Deep encoder-decoder hierarchical convolutional neural networks for conjugate heat transfer surrogate modeling

Takiah Ebbs-Picken[a], David A. Romero[a], Carlos M. Da Silva[a], Cristina H. Amon[a,b,*]

[a]*Department of Mechanical and Industrial Engineering, ATOMS Laboratory, University of Toronto, 5 King's College Road, Toronto, ON M5S 3G8, Canada*

[b]*Chemical Engineering and Applied Chemistry, University of Toronto, 5 King's College Road, Toronto, ON M5S 3G8, Canada*

Abstract

Conjugate heat transfer (CHT) analyses are vital for the design of many energy systems. However, high-fidelity CHT numerical simulations are computationally intensive, which limits their use in applications such as design optimization, where hundreds to thousands of evaluations are required. In this work, we develop a modular deep encoder-decoder hierarchical (DeepEDH) convolutional neural network, a novel deep-learning-based surrogate modeling methodology for computationally intensive CHT analyses. Leveraging convective temperature dependencies, we propose a two-stage temperature prediction architecture that couples velocity and temperature fields. The proposed DeepEDH methodology is demonstrated by modeling the pressure, velocity, and temperature fields for a liquid-cooled cold-plate-based battery thermal management system with variable channel geometry. A computational mesh and CHT formulation of the cold plate is created and solved using the finite element method (FEM), generating a dataset of 1,500 simulations. The FEM results are transformed and scaled from unstructured to structured, image-like meshes to create training and test datasets for DeepEDH models. The DeepEDH architecture's performance is examined in relation to data scaling, training dataset size, and network depth. Our performance analysis covers the impact of the novel architecture, separate DeepEDH models for each field, output geometry masks, multi-stage temperature field predictions, and optimizations of the hyperparameters and architecture. Furthermore, we quantify the influence of the CHT analysis' thermal boundary conditions on surrogate model performance, highlighting improved temperature model performance with higher heat fluxes. Compared to other deep learning neural network surrogate models, such as U-Net and DenseED, the proposed DeepEDH architecture for CHT analyses exhibits up to a 65% enhancement in the coefficient of determination R^2.

Keywords: Surrogate modeling, machine learning, deep-learning, deep encoder-decoder hierarchical (DeepEDH) convolutional neural networks, conjugate heat transfer, battery thermal management

Nomenclature

Acronyms

ANN	artificial neural network
CFD	computational fluid dynamics
CHT	conjugate heat transfer
CIC	convolution-in-convolution
DeepEDH	modular deep convolutional encoder-decoder hierarchical
FCN	fully convolutional network
FEM	finite element method
FPN	feature pyramid network
MSE	mean squared error
PDE	partial differential equation
RMSE	root mean square error
SCC	Spearman's rank correlation coefficient
SOC	state of charge

Symbols

$\boldsymbol{F}$	body force
$\boldsymbol{I}$	identity tensor
$\boldsymbol{K}$	stress tensor
$\boldsymbol{n}$	unit normal vector
$\boldsymbol{q}$	heat flux
$\boldsymbol{u}$	velocity
ΔH	enthalpy difference
$\dot{m}$	mass flow rate
μ	dynamic viscosity
ρ	density
$\forall$	volume
A	area
C_p	specific heat capacity
d_{bc}	boundary condition thickness
h	convective heat transfer coefficient
I	current
k	thermal conductivity
p	pressure
P_0	power source
p_0	outlet pressure
Q	heat source
R^2	coefficient of determination
T	temperature
t	time
T_{ext}	external temperature
V	voltage

Contents

Highlights 1

Nomenclature 1

1 Introduction 4

2 Governing equations and data processing procedures 6
2.1 Governing equations and boundary conditions . . . 6
2.2 Data structuring and scaling procedures . . . 7

3 Surrogate modeling methodology 8
3.1 Data-driven surrogate modeling as image-to-image regression . . . 8
3.2 Modular deep encoder-decoder hierarchical (DeepEDH) convolutional neural network architecture . . . 8
3.2.1 Dense and fully convolutional encoder-decoder neural networks . . . 8
3.2.2 Output geometry mask . . . 10
3.2.3 Two-stage temperature model . . . 10
3.2.4 Final architecture: DeepEDH . . . 10
3.3 Network hyperparameters, training loss, and regularization . . . 11

4 Surrogate implementation and training 11
4.1 Problem definition . . . 12
4.1.1 Datasets . . . 13
4.2 Surrogate model training . . . 14

5 Surrogate model performance and characterization 17
5.1 Evaluation metrics . . . 17
5.2 Characterization results . . . 17
5.2.1 Code dimension . . . 17
5.2.2 Dataset size . . . 20
5.2.3 Data resolution . . . 20
5.3 Overall DeepEDH model performance . . . 21
5.4 DeepEDH-temperature model performance at varying heat fluxes . . . 23
. . . 25

6 Summary and conclusions 25

Data availability 26

Acknowledgments 26

References 26

1. Introduction

Heat transfer analyses are vital for designing many energy systems, with applications ranging from batteries and fuel cells to electric vehicles and power generation. For many of these systems, temperature directly impacts their performance, requiring effective thermal management. Designing and optimizing thermal management systems requires accurately predicting their heat transfer behavior. Many of these applications use convection-based thermal management systems and require the consideration of heat transfer in both solid and fluid domains simultaneously; such analysis is known as conjugate heat transfer (CHT). Different governing partial differential equations (PDEs), the momentum, continuity, and energy equations, exist for the solid and fluid domains and are coupled at the solid-fluid interface. Solving these equations in each domain requires a dynamic update of the interface boundary condition. Analytical solutions for CHT problems are limited and often rely on simplifications, limiting their practical applicability [1]. Instead, numerical approaches are commonly used to solve CHT problems [1], but these approaches can be computationally expensive, especially for high-fidelity three-dimensional solutions. Directly using high-fidelity models is unfeasible in many practical applications due to time and computational constraints. For example, design optimization typically requires thousands of model evaluations, whereas control and digital twin applications often have limited computational resources [2, 3, 4]. To address these challenges, reduced-order surrogate models can approximate the behavior of the system at a decreased computational cost, enabling their use in applications like design optimization [2], control [3], and digital twin systems [4], where high-fidelity models are impractical.

Data-driven surrogate modeling methodologies have been proven effective in approximating the behavior of complex engineering systems. Surrogate modeling requires a regression task where the input needs to be mapped non-linearly to the output. These approaches rely on input-output data observations to construct an approximation for the mapping or relationship of interest. While some works incorporate physics-based constraints into the surrogate model [5, 6, 7, 8], most data-driven approaches are purely data-based, requiring no prior knowledge of the system. Common approaches include polynomial regression, symbolic regression, Kriging, radial basis functions, and artificial neural networks (ANNs). Many studies have applied these approaches to predict specific values, such as extreme temperatures, velocities, or pressures, without reconstructing entire physical fields [2]. For instance, polynomial response surface and Kriging methods have been used for shape optimization of turbine blades from temperature and pressure predictions [9], boiler superheaters have been optimized using polynomial basis functions to predict heat transfer rates [10], and symbolic regressions have been applied in building energy demand response optimization [11]. Artificial neural networks have been used for various predictions, including closure coefficients for turbulent heat flux models in forced convection flows [12], internal fridge temperatures [13], maximum chip temperatures [14], and electronic component temperatures and flow velocities in nanofluid filled enclosures [15, 16, 17, 18]. While these traditional data-driven methodologies are effective for specific value predictions, they are typically incapable of reconstructing full physical fields and handling high-dimensional or permutation-invariant inputs without significant computational overhead, limiting their application to low-dimensional regression problems.

Alternatively, deep neural network data-based surrogate modeling approaches have been effective for applications involving regression of complex and non-linear behaviors. These approaches are characterized by their numerous layers, making them distinct from shallow ANNs and other data-based methods. Comparatively, deep neural networks can accurately reconstruct full physical field results, yielding similar accuracy for specific values derived from these fields [19]. Most research in this area has focused on convolutional neural networks (CNNs), which, initially designed for image classification, have proven effective for high-dimensional surrogate modeling. Levering a combination of linear convolution operations and non-linear activations, CNNs extract features from inputs to predict relevant outputs.

Convolutional neural networks have proven effective for surrogate modeling of engineering systems in various applications. For example, they have been used to reconstruct cooling effectiveness fields for transpiration cooling [20], predict velocity fields around a cylinder [21], predict battery lifetime [22], and estimate electric vehicle spatial distributions for electricity grid planning [23]. Extending CNNs, encoder-decoder architectures use a sequence of convolutional layers to reduce the input size to a low-dimensional latent space, or code dimension, followed by a series of deconvolutional layers to reconstruct the output. Compared to traditional CNNs, encoder-decoder architectures extract multi-scale features from the input more effectively. Initially developed for image reconstruction, these architectures have recently been applied as surrogate models for various engineering systems. For example, encoder-decoder CNN models have been used to predict flow fields around different objects [24] and airfoils [25], shale gas production [26], as well as CHT temperature and velocity fields [27].

To enhance information propagation within encoder-decoder networks, especially in cases with a direct input-output

dependency, the U-Net architecture [28] introduced skip connections to link the encoder and decoder sections. U-Net architectures have since been used to reconstruct temperature and velocity fields in various applications, such as nanofluid-filled finned absorber tubes for natural and forced convection [29] and film cooling in rocket combustors [30]. Dense architectures, which introduce shortcut connections between layers, aim to further improve information propagation and reduce the number of parameters in CNNs. Examples include ResNet [31], Highway Networks [32], and DenseNet [33]. These architectures enable more efficient training of deeper networks with fewer parameters and have been proven effective in image classification and segmentation tasks. In the context of surrogate modeling, these networks have been applied to various engineering systems, including nanofluid-filled finned absorber tubes [29, 34], channelized subsurface flow systems [35, 36, 37, 38], and CO_2 plume migration [39]. Building on these architectures, Zhu and Zabaras [40] extended the fully convolutional DenseNet [41] to develop the DenseED architecture for high-dimensional surrogate modeling. DenseED demonstrated strong performance in surrogate modeling and uncertainty quantification for single-phase flow in heterogeneous media with a 4225-dimensional input and multi-phase flow fields with a 2500-dimensional input [42]. Notably, DenseED showed excellent performance with limited training data and fewer network parameters than non-dense architectures. Romero *et al.* [43] extended DenseED with the DeepWFLO architecture to predict turbine wake fields for wind farm layout optimization, demonstrating the architecture's effectiveness for velocity field predictions.

In this work, we develop a new modular deep encoder-decoder hierarchical (DeepEDH) convolutional neural network architecture based on image-to-image regression, which builds on previous DenseED [40] and fully convolutional DenseNet [41] architectures. Our DeepEDH architecture is tailored for surrogate modeling of CHT problems. While several studies have developed surrogate models for CHT analysis, most are limited to predicting specific values of interest, with few models reconstructing full pressure, velocity, and temperature fields. Notably, none of these studies leverage the coupled nature of flow and heat transfer in CHT problems, instead predicting each field separately. This work's end-to-end surrogate modeling approach effectively addresses these research gaps. Our new DeepEDH convolutional neural network architecture leverages specific physics considerations. It introduces several novel aspects, including output geometry masks, separate models for each field, and a two-stage temperature prediction methodology. Separate models predict individual physical fields, serving as surrogate models for different governing PDEs. We apply an output geometry mask after the decoder to ensure the validity of reconstructed flow fields only within the fluid domain. Additionally, for temperature fields, we use two-stage models where velocity and temperature fields are linked, leveraging the coupled behavior of CHT problems. Combining the DeepEDH architecture, output geometry masks, field-specific models, and the connected two-stage temperature methodology, we address critical research gaps in previous works and demonstrate the ability to reconstruct complete fields for CHT analysis with improved accuracy and efficiency, requiring fewer network parameters than previous models.

Our work introduces several other aspects not explored in previous investigations:

1. A method for translating unstructured CHT simulation results into structured meshes, allowing the creation of convolutional surrogate model training databases from such results.
2. A comprehensive assessment of numerous factors influencing model performance, including architecture modifications, separate DeepEDH models for each field, output geometry masks, multi-stage temperature architecture, and hyperparameter and architecture optimization.
3. A thorough quantification of the effects of the neural network depth, data resolution, dataset size, and the magnitude of heat flux boundary conditions on surrogate model performance, allowing physical insights into the behavior of the CHT system.

Our findings and evaluations of the proposed DeepEDH architecture and end-to-end surrogate modeling methodology provide rich scientific insights to advance future research on surrogate modeling for CHT analyses of complex engineering systems.

The remainder of the paper is structured as follows. In Section 2, we present the governing equations for CHT analysis and outline the procedures for processing unstructured mesh data. Section 3 defines surrogate modeling in the context of image-to-image regression neural networks and introduces the model architecture, training process, regularization techniques, and loss function. In Section 4, we provide details of an implementation of the surrogate modeling methodology, focusing on an application to a liquid-cooled cold-plate-based battery thermal management system. This includes information on the generated datasets, model specifications, and the training process. Section 5 includes a

comprehensive analysis and characterization of the model's performance. Finally, in Section 6, we offer a summary of our findings and present our conclusions.

2. Governing equations and data processing procedures

This section describes the governing equations for CHT analyses and the data processing procedures used to generate the training and validation datasets from unstructured numerical simulation results.

