Machine Learning Driven Metric Based Mesh Adaptation

Kunal Ghosh
Department of Aerospace Engineering
Indian Institute of Science
Bengaluru, Karnataka, India
kunalghosh@alum.iisc.ac.in

Bakhshi Mehul
Department of Aerospace Engineering
Indian Institute of Science
Bengaluru, Karnataka, India
bakhshimehul@iisc.ac.in

Kush Pandya
Department of Aerospace Engineering
Indian Institute of Science
Bengaluru, Karnataka, India
kushpandya@iisc.ac.in

Dipendrasingh Kain
Department of Aerospace Engineering
Indian Institute of Science
Bengaluru, Karnataka, India
kains@iisc.ac.in

Aravind Balan [*]
Department of Aerospace Engineering
Indian Institute of Science
Bengaluru, Karnataka, India
aravindbalan@iisc.ac.in

Ajay Rangarajan
Graduate School AICES
RWTH Aachen University
52062 Aachen, Germany
rangarajan@aices.rwth-aachen.de

Abstract

Mesh adaptation is crucial in CFD to dynamically refine and optimize the computational grid, enhancing accuracy in capturing complex flow features. Metric-field-based mesh adaptation, while mathematically robust, typically relies on adjoint solutions for error estimation, which can significantly increase computational demands. To address this challenge, this research aims to develop a machine learning-driven approach to mesh adaptation in CFD, eliminating the need for computationally intensive adjoint solutions. In pursuit of this goal, we employ ensemble models and Graph Convolutional Networks (GCNs) to predict the local error estimator for each cell of the mesh during the adaptation process. Our findings show that GCNs outperform ensemble models for isotropic meshes, while both models yield similar results in anisotropic meshes. These results demonstrate that our machine learning-driven approach eliminates the need to solve adjoint equations for error estimation, paving the way for more efficient CFD simulations in complex flow scenarios.

1 Introduction

In engineering, the ability to simulate fluid flow with precision is often pivotal to the success of critical applications. A diverse array of fluid flow challenges can be effectively addressed through the numerical solution of partial differential equations (PDEs), which serve as the foundation for these simulations. The advent of advanced numerical methodologies has revolutionized engineering practices, empowering professionals to tackle complex flow phenomena with greater accuracy and efficiency. A vital aspect of this process is the discretization of the computational domain, commonly referred to as meshing, which is essential for obtaining reliable numerical solutions. To further

[*]Corresponding author: aravindbalan@iisc.ac.in

enhance the accuracy and computational efficiency of numerical simulations, mesh adaptation techniques dynamically adjust the mesh resolution within the computational domain.

1.1 Metric-field Based Mesh Adaptation

Building on this foundation, metric-field-based mesh adaptation emerges as a crucial strategy for refining mesh resolution in response to the characteristics of practical fluid flows. Practical fluid flows often exhibit directional features, highlighting the need for anisotropic meshes to enhance both efficiency and accuracy. Metric-field-based mesh adaptation (A. Rangarajan [2020]) provides a robust mathematical framework for implementing anisotropic mesh refinement. This algorithm involves numerically solving a PDE and executing a series of processes for metric-based mesh adaptation, as illustrated in Fig. 1, adapted from A. Balan [2020].

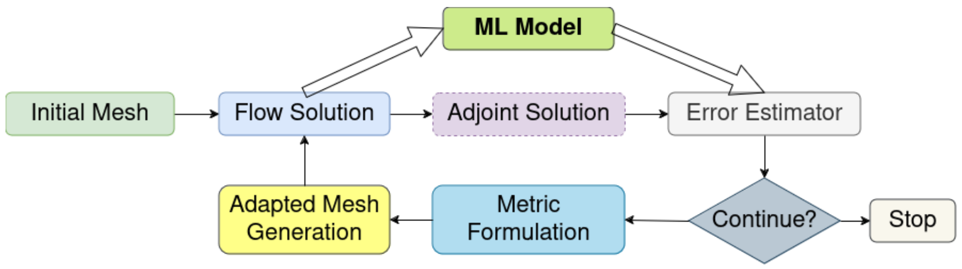

Figure 1: Overview of metric-based mesh adaptation and implimentation of ML model.

