

Fluid Intelligence:
Physics-Informed Neural Networks as CFD Surrogates for Indoor Airflow and Thermal Comfort

Hasibul Hossain

A thesis
submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Architecture

University of Washington

2026

Committee:
Tomás Méndez Echenagucia
Narjes Abbasabadi
Mohammad Tabatabaei Manesh

Program Authorized to Offer Degree:
Architecture

© Copyright 2026

Hasibul Hossain

University of Washington

Abstract

Fluid Intelligence:

Physics-Informed Neural Networks as CFD Surrogates for Indoor Airflow and Thermal Comfort

Hasibul Hossain

Chair of the Supervisory Committee:

Tomás Méndez Echenagucia

Department of Architecture

Predicting indoor airflow is central to thermal comfort, but the computational fluid dynamics (CFD) that does it well is too costly to run inside an architectural design workflow. This study asks whether a deep-learning surrogate can stand in, and which kind. Three architectures are compared on one shared problem: a data-driven artificial neural network (ANN), a physics-informed neural network (PINN), and a graph neural network (GNN), all trained as surrogates for $k\text{-}\epsilon$ RANS CFD of a ventilated room. The comparison spans four room geometries, three training-set sizes, and six evaluation splits that separate interpolation from extrapolation along the velocity and thermal axes, with an ISO 7730 layer carrying the predicted fields through to thermal-comfort measures. Inside the sampled range the three models perform alike, with the GNN and PINN slightly ahead. Outside it they diverge: the PINN holds its temperature accuracy where the ANN and GNN fail, wins nearly every extrapolation case, and reaches better-than-full-data accuracy on temperature from a fraction of the training cases. The comfort predictions follow the field accuracy, leaving the PINN the most reliable. Physics-informed training earns its place when extrapolation and limited data matter, the conditions early-stage design most often meets.

Keywords: PINN, GNN, ANN, CFD surrogate, RANS CFD, ISO 7730, indoor airflow, thermal comfort, deep learning, data efficiency, generalization, Fourier features, parametric design, architectural design

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement	4
1.3	Research Questions	6
1.4	Contributions	7
1.5	Paper Organization	7
2	Background	8
2.1	Computational Fluid Dynamics	8
2.1.1	Governing Equations: Navier-Stokes	8
2.1.2	Turbulence Modeling: The $k-\epsilon$ Model	10
2.1.3	Numerical Methods and Computational Cost	12
2.2	Indoor Environment and Thermal Comfort	13
2.2.1	Indoor Environmental Quality	13
2.2.2	Thermal Comfort: PMV and PPD	14
2.2.3	Draught Rate	16
2.2.4	Mean Radiant Temperature	17
2.3	Deep Learning Foundations	18
2.3.1	Feed-Forward Neural Networks	18
2.3.2	Generalization and Regularization	19
2.4	Surrogate Modeling Approaches	20
2.4.1	Pure Data-Driven Surrogates	21
2.4.2	Physics-Informed Neural Networks	22
2.4.3	Graph Neural Networks for Spatial Fields	24
3	Related Work	25
3.1	Machine Learning for CFD: A Historical Overview	25
3.2	Surrogate Models for Indoor Airflow	26
3.3	Physics-Informed Neural Networks in Fluid Dynamics	28
3.4	Graph Neural Networks for Field Prediction	29

3.5	Integration with Architectural Design Workflows	30
3.6	Research Gap and Positioning	31
3.7	Literature Matrix	33
4	Methodology	34
4.1	Pipeline Overview	34
4.2	CFD Dataset Generation	35
4.2.1	Simulation Environment: COMSOL Multiphysics	35
4.2.2	Governing Physics: Coupled Turbulent Flow and Heat Transfer	37
4.2.3	Geometry and Boundary Conditions	40
4.2.4	Parameter Space	41
4.2.5	Mesh Strategy	43
4.3	Automated Sweep Pipeline	44
4.3.1	Latin Hypercube Sampling	44
4.3.2	Interpolation Gap Design	46
4.3.3	Extrapolation Range Design	48
4.3.4	Staged Data Generation for the Data-Efficiency Study	48
4.3.5	Six-Split Evaluation Design	49
4.4	Wall Configurations: Robustness Study	51
4.5	Surrogate Models	53
4.5.1	ANN: The Data-Driven Baseline	54
4.5.2	PINN: The Physics-Informed Approach	58
4.5.3	GNN: The Graph-Structured Approach	65
4.6	Downstream Comfort Prediction	70
4.6.1	A Deterministic Comfort Layer	70
4.6.2	Field Inputs and Personal Parameters	70
4.6.3	Mean Radiant Temperature	71
4.6.4	PMV, PPD, and Draught Rate	72
4.6.5	Categories, Verification, and Assumptions	73
4.7	Evaluation Framework	74
4.7.1	Field Accuracy	74
4.7.2	Interpolation versus Extrapolation	75
4.7.3	Data Efficiency	75

4.7.4	Downstream Comfort Accuracy	76
5	Results	77
5.1	Evaluation Cases	78
5.2	Reconstructed Fields	78
5.3	Field-Accuracy Metrics	83
5.4	Predicted Comfort	85
5.5	Model Ranking and Data Efficiency	90
5.6	The Pattern Across Geometries	91
6	Discussion	94
6.1	Architecture Performance Patterns	94
6.2	Generalization Behavior	95
6.3	Data Efficiency Implications	96
6.4	Implications for Architectural Design Workflows	97
6.5	Design Principles for Indoor Airflow Surrogates	98
7	Limitations and Future Work	99
7.1	Limitations	99
7.2	Future Directions	102
7.2.1	Near-Term Methodological Extensions	102
7.2.2	Longer-Term Applications	103
8	Conclusions	104
	References	106
A	Hyperparameters and Training Details	114
A.1	Architectural Hyperparameters	114
A.2	Optimisation and Schedule	115
A.3	PINN-Specific Settings	115
A.4	GNN-Specific Settings	116
A.5	Hardware	116
A.6	Approximate Training Times	116
B	Per-Configuration Results	117

C	Parameter Ranges and LHS Configuration	129
C.1	Seven-Dimensional Training Range	129
C.2	Interpolation Gaps	130
C.3	Extrapolation Bands	130
C.4	Latin Hypercube Sampling Configuration	130
C.5	Case Counts per Split per Configuration	131
C.6	Deterministically Nested Data-Efficiency Subsets	131
C.7	Identical Parameter Ranges Across Configurations	131
D	Verification of Thermal Comfort Implementation	132
D.1	Basis of Verification	132
D.2	PMV / PPD Across the Operating Range	133
D.3	Draft Rate against the ISO 7730:2005 Closed Form	134
D.4	MRT View-Factor Radiative Closure	135
D.5	Summary of Verification	135

List of Figures

2.1	Methods for predicting building-related physical fields, adapted from Chen et al. (2025). This paper sits in the surrogate-model branch under machine learning, with reference data from a RANS $k-\varepsilon$ solver.	20
4.1	The four-stage pipeline of this paper. A COMSOL RANS $k-\varepsilon$ sweep generates the reference fields; three surrogate architectures are trained on them; each is evaluated for field accuracy, generalisation, and downstream comfort; and a trained surrogate is deployed inside a parametric design tool.	34
4.2	A representative COMSOL solution for one case of the ventilated room: the steady velocity and temperature fields the sweep produces. These are the reference fields each surrogate is trained to reproduce.	36
4.3	The room geometry as built in COMSOL: the rectangular domain, the concrete envelope, the interior partition, and the inlet and outlet openings.	40
4.4	The two-dimensional room: interior dimensions, the concrete envelope, the interior partition wall, and the inlet and outlet openings with the geometric parameters that vary from case to case. The partition is shown at the quarter-length position.	42
4.5	The physics-controlled COMSOL mesh at the “Finer” element size, with boundary-layer refinement along the walls.	43
4.6	Latin Hypercube Sampling set against plain random sampling on a 10×10 grid. Independent random draws (left) leave some rows and columns empty while doubling up in others; a Latin Hypercube design (right) places exactly one point in every row and every column, so each parameter is sampled evenly across its range. For the same number of cases this removes the wasted, clustered draws, which is why the parameter sweep uses it.	46

4.7	The interpolation test design in the inlet-speed-supply-temperature plane; the box marks the training range. Training cases (grey) fill the plane apart from two held-out interior bands – a vertical velocity gap at $U_{\text{in}} \in [1.00, 1.50]$ m/s and a horizontal thermal gap at $T_{\text{in}} \in [296, 302]$ K. Interpolation test cases (purple) are drawn from inside those bands, surrounded on both sides by training data, so the test measures how well a model fills a gap at parameter values it never trained on. Only one axis is held out at a time: cases in the vertical band vary inlet speed inside the gap while temperature stays in the trained range, and cases in the horizontal band do the reverse.	47
4.8	The six splits in the plane of inlet speed U_{in} and supply temperature T_{in} . Training and validation cases fill the range apart from the two interior gaps removed by rejection sampling; the interpolation splits sample inside those gaps; the extrapolation splits sample strictly above the training maximum on each axis. The remaining five parameters are sampled over their full ranges in every split.	50
4.9	The four room geometries. From left: the open room with no partition, and the partition placed at the quarter, half, and three-quarter points of the room length. The partition dimensions and all seven sampled parameters are identical across the obstructed cases; only the wall position changes.	52
4.10	The ANN baseline: a plain feed-forward network mapping the ten input features of a single query point through six hidden layers of width 256 to the six output fields $(u, v, p, T, k, \varepsilon)$ at that point. It has no Fourier encoding, no gating, no skip connection, and no graph structure – the deliberately minimal data-driven model of the comparison.	55
4.11	The PINN architecture: the spatial coordinates are lifted by random Fourier features and concatenated with the case parameters, passed through a stack of gated hidden blocks with adaptive activations and an input-skip residual, and projected to the six output fields. The governing-equation residuals are formed from the outputs by automatic differentiation and added to the loss.	59
4.12	The GNN architecture: each CFD case becomes a grid graph whose edges follow four-connectivity but are pruned wherever they would cross the partition wall. An encoder lifts the per-node features, four GraphSAGE blocks pass messages along the surviving edges, and a decoder reads out the six fields at every fluid node.	65

5.1	Velocity in the interpolation-velocity case. Top to bottom: streamwise velocity u (m/s) and vertical velocity v (m/s) for the open room, then the same two for the half-partition room. In each panel, the top row is the COMSOL ground truth with the ANN, GNN and PINN predictions, and the bottom row is each model's absolute error.	79
5.2	Temperature in the two interpolation cases. Top to bottom: temperature T ($^{\circ}\text{C}$) in the interpolation-velocity and interpolation-thermal cases for the open room, then the same two for the half-partition room. Panel layout as in Figure 5.1.	80
5.3	Velocity in the extrapolation-velocity case. Top to bottom: streamwise velocity u (m/s) and vertical velocity v (m/s) for the open room, then the same two for the half-partition room. Panel layout as in Figure 5.1.	81
5.4	Temperature in the two extrapolation cases. Top to bottom: temperature T ($^{\circ}\text{C}$) in the extrapolation-velocity and extrapolation-thermal cases for the open room, then the same two for the half-partition room. Panel layout as in Figure 5.1.	82
5.5	Field-accuracy tables. Top to bottom: mean squared error and mean absolute error per model, broken out by test case and field, for the open room, then the same two for the half-partition room. The lowest error in each split-and-field cell is outlined, and the outline marks the winning model.	84
5.6	Predicted comfort fields in the open room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Each row shows the COMSOL ground truth and the ANN, GNN and PINN predictions, with each prediction's mean absolute error against the ground truth annotated.	86
5.7	Predicted comfort fields in the half-partition room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Panel layout as in Figure 5.6.	87
5.8	Comfort-fraction agreement in the open room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Each panel plots each model's predicted comfort fraction against the ground-truth value.	88
5.9	Comfort-fraction agreement in the half-partition room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Panel layout as in Figure 5.8.	89
5.10	Coefficient of determination per model, split, and field, one heatmap per model: (a) the open room and (b) the half-partition room. Lighter cells are higher and white is a perfect fit. The interpolation rows are light for all three models; in the extrapolation-thermal row the temperature cell turns negative for the ANN and GNN – a worse fit than predicting the mean – while the PINN's stays high.	92

5.11 Per-field winners and reduced-data comparison. Top row: the single best model for each field and split, (a) the open room and (b) the half-partition room, with the outline colour marking the winner. Bottom row: the PINN's error trained on reduced data as a percentage of the ANN and GNN baselines trained on the full set, (c) the open room and (d) the half-partition room; below 100 percent (blue) the PINN still wins on less data, above it (red) the PINN loses.	93
B.1 Velocity in the interpolation-velocity case: streamwise velocity u (m/s) and vertical velocity v (m/s) for the quarter-partition room, then the same two for the three-quarter-partition room. Panel layout as in Figure 5.1.	118
B.2 Temperature T (°C) in the interpolation-velocity and interpolation-thermal cases for the quarter-partition room, then the three-quarter-partition room. Panel layout as in Figure 5.2.	119
B.3 Velocity in the extrapolation-velocity case: streamwise velocity u (m/s) and vertical velocity v (m/s) for the quarter-partition room, then the three-quarter-partition room. Panel layout as in Figure 5.3.	120
B.4 Temperature T (°C) in the extrapolation-velocity and extrapolation-thermal cases for the quarter-partition room, then the three-quarter-partition room. Panel layout as in Figure 5.4.	121
B.5 Field-accuracy tables: mean squared error and mean absolute error per model, by test case and field, for the quarter-partition room, then the three-quarter-partition room. The lowest error in each split-and-field cell is outlined. Layout as in Figure 5.5.	122
B.6 Predicted comfort fields in the quarter-partition room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Layout as in Figure 5.6.	123
B.7 Predicted comfort fields in the three-quarter-partition room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Layout as in Figure 5.6.	124
B.8 Comfort-fraction agreement in the quarter-partition room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Layout as in Figure 5.8.	125
B.9 Comfort-fraction agreement in the three-quarter-partition room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Layout as in Figure 5.8.	126
B.10 Coefficient of determination per model, split, and field, one heatmap per model: (a) the quarter-partition room and (b) the three-quarter-partition room. Layout as in Figure 5.10.	127

B.11 Per-field winners (top) and the PINN's reduced-data comparison (bottom): (a, c) the quarter-partition room and (b, d) the three-quarter-partition room. Layout as in Figure 5.11.128

List of Tables

2.1	Thermal-environment categories of ISO 7730:2005, with the matching EN 16798-1:2019 labels.	16
3.1	The closest prior studies set against the axes this paper addresses. <i>Architectures compared</i> counts distinct neural-network families evaluated on one problem; <i>Generalization tested</i> names the held-out regime each study used. Chung reports comfort with an adaptive model rather than a field-based ISO 7730 calculation, and both Sakiyama and Chung couple conventional solvers to the design tool rather than a neural surrogate.	33
4.1	The seven sampled case parameters and their training ranges. The two opening heights are not sampled directly but derived from auxiliary unit variables through the geometry-safe map of Equation 4.8.	42
4.2	The six evaluation splits sampled for each room geometry. The ranges are the sampling bounds for the inlet speed and the supply temperature; the excluded bands are the interior gaps removed by rejection sampling. The remaining five parameters are sampled over their full ranges in every split.	50
4.3	The three surrogate architectures at a glance. The physics-informed and graph models are described in the two subsections that follow; they are previewed here only to show where the data-driven baseline sits.	55
A.1	Architectural hyperparameters for the three surrogate models. Cross-references: ANN Section 4.5.1, PINN Section 4.5.2, GNN Section 4.5.3.	114
A.2	Optimisation, scheduling, and training-loop hyperparameters.	115
A.3	Wall-clock training time in minutes, by architecture, dataset size, and wall configuration.	116
C.1	Training-range bounds for the seven LHS-sampled parameters.	129
C.2	Target case counts per split per wall configuration.	131
D.1	PMV / PPD / ISO 7730 Annex A category across six operating-range environments at $M = 1.2$ met, $I_{cl} = 0.5$ clo, $RH = 50\%$.	133
D.2	Draft Rate against the ISO 7730:2005 Section 6.2, Eq. (6) closed-form evaluation at five benchmark points across the operating range.	134

D.3	MRT view-factor radiative closure verification at three uniform wall temperatures.	135
D.4	Summary of verification: each quantity tested, the reference against which it was tested, the implementation under test, and the maximum absolute deviation observed.	135

List of Symbols

Operators and Notation

∇	Gradient (del) operator
$\nabla \cdot$	Divergence operator
∇^2	Laplacian operator
$\partial/\partial t$	Partial derivative
$\ \cdot\ _2$	Euclidean (L_2) norm
$(\cdot)^T$	Transpose
$\odot$	Element-wise (Hadamard) product
$\oplus$	Vector concatenation
$\circ$	Function composition
$(\hat{\cdot})$	Neural-network prediction / estimated quantity
$\overline{(\cdot)}$	Reynolds (time) average
$(\cdot)'$	Fluctuating component
$\mathbb{R}$	Set of real numbers

Roman Symbols

$\mathbf{B}$	Random Fourier-feature frequency matrix
$\mathbf{b}_\ell$	Bias vector of layer ℓ
c_p	Specific heat capacity of air at constant pressure ($\text{J kg}^{-1} \text{K}^{-1}$)
$C_\mu, C_{\varepsilon 1}, C_{\varepsilon 2}$	Empirical constants of the $k\text{-}\varepsilon$ turbulence model
$d_{\text{in}}, d_{\text{out}}$	Vertical positions of the inlet and outlet openings
DR	Draught Rate, ISO 7730 (%)
E	Edge set of the graph (GNN)
$\mathbf{f}$	Body force per unit mass
f_{cl}	Clothing area factor
$F_{p\text{-}i}$	View factor between an evaluation point and surface i
$f(\mathbf{x}; \theta)$	Neural-network mapping with parameters θ
$\mathbf{g}$	Gravitational acceleration vector
$g(\mathbf{x})$	Prescribed boundary data (PINN)
h_c	Convective heat-transfer coefficient ($\text{W m}^{-2} \text{K}^{-1}$)
$h_{\text{in}}, h_{\text{out}}$	Heights of the inlet and outlet openings
$\mathbf{h}, \mathbf{h}_v^{(\ell)}$	Hidden feature vector (of node v at layer ℓ)
H	Hidden (feature) dimension
H_{room}	Room height
I_{cl}	Clothing thermal insulation ($\text{m}^2 \text{K W}^{-1}$)
$\mathbf{I}$	Identity tensor
k	Turbulent kinetic energy ($\text{m}^2 \text{s}^{-2}$)

$k_{\text{air}}, k_{\text{concrete}}$	Thermal conductivity of air / concrete ($\text{W m}^{-1} \text{K}^{-1}$)
L	Thermal load on the body (PMV); also number of hidden layers
$\mathcal{L}$	Loss function (subscripts denote data, PDE, BC, ... terms)
M	Metabolic rate (W m^{-2})
MAE, MSE	Mean absolute error / mean squared error
$\mathcal{N}(v)$	Neighbourhood (adjacent nodes) of node v
$N, N_{\text{data}}, N_{\text{col}}, N_{\text{BC}}$	Counts of samples / data, collocation and boundary points
p	Pressure (Pa)
p_a	Partial water-vapour pressure (Pa)
P_k	Production of turbulent kinetic energy
$\mathbf{P}$	Geometry / boundary-condition parameter vector
Pr_t	Turbulent Prandtl number
PMV	Predicted Mean Vote, ISO 7730
PPD	Predicted Percentage Dissatisfied, ISO 7730 (%)
Q	Volumetric heat source
$Q_{\text{conv}}, Q_{\text{rad}}, Q_{\text{evap}}, Q_{\text{res}}$	Convective, radiative, evaporative and respiratory heat losses
R^2	Coefficient of determination
$\mathcal{R}, \mathcal{B}$	PDE-residual and boundary-condition operators
R_{mom}, R_k	Momentum and k -transport PDE residuals
Re	Reynolds number
RH	Relative humidity (%)
$s_{\text{cont}}, s_{\text{mom}}, s_{\text{ene}}, s_k$	Residual normalisation scales
t	Time (s)
t_a	Air temperature ($^{\circ}\text{C}$)
t_{cl}	Clothing-surface temperature ($^{\circ}\text{C}$)
$\bar{t}_r$	Mean radiant temperature ($^{\circ}\text{C}$)
T	Temperature field (K)
T_i	Temperature of surface i (K)
T_u	Local turbulence intensity (%)
$T_{\text{in}}, T_{\text{wall}}$	Inlet-air and wall temperatures
$\mathbf{u}$	Velocity vector field
u, v	Horizontal and vertical velocity components (m s^{-1})
u_i, u_j	Velocity components (index notation)
U_{in}	Inlet velocity magnitude (m s^{-1})
v_a	Local air speed (m s^{-1})
V_{train}	Set of training nodes (GNN)
W	External mechanical work (W m^{-2})
$\mathbf{W}_{\ell}, \mathbf{W}_{\text{out}}, \mathbf{W}_{\text{skip}}$	Layer, output and skip-connection weight matrices
$\mathbf{x}$	Spatial position vector (x, y)
x, y	Horizontal and vertical Cartesian coordinates (m)
x_i, x_j	Spatial coordinates (index notation)
x_{wall}	Partition-wall position / extent
$\hat{\mathbf{y}}, \mathbf{y}$	Predicted and ground-truth field outputs

z

Concatenated input embedding (PINN)

Greek Symbols

α

View angle subtended by a surface; opening-blend factor

α_{eff}

Effective thermal diffusivity

γ

Random Fourier-feature encoding

ε

Turbulent dissipation rate ($\text{m}^2 \text{s}^{-3}$)

ε_{min}

Lower clip applied to ε to enforce positivity

θ

Trainable network parameters (weights and biases)

λ

Loss-term weighting coefficient

μ

Dynamic viscosity (Pa s); also batch mean in LayerNorm

μ_t

Turbulent (eddy) viscosity

ν

Kinematic viscosity ($\text{m}^2 \text{s}^{-1}$)

$\nu_{\text{mol}}, \nu_t, \nu_{\text{eff}}$

Molecular, turbulent and effective kinematic viscosity

ρ

Fluid density (kg m^{-3})

σ

Sigmoid activation function; standard deviation

$\sigma_k, \sigma_\varepsilon$

Turbulent Prandtl numbers for k and ε

σ_B

Fourier-feature frequency scale

τ

Viscous stress tensor

ϕ_ℓ

Activation / transformation of layer ℓ

Φ

Viscous dissipation function

List of Acronyms

ANN	Artificial Neural Network
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
BC	Boundary Condition
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
DR	Draught Rate, ISO 7730
FNO	Fourier Neural Operator
GELU	Gaussian Error Linear Unit (activation function)
GNN	Graph Neural Network
GNS	Graph Network Simulator
GraphSAGE	Graph SAmple and aggreGatE (inductive graph network)
HVAC	Heating, Ventilation and Air-Conditioning
ISO	International Organization for Standardization
LAAF	Layer-wise Adaptive Activation Function
LES	Large-Eddy Simulation
LHS	Latin Hypercube Sampling
MAE	Mean Absolute Error
MLP	Multilayer Perceptron
MRT	Mean Radiant Temperature
MSE	Mean Squared Error
PDE	Partial Differential Equation
PIGNN	Physics-Informed Graph Neural Network
PINN	Physics-Informed Neural Network
PMV	Predicted Mean Vote, ISO 7730
POD	Proper Orthogonal Decomposition
PPD	Predicted Percentage Dissatisfied, ISO 7730
RANS	Reynolds-Averaged Navier-Stokes
RH	Relative Humidity
SAGEConv	GraphSAGE convolution layer
SST	Shear-Stress Transport (k - ω turbulence model)

Acknowledgments

I would like to express my deepest gratitude to my advisor and committee chair, Professor Tomás Méndez Echenagucia, for his guidance, encouragement, and support throughout my master's studies. From the very beginning of my enrollment in the program, he helped me shape my academic direction, challenged me to think beyond my comfort zone, and pushed me to work at my highest capacity. His mentorship has not only strengthened this thesis, but also shaped the way I hope to grow as a researcher in the future.

I am also sincerely grateful to Professor Kate Simonen and Professor Tomás Méndez Echenagucia for giving me the opportunity to work as a Research Assistant on the POD-LCA project. This opportunity was deeply meaningful to me, as it helped me continue my second year of graduate study with greater stability and focus, while also allowing me to contribute to important research in the built environment.

I would like to thank Professor Narjes Abbasabadi, my thesis committee member, for her thoughtful guidance and valuable feedback throughout this process. I am especially grateful for the opportunity to work with her in the Sustainable Intelligence Lab, which allowed me to explore the role of artificial intelligence in architecture more deeply and helped shape the direction of my research.

I also extend my sincere thanks to Mohammad Tabatabaei Manesh, my thesis committee member, for his valuable feedback, insight, and support throughout the development of this thesis. His comments helped me refine my ideas and strengthen the work presented here.

I am deeply grateful to the University of Washington College of Built Environments for awarding me the Top Scholar Award and the Architectural Foundation scholarship. These opportunities made it possible for me, as an international student, to begin my higher education journey in the United States. I will always be thankful for the support that opened this path for me.

I would also like to thank Justin Smith, my supervisor at HNTB, for giving me the opportunity to explore the architectural profession through my internship. That experience broadened my understanding of practice, gave me meaningful exposure to the industry, and became an important part of my academic and professional growth during my graduate studies.

Finally, I would like to thank my family, friends, and everyone who supported me along the way. Their encouragement helped me continue during difficult moments. I also want to acknowledge my own perseverance throughout this journey—the long days, the uncertainty, and the effort it took to bring this thesis to completion.

1

Introduction

1.1 Motivation

Most of us live our lives indoors. Surveys put the figure at around 80% of our time spent inside buildings (H. Gao et al. 2024), which means the air we breathe and the conditions we work in are, for the most part, whatever conditions a building hands us. A building is meant to do more than keep the weather out. It should also keep a comfortable and healthy environment inside, and that second job depends almost entirely on ventilation—the steady exchange of stale indoor air for fresh outdoor air (Björling 2021). How that air actually moves through a room is not a minor detail. The way air is organized indoors shapes comfort, energy use, and the health of the people inside, and it has become one of the basic concerns of designing the built environment (Mao et al. 2025). The tool engineers reach for to study this movement is computational fluid dynamics, or CFD, which solves the governing equations of fluid flow, heat, and mass transfer to show how air behaves in a space (Mao et al. 2025).

When ventilation is poor, the consequences are real and measurable. Reviews of the evidence have found a clear link between how well a building is ventilated and how the people in it feel and perform. Better ventilation is tied to better perceived air quality and comfort, to fewer of the symptoms grouped under “sick building syndrome,” to fewer infections and fewer asthma and allergy complaints, and to less short-term sick leave—and it is even tied to how well people do their office work (Björling 2021). That last connection matters. Poor air does not just make a building unpleasant; it can quietly drag down the productivity of everyone who works in it.

The history of this problem is worth pausing on. As builders began putting up tighter, more sealed buildings, complaints started to pile up. In some stretches up to 30% of newly built offices drew an unusual number of them—headaches, fatigue, trouble concentrating, irritated skin and eyes—that nobody could fully explain. In 1984 the World Health Organization’s regional office for Europe gave these scattered symptoms a single name: Sick Building Syndrome. The tighter building envelopes going up at the time looked like a likely cause (Björling 2021).

The price of all this is not small. Indoor air pollution has been estimated to cost roughly two million disability-adjusted life years every year across 26 European countries, which works out to a societal cost of more than 200 billion euros (Björling 2021).

At the same time, buildings are one of the heaviest loads on our energy systems. In the European Union, about 40% of all energy use traces back to buildings, and a large share of that goes simply to conditioning the air inside for comfort—heating it in winter, cooling it in summer (Björling 2021). The global picture looks the same. Buildings account for roughly 30% of the world’s final energy use and more than half of its electricity, and demand for space cooling has been climbing fast: it more than tripled between 1990 and 2016, and cooling is now the main driver of rising energy use in the sector (Chung et al. 2023). On the emissions side, buildings are responsible for something like 40% of total energy use and around 30% of carbon emissions worldwide, which is why building-performance simulation tools have become a standard part of how engineers study and improve the way buildings perform (Markarian and Azar 2023).

Here is where the two stories collide. Ventilation keeps people healthy, but moving and conditioning all that fresh air costs energy—and the moment energy becomes expensive, ventilation starts to look like a place to cut. This is not a hypothetical. When the energy crisis hit in 1973, the energy spent on ventilating buildings suddenly read as a liability, and ventilation requirements were lowered in the standards that followed. Buildings were also sealed more tightly to stop air leaking through the envelope, especially in the Nordic countries, which trimmed the supply of fresh air even further (Björling 2021).

The pressure to keep ventilation low has not always come from energy concerns alone. One of the clearer cases on record involves the ASHRAE ventilation standards of the 1980s. A 1981 proposal set two ventilation levels for offices—a lower one for offices without smokers and a higher one where smoking was allowed. The tobacco industry read this as a direct threat to its business; an internal Philip Morris memo concluded that the higher requirement would at least double the maintenance costs of any workplace that permitted smoking. The 1981 standard was never adopted, and through a string of revisions the industry managed to keep stricter ventilation requirements from being enforced until 2000 (Björling 2021). The episode is a useful reminder that the numbers written into a ventilation standard are not purely technical. They sit at the meeting point of health, energy, and commercial interest.

The trouble is that trading away air quality to save energy is a bad bargain. One estimate puts the projected societal cost to a building’s occupants at roughly nine times the cost of the energy used in that building—a ratio of about nine to one (Björling 2021). Cutting ventilation to save energy can therefore cost society far more than it ever saves. And yet policy often leans the other way. The European Union’s building-energy framework is guided by an “energy efficiency first” principle.

It means that when energy goals collide with other goals—healthy indoor air among them—the energy side tends to win, and the other requirements get pushed down toward the legal minimum (Björling 2021). The standards bear this out. In a standard 10 m² single-occupant office, the required ventilation rate was lowered by 43% between the 2006 and 2019 European guidelines, a change that effectively accepts 30% of occupants being dissatisfied where the older guideline accepted 20% (Björling 2021). So buildings are being sealed tighter at the very moment the rules are letting in less fresh air.

There is a broader shift in the background here as well. It might seem that natural ventilation—letting air move through a building on its own, without fans—would be the more sustainable choice, and in many ways it does have real advantages for the people inside. But mechanical systems tend to win on predictability, control, and heat recovery, and as older buildings are renovated to modern energy standards, many are likely to be fitted with mechanical ventilation in place of what they had (Björling 2021). Whatever form it takes, the point that holds is simple: the goal is adequate ventilation, and the tools we design with need to support it.

Put plainly, this is not only an engineering question but close to a question of basic rights. The WHO has suggested that every person’s right to a healthy indoor environment should be treated as a basic human right, and adequate ventilation sits at the center of meeting it. The old mantra “build tight—ventilate right” captures the balance well, but it is the second half that tends to get forgotten (Björling 2021).

If the goal is to design buildings that ventilate well, designers need a way to predict how air will actually move through a space before it is built. CFD is that way—but it is slow. Cities are growing, buildings are getting more varied and more complex, and practice increasingly needs fast answers early on: quick comparisons between ventilation options, pollutant-dispersion checks, wind-load assessments, all at the stage where decisions are still cheap to change. A defensible CFD run, though, means building a fine mesh, solving a set of nonlinear equations through many iterations, and leaning on serious computing power, and that combination keeps CFD out of the fast, early phase of design where it would help the most (Mao et al. 2025). The scale of the wait is easy to underestimate. A single high-fidelity simulation of airflow in a built environment can take days to finish even on a dedicated supercomputer, which puts any real exploration of many design options out of reach with conventional methods (Caron, Lauret, and Bastide 2025).

This is the gap that machine-learning surrogates are meant to fill. Instead of solving the flow equations from scratch on a dense mesh for every new case, a surrogate model learns the mapping from a building’s design parameters to its flow field and then returns a prediction in a fraction of the time. Physics-informed neural networks (PINNs) are one promising version of the idea.

A PINN is a neural network with the governing physical laws built directly into its training, so it can produce solutions to the Navier-Stokes equations while sidestepping some of CFD’s heaviest costs—the dense meshing, the large training datasets, and the slow march through time steps (Botarelli et al. 2025). A fast, physically grounded surrogate could return flow prediction to the early, iterative part of design—and that is where this paper begins.

1.2 Problem Statement

Surrogate models promise speed, but the idea is still young, and the open problems are real. Reviews of machine learning for CFD in the built environment keep returning to the same short list of difficulties: surrogate models need to generalize better, they need to be more interpretable, the methods need to scale, and the cost of building the models in the first place needs to come down (Caron, Lauret, and Bastide 2025). The physics-informed branch carries a more specific gap. Most of the work on PINNs so far has stayed on clean benchmark problems—comparing ways of imposing boundary and initial conditions, training across consecutive time windows, laying out general best practices—and very little of it has tested whether PINNs actually hold up when they are pointed at a real, applied scenario (Botarelli et al. 2025).

The hardest of these problems is generalization. A surrogate can be very accurate as long as it is asked about cases that resemble the ones it was trained on, and then fall apart the moment it meets a geometry or a boundary condition it has never seen. It helps to name the two cases that behave so differently: *interpolation* is prediction at inputs inside the range of the training data, surrounded by cases the model has seen, while *extrapolation* is prediction beyond that range, at conditions past anything it was trained on. A network is typically strong at the first and weak at the second; the formal statement of both is given in Section 2.3.2. The reason is built into how these models learn. A purely data-driven network reasons by induction from its examples and carries no explicit knowledge of the physical laws—the Navier-Stokes equations—that actually govern the flow (Mao et al. 2025). The way forward that the literature keeps pointing to is to push more physics into the training process and to lean on architectures such as PINNs and graph neural networks, which, for all their promise, are still underused in built-environment work (Caron, Lauret, and Bastide 2025).

Those gaps are written for the machine-learning-CFD research community. For an architect sitting in front of a design, the same problems show up as four concrete, everyday obstacles.

The first is the iteration-time gap. Architectural design moves by rapid iteration—nudge a wall, resize an opening, change a façade—often dozens of small variants in a single working session. A single CFD case in this paper takes about 138 seconds on engineering-class hardware, and the more demanding three-dimensional, time-dependent cases run for hours or even days each (Mao et al. 2025). The iteration a designer wants is measured in seconds; the iteration CFD delivers is measured in hours. This is the plainest, most architect-legible reason a faster surrogate is worth building.

The second is the expertise barrier. Running a CFD case you can actually trust takes a real specialist’s skill set—judging mesh quality, choosing a turbulence model, setting boundary conditions, diagnosing whether the solver has converged, and making sense of the output. Most architects have neither that training nor the time for it. The usual result is that CFD gets handed to an outside consultant once the design has already mostly settled, which quietly moves the analysis out of the design loop and into after-the-fact verification. A surrogate that is fast enough to run from inside the designer’s own modeling tool turns that around, putting flow feedback in the architect’s hands while the design is still moving.

The third is a mismatch with the way design tools work. Contemporary practice runs on parametric modelers—Rhino with Grasshopper, Revit, and the like—that describe a building as a network of geometric operations driven by continuous parameter dials (Rodrigues Marques Sakiyama et al. 2021; Chung et al. 2023). A CFD pipeline that has to rebuild its mesh and restart a fresh solve every time one of those dials is turned simply does not fit that way of working. A surrogate that reads the same dials directly and returns the flow field near-instantly fits it without friction.

The fourth is the generalization bottleneck again, now seen from the surrogate’s own side. A purely data-driven surrogate trained on a finite sweep of CFD cases extrapolates poorly to geometries and conditions outside that sweep (Mao et al. 2025; Chen et al. 2025). This is the central open question for using machine learning as a CFD surrogate in buildings, and it is the main reason this paper compares three architectures rather than settling on one. The hope is that the physics built into a PINN and the spatial structure built into a GNN will improve generalization precisely where the plain data-driven model is weakest.

1.3 Research Questions

The four problems above point to a handful of questions this paper takes on. They move from the most basic—whether this kind of model works at all—through the central comparison that motivates the whole study, and on to the practical question of whether the predictions are good enough to feed the comfort calculations a designer ultimately cares about.

1. Can deep-learning surrogates predict indoor airflow fields with enough accuracy to be useful in architectural design?
2. Does physics-informed training improve generalization compared with a purely data-driven baseline?
3. How does surrogate accuracy scale with the size of the training set, and which architecture is the most data-efficient?
4. How reliably do the predicted fields carry through into downstream thermal-comfort predictions under ISO 7730?

Each of these questions is answered later in the paper. The first two—field accuracy and physics-informed generalization—sit at the center of the architecture comparison: Chapter 4 defines the three models and lays out how interpolation and extrapolation are tested apart from one another, and Chapter 5 reports how each model holds up. The third question, on data efficiency, follows from the staged data-generation design in Chapter 4 and the results that come out of it. The fourth, on downstream comfort, is handled by the comfort pipeline built in Chapter 4 and the comfort results that follow it, with the supporting verification collected in an appendix.

1.4 Contributions

The contributions of this paper follow directly from those questions. In short, the work offers:

- a systematic comparison of three deep-learning architectures—a data-driven artificial neural network (ANN), a physics-informed neural network (PINN), and a graph neural network (GNN)—as surrogates for indoor airflow, all on a single shared problem and dataset;
- a parametric study across four room geometries: an open room, and three variants with an internal partition placed at the quarter, half, and three-quarter points along the room’s length;
- an explicit separation of interpolation from extrapolation, with the velocity and thermal parameter axes tested on their own across six evaluation splits;
- a data-efficiency study at three training-set sizes, which asks how far accuracy can be stretched against the cost of generating CFD data;
- a downstream thermal-comfort pipeline that couples the field surrogate to the ISO 7730 measures—Predicted Mean Vote (PMV), Predicted Percentage Dissatisfied (PPD), and Draught Rate (DR).

Taken together, these shift the question from “can a neural network reproduce a CFD field” toward the more useful one: which kind of network earns a place in a designer’s toolkit, and why.

1.5 Paper Organization

The rest of the paper is laid out as follows. Chapter 2 sets up the theoretical background—computational fluid dynamics, turbulence modeling, thermal comfort, and the deep-learning foundations the later chapters build on. Chapter 3 surveys the related work on machine-learning surrogates for fluid simulation and places this paper against what has come before. Chapter 4 lays out the methodology in full: the CFD setup, the automated parameter sweep that generates the data, the three surrogate architectures, the downstream comfort pipeline, and the evaluation framework that ties them together. Chapter 5 presents the results, opening with a deep dive on the open and half-partition configuration and then widening to a robustness summary across all four geometries. Chapter 6 steps back to discuss what the findings mean for architectural design workflows. Chapter 7 sets out the limitations and the work still left to do, and Chapter 8 concludes.

2

Background

This chapter sets out the theoretical background for the work that follows. It covers four areas in turn: the physics of indoor airflow and heat transfer, and the computational fluid dynamics used to solve it; the thermal-comfort standards that turn airflow and temperature fields into a measure of occupant comfort; the foundations of the neural networks used as surrogate models; and the surrogate-modelling approaches this paper compares.

2.1 Computational Fluid Dynamics

2.1.1 Governing Equations: Navier-Stokes

Computational fluid dynamics, or CFD, is the practice of solving the equations that govern how a fluid moves, on a computer, when those equations are too hard to solve by hand. For the air in a room, “how the fluid moves” means three things at once: where the air goes, how fast it travels, and how warm it is at every point. These three are bound together by conservation laws that physics holds to be true for any fluid – the conservation of mass, of momentum, and of energy. Written out for a fluid in three dimensions, in the general compressible form, they read (Pope 2000; Anderson 1995):

$$\text{Mass: } \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \quad (2.1)$$

$$\text{Momentum: } \rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g}, \quad (2.2)$$

$$\text{Energy: } \rho c_p \left(\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right) = \nabla \cdot (k_{\text{air}} \nabla T) + \Phi + Q. \quad (2.3)$$

Here $\mathbf{u}$ is the velocity of the air – a vector, since air moves in some direction – while ρ is its density, p the pressure, and T the temperature. The remaining symbols are material properties and sources: c_p is the specific heat of air at constant pressure, k_{air} its thermal conductivity, $\mathbf{g}$ gravity, $\boldsymbol{\tau}$ the viscous stress tensor (the internal friction of the fluid), Φ the heat released as that friction dissipates motion.

Q is any heat added from outside, such as a radiator or a warm body. It helps to read each equation as a plain statement before treating it as algebra. The first, continuity, says that mass cannot appear or vanish: whatever air flows into a small region must either flow back out or show up as a change in density, with nothing created or destroyed along the way. The second, the momentum equation, is Newton's second law – force equals mass times acceleration – written for every infinitesimal parcel of air. The left-hand side is the parcel's mass times its acceleration; the right-hand side sums the forces acting on it: a push from pressure differences ($-\nabla p$, since air accelerates from high pressure toward low), a drag from friction against neighbouring parcels ($\nabla \cdot \tau$), and the pull of gravity ($\rho \mathbf{g}$). The third, the energy equation, follows the heat: temperature changes as warmth is carried along by the moving air, conducted through it, generated by friction, or supplied by a source.

The viscous stress τ in the momentum equation is itself a small model of how a fluid resists being sheared. For a Newtonian fluid – and air is one – it is proportional to the rate at which the flow is deforming (Pope 2000):

$$\tau = \mu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) - \frac{2}{3} \mu (\nabla \cdot \mathbf{u}) \mathbf{I}, \quad (2.4)$$

where μ is the dynamic viscosity, a measure of how thick or resistant the fluid is, and $\mathbf{I}$ is the identity tensor. The first term captures the friction from the fluid being stretched and sheared in every direction; the second is a small correction that matters only when the fluid is being compressed. The point worth holding onto is that viscosity is what couples one parcel of air to the next, so that a fast jet from a vent drags the still air around it into motion rather than sliding past it untouched.

These equations are deceptively compact. Written down, the unsteady three-dimensional system fits on a line or two; solved, it has resisted every attempt at a general closed-form answer. Whether smooth solutions to the three-dimensional incompressible equations always exist is still an open question – one of the seven Clay Millennium Prize Problems, with a million-dollar reward attached to a proof (Clay Mathematics Institute 2000). For engineering the consequence is simple: we cannot write the answer down, so we approximate it numerically, which is what CFD does and what the rest of this section builds toward.

For the air in a room the full compressible system can be trimmed. Indoor air moves slowly, far below the speed of sound, and at those speeds its density barely changes, so it can be treated as constant. Holding the density fixed collapses the system into the *incompressible* Navier-Stokes equations, the form found in nearly every introductory CFD text and the one this paper treats as canonical (Pope 2000; Anderson 1995):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \quad \nabla \cdot \mathbf{u} = 0. \quad (2.5)$$

The momentum equation reads exactly as before – pressure, friction, body force – but the friction term is now written with the kinematic viscosity $\nu = \mu/\rho$ and the Laplacian $\nabla^2 \mathbf{u}$, which measures how far a parcel’s velocity sits from the average of its neighbours and so smears sharp differences in the flow out over time. The body force $\mathbf{f}$ gathers anything pushing on the fluid from outside, most commonly buoyancy, the slow rise of warm air that can drive much of the movement in a heated or occupied room. The continuity equation has shrunk to a single tidy constraint, $\nabla \cdot \mathbf{u} = 0$, which is the formal way of saying that same “fluid cannot pile up”: at every point, as much air leaves as arrives.

One simplification is deliberately left undone. Because thermal comfort depends on temperature as much as on air speed, this paper keeps the energy equation; the flow field and the temperature field are solved together. The coupling in the model solved here runs through the temperature dependence of the air’s density rather than through a buoyancy body force, because gravity is switched off; the forced-convection regime this puts the study in is set out with the solver in Section 4.2.2. The incompressible momentum and continuity equations above are the teaching form used throughout the methodology, while the full compressible, weakly-coupled system is held back for the description of the COMSOL solver setup, where the physics as actually solved is laid out in detail.

2.1.2 Turbulence Modeling: The k- ϵ Model

The equations so far describe any flow, smooth or chaotic. The difficulty is that the air in a real room is almost never smooth. Push air through a vent and it breaks into a tangle of swirls and eddies of every size, from room-spanning circulations down to millimetre-scale wisps that flicker and vanish. This is *turbulence*. In principle the Navier-Stokes equations already contain all of it, and one could simply solve them on a mesh fine enough and a time step short enough to capture every eddy. That approach, direct numerical simulation (DNS), is the most accurate option and, for a room, a hopeless one: the spread of scales is so wide that the cost lands far beyond what any design office could afford (Mao et al. 2025).

The practical way out is to stop tracking every eddy and solve instead for the *average* flow, modelling the effect of the turbulence rather than resolving it. There is a hierarchy of how far to take this. Large-eddy simulation (LES) resolves the big, energy-carrying eddies and models only the small ones; it is more accurate than the cheaper alternatives but still costly, since it has to follow the flow as it churns in time (Mao et al. 2025).

The far more common choice for buildings is to average the turbulence out entirely. Because extreme accuracy is rarely needed for an engineering answer, this Reynolds-averaged approach – Reynolds-averaged Navier-Stokes, or RANS – remains the standard for building airflow (Caron, Lauret, and Bastide 2025; Mao et al. 2025).

Averaging starts by splitting every quantity into a steady mean and a fluctuating part. For the velocity,

$$\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \mathbf{u}'(\mathbf{x}, t), \quad (2.6)$$

where $\bar{\mathbf{u}}$ is the time-averaged velocity – what a slow anemometer would read – and $\mathbf{u}'$ is the rapid, zero-average jitter of the turbulence around it. Substitute this split into the momentum equation, average over time, and the result looks almost like the original, with one stubborn addition: an extra stress built from the averaged products of the fluctuations, $\overline{\mathbf{u}'\mathbf{u}'}$, known as the *Reynolds stresses*. This term carries the net effect of all the unresolved eddies on the mean flow, and it cannot be computed from the mean flow alone, so the averaged equations end up with more unknowns than equations. This is the famous *closure problem* of turbulence, and a turbulence model is precisely a recipe for closing it (Pope 2000; Wilcox 2006).

The most widely used recipe rests on an analogy. The Reynolds stresses act on the mean flow much as ordinary viscous friction does, so the *Boussinesq hypothesis* models them in the same mathematical shape, introducing a *turbulent* or *eddy* viscosity μ_t that plays the part for turbulence that μ plays for molecular friction. The eddy viscosity is not a property of the air; it is a property of the flow, larger where the turbulence churns more vigorously. The k - ε model builds μ_t from two scalar fields it solves for directly – the turbulent kinetic energy k , which measures how much energy is held in the fluctuations, and the dissipation rate ε , which measures how fast that energy is ground down into heat (Launder and Spalding 1974; Wilcox 2006):

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}, \quad (2.7)$$

where C_μ is an empirical constant. The label “two-equation model” comes from the fact that k and ε each get a transport equation of their own, of the same convection-diffusion-source shape as the equations already seen:

$$\frac{\partial(\rho k)}{\partial t} + \nabla \cdot (\rho k \bar{\mathbf{u}}) = \nabla \cdot \left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right] + P_k - \rho \varepsilon, \quad (2.8)$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + \nabla \cdot (\rho \varepsilon \bar{\mathbf{u}}) = \nabla \cdot \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \nabla \varepsilon \right] + C_{1\varepsilon} \frac{\varepsilon}{k} P_k - C_{2\varepsilon} \rho \frac{\varepsilon^2}{k}. \quad (2.9)$$

Each equation reads the same way, left to right: the change in k (or ε) over time, plus the amount carried along by the mean flow, equals the amount that spreads by diffusion, plus what is produced, minus what is destroyed. In the k equation, P_k is the production of turbulence – energy drawn out of the mean flow by shear – and $\rho \varepsilon$ is its destruction, the rate at which eddies dissipate into heat. The ε equation carries the same structure, with the production and destruction of k rescaled by ε/k to give them the right units and tuned by two further constants. The five constants take the standard values $C_\mu = 0.09$, $C_{1\varepsilon} = 1.44$, $C_{2\varepsilon} = 1.92$, $\sigma_k = 1.0$, and $\sigma_\varepsilon = 1.3$; none of them is derived from first principles. They were fitted once, decades ago, against a handful of canonical flows, and have been carried forward largely unchanged ever since (Launder and Spalding 1974).

