

Entropy-driven Optimal Sub-sampling of Fluid Dynamics for Developing Machine-learned Surrogates

Wesley Brewer¹, Daniel Martinez², Muralikrishnan Gopalakrishnan Meena¹, Katarzyna Borowiec¹, Aditya Kashi¹, Siyan Liu¹, Christopher Pilmaier³, Greg Burgreen³, Shanti Bhushan³

¹Oak Ridge National Laboratory, ²Science & Technology Corporation, ³Mississippi State University

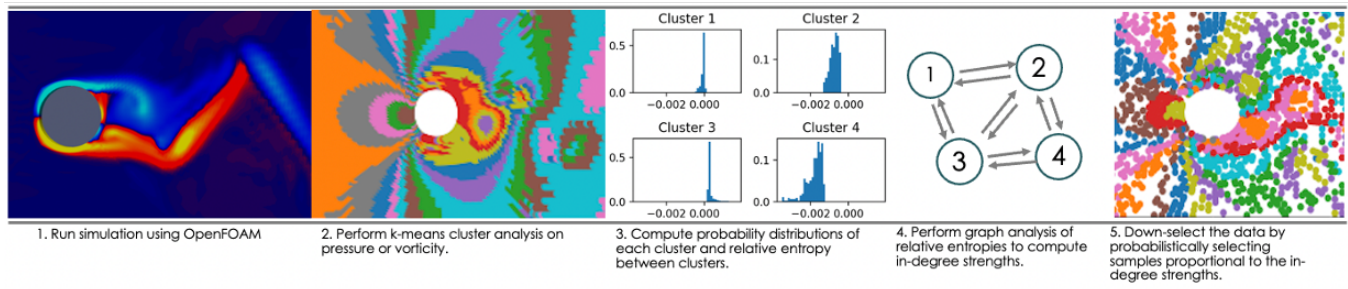


Figure 1: MaxEnt method for optimal sub-sampling applied to flow over a cylinder.

ABSTRACT

Optimal sub-sampling of large datasets from fluid dynamics simulations is essential for training reduced-order machine learned models. A method using Shannon entropy was developed to weight flow features according to their level of information content, such that the most informative features can be extracted and used for training a surrogate model. The method is demonstrated in the canonical flow over a cylinder problem simulated with OpenFOAM. Both time-independent predictions and temporal forecasting were investigated as well as two types of prediction targets: local per-grid-point predictions and global per-time-step predictions. When tested on training a surrogate model, results indicate that our entropy-based sampling method often outperforms random sampling and yields reproducible results in less iterations. Initial findings also highlight the method's application to nekRS LES simulation data of magnetohydrodynamic flows in nuclear fusion reactors.

CCS CONCEPTS

• Applied computing; • Physical sciences and engineering; • engineering;

KEYWORDS

maximum entropy, clustering, sampling, surrogate, reduced-order

ACM Reference Format:

Wesley Brewer¹, Daniel Martinez², Muralikrishnan Gopalakrishnan Meena¹, Katarzyna Borowiec¹, Aditya Kashi¹, Siyan Liu¹, Christopher Pilmaier³, Greg Burgreen³, Shanti Bhushan³, ¹Oak Ridge National Laboratory, ²Science & Technology Corporation, ³Mississippi State University. 2023. Entropy-driven Optimal Sub-sampling of Fluid Dynamics for Developing Machine-learned Surrogates. In *Proceedings of The 4th Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S'23)*. ACM, New York, NY, USA, 7 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

There are a class of AI workflows which involve running simulations to develop a machine-learned surrogate model: Such workflows may be used for developing digital twins to run in near real-time, e.g., [12], or may be used to develop reduced-order models to significantly speed up more computationally expensive workflows, e.g., [20]. In either case, the workflows involve running numerous simulations, then curating a training dataset by extracting data from the simulations, using that data to train the neural network, performing neural architecture search (NAS) to optimize its architecture, and finally deploying the surrogate model for inference on HPC, as shown in Fig. 2. For each stage of the workflow, a number of key decisions need to be made by the researcher.