2.1. Governing equations and boundary conditions

Conjugate heat transfer analyses are crucial for designing and optimizing thermal management systems, for example, those providing indirect liquid cooling through metal cold plates, as the one depicted in the case study used in this work (Section 4.1), featuring a cold plate with fluid channels and solid pin-fins and walls. Additional application illustrations are available in our previous work on the design optimization of pin-fin cold plates for electric vehicle battery packs [44]. In these cold-plate-based systems, the fluid enters the channel with velocity $\boldsymbol{u}$ and heat is transferred to the fluid from the cold plate surface heated by heat flux $\boldsymbol{q}$. Conjugate heat transfer is used to analyze such a system and determine the velocity and temperature fields in the fluid and the temperature field in the solid. This CHT analysis is governed by three equations: conservation of mass, momentum, and energy. Specifically, for three-dimensional, steady, and laminar flow with constant fluid properties within the fluid domain, the governing equations are momentum conservation Equation (1) and mass continuity Equation (2). Energy conservation is applied in both the fluid and solid domains. These domains are coupled at the solid-fluid interface, ensuring energy conservation and continuity of temperature and heat flux across this interface.

$$\begin{aligned} &\rho\left(\boldsymbol{u}\cdot\nabla\right)\boldsymbol{u} = \nabla\cdot\left[-p\boldsymbol{I}+\boldsymbol{K}\right]+\boldsymbol{F},\\ &\boldsymbol{K} = \mu\left(\nabla\boldsymbol{u}+\left(\nabla\boldsymbol{u}\right)^{T}\right), \end{aligned} \tag{1}$$

$$\rho\nabla\cdot\boldsymbol{u} = 0, \tag{2}$$

where $\boldsymbol{u}$ is the velocity vector, p is the pressure, ρ is the density, μ is the dynamic viscosity, $\boldsymbol{K}$ is the stress tensor, $\boldsymbol{F}$ is the body force, and $\boldsymbol{I}$ is the identity tensor. We define boundary conditions for the fluid domain as no-slip walls Equation (3), mass flow inlet Equation (4), pressure outlet Equation (5), and symmetry Equation (6).

$$\boldsymbol{u} = 0, \tag{3}$$

$$-\int_{\partial\Omega}\rho(\boldsymbol{u}\cdot\boldsymbol{n})d_{bc}dS = \dot{m}, \tag{4}$$

$$\left[-\rho\boldsymbol{I}+\boldsymbol{K}\right]\boldsymbol{n} = -p_0\boldsymbol{n}, \tag{5}$$

$$\begin{aligned} &\boldsymbol{u}\cdot\boldsymbol{n} = 0,\\ &\boldsymbol{K_n}-\left[\boldsymbol{K_n}\cdot\boldsymbol{n}\right] = 0, \qquad \boldsymbol{K_n} = \boldsymbol{K}\boldsymbol{n}, \end{aligned} \tag{6}$$

here $\boldsymbol{n}$ is the unit normal vector, d_{bc} is the boundary condition thickness, $\dot{m}$ is the mass flow rate, and p_0 is the outlet pressure. Three-dimensional and transient heat transfer is governed by Equation (7):

$$\begin{aligned} &\rho C_p\frac{\partial T}{\partial t}+\rho C_p\boldsymbol{u}\cdot\nabla T+\nabla\cdot\boldsymbol{q} = Q,\\ &\boldsymbol{q} = -k\nabla T, \end{aligned} \tag{7}$$

where T is the temperature, C_p is the specific heat capacity, t is time, $\boldsymbol{q}$ is the heat flux, Q is the volumetric heat generation rate, and k is the thermal conductivity, with boundary conditions consisting of insulated Equation (8) consisting of zero heat flux, convective heat flux Equation (8) with $q_0 = h(T_{ext}-T)$, inflow Equation (9), outflow Equation (10),

and symmetry Equation (10):

$$-\boldsymbol{n}\cdot\boldsymbol{q} = q_0, \tag{8}$$

$$\boldsymbol{n}\cdot\boldsymbol{q} = \rho\Delta H\boldsymbol{u}\cdot\boldsymbol{n},$$

$$\Delta H = \int_{T_{ustr}}^{T} C_p dT, \tag{9}$$

$$-\boldsymbol{n}\cdot\boldsymbol{q} = 0. \tag{10}$$

For Equations (8) to (10), q_0 is the heat flux, h is the convective heat transfer coefficient, T_{ext} is the external temperature, P_0 is the power source, $\mathcal{V}$ is the volume, and ΔH is the enthalpy difference. In this work's case study (Section 4), the heat source is modeled based on battery heat generation, considering the battery temperature and state-of-charge (SOC) independent Ohmic heating Equation (11) [45].

$$\dot{Q}_{\text{gen}} = |V - VOC(SOC)| \times I, \tag{11}$$

where V is the battery voltage, VOC is the open circuit voltage, and I is the current.

The heat flux, generated by a battery module in this work's case study (Section 4), is utilized as the cold plate's boundary condition for the energy Equation (7). The governing Equations (1), (2) and (7) with appropriate boundary conditions are then solved with the finite element method (FEM). To obtain accurate results across the fluid and solid domains, the mesh is refined at the fluid-solid interface to capture the high temperature and velocity gradients. Further, a conformal mesh at this interface ensures continuity of the temperature and heat flux from the solid to fluid domain, allowing the conjugate heat transfer to be accurately captured.

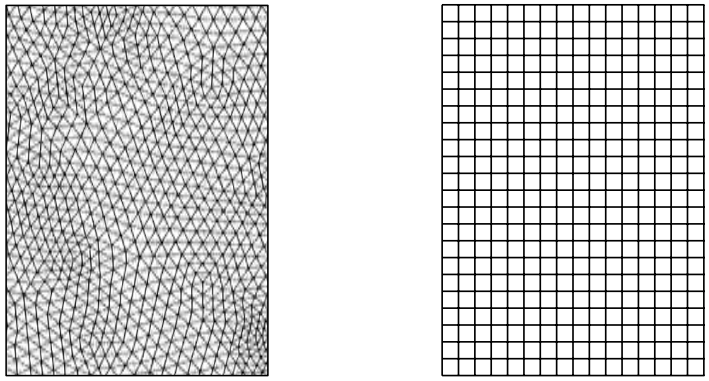
2.2. Data structuring and scaling procedures

Convolutional neural networks were generally designed for image-related tasks, such as recognition or segmentation, where image-like data is used for input and training. For CHT surrogate modeling, previous literature primarily used structured meshes composed of square cells, which resemble image pixels and are suitable for CNNs. However, for solving CHT problems with complex geometries, it is often advantageous to use unstructured meshes composed of irregular polygons without a specific structure. To adapt these numerical solutions for surrogate modeling, we transform the unstructured result data to structured data post-solver. By mapping to a structured grid after solving, we can represent the results as images with lower resolution than the unstructured solver mesh, leading to smaller models with faster and more efficient training.

To transform two-dimensional unstructured mesh data (Figure 1a) into structured mesh data (Figure 1b), we define a structured mesh as a grid of square cells with n_y rows and n_x columns. For each c_{struct} structured cell, we calculate an area-weighted average of the values from overlapping $c_{j_{\text{unstruct}}}$ unstructured cells, repeating this process for all cells in the new mesh. The area-weighted average is calculated according to Equation (12).

$$\Psi(c_{i_{\text{struct}}}) = \frac{\sum_{j=0}^{n} A(c_{j_{\text{unstruct}}}) \cdot \Psi(c_{j_{\text{unstruct}}})}{\sum_{j=0}^{n} A_(c_{j_{\text{unstruct}}})} \qquad \forall c_{j_{\text{unstruct}}} \in dom(c_{i_{\text{struct}}}) \tag{12}$$

where Ψ is the field value of interest, such as pressure, velocity, or temperature, A is the area of the cell, and dom is the domain of a cell.



(a) (b)

Figure 1: Illustration of the application of Equation (12) to temperature, velocity, or pressure field results, transforming (a) an unstructured mesh onto (b) an image-like structured mesh where each cell is representative of an image pixel.

3. Surrogate modeling methodology

3.1. Data-driven surrogate modeling as image-to-image regression

The thermal management systems under consideration in this work are governed by the CHT equations and boundary conditions detailed in Section 2.1. The solution process computes steady-state results, including pressure and velocity fields, as well as transient temperature results, using simulations based on the input geometry and predefined boundary conditions. Assuming fixed boundary conditions and variable geometry, we can view this simulation process as a black-box function, denoted as f, that maps the geometry model $\boldsymbol{X}_s$ to physical fields $\boldsymbol{y}$, as expressed in Equation (13).

$$\boldsymbol{y} = (P, \boldsymbol{u}, T) = f(\boldsymbol{x}, \boldsymbol{X}_s) \tag{13}$$

here $\boldsymbol{y}$ are the pressure (P), velocity ($\boldsymbol{u}$), and temperature (T) at each spatial location $\boldsymbol{x}$ in the domain.

Data-based surrogate modeling aims to replace the computationally expensive simulation process f with a more efficient model. This surrogate model is developed or trained using a limited dataset consisting of n simulation inputs and outputs: $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n}$. By treating the simulation inputs and outputs as images, the surrogate model can be formulated as an image-to-image regression model, similar to established CNN architectures. Both the input and output are considered as images with pixel dimensions of n_y rows and n_x columns, where each pixel is a value to be predicted by the regression model. Given that the simulation results from process f are defined on unstructured meshes, the data processing methods established in Section 2.2 are used to transform $\boldsymbol{y}$ results to structured image-like data. The regression models use the high-dimensional input to predict each pixel in the output, modeling a highly non-linear relationship to approximate the solution of the underlying governing PDE. We can define the surrogate model as a function, denoted as $\hat{f}$, which approximates f as outlined in Equation (14).

$$\hat{\boldsymbol{y}} = (\hat{P}, \hat{\boldsymbol{u}}, \hat{T}) = \hat{f}(\boldsymbol{x}, \boldsymbol{X}_s, \boldsymbol{\theta}) \tag{14}$$

the $\hat{\boldsymbol{y}}$ predictions consist of pressure ($\hat{P}$), velocity ($\hat{\boldsymbol{u}}$), and temperature ($\hat{T}$) results, approximating the true $\boldsymbol{y}$ values. The predictions are determined using the surrogate models with parameters $\boldsymbol{\theta}$, learned from the n training simulations: $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n}$.

3.2. Modular deep encoder-decoder hierarchical (DeepEDH) convolutional neural network architecture

In this section, we briefly introduce the techniques used in developing the DeepEDH architecture presented in this work.

3.2.1. Dense and fully convolutional encoder-decoder neural networks

Training deep convolutional neural networks can be challenging due to issues like vanishing gradients [46] and accuracy degradation [31]. To address these problems, various techniques have been introduced, including normalized initialization and intermediate initialization layers [31, 47, 48], and shortcut connections, as seen in architectures like

ResNet [31] and Highway Networks [32]. Shortcut connections are a fundamental feature of dense convolutional neural networks and enable more efficient training of deeper networks, allowing the output of a layer to be combined with the input to a layer further down the network. DenseNet [33] extended this concept, introducing shortcut connections to all subsequent layers. For network layer L, there will be $K \cdot L$ feature maps added based on the growth factor K. Figure 2 shows an example of a dense block based on DenseNet with $L = 3$ and $K = 8$.

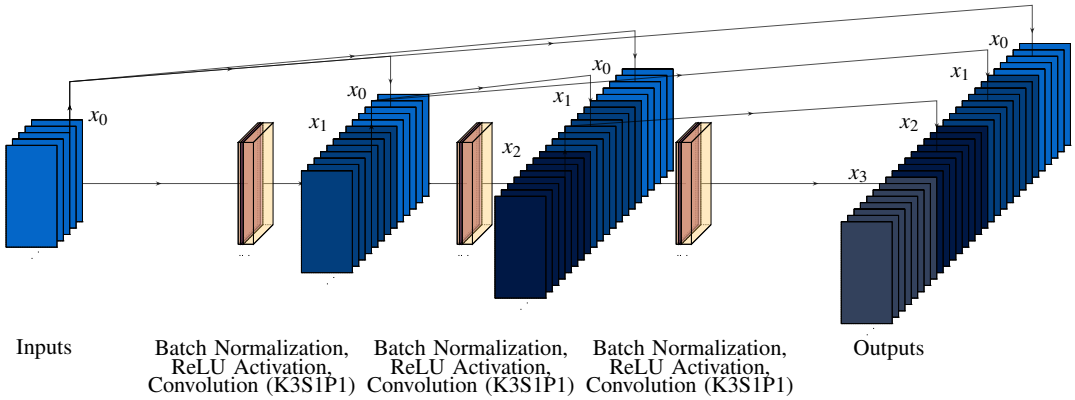


Figure 2: Dense block based on the DenseNet [33] architecture with 3 layers ($L = 3$) and a growth rate of 8 ($K = 8$). The previous feature maps, represented as blue-hued planes in the figure, are appended to the output from each block of batch normalization, ReLU activations, and convolution using a kernel with kernel size (K) of 3, stride (S) of 1, and padding (P) of 1. The feature map sizes remain constant through the dense block, while the number of feature maps grows by K through each layer.

Fully convolutional networks (FCNs) [49] are a class of neural networks that have been used for image-to-image regression tasks. These networks allow for pixel-wise predictions by replacing the fully connected layers with convolutional layers and including up-sampling (decoder) layers to recover the input size. The use of skip connections between down-sampling (encoder) sections and decoder sections enables the preservation of information from the encoding process. Recent works like U-Net [28], Segnet [50], and FPNs [51] have demonstrated the effectiveness of FCNs. By using both encoder and decoder sections, FCNs can learn both low-level and high-level features, while skip connections facilitate the flow of information across multiple scales.

To leverage the advantages of both dense convolutional networks and FCNs, researchers have extended DenseNet to create a fully convolutional architecture [41]. To address varying feature map sizes in encoder-decoder architectures, transition layers, illustrated in Figures 3a and 3b, and dense blocks (Figure 2) were defined [40, 41]. Transition layers, whether down-sampling (encoder) or up-sampling (decoder), serve to connect dense blocks and make the network modular. The DenseED architecture, developed by Zhu and Zabaras [40], was adapted from the fully convolutional DenseNet architecture [41] and tailored for surrogate modeling of PDE-based systems by introducing several key changes: (i) the number of feature maps was reduced using the first layer in each transition layer, (ii) all feature maps from previous layers were kept in each dense block, (iii) no skip connections between the encoder and decoder sections were used, and (iv) convolution with stride 2 was used for down-sampling. The DeepEDH architecture proposed in this work leverages the majority of these changes; however, we introduce several additional enhancements: (i) skip connections between encoder and decoder sections due to the stronger correspondence between the input and output for CHT analyses, (ii) dropout for regularization, (iii) output geometry masks to account for the direct correspondence between the inputs and outputs for flow predictions, and (iv) a two-stage model for temperature prediction to take advantage of the coupling between velocity and temperature fields in CHT analysis.

