FINAL REPORT

Project Title: *Probing Particle Impingement in Boilers Using High-Performance Computing with Parallel CPUs and GPUs*

Award no: DE-FE0031746

University of California, Riverside
c/o Office of Research and Economic Development
200 University Office Building
University of California
Riverside, CA 92521

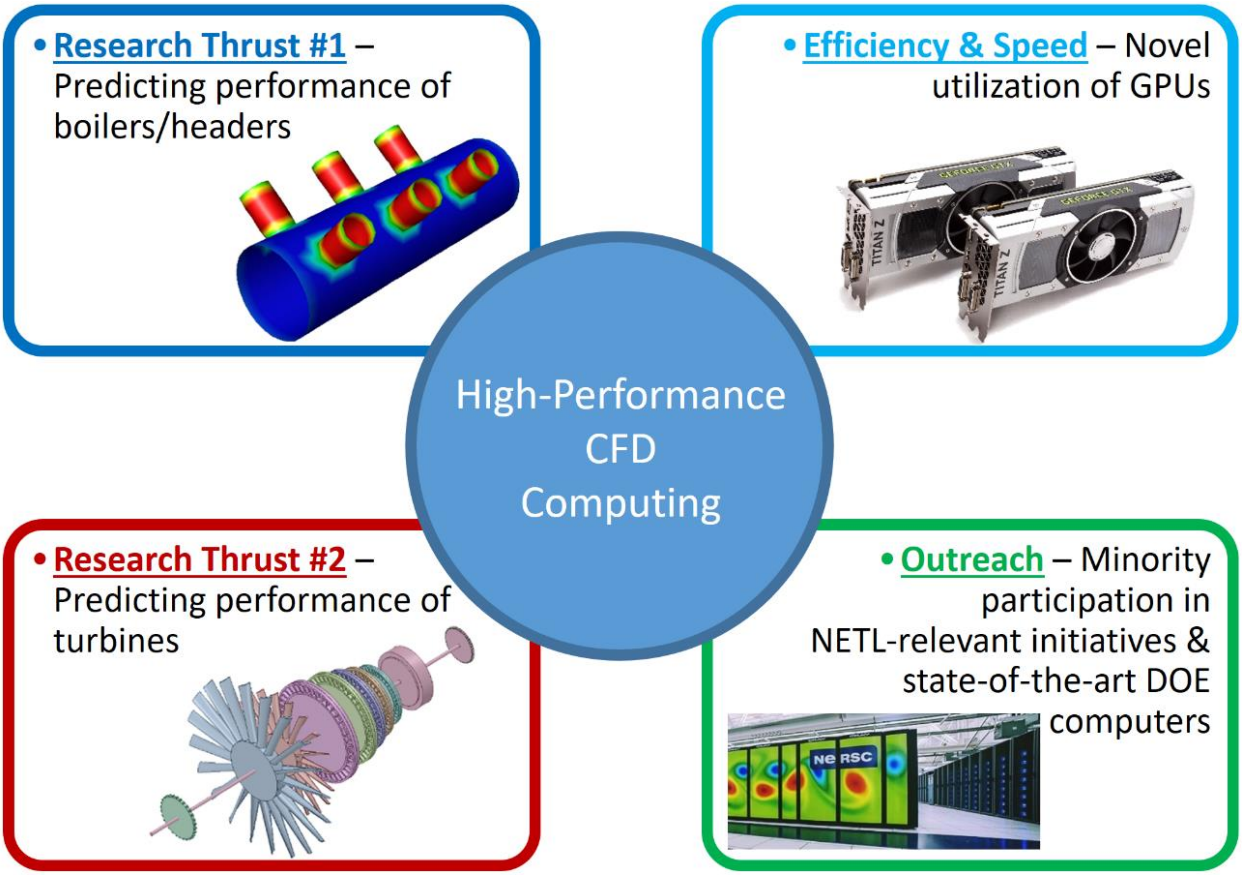
UEI number: MR5QC5FCAVH5

PRINCIPAL INVESTIGATOR

Prof. Bryan M. Wong
(951) 827-2163
bryan.wong@ucr.edu

Abstract/Executive Summary

The major goals of the project are to calculate and analyze particle impingement within boilers, quantify effects of particulates in boilers, and predict damage rates of boilers under different cycling modes. Collectively, these initiatives develop insight into existing coal plant challenges using advanced modeling tools, *particularly those leveraging high-performance computing resources*. As depicted in the figure below, high-performance CFD computing forms a ***central theme*** in this project that will employ a high degree of coordination and communication between these initiatives to realize a final, rigorously sound, and validated computational capability upon completion. These results will create a holistic, comprehensive, systems-level assessment of damage rates under different cycling modes. *Together, these objectives will develop critical insight into damage mechanisms in existing coal plant challenges for accurately and efficiently assessing operating performance in fossil energy power plants*.



This final report is divided into two sections entitled "Section 1: Predicting Complex Erosion Profiles in Steam Distribution Headers with Convolutional and Recurrent Neural Networks" and "Section 2: FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion" which summarize the findings of this project. This work was supported by the U.S. Department of Energy, National Energy Technology Laboratory (NETL) under Award No. DE-FE0031746.

Section 1: Predicting Complex Erosion Profiles in Steam Distribution Headers with Convolutional and Recurrent Neural Networks

Section 1 Abstract: The effects of erosion due to particle impingement continue to be of immense concern in various energy and technology industries. Brute force computational fluid dynamics (CFD) approaches allow accurate predictions of complex erosion processes; however, these large-scale calculations can be very computationally expensive. Specifically, when different initial conditions are required to analyze the system, the CFD simulations must be re-started *de novo* without recourse to previously converged cases. To address these issues, we harness Convolutional Neural Network (CNN) and Long and Short-Term Memory (LSTM) machine learning approaches to predict complex surface erosion profiles in steam distribution headers for the first time. We show that this hybrid machine learning approach can accurately predict *entire particle trajectories* and surface erosion profiles when only initial positions and velocities are inputted into the algorithm. Most importantly, our approach is 600 times faster than conventional CFD calculations and gives impressive R^2 scores of 0.91 and 0.71 for particle trajectories and surface erosion profiles, respectively. Taken together, our hybrid machine learning approach is a promising technique for accurately predicting surface erosion rates but with a significantly reduced computational cost.

1.1. Introduction

Erosion due to particle impingement results in irreversible material degradation due to repeated impact of high-velocity particles on surfaces. This process causes perpetual wear of critical components in various energy and technological industries, such as those used in petroleum refining,[1] helicopter rotor blades,[2] aircraft engine blades,[3] and pipelines in coal power plants.[4] Mitigating these deleterious effects is particularly crucial since the financial loss due to erosion is estimated to cost 1 – 4% of the gross national product in industrialized nations.[5] Since these erosion processes depend on a multitude of control variables, *in situ* investigation of these processes is impractical since the number of experiments is typically immense. Computational fluid dynamics (CFD) can shed insight into these erosion processes by direct solution of the Reynolds-Averaged Navier Stokes (RANS) equations;[6] however, these numerical approaches can be computationally expensive. Specifically, when different initial conditions are required to analyze the system, the CFD simulations must be re-started *de novo* without recourse to previously converged cases.

To address these computational bottlenecks, machine learning (ML) approaches for approximating numerical CFD calculations have started to gain significant attention in the scientific literature. For example, recent studies have utilized new response surface models in conjunction with CFD approaches.[7,8] Tran Anh et al.[9,10] proposed neural network approaches based on Gaussian process modeling (GPM) for predicting erosion on a 3-D impeller wear model. Similarly, Bahrainian et al.[11] and Bakhshesh et al.[12] also used GPM to predict solid particle erosion in standard elbow pipes. Building on this work, Dai et al.[13] used a hybrid model combining dimensional analysis with the Buckingham π theorem and GPM for predicting erosion rates. Finally, Zahedi et al.[14] used a random forest algorithm to predict erosion in elbow pipes, and Pandya et al.[15] trained a neural network from pre-computed numerical CFD calculations, which were used as input to their ML algorithms. Despite the impressive accuracy of these prior approaches, it is worth noting their limitations. For example, GPM-based models are task-specific and may require sophisticated approximate Bayesian inference algorithms that use more

computational power than the deep-learning models themselves.[16] In addition, the neural network by Pandya et al.[15] could only predict the total erosion rate (as opposed to the *spatially varying* distribution of surface erosion) and, therefore, did not provide geometric details for these erosion processes.

In this work, we augment these prior ML approaches to enable predictions of entire particle trajectories and erosion distributions on surfaces for the first time. Most notably, we harness a hybrid approach composed of two ML approaches that accurately predict surface erosion rate distributions given only initial conditions on particle size, main-inlet speed, sub-inlet speed, main-inlet pressure, and sub-inlet pressure. In other words, our hybrid ML approach can side-step the computationally intensive CFD calculations by predicting entire trajectories and surface erosion profiles from minimal information associated with the initial conditions. We give a detailed description of our algorithms and provide performance analyses of our hybrid ML approach, including accuracy tests, feature importance analyses, and computational efficiency metrics. Finally, we conclude with a brief summary and discussion of potential future applications of our hybrid machine learning approach for predicting surface erosion rates in these complex geometries.

1.2. Computational Details

The Eulerian-Eulerian and the Eulerian-Lagrangian models[17] are the two most common approaches for CFD particle tracking in a gas flow. The former approximates both the gas flow and the solid phase as a continuum, whereas the latter treats the gas flow as a continuous phase and the solid phase as discrete. In the Eulerian-Lagrangian model, the movement of each solid phase is calculated separately, which is the approach used in this work. Also known as discrete phase modeling (DPM), this numerical approach has been shown to give accurate predictions of various experimental cases of erosion.[18–20] Within the DPM method, a finite element mesh is used to discretize a given geometry onto which the particle dynamics in the fluid are numerically obtained by solving the Reynolds-averaged Navier–Stokes (RANS) equations:[21]

$$\nabla \cdot \overline{u} = 0, \tag{1}$$

$$\frac{\partial \overline{u}}{\partial t} + \rho(\overline{u} \cdot \nabla)\overline{u} = -\nabla \overline{p} + \eta \Delta \overline{u} - \nabla \cdot \tau^{RS} + \overline{f}_D, \tag{2}$$

where $\overline{u}$ and $\overline{p}$ are the average flow velocity and pressure, respectively, ρ is the fluid density, η is the dynamic viscosity, τ^{RS} is the Reynolds stress term for an eddy-viscosity approach, and $\overline{f}_D$ is the additional body forces (such as gravity and the Stokes drag force). Equations (1) – (2) conserve mass and momentum, respectively. In this study, the *k*-ω Shear Stress Transport (SST) turbulence model with the steady-state method was used. In addition, both ρ and η are approximated as constants since the fluid phase is incompressible. We specifically chose the *k*-ω SST formalism since it is the most commonly used approach in industrial applications due to its high accuracy to expense ratio.[22]

The particle trajectories were obtained by integrating the force balance equation written in a Lagrangian reference frame[23] given by

$$m_p \frac{dv_i^k}{dt} = \left(m_p - m_f\right) g_i + F_d^k, \tag{3}$$

where v_i^k is the k^{th} particles velocity in the i^{th} direction, m_p and m_f are the particle and fluid mass, respectively, g_i is the acceleration due to the gravity, and F_d^k is the Stokes drag force acting on the k^{th} particle due to the relative velocity between the fluid and the particle. The particle motion in

each mesh element is calculated by accounting for the explicit couplings between the differential equations describing the fluid and particle phases.

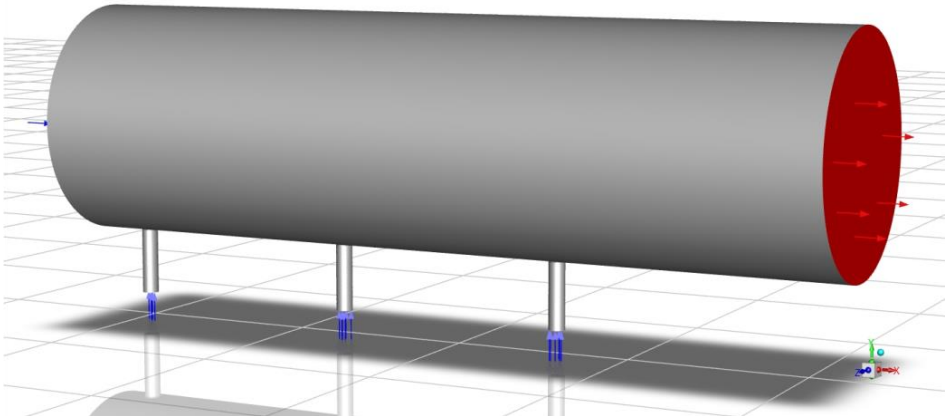
The erosion rate (ER) profile[24] for a given geometry can be calculated with the following expression:

$$\mathrm{ER} = \sum_{p=1}^{N} \frac{\dot{m}_p C(d_p) f(\alpha) v^{b(v)}}{A_{\mathrm{face}}}, \tag{4}$$

where $\dot{m}_p$ is the mass rate of particles impacting the surface, $C(d_p)$ is a Tulsa Angle Dependent Model function[25] of the particle diameter d_p, $f(\alpha)$ is a function of impact angle, v is the relative velocity, $b(v)$ is the Tulsa Angle Dependent Model function of particle relative velocity,[25] and A_{face} is the area of the cell face at the wall. The particle impact velocity is the largest contributor in Equation (4) since the erosion rate is proportional to the n^{th} power of the particle velocity.[26–28] Various erosion models, such as those proposed by Finnie[27] and Oka,[28] suggest different values of n to evaluate ER. The Finnie erosion model recommends setting $n = 2$ for ductile materials,[29,30] which is the convention used in our work since it is the most relevant for the steam distribution header investigated in this project. The ANSYS Fluent 19.2 software was used to obtain the particle trajectories and Finnie erosion profiles that were used as training data for our ML simulations.

We used the OP-650 steam distribution header (commonly used in sub-critical coal plants) as a representative geometry for our ML study on surface erosion distributions.[31–33] Fig. 1a shows the geometry of the steam distribution header with three-branched pipelines, and Fig. 1b shows the finite element mesh surface used in all of our calculations. An extremely fine polyhedral mesh containing a total of 413,685 elements and 38,312 surface elements (each element has an average edge length of 1.5×10^{-2} m) were used to resolve the detailed erosion profiles on the surfaces of the header geometry. We evaluated the quality of our mesh by computing both its orthogonal quality and aspect ratio. The orthogonal quality defines the degree to which the mesh is not orthogonal[33] with values close to 0 indicating higher quality meshes. The mesh grid used in our work has a minimum orthogonal quality of 0.208 and an average value of 0.773 ± 0.114. The aspect ratio defines the ratio of cells in a different dimension,[33] where values closer to 1 indicate higher quality meshes. The mesh grids used in our work have a maximum aspect ratio of 8.682 and an average value of 1.842 ± 0.449. In addition, we carried out a sensitivity study of our CFD calculations to test how particle trajectories and wall erosion patterns are affected by the Reynolds number and the absolute roughness coefficient of the wall.

(a)



(b)

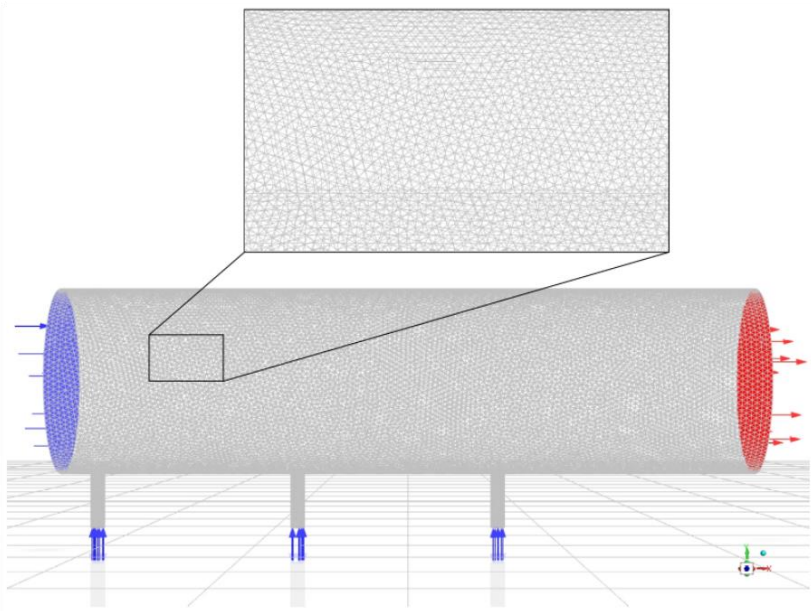


Figure 1. (a) Geometry and (b) surface mesh of the OP-650 steam distribution header used in this work. The inset of panel (b) shows a magnified view of the mesh, which contains a total of 38,312 polyhedral mesh elements, each of which has a mean size of 1.5×10^{-2} m.

