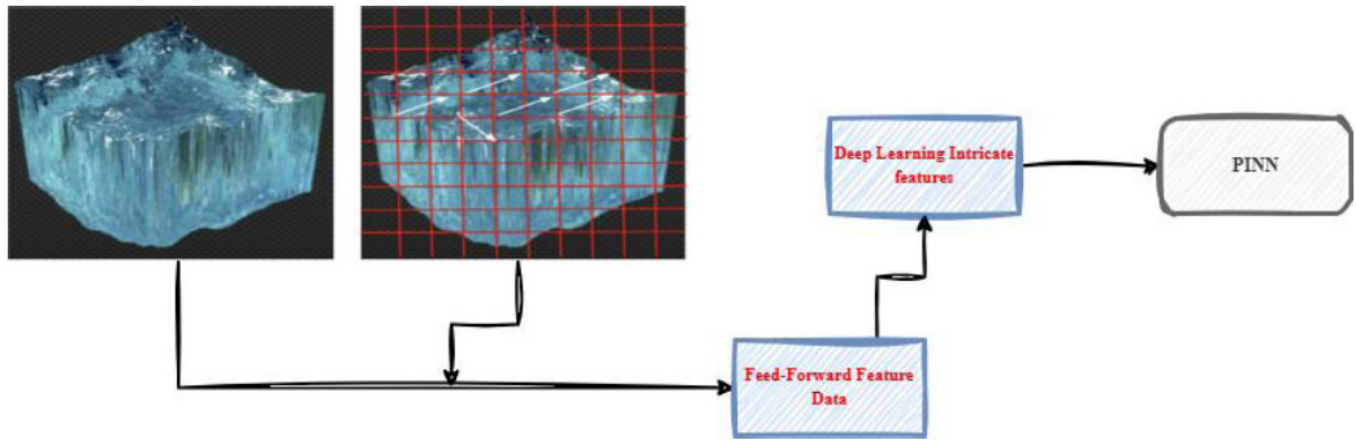
The initial phase involves the preprocessing of unrefined simulation data, encompassing both standardization and normalization techniques. To diminish the high-dimensional data while preserving vital information, Principal Component Analysis (PCA) and Proper Orthogonal Decomposition (POD) are utilized. This procedure streamlines the data while ensuring that significant patterns remain intact. The crux of the architecture is the Physics Informed Neural Network (PINN), which encompasses and supports collaborative physics and neural simulation.

Furthermore, Hybrid deep learning models (Fig. 10) have shown keen development across the use in computational physics and fluid simulation. SSL (Semi-Supervised Learning), MIL (Multi-Instance Learning have), RNNs, and ML4Sci have continuously proved to be deeply intriguing research setups. The combination of PCA, PINN, and regression techniques only, has proved to be a key differentiator, with advancements in residual evaluation, over-minute feature learning, and excellent time efficiency, The use of these three models only revolutionizes the CFD simulation. Furthermore, if these models are used in a hybrid architecture, then it would be safe to say that Machine Learning Techniques will take over the traditional CFD simulation in no time.