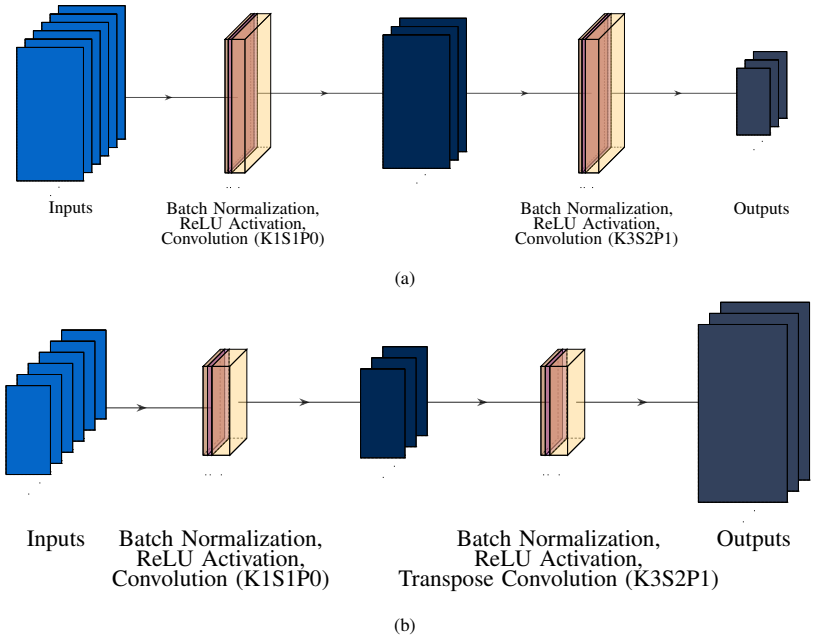


Figure 3: (a) Encoding layer with convolution and (b) decoding layer with convolution and transpose convolution. The first combination of batch normalization, ReLU activations, and convolution for both encoding and decoding layers reduce the number of feature maps, represented as blue-hued planes in the figure, by half using a kernel with kernel size (K) of 1, stride (S) of 1, and padding (P) of 0. The second combination uses a kernel with K=3, S=2, and P=1 to reduce the feature map sizes by half for encoding with convolution layers and doubles the feature map sizes for decoding with transpose convolution layers.

3.2.2. Output geometry mask

An output geometry mask is introduced to our architecture to ensure that solid domain regions are correctly represented in the model output for pressure and velocity predictions. We create the output geometry mask, denoted as $\boldsymbol{\delta}$, by inverting the input image. In the input image, solid regions have a value of 1, while the fluid regions are set to 0. Hence, we define the mask such that $\boldsymbol{\delta} = 1 - \boldsymbol{x}$, where $\boldsymbol{x}$ is the input image. Using the output geometry mask, the final pressure ($\hat{P}$) and velocity ($\hat{\boldsymbol{u}}$) outputs of the surrogate model $\hat{\boldsymbol{y}}_{\text{fluid}}$ are defined using element-wise multiplication with the mask: $\hat{\boldsymbol{y}}_{\text{fluid}} = \boldsymbol{\delta} \odot \hat{\boldsymbol{y}} = \boldsymbol{\delta} \odot \hat{f}(\boldsymbol{x}, \boldsymbol{X}_s, \boldsymbol{\theta})$. Here $\odot$ is the element-wise multiplication operator. This ensures that the predicted pressure and velocity fields are only valid in the fluid domain, improving model accuracy for CHT and flow analyses.

3.2.3. Two-stage temperature model

Due to the significant influence of convection in CHT analyses, temperature exhibits a strong dependence on the velocity field. In order to leverage this interdependence and improve the temperature predictions, we have developed a two-stage model. In this approach, the first stage utilizes the output of the velocity model, which is then incorporated into the input of the second-stage model for temperature prediction. This enables the network to learn the relationship between the temperature and velocity fields, resulting in improved temperature predictions. The velocity model produces an output denoted as $\hat{\boldsymbol{y}}_{\text{vel}} = \hat{f}_{\text{vel}}(\boldsymbol{x}, \boldsymbol{X}_s, \boldsymbol{\theta}_{\text{vel}})$. This output is used as input for the temperature model, represented as $\hat{\boldsymbol{y}}_{\text{temp}} = \hat{f}_{\text{temp}}(\boldsymbol{x}, \boldsymbol{X}_s, \hat{\boldsymbol{y}}_{\text{vel}}, \boldsymbol{\theta}_{\text{temp}}) = \hat{f}_{\text{temp}}(\boldsymbol{x}, \hat{f}_{\text{vel}}(\boldsymbol{x}, \boldsymbol{X}_s, \boldsymbol{\theta}_{\text{vel}}), \boldsymbol{\theta}_{\text{temp}})$.

3.2.4. Final architecture: DeepEDH

This work introduces the new neural network architecture called DeepEDH (modular deep encoder-decoder hierarchical convolutional neural network). DeepEDH uses principles from fully convolutional networks similar to U-Net [28], Segnet [50], and FPNs [51], with dense blocks and skip connections between encoder and decoder sections, similar to fully convolutional DenseNet [41]. We use the changes stated above to fully convolutional DenseNet proposed in DenseED [40]. However, we include skip connections and dropout. DeepEDH is defined by the depth of the network (number of dense blocks), the number of layers within each dense block, the growth rate K through each dense block, and the number of feature maps after the initial convolutional layer. The depth and the number of layers in each dense

block can be defined using a list, L_{dense}. The length of this list defines the network depth, while the value at each index determines the number of layers in each dense block. It is important to note that the length of L_{dense} must be an odd integer, where the middle value defines the number of layers in the bottleneck dense block. An example of DeepEDH with $L_{\text{dense}} = [3, 4, 3]$ and $K = 2$ is shown in Figure 4.

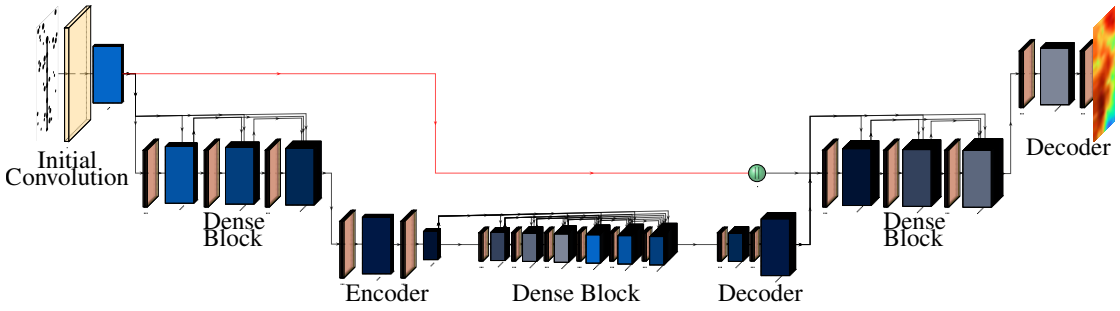


Figure 4: Network architecture: DeepEDH with $L_{\text{dense}} = [3, 4, 3]$ and $K = 2$. This example includes 1 encoding dense block with 3 layers, a bottleneck dense block with 4 layers, and 1 decoding dense block with 3 layers. The number of feature maps, represented as blue-hued planes in the figure, grows by two through each dense block layer as defined by the growth rate.

The encoding section of the network takes the input: $\boldsymbol{X}_s$ for velocity and pressure models and $(\boldsymbol{X}_s, \hat{\boldsymbol{y}}_{\text{vel}})$ for temperature models. An initial convolutional layer is applied to reduce the size of the input by half and extract initial feature maps. The initial feature maps pass through pairs of dense blocks (Figure 2), and encoding transition layers (Figure 3a) until reaching the bottleneck block (bottom of Figure 4), from which code dimension feature maps are extracted. Subsequently, the code dimension feature maps progress through pairs of decoding transition layers (Figure 3b) and dense blocks (Figure 2), leading to the final decoding layer that directly predicts the output. For the pressure and velocity models, the output geometry mask is applied to the network output. In contrast, for temperature predictions, the velocity model output is passed as input in the two-stage architecture.

3.3. Network hyperparameters, training loss, and regularization

To determine the hyperparameters for the DeepEDH architecture, we based our initial values on previous studies such as [40, 41]. We then performed hyperparameter and architecture optimization with a combination of Bayesian optimization (BO) and Hyperband (HB) with the BOHB algorithm [52]. The hyperparameters considered for optimization include the dropout rate, the number of layers in the encoding, bottleneck, and decoding dense blocks, the growth rate, the number of feature maps after the initial convolution layer, the initial learning rate, the learning rate weight decay, and the batch size. It is important to note that hyperparameters were optimized separately for each field. Full details of the architecture and training hyperparameter optimizations are included with the supplementary materials.

During the training process, we minimized the difference between the structured target field $\boldsymbol{y}$, generated by simulation, and the model predicted field $\hat{\boldsymbol{y}}$ for all $N_{\text{cells}} = n_x \cdot n_y$ pixels. We used a regularized mean squared error (MSE) training loss function, as defined in Equation (15).

$$L(\hat{f}(\boldsymbol{x}, \boldsymbol{X}_s, \boldsymbol{\theta}), \boldsymbol{y}) = \frac{1}{N_{\text{cells}}} \sum_{i=1}^{N_{\text{cells}}} \left(\hat{f}(\boldsymbol{x}, \boldsymbol{X}_s, \boldsymbol{\theta})_i - \boldsymbol{y}_i \right)^2 + \alpha \Omega(\boldsymbol{\theta}) \tag{15}$$

This loss function incorporates $\mathcal{L}_2$ regularization with $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}$, defined using weight decay (α) in PyTorch. We used both batch normalization [47] and dropout [53] for additional regularization. The network parameters $\boldsymbol{\theta}$ include the weights of the convolutional and transpose convolutional layer kernels, as well as the scale and shift parameters in the batch normalization layers.

4. Surrogate implementation and training

This section details the implementation of the surrogate modeling methodology and DeepEDH architecture presented in Section 3, considering a liquid-cooled cold-plate-based battery thermal management system. DeepEDH neural network surrogate models were developed as replacements for computational FEM simulations to predict the pressure,

velocity, and temperature fields of the CHT analysis based on the cold plate channel geometry. The dataset generation, data transformations, network architecture, and training details are presented for the test case.

4.1. Problem definition

The surrogate model implementation considered a pin-fin cold plate thermal management system, shown in Figure 5a. The cold plate was designed to cool a battery module made up of 30 pouch cells, where coolant flows into the cold plate through the inlet tube, around the central wall and pin-fin arrangement, and exits through the outlet tube. To simulate a transient battery discharge process with non-uniform heat generation from the battery module, a spatially varying heat flux boundary condition was applied to the top surface of the cold plate. Here, we considered heat flux based on a 3C charge process, where each battery cell in the battery module was charged at a constant current of 3 times its rated capacity. This is representative of a battery fast charge process [45, 54], with heating rates ranging from 600 W to 900 W, amounting to a total heat generation of 2,693,403 kJ considering the 30 pouch cell battery module. An inlet flow rate of 3 LPM with a temperature of 20 °C was used for the cold plate, leading to laminar flow with a channel Reynolds number of approximately 1000.

The cold plate channel geometry, made up of the pin-fin arrangement, is defined by the position of the pin-fin centers (x_i, y_i) and the pin-fin radii r_i for $i = 1, 2, \ldots, n_{\text{pins}}$ pin-fins. Using the channel geometry as input to the DeepEDH surrogate models, we predicted pressure and velocity fields at the mid-plane of the cold plate channel and the temperature field at the surface of the cold plate for the final time step of the transient simulation.

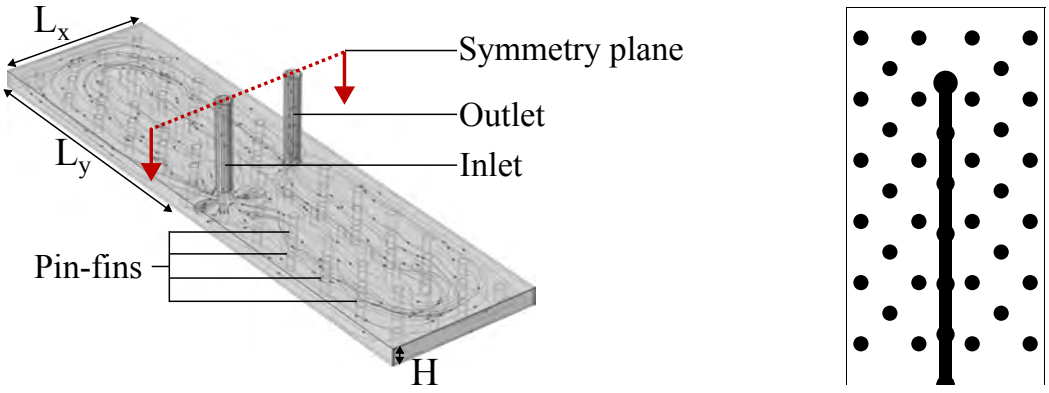


Figure 5: Pin-fin cold plate (a) three-dimensional geometry and (b) channel cross-sectional view with the solid metal region shown in black and the flow region shown in white. Symmetry was applied across the width of the cold plate with (b) showing half of the plate.

The cold plate channel cross-sectional geometry, as shown in Figure 5b, depicts the pin-fin solid regions in black and the fluid region in white. The domain size was fixed at approximately $L_x \approx 0.25$ m and $L_y \approx 0.5$ m, with a constant channel height of $H = 0.005$ m. The design variables included the positions and radii of the pin-fins. These design variables were constrained to ensure that there was no overlap between solid regions. The pin-fin radii were further constrained between $r_{\text{min}} = 0.005$ m and $r_{\text{max}} = 0.015$ m, and the system consisted of $n_{\text{pins}} = 34$ pin-fins. A symmetry boundary condition (Equation (6)) was applied across the width of the cold plate through the inlet and outlet tubes to reduce the domain size. Mass flow (Equation (4)) and pressure (Equation (5)) boundary conditions were applied to the inlet and outlet respectively. No-slip wall (Equation (3)) boundary conditions were used for the internal channel walls. The cold plate was thermally insulated on the bottom (Equation (8) with $q_0 = 0$), while natural convection boundary conditions were applied to the other external cold plate walls using vertical and horizontal wall correlations. The top surface used the spatially and time-varying heat flux boundary condition (Equation (8)) determined from a battery discharge process. Heat inflow and outflow conditions Equations (9) and (10) were specified at the inlet and outlet.

