National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

Graph Neural Networks as Surrogates of Turbulence and Transition Models in Aerodynamics

Diploma Thesis

Giounteri Linta Anastasia

Advisors:

Kyriakos C. Giannakoglou, Professor NTUA

Dr. V. Asouti, Adjunct Lecturer NTUA

Athens, 2026

Acknowledgments

First, I would like to express my sincere gratitude to my advisor, Professor Kyriakos C. Giannakoglou, for guiding the entire process of this diploma thesis. His experience and scientific insight were essential for the successful completion of this work.

I would also like to sincerely thank Dr. Varvara Asouti for implementing the neural network configurations in the PUMA solver, as well as for her continuous guidance and support on several aspects of this research.

I would like to thank Dr. Marina Kontou for her valuable help and support throughout this process, for helping me develop new knowledge and skills, and for providing important scientific background through her PhD work, which formed a foundation for this diploma thesis.

Finally, I would like to thank my family and friends for their moral support and encouragement.

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

Graph Neural Networks as Surrogates of Turbulence and Transition Models in Aerodynamics

Diploma Thesis

Giounteri Linta Anastasia

Advisors:

Kyriakos C. Giannakoglou, Professor NTUA

Dr. V. Asouti, Adjunct Lecturer NTUA

Athens, 2026

Abstract

In this Diploma Thesis, Graph Neural Networks (GNNs) are investigated as surrogate models for turbulence and transition modelling in Computational Fluid Dynamics (CFD). Unlike pointwise neural networks (NNs), such as Fully Connected Neural Networks (FCNNs), which rely solely on local mesh node features, GNNs explicitly incorporate neighbourhood information defined by the mesh connectivity.

Preliminary tests are conducted to validate the end-to-end pipeline and to investigate the influence of mesh resolution on NN predictions for a two-variable function. Four meshes with increasing resolutions are generated, and a polynomial function is used to construct the corresponding datasets. For each mesh resolution, a Graph Attention Network (GAT), a type of GNN, and an FCNN are trained, and the effect of mesh resolution on model performance is evaluated. Subsequently, the network architectures are optimized using the in-house software EASY, with the loss as the objective function and the number of layers, neurons per layer, and activation functions in the hidden and output layers as design variables. The optimized models are then compared in terms of target prediction accuracy for both a simple and a more complex function.

Two CFD test cases are considered: external aerodynamics around an isolated airfoil (NACA4318) and internal aerodynamics around a transonic turbine blade airfoil (LS89). In both cases, the geometries are varied while the flow conditions are kept fixed. For each geometry, C-type meshes with identical node counts are generated. The objective is to develop NN surrogate models capable of replacing the turbulence (Spalart–Allmaras for the airfoil case, $k - \omega \ SST$ for the turbine blade airfoil) and transition model ($\gamma - \tilde{Re}_{\theta t}$ for the turbine blade airfoil), within the in-house CFD solver PUMA. To this end, GATs, Message Passing Neural Networks (MPNNs), both belonging to the class of GNNs, and FCNNs are trained to predict the turbulent viscosity field μ_t based on geometric and flow-related quantities. GNN and FCNN architectures are optimized, trained, and compared in terms of prediction accuracy. Additional tests are conducted for the airfoil case to demonstrate the ability of the NN models to operate on unstructured meshes with variable node counts. NN models are integrated into the PUMA solver to assess their performance in a coupled CFD environment. The GNN and FCNN models are compared using both predictions of turbulent viscosity field and integral quantities, namely lift and drag coefficients for the airfoil case and mass-averaged total pressure losses for the turbine blade airfoil.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστομηχανικής & Βελτιστοποίησης

Νευρωνικά δίκτυα τύπου γράφου ως υποκατάστατα μοντέλων τύρβης και μετάβασης στην αεροδυναμική

Διπλωματική Εργασία

Γιουντέρη Λίντα Αναστασία

Επιβλέποντες:

Κυριάκος Χ.Γιαννάκογλου, Καθηγητής ΕΜΠ

Δρ. Β. Ασούτη, Εντεταλμένη Διδάσκουσα ΕΜΠ

Αθήνα, 2026

Περίληψη

Στην Διπλωματική Εργασία εξετάζεται η χρήση Νευρωνικών Δικτύων Γράφων (GNN) ως υποκατάστατων των μοντέλων τύρβης και μετάβασης στο πλαίσιο της Υπολογιστικής Ρευστοδυναμικής (ΥΔΡ). Σε σύγκριση με τα Πλήρως Συνδεδεμένα Νευρωνικά Δίκτυα (FCNN), τα οποία βασίζονται αποκλειστικά στα τοπικά χαρακτηριστικά των κόμβων του υπολογιστικού πλέγματος, τα GNN ενσωματώνουν πληροφορία από τους γειτονικούς κόμβους μέσω της συνδεσιμότητας του γράφου που προκύπτει από την τοπολογία του υπολογιστικού πλέγματος.

Για την αξιολόγηση της ολοκληρωμένης ροής εργασίας και τη διερεύνηση της επίδρασης της πυκνότητας του πλέγματος στην ακρίβεια των Νευρωνικών Δικτύων (NN), πραγματοποιούνται δοκιμές σε μία συνάρτηση δύο μεταβλητών με τη χρήση Νευρωνικών Δικτύων Γράφων τύπου Δικτύων Προσοχής σε Γράφους (Graph Attention Networks – GAT) και FCNN. Δημιουργούνται τέσσερα πλέγματα με αυξανόμενη πυκνότητα και χρησιμοποιείται μία πολυωνυμική συνάρτηση για την κατασκευή των αντίστοιχων σετ δεδομένων. Για κάθε πυκνότητα πλέγματος εκπαιδεύονται ένα GNN και ένα FCNN και αξιολογείται η επίδραση της πυκνότητας του πλέγματος στην απόδοσή τους. Στη συνέχεια, οι αρχιτεκτονικές των NN βελτιστοποιούνται ως προς το σφάλμα, μέσω του λογισμικού του εργαστηρίου EASY, ώστε να καθοριστεί βέλτιστα ο αριθμός των επιπέδων, ο αριθμός των νευρώνων ανά επίπεδο και οι συναρτήσεις ενεργοποίησης στα κρυφά επίπεδα και στο επίπεδο εξόδου. Τα βελτιστοποιημένα μοντέλα συγκρίνονται ως προς την ακρίβεια πρόβλεψης για μία απλή και μία πιο σύνθετη συνάρτηση.

Εξετάζονται δύο περιπτώσεις ΥΔΡ: η εξωτερική αεροδυναμική γύρω από μία μεμονωμένη αεροτομή (NACA4318) και η εσωτερική αεροδυναμική σε διηχητική 2Δ πτερύγωση στροβίλου (LS89). Και στις δύο περιπτώσεις, οι γεωμετρίες μεταβάλλονται, ενώ οι συνθήκες ροής διατηρούνται σταθερές. Για κάθε γεωμετρία δημιουργούνται πλέγματα τύπου C με ίδιο αριθμό κόμβων. Στόχος είναι η ανάπτυξη NN ικανών να αντικαταστήσουν τα μοντέλα τύρβης (Spalart–Allmaras για τη μεμονωμένη αεροτομή και $k-\omega$ SST για την πτερύγωση στροβίλου) καθώς και το μοντέλο μετάβασης ($\gamma-\widetilde{Re}_{\theta t}$ για την πτερύγωση στροβίλου) στον υπολογιστικό κώδικα ΥΔΡ του εργαστηρίου PUMA. Για τον σκοπό αυτό, GNN τύπου GAT και Νευρωνικών Δικτύων Μετάδοσης Μηνυμάτων (Message Passing Neural Networks – MPNN), καθώς και FCNN, εκπαιδεύονται να προβλέπουν το πεδίο της τυρβώδους συνεκτικότητας μ_t, με είσοδο γεωμετρικές και ροϊκές ποσότητες. Οι αρχιτεκτονικές GNN και FCNN βελτιστοποιούνται, εκπαιδεύονται και συγκρίνονται ως προς την ακρίβεια πρόβλεψης. Επιπλέον, στην περίπτωση της μεμονωμένης αεροτομής πραγματοποιούνται δοκιμές για να εξεταστεί η ικανότητα των NN να λειτουργούν σε μη δομημένα πλέγματα με μεταβαλλόμενο αριθμό κόμβων. Τα εκπαιδευμένα μοντέλα ενσωματώνονται στον επιλυτή PUMA για την αξιολόγηση της απόδοσής τους μέσα στον κώδικα ΥΔΡ. Η σύγκριση των GNN και FCNN πραγματοποιείται μέσω του πεδίου της τυρβώδους συνεκτικότητας αλλά και μέσω ολοκληρωματικών μεγεθών, όπως οι συντελεστές άνωσης και οπισθέλκουσας για την αεροτομή και οι μαζικά σταθμισμένες απώλειες ολικής πίεσης για την πτερύγωση στροβίλου.

Contents

Nomenclature

AI	Artificial Intelligence
AoA	Angle of Attack
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DL	Deep Learning
DNS	Direct Numerical Simulation
EA	Evolutionary Algorithm
EASY	Evolutionary Algorithms SYstem
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GeLU	Gaussian Error Linear Unit
GNN	Graph Neural Network
LES	Large–Eddy Simulation
LHS	Latin Hypercube Sampling
MAE	Mean Absolute Error
ML	Machine Learning
MPNN	Message Passing Neural Network
MSE	Mean Squared Error
NTUA	National Technical University of Athens
NURBS	Non-Uniform Rational B-Splines
PCOpt	Parallel CFD & Optimization Unit
RANS	Reynolds–Averaged Navier–Stokes equations
ReLU	Rectified Linear Unit

SA	Spalart–Allmaras turbulence model

Chapter 1

Introduction

1.1 CFD and Turbulence

Computational Fluid Dynamics (CFD) solves the Navier–Stokes equations on a discretized domain (mesh) to predict fluid flow around complex geometries. In practical aerodynamics, the relevant flows are frequently turbulent, exhibiting fluctuations in velocity and pressure over a wide range of scales.

Laminar vs. turbulent flow The transition from laminar to turbulent flow is governed primarily by the Reynolds number, $\mathrm{Re} = UL/\nu$, where U is a characteristic velocity, L a characteristic length, and ν the kinematic viscosity. Laminar flow features smooth, layered motion with lower skin-friction drag, whereas turbulent flow exhibits enhanced mixing and momentum transfer, typically increasing skin-friction drag but also delaying separation.

Modeling hierarchy: DNS, LES, and RANS Resolving all turbulent scales requires *Direct Numerical Simulation* (DNS), which is prohibitively expensive at engineering Reynolds numbers. *Large–Eddy Simulation* (LES) resolves the large turbulent structures and models the subgrid scales, but remains costly for routine design. *Reynolds–Averaged Navier–Stokes* (RANS) solves for the mean flow and models the effect of all turbulent fluctuations, offering a practical compromise for many aerodynamic applications.

RANS closure and turbulence modeling In RANS, the Reynolds decomposition introduces the Reynolds stresses that must be modeled. Under the Boussinesq hypothesis, these stresses are related to a turbulent (eddy) viscosity μ_t (or $\nu_t = \mu_t/\rho$) [9]. In this work, the turbulence closure depends on the application: external aerodynamics employs the Spalart–Allmaras (SA) model [22], whereas the internal-flow cases use a $k - \omega\ SST$ model coupled with the $\gamma - \tilde{Re}_{\theta t}$ transition model.

1.2 Artificial Intelligence and Machine Learning

Artificial Intelligence (AI) broadly denotes computational methods that perform tasks requiring human-like perception, reasoning, or decision-making. Within AI, *Machine Learning* (ML) focuses on models that learn patterns from data and improve with experience, rather than relying on hand-crafted rules [3, 2]. In most engineering applications, ML is used in a *supervised learning* setting, where a model learns a mapping from input features to target quantities using labeled data. Model parameters are optimized by minimizing a loss function that measures the discrepancy between predictions and reference data, typically using gradient-based algorithms.

Recent advances in *Deep Learning* (DL) have enabled the construction of highly expressive models composed of multiple nonlinear layers. Deep neural networks can approximate complex, high-dimensional relationships and have demonstrated remarkable success in fields such as computer vision, speech recognition, and scientific computing. Their flexibility makes them attractive for modeling nonlinear physical processes that are difficult to describe analytically.

1.3 AI in CFD

In recent years, AI and ML techniques have been increasingly adopted in CFD to enhance predictive accuracy and reduce computational cost, motivated by the high computational expense of resolving complex flow physics with traditional numerical solvers.

One prominent application of machine learning in CFD is the acceleration of high-fidelity simulations through data-driven surrogate models. In this context, Convolutional Neural Networks (CNNs) have been employed to reconstruct high-resolution flow fields from coarse-grid simulations, a process often referred to as flow field super-resolution. By learning a nonlinear mapping between low- and high-resolution solutions, these models can recover fine-scale turbulent structures while significantly reducing computational cost [19].

Beyond purely data-driven surrogates, hybrid approaches have been proposed that couple neural networks with traditional iterative solvers. In this setting, a CNN is used to predict the converged flow field from an intermediate solver state, thereby reducing the number of iterations required to reach convergence while retaining the numerical robustness of the underlying CFD method [23].

While many CNN-based approaches rely on structured grids, industrial CFD simulations typically employ unstructured meshes to accurately represent complex geometries. To address this limitation, Graph Neural Networks (GNNs) have been proposed as surrogate models that operate directly on mesh-based representations. In particular, Chen et al. [4] introduce a Graph Convolutional Neural Network (GCNN) that operates on body-fitted triangular meshes to predict steady laminar flow fields around two-dimensional obstacles. The network takes as input the spatial coordinates of the mesh nodes together with a geometric encoding of the obstacle boundary, and outputs the velocity components and pressure at each node of the mesh.

Extending graph-based learning to unsteady flows, Han et al. [16] propose a data-driven simulator for unsteady fluid dynamics on unstructured meshes that combines GNNs with a transformer-based temporal attention mechanism. The flow field is first represented on a graph corresponding to the finite-volume mesh, where nodes represent mesh cells and edges encode neighborhood connectivity. A GNN is used to locally aggregate information and compress the high-dimensional flow state into a reduced latent representation defined on a small set of pivotal mesh nodes, enabling stable long-term predictions of complex flow dynamics.

1.4 Thesis Aim and Scope

The aim of this thesis is to investigate whether Graph Neural Networks (GNNs), which exploit mesh connectivity to incorporate neighborhood information, offer advantages over pointwise models that treat mesh nodes as independent (e.g., Fully Connected Neural Networks) for CFD surrogate modeling. The study considers two cases: external aerodynamics around an isolated airfoil, and internal aerodynamics around a turbine-blade airfoil. In both cases, the learning models act as data-driven surrogates for turbulence closure by predicting the turbulent viscosity field from geometric and flow-related input features. The geometry is varied across samples while the flow conditions are kept fixed.

As a pointwise baseline, the FCNN architecture is adopted from the PhD thesis of [20], providing an established reference model and enabling a consistent comparison with the proposed graph-based approaches. The comparison focuses on (i) predictive accuracy and spatial error characteristics, (ii) performance when the learned model is coupled into the CFD solver. In addition, preliminary tests are conducted to validate the end-to-end pipeline (data preparation, graph construction, training, and model selection) before applying the methodology to the CFD cases.

1.5 Thesis Organization

Chapter 2: Machine Learning and Neural Networks. This chapter introduces the fundamental concepts required for the remainder of the thesis, including supervised learning, artificial neurons and activation functions, neural network architectures, and the training workflow (forward propagation, loss functions, and backpropagation). The challenges of neural networks are presented, and the need for graph-based models is highlighted. In addition, the evolutionary optimization framework used to tune neural-network architectures is described, including the

Metamodel-Assisted Evolutionary Algorithm (MAEA) strategy and the EASY software adopted in this study.

Chapter 3: Graph Neural Networks. Graph representations and their matrix formulations are introduced, followed by the message-passing paradigm and a discussion of the advantages and challenges of GNNs in scientific computing. The specific GNN architectures employed in this work are presented, together with the general software framework used for their implementation.

Chapter 4: Preliminary Tests: Analytical Function Regression This chapter introduces a controlled benchmark to validate the complete learning pipeline, from graph construction to model training and evaluation. FCNN and GNN models are compared on two-variable function regression to study the influence of mesh resolution. An architecture optimization study is then carried out for both models (FCNN and GNN), enabling a consistent performance comparison.

Chapter 5: NACA4318 Isolated Airfoil. This chapter investigates the implementation of FCNN and GNN models as surrogates for turbulence closure in an external-aerodynamics case: flow around an isolated NACA4318 airfoil. The workflow comprises geometry generation, mesh construction, and the preparation of training and validation datasets derived from RANS–SA solutions. The model architectures are optimized and their predictive accuracy is assessed (offline). The trained models are then coupled into the CFD solver—replacing the turbulence model—and their performance is evaluated in an online setting.

Chapter 6: LS89 Turbine Blade Airfoil. This chapter extends the methodology to a turbine-blade airfoil configuration and discusses the specific aspects of dataset preparation for this geometry. The same training, offline evaluation, and online coupling procedure described in Chapter 6 is applied to the optimized FCNN and GNN models.

Chapter 7: Conclusions and Outlook. The final chapter summarizes the main findings of the thesis, discusses the limitations of the proposed approach, and outlines potential directions for future research.

Chapter 2

Machine Learning and Neural Networks

2.1 Introduction to ML

ML is a subset of AI that focuses on imitating the way humans learn. It improves performance through experience, using data-driven algorithms to analyze information, identify patterns, and make predictions. These statistical algorithms are particularly valuable in situations where designing explicit, rule-based algorithms would be highly complex or impractical [3],[2].

The primary goal of an ML model is to handle unseen data effectively, drawing on knowledge gained during training. This process can be viewed as a black box model: the exact internal transformations from input to output are not explicitly designed by humans but instead shaped by the data itself [2].

2.2 Types of Learning

ML can be divided into three main categories. Their differences lie in the type of data used and the kind of feedback provided to guide the training process [2]:

- **Supervised Learning**: In supervised learning, the data are labeled, meaning input–output pairs are provided. The model generates predictions, and the error (loss) is calculated as the difference between the predicted and the true values. The model then updates its parameters to minimize this error.
- **Unsupervised Learning**: In unsupervised learning, the data are unlabeled. The model receives inputs without corresponding outputs, aiming to discover hidden patterns, groupings, or relationships within the data.

- **Reinforcement Learning**: In reinforcement learning, the model learns by interacting with an environment and receiving rewards or penalties. The goal is to learn an action strategy that maximizes cumulative reward over time.

In this thesis, supervised learning is used to map geometric and flow-related inputs to the turbulent viscosity field μ_t. To perform this mapping, neural network models are employed, which are constructed from interconnected artificial neurons.

2.3 Artificial Neurons

2.3.1 Definition

Artificial neurons are the basic components of modern neural networks (NNs) and are inspired by biological neurons in the nervous system, which transmit information through electrical and chemical signals. A biological neuron consists of dendrites that receive signals, a cell body that processes them, and an axon that transmits the output to other neurons via synapses [21].

Similarly, an artificial neuron receives multiple inputs, computes a weighted summation, and applies an activation function to produce an output. Mathematically, this operation can be expressed as

$$h = g\left(\sum_{i=0}^{m} w_i x_i + b\right), \tag{2.1}$$

where x_i are the input features, w_i the corresponding weights, b a bias term and g the activation function.

This mechanism enables neural networks to model complex and nonlinear relationships between inputs and outputs. When artificial neurons are arranged in multiple layers, including input, hidden, and output layers, they form deep neural networks with high representational capacity [2, 3]. Figure 2.1 illustrates the analogy between a biological neuron and an artificial neural network.

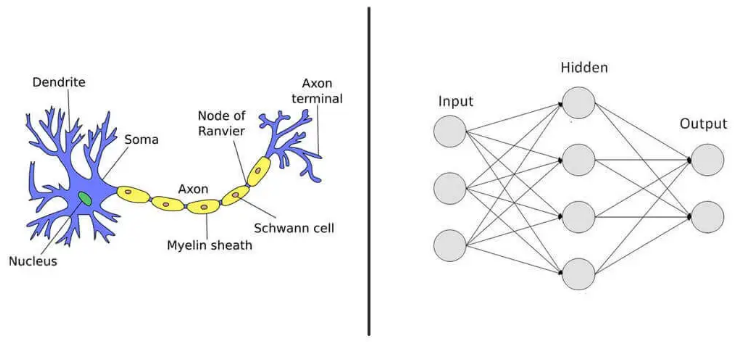


Figure 2.1: *Biological neuron and artificial neural network, adapted from [5].*

2.3.2 Activation functions

While the weighted summation defines a linear mapping, the activation function introduces non-linearity into the model, enabling neural networks to approximate complex functions. The activation function is applied to the weighted sum of the inputs plus the bias, producing the final output, as shown in Equation 2.1 [2, 3].

The activation functions employed in this thesis are (Figure 2.2):

- Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

  The sigmoid maps any input to a value between 0 and 1, making it well-suited for binary classification. It produces a smooth, continuous output and is differentiable [7, 2, 3].

- Hyperbolic Tangent (tanh)

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

  The tanh function outputs values in the range $[-1, 1]$, offering a zero-centered distribution. This often leads to faster convergence during training compared to the sigmoid [7, 2, 3].

- Rectified Linear Unit (ReLU)

$$ReLU(x) = \max(0, x) \tag{2.4}$$

  A key characteristic of the ReLU function is that only certain neurons are activated. When the output is less than zero, the ReLU function returns zero, effectively deactivating the neuron. This makes the function more computationally efficient than sigmoid and tanh. However, it also introduces a drawback: if a neuron remains deactivated for an extended period, it may become permanently inactive, leading to the dying ReLU problem [7, 2, 3].

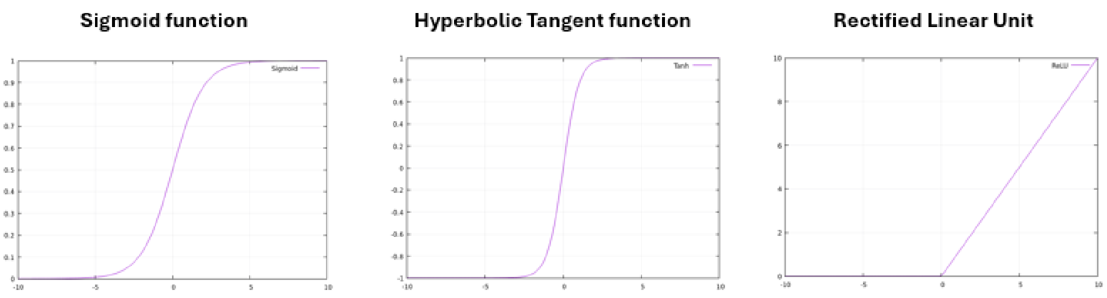


Figure 2.2: *Diagrams of the activation functions : Sigmoid , Hyperbolic Tangent , Rectified Linear Unit*

2.4 Building Neural Networks

2.4.1 Definition of Neural Networks

Neural networks are capable of processing information and learning from data to perform various tasks such as classification, prediction, and pattern recognition [2, 3].

The key components of a neural network are the following [2, 3]:

- Neurons: basic units that receive and process input information.
- Connections: links between neurons that allow the transfer of information.
- Weights and biases: parameters that determine the importance of the information passing through the network.
- Layers: groups of neurons organized to perform computations; neurons within the same layer are not connected.
- Learning mechanism: the process of adjusting weights and biases based on data to improve performance.

2.4.2 Layers

A layer is a group of neurons in a neural network. Neurons within a layer are not directly connected but connect to neurons in the subsequent layer via weighted connections. When every neuron in a layer is connected to all neurons in the next layer, the network is referred to as a Fully Connected Neural Network (FCNN). In this thesis, FCNNs are employed and evaluated against alternative models. An FCNN typically consists of input, hidden, and output layers (Figure 2.3) [2, 3].

Input layer: represents the features of the input data. Each neuron corresponds to a feature in the dataset. In this thesis, the input is defined per mesh node: each node is associated with a feature vector containing geometrical and flow-related quantities. Consequently, each neuron in the input layer corresponds to one component of this node-wise feature vector.

Hidden layer(s): process the weighted information from the input layer. A network may contain one or multiple hidden layers. For prediction tasks such as CFD field reconstruction, multiple hidden layers are typically beneficial due to the high complexity and nonlinearity of the target quantities.

Output layer: produces the final results of the network. In this thesis, each node-wise input feature vector is mapped to a single scalar output; thus, the output layer contains one neuron that predicts the turbulent viscosity associated with the corresponding mesh node.

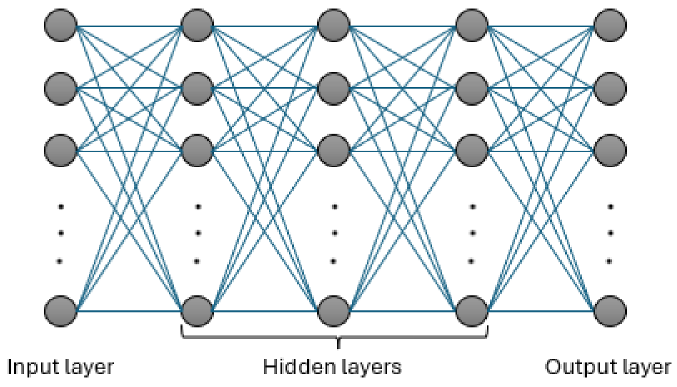


Figure 2.3: *Types of layers in neural networks*

In CFD applications, an FCNN processes each mesh node through the same learned mapping from local input features to the target quantity. An alternative would be to provide the entire field (i.e., all mesh nodes) as a single input vector; however, this leads to an extremely large number of trainable parameters and is not scalable for high-resolution CFD meshes. Consequently, a node-wise FCNN treats nodes independently and cannot capture mesh-based spatial correlations, which are essential in CFD.

2.5 Working principle of Neural Networks

Two main processes take place in a neural network: the feedforward pass and backpropagation [17, 2].

Feedforward Pass
In the feedforward pass, the input data flows from the input layer through the hidden layers and finally to the output layer (Figure 2.4). During this phase, a linear transformation is applied using the weights and biases introduced earlier. For a given neuron, the computation is expressed as follows.

$$h = w_1 h_1 + w_2 h_2 + \cdots + w_n h_n + b \tag{2.5}$$

where:

- h is the value of the current neuron,
- $h_1, ..., h_n$ are the outputs of the neurons in the previous layer,
- $w_1, ..., w_n$ are the weights,
- b is the bias.

The resulting value h is then passed through an activation function.

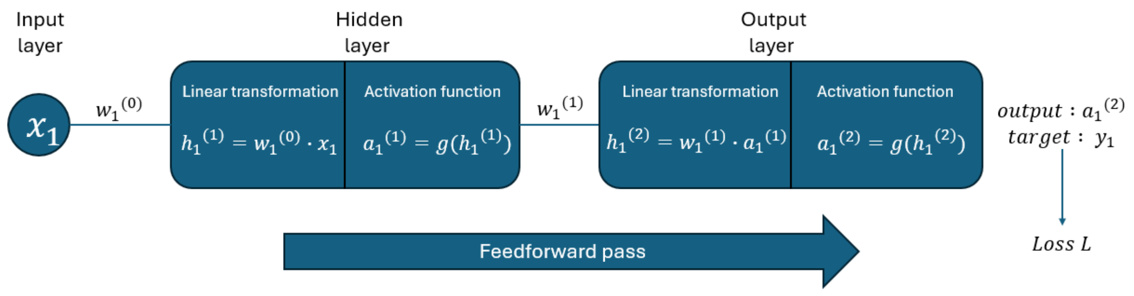


Figure 2.4: *Feedforward pass representation*

Backpropagation

After the forward pass, the network produces an output, and a loss is calculated. The loss measures the discrepancy between the predicted value and the true value. The next step is the calculation of the gradients, where the gradients of the loss function with respect to the weights and biases are computed using the chain rule of calculus (Figure 2.5).

These gradients are then used to update the weights and biases through an optimization algorithm, typically a variant of gradient descent. The update rule is given by:

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial L}{\partial w} \tag{2.6}$$

$$b_{\text{new}} = b_{\text{old}} - \eta \cdot \frac{\partial L}{\partial b} \tag{2.7}$$

where:

- w and b are the weights and biases,
- L is the loss function,
- η is the learning rate, a hyperparameter that controls the step size of the updates.

By iteratively updating the weights and biases in the negative direction of the gradients the network gradually minimizes the loss.

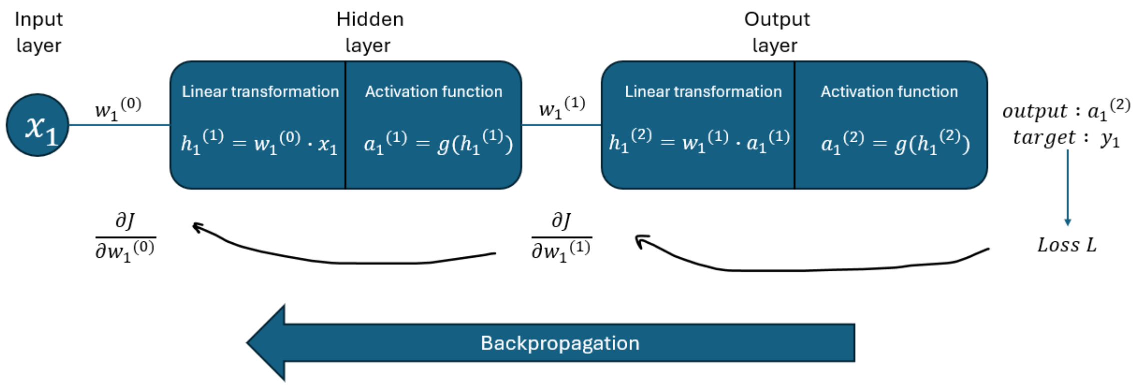

Input layer
Hidden layer
Output layer
x_1
$w_1^{(0)}$
Linear transformation
$h_1^{(1)} = w_1^{(0)} \cdot x_1$
Activation function
$a_1^{(1)} = g(h_1^{(1)})$
$w_1^{(1)}$
Linear transformation
$h_1^{(2)} = w_1^{(1)} \cdot a_1^{(1)}$
Activation function
$a_1^{(2)} = g(h_1^{(2)})$
$output : a_1^{(2)}$
$target : y_1$
$Loss\ L$
$\frac{\partial J}{\partial w_1^{(0)}}$
$\frac{\partial J}{\partial w_1^{(1)}}$
Backpropagation

Figure 2.5: *Backpropagation*

2.6 Loss functions

The discrepancy between the predicted and true values is measured by a function called the loss function, which is minimized by adjusting the model's weights [8, 2, 3].

For regression tasks, two commonly used loss functions are:

- **Mean Squared Error (MSE)**: Mean Squared Error is defined as the average of the squared differences between the actual and predicted values:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{2.8}$$

  where N is the number of data points, y_i is the actual value, and $\hat{y}_i$ is the predicted value. Taking the mean ensures that the loss function remains independent of the number of data points in the training set, while squaring the differences penalizes larger deviations from the target more heavily.

- **Mean Absolute Error (MAE)**: Mean Absolute Error is defined as the average of the absolute differences between the actual and predicted values:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|, \tag{2.9}$$

  where N is the number of data points, y_i is the actual value, and $\hat{y}_i$ is the predicted value. Taking the mean again ensures independence from the number of data points. The absolute value guarantees that all errors are treated equally, without disproportionately penalizing larger ones. In this thesis, the MAE is adopted for training, as it is less sensitive to outliers than MSE.

2.7 Challenges in Neural Networks

Neural networks often face several challenges that affect their effectiveness and efficiency [2, 3, 26]:

- High Data Requirements and Computational Cost
  Neural networks typically require large amounts of data and significant computational resources to train. This is a trade-off: while they are capable of modeling highly complex and nonlinear problems that are beyond the reach of simpler models, their expressive power comes at the cost of longer training times and higher memory usage. This is particularly relevant for CFD applications, such as those considered in this thesis, where training data consist of high-dimensional flow fields obtained from numerical simulations. Generating these labeled datasets is computationally expensive, and increasing mesh resolution leads to larger input/output representations, resulting in significant memory requirements and longer training times.

- Overfitting
  A major difficulty in training neural networks is avoiding overfitting. Overfitting occurs when the model learns the training data too well, including its noise, and fails to generalize to unseen data. Factors that can contribute to overfitting include the size of the dataset, the depth of the network, and the number of training epochs. In this thesis, overfitting was assessed by comparing training and validation loss curves. For the final models, the validation loss followed the training loss without a persistent divergence, indicating satisfactory generalization on the available dataset.