The main and reheat steam parameters used in this work are (540 °C, 14 MPa) and (540 °C, 2.3 MPa), respectively.[32] The minimum and maximum pressures used for the steam line in this work were based on the pressure profiles of the OP-650 boiler under operation.[32,34] The steam flow rate profile in the mainstream was used to calculate the maximum and minimum initial flow velocities,[35] which were obtained by multiplying the flow rate with the specific volume and dividing by the cross-sectional area of the pipe. The anthracite particle sizes investigated in this study ranged from 40 to 60 μm, which have been shown to be mixed in the macro steam line.[36] Our ML study focused on five parameters that contribute significantly to erosion: particle size, main-inlet speed, sub-inlet speed, main-inlet pressure, and sub-inlet pressure. The minimum and maximum ranges for each parameter were based on the actual operating range of the OP-650

boiler. To thoroughly explore this large parameter space required for our ML algorithms, we divided the range of each parameter into five intervals to obtain 5^5 = 3,125 combinations, schematically shown in Fig. 2.

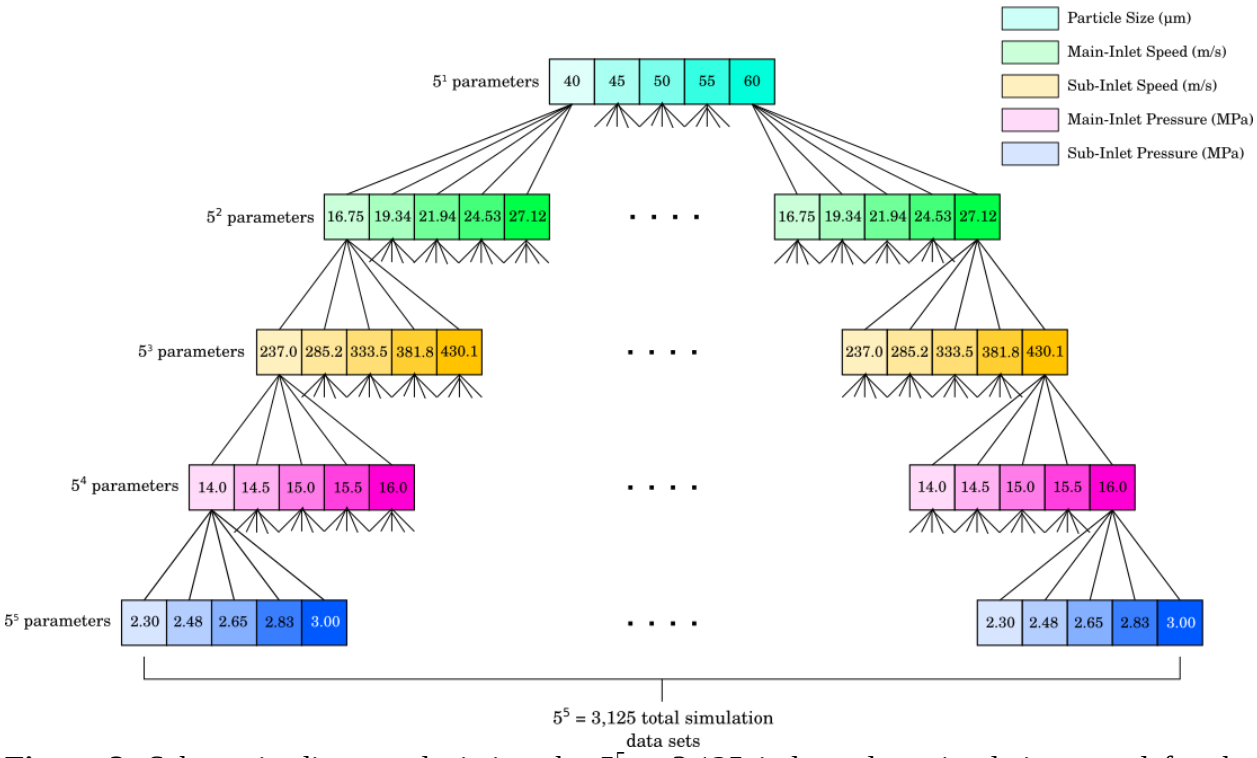


Figure 2. Schematic diagram depicting the 5^5 = 3,125 independent simulations used for the machine learning algorithms in this work. The 3,125 simulations span the parameter space of particle sizes, main-inlet speeds, sub-inlet speeds, main-inlet pressures, and sub-inlet pressures used in an OP-650 steam distribution header.

Fig. 3 shows a schematic of our deep-learning-based erosion prediction model. First, the particle trajectories and surface erosion profiles were organized into a 4-D array with dimensions of 3,125×50×196×3, where 3,125 is the number of simulation datasets, 50 is the number of time steps (each time-step interval = 2.49 ms), 196 is the number of particles, and the last dimension contains the (x, y, z) Cartesian coordinates of the particles. As mentioned in the Introduction, the intent of our work is to harness ML to predict entire particle trajectories and surface erosion rate distributions *when only initial positions and velocities are inputted into the algorithm*. As such, the input to the LSTM ML model excludes all time-series data (i.e., the 2nd dimension in the original 3,125×50×196×3 array is removed) and includes the five initial parameters (particle size, main-inlet/sub-inlet speed, and main-inlet/sub-inlet pressure), resulting in a final dataset of size 3,125×196×8. Further details of the LSTM algorithm are given further below.

The predicted trajectories from the LSTM algorithm (having dimensions of 3,125×50×196×3) were then used as input to the CNN ML model. As before, the five initial parameters were also added, giving an augmented dataset of size 3,125×50×196×8. The output of the CNN model is a vector containing 38,312 data points, which corresponds to the number of elements in the surface mesh. To train our ML model, we used 32 Intel Broadwell CPU cores with 15 Gigabytes of RAM per CPU core.

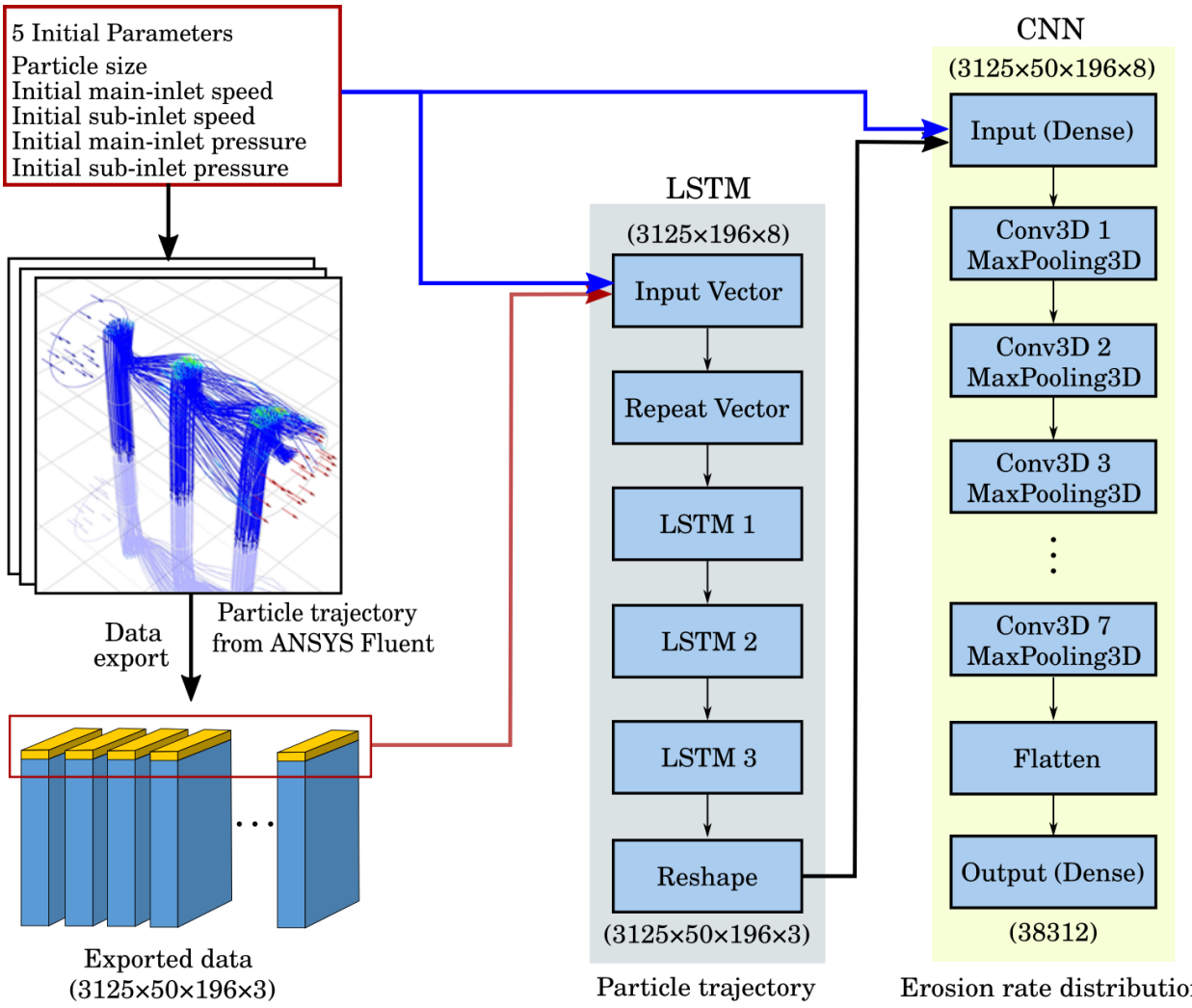


Figure 3. Workflow diagram for the LSTM and CNN architectures used in this work. Five initial parameters were used for the CFD dataset generation, which were exported into a data frame. The LSTM algorithm predicts entire particle trajectories given only initial positions and velocities. The output of LSTM is used as input to a CNN, which predicts surface erosion rate distributions.

We now briefly discuss the algorithms used to predict the particle trajectories and surface erosion rate distributions in the LSTM and CNN methods, respectively. The LSTM approach utilizes a chain-like structure with a series of connected LSTM units (also known as cells). In short, these units detect features from an input sequence and attempt to learn long-term dependencies.[37] To accomplish this, each LSTM unit (shown in Fig. 4) consists of a cell state and three gates that regulate information flow: a forget gate, an input gate, and an output gate.[38] The cell state maintains the information for learning long-term dependencies and is transmitted into the next LSTM unit. The forget gate decides which data from the previous hidden state should be forgotten, and the input gate updates the cell state using the hidden state and current input. Lastly, the output gate determines the value of the next hidden state.

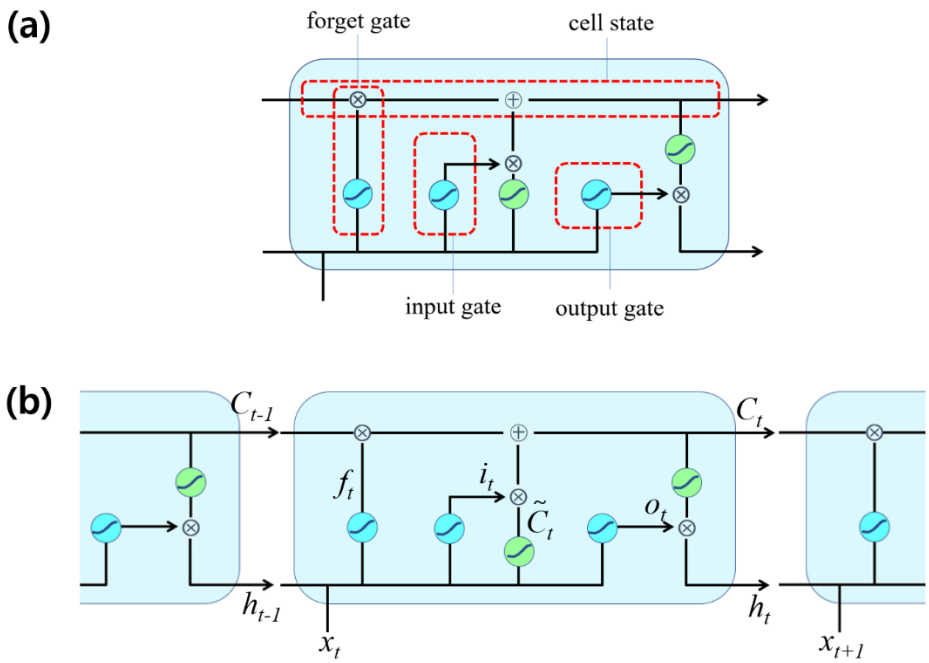


Figure 4. (a) LSTM unit and (b) LSTM unit at time *t* between two adjacent LSTM units.

The blue circles in Figs. 4a and b represent the sigmoid activation function, and the green circles denote the hyperbolic tangent activation function. The ⊗ and ⊕ symbols are the element-wise multiplication and addition operator, respectively. The LSTM unit at time *t* performs the calculations described in the following equations:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \quad (5)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad (6)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \quad (7)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t, \quad (8)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad (9)$$

$$h_t = o_t \cdot \tanh(C_t), \quad (10)$$

where f_t is the forget gate vector, i_t is the input gate vector, $\tilde{C}_t$ is the candidate cell state, C_t is the cell state, o_t is the output gate vector, and h_t is the hidden state at time *t*. W_f, W_i, W_C, and W_o are the weights of their given subscripts, and b_f, b_i, b_C, and b_o are the biases on the assigned gates or states. The cell state C_t is updated based on the previously hidden state h_{t-1} and the current input x_t; that is, f_t decides what information should be forgotten from the cell state. The input gate decides which information, i_t, should be kept in the cell state with a vector $\tilde{C}_t$ containing new candidate values. The hidden state, h_t, is computed with o_t for the next LSTM unit. Collectively, the LSTM approach removes or adds information to the cell state to carry long-term dependencies over arbitrary time intervals. The weights (W_f, W_i, W_C, and W_o) and biases (b_f, b_i, b_C, and b_o) are adjusted by backpropagation. In the context of our CFD erosion calculations, the LSTM model automatically learns the positions of the particles for each time step *when only the initial conditions are specified*.

The predicted trajectories from the LSTM model are then inputted into a CNN algorithm to predict the resulting surface erosion rate distributions. The CNN algorithm plays a key role in our work since the 1D trajectories from LSTM alone are insufficient to accurately predict the 2D erosion profiles on header surfaces. CNN ML algorithms have been previously used for video recognition and 3D-image analyses such as magnetic resonance imaging[39] and computed

tomography.[40] In a similar fashion, we harness the image-processing capabilities of these CNN algorithms to predict the complex erosion profiles on the surfaces of the OP-650 boiler geometry. Our CNN model consists of seven consecutive sequences with 3D-Convolution and 3D-Maxpooling layers[41–43] (described further below) for capturing the complex dynamics and trends in surface erosion rates.

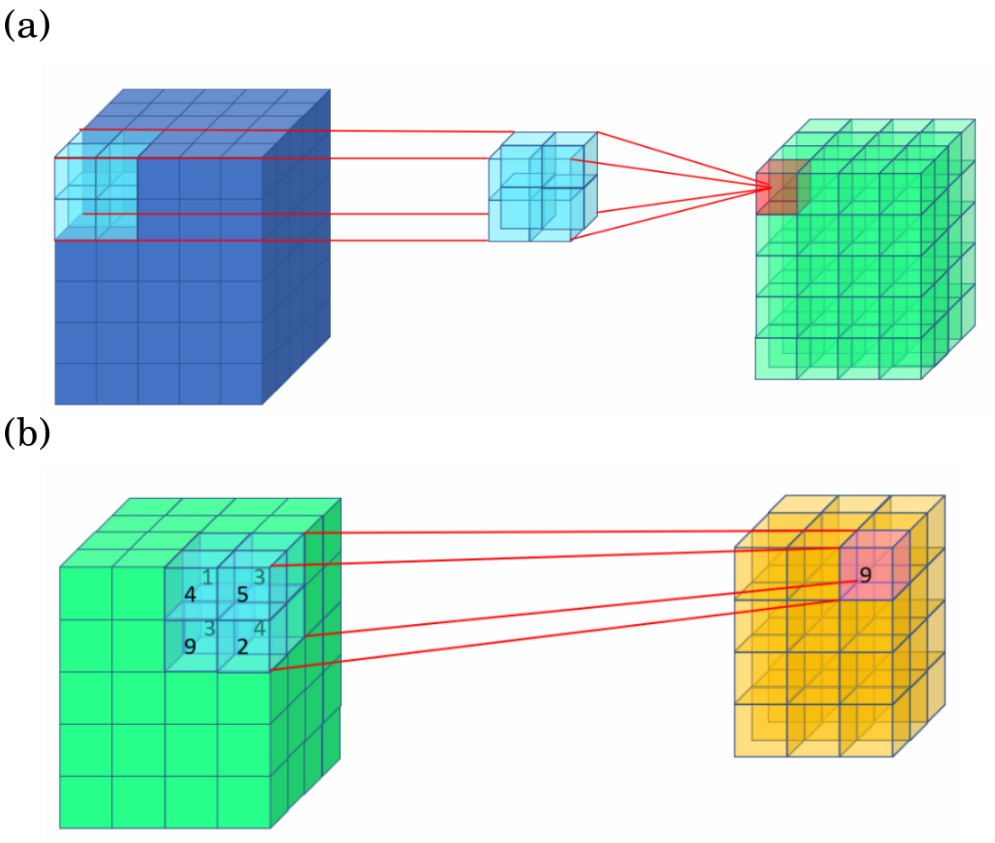


Figure 5. (a) Schematic of 3D-Convolution algorithm: the cyan-colored 2×2×2 kernel is processed via filters and summed to construct the red-colored cell. (b) Schematic of the 3D-Maxpooling algorithm: the maximum number in the cyan-colored 2×2×2 kernel is assigned to the red cube.