That history is exactly why k - ε is the default for indoor air. It is stable, it is cheap, and it gives reasonably accurate results for the kinds of airflow found in most rooms, which is why one study after another has judged it the best balance of performance and cost for ventilation problems (Rodrigues Marques Sakiyama et al. 2021; Mao et al. 2025). It is not flawless: it can struggle where buoyancy is strong or temperature gradients are steep, conditions its constants were never tuned for (Rodrigues Marques Sakiyama et al. 2021). For the ventilated room studied here those conditions stay mild, and k - ε is the closure used to generate every CFD case in this paper.

2.1.3 Numerical Methods and Computational Cost

Because the Navier-Stokes equations cannot be solved exactly, CFD solves them approximately, by chopping the room into a mesh of small cells and turning the continuous equations into a large set of algebraic ones – one set per cell – that a computer can grind through. The classical ways of doing this, whether finite differences, finite volumes, finite elements, or spectral methods, all share a catch: they are accurate only if the flow varies smoothly from one cell to the next, so the mesh has to be fine enough to resolve the smallest feature that matters (Kochkov et al. 2021). Sharpen the flow and the mesh must be refined, and a finer mesh means more cells, more equations, and more time.

The equations are also nonlinear – the velocity appears multiplied by itself in the $(\mathbf{u} \cdot \nabla)\mathbf{u}$ term – so they cannot be solved in a single pass. The solver has to guess a flow field, check how badly the guess violates the equations, correct it, and repeat, iterating until the leftover error falls below a tolerance. The combined cost of a fine mesh and a long iterative solve climbs steeply with how turbulent the flow is: as a rule of thumb, a tenfold increase in Reynolds number – a tenfold step up in turbulence – multiplies the computational cost roughly a thousandfold (Kochkov et al. 2021).

For airflow in the built environment the bill comes from two places in particular: the fine meshes that complex geometry such as façades and ventilation ducts demands, and the high-fidelity turbulence models needed to resolve transient detail (Mao et al. 2025). Put together, these push the time for a single defensible simulation into the range of hours to days, and any study that needs many scenarios pays that cost again for every one, scaling roughly in proportion to the number of cases (Mao et al. 2025). This is the wall the rest of the paper sets out to get around – not by making CFD itself faster, but by training a model to imitate its output and skip the solve entirely.

2.2 Indoor Environment and Thermal Comfort

2.2.1 Indoor Environmental Quality

A flow field on its own says nothing about whether a room is a good place to be. To matter, the airflow has to be tied to how the people in the room actually feel, and that connection runs through indoor environmental quality – the bundle of air quality, temperature, humidity, and air movement that together decide whether an occupant is comfortable and healthy. Two things shape it: how much fresh air a space receives, and how well that air is distributed once it arrives. Both matter, and a room can fail on the second even when it succeeds on the first (Björling 2021).

The stakes are not small. Good ventilation improves air quality, and with it the health, the cognitive performance, and the work efficiency of the people inside; during the recent pandemic it was singled out as one of the clearest ways to slow the spread of airborne infection by bringing in outdoor air (Son, Kim, and Koo 2024). The catch, as always, is that ventilation also costs energy, which is what keeps it in tension with the drive for efficient buildings (Son, Kim, and Koo 2024). Of the several strands of indoor environmental quality, the one this paper predicts is thermal comfort – whether the combination of temperature and air movement at a point in the room would leave an occupant feeling neither too warm nor too cool.

2.2.2 Thermal Comfort: PMV and PPD

Thermal comfort sounds subjective, and for any one person it is. The standard way to make it measurable comes from treating the human body as a small heat engine that has to stay in balance. The body produces heat through metabolism and loses it to its surroundings, and comfort is the state in which production and loss are matched without the body having to work hard to keep them so. Written as an energy balance, this is (Guo et al. 2025):

$$Q_{\text{balance}} = (M - W) - (Q_{\text{conv}} + Q_{\text{rad}} + Q_{\text{evap}} + Q_{\text{res}}), \quad (2.10)$$

where M is the metabolic heat the body generates, W is any mechanical work it does on its surroundings, and the four loss terms are heat carried off by convection (Q_{conv}), radiated to surrounding surfaces (Q_{rad}), lost through the evaporation of sweat (Q_{evap}), and exhaled in breathing (Q_{res}). When Q_{balance} is zero the body is in equilibrium; when it is not, the person feels too warm or too cold.

The most widely used model built on this idea is the Predicted Mean Vote, or PMV. It predicts the average thermal sensation a large group of people would report in a given environment, on a seven-point scale running from -3 (cold) through 0 (neutral) to $+3$ (hot), from six inputs: air temperature, radiant temperature, humidity, air velocity, the occupants' metabolic rate, and their clothing insulation (Chung et al. 2023). Because it rests on the steady-state heat balance rather than on how people psychologically adapt to their surroundings, PMV is at its most reliable in conditioned spaces and tends to drift from reported comfort in naturally ventilated buildings, where occupants open windows and adjust their expectations (Chung et al. 2023). The mechanically conditioned room studied here sits squarely in the regime PMV was built for.

The model maps the environment and the occupant onto the sensation scale through a single expression (ISO 2005; Fanger 1970):

$$\text{PMV} = [0.303 \exp(-0.036 M) + 0.028] L. \quad (2.11)$$

The bracketed factor is a sensitivity that shrinks as the metabolic rate M rises — a more active body registers a given thermal imbalance as a smaller sensation — and L is the *thermal load* on the body, the net heat imbalance left over once every avenue of heat loss has been accounted for. The load expands into the metabolic heat production minus six separate loss channels (ISO 2005):

$$\begin{aligned}
L = & (M - W) \\
& - 3.05 \times 10^{-3} [5733 - 6.99 (M - W) - p_a] \\
& - 0.42 [(M - W) - 58.15] \\
& - 1.7 \times 10^{-5} M (5867 - p_a) \\
& - 0.0014 M (34 - t_a) \\
& - 3.96 \times 10^{-8} f_{cl} [(t_{cl} + 273)^4 - (\bar{t}_r + 273)^4] \\
& - f_{cl} h_c (t_{cl} - t_a).
\end{aligned} \tag{2.12}$$

Read top to bottom, the equation is a heat ledger. The first line, $(M - W)$, is the heat the body produces internally. Each line after it subtracts one way that heat leaves. The first subtracted term is the slow diffusion of moisture through the skin; the second is evaporative loss from regulatory sweating, which only switches on once the body is working hard enough – it applies when $M - W$ rises above 58.15 W/m^2 , the metabolic rate of a person at rest (one “met”), and is set to zero below that. The third and fourth terms are heat carried out in the breath, latent (the moisture in exhaled air) and dry (its warmth) respectively. The last two are the ones that matter most in a room: the fifth is radiative loss to the surrounding surfaces, written as a difference of fourth powers of absolute temperature because radiation follows the Stefan-Boltzmann law, and the sixth is convective loss to the moving air. The remaining symbols are the occupant’s water-vapour pressure p_a , the air temperature t_a , the clothing surface temperature t_{cl} , the mean radiant temperature $\bar{t}_r$, the clothing area factor f_{cl} , and the convective heat-transfer coefficient h_c . The fixed numbers are empirical: 5733 Pa is the saturated vapour pressure at the body’s skin temperature, and the coefficient 3.96×10^{-8} folds together the Stefan-Boltzmann constant, an assumed clothing emissivity near 0.97, and a unit conversion.

For design it is more useful to know not the average vote but the share of people who will be unhappy. The Predicted Percentage Dissatisfied, PPD, converts a PMV value into exactly that (ISO 2005; Fanger 1970):

$$PPD = 100 - 95 \exp(-0.03353 \text{ PMV}^4 - 0.2179 \text{ PMV}^2). \tag{2.13}$$

The curve bottoms out at $\text{PMV} = 0$, where it does not reach zero but flattens at 5% – Fanger’s reading of the fact that in any group a few people will be dissatisfied no matter how carefully the room is tuned – and climbs steeply as PMV moves away from neutral in either direction. Two coefficients are Fanger’s original 1970 regression constants, and they have carried through every edition of the standard unchanged.

The PMV and PPD calculations in this paper are not re-implemented from scratch. They use a published, validated port of the reference program printed in ISO 7730:2005 Annex D, which its authors checked against the ten canonical test cases in that annex and matched to within 0.01 PMV units and half a percentage point of PPD (Tartarini and Schiavon 2020). That agreement is taken as the formal verification of the comfort calculation used throughout.

Finally, the standard sorts indoor environments into three quality categories, each setting a simultaneous ceiling on PMV, PPD, and the draught rate introduced next. Table 2.1 lists them; the same three appear, with identical thresholds and the labels I-III, in EN 16798-1:2019 (European Committee for Standardization 2019).

Table 2.1: Thermal-environment categories of ISO 7730:2005, with the matching EN 16798-1:2019 labels.

Category	PMV	PPD	DR
A (I)	< 0.2	< 6 %	< 10 %
B (II)	< 0.5	< 10 %	< 20 %
C (III)	< 0.7	< 15 %	< 30 %

Throughout the results, the comfort fraction reported for a room is the share of grid cells that satisfy $|PMV| \leq 0.5$ – Category B, the usual design target for general office space and the threshold most often quoted in the CFD-comfort literature (Stamou and Katsiris 2006; Sevilgen and Kilic 2011).

2.2.3 Draught Rate

Comfort is not only about the average warmth of a room. A person can be comfortable on the whole and still be annoyed by a cold draught on the back of the neck, and air movement that feels pleasant in summer can feel like a complaint in winter. Standards pull on this in two directions. Elevated air speed can *extend* the comfort range in warm conditions: ASHRAE 55 lets the average air speed rise with temperature up to 1.2 m/s, and EN 16798-1 permits up to 1.5 m/s, which can widen the acceptable temperature band by as much as 4 °C (Rodrigues Marques Sakiyama et al. 2021). In neutral or cool conditions, though, unwanted air movement becomes a source of local discomfort in its own right.

That local discomfort is quantified by the draught rate, DR – the predicted percentage of people who would be bothered by air movement on the exposed neck (ISO 2005; Fanger et al. 1988):

$$DR = (34 - t_a) (v_a - 0.05)^{0.62} (0.37 v_a T_a + 3.14). \quad (2.14)$$

Each of the three factors carries a piece of the intuition. The first, $(34 - t_a)$, is how far the air sits below skin temperature: cooler air pulls away more heat and feels more like a draught, and when the air reaches skin temperature the whole expression vanishes. The second, $(v_a - 0.05)^{0.62}$, makes the draught grow with air speed v_a but with diminishing returns, and builds in a threshold – below 0.05 m/s the air is effectively still and the rate is taken as zero. The third factor adds the effect of *fluctuation*: steady air is less annoying than gusty air at the same average speed, so a higher turbulence intensity T_u raises the rate. The model is defined for air speeds between 0.05 and 3 m/s and for people at light, mostly seated activity; the standard notes that it can overstate discomfort at the arms and feet and understate it during more active tasks (ISO 2005).

The turbulence intensity the draught model needs is itself read off the CFD solution. Under the isotropic-turbulence assumption built into the k - ε closure, it follows directly from the turbulent kinetic energy k and the local air speed (Hinze 1975; ANSYS, Inc. 2024):

$$T_u = 100 \cdot \frac{\sqrt{2k/3}}{v_a}. \quad (2.15)$$

The numerator is the typical size of the velocity fluctuations implied by k , and dividing by the mean speed turns it into a percentage. Computing T_u this way, point by point from the resolved k field, is standard in the CFD-comfort literature (Koskela et al. 2001; Stamou and Katsiris 2006; Sevilgen and Kilic 2011; Cheng, Niu, and N. Gao 2012), and it avoids the cruder fallback the standard offers when no field data are available, which is simply to assume a fixed 40% (ISO 2005).

2.2.4 Mean Radiant Temperature

One symbol in the PMV load was left unexplained: the mean radiant temperature $\bar{t}_r$. It matters because a body exchanges heat with its surroundings not only by touching the air but by radiating to the surfaces around it, which is why a room with cold walls feels chilly even when the air is warm, and why sitting beside a sunlit window feels hot. The mean radiant temperature is the single equivalent temperature of a uniform enclosure that would radiate to the occupant exactly as the real, unequal surfaces do. It is computed by weighting each surface's temperature by how much of the occupant's field of view that surface fills (ISO 1998):

$$\bar{t}_r = \left(\sum_i T_i^4 F_{p-i} \right)^{1/4} - 273.15, \quad (2.16)$$

where T_i is the temperature of surface i in kelvin and F_{p-i} is the *view factor* from the evaluation point

to that surface – the fraction of the point’s surroundings that surface takes up. The temperatures enter as fourth powers because radiation obeys the Stefan-Boltzmann law, and the final subtraction of 273.15 converts the result from kelvin back to degrees Celsius. For a closed room the view factors must account for everything the point can see, so they sum to one, $\sum_i F_{p-i} = 1$ (Modest 2013). In the two-dimensional room analysed here, the view factor to each of the four walls is simply the share of the full circle the wall subtends at the point, $F_{p-i} = \alpha_i/2\pi$, where α_i is the angle the wall spans as seen from that point.

This treats the occupant as a small point that radiates equally in every direction, the natural two-dimensional reduction of the small-sensor model the standard recommends when detailed angle factors for a human body are not available (ISO 1998). The fuller, direction-dependent angle factors for a seated or standing person would refine this in three dimensions, but they do not fit the two-dimensional slice studied here. Mean radiant temperature is properly a three-dimensional, occupant-position-dependent quantity, so the two-dimensional point reduction is a deliberate approximation, and it is carried as one of the comfort-model limitations of this study. Air temperature does not enter the equation at all: across the path lengths of a normal room, air is effectively transparent to the long-wave radiation that carries this heat, so only the bounding surfaces contribute.

2.3 Deep Learning Foundations

2.3.1 Feed-Forward Neural Networks

The physics above describes what a CFD solver computes. The rest of this paper is about replacing that solver with something far faster, and the tool it reaches for is the neural network. Deep learning has become the dominant approach in machine learning for a few practical reasons: deep networks can approximate essentially any continuous function, they cope well with large datasets, and they train efficiently on graphics processors (Caron, Lauret, and Bastide 2025). Those traits are what make a network a plausible stand-in for an expensive simulation.

At its simplest a neural network is a supervised learner. It is handed a set of examples in which both the input x and the desired output y are known – here, a set of design parameters and the CFD field they produce – and it adjusts itself until it reproduces that mapping. Once trained, the same fixed function can be applied to inputs it has never seen and, if all goes well, return an accurate prediction (Calzolari and Liu 2021).

The feed-forward network, the basic form, builds this mapping by stacking simple layers. A network with L hidden layers and parameters $\theta = \{W_\ell, \mathbf{b}_\ell\}$ computes

$$f(\mathbf{x}; \theta) = W_{L+1} \phi_L(W_L \phi_{L-1}(\cdots \phi_1(W_1 \mathbf{x} + \mathbf{b}_1) \cdots) + \mathbf{b}_L) + \mathbf{b}_{L+1}, \quad (2.17)$$

reading from the inside out. The input $\mathbf{x}$ is multiplied by a weight matrix W_1 and shifted by a bias vector $\mathbf{b}_1$; the result passes through a nonlinear activation ϕ_1 — a simple function such as ReLU, GELU, or tanh applied to each component — and the output of that becomes the input to the next layer, and so on up the stack. The weights and biases are the knobs the network turns during training; the activations are what let it represent something more interesting than a straight line, since a stack of purely linear layers would collapse back into a single linear map. “Feed-forward” means the information flows one way, from input to output, with no loops back, which is what separates this architecture from recurrent networks and from the graph-structured networks met later.

The reason a network of this kind is even a candidate for the job is a theoretical guarantee called the *universal approximation theorem*: a feed-forward network with a single hidden layer of finite width and a suitable nonlinear activation can approximate any continuous function on a bounded domain to any accuracy desired (Hornik, Stinchcombe, and White 1989; Cybenko 1989). This is the formal statement behind the loose claim that neural networks “can approximate anything.” It is worth being clear about what it does not say. It is an existence result, not a recipe: it promises that some set of weights works, but says nothing about how wide the layer must be, how to find those weights, or whether a network that fits its training data will do anything sensible on data drawn from somewhere else. Those last gaps are exactly what the next two parts are about.

2.3.2 Generalization and Regularization

Fitting the training data is not the goal; performing well on data the network has not seen is. A model can drive its training loss to almost nothing and still be useless, because it has memorised the quirks and noise of the training set rather than learning the underlying pattern — the failure known as *overfitting*, which comes bundled with the broader difficulties of poor generalization and weak interpretability that complex models tend to bring (Caron, Lauret, and Bastide 2025). The countermeasures, grouped under *regularization*, all push the model toward simpler solutions that are likelier to transfer: penalising large weights, randomly switching off units during training, and stopping the training before it begins chasing noise are the common ones.

There is a sharper way to see where generalization holds and where it breaks, and it runs straight to the heart of this paper. A neural network’s real strength is *interpolation*. Given enough training data, it predicts well on test cases that fall within the same range and distribution as what it was trained on. *Extrapolation* – predicting outside that range – is a different matter, and networks can fail at it even in simple cases (Calzolari and Liu 2021). For a CFD surrogate this is the whole question. A model trained on rooms with certain inlet speeds and temperatures will likely do fine on a new room whose conditions sit inside that envelope, but a designer will sooner or later push a parameter past the edge of the training range, and that is where a purely data-driven model is least trustworthy. Separating interpolation from extrapolation, and asking whether building physics into the model shores up the extrapolation case, is the central experiment of this paper, and the evaluation framework is built around exactly that split.

2.4 Surrogate Modeling Approaches

With the physics and the neural-network machinery in place, the remaining background is the family of methods that sit between them: surrogate models, which learn to imitate a simulation. It helps to start with a map of the territory. Chen and colleagues organise the methods used to predict building-related physical fields into two broad branches (Chen et al. 2025).

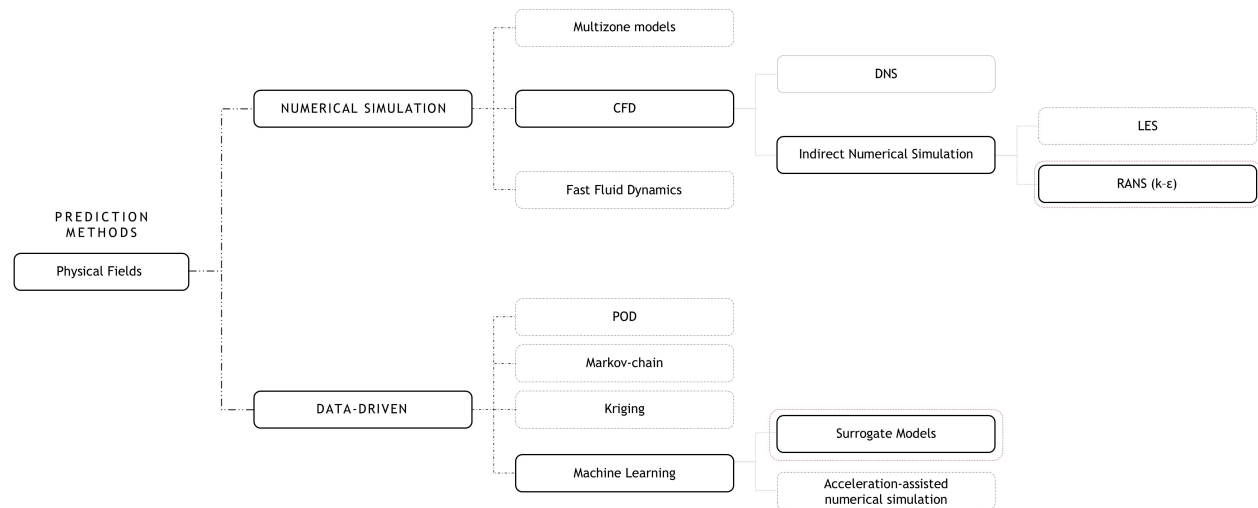


Figure 2.1: Methods for predicting building-related physical fields, adapted from Chen et al. (2025). This paper sits in the surrogate-model branch under machine learning, with reference data from a RANS $k-\epsilon$ solver.

The first is *numerical simulation* – multizone models, fast fluid dynamics, and CFD itself, which splits in turn into direct numerical simulation, where every turbulent scale is resolved, and indirect approaches such as LES and RANS, which model the smallest scales rather than resolving them. The $k-\varepsilon$ closure used in this paper is a two-equation RANS model and lives on this branch. The second is *data-driven* – proper orthogonal decomposition, Kriging, and machine-learning models, among others. Machine learning then divides once more, into *surrogate models*, which replace the solver outright with a learnable mapping, and acceleration methods, which keep a conventional solver and use learning only to speed parts of it up.

Figure 2.1 lays out this organisation. It is worth being explicit about where this paper sits within it. The CFD reference data are generated by a steady RANS $k-\varepsilon$ solver, on the indirect-numerical branch; the three models compared – the ANN, the PINN, and the GNN – are all *surrogate models* in Chen’s sense, learnable mappings that stand in for the solver rather than accelerate it. The work does not touch the acceleration branch: there is no enhanced turbulence model and no learning bolted onto the solver’s internals. The rest of this section takes the three surrogate families in turn.

2.4.1 Pure Data-Driven Surrogates

A surrogate model, in the plainest terms, is a data-driven stand-in for an expensive physics-based model. The recipe has three steps: a simulation tool generates a dataset, a portion of that data is used to build and train the surrogate, and a held-back portion is used to test that the surrogate really does reproduce the original model (Markarian and Azar 2023). Once trained, the surrogate answers in a fraction of the time the simulation would take, which is the whole point of building one.

Surrogates need not be neural networks. The classical options – polynomial regression, response-surface models, Kriging – fit a smooth function to the input-output data and work well for low-dimensional, gently behaved problems, but their accuracy falls away on the high-dimensional, strongly nonlinear ones (Mao et al. 2025). Indoor airflow, with its many design parameters and its sharp jets and recirculations, is firmly in the hard category, which is why this paper uses a neural network as its data-driven baseline rather than one of the classical fits.

Whatever form it takes, a purely data-driven surrogate carries a built-in dependence on its training data. Its accuracy hinges on how many samples it sees and how well they are spread across the input space: too few, or poorly distributed, and the predictions skew (Mao et al. 2025). And since each training sample is a full CFD run, generating enough of them is exactly the expense the surrogate was meant to avoid, so there is a real tension between how much data a surrogate needs and how much that data costs to produce (Mao et al. 2025).

Underneath all of it is a basic trade: a surrogate buys speed, and it usually pays for that speed with some loss of accuracy against the simulation it imitates (Calzolari and Liu 2021). The architectures that follow are, in large part, attempts to get a better deal in this trade – more accuracy, or less data, for the same speed.

2.4.2 Physics-Informed Neural Networks

A pure data-driven surrogate learns only from examples. It has no idea that the fields it predicts are meant to obey the laws of fluid motion, so nothing stops it from returning a flow that looks plausible but quietly violates the conservation of mass or momentum. A physics-informed neural network (PINN) closes that gap by writing the governing equations directly into the training objective. The network still takes coordinates as input and predicts the field, but alongside the usual penalty for disagreeing with data, it is also penalised for disagreeing with the physics – and the derivatives needed to check the physics are obtained by automatic differentiation of the network itself – the exact, chain-rule differentiation of the network’s output with respect to its input coordinates (Botarelli et al. 2025; Raissi, Perdikaris, and George E. Karniadakis 2019). Folding this prior knowledge into the model reduces its reliance on data and tends to help most in exactly the data-scarce settings where a pure data-driven model struggles (Chen et al. 2025; Zhao et al. 2024).

The physics enters through the loss. Where an ordinary network minimises a single data term, a PINN minimises a weighted sum of three (Raissi, Perdikaris, and George E. Karniadakis 2019; Ma et al. 2025):

$$\mathcal{L}_{\text{PINN}} = \lambda_{\text{data}} \mathcal{L}_{\text{data}} + \lambda_{\text{pde}} \mathcal{L}_{\text{pde}} + \lambda_{\text{bc}} \mathcal{L}_{\text{bc}}, \quad (2.18)$$

where the weights λ_{data} , λ_{pde} , and λ_{bc} set how much each term counts against the others. Each of the three deserves its own look.

The *data term* is the familiar one. It measures how far the network’s prediction $\hat{u}_\theta$ sits from the known CFD values u_i at the N_{data} points where the simulation has already given an answer:

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \|\hat{u}_\theta(\mathbf{x}_i) - u_i\|^2. \quad (2.19)$$

This is just the mean squared error a data-driven network would use on its own; it pins the prediction to the simulation wherever simulation data exist.

The *physics term* is what makes the network physics-informed. The governing equations can always be rearranged so that every term sits on one side and the other side is zero – a true solution drives the leftover, the *residual*, exactly to zero. Writing $\mathcal{R}(\cdot)$ for that residual, the physics loss is the mean squared residual when the network’s own output is substituted in, measured at a set of N_{col} *collocation points* scattered through the domain:

$$\mathcal{L}_{\text{pde}} = \frac{1}{N_{\text{col}}} \sum_{j=1}^{N_{\text{col}}} \|\mathcal{R}(\hat{u}_{\theta}(\mathbf{x}_j))\|^2. \quad (2.20)$$

The derivatives inside $\mathcal{R}$, such as gradients and Laplacians, are calculated using automatic differentiation. This means the network is differentiated with respect to its input coordinates directly, instead of using the kind of mesh needed in a finite-volume or finite-element solver.

The collocation points are still sampled points in the domain, so the method does not remove spatial sampling completely. It replaces a fixed solver mesh with a set of sampled points where the physics residual is checked.

By reducing this residual, the model is encouraged to satisfy the governing physics even at locations where no COMSOL data is provided. This is one reason a PINN can give reasonable predictions beyond the training data.

The *boundary term* enforces the conditions at the edges of the domain – the inlet velocity, the wall temperatures, the no-slip condition at solid surfaces – which the interior equations alone do not fix. With $\mathcal{B}(\cdot)$ the boundary operator and g the prescribed boundary value, it penalises any mismatch at the N_{bc} boundary points:

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_{k=1}^{N_{\text{bc}}} \|\mathcal{B}(\hat{u}_{\theta}(\mathbf{x}_k)) - g(\mathbf{x}_k)\|^2. \quad (2.21)$$

There is an important catch in all of this, and it is built into the word *penalised*. The physics and the boundary conditions are imposed softly – as terms the optimiser is asked to make small, not as hard constraints it must satisfy exactly. Minimising the loss shrinks the residuals but never guarantees they reach zero, so the physical laws end up only approximately obeyed (Caron, Lauret, and Bastide 2025; Ma et al. 2025). A PINN therefore trades the hard guarantees of a conventional solver for the flexibility of a learned model, and part of the craft of training one is balancing the three weights so that no single term overwhelms the others. This three-term form is the canonical teaching version of the PINN loss. The version implemented here expands the physics term into separate residuals – continuity, momentum, energy, and turbulence transport – and adds further terms, all of which are set out in the methodology.

2.4.3 Graph Neural Networks for Spatial Fields

The third architecture takes a different view of the problem. An ANN treats each prediction point in isolation, and a PINN ties points together only through the shared physics in its loss; neither is told, as part of its structure, which points lie near which. A graph neural network (GNN) builds that spatial structure in from the start. The idea is to apply neural networks to data laid out as a graph – a set of nodes joined by edges – which suits fluid problems naturally, since the mesh a CFD solver already uses is itself a graph: each cell a node, each shared face an edge (Caron, Lauret, and Bastide 2025).

A GNN works by letting nodes exchange information with their neighbours, a procedure called *message passing*. At each layer, every node gathers the current states of the nodes it connects to, combines them, and updates its own state accordingly. For a node v at layer ℓ , the update is (Ma et al. 2025; Gilmer et al. 2017; Hamilton, Ying, and Leskovec 2017):

$$\mathbf{h}_v^{(\ell+1)} = \sigma\left(W^{(\ell)} \cdot \text{AGG}(\{\mathbf{h}_u^{(\ell)} : u \in \mathcal{N}(v)\})\right), \quad (2.22)$$

where $\mathbf{h}_v^{(\ell)}$ is the hidden state of node v at layer ℓ , $\mathcal{N}(v)$ is the set of its neighbours in the graph, AGG is a permutation-invariant aggregation such as a mean, sum, or maximum, $W^{(\ell)}$ is a learned weight matrix for that layer, and σ is a nonlinear activation. In plain terms, every node listens to its neighbours, pools what it hears into a single summary, passes that through a learned linear map, and applies a nonlinearity – like a gossip network in which each grid point updates its view from what the points around it are saying. Stack a few such layers and the information spreads outward node by node, so that after ℓ rounds a node has been shaped by everything within ℓ steps of it on the graph.

Because the aggregation ignores the order of the neighbours, a GNN naturally respects the irregular, unstructured layout of a real mesh, and because the same learned weights are reused at every node, it can in principle be applied to graphs of different sizes and shapes than the ones it trained on. The particular variant used in this paper, its mean-based aggregation, and the way the graph is built around the room’s geometry are detailed in the methodology.

3

Related Work

This chapter reviews the prior work this paper builds on and sets itself against. It works from the general toward the specific: first the broad roles machine learning has played in computational fluid dynamics, then the narrower problem of surrogate models for indoor airflow, then the two architecture families this study puts at its center, physics-informed neural networks and graph neural networks, and finally the work that has tried to bring such models into architectural design tools. The chapter ends by drawing together the gaps in this body of work and stating where the present paper sits within it.

3.1 Machine Learning for CFD: A Historical Overview

Machine learning has been brought into computational fluid dynamics in more than one role, and it is worth separating them before narrowing to the one this paper works in. Caron, Lauret, and Bastide (2025), reviewing the field for built-environment applications, group the work into three broad uses: speeding up direct numerical simulation, improving the turbulence models that close the averaged equations, and building reduced-order or surrogate models that stand in for the solver. The first two keep a conventional solver and use learning to make some part of it cheaper or more accurate. The third replaces the solver outright with a learned mapping, and that is the role this paper belongs to.

The surrogate idea, in its plainest form, is to train a model to reproduce the input-output behaviour of a CFD run and then call the model instead of the solver. Caron, Lauret, and Bastide (2025) note that this works well enough in narrow, specific applications but tends to generalize poorly once the model is asked about conditions it did not see during training. That weakness is the reason the field has moved the way it has. Rather than give up on surrogates, researchers have shifted toward models that carry some physics with them, constraining or guiding the network with the governing equations so that its predictions stay physically consistent. This move from purely data-driven surrogates toward physics-aware ones is the trend that most shapes the methods compared in this paper.

Within the built environment specifically, Mao et al. (2025) sort surrogate models into three types: reduced-order models, direct surrogate models, and the two combined. A reduced-order model first compresses the high-dimensional flow field down to a handful of dominant modes and learns the behaviour in that smaller space. A direct surrogate model skips the compression and learns the full mapping from design parameters to flow field in one step. The combined approach uses a reduced-order representation as the target that a direct surrogate is trained to predict. Whichever form is used, the payoff is similar: once trained, these models reproduce the output of an expensive simulation at a small fraction of the cost, often several orders of magnitude faster (Mao et al. 2025). The three architectures compared later in this paper are all direct surrogate models in this sense. Each learns the parameter-to-field mapping directly, with no intermediate reduced basis.

A more recent line of work reframes the problem again. Operator networks such as the deep operator network (Lu et al. 2021) and the Fourier neural operator (Z. Li et al. 2021) do not learn a mapping for one fixed grid. They learn the operator itself, the mapping between whole function spaces, which makes their predictions in principle independent of the resolution they were trained at. Caron, Lauret, and Bastide (2025) report that these architectures are among the strongest performers for real-time inference, while also running into real limits on large, tightly coupled multiphysics problems. Operator learning is the immediate backdrop for the indoor-airflow surrogates taken up in the next section, where one of the closest precedents to this paper uses exactly these tools.

3.2 Surrogate Models for Indoor Airflow

Inside a room, what a surrogate has to learn is the mapping from a set of design or operating parameters to the resulting flow field, which in practice means the velocity, the pressure, and the turbulence quantities at every point in the space (Mao et al. 2025). A fair amount of work has gone into this over the past decade using a range of methods, and it is worth walking through it roughly in order of how much physics each approach carries.

The earlier efforts were data-driven through and through. Surveying the area, Mao et al. (2025) describe a back-propagation network from Tian and colleagues that predicted energy use, thermal comfort, and air quality in thermally stratified rooms, and a reduced-order model from Luo and colleagues that paired proper orthogonal decomposition with a radial-basis network to reconstruct full three-dimensional temperature and velocity fields in non-uniform indoor environments at high speed.

A later hybrid from Wang and colleagues combined a convolutional and a long short-term memory network, trained on CFD data, to predict thermal comfort in crowded spaces, with good accuracy on dynamic quantities such as local air velocity and temperature. These studies established that a learned model can stand in for an indoor CFD run and return useful environmental quantities quickly. What they share is that the model learns from data alone, with nothing in it that knows the flow has to obey any physical law.

The two studies closest to what this paper does take that limitation as their starting point, and they answer it in opposite ways. H. Gao et al. (2024) framed indoor airflow prediction as an operator-learning task, using an improved DeepONet together with a Fourier neural operator to learn the mapping from a flow case to its solution field. Their trained models returned a field in under 0.01 seconds, somewhere between three and six orders of magnitude faster than the CFD they replaced, and they did it with a fairly small training set while still predicting reasonably on a domain they had not trained on. The strength of the operator approach is that it learns a mapping between function spaces rather than on a single fixed grid, so the prediction does not in principle depend on the resolution it was trained at.

Wei and Ooka (2023) went the other way. Rather than rely on data alone, they built the governing equations into the network as a physics-informed model and used it to reconstruct a two-dimensional airflow field from only a handful of point measurements. The comparison they ran is the one that matters here: a plain artificial neural network given the same sparse data returned unphysical fields, while the physics-informed model did not, and the reconstruction cut the computation against a conventional CFD run by roughly 40%. Their result is the clearest single piece of evidence that, for indoor airflow specifically, building in the physics pays off most where the data runs thin.

Put together, this body of work points to a consistent lesson and an equally consistent gap. The lesson is that a purely data-driven surrogate is fast but goes unphysical when the data is sparse, and that adding physics is the most reliable fix (Mao et al. 2025). The gap is that every one of these studies commits to a single kind of model from the outset, whether an operator network, a reduced-order model, or a physics-informed network, and reports it against CFD on its own terms. None of them puts a data-driven network, a physics-informed network, and a graph network on the same indoor problem, with the same data and the same test, to see which of the three makes the best use of what it is given. That comparison is what this paper sets out to provide.

3.3 Physics-Informed Neural Networks in Fluid Dynamics

The physics-informed neural network was introduced by Raissi, Perdikaris, and George E. Karniadakis (2019) as a way to solve and reconstruct the solutions of partial differential equations by writing the equations themselves into the training loss; the form that loss takes was set out in Chapter 2. What made the idea spread so quickly through fluid dynamics is the combination it offers: a model that can produce a flow field while respecting the governing equations, and that needs far less data than a network trained on examples alone (George Em Karniadakis et al. 2021). PINNs have since been applied across a wide range of flow problems, and it is worth looking at how that work has gone, both where it has succeeded and where it has struggled.

The motivation is easiest to see by contrast with the data-only approach. Zhao et al. (2024) point out that a model trained on data alone is only as good as that data: it overfits when examples are scarce or hard to obtain, and as a black box it tends to miss the mathematical structure of a PDE solution and has no way to take in physical knowledge it was not shown. A PINN does not have these problems in the same way. Because it carries the equations, it adapts well to exactly the awkward conditions that trouble a data-driven model, including sparse or noisy measurements and boundary conditions that change over time (Zhao et al. 2024). The physics acts as a standing constraint that holds the prediction in place where the data thins out.

Most early PINN work in fluids stayed with laminar or canonical flows, where the equations are the full incompressible Navier-Stokes system with no turbulence model attached. A smaller body of work has pushed into turbulence by way of the Reynolds-averaged equations, which is the regime this paper operates in. Eivazi et al. (2022) applied PINNs directly to the Reynolds-averaged Navier-Stokes equations, showing that the averaged turbulent equations, and not only their laminar counterparts, can be solved within the physics-informed framework. Pioch et al. (2023) went a step further and compared several RANS turbulence-model formulations inside a PINN on the backward-facing step, a standard separated-flow benchmark, and found that the choice of closure has a real effect on how well the network does. These two studies matter here because the training data for this paper come from a $k-\varepsilon$ RANS solver, so the physics-informed model is being asked to learn a RANS solution, the same kind of averaged field these papers work with.

None of this should make PINNs sound easy to train, because they are not. Botarelli et al. (2025) make the point plainly in their review: most published PINN work has stayed on PDE and fluid-dynamics benchmarks.

Many of those studies fed the network numerical data drawn from a simulation, which weakens any claim that the PINN is replacing a solver rather than imitating one, and very few report how execution time trades off against accuracy. The training itself can fail outright. They point to work by Chuang and Barba that documented failure cases on the Taylor-Green vortex and on flow past a cylinder when the network was trained with no data at all, and to the catalogue of training pitfalls and best practices assembled by S. Wang, Sankaran, et al. (2023) for getting PINNs to converge on a range of equations. It is a promising method that is also hard to train reliably, and the reviews are candid about it.

Two things about this paper follow from that picture. First, the failure cases cluster in the data-free setting, where the network has nothing but the equations and a set of collocation points to learn from. This paper does not work there. Its PINN is trained against CFD data as well as the physics, a hybrid that leans on whichever resource is more reliable at each point and is far less exposed to the failure modes that catch a purely physics-driven model. Second, the test here is an applied indoor-airflow scenario rather than a clean benchmark, which is the kind of real-world assessment Botarelli et al. (2025) say the literature is still short of.

3.4 Graph Neural Networks for Field Prediction

A graph neural network earns its place in this comparison for a structural reason. A CFD mesh is already a graph: the cells are nodes and the faces between them are edges, so the connectivity a solver works on maps onto a GNN with no translation (Caron, Lauret, and Bastide 2025). Chapter 2 set out how message passing turns that connectivity into a prediction, with each node updating its state from its neighbours. The promise of the approach is that, unlike a network that treats each point in isolation, a GNN is told from the start which points sit next to which.

The idea that a graph network can stand in for a physics simulator was established outside the built environment. Sanchez-Gonzalez et al. (2020) trained a graph network to simulate complex physical systems, representing the material as a set of interacting particles and learning the dynamics from data, and showed it could reproduce fluids, sand, and deformable solids. Pfaff et al. (2021) carried this onto unstructured meshes, predicting physical fields directly on the mesh and generalizing to mesh sizes and shapes the model had not seen during training. Both build on the general graph-network framework laid out by Battaglia et al. (2018). Taken together, this line of work showed that a learned graph simulator can be both accurate and transferable, which is the property a CFD surrogate most needs.

The work closest to this paper brings the graph network into building-related flow. Shao et al. (2023) proposed a physics-informed graph neural network, combining the graph structure with a PINN-style loss, to predict urban wind fields on unstructured grids. By embedding the Reynolds-averaged equations into the network, the model reached accuracy comparable to CFD while running 10 to 100 times faster, and its authors argued that the physical constraint is what fixes the poor generalization of a purely data-driven graph model (Mao et al. 2025). A related two-stage approach reported by Mao et al. (2025) coupled a steady RANS solve with a graph network, using a coarse-grid CFD field to warm-start the prediction, and reported a computational efficiency around 280 times that of a large-eddy simulation at comparable accuracy. The authors of the urban-wind model were careful to note that it still needed testing on more complex scenarios (Caron, Lauret, and Bastide 2025).

The appeal of the graph approach is a structural one: adjacency is built into the model rather than learned, and because the same weights are reused at every node it can in principle be applied to geometries of a different size or shape than it trained on. What the precedents do not offer is a test of that promise against the alternatives on an indoor problem. The foundational simulators work on generic physics, Shao's model is outdoor urban wind with no comfort calculation attached, and each is evaluated on its own rather than set beside a data-driven and a physics-informed network. The graph network in this paper is built to fill exactly that slot, as the third member of a comparison on a single indoor-airflow problem.

3.5 Integration with Architectural Design Workflows

A surrogate is only useful to an architect if it reaches the tools an architect actually works in. Most contemporary practice runs through parametric modelers, with Rhinoceros and its Grasshopper editor the common case, and a long line of design and simulation studies has been built on top of them. Rodrigues Marques Sakiyama et al. (2021) note that despite this, there is still a real disconnect between the parametric design platform and building-performance simulation, and that the disconnect is widest for natural ventilation, where the airflow analysis tends to sit outside the modeling environment rather than inside it.

Their own response was to build the connection by hand. Rodrigues Marques Sakiyama et al. (2021) assembled a workflow inside Rhino and Grasshopper that used Butterfly, a Ladybug tool, to set up and run CFD simulations through OpenFOAM, and then post-processed and visualized the airflow results with Grasshopper's own components. The geometry stayed parametric, the simulation was automated, and the results came back into the same environment the designer was working in.

What sat under the workflow, though, was still a full OpenFOAM solve, so the feedback ran at the speed of conventional CFD rather than the speed of design iteration. Chung et al. (2023) brought performance analysis into design from a different direction. Their workflow generated a varied set of plausible floor plans with the Wave Function Collapse algorithm, then evaluated them with post-pandemic occupant-presence models and adaptive thermal-comfort models, testing multi-mode cooling strategies that mix conventional air conditioning with natural ventilation. The finding was that the mixed strategies could cut energy use noticeably while giving up only a little comfort. Here comfort and occupancy are genuinely in the loop, but the analysis rests on building-performance simulation rather than a learned surrogate, and the comfort side uses an adaptive model rather than a field-based calculation of the kind this paper attaches downstream.

Looking ahead, Mao et al. (2025) expect machine-learning surrogates to sharpen both early-stage and detailed design work by giving fast feedback and supporting the optimization of ventilation, energy, and comfort together. The gap between that expectation and the integrations just described is the one this paper works in. Both Sakiyama and Chung wire a conventional engine into the design tool, and a conventional engine, however well connected, still answers at its own pace. None of these workflows drops a fast neural surrogate into the parametric environment with a comfort calculation attached to its output.

3.6 Research Gap and Positioning

The work surveyed so far has carried the field a long way, but read together it leaves four gaps that this paper is built to close.

The clearest is the absence of a like-for-like comparison. The indoor-airflow surrogates of the last decade each report a single architecture against CFD: an operator network in one study, a reduced-order model in another, a physics-informed network in a third (H. Gao et al. 2024; Wei and Ooka 2023). No published study has yet set a data-driven network, a physics-informed network, and a graph network on the same problem, with the same dataset and the same evaluation protocol, so that their differences can be read directly rather than pieced together across papers that each used different data and different tests. The comparison built in Chapter 4 and reported in Chapter 5 is meant to fill this gap.

A second gap runs through how generalization is tested. Most surrogate studies report a single held-out test set, which mixes two very different things: filling gaps inside the range the model was trained on, and reaching outside that range altogether.

The literature is clear that the second is where data-driven models are weakest (Mao et al. 2025; Chen et al. 2025), yet a single split cannot separate the two, and it usually averages over all the input parameters at once, so it cannot say which parameter the model struggles with. This paper splits the evaluation into distinct interpolation and extrapolation regimes and further separates the velocity parameters from the thermal ones, so the weakness of each model can be located rather than just measured. The framework that does this is set out in Chapter 4.

The third gap is about what the training data costs. Every training sample is a full CFD run, so the number of samples a surrogate needs is a direct expense, and a designer deciding whether to build one wants to know how accuracy trades against that expense. The closest precedent reports this for a single architecture, finding that accuracy saturates once enough measurements are supplied (Wei and Ooka 2023), but no published study maps the trade for several architectures together. This paper trains all three models at three dataset sizes, so a data-efficiency curve can be drawn for each and compared. These results appear in Chapter 5.

The last gap is the missing end of the pipeline. The design-workflow integrations bring conventional solvers into parametric tools, and some of them include comfort, but none couples a fast neural surrogate to a validated comfort calculation inside the design environment. A predicted velocity and temperature field is only halfway to what a designer needs; the real question is whether the room is comfortable, which means carrying the field through the ISO 7730 measures. This paper closes that end with a downstream comfort calculation and a prototype Grasshopper plugin, and the comfort calculation itself is checked against the reference cases of the standard in an appendix.

Set against all of this, the position of this paper is straightforward. It is, as far as the published literature shows, a systematic comparison of a data-driven, a physics-informed, and a graph neural network as surrogates for indoor airflow, with interpolation and extrapolation separated by parameter axis, a data-efficiency study across three dataset sizes, a downstream ISO 7730 comfort calculation, and an architect-facing parametric-design integration. The next section sets the same positioning against the closest comparator studies in a single table.

3.7 Literature Matrix

Table 3.1 collects the comparison drawn out in the preceding sections, setting the closest prior studies against the axes this paper works on. Read down the capability columns and no single study spans them: each prior work covers one or two and leaves the rest aside, which is the gap stated above shown in one view.

Table 3.1: The closest prior studies set against the axes this paper addresses. *Architectures compared* counts distinct neural-network families evaluated on one problem; *Generalization tested* names the held-out regime each study used. Chung reports comfort with an adaptive model rather than a field-based ISO 7730 calculation, and both Sakiyama and Chung couple conventional solvers to the design tool rather than a neural surrogate.