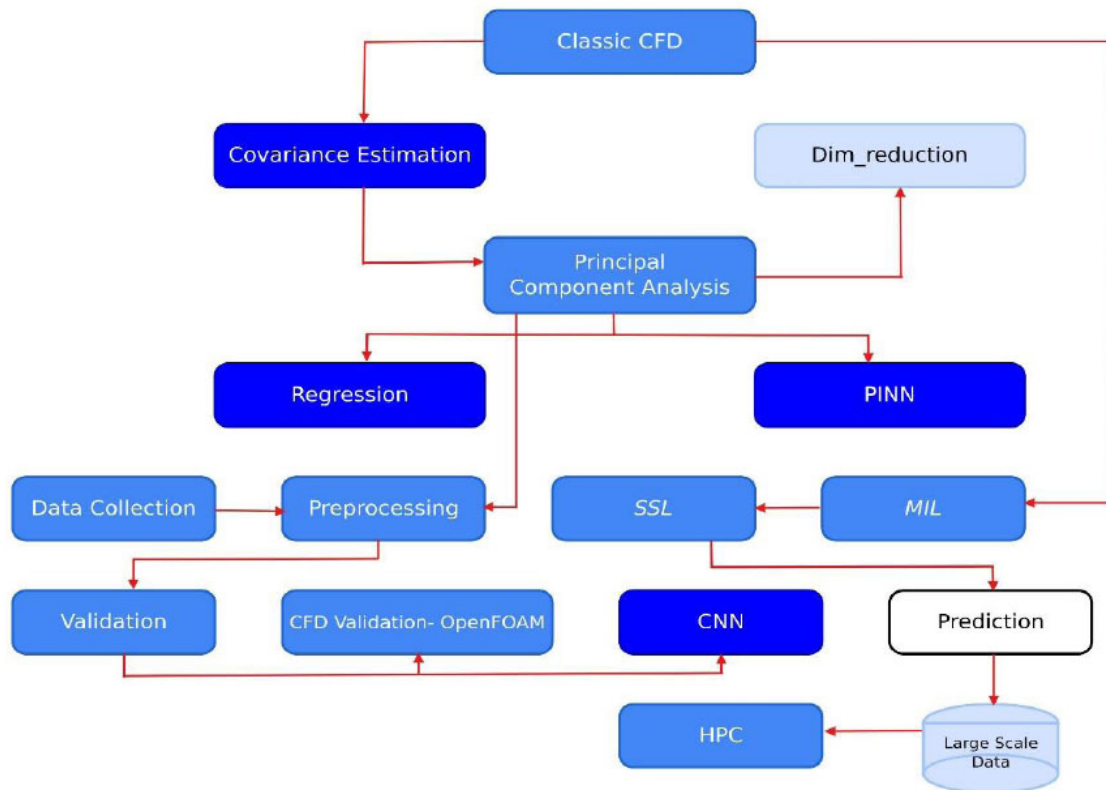
The Architecture also includes the use of Computer Vision to validate theoretical and predictive results with real-time scenarios, and the management of big data with High - Performance Computing.

### 5.5. Computational Neural Networks for Validation and Real-time Training

Convolutional Neural Networks are significantly important in classifying patterns and features in spatial data, during the real-time experimental setup. Besides the above, they can also be used to validate the predictions made by the hybrid model learning on physical and feed-forward data. Thus, using CNN shall enhance the modality of our predictive and simulative architecture, and arm it to compare vector results with real-time flow analysis. such as structures, vortices, and turbulence.



**Fig. (9).** Data feature input for PINN. (A higher resolution / colour version of this figure is available in the electronic copy of the article).



**Fig. (10).** Hybrid ML-architecture. (A higher resolution / colour version of this figure is available in the electronic copy of the article).

The image dataset we have segmented from synthetic data generated from localized simulations, and data imported from PRACE, and OpenFOAM archives, shall serve as the training dataset for our model, furthermore, ~14% of the dataset has been reserved for real-time validation purposes. The definition of the CNN architecture has been provided below:

```
model = Sequential([ Conv2D(32, (3, 3), activation='relu', input_shape=(image_height, image_width, num_channels)), MaxPooling2D((2, 2)), Conv2D(64, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Conv2D(128, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Flatten(), Dense(128, activation='relu'), Dense(num_classes, activation='softmax')])
```

The Adam optimizer has been used in the defined CNN architecture, where localization and segmentation have been improved for complex geometries, as it is obvious that we are dealing with flowing regions, where segmentation and localization is considered tough, hence, a tweaked version CNN, has been initiated, wherein the gradient descent (SGD), namely AdaGrad and RMSProp, Adam is widely used because it's computationally feasible, and a truly responsive process-function, when it comes to layered dynamic geometries.

Certain hyperparameters like, rate schedulers have been used, and the generic Adam optimizer has been configured mimicking Nadam and Ranger optimizers, however, they could not be used, because they can be resource consuming, and tend to lose learning rate with higher entropy of vector-flow.