In running simulations, how many simulations need to be run? What kinds of parametric variations need to be explored? What is the optimal subset of data that needs to be extracted from the simulation in order to train the neural network (this subject of this paper)? What features are most informative for training the network? How can we ensure an equal representation of the various features in the dataset? Should we non-dimensionalize the features, and if so, what kind of method should we use (e.g., standard scaler, minmax, Yeo-Johnson, etc.)? What type of neural network architecture should be used, e.g., a convolutional neural network, a recurrent neural network, etc.? How many layers should the network have, and how many units or filters per layer? What are the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

AI4S'23, November 13, 2023, Denver, CO

© 2023 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00

<https://doi.org/XXXXXXX.XXXXXXX>

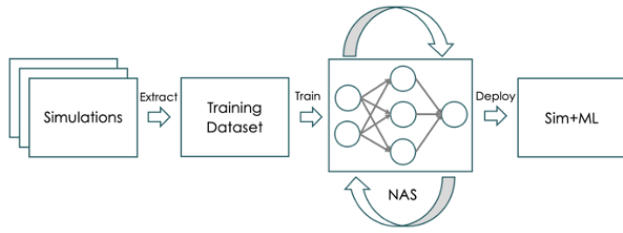


Figure 2: Workflow for training and deploying an ML surrogate.

optimal activation functions to use? How long should we train the network? What measures can we take to prevent overfitting the network to the training data and, as a result, ensure the model’s ability to generalize well? What is the optimal way to deploy the model [6]? How can we ensure that the unseen test data is not out-of-distribution (OOD) [17], and was sufficiently covered by the training data?

The highlighted queries underscore the intricate challenges associated with the development and deployment of machine learning models, particularly in the realm of engineering applications. Such workflows naturally raise the question of how these methods can be automated. This is where Automated Machine Learning (AutoML) comes into play. While there has been considerable research in AutoML for optimizing model architectures, as evidenced by the plethora of frameworks for hyperparameter optimization (e.g., KerasTuner [24], DeepHyper [2], MENDDL [32], etc.), analogous methods for generating optimal training datasets have been conspicuously absent.

The general thought in the AI community seems to universally be that more data produces better machine-learned models. However, this notion is increasingly being challenged [22], especially in the case of highly skewed datasets [5]; in these cases, the information of interest is significantly under-represented, which can make developing an accurate surrogate model extremely challenging. Fluid dynamics datasets, such as turbulence, can be massive in size, e.g., a data snapshot for a single file for a single variable of a DNS simulation on leadership class computing facilities can be on the order of ten terabytes [8]. Large datasets like these necessitate the use of intelligent sampling tools.

In this work, we develop an approach based on the principle of maximum entropy for optimal curation of training datasets, with the aim to produce highly accurate models; ‘curation’ refers specifically to sub-sampling simulation data. It has long been known since Jaynes seminal work on information theory and statistical mechanics [11], that the probability distribution which best represents the current state of knowledge about a system is the one which exhibits maximum entropy. If there is a way to incorporate this principle into the process of curating datasets, at least in theory, one could automatically design an optimal training dataset. To this end, we provide the following contributions:

- (1) Foundational work of representing fluid dynamics in the context of information theory, such that neural networks can better understand the data – i.e., developing a novel

approach of how to implement the computation of Shannon entropy on fluid dynamics datasets,

- (2) Studying the parameters and types of conditions where the method works well and where it performs not so well yet,
- (3) Providing an open source framework, DataSiftML, that can be used to reproduce this work, as well as can be used on other datasets.

2 BACKGROUND

In reducing the available data and select optimal subsets, there are a numerous approaches that have been used. We will not cover all of them here, but generally they are listed as:

- (1) *Projection-based methods* – proper orthogonal decomposition (POD) [3], dynamic mode decomposition (DMD) [27]
- (2) *Neural network-based methods* – autoencoders, active learning, uncertainty-based sampling
- (3) *Cluster-based sampling* – first clustering the features using a clustering algorithms such as k-means, and then drawing samples from the clusters
- (4) *Random-sampling* – random sampling, random-weighted sampling, latin hypercube sampling (LHS)
- (5) *Importance sampling* – draw samples according to their importance, such as gradient, variance, or in our case entropy

We will narrow our review of these methods to those works which have used such methods in the application of reduced-order modeling of fluid dynamics.

Projection-based methods, or modal methods, attempt to find a low-dimensional *global* representation of the most of important characteristics of the solution field. These methods assume linear dynamics and learn global characteristic modes of the data. These modes can then be used as orthogonal basis functions to represent the data using (much) fewer degrees of freedom, assuming most of the important phenomena in the data can be represented reasonably accurately in a lower-dimensional space. The strengths of these methods, when they are successful, are that they identify the most important patterns (modes) in the data and are quite interpretable because: for example, the modes are functions over space that can be visualized. However, the drawbacks of these methods are (1) they are linear, so they are of limited applicability to approximating highly nonlinear dynamics, and (2) it is not obvious how to choose the number of modes to represent the data *a priori*. Learning latent spaces from autoencoders can also be thought of as a modal method, since this is also a data-driven dimension reduction that attempts to capture globally-important features. These are typically nonlinear methods, as they typically use neural networks with nonlinear activation functions. There has also been recent work in this field, of using such methods in combination with machine learning to learn optimal sensor placement for reconstruction [19].