Study	Method	Domain	Architectures compared	Generalization tested	Data efficiency	Comfort coupling	Design tool
H. Gao et al. (2024)	DeepONet + FNO	Indoor airflow	One operator family	Unseen domain, single split	Small-set emphasis	Not addressed	Not addressed
Wei and Ooka (2023)	PINN vs ANN	Indoor airflow	Two (PINN, ANN)	Single test set	Sparse-data only	Not addressed	Not addressed
Son, Kim, and Koo (2024)	PINN	Ventilation rates	One	Single setting	Not addressed	Not addressed	Not addressed
Eivazi et al. (2022)	PINN	RANS flow (generic)	One	Single benchmark	Not addressed	Outside scope	Outside scope
Pioch et al. (2023)	PINN (RANS closures)	Step-flow benchmark	One (closures compared)	Single benchmark	Not addressed	Outside scope	Outside scope
Shao et al. (2023)	PIGNN (PINN + GNN)	Urban wind	One hybrid	Single grid	Not addressed	Not addressed	Not addressed
Pfaff et al. (2021)	GNN (MeshGraphNets)	Mesh physics (generic)	One	Across mesh size, shape	Not addressed	Outside scope	Outside scope
Sanchez-Gonzalez et al. (2020)	GNN (GNS)	Particle physics (generic)	One	Scale-up, larger domains	Not addressed	Outside scope	Outside scope
Rodrigues Marques Sakiyama et al. (2021)	CFD (OpenFOAM)	Indoor, NV	n/a (not a surrogate)	n/a	Outside scope	Not addressed	Grasshopper + OpenFOAM
Chung et al. (2023)	BPS + generative	Building, NV	n/a (not a surrogate)	n/a	Outside scope	Adaptive comfort	Generative GH workflow

4

Methodology

This chapter sets out how the study was built and run, from the CFD data underneath to the trained surrogates and the comfort predictions on top. It works from the bottom up: how the reference CFD dataset is generated in COMSOL, how the parameter sweep is designed and sampled, the four room geometries that make up the robustness study, the three surrogate architectures and how each one is trained, the downstream comfort calculation, and the framework used to evaluate all of it.

4.1 Pipeline Overview

The work runs in four stages, shown in Figure 4.1. The first generates the data, the second trains the models on it, the third evaluates them, and the fourth puts a trained model to work inside a design tool. Each stage feeds the next, and the whole chain is built once and then re-run, unchanged, for each of the four room geometries.

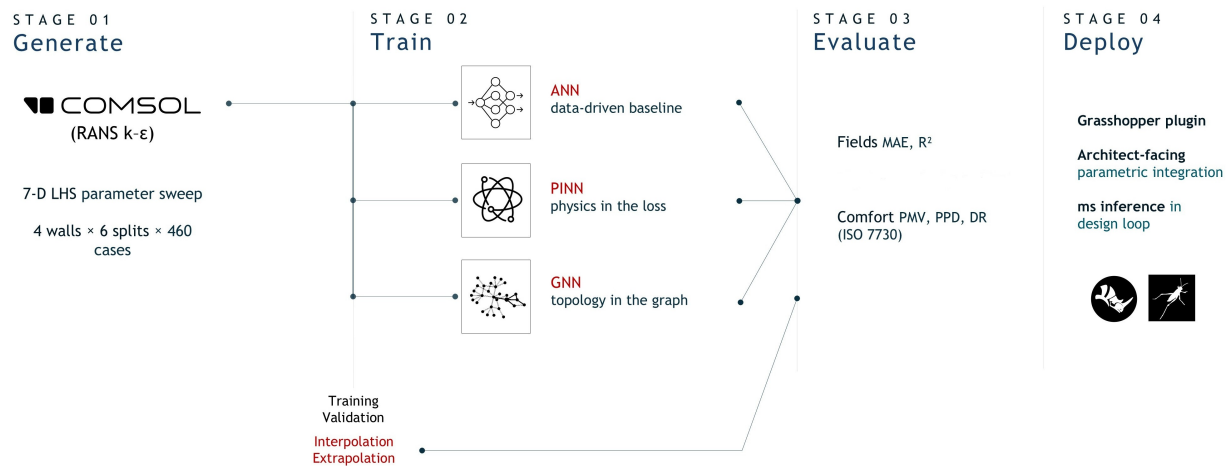


Figure 4.1: The four-stage pipeline of this paper. A COMSOL RANS $k-\epsilon$ sweep generates the reference fields; three surrogate architectures are trained on them; each is evaluated for field accuracy, generalisation, and downstream comfort; and a trained surrogate is deployed inside a parametric design tool.

The first stage produces the reference data. A COMSOL solve of the steady RANS equations with the $k-\varepsilon$ closure (Section 4.2) is the ground truth the surrogates learn to imitate. Throughout this paper the surrogates are assessed against this CFD reference rather than against physical measurements, so accuracy here means agreement with the COMSOL $k-\varepsilon$ solution; experimental validation of the reference itself is left as a limitation. Rather than hand-pick a handful of cases, the study sweeps a seven-dimensional parameter space with Latin Hypercube Sampling (Section 4.3.1) and keeps 460 successfully-solved cases for each room geometry. Those cases are split six ways: a training set, a validation set, and four held-out test sets that separate interpolation from extrapolation (Section 4.3.5). The data design itself therefore encodes the generalisation question the paper is built around, before any model is trained.

The second stage trains three surrogates on that data. The first is a plain data-driven network that serves as the baseline; the second is a physics-informed network that carries the governing equations in its loss; the third is a graph network that carries the room’s spatial layout in its connectivity (Section 4.5). The third stage then evaluates the three on the held-out splits along three lines at once: how accurately each reproduces the CFD fields, how well each generalises to held-out interpolation and extrapolation cases, and how faithfully the predicted fields carry through into the ISO 7730 comfort measures. The interpolation and extrapolation splits are the instrument for the central comparison, since they are where a purely data-driven model and a physics- or structure-informed one are most likely to part ways.

The fourth stage is the practical payoff. A trained surrogate returns a full field far faster than the roughly 138 seconds a fresh CFD case takes, which is fast enough to sit inside the loop a designer actually works in. The rest of this chapter describes each stage in turn.

4.2 CFD Dataset Generation

Everything downstream rests on the reference data, so it is worth being precise about how it is produced. A surrogate can only ever be as good as the simulations it learns from, and any error the CFD carries is inherited by all three models alike. This section describes the solver, the physics it solves, the room and its boundary conditions, the parameters that vary from case to case, and the mesh.

4.2.1 Simulation Environment: COMSOL Multiphysics

Every CFD case in this paper was solved in COMSOL Multiphysics 6.3 (COMSOL AB 2026), using the CFD and Heat Transfer modules. The model is two-dimensional and steady. Two physics interfaces run together: a turbulent $k-\varepsilon$ flow solve, and a heat-transfer solve that covers both the air and the surrounding concrete envelope. A nonisothermal-flow coupling node links them, for reasons set out in Section 4.2.2.

The solve is segregated rather than fully coupled. It runs in two steps per outer iteration: the first solves the nonisothermal flow block (the velocity components, pressure, and temperature together) with a damping factor of 0.5, and the second solves the two turbulence variables k and ε with a damping factor of 0.45 and three inner iterations. Both steps use PARDISO as the direct linear solver, and pseudo-time stepping with an Anderson-acceleration switch stabilises the march toward the steady state. Lower limits hold k , ε , and T at or above zero so the solution never drifts into physically impossible values, and the solve is capped at 300 outer iterations. On the workstation used here a single case takes around 138 seconds.

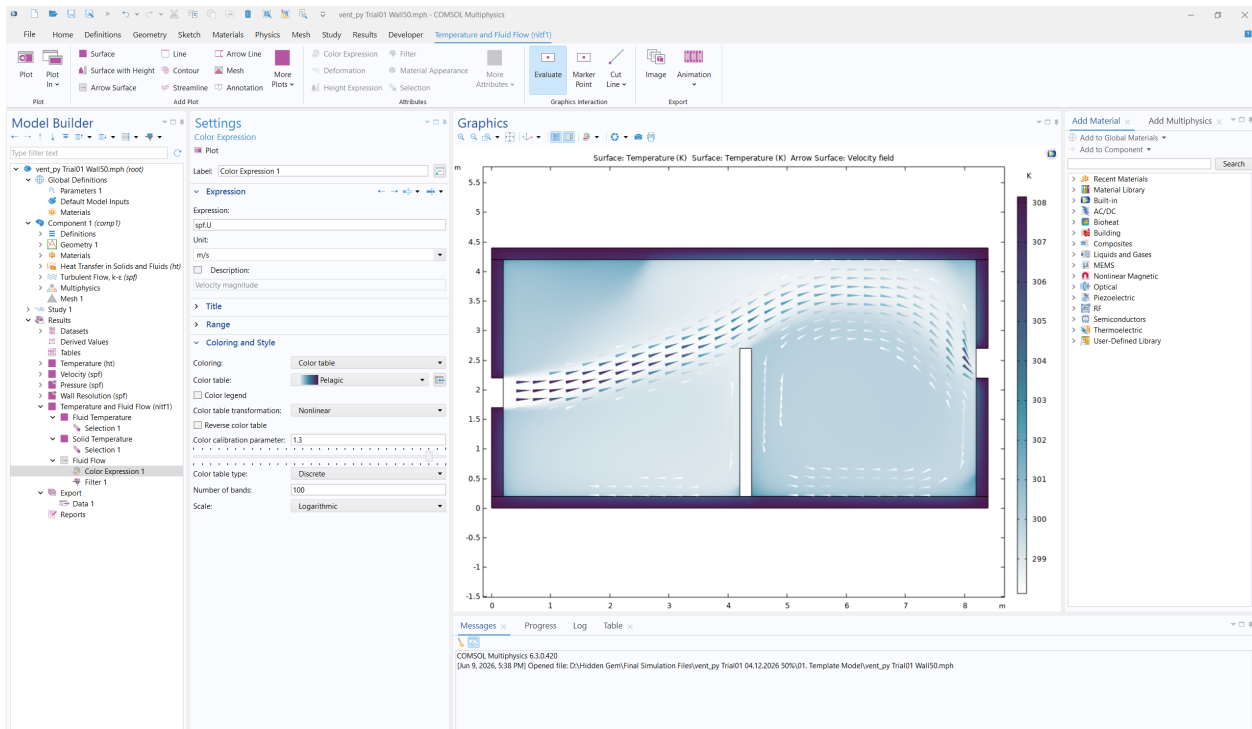


Figure 4.2: A representative COMSOL solution for one case of the ventilated room: the steady velocity and temperature fields the sweep produces. These are the reference fields each surrogate is trained to reproduce.

Solving 460 cases by hand was never going to work, so the sweep was scripted end to end through the open-source `mph` library (Hennig 2024), which exposes COMSOL’s Java API to Python. A reference `.mph` file holds the full setup, geometry through solver, plus a named export node, and the script loads it once into a persistent session. For each case it overrides the seven swept parameters, re-runs the stationary study with the existing solver configuration, and exports the solution onto a uniform grid as a CSV. Because every case starts from the same template and only the seven numbers change, the runs stay directly comparable. Geometry relationships, material models, mesh strategy, solver settings, and export grid are all identical across the sweep, so nothing varies between cases except what is supposed to.

4.2.2 Governing Physics: Coupled Turbulent Flow and Heat Transfer

A non-isothermal ventilation flow cannot be solved by computing the velocity field first and the temperature field afterwards, because the two are bound together at several points and each depends on the other. The clearest binding is through density. Air is treated as an ideal gas whose density depends on temperature, $\rho = p_A M / (RT)$ with molar mass $M = 0.02897$ kg/mol, so a difference of up to 20 K between the inlet stream and the exterior walls produces a several-percent variation in density across the room. The continuity equation the solver enforces is the weakly-compressible form $\nabla \cdot (\rho \mathbf{u}) = 0$, which means the velocity field cannot be found without the temperature field that sets ρ . The transport properties – viscosity, specific heat, thermal conductivity – are likewise temperature-dependent and are re-evaluated from the current temperature at every outer iteration.

Gravity is deliberately switched off in this study, so the coupling does not act through a buoyancy body force; it acts entirely through the equation of state and the temperature-dependent properties just described. At the forced-flow inlet velocities used here, 0.30 to 3.00 m/s, the inlet momentum dominates any thermal plume, so a forced-convection assumption is consistent with the operating regime. Re-enabling gravity to reach the mixed- and natural-convection regimes is left to future work.

A third coupling runs through the turbulence model. Turbulent heat transport is carried by an eddy thermal conductivity $k_t = \mu_t c_p / \text{Pr}_t$, where μ_t comes from the k - ε model and the turbulent Prandtl number Pr_t is itself flow-dependent. COMSOL closes this with the Kays-Crawford model (Kays 1994; Kays and Crawford 1993; Weigand, Ferguson, and Crawford 1997), in which Pr_t is a function of the molecular Prandtl number and a turbulent Peclet number, approaching about 0.85 for air far from walls. Evaluating it needs the turbulent-viscosity field from the flow side and the temperature field from the heat side at the same time. The fourth coupling is at the walls: the thermal wall function computes a wall heat flux from the resolved wall temperature, the friction velocity, and Pr_t , and feeds it back into the energy equation, while the momentum wall function depends on a viscosity that is itself temperature-dependent.

For these reasons the model uses COMSOL's Nonisothermal Flow coupling node, which ties the k - ε flow interface to the heat interface and shares the density, the reference temperature, the velocity, the absolute pressure, and the turbulent heat-transport closure between them. Viscous dissipation and the kinetic-energy contribution to the energy equation are left out; both are small next to advective transport at these velocities. Solving the two interfaces together rather than in sequence is the standard treatment in the indoor-airflow literature whenever wall and inlet temperatures differ by more than a few kelvin (Stamou and Katsiris 2006; Sevilgen and Kilic 2011; Koskela et al. 2001; Cheng, Niu, and N. Gao 2012).

The turbulence closure is the standard two-equation $k-\varepsilon$ model with wall functions, chosen over the shear-stress-transport (SST) alternative for two reasons. First, $k-\varepsilon$ with wall functions is the long-standing default for indoor airflow CFD (Stamou and Katsiris 2006; Sevilgen and Kilic 2011; Koskela et al. 2001; Cheng, Niu, and N. Gao 2012), which keeps the reference data comparable with the comfort literature this paper draws on. Second, wall functions bridge the viscous sublayer instead of resolving it, so the near-wall mesh can be far coarser than SST’s wall-resolved formulation needs; with 460 cases to solve per geometry, that saving is the difference between a tractable sweep and an impractical one. The cost is that $k-\varepsilon$ under-predicts separation under strong adverse pressure gradients and is less accurate very close to the wall. That trade is acceptable here for a specific reason: the surrogate is trained against the $k-\varepsilon$ solutions themselves, so it inherits whatever error the closure carries, and that error is identical across the ANN, PINN, and GNN. It cannot tilt the architecture comparison, which is the actual question. Gao and colleagues, by contrast, chose SST for confined indoor flow; the choice here is governed by the sweep budget and by comparability rather than by single-case fidelity, and a move to SST or a hybrid RANS/LES closure is noted as future work.

The flow interface solves the steady, weakly-compressible RANS equations with the $k-\varepsilon$ closure (Launder and Spalding 1974). In index notation, with summation over the repeated index j across the two Cartesian directions,

$$\rho u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left[(\mu + \mu_t) \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right], \quad (4.1)$$

$$\frac{\partial(\rho u_i)}{\partial x_i} = 0. \quad (4.2)$$

Equation (4.1) is the steady RANS form of the momentum balance introduced in Chapter 2. The left-hand side is advection of momentum by the mean flow; the right-hand side balances it against the pressure gradient and the divergence of the total stress, in which the molecular viscosity μ is augmented by an eddy viscosity μ_t that stands in for the averaged-out turbulence. Equation (4.2) is the weakly-compressible continuity statement $\nabla \cdot (\rho \mathbf{u}) = 0$ rather than the constant-density $\partial u_i / \partial x_i = 0$ of Chapter 2, precisely because ρ varies with the temperature field, as described above.

The turbulence is carried by two transport equations, one for the turbulent kinetic energy k and one for its dissipation rate ε , both introduced in Chapter 2:

$$\rho u_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] + P_k - \rho \varepsilon, \quad (4.3)$$

$$\rho u_j \frac{\partial \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] + C_{\varepsilon 1} \frac{\varepsilon}{k} P_k - C_{\varepsilon 2} \rho \frac{\varepsilon^2}{k}. \quad (4.4)$$

Each balances advection of k (or ε) by the mean flow against diffusion, a production term P_k that draws energy out of the mean shear, and a destruction term. The two equations are joined to the momentum equation through the eddy viscosity,

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}, \quad (4.5)$$

the same Boussinesq closure given in Chapter 2, with the standard constants $C_\mu = 0.09$, $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$, $\sigma_k = 1.00$, and $\sigma_\varepsilon = 1.30$ (Launder and Spalding 1974; Pope 2000; Wilcox 2006). The wall treatment uses standard logarithmic wall functions, which assume a log-law velocity profile in the near-wall cell rather than resolving the sublayer; this is what the mesh of Section 4.2.5 is built to support.

The heat interface solves the energy equation in both the air and the concrete envelope. In the fluid, advection of heat by the mean flow is balanced against conduction and turbulent thermal diffusion,

$$\rho c_p u_j \frac{\partial T}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\left(k_{\text{air}} + \frac{\mu_t c_p}{\text{Pr}_t} \right) \frac{\partial T}{\partial x_j} \right], \quad (4.6)$$

where the turbulent contribution enters through the eddy thermal conductivity $\mu_t c_p / \text{Pr}_t$ with the Kays-Crawford Pr_t . In the solid envelope there is no flow, so the equation reduces to steady conduction,

$$0 = \frac{\partial}{\partial x_j} \left(k_{\text{concrete}} \frac{\partial T}{\partial x_j} \right), \quad (4.7)$$

with the temperature and the heat flux matched at the air-concrete interface. The concrete properties are held constant: density 2300 kg/m³, conductivity 1.8 W/(m K), and specific heat 880 J/(kg K), all from the COMSOL material library.

4.2.3 Geometry and Boundary Conditions

The computational domain is a two-dimensional rectangular room of interior dimensions $L_{\text{room}} = 8.0$ m by $H_{\text{room}} = 4.0$ m, wrapped in a concrete envelope 0.20 m thick on all four sides. A single interior partition wall, 0.20 m thick and 2.5 m tall, stands at the quarter, half, or three-quarter point of the room depending on the configuration (Section 4.4); the 2.5 m height leaves an air gap between the top of the partition and the 4 m ceiling. All of these pieces — floor, ceiling, the four wall segments around the openings, and the partition — are combined into one connected geometry through a Form Union operation. Figure 4.4 shows the layout.

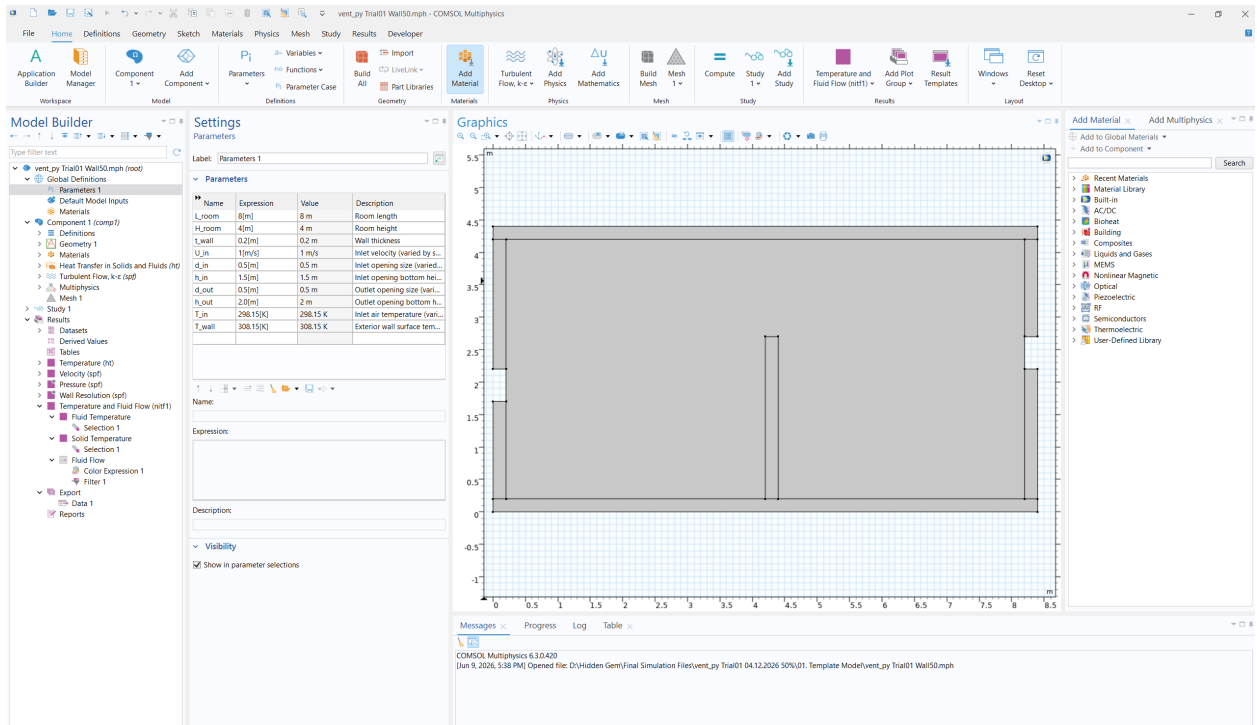


Figure 4.3: The room geometry as built in COMSOL: the rectangular domain, the concrete envelope, the interior partition, and the inlet and outlet openings.

The inlet opening sits on the left exterior wall at height h_{in} from the floor, with vertical extent d_{in} ; the outlet sits on the right wall at height h_{out} with extent d_{out} . The flow boundary conditions are straightforward. The inlet carries a velocity condition with normal speed U_{in} , and its turbulence is specified through a turbulent intensity of 0.05 — the low end of the fully-turbulent range the COMSOL documentation recommends for ventilation inlets — and a turbulent length scale of $0.07 d_{\text{in}}$, the standard fully-developed pipe-flow correlation also used in the ANSYS Fluent theory guide (ANSYS, Inc. 2024) and widely adopted in indoor-airflow CFD when inlet turbulence has not been measured. The outlet carries a pressure condition at 0 Pa gauge, and every solid surface carries a no-slip condition.

The thermal boundary conditions set up the energy balance the coupled solve resolves. The whole domain is seeded at 293.15 K. The inlet air enters at a fixed temperature T_{in} , the outlet uses a convective-outflow condition that lets the flow carry heat out without imposing a gradient, and the outward-facing surfaces of the concrete envelope are held at a fixed exterior temperature T_{wall} . Every interior surface not otherwise assigned is treated as adiabatic; in particular the interior partition conducts no heat, the usual assumption for a partition separating two rooms held at the same envelope temperature. The combination of a cool inlet stream (typically 25 °C) and a warm envelope (typically 35 °C, a sun-loaded wall in summer) is what drives the non-trivial temperature field across the room and makes the coupled solve of Section 4.2.2 necessary.

4.2.4 Parameter Space

Each case is fully specified by seven scalar inputs, listed with their training ranges in Table 4.1; Figure 4.4 shows where the geometric parameters sit on the room.

The two opening heights are handled with a little care. Rather than sample h_{in} and h_{out} directly, which could place an opening through the floor or ceiling for some combinations of height and size, the sweep samples two auxiliary unit variables $\alpha_{\text{in}}, \alpha_{\text{out}} \in [0.05, 0.95]$ and maps each to an opening height through a geometry-safe transformation,

$$h_{\text{open}} = m + \alpha (H_{\text{room}} - d_{\text{open}} - 2m), \quad m = 0.20 \text{ m}. \quad (4.8)$$

With the margin m set to 0.20 m, this guarantees for any sampled pair that the opening clears the floor ($h_{\text{open}} \geq m$) and the ceiling ($h_{\text{open}} + d_{\text{open}} \leq H_{\text{room}} - m$), so no candidate case can ever produce an opening that intersects a slab. The map is monotone in α , so an α uniform over its range places the opening centroid uniformly within the admissible band for the sampled size. The eighth dial, the partition position $x_{\text{wall_pos}}$, is fixed within a configuration and changes only between the four geometries of Section 4.4; it is therefore a label on the study rather than a sampled parameter.

Each case is thus fixed by these seven numbers plus the wall position. A surrogate query adds the two spatial coordinates (x, y) of the point being asked about, giving nine input features per query point, and predicts the six output fields there: the two velocity components u and v , the pressure p , the temperature T , and the two turbulence quantities k and ε .

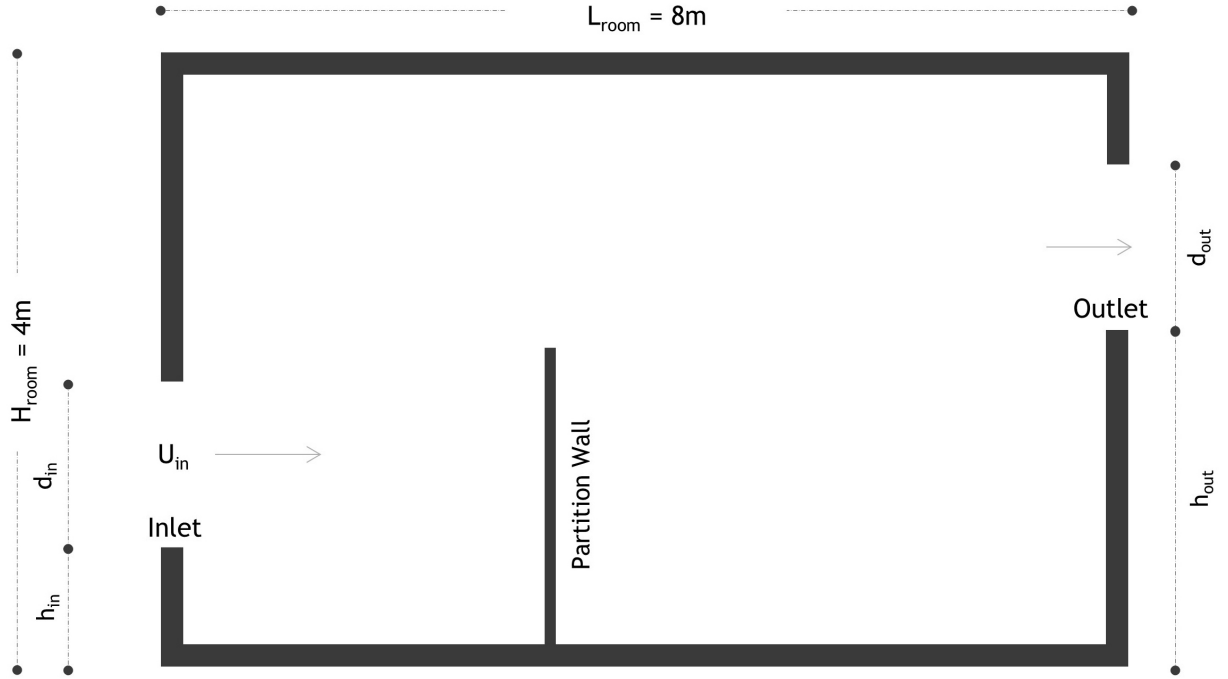


Figure 4.4: The two-dimensional room: interior dimensions, the concrete envelope, the interior partition wall, and the inlet and outlet openings with the geometric parameters that vary from case to case. The partition is shown at the quarter-length position.

Table 4.1: The seven sampled case parameters and their training ranges. The two opening heights are not sampled directly but derived from auxiliary unit variables through the geometry-safe map of Equation 4.8.

Symbol	Physical meaning	Training range
U_{in}	Inlet normal velocity (m/s)	0.30 - 2.00
d_{in}	Inlet opening size (m)	0.10 - 1.20
h_{in}	Inlet height from floor (m)	derived from α_{in}
d_{out}	Outlet opening size (m)	0.10 - 1.20
h_{out}	Outlet height from floor (m)	derived from α_{out}
T_{in}	Inlet air temperature (K)	288 - 308
T_{wall}	Exterior wall surface temperature (K)	293 - 318

The two range gaps used to test interpolation and the bands used to test extrapolation are specified in Section 4.3.2 and Section 4.3.3; all ranges are held identical across the four wall configurations so the geometries can be compared fairly.

4.2.5 Mesh Strategy

The CFD mesh is physics-controlled at the “Finer” element size. COMSOL reads the active interfaces — here the $k-\varepsilon$ flow and the heat transfer — and sizes the boundary-layer and bulk elements to satisfy the turbulence model’s wall-function requirement. For $k-\varepsilon$ with standard wall functions this places the first cell centre in the log-law region, at a dimensionless wall distance of roughly $y^+ = 30$ to 300, which is what the closure expects: it relies on the log-law rather than resolving the viscous sublayer, so a wall-resolved mesh at $y^+ < 1$ would be both unnecessary and far too expensive to repeat across 460 cases.

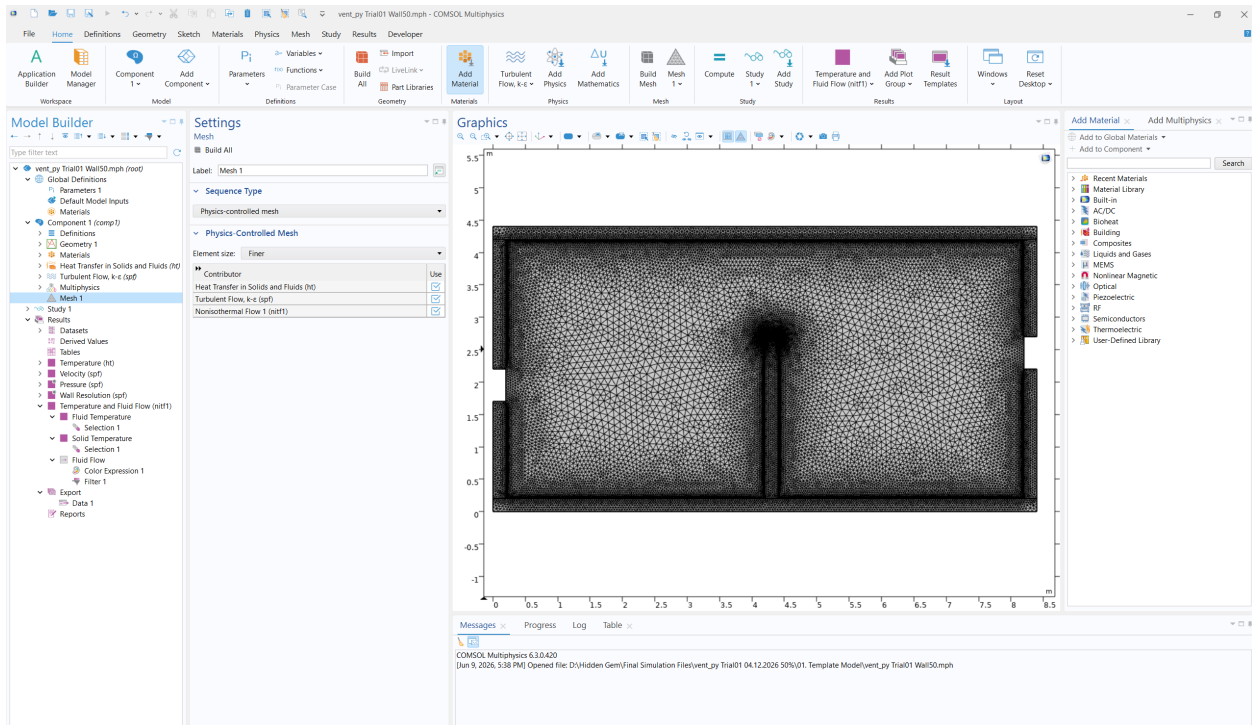


Figure 4.5: The physics-controlled COMSOL mesh at the “Finer” element size, with boundary-layer refinement along the walls.

The post-processed solution is not exported on that triangulation, though. It is sampled onto a separate uniform Cartesian grid that becomes the training grid for the surrogates: 161 points across the room in x and 81 in y , spaced about 0.05 m apart, for 13,041 grid points per case, written to CSV with the points that fall inside the partition wall marked as not-a-number. Keeping the two grids distinct is deliberate. The solver works on its own physics-adapted mesh for accuracy, while the surrogate sees a regular sampling of the result, so that every case yields the same point cloud at the same coordinates. That regularity is what makes batching simple for the ANN and PINN and graph construction simple for the GNN, all described in Section 4.5.

One verification step is deliberately out of scope. The mesh density was fixed at the physics-controlled “Finer” setting across the whole sweep and was not varied to confirm grid independence, so no formal mesh-sensitivity or grid-convergence study was carried out. The fields the surrogates learn are therefore those of this particular mesh resolution; a grid-convergence check on the CFD reference is noted as a limitation and left to future work.

4.3 Automated Sweep Pipeline

The parameter ranges in Section 4.2.4 set the bounds of a seven-dimensional space of rooms and operating conditions. On their own they are only bounds; producing training data means deciding where inside that space to actually run COMSOL, solving each case, and recording it in a form the surrogates can read. Two requirements shaped how this was done. The sampling had to fill the space evenly at a tractable number of cases, since each solve is expensive. And the division between what the surrogates see during training and what they are tested on had to be built into the sampling itself, so that interpolation and extrapolation could be measured separately rather than inferred after the fact. What follows traces the data from those ranges to the 460 solved cases per geometry that the models are trained and evaluated on, and the same pipeline is run, unchanged, for each of the four room geometries.

4.3.1 Latin Hypercube Sampling

The sweep places its cases with Latin Hypercube Sampling (LHS), the stratified space-filling scheme introduced by introduced by McKay, Beckman, and Conover (1979). The problem it solves is specific to sampling a high-dimensional space on a budget. With seven parameters and only a few hundred cases to spend, neither of the obvious alternatives works. A regular grid spends its points wastefully: even a coarse five-level grid is $5^7 = 78,125$ cases, two orders of magnitude past what the COMSOL budget allows, and most of those points would be redundant along any single axis. Drawing the cases independently at random avoids that cost but tends to clump, leaving stretches of some parameters unsampled by chance while doubling up elsewhere. LHS keeps the low cost of random sampling and removes the clumping, by forcing each parameter to be sampled evenly across its range, as Figure 4.6 illustrates.

For N cases in $d = 7$ dimensions, each axis is divided into N equal-width strata, and the sample is built so that exactly one case lands in each stratum on every axis. The implementation generates a unit-cube design first and then scales it to the parameter ranges. The unit-cube coordinate of case i on axis j is

$$x_{ij} = \frac{\sigma_j(i) + u_{ij}}{N}, \quad (4.9)$$

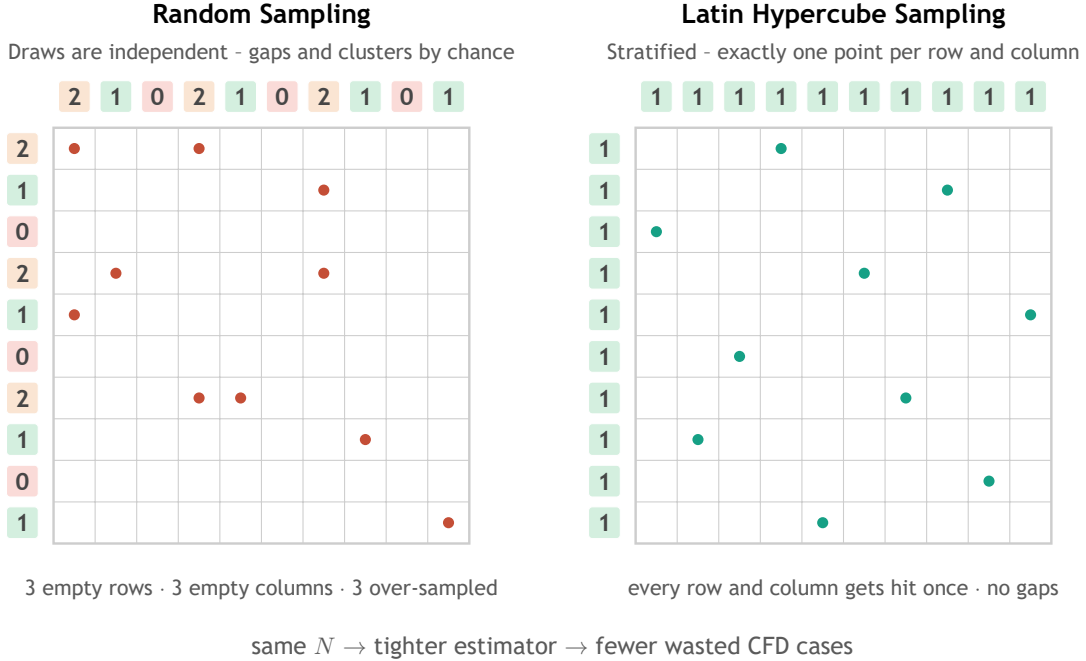
where σ_j is an independent, uniformly random permutation of the stratum indices $\{0, 1, \dots, N - 1\}$ drawn for axis j , and $u_{ij} \sim \mathcal{U}[0, 1)$ is an independent uniform draw. The term $\sigma_j(i)/N$ pins case i to one stratum on axis j ; the independent permutation per axis is what decorrelates the axes, so the strata are not visited in the same order on each one; and the u_{ij}/N term jitters the case to a random position inside its assigned stratum, so the design is a stratified random sample rather than a fixed lattice. Each unit coordinate is then mapped onto its parameter range by the linear scaling

$$p_{ij} = \ell_j + x_{ij}(h_j - \ell_j), \quad (4.10)$$

with ℓ_j and h_j the lower and upper bounds of parameter j from Table 4.1. The result is a set of N cases in which every parameter is represented once in each of its N strata, with no axis left with gaps or pile-ups.

The seven sampled dimensions, in the order they are drawn, are the inlet speed U_{in} , the inlet opening size d_{in} , the inlet placement variable α_{in} , the outlet size d_{out} , the outlet placement variable α_{out} , the supply-air temperature T_{in} , and the wall temperature T_{wall} . The two placement variables are not opening heights themselves. They are the auxiliary unit variables of Section 4.2.4, sampled in $[0.05, 0.95]$ and converted to physical heights h_{in} and h_{out} through the geometry-safe map of Equation 4.8 only after sampling. Sampling in α rather than in height is what lets the design run over a fixed unit interval while still guaranteeing that every opening clears the floor and ceiling.

The case for LHS over plain random sampling is not only the even coverage but a provable reduction in estimator variance. McKay, Beckman, and Conover (1979) show that for an output that is monotone in the inputs, the variance of a mean estimated from an LHS design is no larger than that from a simple random sample of the same size, and is strictly smaller when the output has additive structure in the parameters. The CFD fields here have that structure to a good approximation: U_{in} is the dominant control on velocity magnitude, and the difference $T_{\text{in}} - T_{\text{wall}}$ is the dominant control on the temperature field. An evenly stratified sample of those axes therefore captures the bulk of the field variation at far fewer cases than random placement would need.



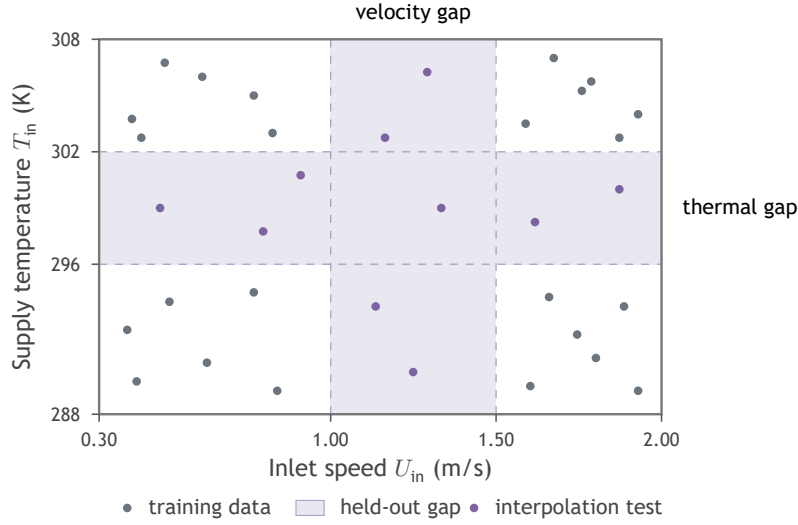


Figure 4.7: The interpolation test design in the inlet-speed-supply-temperature plane; the box marks the training range. Training cases (grey) fill the plane apart from two held-out interior bands – a vertical velocity gap at $U_{in} \in [1.00, 1.50]$ m/s and a horizontal thermal gap at $T_{in} \in [296, 302]$ K. Interpolation test cases (purple) are drawn from inside those bands, surrounded on both sides by training data, so the test measures how well a model fills a gap at parameter values it never trained on. Only one axis is held out at a time: cases in the vertical band vary inlet speed inside the gap while temperature stays in the trained range, and cases in the horizontal band do the reverse.

During candidate generation for the training and validation splits, any case whose inlet speed falls in $[1.00, 1.50]$ m/s, or whose supply temperature falls in $[296, 302]$ K, is rejected before it is ever sent to COMSOL. The two bands sit in the interior of the training ranges $U_{in} \in [0.30, 2.00]$ m/s and $T_{in} \in [288, 308]$ K, so the training and validation sets end up with no support in either band while still spanning the rest of each range. The two interpolation test splits then sample inside those same bands: the velocity-interpolation split draws U_{in} from $[1.00, 1.50]$ m/s, and the thermal-interpolation split draws T_{in} from $[296, 302]$ K. Only one axis is held out at a time. In the velocity-interpolation split the temperature is drawn over its full training range, and in the thermal-interpolation split the speed is, so that each test isolates the model’s behaviour along a single parameter rather than in a joint corner.

Cutting an interior band rather than trimming an edge is what makes this a genuine interpolation test. The held-out cases are convex combinations of training cases, surrounded on both sides by data the model has seen, at parameter values it never encountered. A surrogate that has learned the underlying mapping, rather than memorised the training cases, should fill that gap accurately. A surrogate that has only learned to blend nearby training samples will show its first cracks here, before extrapolation is even tested.

4.3.3 Extrapolation Range Design

Where the interpolation gaps sit inside the training range, the extrapolation bands sit strictly above it. The velocity-extrapolation split draws U_{in} from $[2.10, 3.00]$ m/s against a training maximum of 2.00 m/s, and the thermal-extrapolation split draws T_{in} from $[310, 320]$ K against a training maximum of 308 K. As with the interpolation splits, only one axis is pushed out of range at a time; the other six parameters are sampled over their normal training ranges. Extrapolation here therefore means operating beyond the trained range on the inlet-speed and supply-temperature axes, not transferring to an unseen room shape; generalisation across geometry is a separate question, addressed by the four wall configurations of Section 4.4 rather than by these splits.

The small buffers between the training ceiling and the start of each extrapolation band, 0.10 m/s for velocity and 2 K for temperature, are deliberate. They keep the training and extrapolation distributions from touching even at their shared edge, so that a case labelled extrapolation is unambiguously outside everything the model trained on. The automation script enforces the separation rather than trusting the constants: a preflight check refuses to start if either extrapolation band overlaps the corresponding training range, halting the sweep before any solve when the configuration is inconsistent.

Testing in extrapolation, and not only in the interior gaps, is the empirical heart of the architecture comparison. The hypothesis the paper sets out to test is that physics-informed and graph-structured inductive biases should help a surrogate hold up at operating conditions beyond its training data, exactly where a purely data-driven network is expected to degrade (Ma et al. 2025; Calzolari and Liu 2021). A plain regression model has no reason to behave sensibly outside the span of its data; a model that carries the governing equations in its loss, or the room’s spatial structure in its connectivity, has at least some constraint pulling its predictions toward physically admissible behaviour out there. The two extrapolation splits are where that claim is put to the test, and the results chapter reports how far each architecture actually holds.

4.3.4 Staged Data Generation for the Data-Efficiency Study

Beyond accuracy in and out of distribution, the third research question asks how each architecture’s accuracy scales with the amount of training data, since the CFD data is the expensive part of building a surrogate. This is measured by training every architecture at three training-set sizes: 50 cases, 150 cases, and the full 300. The three sets are nested and deterministic. The 50-case set is the first 50 cases of the 150-case set, which is in turn the first 150 of the full 300, all taken from the single training-split LHS sample of Section 4.3.1.

No extra COMSOL runs are needed for the smaller sizes; they are subsets of data already solved. Nesting the subsets keeps the comparison honest in two ways. Across architectures, the 50-case ANN, PINN, and GNN all see exactly the same 50 cases, so any difference between them at that size is a difference between the models and not between their training samples. Across sizes, moving from 50 to 150 to 300 only ever adds cases and never swaps them, so the data-efficiency curve for a given architecture measures the marginal value of more data rather than the noise of a fresh draw. The validation set and the four test splits are held fixed across all three sizes; only the training set changes.

The framing follows Wei and Ooka (2023), who studied how reconstruction accuracy depends on the amount of training information through an information-saturation ratio, the share of the data needed to capture essentially all the information in the full set. Their finding is the one this study is set up to probe across architectures: as the saturation ratio drops, every method loses accuracy, but the physics-informed network stays more accurate and more robust than the data-driven one once the ratio is above roughly a third. Treating the full 300-case training pool as the saturation reference, the three sizes here, 50, 150, and 300 cases, sit at one-sixth, one-half, and all of that pool, which spans the regime where Wei and Ooka saw the physics-informed advantage open up. Whether that advantage reappears for indoor airflow fields, and whether the graph model shows a different efficiency profile again, is one of the questions the results chapter answers.

4.3.5 Six-Split Evaluation Design

The sampling rules above resolve into six splits per geometry, listed in Table 4.2. Two are for fitting the model, a training set of 300 cases and a validation set of 60, and four are held out for testing: an interpolation and an extrapolation split on each of the velocity and thermal axes. The successful-case targets sum to 460 per room geometry. The training and validation splits are drawn from identical distributions, with the same ranges and the same excluded gaps, and differ only in their random seed. That is what makes the validation set an honest estimate of in-distribution error: it is a fresh draw from precisely the distribution the model was trained on, neither easier nor harder. The four test splits, by contrast, are drawn from distributions the training set was specifically built to avoid, which is what lets them measure generalisation rather than fit. Figure 4.8 shows the design in the plane of the two axes the splits separate, the inlet speed and the supply temperature. The training cases fill the range densely except for the two interior bands left empty by rejection; the interpolation test cases sit inside those bands; and the extrapolation test cases sit beyond the upper edge of the training cloud on each axis. The five other parameters, not shown, are sampled over their full ranges throughout.

Table 4.2: The six evaluation splits sampled for each room geometry. The ranges are the sampling bounds for the inlet speed and the supply temperature; the excluded bands are the interior gaps removed by rejection sampling. The remaining five parameters are sampled over their full ranges in every split.