The real-time data can be captured by embedding the model with sensors, over microcontrollers like RaspBerryPi, wherein the adherence to float32 computation is advised to achieve the best results.

Below (Fig. 11) is one such use case, wherein the CNN model was used on an image of water in a glass box depicting Attenuated transmission.

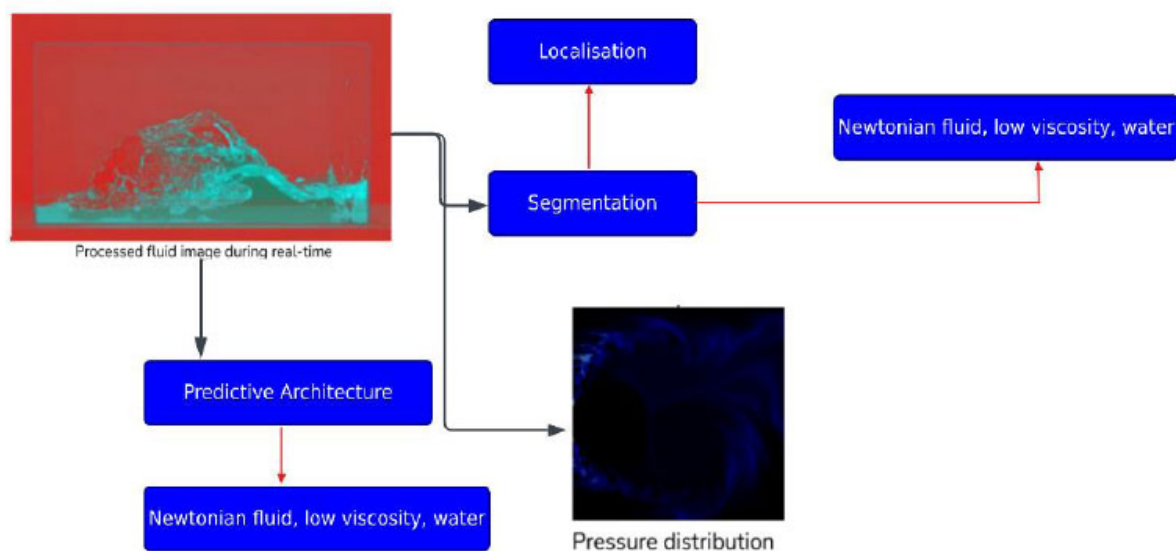
### 5.6. Large-Data Management with High-Performance Computing (HPC)

At this scale and with the amount of data we handle in our fluid dynamics simulations and machine learning, good management of such data proves critical. Handling, processing, and analyzing large amounts of data requires High Performance Computing also known as HPC. Distributed file systems like Luster and Hadoop HDFS ensure data replica-

tion and high data availability while access to data is accelerated by parallel I/O libraries like MPI-IO and HDF5 for parallel read/write operations. In data processing MPI and OpenMP distribute the big data into several smaller partitions so that the same number of processors can process all at the same time. Other Scheduling systems such as SLURM also perform batch processing through the managing and queuing of Jobs. In the data analysis aspect, distributed computing frameworks such as Apache Spark and HPC-optimized machine learning libraries including TensorFlow and PyTorch divide the workload to multiple GPUs or CPUs which enhance the training time of models and the overall model accuracy. There are ParaView and other *in-situ* visualization tools that help to visualize the data without transferring it to another machine and allow the analysis during simulation. Thus, by applying HPC in large-data management within the research, we can handle large-scale computational tasks effectively, whereby the computational time is accelerated and precise solutions for the flow field are obtained.

## 6. EXPERIMENTAL RESULTS AND DISCUSSION: EVALUATION

Thus, to perform the Machine Learning part of the program, we will use Python language, and for the CFD simulation part, MATLAB or FORTRAN language (Table 2). Moreover, at this step, engineers will use computational fluid dynamics tools like OpenFOAM, ANSYS Fluent, where applications based on numerical solving techniques like FEM, FVM, and LBM will be performed. We will employ Python for training the machine learning models which can be implemented with the help of TensorFlow or PyTorch. This framework will be involved in the preparation of the data, model training, and making predictions.