Active learning is a strategy used in machine learning where the algorithm, instead of being passively fed with data, actively queries or selects the most “informative” data points to be labeled [23]. Chen [7] used active learning in the context of aerodynamic optimization of shape, in which he was able to use the method to automatically search for optimal drag-reduced candidate shapes, thereby reducing the number of required data points from 8000 down to 625. There are numerous approaches for actively querying

which data points should be included, which include some of the methods listed above such as random-sampling, but also methods such as uncertainty sampling [26].

Researchers have used cluster-based methods to simplify representations of flowfields. For example, Liu et al. [16] used principal component analysis (PCA) along with stream lines and surfaces to simplify the representation of 3D flowfields. Bhushan et al. [4] used Birch clustering to curate a reduced training dataset from a much larger DNS simulation dataset of turbulent channel flow. The purpose of the clustering was to reduce the skew of the data. They showed with using just 10% of the total dataset, they were able to achieve as good of performance as training on the entire dataset.

Regarding neural network based approaches, Ramos et al. [25] studied a comparison of DMD and convolutional autoencoders for numerical and video data analysis of fluid flow. They generally show that the DMD is more computationally efficient and produces better results than the convolution autoencoder approach. Moreover, Fukami et al. [10] performed an analysis of using autoencoders to learn a latent representation of several cases, including the two-dimensional cylinder wake flow-field as well as turbulent channel flow. While the approach is generally successful in reducing the order of the model, they mention one of the main challenges in using autoencoders is understanding and interpreting the learned latent space representation of the flows. Another notable work that uses autoencoders to learn the effective dynamics of fluid flows is the work by Vlachas et al. [30]. They use autoencoders along with recurrent neural networks to learn the macrodynamics of a flowfield.

In this work, we focus on an importance sampling method along with cluster analysis in order to find optimal clusters of data and sample from the optimal clusters to curate the training data set. Recently, maximum-entropy (MaxEnt) approaches have been used in sub-sampling molecular dynamics simulations [13, 21]. These studies demonstrate that automated methods leveraging maximum entropy outperform manually crafted datasets created by subject matter experts. The adoption of maximum entropy techniques allows for more efficient and effective data sub-sampling, enabling enhanced representations of complex molecular dynamics processes. This work inspired the work of Boyer et al. [5] to implement a more generalized MaxEnt approach that could be used to used to downselect features of flight tests data in order to train a machine-learned cruise guide indicator. We follow a similar approach of Boyer et al. [5], but adapt the approach for fluid dynamics datasets, which involve investigating the proper high-order descriptor to use when clustering, and also developing a graph-analysis approach to ranking the importance of the clusters.

3 METHODS

3.1 Running Simulations to Generate Raw Data

We performed our simulations using OpenFOAM (Open Field Operation and Manipulation), an open-source computational fluid dynamics (CFD) software package widely used for simulating complex fluid flow problems [31]. We are using a simplified version of OpenFOAM called SimpleFOAM, which numerically solves the Navier-Stokes equations and conservation of mass equations, given as:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (2)$$

where:

- \mathbf{u} = velocity vector,
- t = time,
- p = pressure,
- ρ = fluid density,
- ν = kinematic viscosity,
- \mathbf{f} = external body forces (e.g., gravity)

3.2 Sub-sampling via MaxEnt Method

In this section we describe the algorithm we developed for implementing the principle of maximum entropy. In implementing such a technique, we must first consider how to compute information entropy for a flowfield. We use a combination of clustering on high-level descriptor, which we describe as follows.

Step 1: Define and compute a high-level cluster variable of the simulation. We first considered how to describe the workflow with a descriptor-based method using vorticity as a high-level flow descriptor that concisely captures the flow dynamics. We have considered such parameters as vorticity, pressure, the stream function ψ , and combinations of such features. Vorticity is defined as:

$$\vec{\omega} \equiv \nabla \times \vec{u} \quad (3)$$

and in two-dimensions it may be computed using the following formula:

$$\omega = \partial v / \partial x - \partial u / \partial y \quad (4)$$

Step 2: Perform cluster analysis on cluster variable. Now, we perform a cluster analysis on the high-level descriptor using a clustering algorithm such as k -means – a centroid-based clustering algorithm that partitions data into k clusters by iteratively assigning points to the nearest cluster center and updating the centers based on the mean of the assigned points [18]. We have found that using a much higher value of k such as 20 generally worked better for fluid flows.