#	Split	Cases	U_{in} range (m/s)	T_{in} range (K)	U_{in} excl.	T_{in} excl.
1	Training	300	[0.30, 2.00]	[288, 308]	[1.00, 1.50]	[296, 302]
2	Validation	60	[0.30, 2.00]	[288, 308]	[1.00, 1.50]	[296, 302]
3	Interpolation, velocity	20	[1.00, 1.50]	[288, 308]	—	—
4	Interpolation, thermal	20	[0.30, 2.00]	[296, 302]	—	—
5	Extrapolation, velocity	30	[2.10, 3.00]	[288, 308]	—	—
6	Extrapolation, thermal	30	[0.30, 2.00]	[310, 320]	—	—
Total		460				

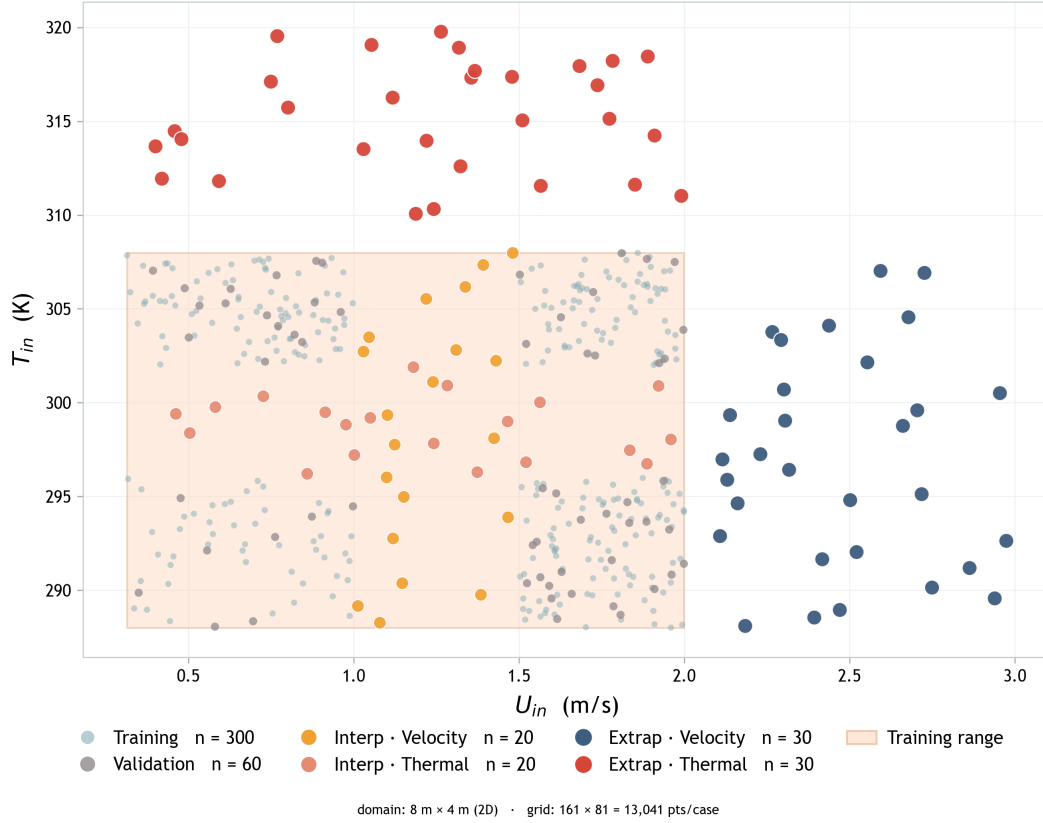


Figure 4.8: The six splits in the plane of inlet speed U_{in} and supply temperature T_{in} . Training and validation cases fill the range apart from the two interior gaps removed by rejection sampling; the interpolation splits sample inside those gaps; the extrapolation splits sample strictly above the training maximum on each axis. The remaining five parameters are sampled over their full ranges in every split.

Two implementation details affect the case counts and are worth stating. First, the segregated $k\text{-}\varepsilon$ solver does not converge on every candidate, particularly toward the corners of the parameter space where a small outlet is paired with a high inlet speed. The sweep absorbs this by oversampling: it generates candidates in batches of $2.5\times$ the number still needed, solves them with a 900-second per-case timeout, and accepts only the cases that converge, drawing further batches until the target is met or a batch limit is reached. The targets in Table 4.2 are therefore counts of successful solves, not of attempts. Second, every solved case is appended to a manifest the moment it finishes, recording its split, its seven parameters, the output path, and the solver wall-clock time. If the sweep is interrupted, it reloads the manifest, counts what each split already has, and resumes from there, which is what makes a 460-case sweep spread across days of wall-clock time practical on a single workstation.

4.4 Wall Configurations: Robustness Study

The geometry described in Section 4.2.3 was a single room, but the study runs four. The reason is to keep the architecture comparison from resting on one convenient geometry. A result that holds only for an open, unobstructed room is weak evidence about the models; a result that holds whether or not the room contains an internal partition, and wherever that partition sits, is much stronger. The four geometries are therefore treated as four separate experiments, each one run through the same pipeline of Section 4.3 from scratch.

The first geometry is the open room, with no internal partition. It has the simplest topology and the smoothest fields, and it serves as the baseline against which the obstructed cases are read. The other three introduce a single interior partition wall at one of three positions along the room: at the quarter point ($x = 0.25 L_{\text{room}}$), at the half point ($x = 0.50 L_{\text{room}}$), and at the three-quarter point ($x = 0.75 L_{\text{room}}$). The open room and the half-partition case are the two carried through in depth—the open room as the smooth baseline and the half-partition because its symmetric placement produces the strongest recirculation behind the wall and so stresses the surrogate the most. The quarter and three-quarter cases are asymmetric mirror images of each other, a short stub near the inlet side against a long one near the outlet, and they test whether the comparison survives a partition that breaks the room’s symmetry. Figure 4.9 shows the four layouts together.

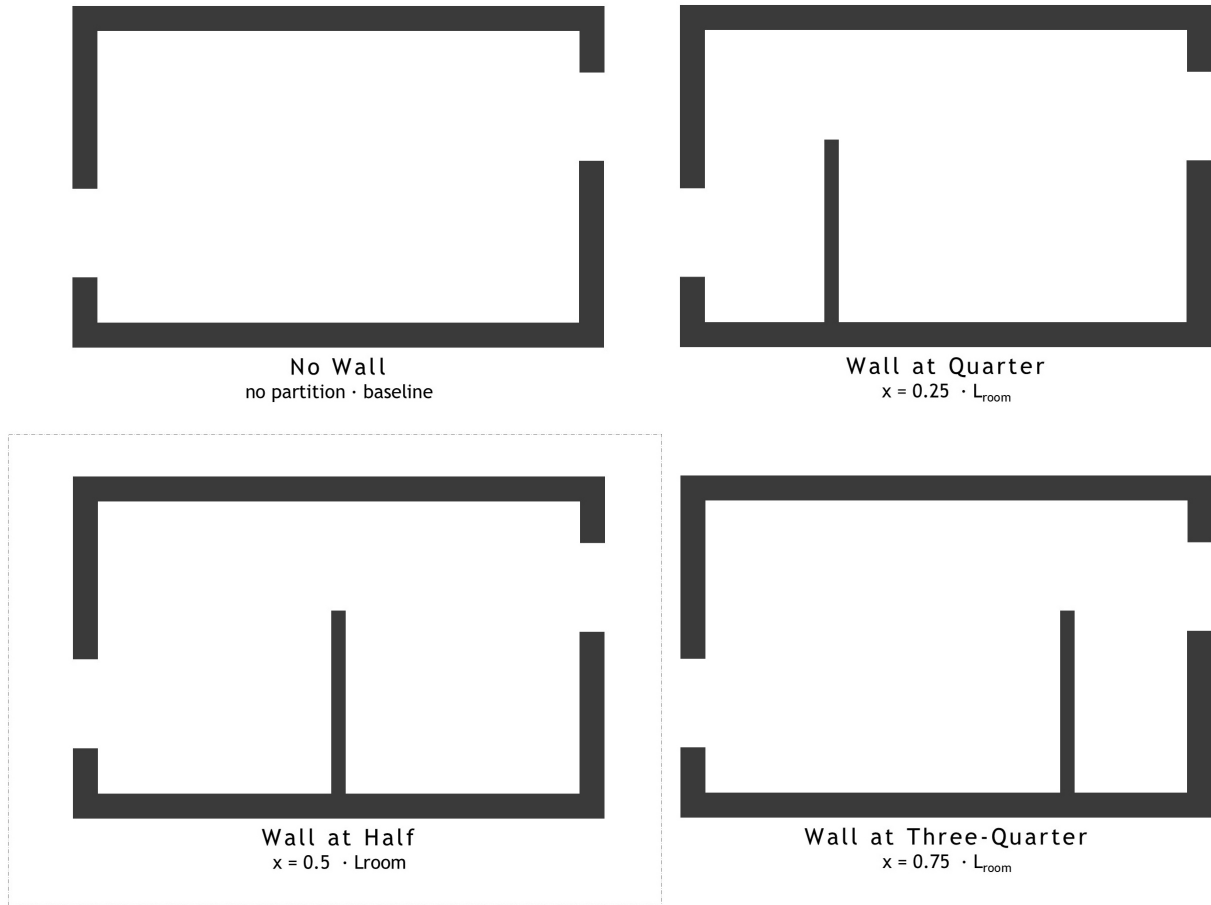


Figure 4.9: The four room geometries. From left: the open room with no partition, and the partition placed at the quarter, half, and three-quarter points of the room length. The partition dimensions and all seven sampled parameters are identical across the obstructed cases; only the wall position changes.

The partition itself is the same in all three obstructed configurations, the wall introduced in Section 4.2.3: 0.20 m thick and 2.5 m tall, which leaves a 1.5 m air gap between its top and the 4 m ceiling so that flow can pass over it rather than being fully blocked. The seven sampled parameters of Section 4.2.4 vary across the same ranges in every configuration. The only thing that changes between the four geometries is the wall's position along the room, which is why it is treated as a label on each experiment rather than as a sampled parameter.

Each configuration is run as a self-contained experiment. It gets its own seven-dimensional LHS sample under the six-split design of Section 4.3.5, 460 successfully-solved cases in all; its own three deterministically-nested training subsets at sizes 50, 150, and 300 (Section 4.3.4); and its own three trained surrogate models (Section 4.5). The seed offset described in Section 4.3.1 makes the four LHS samples independent of one another, so the geometries are not just the same cases re-solved on a moved wall. At no point is a model trained on one geometry evaluated on another geometry's data.

Whether a surrogate trained on the half-partition room could predict the three-quarter room is a different and harder question, the question of cross-geometry transfer, and it is not what this study claims; it is set aside as a limitation and as future work in the closing chapters.

What the four geometries test together is a robustness claim about the comparison itself. The interesting outcome is not any single accuracy number but a pattern: if the physics-informed network turns out to be the most reliable extrapolator, the graph network the most data-efficient, and the plain network the strongest on smooth interior fields, then that ordering should recur across all four geometries even though the absolute accuracy worsens as the room becomes more obstructed. A pattern that appears only in the open room would be a quirk of an easy case. A pattern that survives the partition, in three different positions, is a finding that says something about the architectures rather than about one room. The open-room and half-partition configurations are reported in depth in the results chapter; the quarter and three-quarter configurations are reported at equivalent depth in an appendix, and the cross-geometry comparison of the patterns is what turns the deep dive into a robustness result.

4.5 Surrogate Models

Three surrogate models are trained on the CFD data: a plain feed-forward network, a physics-informed network, and a graph network. They are not three interchangeable attempts at the same task. The plain network is the baseline, the strongest purely data-driven model the comparison can muster. The physics-informed network adds the governing equations to its training objective. The graph network adds the room’s spatial layout to its structure. Reading the three against one another is what isolates the contribution of each modelling approach: a gain the physics-informed model shows over the baseline points to the physics-informed approach as a whole, and a gain the graph model shows points to the graph-structured approach, because the approaches are changed one at a time rather than together. The physics-informed model differs from the baseline in more than its loss, though — it also carries Fourier-feature inputs and gated layers — so a gain over the baseline cannot be pinned on the physics residual alone; separating the residual’s own contribution from these architectural choices would need an ablation that holds the architecture fixed and switches only the physics terms, which is left to future work. To keep the approach-level comparison as clean as possible, the three models share as much of their training setup as the comparison allows — the same optimiser, the same base learning rate and weight decay, the same early-stopping rule on validation error, and the same z-score normalisation of inputs and outputs. Where they differ, they differ only in the one thing each is built to test.

4.5.1 ANN: The Data-Driven Baseline

The baseline is a plain feed-forward multi-layer perceptron. It maps a single query point to the six field values at that point, one point at a time, with no awareness of any other point. The input is the nine per-query features of Section 4.2.4 – the two spatial coordinates (x, y) and the seven case parameters – together with the wall position carried as a tenth input that stays constant within a configuration, for an input dimension of ten. The output is the six fields $(u, v, p, T, k, \varepsilon)$ at the query point. There is no Fourier encoding of the coordinates, no gating between layers, no skip connection, no graph, and nothing in the loss but the data. That sparseness is deliberate. The model is meant to be the honest data-driven baseline against which the physics-informed and graph networks that follow are judged, so that the comparative question stays sharp: does adding physics, or adding spatial structure, buy anything over a competent network that has only seen data?

The forward pass, sketched in Figure 4.10, is a stack of $L = 6$ hidden layers, each 256 units wide, between an input projection and an output projection,

$$\hat{\mathbf{y}} = \mathbf{W}_{\text{out}} (\phi_L \circ \phi_{L-1} \circ \cdots \circ \phi_1)(\mathbf{x}) + \mathbf{b}_{\text{out}}, \quad \phi_\ell(\mathbf{h}) = \text{Dropout}(\text{GELU}(\mathbf{W}_\ell \mathbf{h} + \mathbf{b}_\ell)), \quad (4.11)$$

where $\mathbf{x}$ is the ten-dimensional input vector, each ϕ_ℓ is one hidden layer, and $\hat{\mathbf{y}}$ is the six-dimensional prediction. A hidden layer applies an affine map $\mathbf{W}_\ell \mathbf{h} + \mathbf{b}_\ell$, passes the result through the Gaussian Error Linear Unit activation, and then drops a fraction of the activations at random during training. The activation is $\text{GELU}(x) = x \Phi(x)$, with Φ the standard-normal cumulative distribution function, from Hendrycks and Gimpel (2016); it is the same activation used inside the graph network’s processor blocks, held common across the two so the activation is not a confound in the comparison. The dropout rate is $p = 0.1$, following Srivastava et al. (2014), which regularises the network by keeping it from leaning too heavily on any single unit. The linear weights are initialised with the Xavier-uniform scheme (Glorot and Bengio 2010) and the biases at zero, so that the signal variance is preserved through the depth of the stack at the start of training.

With a ten-dimensional input, a six-dimensional output, width 256, and six hidden layers, the network has 333,318 trainable parameters, about a third of a million. These divide into the input projection from ten to 256 (2,816 parameters), five hidden maps of 256 to 256 (65,792 each, 328,960 together), and the output projection from 256 to six (1,542). The count matters for the three-way comparison. Table 4.3 places the baseline against the other two models, where it sits in the middle: larger than the graph network, smaller than the physics-informed one.

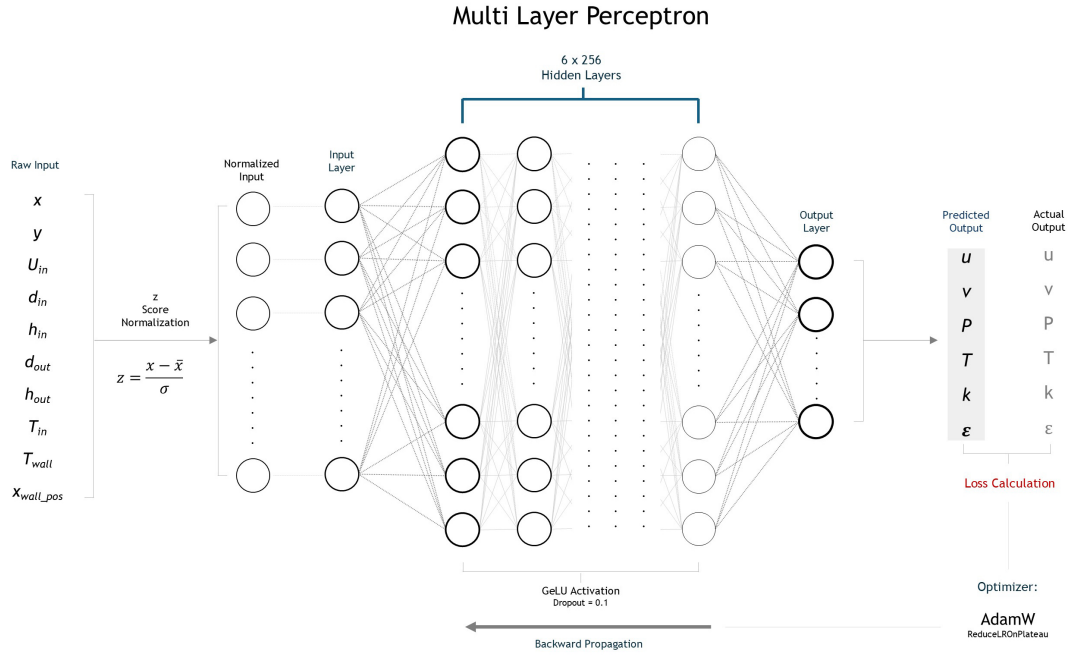


Figure 4.10: The ANN baseline: a plain feed-forward network mapping the ten input features of a single query point through six hidden layers of width 256 to the six output fields $(u, v, p, T, k, \epsilon)$ at that point. It has no Fourier encoding, no gating, no skip connection, and no graph structure – the deliberately minimal data-driven model of the comparison.

Table 4.3: The three surrogate architectures at a glance. The physics-informed and graph models are described in the two subsections that follow; they are previewed here only to show where the data-driven baseline sits.

Model	Parameters	Loss	Spatial encoding	Boundary handling
ANN	333,318 (≈ 0.33 M)	Pure data MSE	Plain (x, y) and tiled case parameters	None; implicit in the training data
PINN	696,844 (≈ 0.70 M)	Data MSE plus physics residuals	Random Fourier features and parameters	Soft Dirichlet penalty in the loss
GNN	151,302 (≈ 0.15 M)	Pure data MSE	Grid-graph topology, wall-pruned	Implicit, via wall-aware edge removal

The baseline uses the simplest possible spatial encoding, shares the graph network’s pure-MSE loss, and shares the physics-informed network’s pointwise forward pass. That middle position is chosen on purpose. It lets the comparison separate the effect of the physics constraints, read off in the difference between the physics-informed model and the baseline, from the effect of the graph structure, read off in the difference between the graph model and the baseline, without the baseline itself introducing some architectural advantage that would muddy either contrast.

The loss is the mean squared error between the predicted and the true fields, with nothing else added,

$$\mathcal{L}_{\text{ANN}}(\theta) = \frac{1}{6 N_{\text{pts}}} \sum_{i=1}^{N_{\text{pts}}} \|\hat{\mathbf{y}}(\mathbf{x}_i, \mathbf{P}_i; \theta) - \mathbf{y}(\mathbf{x}_i, \mathbf{P}_i)\|_2^2. \quad (4.12)$$

Here θ collects all of the network’s weights and biases from Equation 4.11; $(\mathbf{x}_i, \mathbf{P}_i) \in \mathbb{R}^{10}$ is the input feature vector at fluid grid point i , the spatial coordinates together with the tiled case parameters $\mathbf{P}$ (Section 4.2.4); $\hat{\mathbf{y}} \in \mathbb{R}^6$ is the predicted $(u, v, p, T, k, \varepsilon)$ at that point, expressed in the per-channel z-score units of the training-split statistics; $\mathbf{y}$ is the corresponding COMSOL ground truth in the same units; $\|\cdot\|_2$ is the Euclidean norm over the six channels; and N_{pts} is the number of fluid grid points in the current mini-batch. The factor $1/6$ averages over the six channels – the channel-mean reduction of PyTorch’s mean-squared-error loss (Paszke et al. 2019) – so the loss reads as a normalised error per channel regardless of how many channels there are. Wall cells never enter this sum; they are masked out as missing values during preprocessing, so the network is only ever trained and scored on fluid points.

This is the standard objective for multi-output regression. Both Goodfellow, Bengio, and Courville (2016) and Bishop (2006) derive mean squared error as the maximum-likelihood estimate under a Gaussian noise model with constant variance, which is to say the network output is being treated as the mean of a Gaussian observation with the same spread on every channel. Beyond the weight decay built into the optimiser, the ANN carries no other term: no equation residual, no boundary penalty, no positivity constraint. Those are exactly the terms the physics-informed network adds, and leaving them out here is what makes this the minimum-assumption model in the set. Boundary conditions reach it only through the data, in the distribution of training points near the walls and openings, which is the ordinary way a supervised regressor learns them and a further reason the plain network is the natural baseline.

Training uses the AdamW optimiser (Loshchilov and Hutter 2019) with a base learning rate of 10^{-3} and weight decay 10^{-4} , the same optimiser, base rate, and decay used for the other two models. The learning rate is cut when progress stalls, halved after sixty epochs without an improvement in validation error and floored at 10^{-7} . Training runs for up to 1500 epochs and stops early once the validation error has not improved for 250 epochs (Prechelt 1998), a point the ANN reaches well before the cap on its relatively smooth loss surface. Mini-batches are drawn from the flattened pool of fluid points at 8192 points per step.

Two of these choices are simpler than what the other models need, and the simplicity is the point. Set against the physics-informed network, the ANN has no curriculum schedule, since it has no physics terms to ramp up gradually; no collocation sampling, since its loss lives on the fixed data grid rather than on freshly drawn interior points; no gradient clipping, since a single well-behaved data term does not produce the loss spikes that physics residuals can; and no Fourier encoding, since it reads the raw normalised coordinates directly. Set against the graph network, it batches over individual points rather than over whole cases as connected graphs, and it builds no graph at all, so every prediction is independent of every other. Each of these is something the baseline does without on purpose, so that any advantage the other two show can be traced to the feature they add rather than to an incidental difference in how the baseline was trained.

4.5.2 PINN: The Physics-Informed Approach

The physics-informed network keeps the pointwise mapping of the baseline but changes what the network is asked to satisfy. Raissi, Perdikaris, and George E. Karniadakis (2019) introduced the idea: a network that takes coordinates as input and predicts the field can be trained not only to match data but to satisfy the governing equations, by adding the residual of those equations — evaluated through automatic differentiation of the network’s own output — as extra terms in the loss. A network that drives those residuals to zero is, in effect, solving the equations, with the data acting as anchoring conditions rather than as the only source of supervision. Here the network $N(x, y, \mathbf{P}; \theta)$ takes the same ten inputs as the baseline of Section 4.5.1, the two spatial coordinates and the eight case-level values, and predicts the six fields $(u, v, p, T, k, \varepsilon)$ at the queried point. Its loss carries a data term and a set of physics terms, evaluated at collocation points sampled through the domain,

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \sum_i \lambda_i \mathcal{L}_{\text{PDE}, i}, \quad (4.13)$$

with the residuals $\mathcal{L}_{\text{PDE}, i}$ and their weights λ_i spelled out in subsubsection 4.5.2. All of the derivatives in those residuals are taken by differentiating the network output with respect to its coordinate inputs, so the physics is enforced on the model itself rather than estimated from a grid.

Two well-documented failure modes of the basic formulation shape the architecture used here. The first is spectral bias: a plain MLP learns the low-frequency part of a function long before the high-frequency part, and S. Wang, H. Wang, and Perdikaris (2021) showed through a neural-tangent-kernel analysis that this is the dominant way physics-informed networks fail near walls and other sharp features — precisely the thin shear layers along the bounding walls that matter most in indoor ventilation. The response, following Tancik et al. (2020), is to encode the coordinates through random Fourier features before they enter the network, which surfaces the high-frequency content the MLP would otherwise be slow to learn. The second failure mode is optimisation: a loss with several stiff residual terms pulling at once is hard to balance and can stall in poor minima. Krishnapriyan et al. (2021) document this and propose curriculum regularisation — introducing the physics terms gradually, after the network has warmed up on data — as the remedy. Both responses are built into the model below, Fourier-encoded inputs and gated layers for the first and a staged loss schedule for the second.

With the Fourier encoding, six hidden layers of width 256, and the gated construction described next, the network has 696,844 trainable parameters, roughly 0.70 M, about $4.6\times$ the graph network’s count, though the two are trained on very different objectives. Figure 4.11 shows the architecture.

Architecture

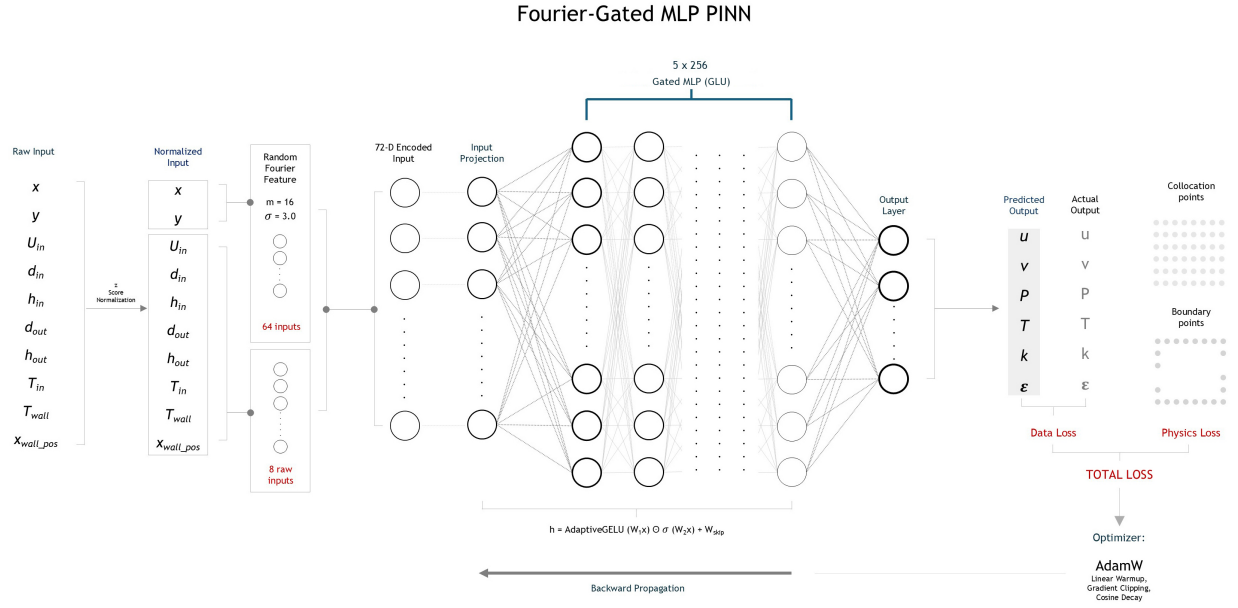


Figure 4.11: The PINN architecture: the spatial coordinates are lifted by random Fourier features and concatenated with the case parameters, passed through a stack of gated hidden blocks with adaptive activations and an input-skip residual, and projected to the six output fields. The governing-equation residuals are formed from the outputs by automatic differentiation and added to the loss.

Only the spatial coordinates are encoded. The eight case parameters are dimensionless scalars not subject to spectral bias, and they pass through unchanged. Each coordinate vector $\mathbf{v} = (x, y)$ is mapped to random Fourier features (Tancik et al. 2020),

$$\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T, \quad (4.14)$$

$$\mathbf{B} \in \mathbb{R}^{m \times n_c}, \quad B_{ij} \sim \mathcal{N}(0, \sigma_B^2).$$

where the frequency matrix $\mathbf{B}$ carries $m = 16$ frequencies per coordinate, drawn once from a Gaussian of scale $\sigma_B = 3.0$ at construction with a fixed seed and held non-trainable. Stacking the cosine and sine parts gives $2mn_c = 64$ features for the two coordinates, which are concatenated with the eight case parameters to form a 72-dimensional encoded input. Tancik et al. (2020) show that the scale σ_B matters far more than the shape of the sampling distribution, which is why it is the one Fourier hyperparameter tuned here; S. Wang, H. Wang, and Perdikaris (2021) justify the same encoding inside a physics-informed network specifically.

Each hidden layer is a gated block rather than a plain affine map and activation. It splits its input across two branches and multiplies them,

$$\mathbf{h} = \text{GELU}(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \odot \sigma(\mathbf{W}_2 \mathbf{x} + \mathbf{b}_2), \quad (4.15)$$

with $\odot$ the elementwise product and σ the logistic sigmoid. One branch carries the activated signal; the other carries a sigmoid gate in $[0, 1]$ that modulates it pointwise, so the layer can pass or suppress features depending on their value. This is a gated-linear-unit construction, pairing the sigmoid gate of Dauphin et al. (2017) with the GELU value branch of Shazeer (2020). Five such blocks of width 256 follow the input projection.

The GELU activation inside each block carries a learnable slope, $\text{GELU}(ax)$ with one trainable scalar a per layer, the layer-wise adaptive activation of Jagtap, Kawaguchi, and George Em Karniadakis (2020b) (the global-slope precursor is Jagtap, Kawaguchi, and George Em Karniadakis (2020a)). Each slope is initialised at one, so the network starts as a standard-GELU stack and adapts the slopes from there; six are learned in all, one at the input projection and one in each hidden block. One detail belongs in the open: the slope-recovery penalty that Jagtap, Kawaguchi, and George Em Karniadakis (2020b) add to keep slopes from collapsing toward zero is not part of the objective here. The slopes stayed bounded away from zero in practice, but that penalty was not imposed, and the implementation should not be read as a verbatim reproduction of their scheme.

A projection of the 72-dimensional encoded input is added at the output of every gated block,

$$\begin{aligned} \mathbf{h}_{\ell+1} &= \text{GatedBlock}_{\ell}(\mathbf{h}_{\ell}) + \mathbf{W}_{\text{skip}} \mathbf{z}, \\ \mathbf{z} &= \gamma(x, y) \oplus \mathbf{P}. \end{aligned} \quad (4.16)$$

where $\oplus$ is concatenation and $\mathbf{W}_{\text{skip}}$ is a learnable 256×72 projection. This is the projection-shortcut residual of He et al. (2016), in the variant used when the shortcut input and output widths differ. Re-injecting the Fourier-encoded input at every layer keeps the high-frequency content from being washed out by the depth of the gated stack; S. Wang, B. Li, et al. (2024) argue for exactly this input-skip residual inside physics-informed networks. All of the linear weights – the input projection, the two maps in each gated block, the skip projection, and the output layer – use Xavier-uniform initialisation (Glorot and Bengio 2010) with zero biases.

The Seven-Term Loss

The training loss is a weighted sum of seven terms, one fitting the data and six enforcing physics,

$$\begin{aligned}\mathcal{L}_{\text{total}} = & \mathcal{L}_{\text{data}} + \lambda_{\text{cont}}\mathcal{L}_{\text{cont}} + \lambda_{\text{mom}}\mathcal{L}_{\text{mom}} + \lambda_{\text{ene}}\mathcal{L}_{\text{ene}} \\ & + \lambda_{\text{ktrans}}\mathcal{L}_{\text{ktrans}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \lambda_{\text{pos}}\mathcal{L}_{\text{pos}}.\end{aligned}\quad (4.17)$$

The data term is the supervised anchor; the next four are residuals of the governing equations of Section 4.2.2, evaluated at collocation points by automatic differentiation; and the last two enforce the boundary conditions and the physical positivity of the turbulence quantities. Each is defined below, leading with what it enforces. Throughout, a hat marks a network prediction, the repeated index j is summed over the two coordinate directions, and N_{col} is the number of collocation points, re-sampled uniformly over the domain and the training-parameter set each epoch (Wu et al. 2023).

The data term holds the prediction to the COMSOL solution on the training points, the same channel-averaged mean squared error as the baseline (Equation 4.12):

$$\mathcal{L}_{\text{data}} = \frac{1}{6 N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \|\hat{\mathbf{y}}(\mathbf{x}_i, \mathbf{P}_i; \boldsymbol{\theta}) - \mathbf{y}(\mathbf{x}_i, \mathbf{P}_i)\|_2^2, \quad (4.18)$$

with N_{data} the fluid points in the batch and the norm taken over the six channels.

Continuity enforces mass conservation – that the predicted velocity field be divergence-free, the incompressible continuity equation of Equation 4.2:

$$\mathcal{L}_{\text{cont}} = \frac{1}{N_{\text{col}} s_{\text{cont}}^2} \sum_{i=1}^{N_{\text{col}}} \left(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y} \right)^2. \quad (4.19)$$

The bracket is the velocity divergence, which should vanish; $s_{\text{cont}} = U_{\text{ref}}/L_{\text{ref}}$ with $U_{\text{ref}} = 0.5$ m/s and $L_{\text{ref}} = 8.0$ m is a fixed reference scale that renders the residual dimensionless.

The momentum term enforces the steady RANS momentum balance of Equation 4.1 in each direction:

$$\begin{aligned}\mathcal{L}_{\text{mom}} = & \frac{1}{2 N_{\text{col}} s_{\text{mom}}^2} \sum_{i=1}^{N_{\text{col}}} \sum_{d \in \{x, y\}} R_{\text{mom}, d}^2, \\ R_{\text{mom}, d} = & \hat{u}_j \frac{\partial \hat{u}_d}{\partial x_j} + \frac{1}{\rho} \frac{\partial \hat{p}}{\partial x_d} - \nu_{\text{eff}} \nabla^2 \hat{u}_d, \\ \nu_{\text{eff}} = & \nu_{\text{mol}} + \nu_t, \quad \nu_t = C_\mu \hat{k}^2 / \hat{\varepsilon}.\end{aligned}\quad (4.20)$$

The residual $R_{\text{mom},d}$ in direction d collects advection of momentum, the pressure gradient, and diffusion by the effective viscosity ν_{eff} , whose turbulent part is the Boussinesq eddy viscosity $\nu_t = C_\mu \hat{k}^2 / \hat{\varepsilon}$ with $C_\mu = 0.09$ – the standard k - ε closure of Launder and Spalding (1974), given as Equation 4.5. The scale is $s_{\text{mom}} = U_{\text{ref}}^2 / L_{\text{ref}}$. Because ν_t depends on $\hat{k}$ and $\hat{\varepsilon}$, this single residual couples all six predicted fields together.

The energy term enforces steady heat advection-diffusion, the fluid energy equation of Equation 4.6:

$$\mathcal{L}_{\text{ene}} = \frac{1}{N_{\text{col}} s_{\text{ene}}^2} \sum_{i=1}^{N_{\text{col}}} \left(\hat{u}_j \frac{\partial \hat{T}}{\partial x_j} - \alpha_{\text{eff}} \nabla^2 \hat{T} \right)^2. \quad (4.21)$$

Temperature advected by the velocity is balanced against diffusion by the effective thermal diffusivity α_{eff} (molecular plus turbulent), with reference scale $s_{\text{ene}} = U_{\text{ref}} T_{\text{ref}} / L_{\text{ref}}$ and $T_{\text{ref}} = 15$ K.

The k -transport term enforces the transport equation for turbulent kinetic energy of Equation 4.3:

$$\begin{aligned} \mathcal{L}_{\text{ktrans}} &= \frac{1}{N_{\text{col}} s_k^2} \sum_{i=1}^{N_{\text{col}}} R_k^2, \\ R_k &= \hat{u}_j \frac{\partial \hat{k}}{\partial x_j} - \nabla \cdot ((\nu_{\text{mol}} + \nu_t / \sigma_k) \nabla \hat{k}) - P_k + \hat{\varepsilon}, \\ P_k &= \nu_t \left(\frac{\partial \hat{u}_i}{\partial x_j} + \frac{\partial \hat{u}_j}{\partial x_i} \right) \frac{\partial \hat{u}_i}{\partial x_j}. \end{aligned} \quad (4.22)$$

The residual R_k collects advection of k , diffusion at turbulent Prandtl number $\sigma_k = 1.0$, production by mean shear P_k , and dissipation $\hat{\varepsilon}$; the scale is $s_k = U_{\text{ref}} K_{\text{ref}} / L_{\text{ref}}$ with $K_{\text{ref}} = 0.01 \text{ m}^2/\text{s}^2$. Here a deliberate limitation must be stated plainly. The companion ε -transport equation of the k - ε model (Equation 4.4) is *not* enforced as a residual. ε enters the physics only algebraically, through the eddy viscosity $\nu_t = C_\mu \hat{k}^2 / \hat{\varepsilon}$, and is otherwise held only by the data term and by the positivity penalty below. The model thus enforces k -transport while leaving ε to data and positivity, which places it between a full RANS- k - ε PINN such as that of Pioch et al. (2023) and a turbulence-model-free RANS-PINN that learns the Reynolds stresses directly such as that of Eivazi et al. (2022), and identical to neither.

The boundary term applies the boundary conditions of Section 4.2.3 as soft penalties rather than hard constraints:

$$\mathcal{L}_{\text{BC}} = \frac{1}{6 N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} \|\hat{\mathbf{y}}(\mathbf{x}_i^{\text{BC}}, \mathbf{P}_i; \boldsymbol{\theta}) - \mathbf{y}_{\text{BC}}\|_2^2. \quad (4.23)$$

On points sampled along the boundaries it penalises departure from the prescribed values: $u = U_{\text{in}}$, $v = 0$, $T = T_{\text{in}}$ at the inlet; no-slip $u = v = 0$ and $T = T_{\text{wall}}$ at the solid walls; and zero gauge pressure at the outlet. These are Dirichlet conditions only; no gradient conditions are imposed anywhere.

Soft boundary penalties are the standard PINN treatment, equivalent to adding boundary points to the data with their own weight (Raissi, Perdikaris, and George E. Karniadakis 2019).

The positivity term keeps the turbulence quantities physical, since neither k nor ε can be negative:

$$\mathcal{L}_{\text{pos}} = \frac{1}{N_{\text{col}}} \sum_{i=1}^{N_{\text{col}}} \left[\max(0, -\hat{k})^2 + \max(0, \varepsilon_{\min} - \hat{\varepsilon})^2 \right], \quad \varepsilon_{\min} = 10^{-8}. \quad (4.24)$$

Each one-sided penalty is zero when the quantity is in range and grows quadratically when it is not, with the floor $\varepsilon_{\min}$ holding the eddy-viscosity denominator away from zero. This is the textbook quadratic penalty for an inequality constraint (Nocedal and Wright 2006); it is not a log-barrier or an interior-point scheme.

Each physics residual is divided by a fixed dimensional scale built from the reference conditions, rather than rebalanced on the fly during training. Fixed non-dimensionalisation is the classical CFD approach (Pope 2000), and because the surrogate spans many cases that share the same reference scales, fixed weights stay interpretable and stable across the sweep. Adaptive alternatives exist – gradient-norm balancing (S. Wang, Teng, and Perdikaris 2021), trainable per-point attention weights (McClenny and Braga-Neto 2020), and the broader training practices surveyed by S. Wang, Sankaran, et al. (2023) – and are noted here for contrast rather than used.

The physics terms are not all switched on at once. They enter one at a time on a curriculum schedule ordered by stiffness, so the network first learns to fit the data and the easiest constraints before the stiffest ones are asked of it (Krishnapriyan et al. 2021). Training begins with the data term alone for the first 200 epochs; continuity joins at epoch 200, the boundary penalty at 300, momentum at 600, energy at 1000, and k -transport last at 1500, each ramped linearly from zero to its target weight over the following 400 epochs. The k -transport term comes last because its eddy-viscosity dependence $\hat{k}^2/\hat{\varepsilon}$ is the stiffest part of the system and benefits most from a velocity field that has already settled.

Training uses AdamW (Loshchilov and Hutter 2019) with base learning rate 10^{-3} and weight decay 10^{-4} , the shared optimiser of all three models. The learning rate ramps linearly from zero to base over a warmup (Goyal et al. 2017) and then anneals along a cosine to a small floor (Loshchilov and Hutter 2017). Gradients are clipped to a global norm of 1.0 (Pascanu, Mikolov, and Bengio 2013), which matters most at the moments a new physics term opens its curriculum gate and briefly spikes the loss.

The residual equations used here are simpler than the equations solved in COMSOL, so this difference should be made clear. The COMSOL model in Section 4.2.2 is weakly compressible. Its continuity equation is written as $\nabla \cdot (\rho, \mathbf{u}) = 0$, where density changes with temperature (Equation 4.2), and its momentum equation also includes density and the full stress-divergence term (Equation 4.1).

In the PINN residuals, these terms are simplified. Continuity is enforced as the constant-density condition $\partial \hat{u}/\partial x + \partial \hat{v}/\partial y = 0$ (Equation 4.19), and the momentum residual also assumes constant density and constant fluid properties (Equation 4.20). The energy and k -transport residuals follow the same general advection-diffusion form as the COMSOL equations, but they are also written with constant properties. The ε -transport equation (Equation 4.4) is not included in the residual loss, as mentioned earlier.

These simplifications are acceptable for the cases studied here. The inlet velocities are low, with a Mach number around 0.01, so the flow can be treated as nearly incompressible. Since gravity is turned off in the COMSOL setup (Section 4.2.1), the temperature-dependent density change is also small, only a few percent for the wall-to-inlet temperature differences used in this study. The constant-density residuals therefore neglect a small effect rather than a dominant one.

More importantly, the residuals are not used as the main source of truth. The data loss (Equation 4.18) trains the network directly against the full weakly-compressible k - ε COMSOL solution. The residual terms only act as a soft physical guide. They help the model stay close to the main physical balances, especially where the data coverage is limited or in the extrapolation cases. In that role, the simpler incompressible residuals are useful because they guide the solution toward physically reasonable behaviour without replacing the COMSOL reference solution itself.

4.5.3 GNN: The Graph-Structured Approach

The graph network shares the baseline’s pure-data objective but changes the forward pass. Where the ANN predicts each point in isolation, the GNN predicts each point from its neighbourhood in a graph laid over the room, so a node’s prediction depends on the cases around it rather than on its own coordinates alone. The scaffold is the encoder-processor-decoder graph network of Battaglia et al. (2018): an encoder lifts the per-node inputs into a latent representation, a processor of several graph blocks refines that representation by exchanging information along the edges, and a decoder reads out the six fields at each node. This scaffold has become the standard one for physics surrogates (Sanchez-Gonzalez et al. 2020; Pfaff et al. 2021). The processor here uses four independently parameterised blocks rather than one shared block applied four times, the no-weight-sharing variant of Battaglia et al. (2018), and each block is a GraphSAGE layer (Hamilton, Ying, and Leskovec 2017).

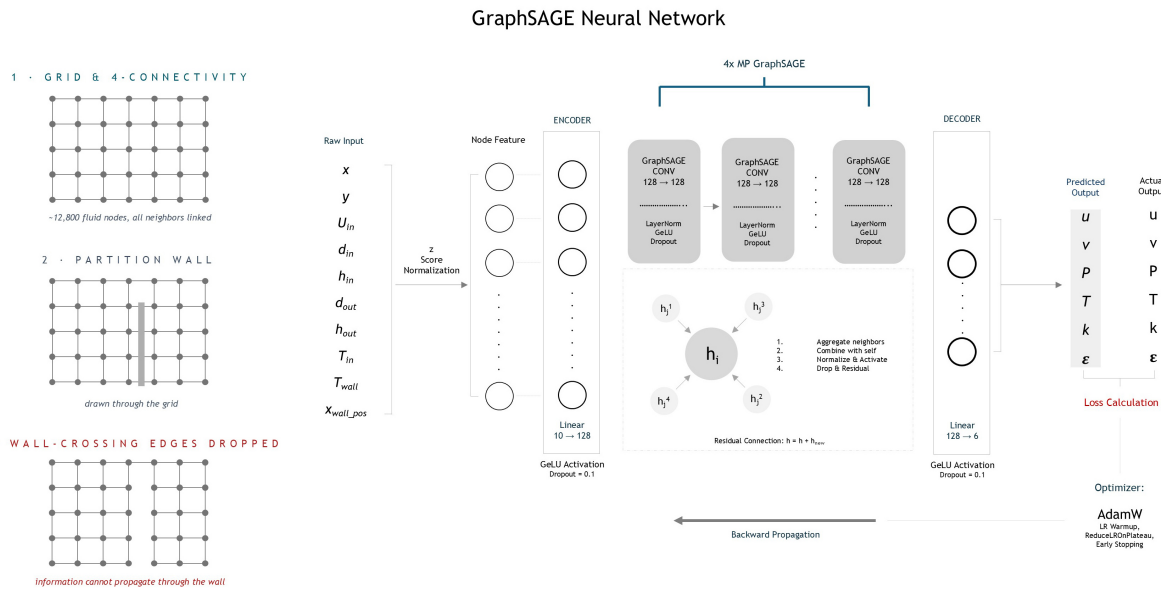


Figure 4.12: The GNN architecture: each CFD case becomes a grid graph whose edges follow four-connectivity but are pruned wherever they would cross the partition wall. An encoder lifts the per-node features, four GraphSAGE blocks pass messages along the surviving edges, and a decoder reads out the six fields at every fluid node.

It is worth being precise about what this model is and is not. It is a data-driven surrogate: there is no equation residual in its loss, no boundary penalty, and no positivity term. Wall geometry enters the model in one place only, through the connectivity of the graph, as the next subsection describes. It is therefore not a physics-informed graph network, and it is not a reimplementaion of MeshGraphNets.

It is an encoder-processor-decoder GNN with GraphSAGE message-passing blocks, trained by plain data regression against the COMSOL fields, with wall-aware graph construction as its only inductive bias toward geometry. That deliberate sparseness is the point of the comparison: holding the loss identical to the baseline’s isolates the effect of the graph-structured forward pass.

With a hidden width of 128 and four message-passing blocks, the network has 151,302 trainable parameters, roughly 0.15 M — about five times fewer than the physics-informed network of Section 4.5.2, and trained on a single loss term rather than seven. Figure 4.12 shows the architecture.

Graph Construction and Node Features

Each CFD case is turned into a graph $G = (V, E)$. The nodes are the fluid grid points of the 161×81 export grid of Section 4.2.5, with the cells that fall inside a wall left out, giving roughly twelve to thirteen thousand nodes per case depending on the wall configuration. The edges follow four-connectivity — each interior node joined to its right and upper neighbours, made bidirectional — with one rule that does all the geometric work: an edge is kept only if both of its endpoints are fluid nodes. If either endpoint lies inside the partition wall, the edge is dropped.

That single rule is the model’s entire inductive bias toward wall geometry. Because no edge crosses the partition, information cannot propagate through it during message passing; the wall is felt not as a term in the loss or a label on a node but as an absence of connections. This is a different mechanism from the physics-informed network’s soft boundary penalty (subsubsection 4.5.2), which acts in the loss, and from the node-type labelling used in MeshGraphNets (Pfaff et al. 2021), which tags boundary nodes and lets the network learn how to treat them. Edge removal, soft penalty, and node typing are three distinct design choices, and the graph model commits to the first.

The reach of that connectivity follows from the depth. With four message-passing blocks on four-connectivity, each output node draws on its four-hop neighbourhood, which on the open interior grid is the set of cells within taxicab distance four — a diamond of 41 cells that fits inside a nine-by-nine box, not a filled nine-by-nine square. Near a wall the diamond is clipped by the edge-removal rule. This finite receptive field is the graph model’s counterpart to the physics-informed network’s Fourier encoding, but it is a spatial bias on which nodes can talk to which, not a spectral one on how coordinates are represented; the two are not the same kind of prior, and the contrast is taken up in the discussion.

The per-node features are the node’s coordinates concatenated with the case parameters, the same ten inputs the other two models use,

$$\mathbf{x}_{\text{node}} = [x, y, U_{\text{in}}, h_{\text{in}}, d_{\text{in}}, h_{\text{out}}, d_{\text{out}}, T_{\text{in}}, T_{\text{wall}}, x_{\text{wall}}] \in \mathbb{R}^{10}, \quad (4.25)$$

with the eight case-level scalars broadcast identically onto every fluid node of the case and the whole vector z-score-normalised per channel over the training split. Attaching global properties to every node this way is the usual pattern in graph-network surrogates (Sanchez-Gonzalez et al. 2020). It is also frankly redundant, since every node of a case carries the same eight values, which costs memory at the hidden layers; the leaner alternative would be to carry the case parameters as a single graph-level attribute and condition the block updates on it (Battaglia et al. 2018). This work takes the simpler tile-and-concatenate route, and records it as a simplification rather than a contribution.