For the numerical solution of the CHT problem, three-dimensional unstructured meshes were generated and solved using the FEM for each geometry in COMSOL Multiphysics 6.0 with the conjugate heat transfer physics. From the three-dimensional FEM simulation results the velocity and pressure fields were extracted at the mid-plane of the cold plate channel depth, while the temperature field was extracted at the surface of the cold plate. The DeepEDH models were developed based on these simulation results to act as surrogates for the FEM models.

4.1.1. Datasets

A set of 1,500 pin-fin arrangements was defined using Latin hypercube sampling, where each arrangement is characterized by the positions and radii of the pin-fins. Several examples of these pin-fin arrangements are displayed in Figures 6a to 6d.

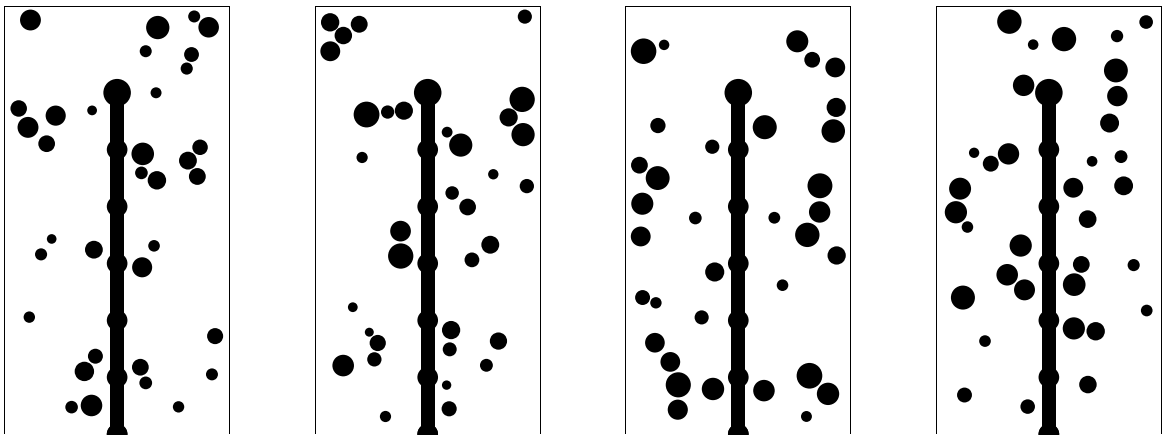


Figure 6: Channel cross-sectional views of selected pin-fin cold plate geometry sample points from the Latin Hypercube sample plan. Solid metal regions are shown in black and the flow region is shown in white.

For each pin-fin arrangement, the FEM CHT simulations determined the output fields at each time step. These output fields include the pressure (p) and velocity components (u, v, and w) at the mid-plane of the cold plate channel, as well as temperature (T) at the heated surface of the cold plate. Parallel computing enabled 50 FEM models to be executed simultaneously, with each simulation taking approximately 0.3 - 0.5 hours, allowing each dataset to be generated in approximately 10 hours. Each simulation, run in parallel, used 48 GB of memory, 2 GB of storage, and 12 cores on a high-performance computing cluster.

It is important to note that not all of the samples defined by the sampling plan resulted in converged numerical solutions. Those non-converged solutions were excluded from the final dataset. The complete dataset consists of approximately 1,500 simulations, considering only those with converged solutions. These simulations included various input pin-fin arrangements, heat flux profiles, and output fields for each time step. An example of a single simulation sample point is presented in Figures 7a to 7g.

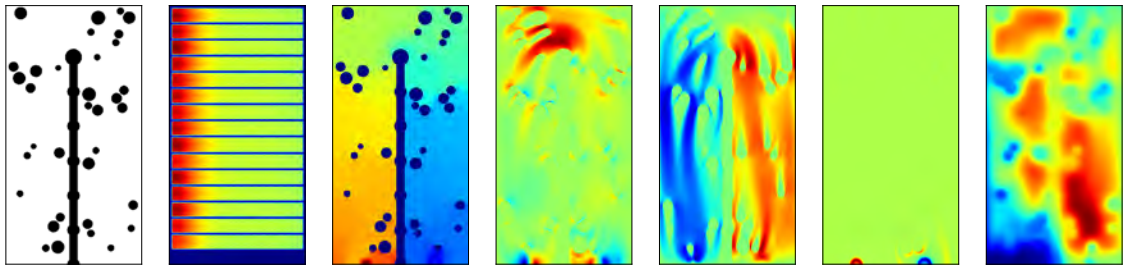
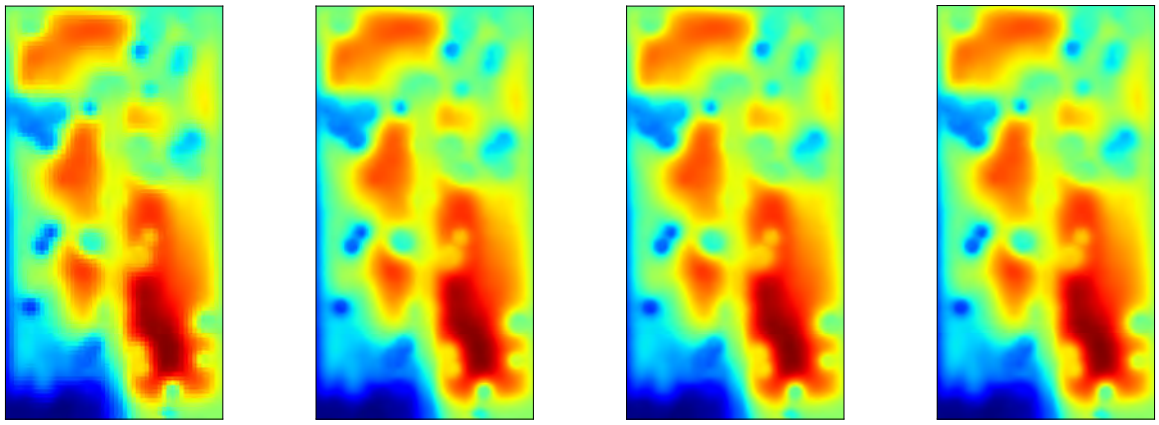


Figure 7: Dataset example: (a) geometry input, (b) heat flux input, (c) pressure output, (d) u velocity output, (e) v velocity output, (f) w velocity output, and (g) temperature output. Pressure and velocity field outputs were extracted at the mid-plane of the cold plate channel depth, while the temperature field was extracted at the surface of the cold plate

The simulation results were based on an unstructured mesh of approximately 3,000,000 cells. This mesh was composed of approximately 250,000 cells for fluid cross-sections and approximately 200,000 cells for solid cross-sections. The results were processed using the data transformations described in Section 2.2 to structure the results and generate datasets for training and testing the surrogate models. Four data resolutions were considered, as shown in Figures 8a

to 8d. These resolutions were determined by selecting n_x and calculating the appropriate n_y to ensure that the structured cells are physically square. The resulting data resolutions were 50×95 with 4,750 cells, 100×190 with 19,000 cells, 200×380 with 76,000 cells, and 400×761 with 304,400 cells, representing cell sizes of approximately 5.2 mm/px, 2.6mm/px, 1.3 mm/px, and 0.65 mm/px respectively. In relation to the original unstructured mesh, these data resolutions represent approximately 2%, 8%, 33%, and 133% of the original mesh size, providing a range of data scaling.



(a) (b) (c) (d)

Figure 8: Data resolutions, shown here for a sample temperature field: (a) 50×95, (b) 100×190, (c) 200×380, and (d) 400×761.

4.2. Surrogate model training

DeepEDH models were developed and trained to predict the pressure, velocity, and temperature fields, resulting in three separate DeepEDH field models: DeepEDH-pressure, DeepEDH-velocity, and DeepEDH-temperature. DeepEDH-temperature used a two-stage architecture, coupling the velocity and temperature fields. The network architecture for each model was determined through the results of network characterization, hyperparameter optimization, and architecture optimization. The final architectures for each field model are outlined in Table 1. The outputs of each layer depend on various factors, including the depth of the network, the number of layers in each dense block, and the growth rate K. Each dense block increases the number of feature maps by $K \cdot L$, while the transition layers reduce the number of feature maps by half. The encoding transition layers halve the feature map size, while decoding transition layers double the feature map size.

Table 1: Network architecture details for DeepEDH-pressure, DeepEDH-velocity, and DeepEDH-temperature models.

Network Section	Layer	Layer Parameters			Output
		DeepEDH-pressure	DeepEDH-velocity	DeepEDH-temperature	
Initial Convolution	Inputs	1	1	2	$(n_x, n_y,$ Inputs$)$
	Convolution (K7S2P3)	Output (IC): 16	Output (IC): 48	Output (IC): 16	$(\frac{n_x}{2}, \frac{n_y}{2}, 1,$ IC$)$
Encoder	Dense Blocks (n_{enc})	$K_{enc} = 32$, $L_{enc} = 12$	$K_{enc} = 16$, $L_{enc} = 7$	$K_{enc} = 16$, $L_{enc} = 5$	$\left(\frac{n_x}{2^{i_{enc}}}, \frac{n_y}{2^{i_{enc}}}, \frac{IC+(K_{enc}\times L_{enc})\cdot i_{enc}}{2^{i_{enc}-1}}\right)$ for $i_{enc} = 1$ to n_{enc}
	Encodings (n_{enc})	BN, ReLU, K1S1P0 & BN, ReLU, K3S2P1			$\left(\frac{n_x}{2^{i_{enc}+1}}, \frac{n_y}{2^{i_{enc}+1}}, \frac{IC+(K_{enc}\times L_{enc})\cdot i_{enc}}{2^{i_{enc}}}\right)$ for $i_{enc} = 1$ to n_{enc}
Bottleneck	Dense Block	$K_{bot} = 32$, $L_{bot} = 5$	$K_{bot} = 16$, $L_{bot} = 3$	$K_{bot} = 16$, $L_{bot} = 3$	$\left(\frac{n_x}{2^{n_{enc}+1}}, \frac{n_y}{2^{n_{enc}+1}}, FM_{bot}\right)$; $FM_{bot} = \frac{IC+(K_{enc}\times L_{enc})\cdot n_{enc}}{2^{n_{enc}}} + K_{bot} \times L_{bot}$
Decoder	Decodings (n_{dec})	BN, ReLU, K1S1P0 & BN, ReLU, Transpose K3S2P1			$\left(\frac{n_x}{2^{n_{enc}-i_{dec}+1}}, \frac{n_y}{2^{n_{enc}-i_{dec}+1}}, \frac{FM_{bot}+(K_{dec}\times L_{dec})\cdot(i_{dec}-1)}{2^{i_{dec}}}\right)$ for $i_{dec} = 1$ to n_{dec}
	Dense Blocks (n_{dec})	$K_{dec} = 32$, $L_{dec} = 9$	$K_{dec} = 16$, $L_{dec} = 10$	$K_{dec} = 16$, $L_{dec} = 10$	$(\frac{n_x}{2^{n_{enc}-i_{dec}+1}}, \frac{n_y}{2^{n_{enc}-i_{dec}+1}}, \frac{FM_{bot}+(K_{dec}\times L_{dec})\cdot(i_{dec}-1)}{2^{i_{dec}}}$ $+(K_{dec} \times L_{dec}))$ for $i_{dec} = 1$ to n_{dec}
Outputs		1			$(n_x, n_y, 1)$

The data was split into three subsets: 80% for training $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n_{\text{train}}}$, 10% for testing $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n_{\text{test}}}$, and 10% for validation $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n_{\text{validation}}}$. To prepare data for training and testing, data transformations were applied to the datasets, including

minimum-maximum scaling and Z-score standardization, as defined in Equations (16) and (17).

$$\boldsymbol{y}_{\text{scaled}} = \frac{\boldsymbol{y} - y_{\min}}{y_{\max} - y_{\min}} \tag{16}$$

$$\boldsymbol{y}_{\text{standardized}} = \frac{\boldsymbol{y} - \overline{\boldsymbol{y}}}{\sigma_{\boldsymbol{y}}} = \frac{\boldsymbol{y} - \overline{\boldsymbol{y}}}{\sqrt{\frac{\sum_{i=1}^{n} \boldsymbol{y}_i - \overline{\boldsymbol{y}}}{n}}} \tag{17}$$

where $y_{\min}$, $y_{\max}$, $\overline{\boldsymbol{y}}$, and $\sigma_{\boldsymbol{y}}$ are the minimum, maximum, mean, and standard deviation values of the target field $\boldsymbol{y}$ based on the training data. Minimum-maximum scaling bounds the data between 0 and 1, while Z-score standardization results in a distribution with a mean of 0 and a standard deviation of 1. We found that applying standardization to temperature and velocity outputs and scaling to temperature inputs with no transformations for pressure data gave the best performance for each model. Before evaluating the performance of the models, inverse data transforms were applied to the predictions to revert them to their original scale.

The training loss function (Equation (15)) was minimized by tuning the model parameters $\boldsymbol{\theta}$ from Equation (14). Gradients of the training loss function with respect to $\boldsymbol{\theta}$ were computed through backpropagation in the DeepEDH models. The adaptive moment estimation (ADAM) [55] optimization algorithm was used with an initial learning rate of 10^{-3}, weight decay of 10^{-4}, and batch size of 8. A learning rate scheduler that reduced the learning rate on a loss plateau was used with a relative tolerance of 1×10^{-4}, decay factor of 10, and patience of 10 epochs. This means that if the loss did not decrease relatively by 1×10^{-4} for 10 epochs, the learning rate was reduced by a factor of 10. The training process was carried out over 500 epochs, with sample learning curves provided in Figures 9a to 9c for each model and data resolution.