1.2 ML for Mesh Adaptation

In recent years, the application of machine learning techniques for adaptive mesh refinement has garnered significant attention as a means to enhance computational efficiency and accuracy. For instance, S. Sanchez-Gamero [2024] utilized Artificial Neural Networks (ANNs) to predict spacing functions for constructing meshes tailored to unseen turbulent flows. Similarly, C. Foucart [2023] investigated the implementation of deep reinforcement learning in the isotropic mesh adaptation process. Additional studies, including L. Manevitz [2005], J. Bohn [2021], J. Yang [2023], J. Yang [2022], and K. Tlales [2024], have demonstrated various applications of machine learning in mesh adaptation. While most research has focused on heuristic-based isotropic mesh adaptation, some notable works by K.J. Fidkowski [2021], V. Ojha [2022], and G. Chen [2021] have proposed the use of ANNs and Convolutional Neural Networks (CNNs) for metric-based mesh adaptation, motivated by its mathematical rigour for the error estimation.

This study aims to build on previous research by Ghosh [2024] involving ensemble learning and GCNs integration into a metric-based mesh adaptation algorithm of Rangarajan [2021]. The goal is to predict errors using the mesh and flow solution data, thereby accelerating the adaptation process. The predicted errors are utilized to determine isotropic mesh density (d), while the anisotropic parameters (β, θ), essential for constructing the metric, are derived from interpolation error.

2 Dataset Creation

2.1 Data Acquisition

Numerical simulation data was generated using our in-house Hybrid Discontinuous Galerkin (HDG) solver (Hecht [1998], S. Balay, Rangarajan [2021], A. Balan [2020]) with a polynomial order of $p = 2$. The simulations started with an unstructured, uniform mesh that was progressively refined through multiple iterations using the metric-based mesh adaptation algorithm, targeting approximately 3000 elements until the error stabilized. For this study, we analytically computed the source term using the manufactured solution for the advection-diffusion equation from Rangarajan [2021], and then solved the equation with the HDG solver to obtain the numerical solution, u_h.

$$\beta_1 \frac{\partial u}{\partial x} + \beta_2 \frac{\partial u}{\partial y} = \epsilon(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + s$$

Here, ϵ represents diffusion coefficients, β_1 and β_2 denote convection coefficients in x and y directions, respectively, and s is the source term. We generated 1000 data points with ϵ ranging from 0.1 to 1 and $\beta_{1,2}$ ranging from 1 to 5.

2.2 Inputs and Output

The inputs to the machine learning models included the coefficients of the equation (ϵ, β_1, β_2), elemental properties (element area I, β, θ), numerical solutions (u_h), gradients of the numerical solutions ($\vec{\nabla} u_h$), as well as the jumps in numerical solutions at the element edges ([[u_h]]) and the jumps in gradients of the numerical solutions at the element edges ([[$\vec{\nabla} u_h$]]). For each integration point (i) within the element, we predicted the error (ϵ_i) using the formula:

$$\epsilon_i = \mid u_i - u_{hi} \mid$$

Rather than predicting ϵ_i directly, our models focus on predicting the normalized error ($\widetilde{\epsilon_i}$). This approach was chosen because ϵ_i can vary significantly across orders of magnitude, while $\widetilde{\epsilon_i}$ remains relatively consistent. The relationship between ϵ_i and $\widetilde{\epsilon_i}$ is given by, $\widetilde{\epsilon_i} = log(\epsilon_i^2)$.

To improve the performance of the ensemble model, Z-score and Min-Max normalization techniques were applied sequentially to normalize both the input and output data.

3 Methodology

3.1 Ensemble Models

Ensemble models harness the strength of diversity by integrating distinct algorithms, each capturing unique aspects of the data. We selected ensemble models for our study due to their effectiveness in balancing the bias-variance trade-off (Kunapuli [2023]).

To capitalize on the strengths of weak learners, we implemented a sequential ensemble model, as illustrated in Fig. 2, utilizing TensorFlow M. Abadi [2015]. We opted for three base learners because using only two resulted in underfitting, whereas incorporating four led to overfitting.

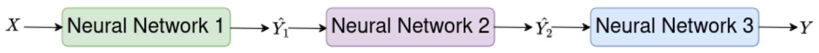

Figure 2: Sequential ensemble model with 3 base learners.

A novel aggregation method combining boosting and stacking aggregation was used to optimize the predictive performance of the ensemble models, as detailed in Ghosh [2024]. In this framework, the output of each base learner is utilized as the input for the subsequent learner, creating a sequential learning process. The hyperparameters for our neural network base learners are outlined in Table 1. We opted for ensemble models with consistent hyperparameters across all base learners, as this strategy has been shown to yield better performance compared to those with varying hyperparameters.