2.8 Transition to GNNs

FCNNs provide a flexible baseline for learning nonlinear mappings from local inputs to targets. However, when applied node-wise on CFD meshes, they ignore mesh connectivity and cannot capture spatial interactions between neighbouring nodes. Since CFD fields are strongly coupled in space through the governing equations and mesh topology, incorporating neighbourhood information is essential. This motivates the use of GNNs, which explicitly propagate information between connected mesh nodes.

2.9 NN Optimization with EASY

Beyond the choice between FCNN and GNN architectures, the predictive performance of NNs is sensitive to architectural design choices. To systematically explore this design space, an evolutionary optimization strategy is adopted, as described in this section.

Evolutionary algorithms (EAs) are stochastic optimization methods inspired by natural evolution. They iteratively improve a population of candidate solutions through selection, reproduction, and mutation. In this work, an EA is used to explore neural-network architectures in a discrete design space.

An optimization problem is defined by a set of *design variables* and an *objective function*. Here, the objective is to minimize the prediction loss of the neural networks, while the design variables describe the architecture (e.g., number of layers, number of neurons per layer, and activation functions).

A key limitation of EAs is their computational cost, since they typically require many objective-function evaluations. In the present application, each evaluation corresponds to training and validating a candidate network. To reduce the number of costly evaluations, *Metamodel-Assisted Evolutionary Algorithms (MAEAs)* incorporate surrogate models (metamodels) that approximate the objective function. After an initial set of architectures is evaluated with the true objective, a metamodel is trained to predict the loss of new candidates. The most promising candidates are then re-evaluated with the true objective to maintain reliability [10].

The *Evolutionary Algorithms SYstem (EASY)* software, developed by the PCOpt group at NTUA, is used as the optimization framework. EASY supports single- and multi-objective evolutionary optimization and provides metamodel-assisted strategies, which are leveraged here for neural-network architecture search [11].

Chapter 3

Graph Neural Networks

3.1 Introduction to GNNs

GNNs are a class of neural networks designed to operate on data represented as graphs. Graphs naturally arise in various real-world applications, including CFD meshes. A key advantage of GNNs is their ability to exploit relationships between nodes, allowing them to learn from both node features and graph connectivity. This is achieved through message passing, which is particularly beneficial for CFD applications on unstructured meshes, where connectivity is irregular, and neighborhood size varies across the domain [25].

3.1.1 Graphs

A **graph** is a data structure commonly denoted as $G = (V, E)$, where V is the set of **nodes** (or **vertices**) and $E \subseteq V \times V$ is the set of **edges** (or **links**) that define the relationships between nodes (Figure 3.1) [25].

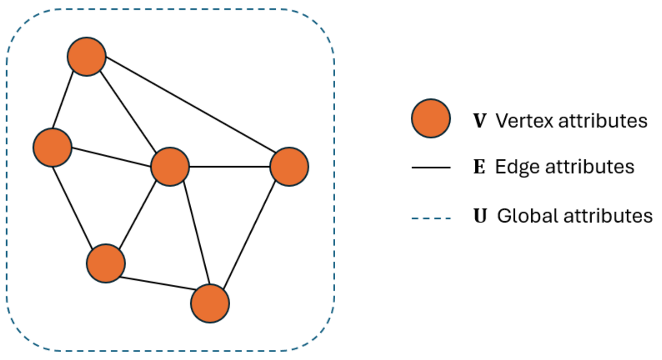


Figure 3.1: *Representation of the three types of attributes in a graph*

In more expressive formulations—especially in the context of machine learning—graphs can include [25]:

- **Node attributes** $v_i \in \mathbb{R}^F$, representing the feature vector (or *embedding*) of node i. In CFD applications, each node typically corresponds to a mesh vertex, and v_i may include geometric information (e.g., coordinates (x, y), wall distance) as well as local flow quantities (e.g., velocity components, pressure).
- **Edge attributes** $e_{ij} \in \mathbb{R}^S$, containing features of the edge from node i to node j. For CFD meshes, an edge represents a neighborhood relation between two mesh entities, and e_{ij} can encode geometric relations such as the Euclidean distance and the relative orientation (e.g., angle) between them.
- **Global attributes** $u \in \mathbb{R}^G$, which describe properties of the graph as a whole. In CFD, these correspond to case-level parameters that are shared by all nodes and edges, such as Reynolds number, Mach number, or angle of attack.

Graphs can be categorized into [25]:

- **Undirected graphs**, where edges have no orientation, i.e., if $(i, j) \in E$, then $(j, i) \in E$ as well. In this thesis, mesh connectivity is treated as undirected, as information exchange between neighboring mesh nodes is not naturally directional.
- **Directed graphs**, where each edge has a specific direction from node i to node j, representing asymmetric relationships.

3.1.2 Types of Graph Prediction Tasks

GNNs can be applied to a variety of learning tasks depending on the type of the prediction. The three most common categories are [25, 14]:

- **Node-level tasks**: The objective is to predict a label or a property for each individual node in the graph. In this thesis, the objective is to predict the turbulent viscosity at each mesh node (Figure 3.2).

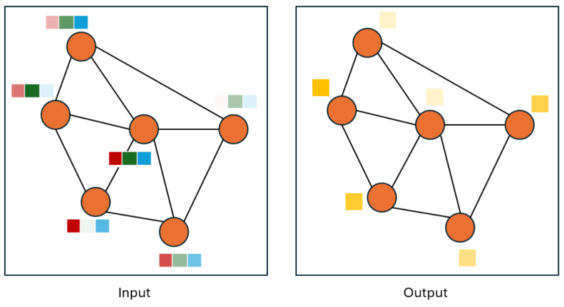


Figure 3.2: *Node-level task representation*

- **Edge-level tasks**: These tasks involve predicting properties of edges or determining the presence or absence of an edge between two nodes (Figure 3.3).

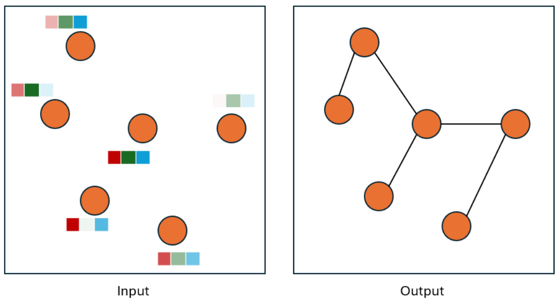


Figure 3.3: *Edge-level task representation*

- **Graph-level tasks**: The prediction targets the graph as a whole (Figure 3.4).

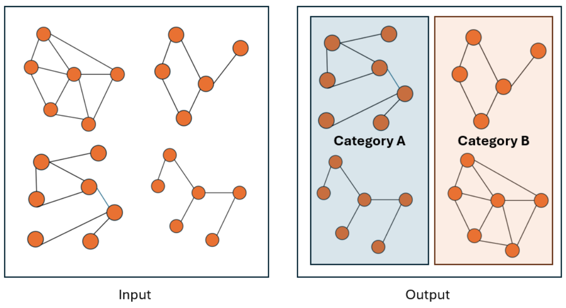


Figure 3.4: *Graph-level task representation*

3.1.3 Representation of Graphs in ML

In ML, data is usually represented as rectangular arrays (e.g., matrices or tensors). To use graphs within ML models, we must represent their components—nodes, edges, and relationships—in matrix form.

A graph encodes the following types of information (Figure 3.5) [25, 14]:

- **Node feature matrix** $X \in \mathbb{R}^{N \times F}$: This matrix stores features for each node in the graph, where N is the number of nodes and F is the number of features per node. In the applications considered in this thesis, N and F vary across cases: $N = 29848$, $F = 8$ for the airfoil mesh and $N = 49986$, $F = 9$ for the turbine blade airfoil mesh.

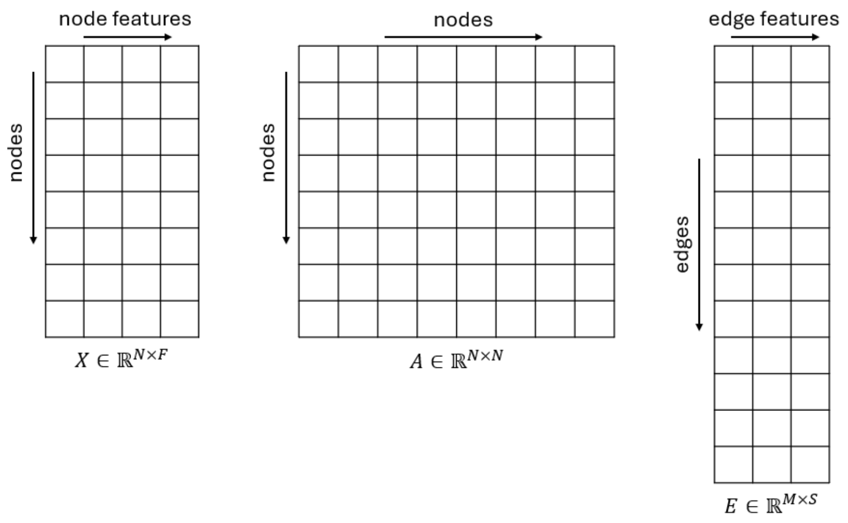

node features
nodes
$X \in \mathbb{R}^{N\times F}$
nodes
nodes
$A \in \mathbb{R}^{N\times N}$
edge features
edges
$E \in \mathbb{R}^{M\times S}$

Figure 3.5: *Representation of matrices X, A, E*

- **Adjacency matrix** $A \in \mathbb{R}^{N\times N}$: This matrix represents the connections between nodes. For unweighted graphs, $A_{ij} = 1$ indicates an edge between node i and node j, and $A_{ij} = 0$ otherwise. In the applications considered in this thesis, the airfoil case leads to 29848×29848 matrix, while the turbine blade case leads to 49986×49986 matrix. Such matrices become prohibitively memory-intensive if stored in dense form; therefore, efficient sparse representations and memory-aware implementations are essential for training and inference on large CFD meshes.

- **Edge feature matrix** $E \in \mathbb{R}^{M\times S}$: This matrix captures attributes of the edges, where M is the number of edges and S is the number of features per edge. In this thesis, the edge feature is the Euclidean distance between two connected mesh nodes; therefore, $S = 1$. In the considered applications, the airfoil mesh contains $M = 118664$ edges, whereas the turbine blade airfoil mesh contains $M = 394896$ edges. Note that M is reported as twice the number of unique neighbor pairs because each undirected connection $\{i, j\}$ is represented by two directed edges (i, j) and (j, i).

- **Global feature vector** $u \in \mathbb{R}^{G}$: Global features summarize information about the entire graph, such as overall density or category, where G is the number of global features. These features are not included in our work. They could be useful when flow conditions vary.

These representations allow GNNs to process and learn from graph data using standard ML frameworks.

3.1.4 Message Passing

Most modern GNN architectures follow the **message-passing paradigm**, which proceeds in iterative steps. At each layer (or time step), every node receives information (messages) from its neighbors, aggregates these messages, and updates its own representation accordingly [25, 12, 15].

Formally, at time step t, the representation $h_i^{(t)}$ of node i is updated through two main operations (Figure 3.6) [12, 15]:

1. **Aggregation**:
$$\mathbf{m_i^{(t+1)}} = \sum_{j\in\mathcal{N}(i)} \mathbf{M_t}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e_{ij}}) \tag{3.1}$$

2. **Update**:
$$\mathbf{h_i^{(t+1)}} = \mathbf{U_t}\left(\mathbf{h_i^{(t)}},\, \mathbf{m_i^{(t+1)}}\right) \tag{3.2}$$

Where:

- $\mathcal{N}(i)$ is the set of neighboring nodes of node i.
- $\mathbf{M_t}$ is the **message function**, which computes information sent from neighbor j to node i. It can be a neural network, a linear layer, or any differentiable function.
- $\mathbf{U_t}$ is the **update function**, responsible for updating the node's representation using its previous state and the aggregated message.
- $\mathbf{e_{ij}}$ represents the **edge features** between nodes i and j, such as distances, angles, or physical quantities.

The aggregation operator (commonly sum, mean, or max) must be **permutation-invariant** to ensure the representation does not depend on the order of neighbors.

After several message-passing steps, each node obtains a final representation $\mathbf{h_i^{(T)}}$ [12]. This neighborhood-based formulation aligns better with the local coupling present in the governing equations of CFD than node-wise independent models, such as the FCNN model described in Section 2.4.2.

Moreover, a key practical distinction is that message passing performs computations along edges (messages are computed for each edge and then aggregated at each node). Therefore, compared to an FCNN—which processes nodes independently—the computational cost of a GNN layer scales not only with the number of nodes N, but also with the number of edges M (as defined in Section 3.1.3).

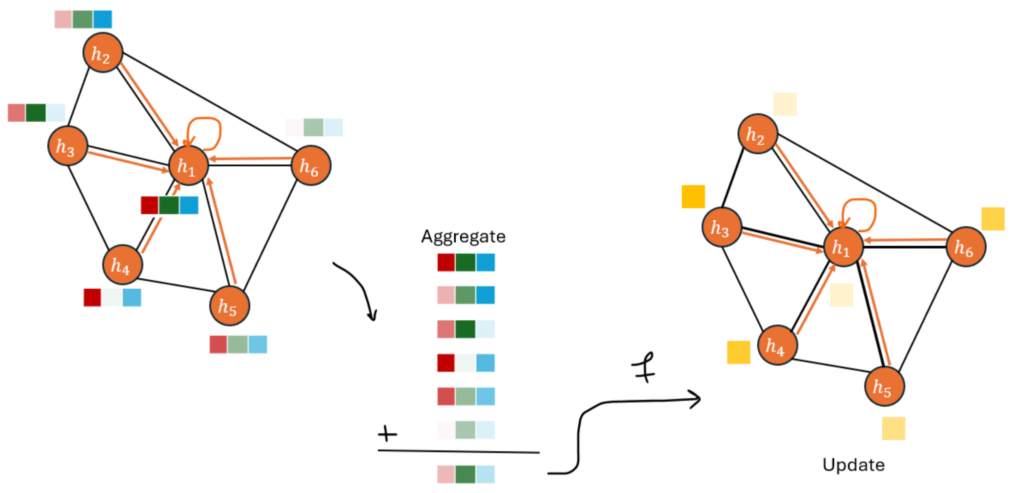


Figure 3.6: *Illustration of the message passing mechanism with aggregation and update steps*

3.1.5 Advantages of GNNs

GNNs offer several advantages that make them particularly well-suited for tasks involving graph-structured and relational data [25].

Ability to model relationships
A key advantage of GNNs lies in their ability to **leverage the connections between nodes**. By aggregating and propagating information through edges, GNNs capture dependencies and interactions that are crucial in many domains, especially in CFD. This capability represents a fundamental advantage over FCNNs, which typically process each node independently.

Flexibility with graph size and structure
The message-passing mechanism enables a single model to operate on meshes with different numbers of nodes and edges during training and inference. Since each node aggregates information from a variable-size neighborhood, GNNs are well suited to unstructured CFD meshes, where connectivity is irregular.

Permutation invariance
Since the aggregation functions used in GNNs (e.g., sum, mean, max) are permutation-invariant, the learned representations do not depend on the ordering of nodes or edges. This property is particularly important for unstructured CFD meshes, where node indexing is arbitrary.

Expressiveness
Finally, GNNs provide a way to combine both **node-level features** and **structural information**. In particular, they can incorporate edge attributes between neighboring mesh nodes, enabling a more expressive representation.

3.1.6 GNN Difficulties

While GNNs are powerful tools for learning on graph-structured data, they also come with computational challenges and limitations. Two of the most prominent issues are memory usage and over-smoothing.

Memory usage
One of the primary challenges in scaling GNNs lies in their **memory requirements**. In large-scale graphs, such as CFD meshes with high resolution, the storage and computation of node features, edge features, and message-passing operations become prohibitively expensive (Subsection 3.1.3).

A particular bottleneck is the representation of graph connectivity through the adjacency matrix $\mathbf{A}$. Fortunately, CFD mesh connectivity is **sparse**, meaning that the majority of entries in $\mathbf{A}$ are zero. Exploiting this sparsity enables the use of **sparse matrix representations**, which provide a reduced memory footprint.

Over-smoothing
Another fundamental limitation is the **over-smoothing problem**. As the number of layers in a GNN increases, repeated message passing causes node embeddings to become more similar to each other. Eventually, the representations of different nodes may converge to nearly identical vectors, regardless of their original features or local neighborhood structure. To mitigate over-smoothing, practical models often limit the number of layers or incorporate techniques such as residual connections and normalization [28]. This is particularly relevant in CFD, where sharp gradients may be smoothed out if the network becomes excessively deep.

3.2 GNN Models Used

This section presents the specific types of GNNs employed in this work, together with a simpler reference model introduced for conceptual clarity. All considered architectures follow the general message-passing formulation introduced in Section 3.1.4, but differ in how neighborhood information is aggregated and incorporated into the node representations [24].

Figure 3.7 illustrates the generic layer-wise propagation mechanism common to all GNN models. Node features are iteratively updated through successive layers, allowing information to propagate across increasingly larger neighborhoods as the network depth increases. While this propagation scheme is shared across models, the specific definition of the aggregation or message function varies between architectures.

Three architectures are considered: GCN (to introduce graph convolution concepts), GAT and MPNN (used in the applications considered in this thesis):

- The **Graph Convolutional Network (GCN)** [18], which aggregates information from neighboring nodes using fixed, normalized weights derived from the graph connectivity.

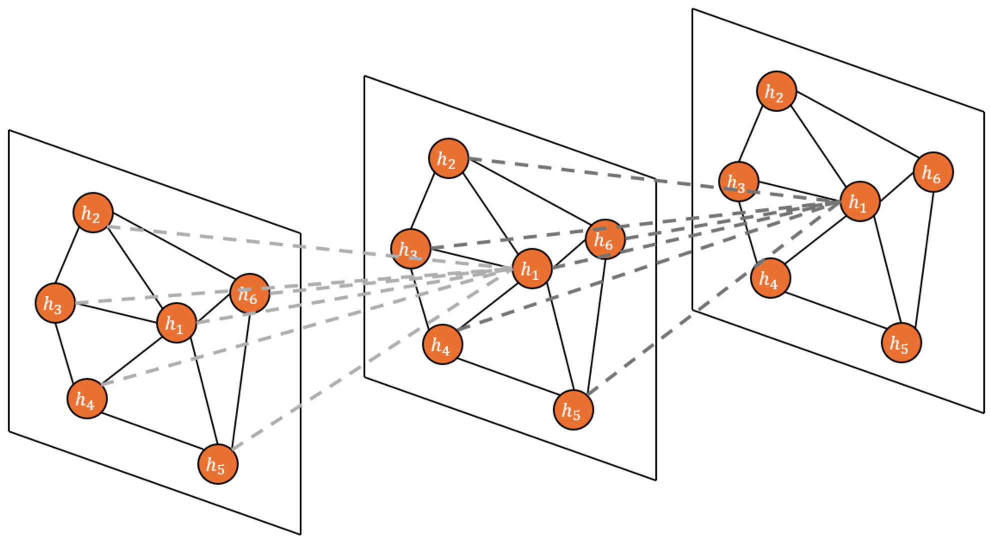


Figure 3.7: *Illustration of layer-wise feature propagation in a GNN. Each layer updates node representations by aggregating information from neighboring nodes. Stacking multiple layers allows information to propagate across larger graph neighborhoods. The specific aggregation mechanism depends on the chosen GNN architecture.*

- The **Graph Attention Network (GAT)** [27], which introduces an attention mechanism to learn adaptive, data-dependent weights for neighboring nodes.
- The **Message Passing Neural Network (MPNN)** [12], which follows the general message-passing framework and is instantiated in this work through specific design choices.

Each of the following subsections details the formulation of these models [6].

3.2.1 Graph Convolutional Network (GCN)

In the GCN formulation, the message function assigns equal importance to all neighboring nodes through fixed, degree-based normalization [18]. For a given node i, the aggregation step at layer t can be written as

$$\mathbf{m}_i^{(t+1)} = \sum_{j \in \mathcal{N}(i)} z_{ij} \, \mathbf{h}_j^{(t)}, \tag{3.3}$$

where

$$z_{ij} = \frac{1}{\sqrt{d_i d_j}} \tag{3.4}$$

denotes the normalization coefficient associated with the edge (i, j), and d_i is the degree of node i (number of neighbors). The normalization coefficient z_{ij} relies on the assumption that connected nodes are likely to share similar labels and reduces the influence of information originating from nodes with a large number of neighbors.

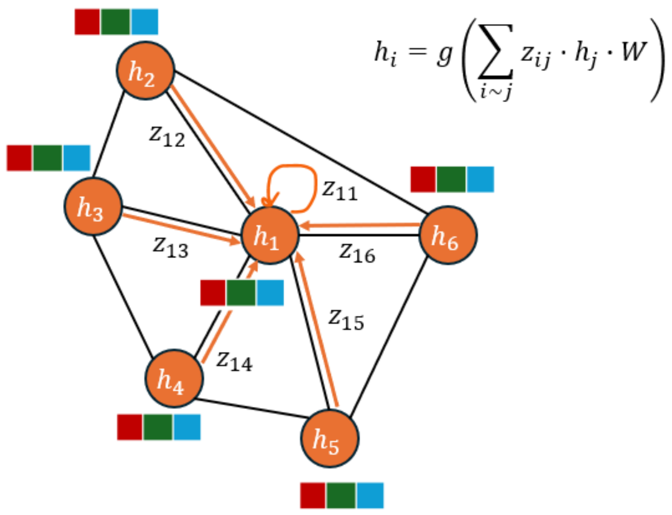


Figure 3.8: *Illustration of a GCN layer. The representation of node i is updated by aggregating feature vectors from its neighbors using fixed, degree-normalized coefficients z_{ij}.*

The aggregated message is then passed through a linear transformation followed by a nonlinear activation function:

$$\mathbf{h}_i^{(t+1)} = g\left(\mathbf{W}^{(t)}\mathbf{m}_i^{(t+1)} + \mathbf{b}^{(t)}\right), \tag{3.5}$$

where $\mathbf{W}^{(t)}$ and $\mathbf{b}^{(t)}$ are the learnable weights and biases, and $g(\cdot)$ denotes an activation function.

In this thesis, GCN is not used in the applications because it assigns equal importance to neighbors, which may limit expressiveness for the present regression task.

3.2.2 Graph Attention Networks (GAT)

Unlike GCNs, which aggregate neighbor features using fixed, degree-based weights, GATs compute *learnable attention coefficients* that allow different neighbors to contribute unequally to the node representation update [27].

At layer t, the aggregated message at node i is computed as a weighted sum of its neighbors' features:

$$\mathbf{m}_i^{(t+1)} = \sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(t)}\mathbf{h}_j^{(t)}, \tag{3.6}$$

where $\alpha_{ij}^{(t)}$ denotes the attention coefficient associated with the edge (i, j).

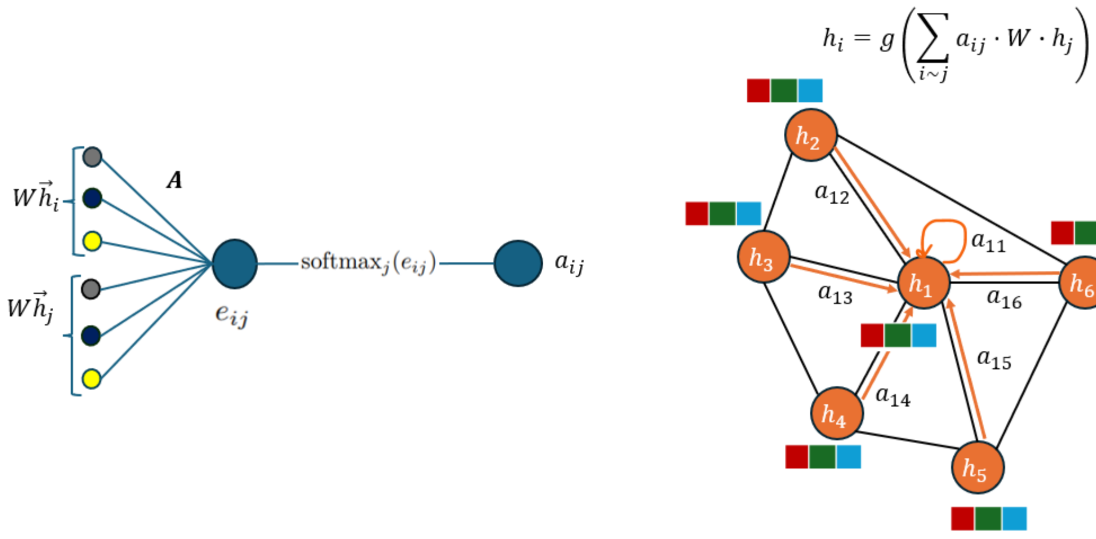


Figure 3.9: *Illustration of a GAT layer. Attention coefficients α_{ij} are computed from transformed node features and normalized using a softmax function. The representation of node i is updated as a weighted sum of its neighbors' features, allowing different neighbors to contribute unequally.*

The attention coefficients are obtained by first computing unnormalized attention scores:

$$e_{ij}^{(t)} = \sigma\left(\mathbf{A}^{(t)}\left[\mathbf{W}^{(t)}\mathbf{h}_i^{(t)} \parallel \mathbf{W}^{(t)}\mathbf{h}_j^{(t)}\right]\right), \tag{3.7}$$

where $\mathbf{W}^{(t)}$ is a shared linear transformation, $\mathbf{A}^{(t)}$ is a learnable attention vector, $\parallel$ denotes concatenation, and $\sigma(\cdot)$ is a nonlinear activation function. The normalized attention coefficients are then computed using a softmax function over the neighborhood of node i:

$$\alpha_{ij}^{(t)} = \frac{\exp(e_{ij}^{(t)})}{\sum_{k\in\mathcal{N}(i)} \exp(e_{ik}^{(t)})}. \tag{3.8}$$

Finally, the node features are updated by applying a linear transformation and a nonlinear activation function to the aggregated message:

$$\mathbf{h}_i^{(t+1)} = \sigma\left(\mathbf{W}^{(t)}\mathbf{m}_i^{(t+1)} + \mathbf{b}^{(t)}\right). \tag{3.9}$$

This model is a primary architecture in this thesis, as the attention mechanism enables adaptive weighting of neighbor contributions during aggregation.

3.2.3 Message Passing Neural Networks (MPNN)

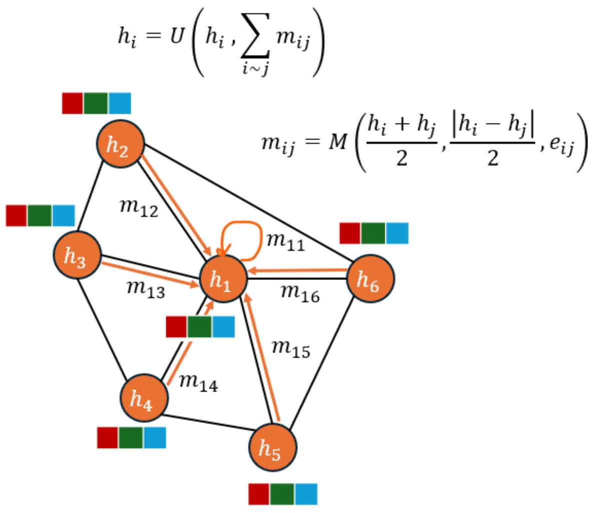
In contrast to fixed architectures such as GCNs or GATs, MPNNs allow explicit definition of the message and update functions, providing greater flexibility in incorporating node and edge features. In this thesis, each MPNN layer parameterizes the *message* and *update* functions using two single fully connected (dense) layers; one dense layer is applied to edge-level computations (to generate messages), and a second dense layer is applied at the node level to update the node embeddings:

For each edge $(j \to i)$, the message function operates on a concatenation of symmetric combinations of node features and edge features, defined as

$$\mathbf{m}_{ij} = \left[\frac{\mathbf{h}_i + \mathbf{h}_j}{2}, \ \frac{|\mathbf{h}_i - \mathbf{h}_j|}{2}, \ \mathbf{e}_{ij} \right], \tag{3.10}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ denote the feature vectors of nodes i and j, respectively, and $\mathbf{e}_{ij}$ represents the edge features, which in this thesis is the Euclidean distance between the nodes.

The resulting message vectors are transformed by a learnable message function implemented as a dense neural layer. Messages from all neighbors are then aggregated using a summation operator, ensuring permutation invariance. The aggregated message is combined with the previous node features and passed through a second dense neural layer, serving as the update function, to obtain the updated node representation.



$h_i = U\left(h_i, \sum_{i\sim j} m_{ij}\right)$
$m_{ij} = M\left(\frac{h_i + h_j}{2}, \frac{|h_i - h_j|}{2}, e_{ij}\right)$
h_1
h_2
h_3
h_4
h_5
h_6
m_{11}
m_{12}
m_{13}
m_{14}
m_{15}
m_{16}

Figure 3.10: *Illustration of the MPNN layer used in this work. For each edge, messages are constructed from node and edge features and aggregated with the current node state to update the node representations.*

3.3 GNN Implementation in Python-Spektral

Python and the Spektral library are used to build and train the GNN models considered in this work. Spektral provides high-level tools that integrate well with TensorFlow and support a wide range of GNN architectures [13, 1].

In Spektral, each sample is stored as a `Graph` object containing the node features $\mathbf{x} \in \mathbb{R}^{n \times f}$, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ (stored in sparse format), optional edge features $\mathbf{e} \in \mathbb{R}^{m \times s}$, and the target values $\mathbf{y}$ (see Section 3.1.3). These `Graph` objects are grouped into a `Dataset`. Once defined, the dataset can be passed to different types of loaders depending on the chosen mode. The loader handles batching, preprocessing, and feeding data into the model during training and evaluation [13, 1].

In this thesis the *disjoint mode* is adopted, as it provides an efficient and flexible representation for datasets consisting of multiple graphs with varying sizes and sparse connectivity.

Disjoint mode. In disjoint mode, multiple graphs are combined into a single disconnected super-graph with a block-diagonal adjacency matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$, where K denotes the total number of nodes across all graphs. No edges exist between nodes belonging to different graphs.

Node features are stacked into a single matrix $\mathbf{X} \in \mathbb{R}^{K \times F}$, and edge features into $\mathbf{E} \in \mathbb{R}^{L \times S}$, where L is the total number of edges. An additional index vector $\mathbf{g} \in \{1, \ldots, G\}^K$ encodes the graph membership of each node.

Disjoint mode allows training on datasets containing graphs of different sizes (e.g., CFD meshes with different resolutions or geometries). Although the datasets used in this thesis employ fixed meshes within each case, the ability to handle variable-size meshes is a useful property for more general CFD workflows.

Chapter 4

Preliminary Tests: Analytical Function Regression

4.1 Introduction and Motivation

Before applying GNNs (see Chapter 3) to the complex case of CFD, a preliminary test is designed based on the prediction of a two-variable polynomial function. This test serves two purposes. First, it enables validation of the end-to-end workflow of transforming a spatial domain into a graph, generating datasets, optimizing architectures, and training NN models. Second, it provides a comparison between FCNNs, (see Section 2.4) and GNNs, highlighting their respective strengths and limitations in a setting where neighborhood relations are not inherently important.

4.2 Problem Setup

The regression task is defined using a polynomial function of two variables:

$$f(x, y) = b_0x^2 + b_1y^2 + b_2yx^3 + b_3xy^3 + b_4x^5, \tag{4.1}$$

where the coefficients $\{b_0, b_1, b_2, b_3, b_4\}$ are sampled from the following ranges:

$$b_0 \in [-4, -1], \quad b_1 \in [1, 4], \quad b_2 \in [-8, -5], \quad b_3 \in [7, 10], \quad b_4 \in [-7, -3].$$

The learning objective is to predict the polynomial values

$$z = f(x, y),$$

given the input coordinates (x, y) and the coefficients $(b_0, \ldots, b_4)$.

4.3 Domain Discretization and Data Generation

The spatial domain is defined as the unit square: $(x, y) \in [0, 1]\times[0, 1]$. This domain is discretized into four triangular meshes of increasing resolution, containing 181, 549, 3327, and 13048 nodes, respectively. The four meshes are illustrated in Figure 4.1, where the mesh vertices correspond to the graph nodes.