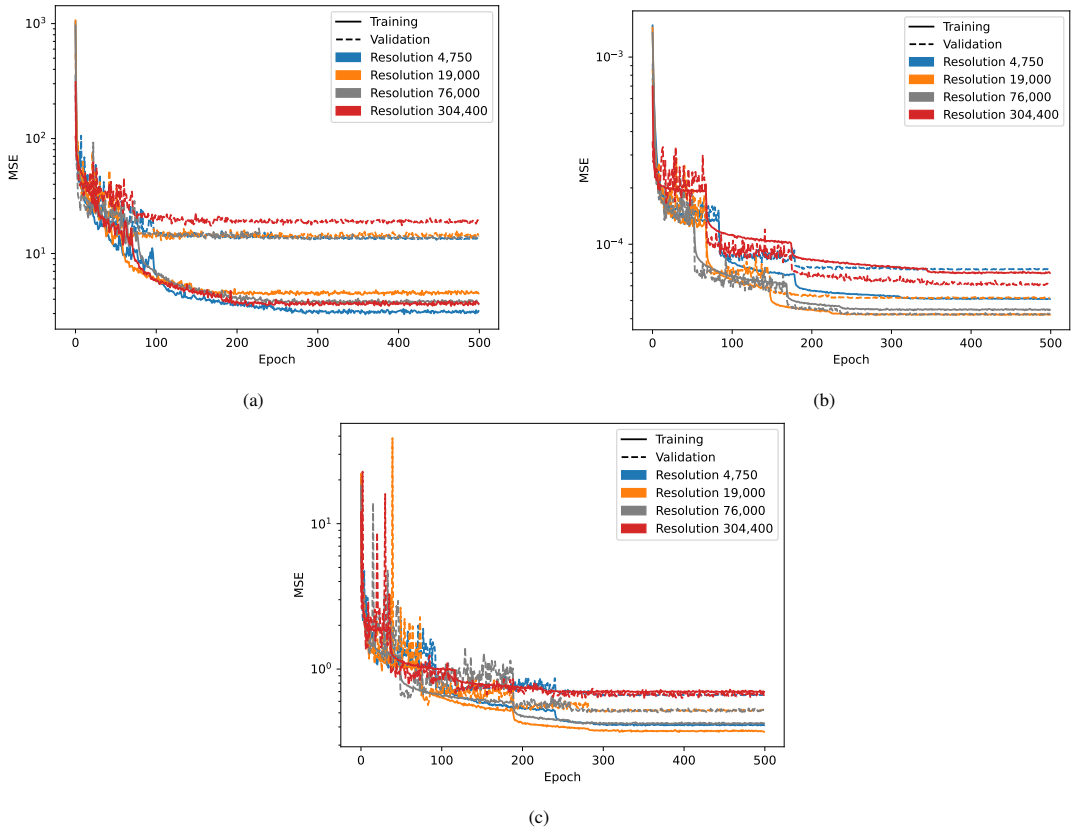


Figure 9: Training curves for (a) DeepEDH-pressure [Pa^2], (b) DeepEDH-velocity [$\text{m}^2\,\text{s}^{-2}$], and (c) DeepEDH-temperature [K^2] models. These curves show loss computed from the validation dataset.

Figures 9a to 9c indicate good convergence of the training loss for each model and do not show signs of over-fitting.

The full surrogate modeling methodology, from data generation and processing, to final field prediction is outlined in Figure 10.

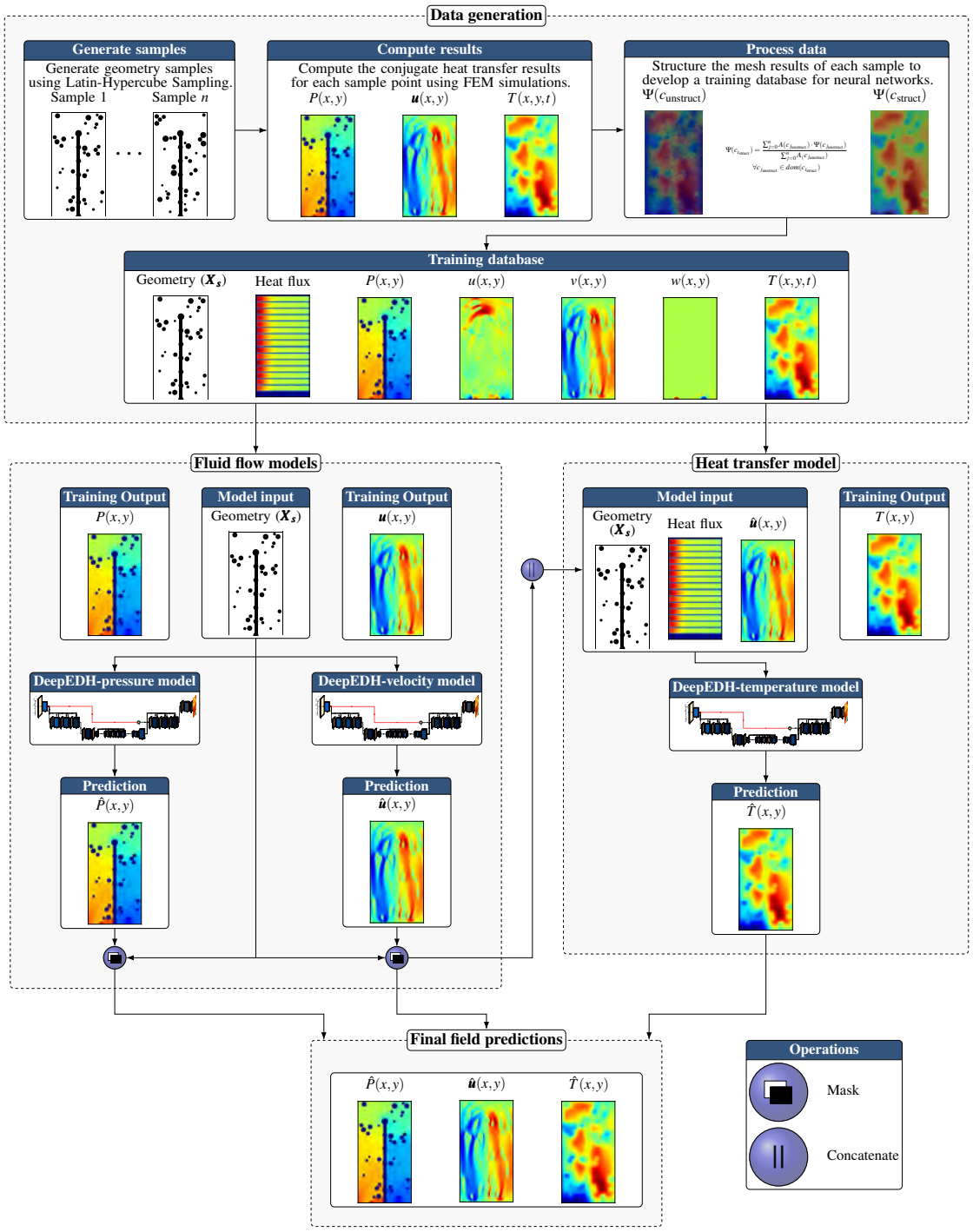


Figure 10: Overview of surrogate methodology from data generation and processing, to final field prediction with the output geometry mask and two-stage temperature architecture.

5. Surrogate model performance and characterization

We first characterized the effect of network depth and dataset size on the surrogate model performance before completing hyperparameter and architecture optimization to determine the final architectures of each DeepEDH model. To demonstrate the advantage of the surrogate methodology proposed in this work, we compared the performance of our DeepEDH models with U-Net [28] and DenseED [40] models, specifically examining the impact of each change. We characterized the effect of the DeepEDH network architectures, output geometry masks, two-stage temperature architecture, and hyperparameter and architecture optimization. Finally, we present the impact of the heat flux magnitude on the temperature model's performance, demonstrating excellent performance for a range of thermal boundary conditions.

5.1. Evaluation metrics

To evaluate and characterize the performance of the surrogate models, we considered three metrics based on the test dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n_{\text{test}}}$. The coefficient of determination (R^2), calculated using Equation (18), quantifies the proportion of variance in the data that the model can explain.

$$R^2 = 1 - \frac{\sum_{i=1}^{n_{\text{test}}} \left\| \boldsymbol{y}_i - \hat{\boldsymbol{y}}_i \right\|_2^2}{\sum_{i=1}^{n_{\text{test}}} \left\| \boldsymbol{y}_i - \overline{\boldsymbol{y}} \right\|_2^2} \tag{18}$$

The mean of the target fields, $\boldsymbol{y}$, is $\overline{\boldsymbol{y}}$, while $\hat{\boldsymbol{y}}$ are the surrogate model outputs. The R^2 value ranges from 0 to 1, with 1 indicating the model can explain 100% of the variance in data while 0 shows no correlation between the model prediction and the target field. The root mean square error (RMSE), defined by Equation (19), measures the average error between the model predictions and target fields.

$$RMSE = \sqrt{\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \left\| \hat{\boldsymbol{y}}_i - \boldsymbol{y}_i \right\|_2^2} \tag{19}$$

The RMSE metric is expressed in the same units as the target field, and a lower RMSE indicates a model with a lower average error. Spearman's rank correlation coefficient (SCC), Equation (20), assesses the ordered correlation between the model predictions and the target field.

$$SCC = \frac{\sum_{i=1}^{n_{\text{test}}} \left(R(\boldsymbol{y}_i) - \overline{R(\boldsymbol{y})} \right) \cdot \left(R(\hat{\boldsymbol{y}}_i) - \overline{R(\hat{\boldsymbol{y}}_i)} \right)}{\sqrt{\sum_{i=1}^{n_{\text{test}}} \left(R(\boldsymbol{y}_i) - \overline{R(\boldsymbol{y})} \right)^2} \cdot \sqrt{\sum_{i=1}^{n_{\text{test}}} \left(R(\hat{\boldsymbol{y}}_i) - \overline{R(\hat{\boldsymbol{y}}_i)} \right)^2}} \tag{20}$$

here $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ are converted to their corresponding rank variables $R(\boldsymbol{y}_i)$ and $R(\hat{\boldsymbol{y}}_i)$, while the means of the rank variables are $\overline{R(\boldsymbol{y})}$ and $\overline{R(\hat{\boldsymbol{y}})}$. The SCC value ranges from -1 to 1 and measures the monotonic relationship, with 1 indicating a positive correlation and -1 indicating an inverse correlation. The SCC is particularly useful in applications where preserving the relative rank between alternatives is crucial, such as design optimization. These three metrics collectively offer a comprehensive evaluation of the surrogate model's performance. The R^2 and SCC values provide insights into the overall regression performance and correlation, while RMSE quantifies the average prediction error.

5.2. Characterization results

To characterize the model performance, we examined the effect of the network depth, dataset size, data resolution, and thermal boundary condition magnitude. All three metrics introduced in Section 5.1 show similar trends. Hence, characterization results in this section are presented with the R^2 metric, while the RMSE and SCC results are included in the supplementary materials. Additionally, we include model field predictions for the extreme values of each characterization parameter and tabulated data for all results presented in this section in the supplementary materials.

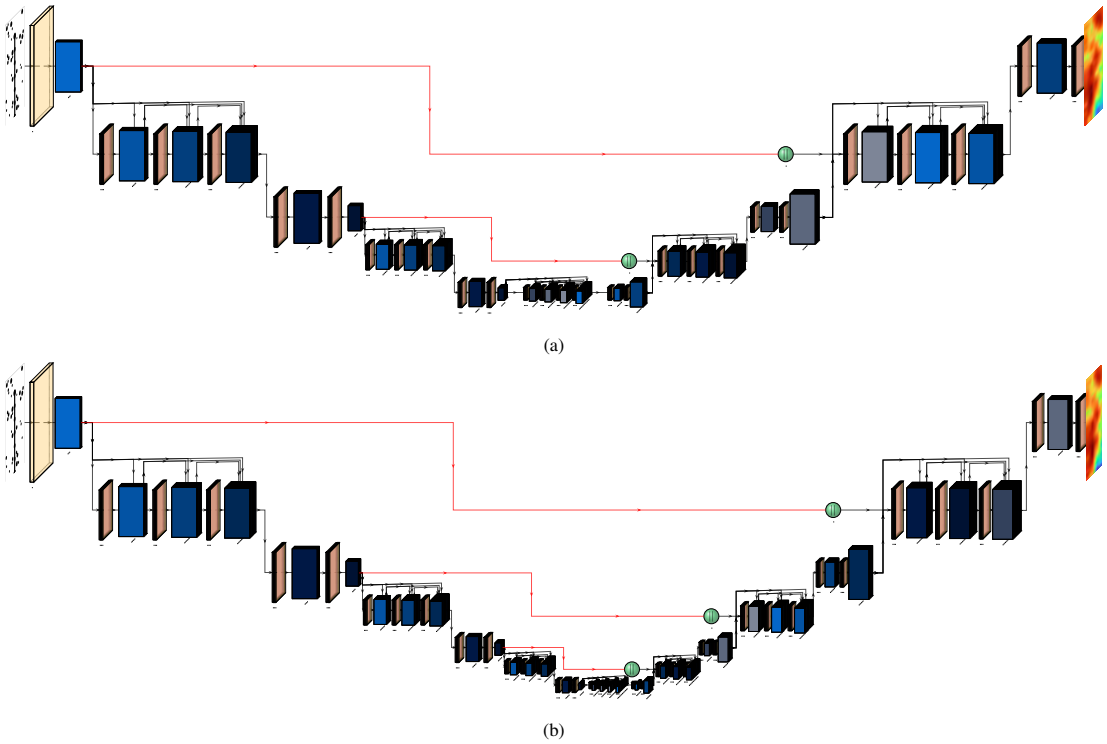
5.2.1. Code dimension

The term code dimension, in the context of the DeepEDH network, refers to the spatial dimension or number of pixels in the feature maps at the network's bottleneck layer. The number of dense blocks used in the DeepEDH network and data resolution define the code dimension. Unlike fully connected networks, where the entire input affects each unit in the network, units within a convolutional network are only dependent on a region of the input, while only a region

of the output impacts each unit in the network during backpropagation. This region is referred to as the receptive field or field of view [56] and is related to the code dimension, which represents a mapping of the input geometry to the output field.