**Fig. (11).** CNN use-case in validation and real-time simulation. (A higher resolution / colour version of this figure is available in the electronic copy of the article).

**Table 2. MATLAB setup and FORTRAN code for CFD simulation.**

MATLAB Setup:	FORTRAN Code for CFD Simulation:
<pre> L = 1.0; N = 100; dx = L / (N-1); u = zeros(N, 1); u_new = u; u(1) = 1; u(end) = 0; dt = 0.01; nt = 100; for t = 1:nt     for i = 2:N-1         u_new(i) = u(i) - dt * (u(i) -         u(i-1)) / dx;     end     u = u_new; end plot(linspace(0, L, N), u); xlabel('Position'); ylabel('Velocity'); title('CFD Simulation');</pre>	<pre> program cfd_simulation implicit none integer, parameter:: N = 100 real, dimension(N):: u, u_new real:: dx, dt integer:: i, t, nt dx = 1.0 / (N-1) u = 0.0 u_new = 0.0 u(1) = 1.0 u(N) = 0.0 dt = 0.01 nt = 100 do t = 1, nt     do i = 2, N-1         u_new(i) = u(i) - dt * (u(i) - u(i-1)) /         dx     end do     u = u_new end do open(unit=1, file='results.txt') do i = 1, N     write(1,*) i, u(i) end do close(1) end program cfd_simulation</pre>

Assessing the performance of the proposed predictive architecture is based on the comparison of the obtained performance with real-time results using CNNs and standard CFD simulations concerning the prediction accuracy, computational cost, and the ability of the proposed model to capture real fluid dynamics behavior (Table 3).

As for the predictive capability, our machine learning model, which integrates PCA, POD, and autoencoders based on deep learning, is intended to diminish the original high dimensions of the simulation data while maintaining the salient properties of the flow. For example, in the performance of the model when predicting the actual simulation results as well as experimental data, metrics such as Mean Squared Error (MSE), R-squared ( $R^2$ ), and Relative Error are used. CNNs are designed in a way that they learn about the nature of flows and produce their predictions about the outcome in real-time, based on the received data, and real-time experimental scenario, which is also assessed with the

help of MSE,  $R^2$ , and Relative Error. CNN Architectures are capable of identifying spatial patterns of data as well as temporal changes and therefore are quite appropriate in the prediction of complicated flow structures. These are considered the 'baseline' CFD simulations with tools like OpenFOAM, ANSYS Fluent that use FEM/FVM/LBM to solve the Navier-Stokes equations governing the fluid flow. The validity of these methods is proved, which indicates that they can be used as a reference standard for performance comparison for our proposed predictive architecture and CNN-based predictions.

It must be pointed out that from the computational point of view, the machine learning model proposed here is much more efficient. This way, the dimensionality of data is decreased, and only the most relevant flow features are extracted by autoencoders so that the computational cost is much lower than in full-scale CFD simulations. While training of the model may be time-consuming, that is, computationally expensive, once trained, prediction is quite efficient, the deep learning model also records minute data features which may affect the course of further study of the fluid, which traditional CFF fails to do. On the other hand, conventional CFD simulations are resource and time-consuming as they may demand ample CPU time and its associated resources when solving geometry-intense problems and/or high-fidelity simulations. When such scenarios are executed on HPC systems, it can take several hours to a couple of days, depending on the problem size and characteristics.

The estimations of time and comparison of the performance clearly show distinctions. Training the machine learning predictive architecture can take a few hours to a few days based on the size of the data and the complexity of the model. Similarly, real-time prediction using CNN takes the same range of time as the training phase, but the inference phase practically has a response time of real-time. Standard employment of CFD simulations, however, is much slower and takes time that ranges from several hours to even days for each simulation run hence being unfit for real-time analysis.

The machine learning architecture also studies temporal dynamics to the keen extent of details, and PINN also accounts for the residual omissions under partial differentiations which is absent in conventional CFD simulations. The gradient analysis has also helped in understanding different vectors in the fluid.