Message Passing and Block Composition

Each processor block passes messages with the mean-aggregator form of GraphSAGE (Hamilton, Ying, and Leskovec 2017). A node’s updated representation combines a transform of its own state with a transform of the average of its neighbours,

$$\mathbf{h}'_i = \mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \cdot \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j, \quad (4.26)$$

where $\mathcal{N}(i)$ is the set of i ’s graph neighbours. This is algebraically the concatenate-then-project form of the original algorithm with the weight matrix split into a self block $\mathbf{W}_1$ and a neighbour block $\mathbf{W}_2$; the non-linearity is applied not inside the operator but in the surrounding block. The broader framework in which this sits – a message function, a node update, and a readout – is the message-passing neural network of Gilmer et al. (2017); GraphSAGE is the specific variant used here.

Three honest caveats place this exactly. The aggregation runs over the full four-connectivity neighbourhood rather than a sampled subset, which is tractable because each graph has only around twelve thousand nodes; the per-layer embedding normalisation of the original algorithm is not used (the library default is off, and that default is kept); and the edges carry no features, only binary connectivity, with no edge length or boundary tag. Each of these is a deliberate simplification.

Around the message-passing operator, each of the four blocks adds normalisation, activation, dropout, and a residual connection,

$$\mathbf{h} \leftarrow \mathbf{h} + \text{Dropout}(\text{GELU}(\text{LayerNorm}(\text{SAGEConv}(\mathbf{h}, E))))). \quad (4.27)$$

The residual is an identity shortcut in the sense of He et al. (2016), available here because each block’s input and output share the hidden width of 128, so no projection is needed. This differs from the physics-informed network’s projection-shortcut residual (Equation 4.16), which had to project the encoded input up to the hidden width; both are variants of the same idea but they carry different content. The normalisation is layer normalisation (Ba, Kiros, and Hinton 2016), applied to each node’s feature vector,

$$\begin{aligned} \mu &= \frac{1}{H} \sum_{i=1}^H a_i, & s &= \sqrt{\frac{1}{H} \sum_{i=1}^H (a_i - \mu)^2}, \\ \text{LayerNorm}(\mathbf{a}) &= \frac{\mathbf{g}}{s} \odot (\mathbf{a} - \mu) + \mathbf{b}, \end{aligned} \quad (4.28)$$

where H is the feature dimension, μ and s are the mean and standard deviation across that node’s features, and $\mathbf{g}, \mathbf{b}$ are learnable per-channel gain and bias initialised at one and zero. It sits after the message-passing operator and before the activation. The activation is GELU (Hendrycks and Gimpel 2016), the same one used in the baseline, and dropout at rate $p = 0.1$ (Srivastava et al. 2014) is applied in the encoder, in every block, and in the decoder.

One initialisation detail is worth stating rather than glossing. The Xavier-uniform scheme (Glorot and Bengio 2010) is applied by walking the model for standard linear layers, which catches the encoder and decoder projections but not the message-passing weights or the normalisation parameters – those are not stock linear layers and keep their library defaults. So only about an eighth of the network’s parameters are Xavier-initialised; the message-passing weights keep the library’s Kaiming default and the normalisation parameters keep the identity initialisation. The claim is not that the whole network is Xavier-initialised, only the encoder and decoder.

Loss and Training

The loss is the channel-averaged mean squared error between the predicted and the true fields at every fluid node,

$$\mathcal{L}_{\text{GNN}}(\boldsymbol{\theta}) = \frac{1}{6 |V_{\text{train}}|} \sum_{v \in V_{\text{train}}} \|\hat{\mathbf{y}}_v(\boldsymbol{\theta}) - \mathbf{y}_v\|_2^2, \quad (4.29)$$

and that is the whole of it. There is no physics residual, no boundary penalty, no positivity term; V_{train} is the combined set of fluid nodes across the graphs in the current mini-batch, the norm runs over the six channels, and the factor $1/6$ is the same channel-mean reduction the other two models use (Paszke et al. 2019), which keeps the three data losses comparable in magnitude.

Two comparisons make the design legible. In form, Equation 4.29 is identical to the baseline’s data loss (Equation 4.12): both are six-channel mean squared error against the same COMSOL targets in the same normalisation. The difference between the two models is not in the loss but in the prediction. The baseline’s output at a point depends only on that point’s own feature vector, while the graph model’s output at a node depends on the features across its four-hop neighbourhood, with the wall-aware pruning cutting some of those connections. Any improvement of the graph model over the baseline on a given split can therefore be put down to the graph-structured forward pass and the wall-aware connectivity, not to any difference in objective.

The graph model’s loss is also not the physics-informed network’s loss with the physics weights set to zero, and the difference is more than bookkeeping. The graph model batches over whole cases as graphs rather than over individual points; its loss is evaluated only at the observed fluid nodes, whereas the physics-informed network’s residuals live at collocation points sampled anywhere in the domain; and it has no curriculum, no gradient clipping, and no soft boundary penalty, since its one geometric prior is built into the graph rather than into the loss. The three together — a data-only baseline, a seven-term physics-informed network, and a data-only graph model with wall-aware connectivity — are the axis the whole comparison turns on.

Training uses AdamW (Loshchilov and Hutter 2019) with base learning rate 10^{-3} and weight decay 10^{-4} , the shared optimiser again, but with no curriculum since there is only one loss term. The learning rate ramps linearly over the first 50 epochs (Goyal et al. 2017), after which it is halved whenever the validation error fails to improve for 80 epochs, down to a floor of 10^{-7} ; the plateau scheduler is held off until the warmup finishes so the two do not interfere. Training runs for up to 1500 epochs and stops early after 300 epochs without improvement on the validation error (Prechelt 1998). Mini-batching uses the graph library’s loader, which packs eight case graphs into one block-diagonal batch per step. Set against the physics-informed network’s training, the graph model uses no gradient clipping (it was stable without it across all three data-efficiency sizes), no curriculum, and no collocation sampling.

4.6 Downstream Comfort Prediction

The surrogates predict fields; what an architect actually wants to know is whether the room is comfortable. The bridge between the two is a fixed set of standard equations, and this section describes how those equations are applied to the predicted fields. Nothing about comfort is learned. The thermal-comfort indices are computed as a deterministic post-processing layer that takes the predicted velocity, temperature, and turbulence and returns the comfort measures defined in Section 2.2, using ISO 7730 (ISO 2005) for the whole-body indices PMV and PPD and the local index of draught, and ISO 7726 (ISO 1998) for the mean radiant temperature.

4.6.1 A Deterministic Comfort Layer

Keeping the comfort layer deterministic and separate from the surrogate is a deliberate design choice with a useful consequence. Because the same fixed equations are applied to the predicted fields and to the COMSOL ground truth, any difference between a predicted comfort map and the true one comes entirely from error in the underlying field predictions, not from any modelling choice in the comfort layer itself. The standard becomes a transparent, physically interpretable amplifier of the field error: it lets a small error in predicted velocity or temperature be read directly as its effect on comfort. This mapping of CFD velocity, temperature, and turbulence onto the ISO 7730 indices is the standard practice in CFD-based comfort studies (Stamou and Katsiris 2006; Sevilgen and Kilic 2011; Koskela et al. 2001; Cheng, Niu, and N. Gao 2012); the only thing replaced here is the source of the fields.

4.6.2 Field Inputs and Personal Parameters

For each grid cell the comfort layer reads four quantities. Three come straight from the fields: the air temperature $t_a = T - 273.15$ in °C, the air-speed magnitude $v_a = \sqrt{u^2 + v^2}$, and the turbulent kinetic energy k . The fourth, the mean radiant temperature $\bar{t}_r$, is computed from the wall temperatures by the view-factor calculation of the next subsection. The remaining inputs to PMV are personal rather than environmental, and they are held at standard sedentary-office values throughout: a metabolic rate of 1.2 met, clothing insulation of 0.5 clo, and relative humidity of 50 percent, with the conversions $1 \text{ met} = 58.15 \text{ W/m}^2$ and $1 \text{ clo} = 0.155 \text{ m}^2 \text{ K/W}$ as defined in ISO 7730 (ISO 2005). All of the CFD cases analysed fall inside the standard's stated applicability ranges for air temperature, radiant temperature, air speed, metabolic rate, and clothing.

4.6.3 Mean Radiant Temperature

Mean radiant temperature at a point is the fourth-power-weighted average of the surrounding surface temperatures (the general form given as Equation 2.16). In the two-dimensional room slice it is computed by weighting the four enclosing wall temperatures by the angle each wall subtends at the point, following the view-factor form of ISO 7726 Annex B (ISO 1998),

$$\bar{t}_r = \left(\sum_i T_i^4 F_{p-i} \right)^{1/4} - 273.15, \quad F_{p-i} = \frac{\alpha_i}{2\pi}, \quad (4.30)$$

where T_i is the temperature of wall i in kelvin and F_{p-i} is the view factor from the point to that wall. The view factor is the subtended angle α_i divided by the full 2π , and the four angles are obtained from the endpoints of each wall as seen from the point (x, y) ,

$$\begin{aligned} \alpha_{\text{floor}} &= \arctan \frac{x - x_{\min}}{y - y_{\min}} + \arctan \frac{x_{\max} - x}{y - y_{\min}}, \\ \alpha_{\text{ceiling}} &= \arctan \frac{x - x_{\min}}{y_{\max} - y} + \arctan \frac{x_{\max} - x}{y_{\max} - y}, \\ \alpha_{\text{left}} &= \arctan \frac{y - y_{\min}}{x - x_{\min}} + \arctan \frac{y_{\max} - y}{x - x_{\min}}, \\ \alpha_{\text{right}} &= \arctan \frac{y - y_{\min}}{x_{\max} - x} + \arctan \frac{y_{\max} - y}{x_{\max} - x}. \end{aligned} \quad (4.31)$$

These four angles sum to exactly 2π at any interior point, so the view factors sum to one and the weighting is exact. The construction is the planar reduction of the crossed-strings view-factor method (Modest 2013), and it treats the occupant as a small isotropic radiator at the evaluation point, the natural two-dimensional version of the small-sphere sensor model in ISO 7726 (ISO 1998). The full sitting- and standing-person angle factors of Fanger (1970) would refine this in three dimensions but do not apply to the two-dimensional slice analysed here. Air does not enter the calculation, since over the path lengths of a room it is effectively transparent in the longwave band and only the bounding surfaces radiate.

4.6.4 PMV, PPD, and Draught Rate

The predicted mean vote and the predicted percentage dissatisfied are computed from the heat-balance form of Equation 2.11 and Equation 2.13, with the air temperature, air speed, and radiant temperature taken from the maps above. Three auxiliary quantities feed the heat balance: the clothing area factor, the convective heat-transfer coefficient, and the water-vapour partial pressure,

$$f_{cl} = \begin{cases} 1.00 + 1.290 I_{cl} & I_{cl} \leq 0.078 \text{ m}^2\text{K/W}, \\ 1.05 + 0.645 I_{cl} & \text{otherwise,} \end{cases} \quad (4.32)$$

$$h_c = \max(12.1\sqrt{v_a}, 2.38|t_{cl} - t_a|^{0.25}),$$

$$p_a = \frac{\text{RH}}{100} \cdot 1000 \cdot \exp\left(16.6536 - \frac{4030.183}{t_a + 235}\right).$$

The convective coefficient takes the larger of a forced-convection limit (Mitchell 1974) and a natural-convection limit (Nishi and Gagge 1970), so the model uses whichever mode dominates at the local air speed. The clothing surface temperature t_{cl} that these feed is itself defined implicitly, appearing on both sides of its own balance,

$$t_{cl} = 35.7 - 0.028(M - W) - I_{cl} \left\{ 3.96 \times 10^{-8} f_{cl} [(t_{cl} + 273)^4 - (\bar{t}_r + 273)^4] + f_{cl} h_c (t_{cl} - t_a) \right\}. \quad (4.33)$$

The fourth-power radiative term and the dependence of h_c on t_{cl} leave Equation 4.33 with no closed form, so ISO 7730 specifies a damped iteration – averaging the previous and updated estimates before resubstituting – rather than plain fixed-point iteration, which is unstable once the clothing insulation is large enough that the right-hand side stops being a contraction. To avoid re-implementing a numerically delicate scheme, PMV and PPD are evaluated through the validated reference port `pythermalcomfort` (Tartarini and Schiavon 2020), which transcribes the ISO 7730 Annex D procedure directly. The percentage dissatisfied then follows from PMV by the fixed relation of Equation 2.13, unchanged across editions of the standard.

The draught rate is computed from the same fields by Equation 2.14, with the turbulence intensity taken pointwise from the predicted turbulent kinetic energy through Equation 2.15 rather than set to the standard’s default of 40 percent. Computing the intensity from the resolved k field is the whole reason k is carried as a predicted channel: it lets the draught estimate respond to the actual local turbulence rather than a fixed assumption.

4.6.5 Categories, Verification, and Assumptions

The pointwise indices are summarised against the acceptability categories of Table 2.1. The comfort fraction reported in the results is the area fraction of a room satisfying $|\text{PMV}| \leq 0.5$, which is Category B of ISO 7730 and aligns with the corresponding band of EN 16798 (European Committee for Standardization 2019) and ASHRAE 55 (ASHRAE 2023).

The implementation was checked against three independent references. The PMV and PPD port reproduces the ten canonical cases of the ISO 7730 Annex D table to within 0.01 PMV units and half a percentage point of PPD; the draught-rate routine matches a closed-form evaluation of the standard’s equation to machine precision; and the view-factor mean radiant temperature satisfies the radiative-closure identity, that the view factors sum to one, to within machine precision at every interior point under uniform-wall test cases. The full verification table is given in an appendix.

A few assumptions are built into this layer and are carried forward to the limitations discussion. The personal parameters are fixed at sedentary-office values rather than varied, so the maps describe one occupant type. The two-dimensional slice resolves only the four in-plane walls, so the radiant calculation omits any out-of-plane surfaces a full three-dimensional room would contribute. Only the whole-body indices and the draught rate are computed; the other local-discomfort indices of ISO 7730 are out of scope. And the turbulence intensity that drives the draught estimate inherits the isotropy assumption of the $k\text{-}\varepsilon$ closure, so it is only as good as that closure’s treatment of near-wall turbulence.

4.7 Evaluation Framework

The three surrogates are judged along four lines, each tied to one of the questions the paper set out to answer. The first is raw field accuracy: how closely each model reproduces the CFD fields. The second is generalisation: whether that accuracy holds up inside the interior gaps and outside the training range, read separately. The third is data efficiency: how the accuracy scales with the amount of training data. The fourth is downstream comfort accuracy: whether the predicted fields carry through to the right comfort verdict. The instruments for all four are the six evaluation splits of Section 4.3.5 and the staged training sizes of Section 4.3.4, applied identically to every model and, under the robustness study of Section 4.4, to every geometry.

4.7.1 Field Accuracy

Field accuracy is measured with three metrics, computed per output channel over the fluid grid points of a split,

$$\begin{aligned} \text{MSE} &= \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2, \\ \text{MAE} &= \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i|, \\ R^2 &= 1 - \frac{\sum_{i=1}^N (\hat{y}_i - y_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}. \end{aligned} \tag{4.34}$$

where $\hat{y}_i$ and y_i are the predicted and true values of one channel at point i , $\bar{y}$ is the mean of the true values over the split, and N is the number of fluid points. The three say different things. The mean squared error is the primary training and ranking metric; squaring the error makes it punish large local mistakes heavily, so it rewards a model that avoids bad outliers. The mean absolute error is the average error in the field's own units, which makes it the easiest to read physically – so many meters per second of velocity error, so many kelvin of temperature error. The coefficient of determination R^2 rescales the error against the variance of the field itself: it is one for a perfect prediction and zero for a model no better than guessing the field's mean everywhere, which makes it comparable across channels whose magnitudes differ by orders of magnitude. Reporting all three keeps the picture honest, since a model can look good on one and poor on another – a low MAE with a low R^2 , for instance, flags a field that is nearly flat and barely being predicted at all.

These are accuracy metrics, and accuracy is only one axis of what a physics-informed model is meant to improve; Ma et al. (2025) note that the same error measures used for data-driven models do not, on their own, capture the physical consistency or robustness that a physics-informed approach is supposed to add. That caveat is taken seriously in the framing of the results, but the field metrics reported here are deliberately held to these three so that the three architectures are compared on exactly the same footing.

4.7.2 Interpolation versus Extrapolation

The central comparison is not overall accuracy but where each model’s accuracy holds and where it breaks. The six-split design makes this directly measurable: the three field metrics are reported separately on the two interpolation splits, where the test cases sit inside the interior gaps the training set was made to avoid (Section 4.3.2), and on the two extrapolation splits, where they sit beyond the training range (Section 4.3.3). Keeping the velocity and thermal axes apart means the breakdown can be read one parameter at a time, so a model that generalises well in temperature but poorly in velocity shows that pattern rather than having it averaged away.

This separation is the empirical form of the paper’s hypothesis. A purely data-driven model is expected to interpolate within its training support and to degrade outside it, while the physics and graph priors are expected to help most exactly where the data runs out. Ma et al. (2025) make testing in extrapolation, and on limited data, a principle for evaluating physics-informed models rather than an afterthought, which is the principle the extrapolation splits put into practice. The gap between a model’s in-range and out-of-range accuracy, rather than either number alone, is what the architecture comparison turns on.

4.7.3 Data Efficiency

Data efficiency is measured by repeating the whole evaluation at the three nested training sizes of Section 4.3.4 – 50, 150, and the full 300 cases – for each architecture, and reading the field metrics as a function of training-set size. Because the smaller sets are deterministic subsets of the larger ones and the test splits are held fixed, the resulting curves measure the marginal value of more CFD data for each model rather than the noise of a fresh draw. The question they answer is the one transposed from Wei and Ooka (2023): whether the physics-informed model holds its accuracy as the data shrinks better than the data-driven one does, and whether the graph model shows a different efficiency profile again.

A model whose accuracy collapses between 300 and 50 cases is expensive to deploy, since every case is a CFD solve; a model that stays accurate on 50 is the more practical surrogate even if it is a little behind at full data.

4.7.4 Downstream Comfort Accuracy

The last line of evaluation asks whether the field errors matter where it counts, in the comfort verdict an architect would actually read. Because the comfort layer of Section 4.6 is deterministic, the comfort error is nothing more than the field error pushed through the ISO 7730 equations, which lets the standard act as an interpretable amplifier of the field metrics above. Comfort accuracy is reported as the mean absolute error of the predicted PMV against the value computed from the ground-truth fields, in PMV units, and the same for PPD and the draught rate, both in percentage points. Two classification measures sit alongside these: the fraction of grid cells the model places in the correct ISO 7730 comfort category, and the fraction where it agrees with the ground truth on the Category B test, whether the predicted comfort satisfies $|\text{PMV}| \leq 0.5$. The first asks how close the comfort numbers are; the second asks whether the model would lead a designer to the same decision about the room. Reporting comfort error against deterministically computed ground-truth comfort, at the full grid of each case, follows the validation practice of Botarelli et al. (2025), and it closes the loop from a predicted field back to the design judgement the surrogate is ultimately meant to support.

5

Results

This chapter reports how the three surrogates perform when the evaluation framework of Section 4.7 is applied to the trained models. It follows the same four lines that framework set out: how closely each model reproduces the CFD fields, how that accuracy holds inside the interpolation gaps and outside the training range, how it propagates into the ISO 7730 comfort measures, and how it scales with the amount of training data. Two of the four room geometries are shown here in full, the open room and the half-partition room of Section 4.4. The quarter- and three-quarter-partition rooms are reported with the identical set of figures in the appendix, which is what lets the comparison be read as a robustness check rather than a single result.

The aim throughout is an honest comparison, not a verdict. The figures are read for what they show – which model leads on which split, by how much, and where each one holds or breaks – and the reading stays at the level of observation. Why the models behave as they do, how the pattern lines up with the prior work of Chapter 3, and what it means for the design workflow are taken up in the discussion that follows. Where the data does not favour the physics-informed model, that is reported as plainly as where it does.

A few conventions hold across the chapter. In the comparison figures the models keep a fixed colour code, the ANN in teal, the PINN in red, and the GNN in amber, with the COMSOL ground truth in navy. In the field maps, each panel is laid out the same way: the top row shows the COMSOL ground truth alongside the ANN, GNN and PINN predictions, and the bottom row shows each model’s absolute error against that ground truth. Temperature is shown in degrees Celsius and the velocity components in metres per second. The four test cases referred to throughout are the four held-out splits of Section 4.3.5: interpolation and extrapolation, each on the velocity and on the thermal axis.

5.1 Evaluation Cases

The evaluation setup is important for reading the results. As described in Chapter 4, each room geometry was tested using a LHS sweep over seven case parameters (Section 4.3.1). Of these, 460 COMSOL cases converged and were divided into the six splits in Section 4.3.5. 300 cases were used for training, 60 for validation, and four held-out splits were kept for testing. The four test splits were designed to check generalisation. For both inlet speed and supply temperature, one split tests interpolation inside the training range, while another tests extrapolation beyond the training maximum. Only one parameter is moved out of range at a time, so its effect can be seen clearly. All errors in this chapter are measured against the COMSOL $k\text{-}\varepsilon$ solution of Section 4.2, not physical experiments. Therefore, accuracy here means agreement with the CFD reference; experimental validation is left for future work.

5.2 Reconstructed Fields

The first read on the models is the simplest one: do the predicted fields look like the CFD? Figure 5.1 through Figure 5.4 place the predictions next to the ground truth for each test case, in both rooms, with the per-model error maps directly beneath. They are the qualitative counterpart to the metrics that follow, and they show where in the room the differences actually sit rather than collapsing them to a single number. In the interpolation cases (Figure 5.1 and Figure 5.2), all three reconstructions track the ground truth, but the error panels already begin to separate the models, and they separate on temperature first. The ANN's temperature error is visibly the largest of the three, while the GNN and PINN hold the temperature field tight against the ground truth. On the velocity components the three are much closer, with the GNN and PINN error panels a little fainter than the ANN's and what error there is concentrating where the gradients are steepest – along the inlet jet and in the thin shear layers next to the walls. The same ordering holds in both rooms, and in the half-partition case all three pick out the recirculation behind the wall, at least in outline. In the extrapolation cases (Figure 5.3 and Figure 5.4) the separation becomes stark, and again it is temperature that drives it. In the thermal-extrapolation case, as the supply temperature is pushed past the training range, the ANN and GNN temperature error panels saturate – their predicted temperature fields sit far from the ground truth across much of the room – while the PINN's temperature panel stays close to it. The velocity error panels grow for all three out of range, but least for the PINN. The qualitative picture is a physics-informed model that holds its fields beyond the training range while the data-driven and graph models break down, most dramatically on temperature; the next section puts numbers to how large that gap is.

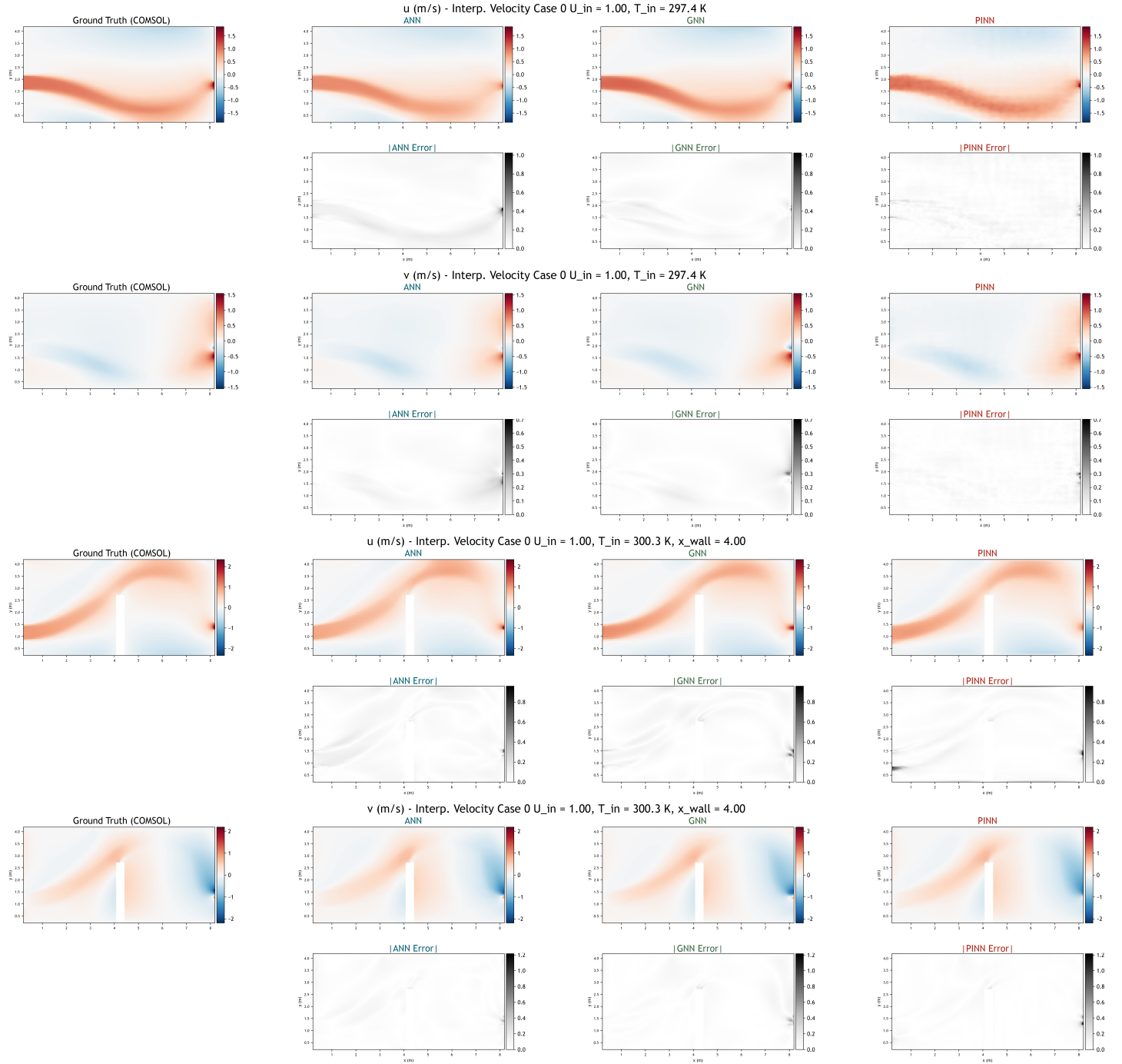


Figure 5.1: Velocity in the interpolation-velocity case. Top to bottom: streamwise velocity u (m/s) and vertical velocity v (m/s) for the open room, then the same two for the half-partition room. In each panel, the top row is the COMSOL ground truth with the ANN, GNN and PINN predictions, and the bottom row is each model's absolute error.

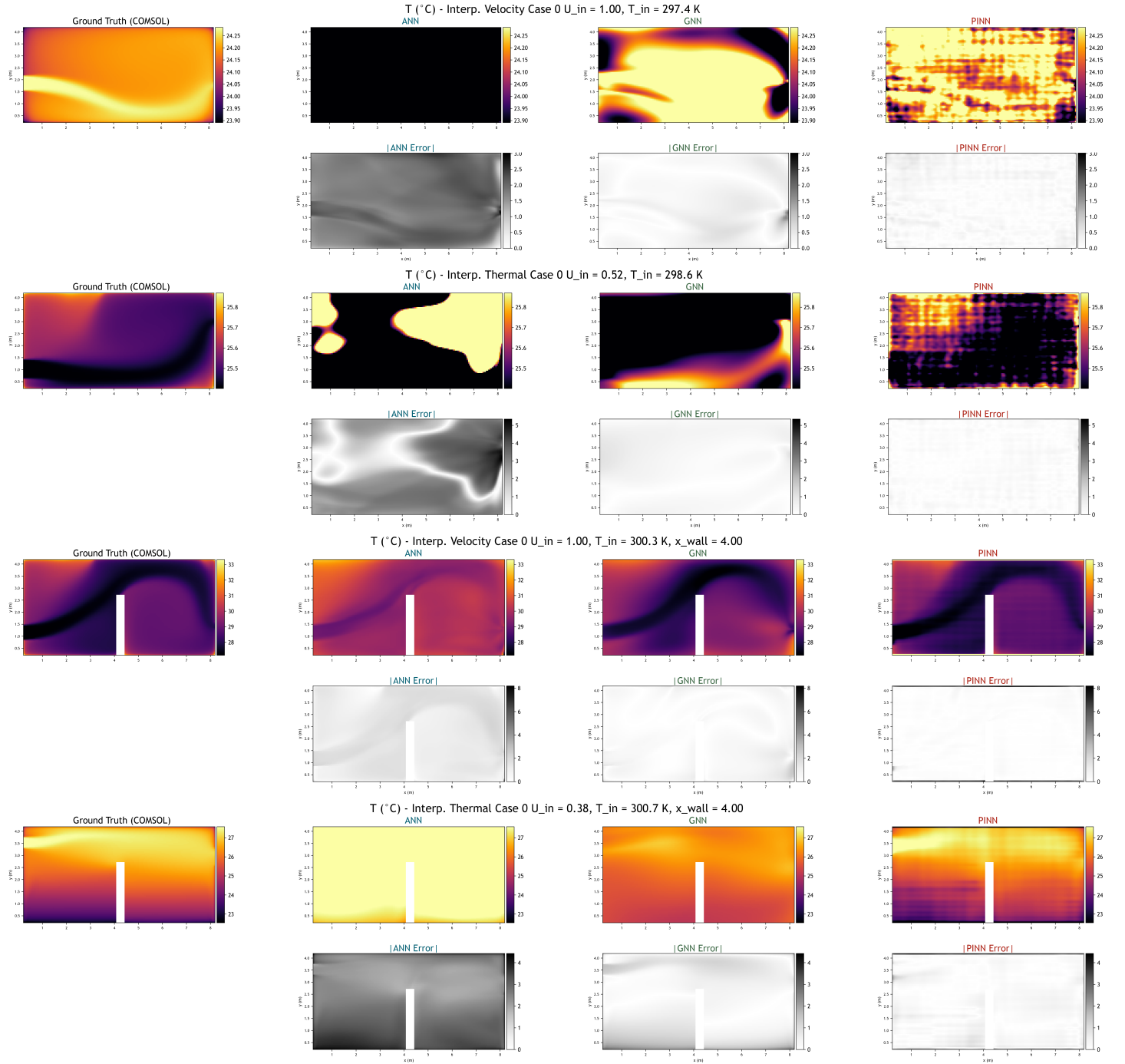


Figure 5.2: Temperature in the two interpolation cases. Top to bottom: temperature T ($^{\circ}\text{C}$) in the interpolation-velocity and interpolation-thermal cases for the open room, then the same two for the half-partition room. Panel layout as in Figure 5.1.

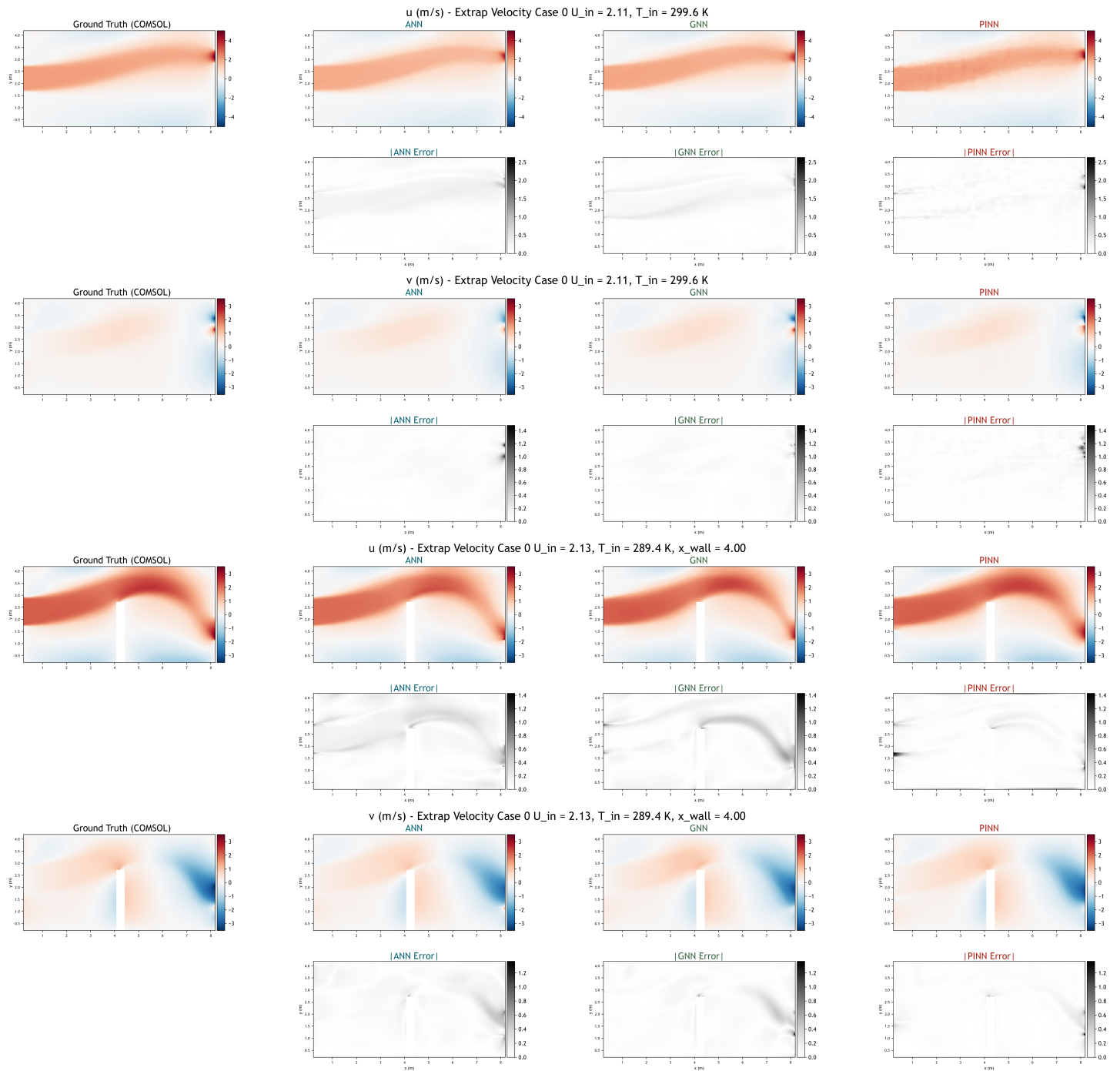


Figure 5.3: Velocity in the extrapolation-velocity case. Top to bottom: streamwise velocity u (m/s) and vertical velocity v (m/s) for the open room, then the same two for the half-partition room. Panel layout as in Figure 5.1.

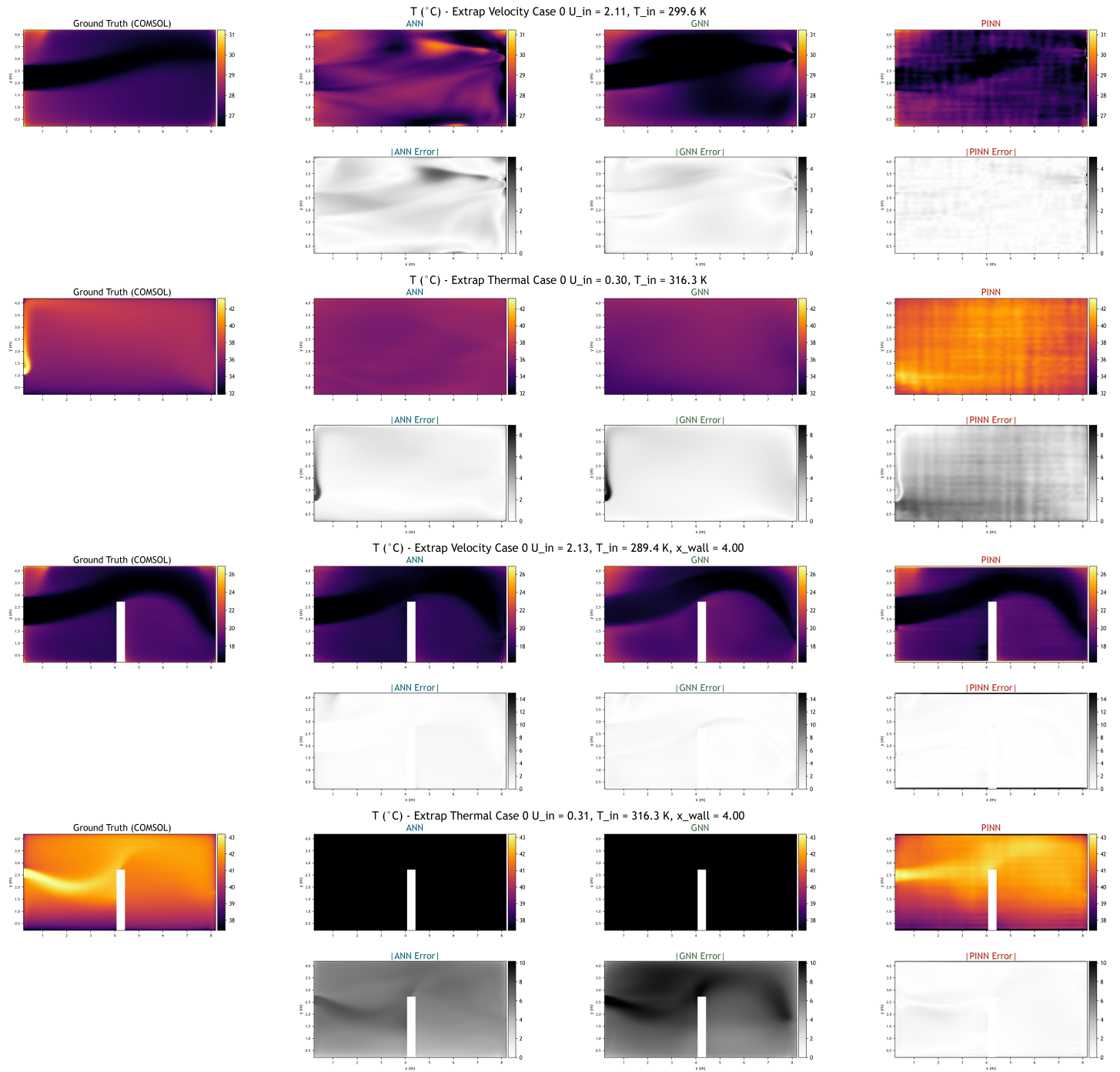


Figure 5.4: Temperature in the two extrapolation cases. Top to bottom: temperature T ($^{\circ}\text{C}$) in the extrapolation-velocity and extrapolation-thermal cases for the open room, then the same two for the half-partition room. Panel layout as in Figure 5.1.

5.3 Field-Accuracy Metrics

The maps showed where the errors sit; Figure 5.5 puts numbers on them. It reports the mean squared error and the mean absolute error for each model, split by test case and by field, for both rooms, with the lowest error in each split-and-field cell outlined. The numbers fall into two regimes, inside the training range and outside it, and the two regimes have different winners.

Temperature is where the models are furthest apart, and the reading is not uniform across the rooms. In the open room the PINN reconstructs temperature best in every test case: its mean squared error in the two interpolation cases is about 0.09 and 0.12, against roughly 0.21 and 0.24 for the GNN and 2.95 and 5.71 for the ANN. In the half-partition room that interpolation lead passes to the GNN, whose temperature error drops to about 0.16 and 0.20 while the PINN sits at 0.71 and 0.55 and the ANN again trails far behind. The picture then changes completely in thermal extrapolation. There the PINN's temperature error stays near 0.79 in both rooms, essentially unchanged from its in-range value, while the ANN climbs to about 28 in the open room and 33 in the half-wall, and the GNN climbs further still, to roughly 38 and 53. The GNN, the strongest temperature interpolator, becomes the weakest temperature extrapolator — worse even than the plain baseline once the supply temperature leaves the training range.

The velocity components tell a closer and steadier story. In range the GNN is the most accurate on velocity, if narrowly: its interpolation-velocity streamwise error is about 0.017 in the open room and 0.0054 in the half-wall, just ahead of the PINN's 0.0175 and 0.0055 and clearly ahead of the ANN's 0.030 and 0.0090. Out of range the order swaps, as it did for temperature: the PINN takes the velocity lead, with an extrapolation-velocity streamwise error near 0.11 in the open room and 0.039 in the half-wall, against the GNN's 0.15 and 0.093 and the ANN's 0.18 and 0.13. The mean absolute error table carries the same ordering throughout, with the gaps compressed, since it is less sensitive than the squared metric to the few large local errors.

Read together, the tables divide cleanly. Inside the training range the GNN is the most accurate field reconstructor, leading on velocity in both rooms and on temperature in the half-wall, with the PINN close behind and the ANN trailing. Outside it the PINN is ahead on every field, and on temperature it is ahead by one to two orders of magnitude, holding its accuracy almost flat across the boundary the other two fall off. The ANN is the weakest model in range but, on temperature extrapolation, is not the worst — that is the GNN, whose graph-structured fit to the training range does not carry beyond it. This split between an interpolation winner and an extrapolation winner is the central quantitative result, and it holds in both geometries.

Mean Squared Error — $\text{mean}[(\text{pred} - \text{truth})^2]$ in PHYSICAL UNITS: u, v in $(\text{m/s})^2$, p in Pa^2 , T in K^2 , k in $(\text{m}^2/\text{s}^2)^2$, ϵ in $(\text{m}^2/\text{s}^2)^2$. Scale differs per field. **Dark outline = best model per split×field.**

model	ANN										GNN										PINN									
field	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ
split																														
Training	0.1273	1.612e-05	1.51	0.001199	0.0004441	0.009962	0.1252	3.66e-05	1.494	0.002177	0.0008826	0.04357	0.06043	2.652e-05	2.202	0.002062	0.0009953	0.03179												
Validation	0.947	0.0001332	1503	0.01639	0.005825	0.06276	0.3233	0.0001349	1111	0.01596	0.005824	0.08262	0.1568	0.0001383	850.8	0.02318	0.00756	0.07176												
Interp Velocity	2.95	0.0001106	87.29	0.02979	0.00909	0.01298	0.2095	6.192e-05	8.987	0.01711	0.006676	0.0145	0.08856	5.152e-05	41.11	0.01753	0.006969	0.01102												
Interp Thermal	5.709	0.0002098	713.7	0.04632	0.01781	0.2582	0.2444	0.0001572	935.7	0.02613	0.01192	0.2259	0.1223	9.537e-05	1383	0.01886	0.007452	0.1928												
Extrap Velocity	4.27	0.003238	737.6	0.1776	0.06872	1.058	1.351	0.002244	262.5	0.1463	0.05123	1.028	0.313	0.001296	216.6	0.1109	0.04612	0.8242												
Extrap Thermal	28.36	0.0008328	337.7	0.1577	0.03802	0.06702	38.09	0.0003841	172	0.1257	0.03694	0.05043	0.7942	0.0002129	162.3	0.04708	0.01781	0.04804												

Mean Absolute Error — $\text{mean}[(\text{pred} - \text{truth})]$ in PHYSICAL UNITS: u, v in m/s , p in Pa , T in K , k in m^2/s^2 , ϵ in m^2/s^2 . **Dark outline = best model per split×field.**

model	ANN										GNN										PINN									
field	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ
split																														
Training	0.2551	0.0020	0.8951	0.0217	0.0113	0.0045	0.2444	0.0025	0.8214	0.0279	0.0140	0.0077	0.1712	0.0022	0.8415	0.0250	0.0127	0.0106												
Validation	0.6844	0.0054	8.8566	0.0602	0.0316	0.0066	0.4162	0.0055	7.3260	0.0658	0.0321	0.0095	0.2749	0.0051	6.5909	0.0673	0.0340	0.0144												
Interp Velocity	1.2228	0.0066	4.4545	0.0942	0.0443	0.0067	0.3444	0.0040	1.7130	0.0620	0.0353	0.0080	0.2153	0.0040	2.6468	0.0677	0.0365	0.0125												
Interp Thermal	2.1095	0.0060	8.8920	0.1066	0.0537	0.0083	0.3658	0.0045	9.6663	0.0792	0.0468	0.0093	0.2340	0.0037	12.5338	0.0683	0.0382	0.0156												
Extrap Velocity	1.5510	0.0311	12.3050	0.2456	0.1182	0.0438	0.8686	0.0236	6.7083	0.2132	0.0962	0.0377	0.4114	0.0183	6.4554	0.1622	0.0929	0.0470												
Extrap Thermal	4.7232	0.0147	8.8803	0.2328	0.1010	0.0136	5.1234	0.0097	5.3429	0.1891	0.0885	0.0161	0.5294	0.0076	4.7441	0.1165	0.0613	0.0215												

Mean Squared Error — $\text{mean}[(\text{pred} - \text{truth})^2]$ in PHYSICAL UNITS: u, v in $(\text{m/s})^2$, p in Pa^2 , T in K^2 , k in $(\text{m}^2/\text{s}^2)^2$, ϵ in $(\text{m}^2/\text{s}^2)^2$. Scale differs per field. **Dark outline = best model per split×field.**

model	ANN										GNN										PINN									
field	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ
split																														
Training	0.1756	3.113e-05	1.806	0.002378	0.001066	0.004313	0.1394	3.966e-05	1.811	0.002798	0.001307	0.01546	0.8027	2.872e-06	0.7821	0.004406	0.001326	0.0008245												
Validation	0.7668	8.435e-05	290.7	0.005561	0.002801	0.0253	0.3575	0.0001276	403	0.005654	0.002697	0.05933	0.8138	3.081e-05	167.4	0.006658	0.002808	0.07092												
Interp Velocity	1.675	5.166e-05	6.396	0.008964	0.003093	0.002092	0.1619	3.52e-05	3.452	0.005424	0.002275	0.004344	0.7052	1.106e-05	1.232	0.005516	0.002081	0.002494												
Interp Thermal	6.331	7.274e-05	109.7	0.01138	0.003149	0.006408	0.1978	6.996e-05	55.62	0.01167	0.003352	0.01274	0.5494	2.777e-05	28.84	0.01157	0.003048	0.009311												
Extrap Velocity	4.294	0.003279	3160	0.1254	0.03681	1.967	1.599	0.002195	3679	0.09265	0.03628	1.975	1.124	0.0005541	3508	0.03942	0.012	1.263												
Extrap Thermal	32.97	0.0001593	193.8	0.01461	0.005378	0.01174	53.07	9.58e-05	21.22	0.007229	0.002987	0.01592	0.7945	3.006e-05	30.65	0.006381	0.002331	0.02199												

Mean Absolute Error — $\text{mean}[(\text{pred} - \text{truth})]$ in PHYSICAL UNITS: u, v in m/s , p in Pa , T in K , k in m^2/s^2 , ϵ in m^2/s^2 . **Dark outline = best model per split×field.**

model	ANN										GNN										PINN									
field	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ	T	k	p	u	v	ϵ
split																														
Training	0.3113	0.0031	1.0020	0.0326	0.0211	0.0046	0.2687	0.0029	0.8628	0.0325	0.0209	0.0061	0.2469	0.0009	0.2986	0.0241	0.0134	0.0038												
Validation	0.5570	0.0046	5.2666	0.0447	0.0290	0.0067	0.3931	0.0039	4.8652	0.0421	0.0273	0.0081	0.2604	0.0019	3.2326	0.0317	0.0188	0.0058												
Interp Velocity	0.8533	0.0048	1.6512	0.0608	0.0332	0.0047	0.2988	0.0032	1.1713	0.0399	0.0258	0.0053	0.2262	0.0018	0.5368	0.0334	0.0196	0.0038												
Interp Thermal	2.2166	0.0046	4.2215	0.0549	0.0302	0.0051	0.3116	0.0036	2.7890	0.0472	0.0272	0.0062	0.2375	0.0021	1.8325	0.0386	0.0202	0.0043												
Extrap Velocity	1.4850	0.0323	20.9034	0.2213	0.1007	0.0519	0.9441	0.0208	19.6384	0.1743	0.0987	0.0506	0.3398	0.0113	23.0918	0.0922	0.0472	0.0339												
Extrap Thermal	5.3585	0.0069	6.3727	0.0770	0.0446	0.0078	6.6128	0.0048	2.1399	0.0561	0.0335	0.0070	0.3715	0.0026	1.7521	0.0380	0.0227	0.0051												

Figure 5.5: Field-accuracy tables. Top to bottom: mean squared error and mean absolute error per model, broken out by test case and field, for the open room, then the same two for the half-partition room. The lowest error in each split-and-field cell is outlined, and the outline marks the winning model.