Smaller code dimensions correspond to deeper networks, mapping the input field to a lower dimension space, resulting in more information loss. However, by implementing skip connections, this information can be retained, allowing smaller code dimensions to generate more features and capture fine-grained information from the input geometry. This results in less smooth outputs. Conversely, these deeper networks require more parameters, which increases their computational cost and makes them more difficult to train. In comparison, shallower networks with larger code dimensions and fewer parameters can capture only more high-level or coarse information, resulting in smoother outputs. For the dense prediction task considered here, the code dimension must be a suitable size such that the receptive field can capture the required information from the input and output images, while limiting the network size for computational efficiency and ease of training. The code dimension enables some physical interpretations, corresponding to the spatial resolution of input features that impact the output field. With a code dimension that is too large, the model cannot capture the small features from the input geometry that impact the output fields. Hence, investigating model performance across different code dimensions allows the minimum geometry size that impacts output fields to be determined.

When comparing results across different data resolutions, it is important to maintain a consistent code dimension. Therefore, different network depths are required for each resolution. For example, for a 50×95 data resolution with a DeepEDH architecture defined as $L_{\text{dense}} = [3, 4, 3]$, shown in Figure 4, the resulting code dimension is 312 pixels with dimensions 13×24. The input size of 50×95 is halved by the initial convolution layer, then halved again by the encoding layer, resulting in the corresponding code dimension of 13×24. Figure 11 shows DeepEDH architectures with varying depths of $L_{\text{dense}} = [3, 3, 4, 3, 3]$ in Figure 11a and $L_{\text{dense}} = [3, 3, 3, 4, 3, 3, 3]$ in Figure 11b.



(a)

(b)

Figure 11: DeepEDH architecture with varying depths: (a) 5 dense blocks with $L_{\text{dense}} = [3, 3, 4, 3, 3]$ and $K = 2$, and (b) 7 dense blocks with $L_{\text{dense}} = [3, 3, 3, 4, 3, 3, 3]$ and $K = 2$.

The network architectures in Figures 11a and 11b result in code dimensions of 84 and 24 pixels, respectively, for the 50×95 data resolution. To achieve code dimensions of 312, 84, and 24, the dataset with 400×761 resolution would require 9, 11, and 13 dense blocks. We considered code dimensions ranging from 1200 pixels, based on a single dense block for the 50×95 dataset, to 2 pixels. A summary of the code dimensions for each data resolution with the

corresponding number of dense blocks is shown in Table 2.

Table 2: Network depth, data resolution, and corresponding code dimension.

Dense Blocks	Data Resolution (Cells [W×H])			
	4,750 [50×95]	19,000 [100×190]	76,000 [200×380]	304,400 [400×761]
1	1,200 [25×48]	-	-	-
3	312 [13×24]	1,200 [25×48]	-	-
5	84 [7×12]	312 [13×24]	1,200 [25×48]	-
7	24 [4×6]	84 [7×12]	312 [13×24]	1,200 [5×48]
9	6 [2×3]	24 [4×6]	84 [7×12]	312 [13×24]
11	2 [1×2]	6 [2×3]	24 [4×6]	84 [7×12]
13	-	2 [1×2]	6 [2×3]	24 [4×6]
15	-	-	2 [1×2]	6 [2×3]
17	-	-	-	2 [1×2]

To investigate the impact of code dimension, we used the same network architecture for each DeepEDH model, with the code dimension defined by the number of dense blocks. Here, architectures used encoding and decoding dense blocks with 3 layers and a bottleneck block with 6 layers; all blocks had a growth rate of 16. The models considered 48 initial feature maps and a dropout of 0.05. The networks were trained for 500 epochs with a batch size of 16, learning rate of 3×10^{-3}, and learning rate decay of 5×10^{-4}. For each data resolution, models were trained with the appropriate number of dense blocks to achieve code dimensions ranging from 1200 to 2 pixels as defined in Table 2. Figures 12a to 12c show the R^2 results for each field and data resolution.

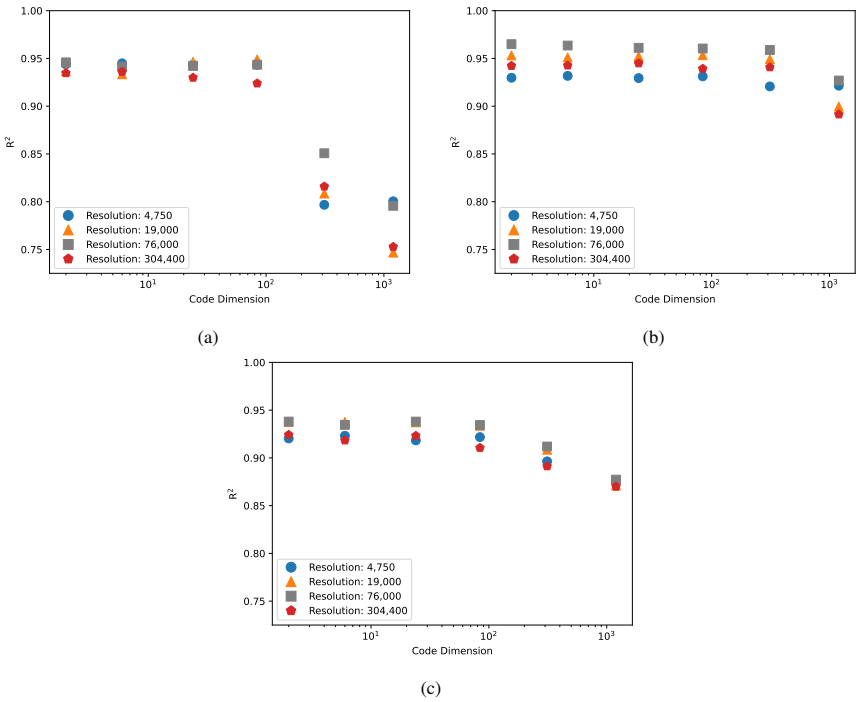


Figure 12: Model code dimension characterization R^2 results for: (a) DeepEDH-pressure, (b) DeepEDH-velocity, and (c) DeepEDH-temperature models.

Figures 12a to 12c show that the code dimension has a significant impact on the model performance - specifically at larger code dimensions, which result in poor performance for all fields and data resolutions. This is due to the inability of the higher code dimension models to capture small input features and the resulting smooth nature of the output

fields. However, as the code dimension is reduced, model performance improves across all fields and data resolutions before plateauing at a code dimension of 24 pixels. Based on the physical interpretation of the code dimension, this is expected: as the code dimension decreases the minimum geometry size that impacts the output fields is captured. Further refinement past this point does not improve model performance, as the model already captures the necessary information from the input and output fields. For optimal performance, a code dimension small enough to capture the minimum information scale is required. However, this must be balanced against the higher computational training cost and training difficulty associated with the increased number of network parameters from smaller code dimensions. Considering these results, future experiments use a code dimension of 24 pixels to limit the number of network parameters while maintaining model performance.

5.2.2. *Dataset size*

Dataset size refers to the number of simulations in the dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n_{\text{train}}}$ used to train the surrogate models. For surrogate modeling, generating the dataset is typically the most computationally expensive step. Increased dataset size also results in longer training times. However, this is generally less significant than the simulation time required to generate the dataset. By selecting an appropriate dataset size we can reduce the computational cost of generating the dataset while maintaining good model performance. We considered dataset sizes ranging from 25 to 1000 simulations, corresponding to approximately 1.5% and 66% of the total pre-generated dataset size. Figures 13a to 13c show the R^2 results for each field, data resolution, and dataset size.

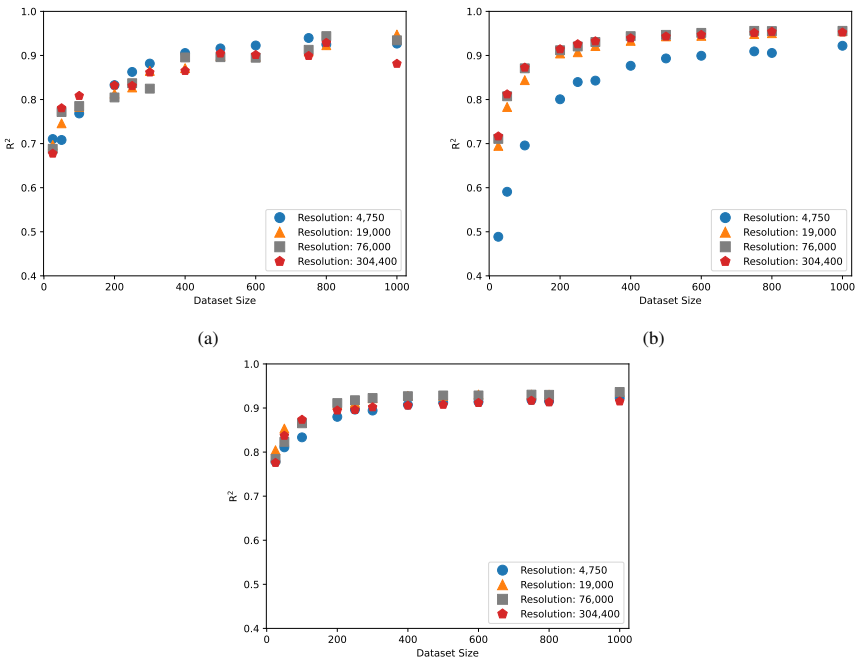


Figure 13: Model training dataset size and dimension characterization R^2 results for: (a) DeepEDH-pressure, (b) DeepEDH-velocity, and (c) DeepEDH-temperature models.

Figures 13a to 13c show that increasing the dataset size results in improved DeepEDH model performance with diminishing returns. Increases from 25 to approximately 200 simulations significantly improve model performance, while performance plateaus at approximately 500 simulations.

5.2.3. *Data resolution*

From the results presented in Sections 5.2.1 and 5.2.2, we determined that a code dimension of 24 pixels and dataset size of 500 simulations resulted in good model performance while limiting the computational cost of training. We have considered data resolutions ranging from 50×95 to 400×761, corresponding to 4,750 and 304,400 pixels with dimensions of approximately 5.2 mm/px and 0.65 mm/px. Physically, the data resolution indicates the required spatial

resolution of the dataset representation necessary to accurately capture the input geometry and output field features. Lower resolutions can result in the loss of finer geometry and field features, whereas higher resolutions can overrefine the input and output fields, adding redundant information. Figures 14a to 14c show the R^2 results for each field and data resolution. As the data resolution increases the computational cost of model training increases significantly and does not improve model performance. From Figures 14a to 14c resolutions of 19,000 or 76,000 pixels, corresponding to 2.6 mm/px and 1.3 mm/px, show the best performance across all fields. Lower resolutions result in poor model performance compared to full-resolution data due to the reduction of fidelity in the input and output fields, leading to unpredictable input-to-output behavior. Particularly for the circular geometry of the pin-fins and central wall, at low structured resolutions, the geometry becomes significantly distorted. At the other extreme, higher resolutions require many more pixel values to be predicted, increasing the difficulty of the regression task and reducing model performance.

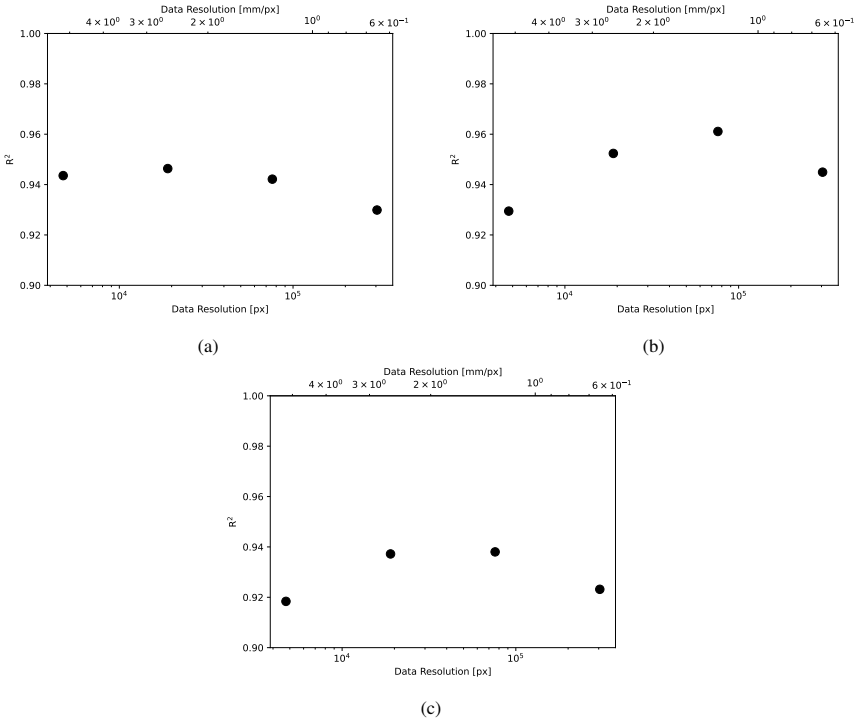


Figure 14: Model data resolution characterization R^2 results for: (a) DeepEDH-pressure, (b) DeepEDH-velocity, and (c) DeepEDH-temperature models. The bottom axis shows results for the total number of pixels, while the top shows the corresponding pixel resolutions.

5.3. Overall DeepEDH model performance

Through the DeepEDH model characterization, we determined the optimal code dimension, dataset size, and data resolution for the surrogate models, which allowed for good model performance while limiting computational cost. After selecting a code dimension of 24 pixels and a resolution of 19,000 pixels based on the characterization results presented in Section 5.2, a new hyperparameter and architecture optimization effort was conducted to determine the final architectures. We employed Bayesian optimization (BO) and Hyperband (HB) techniques using the BOHB algorithm [52] to determine the final network architectures, which are detailed in Table 1. An overview of the characterization and optimization process is illustrated in Figure 15 .