Note, that there are alternative clustering methods that may be used to perform a similar analysis that we have not yet tried but would like to test in the future, such as self-organized map (SOM) and DBSCAN. SOM is a neural network-based unsupervised learning algorithm that maps high-dimensional data onto a lower-dimensional grid, preserving topological relationships to reveal underlying patterns and structure in the data [14]. DBSCAN is a density-based clustering algorithm that groups data points into clusters based on their density, distinguishing dense regions from sparse ones and identifying noise points as well [9].

Step 3: Compute probability distributions of the clusters. Next, we compute probability distributions of the clusters coming out of Step 2. This first requires creating histograms of the cluster variable using `np.histogram` with a consistent bin range and count across all the histograms. Then, we convert the histograms to

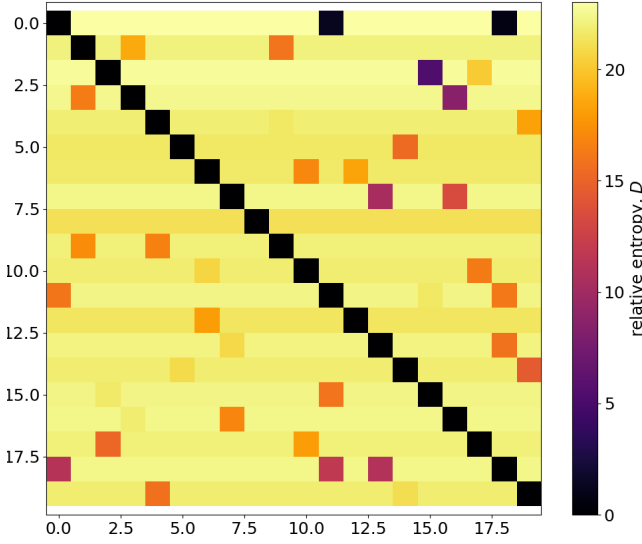


Figure 3: Adjacency matrix of relative entropies for 20 clusters of pressure.

probability distributions by dividing by the total number of counts in the histogram.

Step 4: Compute the relative entropy between all the probability distributions. The next step is to use Kullback-Leibler divergence to compute the relative entropy between the probability distributions, defined as:

$$D = \sum p \log(p/q) \quad (5)$$

For this we can simply use the `scipy.stats.entropy` function, for two probability p and q . An example of the adjacency matrix for 20 clusters is shown as follows in Fig. 3, defined by:

$$A_{ij} = \sum P(C_i) \log(P(C_i)/P(C_j)) \quad (6)$$

where A is the adjacency matrix, the elements of which are the relative entropies between the various clusters. Here, $P(C_i)$ is the probability distribution of pressure of cluster members of cluster C_i .

Step 5: Down-select optimal subset of samples based on in-degree strength. We next use the `networkx` library to convert the adjacency matrix into a directed graph. We compute the in-degree strength by summing the rows in the adjacency matrix. Next, we perform an `np.argsort` on all the in-strength values, to sort the clusters according to their in-strength. We can now sample from the optimal clusters for each time step to generate the training dataset. For this, we first convert the in-strength values into probabilities by using a MinMax scaler defined as:

$$w_{\text{scaled}} = \frac{w - w_{\min}}{w_{\max} - w_{\min}} \quad (7)$$

Now, we can simply use `np.random.choice` given these in-strength probabilities to select samples proportional to the cluster in-strengths.

3.3 Training the Neural Network on Data Subset

We used Keras with TensorFlow 2.11 to train the neural network on the downsampled data. Part of the process of training a good model involves not only designing a good baseline architectures, but also performing some level of neural architecture search (NAS) as shown in Fig. 2. Our DataSimML framework has built-in support for pluggable neural network architectures, such as multi-layer perceptrons (MLP), LSTM architectures, transformer architectures, etc., as well as built-in support for hyperparameter tuning via Keras-Tuner. We generally use hyperparameter tuning to select quantities such as the number of units or filters per layer, number of layers, dropout rate, learning rate, batch size, activation function type, etc.

In the case of unsteady flows, we needed to treat the problem as a temporal forecasting problem, in that we first need to create time sequences, so that we train on x_{i-2}, x_{i-1}, x_i in order to predict the future time step for the target y_{i+1} [1]. Lim and Zohren [15] provide a good overview of the types of networks and methods required to perform temporal forecasting using neural networks. The architectures generally include using either: 1D convolutional neural networks with causal dilated convolution layers, long short-term memory layers (LSTM), or attention-based architectures, namely Transformers [29].