3.2 Graph Neural Networks

The error (ϵ_i) at an integration point (i) is interdependent with the errors at neighbouring integration points, rather than being independent and identically distributed (IID). This interdependence prompted us to employ graph neural networks for node regression, utilizing PyTorch A. Paszke [2019] and PyTorch Geometric M. Fey [2019] as detailed by Hamilton [2020].

We used GCN with symmetric-normalized aggregation and the self-loop update method. The hyper-parameters used for isotropic and anisotropic GCN models are specified in Table 1. The architectures of the two models were the same, with the only difference being an additional normalization layer after the first graph convolution layer in the anisotropic model as detailed in Ghosh [2024].

Table 1: Hyperparameters for Machine Learning models.

	Ensemble	GCN	
Hyperparameter		Isotropic	Anisotropic
Loss function	Huber loss	Huber loss	Huber loss
Learning rate	0.01	0.01	0.01
Optimizer	Adam	Adam	Adam
Checkpoint metric	R^2 score of validation set	Loss	Loss
Batch size	4096 elements	128 graphs	32 graphs

4 Results

Trained ensemble and GCN models were deployed and tested on the advection-diffusion case with parameters $\epsilon = 0.01$, $\beta_1 = 2.0$, and $\beta_2 = 2.0$. We intentionally selected ϵ outside the training data range of [0.1, 1.0] to evaluate the models' generalization capabilities. These particular coefficient values were chosen to produce a solution with a sharp boundary layer, highlighting the necessity of accurately capturing this critical feature through machine learning models.

4.1 Isotropic Mesh Adaptation

The ensemble and GCN models effectively captured the sharp boundary layer along the top and right edges. As illustrated in Fig. 3, the adapted mesh generated by these models accurately represents this feature, along with the corresponding solution. We assessed the models' performance by comparing their results to those obtained from true error-based adaptation (adapt-exact).

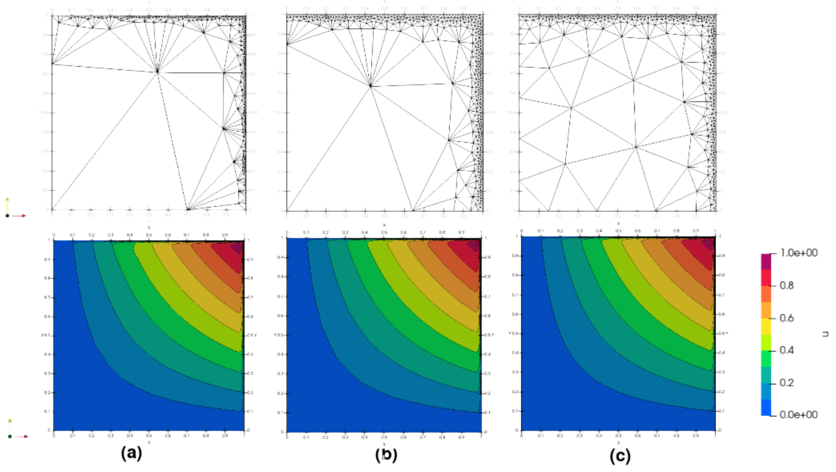

Figure 3: Isotropic adapted mesh and solution (a) Adapt-exact (727 elements) (b) Ensemble (839 elements) (c) GCN (828 elements).

Fig. 4 displays the error convergence plot of the L_2 norm of error in the domain against representative mesh size (h). Both the ensemble and GCN models exhibit approximately the same slope of 3. Notably, the error in the mesh generated by the GCN model is lower than that of the ensemble model, except in cases with a larger total number of elements.

4.2 Anisotropic Mesh Adaptation

Anisotropic-adapted meshes are shown in Fig. 5. While both models captured the boundary layers effectively, GCN models can be observed struggling to produce coarse anisotropic elements compared to adapt-exact and ensemble models.