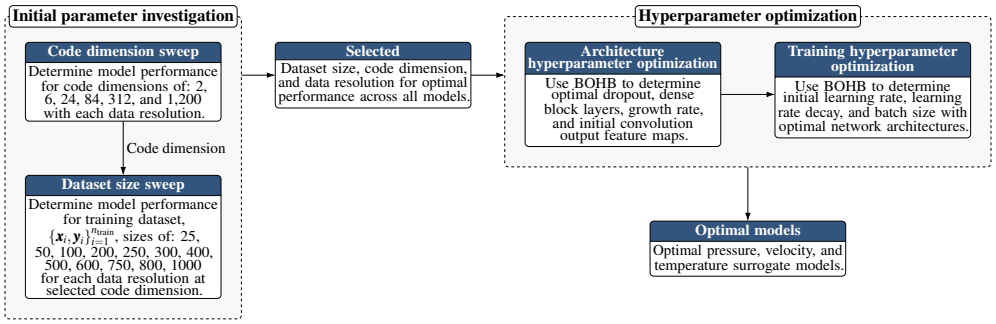


Figure 15: Overview of DeepEDH surrogate model characterization and optimization process.

We evaluated our proposed DeepEDH surrogate models in comparison to U-Net [28] and DenseED [40] models. For this comparison, we maintained the same training dataset size of 80%, a code dimension of 24 pixels for both DenseED and DeepEDH, and data resolution of 19,000. These comparison results are included in Table 3.

Table 3: DeepEDH model performance with various features compared to U-Net [28] and DenseED [40]. Each column considers an added feature, where features in previous columns are also included. For example the *DeepEDH Optimized* column includes output geometry masking for flow fields, multi-stage temperature architecture, and optimized hyperparameters.

Model	Metric	Architecture					
		U-Net [28]	DenseED [40]	DeepEDH			
				Base	Output geometry mask	Multi-Stage	Hyperparameter Optimized
Pressure	RMSE [Pa]	4.9571	5.6956	3.6105	3.5472	-	3.3031
	R^2	0.8578	0.8123	0.9464	0.9465	-	0.9550
	SCC	0.9929	0.9855	0.9941	0.9951	-	0.9958
Velocity	RMSE [$m\,s^{-1}$]	0.01932	0.01443	0.007154	0.007115	-	0.006054
	R^2	0.6302	0.7938	0.9524	0.9524	-	0.9657
	SCC	0.8591	0.9250	0.9807	0.9808	-	0.9845
Temperature	RMSE [K]	1.8649	1.4476	0.8922	-	0.7127	0.6889
	R^2	0.5701	0.7409	0.9015	-	0.9372	0.9413
	SCC	0.9185	0.9506	0.9823	-	0.9888	0.9891

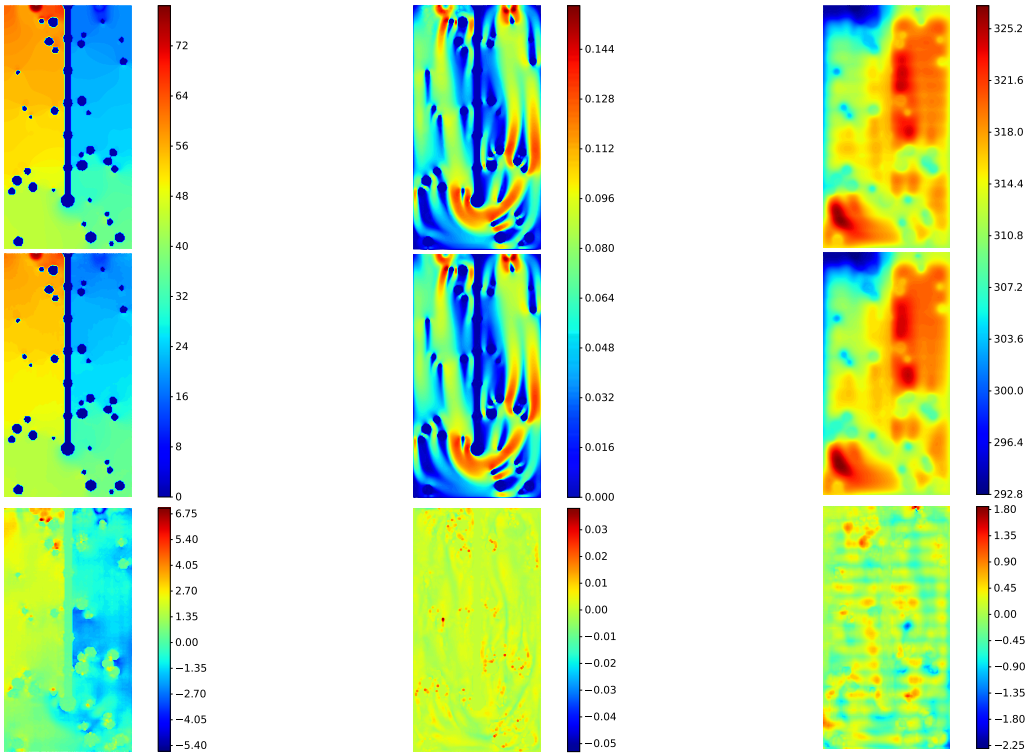
The performance comparison presented in Table 3 demonstrates that DeepEDH outperforms U-Net and DenseED for all models. Compared to DenseED, DeepEDH results in 18%, 22%, and 27% improvements for R^2 of pressure, velocity, and temperature predictions respectively. Similarly, R^2 improvements of 11%, 53%, and 65% for pressure, velocity, and temperature predictions were achieved compared to U-Net. The largest improvements are for the temperature predictions due to the coupled model architecture. Model parameters and training time for each network are summarized in Table 4. The training times are based on training completed using a Nvidia GeForce RTX 3080 GPU with 10 GB of GPU memory and 64 GB of RAM. From Table 4, training time of DeepEDH is approximately 26% less than U-Net and requires 91% fewer parameters. Compared to DenseED, DeepEDH uses individual networks for each model and requires sequential training for the temperature model, resulting in roughly doubled training time and tripled parameter count.

Table 4: Model parameters and training time for U-Net [28], DenseED [40], and DeepEDH.

Architecture	Parameters	Training Time [s]
U-Net [28]	31,037,187	4,872
DenseED [40]	593,489	1,886
DeepEDH	2,684,391	3,596

Figures 16a to 16c show the actual, predicted, and error fields for the best performing DeepEDH models, while Table 5 shows the tabulated performance metrics. Qualitatively the results shown in Figures 16a to 16c show that the model

is able to capture the general trends of the field, while the metrics presented in Table 3 show that the model is able to capture the field with high accuracy and low error.



(a) (b) (c)

Figure 16: Optimized model predictions: (a) DeepEDH-pressure [Pa], (b) DeepEDH-velocity [$\mathrm{m\,s^{-1}}$], and (c) DeepEDH-temperature [K] fields. Within each subfigure, the first row shows the target fields, while the second row shows the model predictions, and the third row shows the error.

Table 5: Best DeepEDH model performance metrics.

Model	R^2	RMSE	SCC
DeepEDH-pressure	0.9550	3.3073 Pa	0.9958
DeepEDH-velocity	0.9657	0.006071 $\mathrm{m\,s^{-1}}$	0.9845
DeepEDH-temperature	0.9413	0.6889 K	0.9892

5.4. DeepEDH-temperature model performance at varying heat fluxes

This section quantifies the impact of the heat flux boundary condition on the performance of the DeepEDH-temperature model. We varied the heat flux by scaling the input heat flux field by factors of 1/10, 1/5, 2/5, 2, 5, and 10. The accumulated heat was calculated using Equation (21) for each heat flux scaling factor. DeepEDH-temperature performance is summarized in Figure 17, with tabulated results included in the supplementary materials.

$$Q_{\text{accumulated}} = \int_0^{t_{\text{final}}} \left[\int_{\partial\Omega} q_0(x, y, t) dS \right] dt - \int_0^{t_{\text{final}}} \dot{m} C_p (T_{\text{outlet}}(t) - T_{\text{inlet}}(t)) dt \tag{21}$$

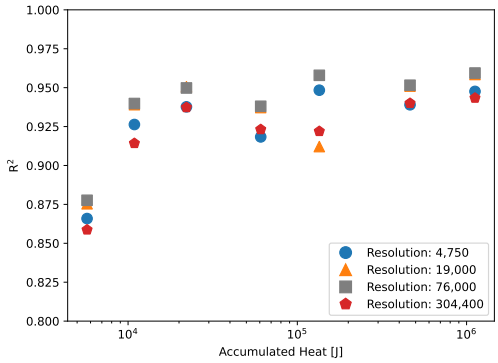


Figure 17: DeepEDH-temperature model R^2 performance for varying heat flux boundary conditions.

Figure 17 shows that the DeepEDH-temperature model performance improves and then plateaus as heat flux increases. At lower heat fluxes, the accumulated heat is limited, resulting in a temperature profile with lower variation. For example, consider Figures 18a to 18e that show the actual, predicted, and error fields for a sample point in the test dataset. Figures 18a to 18e show the limited variation of the temperature field at low heat fluxes, while at higher values the effect of geometry is more clear. It is important to note the difference in scale for the error results in Figures 18a to 18e. Physically, this indicates that the cold plate geometry has a limited impact on the temperature field at lower heat fluxes; in contrast, the temperature field is more sensitive to the cold plate geometry at higher heat fluxes. Interpreting these results allows for the identification of the level of heating required for changes in the cold plate geometry to significantly impact the temperature field and, hence, the cold plate's thermal performance. With heat fluxes below this limit, changes in the cold plate geometry will have a more limited impact.

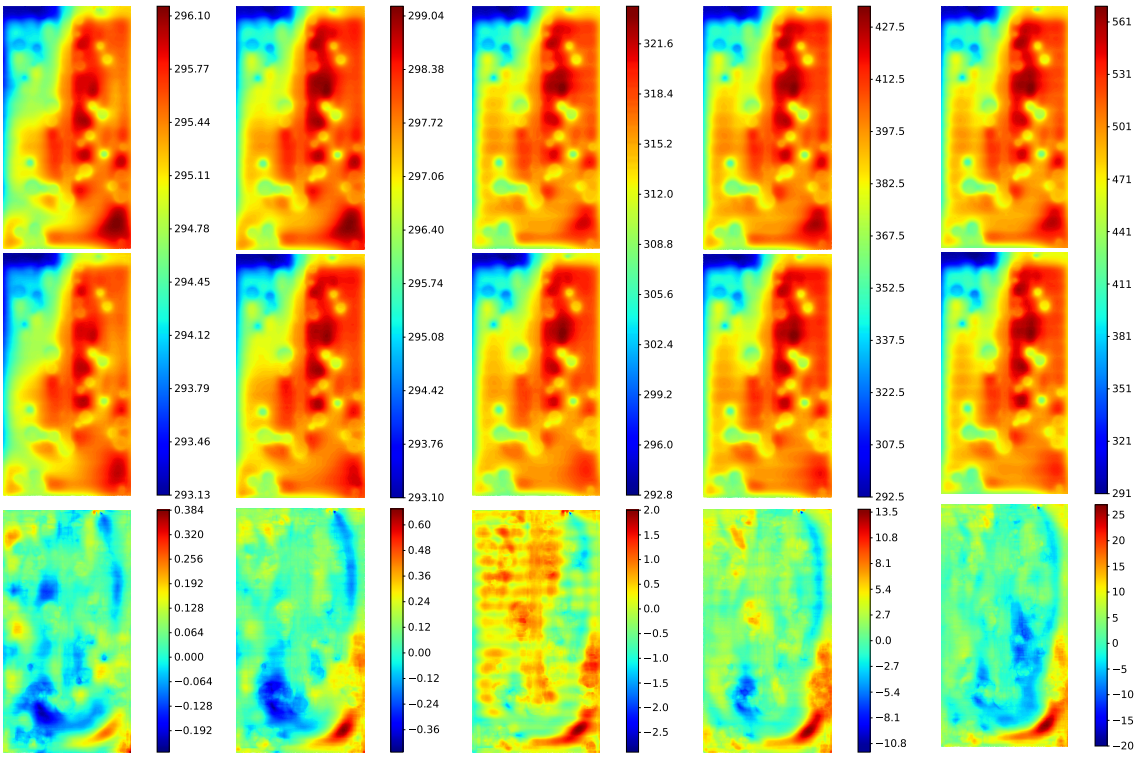


Figure 18: Temperature [K] field predictions with heat flux factors of: (a) 1/10, (b) 1/5, (c) 1, (d) 5, and (e) 10. Within each subfigure, the first row shows the target fields, while the second row shows the model predictions, and the third row shows the error. Note the difference in scale for the error results.

5.5. *Application feasibility*

The results presented in this work demonstrate the effectiveness of the DeepEDH surrogate modeling methodology for CHT analyses. This section examines the feasibility of applying these models to common surrogate modeling tasks, including design optimization, real-time control systems, and real-time digital twin applications. For design optimization, thousands of objective evaluations are required to evaluate design candidates during the optimization process. By using DeepEDH models to replace the FEM numerical models, the optimization computational time can be reduced by approximately 97% as summarized in Table 6 . For real-time control or digital twin applications, the computational resources and execution time required to run the FEM CHT simulations in real time are often infeasible. DeepEDH requires limited computational resources and can be executed in less than 1 s, as shown in Table 6 . Compared to the FEM models, the storage and memory requirements for the DeepEDH models are also significantly reduced by approximately 98% and 92%, respectively. Overall, these results demonstrate the feasibility of using DeepEDH surrogate models for these typical surrogate applications.

Table 6: Application feasibility summary based on the models used in the work, comparing computational time and memory requirements. The computational time estimates are based on executing 50 FEM simulations in parallel. Memory estimates are based on disk space, COMSOL memory, and PyTorch memory usage.