4 RESULTS

In considering a sample problem to demonstrate the MaxEnt method, we considered the following possible use cases:

- Predicting drag on a cylinder
- Predicting lift and drag on airfoil
- Reconstructing flow-field given some sparse data points
- Super-resolution/coarse-graining [33]
- Mixing efficiency or energy decay of the system

We decided to focus on the flow over a cylinder case, that while geometrically simple, provides a rich unsteady flowfield to analyze via its Kármán vortex street wake. Moreover, in considering how to formulate the problem, there are two general approaches considered for demonstrating the MaxEnt approach:

- (1) *Local field prediction* – at each grid point local field values are used as features/inputs to the network, e.g., u , v , and predict local field values such as turbulent kinetic energy (TKE).
- (2) *Global quantity prediction* – use sampled data at each timestep, e.g., u and v , to predict a global value, such as lift or drag forces.

4.1 Laminar Flow over a Cylinder

With the goal of starting simple and working to more complex workflows, we started with 2D laminar flow over a cylinder simulation using OpenFOAM v7 official Docker release. We ran the simulations at Reynolds number 1267 as shown in Fig. 4 and experimented with our MaxEnt approach to extract out features per time step with the highest information content. We started with an existing setup for the OpenFOAM simulation (from <https://github.com/AsmaaHADANE>) and then modified the Reynolds number, added a code block to the `controlDict` file to compute the forces, as well as ran post-processing command `postProcess` to compute the cell centers and vorticity. We use `blockMesh` to generate a

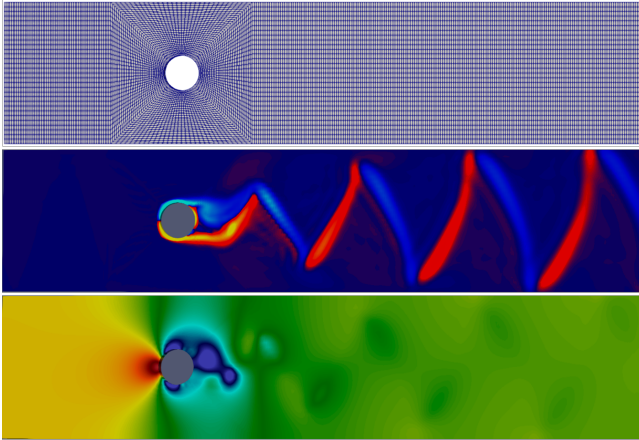


Figure 4: Computational grid (top), vorticity (middle) and pressure (bottom) for timestep 7200 rendered in ParaView.

multi-block grid around the cylinder, and specify slip conditions on the top and bottom walls, a no-slip condition on the cylinder surface, a uniform inflow value in the x -direction of 0.054 m/s, and a zero gradient condition at the exit.

In the case of *local field prediction*, the input shape for our neural network is (98, 3, 21600) and the output shape is (98, 3, 10800). Here the 98 represents the number of time sequences, each containing three time steps, and 10800 represents the number of grid points per time step per variable. On the other hand, for *global quantity prediction* our input shape is (98, 3, 21600) and our output shape for drag is (98, 3), since we are predicting a single quantity of drag for each time step.

We considered several different options for choosing the proper cluster variable. For this, we wanted a quantity which represented a higher-level statistic of the flow-field. So, we considered vorticity ω , as defined in Equation 4, computing the stream function ψ , using pressure p , or combinations of such variables. We can compute the total entropy, H , at each timestep by summing all the relative entropies in Fig. 3. In doing so, we found that pressure gives a higher entropy per timestep ($H=1907$) versus vorticity ($H=1886$). We could also consider clustering on combinations of variables, which we plan to do in the future.

To find the optimal value of k , we used the “elbow method”, which requires performing a study of a range of k values and plotting the sum of squared errors (SSE). Using such a method, the optimal k was identified to be about five to seven clusters for our cylinder test case. However, in practice, we found that using a much higher value of k of 20 worked better in terms of training loss.

In Fig. 5 we perform a MaxEnt cluster analysis of time step 7200 using seven clusters (NC) and show the probability distribution of the entire pressure for the time step in grey with red line edge, and then show the individual clusters. It is interesting to note how MaxEnt tends to weight clusters that are generally underrepresented in the flowfield.