Figure 5(a) shows a schematic of 3D-Convolution layer. In this example, the original input 3D data has dimensions of 5×5×6, and the convolution filter size is 2×2×2. Each cell in the convolution filter has a random value, which is updated later with backpropagation. The kernel is the original dataset but has the same dimensions as the convolution filter. The kernel values are multiplied with the filter's values, then summed via a dot product between the kernel and the filter. The 2×2×2 kernel slides over by one voxel, repeating the process until the kernel covers the entire original data. Figure 5(b) shows a schematic of the 3D-Maxpooling process. After convolution, the data are scanned with the 2×2×2 kernel. The number of voxels to be shifted over is known as a stride; for example, if the stride is 2, then the 2×2×2 kernel shifts over two voxels. The maximum value in the kernel is saved in the output; the kernel then strides by one voxel and repeats the same operation. In short, both the convolution and maxpooling operations reduce the dimensions of the original dataset after the data is processed. For example, original data with dimensions of 5×5×6

will be reduced to a 4×4×5 dataset after the 3D-Convolution process and finally to a 3×3×4 dataset after the 3D-Maxpooling process. Padding is used on each operation layer to maintain the spatial resolution of the data; i.e., zero-valued voxels are added at the edge of the original dataset to preserve its dimensions. In our study, we performed successive 3D-Convolution and 3D-Maxpooling operations for multiple levels of abstractions and epochs to learn trends and validate their values with backpropagation.

To quantitatively assess the performance of our ML predictions, we computed the R^2 score (the coefficient of determination) and the Root Mean Square Error (RMSE). The R^2 score measures the total variation in the output from the model, which is calculated using the following equation44:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}\left(y_i - \overline{y}_i\right)^2}, \tag{11}$$

where N is the size of the dataset, y_i is the i^{th} value of the validation set, $\hat{y}_i$ is the predicted value, and $\overline{y}_i$ is the mean of the validation set. While the R^2 score is a relative measure of how well the model fits the observed data, the RMSE gives an absolute measure of the goodness for the fit (values close to 0 indicate the prediction is close to the original data), given by the following equation:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}. \tag{12}$$

A K-fold cross-validation approach,45,46 which is widely used for validating ML models, was used to mitigate overfitting. The K-fold model divides all samples into k groups of equal size and categorizes them into training and validation sets with a specific ratio. The training set is used to train the ML model, then tested with the validation set, which is repeated k times. We set k = 5, and the ratio of the training set to the validation was set to 8:2. Finally, we calculated the R^2 score and RMSE for each k value and averaged them to quantitatively assess the performance of our ML models. The computer codes for processing the CFD data, training, and ML models can be downloaded from a Github repository at https://github.com/SDY159/CFD_ML.

1.3. Results and Discussion

1.3.1. ML Performance for Predicting Particle Trajectories and Surface Erosion Profiles

Training our ML algorithms over several epochs to minimize the loss gives LSTM and LSTM+CNN predictions of particle trajectories and surface erosion profiles, respectively. For trajectories predicted by the LSTM model, we obtained an impressive mean R^2 score of 0.909 $\pm$ 0.006 and a mean RMSE of 0.172 $\pm$ 0.002 m. For surface erosion rates predicted by the LSTM+CNN model, we obtained a mean R^2 score of 0.71 $\pm$ 0.012 and a mean RMSE of 0.00010 $\pm$ 0.00001 kg/m^2·s. Fig. 6 depicts two representative simulations comparing our LSTM+CNN predictions against the brute-force CFD results. We specifically chose simulation #296 among our 3,125 simulation datasets and a validation dataset not included in the original simulation datasets to validate our ML model. The parameters used in the validation data are as follows: particle size: 42 μm, main-inlet speed: 20 m/s, sub-inlet speed: 300 m/s, main-inlet pressure: 15.25 MPa, and sub-inlet pressure: 2.5 MPa.

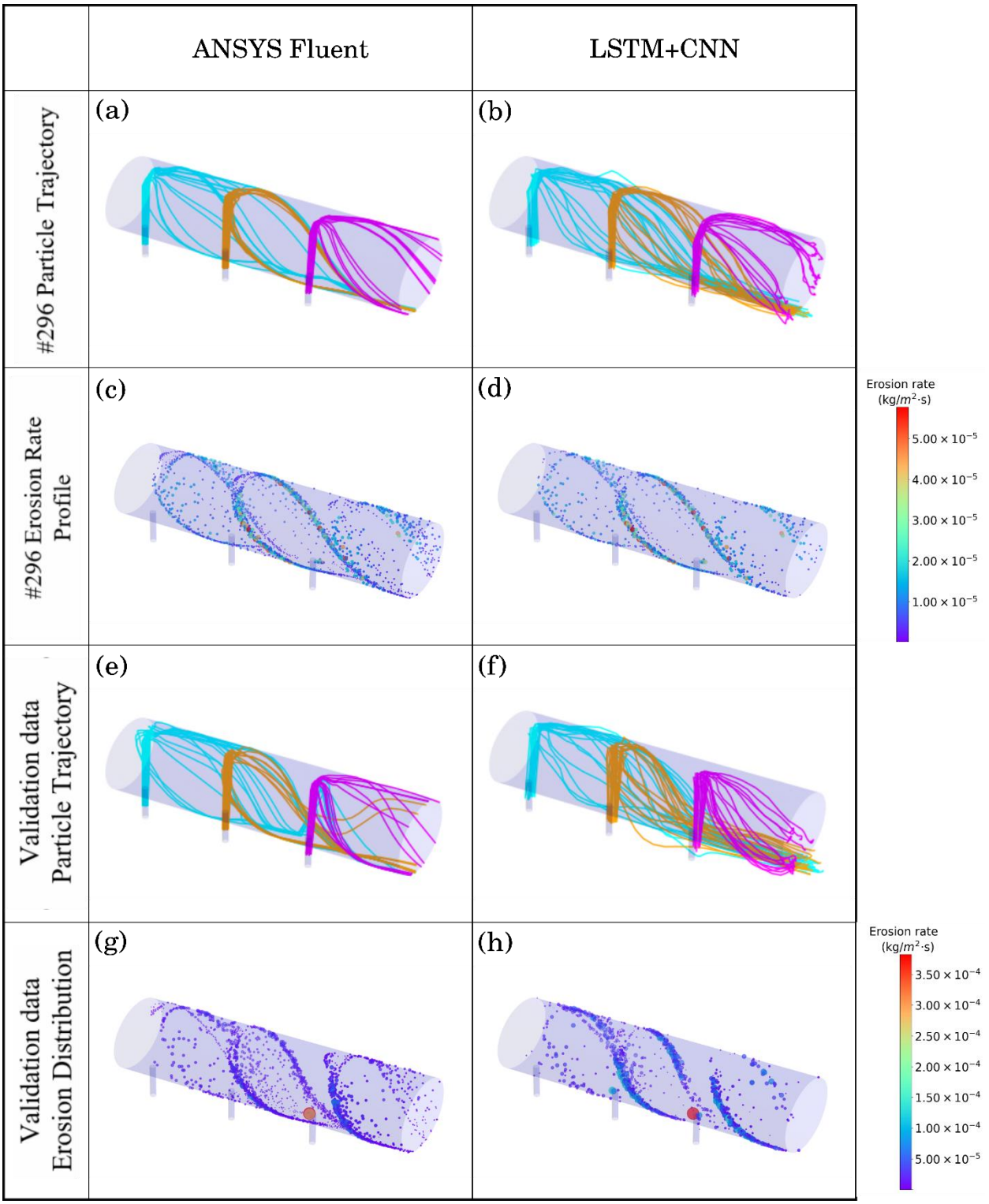


Figure 6. Comparison between numerical ANSYS Fluent CFD calculations and predictions from our LSTM+CNN approach. Panels (a) and (c) depict ANSYS Fluent CFD particle trajectories and surface erosion profiles, respectively, and panels (b) and (d) show the corresponding results predicted by the LSTM+CNN ML algorithm for simulation #296 (which exhibits laminar flow). Panels (e) and (g) depict ANSYS Fluent CFD particle trajectories and surface erosion profiles,

respectively, and panels (f) and (h) show the corresponding results predicted by the LSTM+CNN ML algorithm for the validation data.

As shown in Figs. 6a, b, e, and f, the trajectories predicted by our hybrid LSTM+CNN ML approach match extremely well with the numerical CFD calculations. This is particularly impressive since the LSTM+CNN algorithm utilizes *only initial positions and velocities* but can nonetheless still predict entire trajectories of the particles in the system. The LSTM+CNN algorithm also gives good agreement for our validation data (panels e and f), and only a handful of trajectories deviate from the CFD numerical calculations for this complex set of conditions. We observe a similarly good agreement between our ML predictions and CFD calculations for surface erosion profiles shown in Figs. 6c, d, g, and h. Compared to our trajectory predictions, the erosion rates are slightly underestimated by the LSTM+CNN approach; however, surfaces with the highest erosion rates (which are of primary concern to energy and technological industries) are predicted quite accurately.

1.3.2. Feature Importance Analysis

In addition to the trajectories and erosion profiles discussed in the previous section, we also carried out a feature importance analysis of the initial parameters to give insight into which specific variables primarily contribute to the performance of the underlying algorithm.[40] To carry out this analysis, we excluded each initial parameter, one at a time, and re-trained our ML model against the benchmark CFD results. The RMSE and R^2 scores were then re-computed to assess the significance of each excluded initial parameter in our ML model. Fig. 7 shows a summary of our findings compared against our "full" ML models that include all initial parameters, represented by the dashed red line in each panel.

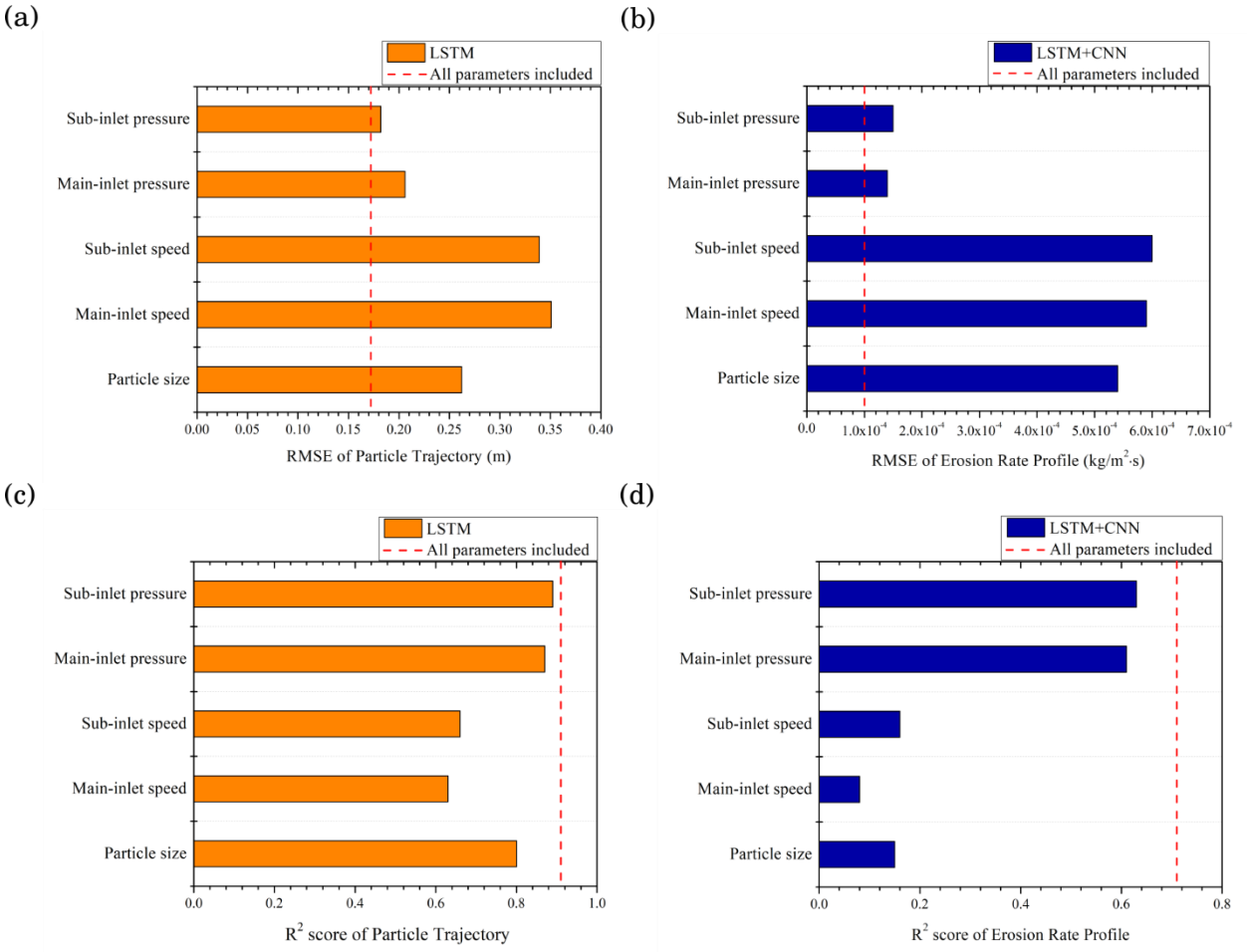


Figure 7. Feature importance analysis for the five initial conditions/settings explored in this work. Panels (a) and (b) show RMSEs for particle trajectories and surface erosion rates predicted by LSTM and LSTM + CNN, respectively. Panels (c) and (d) depict R^2 scores for particle trajectories and surface erosion rates predicted by LSTM and LSTM + CNN, respectively. The red dashed lines in all of the panels represent our "full" model, which includes all initial parameters.

Figures 7a and b show that the RMSE increases significantly when the sub-inlet or main-inlet speed is excluded. Similarly, Figs. 7c and d also indicate that the R^2 scores are low when the sub-inlet speed or the main-inlet speed information is not provided. Interestingly, we find that including data on the initial pressure does not give significantly different results; however, including particle size information appears to be more crucial. As shown in all panels of Fig. 7, the initial speed is the most important contributor to ML accuracy. This is perhaps not too surprising since the erosion rate is proportional to the n^{th} power of the particle velocity, as discussed in Section 2. However, it is important to note that none of the ML algorithms were directly trained on the functional form of Eq. 4, yet the LSTM and LSTM+CNN methods automatically recognize this strong dependence and designate this variable to be the most determining contributor in our study.

1.3.3. Assessment of Computational Efficiency

In addition to the performance metrics and feature importance analyses described previously, we compare the computational efficiency of our finalized ML models against the ANSYS Fluent CFD calculations in this section. At this point, it is worth mentioning some of the limitations inherent to our ML approach. In particular, the LSTM+CNN approach requires numerous CFD calculations for constructing a training dataset for a given geometry. Despite this limitation, once the LSTM+CNN model is constructed, it can be easily utilized/transferred to other users who (1) may not have the expertise to use advanced CFD software or (2) have access to high-performance computing resources. For example, using the ANSYS Fluent CFD software correctly requires an intermediate-to-advanced level of expertise to set numerous computational parameters including finite element mesh size, specific erosion model, flow velocity, temperature, particle size, main-inlet speed, sub-inlet speed, main-inlet pressure, sub-inlet pressure, and many others. In contrast, the LSTM+CNN model constructed in this work only requires 5 input parameters but gives similar accuracy as the full CFD model and is more easily accessible to non-CFD experts requiring erosion profiles to diagnose steam distribution headers. Similarly, for users who may not have access to high-performance computing (such as developing countries) or field workers unable to access supercomputing resources but need to diagnose their system "on the fly," the LSTM+CNN model developed in this work can provide accurate erosion profiles on a conventional laptop. To this end, Fig. 8 compares the computational times of our finalized LSTM+CNN model against the ANSYS Fluent CFD calculations for the validation datasets. From these representative calculations, we find that the average computation times for our CFD and ML calculations are 1093.97 ± 69.82 s and 1.82 ± 0.11 s, respectively. On average, our ML approach is more than 600 times faster than the numerically intensive CFD calculations and gives a similar accuracy for predicting surface erosion rates but with significantly less computational time/effort.