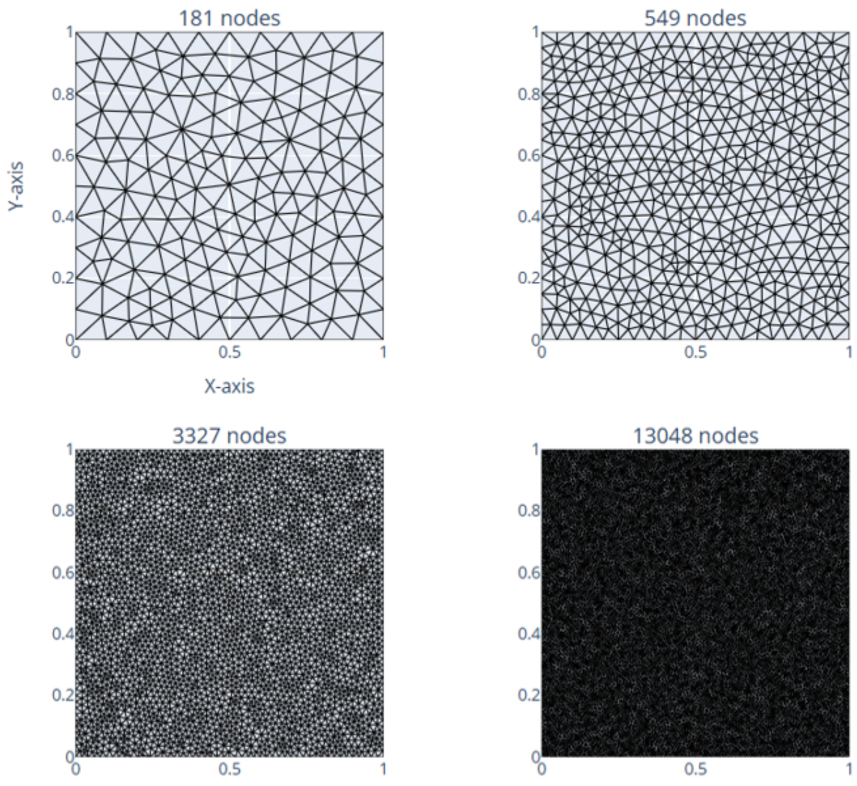


Figure 4.1: *Polynomial Surface Reconstruction: Discretization of the unit square into triangular meshes of increasing resolution (181, 549, 3327, and 13048 nodes).*

For each set of coefficient samples $(b_0, \ldots, b_4)$, the polynomial values $z = f(x, y)$ are computed at all the mesh nodes in the four meshes, providing ground-truth labels. In total, 105 different coefficient sets are generated, resulting in 105 dataset samples. Of these, 100 samples are used for training and 5 for validation. A representative training sample is shown as a contour plot in Figure 4.2, illustrating the type of field that the model aims to predict.

The ranges of the x, y coordinates, the coefficients $(b_0, \ldots, b_4)$, and the polynomial values z differ significantly. To enable effective training, all inputs and outputs are scaled using Min–Max normalization, defined as

$$\hat{F} = \frac{F - F_{\text{min}}}{F_{\text{max}} - F_{\text{min}}},$$

where F denotes the true value, and $F_{\min}$ and $F_{\max}$ are the minimum and maximum values of the corresponding variable. This normalization is applied separately to each variable, ensuring that all features and labels are mapped to the range $[0, 1]$.

For evaluation, the predicted values are denormalized back to the original scale using

$$F = \hat{F} \cdot (F_{\max} - F_{\min}) + F_{\min},$$

so that the error metrics are computed in the physical units of the problem.

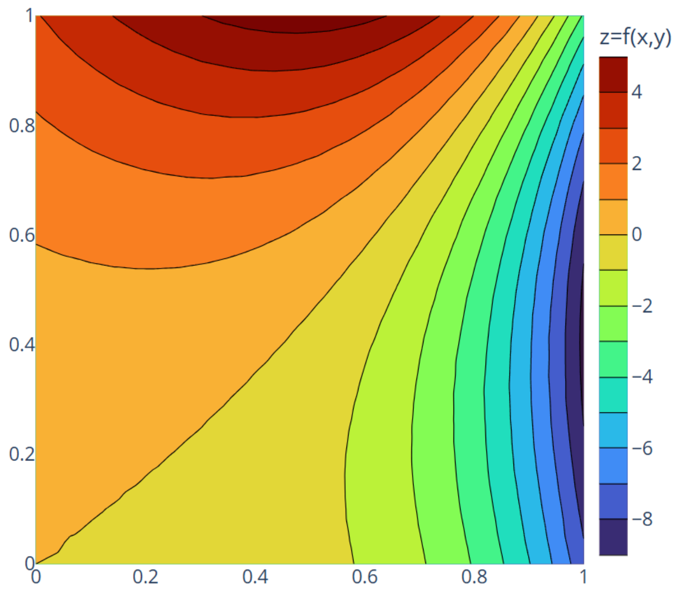


Figure 4.2: *Polynomial Surface Reconstruction: Contour representation of a training sample on the mesh domain.*

4.4 Mesh Selection

The present section examines how mesh resolution influences model performance for polynomial surface reconstruction and compares the behavior of GNNs and FCNNs. The four triangular meshes introduced in Section 4.3, containing 181, 549, 3327, and 13048 nodes, are used throughout this study.

For each model type, a fixed network architecture and training procedure are employed, and the model is trained independently on each mesh resolution. Within a given model, the only varying factor across experiments is the mesh resolution, allowing the effect of mesh refinement on predictive accuracy to be systematically analyzed between GNNs and FCNNs.

4.4.1 GNN: Training and Results

A GAT (see Section 3.2) consisting of six hidden layers with dimensions $[128, 256, 2048, 128, 32, 512]$ is employed. The model is trained for 800 epochs on each of the four meshes (181, 549, 3327, and 13048 nodes). The validation loss is measured using the Mean Absolute Error (MAE), as described in Section 2.6. The validation losses during training are shown in Figure 4.3.

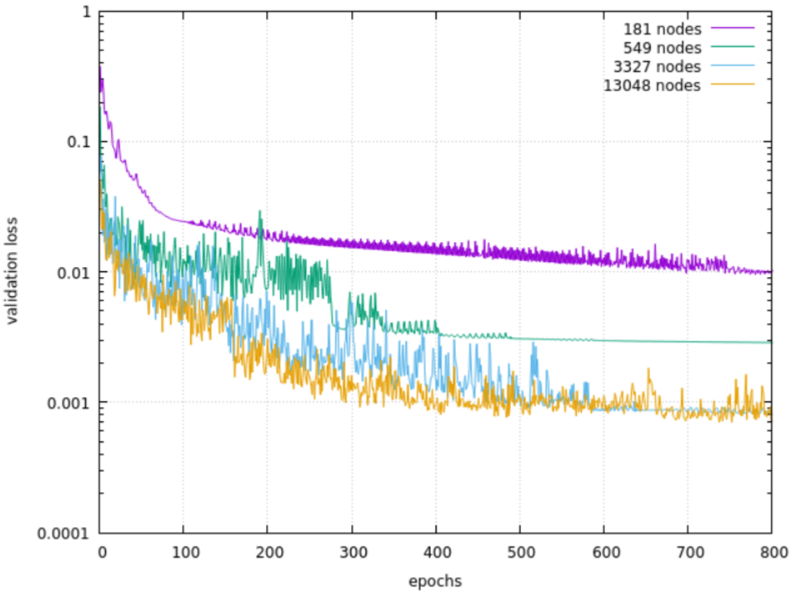


Figure 4.3: *Polynomial surface reconstruction: Validation losses during training of the GAT model on meshes with 181, 549, 3327, and 13048 nodes. The vertical axis is plotted on a logarithmic scale to emphasize differences across resolutions.*

Figure 4.3 shows a sharp improvement when increasing the resolution from 181 to 549 nodes, indicating that the coarsest mesh is insufficient for capturing the polynomial variations. Further refinement to 3327 nodes leads to a substantial reduction in validation loss, while the difference between 3327 and 13048 nodes is minimal. This suggests that convergence with respect to mesh resolution has been reached.

The absolute error distributions shown in Figure 4.4 confirm this trend. The error magnitude decreases consistently as the resolution increases, and the predictions become progressively more accurate.

Dependence on Total Node and Edge Count The GNN model operates by sequentially passing information along the edges of a mesh while updating shared weights. Consequently, higher-resolution meshes yield improved accuracy primarily because they provide a larger number of training nodes and edges. This observation does not imply, however, that coarse meshes are inherently unable to capture the underlying field. To illustrate this, an additional test was conducted using the GAT model. Instead of 100 training meshes and 5 validation meshes, for the 181-node mesh, the setup was expanded to 304 training meshes and 16 validation meshes, corresponding to the same total node count (product of mesh resolution and the number of meshes) as the experiments with the 549-node mesh. As shown in Figure 4.5, the 181-node mesh achieves predictive performance comparable to the 549-node mesh when the total node count is the same.

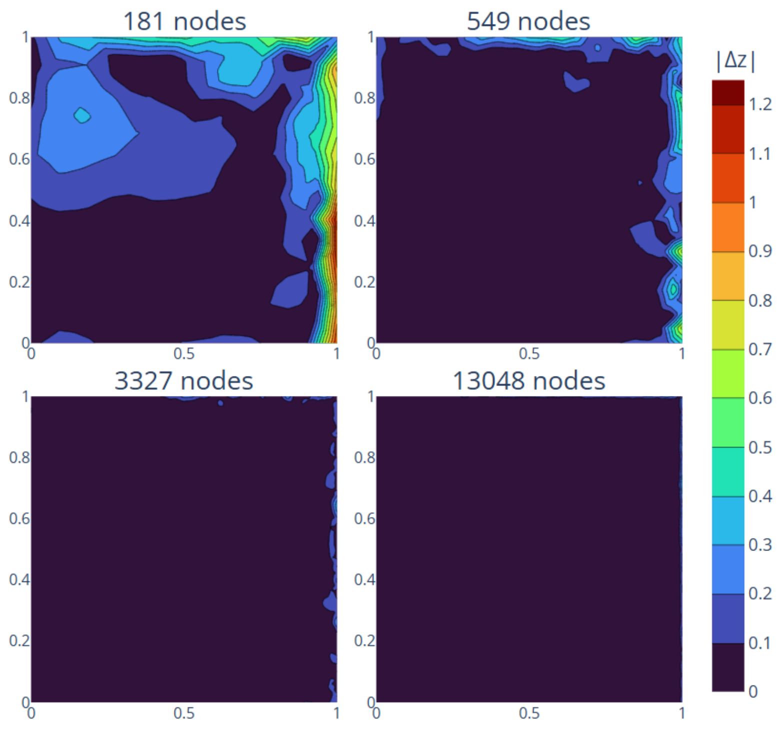


Figure 4.4: *Polynomial surface reconstruction: Absolute error distributions of the GAT model on meshes with 181, 549, 3327, and 13048 nodes.*

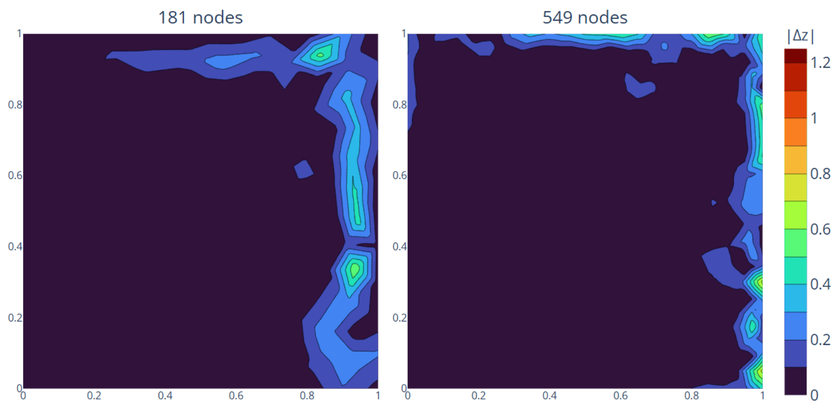


Figure 4.5: *Polynomial surface reconstruction: Comparison of absolute error distributions for meshes with 181 and 549 nodes, adjusted to the same total node count by varying the number of training and validation meshes.*

4.4.2 FCNN: Training and Results

The same polynomial reconstruction experiment is repeated using an FCNN. The network consists of six hidden layers with dimensions $[1024, 1024, 1024, 512, 64, 512]$ and is trained for 800 epochs on each of the four meshes. The validation losses during training are shown in Figure 4.6.

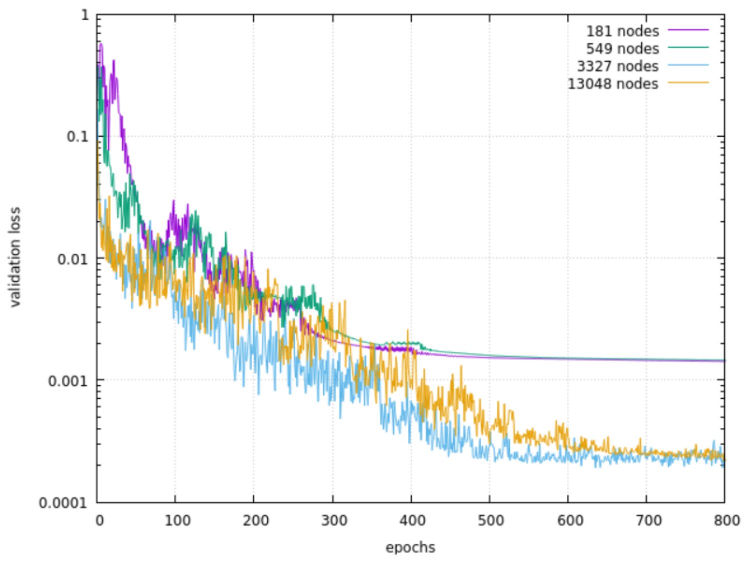


Figure 4.6: *Polynomial surface reconstruction: Validation losses during training of the FCNN on meshes with 181, 549, 3327, and 13048 nodes. The vertical axis is plotted on a logarithmic scale.*

As illustrated in Figure 4.6, the dependence of FCNN performance on mesh resolution differs from that observed for the GNN. The validation losses obtained on the 181-node and 549-node meshes are very similar, indicating that the modest increase in node count does not significantly affect the learning process. A noticeable reduction in validation loss is observed when increasing the resolution to 3327 nodes, while further refinement to 13048 nodes yields no substantial additional improvement.

This behavior is consistent with the node-wise nature of FCNN training. Increasing the mesh resolution primarily affects performance by increasing the total number of training samples. When the increase in node count is sufficiently large, as in the transition from 549 to 3327 nodes, the model benefits from the additional data. However, beyond a certain resolution, the polynomial field is already well resolved.

The absolute error distributions shown in Figure 4.7 further support this interpretation. The errors decrease significantly when moving from coarse to intermediate resolutions, while the difference between the 3327-node and 13048-node meshes is minimal.

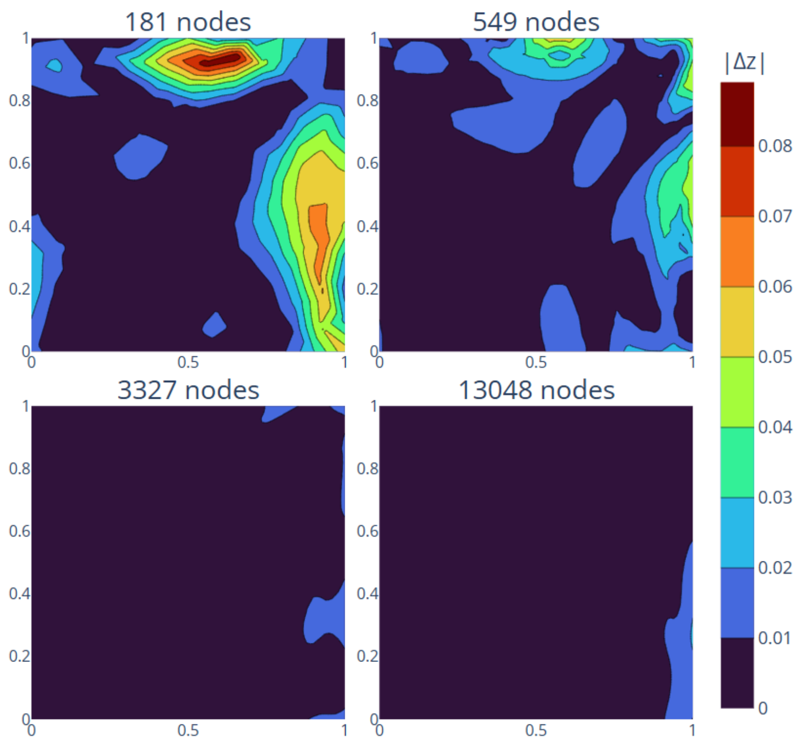


Figure 4.7: *Polynomial surface reconstruction: Absolute error distributions of the FCNN on meshes with 181, 549, 3327, and 13048 nodes.*

4.4.3 Discussion: Mesh Resolution in GNNs vs. FCNNs

The results indicate that the improved performance observed with increasing mesh resolution is primarily related to the increased amount of data available during training. For the FCNN, which processes each node independently, mesh refinement leads to a larger number of node samples being passed through the network. As a result, the model benefits from increased data exposure.

In contrast, the GAT operates on both nodes and edges through message passing. Increasing the mesh resolution therefore not only increases the number of node samples, but also the total number of edges and neighborhood interactions involved in training. Consequently, mesh refinement has a stronger impact on GAT performance than on the FCNN, as it simultaneously increases both sample count and interaction complexity.

4.5 Model Optimization and Comparison

The previous section investigated how mesh resolution affects the performance of the GAT and FCNN models. Building on those findings, this section focuses on a fair comparison between the two approaches by performing an architecture optimization study for each model. The mesh with 3327 nodes is used throughout this section.

4.5.1 Optimization Setup

The architecture of a NN has a critical impact on its performance. Key hyperparameters include the number of layers, the number of neurons per layer, and the choice of activation functions. These parameters directly influence the ability of the network to capture nonlinear relationships and, consequently, the validation loss achieved during training.

To identify suitable architectures for both the GAT and the FCNN, the problem is formulated as an optimization task. The design space includes: the number of hidden layers (3–10), the number of neurons per layer (2^5–2^{12}), the activation function in the hidden layers (ReLU, GeLU, tanh, sigmoid), and the activation function in the output layer (same set).

The optimization objective is the minimization of the validation loss (MAE; see Section 2.6). The EASY framework is employed for this task as described in Section 2.9. Each optimization run evaluates 200 candidate architectures, training each candidate for 500 epochs.

4.5.2 Optimization Results

In total, 200 GAT and 200 FCNN architectures are evaluated. Figure 4.8 shows the distribution of validation losses, illustrating the variability in performance across candidate models.

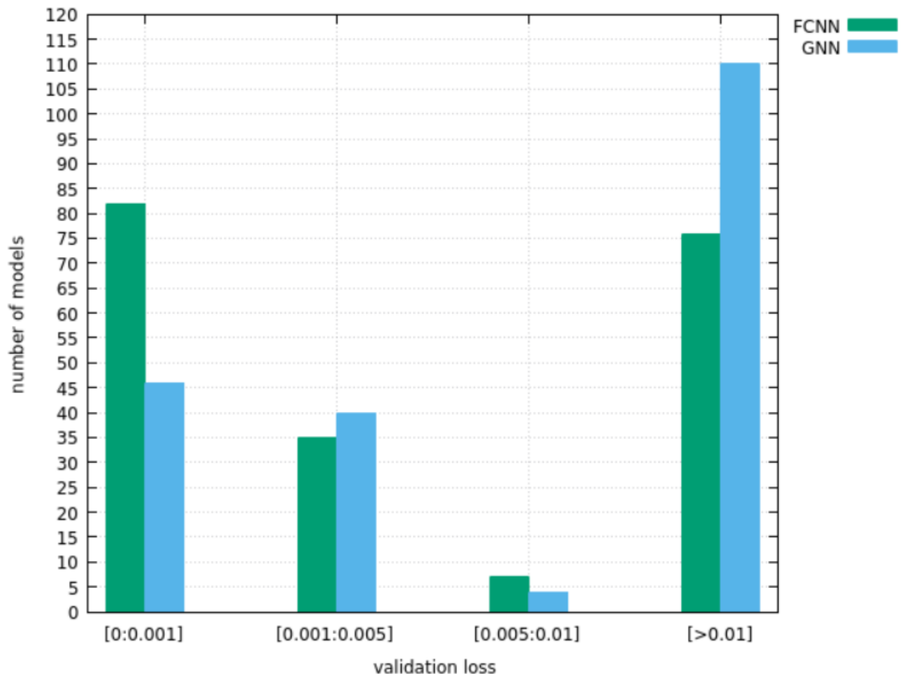


Figure 4.8: *Polynomial surface reconstruction: Distribution of validation losses across 200 GAT and 200 FCNN candidate architectures.*

The histogram indicates that 82 out of 200 FCNN models achieve a validation loss below 10^{-3}, compared with 46 out of 200 GAT models. This suggests that, in this benchmark, FCNNs generally outperform GATs in terms of raw predictive accuracy.

The best-performing architectures are:

- GAT: 3 hidden layers with [256, 1024, 512] neurons, GeLU activation in the hidden layers, and sigmoid activation in the output layer,
- FCNN: 7 hidden layers with [1024, 256, 32, 1024, 256, 1024, 512] neurons, GeLU activation in the hidden layers, and sigmoid activation in the output layer.

The two optimized architectures are trained further until the validation loss ceases to decrease. The GAT reaches a minimum validation loss of 6.0×10^{-4}, while the FCNN achieves a minimum validation loss of 2.0×10^{-4}.

4.5.3 Qualitative Evaluation and Attention Analysis

Figure 4.9 presents the absolute error between predicted and reference values.

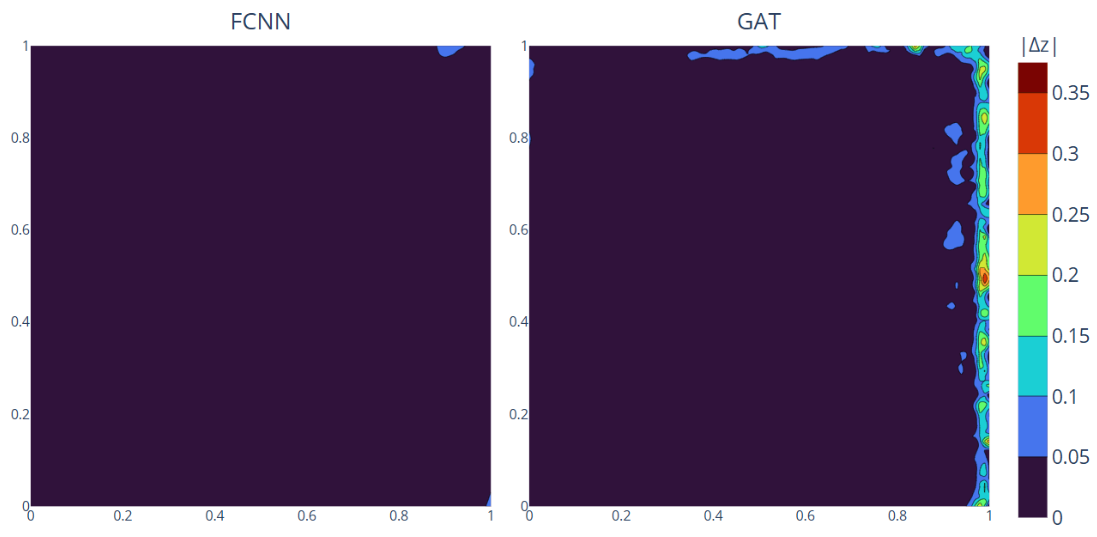


Figure 4.9: *Polynomial surface reconstruction: Absolute error distributions of the optimized GAT (left) and optimized FCNN (right).*

The corresponding absolute error maps shown in Figure 4.9 indicate that the FCNN achieves lower overall errors compared to the GAT. In the GAT predictions, the largest discrepancies are localized in regions where the target function exhibits strong spatial variations, whereas the FCNN errors remain uniformly low across the domain.

Attention–weight visualization. An additional benefit of GATs is their interpretability through attention coefficients. For each directed edge $j \to i$, the coefficient $\alpha^{(\ell)}_{j\to i}$ expresses the relative importance of neighbor j for updating node i at layer ℓ. To obtain a single representative value per edge, the attention coefficients are averaged across layers:

$$\bar{\alpha}_{j\to i} = \frac{1}{L}\sum_{\ell=1}^{L} \tilde{\alpha}^{(\ell)}_{j\to i}.$$

These values are visualized in Figures 4.10–4.12. Figures 4.10 and 4.11 display the directed edge attention coefficients, with Part A and Part B representing complementary message directions between neighboring nodes. Figure 4.12 illustrates the self-loop attention, highlighting the relative importance assigned to a node's own value. The background color map in all cases represents the polynomial field $z = f(x, y)$ in the region $x, y \in [0.8, 1]$, which is selected due to its large variability in values.

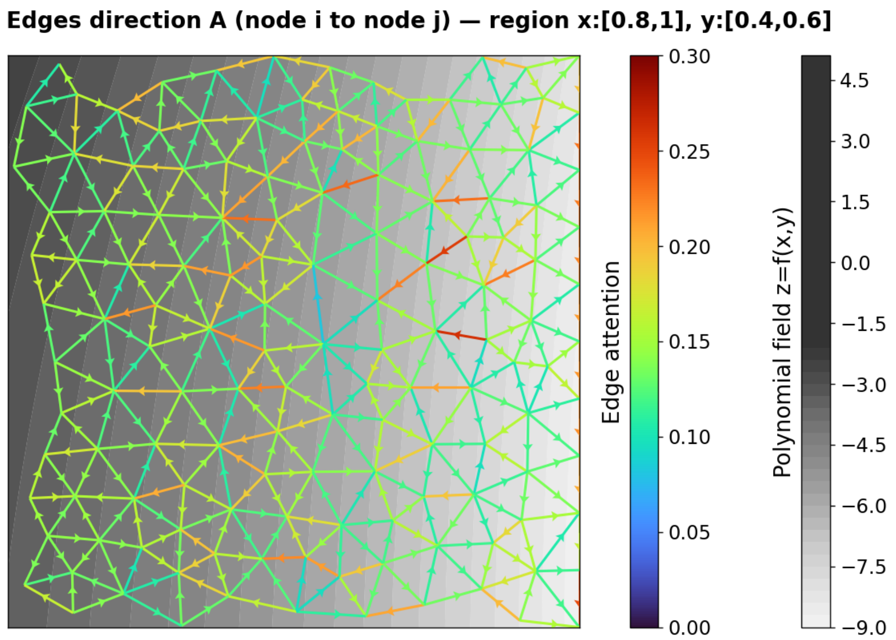


Figure 4.10: *Polynomial Surface Reconstruction: Directed messages, Part A — Attention strengths from sender to receiver nodes, with the polynomial field $z = f(x, y)$ shown in the background.*

Figures 4.10–4.12 indicate that the model learns the relative importance of neighboring nodes. The comparable magnitude of edge and self-loop weights further indicates that both the node's own value and those of its neighbors contribute substantially to the prediction.

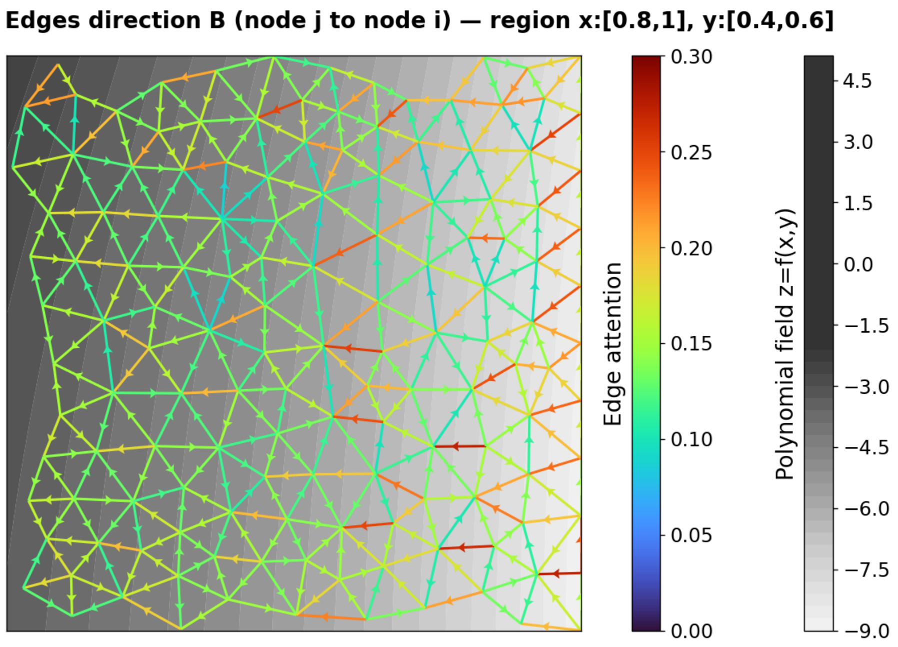


Figure 4.11: *Polynomial Surface Reconstruction: Directed messages, Part B — Complementary directions to those in Part A, with the polynomial field $z = f(x, y)$ shown in the background.*

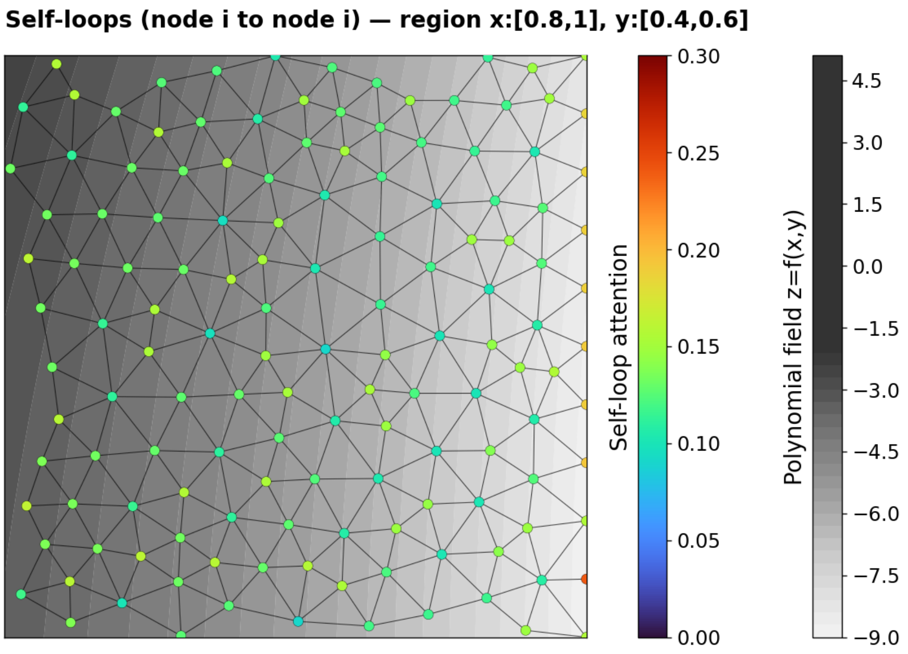


Figure 4.12: *Polynomial Surface Reconstruction: Self-loop attention values at each node, with the polynomial field $z = f(x, y)$ shown in the background.*

4.6 Increasing Function Complexity

The previous results showed that, for the initial polynomial reconstruction problem, the FCNN consistently outperformed the GAT model. In that setting, the dataset consisted of polynomial surfaces that were relatively smooth and similar in structure. In this section, the learning task is made more challenging by replacing the polynomial formulation with a function that introduces stronger nonlinearities.

The function used to generate the new dataset is defined as:

$$f(x, y) = b_0 \sin(b_1 xy) + \sin(b_2 y) + b3 \frac{x^2 + b4xy}{b5xy^2 - b6\sqrt{x} + 10}, \tag{4.2}$$

where the coefficients are sampled from the following ranges:

$$b_0 \in [2, 5], \quad b_1 \in [10, 20], \quad b_2 \in [10, 20], \quad b_3 \in [2, 8],$$

$$b_4 \in [2, 5], \quad b_5 \in [4, 8], \quad b_6 \in [1/4, 3/4]$$

A representative sample from the resulting dataset is shown in Figure 4.13.