5.4 Predicted Comfort

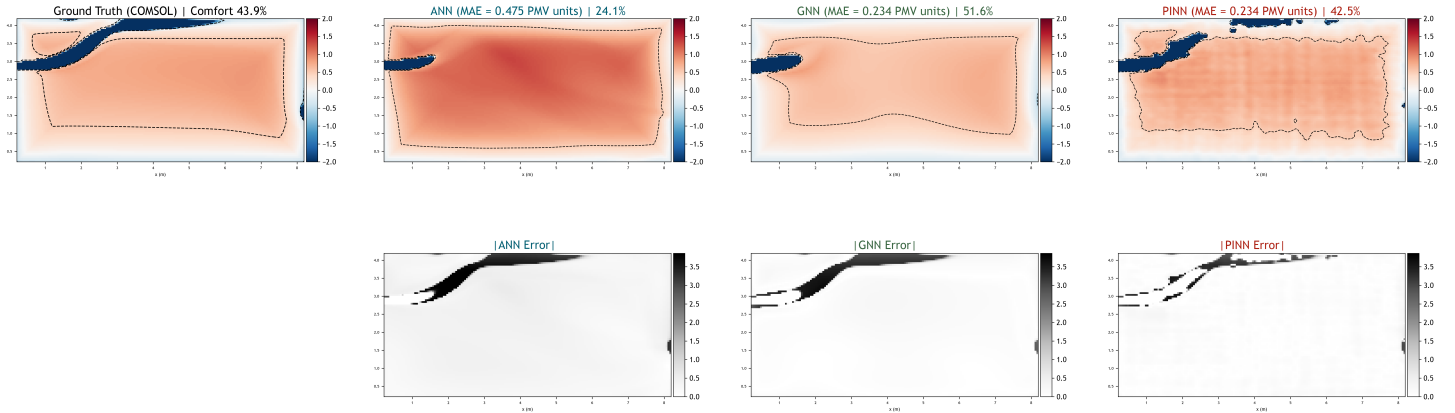
Because the comfort layer is deterministic (Section 4.6), the comfort error is the field error carried through the ISO 7730 equations rather than anything the models learn separately. The comfort results therefore follow the field results, and this section reads how far. There are two views: the predicted comfort fields set against the ground truth (Figure 5.6 and Figure 5.7), and the agreement on the room-level comfort fraction, case by case (Figure 5.8 and Figure 5.9).

The comfort maps show the PMV, the predicted percentage dissatisfied, and the draught rate for each model beside the ground truth, each panel annotated with its mean absolute error against the CFD. The PINN's comfort fields sit closest to the ground truth in both rooms, and the per-panel error annotations bear this out across all three indices. Because the whole-body indices weight temperature most heavily, the PINN's temperature accuracy is what carries it to the cleanest PMV and percentage-dissatisfied maps; the draught rate, which also leans on velocity and turbulence, separates the models less. The next view puts numbers on the agreement.

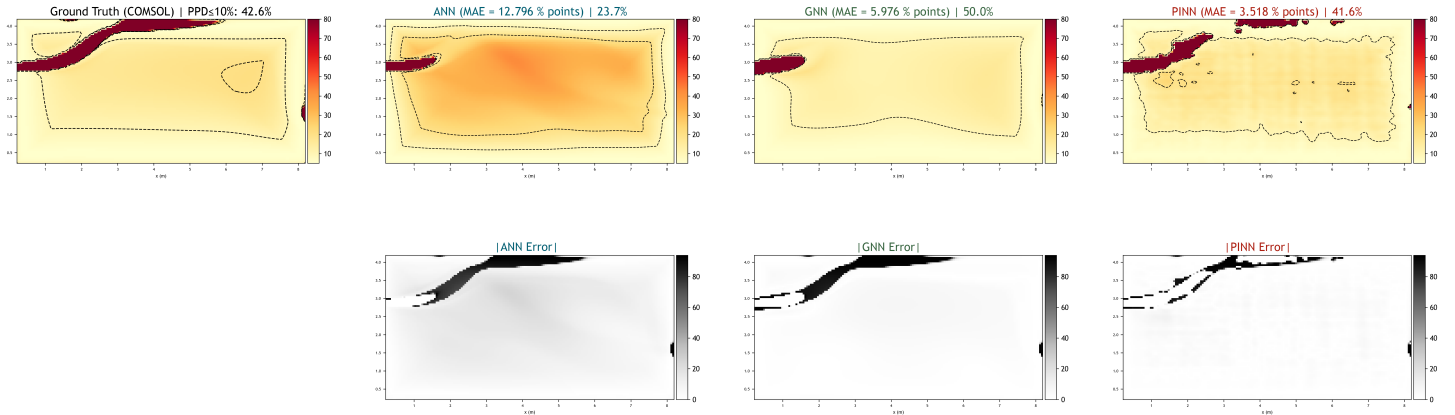
The comfort-fraction view asks the design question directly: would the surrogate report the same fraction of the room as comfortable as the CFD does? Each panel plots a model's predicted comfort fraction against the ground-truth value case by case, with the absolute deviation beside it and the best model per split outlined. The PINN's fractions track the ground-truth line most closely on all three indices and in both rooms. Its mean error on the PMV-based comfort fraction is about 1.5 percentage points in the open room and 0.7 in the half-wall, against the GNN's 2.1 and 1.0 and the ANN's 5.1 and 4.5; the percentage-dissatisfied fraction lands almost identically. The draught-rate fraction is the loosest for every model, but the order is unchanged – the PINN off by 2.7 points in the open room and 1.2 in the half-wall, the GNN by 3.7 and 3.6, the ANN by 5.4 and 5.2. The GNN is second on the comfort fraction in both rooms, comfortably ahead of the ANN.

Across both rooms the comfort ordering is the same as the field ordering it descends from: the PINN closest to the ground truth, the ANN and GNN close behind and swapping second place between the geometries. The model that reconstructs the fields best therefore also gives the most trustworthy comfort verdict – by a clear margin on the temperature-driven PMV and percentage dissatisfied, and a narrower one on draught.

PMV (range - [-3, +3]) - Valid Case 16



PPD (%) - Predicted Percentage Dissatisfied - Valid Case 16



DR (%) - Draught Rate - Valid Case 16

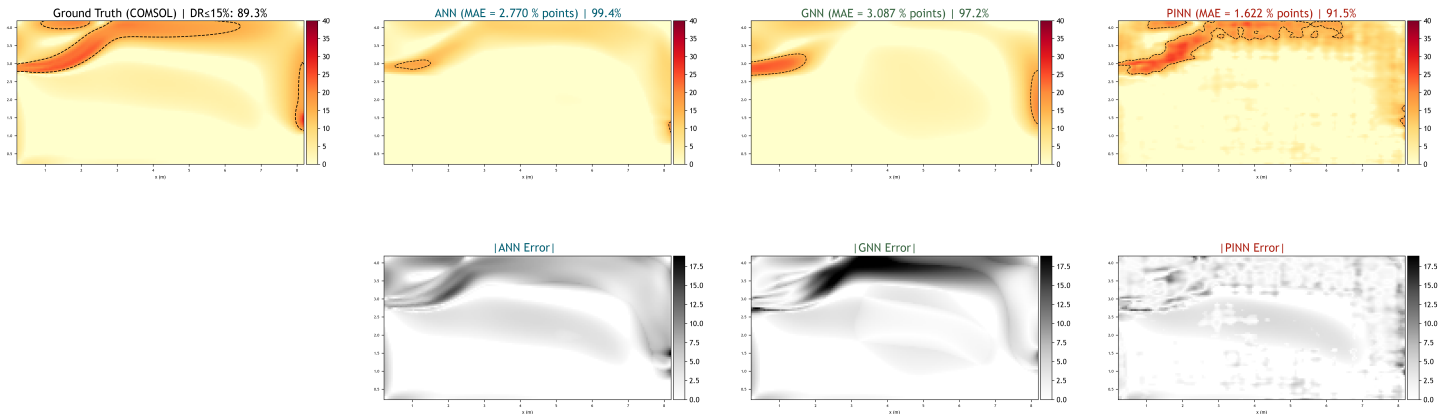
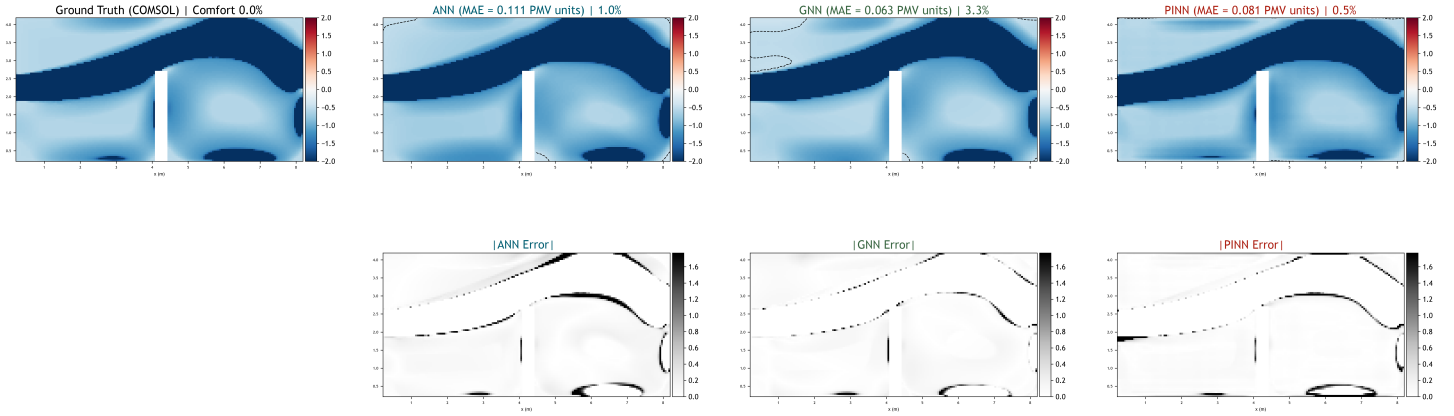
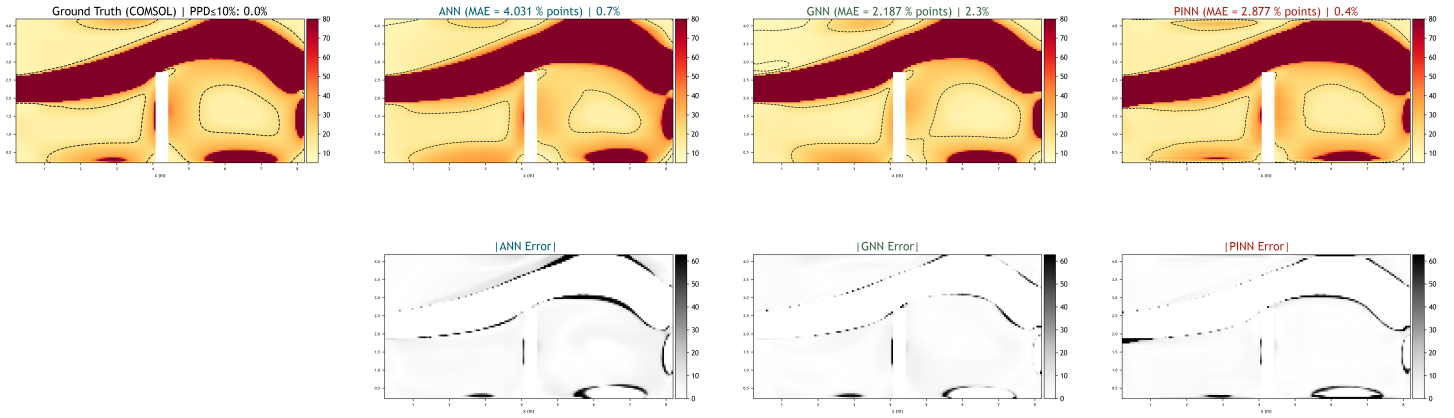


Figure 5.6: Predicted comfort fields in the open room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Each row shows the COMSOL ground truth and the ANN, GNN and PINN predictions, with each prediction's mean absolute error against the ground truth annotated.

PMV (range = [-3, +3]) - Valid Case 26



PPD (%) - Predicted Percentage Dissatisfied - Valid Case 26



DR (%) - Draught Rate - Valid Case 26

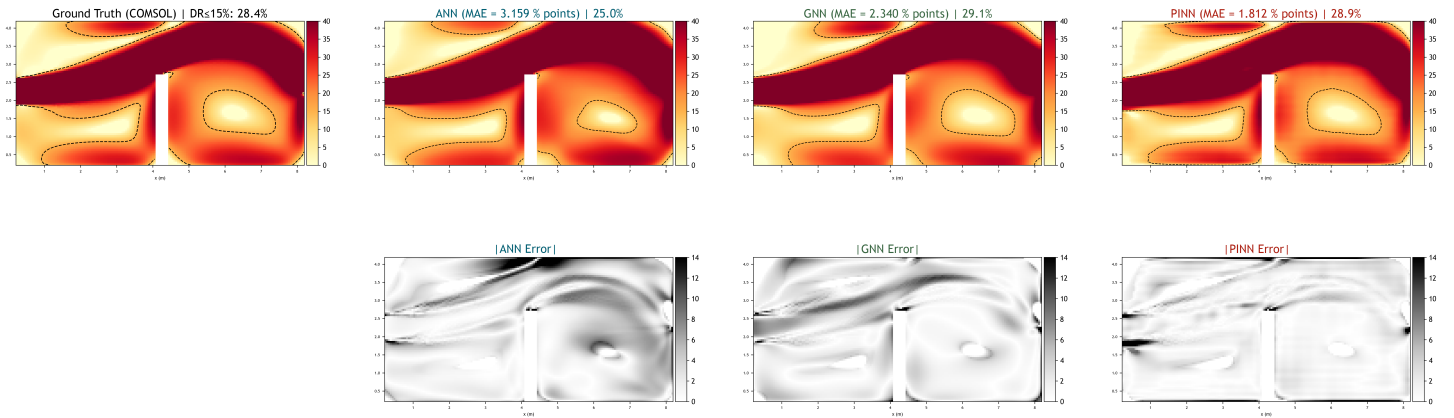


Figure 5.7: Predicted comfort fields in the half-partition room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Panel layout as in Figure 5.6.

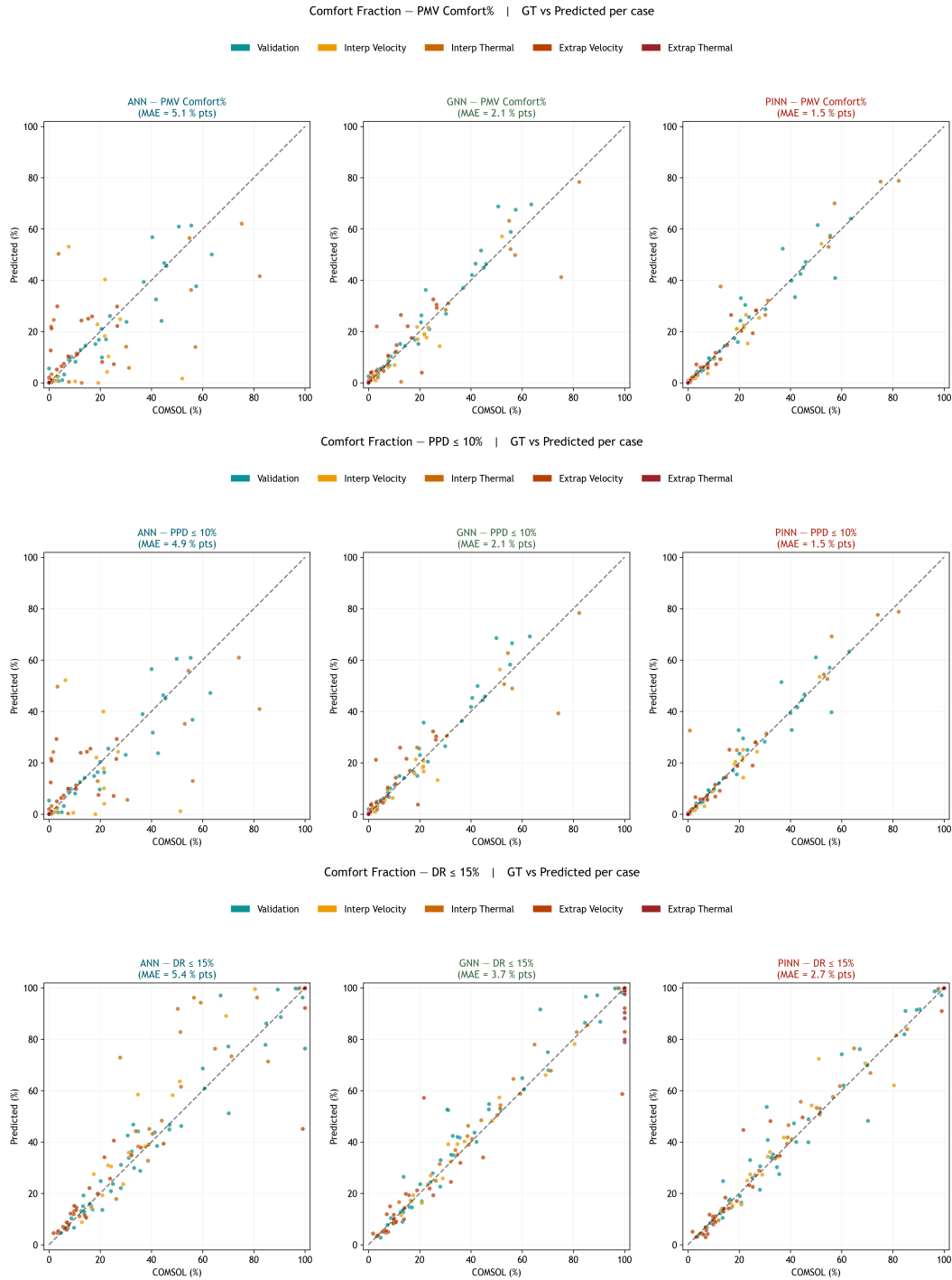


Figure 5.8: Comfort-fraction agreement in the open room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Each panel plots each model’s predicted comfort fraction against the ground-truth value.

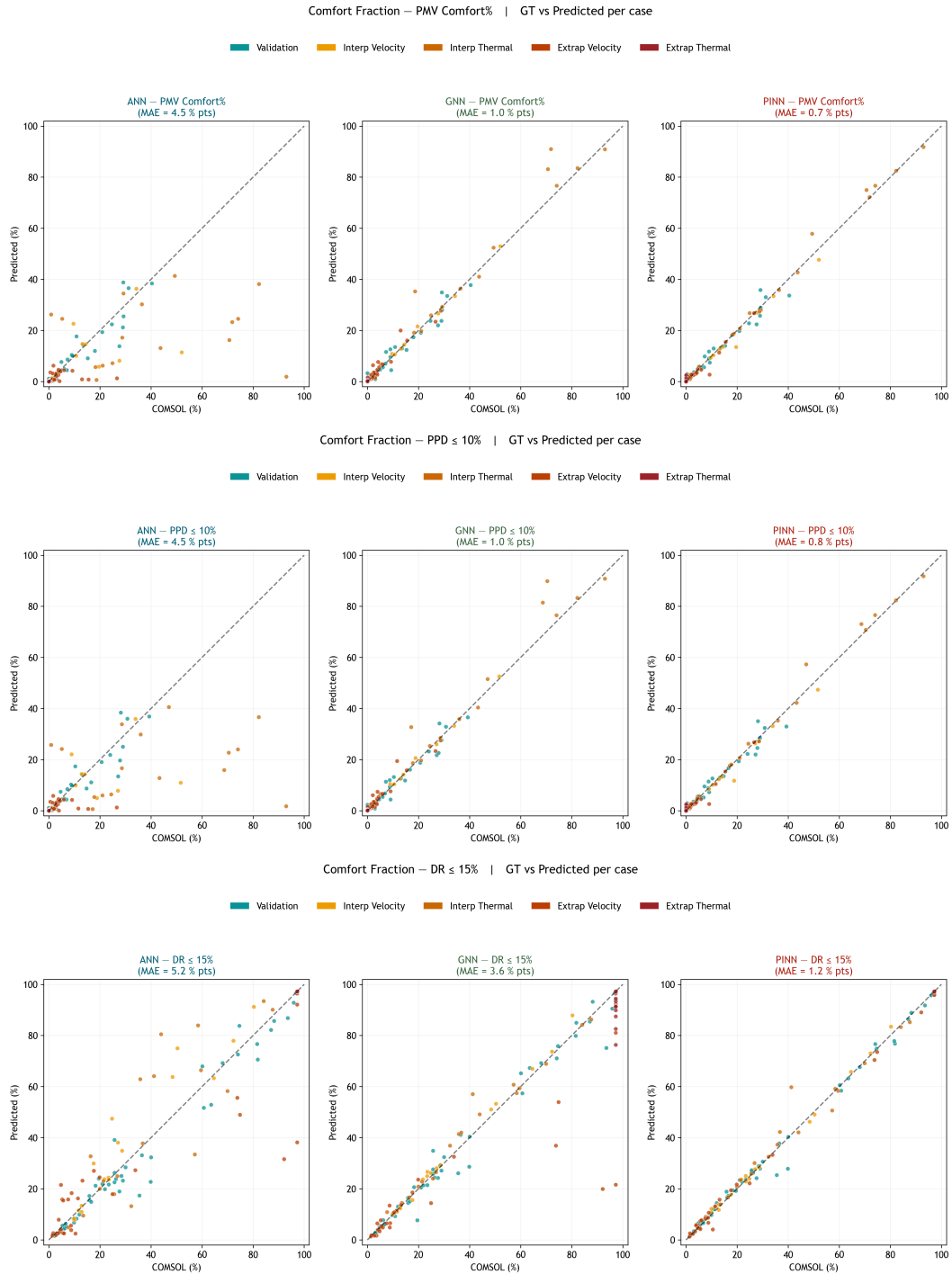


Figure 5.9: Comfort-fraction agreement in the half-partition room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Panel layout as in Figure 5.8.

5.5 Model Ranking and Data Efficiency

The per-split tables condense into two summary views, and a third panel turns to the data-efficiency question. Figure 5.10 reports the variance each model explains, Figure 5.11 counts the per-field winners and plots accuracy against training-set size.

Figure 5.10 shows the coefficient of determination for every model, split, and field, one heatmap per model in each room, with lighter cells marking a higher value and white a perfect fit. The in-range rows are uniformly strong: across the interpolation splits most cells sit above 0.8, and the velocity and temperature cells climb into the high 0.90s for the GNN and PINN. The temperature cell is what separates them. In the open room the PINN's interpolation-temperature R-squared (around 0.98 to 1.00) edges the GNN's (0.95 to 0.99), with the ANN below both and far below on the thermal split; in the half-wall the order flips, the GNN's temperature cells (0.93 to 0.99) leading the PINN's (0.82 to 0.97). The extrapolation rows are where the heatmaps come apart, and the extrapolation-thermal temperature cell shows it most starkly: the PINN's R-squared there is 0.92 in the open room and 0.88 in the half-wall, while the ANN's drops to -1.8 and -3.8 and the GNN's to -2.8 and -6.8 — strongly negative, a fit far worse than simply predicting the field's mean. The out-of-range velocity cells stay highest for the PINN as well. The visual is a PINN whose map stays light across both regimes against an ANN and GNN whose extrapolation rows go dark.

The per-field winner grid in Figure 5.11 counts the same result cell by cell. The PINN is the single best model in the largest share of the field-and-split cells, about two-thirds of the grid in the open room and just over half in the half-wall; the ANN takes most of the remainder and the GNN the smallest share of the three. That the best in-range reconstructor ends with the fewest outright wins is the counting version of the two-regime story: the GNN takes a cluster of interpolation cells and then loses every extrapolation cell, while the PINN wins the whole extrapolation block and enough interpolation cells to lead overall.

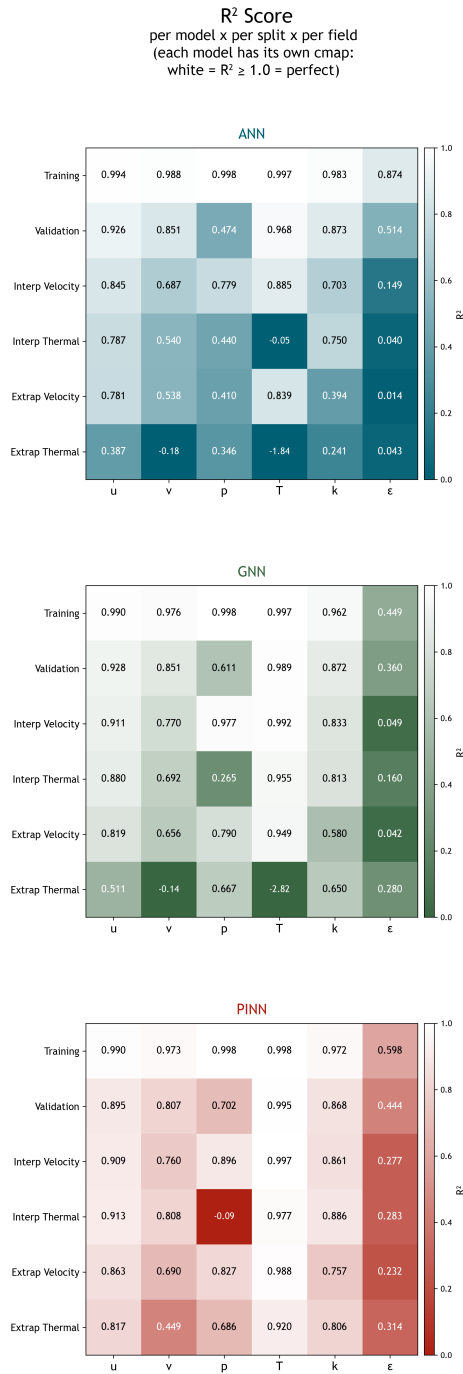
The data-efficiency panels in Figure 5.11 pose the practical question directly: can the PINN, trained on a reduced set of cases, match the ANN and GNN trained on the full set (the nested subsets of Section 4.3.4)? Each cell reports the PINN's error as a percentage of a full-data baseline's, so below 100 percent the PINN wins on less data and above it loses. The result splits along the same seam as the rest. On temperature the PINN trained on a fraction of the data beats the full-data baselines outright: in the open room its thermal-extrapolation error on 150 cases is 32 percent of the ANN's and 56 percent of the GNN's, and on 50 cases it still matches or beats them. The half-wall repeats this, the PINN on 150 cases reaching 29 percent of the ANN's full-data thermal-extrapolation error.

On velocity the result reverses, most sharply against the GNN: the PINN on 50 cases carries about twelve times the GNN’s full-data error on open-room velocity interpolation, since that is the field the GNN reconstructs best. The physics prior buys real data efficiency on the field it constrains, temperature, and gives it back on the field it does not.

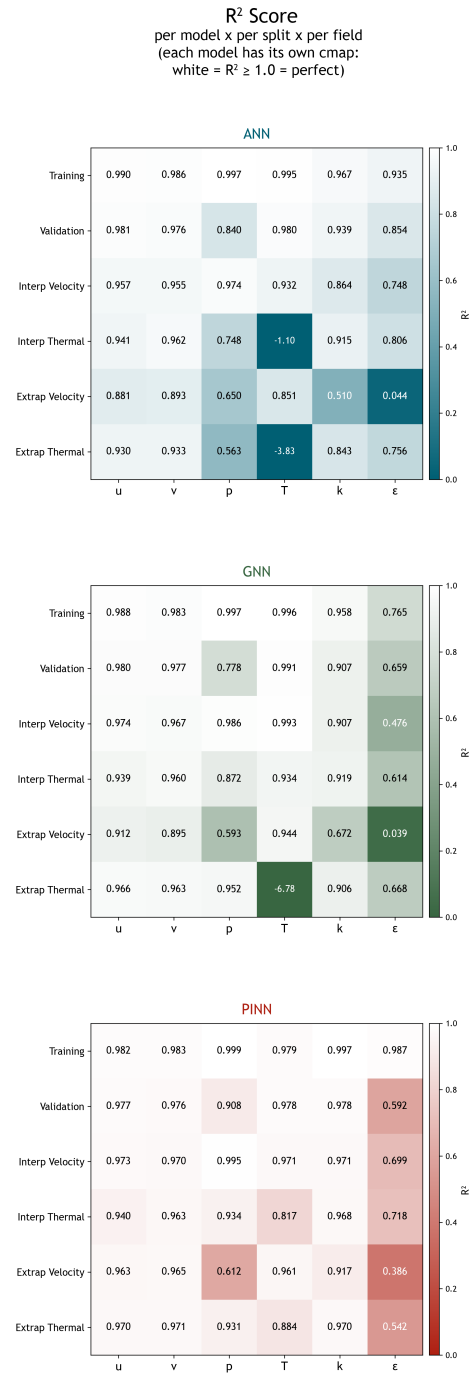
5.6 The Pattern Across Geometries

Reading the two rooms together is the point of showing them side by side, and the shape of the comparison is the same in both. The PINN is the strongest model overall and the only one that holds out of range, by a wide margin on temperature; in range it runs level with the GNN, the two trading the lead between the rooms while both stay ahead of the ANN; and the comfort verdict follows the fields, with the PINN closest in each room. The same order recurs across the chapter’s figures – the field-accuracy tables, the R-squared heatmaps, the comfort errors, and the per-field winner grid all place the PINN ahead overall, with the GNN and ANN trading second place between them.

What shifts with the partition is the magnitude, not the shape. The half-wall’s recirculation raises some errors and the open room’s smoother field lowers others, and two details are geometry-dependent: the in-range temperature lead passes from the PINN in the open room to the GNN in the half-wall, and the GNN’s temperature extrapolation is at its worst in the half-wall. But the ordering that matters holds across both rooms – physics carries the model beyond its training data, graph structure sharpens it within, and both beat the plain baseline on the fields. The quarter- and three-quarter-partition rooms appear in the same set of figures in the appendix; whether the same shape recurs there is the test of how far the pattern generalises. Why it takes this shape, with the physics residual helping out of range while the graph connectivity helps within it, is the question the discussion takes up.

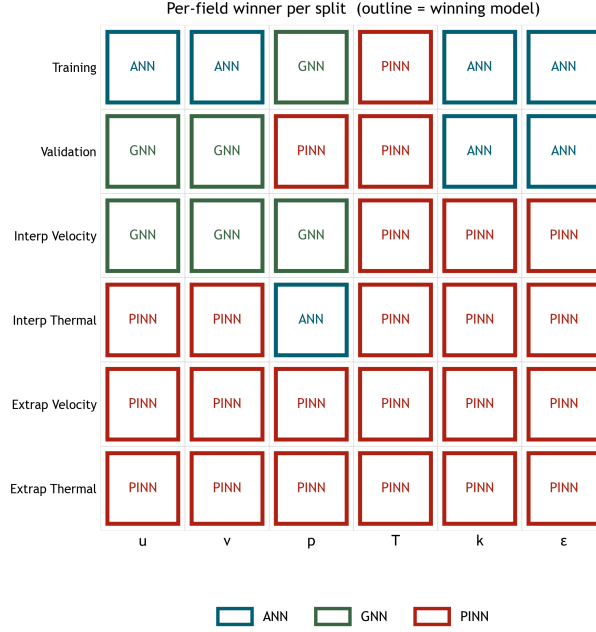


(a) Open room

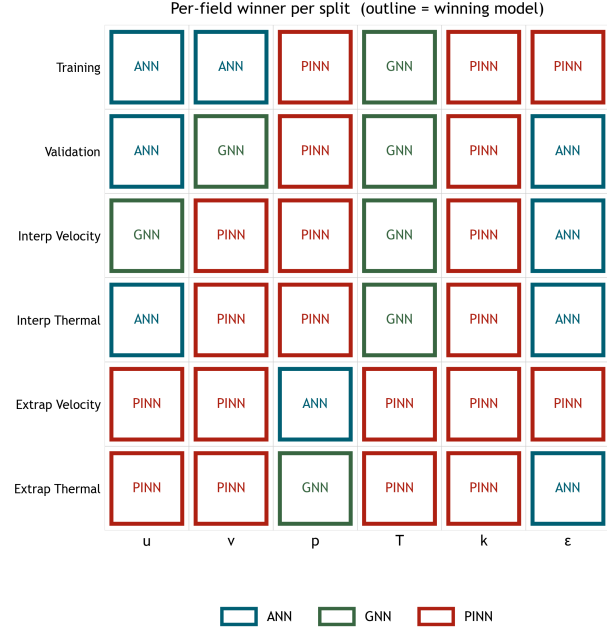


(b) Half-partition room

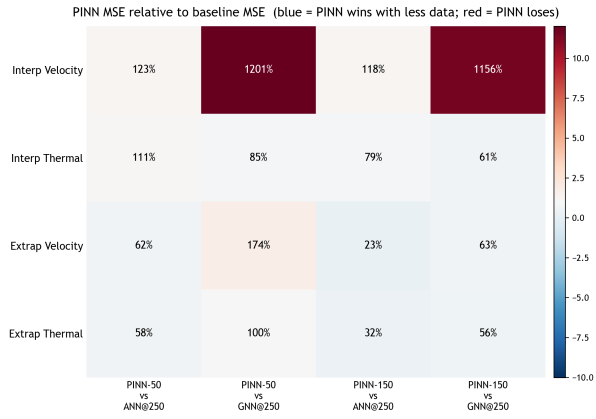
Figure 5.10: Coefficient of determination per model, split, and field, one heatmap per model: (a) the open room and (b) the half-partition room. Lighter cells are higher and white is a perfect fit. The interpolation rows are light for all three models; in the extrapolation-thermal row the temperature cell turns negative for the ANN and GNN – a worse fit than predicting the mean – while the PINN’s stays high.



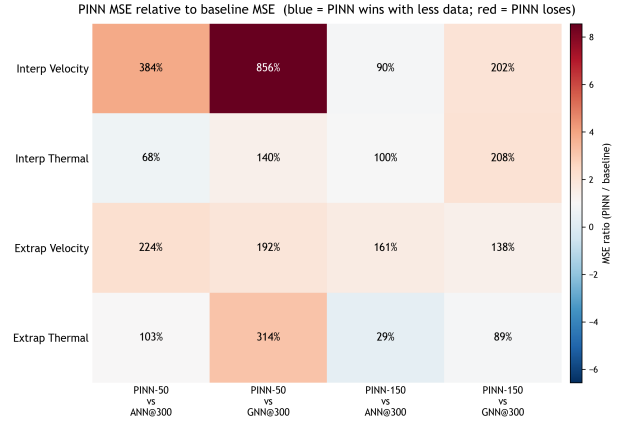
(a) Per-field winner, open room



(b) Per-field winner, half-partition room



(c) PINN on reduced data vs full-data baselines, open room



(d) PINN on reduced data vs full-data baselines, half-partition room

Figure 5.11: Per-field winners and reduced-data comparison. Top row: the single best model for each field and split, (a) the open room and (b) the half-partition room, with the outline colour marking the winner. Bottom row: the PINN's error trained on reduced data as a percentage of the ANN and GNN baselines trained on the full set, (c) the open room and (d) the half-partition room; below 100 percent (blue) the PINN still wins on less data, above it (red) the PINN loses.

6

Discussion

This chapter interprets the results of Chapter 5, following the research questions set out in Section 1.3. The results chapter reported what the figures show. This one asks why they show it, and where a reasonable expectation turns out to be wrong.

6.1 Architecture Performance Patterns

The winner grid and the field tables (Section 5.5, Section 5.3) point to one pattern. The PINN is the strongest model overall and clearly the strongest out of range. In range it runs level with the GNN, which holds a narrow lead on velocity. The ANN trails both. A few reasonable guesses about why each model wins where it does turn out to be wrong, and it is worth taking them one at a time.

The first guess is that the PINN, because it carries explicit physics, should be strongest on the most physics-constrained field, pressure. It is not. Its pressure accuracy is uneven, and in one interpolation case its coefficient of determination drops just below zero (Section 5.5), worse than predicting the field's mean. The reason is that the PINN never enforces a pressure equation directly (subsubsection 4.5.2). Pressure enters only through the momentum residual, and that indirect link gives it no special accuracy. The field the physics does help is temperature, which the energy residual constrains directly.

The second guess runs the other way. The plain data-driven baseline should handle the smooth temperature field well, since a smooth parameter-to-field map is what a regressor fits most easily. The opposite happens. Temperature is the ANN's weakest field by a wide margin, with an interpolation error more than ten times that of the other two models (Section 5.3). The smoothness is misleading. The temperature level depends on the inlet and wall conditions in a way the unconstrained network only approximates, so it smears the field, while the physics residual and the graph's spatial coupling both hold it in place.

The one guess the data confirms is that physics helps most when the model is pushed outside its training data. The PINN wins every extrapolation cell of the winner grid in the open room and all but a few in the half-wall (Section 5.5), and it is the only model whose temperature fit holds up under thermal extrapolation. This is the central result, and the next section looks at it more closely.

A last guess is that the turbulence quantities, k and the dissipation rate ε , should favour the graph model, since spatial coherence matters most there. The data does not support this either. The dissipation rate is the hardest field for every model, with a low coefficient of determination across the board (Section 5.5), and none of the three does well on it. The kinetic energy k is reconstructed solidly by all three, and where it is reconstructed best, that is the PINN, not the GNN. The graph structure helps on velocity, not on the turbulence fields. So the pattern is narrower than the guesses suggest. Physics helps on temperature and on extrapolation. Graph structure helps in range, mostly on velocity. And the plain baseline is weakest not on the rough fields but on the smooth one.

6.2 Generalization Behavior

The sharpest divide in the comparison is between interpolation and extrapolation. Inside the sampled parameter range the models are close. All three are bunched on velocity, and the GNN and PINN are tight on temperature as well; the only in-range gap is the ANN's loose temperature (Section 5.3). Outside the range they come apart, and they come apart for a structural reason. The PINN carries a physics residual that constrains its output everywhere, in range or out. The ANN and GNN have no such constraint and can only extrapolate from the training data. This is the main answer to the generalisation question. Among the three models, the PINN loses the least accuracy when the test cases move outside the training range. This supports the idea that the physics residual helps guide the predicted field when there is less training data to rely on. At the same time, this result should not be credited to the residual alone. The PINN also uses Fourier features and gated layers, which the baseline model does not have (Section 4.5). Without an ablation study that removes these parts one by one, it is not possible to say exactly how much of the improvement comes from the physics loss itself. What the comparison shows clearly is that the physics-informed model, as a full model design, performs best out of range..

The two extrapolation directions are not equally hard. Pushing the inlet velocity past its training range degrades all three models but leaves their fields recognisable. The velocity fit stays positive in velocity extrapolation and, at worst, dips just below zero in the thermal case (Section 5.5). Pushing the supply temperature past its range is where the ANN and GNN break. Their temperature fit turns strongly negative (Section 5.5), worse than guessing the field's mean. The reason is simple.

A network with no thermal constraint, asked for a temperature it never saw in training, falls back toward the temperatures it did see. It saturates near the training range, which is what the extrapolation-thermal maps show (Section 5.2). The velocity field is better fixed by the geometry and inlet the model can see, so the same fallback does less harm.

It would be easy to assume the plain network degrades most out of range, but that is not what happens. On temperature extrapolation the worst model is the GNN, not the ANN (Section 5.3). The same graph structure that sharpens its in-range reconstruction makes its extrapolation the most brittle. This says what each model’s built-in assumption is worth out of range. The graph topology helps fit the sampled range but gives no guide for leaving it; it sharpens the in-range fit, then overfits the training range harder than the plain baseline once the temperature moves past it. Of the three, only the physics residual extends beyond the training data, and only the PINN holds its accuracy there.

Compared against the known failure modes of physics-informed training (Section 3.3), the serious ones did not appear here. Training did not fail to converge, and the physics loss did not swamp the data term to give a smooth but wrong field; the PINN’s in-range accuracy keeps pace with the GNN’s. The weaknesses that did show up are the milder, expected ones: weak pressure, where no pressure equation is enforced, and a poorly resolved dissipation rate, which follows from enforcing only the k -transport residual and not the ε -transport equation (subsubsection 4.5.2). These are limits of this particular loss, not of the method, and they point to where a fuller closure would help.

6.3 Data Efficiency Implications

The data-efficiency panels (Section 5.5) ask the practical question: how much of the expensive CFD data can the physics prior replace? For temperature, a large part of it. A PINN trained on 150 cases reaches about a third of the ANN’s thermal-extrapolation error and just over half of the GNN’s, with both baselines trained on the full set; on 50 cases it still matches or beats them. Each CFD case takes 138 seconds to solve (Section 4.2.1), so on the temperature field, the field that matters most for comfort, the PINN reaches better-than-full-data accuracy on far fewer solves.

This does not hold on every field. On velocity, and most clearly on velocity interpolation, the reduced-data PINN falls well behind the full-data baselines, by about ten times against the GNN, which is the field the GNN reconstructs best (Section 5.5). Physics buys data efficiency where it constrains the field, on temperature, and not where it does not. So the result should be read field by field. If temperature and comfort are the goal, the PINN earns its cost on far fewer cases. If the velocity field itself is the goal, the data-driven models still need the full set.

One thing the experiment does not measure is how the ANN and GNN themselves hold up as their own training sets shrink; it tracks only the PINN's reduced-data accuracy against the full-data baselines. How well the graph model degrades with less data is left open. The literature suggests message-passing models need enough examples to learn stable propagation, and that indoor-airflow surrogates reach a point where more cases stop helping, a result reported by Wei and Ooka (2023). Whether the GNN reaches that point sooner or later than the others is a straightforward extension of this study, not something the present figures settle.

6.4 Implications for Architectural Design Workflows

The first implication is speed. In this study, one COMSOL case takes about 138 seconds to solve (Section 4.2.1). Once the surrogate model is trained, it can predict the field much faster. This makes it more suitable for use during design, where many changes may need to be tested quickly.

A detailed timing study is not included here. Future work should measure the full process more carefully, including model inference, pre-processing, post-processing, and any plugin overhead. The main point in this chapter is the change in workflow. Instead of waiting minutes, or setting up a separate simulation run, the designer can get feedback almost immediately.

This matters because airflow prediction can then become part of the design loop. An architect could move an inlet, change the supply temperature, or test a different option and quickly see how the airflow field and comfort result change. In this way, the surrogate shifts CFD-style feedback from a later consultant report into an earlier and more interactive design process.

The second implication is what gets predicted. The comfort layer is a fixed calculation on the field (Section 4.6), so the surrogate returns PMV, the percentage dissatisfied, and draught rate in the same step as the field, with no extra modelling stage. Airflow and comfort can be worked on together instead of as a field prediction now and a comfort study later. The comfort fractions the PINN returns are the closest to the CFD (Section 5.4), and the comfort fraction is the number a designer acts on.

The third implication is trust, and here the generalisation result becomes a design rule. The surrogate is reliable inside the sampled parameter range; for the ANN and GNN it is not reliable outside it, and a designer cannot be expected to remember which inlet speeds and temperatures were sampled. A usable tool should show that boundary, for instance with a parameter dial that reads green inside the trained range, amber at its edge, and red beyond it, so the prediction comes with its own warning. In the extrapolation ranges tested here, the PINN was the only model that kept its accuracy reasonably well. This makes it the better choice for a design tool that is meant to test options beyond the training cases.

However, this only applies to the extrapolation range studied in this work. If the design moves much further outside that range, none of the surrogate models has been tested there. In those cases, the safer option is to run a full CFD simulation instead of relying on the surrogate prediction.

6.5 Design Principles for Indoor Airflow Surrogates

A few practical guidelines follow from the comparison. They are defaults for the next study of this kind rather than new findings. The first is about features. Fourier features are worth using when the field has sharp gradients near walls or openings (Tancik et al. 2020; S. Wang, H. Wang, and Perdikaris 2021), which is true of every indoor case here; without them a coordinate network blurs the thin shear layers that the comfort calculation depends on most. The second is about when to use physics. Physics-informed training is worth its slower convergence and harder loss balancing when CFD data is scarce and extrapolation matters. When it is not, a plain or graph model reaches the same accuracy inside a dense, well-sampled range for much less trouble.

The third is about structure. Graph architectures pay off when wall-aware spatial coherence matters and the geometry is regular enough to write as a structured graph, which is where the GNN’s in-range velocity lead comes from. On a field with no strong spatial pattern, such as the temperature level, the structure helps less. The last two guidelines are about evaluation. Interpolation and extrapolation should be scored separately, because combining them hides the failure that matters most for design: the collapse out of range, which an averaged score would bury under strong in-range numbers. And field prediction should always be carried through to the comfort metric, because the ISO 7730 non-linearities can shrink or magnify a field error in ways a field-only score does not show. The comfort layer is where the prediction meets the decision.

7

Limitations and Future Work

This work gives a starting point for comparing different deep-learning surrogate models for indoor airflow. At the same time, the study has clear limitations, and these limits affect how far the results can be taken. This chapter discusses those limitations directly and also points to the future work that follows from them. Some limitations come from deliberate choices made to keep the study focused, while others are constraints of the current method, data, or validation. In both cases, they show where later studies can build on this work.

7.1 Limitations

Non-neural surrogate baselines. The comparison centres on three deep-learning paradigms measured against a data-driven neural baseline. A non-neural surrogate — Kriging, radial-basis-function regression, or a POD reduced-order model (Markarian and Azar 2023; Mao et al. 2025) — sits outside the present scope. Adding one would let a later study separate what deep learning specifically contributes from what any sufficiently expressive surrogate would achieve.

Architectural-component ablations. Each architecture is evaluated as a complete unit. Component-level ablations — the PINN with and without Fourier features, with and without curriculum scheduling, with fixed versus adaptive loss weighting, and the GNN with and without wall-aware edge pruning — would attribute the performance differences to individual design choices rather than to the architecture as a whole.