Fig. 6 compares random sampling versus MaxEnt sampling clustered on vorticity and pressure. MaxEnt method does a good job of adapting to flow features downstream of the cylinder (for the case

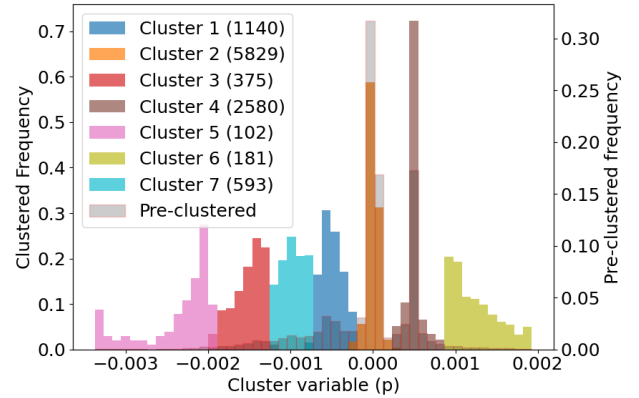


Figure 5: Clustering on pressure for time step 5500 (NC=7).

of vorticity), and upstream of the cylinder (for the case of pressure). Fig. 7 shows an example of the subsampled probability distributions of vorticity. Random sampling selects many samples near zero, whereas MaxEnt tends to select more samples in underpopulated regions of the flowfield, that have features of interest, e.g., in the vortex street.

Because random sampling is influenced by the grid adaption around the cylinder, we wrote a script in ParaView Python to first convert the OpenFOAM simulation to VTK format, and then interpolate the data onto a cartesian grid with equidistant spacing between the grid points. We trained a model to predict drag by first extracting 2000 samples from the simulation using both MaxEnt and Random sampling. Our neural network is a fully connected neural network with two hidden layers, each with 2000 units, using 100 epochs at batch size one. The results are shown in Table 1. While the training errors for the two methods are similar, the test error for the MaxEnt approach is about 12% less than the random sampling approach and the standard deviation is five times less.

	MaxEnt Sampling		Random Sampling	
	Train loss	Test loss	Train loss	Test loss
Train 1	0.0316	0.2004	0.0189	0.1931
Train 2	0.0265	0.1982	0.0443	0.2298
Train 3	0.0294	0.1903	0.0169	0.2391
avg	0.029	0.196	0.027	0.221
std	0.003	0.005	0.015	0.024

Table 1: Training performance comparing MaxEnt sampling vs Random sampling (target: drag).

4.2 Turbulent Magnetohydrodynamics Flow

The actual application that we are needing such a sub-sampling tool is for the development of a machine-learned turbulence heat flux model for magnetohydrodynamic (MHD) flows, with applications in nuclear fusion reactors [28]. NekRS simulations of MHD flow over a heated cylinder at Reynolds number 3900 were run on the Summit

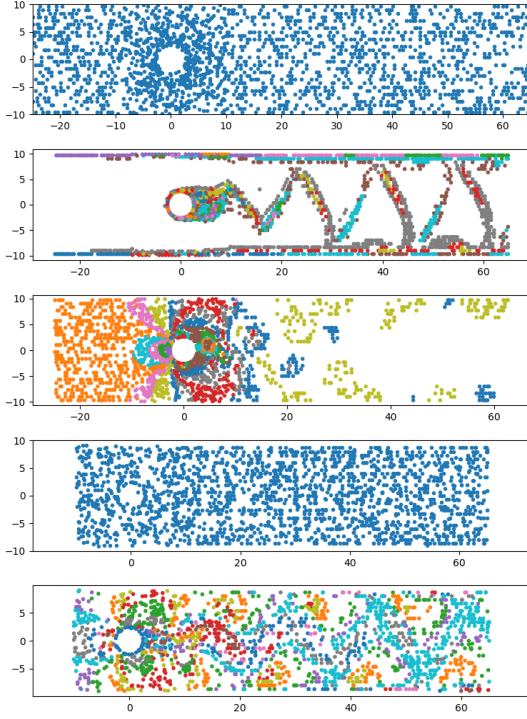


Figure 6: From top to bottom: Random sampling on original computational grid, MaxEnt sampling clustered on vorticity (time 7100), MaxEnt clustered on pressure (7200), Random sampling on cartesian projection, MaxEnt clustered on velocity for cartesian projection (time 9000).