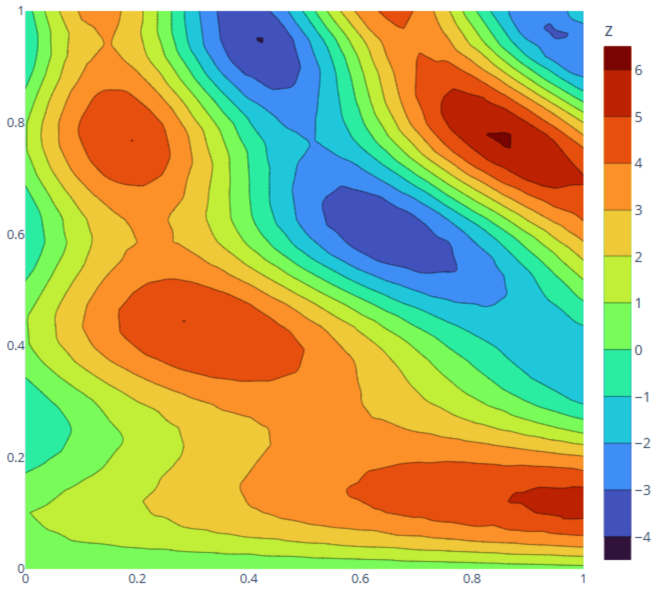


Figure 4.13: *Higher-complexity surface reconstruction: Representative sample from the dataset generated using* (4.2).

To keep training time manageable, all experiments in this section are conducted on the 549-node mesh. The dataset comprises 200 training and 20 validation samples. As before, architecture optimization is performed for both models.

The best FCNN achieves a validation loss of 6×10^{-3}, while the best GAT attains 9×10^{-3}. Figure 4.14 compares the corresponding absolute error fields.

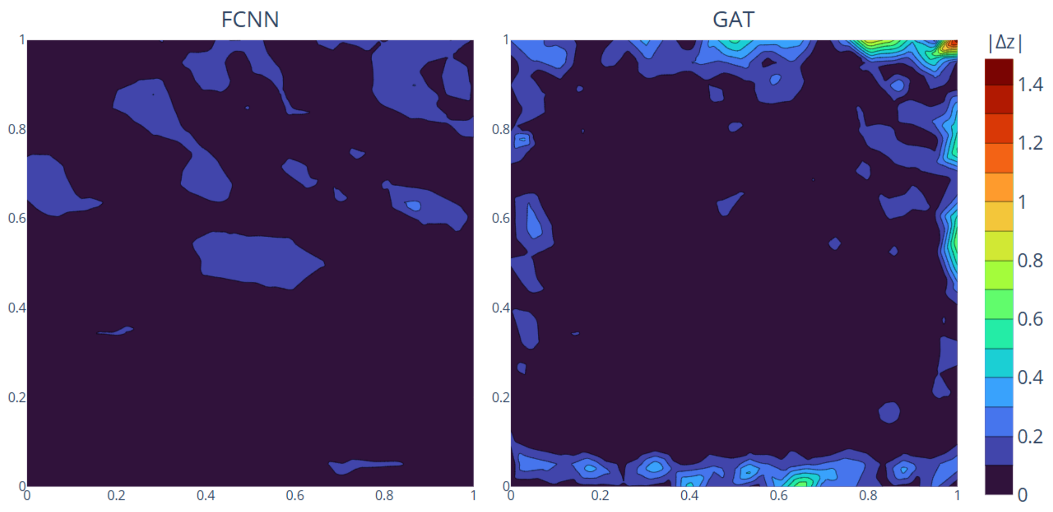


Figure 4.14: *Higher-complexity surface reconstruction: Absolute error fields for the best FCNN and GAT models.*

The results indicate that the added complexity of GAT does not improve performance in this setting. Although GAT aggregates information from neighboring nodes, this adds limited benefit when topology is fixed and the target depends on local input features.

4.7 Key Findings

The main findings are summarized as follows:

- The predictive performance of both models improves with increasing mesh resolution due to the higher number of training nodes. When the total node count (number of meshes $\times$ nodes per mesh) is similar, coarse meshes can achieve performance comparable to finer meshes.
- FCNN performance depends mainly on the total number of training nodes, whereas GAT performance depends on both nodes and edges. Consequently, GAT is more sensitive to mesh resolution due to increased connectivity.
- In both simple and complex regression tasks, FCNN consistently outperforms GAT in predictive accuracy.
- In this regression problem, mesh topology adds limited benefit since the target depends only on local features; therefore, GAT offers no advantage over pointwise FCNN.

These findings motivate the transition to CFD problems in the next chapter, where spatial interactions and mesh connectivity become physically meaningful and are expected to favor graph-based models.

Chapter 5

NACA4318 Isolated Airfoil

5.1 Objective and Structure of the Case Study

In the previous chapter, GNNs were evaluated on a controlled polynomial regression problem. It was shown that, in such pointwise regression tasks, graph-based models do not offer an advantage over FCNNs.

The present chapter extends this investigation to a CFD application, where mesh connectivity and neighborhood interactions play a key role in the underlying physics. The case study focuses on turbulent flow over an isolated NACA4318 airfoil.

The primary objective of this chapter is to develop and train a GNN capable of replacing the SA turbulence model within the in-house CFD solver **PUMA**.

5.2 NACA4318 Isolated Airfoil Case Setup

5.2.1 Airfoil Geometry and Parameterization

The reference geometry considered in this study is the NACA4318 airfoil, shown in Figure 5.1. To generate a dataset suitable for training, the baseline profile was embedded in a 5×3 Non-Uniform Rational B-Splines (NURBS) control box, illustrated in Figure 5.1. Out of the total 15 control points, 13 were allowed to move within $\pm 10\%$ of their reference position in both chordwise and normal-to-chord directions, while 2 remained fixed.

Using Latin Hypercube Sampling (LHS), 100 distinct airfoil geometries were generated by varying the control points within their prescribed ranges. The resulting set of geometries, shown in Figure 5.2, ensures diversity while retaining aerodynamic relevance.

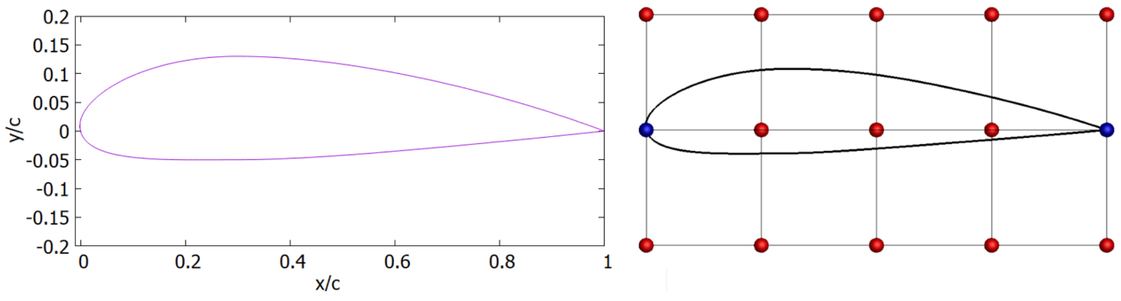


Figure 5.1: *NACA4318 Case: Baseline geometry (Left). NURBS control box around the airfoil. Blue points: fixed; Red points: movable (Right).*

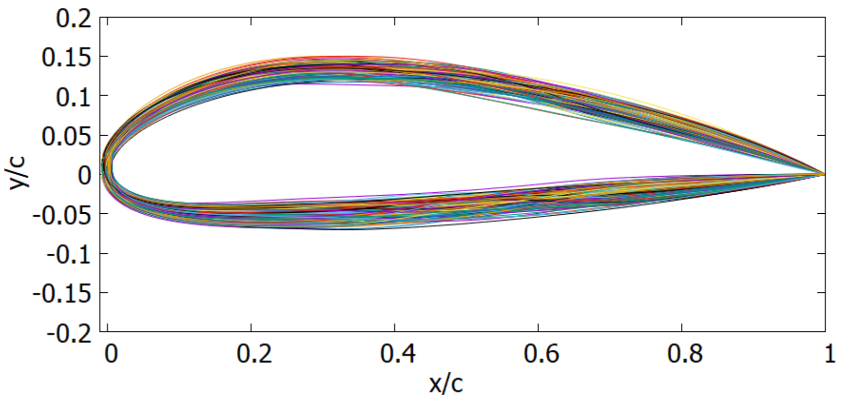


Figure 5.2: *NACA4318 Case: Set of 100 airfoil geometries generated using LHS.*

5.2.2 Mesh Generation

For each of the 100 geometries, a structured C-type computational mesh was generated to provide consistent discretization across the dataset. An example mesh is shown in Figure 5.3. Each mesh contains 29848 nodes and 118664 edges, with strong refinement near the airfoil surface and in the wake region to accurately capture turbulence.

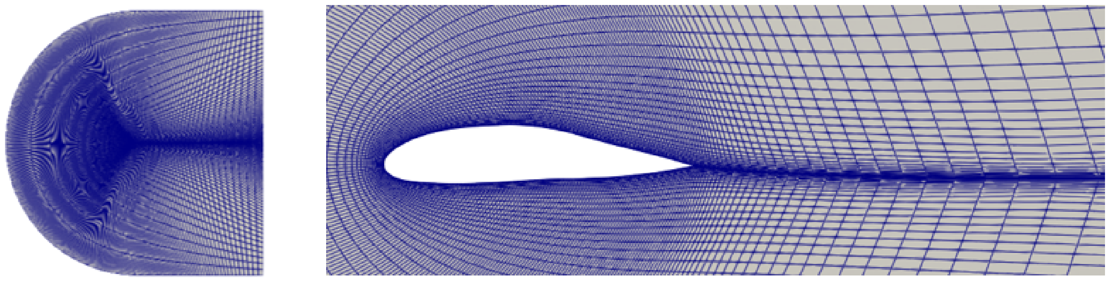


Figure 5.3: *NACA4318 Case: Computational mesh around the airfoil. Left: full view including the farfield boundary. Right: close-up view near the airfoil surface.*

5.2.3 Flow Conditions

The simulations were carried out by solving the RANS equations with turbulence closure provided by the SA model. The flow conditions are fixed and summarized in Table 5.1.

Quantity	**Symbol**	**Value**
Freestream Mach number	M_∞	0.13
Reynolds number ($\times 10^6$)	Re	3.8
Angle of attack (°)	AoA	2.2

Table 5.1: *NACA4318 Case: Flow conditions.*

5.2.4 Baseline Flow Analysis

The baseline NACA4318 airfoil serves as a reference configuration. Figure 5.4 shows the pressure coefficient distribution over the baseline profile. As expected, pressure is higher on the lower surface (pressure side) and lower on the upper surface (suction side), thereby generating lift.

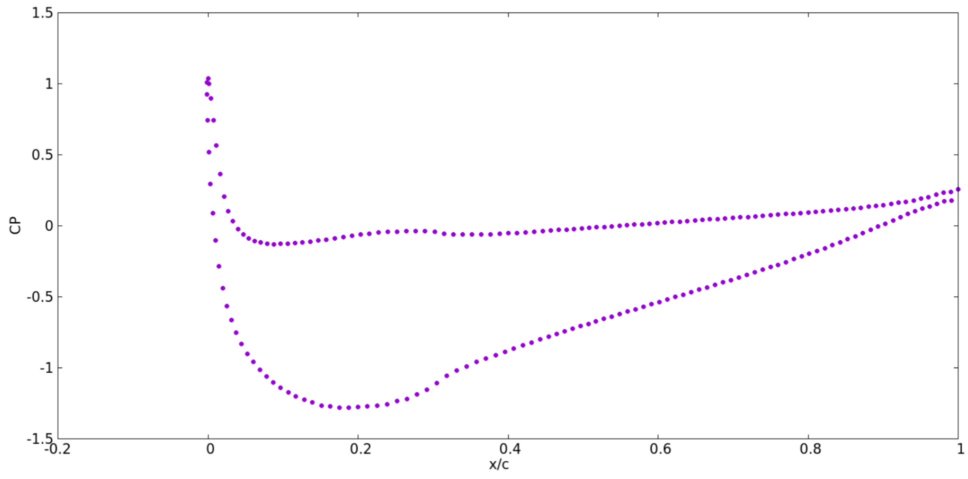


Figure 5.4: *NACA4318 Case: Pressure coefficient distribution on the baseline NACA4318 geometry.*

5.3 Dataset Preparation

5.3.1 Selection of Input and Output Quantities

The CFD simulations of the NACA4318 geometries provide a rich set of flow and geometric variables. To ensure consistency and enable direct comparison between the GNN and the FCNN models, the same set of input quantities as those used in the framework of the PhD thesis [20] is adopted.

Accordingly, the following quantities are used as inputs to the NNs:

- Nodal coordinates (x, y),
- Velocity components (u, v),
- Static pressure p,
- Vorticity Ω,
- Strain rate S,
- Wall distance d.

The output of the network is the turbulent kinematic viscosity μ_t. An example distribution of μ_t for the baseline NACA4318 airfoil is shown in Figure 5.5. As expected, μ_t is nearly zero in most of the computational domain, except in the wake region behind the airfoil, where higher values are present. Capturing this highly localized distribution constitutes the main challenge for the NN.

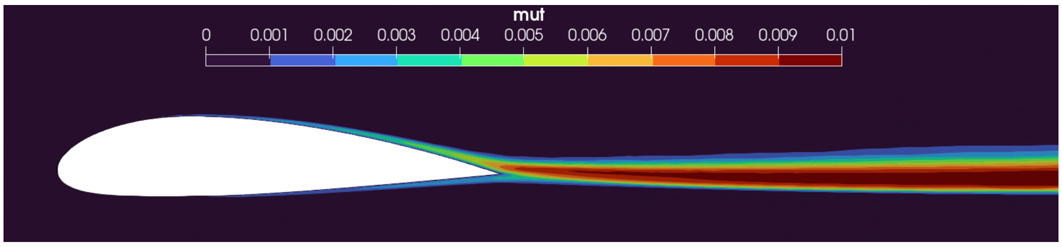


Figure 5.5: *NACA4318 Case: Turbulent viscosity field for the baseline geometry.*

5.3.2 Normalization Strategy

The input features and the output variable exhibit widely different scales; therefore, all quantities are scaled using Min–Max normalization as described in Section 4.3. The minimum and maximum values for each variable are summarized in Table 5.2.

Quantity	Minimum	Maximum
x-coordinate	-47.7	50.0
y-coordinate	-50.0	50.0
u - velocity	-1.3	71.0
v - velocity	-23.0	40.6
Pressure	99415.0	102656.0
Vorticity	0.0	798569.0
Strain rate	1e-4	798262.0
Wall distance	0.0	70.0
Turbulent viscosity μ_t	0	0.0127

Table 5.2: *NACA4318 Case: Min and max values of the input and output quantities.*

During inference, predictions are obtained by normalizing the inputs of a new mesh with the minimum and maximum values determined from the training dataset. Consequently, if a new geometry produces flow variables that deviate significantly from these ranges, the normalized values may lie outside the interval $[0, 1]$. In such cases, the network is effectively extrapolating, which can reduce prediction accuracy.

5.4 Model Development and Evaluation

Three NN models are applied to the NACA4318 dataset: an FCNN, a GAT, and an MPNN, as introduced in Sections 2.4 and 3.2.

The dataset consists of 100 geometries, with 90 cases used for training and 10 reserved for validation. Each geometry is represented as a complete graph containing its mesh and associated flow variables. The data are processed in *disjoint mode* (see Section 3.3).

5.4.1 Optimization of Architectures

The architectures of the three models (FCNN, GAT, and MPNN) were optimized using the EASY framework (see Section 2.9), following the same procedure as in Section 4.5. The design variable ranges for each model are summarized in Table 5.3.

Design variable	**FCNN**	**GAT**	**MPNN**
Hidden layers	3–10	3–10	4–6
Neurons per layer	2^5–2^{12}	2^5–2^{12}	2^2–2^9
Hidden activations	Activation set	Activation set	Activation set
Output activation	Activation set	Activation set	Activation set

Table 5.3: *NACA4318 Case: Ranges of design variables for FCNN, GAT, and MPNN optimization. The activation set is defined as {ReLU, GeLU, tanh, sigmoid}.*

For the FCNN and GAT models, the ranges of the design variables were common, whereas for the MPNN the ranges were restricted to reduce memory consumption.

This restriction was necessary because MPNNs are more demanding computationally due to the additional memory required to store the edge-feature matrix (see Section 3.2).

The FCNN architecture was previously optimized in the framework of a PhD thesis [20] using the same dataset and is therefore not re-optimized here. The best-performing FCNN, GAT, and MPNN architectures identified are:

- **FCNN**: 6 hidden layers with widths (64, 128, 256, 1024, 4096, 512), ReLU activations in the hidden layers, and tanh in the output layer, resulting in approximately 6×10^6 trainable parameters [20].

- **GAT**: 5 hidden layers with widths (256, 512, 512, 256, 256), GeLU activations in the hidden layers, and tanh in the output layer, resulting in approximately 6×10^5 trainable parameters.
- **MPNN**: 6 hidden layers with widths (128, 256, 64, 256, 16, 32), ReLU activations in the hidden layers, and GeLU in the output layer, resulting in approximately 2×10^5 trainable parameters.

During the optimization of the GAT architecture, only 30 out of 200 evaluations were successful due to memory limitations. For the MPNN, 123 out of 200 evaluations were successful, owing to the more restricted search space.

The minimum validation losses obtained from the optimization of the GAT and MPNN models are summarized in Table 5.4, both for the 500-epoch optimization runs and for extended training where models were trained until the validation loss ceased to decrease. For the FCNN, which was already optimized in the framework of a PhD thesis [20], only the validation loss from extended training is reported.

Model	**Validation loss (500 epochs)**	**Validation loss (further training)**
FCNN	–	3.5×10^{-3}
GAT	1.8×10^{-3}	1.5×10^{-3}
MPNN	1.3×10^{-3}	1.1×10^{-3}

Table 5.4: *NACA4318 Case: Minimum validation loss achieved by the optimized architectures.*

5.4.2 Turbulent Viscosity Prediction on Validation Cases

The predictive performance of the three models — FCNN, GAT, and MPNN — is evaluated on structured meshes of the NACA4318 validation dataset (unseen geometries). For each NN, the CFD reference solution, the model prediction, and the absolute error are presented.

FCNN Figure 5.6 shows the results obtained with the FCNN model. The network successfully captured both the magnitude and overall distribution of the turbulent viscosity field, with max absolute error 3.6×10^{-3}. A closer view of the trailing-edge wake in Figure 5.7 shows that, although the FCNN captures the overall wake structure, the predicted field is less smooth than the CFD reference. This is a direct consequence of the FCNN's pointwise formulation, which lacks explicit neighborhood coupling and therefore struggles to enforce spatial coherence in regions with strong gradients.

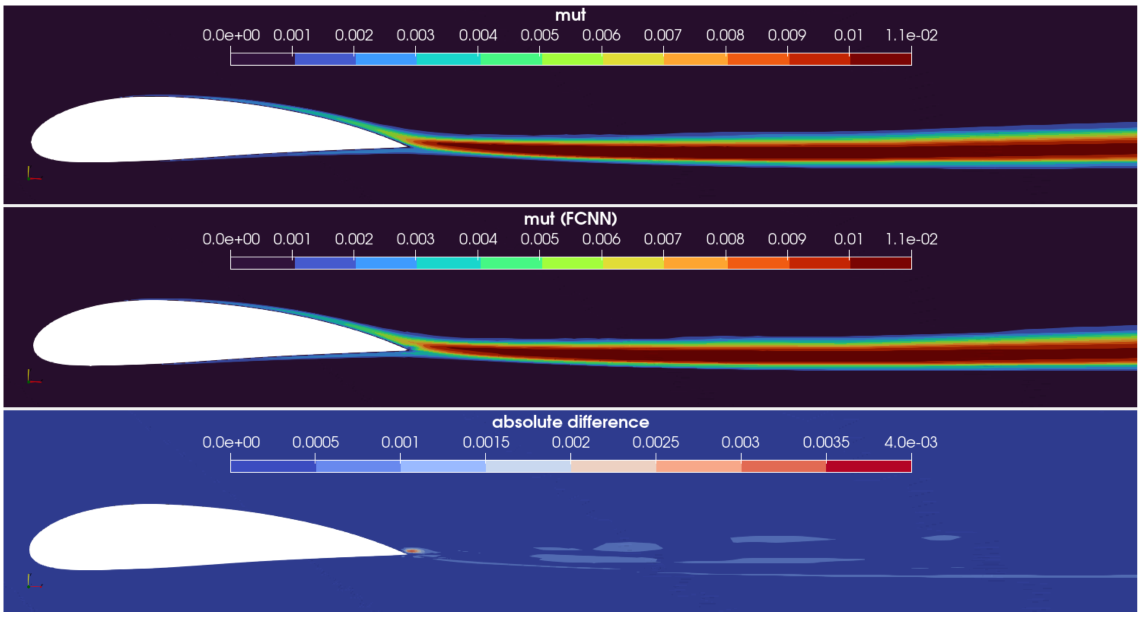


Figure 5.6: *NACA4318 Case: Comparison of CFD results, FCNN predictions, and absolute error for a validation airfoil.*

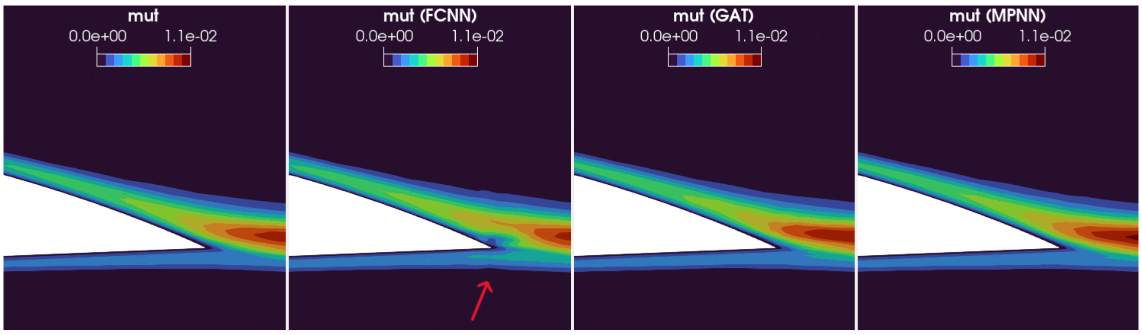


Figure 5.7: *NACA4318 case: Detailed comparison of CFD results and predictions from the FCNN, GAT, and MPNN in the wake region. The arrow highlights a less smooth region in the FCNN prediction.*

GAT Figure 5.8 presents the results of the GAT model. Compared to the FCNN, the GAT reproduced the turbulent viscosity distribution with higher accuracy, achieving max absolute error 1.1×10^{-3}. The wake was resolved more smoothly at the trailing edge, compared to the FCNN (Figure 5.7). This improvement is attributed to the GAT's ability to incorporate neighborhood relations.

Attention–weight visualization. The attention coefficients of the trained GAT model are visualized as described in Paragraph 4.5.3. The two directions are shown separately in Figures 5.9 and 5.10, while Figure 5.11 displays the *self–loop* attention as node colors. In all edge plots, the **tail of the arrow corresponds to the sender**, and the **head corresponds to the receiver**.

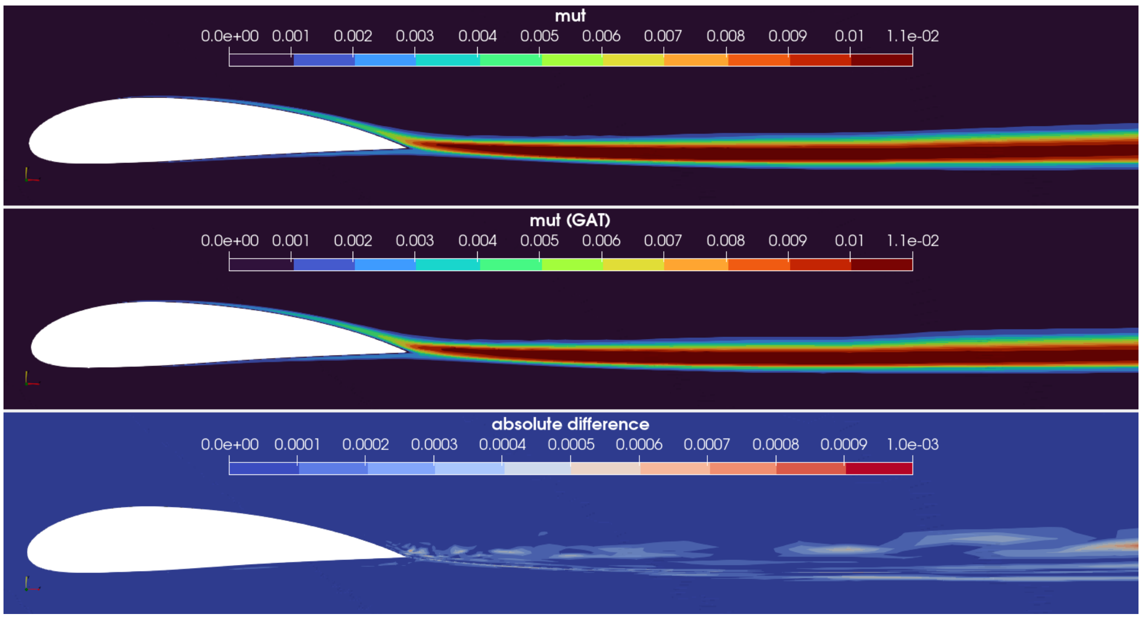


Figure 5.8: *NACA4318 Case: Comparison of CFD results, GAT predictions, and relative error for a validation airfoil.*

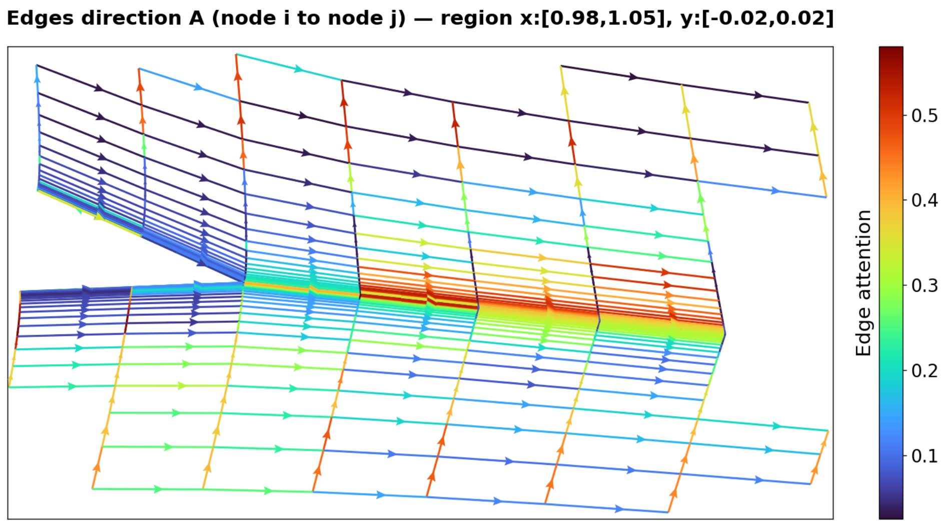


Figure 5.9: *NACA4318 Case: Directed messages, part A - information flow from sender to receiver nodes, colored by attention strength.*

For clarity, the visualization focuses on the trailing-edge wake region ($x \in [0.98, 1.05]$, $y \in [-0.02, 0.02]$). In this zone, the attention weights assigned to upstream-directed messages (Figure 5.10) are generally stronger than those aligned with the freestream direction (Figure 5.9). Although the underlying mesh connectivity is undirected, the GAT assigns independent attention coefficients to each directed message, allowing the model to learn asymmetric importance between neighboring nodes.

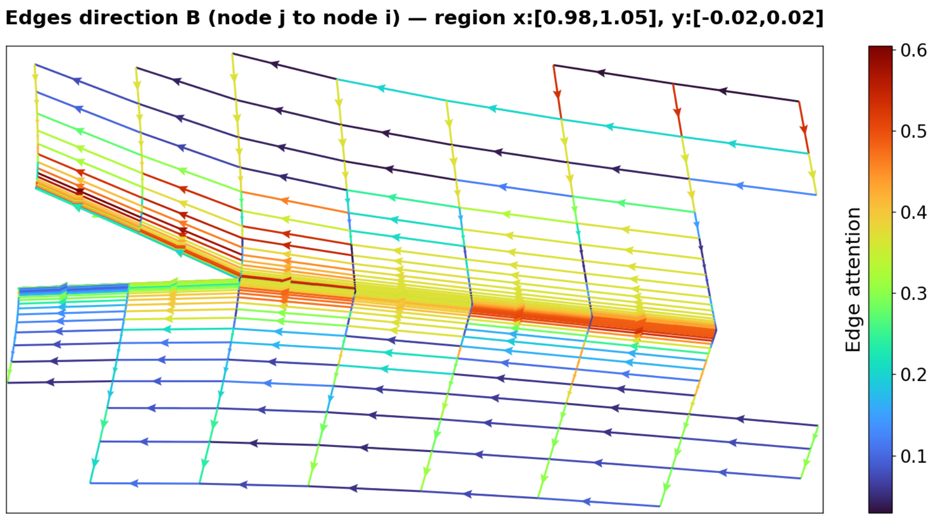


Figure 5.10: *NACA4318 Case: Directed messages, part B - complementary directions to those shown in part A.*

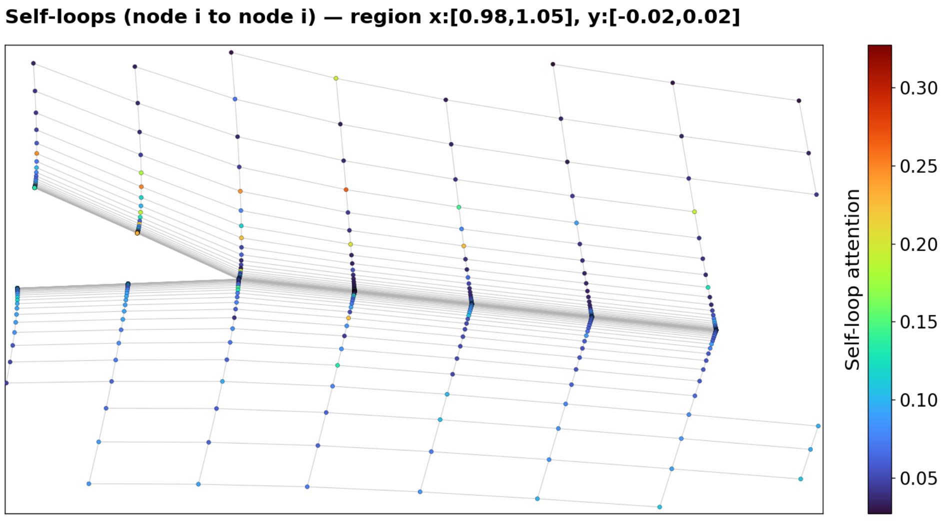


Figure 5.11: *NACA4318 Case: Self-loop attention at each node, with the mesh shown faintly for reference.*

Furthermore, self-loop attention (Figure 5.11) is typically lower than the strongest neighbor-edge attention, suggesting that the model relies more heavily on neighborhood information than on each node's own features. This behavior contrasts with the polynomial regression case studied in the previous chapter, where self-loop and neighbor attention weights exhibited comparable magnitudes, reflecting the pointwise nature of that problem.

MPNN Figure 5.12 illustrates the predictions of the MPNN. The model accurately reproduced both the magnitude and spatial distribution of the turbulent viscosity, with max absolute error 6.7×10^{-4}. The trailing-edge wake is represented smoothly, confirming the ability of the MPNN to exploit mesh connectivity (Figure 5.7).