From a 2D slice to 3D geometry. The simulation domain is a fixed 8 m by 4 m two-dimensional rectangle in which only the internal-wall position varies (Section 4.4). Irregular plan shapes, full three-dimensional simulation, and rooms with several internal partitions are the natural geometric extensions.

From steady state to transient flow. The CFD ground truth is a steady-state solution. Transient dynamics – airflow ramp-up after a door opens, time-varying boundary temperatures, occupant-driven disturbances – are the next regime, and the one where the surrogate speedup of Section 6.4 matters most in absolute terms.

The turbulence closure as reference. The CFD reference is the k - ε RANS closure (Section 2.1.2) rather than large-eddy or direct numerical simulation. The surrogates are trained against k - ε output, so any inaccuracy of that closure in the simulated regime – near corners, in strong buoyancy, or in highly anisotropic turbulence (Rodrigues Marques Sakiyama et al. 2021) – is carried into the predictions. A higher-fidelity reference is a future axis of the same study design.

Mesh-independence of the CFD reference. The CFD mesh is fixed at COMSOL’s physics-controlled “Finer” setting and was not varied to confirm grid independence (Section 4.2.5), so no formal mesh-sensitivity or grid-convergence study was run. The reference fields, and the surrogates trained on them, therefore carry whatever discretisation error that resolution holds; a grid-convergence check is a straightforward addition to the same pipeline.

A fuller PINN closure. The PINN enforces the k -transport residual but not the ε -transport equation of the standard k - ε model (subsubsection 4.5.2), a deliberate simplification motivated by the stiffness of ε -transport. As the discussion noted, the dissipation rate is the weakest field for every model; enforcing both transport equations, after the full RANS- k - ε PINN of Pioch et al. (Pioch et al. 2023), is the clearest place that accuracy would improve.

Experimental validation. The COMSOL model was not validated against experimental measurements for the geometries studied; the surrogates approximate the COMSOL output rather than the underlying physical flow. Validating both the CFD reference and the surrogate predictions against a physical mock-up would anchor the whole pipeline to measurement.

Cross-geometry generalisation. Each wall configuration is trained as a separate, independent study (Section 4.4); the robustness claim is that the architecture-comparison pattern recurs across configurations, not that a single model spans them. Training one model to handle several geometries is a direction in its own right.

Broader thermal regimes. The parameter ranges (Section 4.2.4) cover a moderate thermal-comfort regime. Extreme summer and winter conditions, and the boundary-condition extremes that come with them, would widen the envelope the surrogates are asked to cover.

Uncertainty quantification. The predictions are point estimates. Bayesian neural networks or ensemble methods that return a confidence interval alongside each prediction would let a design tool report not just the predicted comfort fraction but the confidence in it – a genuinely useful addition for decision-making.

Extending the comfort layer. The comfort post-process (Section 4.6) holds the personal parameters fixed at sedentary-office values, computes mean radiant temperature from the four in-plane walls only, and evaluates PMV, PPD, and draught rate but not the three further local-discomfort indices of ISO 7730 (ISO 2005). Sweeping clothing and metabolic rate, extending the radiative geometry to a six-wall enclosure (ISO 1998), and adding the remaining indices are each natural extensions of the layer.

7.2 Future Directions

The limitations above point to a coherent programme of extensions, and a few directions reach beyond them. The nearer-term ones extend the comparison itself; the longer-term ones put a mature surrogate to work.

7.2.1 Near-Term Methodological Extensions

3D and transient simulation. Move from 2D steady to 3D unsteady CFD ground truth. This costs substantially more compute per case, but it is the regime where the surrogate speedup matters most in absolute terms.

A mesh-independence study. Repeat a sample of cases at refined and coarsened mesh densities to establish grid convergence of the CFD reference (Section 4.2.5), bounding the discretisation error the surrogates inherit.

Cross-geometry generalisation. Train a single model across wall configurations by adding wall position as an input and exploiting transfer between configurations; the MeshGraphNets node-type encoding (Pfaff et al. 2021) is a natural starting point.

Experimental validation. Validate both the CFD reference and the surrogate predictions against measured airflow from a physical mock-up of the room geometry.

Operator-learning methods. Evaluate DeepONet (Lu et al. 2021) and the Fourier neural operator (Z. Li et al. 2021) against the same framework used here; the closest indoor-airflow precedent is Gao et al. (H. Gao et al. 2024).

Hybrid reduced-order models. Combine the speed of a POD reduced-order model with the parametric coverage of a neural surrogate (Mao et al. 2025), which would also supply the non-neural comparison point noted above.

Multi-fidelity training. Pre-train on cheaper low-resolution simulations and fine-tune on the expensive high-resolution data, to push data efficiency further than the present results.

Comfort sensitivity and richer indices. Sweep clothing insulation and metabolic rate across realistic occupant configurations, extend the radiant calculation to a full enclosure (Fanger 1970; ISO 1998), and add the local-discomfort indices of ISO 7730 (ISO 2005).

7.2.2 Longer-Term Applications

Reinforcement learning for HVAC control. A field surrogate that returns the airflow state near-instantly is the missing piece for closed-loop HVAC control trained by reinforcement learning, where it can serve as the world model for the agent.

A production design tool. Carry the prototype Grasshopper integration (Section 6.4) to a production-quality plugin, with a parameter-envelope overlay and an uncertainty display.

Conclusions

Indoor airflow drives thermal comfort, and using it in architectural design means having a model that is accurate and that fits inside a parametric workflow. CFD provides the accuracy but is too costly to run in a design loop, so the practical question is whether a learned surrogate can stand in for it. A surrogate is only useful if it holds up where a designer needs it: across a range of conditions, including ones outside its training data, and accurately enough that the comfort measures built on its output can be trusted. This paper asked which kind of deep-learning surrogate meets that bar, and why.

The study compared three architectures on one shared problem: a data-driven artificial neural network (ANN), a physics-informed neural network (PINN), and a graph neural network (GNN). All three were trained as surrogates for $k\text{-}\epsilon$ RANS CFD of a ventilated room, across four geometries, at three training-set sizes, and tested on six splits that separate interpolation from extrapolation along the velocity and thermal axes. On top of the predicted fields, an ISO 7730 layer produced PMV, predicted percentage dissatisfied (PPD), and draught rate, so the models could be judged on the comfort verdict a designer uses, not only on the raw fields.

The results answer the four questions the paper set out with. On the first, whether these surrogates are accurate enough to be useful, the answer inside the sampled range is yes. All three reconstruct the fields well, with the GNN and PINN running level and reaching R^2 in the high 0.90s on velocity and temperature. On the second, whether physics-informed training improves generalisation, the answer is clear, and it is the central finding of the paper. Outside the training range the PINN holds its temperature fit at an R^2 of 0.92 in the open room and 0.88 in the half-wall, while the ANN and GNN turn negative, worse than guessing the field's mean. The PINN wins about two-thirds of the field-and-split cells in the open room and just over half in the half-wall, and it takes nearly every extrapolation cell.

On the third question, data efficiency, the PINN is the most efficient model where physics constrains the field. Trained on 150 cases it reaches about a third of the ANN's full-data thermal-extrapolation error and just over half of the GNN's; on velocity, the field the GNN reconstructs best, the data-driven models still need the full set.

On the fourth, whether the predicted fields carry through into comfort, they do, and the comfort ordering follows the field ordering. The PINN's comfort-fraction error is about 1.5 points in the open room and 0.7 in the half-wall, the closest of the three to the CFD, which makes its comfort verdict the most trustworthy.

These results rest on a handful of contributions. The three architectures were compared on one shared problem and dataset, so the comparison is direct rather than stitched together from separate studies. That comparison ran across four room geometries, from an open room to an internal partition at the quarter, half, and three-quarter points. Interpolation and extrapolation were kept apart, with the velocity and thermal axes tested on their own across six splits, which is what made the out-of-range collapse visible instead of hidden inside an average. Data efficiency was measured at three training-set sizes, setting accuracy against the cost of generating CFD data. The field surrogate was coupled to an ISO 7730 comfort layer that carries the prediction through to PMV, PPD, and Draught Rate. And a prototype Grasshopper plugin showed the trained surrogates running inside a working design environment.

These findings come with limits worth stating alongside them. The accuracy reported here is accuracy relative to the COMSOL $k-\varepsilon$ RANS reference, not to physical measurement; the surrogates reproduce that CFD solution rather than the real flow, and the closure's own error is carried into their predictions. The PINN's advantage is also specific rather than uniform: it is strongest on thermal extrapolation and on the comfort outputs that lean on temperature, while in range it only matches the GNN and on velocity it does not lead. And because the physics-informed model differs from the baseline in its Fourier features and gated layers as well as its loss, the share of its advantage that comes from the physics residual itself cannot be separated without an ablation. The clearest next steps follow from these limits: validating both the CFD reference and the surrogates against physical measurements, a mesh-sensitivity study of the CFD itself, and a wider range of comfort conditions than the single mechanically-ventilated regime studied here. Chapter 7 sets these out in full.

The broader point is about where physics belongs in a surrogate. A purely data-driven model is enough when the design stays inside a densely sampled range, and a graph model sharpens that fit. But the cases that matter most in early design are often the unfamiliar ones, the inlet or supply temperature a designer has not tried before, and there the physics-informed model is the only one that stays reliable. A surrogate that carries its physics is what lets a CFD-grade airflow prediction, and the comfort verdict that depends on it, sit inside the design loop rather than in a separate study after the fact. That is the bridge this work set out to build between CFD and architectural design.

References

- Anderson, John D. (1995). *Computational Fluid Dynamics: The Basics with Applications*. New York: McGraw-Hill.
- ANSYS, Inc. (2024). *ANSYS Fluent Theory Guide, Release 2024 R1*. Canonsburg, PA.
- ASHRAE (2023). *ANSI/ASHRAE Standard 55-2023 Thermal Environmental Conditions for Human Occupancy*. Atlanta, GA.
- Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML]. URL: <https://arxiv.org/abs/1607.06450>.
- Battaglia, Peter W., Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. (2018). *Relational inductive biases, deep learning, and graph networks*. arXiv: 1806.01261.
- Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer.
- Björling, Mikael (2021). “Building for Sustainable Ventilation and Air Quality”. In: *Urban Transition - Perspectives on Urban Systems and Environments*. Ed. by Marita Wallhagen and Mathias Cehlin. London: IntechOpen. Chap. 10. DOI: 10.5772/intechopen.97719.
- Botarelli, Tommaso, Marco Fanfani, Paolo Nesi, and Lorenzo Pinelli (2025). “Using Physics-Informed neural networks for solving Navier-Stokes equations in fluid dynamic complex scenarios”. In: *Engineering Applications of Artificial Intelligence* 148, p. 110347. DOI: 10.1016/j.engappai.2025.110347.
- Calzolari, Giovanni and Wei Liu (2021). “Deep learning to replace, improve, or aid CFD analysis in built environment applications: A review”. In: *Building and Environment* 206, p. 108315. DOI: 10.1016/j.buildenv.2021.108315.
- Caron, Clément, Philippe Lauret, and Alain Bastide (2025). “Machine Learning to speed up Computational Fluid Dynamics engineering simulations for built environments: A review”. In: *Building and Environment* 267, p. 112229. DOI: 10.1016/j.buildenv.2024.112229.
- Chen, Tianle, Ruibin Li, Xuesong Hu, Bingyang Zhang, Yuanyuan Liu, Liangzhu (Leon) Wang, and Naiping Gao (2025). “Machine learning as CFD surrogate models for rapid prediction of building-related physical fields: A review of methods and state-of-the-art”. In: *Building and Environment* 285, p. 113667. DOI: 10.1016/j.buildenv.2025.113667.

- Cheng, Y., J. Niu, and N. Gao (2012). “Thermal comfort models: A review and numerical investigation”. In: *Building and Environment* 47, pp. 13-22. DOI: 10.1016/j.buildenv.2011.05.011.
- Chung, Jihoon, Nastaran Shahmansouri, Rhys Goldstein, James Stoddart, and John Locke (2023). *Sustainability through Optimal Design of Buildings for Natural Ventilation using Updated Comfort and Occupancy Models*. arXiv: 2312.06498 [cs.CE].
- Clay Mathematics Institute (2000). *Navier-Stokes Equation*. Millennium Prize Problems. URL: <https://www.claymath.org/millennium/navier-stokes-equation/>.
- COMSOL AB (2026). *COMSOL Multiphysics, Version 6.3.0.420 – CFD Module and Heat Transfer Module*. Stockholm, Sweden. URL: <https://www.comsol.com>.
- Cybenko, George (1989). “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303-314. DOI: 10.1007/BF02551274.
- Dauphin, Yann N., Angela Fan, Michael Auli, and David Grangier (2017). “Language Modeling with Gated Convolutional Networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*.
- Eivazi, Hamidreza, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa (2022). “Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations”. In: *Physics of Fluids* 34.7, p. 075117. DOI: 10.1063/5.0095270.
- European Committee for Standardization (2019). *EN 16798-1:2019 Energy Performance of Buildings – Ventilation for Buildings – Part 1: Indoor Environmental Input Parameters for Design and Assessment of Energy Performance of Buildings*. Brussels, Belgium.
- Fanger, P. O. (1970). *Thermal comfort: Analysis and applications in environmental engineering*. Copenhagen: Danish Technical Press.
- Fanger, P. O., A. K. Melikov, H. Hanzawa, and J. Ring (1988). “Air turbulence and sensation of draught”. In: *Energy and Buildings* 12.1, pp. 21-39. DOI: 10.1016/0378-7788(88)90053-9.
- Fey, Matthias and Jan Eric Lenssen (2019). “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
- Gao, Hu, Weixin Qian, Jiankai Dong, and Jing Liu (2024). “Rapid prediction of indoor airflow field using operator neural network with small dataset”. In: *Building and Environment* 251, p. 111175. DOI: 10.1016/j.buildenv.2024.111175.
- Gilmer, Justin, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl (2017). “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1263-1272.

- Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 249-256.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Cambridge, MA: MIT Press. URL: <http://www.deeplearningbook.org>.
- Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He (2017). “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”. In: *arXiv preprint arXiv:1706.02677*. URL: <http://arxiv.org/abs/1706.02677>.
- Guo, Hongshan, Kanxuan He, Yongqiang Luo, and Yu Chang (2025). “Physics-Informed Neural Networks for robust thermal comfort prediction: Overcoming data quality limitations through physiological constraints”. In: *Building and Environment* 285, p. 113588. DOI: 10.1016/j.buildenv.2025.113588.
- Hamilton, William L., Rex Ying, and Jure Leskovec (2017). “Inductive representation learning on large graphs”. In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 1024-1034. URL: <http://arxiv.org/abs/1706.02216>.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778. DOI: 10.1109/CVPR.2016.90.
- Hendrycks, Dan and Kevin Gimpel (2016). “Gaussian Error Linear Units (GELUs)”. In: *arXiv preprint arXiv:1606.08415*. URL: <http://arxiv.org/abs/1606.08415>.
- Hennig, John (2024). *MPh: Pythonic Scripting Interface for COMSOL Multiphysics*. Python package, version 1.3.x, MIT licence. URL: <https://github.com/MPh-py/MPh>.
- Hinze, J. O. (1975). *Turbulence*. 2nd ed. New York: McGraw-Hill.
- Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5, pp. 359-366. DOI: 10.1016/0893-6080(89)90020-8.
- International Organization for Standardization (1998). *ISO 7726:1998 Ergonomics of the Thermal Environment – Instruments for Measuring Physical Quantities*. Geneva, Switzerland.
- (2005). *ISO 7730:2005 Ergonomics of the thermal environment – Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria*.
- Jagtap, Ameya D., Kenji Kawaguchi, and George Em Karniadakis (2020a). “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks”. In: *Journal of Computational Physics* 404, p. 109136. DOI: 10.1016/j.jcp.2019.109136.

- Jagtap, Ameya D., Kenji Kawaguchi, and George Em Karniadakis (2020b). “Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks”. In: *Proceedings of the Royal Society A* 476.2239, p. 20200334. DOI: 10.1098/rspa.2020.0334.
- Karniadakis, George Em, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang (2021). “Physics-informed machine learning”. In: *Nature Reviews Physics* 3.6, pp. 422-440. DOI: 10.1038/s42254-021-00314-5.
- Kays, W. M. (1994). “Turbulent Prandtl number — where are we?” In: *ASME Journal of Heat Transfer* 116.2, pp. 284-295. DOI: 10.1115/1.2911398.
- Kays, W. M. and M. E. Crawford (1993). *Convective Heat and Mass Transfer*. 3rd ed. New York: McGraw-Hill.
- Kochkov, Dmitrii, Jamie A. Smith, Ayta Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer (2021). “Machine learning-accelerated computational fluid dynamics”. In: *Proceedings of the National Academy of Sciences* 118.21, e2101784118. DOI: 10.1073/pnas.2101784118.
- Koskela, H., J. Heikkinen, R. Niemelä, and T. Hautalampi (2001). “Turbulence correction for thermal comfort calculation”. In: *Building and Environment* 36.2, pp. 247-255. DOI: 10.1016/S0360-1323(00)00002-0.
- Krishnapriyan, Aditi, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W. Mahoney (2021). “Characterizing possible failure modes in physics-informed neural networks”. In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 26548-26560. URL: <https://arxiv.org/abs/2109.01050>.
- Lauder, B. E. and D. B. Spalding (1974). “The numerical computation of turbulent flows”. In: *Computer Methods in Applied Mechanics and Engineering* 3.2, pp. 269-289. DOI: 10.1016/0045-7825(74)90029-2.
- Li, Zongyi, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar (2021). “Fourier neural operator for parametric partial differential equations”. In: *International Conference on Learning Representations (ICLR)*. arXiv: 2010.08895.
- Loshchilov, Ilya and Frank Hutter (2017). “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations (ICLR)*. URL: <http://arxiv.org/abs/1608.03983>.
- (2019). “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations (ICLR)*. URL: <http://arxiv.org/abs/1711.05101>.
- Lu, Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis (2021). “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature Machine Intelligence* 3.3, pp. 218-229. DOI: 10.1038/s42256-021-00302-5.

- Ma, Zhihao, Gang Jiang, Yuqing Hu, and Jianli Chen (2025). "A review of physics-informed machine learning for building energy modeling". In: *Applied Energy* 381, p. 125169. DOI: 10.1016/j.apenergy.2024.125169.
- Mao, Rui, Yuer Lan, Linfeng Liang, Tao Yu, Minhao Mu, Wenjun Leng, and Zhengwei Long (2025). "Rapid CFD Prediction Based on Machine Learning Surrogate Model in Built Environment: A Review". In: *Fluids* 10, p. 193. DOI: 10.3390/fluids10080193.
- Markarian, Elin and Elie Azar (2023). "Predicting energy consumption and thermal comfort in buildings using a hybrid machine learning and building performance simulation approach". In: *Proceedings of the 18th IBPSA Conference*. Shanghai, China. DOI: 10.26868/25222708.2023.1398.
- McClenny, Levi and Ulisses Braga-Neto (2020). "Self-adaptive physics-informed neural networks using a soft attention mechanism". In: *arXiv preprint arXiv:2009.04544*. URL: <https://arxiv.org/abs/2009.04544>.
- McKay, M. D., R. J. Beckman, and W. J. Conover (1979). "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2, pp. 239-245. DOI: 10.1080/00401706.1979.10489755.
- Mitchell, D. (1974). "Convective heat transfer from man and other animals". In: *Journal of Applied Physiology* 37.3, pp. 412-418.
- Modest, M. F. (2013). *Radiative Heat Transfer*. 3rd ed. Oxford: Academic Press.
- Nishi, Y. and A. P. Gagge (1970). "Direct evaluation of convective heat transfer coefficient by naphthalene sublimation". In: *Journal of Applied Physiology* 29.6, pp. 830-838. DOI: 10.1152/jappl.1970.29.6.830.
- Nocedal, Jorge and Stephen J. Wright (2006). *Numerical Optimization*. 2nd. New York: Springer.
- Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1310-1318. URL: <http://arxiv.org/abs/1211.5063>.
- Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 8024-8035. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

- Pfaff, Tobias, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia (2021). “Learning Mesh-Based Simulation with Graph Networks”. In: *International Conference on Learning Representations (ICLR)*. URL: <https://arxiv.org/abs/2010.03409>.
- Pioch, Fabian, Jan Hauke Harmening, Andreas Maximilian Müller, Franz-Josef Peitzmann, Dieter Schramm, and Ould el Moctar (2023). “Turbulence Modeling for Physics-Informed Neural Networks: Comparison of Different RANS Models for the Backward-Facing Step Flow”. In: *Fluids* 8.2, p. 43. DOI: 10.3390/fluids8020043.
- Pope, Stephen B. (2000). *Turbulent Flows*. Cambridge, UK: Cambridge University Press.
- Prechelt, Lutz (1998). “Early Stopping — But When?” In: *Neural Networks: Tricks of the Trade*. Ed. by Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller. Vol. 1524. LNCS. Springer, pp. 55-69. DOI: 10.1007/978-3-642-35289-8_5.
- Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis (2019). “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378, pp. 686-707. DOI: 10.1016/j.jcp.2018.10.045.
- Rodrigues Marques Sakiyama, Nayara, Jurgen Frick, Timea Bejat, and Harald Garrecht (2021). “Using CFD to Evaluate Natural Ventilation through a 3D Parametric Modeling Approach”. In: *Energies* 14. DOI: 10.3390/en14082197.
- Sanchez-Gonzalez, Alvaro, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia (2020). “Learning to Simulate Complex Physics with Graph Networks”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*.
- Sevilgen, G. and M. Kilic (2011). “Numerical analysis of air flow, heat transfer, moisture transport and thermal comfort in a room heated by two-panel radiators”. In: *Energy and Buildings* 43.1, pp. 137-146. DOI: 10.1016/j.enbuild.2010.08.034.
- Shao, Xinhe, Zhao Liu, Shengwei Zhang, Zhixin Zhao, and Cheng Hu (2023). “PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh”. In: *Building and Environment* 232, p. 110056. DOI: 10.1016/j.buildenv.2023.110056.
- Shazeer, Noam (2020). “GLU Variants Improve Transformer”. In: *arXiv preprint arXiv:2002.05202*. URL: <https://arxiv.org/abs/2002.05202>.
- Son, Jiin, Jihoon Kim, and Junemo Koo (2024). “Analysis of ventilation and infiltration rates using physics-informed neural networks: Impact of space operation and meteorological factors”. In: *Building and Environment* 267, p. 112249. DOI: 10.1016/j.buildenv.2024.112249.

- Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15, pp. 1929-1958. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.
- Stamou, A. and I. Katsiris (2006). “Verification of a CFD model for indoor airflow and heat transfer”. In: *Building and Environment* 41.9, pp. 1171-1181. DOI: 10.1016/j.buildenv.2005.06.029.
- Tancik, Matthew, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng (2020). “Fourier features let networks learn high frequency functions in low dimensional domains”. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 7537-7547. URL: <https://arxiv.org/abs/2006.10739>.
- Tartarini, F. and S. Schiavon (2020). “pythermalcomfort: A Python package for thermal comfort research”. In: *SoftwareX* 12, p. 100578. DOI: 10.1016/j.softx.2020.100578.
- Wang, Sifan, Bowen Li, Yuhao Chen, and Paris Perdikaris (2024). *PirateNets: Physics-Informed Deep Learning with Residual Adaptive Networks*. arXiv: 2402.00326. URL: <https://arxiv.org/abs/2402.00326>.
- Wang, Sifan, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris (2023). *An expert’s guide to training physics-informed neural networks*. arXiv: 2308.08468. URL: <https://arxiv.org/abs/2308.08468>.
- Wang, Sifan, Yujun Teng, and Paris Perdikaris (2021). “Understanding and mitigating gradient pathologies in physics-informed neural networks”. In: *SIAM Journal on Scientific Computing* 43.5, A3055-A3081. DOI: 10.1137/20M1318043.
- Wang, Sifan, Hanwen Wang, and Paris Perdikaris (2021). “On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 384, p. 113938. DOI: 10.1016/j.cma.2021.113938.
- Wei, Chenghao and Ryoza Ooka (2023). “Indoor airflow field reconstruction using physics-informed neural network”. In: *Building and Environment* 242, p. 110563. DOI: 10.1016/j.buildenv.2023.110563.
- Weigand, B., J. R. Ferguson, and M. E. Crawford (1997). “An Extended Kays and Crawford Turbulent Prandtl Number Model”. In: *International Journal of Heat and Mass Transfer* 40.17, pp. 4191-4196. DOI: 10.1016/S0017-9310(97)00084-7.
- Wilcox, David C. (2006). *Turbulence Modeling for CFD*. 3rd. La Cañada, CA: DCW Industries.
- Wu, Chenxi, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu (2023). “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 403, p. 115671. DOI: 10.1016/j.cma.2022.115671.

Zhao, Chi, Feifei Zhang, Wenqiang Lou, Xi Wang, and Jianyong Yang (2024). "A comprehensive review of advances in physics-informed neural networks and their applications in complex fluid dynamics". In: *Physics of Fluids* 36, p. 101301. doi: 10.1063/5.0226562.

A

Hyperparameters and Training Details

This appendix documents the full hyperparameter configuration for the three surrogate architectures, the optimisation settings used during training, the hardware on which training was performed, and the wall-clock training time for every configuration.

A.1 Architectural Hyperparameters

Table A.1: Architectural hyperparameters for the three surrogate models. Cross-references: ANN Section 4.5.1, PINN Section 4.5.2, GNN Section 4.5.3.

Hyperparameter	ANN	PINN	GNN
Input dimension	10	10	10
Output dimension	6	6	6
Hidden width	256	256	128
Hidden layers	6	5 gated blocks	4 SAGEConv blocks
Activation	GELU	AdaptiveGELU (L-LAAF)	GELU
Dropout	0.1	—	0.1
Fourier features	—	$m=16, \sigma=3.0$	—
Skip connections	none	input-skip projection	identity per block
Normalisation	—	—	LayerNorm per block
Initialisation	Xavier-uniform	Xavier-uniform	Xavier on encoder/decoder
Total parameters	333 318	696 844	151 302

A.2 Optimisation and Schedule

All three architectures use AdamW (Loshchilov and Hutter 2019) with base learning rate 10^{-3} and weight decay 10^{-4} .

Table A.2: Optimisation, scheduling, and training-loop hyperparameters.

Hyperparameter	ANN	PINN	GNN
Optimiser	AdamW (Loshchilov and Hutter 2019)	AdamW	AdamW
Base learning rate	10^{-3}	10^{-3}	10^{-3}
Weight decay	10^{-4}	10^{-4}	10^{-4}
LR schedule	ReduceLRonPlateau	warmup + cosine (Goyal et al. 2017; Loshchilov and Hutter 2017)	warmup + plateau (Goyal et al. 2017)
LR floor	10^{-7}	10^{-7}	10^{-7}
Gradient clipping	none	$\ \nabla\ \leq 1.0$ (Pascanu, Mikolov, and Bengio 2013)	none
Batch size	8192 points	full-batch collocation	8 graphs
Maximum epochs	1500	6000	1500
Early stopping	patience 250 (Prechelt 1998)	not applicable	patience 300 (Prechelt 1998)

A.3 PINN-Specific Settings

Fixed residual scales. Continuity scale $U_{\text{REF}}/L_{\text{REF}}$, momentum scale $U_{\text{REF}}^2/L_{\text{REF}}$, energy scale $U_{\text{REF}}T_{\text{REF}}/L_{\text{REF}}$, k -transport scale $U_{\text{REF}}k_{\text{REF}}/L_{\text{REF}}$, with $U_{\text{REF}} = 0.5 \text{ m s}^{-1}$, $L_{\text{REF}} = 8 \text{ m}$, $T_{\text{REF}} = 15 \text{ K}$, $k_{\text{REF}} = 0.01 \text{ m}^2 \text{ s}^{-2}$ (Pope 2000).

Curriculum schedule start epochs. Data 0 (always on); continuity 200; soft-Dirichlet BC 300; momentum 600; energy 1000; k -transport 1500. Each term is ramped linearly from weight 0 to target weight over 400 epochs from its start point (Krishnapriyan et al. 2021).

Collocation sampling. Uniform random collocation points are re-sampled every epoch over the spatial domain $\times$ training-parameter set (Wu et al. 2023).

A.4 GNN-Specific Settings

The SAGEConv operator (Hamilton, Ying, and Leskovec 2017; Fey and Lenssen 2019) is used with default settings: `aggr='mean'`, `root_weight=True`, `normalize=False`, no neighbourhood sampling. Mini-batching uses PyTorch Geometric’s `DataLoader` with batch size 8 graphs per step.

A.5 Hardware

All training and evaluation reported in this paper was performed on a single workstation: NVIDIA RTX 5070 GPU (12 GB VRAM), AMD Ryzen 9 7800X CPU, 64 GB system RAM. Operating system and software-stack versions (Python, PyTorch, PyTorch Geometric, CUDA) are listed alongside the released source code repository.

A.6 Approximate Training Times

Table A.3 gives the wall-clock training time for every combination of architecture, dataset size, and wall configuration, in minutes. The ordering is wide and holds across configurations. The GNN trains fastest, in six to about seventy minutes. The ANN takes longer, from twenty minutes up to roughly fourteen hours on the full set. The PINN is the most expensive by a large margin, from about eleven hours on the smallest set to nearly three days at full data in the half-wall room. This is a one-time training cost; once trained, all three models run inference at the same low cost. The PINN’s expense follows from its longer schedule and the physics residuals evaluated at every step (Table A.2), and it is the cost that its data efficiency partly offsets, since the model reaches useful temperature accuracy on far fewer cases.

Table A.3: Wall-clock training time in minutes, by architecture, dataset size, and wall configuration.

Model	Cases	Open	$0.25 L_{\text{room}}$	$0.50 L_{\text{room}}$	$0.75 L_{\text{room}}$
ANN	50	30	20	39	41
	150	118	190	237	112
	Full	374	503	831	344
PINN	50	709	652	1049	1171
	150	1389	1367	1621	1683
	Full	2306	2847	4380	3000
GNN	50	6	6	10	9
	150	8	9	24	30
	Full	46	30	65	68

B

Per-Configuration Results

This appendix reproduces the evaluation of Appendix 5 for the two room geometries not shown there in full: the quarter-partition room ($x_{\text{wall}} = 0.25 L_{\text{room}}$) and the three-quarter-partition room ($x_{\text{wall}} = 0.75 L_{\text{room}}$) of Section 4.4. The figures use the same layout, ordering, and colour conventions as the main results, with the quarter-partition room shown first and the three-quarter-partition room second in each figure. Captions are kept brief and point to the corresponding main-text figure. Read alongside the open- and half-partition figures of Appendix 5, these let the cross-configuration robustness claim of Section 5.6 be checked against the full evidence.

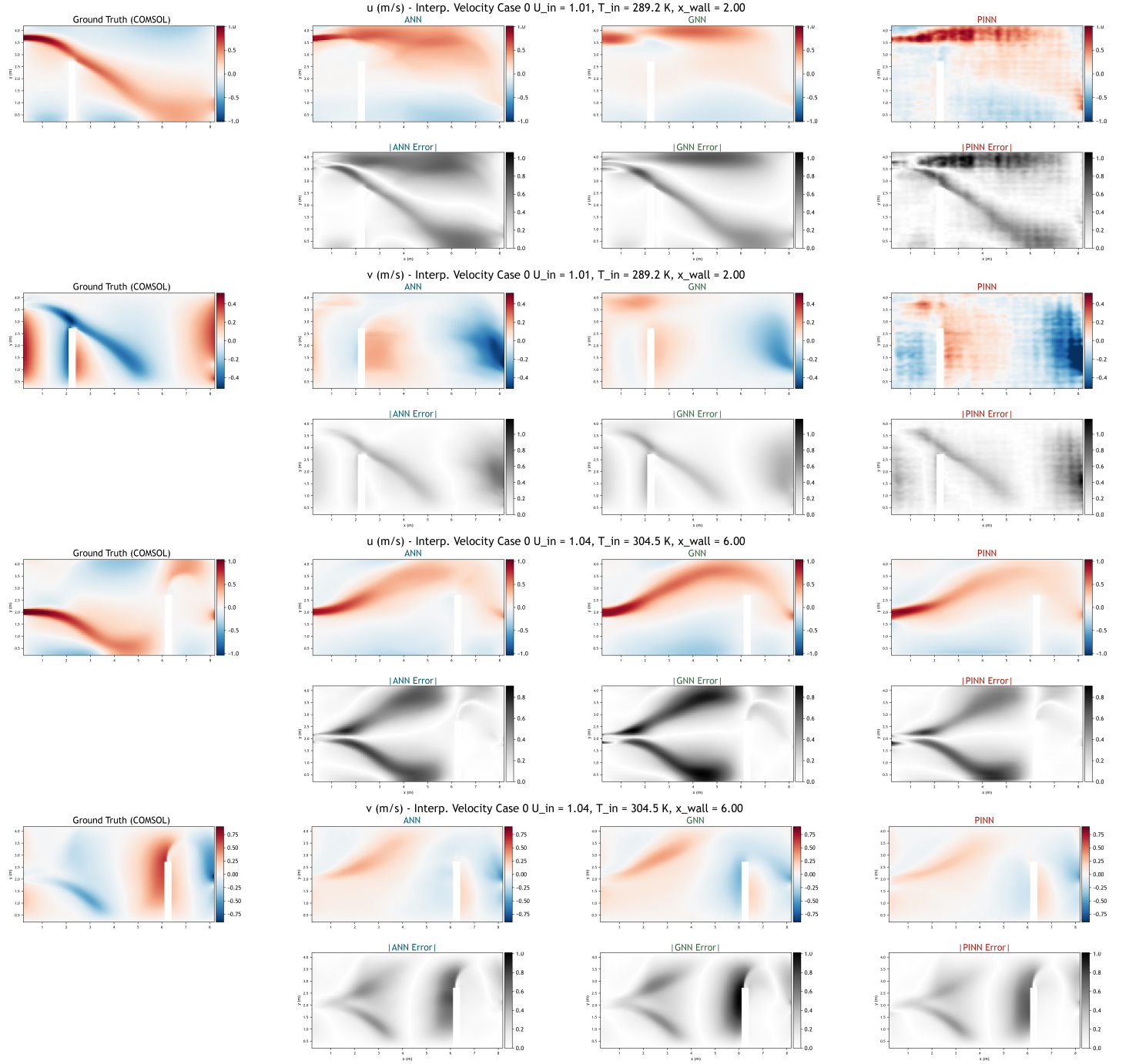


Figure B.1: Velocity in the interpolation-velocity case: streamwise velocity u (m/s) and vertical velocity v (m/s) for the quarter-partition room, then the same two for the three-quarter-partition room. Panel layout as in Figure 5.1.

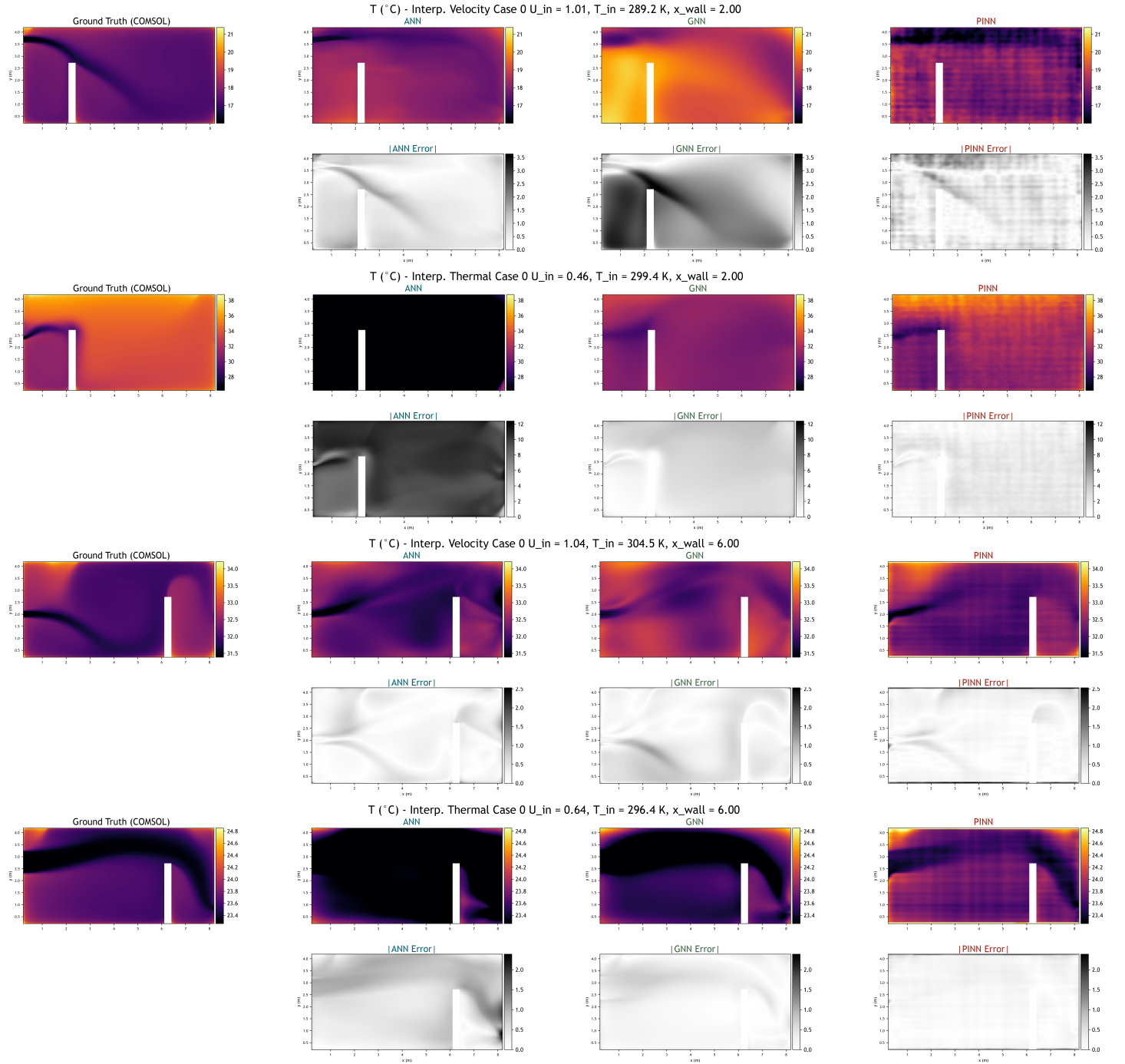


Figure B.2: Temperature T (°C) in the interpolation-velocity and interpolation-thermal cases for the quarter-partition room, then the three-quarter-partition room. Panel layout as in Figure 5.2.

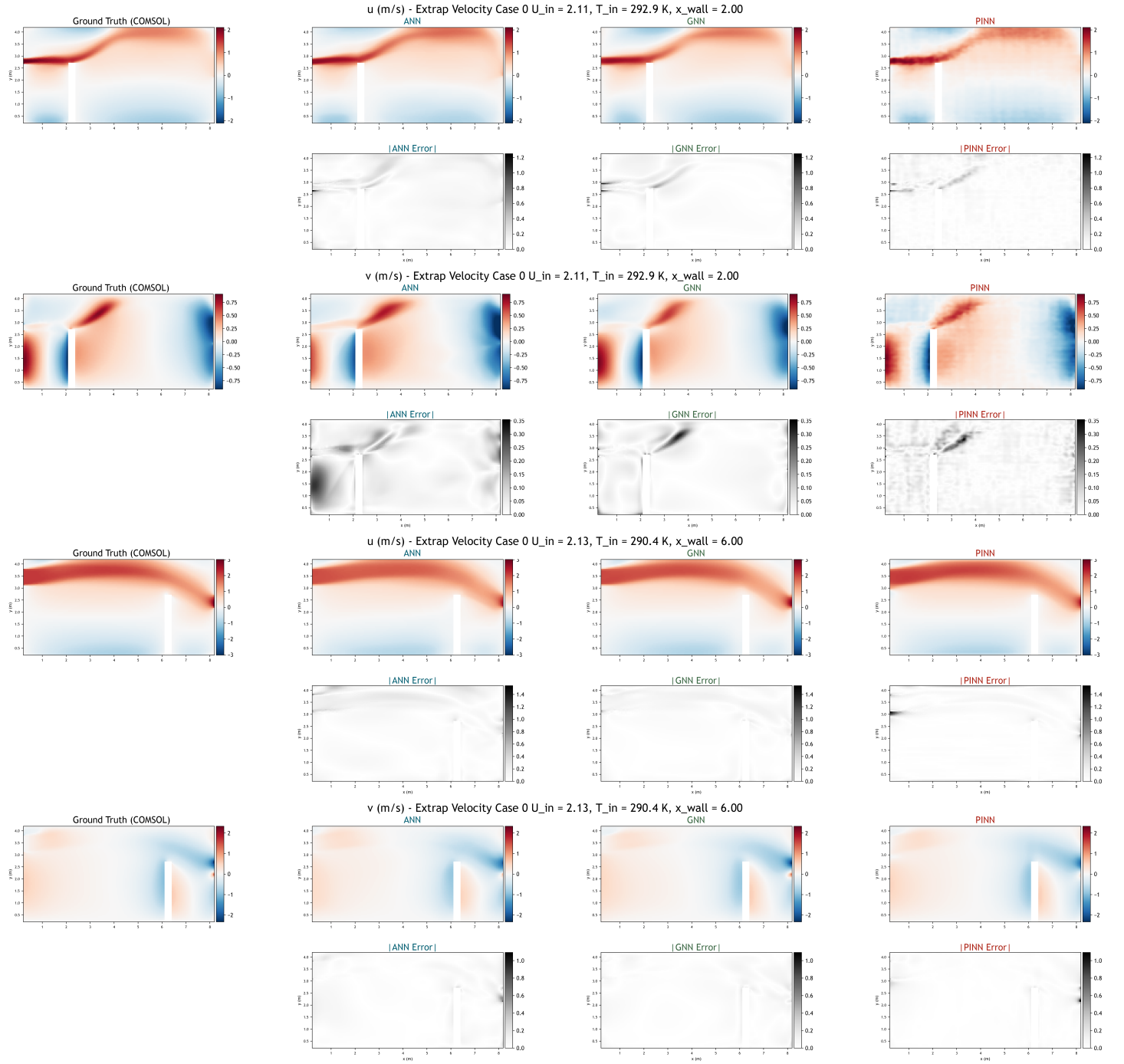


Figure B.3: Velocity in the extrapolation-velocity case: streamwise velocity u (m/s) and vertical velocity v (m/s) for the quarter-partition room, then the three-quarter-partition room. Panel layout as in Figure 5.3.

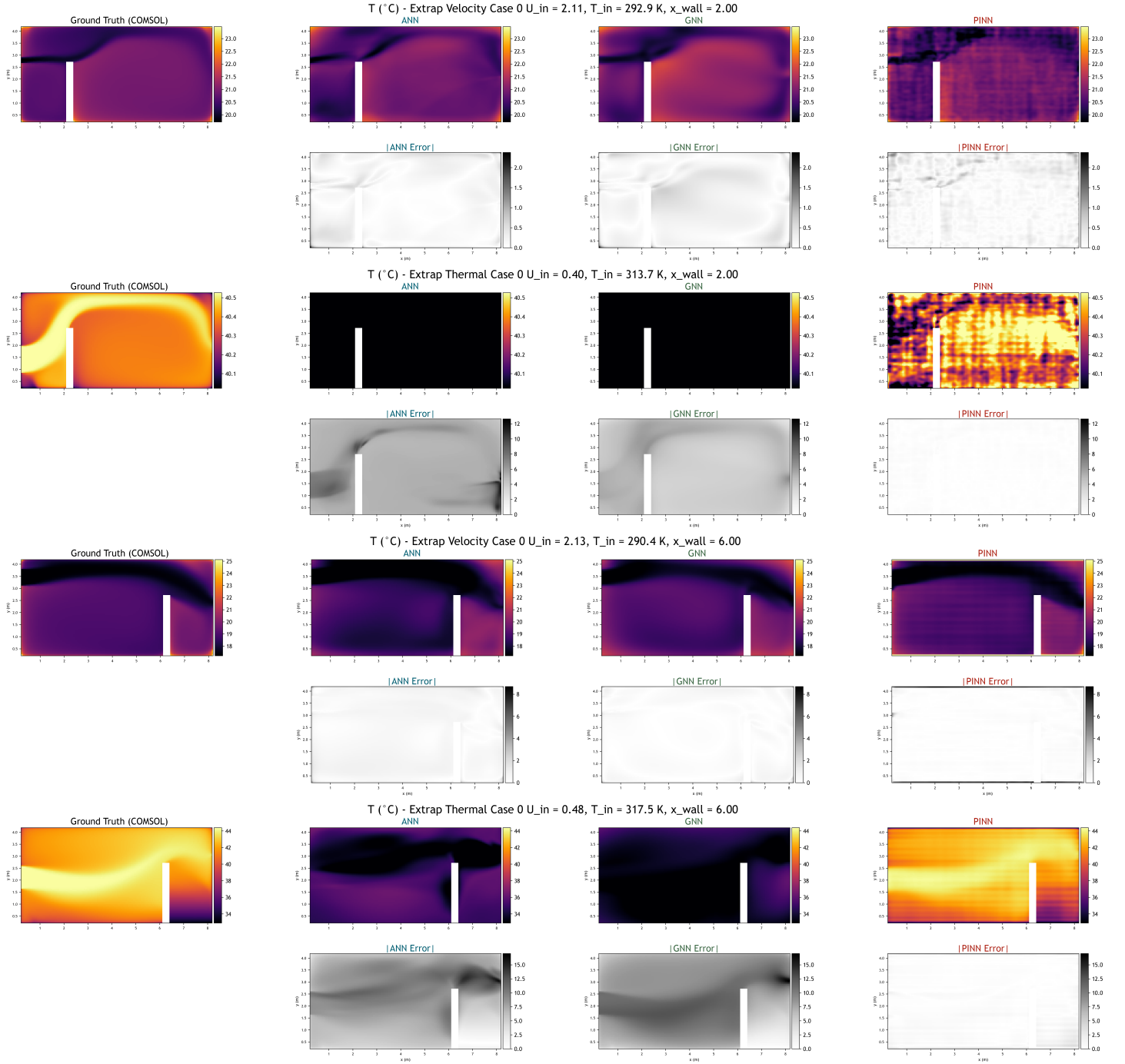


Figure B.4: Temperature T (°C) in the extrapolation-velocity and extrapolation-thermal cases for the quarter-partition room, then the three-quarter-partition room. Panel layout as in Figure 5.4.