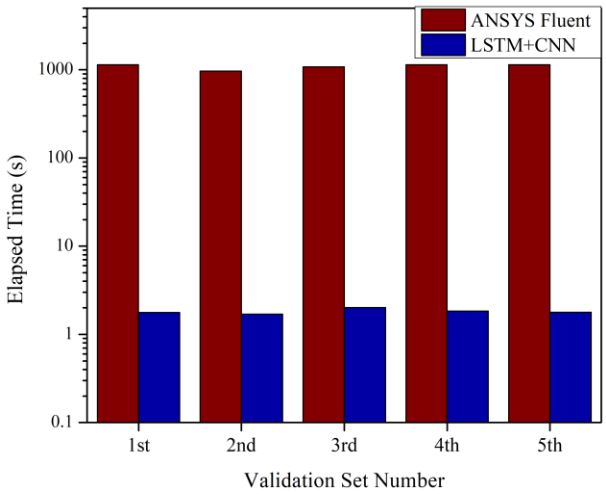


Figure 8. Comparison of computational timings between ANSYS Fluent CFD calculations and the LSTM+CNN ML algorithm.

1.4. Conclusion

In summary, we have presented the first hybrid LSTM+CNN ML approach for accurately and efficiently predicting surface erosion rates on a representative steam distribution header geometry. The most salient result of our study is that our hybrid LSTM+CNN approach accurately

predicts *entire particle trajectories and surface erosion rate distributions* when only initial positions and velocities are inputted into the algorithm. Our hybrid ML model was constructed by carrying out a series of CFD calculations on fluid flow and particle transport obtained from a variety of initial parameters, including particle size, main-inlet speed, sub-inlet speed, main-inlet pressure, and sub-inlet pressure. An LSTM algorithm was then trained to predict these trajectories (given only particle positions and velocities), which were subsequently used as input to a CNN algorithm for predicting surface erosion profiles. Taken together, our hybrid LSTM+CNN approach gives impressive R^2 scores of 0.91 and 0.71 for particle trajectory and surface erosion predictions, respectively. We also carried out a feature importance analysis, which showed the initial speed to be the largest contributing factor for ML accuracy. Finally, a computational analysis of our results demonstrates that our LSTM+CNN approach is more than 600 times faster than numerical CFD calculations without loss of accuracy. Because of its simplicity and computational efficiency, our LSTM+CNN model can be directly utilized/transferred to users that may not have the expertise to run advanced CFD calculations or have access to high-performance computing to calculate these erosion profiles in their system.

A Github repository containing all the CVD and ML models used in this work can be accessed at https://github.com/SDY159/CFD_ML

Section 2: FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion

The following work was published in its final form in Ref. 74.

Section 2 Abstract: The deleterious impact of erosion due to high-velocity particle impingement adversely affects a variety of engineering/industrial systems, resulting in irreversible mechanical wear of materials/components. Brute force computational fluid dynamics (CFD) calculations are commonly used to predict surface erosion by directly solving the Navier Stokes equations for the fluid and particle dynamics; however, these numerical approaches often require significant computational resources. In contrast, recent data-driven approaches using machine learning (ML) have shown immense promise for more efficient and accurate predictions to sidestep the computationally demanding CFD calculations. To this end, we have developed FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer), a new hybrid ML architecture for accurately predicting particle trajectories and erosion on an industrial-scale steam header geometry. Our FLUID-GPT approach utilizes a Generative Pre-Trained Transformer 2 (GPT-2) with a Convolutional Neural Network (CNN) for the first time to predict surface erosion using only information from five initial conditions: particle size, main-inlet speed, main-inlet pressure, sub-inlet speed, and sub-inlet pressure. Compared to the Long- and Short-Term Memory (LSTM) ML techniques used in previous work, our FLUID-GPT model is much more accurate (a 54% decrease in mean squared error) and efficient (70% less training time). Our work demonstrates that FLUID-GPT is an accurate and efficient ML approach for predicting time-series trajectories and their subsequent spatial erosion patterns in these complex dynamic systems.

2.1. Introduction

Erosion due to high-velocity particle impingement continues to be a topic of pressing concern due to its deleterious effects in a variety of energy and technological industries. For example, erosion processes result in irreversible mechanical wear of materials/components in petroleum refining,[1] aircraft rotor/engine blades,[2,3] and pipelines in coal-fired power plants.[48,49] Recent studies have estimated that the financial loss due to erosion can reach up to several billions of US dollars in industrialized nations.[5] To mitigate these effects, computational fluid dynamics (CFD) is commonly used to solve the Navier-Stokes equations for the fluid and particle dynamics to shed insight into the specific mechanisms involved in these erosion processes. However, CFD calculations often require significant computational time and high-performance computing hardware, particularly for large-scale structures used in industrial power plants.

In recent years, there has been a rapid paradigm shift from "traditional" computational approaches to data-driven science, largely due to advances in computational power and the increasing availability of data. In the context of CFD and erosion calculations, recent machine learning (ML) approaches have shown remarkable accuracy and efficiency in a variety of systems.[10–15,50–55] In particular, previous work by us utilized a hybrid Long- and Short-Term Memory (LSTM) with a 3-dimensional Convolutional Neural Network (CNN) to predict particle trajectories and surface erosion, respectfully, in an industrial-scale boiler header.[55] While our approach was able to successfully predict surface erosion using only five initial conditions as input,

the LSTM training was computationally expensive due to its recurrence-based architecture, taking approximately 26 hours on 32 parallel CPUs.

To overcome this computational bottleneck, we present a new hybrid ML method, codenamed FLUID-GPT: Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer. FLUID-GPT utilizes a Generative Pre-Trained Transformer 2 (GPT-2) with a 3D CNN. The primary objective of FLUID-GPT revolves around the accurate prediction of particle trajectories and erosion phenomena within a sprawling industrial-scale boiler header.[21] GPT-2 is an attention-based model that falls under the umbrella of transformer models. Initially developed for tasks like natural language processing (NLP) and translation,[56] it has recently gained widespread attention due to its utilization in interactive Artificial Intelligence (AI) chatbots and forecasting models for time-series data.[57–59] A transformer model employs an encoder-decoder architecture in which the encoder generates a representation capturing relevant information from the input data. Subsequently, the decoder utilizes this representation to create output sequences through the attention mechanism iteratively. Recent research has demonstrated the effectiveness of focusing exclusively on the decoder for language modeling, leading to the development of a GPT model.[60]

This study employs our FLUID-GPT ML approach to forecast particle trajectories based on five initial parameters: particle size, main-inlet speed, main-inlet pressure, sub-inlet speed, and sub-inlet pressure. Subsequently, erosion predictions are generated using a CNN-based approach, utilizing the trajectories produced by our time-series models. We comprehensively describe our algorithms and conduct a comparative analysis of training efficiency and accuracy between our FLUID-GPT approach, LSTM, and Bidirectional LSTM (BiLSTM) for predicting particle trajectories and surface erosion. Finally, we offer a concise summary and discuss potential future applications of our hybrid ML approach. In conclusion, our study underscores the efficiency and accuracy of FLUID-GPT in analyzing sequential data, marking a significant step forward in predicting intricate particle dynamics.

2.2. Computational Methods

Our FLUID-GPT hybrid machine learning model combines GPT-2 and CNN within the PyTorch framework. We customized the GPT-2 architecture from the Transformer PhysX (TrphysX) package[58] to train our CFD dataset, which we carried out on a single Nvidia K80 GPU. The following sections describe the GPT-2 and CNN architecture, our data collection, and transformation process. We then describe our model training approach, including an early stopping criterion, which we used to compare computational efficiency across the various machine learning algorithms.

2.2.1. GPT-2 Model Architecture

Our utilization of the GPT-2 model for time-series analysis stems from its inherent ability to discern patterns and interconnections within sequential data points autonomously. It is important to highlight that the original GPT-1 algorithm was primarily designed for pre-training and focusing on acquiring linguistic structure and grammar. Subsequently, this foundational model undergoes refinement through supervised fine-tuning, requiring substantial labeled data pertinent to the specific task.[61] However, this reliance on supervised fine-tuning can present limitations as it necessitates the acquisition and processing of considerable labeled datasets, potentially restraining the model's versatility.

The more recent GPT-2 model, distinguished by its augmented input capacity and model size compared to GPT-1, introduced a groundbreaking approach known as "unsupervised fine-tuning." This innovation empowers the model to fine-tune for specialized tasks without requiring extensive task-specific data volumes. This flexibility enables the model to seamlessly adapt to varying tasks with reduced data prerequisites.[61] We leverage this unique feature by augmenting our input data to incorporate the five initial parameters (discussed in the Introduction) utilized in CFD calculations. This augmentation ensures the model's comprehensive learning of particle trajectories, preventing the need for supplementary training. Further elaboration on our feature engineering can be found in Section 2.2.3.1 - Data Collection.

GPT-2 model training is accomplished by calculating attention scores for each timestep, which measures the degree of association between the current timestep and other timesteps in the sequence.[60] In contrast to recurrent ML models, which process timesteps one at a time and propagate information from one stage to the next through a hidden state, the GPT-2 model employs self-attention. This enables the model to capture more complex relationships between the different timesteps in the sequence.

Positional encoding[60] is pivotal in enabling the GPT-2 model to distinguish between distinct timesteps accurately. This involves the introduction of unique patterns for each timestep, facilitating the model's ability to discern temporal variations. The GPT-2 model adeptly incorporates the foundational Transformer model's positional encoding approach. These patterns specify each timestep by integrating sine- and cosine-based positional patterns into the data's feature dimension. This can be equated to the addition of timestamps to a sequence of events, facilitating well-informed predictions and upholding chronological coherence throughout the prediction sequence.[60]

Figure 9 depicts the GPT-2 architecture,[61] composed of vertically stacked transformer decoder blocks integrated with positional encoding. Within each decoder block, a multi-head masked attention and a multi-layer perceptron (MLP) are enveloped by normalization[62] and dropout layers. The input time-series data is organized as an array (samples, timesteps, features) in the multi-head masked attention, where each timestep corresponds to a distinct set of features. This array's feature dimension is partitioned and allocated to individual attention heads, enabling simultaneous processing and comprehensive treatment of diverse aspects within the input sequence. This parallelization boosts the model's capability to capture short- and long-range dependencies, culminating in heightened predictive stability and precision.

As shown in the example in Figure 9, the input is distributed across four attention heads. Each attention head autonomously transforms the input sequence into three distinct vectors: query (Q), key (K), and value (V). These vectors are subsequently employed to calculate attention scores using the following formula:[60]

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{13}$$

where A designates the attention score matrix, Q stands for the query vector, K^T denotes the transposed key vector, V represents the value vector, and d_k signifies the dimension of the Q and K vectors. The Q vector corresponds to the current timestep within the considered input sequence, while K encompasses insights about all timesteps within the input sequence. The dot product between the Q and K establishes the similarity or association between different timesteps, enabling the model to concentrate on relevant parts of the data. The V vector carries the data to which the model directs its attention and is used to generate the output. After the computation of attention

scores, the individual attention heads merge via concatenation in the final linear layer, culminating in deriving the final attention matrix output.

Figure 10(a) illustrates the learning process involving time-series data. This data is segmented into portions of length **ω**, where **ω** is the window size. These segments undergo systematic processing utilizing a "striding" technique, wherein the window traverses the sequence with stride **s**, resulting in overlapping parts. In the training phase, the last data point within the window is omitted from the training input. In contrast, the initial data point is excluded from the training label – a visual representation of this is provided in Figure 10(b). This strategic arrangement entrusts the model to assimilate insights from neighboring segments and derive contextual understanding across the sequence.

Subsequently, the multi-head masked attention layer generates Q, K, and V vectors employing the training input data. These vectors then play a role in computing attention scores through Equation (13), producing the attention output depicted in Figure 10(c). These scores are used to predict subsequent timestep values. Training the GPT model involves the iterative refinement of neural network components responsible for generating appropriate Q, K, and V vectors. This process enables the model to learn and generate attention scores that contribute to accurate predictions of future values within the time series.

The input sequence of GPT-2 undergoes a twofold segmentation process. Firstly, the time steps are partitioned into windows, each encapsulating a localized context of consecutive timesteps. Following the temporal segmentation, the feature dimension is distributed across separate attention heads, facilitating parallelized processing throughout the sequence. This strategic fusion of windowing and attention head allocation endows the GPT architecture to effectively process and predict time-series data, capturing intricate sequential relationships comprehensively.

Figure 11 shows how the GPT-2 algorithm predicts sequential values. We only use the initial timesteps as input and the entire 50 timesteps as label data. The trained GPT-2 model iteratively predicts the next timesteps until it reaches the last (50th) step. Notably, GPT-2 is an autoregressive model[63,64] where each timestep is generated based on all the previous timesteps. For example, the *n*th timestep, t_n, is predicted based on t_{n-1}, t_{n-2}, ..., t_2, and t_1. This timestep prediction method has an advantage over LSTM, which predicts t_n based only on t_{n-1}. Furthermore, as described in our previous machine learning study, the "forget" gate in the LSTM algorithm can forget previous information, which could adversely affect the predictions.[55] For these reasons, GPT-2 possesses several advantages in accuracy and training efficiency since it simultaneously accepts all timesteps and calculates attention scores (via the dot product) to generate contextually appropriate predictions for the next time step.

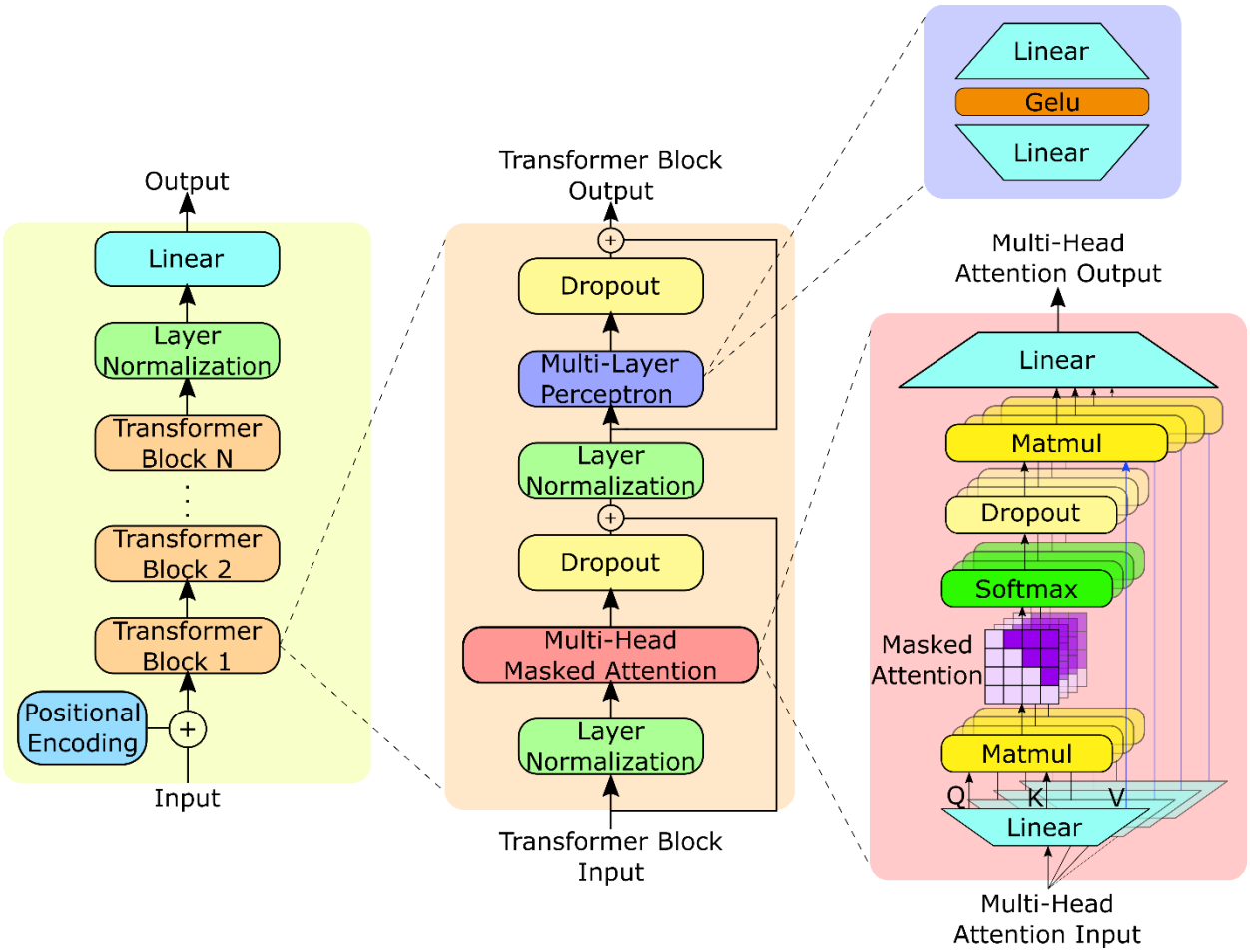


Figure 9. GPT-2 model architecture. The GPT-2 model contains *N* Transformer decoder blocks, as shown in the left panel. Each decoder block (center panel) includes a multi-head masked attention layer, a multi-layer perceptron layer, normalization, and dropout layers. The residual connection (branching line to the addition operator) allows the block to learn from the previous block's input. The multi-head masked attention layer (right panel) calculates attention scores using *Q*, *K*, and *V* vectors to capture sequential relationships in the input sequence.