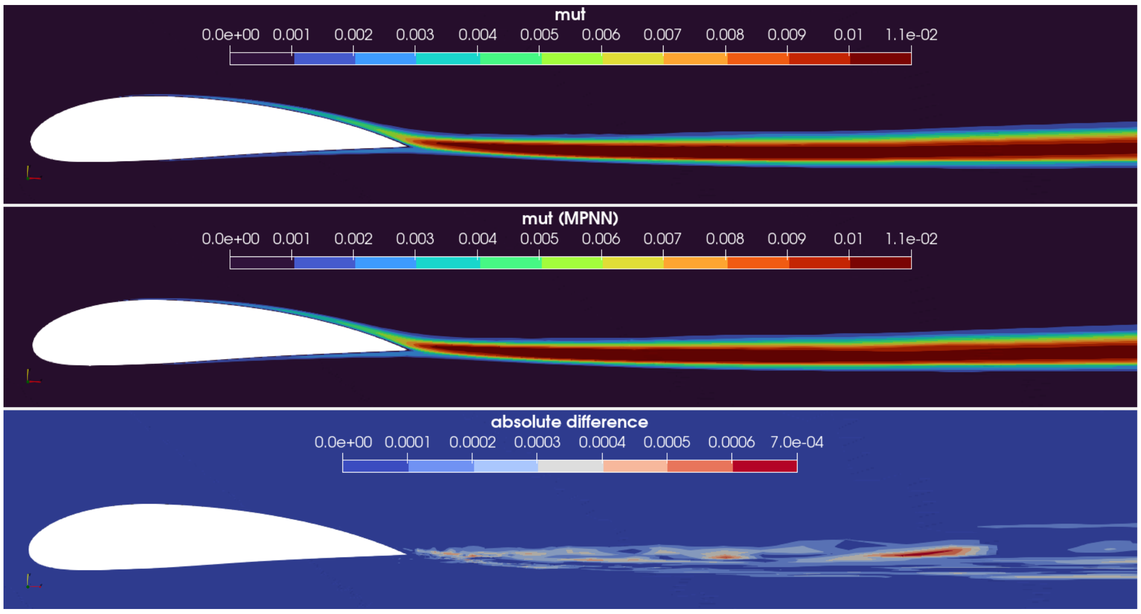


Figure 5.12: *NACA4318 Case: Comparison of CFD results, MPNN predictions, and absolute error for a validation airfoil.*

Quantitative comparison on the validation set To complement the qualitative analysis presented above, the predictive performance of the three models is quantified over the entire NACA4318 validation dataset. Figures 5.13 and 5.14 report the mean and maximum absolute errors, respectively, for each validation airfoil and for each model.

Across all validation cases, both graph-based models outperform the FCNN in terms of mean and maximum absolute error, with the MPNN consistently achieving the lowest error levels.

The training and prediction times for each model are summarized in Table 5.5. Compared to the FCNN, the GAT and MPNN require substantially more memory due to the additional graph-related data structures (adjacency and edge-feature matrices). As a result, smaller batch sizes (number of meshes processed simultaneously) are necessary to avoid memory limitations, which in turn increases training time per epoch. In this study, training was performed with a batch size of 5 for the FCNN and a batch size of 1 for both GAT and MPNN.

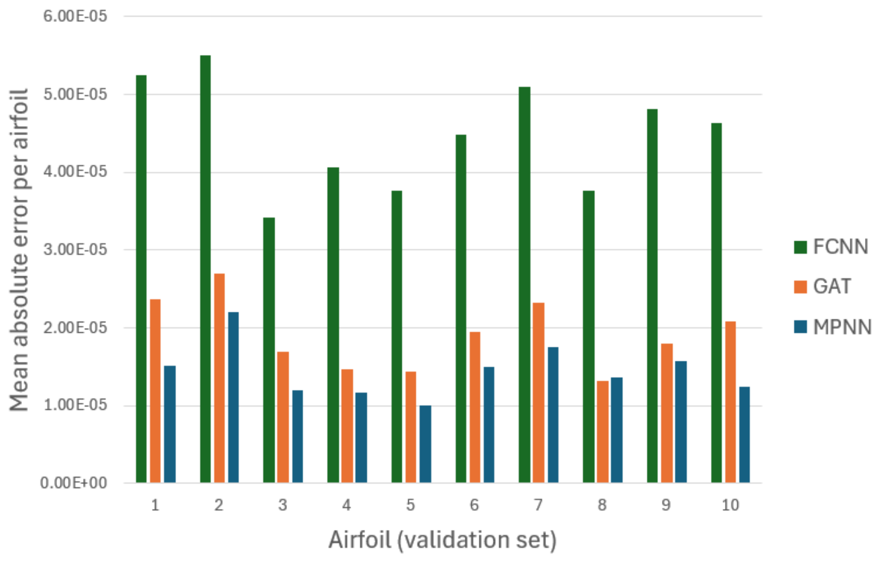


Figure 5.13: *NACA4318 Case: Mean absolute error per validation airfoil for FCNN, GAT, and MPNN models.*

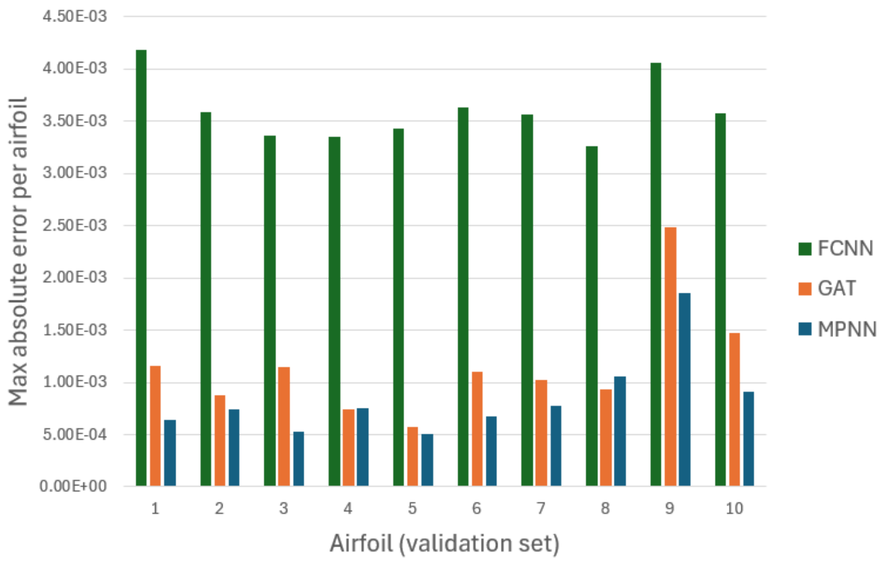


Figure 5.14: *NACA4318 Case: Maximum absolute error per validation airfoil for FCNN, GAT, and MPNN models.*

Despite the smaller batch size, the FCNN exhibits the longest total training time due to its significantly larger number of trainable parameters. In contrast, the graph-based models converge faster overall. During inference, the FCNN achieves the lowest prediction time, as it operates solely on node features and requires less data loading. Training times reported here exclude data-loading overhead, whereas prediction times include both data loading and inference.

Model	Training time [h]	Prediction time [s]
FCNN	13.0	7.45
GAT	9.9	16.46
MPNN	5.2	13.14

Table 5.5: *NACA4318 Case: Training and prediction time comparison of FCNN, GAT, and MPNN models.*

Relative error All models are trained using an MAE loss; therefore, the optimization objective is to minimize the absolute error on the available dataset. The resulting absolute errors are typically on the order of 10^{-3}. However, the turbulent viscosity field is close to zero over a large portion of the domain. In these near-zero regions, predicted values can be on the order of 10^{-4} while the reference values may be as small as 10^{-8}. Although this discrepancy is negligible in absolute terms, it leads to very large relative errors due to the small denominator. For this reason, relative-error metrics are computed only in regions where the turbulent viscosity exceeds 10^{-3}. This masking restricts the evaluation to the physically relevant part of the flow, where turbulent viscosity is non-negligible, and the largest gradients occur.

Figures 5.15 and 5.16 report the mean and maximum absolute relative error for each validation airfoil. Both metrics indicate that the graph-based models outperform the FCNN, with the MPNN achieving the lowest errors overall.

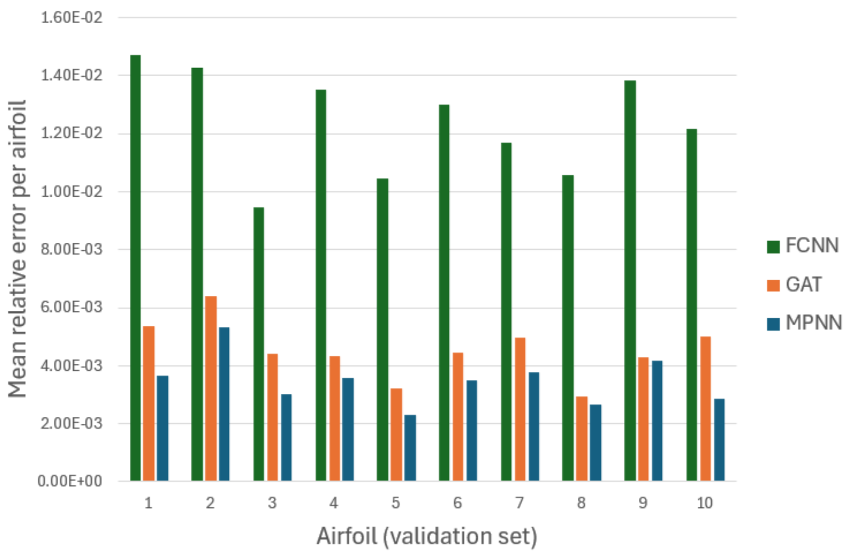


Figure 5.15: *NACA4318 Case: Mean absolute relative error per validation airfoil for FCNN, GAT, and MPNN models.*

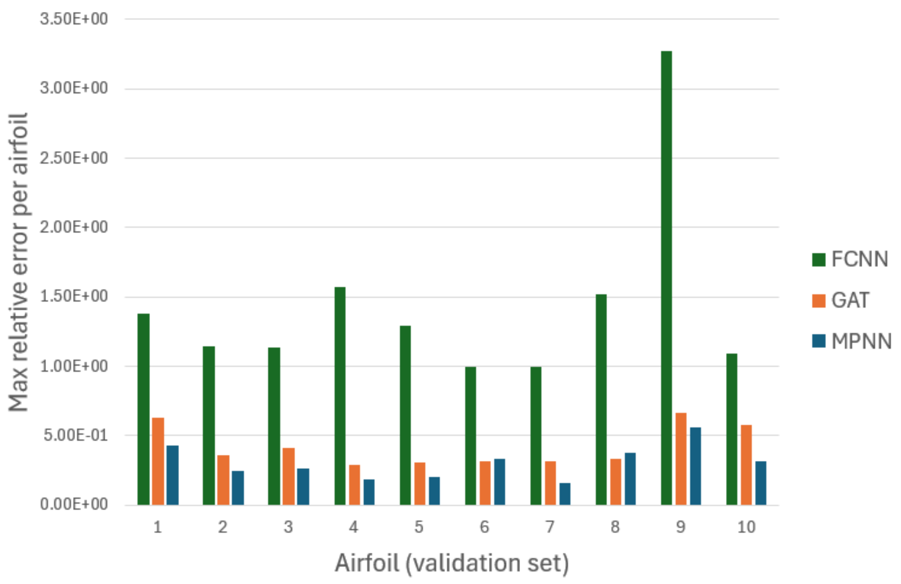


Figure 5.16: *NACA4318 Case: Max absolute relative error per validation airfoil for FCNN, GAT, and MPNN models.*

To further localize the discrepancies, Figure 5.17 compares the spatial distribution of the relative error using a common color scale.

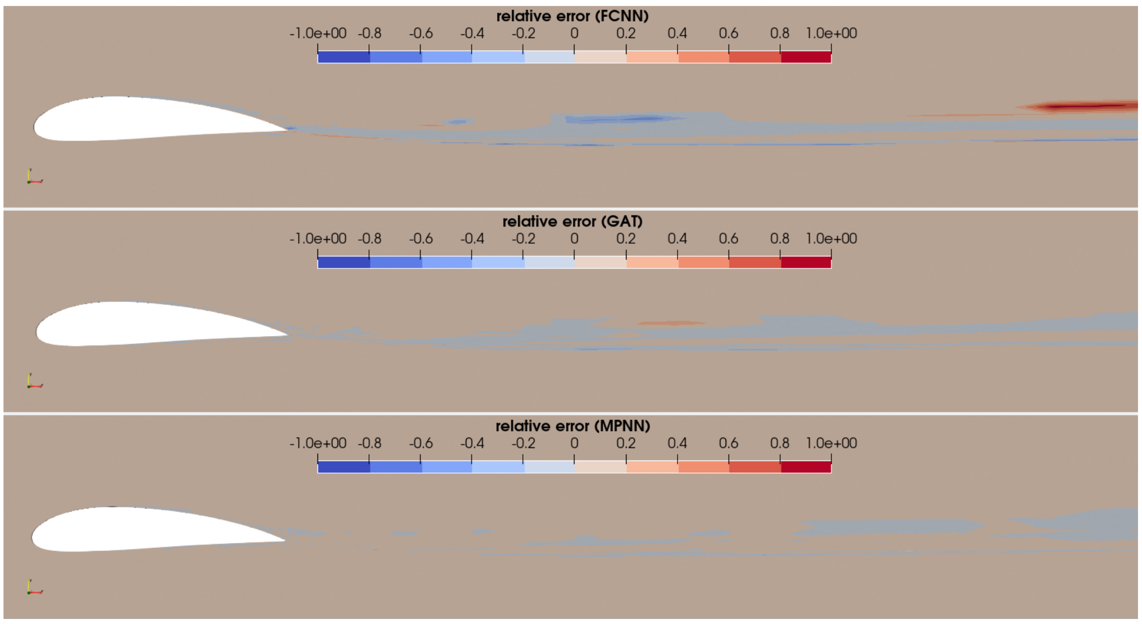


Figure 5.17: *NACA4318 case: Comparison of the relative error fields for a validation airfoil for FCNN, GAT, and MPNN (common color scale)*

The FCNN exhibits its largest errors near the trailing edge, i.e., in a region where the airfoil geometry changes. In contrast, the graph-based models substantially reduce these localized peaks and yield a smoother error distribution. This indicates that exploiting neighborhood connectivity helps the GNNs adapt to geometry-driven variations that are harder to capture using pointwise features alone.

Conclusion Overall, these results demonstrate that graph-based models, and in particular GAT and MPNN, provide a more faithful reconstruction of the turbulent viscosity field than purely feature-based networks, highlighting their suitability for CFD prediction tasks.

5.4.3 Demonstrating the Mesh Flexibility of NNs

The training dataset is composed of structured C-type meshes sharing a common farfield extent and an identical number of nodes. While nodal coordinates vary slightly across different airfoil geometries, the underlying mesh topology remains fixed. This setup allows the NNs to learn from a consistent discretization during training. To assess the ability of different NN architectures to generalize beyond the training mesh, two mesh-modification tests were performed.

Test 1: Unstructured triangulation with identical farfield and node count. In the first test, the baseline NACA4318 airfoil mesh was converted into an unstructured triangular mesh by subdividing each quadrilateral cell of the structured C-mesh. This transformation preserved the original node positions and total node count, while increasing the number of edges from 118664 to 177632. A zoomed view of the resulting triangulation is shown in Figure 5.18.

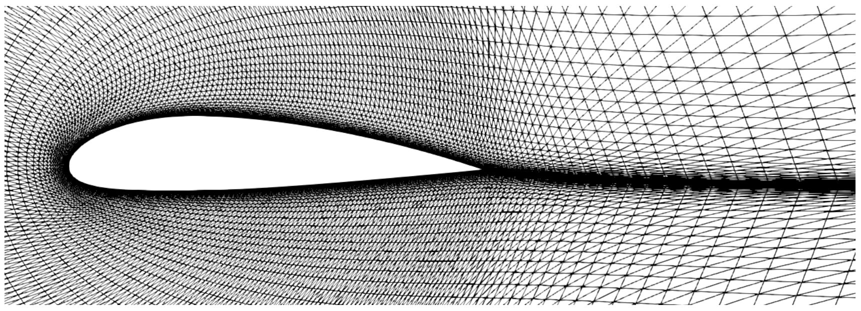


Figure 5.18: *NACA4318 Case: Zoomed view of the unstructured triangular mesh generated from the structured C-type training mesh with identical farfield and node count (Test 1).*

Figure 5.19, compares the turbulent viscosity (μ_t) predictions obtained with the FCNN, GAT, and MPNN models against the RANS reference solution. All three models reproduce the global flow structure; however, they tend to overpredict the wake thickness. The FCNN and GAT predictions remain within the correct order of magnitude, whereas the MPNN produces peak turbulent viscosity levels nearly twice the maximum value in the RANS solution, indicating a sensitivity to the modified graph connectivity.

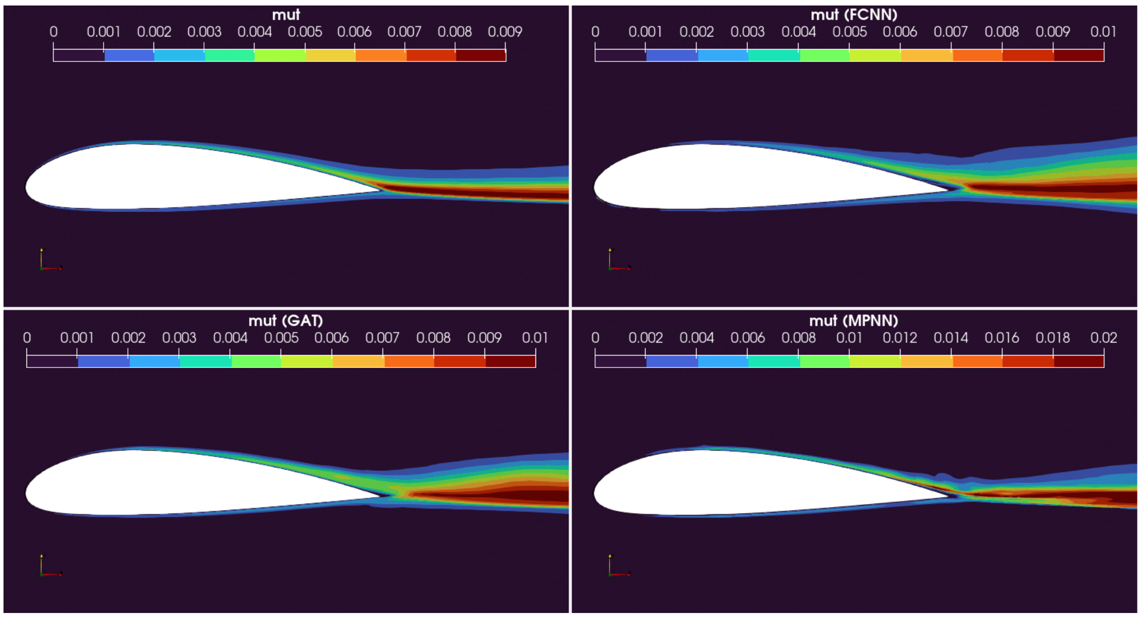


Figure 5.19: *NACA4318 Case: Comparison of the predictions for FCNN, GAT, and MPNN models (Test 1).*

Test 2: Unstructured mesh with different farfield and node count. A second and more challenging evaluation was carried out using an unstructured triangular mesh with a different farfield extent and a reduced number of nodes (56862). This mesh configuration, illustrated in Figure 5.20, introduces both geometric and topological changes relative to the training data.

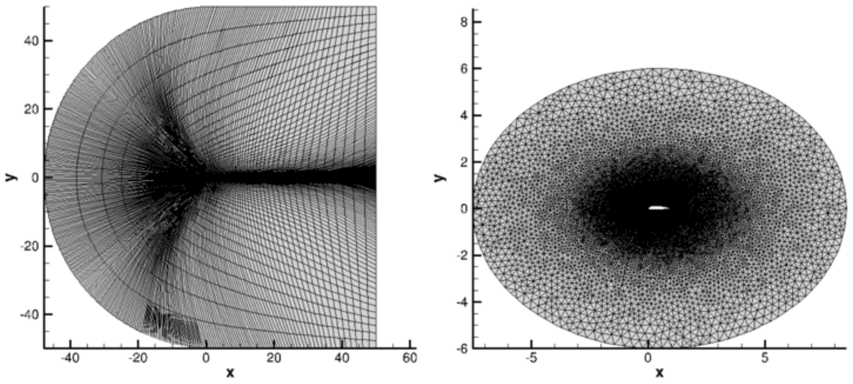


Figure 5.20: *NACA4318 Case: Training mesh (left) and unstructured triangular mesh with different farfield and node count used for Test 2 (right).*

Figure 5.21 show that both the FCNN and GAT models maintain physically plausible predictions under these conditions. The predicted turbulent viscosity values remain within a realistic range, and the wake region is correctly identified downstream of the trailing edge.

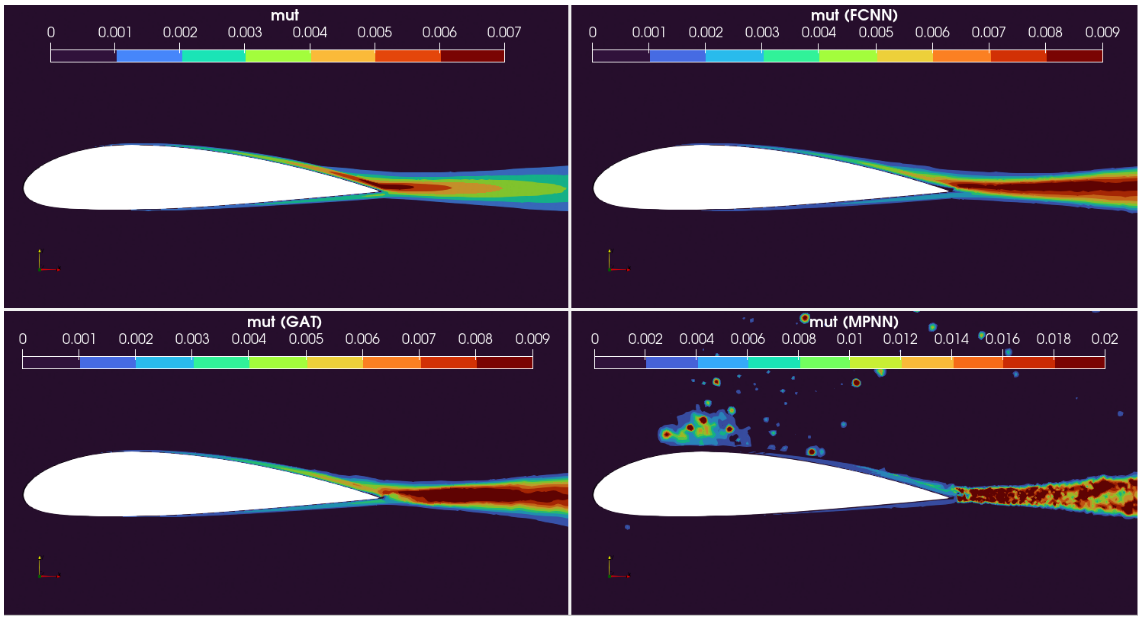


Figure 5.21: *NACA4318 Case: Comparison of the predictions for FCNN, GAT, and MPNN models (Test 2).*

In contrast, the MPNN fails to generalize to this mesh configuration (Figure 5.21). The predicted eddy viscosity field deviates substantially from the RANS solution, with a distorted wake structure and incorrect spatial distribution. This behavior suggests that the MPNN architecture, as configured in this study, is strongly dependent on the graph topology encountered during training.

5.5 Integration of NNs into the CFD Solver

In this section, the feasibility of replacing the SA turbulence model within the in-house CFD solver **PUMA** is investigated. The FCNN and GAT models developed and trained in the previous section are employed. The resulting configurations are referred to as *RANS–FCNN* and *RANS–GAT*, respectively.

5.5.1 Convergence of Aerodynamic Coefficients

Figures 5.22 and 5.23 show the convergence histories of the lift and drag coefficients for a representative validation airfoil. Results obtained with the RANS–FCNN and RANS–GAT closures are compared against the RANS–TM reference solution.

After an initial transient phase, both NN-based simulations converge toward steady values of C_L and C_D, although small residual oscillations remain visible in the zoomed region of the convergence histories. The GAT closure reaches values that more closely match those of the SA turbulence model than the FCNN.

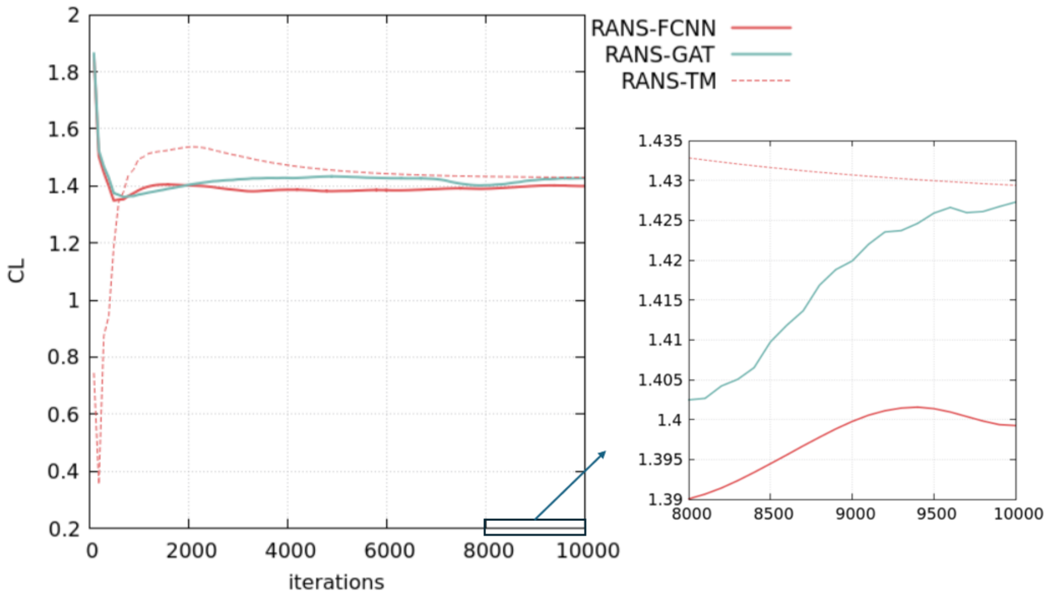


Figure 5.22: *NACA4318 Case: Convergence history of the lift coefficient C_L for RANS–FCNN, RANS–GAT, and RANS–TM simulations.*

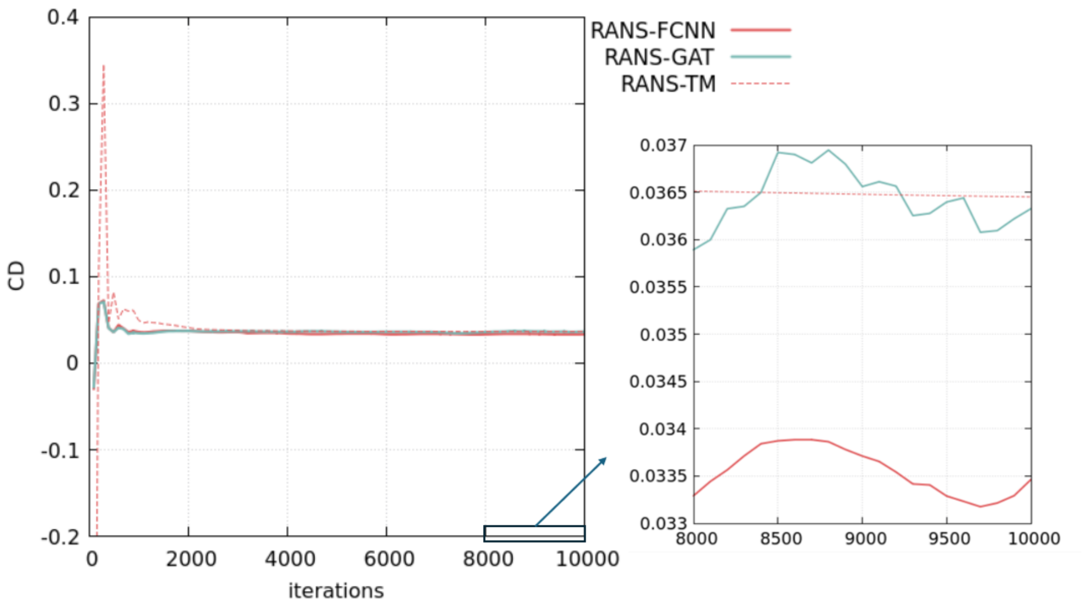


Figure 5.23: *NACA4318 Case: Convergence history of the drag coefficient C_D for RANS–FCNN, RANS–GAT, and RANS–TM simulations.*

5.5.2 Comparison of Turbulent Viscosity Fields

Figures 5.24 and 5.25 compare the turbulent viscosity fields obtained with the SA turbulence model and the NN-based closures.

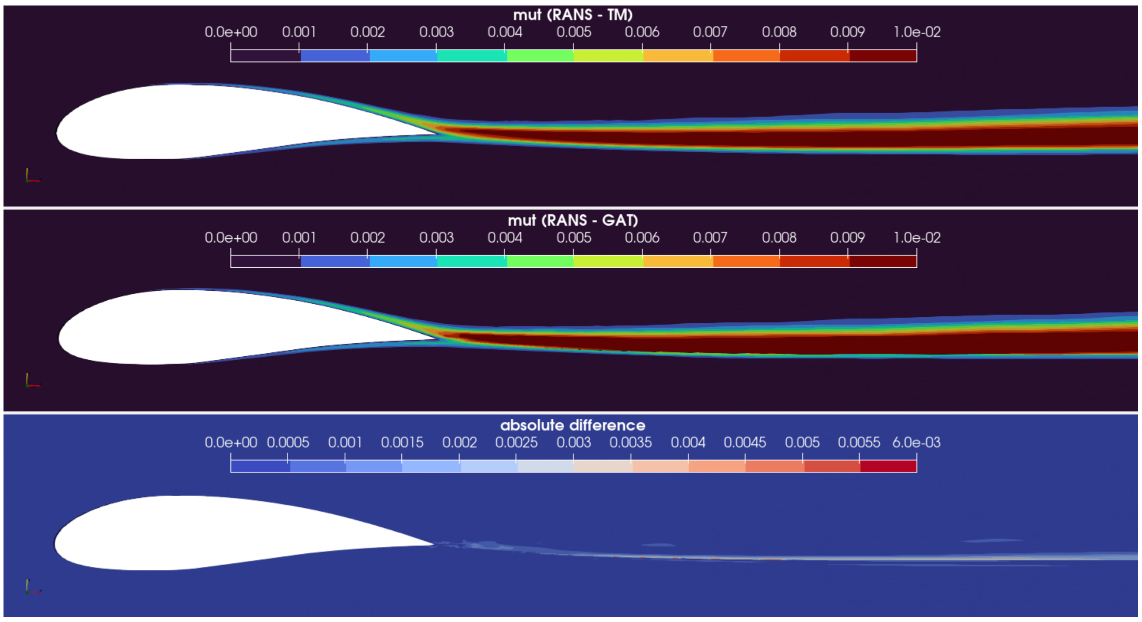


Figure 5.24: *NACA4318 Case: Comparison of turbulent viscosity fields obtained with RANS–TM (top), RANS–GAT (middle), and absolute difference (bottom).*

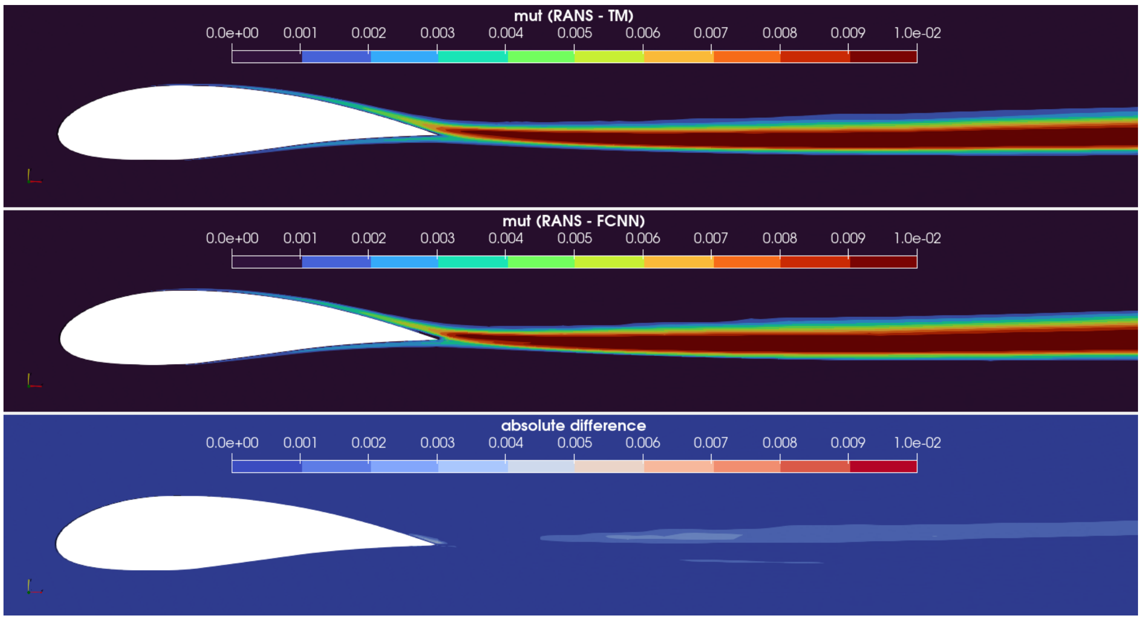


Figure 5.25: *NACA4318 Case: Comparison of turbulent viscosity fields obtained with RANS–TM (top), RANS–FCNN (middle), and absolute difference (bottom).*

The RANS–GAT solution accurately reproduces both the spatial distribution and magnitude of the turbulent viscosity. The absolute difference with respect to the RANS–TM reference remains localized and small relative to the peak values. By contrast, the RANS–FCNN prediction exhibits larger discrepancies in the wake region, with noticeably larger absolute errors.