Mean Squared Error — mean((pred – truth) ²) in PHYSICAL UNITS ² : u, v in (m/s) ² , p in Pa ² , T in K ² , k in (m ² /s ²) ² , ε in (m ² /s ³) ² . Scale differs per field. Dark outline = best model per split×field.																				
model	ANN							GNN							PINN					
field	T	k	p	u	v	ε	T	k	p	u	v	ε	T	k	p	u	v	ε		
split																				
Training	0.2226	0.0001238	2.643	0.00796	0.005521	0.00538	0.1275	8.106e-05	2.194	0.004597	0.002115	0.0452	0.0669	5.309e-05	2.746	0.003604	0.002081	0.1217		
Validation	0.4584	0.0001686	178.2	0.01466	0.008572	0.01587	0.275	0.0001142	103.8	0.01173	0.004795	0.04131	0.1024	8.674e-05	42.68	0.01123	0.005582	0.05763		
Interp Velocity	1.951	0.0001403	668.5	0.02033	0.008334	0.0293	0.5185	8.145e-05	459.9	0.01179	0.004857	0.04774	0.1322	6.48e-05	194.5	0.01103	0.006749	0.06866		
Interp Thermal	10.88	0.0001489	39.29	0.009983	0.006553	0.005138	0.6331	5.515e-05	11.9	0.00549	0.00247	0.01004	0.2259	3.784e-05	3.781	0.004161	0.002301	0.03357		
Extrap Velocity	3.116	0.009279	3017	0.0883	0.04142	11.16	0.5352	0.007442	906.3	0.08791	0.0653	11.23	0.2853	0.002731	227.3	0.02571	0.01685	8.633		
Extrap Thermal	6027	2.32e+04	6.63e+06	2.634e+04	6605	1.435e+05	60.99	0.0001933	443.9	0.02267	0.008585	0.107	0.1987	0.0001039	195.8	0.0201	0.00844	0.3026		

Mean Absolute Error — mean(|pred – truth|) in PHYSICAL UNITS: u, v in m/s, p in Pa, T in K, k in m²/s², ε in m²/s³. **Dark outline = best model per split×field.**

model	ANN							GNN							PINN					
field	T	k	p	u	v	ε	T	k	p	u	v	ε	T	k	p	u	v	ε		
split																				
Training	0.3479	0.0075	1.1971	0.0619	0.0432	0.0109	0.2525	0.0042	0.9047	0.0394	0.0245	0.0130	0.1732	0.0036	0.9566	0.0339	0.0232	0.0197		
Validation	0.5073	0.0084	4.2828	0.0748	0.0509	0.0116	0.3540	0.0052	3.1443	0.0533	0.0318	0.0132	0.2144	0.0047	2.2762	0.0500	0.0330	0.0212		
Interp Velocity	0.9591	0.0087	8.2264	0.0881	0.0536	0.0103	0.4295	0.0048	5.9262	0.0547	0.0330	0.0117	0.2256	0.0040	4.1038	0.0489	0.0346	0.0204		
Interp Thermal	2.6044	0.0081	2.6456	0.0699	0.0487	0.0103	0.4231	0.0041	1.5976	0.0459	0.0281	0.0111	0.2521	0.0036	1.0442	0.0402	0.0278	0.0172		
Extrap Velocity	1.3078	0.0353	23.1019	0.1766	0.1142	0.0813	0.5389	0.0262	12.9127	0.1511	0.0985	0.0873	0.3176	0.0154	7.2808	0.0812	0.0589	0.0741		
Extrap Thermal	35.6371	27.2069	515.4531	29.4579	17.7798	64.2495	6.6728	0.0071	5.4655	0.0824	0.0465	0.0396	0.2858	0.0050	3.6553	0.0600	0.0382	0.0236		

Mean Squared Error — mean((pred – truth)²) in PHYSICAL UNITS²: u, v in (m/s)², p in Pa², T in K², k in (m²/s²)², ε in (m²/s³)². Scale differs per field. **Dark outline = best model per split×field.**

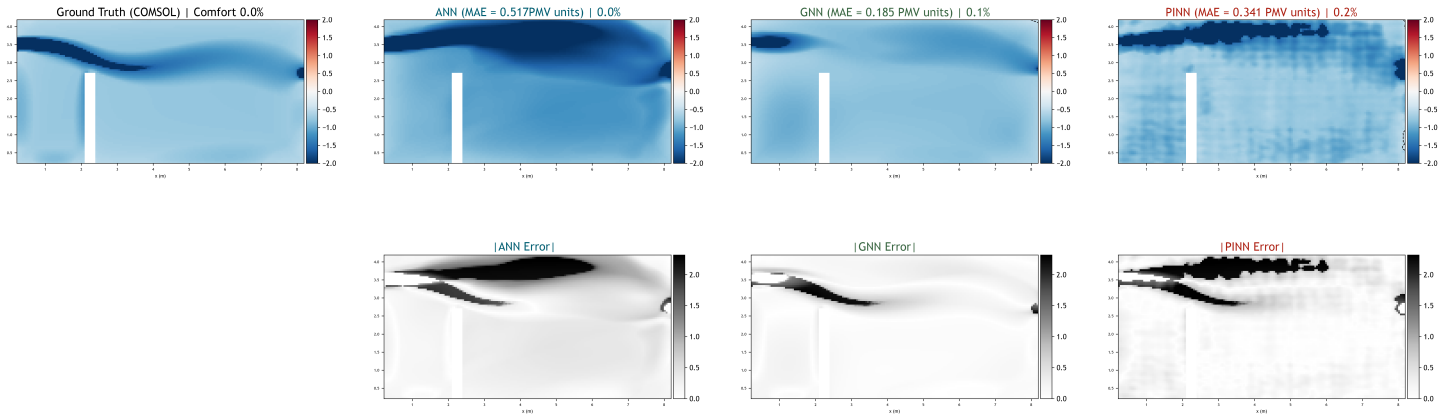
model	ANN							GNN							PINN					
field	T	k	p	u	v	ε	T	k	p	u	v	ε	T	k	p	u	v	ε		
split																				
Training	0.1821	6.919e-05	3.011	0.003295	0.001307	0.1536	0.1215	4.487e-05	1.9	0.002309	0.001188	0.08188	0.6172	6.038e-06	0.8761	0.003286	0.001188	0.001951		
Validation	0.4871	0.0001101	7.208	0.007366	0.003192	0.0122	0.2747	0.0001018	28.41	0.008978	0.004836	0.01142	0.5169	6.852e-05	1.466	0.009065	0.002975	0.0117		
Interp Velocity	1.469	0.0001507	102	0.02387	0.009398	0.006231	0.2325	8.65e-05	26.79	0.0154	0.007544	0.009461	0.4952	3.55e-05	9.83	0.008835	0.004496	0.002085		
Interp Thermal	5.549	0.0001595	123.2	0.01085	0.003253	0.0323	0.1499	0.000102	139.9	0.006965	0.00277	0.05019	0.4039	2.841e-05	50.32	0.005192	0.002224	0.03613		
Extrap Velocity	5.451	0.005606	2455	0.1627	0.04999	2.218	1.069	0.005222	867.6	0.1243	0.0575	2.431	0.4898	0.001845	2143	0.08953	0.03385	2.01		
Extrap Thermal	46.18	0.001119	1097	0.06197	0.02312	0.1239	79.89	0.0003039	187.4	0.01709	0.007652	0.09415	0.7768	0.0001139	195	0.008837	0.004764	0.08134		

Mean Absolute Error — mean(|pred – truth|) in PHYSICAL UNITS: u, v in m/s, p in Pa, T in K, k in m²/s², ε in m²/s³. **Dark outline = best model per split×field.**

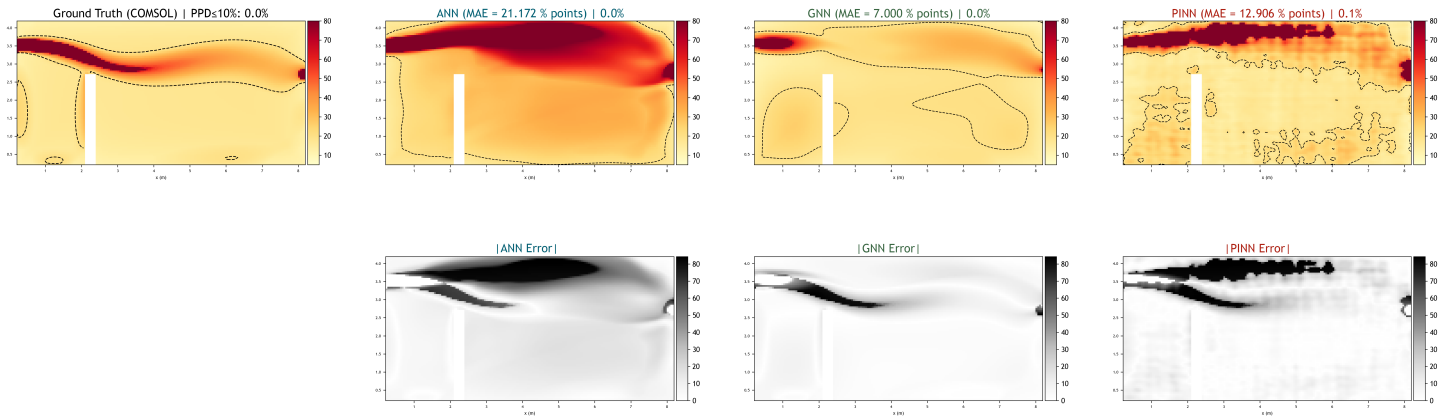
model	ANN							GNN							PINN					
field	T	k	p	u	v	ε	T	k	p	u	v	ε	T	k	p	u	v	ε		
split																				
Training	0.3032	0.0039	1.2384	0.0348	0.0211	0.0098	0.2395	0.0032	0.9030	0.0296	0.0192	0.0100	0.2241	0.0014	0.2973	0.0238	0.0129	0.0061		
Validation	0.5155	0.0052	1.5228	0.0512	0.0314	0.0090	0.3749	0.0046	1.7260	0.0485	0.0322	0.0092	0.2395	0.0034	0.4659	0.0405	0.0234	0.0071		
Interp Velocity	0.8449	0.0076	3.6240	0.0891	0.0509	0.0095	0.3563	0.0051	2.1766	0.0585	0.0392	0.0096	0.2239	0.0031	1.0811	0.0431	0.0281	0.0062		
Interp Thermal	2.1018	0.0067	4.3481	0.0612	0.0334	0.0125	0.2835	0.0050	5.6644	0.0462	0.0275	0.0128	0.1967	0.0026	2.4341	0.0311	0.0166	0.0080		
Extrap Velocity	1.6935	0.0401	16.5549	0.2501	0.1300	0.0689	0.7605	0.0347	10.7383	0.2071	0.1293	0.0693	0.2779	0.0231	12.5745	0.1537	0.0884	0.0512		
Extrap Thermal	6.2919	0.0167	11.4854	0.1497	0.0822	0.0215	8.2103	0.0087	5.7945	0.0773	0.0453	0.0213	0.3771	0.0045	4.7989	0.0454	0.0268	0.0105		

Figure B.5: Field-accuracy tables: mean squared error and mean absolute error per model, by test case and field, for the quarter-partition room, then the three-quarter-partition room. The lowest error in each split-and-field cell is outlined. Layout as in Figure 5.5.

PMV (range - [-3, +3]) - Valid Case 2



PPD (%) - Predicted Percentage Dissatisfied - Valid Case 2



DR (%) - Draught Rate - Valid Case 2

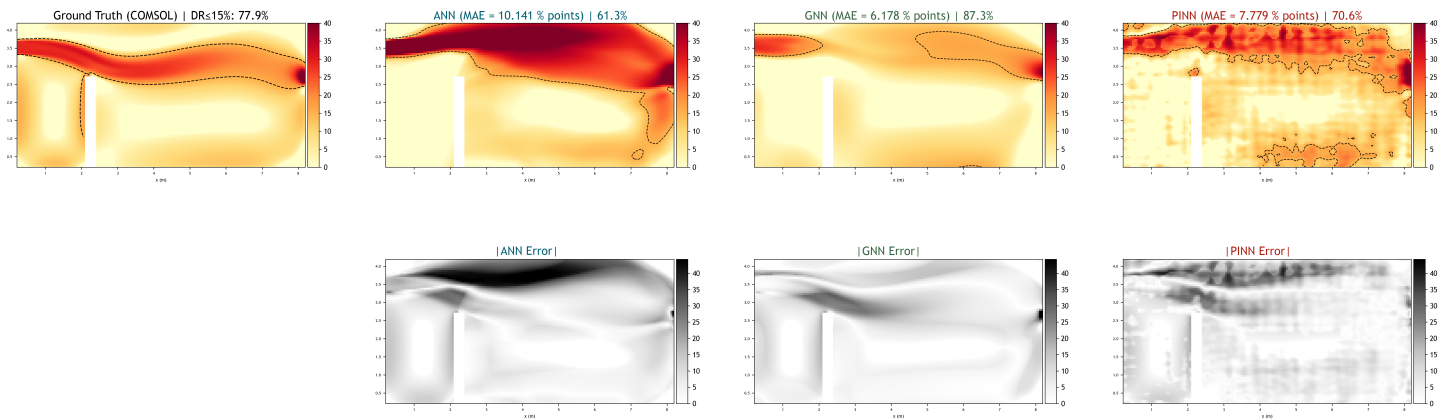
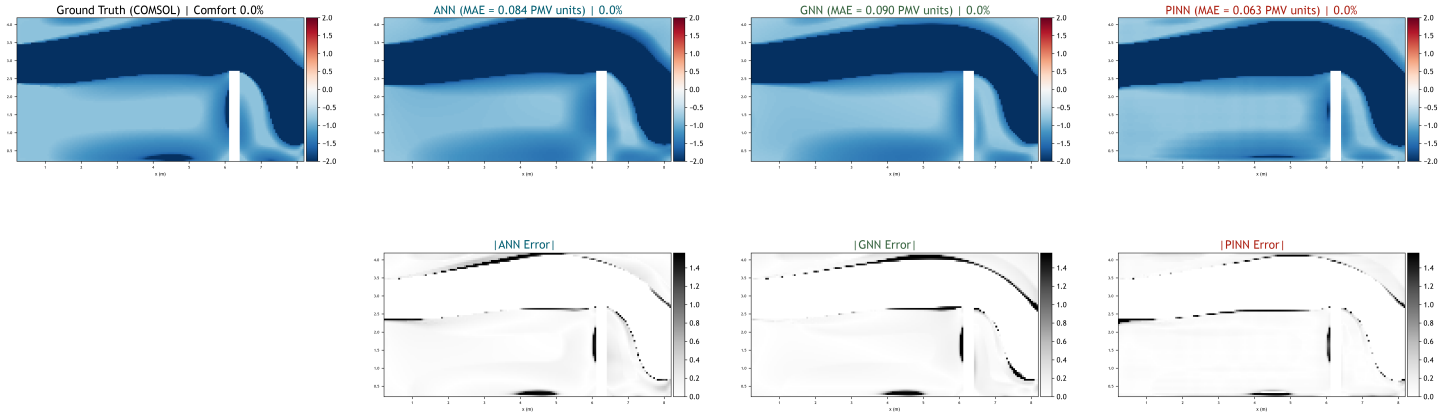
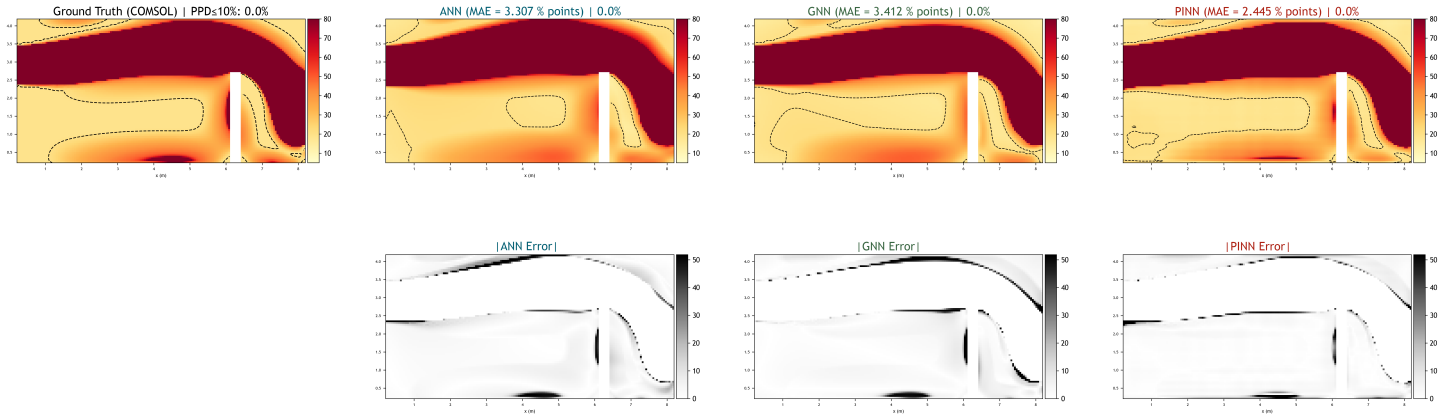


Figure B.6: Predicted comfort fields in the quarter-partition room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Layout as in Figure 5.6.

PMV (range = [-3, +3]) - Valid Case 30



PPD (%) - Predicted Percentage Dissatisfied - Valid Case 30



DR (%) - Draught Rate - Valid Case 30

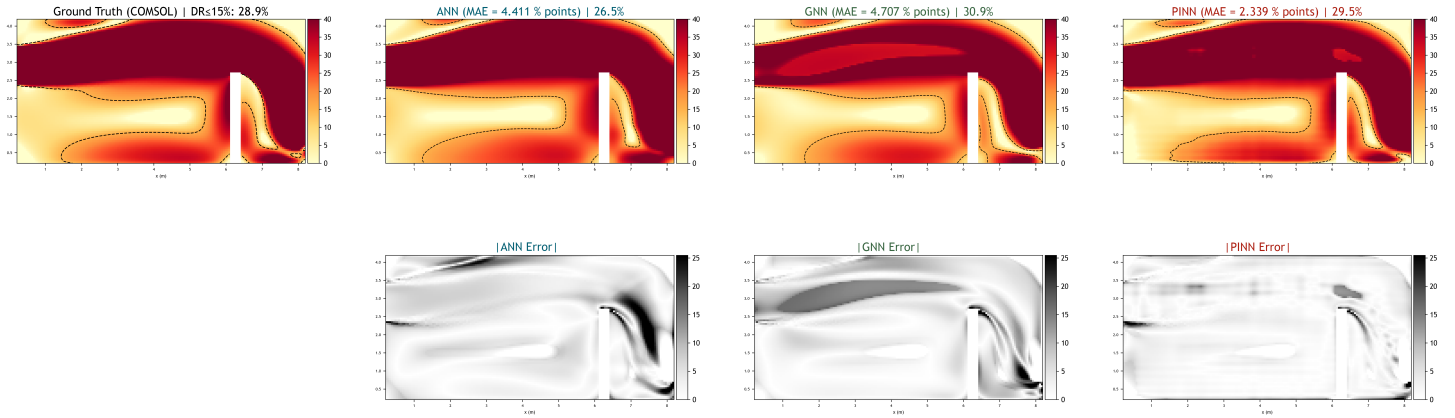


Figure B.7: Predicted comfort fields in the three-quarter-partition room. Top to bottom: PMV, predicted percentage dissatisfied (%), and draught rate (%). Layout as in Figure 5.6.

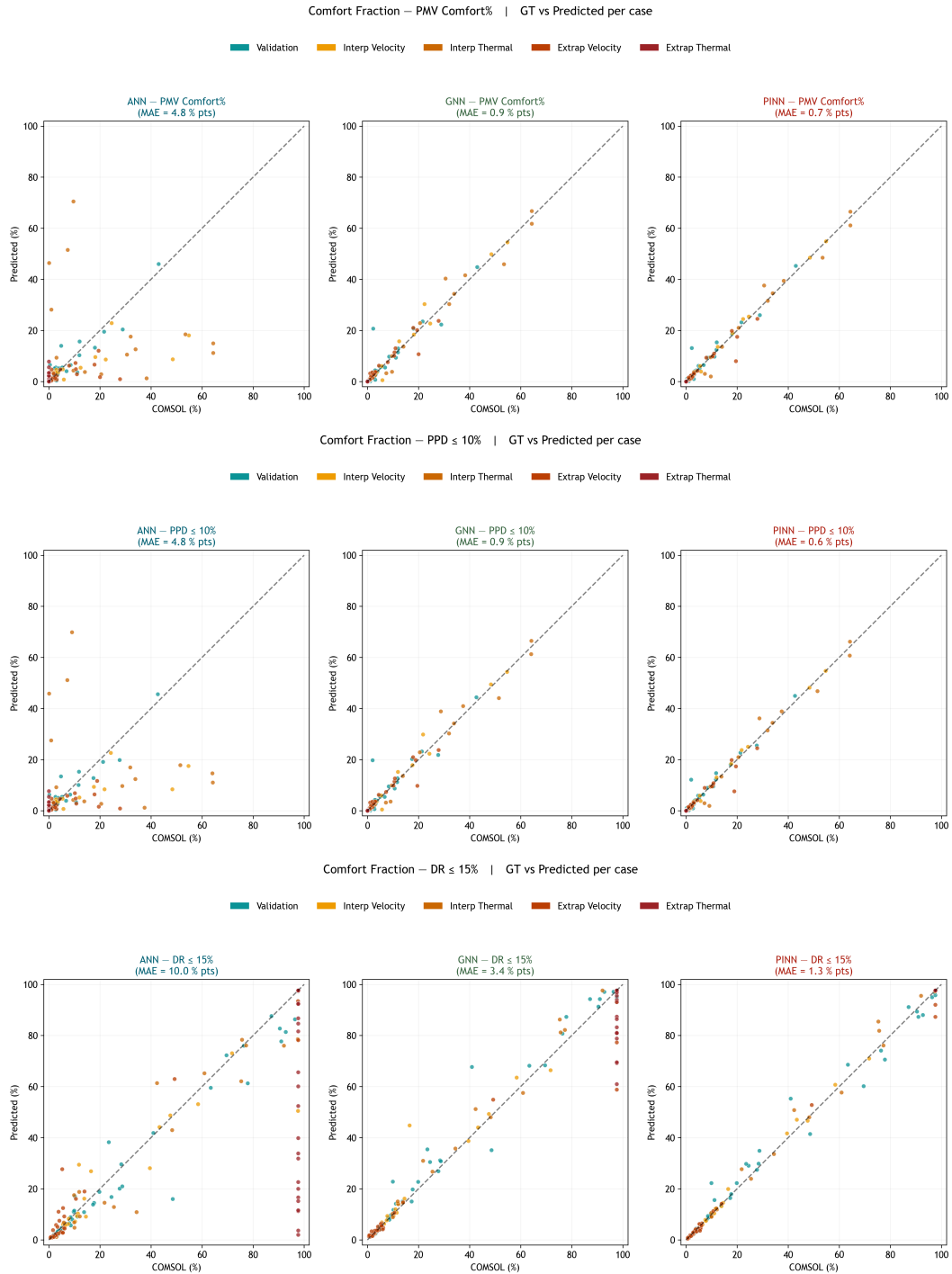


Figure B.8: Comfort-fraction agreement in the quarter-partition room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Layout as in Figure 5.8.

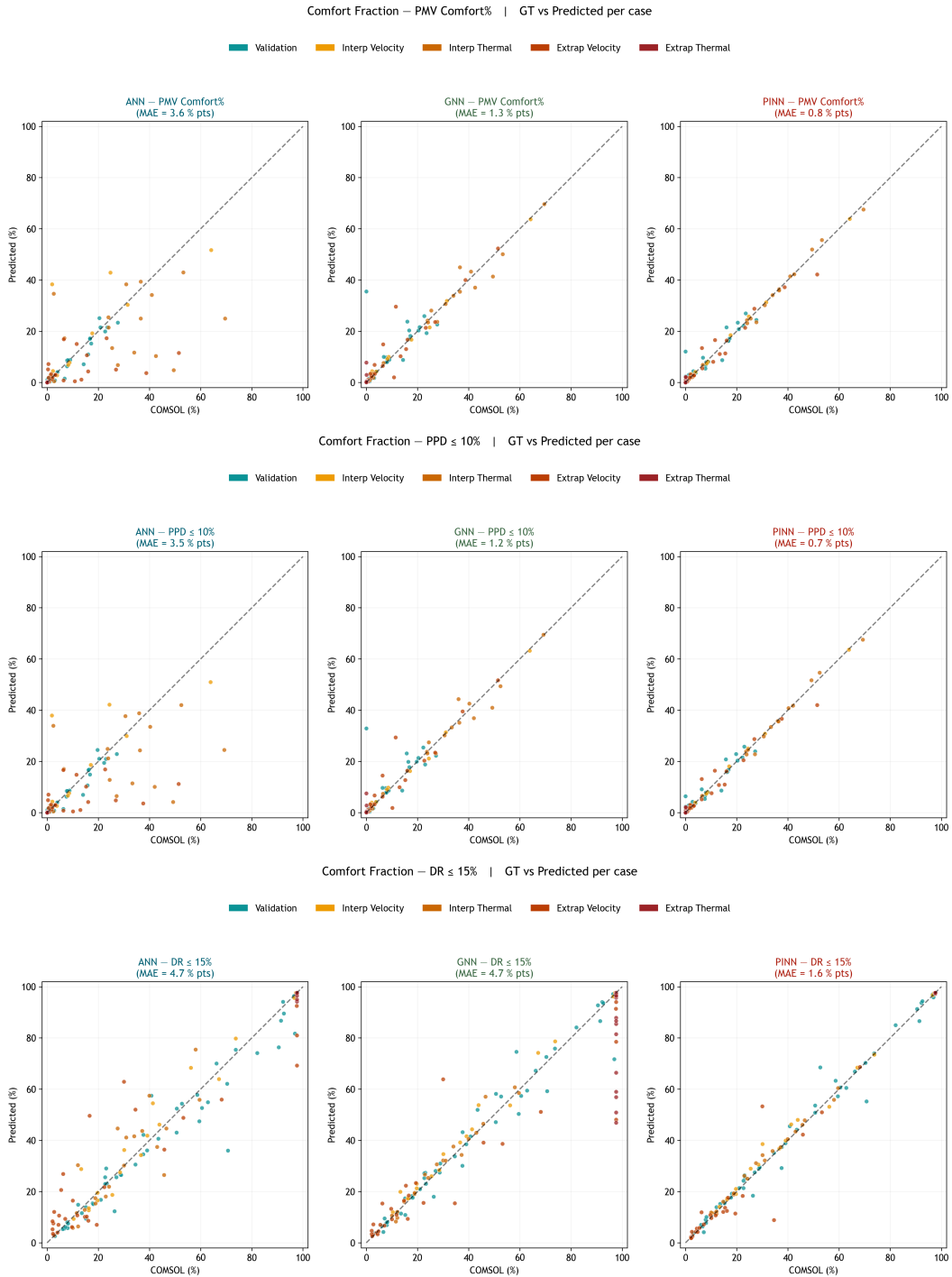
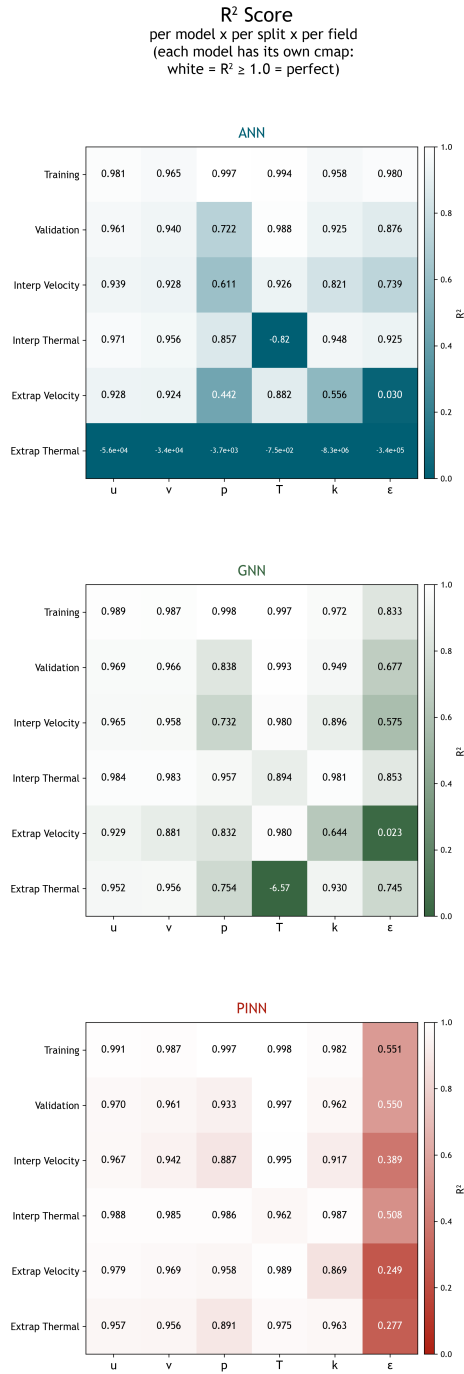
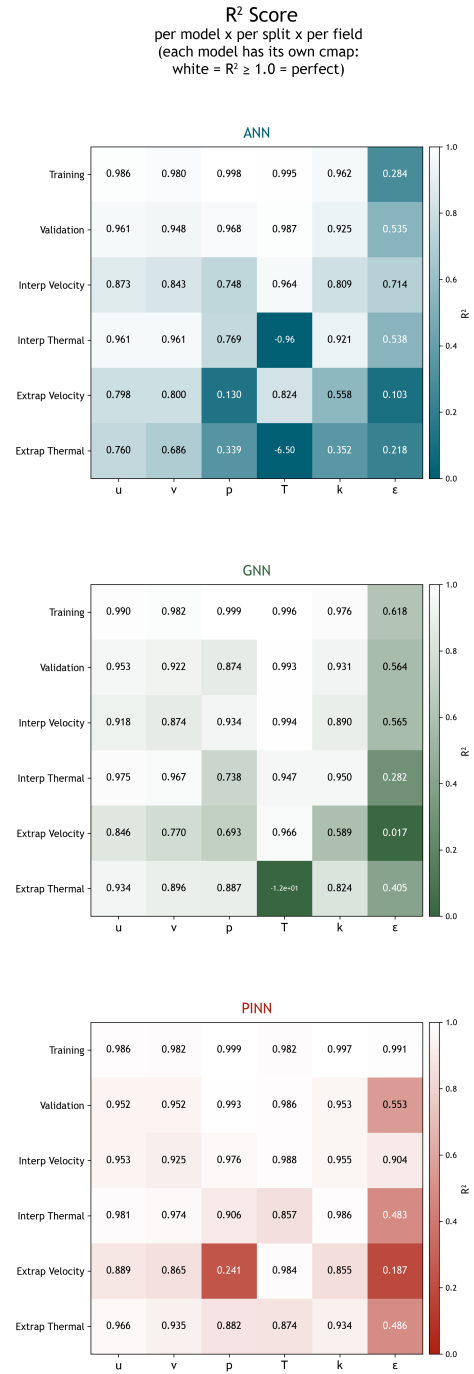


Figure B.9: Comfort-fraction agreement in the three-quarter-partition room. Top to bottom: the comfort fraction from PMV, predicted percentage dissatisfied, and draught rate. Layout as in Figure 5.8.

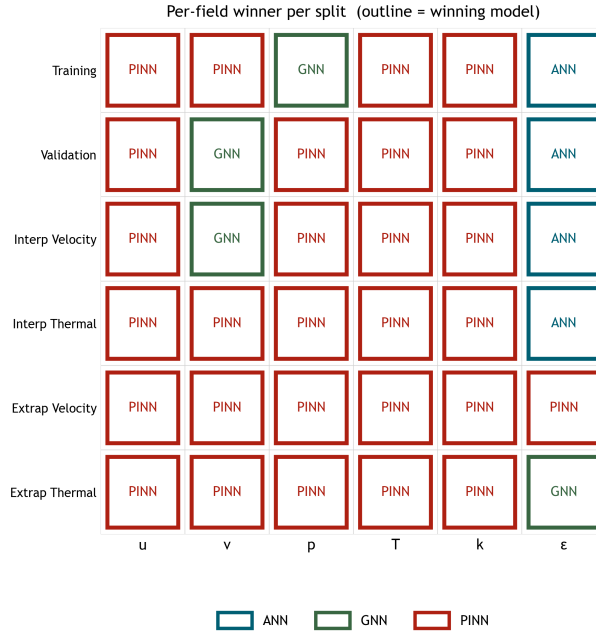


(a) Quarter-partition room

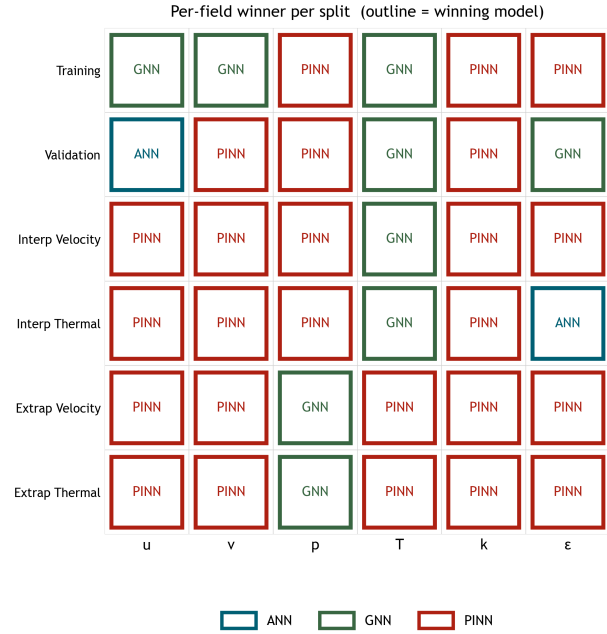


(b) Three-quarter-partition room

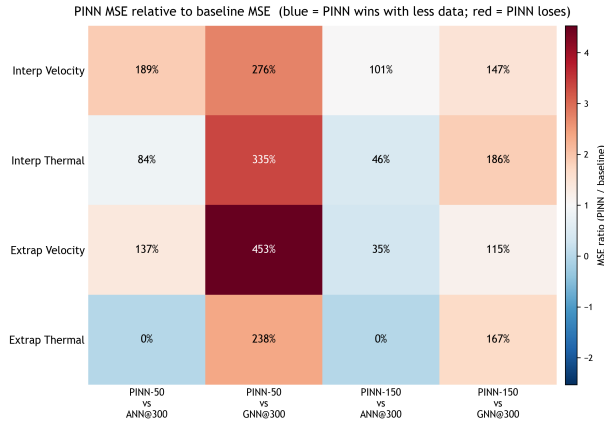
Figure B.10: Coefficient of determination per model, split, and field, one heatmap per model: (a) the quarter-partition room and (b) the three-quarter-partition room. Layout as in Figure 5.10.



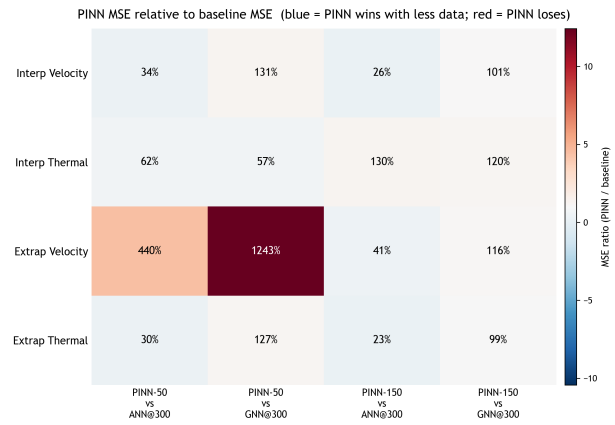
(a) Per-field winner, quarter-partition room



(b) Per-field winner, three-quarter-partition room



(c) PINN on reduced data vs full-data baselines, quarter-partition room



(d) PINN on reduced data vs full-data baselines, three-quarter-partition room

Figure B.11: Per-field winners (top) and the PINN's reduced-data comparison (bottom): (a, c) the quarter-partition room and (b, d) the three-quarter-partition room. Layout as in Figure 5.11.

C

Parameter Ranges and LHS Configuration

This appendix records the exact parameter ranges, interpolation gaps, extrapolation bands, LHS seeding scheme, and per-split case counts used in the CFD parameter sweep of Section 4.3. The intent is single-place reproducibility: anyone reading this appendix should be able to regenerate the parameter manifest exactly.

C.1 Seven-Dimensional Training Range

The seven LHS-sampled parameters and their training-range bounds follow Table 4.1, restated here in the form actually sampled: the two opening placements appear as the auxiliary unit variables α_{in} , α_{out} of Section 4.2.4 rather than as the derived physical heights.

Table C.1: Training-range bounds for the seven LHS-sampled parameters.

Symbol	Physical meaning	Training range
U_{in}	Inlet normal velocity (m s^{-1})	[0.30, 2.00]
d_{in}	Inlet opening size (m)	[0.10, 1.20]
α_{in}	Auxiliary unit variable for inlet height	[0.05, 0.95]
d_{out}	Outlet opening size (m)	[0.10, 1.20]
α_{out}	Auxiliary unit variable for outlet height	[0.05, 0.95]
T_{in}	Inlet air temperature (K)	[288, 308]
T_{wall}	Exterior wall surface temperature (K)	[293, 318]

The geometry-safe transformation that maps the auxiliary unit variables to physical opening heights is Equation 4.8 of Section 4.2.4, with margin $m = 0.20$ m.

C.2 Interpolation Gaps

Training cases are rejected if they fall in either of the following two interpolation gaps (Section 4.3.2):

- $U_{\text{in}} \in [1.00, 1.50] \text{ m s}^{-1}$ – velocity interpolation gap.
- $T_{\text{in}} \in [296, 302] \text{ K}$ – thermal interpolation gap.

The interp-velocity and interp-thermal test splits sample exclusively inside these gaps, one axis at a time, with the other parameters drawn over their full training ranges.

C.3 Extrapolation Bands

The extrapolation test splits sample strictly above the training range (Section 4.3.3):

- $U_{\text{in}} \in [2.10, 3.00] \text{ m s}^{-1}$ – velocity extrapolation band, above the training maximum of 2.00 m s^{-1} .
- $T_{\text{in}} \in [310, 320] \text{ K}$ – thermal extrapolation band, above the training maximum of 308 K .

Only one axis is pushed out of range at a time; the other six parameters are sampled over their normal training ranges. The buffers between the training ceiling and the start of each band, 0.10 m s^{-1} for velocity and 2 K for temperature, keep the training and extrapolation distributions from touching at their shared edge.

C.4 Latin Hypercube Sampling Configuration

The sampling is performed by the McKay, Beckman, and Conover (1979) seven-dimensional Latin Hypercube scheme with:

- Oversample factor 2.5 to enable rejection sampling of interpolation-gap candidates.
- Manifest CSV recording every attempted case (parameter values, convergence status) to enable resume logic after solver failures.
- COMSOL solve timeout of 900 s per case.
- Seeding via NumPy's `default_rng`, with an independent base seed per split, each batch offset by $10,000b$ for batch index b , and each wall geometry offset by a further 100,000 (Section 4.3.1); the base seed values are recorded with the released source code.

C.5 Case Counts per Split per Configuration

The six-split case counts per wall configuration are:

Table C.2: Target case counts per split per wall configuration.

Split	Cases
train	300
valid	60
interp-velocity	20
interp-thermal	20
extrap-velocity	30
extrap-thermal	30
Total per config	460
Total across four wall configurations	1840

C.6 Deterministically Nested Data-Efficiency Subsets

The 50-case and 150-case subsets used in the data-efficiency study (Section 4.3.4) are nested prefixes of the single training-split LHS sample (Section 4.3.1):

$$\text{subset}_{50} \subset \text{subset}_{150} \subset \text{full training set},$$

where the 50-case set is the first 50 cases of the 150-case set, which is in turn the first 150 of the full 300. Because the order is fixed by the seeded LHS draw, the same cases are used by every architecture at a given size, so the data-efficiency comparison is not confounded by sample identity. No additional CFD runs are needed for the smaller sizes; they are subsets of data already solved.

C.7 Identical Parameter Ranges Across Configurations

All parameter ranges, interpolation gaps, extrapolation bands, and case counts above are held **identical across the four wall configurations** (open, quarter, half, three-quarter). Only the LHS seed differs between geometries (Section 4.3.1), so each is a genuinely different sample drawn from the same ranges. This is the structural assumption that enables the cross-configuration robustness comparison of Section 4.4 and Appendix B.

D

Verification of Thermal Comfort Implementation

This appendix records the verification of the thermal-comfort post-processing layer described in Section 4.6 against its source standards: ISO 7730:2005 (ISO 2005) for PMV, PPD, and DR, and ISO 7726:1998 (ISO 1998) for MRT. Three independent checks are performed: against the standard's published reference output for PMV and PPD, against the closed form of ISO 7730:2005 Section 6.2, Eq. (6) for DR, and against the radiative-closure identity $\sum_i F_{p-i} = 1$ of ISO 7726:1998 Annex B for MRT.

D.1 Basis of Verification

PMV and PPD are computed through `pythermalcomfort` (Tartarini and Schiavon 2020), specifically the function `pmv_ppd_iso(..., model="7730-2005")`, which is a direct Python port of the BASIC reference program published as ISO 7730:2005 Annex D. The package authors validated this port against the ten canonical reference cases listed in ISO 7730:2005 Annex D Table D.1, with a maximum absolute deviation of ≤ 0.01 PMV units and ≤ 0.5 percentage points in PPD (Tartarini and Schiavon 2020, Table 1). That validation is taken as the formal verification of the PMV/PPD calculation in this work.

DR and the view-factor MRT are implemented in plain NumPy following ISO 7730:2005 Section 6.2, Eq. (6) and ISO 7726:1998 Annex B Eq. (B.1) respectively; these implementations are verified by direct comparison against the closed-form expressions of the standards, with the results shown in Section D.3 and Section D.4 below.

D.2 PMV / PPD Across the Operating Range

Table D.1 reports PMV, PPD, and the corresponding ISO 7730:2005 Annex A acceptability category for six environments spanning the operating range of the CFD case sweep, at the personal-parameter configuration of this study ($M = 1.2$ met, $I_{cl} = 0.5$ clo, $RH = 50\%$). Values are produced by `pythermalcomfort v3.x` (Tartarini and Schiavon 2020).

Table D.1: PMV / PPD / ISO 7730 Annex A category across six operating-range environments at $M = 1.2$ met, $I_{cl} = 0.5$ clo, $RH = 50\%$.

t_a (°C)	$\bar{t}_r$ (°C)	v_a (m s ⁻¹)	PMV	PPD (%)	Category
20.0	20.0	0.10	-1.43	47.3	outside
22.0	22.0	0.10	-0.81	18.9	outside
24.0	24.0	0.10	-0.21	5.9	B (II)
24.0	24.0	0.30	-0.70	15.2	outside
26.0	26.0	0.10	+0.38	8.1	B (II)
26.0	26.0	0.30	+0.00	5.0	A (I)

The sweep covers thermal sensations from cool (“outside” the comfort categories on the cold side) through neutral (Categories A and B) to slightly warm; PPD bottoms out at the 5% floor of the Fanger model at $PMV = 0$ as expected (ISO 2005; Fanger 1970).

D.3 Draft Rate against the ISO 7730:2005 Closed Form

The closed-form evaluation uses ISO 7730:2005 Section 6.2, Eq. (6) directly with an externally-set turbulence intensity T_u ; this work's implementation first derives T_u from the resolved turbulent kinetic energy k under the isotropy relation

$$T_u [\%] = 100 \cdot \sqrt{2k/3} / v_a,$$

then applies Eq. (6). For verification, k is back-solved from a target T_u via $k = 1.5 \cdot (T_u \cdot v_a/100)^2$.

Table D.2: Draft Rate against the ISO 7730:2005 Section 6.2, Eq. (6) closed-form evaluation at five benchmark points across the operating range.

t_a (°C)	v_a (m s ⁻¹)	T_u (%)	DR closed-form (%)	DR this work (%)	$ \Delta $
20.0	0.20	40.0	26.3411	26.3411	3.6×10^{-15}
23.0	0.15	40.0	14.1435	14.1435	3.6×10^{-15}
24.0	0.30	40.0	32.0916	32.0916	7.1×10^{-15}
22.0	0.10	20.0	7.2673	7.2673	8.9×10^{-16}
26.0	0.40	60.0	50.1553	50.1553	7.1×10^{-15}

Maximum absolute deviation across the five benchmark points is 7.11×10^{-15} %, i.e. double-precision machine epsilon. The DR implementation reproduces ISO 7730:2005 Section 6.2, Eq. (6) exactly.

D.4 MRT View-Factor Radiative Closure

Per ISO 7726:1998 Annex B Eq. (B.1) (ISO 1998), the view factors from any interior point of an enclosure to its bounding walls must satisfy $\sum_i F_{p-i} = 1$, and consequently if every wall is at a single temperature T_{wall} the predicted $\bar{t}_r$ must equal T_{wall} everywhere within the enclosure. The implementation was tested on an 81×41 grid covering the $8 \text{ m} \times 4 \text{ m}$ domain at three uniform wall temperatures:

Table D.3: MRT view-factor radiative closure verification at three uniform wall temperatures.

Test	T_{wall} (°C)	MRT range (°C)	$\ \text{MRT} - T_{\text{wall}}\ _{\infty}$	$\sum_i F_{p-i}$ range
Uniform A	18.0	[18.000000, 18.000000]	$5.7 \times 10^{-14} \text{ °C}$	[1.000000000, 1.000000000]
Uniform B	22.0	[22.000000, 22.000000]	$5.7 \times 10^{-14} \text{ °C}$	[1.000000000, 1.000000000]
Uniform C	26.0	[26.000000, 26.000000]	$5.7 \times 10^{-14} \text{ °C}$	[1.000000000, 1.000000000]

In all three uniform-wall cases the predicted $\bar{t}_r$ equals T_{wall} to machine precision ($\leq 6 \times 10^{-14} \text{ °C}$ across all 79×39 interior points), and the sum of view factors is exactly 1 to nine decimal places.

For an asymmetric configuration (floor 28 °C , ceiling 18 °C , side walls 23 °C – a heated-floor / cool-ceiling room) the predicted MRT is spatially non-uniform with range $[21.70, 24.45] \text{ °C}$, while $\sum_i F_{p-i}$ remains 1.000000000 throughout the interior. The spatial variation arises entirely from the directional asymmetry of the four view factors, as ISO 7726:1998 Annex B prescribes (ISO 1998).

D.5 Summary of Verification

Table D.4: Summary of verification: each quantity tested, the reference against which it was tested, the implementation under test, and the maximum absolute deviation observed.

Quantity	Reference	Implementation	Max abs. deviation
PMV	ISO 7730:2005 Annex D Table D.1, ten reference cases	pmv_ppd_iso (model 7730-2005), validated in (Tartarini and Schiavon 2020, Table 1)	≤ 0.01 PMV units
PPD	ISO 7730:2005 Annex D Table D.1, ten reference cases	as above	≤ 0.5 percentage points
DR	ISO 7730:2005 Section 6.2, Eq. (6); closed-form check at five operating-range points (Section D.3)	NumPy implementation, this work	$7.1 \times 10^{-15} \%$ (machine ϵ)
MRT	ISO 7726:1998 Annex B Eq. (B.1), radiative closure $\sum_i F = 1$ at every interior point (Section D.4)	NumPy implementation, this work	$6 \times 10^{-14} \text{ °C}$ (machine ϵ)

The thermal comfort post-processing layer is therefore demonstrably faithful to its source standards across all four quantities reported in the results sections.