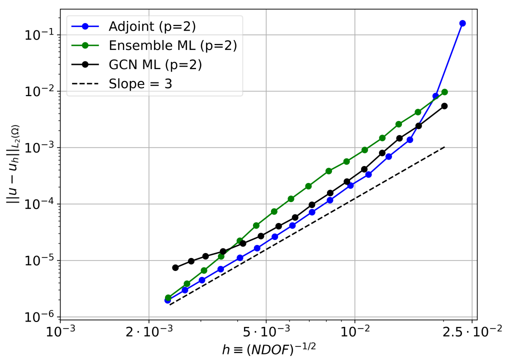

Figure 4: Error vs h for isotropic case, $\epsilon = 0.01$, $\beta_1 = 2.0$ and $\beta_2 = 2.0$.

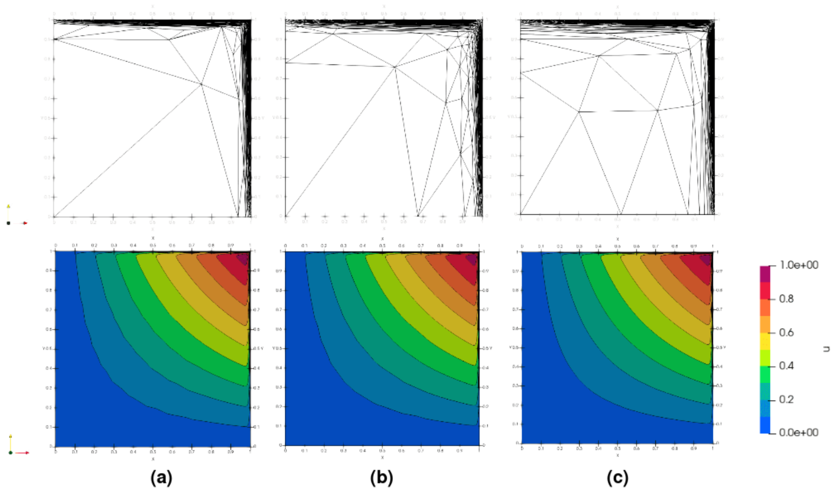

Figure 5: Anisotropic adapted mesh (a) Adapt-exact (560 elements) (b) Ensemble (590 elements) (c) GCN (575 elements).

Fig. 6 presents the plot of the L_2 norm of error in the domain versus h for anisotropic meshes. Both the ensemble and GCN models exhibit approximately the same slope of 3. Notably, the error produced by the anisotropic meshes is an order of magnitude lower than that of the isotropic meshes. However, the ensemble model demonstrated slightly better performance than the GCN model in the context of anisotropic mesh adaptation.

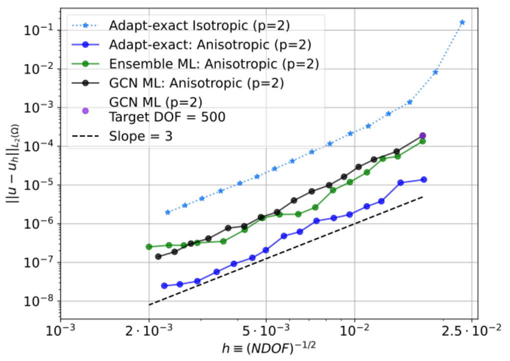

Figure 6: Error vs h for anisotropic case, $\epsilon = 0.01$, $\beta_1 = 2.0$ and $\beta_2 = 2.0$.

5 Conclusion

In summary, ensemble learning and GCN models effectively predict errors for mesh adaptation in the convection-diffusion problem, with GCNs excelling for isotropic meshes and ensemble models marginally outperforming GCNs in anisotropic adaptation. Future work will extend these models to more complex scenarios, including inviscid and viscous flow around airfoils, considering varying angles of attack and Mach numbers across different flow regimes.

References

S.L. Wood W.K Anderson A. Balan, M.A. Park. Verification of anisotropic mesh adaptation for complex aerospace applications. In *AIAA scitech 2020 forum*, page 0675, 2020.

F. Massa A. Lerer J. Bradbury G. Chanan T. Killeen Z. Lin N. Gimelshein L. Antiga A. Desmaison A. Kopf E. Yang Z. DeVito M. Raison A. Tejani S. Chilamkurthy B. Steiner L. Fang J. Bai S. Chintala A. Paszke, S. Gross. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

V. Dolejsi A. Rangarajan, G. May. Adjoint-based anisotropic hp-adaptation for discontinuous galerkin methods using a continuous mesh model. *Journal of Computational Physics*, 409:109321, 2020.

P.F.J. Lermusiaux C. Foucart, A. Charous. Deep reinforcement learning for adaptive mesh refinement. *Journal of Computational Physics*, 491:112381, 2023.