5.5.3 Quantitative Error Analysis over the Validation Set

To assess the robustness of the proposed approaches, the relative errors in lift and drag coefficients are evaluated for all ten airfoils in the validation dataset. Figures 5.26 and 5.27 report the relative errors of RANS–FCNN and RANS–GAT with respect to the RANS–TM reference.

Across all validation cases, the GAT-based closure consistently yields lower relative errors than the FCNN-based approach for both C_L and C_D.

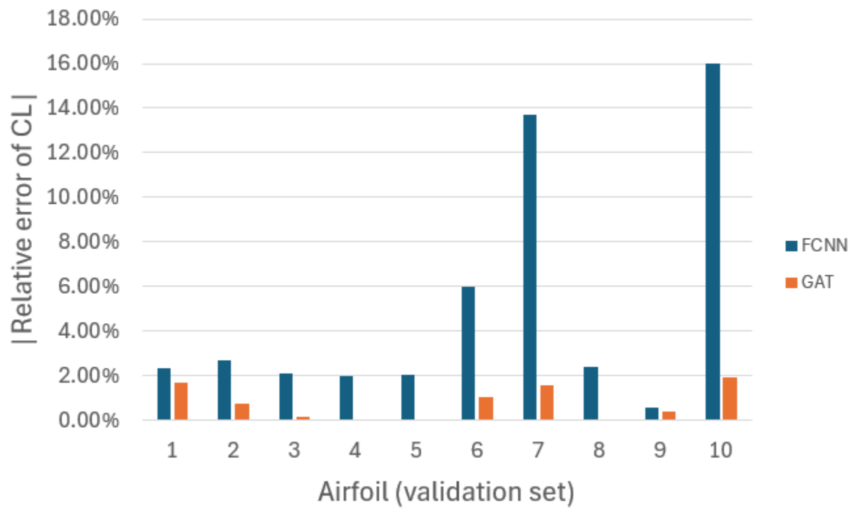


Figure 5.26: *NACA4318 Case: Relative error in lift coefficient C_L for RANS–FCNN and RANS–GAT across the validation dataset.*

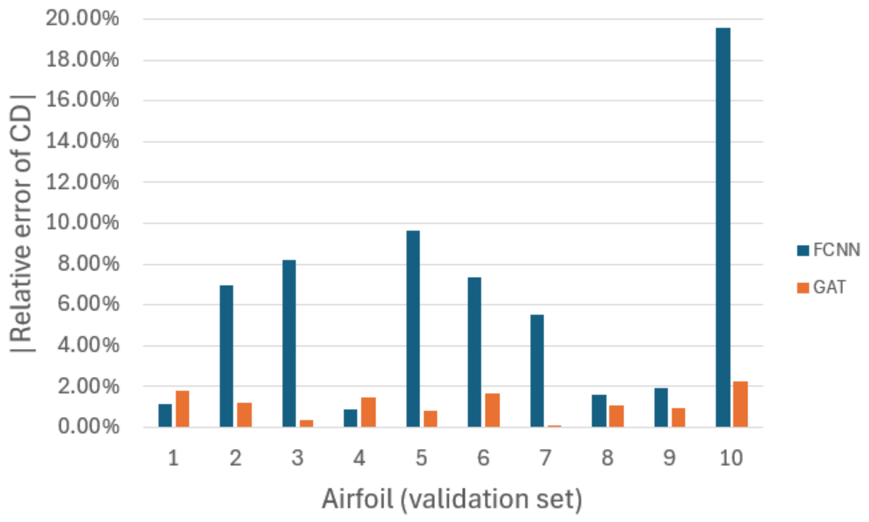


Figure 5.27: *NACA4318 Case: Relative error in drag coefficient C_D for RANS–FCNN and RANS–GAT across the validation dataset.*

5.6 Summary and Key Findings

This chapter examined the use of NNs for turbulence modeling in the turbulent flow over the isolated NACA4318 airfoil. The main findings can be summarized as follows:

- A dataset of 100 airfoil geometries was generated, and RANS-SA simulations were used to provide reference solutions for training and validation.
- Three neural network architectures were evaluated: an FCNN, a GAT, and an MPNN. The turbulent viscosity field was selected as the prediction target.
- On structured validation meshes, both GAT and MPNN outperformed the FCNN in terms of absolute, and relative error.
- Visualization of attention weights demonstrated that graph-based models exploit physically meaningful neighborhood information, particularly in the wake region, whereas the FCNN operates purely pointwise.
- Mesh-flexibility tests showed that the FCNN and GAT generalize more robustly to unseen mesh configurations, while the MPNN exhibits stronger sensitivity to changes in mesh topology.
- Integration of the FCNN and GAT models into the in-house CFD solver **PUMA** confirmed that NN-based turbulence closures can be coupled with a RANS solver, while retaining numerical convergence. The GAT closure consistently yielded more accurate predictions than the FCNN approach.

Overall, the results demonstrate that explicitly incorporating mesh connectivity through GNNs improves the accuracy and physical consistency of turbulence modeling compared to purely pointwise approaches.

Chapter 6

LS89 Turbine Blade Airfoil

6.1 Objective and Scope of the Case Study

This chapter extends the analysis presented for the NACA4318 airfoil to a more complex turbomachinery configuration, namely the transonic LS89 turbine blade airfoil.

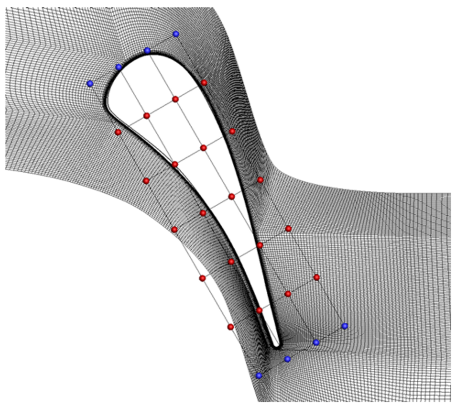
As in the previous chapter, the goal is to develop a NN-based turbulence closure capable of replacing the $k - \omega\ SST$ turbulence and the $\gamma - \tilde{Re}_{\theta t}$ transition models by directly predicting the turbulent viscosity field. The LS89 case serves as a more challenging benchmark, allowing the robustness and generalization capabilities of graph-based models to be assessed in internal aerodynamic flows as well.

6.2 LS89 Turbine Blade Airfoil Case Setup

6.2.1 Geometry Parameterization

The LS89 turbine blade airfoil geometry is parameterized using a NURBS box, following the methodology introduced in the airfoil case. As illustrated in Figure 6.1, the baseline blade profile is embedded within a 7×4 NURBS control box.

Control points shown in red are allowed to move both in the chordwise and normal-to-the-chord directions within $\pm 10\%$ of their reference positions. The eight control points located at the inlet and outlet boundaries (shown in blue) are constrained to move only in the normal-to-the-chord direction in order to preserve the inflow and outflow alignment of the blade. This parameterization results in a total of 48 design variables. Using this representation, a dataset of 150 geometrically distinct turbine blade airfoils is generated by sampling the design space within the prescribed bounds using Latin Hypercube Sampling (LHS). The resulting geometries are shown in Figure 6.1.



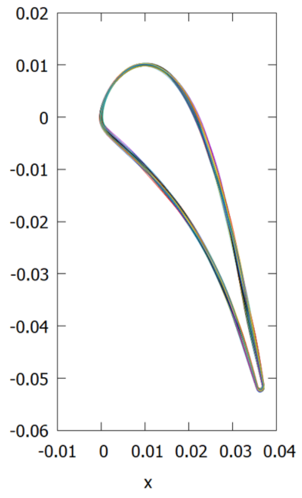


Figure 6.1: *LS89 case: NURBS-box parameterization of the turbine blade airfoil geometry. Red control points are free to move in both chordwise and normal-to-the-chord directions, while blue control points are constrained to normal displacement only. Adapted from [20] (left). Set of 150 geometries generated using LHS (right).*

6.2.2 Mesh Generation

A structured C-type computational mesh is generated for each turbine blade airfoil geometry. The mesh is strongly refined near the blade surface and in the trailing-edge wake region to accurately capture boundary-layer development and wake dynamics. An example mesh is shown in Figure 6.2.

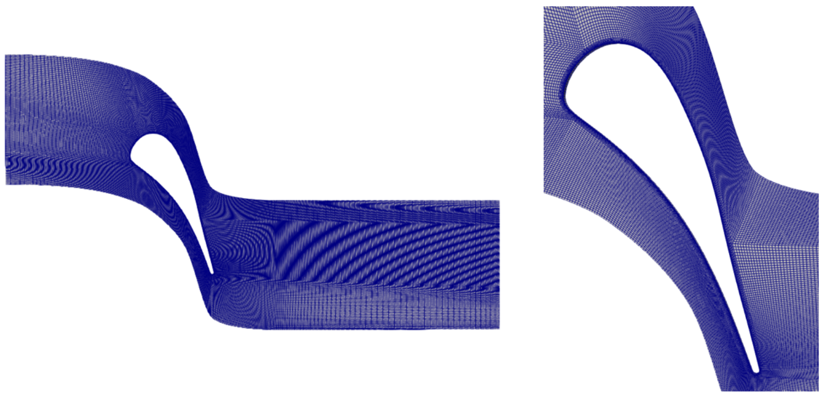


Figure 6.2: *LS89 Case: Example of the structured C-type computational mesh used for the turbine blade airfoil simulations. Left: full view of the computational domain. Right: close-up view near the blade surface.*

Each mesh consists of approximately 50000 nodes and 395000 edges, representing a substantially higher spatial resolution than in the airfoil case. The increased number of nodes and edges also increases the memory requirements during training and inference.

6.2.3 Flow Conditions

The LS89 turbine blade airfoil case considered in this work corresponds to a transonic turbine cascade configuration. The operating point is prescribed by the inlet total conditions, the outlet static pressure, and the inlet flow angle. The inlet turbulence level is specified through the turbulence intensity and the inlet viscosity ratio. All reference simulations are performed using the $k-\omega$ SST turbulence model coupled with the $\gamma-\tilde{Re}_{\theta t}$ transition model. The flow conditions are summarized in Table 6.1.

Quantity	Symbol	Value
Inlet Total Pressure (bar)	p_t^{in}	3.269
Inlet Total Temperature (K)	T_t^{in}	418.90
Outlet Static Pressure (bar)	p^{out}	1.550
Wall Static Temperature (K)	T^w	297.55
Inlet Flow Angle (°)	α^{in}	0.0
Inlet Turbulence Intensity (%)	Tu^{in}	0.8
Inlet Viscosity Ratio	$(\mu_t/\mu)^{in}$	11

Table 6.1: *LS89 Case: Inlet, outlet, and wall conditions.*

6.2.4 Flow Analysis

Figure 6.3 shows the Mach-number distribution in the blade passage. The flow accelerates rapidly along the suction side, reaching locally supersonic conditions ($M > 1$). Downstream, the supersonic region is terminated by a shock wave, visible as a sharp transition from supersonic to subsonic Mach numbers. This confirms the transonic nature of the flow at this operating point and indicates shock-related total-pressure losses.

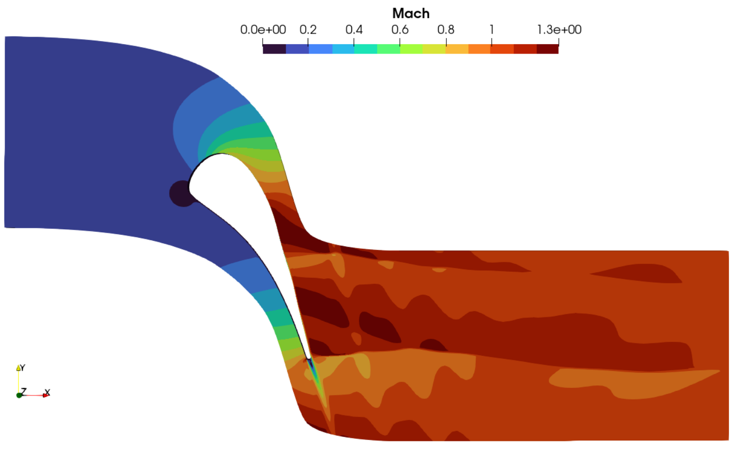


Figure 6.3: *LS89 case: Mach-number field obtained from the reference RANS simulation for the turbine blade airfoil geometry.*

6.3 Dataset Preparation

The selection of input and output quantities follows the same approach adopted in previous work [20], enabling direct comparison between FCNN and GNN.

The neural network inputs consist of nodal coordinates (x, y), local flow variables $(u, v, \rho, p, \Omega, S)$, and wall distance d. The target variable is the turbulent kinematic viscosity μ_t. An example reference distribution is shown in Figure 6.4, where high-value areas can be seen in the trailing-edge wake region.

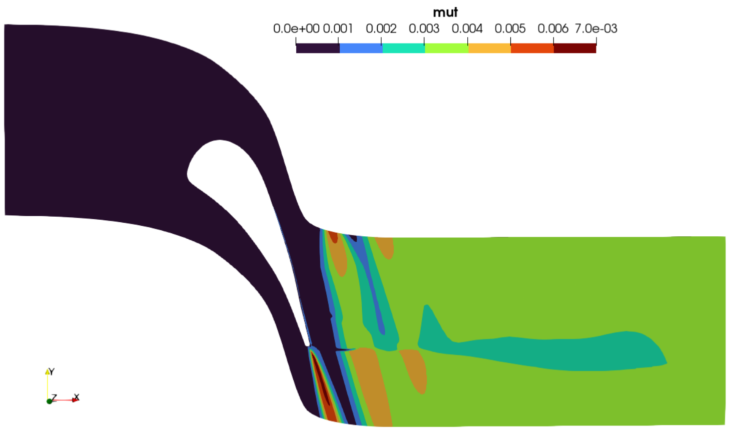


Figure 6.4: *LS89 Case: Turbulent viscosity μ_t obtained from the reference RANS simulation for the turbine blade airfoil geometry. Elevated values are concentrated in the trailing-edge wake region.*

All quantities are scaled using Min–Max normalization, following the procedure in Section 5.3.2. The normalization ranges are computed from the training data and summarized in Table 6.2. The dataset includes 120 training and 30 validation cases.

Quantity	Minimum	Maximum
x-coordinate	-0.0550	0.1640
y-coordinate	-0.0771	0.0451
Density ρ	1.0176	3.8288
Velocity component u	-119.1354	310.7952
Velocity component v	-496.3054	102.5835
Pressure p	86,763.89	327,012.37
Vorticity Ω	0.0	7.13×10^7
Strain rate S	3.44	7.06×10^7
Wall distance d	0.0	0.1305
Turbulent viscosity μ_t	0.0	0.01227

Table 6.2: *LS89 Case: Minimum and maximum values of the input and output quantities used for Min–Max normalization.*

6.4 Model Optimization and Training

Two NN architectures are evaluated in this case: an FCNN previously optimized in the PhD study [20] and a GAT optimized in the present work.

The optimized FCNN consists of five hidden layers with $[512, 512, 4096, 4096, 256]$ neurons. ReLU activation functions are used in the hidden layers, while a *tanh* activation is employed in the output layer. The GAT architecture is optimized following the same procedure as in the airfoil case and results in five hidden layers with $[1024, 128, 64, 64, 32]$ neurons, using the same activation functions as the FCNN.

After selecting the optimal architectures, both models are further trained until convergence. The minimum validation losses achieved are:

- FCNN: 9.9×10^{-3}
- GAT : 8.5×10^{-3}

6.5 Turbulent Viscosity Prediction

Figures 6.5 and 6.6 compare the turbulent viscosity predictions obtained with the FCNN and GAT models for a turbine blade airfoil case from the validation set. In both cases, the models correctly identify the wake region downstream of the trailing edge, where elevated turbulent viscosity values are expected.

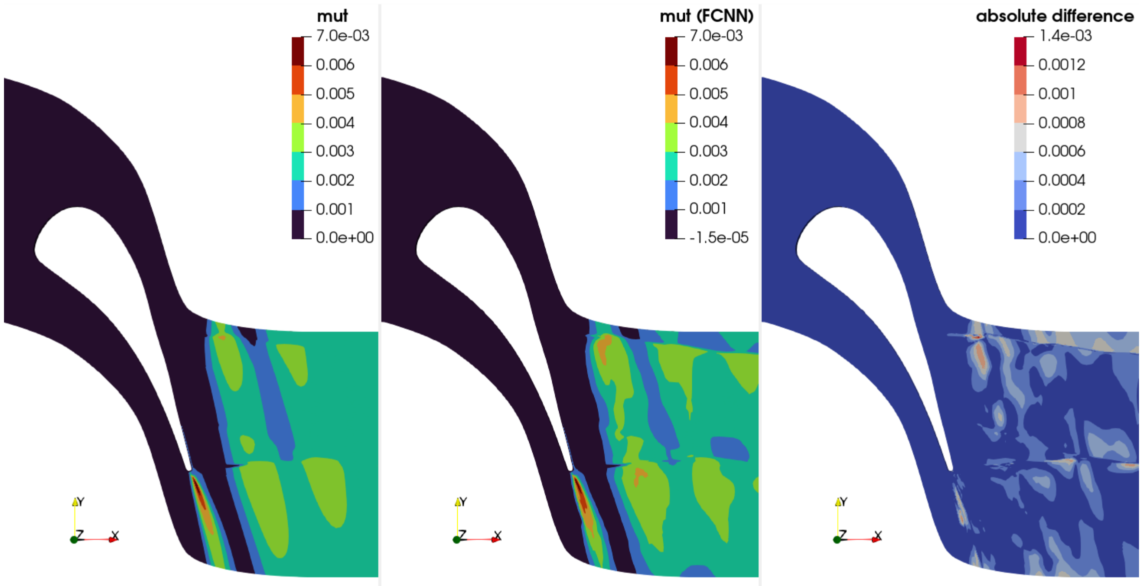


Figure 6.5: *LS89 Case: Comparison of the reference RANS turbulent viscosity field (left), FCNN prediction (middle), and absolute difference (right) for a representative turbine blade airfoil from the validation set.*

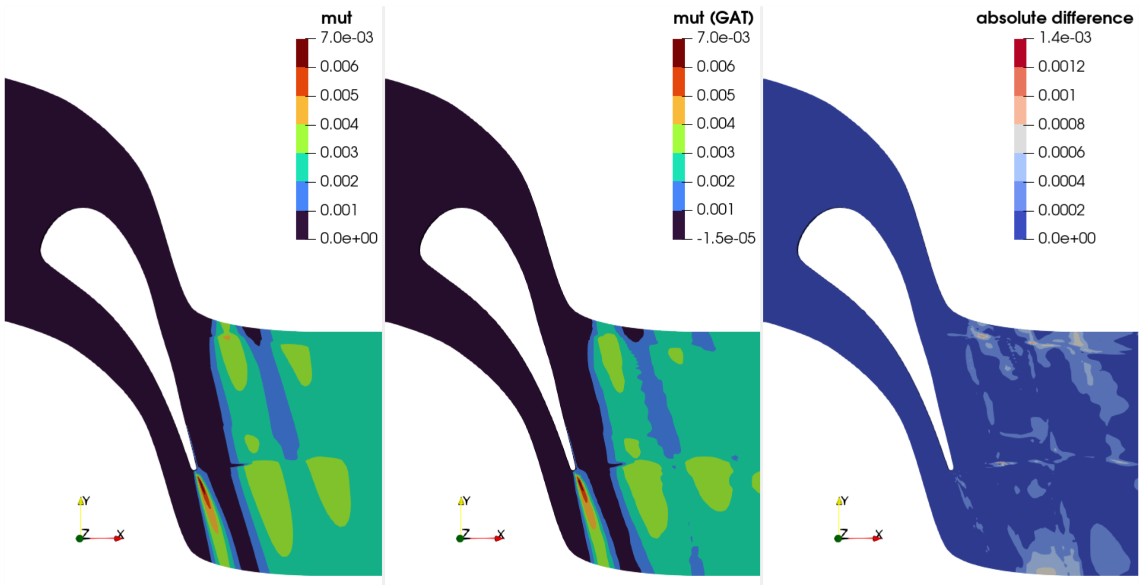


Figure 6.6: *LS89 Case: Comparison of the reference RANS turbulent viscosity field (left), GAT prediction (middle), and absolute difference (right) for a representative turbine blade airfoil from the validation set.*

To quantitatively assess performance across the validation set, Figures 6.7 and 6.8 report, for each of the 30 blade airfoils, the mean and maximum absolute errors achieved by the FCNN and GAT models. The dataset-level averages (dashed lines) are also shown to support comparison.

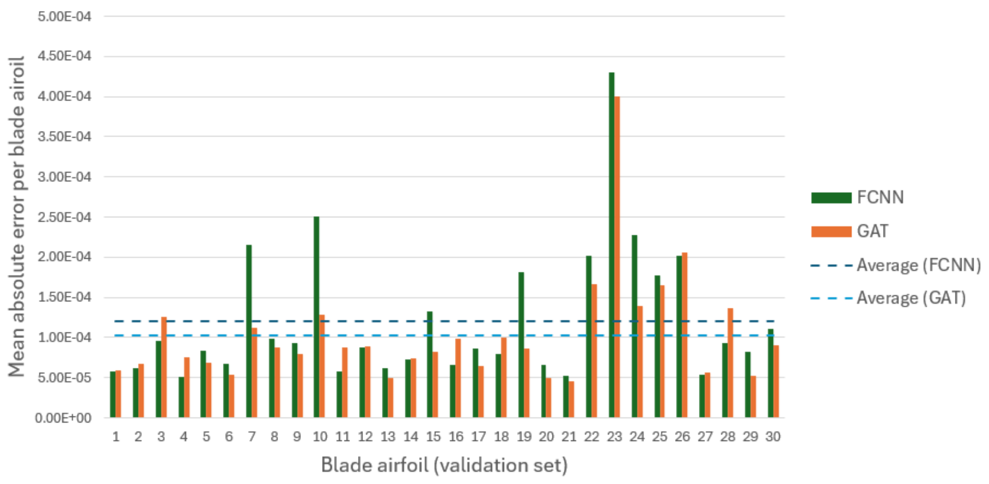


Figure 6.7: *LS89 Case: Mean absolute error of the turbulent viscosity prediction for each turbine blade airfoil in the validation set, comparing FCNN and GAT models.Dashed lines indicate averages over the 30 airfoils.*

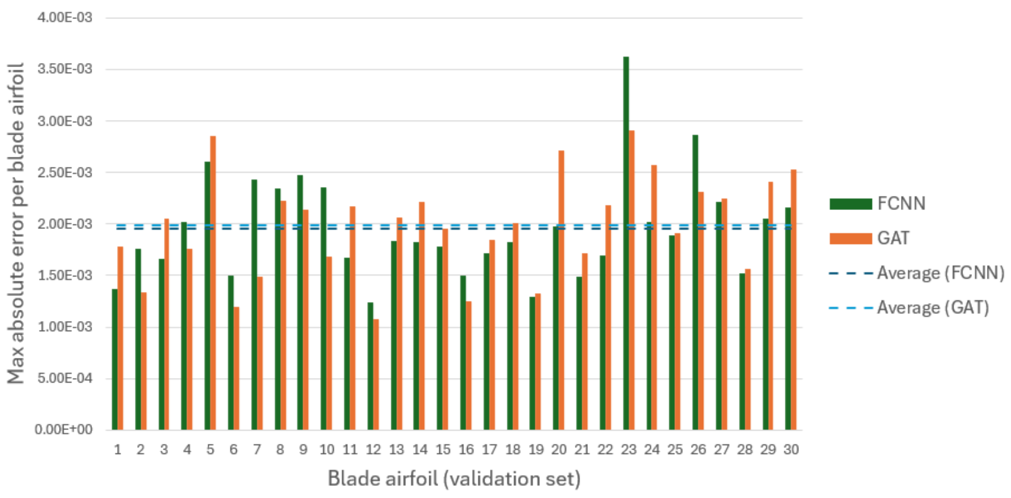


Figure 6.8: *LS89 Case: Maximum absolute error of the turbulent viscosity prediction for each turbine blade airfoil in the validation set, comparing FCNN and GAT models.Dashed lines indicate averages over the 30 airfoils.*

Overall, the GAT achieves a lower average mean absolute error than the FCNN, indicating slightly better accuracy. In terms of maximum absolute error, the two models perform similarly, with very close average values across the validation set.

Additionally, the relative error is computed following the same procedure used for the isolated NACA4318 airfoil case (see 5.4.2). The mean and maximum absolute relative errors for each of the 30 blade airfoils are illustrated in Figures 6.9 and 6.10.

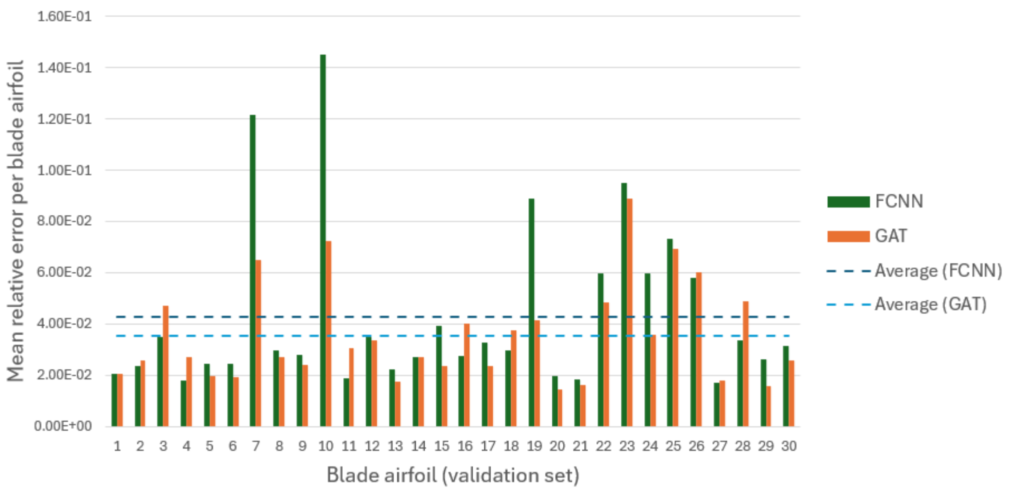


Figure 6.9: *LS89 Case: Mean absolute relative error of the turbulent viscosity prediction for each turbine blade airfoil in the validation set, comparing FCNN and GAT models. Dashed lines indicate averages over the 30 airfoils.*

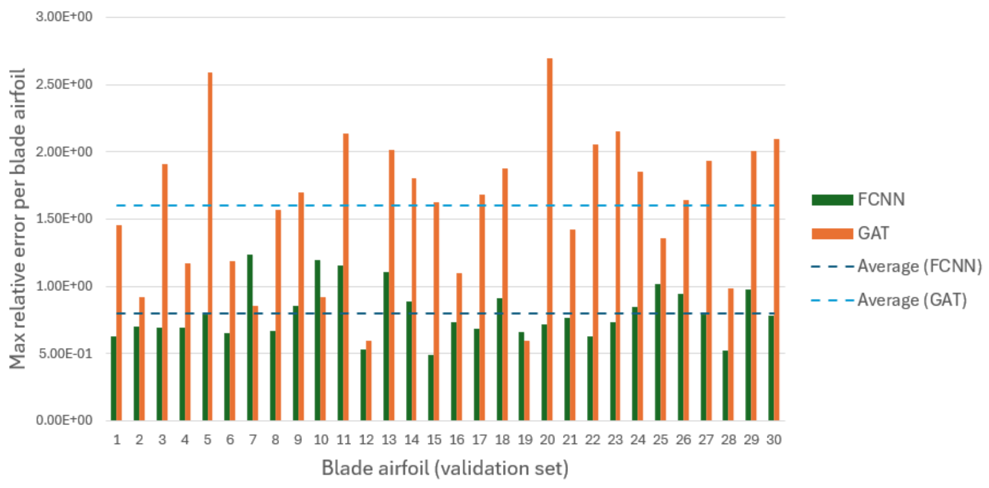


Figure 6.10: *LS89 Case: Max absolute relative error of the turbulent viscosity prediction for each turbine blade airfoil in the validation set, comparing FCNN and GAT models. Dashed lines indicate averages over the 30 airfoils.*

Overall, the GAT achieves a lower average mean relative error, indicating better accuracy across the validation set. In contrast, the FCNN attains a lower maximum relative error on average, suggesting slightly better worst-case performance.

The time for training and prediction of each model are summarized in Table 6.3.

Model	Training time [h]	Prediction time [s]
FCNN	15.5	14
GAT	5.5	16

Table 6.3: *LS89 Case: Training and prediction time comparison of FCNN and GAT.*

The FCNN model needs much more time for training due to the larger number of trainable parameters than GAT (20199425 parameters for FCNN, 158595 parameters for GAT). In the prediction time the data loading is including which requires more time for GAT than FCNN.

6.6 Integration of NNs into the CFD solver

Following Section 5.5, the trained NN-based surrogates are integrated into the CFD solver PUMA. In contrast to the previous setup, both the turbulence and the transition model are bypassed, and the required closure information is provided by the NN. Two solver configurations are considered: (i) *RANS–FCNN* and (ii) *RANS–GAT*, employing the FCNN and the GAT trained in the previous section.

6.6.1 Convergence of mass-averaged total pressure losses

Figure 6.11 presents the convergence history of the mass-averaged total pressure loss, $\Delta p_{t,m}$, for the reference simulation *RANS–TM* (RANS with turbulence and transition models) and the two NN-driven configurations. Both models converge to steady values. However, RANS–GAT shows a smaller deviation from the reference $\Delta p_{t,m}$, while RANS–FCNN reaches a higher plateau.

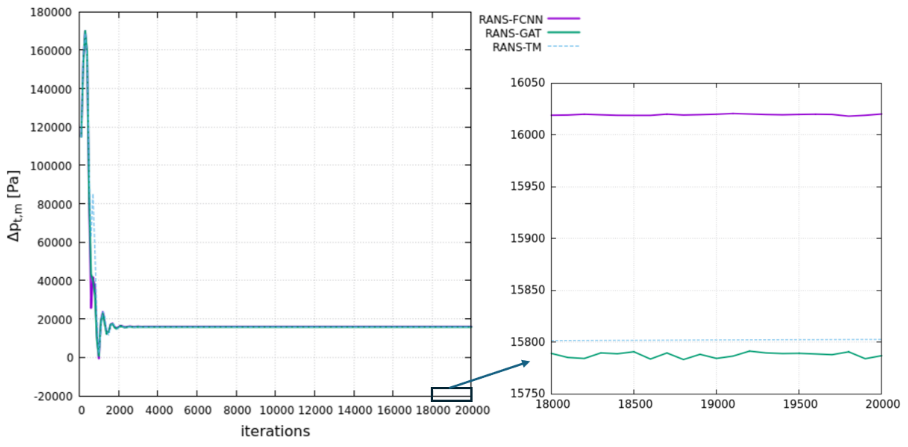


Figure 6.11: *LS89 case: Convergence of mass-averaged total pressure losses, $\Delta p_{t,m}$, for reference RANS–TM and NN-integrated RANS–FCNN and RANS–GAT.*

6.6.2 Comparison of turbulent viscosity field

Figures 6.12 and 6.13 compare the turbulent viscosity field, μ_t, obtained with the reference RANS–TM and with the NN-integrated simulations.