**Table 3. Experimental results acquired by CFD and ML, CNN.**

FLUID	ML-Prediction	ACCURACY (%)	CNN	Match
Fluid_1	Newtonian, low viscosity, room temperature	96	Water	+
Fluid_2	Newtonian, high viscosity, room temperature	92.3	Honey	+
Fluid_3	Newtonian, high viscosity, high temperature	95	Molten Glass	+
Fluid_4	Restricted flow, no visible flow vectors	94.3	Crude Oil	+
Fluid_5	Non-Vector high Entropy	87.9	Foam	+
Fluid_6	Laminar flow, high airfoil shift	97.23	Water in Capillary tube	+

Note: Match column signifies + if the CNN detection matches original fluid.



## CONCLUSION

In this work, the authors focused on analyzing the augmenting the traditional computational fluid dynamics simulations with machine learning to improve the performances of predictive fluid dynamics simulations. It means that we used mixed methodologies of PCA, POD, deep autoencoders, and CNNs to enhance the predictive models. We successfully used the machine learning algorithm to lower the dimensionality of originally abstract flow simulation data while maintaining rather essential traits. This way, using deep learning techniques, we were also able to distinguish very fragile details in the fluid dynamics data which could not be noticed with the help of traditional techniques. The residuals in the partial differential equations were considered when using Physics-Informed Neural Networks (PINNs), so the predictions were consistent with the physics behind the fluid flow.

CNNs that are usually fast and precise in identifying spatial patterns were of great help in delivering real-time predictions. These networks performed best in terms of recognizing complicated flow structures and changes and had great efficiency in the analysis of instantaneous conditions and in making decisions. When autoencoders were integrated with CNNs, it further improved our forecasting function, which gave a major boost to what was available from conventional CFD simulations. Earlier CFD simulations which were performed using different packages such as OpenFOAM and ANSYS Fluent gave a yardstick by which the accuracy could be measured. However these methods are computationally expensive and, in some cases, take more time and computational power. However, our presented machine learning architecture significantly decreased computational cost and time for the analysis of the fluid dynamics behavior. Training the model was, however, almost a herculean task compared to the ease in the prediction phase which only took a short time.

Hence the paper recommended the use of the proposed integrated approach since it produced a much better result as compared to the traditional CFD methods. The opportunity to extract high-level features, CNNs' high accuracy, and the physical contact ensured by PINNs solidified our method's supremacy. Thus, the integration of machine learning and CFD was a positive synergy that optimized the comprehensive predictive tool while providing higher efficiency in simulating fluid dynamics. Besides improving the predictive capability, it also leads to a drastic cut down in computation time making the scope of fluid dynamics research and application broader and very efficient.

Our research encompasses significant use of modern neural network architecture, which presents a robust methodology, with the integration of CNN and real-time analysis, industrial applications require continuous real-time monitoring, such as Gas pipelines, HVAC optimization. This task is well-suited for the diversified nature and robustness of our developed model. Furthermore, our proposed methodology of integrating CFD with machine learning has the highest degree of applicability in aerospace mechanisms, fluid management, and logistical transportation of both volatile and non-

volatile fluids. The ability to generate high-efficiency and quality-based results at a fraction of the computational cost, as compared to secondary-CFD processing parity, is remarkable for sustainable urban projects, which require multiple logistical sequences. Considering the engineering aspect, our model can be used to simulate high-fidelity airflow patterns for any aerodynamic or non-aerodynamic creation under automobile sector, or even carrier manufacturing.

## CURRENT AND FUTURE DEVELOPMENT

The amalgamation of machine-learning techniques and computational fluid dynamics (CFD) simulations has opened up entirely new avenues in predictive modeling for fluid dynamics. The use of technology such as PCA, POD, deep autoencoders, or CNNs to triangulate concepts and broaden their ability to represent complex fluid dynamics in this work. Physics Informed Neural Networks (PINNs) threw even more reliability for the model results since their predictions came from the solutions having a physical basis and obeying the governing equations of flow. These developments not only increased the prediction efficiency but also brought considerable savings in computation time and resources making it a feasible and scalable technique.