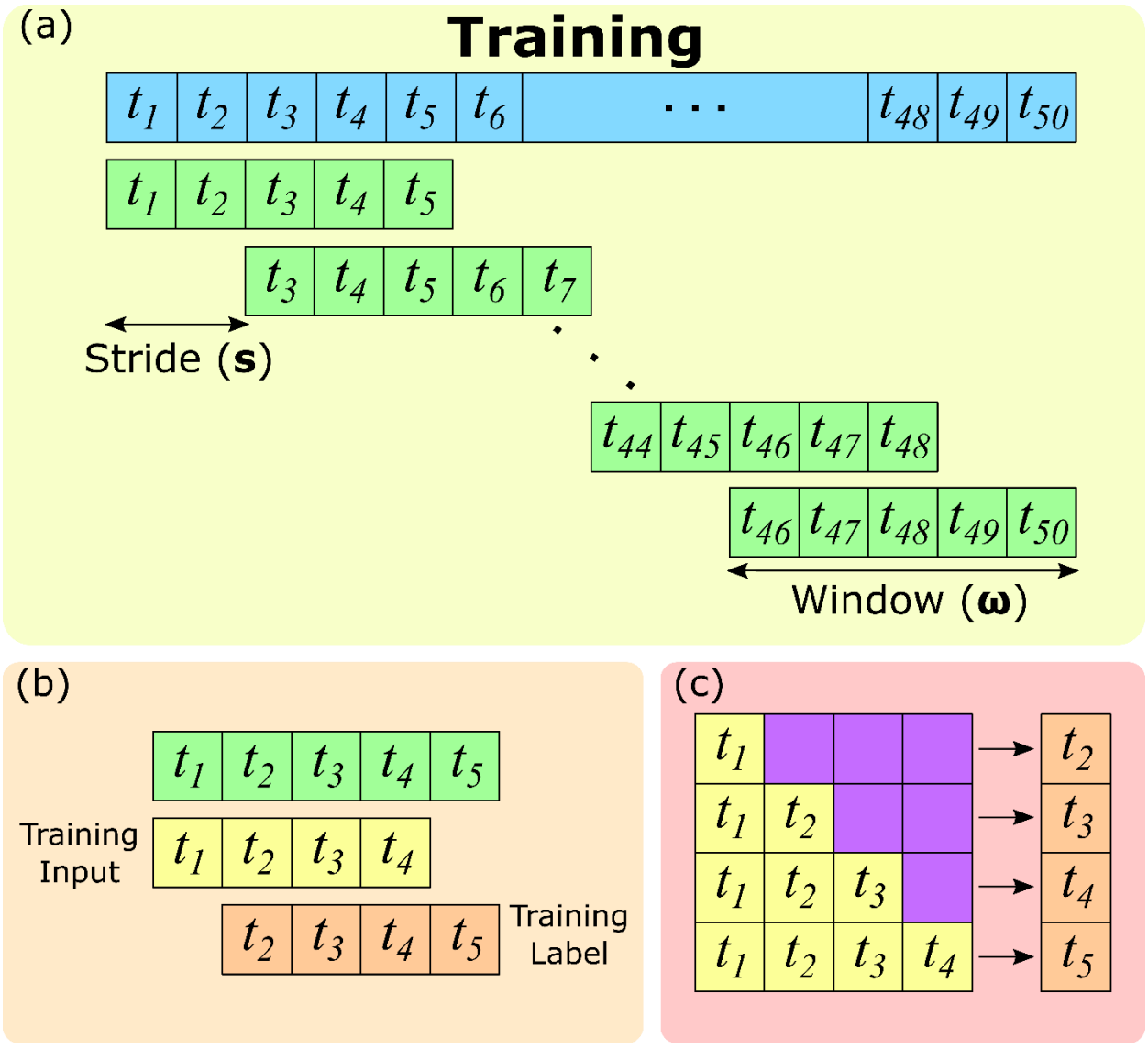


Figure 10. GPT-2 training process. Panel (a) shows the division of the 50 timesteps into segments using window and stride parameters (**ω** = 5 and **s** = 2, respectively). Panel (b) displays each segment's input and labels data points during model training. Panel (c) demonstrates the computation of the attention layer's output using Equation 1, which involves masking certain positions in the input sequence to zero out their attention scores (shown as purple boxes).

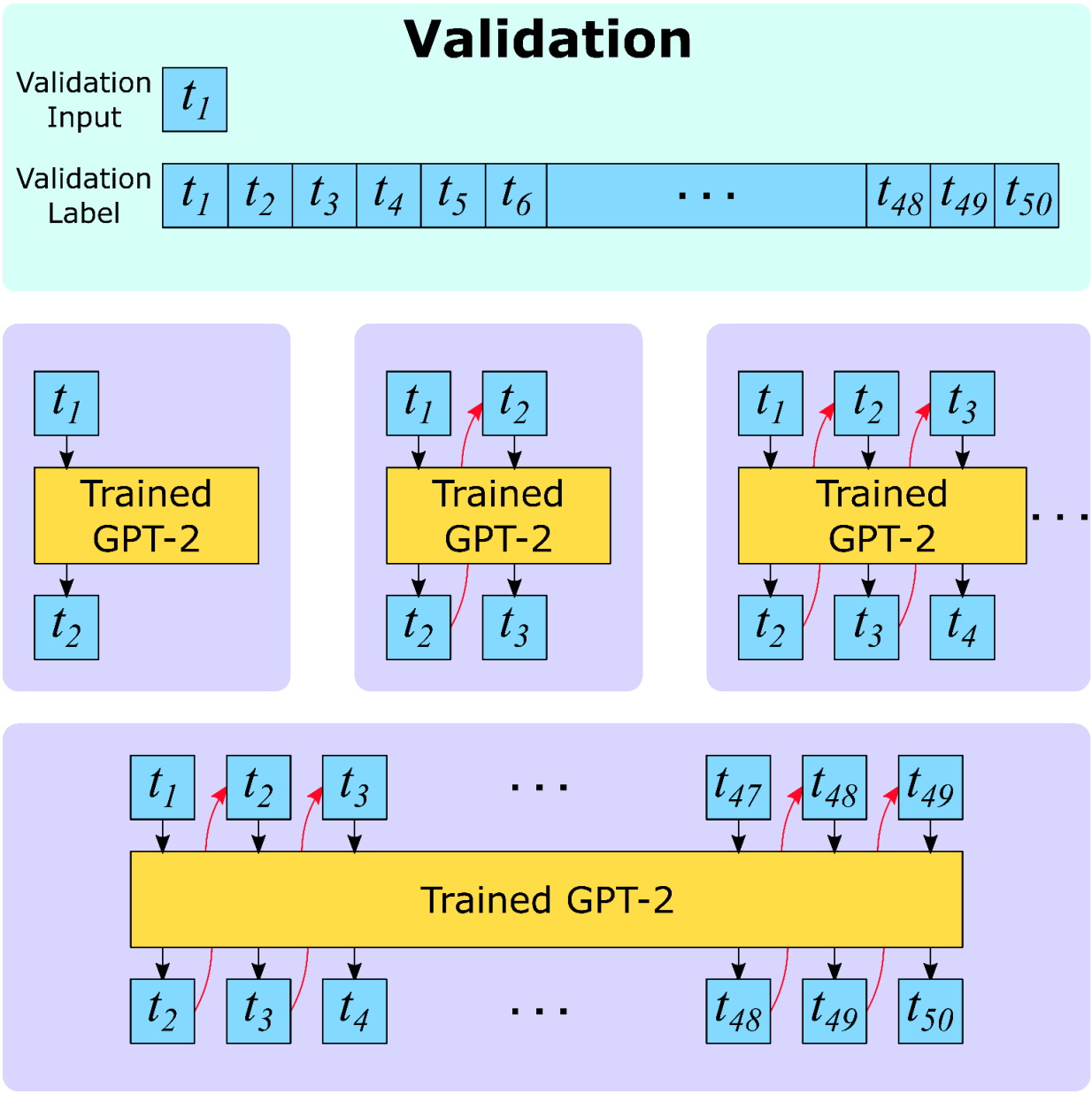


Figure 11. GPT-2 validation process. The trained GPT-2 model predicts all 50 timesteps based on the initial timestep, generating subsequent timesteps based on the previous ones until it reaches the last timestep of the labeled data.

2.2.2. CNN Architecture

The trajectories generated by the time-series models (GPT-2, BiLSTM, or LSTM) serve as input data for a 3D CNN model designed to predict the surface erosion rate. Utilizing the strengths of a 3D CNN (a type of advanced neural network) becomes crucial for our goals. This model is adept at recognizing patterns in 3D data, such as our time-series trajectories, making it better at predicting erosion. Enhancing its ability to understand complex spatial relationships contributes to its prediction accuracy.[65]

The 3D CNN utilizes specific filters known as kernels to analyze localized portions of input data and extract relevant information. These kernels, characterized by designated squared

dimensions, filter the data and emphasize significant features while discarding unnecessary details. The process includes a stride, indicating the kernel's step size as it moves across the data. Additionally, max pooling is applied to further condense and preserve key spatial insights. As the process unfolds, gathered details are progressively integrated, enhancing the network's understanding of spatial relationships within the data. Introducing supplementary layers helps streamline the computation and maintain vital spatial information. The 3D CNN identifies and captures intrinsic data features through an iterative approach.

Our CNN model's success in predicting erosion is due to its ability to recognize spatial patterns that greatly affect erosion in the boiler header. Specifically, CNN identifies areas where particles encounter the surface because of fluid movement, which matches places with higher erosion rates. 3D convolutional layers allow the model to utilize initial conditions and how particles are positioned, revealing important connections between these factors and erosion values. These abilities make the CNN model a good fit for predicting erosion in fluid flow systems.

2.2.3. Data Collection, Model Training, and Early Stopping

2.2.3.1. Data Collection

We obtained particle trajectories and erosion data by running simulations in ANSYS Fluent 19.2 using the geometry of an OP-650 boiler header.[31,32,66] The obtained erosion dataset from our CFD simulations captures erosion observations at the simulations' final time. The particle trajectory data extracted from the CFD simulations constitutes a time-series dataset spanning 50 timesteps with a 2.49 ms time interval. To investigate the effects of various initial conditions, we adjusted the particle size, main-inlet speed, main-inlet pressure, sub-inlet speed, and sub-inlet pressure. Our previous study[55] provides further details and background on the CFD equations, configurations, and initial condition settings, which we also used for this work. In short, ANSYS Fluent calculates erosion by integrating force balance equations[23] over the particle trajectories, which we expanded upon in two stages. First, we utilized a time-series machine learning architecture to forecast particle trajectories based on initial conditions. Subsequently, we leveraged a 3D convolutional machine learning model that processes the predicted trajectories to predict surface erosion profiles.

Our dataset encompasses 3125 samples, each comprising x, y, and z coordinates of 196 particles across 50 timesteps. The 196-particle order from the CFD data was shuffled to minimize the data order dependency. Accounting for fluid flow fluctuations, we augmented each particle's trajectory with five initial condition parameters for every timestep. This yielded a dataset with $3125 \times 50 \times 196 \times 8$ dimensions. The last two dimensions were merged to obtain a final trajectory dataset with a shape of $3125 \times 50 \times 1568$, signifying the number of samples, time steps, and features, respectively.

Furthermore, an erosion dataset (3125×38312) was crafted, characterizing erosion values across the surface mesh of the boiler header. We shuffled the 3125 samples and divided them into training, validation, and test sets. For this division, the test set comprised 10% of the overall data, while the remainder was split between the training and validation sets in an 8:2 ratio. To avoid any risk of overfitting, we applied K-fold cross-validation[46,47] with $k = 5$. This process yielded training, validation, and test sets containing 2250, 562, and 313 samples, respectively.

2.2.3.2. Model Training

The FLUID-GPT approach developed in this study is a hybrid machine-learning model to predict particle trajectories and surface erosion rates in the OP-650 boiler header. Specifically, we used the predicted trajectories from the time-series models (GPT-2, BiLSTM, or LSTM) as input data to a CNN model to predict the surface erosion rate. It is worth emphasizing that our FLUID-GPT approach can predict surface erosion rates *using only information from five initial conditions*: particle size, main-inlet speed, main-inlet pressure, sub-inlet speed, and sub-inlet pressure.

We fine-tuned the hyperparameters of three time-series models: GPT-2, LSTM, and BiLSTM, employing the CFD dataset. BiLSTM served as a reference to assess the sequential memory dependency and accuracy effects of LSTM.[67,68] Our GPT-2 model comprises 120,558,816 parameters, while LSTM and BiLSTM each entail 305,024,608 parameters. We adopted 4 LSTM and BiLSTM layers, observing that reduced layers led to inadequate prediction performance.

Hyperparameters for the GPT-2 model included the number of decoder blocks, attention heads, and window/stride dimensions. Our GPT-2 model employed an MLP with the Gaussian Error Linear Unit (GELU) activation function. We chose this particular configuration due to its superior performance over the Rectified Linear Unit (ReLU) in transformer models for NLP tasks. GELU effectively mitigates the vanishing gradient problem and exhibits a smoothness property, contributing to improved performance.[69,70]

We optimized learning rates utilizing the CyclicLR (CLR) scheduler,[71] which progressively reduces the periodic peak's magnitude as training advances. CLR has shown superior performance over alternative learning rate schedules, such as step and exponential decay, resulting in accelerated convergence and improved accuracy.[71,72] The Adam optimization method was used for each model training.

The CNN processes the predicted trajectory from either GPT-2 or BiLSTM, along with five initial parameters, to forecast surface erosion rates. Our method incorporates four layers of 3D convolution and max pooling, employing the GELU activation function. We optimize CNN performance through systematic enhancements involving kernel size (2, 3), stride (1, 2) variations in convolutional layers, and kernel sizes (2, 3) in max pooling. Additionally, we explore different filter combinations for each set of four convolution layers, denoted as (2-4-8-16), (4-8-16-32), (8-16-32-32), and (10-20-30-40). This comprehensive optimization approach ensures a thorough exploration of hyperparameters, contributing to the improved effectiveness of our CNN model. We employed this optimization strategy for two time-series models: FLUID-GPT (GPT-2+CNN) and BiLSTM+CNN. Performance evaluation encompasses key metrics, including each algorithm's MSE, R^2 score, and training time.

2.2.3.3. Early Stopping

While training our GPT-2 and LSTM models, we employed convergence criteria to prevent overfitting and optimize computational efficiency. Specifically, we utilized an early stopping method, where the training process was stopped when the MSE fell below a predetermined threshold for three consecutive epochs. We set the threshold value at 0.0065 based on the results of the CLR scheduler learning rate optimization procedure. We applied the same threshold value and counter limits to ensure that both models were trained using similar convergence criteria for a fair comparison of their respective performances.

2.3. Results and Discussion

2.3.1. GPT and BiLSTM Hyperparameter Optimization

2.3.1.1 Learning Rates and Schedulers

We optimized the learning rate for the GPT-2 architectures using the CLR scheduler, starting at a value of 1×10^{-3}. We varied the learning rate using an order of magnitude reduction strategy, where each new learning rate was reduced by a power of 10 to determine the optimal value for efficient and accurate convergence, as shown in Figure 12. Our simulations showed that a learning rate of 1×10^{-7} achieved the best performance, as shown in Figures 12(a) and (b). These results demonstrate the importance of optimizing the learning rate for optimal convergence in GPT-2 architectures.