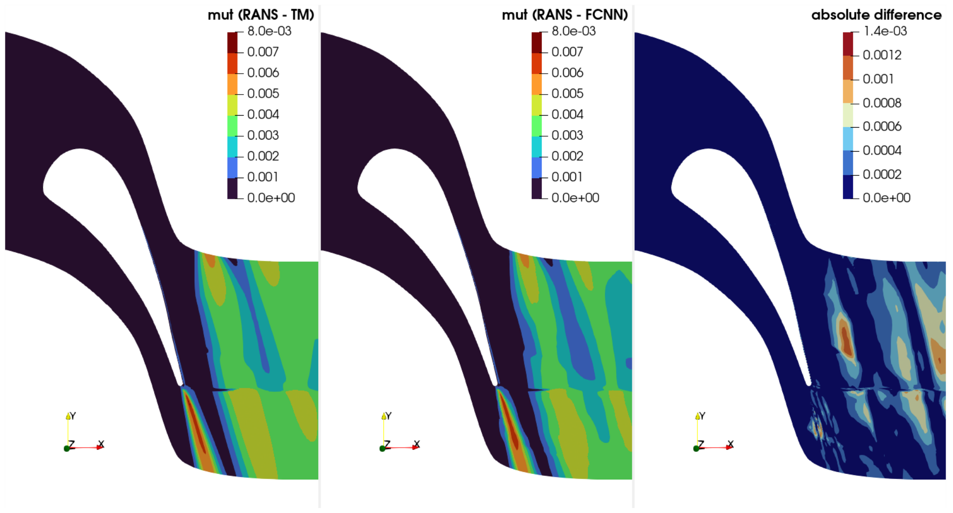


Figure 6.12: *LS89 Case: Turbulent viscosity field μ_t for RANS–TM (reference), RANS–FCNN, and the absolute difference.*

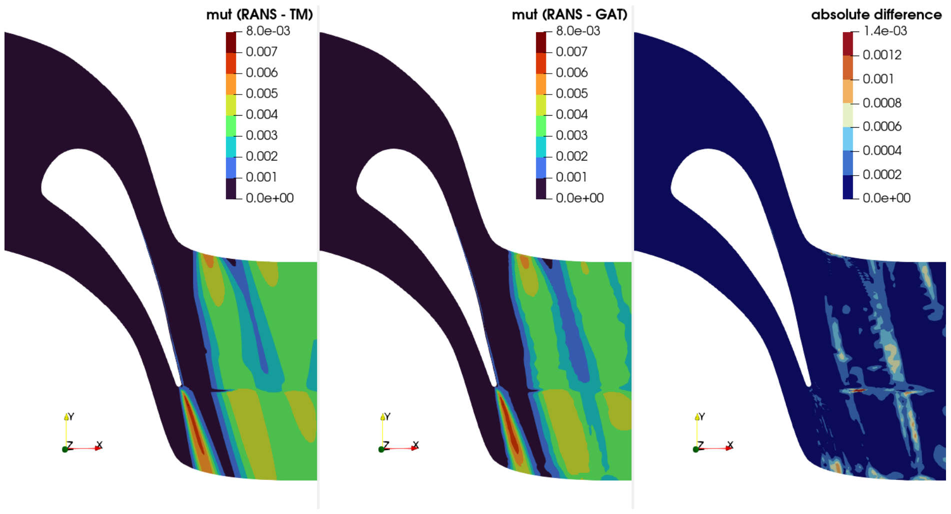


Figure 6.13: *LS89 Case: Turbulent viscosity field μ_t for RANS–TM (reference), RANS–GAT, and the absolute difference.*

Both approaches recover the overall spatial distribution of μ_t. The RANS–GAT prediction is in closer agreement with the reference field, while the RANS–FCNN solution exhibits larger discrepancies in several regions.

6.6.3 Quantitative error analysis over the validation set

To quantify the predictive performance, 12 blade airfoils from the validation set are simulated with the NN-integrated solver. For each case, the relative error of the mass-averaged total pressure loss is computed with respect to the reference RANS–TM solution and summarized in Figure 6.14.

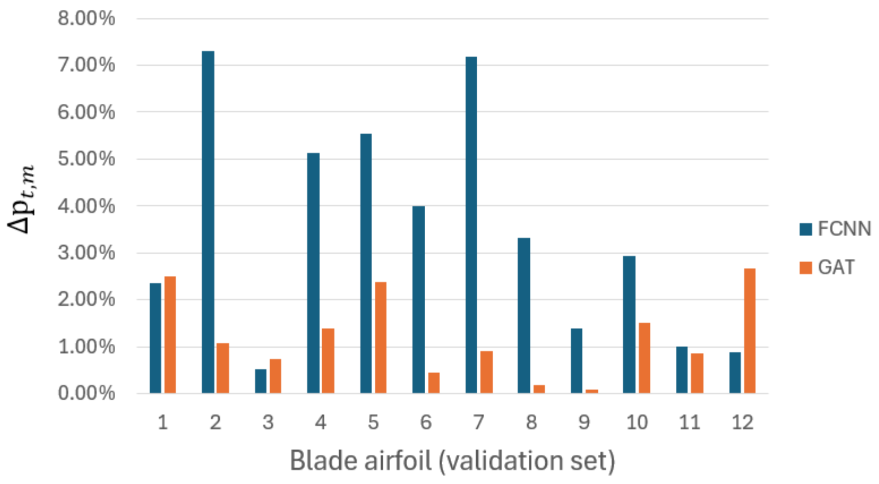


Figure 6.14: *LS89 Case: Relative error of $\Delta p_{t,m}$ with respect to the RANS–TM reference for 12 validation blade airfoils.*

Overall, RANS–GAT yields consistently lower errors than RANS–FCNN for most geometries, and in several cases the improvement is substantial.

6.7 Summary and Key Findings

This chapter examined the NN-based turbulence-closure framework for the transonic LS89 turbine cascade. The main findings are summarized as follows:

- A dataset of 150 geometrically distinct LS89 turbine blade airfoils was generated, and steady RANS simulations were used to produce reference solutions for training and validation.
- Two NN architectures were evaluated: a pointwise FCNN adopted from previous work and a GAT model optimized within this thesis. Both models captured the spatial distribution of the turbulent viscosity μ_t, particularly in the trailing-edge wake. Over the 30-airfoil validation set, the GAT achieved lower average mean absolute and mean relative errors, whereas the FCNN exhibited slightly lower worst-case (maximum) relative error on average.
- The trained surrogates were integrated into the in-house CFD solver **PUMA** by bypassing the turbulence and transition models and supplying the closure through NN-predicted μ_t. Both NN-driven configurations converged robustly to steady solutions for the investigated cases.
- In coupled CFD simulations, the GAT-based closure reproduced the reference mass-averaged total pressure loss $\Delta p_{t,m}$ more accurately than the FCNN-based closure, both in terms of convergence history and in relative-error comparisons across 12 validation blade simulations. Field-level comparisons further indicated closer agreement of the GAT predictions with the reference μ_t distribution over most of the domain.

Overall, the LS89 results reinforce the conclusions from the airfoil case: incorporating mesh connectivity through graph attention improves the accuracy and physical consistency of NN-based turbulence closures, and these benefits persist when the surrogate is deployed inside a transonic CFD solver where shocks, transition, and wake dynamics play a central role.

Chapter 7

Conclusions

This thesis investigates Graph Neural Networks (GNNs) as surrogate closure models for Reynolds-Averaged Navier–Stokes (RANS) simulations in aerodynamic analysis. The central hypothesis is that explicitly incorporating mesh connectivity and neighborhood information can improve the accuracy and physical consistency of data-driven turbulence closures compared with node-wise Fully Connected Neural Networks (FCNNs), which treat each mesh node independently.

Preliminary tests Preliminary tests were first conducted on a simplified regression task, namely the prediction of a two-variable function $z = f(x, y)$. These tests served three purposes. First, they validated the end-to-end pipeline for dataset generation, training, and evaluation in a controlled setting. Second, they highlighted a fundamental difference between models: while FCNNs operate purely on nodes, Graph Attention Networks (GATs) operate on nodes and edges, and their computational cost increases more strongly with mesh refinement due to the growth in the number of edges. Third, in this simplified setting the FCNN outperformed the GAT, because the mesh geometry was fixed and the target field was fully determined from the local inputs (x, y) and the function coefficients, leaving limited benefit for neighborhood aggregation.

NACA4318 isolated airfoil The first CFD case study focused on turbulent flow over an isolated NACA4318 airfoil. The baseline airfoil was parameterized to generate a dataset of geometrically distinct airfoils, for which RANS simulations provided reference solutions. The learning objective was to replace the turbulence model by directly predicting the turbulent viscosity field from geometric and local flow features. Optimized FCNN, GAT, and MPNN architectures were compared using field-level accuracy (absolute and relative errors) and qualitative agreement of the wake structure. On structured validation meshes, the graph-based models (GAT and MPNN) achieved consistently lower errors and produced smoother, more physically plausible wake predictions than the FCNN.

Generalization to unseen mesh configurations was then assessed through two mesh-flexibility tests: (i) unstructured triangulations with identical farfield extent and node count and (ii) unstructured meshes with different farfield extent and node count. In both tests, the FCNN and GAT preserved a reasonable wake structure, whereas the MPNN produced large turbulent-viscosity values and, in the second test, a qualitatively incorrect wake. Based on this sensitivity to mesh topology, the MPNN was not considered in subsequent CFD-coupled experiments. Finally, the FCNN and GAT closures were integrated into the in-house CFD solver **PUMA** by replacing the turbulence model. Both configurations converged robustly. The GAT-based closure achieved closer agreement with the reference RANS solutions, both in terms of turbulent-viscosity fields and aerodynamic outputs, as quantified by the relative errors of lift and drag coefficients over 10 validation airfoils.

LS89 turbine blade airfoil The second CFD case study considered the transonic LS89 turbine blade cascade, which introduces stronger pressure gradients, shock-wave, transition effects, and more complex wake dynamics. The same workflow was applied, using FCNN and GAT models. In this case, the NN-based closure bypassed both the turbulence and transition models in **PUMA** by predicting the turbulent viscosity field directly. In offline evaluation, the GAT achieved lower average errors across the validation set, while the FCNN exhibited slightly better worst-case (maximum) relative-error performance. In coupled CFD simulations, both NN-driven configurations converged. The GAT-based closure reproduced the reference mass-averaged total pressure loss more accurately and yielded turbulent-viscosity fields closer to the reference, as confirmed by the relative-error assessment over 12 validation blade airfoils.

Overall, the results demonstrate that incorporating mesh connectivity through graph attention improves the predictive accuracy and physical consistency of NN-based turbulence closures, and that these benefits persist when the surrogate is deployed inside a CFD solver for both external and internal aerodynamic flows.

Future Work Based on the findings of this diploma thesis, the following directions are proposed:

- **Expand dataset diversity in geometry and operating conditions:** The two CFD datasets were adopted from previous work [20] to enable direct comparison with the established FCNN configuration. However, the geometry perturbations are bounded around a baseline shape and the flow conditions are kept fixed for each case. This restricted variability may limit the scenarios in which neighborhood aggregation provides a strong advantage and can therefore favour pointwise models. Future work should consider constructing richer datasets with broader geometric variations and multiple operating points to better assess robustness and generalization.

- **Evaluate performance on unstructured meshes:** A key practical advantage of graph-based models is their ability to operate directly on unstructured discretizations. In the present work, training and validation were primarily performed on structured meshes, and mesh-flexibility was assessed only through limited tests. Future studies should generate and include fully unstructured-mesh datasets and evaluate training, accuracy, and stability of the GAT-based closure directly on such meshes.

Bibliography

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

[2] Aggarwal, C.: Neural Networks and Deep Learning: A Textbook. Springer (2023). https://doi.org/10.1007/978-3-319-94463-0

[3] Bishop, C.: Pattern Recognition and Machine Learning. Springer (2007). https://doi.org/10.5555/1162264

[4] Chen, J., Hachem, E., Viquerat, J.: Graph neural networks for laminar flow prediction around random 2d shapes (2021). https://doi.org/arXiv:2107.11529

[5] CleverTap: How do neural networks mimic the human brain? (2019), `https://clevertap.com/blog/neural-networks-brain/`

[6] Daigavane, A., Ravindran, B., Aggarwal, G.: Understanding convolutions on graphs. Distill (2021). https://doi.org/10.23915/distill.00032, https://distill.pub/2021/understanding-gnns

[7] Dubey, S., Singh, S., Chaudhuri, B.: Activation functions in deep learning: A comprehensive survey and benchmark. arXiv preprint arXiv:2109.14545 (2022). https://doi.org/10.48550/arXiv.2109.14545, https://arxiv.org/abs/2109.14545

[8] Elharrouss, O., Mahmood, Y., Bechqito, Y., Serhani, M., Badidi, E., Riffi, J., Tairi, H.: Loss functions in deep learning: A comprehensive review. arXiv preprint arXiv:2504.04242v1 (2025), http://arxiv.org/html/2504.04242v1

[9] Giannakoglou, K.C.: Computational methods in turbomachines. Tech. rep., National Technical University of Athens (NTUA), Athens, Greece (2004), `http://147.102.55.162/kgianna/vft/distr/YMS-book.pdf`

[10] Giannakoglou, K.C.: Optimization methods in aerodynamics. Tech. rep., National Technical University of Athens (NTUA), Athens, Greece (2006), http://velos0.ltt.mech.ntua.gr/kgianna/optim/distr/Book-Optim-Giannakoglou.pdf

[11] Giannakoglou, K.C.: EASY: The Evolutionary Algorithm SYstem, v2.0. PCOpt/NTUA (2008), http://velos0.ltt.mech.ntua.gr/EASY/

[12] Gilmer, J., Schoenholz, S., Riley, P., Vinyals, O., Dahl, G.: Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212 (2017). https://doi.org/10.48550/arXiv.1704.01212, https://arxiv.org/abs/1704.01212

[13] Grattarola, D., A.C.: Graph neural networks in tensorflow and keras with spektral. arXiv preprint arXiv:2006.12138 (2020). https://doi.org/10.48550/arXiv.2006.12138, https://arxiv.org/abs/2006.12138

[14] Grattarola, D.: A practical introduction to gnns - part 1 (2021), https://danielegrattarola.github.io/posts/2021-03-03/gnn-lecture-part-1.html

[15] Grattarola, D.: A practical introduction to gnns - part 2 (2021), https://danielegrattarola.github.io/posts/2021-03-12/gnn-lecture-part-2.html

[16] Han, X., Gao, H., Pfaff, T., Wang, J., L.P., L.: Predicting physics in mesh-reduced space with temporal attention (2022). https://doi.org/arXiv:2201.09113

[17] Jordan, J.: Neural networks: training with backpropagation. (2017), https://www.jeremyjordan.me/neural-networks-training/

[18] Kipf, T., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2017). https://doi.org/10.48550/arXiv.1609.02907, https://arxiv.org/abs/1609.02907

[19] Kochkov, D., Jamie, A., Smith, J., Alieva, A., Wang, Q., Brenner, M., Hoyer, S.: Machine learning accelerated computational fluid dynamics (2021). https://doi.org/arXiv:2102.01010

[20] Kontou, M.: The Continuous Adjoint Method with Consistent Discretization Schemes for Transitional Flows and the Use of Deep Neural Networks in Shape Optimization in Fluid Mechanics. Ph.D. thesis, National Technical University of Athens, Athens, Greece (2023), https://dspace.lib.ntua.gr/xmlui/handle/123456789/58743

[21] Kriesel, D.: A Brief Introduction to Neural Networks (2007), availableathttp://www.dkriesel.com

[22] NASA Langley Research Center: Turbulence modeling resource: Spalart–allmaras (sa) model. https://turbmodels.larc.nasa.gov/spalart.html

[23] Obiols-Sales, O., Vishnu, A., Malaya, N., Chandramowlishwaran, A.: Cfdnet: a deep learning-based accelerator for fluid simulations (2020). https://doi.org/arXiv:2005.04485

[24] Rosales, A.: Graph neural networks using pytorch (2024), https://medium.com/@andrea.rosales08/introduction-to-graph-neural-networks-78cbb6f64011

[25] Sanchez-Lengeling, B., Reif, E., Pearce, A., Wiltschko, A.B.: A gentle introduction to graph neural networks. Distill (2021). https://doi.org/10.23915/distill.00033, https://distill.pub/2021/gnn-intro

[26] Tsimenidis, S.: Limitations of deep neural networks: a discussion of g. marcus' critical appraisal of deep learning (2020), https://arxiv.org/pdf/2012.15754

[27] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2018). https://doi.org/10.48550/arXiv.1710.10903, https://arxiv.org/abs/1710.10903

[28] Wu, X., Ajorlou, A., Wu, Z., Jadbabaie, A.: Demystifying oversmoothing in attention-based graph neural networks (2023), https://openreview.net/pdf?id=Kg65qieiuB

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστομηχανικής
& Βελτιστοποίησης

Νευρωνικά δίκτυα τύπου γράφου ως υποκατάστατα μοντέλων τύρβης και μετάβασης στην αεροδυναμική

Διπλωματική Εργασία - Εκτενής Περίληψη στα Ελληνικά

Γιουντέρη Λίντα Αναστασία

Επιβλέποντες:

Κυριάκος Χ.Γιαννάκογλου, Καθηγητής ΕΜΠ

Δρ. Β. Ασούτη, Εντεταλμένη Διδάσκουσα ΕΜΠ

Αθήνα, 2026

Εισαγωγή

Στην παρούσα διπλωματική εργασία, πρωταρχικός στόχος είναι η διερεύνηση της χρήσης Νευρωνικών Δικτύων Γράφων (Graph Neural Networks - GNNs), σε σύγκριση με Πλήρως Συνδεδεμένα Νευρωνικά Δίκτυα (Fully Connected Neural Networks - FCNNs), ως υποκατάστατων των μοντέλων τύρβης και μετάβασης στην αεροδυναμική ανάλυση.

Εξετάζονται δύο διαφορετικές περιπτώσεις ΥΡΔ: η εξωτερική αεροδυναμική γύρω από μία μεμονωμένη αεροτομή NACA4318 και η εσωτερική αεροδυναμική σε διηχητική 2Δ πτερύγωση στροβίλου LS89. Στις περιπτώσεις αυτές, GNN και FCNN εκπαιδεύονται να προβλέπουν το πεδίο της τυρβώδους συνεκτικότητας, με είσοδο γεωμετρικά και ροϊκά πεδία για διαφορετικές γεωμετρίες, διατηρώντας τις συνθήκες ροής σταθερές.

Τα GNN και τα FCNN αξιολογούνται ως προς (i) την ακρίβεια πρόβλεψης του πεδίου τυρβώδους συνεκτικότητας εκτός του λογισμικού ΥΔΡ του εργαστηρίου, PUMA και (ii) την επίδρασή τους στο τελικό αποτέλεσμα όταν ενσωματώνονται εντός του PUMA.

Πλήρως Συνδεδεμένα Νευρωνικά Δίκτυα

Τα Πλήρως Συνδεδεμένα Νευρωνικά Δίκτυα (FCNN) αποτελούν μία από τις απλούστερες κατηγορίες τεχνητών νευρωνικών δικτύων. Οργανώνονται σε διαδοχικά επίπεδα, όπου κάθε επίπεδο περιλαμβάνει έναν αριθμό νευρώνων. Οι νευρώνες του ίδιου επιπέδου δεν συνδέονται μεταξύ τους, αλλά συνδέονται με τους νευρώνες του επόμενου επιπέδου. Ονομάζονται *πλήρως συνδεδεμένα*, διότι κάθε νευρώνας ενός επιπέδου συνδέεται με όλους τους νευρώνες του επόμενου.

Οι συνδέσεις αυτές αντιστοιχούν σε μαθηματικές πράξεις που εκτελεί το δίκτυο ώστε, για δεδομένη είσοδο, να παραχθεί η αντίστοιχη έξοδος. Οι πράξεις καθορίζονται από παραμέτρους που ονομάζονται *βάρη*. Ο προσδιορισμός τους γίνεται μέσω εκπαίδευσης, κατά την οποία χρησιμοποιούνται ζεύγη δεδομένων εισόδου–εξόδου. Η είσοδος προωθείται στο δίκτυο, υπολογίζεται η έξοδος με βάση μία αρχικοποίηση των βαρών και στη συνέχεια εκτιμάται το σφάλμα σε σχέση με την πραγματική τιμή. Μέσω διαδικασίας βελτιστοποίησης, τα βάρη ενημερώνονται επαναληπτικά με στόχο την ελαχιστοποίηση του σφάλματος πρόβλεψης.

Στο πλαίσιο εφαρμογών ΥΡΔ σε πλέγματα, ένα FCNN μπορεί να χρησιμοποιηθεί με δύο βασικούς τρόπους. Έστω ότι κάθε κόμβος του πλέγματος περιγράφεται από δύο χαρακτηριστικά (π.χ. συντεταγμένες x, y) και ζητείται ως έξοδος ένα μέγεθος ανά κόμβο (π.χ. η συνιστώσα ταχύτητας u). Αν το πλέγμα διαθέτει 100 κόμβους, τότε: (i) το δίκτυο μπορεί να λαμβάνει ως είσοδο τα χαρακτηριστικά κάθε κόμβου και να προβλέπει την αντίστοιχη έξοδο *ανά κόμβο* ή (ii) μπορεί να λαμβάνει ως είσοδο τα χαρακτηριστικά *όλων* των κόμβων ταυτόχρονα (200 χαρακτηριστικά) και να επιστρέφει ως έξοδο τις 100 ζητούμενες τιμές.

Η δεύτερη προσέγγιση παρουσιάζει δύο σημαντικά μειονεκτήματα: πρώτον, ο αριθμός των παραμέτρων αυξάνεται σημαντικά, γεγονός που επιβαρύνει τη μνήμη και το υπολογιστικό κόστος, ιδιαίτερα για μεγάλα πλέγματα ΥΡΔ· δεύτερον, το μοντέλο περιορίζεται σε πλέγματα με σταθερό αριθμό κόμβων, καθώς η διάσταση της εισόδου είναι συγκεκριμένη. Αντίθετα, η πρώτη προσέγγιση είναι πιο πρακτική και γενικεύσιμη, όμως δεν ενσωματώνει την τοπολογία του πλέγματος και τη συσχέτιση με τους γειτονικούς κόμβους, καθώς κάθε κόμβος επεξεργάζεται ανεξάρτητα. Το συγκεκριμένο μειονέκτημα έρχονται να αντιμετωπίσουν τα Νευρωνικά Δίκτυα Γράφων.

Νευρωνικά Δίκτυα Γράφων

Τα Νευρωνικά Δίκτυα Γράφων (GNN) εφαρμόζονται, όπως υποδηλώνει και το όνομά τους, σε δομές γράφων. Ένας γράφος αποτελείται από κόμβους και ακμές (συνδέσεις) μεταξύ των κόμβων. Τόσο οι κόμβοι όσο και οι ακμές μπορούν να φέρουν χαρακτηριστικά (γνωρίσματα), τα οποία χρησιμοποιούνται ως είσοδοι για την πρόβλεψη ζητούμενων μεγεθών.

Αντίστοιχα με τα FCNN, τα GNN οργανώνονται σε διαδοχικά επίπεδα. Ωστόσο, σε κάθε επίπεδο η αναπαράσταση ενός κόμβου δεν ενημερώνεται μόνο από τα δικά του χαρακτηριστικά, αλλά και από την πληροφορία των γειτονικών κόμβων με τους οποίους συνδέεται. Η βασική διεργασία με την οποία συλλέγεται και συνδυάζεται αυτή η πληροφορία ονομάζεται *διάδοση μηνυμάτων* (message passing). Κατά τη διάδοση μηνυμάτων, κάθε κόμβος συγκεντρώνει πληροφορία από τους γείτονές του (και, συνήθως, και από τον ίδιο τον κόμβο μέσω αυτο-σύνδεσης) και παράγει μία ενημερωμένη αναπαράσταση, η οποία χρησιμοποιείται στο επόμενο επίπεδο.

Υπάρχουν αρχιτεκτονικές GNN όπου όλες οι εισερχόμενες πληροφορίες από τους γείτονες αντιμετωπίζονται ισότιμα μέσω μιας μορφής συγκέντρωσης (π.χ. άθροιση ή μέσος όρος). Αν και τα μοντέλα αυτά είναι σχετικά απλά, συχνά παρουσιάζουν περιορισμένη εκφραστικότητα, ιδίως σε μη δομημένα πλέγματα, όπου η συνεισφορά διαφορετικών γειτόνων δεν είναι κατ᾽ ανάγκη ισοδύναμη. Το μειονέκτημα αυτό αντιμετωπίζεται από μοντέλα που ενσωματώνουν *μηχανισμούς προσοχής* (attention), οι οποίοι αποδίδουν διαφορετικά βάρη στη συνεισφορά κάθε γειτονικού κόμβου. Οι συντελεστές προσοχής προκύπτουν από επιπρόσθετες εκπαιδεύσιμες παραμέτρους και επιτρέπουν στο δίκτυο να μαθαίνει ποιοι γείτονες είναι σημαντικότεροι για την εκάστοτε πρόβλεψη. Όπως και στην περίπτωση των FCNN (όταν η πρόβλεψη γίνεται ανά κόμβο), τα βάρη και οι επιπρόσθετες παράμετροι των GNN είναι κοινά για όλους τους κόμβους.

Στο πλαίσιο πλεγμάτων ΥΡΔ, όταν το πλέγμα μεταβάλλεται (π.χ. λόγω αλλαγής γεωμετρίας), είναι κρίσιμη η αξιοποίηση της συσχέτισης με τους γειτονικούς κόμβους, ώστε η πρόβλεψη να ενσωματώνει τοπική πληροφορία σχετική με τη γεωμετρία και τη ροή. Επιπλέον, η λειτουργία των GNN παρουσιάζει αντιστοιχία με τον τρόπο διακριτοποίησης και επίλυσης των εξισώσεων στη ΥΡΔ, όπου η πληροφορία ανταλλάσσεται τοπικά μεταξύ γειτονικών κελιών/κόμβων.

Παρότι τα GNN είναι ιδιαίτερα ευέλικτα, συνοδεύονται από ορισμένους πρακτικούς περιορισμούς. Ειδικότερα, παρουσιάζουν αυξημένες απαιτήσεις μνήμης λόγω της ανάγκης αποθήκευσης και διαχείρισης της συνδεσιμότητας (ακμών) και, ανάλογα με την αρχιτεκτονική, επιπρόσθετων εκπαιδεύσιμων παραμέτρων. Επιπλέον, το βάθος του δικτύου πρέπει να επιλέγεται προσεκτικά, ώστε να αποφεύγεται η *υπερ-εξομάλυνση* (over-smoothing) των αναπαραστάσεων: καθώς αυξάνεται ο αριθμός επιπέδων, διευρύνεται η γειτονιά από την οποία αντλείται πληροφορία και οι αναπαραστάσεις των κόμβων τείνουν να γίνονται πιο όμοιες. Παρά τα παραπάνω, τα GNN μπορούν να εφαρμοστούν σε μη δομημένα πλέγματα και να χειριστούν γράφους με διαφορετικό αριθμό κόμβων, κάτι ιδιαίτερα χρήσιμο σε εφαρμογές ΥΡΔ.

Μεμονωμένη αεροτομή NACA4318

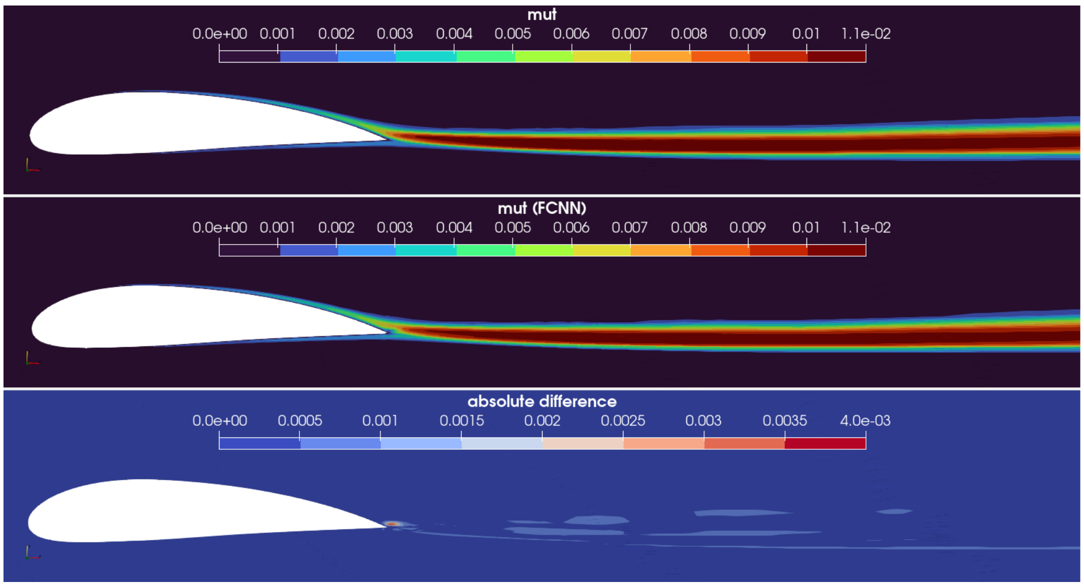
Η μεμονωμένη αεροτομή NACA4318 χρησιμοποιείται ως αεροτομή αναφοράς και παραμετροποιείται με ογκομετρικές NURBS, όπου 13 σημεία ελέγχου επιτρέπεται να μετακινηθούν κατά $\pm 10\%$ της αρχικής τους θέσης. Με τη μέθοδο Latin Hypercube Sampling (LHS) δημιουργούνται 100 διαφορετικές γεωμετρίες. Για κάθε γεωμετρία παράγονται δομημένα πλέγματα τύπου C, με ίδιο αριθμό κόμβων.

Για τη συλλογή των δεδομένων αναφοράς πραγματοποιούνται προσομοιώσεις ΥΡΔ με τον κώδικα του εργαστηρίου (PUMA), όπου επιλύονται οι εξισώσεις ροής με το μοντέλο τύρβης Spalart–Allmaras. Οι συνθήκες ροής είναι $\mathbf{M_\infty = 0.13}$, $\mathbf{Re = 3.8 \times 10^6}$ και γωνία πρόσπτωσης $\alpha = \mathbf{2.2^\circ}$, συνθήκες υπό τις οποίες η αεροτομή παράγει άνωση.

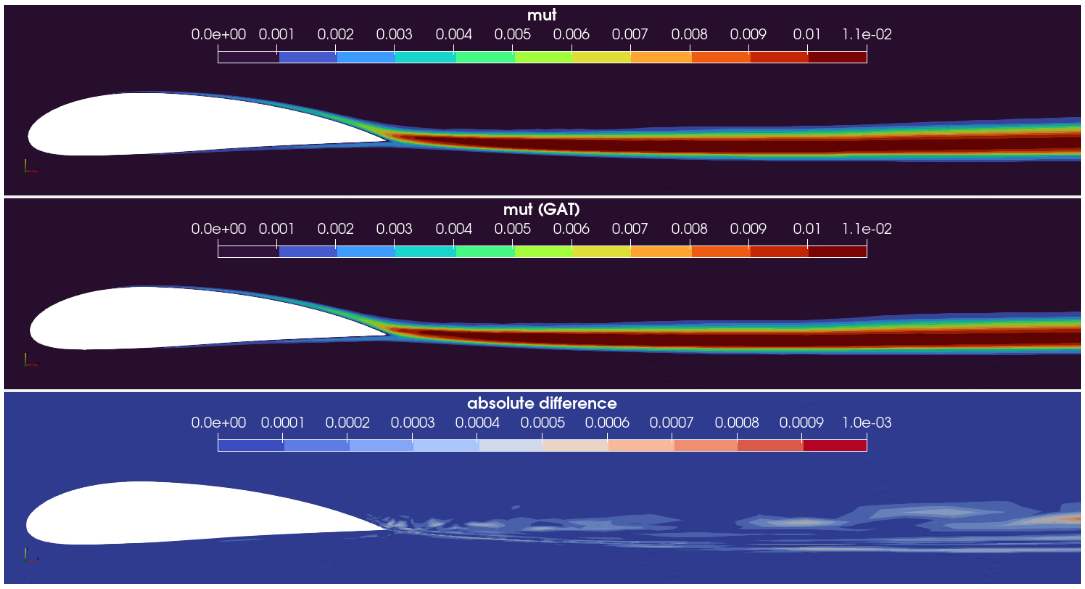
Ως είσοδοι στα μοντέλα επιλέγονται ροϊκές ποσότητες αντίστοιχες με αυτές που χρησιμοποιούνται στη διδακτορική διατριβή [20], ώστε να καταστεί δυνατή η άμεση σύγκριση του FCNN με το αντίστοιχο GNN. Συγκεκριμένα, ως είσοδοι χρησιμοποιούνται οι συντεταγμένες x, y, οι συνιστώσες της ταχύτητας u, v, η πίεση p, η στροβιλότητα Ω, ο ρυθμός παραμόρφωσης S και η απόσταση από το τοίχωμα d. Ως ζητούμενη έξοδος ορίζεται η τυρβώδης συνεκτικότητα μ_t. Τα ζεύγη εισόδου–εξόδου κανονικοποιούνται ως προς τις ελάχιστες και μέγιστες τιμές τους, καθώς τα μεγέθη εμφανίζουν σημαντικά διαφορετικές τάξεις μεγέθους.