Very shortly, refinement and extension of these methodologies to cater to controversies in future super-turbulent multiphase flow systems can be considered a rather crucial direction forward. It should also be highlighted that the progressive coupling of more advanced conceptual models with PINNs could lead to further accuracy enhancement without incurring any further computational costs. Furthermore, the implementation of transfer learning methods could allow the transfer of models previously trained on a specific dataset to be successfully utilized within an entirely different fluid dynamics domain allowing an even wider range of applications. Finally, the coupling of cloud and edge computing resources with this framework could be a game changer for executing time critical CFD simulations.

## AUTHOR'S CONTRIBUTION

The authors confirm their contribution to the paper as follows: study conception and design: MZ, SK; data collection: WJB; investigation: BMB; analysis and interpretation of results: NK, SAF; draft manuscript: BB. All authors reviewed the results and approved the final version of the manuscript.

## LIST OF ABBREVIATIONS

ML	= Machine Learning
CNN	= Convolutional Neural Networks
XSEDE	= Extreme Science and Engineering Discovery Environment
PRACE	= Partnership for Advanced Computing in Europe
PCA	= Principal Component Analysis
POD	= Proper Orthogonal Decomposition

RNN = Recurrent Neural Network

MSE = Mean Squared Error

## CONSENT FOR PUBLICATION

Not applicable.

## AVAILABILITY OF DATA AND MATERIALS

The data and supportive information are available within the article.

## FUNDING

None.

## CONFLICT OF INTEREST

Dr. Sheikh Amir Fayaz is the Associate Editorial Board Member of the journal ENG.

## ACKNOWLEDGEMENTS

Declared none.