The BiLSTM model was trained in a similar fashion, as shown in Figures 12(c) and (d). A learning rate of 1×10^{-4} showed the lowest MSE loss among our variations, with the most stable training and validation loss profiles. The LSTM model was also trained, and we also tested the linear and Cosine Annealing Warm Restarts schedulers.[73]

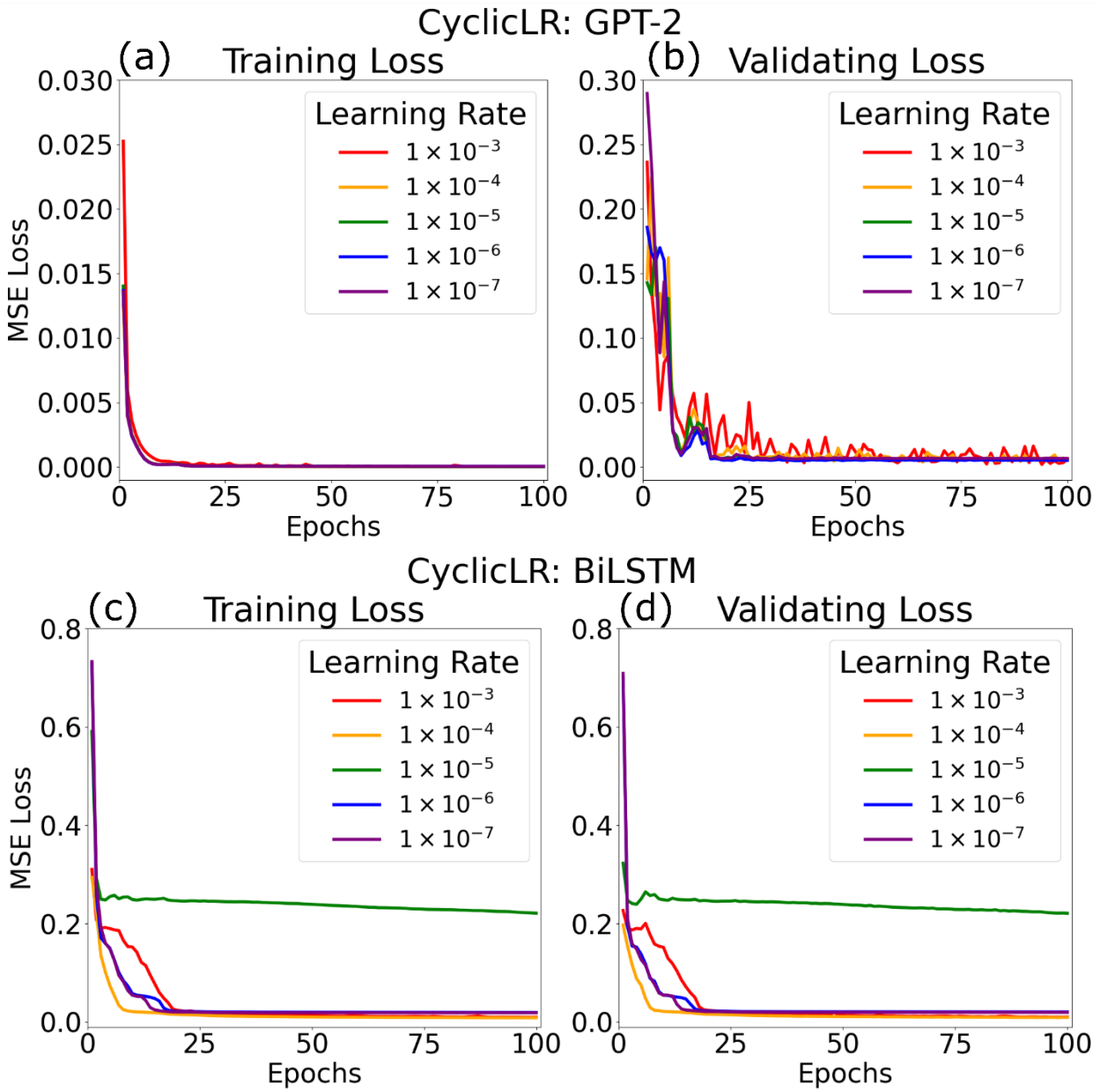


Figure 12. (a)-(b): Training and validation MSEs using the CLR scheduler for the GPT-2 architecture and (c)-(d) the BiLSTM model. The panels show the MSEs for different learning rates during training and validation.

2.3.1.2. Optimizing Window (ω) and Stride (s)

In the hyperparameter tuning process, we employed a grid search to determine the optimized values, with specific ranges, for each parameter: window (**ω** = 2, 4, 8, 16), stride (**s** = 2, 4, 8, 16), the number of decoder layers (2, 3, 4), and the number of attention heads (2, 4, 8, 16). Our results showed that 4 decoder block layers with 2 attention heads provided the most favorable outcome, striking a balance between the MSE and training duration. Figures 13(a) and (b) show the GPT-2 training duration and MSE loss for a range of **ω** and **s** pairs, respectively. Our observations unveiled a noteworthy trend: as the stride (**s**) decreased, the corresponding MSE loss

exhibited a reduction, indicating an enhancement in prediction accuracy, especially when windows overlapped. However, it is essential to note that a decrease in the stride (**s**) also increased training duration. This effect can be attributed to generating more window steps, expanding the input data volume due to larger temporal overlaps.

We, therefore, sought to find a balance between accuracy and efficiency by selecting an appropriate value for **s**. We tested various values of **ω** and **s** and found that the early stopping criteria were triggered at different epochs for each combination of **ω** and **s**. Specifically, the early stopping was activated for 2/2, 4/2, 8/2, 16/2, and 8/4 (**ω**/**s**) pairs, resulting in a much shorter training duration for these five cases. In addition, we noticed that a **ω**:**s** ratio of 1:1 showed higher MSEs than other settings, as the windows did not overlap, and the relationship between the windows could not be learned. As shown in Figure 13, there is an optimal **ω** for each **s** that gives the lowest MSE. After carefully considering these factors, we chose **ω** = 4 and **s** = 2 as the best combination of parameters, resulting in the best balance between MSE loss and training duration compared to other settings.

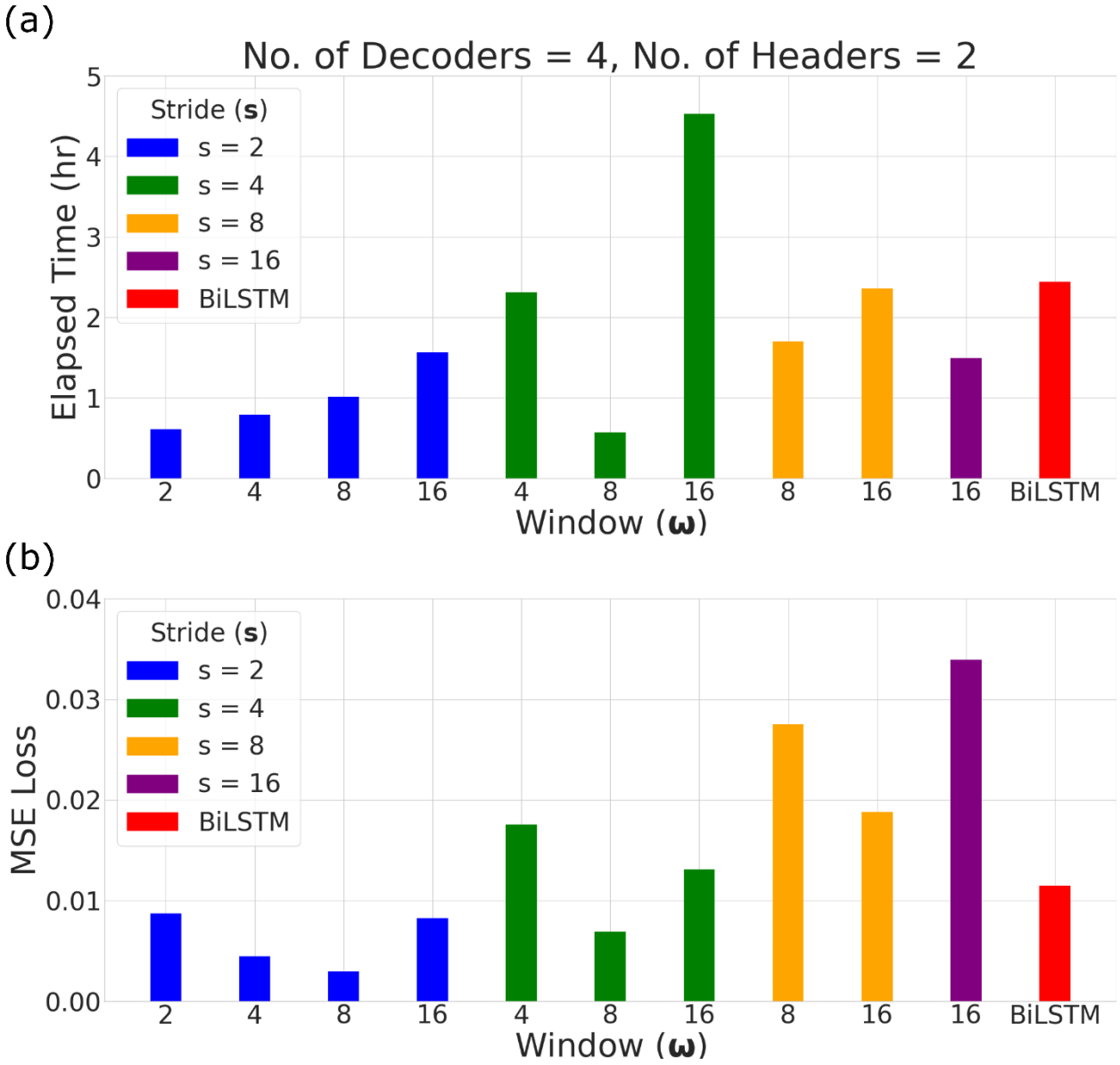


Figure 13. Window (**ω**) and stride (**s**) optimization for GPT-2 with a fixed number of decoder blocks and attention heads. Panel (a) displays the training duration, while panel (b) shows the validating MSE loss for different values of **ω** and **s** pairs. The colors indicate the value of **s** for each **ω**.

2.3.1.3. Number of Decoder Layers and Attention Heads

Our study extended to the influence of varying decoder block layers and attention heads. Training time accelerates as we reduce the number of decoders, albeit with a trade-off in increased MSE values. Figure 13 provides a general overview for configurations with 4 decoder blocks and 2 attention heads. Smaller strides (**s** = 2 or 4) yield heightened performance, particularly with aligned attention head and stride values. These attributes modulate information blending and segment overlap. Smaller strides excel at finer-grained pattern capture, while larger strides reveal broader trends. Larger attention head numbers augment complex data handling and overall

performance. The pivotal interplay between attention heads and stride is essential for orchestrating sequential information flow in input data, underpinning our analysis.

2.3.2. CNN Hyperparameter Optimization

We optimized the hyperparameters for two CNN models: FLUID-GPT (GPT-2+CNN) and BiLSTM+CNN. Figure 14 illustrates the optimization of our FLUID-GPT model, utilizing a (8-16-32-32) convolution filter combination. As observed in optimizing GPT-2, smaller stride yields improved MSE performance, albeit at the expense of longer training duration. This trade-off aligns with expectations, as a larger stride leads to larger steps between convolution operations, potentially affecting prediction accuracy.

However, a distinct relationship emerged for the kernel sizes in convolution and max pooling layers. Smaller convolution kernel sizes correlated with higher MSE values, while reduced max pooling kernel sizes were associated with lower MSEs. This phenomenon arises from the operational disparities between these layers. Smaller convolution kernels capture fewer features per operation, potentially resulting in information loss when critical components span a wider receptive field. Conversely, smaller pooling kernels keep more information by down-sampling less, which is favorable for tasks requiring precise localization and leads to diminished MSEs. By evaluating MSEs and training duration, we identified the optimal hyperparameter configuration: a convolution stride of 2, convolution kernel size of 3, pooling kernel size of 2, and an (8-16-32-32) convolution filter combination. Notably, this optimal configuration is also held for BiLSTM, reinforcing our adoption of the same CNN architecture for erosion prediction.

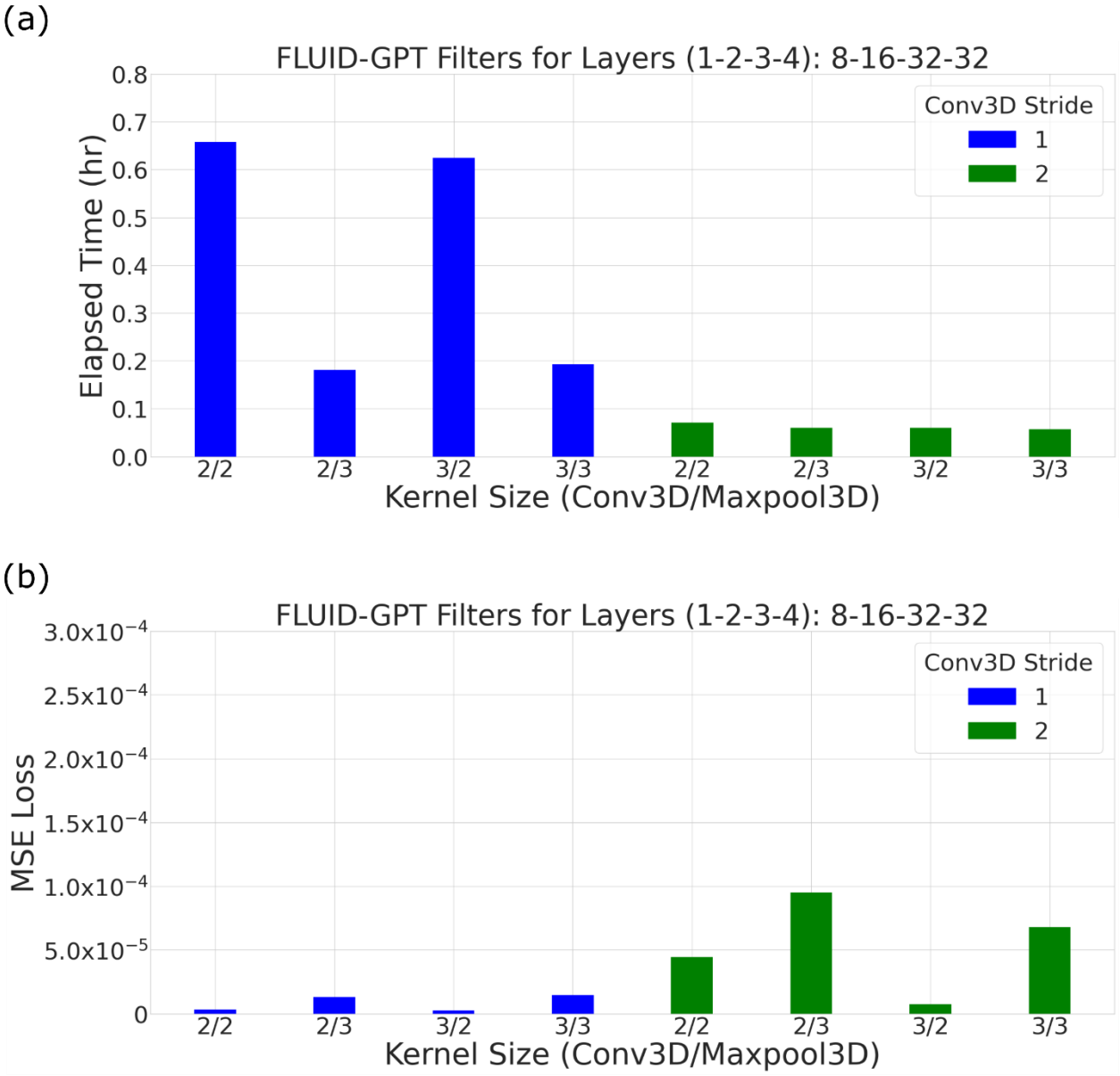


Figure 14. Optimizing CNN: Variations in stride and kernel sizes in the convolution layer and variations in kernel sizes in the max pooling layer with an (8-16-32-32) convolution filter combination. Panel (a) displays the training duration, while panel (b) shows the MSE loss validation for different hyperparameter values. The optimal hyperparameter configuration is a stride of 2, convolution kernel size of 3, pooling kernel size of 2, and an (8-16-32-32) convolution filter.

2.3.3. GPT-2 vs. BiLSTM Performance

After optimizing the GPT-2, BiLSTM, and their CNN models, we compared their prediction performance and training efficiency. Figures 15(a) and (b) show the training and validation loss of particle trajectory predictions from GPT-2 vs. BiLSTM and erosion predictions from FLUID-GPT vs. BiLSTM+CNN, respectively. Our early stopping criteria caused the GPT-2 training to stop after 18 epochs, while BiLSTM continued training up to 100 epochs. This preliminary stopping arises since GPT-2 divides the timesteps into segments with window and

stride sizes. It subsequently processes them with multiple attention heads, allowing the model to review previous sequence information during each new segment. In contrast, the BiLSTM approach learns consecutively without recalling the previous timesteps. Reviewing past information by GPT-2 leads to more efficient learning but a longer training duration per epoch than BiLSTM, which resembles human learning. With early stopping, GPT-2 achieved faster convergence of the validating loss than BiLSTM, resulting in more accurate and efficient predictions.

We also compared the individual CNN models from GPT-2 and BiLSTM. However, despite having an identical CNN architecture, the BiLSTM+CNN model exhibits lower accuracy. Our results indicate that errors associated with the previous model's predicted trajectories may worsen the subsequent CNN performance. Evaluation using a test dataset and K-fold cross-validation yielded average MSEs, R^2 scores, and training durations (see Table 1). GPT-2 errors were nearly half the average MSE of BiLSTM, with better training efficiency. The average MSE of FLUID-GPT was 0.35 times lower than BiLSTM+CNN. The training duration was similar between FLUID-GPT and BiLSTM+CNN due to the shared CNN model.