Για την εκπαίδευση χρησιμοποιούνται 90 γεωμετρίες, ενώ 10 γεωμετρίες (αεροτομές) διατηρούνται για την αξιολόγηση. Η αρχιτεκτονική του FCNN (αριθμός επιπέδων, αριθμός νευρώνων ανά επίπεδο και συναρτήσεις ενεργοποίησης) έχει βελτιστοποιηθεί στη [20]. Στην παρούσα εργασία βελτιστοποιείται η αρχιτεκτονική του GNN. Το βέλτιστο FCNN διαθέτει 6 κρυφά επίπεδα με αριθμό νευρώνων ανά επίπεδο $[64, 128, 256, 1024, 4096, 512]$ και συναρτήσεις ενεργοποίησης ReLU στα κρυφά επίπεδα και tanh στο επίπεδο εξόδου. Το βέλτιστο GNN διαθέτει 5 κρυφά επίπεδα με αριθμό νευρώνων ανά επίπεδο $[256, 512, 512, 256, 256]$ και συναρτήσεις ενεργοποίησης GeLU στα κρυφά επίπεδα και tanh στο επίπεδο εξόδου. Παρόλο που οι μεταβλητές σχεδιασμού στη διαδικασία βελτιστοποίησης ορίστηκαν στο ίδιο εύρος με εκείνο του FCNN, οι αυξημένες απαιτήσεις μνήμης περιόρισαν τον αριθμό επιτυχών αξιολογήσεων.

Ως προς την πρόβλεψη εκτός PUMA, το FCNN επιτυγχάνει μέσο απόλυτο σφάλμα στις 10 αεροτομές αξιολόγησης $\mathbf{3.5 \times 10^{-3}}$, ενώ το GNN $\mathbf{1.5 \times 10^{-3}}$. Για μία αεροτομή, το πεδίο αναφοράς της μ_t, η πρόβλεψη του FCNN και η απόλυτη διαφορά τους παρουσιάζονται στο Σχήμα 7.1.



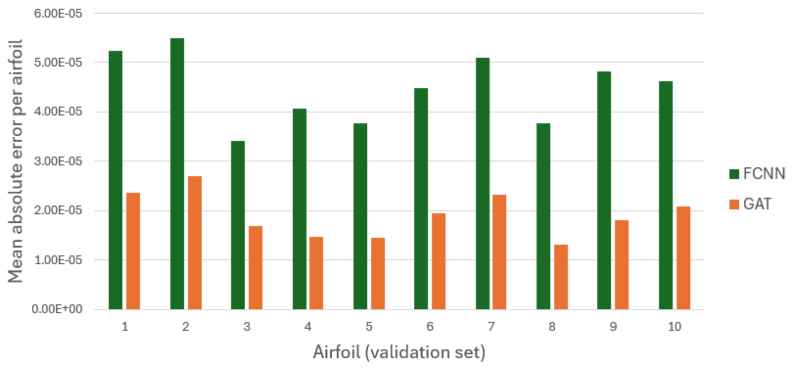
Σχήμα 7.1: *NACA4318: Πεδίο αναφοράς τυρβώδους συνεκτικότητας μ_t, πρόβλεψη FCNN (FCNN) και απόλυτη διαφορά.*



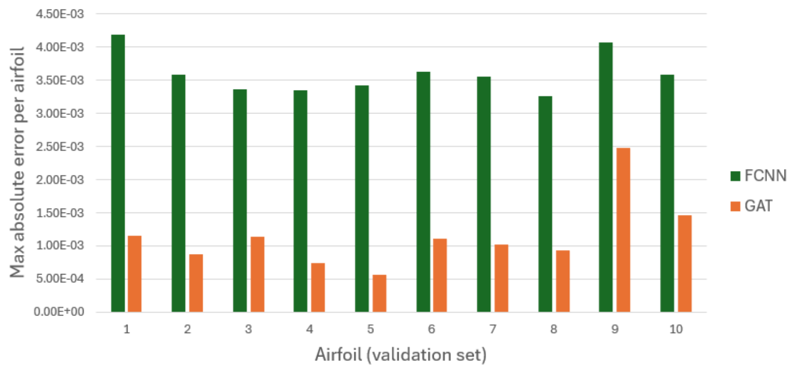
Σχήμα 7.2: *NACA4318: Πεδίο αναφοράς τυρβώδους συνεκτικότητας μ_t, πρόβλεψη GNN (GAT) και απόλυτη διαφορά.*

Από το Σχήμα 7.1 παρατηρείται ότι το FCNN αποτυπώνει σε ικανοποιητικό βαθμό τη μορφολογία του ομόρρου. Ωστόσο, εμφανίζεται αυξημένο απόλυτο σφάλμα κοντά στο χείλος εκφυγής, γεγονός που μπορεί να αποδοθεί στην ανεξάρτητη επεξεργασία των κόμβων. Στο Σχήμα 7.2 παρουσιάζονται τα αντίστοιχα αποτελέσματα για το GNN, το οποίο εμφανίζει μικρότερα σφάλματα και δεν παρουσιάζει την ίδια τοπική αστοχία στην περιοχή του χείλους εκφυγής, λόγω της ενσωμάτωσης πληροφορίας από γειτονικούς κόμβους μέσω της διάδοσης μηνυμάτων.

Για ποσοτική ανάλυση στις 10 αεροτομές αξιολόγησης υπολογίζονται το μέσο και το μέγιστο απόλυτο σφάλμα και παρουσιάζονται στα Σχήματα 7.3 και 7.4. Όπως φαίνεται στα Σχήματα, το GNN παρουσιάζει βελτιωμένη απόδοση έναντι του FCNN, καθώς οδηγεί σε χαμηλότερο μέσο και μέγιστο απόλυτο σφάλμα στις αεροτομές αξιολόγησης.



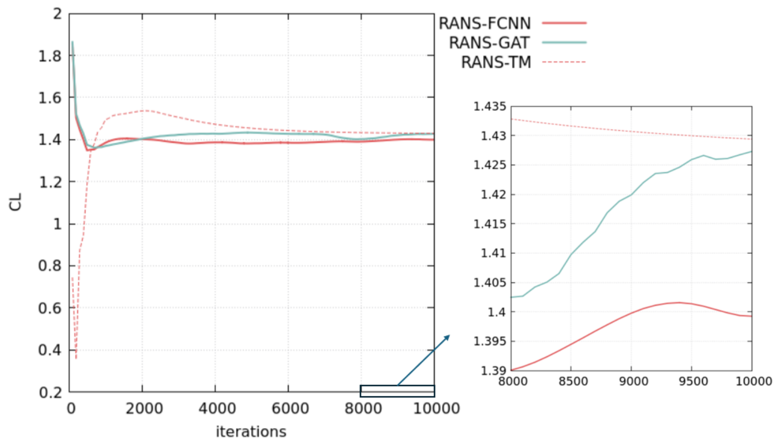
Σχήμα 7.3: *NACA4318: Μέσο σφάλμα πρόβλεψης μ_t για καθεμία από τις 10 αεροτομές αξιολόγησης για τα μοντέλα FCNN (FCNN) και GNN (GAT).*



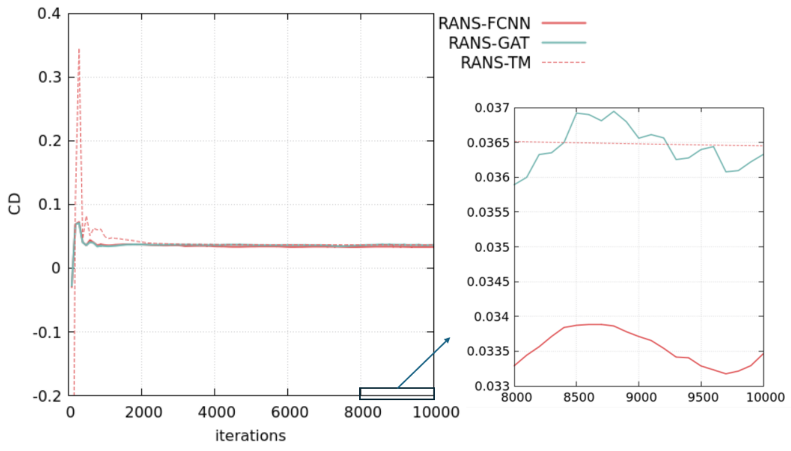
Σχήμα 7.4: *NACA4318: Μέγιστο σφάλμα πρόβλεψης μ_t για καθεμία από τις 10 αεροτομές αξιολόγησης για τα μοντέλα FCNN (FCNN) και GNN (GAT).*

Στη συνέχεια, τα δύο μοντέλα ενσωματώνονται στον κώδικα ΥΡΔ PUMA, αντικαθιστώντας το μοντέλο τύρβης. Για την αξιολόγηση της συμπεριφοράς εντός του επιλυτή παρουσιάζεται η σύγκλιση των αεροδυναμικών συντελεστών άνωσης και οπισθέλκουσας στα Σχήματα 7.5 και 7.6 (RANS–FCNN για το FCNN και RANS–GAT για το GNN). Ως αναφορά περιλαμβάνεται και η καμπύλη του κλασικού μοντέλου (RANS–TM, όπου TM δηλώνει το *turbulence model*).

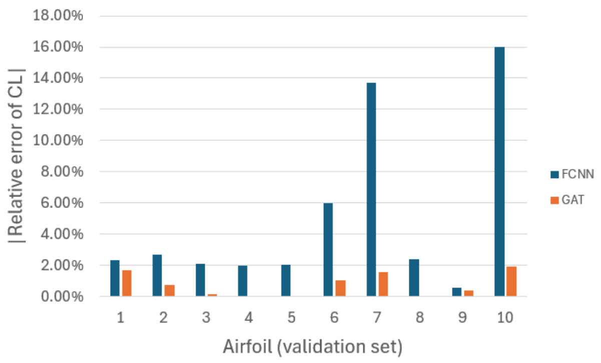
Από τα Σχήματα 7.5 και 7.6 προκύπτει ότι ο επιλυτής μπορεί να συγκλίνει με αντικατεστημένο μοντέλο τύρβης ένα νευρωνικό υποκατάστατο, ενώ το GNN τείνει να συγκλίνει σε τιμές πιο κοντινές στη λύση αναφοράς σε σχέση με το FCNN. Για συνολική αποτίμηση στις 10 αεροτομές αξιολόγησης υπολογίζεται το σχετικό σφάλμα των C_L και C_D, το οποίο παρουσιάζεται στα Σχήματα 7.7 και 7.8. Σε όλες τις περιπτώσεις το GNN εμφανίζει χαμηλότερο σχετικό σφάλμα από το FCNN.



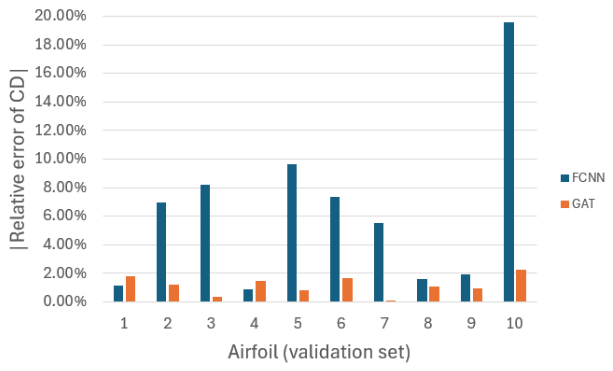
Σχήμα 7.5: *NACA4318: Σύγκλιση του συντελεστή άνωσης C_L για μία αεροτομή αξιολόγησης (RANS–FCNN, RANS–GAT, RANS–TM).*



Σχήμα 7.6: *NACA4318: Σύγκλιση του συντελεστή οπισθέλκουσας C_D για μία αεροτομή αξιολόγησης (RANS–FCNN, RANS–GAT, RANS–TM).*



Σχήμα 7.7: *NACA4318: Σχετικό σφάλμα του συντελεστή άνωσης C_L για FCNN (FCNN) και GNN (GAT).*



Σχήμα 7.8: *NACA4318: Σχετικό σφάλμα του συντελεστή οπισθέλκουσας C_D για FCNN (FCNN) και GNN (GAT).*

Συνοψίζοντας, τα αποτελέσματα για τη συγκεκριμένη εφαρμογή δείχνουν ότι τα GNN, τα οποία ενσωματώνουν ρητά τη γειτονική πληροφορία μέσω διάδοσης μηνυμάτων, προσφέρουν πλεονέκτημα έναντι των FCNN που επεξεργάζονται κάθε κόμβο ανεξάρτητα, τόσο ως προς την ακρίβεια πρόβλεψης της μ_t εκτός επιλυτή όσο και ως προς την τελική αεροδυναμική απόκριση όταν χρησιμοποιούνται εντός του PUMA.

Πτερύγωση στροβίλου LS89

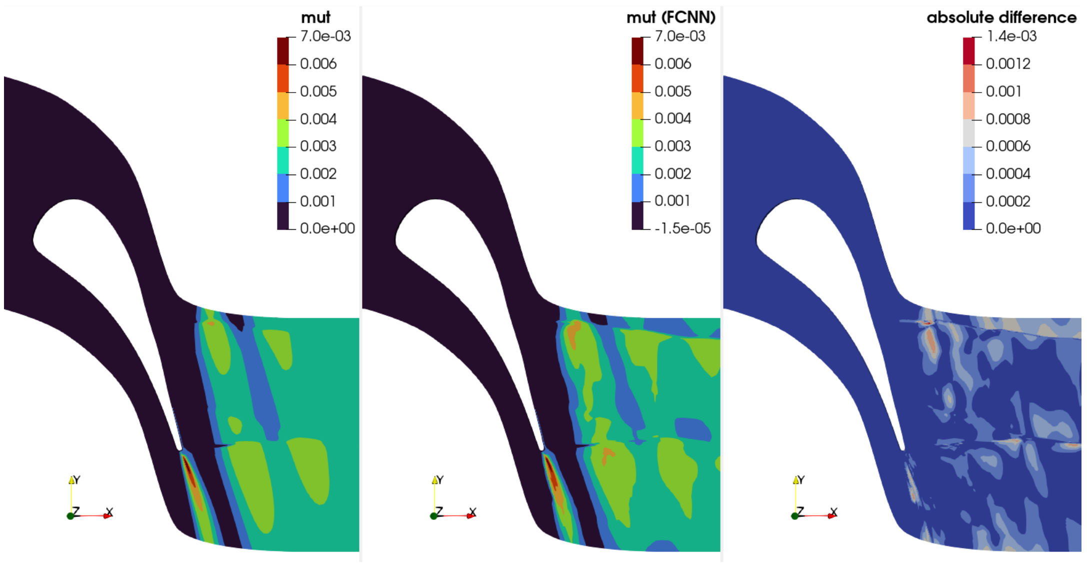
Η διηχητική πτερύγωση στροβίλου LS89 χρησιμοποιείται ως γεωμετρία αναφοράς και παραμετροποιείται με ογκομετρικές NURBS, όπως και στην εφαρμογή της μεμονωμένης αεροτομής. Με τη μέθοδο LHS δημιουργούνται 150 διαφορετικές γεωμετρίες, για τις οποίες παράγονται δομημένα πλέγματα τύπου C με ίδιο αριθμό κόμβων.

Για τη δημιουργία των δεδομένων αναφοράς επιλύονται οι εξισώσεις ροής με το μοντέλο τύρβης $k-\omega$ SST και το μοντέλο μετάβασης $\gamma-\tilde{Re}_{\theta t}$. Οι οριακές συνθήκες/συνθήκες ροής δίνονται ως: $p_t^{in} = 3.269$ bar, $T_t^{in} = 418.90$ K, $p^{out} = 1.550$ bar, $T^w = 297.55$ K, $\alpha^{in} = 0°$, $Tu^{in} = 0.8\%$ και $(\mu_t/\mu)^{in} = 11$.

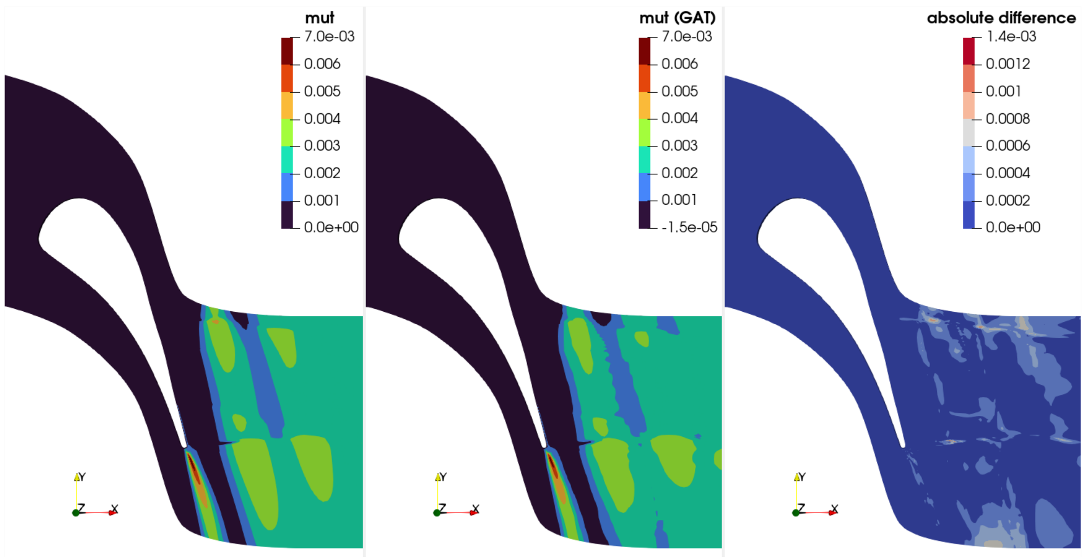
Ως είσοδοι των μοντέλων επιλέγονται ροϊκές ποσότητες σύμφωνες με τη [20]. Συγκεκριμένα, χρησιμοποιούνται οι συντεταγμένες x, y, οι συνιστώσες της ταχύτητας u, v, η πυκνότητα ρ, η πίεση p, η στροβιλότητα Ω, ο ρυθμός παραμόρφωσης S και η απόσταση από το τοίχωμα d. Ως ζητούμενη έξοδος ορίζεται η τυρβώδης συνεκτικότητα μ_t. Τα δεδομένα εισόδου–εξόδου κανονικοποιούνται ως προς τις ελάχιστες και μέγιστες τιμές τους.

Για την εκπαίδευση χρησιμοποιούνται 120 γεωμετρίες, ενώ 30 γεωμετρίες διατηρούνται για αξιολόγηση. Η αρχιτεκτονική του FCNN έχει βελτιστοποιηθεί στη [20], ενώ στο πλαίσιο της παρούσας εργασίας πραγματοποιείται βελτιστοποίηση της αρχιτεκτονικής του GNN. Το βέλτιστο FCNN διαθέτει 5 κρυφά επίπεδα με αριθμό νευρώνων $[512, 512, 4096, 4096, 256]$, ενώ το βέλτιστο GNN 5 κρυφά επίπεδα με $[1024, 128, 64, 64, 32]$. Και στα δύο μοντέλα χρησιμοποιείται ReLU στα κρυφά επίπεδα και tanh στην έξοδο.

Ως προς την πρόβλεψη εκτός επιλυτή, το FCNN επιτυγχάνει μέσο απόλυτο σφάλμα $\mathbf{9.9 \times 10^{-3}}$, ενώ το GNN $\mathbf{8.5 \times 10^{-3}}$. Για μία πτερύγωση, το πεδίο αναφοράς της μ_t, η πρόβλεψη των δύο μοντέλων και η απόλυτη διαφορά παρουσιάζονται στα Σχήματα 7.9 και 7.10.



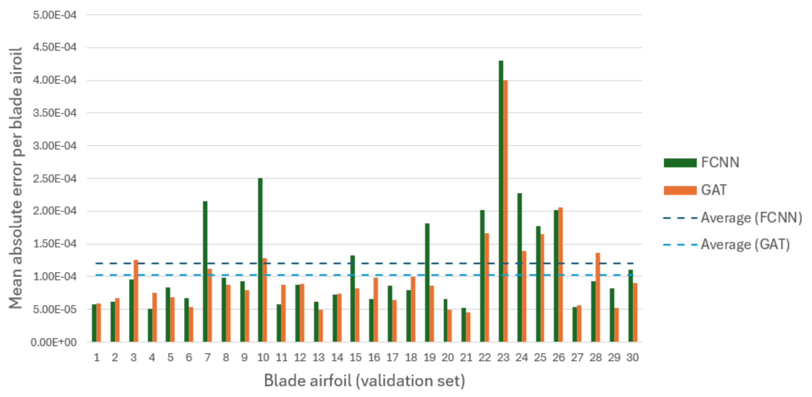
Σχήμα 7.9: *LS89: Πεδίο αναφοράς τυρβώδους συνεκτικότητας μ_t, πρόβλεψη FCNN (FCNN) και απόλυτη διαφορά.*



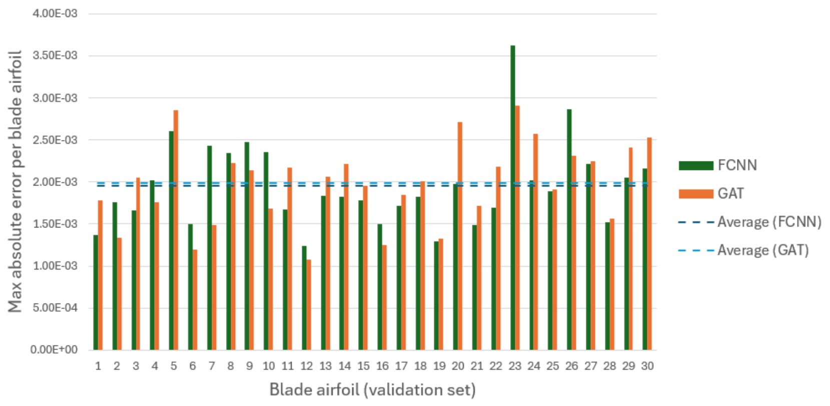
Σχήμα 7.10: *LS89: Πεδίο αναφοράς τυρβώδους συνεκτικότητας μ_t, πρόβλεψη GNN (GAT) και απόλυτη διαφορά.*

Από τα Σχήματα 7.9–7.10 προκύπτει ότι και τα δύο μοντέλα αναπαράγουν ικανοποιητικά τα κύρια χαρακτηριστικά του πεδίου μ_t στην περιοχή κατάντη του πτερυγίου. Ωστόσο, η πρόβλεψη με GNN εμφανίζει συνολικά περιορισμένες περιοχές αυξημένου σφάλματος, στοιχείο που συνάδει με την αξιοποίηση γειτονικής πληροφορίας.

Για ποσοτική εκτίμηση της απόδοσης στις 30 γεωμετρίες αξιολόγησης υπολογίζονται ο μέσος και ο μέγιστος απόλυτος δείκτης σφάλματος, οι οποίοι παρουσιάζονται στα Σχήματα 7.11 και 7.12. Παρατηρείται ότι το GNN υπερέχει ως προς το μέσο απόλυτο σφάλμα, ενώ ως προς το μέγιστο σφάλμα οι διαφορές μεταξύ των δύο προσεγγίσεων είναι περιορισμένες.

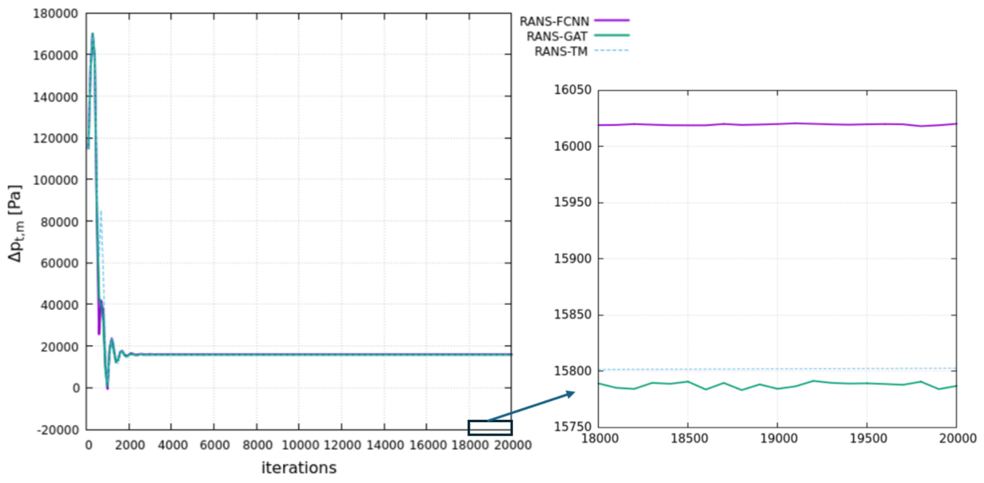


Σχήμα 7.11: *LS89: Μέσο σφάλμα πρόβλεψης μ_t για καθεμία από τις 30 γεωμετρίες αξιολόγησης για FCNN (FCNN) και GNN (GAT). Ο συνολικός μέσος όρος υπολογίζεται και απεικονίζεται για κάθε μοντέλο.*



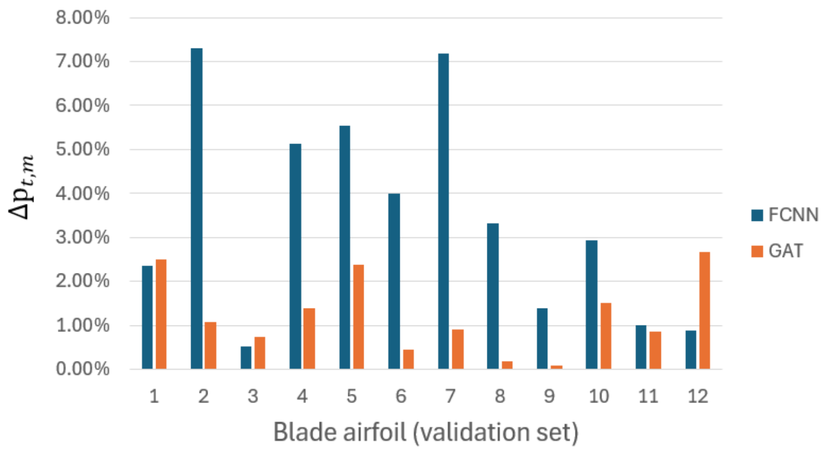
Σχήμα 7.12: *LS89: Μέγιστο σφάλμα πρόβλεψης μ_t για καθεμία από τις 30 γεωμετρίες αξιολόγησης για FCNN (FCNN) και GNN (GAT). Ο συνολικός μέσος όρος υπολογίζεται και απεικονίζεται για κάθε μοντέλο.*

Στη συνέχεια, τα μοντέλα ενσωματώνονται στον κώδικα ΥΡΔ PUMA ως υποκατάστατα του μοντέλου τύρβης και μετάβασης (RANS–FCNN και RANS–GAT στα αντίστοιχα διαγράμματα). Για την αξιολόγηση της συμπεριφοράς του επιλυτή παρουσιάζεται η σύγκλιση των *μαζικά σταθμισμένων απωλειών ολικής πίεσης* $\Delta p_{t,m}$ στο Σχήμα 7.13. Ως αναφορά περιλαμβάνεται η καμπύλη της κλασικής προσομοίωσης (RANS–TM, όπου TM δηλώνει turbulence model).



Σχήμα 7.13: *LS89: Σύγκλιση των μαζικά σταθμισμένων απωλειών ολικής πίεσης $\Delta p_{t,m}$ για μία γεωμετρία αξιολόγησης (RANS–FCNN, RANS–GAT, RANS–TM).*

Από το Σχήμα 7.13 διαπιστώνεται ότι ο επιλυτής συγκλίνει και στις δύο περιπτώσεις, όταν το μοντέλο τύρβης/μετάβασης αντικαθίσταται από νευρωνικό υποκατάστατο. Για το συγκεκριμένο παράδειγμα, το GNN οδηγεί σε τελική τιμή πιο κοντινή στην αναφορά σε σύγκριση με το FCNN. Για συνολική αποτίμηση, υπολογίζεται το σχετικό σφάλμα της $\Delta p_{t,m}$ για 12 γεωμετρίες αξιολόγησης και παρουσιάζεται στο Σχήμα 7.14, όπου παρατηρείται βελτιωμένη απόδοση του GNN.



Σχήμα 7.14: *LS89: Σχετικό σφάλμα των μαζικά σταθμισμένων απωλειών ολικής πίεσης $\Delta p_{t,m}$ για FCNN (FCNN) και GNN (GAT).*

Συνοψίζοντας, ακόμη και στη συνθετότερη διηχητική περίπτωση της LS89, τα GNN εμφανίζουν πλεονέκτημα έναντι των FCNN, καθώς η ενσωμάτωση της τοπικής γειτονικής πληροφορίας οδηγεί σε πιο αξιόπιστη πρόβλεψη του πεδίου μ_t και, κατ΄ επέκταση, σε καλύτερη συμφωνία των υπολογιζόμενων μαζικά σταθμισμένων απωλειών ολικής πίεσης $\Delta p_{t,m}$ όταν τα μοντέλα χρησιμοποιούνται εντός του επιλυτή PUMA.

Συμπεράσματα

Στην παρούσα εργασία διερευνήθηκε η δυνατότητα αντικατάστασης μοντέλων τύρβης και μετάβασης από νευρωνικά υποκατάστατα, συγκρίνοντας Πλήρως Συνδεδεμένα Νευρωνικά Δίκτυα (FCNN) και Νευρωνικά Δίκτυα Γράφων (GNN) για την πρόβλεψη της τυρβώδους συνεκτικότητας μ_t σε πλέγματα ΥΡΔ. Η αξιολόγηση πραγματοποιήθηκε σε δύο διαφορετικές εφαρμογές: μία μεμονωμένη αεροτομή (NACA4318) και μία διηχητική πτερύγωση στροβίλου (LS89).

Για την περίπτωση της NACA4318, το GNN υπερείχε σαφώς του FCNN τόσο ως προς την ακρίβεια πρόβλεψης του πεδίου μ_t εκτός επιλυτή όσο και ως προς την αναπαραγωγή των αεροδυναμικών συντελεστών όταν ενσωματώθηκε στον PUMA. Η βελτιωμένη απόδοση αποδίδεται στη ρητή αξιοποίηση της γειτονικής πληροφορίας (συνδεσιμότητα πλέγματος), η οποία μειώνει τοπικές αστοχίες που προκύπτουν όταν οι κόμβοι αντιμετωπίζονται ανεξάρτητα.

Στη συνθετότερη διηχητική περίπτωση της LS89, η διαφορά στην πρόβλεψη της μ_t εκτός επιλυτή ήταν μικρότερη, ωστόσο το GNN διατήρησε πλεονέκτημα ως προς τον μέσο δείκτη σφάλματος και, κυρίως, ως προς την απόδοση εντός επιλυτή: οι μαζικά σταθμισμένες απώλειες ολικής πίεσης $\Delta p_{t,m}$ προσεγγίστηκαν συστηματικά πιο κοντά στη λύση αναφοράς σε σχέση με το FCNN.

Συνολικά, τα GNN αναδείχθηκαν ως πιο κατάλληλα υποκατάστατα για εφαρμογές ΥΡΔ σε πλέγματα, λόγω της φυσικής τους συμβατότητας με μη δομημένες/δομημένες διακριτοποιήσεις και της δυνατότητας αξιοποίησης γειτονικών συσχετίσεων.