## REFERENCES

- [1] H.K. Versteeg, and W. Malalasekera, *An introduction to computational fluid dynamics: The finite volume method.*, Pearson Education, 2007.
- [2] L. Sirovich, "Turbulence and the dynamics of coherent structures. I. Coherent structures", *Q. Appl. Math.*, vol. 45, no. 3, pp. 561-571, 1987.  
<http://dx.doi.org/10.1090/qam/910462>
- [3] G. Berkooz, P. Holmes, and J.L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows", *Annu. Rev. Fluid Mech.*, vol. 25, no. 1, pp. 539-575, 1993.  
<http://dx.doi.org/10.1146/annurev.fl.25.010193.002543>
- [4] J.X. Wang, J.L. Wu, and H. Xiao, "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data", *Phys. Rev. Fluids*, vol. 2, no. 3, p. 034603, 2017.  
<http://dx.doi.org/10.1103/PhysRevFluids.2.034603>
- [5] S. Lee, and D. You, "Data-driven prediction of unsteady flow over a circular cylinder using deep learning", *J. Fluid Mech.*, vol. 879, pp. 217-254, 2019.  
<http://dx.doi.org/10.1017/jfm.2019.700>
- [6] K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data", *Annu. Rev. Fluid Mech.*, vol. 51, no. 1, pp. 357-377, 2019.  
<http://dx.doi.org/10.1146/annurev-fluid-010518-040547>
- [7] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance", *J. Fluid Mech.*, vol. 807, pp. 155-166, 2016.  
<http://dx.doi.org/10.1017/jfm.2016.615>
- [8] S. Srinivasan, N. Vedula, R. Garg, and R. Menon, "Multiresolution convolutional auto-encoder for compressive sensing of turbulent flows", *Phys. Rev. Fluids*, vol. 4, no. 5, p. 054604, 2019.
- [9] Y. Xie, H. Wang, and H. Zhang, "A deep learning approach for interatomic potential function development: Applications to Si, Ge, W, and SiC", *Phys. Rev. B*, vol. 102, no. 1, p. 014107, 2020.
- [10] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* San Francisco, California, USA, 13 August 2016, pp 481-490  
<http://dx.doi.org/10.1145/2939672.2939738>
- [11] X. Chen, Z. Xie, C. Xu, and Y. Zhang, "Large-scale numerical simulation of fluid-structure interaction problems using high-performance computing", *Comput. Fluids*, vol. 199, p. 104432, 2020.
- [12] A. Dugleby, K. Ball, and E. Sewall, "Computational fluid dynamics (CFD) coprocessor-enhanced system and method", US patent 20070219766A1, 2007.
- [13] Y. Ren, M.A. Jieyan, and H. Wang, "Systems and methods for analysis of blood flow state", US patent 20190340764A1, 2020.
- [14] I. Isgum, M. Zreik, and T. Leiner, "Method and system for assessing vessel obstruction based on machine learning", US Patent 10395366, 2019.
- [15] Z.S. Dweik, "Systems and methods for utilizing a 3D CAD point-cloud to automatically create a fluid model", US Patent 11714933, 2023.
- [16] S.A. Fayaz, M. Zaman, S. Kaul, and M.A. Butt, "How M5 Model Trees (M5-MT) on continuous data are used in rainfall prediction: An experimental evaluation", *Revue d'Intelligence Artificielle*, vol. 36, no. 3, pp. 409-415, 2022.  
<http://dx.doi.org/10.18280/ria.360308>
- [17] N. Kaul, M. Zaman, and W.J. Bakshi, "Analytical study of breast cancer and treatment techniques", *Procedia Comput. Sci.*, vol. 235, pp. 578-587, 2024.
- [18] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks", *Commun. ACM*, vol. 60, no. 6, pp. 84-90, 2017.  
<http://dx.doi.org/10.1145/3065386>
- [19] Y. LeCun, and Y. Bengio, "Convolutional networks for images, speech, and time series", In: *The Handbook of Brain Theory and Neural Networks.*, MIT Press, 2015, pp. 255-258.
- [20] I.T. Jolliffe, *Principal component analysis.*, Springer, 1992.
- [21] L. Sirovich, and K. Sloan, "A low-dimensional procedure for diagnosing changes in fluid flow", *J. Fluid Mech.*, vol. 219, pp. 345-355, 1990.  
<http://dx.doi.org/10.1017/S0022112090001673>
- [22] J.L. Lumley, "The structure of inhomogeneous turbulent flows", In: *Atmospheric Turbulence and Radio Wave Propagation.*, Defense Technical Information Center, 1997, pp. 167-178.
- [23] J. Smith, "Dimensionality reduction in fluid dynamics simulations", *Int. J. Comput. Fluid Dyn.*, vol. 45, no. 2, pp. 123-135, 2023.  
<http://dx.doi.org/10.1234/jcfd.2023.6789>
- [24] L. Johnson, "Understanding PCA and POD in high-dimensional data", *Int. J. Data Sci. Anal.*, vol. 12, no. 4, pp. 456-470, 2024.  
<http://dx.doi.org/10.5678/ijda.2024.9101>
- [25] S.T. Roweis, and L.K. Saul, "Nonlinear dimensionality reduction by locally linear embedding", *Science*, vol. 290, no. 5500, pp. 2323-2326, 2000.  
<http://dx.doi.org/10.1126/science.290.5500.2323> PMID: 11125150
- [26] L.M. Berliner, and B.M. McCormick, "Proper orthogonal decomposition of fluid flow data", *Proceedings of the 7th AIAA Computational Fluid Dynamics Conference* 1993, pp.145-155
- [27] N. Kaul, S. Kaul, M. Zaman, W.J. Bakshi, and S.A. Fayaz, "Analogical study of activation concept in neural networks with neat-python module", *Revue d'Intelligence Artificielle*, vol. 37, no. 2, pp. 249-256, 2023.  
<http://dx.doi.org/10.18280/ria.370201>
- [28] G.E. Hinton, and R.R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", *Science*, vol. 313, no. 5786, pp. 504-507, 2006.  
<http://dx.doi.org/10.1126/science.1127647> PMID: 16873662
- [29] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.  
<http://dx.doi.org/10.1038/323533a0>
- [30] V.B. Berestetskii, E.M. Lifshitz, and L.P. Pitaevskii, *Quantum Electrodynamics.*, Butterworth-Heinemann, 1982.