Model	**MSE**	**R^2 Score**	**Training Duration**	
			Total (hr)	Epoch Time (min)
GPT-2	0.0053 ± 0.0004	0.9807 ± 0.0017	0.7362 ± 0.0474	2.4540
BiLSTM	0.0115 ± 0.0019	0.9427 ± 0.0102	2.4420 ± 0.1333	1.4652
FLUID-GPT	$7.2223 \times 10^{-6} \pm 5.0736 \times 10^{-7}$	0.9899 ± 0.0080	0.0651 ± 0.0016	0.0250
BiLSTM+CNN	$2.0650 \times 10^{-5} \pm 4.4573 \times 10^{-6}$	0.9707 ± 0.0061	0.0617 ± 0.0011	0.0246

Table 1. Comparison of prediction accuracy and training efficiency of GPT-2, BiLSTM, FLUID-GPT, and BiLSTM+CNN for particle trajectory and erosion predictions.

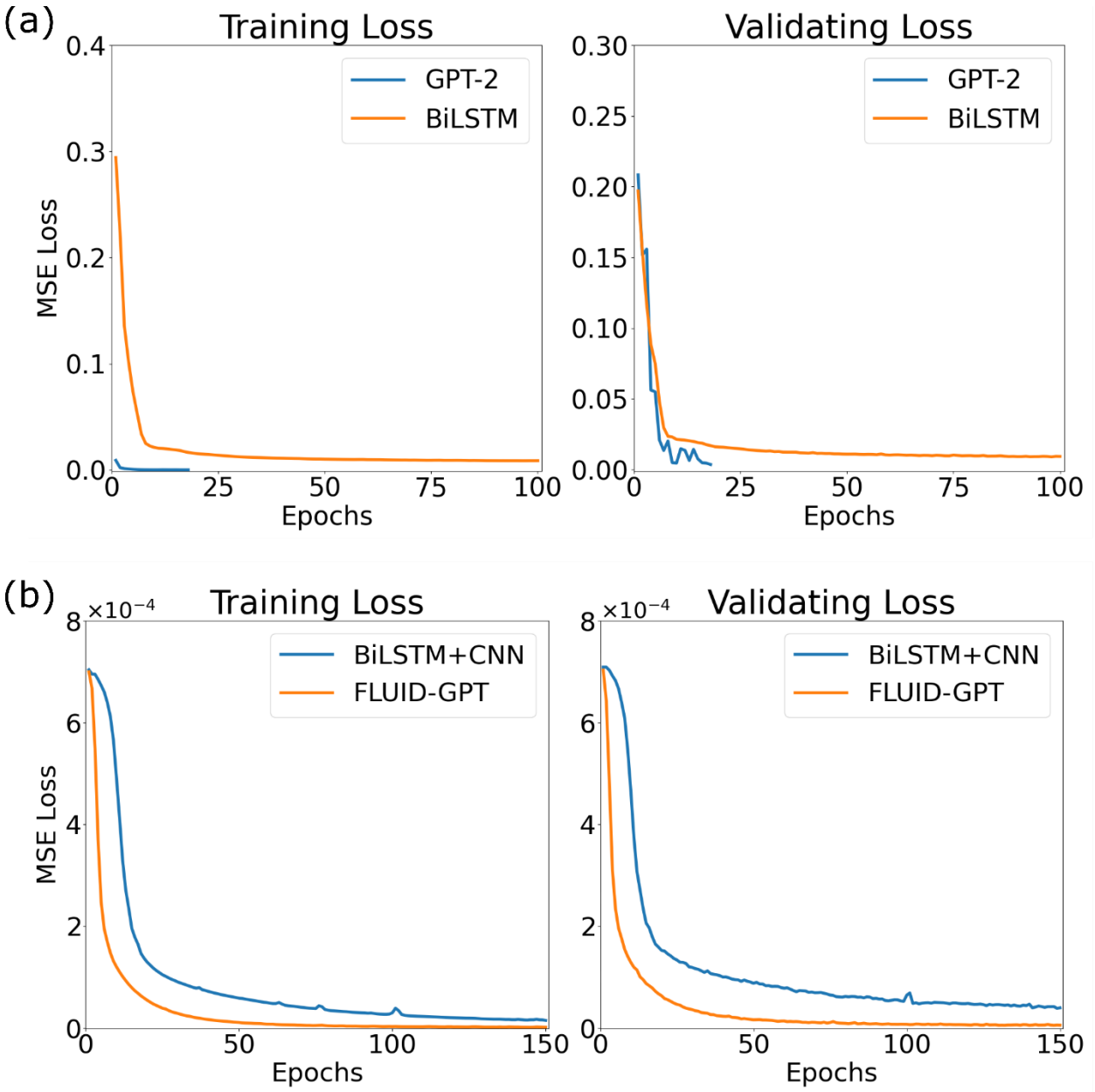


Figure 15. Training and validation loss comparison between (a) GPT-2 vs. BiLSTM, and (b) FLUID-GPT vs. BiLSTM+CNN, using optimized hyperparameters. The early stopping activates on GPT-2 training, converging much faster than BiLSTM. FLUID-GPT shows faster convergence, lower MSE, and a more stable learning profile than BiLSTM+CNN.

2.3.4. Performance in Predicting Trajectories and Erosion Rates

Figure 16 compares the predicted particle trajectories and surface erosion rates obtained from the brute-force CFD calculations and our optimized GPT-2 and BiLSTM models for two representative simulations (sample #50 and #70) from the test dataset. Figure 17 shows the results of our FLUID-GPT approach and BiLSTM+CNN, both of which use CNN on the output data for each respective time-series model.

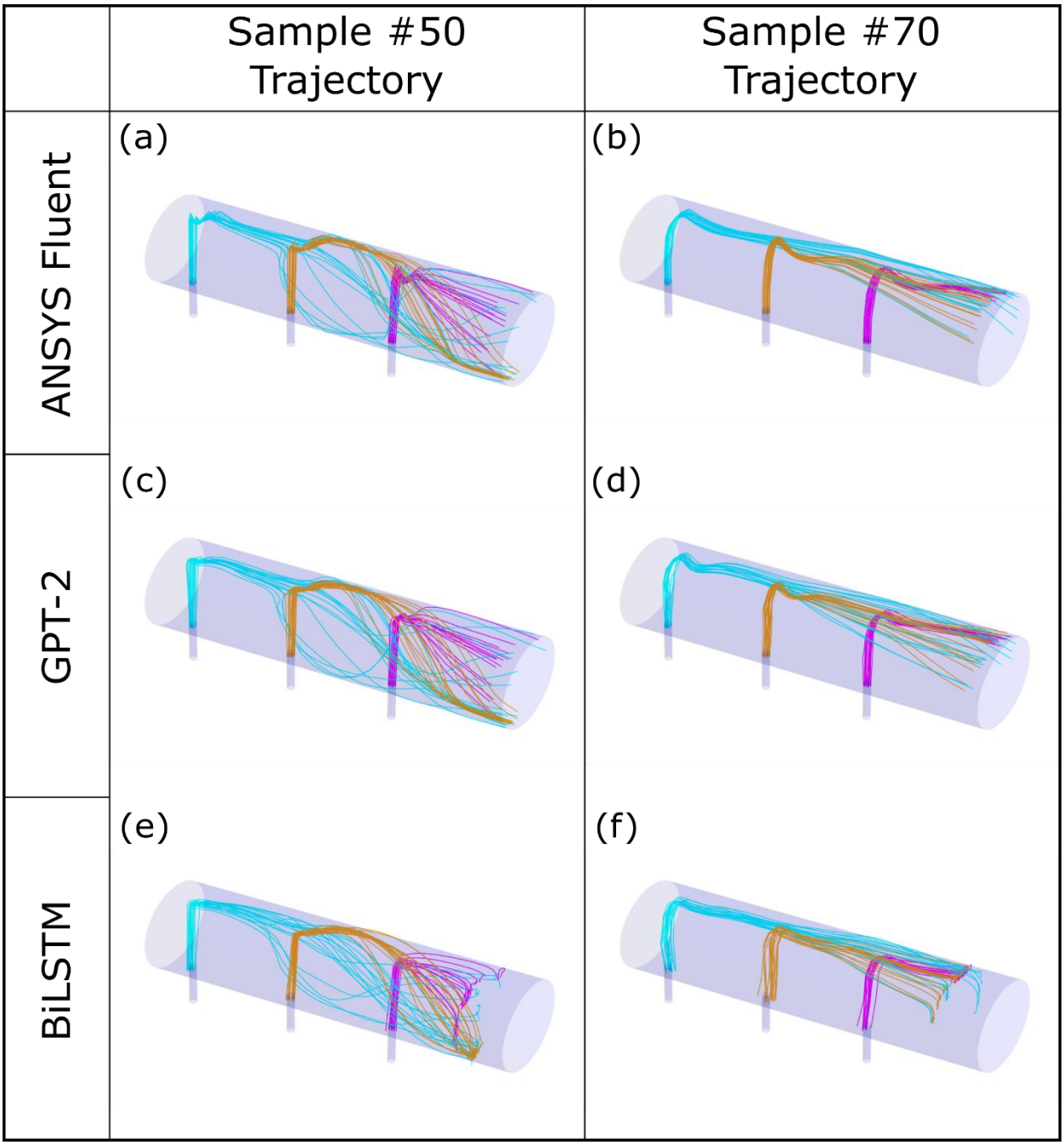


Figure 16. Comparison of particle trajectories predicted by GPT-2 and BiLSTM against ANSYS Fluent CFD simulations for representative simulations from the test dataset (samples #50 and #70).

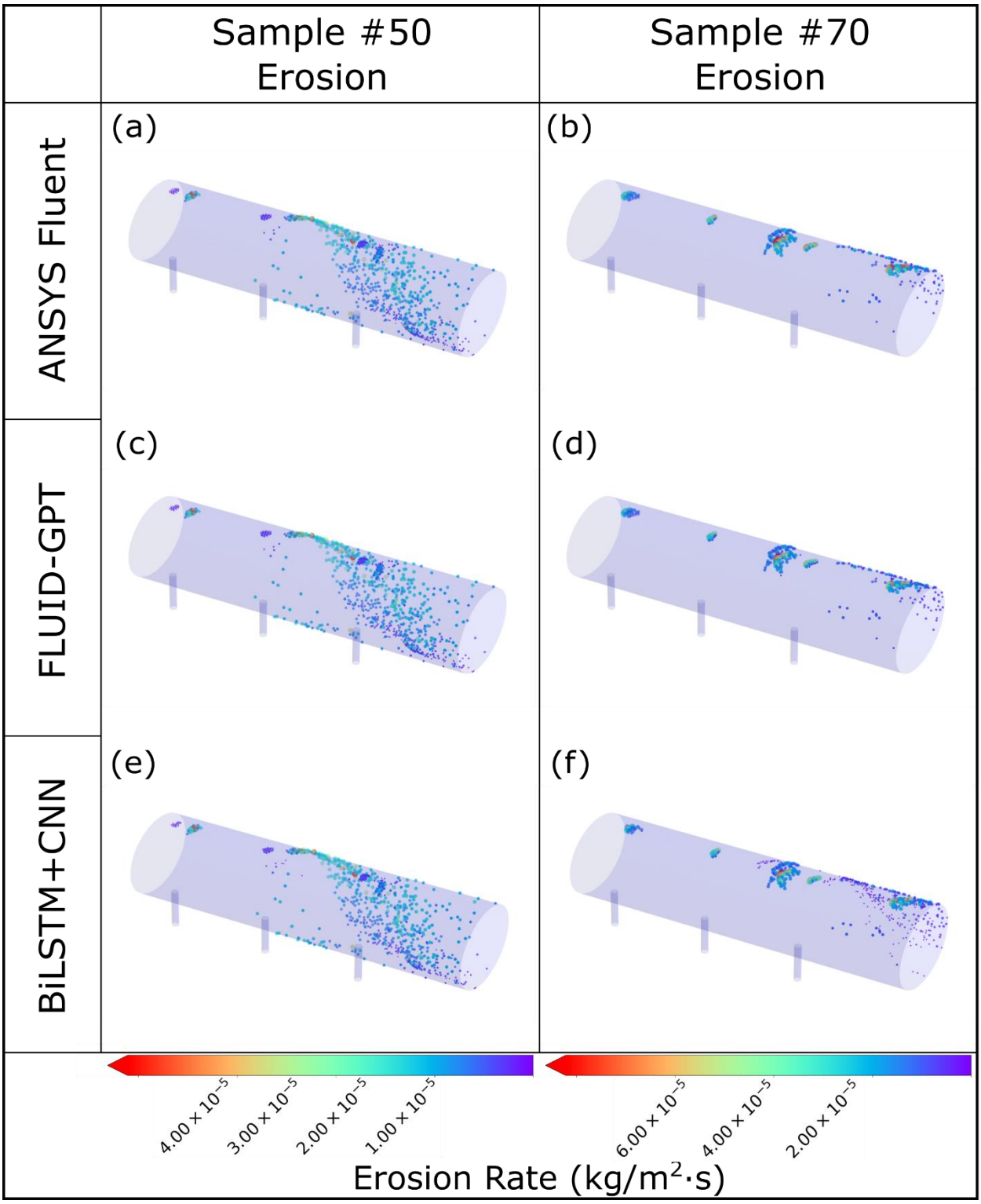


Figure 17. Comparison of surface erosion predicted by FLUID-GPT and BiLSTM+CNN against ANSYS Fluent CFD simulations for representative simulations from the test dataset (samples #50 and #70).

Figures 16(a)-(d) show that the GPT-2 model performs exceptionally well in predicting particle trajectories under turbulent flow conditions. In contrast, the trajectories predicted by BiLSTM exhibit noticeable deviations from the benchmark CFD data in Figures 16(e) and (f). These discrepancies can be attributed to the recurrent nature of the models. In the context of time-series data, GPT-2 is advantageous because it requires less contextual information and can capture dependencies among all timesteps. However, BiLSTM requires more details on the preceding and succeeding data points in the time series, which can be difficult to capture and may lead to lower accuracy. In addition, the forget gate in the BiLSTM algorithm can cause information loss from previous timesteps, further affecting prediction accuracy. Figures 17(a)-(d) highlight the good agreement between the FLUID-GPT and CFD benchmark calculations. Figures 17(e) and (f) present erosion predictions by the BiLSTM+CNN model, demonstrating satisfactory agreement in regions with significant erosion but displaying errors in areas with lower erosion than the CFD benchmarks.

In general, GPT-2 outperformed BiLSTM in both accuracy and efficiency, with a 54% decrease in average MSE (from 0.0115 to 0.0053) and a 70% reduction in training duration (from 2.45 hours to 0.73 hours). This advantage can be attributed to two algorithmic improvements in GPT-2: (1) information control, which segments the input data into windows and strides, enabling greater control over information mixing, and (2) autoregressive modeling, which considers all previous timesteps when predicting the next step. In contrast, BiLSTM processes input data sequentially and only predicts the subsequent step based on the previous step, resulting in lower accuracy and efficiency than GPT-2. Further deficiencies of BiLSTM can be attributed to the lack of information control and inability to consider all previous timesteps to predict the next step, which are algorithmic improvements implemented in GPT-2. For predicting erosion, the training duration of FLUID-GPT and BiLSTM+CNN is similar since the same CNN model is used. However, the average MSE of FLUID-GPT improves by 65% compared to BiLSTM+CNN (from 2.0650×10^{-5} to 7.2223×10^{-6}). We attribute this improvement to the superior performance of GPT-2 in predicting particle trajectories. The BiLSTM model's initial and final segments of the predicted trajectories deviate from the CFD benchmarks, which negatively impacts erosion prediction accuracy. In contrast, the GPT-2 model performs exceptionally well in predicting particle trajectories, even under turbulent flow conditions, which allows the model to capture erosion features more accurately.

2.4. Conclusion

In conclusion, we have developed FLUID-GPT, a new hybrid GPT-2+CNN machine-learning architecture for accurately predicting particle trajectories and erosion on an industrial-scale steam header geometry. Our FLUID-GPT approach can predict surface erosion rates using only information from five initial conditions: particle size, main-inlet speed, main-inlet pressure, sub-inlet speed, and sub-inlet pressure. We optimized our GPT-2 model through systematic variation of learning rates and schedulers, implementation of early stopping criteria, and comprehensive analysis of hyperparameters such as the number of decoder layers, attention heads, and window/stride sizes. Furthermore, we refined the CNN component by fine-tuning kernel and stride sizes within convolutional layers, optimizing the kernel size for max pooling, and carefully selecting filter combinations across various convolutional layer variations. The training time for FLUID-GPT was approximately 47 minutes on a single GPU, with an impressive R^2 score of 0.98 and 0.99 for predicting particle trajectories and erosion, respectively.

Our study shows that the FLUID-GPT hybrid ML approach outperformed traditional time-series models such as LSTM and BiLSTM for predicting particle trajectories and erosion prediction. Specifically, GPT-2 showed a 54% decrease in MSE and was 70% faster than BiLSTM. For predicting erosion, FLUID-GPT showed a 65% improvement in MSE compared to BiLSTM+CNN. Our results demonstrate that the FLUID-GPT hybrid ML approach significantly improves upon our previous LSTM study for predicting trajectories and surface erosion. In particular, the GPT-2 algorithm yields impressive accuracy and has a fast-training duration compared to LSTM. Overall, our work demonstrates that FLUID-GPT is an accurate and efficient approach for predicting complex trajectories and their subsequent erosion patterns. As such, this approach could have promising widespread applications in other research areas requiring time-series analyses or predictions of complex spatial properties arising from time-dependent phenomena.