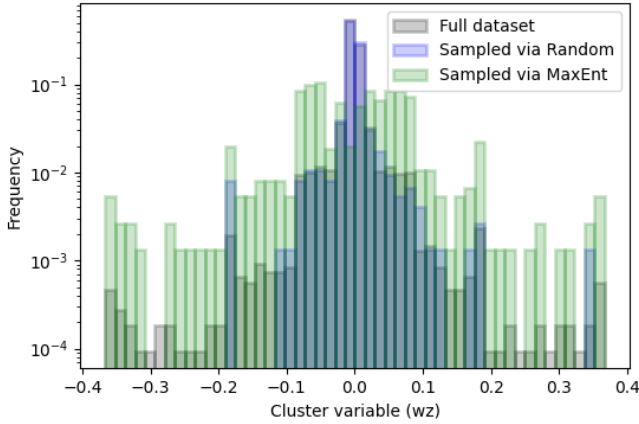


Figure 7: MaxEnt vs random sampling on vorticity.

supercomputer using six GPUs. In addition to solving the Navier-Stokes equations given in Equation 1, MHD flows also require solving two more equations, the energy equation and the transport equation for the magnetic field. The energy equation is given as:

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \alpha \nabla^2 T + \frac{\phi}{\rho c_p} \quad (8)$$



Figure 8: Clustering NekRS LES simulation on TKE.

$$\alpha = \frac{\kappa}{\rho c_p} + \frac{\mu_t}{\rho c_p Pr_t} \quad (9)$$

and the magnetic field is governed by:

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) \quad (10)$$

where \mathbf{B} is the magnetic field, \mathbf{u} is the velocity field, and t is time. Fig. 8 shows our clustering analysis on the cluster variable turbulent kinetic energy (TKE). We are in the process of developing a machine-learned turbulence model to predict the turbulent Prandtl number.

5 CONCLUSIONS

We present a novel method for optimally sub-sampling data from computational fluid dynamics simulations using the principle of maximum entropy. The method first performs clustering analysis on the flowfield using a cluster variable, such as vorticity, and then computes the relative entropy between the clusters using Kullbeck-Leibler divergence. Graph analysis is then performed on relative entropies in order to compute in-degree strengths; samples are then probabilistically drawn from the clusters according to their in-degree strength. The resulting dataset is then used to train a neural network. We test the method on an OpenFOAM simulation of two-dimensional laminar flow over a cylinder, treating each frame as independent events using a multi-layer perceptron, also investigate the temporal dynamics of the flowfield by first converting the data into temporal sequences and training on an LSTM architecture. We consider both local per-grid-point predictions as opposed to predicting a global quantity such as drag per timestep. We find the method to be very effective for extracting the essence of the dynamics required to train a neural network. Compared against random sampling, the MaxEnt approach produced a 12% lower test error on average with a 5x lower standard deviation. The code is freely available at <https://github.com/at-aaims/DataSiftML>¹.

6 ACKNOWLEDGMENTS

This research was sponsored by and used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility at the Oak Ridge National Laboratory (ORNL) supported by the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Moreover, we would like to express our appreciation to a number of colleagues who provided useful advice during the course of this work, including Joshua Brown, Jong Choi, and Mariia Karabin of ORNL.

¹To be made public by 11/12/23.