K.J. Fidkowski G. Chen. Output-based adaptive aerodynamic simulations using convolutional neural networks. *Computers and Fluids*, 223:104947, 2021.

K. Ghosh. Mesh intelligence ml-driven error estimation for mesh adaptation. 2024. URL `https://www.academia.edu/122348532/Mesh_Intelligence_ML_Driven_Error_Estimation_for_Mesh_Adaptation_Thesis_`.

W.L. Hamilton. *Graph representation learning*. Morgan and Claypool Publishers, 2020. ISBN 978-1681739649.

F. Hecht. Bamg: bidimensional anisotropic mesh generator. *User Guide. INRIA, Rocquencourt*, 17, 1998.

M. Feischl J. Bohn. Recurrent neural networks as optimal mesh refinement strategies. *Computers Mathematics with Applications*, 97:61–76, 2021.

B. Petersen J. Kudo K. Mittal V. Tomov J. Camier T. Zhao H. Zha T. Kolev R. Anderson D. Faissol J. Yang, T. Dzanic. E2n: error estimation networks for goal-oriented mesh adaptation. *arXiv preprint*, 2207:11233, 2022.

B. Petersen J. Kudo K. Mittal V. Tomov J. Camier T. Zhao H. Zha T. Kolev R. Anderson D. Faissol J. Yang, T. Dzanic. Reinforcement learning for adaptive mesh refinement. In *International Conference on Artificial Intelligence and Statistics. PMLR*, pages 5997–6014, 2023.

G. Ntoukas G. Rubio E. Ferrer K. Tlales, K.E. Otmani. Machine learning mesh-adaptation for laminar and turbulent flows: applications to high-order discontinuous galerkin solvers. *Engineering with Computers*, pages 1–23, 2024.

G. Chen K.J. Fidkowski. Metric-based, goal-oriented mesh adaptation using machine learning. *Journal of Computational Physics*, 426:109957, 2021.

G. Kunapuli. *Ensemble Methods for Machine Learning*. Manning Publications Co., 20 Baldwin Road, Shelter Island, New York, 2023. ISBN 978-1617297137.

D. Givoli L. Manevitz, A. Bitar. Neural network time series forecasting of finite-element mesh adaptation. *Neurocomputing*, 63:447—-463, 2005.

P. Barham E. Brevdo Z. Chen C. Citro G.S. Corrado A. Davis J. Dean M. Devin S. Ghemawat I. Goodfellow A. Harp G. Irving M. Isard Y. Jia R. Jozefowicz L. Kaiser M. Kudlur J. Levenberg D. Mané R. Monga S. Moore D. Murray C. Olah M. Schuster J. Shlens B. Steiner I. Sutskever K. Talwar P. Tucker V. Vanhoucke V. Vasudevan F. Viégas O. Vinyals P. Warden M. Wattenberg M. Wicke Y. Yu X. Zheng M. Abadi, A. Agarwal. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

J.E. Lenssen M. Fey. *Fast Graph Representation Learning with PyTorch Geometric*. URL: `https://github.com/pyg-team/pytorch_geometric`, 2019.

A. Rangarajan. Metric based hpadaptation using a continuous mesh model for higher order schemes. 2021. URL `https://publications.rwth-aachen.de/record/817176`.

M.F. Adams S. Benson J. Brown P. Brune K. Buschelman E.M. Constantinescu L. Dalcin A. Dener V. Eijkhout J. Faibussowitsch W.D. Gropp V. Hapla T. Isaac P. Jolivet D. Karpeev D. Kaushik M.G. Knepley F. Kong S. Kruger D.A. May L.C. McInnes R. Mills Tran L. Mitchell T. Munson J.E. Roman K. Rupp P. Sanan J. Sarich B.F. Smith S. Zampini H. Zhang J. Zhang S. Balay, S. Abhyankar. Petsc/tao users manual (rev. 3.20). doi: 10.2172/2205494. URL `https://www.osti.gov/biblio/2205494`.

R. Sevilla S. Sanchez-Gamero, O. Hassan. A machine learning approach to predict near-optimal meshes for turbulent compressible flow simulations. *arXiv preprint*, 2406:16057, 2024.

K.J. Fidkowski V. Ojha, G. Chen. Initial mesh generation for solution-adaptive methods using machine learning. In *AIAA scitech 2022 forum*, page 1244, 2022.