A Github repository containing all the CFD and ML models used in this work can be accessed at https://github.com/SDY159/CFD_ML2.

References

(1) Raghu, D.; McKee, B.; Wu, J. B. C.; Sheriff, C. High Temperature Erosion Resistant Materials for Petroleum Refinery Equipment. In *CORROSION 2001*; OnePetro, 2001.

(2) Pepi, M.; Squillacioti, R.; Pfledderer, L.; Phelps, A. Solid Particle Erosion Testing of Helicopter Rotor Blade Materials. *Journal of failure analysis and prevention* **2012**, *12* (1), 96–108.

(3) Swadźba, L.; Formanek, B.; Gabriel, H. M.; Liberski, P.; Podolski, P. Erosion-and Corrosion-Resistant Coatings for Aircraft Compressor Blades. *Surf Coat Technol* **1993**, *62* (1–3), 486–492.

(4) Chawla, V.; Chawla, A.; Puri, D.; Prakash, S.; Gurbuxani, P. G.; Sidhu, B. S. Hot Corrosion & Erosion Problems in Coal Based Power Plants in India and Possible Solutions–a Review. *Journal of minerals and materials characterization and Engineering* **2011**, *10* (04), 367.

(5) Aramide, B. P.; Popoola, A. P. I.; Sadiku, E. R.; Aramide, F. O.; Jamiru, T.; Pityana, S. L. Wear-Resistant Metals and Composites. *Handbook of Nanomaterials and Nanocomposites for Energy and Environmental Applications* **2021**, 731–755.

(6) Parsi, M.; Najmi, K.; Najafifard, F.; Hassani, S.; McLaury, B. S.; Shirazi, S. A. A Comprehensive Review of Solid Particle Erosion Modeling for Oil and Gas Wells and Pipelines Applications. *J Nat Gas Sci Eng* **2014**, *21*, 850–873.

(7) Zhang, Y.; McLaury, B. S.; Shirazi, S. A. Improvements of Particle Near-Wall Velocity and Erosion Predictions Using a Commercial CFD Code. *J Fluids Eng* **2009**, *131* (3).

(8) Arabnejad, H.; Mansouri, A.; Shirazi, S. A.; McLaury, B. S. Development of Mechanistic Erosion Equation for Solid Particles. *Wear* **2015**, *332*, 1044–1050.

(9) Tran, A.; Wang, Y.; Furlan, J.; Pagalthivarthi, K.; Cutright, A.; Garman, M.; Visintainer, R. *WearGP: A Machine Learning Wear Prediction Framework for Slurry Pump Impellers and Casings under Different Operating Conditions.*; Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2019.

(10) Tran, A.; Furlan, J. M.; Pagalthivarthi, K. v; Visintainer, R. J.; Wildey, T.; Wang, Y. WearGP: A Computationally Efficient Machine Learning Framework for Local Erosive Wear Predictions via Nodal Gaussian Processes. *Wear* **2019**, *422*, 9–26.

(11) Bahrainian, S. S.; Bakhshesh, M.; Hajidavalloo, E.; Parsi, M. A Novel Approach for Solid Particle Erosion Prediction Based on Gaussian Process Regression. *Wear* **2021**, *466*, 203549.

(12) Bakhshesh, M.; Bahrainian, S. S.; Hajidavalloo, E.; Parsi, M. Developing a Dimensionless Number for Solid Particle Erosion with Gas-Liquid-Solid Flow in Standard Elbows. *Wear* **2021**, *478*, 203769.

(13) Dai, W.; Mohammadi, S.; Cremaschi, S. A Hybrid Modeling Framework Using Dimensional Analysis for Erosion Predictions. *Comput Chem Eng* **2022**, *156*, 107577.

(14) Zahedi, P.; Parvandeh, S.; Asgharpour, A.; McLaury, B. S.; Shirazi, S. A.; McKinney, B. A. Random Forest Regression Prediction of Solid Particle Erosion in Elbows. *Powder Technol* **2018**, *338*, 983–992.

(15) Pandya, D. A.; Dennis, B. H.; Russell, R. D. A Computational Fluid Dynamics Based Artificial Neural Network Model to Predict Solid Particle Erosion. *Wear* **2017**, *378*, 198–210.

(16) Miyanawala, T. P.; Jaiman, R. K. An Efficient Deep Learning Technique for the Navier-Stokes Equations: Application to Unsteady Wake Flow Dynamics. *arXiv preprint arXiv:1710.09099* **2017**.
(17) Durst, F.; Miloievic, D.; Schönung, B. Eulerian and Lagrangian Predictions of Particulate Two-Phase Flows: A Numerical Study. *Appl Math Model* **1984**, *8* (2), 101–115.
(18) Peng, Z.; Doroodchi, E.; Moghtaderi, B. Heat Transfer Modelling in Discrete Element Method (DEM)-Based Simulations of Thermal Processes: Theory and Model Development. *Prog Energy Combust Sci* **2020**, *79*, 100847.
(19) Stone, L.; Hastie, D.; Zigan, S. Using a Coupled CFD–DPM Approach to Predict Particle Settling in a Horizontal Air Stream. *Advanced Powder Technology* **2019**, *30* (4), 869–878.
(20) Pico, P.; Ratkovich, N.; Muñoz, F.; Dufaud, O. CFD-DPM and Experimental Study of the Dynamics of Wheat Starch Powder/Pyrolysis Gases Hybrid Mixtures in the 20-L Sphere. *Powder Technol* **2020**, *372*, 638–658.
(21) Greifzu, F.; Kratzsch, C.; Forgber, T.; Lindner, F.; Schwarze, R. Assessment of Particle-Tracking Models for Dispersed Particle-Laden Flows Implemented in OpenFOAM and ANSYS FLUENT. *Engineering Applications of Computational Fluid Mechanics* **2016**, *10* (1), 30–43.
(22) Andersson, B.; Andersson, R.; Håkansson, L.; Mortensen, M.; Sudiyo, R.; van Wachem, B. *Computational Fluid Dynamics for Engineers*; Cambridge university press, 2011.
(23) Horwitz, J. A. K.; Mani, A. Accurate Calculation of Stokes Drag for Point–Particle Tracking in Two-Way Coupled Flows. *J Comput Phys* **2016**, *318*, 85–109.
(24) Veritas, D. N. Recommended Practice RP O501 Erosive Wear in Piping Systems. *DNV Recommended Practice* **2007**, *4*.
(25) Qu, Z.; Otshwe, J. N. 2D Materials as Protective Coating against Low and Middle Temperature (100° C-300° C) Corrosion-Erosion in Waste to Energy Plant: Case of Graphene. *Graphene* **2021**, *10* (2), 13–39.
(26) Dhar, S.; Krajac, T.; Ciampini, D.; Papini, M. Erosion Mechanisms Due to Impact of Single Angular Particles. *Wear* **2005**, *258* (1–4), 567–579.
(27) Finnie, I.; McFadden, D. H. On the Velocity Dependence of the Erosion of Ductile Metals by Solid Particles at Low Angles of Incidence. *Wear* **1978**, *48* (1), 181–190.
(28) Oka, Y. I.; Okamura, K.; Yoshida, T. Practical Estimation of Erosion Damage Caused by Solid Particle Impact: Part 1: Effects of Impact Parameters on a Predictive Equation. *Wear* **2005**, *259* (1–6), 95–101.
(29) Finnie, I. Some Observations on the Erosion of Ductile Metals. *wear* **1972**, *19* (1), 81–90.
(30) Huang, C.; Chiovelli, S.; Minev, P.; Luo, J.; Nandakumar, K. A Comprehensive Phenomenological Model for Erosion of Materials in Jet Flow. *Powder Technol* **2008**, *187* (3), 273–279.
(31) Taler, J.; Węglowski, B.; Taler, D.; Sobota, T.; Dzierwa, P.; Trojan, M.; Madejski, P.; Pilarczyk, M. Determination of Start-up Curves for a Boiler with Natural Circulation Based on the Analysis of Stress Distribution in Critical Pressure Components. *Energy* **2015**, *92*, 153–159.
(32) Modliński, N.; Szczepanek, K.; Nabagło, D.; Madejski, P.; Modliński, Z. Mathematical Procedure for Predicting Tube Metal Temperature in the Second Stage Reheater of the Operating Flexibly Steam Boiler. *Appl Therm Eng* **2019**, *146*, 854–865.

(33) Modliński, N.; Madejski, P.; Janda, T.; Szczepanek, K.; Kordylewski, W. A Validation of Computational Fluid Dynamics Temperature Distribution Prediction in a Pulverized Coal Boiler with Acoustic Temperature Measurement. *Energy* **2015**, *92*, 77–86.
(34) Pilarczyk, M.; Węglowski, B. Analiza Cieplno-Wytrzymałościowa Rozruchu Kotła Parowego Na Przykładzie Kotła OP-650. *Zeszyty Naukowe Politechniki Rzeszowskiej* **2014**, *290*, 67–78.
(35) Nabagło, D.; Madejski, P. Combustion Process Analysis in Boiler OP-650k Based on Acoustic Gas Temperature Measuring System. In *Proceedings of 3rd International Conference on Contemporary Problems of Thermal Engineering CPOTE*; 2012; pp 18–20.
(36) Li, J.; Jankowski, R.; Kotecki, M.; Yang, W.; Szewczyk, D.; Brzdekiewicz, A.; Blasiak, W. CFD Approach for Unburned Carbon Reduction in Pulverized Coal Boilers. *Energy & fuels* **2012**, *26* (2), 926–937.
(37) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput* **1997**, *9* (8), 1735–1780.
(38) Mohan, A.; Daniel, D.; Chertkov, M.; Livescu, D. Compressed Convolutional LSTM: An Efficient Deep Learning Framework to Model High Fidelity 3D Turbulence. *arXiv preprint arXiv:1903.00033* **2019**.
(39) Zou, L.; Zheng, J.; Miao, C.; Mckeown, M. J.; Wang, Z. J. 3D CNN Based Automatic Diagnosis of Attention Deficit Hyperactivity Disorder Using Functional and Structural MRI. *IEEE Access* **2017**, *5*, 23626–23636.
(40) Ker, J.; Singh, S. P.; Bai, Y.; Rao, J.; Lim, T.; Wang, L. Image Thresholding Improves 3-Dimensional Convolutional Neural Network Diagnosis of Different Acute Brain Hemorrhages on Computed Tomography Scans. *Sensors* **2019**, *19* (9), 2167.
(41) Rao, C.; Liu, Y. Three-Dimensional Convolutional Neural Network (3D-CNN) for Heterogeneous Material Homogenization. *Comput Mater Sci* **2020**, *184*, 109850.
(42) Lu, X.; Duan, X.; Mao, X.; Li, Y.; Zhang, X. Feature Extraction and Fusion Using Deep Convolutional Neural Networks for Face Detection. *Math Probl Eng* **2017**, *2017*.
(43) Li, Z.; Wang, S.; Fan, R.; Cao, G.; Zhang, Y.; Guo, T. Teeth Category Classification via Seven-layer Deep Convolutional Neural Network with Max Pooling and Global Average Pooling. *Int J Imaging Syst Technol* **2019**, *29* (4), 577–583.
(44) Glantz, S.; Slinker, B. *Primer of Applied Regression & Analysis of Variance, Ed*; McGraw-Hill, Inc., New York, 2001.
(45) Steel, R. G. D.; Torrie, J. H. Principles and Procedures of Statistics. *Principles and procedures of statistics.* **1960**.
(46) Allen, D. M. The Relationship between Variable Selection and Data Agumentation and a Method for Prediction. *technometrics* **1974**, *16* (1), 125–127.
(47) Wojtas, M.; Chen, K. Feature Importance Ranking for Deep Learning. *arXiv preprint arXiv:2010.08973* **2020**.
(48) Vijiapurapu, S.; Cui, J.; Munukutla, S. CFD Application for Coal/Air Balancing in Power Plants. *Appl Math Model* **2006**, *30* (9), 854–866.
(49) Gandhi, M. B.; Vuthaluru, R.; Vuthaluru, H.; French, D.; Shah, K. CFD Based Prediction of Erosion Rate in Large Scale Wall-Fired Boiler. *Appl Therm Eng* **2012**, *42*, 90–100.
(50) Calzolari, G.; Liu, W. Deep Learning to Replace, Improve, or Aid CFD Analysis in Built Environment Applications: A Review. *Build Environ* **2021**, *206*, 108315.
(51) Tao, J.; Sun, G. Application of Deep Learning Based Multi-Fidelity Surrogate Model to Robust Aerodynamic Design Optimization. *Aerosp Sci Technol* **2019**, *92*, 722–737.

(52) Morozova, N.; Trias, F. X.; Capdevila, R.; Schillaci, E.; Oliva, A. A CFD-Based Surrogate Model for Predicting Flow Parameters in a Ventilated Room Using Sensor Readings. *Energy Build* **2022**, *266*, 112146.
(53) Du, P.; Zhu, X.; Wang, J.-X. Deep Learning-Based Surrogate Model for Three-Dimensional Patient-Specific Computational Fluid Dynamics. *Physics of Fluids* **2022**, *34* (8), 081906.
(54) Tran, A.; Wang, Y.; Furlan, J.; Pagalthivarthi, K. V; Garman, M.; Cutright, A.; Visintainer, R. WearGP: A UQ/ML Wear Prediction Framework for Slurry Pump Impellers and Casings. In *Fluids Engineering Division Summer Meeting*; American Society of Mechanical Engineers, 2020; Vol. 83723, p V002T04A008.
(55) Yang, S. D.; Ali, Z. A.; Kwon, H.; Wong, B. M. Predicting Complex Erosion Profiles in Steam Distribution Headers with Convolutional and Recurrent Neural Networks. *Ind Eng Chem Res* **2022**, *61* (24), 8520–8529.
(56) Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. **2018**.
(57) Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A. Language Models Are Few-Shot Learners. *Adv Neural Inf Process Syst* **2020**, *33*, 1877–1901.
(58) Geneva, N.; Zabaras, N. Transformers for Modeling Physical Systems. *Neural Networks* **2022**, *146*, 272–289.
(59) Shalova, A.; Oseledets, I. Deep Representation Learning for Dynamical Systems Modeling. *arXiv preprint arXiv:2002.05111* **2020**.
(60) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Adv Neural Inf Process Syst* **2017**, *30*.
(61) Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models Are Unsupervised Multitask Learners. *OpenAI blog* **2019**, *1* (8), 9.
(62) Xu, J.; Sun, X.; Zhang, Z.; Zhao, G.; Lin, J. Understanding and Improving Layer Normalization. *Adv Neural Inf Process Syst* **2019**, *32*.
(63) Li, C.; Zhang, M.; He, Y. Curriculum Learning: A Regularization Method for Efficient and Stable Billion-Scale Gpt Model Pre-Training. *arXiv preprint arXiv:2108.06084* **2021**.
(64) Meng, K.; Bau, D.; Andonian, A.; Belinkov, Y. Locating and Editing Factual Associations in Gpt. *Adv Neural Inf Process Syst* **2022**, *35*, 17359–17372.
(65) Maturana, D.; Scherer, S. Voxnet: A 3d Convolutional Neural Network for Real-Time Object Recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*; IEEE, 2015; pp 922–928.
(66) Modliński, N.; Madejski, P.; Janda, T.; Szczepanek, K.; Kordylewski, W. A Validation of Computational Fluid Dynamics Temperature Distribution Prediction in a Pulverized Coal Boiler with Acoustic Temperature Measurement. *Energy* **2015**, *92*, 77–86.
(67) Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv preprint arXiv:1508.01991* **2015**.
(68) Schuster, M.; Paliwal, K. K. Bidirectional Recurrent Neural Networks. *IEEE transactions on Signal Processing* **1997**, *45* (11), 2673–2681.
(69) Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (Gelus). *arXiv preprint arXiv:1606.08415* **2016**.
(70) Hendrycks, D.; Gimpel, K. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR, abs/1606.08415* **2016**, *3*.

(71) Smith, L. N. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*; IEEE, 2017; pp 464–472.
(72) Lee, H.; Lee, G.; Kim, J.; Cho, S.; Kim, D.; Yoo, D. Improving Multi-Fidelity Optimization with a Recurring Learning Rate for Hyperparameter Tuning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*; 2023; pp 2309–2318.
(73) Loshchilov, I.; Hutter, F. Sgdr: Stochastic Gradient Descent with Warm Restarts. *arXiv preprint arXiv:1608.03983* **2016**.
(74) Yang, S. D.; Ali, Z. A.; Wong, B. M. FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion. *Ind Eng Chem Res* **2023**, DOI: 10.1021/acs.iecr.3c01639.