REFERENCES

- [1] Google AI. 2023. Time series forecasting. https://www.tensorflow.org/tutorials/structured_data/time_series
- [2] Prasanna Balaprakash, Michael Salim, Thomas D Uram, Venkat Vishwanath, and Stefan M Wild. 2018. DeepHyper: Asynchronous hyperparameter search for deep neural networks. In *2018 IEEE 25th international conference on high performance computing (HiPC)*. IEEE, 42–51.
- [3] Gal Berkooz, Philip Holmes, and John L. Lumley. 1993. The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annual Review of Fluid Mechanics* 25 (Jan. 1993), 539–575. <https://www.annualreviews.org/doi/abs/10.1146/annurev.fl.25.010193.002543>
- [4] Shanti Bhushan, Greg W Burgreen, Wesley Brewer, and Ian D Dettwiller. 2021. Development and validation of a machine learned turbulence model. *Energies* 14, 5 (2021), 1465.
- [5] Mathew Boyer, Wesley Brewer, Jeff Finckenor, Chris Brackbill, Daniel Martinez, and Andrew Wissink. 2023. Development of a Machine-Learned Cruise Guide Indicator for Rotorcraft. In *Proceedings of the 79th Annual Forum of the Vertical Flight Society*. West Palm Beach, Florida. <https://doi.org/10.4050/F-0079-2023-18164>
- [6] Wesley Brewer, Daniel Martinez, Mathew Boyer, Dylan Jude, Andy Wissink, Ben Parsons, Junqi Yin, and Valentine Anantharaj. 2021. Production deployment of machine-learned rotorcraft surrogate models on hpc. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*. IEEE, 21–32.
- [7] Yang Chen. 2020. Active learning over dnn: Automated engineering design optimization for fluid dynamics based on self-simulated dataset. *arXiv preprint arXiv:2001.08075* (2020).
- [8] Miles MP Couchman, Stephen M de Bruyn Kops, and P Caulfield Colm-cille. 2023. Mixing across stable density interfaces in forced stratified turbulence. *Journal of Fluid Mechanics* 961 (2023), A20.
- [9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. 226–231.
- [10] Kai Fukami, Kazuto Hasegawa, Taichi Nakamura, Masaki Morimoto, and Koji Fukagata. 2021. Model order reduction with neural networks: Application to laminar and turbulent flows. *SN Computer Science* 2 (2021), 1–16.
- [11] Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical review* 106, 4 (1957), 620.
- [12] Michael G Kapteyn, David J Knezevic, DBP Huynh, Minh Tran, and Karen E Willcox. 2022. Data-driven physics-based digital twins via a library of component-based reduced-order models. *Internat. J. Numer. Methods Engrg.* 123, 13 (2022), 2986–3003.
- [13] Mariia Karabin and Danny Perez. 2020. An entropy-maximization approach to automated training set generation for interatomic potentials. *The Journal of Chemical Physics* 153, 9 (2020).
- [14] T. Kohonen. 1990. The self-organizing map. *Proc. IEEE* 78, 9 (1990), 1464–1480.
- [15] Bryan Lim and Stefan Zohren. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A* 379, 2194 (2021), 20200209.
- [16] Fan Liu, Wensheng Zhou, Bingxuan Liu, Ke Li, Kai Zhang, Chenming Cao, Guoyu Qin, Chen Cao, and Renfeng Yang. 2022. Flow Field Description and Simplification Based on Principal Component Analysis Downscaling and Clustering Algorithms. *Frontiers in Earth Science* 9 (2022), 804617.
- [17] Siyan Liu, Pei Zhang, Dan Lu, and Guannan Zhang. 2021. PI3NN: Out-of-distribution-aware prediction intervals from three neural networks. *arXiv preprint arXiv:2108.02327* (2021).
- [18] J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1*. 281–297.
- [19] Krithika Manohar, Bingni W. Brunton, J. Nathan Kutz, and Steven L. Brunton. 2018. Data-Driven Sparse Sensor Placement for Reconstruction: Demonstrating the Benefits of Exploiting Known Patterns. *IEEE Control Systems Magazine* 38, 3 (June 2018), 63–86. <https://doi.org/10.1109/MCS.2018.2810460> Conference Name: IEEE Control Systems Magazine.
- [20] Daniel A. Martinez-Gonzalez, Dylan Jude, and Andrew Wissink. 2022. ROAM-ML: A reduced order aerodynamic module augmented with neural network digital surrogates. *AIAA SCITECH 2022 Forum* (2022).
- [21] David Montes de Oca Zapiain, Mitchell A Wood, Nicholas Lubbers, Carlos Z Pereyra, Aidan P Thompson, and Danny Perez. 2022. Training data selection for accuracy and transferability of interatomic potentials. *npj Computational Materials* 8, 1 (2022), 189.
- [22] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2021. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment* 2021, 12 (2021), 124003.
- [23] OpenAI. n.d.. ChatGPT: An AI Language Model based on GPT-4 Architecture. <https://www.openai.com/>. Accessed: [May 3, 2023].
- [24] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, G de Marmiesse, Y Fu, J Podivin, F Schäfer, et al. 2023. Keras Tuner. 2019. Available online: github.com/keras-team/kerastuner (accessed on 2 April 2022) (2023).
- [25] Eliaquim M Ramos, Gabriella M Darze, Francisco RT do Nascimento, José Luiz H Faccini, and Gilson A Giraldo. 2020. Comparison of dynamic mode decomposition and deep learning techniques for two-phase flows analysis. *Flow, Turbulence and Combustion* 105 (2020), 1345–1379.
- [26] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)* 54, 9 (2021), 1–40.
- [27] Peter J. Schmid. 2010. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics* 656 (2010), 5–28. <https://doi.org/10.1017/S0022112010001217>
- [28] Arpan Sircar, Jin Whan Bae, Ethan Peterson, Jerome Solberg, and V Badalasi. 2022. FERM: A multi-physics simulation environment for fusion reactor blanket. Technical Report. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [30] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. 2022. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence* 4, 4 (2022), 359–366.
- [31] Henry G Weller, Gavin Tabor, Hrvoje Jasak, and Christer Fureby. 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics* 12, 6 (1998), 620–631.
- [32] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the workshop on machine learning in high-performance computing environments*. 1–5.
- [33] Jiawei Zhuang, Dmitrii Kochkov, Yohai Bar-Sinai, Michael P Brenner, and Stephan Hoyer. 2021. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids* 6, 6 (2021), 064605.

Received 18 August 2023