	Finite element model	DeepEDH model
Single model evaluation [s]	1,250	0.5
Dataset generation (1500 samples) [s]	-	37,500
DeepEDH training [s]	-	3,600
Optimization (1000 evaluations) [s]	1,250,000	500
Complete optimization [s]	1,250,000 (≈14 days)	41,600 (≈0.5 days)
Model storage [GB]	2	0.05
Execution memory [GB]	48	4

The values presented in Table 6 are approximate based on the FEM simulations, DeepEDH training, and DeepEDH execution times. Further, the times included in Table 6 are based on executing 50 numerical models in parallel, as completed in this work. The database generation for surrogate model training can be executed in parallel as the simulations are independent of each other. In contrast, for optimization, future simulations depend on the previous simulations' results and must be executed sequentially.

6. Summary and conclusions

In this work, we developed a data-driven surrogate modeling methodology for conjugate heat transfer (CHT) analyses, featuring a novel image-to-image regression convolutional neural network architecture called DeepEDH. This architecture is a novel modular deep encoder-decoder hierarchical convolutional neural network that uses dense blocks at each layer, along with skip connections between the encoding and decoding sections. We introduced output geometry masks, multi-stage temperature architectures, and separate models for each field to enhance the surrogate model performance for CHT analyses.

We demonstrated the effectiveness of the proposed DeepEDH methodology modeling a liquid-cooled pin-fin cold plate for a battery thermal management system application. The field-specific DeepEDH surrogate models, DeepEDH-pressure, DeepEDH-velocity, and DeepEDH-temperature, were trained and tested using a dataset of 1,500 transient CHT finite element method simulations with varying pin-fin geometries. These numerical results were processed from unstructured meshes into image-like data with resolutions ranging from 50×95 to 400×761 pixels. The DeepEDH model performances were characterized for varying code dimensions, dataset sizes, and data resolutions. The results highlighted the significant impact of code dimension, with larger code dimensions resulting in poor performance due to the smoothness of the output fields. As the code dimension was reduced, the models' performances improved significantly before plateauing at a code dimension of 4×6 (24) pixels. Similarly, increasing the dataset size also improved model performances, but with diminishing returns and performance plateauing at approximately 500 simulations. Data resolution had a limited impact on the model performances compared to the code dimension and dataset size. Regardless, a compressed resolution of 100×190 (19,000) or 200×380 (76,000) pixels, corresponding to physical sizes

of 2.6mm/px and 1.3 mm/px, had the best performance across all models. Optimal network hyperparameters and architectures were determined through Bayesian optimization (BO) and Hyperband (HB) using the BOHB algorithm.

Our proposed DeepEDH convolutional neural network was demonstrated to be an effective surrogate modeling methodology for CHT analyses, outperforming U-Net and DenseED for all pressure, velocity, and temperature field predictions. The methodology and architecture components introduced in this work, including the output geometry mask, multi-stage temperature architecture, and hyperparameter optimization, all contributed to the improved performance of the DeepEDH models. Finally, we quantified the impact of the heat flux boundary condition on the temperature model performance. Decreased temperature field prediction performance for low heat flux values suggests that the temperature field is less sensitive to cold plate geometry under such conditions. The coupling methodology introduced here for the temperature model yielded excellent results, improving performance by up to 65% compared to other architectures, offering the potential for application in other multiphysics surrogate modeling tasks.

The methodology presented here could enable advancements in the design of systems governed by CHT, with applications in the aerospace, automotive, and electronics industries. The most significant challenges facing surrogate modeling for CHT problems are the computational cost of generating the dataset, the complexity of the models required to capture the relevant physics, and obtaining accurate full-field predictions. The proposed DeepEDH methodology addresses these challenges by reducing the required dataset size and improving model performance and accuracy of the full-field predictions. Future works can use the DeepEDH methodology and the characterization and hyperparameter optimization techniques presented here to develop optimal surrogate models for other CHT systems. The optimal hyperparameters and architectures determined in this work can be used as a starting point for future surrogate modeling tasks, reducing the computational cost of developing new models. Finally, the modular blocks and hierarchical architecture of the DeepEDH architecture can inform other neural network architecture designs, with the potential to extend to other systems governed by coupled physics, including electro-thermal or electro-magnetic surrogate modeling tasks.

Future directions for this work include expanding DeepEDH models to predict field results at all time steps and different cross-sections of three-dimensional geometries. Examining different system conditions, such as turbulent flow or conjugate radiative heat transfer, could demonstrate the DeepEDH models' effectiveness for different applications. In addition, the introduction of more advanced neural network architecture features and techniques such as physics-based constraints, cross-domain generalization, or adaptive loss functions could further improve model performance and reduce the required dataset size.

Data availability

The code developed and used for this study is available in the GitHub repository at:
takiahebbspicken/conjugate-heat-transfer-surrogate-modeling

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), MITACS, and the Ford Motor Company of Canada Ltd. Simulations were facilitated by CMC Microsystems software licences and performed on the SciNet high performance computing clusters supported by the Canada Foundation for Innovation.

References

[1] B. John, P. Senthilkumar, and S. Sadasivan, "Applied and theoretical aspects of conjugate heat transfer analysis: A review," *Archives of Computational Methods in Engineering*, vol. 26, no. 2, p. 475–489, 2019.

[2] T. Ebbs-Picken, C. M. D. Silva, and C. H. Amon, "Design optimization methodologies applied to battery thermal management systems: A review," *Journal of Energy Storage*, vol. 67, p. 107460, 2023.

[3] E. Hachem, H. Ghraieb, J. Viquerat, A. Larcher, and P. Meliga, "Deep reinforcement learning for the control of conjugate heat transfer," *Journal of Computational Physics*, vol. 436, p. 110317, 2021.

[4] F. Naseri, S. Gil, C. Barbu, E. Cetkin, G. Yarimca, A. Jensen, P. Larsen, and C. Gomes, "Digital twin of electric vehicle battery systems: Comprehensive review of the use cases, requirements, and platforms," *Renewable and Sustainable Energy Reviews*, vol. 179, p. 113280, 2023.

[5] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, p. 686–707, 2019.

[6] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems," *Journal of Computational Physics*, vol. 401, p. 109020, 2020.

[7] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations," *Journal of Computational Physics*, vol. 426, p. 109951, 2021.

[8] L. D. Natale, B. Svetozarevic, P. Heer, and C. Jones, "Physically consistent neural networks for building thermal modeling: Theory and analysis," *Applied Energy*, vol. 325, p. 119806, 2022.

[9] K. Yu, X. Yang, and Z. Yue, "Aerodynamic and heat transfer design optimization of internally cooling turbine blade based different surrogate models," *Structural and Multidisciplinary Optimization*, vol. 44, no. 1, p. 75–83, 2011.

[10] V. Maakala, M. Järvinen, and V. Vuorinen, "Optimizing the heat transfer performance of the recovery boiler superheaters using simulated annealing, surrogate modeling, and computational fluid dynamics," *Energy*, vol. 160, p. 361–377, 2018.

[11] H. Ren, D.-c. Gao, Z. Ma, S. Zhang, and Y. Sun, "Data-driven surrogate optimization for deploying heterogeneous multi-energy storage to improve demand response performance at building cluster level," *Applied Energy*, vol. 356, p. 122312, 2024.

[12] M. Fiore, L. Koloszar, M. A. Mendez, M. Duponcheel, and Y. Bartosiewicz, "Turbulent heat flux modelling in forced convection flows using artificial neural networks," *Nuclear Engineering and Design*, vol. 399, p. 112005, 2022.

[13] C. C. a. António and C. Afonso, "Air temperature fields inside refrigeration cabins: A comparison of results from cfd and ann modelling," *Applied Thermal Engineering*, vol. 31, no. 6–7, p. 1244–1251, 2011.

[14] A. Ozsunar, E. Arcaklıoglu, and F. N. Dur, "The prediction of maximum temperature for single chips' cooling using artificial neural networks," *Heat and Mass Transfer*, vol. 45, no. 4, p. 443–450, 2009.

[15] A. Kargar, B. Ghasemi, and S. M. Aminossadati, "An artificial neural network approach to cooling analysis of electronic components in enclosures filled with nanofluids," *Journal of Electronic Packaging*, vol. 133, no. 1, p. 011010, 2011.

[16] A. Ben-Nakhi, M. A. Mahmoud, and A. M. Mahmoud, "Inter-model comparison of cfd and neural network analysis of natural convection heat transfer in a partitioned enclosure," *Applied Mathematical Modelling*, vol. 32, no. 9, p. 1834–1847, 2008.

[17] Y. Varol, E. Avci, A. Koca, and H. F. Oztop, "Prediction of flow fields and temperature distributions due to natural convection in a triangular enclosure using adaptive-network-based fuzzy inference system (anfis) and artificial neural network (ann)," *International Communications in Heat and Mass Transfer*, vol. 34, no. 7, p. 887–896, 2007.

[18] M. A. Mahmoud and A. E. Ben-Nakhi, "Neural networks analysis of free laminar convection heat transfer in a partitioned enclosure," *Communications in Nonlinear Science and Numerical Simulation*, vol. 12, no. 7, p. 1265–1276, 2007.

[19] Q. Wang, W. Zhou, L. Yang, and K. Huang, "Comparison between conventional and deep learning-based surrogate models in predicting convective heat transfer performance of u-bend channels," *Energy and AI*, vol. 8, p. 100140, 2022.

[20] L. Yang, Z. Min, T. Yue, Y. Rao, and M. K. Chyu, "High resolution cooling effectiveness reconstruction of transpiration cooling using convolution modeling method," *International Journal of Heat and Mass Transfer*, vol. 133, p. 1134–1144, 2019.

[21] X. Jin, P. Cheng, W.-L. Chen, and H. Li, "Prediction model of velocity field around circular cylinder over various reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," *Physics of Fluids*, vol. 30, no. 4, p. 047105, 2018.

[22] N. He, Q. Wang, Z. Lu, Y. Chai, and F. Yang, "Early prediction of battery lifetime based on graphical features and convolutional neural networks," *Applied Energy*, vol. 353, p. 122048, 2024.

[23] V. Tikka, J. Haapaniemi, O. Räisänen, and S. Honkapuro, "Convolutional neural networks in estimating the spatial distribution of electric vehicles to support electricity grid planning," *Applied Energy*, vol. 328, p. 120124, 2022.

[24] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 481–490, 2016.

[25] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik, "Prediction of aerodynamic flow fields using convolutional neural networks," *Computational Mechanics*, vol. 64, no. 2, p. 525–545, 2019.

[26] W. Zhou, X. Li, Z. Qi, H. Zhao, and J. Yi, "A shale gas production prediction model based on masked convolutional neural network," *Applied Energy*, vol. 353, p. 122092, 2024.

[27] J.-Z. Peng, X. Liu, Z.-D. Xia, N. Aubry, Z. Chen, and W.-T. Wu, "Data-driven modeling of geometry-adaptive steady heat convection based on convolutional neural networks," *Fluids*, vol. 6, no. 12, p. 436, 2021.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv*, 2015.

[29] Y. Hua, C.-H. Yu, J.-Z. Peng, W.-T. Wu, Y. He, and Z.-F. Zhou, "Thermal performance estimation of nanofluid-filled finned absorber tube using deep convolutional neural network," *Applied Sciences*, vol. 12, no. 21, p. 10883, 2022.

[30] H. Ma, Y.-x. Zhang, O. J. Haidn, N. Thuerey, and X.-y. Hu, "Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks," *Acta Astronautica*, vol. 175, p. 11–18, 2020.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 770–778, 2016.

[32] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[33] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *arXiv*, 2016.

[34] Y. Hua, C.-H. Yu, Q. Zhao, M.-G. Li, W.-T. Wu, and P. Wu, "Surrogate modeling of heat transfers of nanofluids in absorbent tubes with fins based on deep convolutional neural network," *International Journal of Heat and Mass Transfer*, vol. 202, p. 123736, 2023.

[35] M. Tang, Y. Liu, and L. J. Durlofsky, "A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems," *Journal of Computational Physics*, vol. 413, p. 109456, 2020.

[36] *History Matching Complex 3D Systems Using Deep-Learning-Based Surrogate Flow Modeling and CNN-PCA Geological Parameterization*, vol. Day 1 Tue, October 26, 2021 of *SPE Reservoir Simulation Conference*, 10 2021.

[37] M. Tang, Y. Liu, and L. J. Durlofsky, "Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3d subsurface flow," *Computer Methods in Applied Mechanics and Engineering*, vol. 376, p. 113636, 2021.

[38] S. Jiang and L. J. Durlofsky, "Use of multifidelity training data and transfer learning for efficient construction of subsurface flow surrogate models," *Journal of Computational Physics*, vol. 474, p. 111800, 2023.

[39] G. Wen, M. Tang, and S. M. Benson, "Towards a predictor for co2 plume migration using deep neural networks," *International Journal of Greenhouse Gas Control*, vol. 105, p. 103223, 2021.

[40] Y. Zhu and N. Zabaras, "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification," *Journal of Computational Physics*, vol. 366, p. 415–447, 2018.

[41] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[42] S. Mo, Y. Zhu, N. Zabaras, X. Shi, and J. Wu, "Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media," *Water Resources Research*, vol. 55, no. 1, p. 703–728, 2019.

[43] D. A. Romero, S. Hasanpoor, E. G. A. Antonini, and C. H. Amon, "Predicting wind farm wake losses with deep convolutional hierarchical encoder–decoder neural networks," *APL Machine Learning*, vol. 2, no. 1, p. 016111, 2024.

[44] T. Ebbs-Picken, C. M. D. Silva, and C. H. Amon, "Multi-objective design optimization of pin-fin cold plates for electric vehicle battery packs using convolutional neural networks and genetic algorithms," in *2024 23nd IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pp. 1–10, 2024, to appear.

[45] M. Al-Zareer, C. D. Silva, and C. H. Amon, "Predicting anisotropic thermophysical properties and spatially distributed heat generation rates in pouch lithium-ion batteries," *Journal of Power Sources*, vol. 510, p. 230362, 2021.

[46] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, p. 107–116, 1998.

[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv*, 2015.

[48] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010.

[49] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 431–440, 2015.

[50] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, p. 2481–2495, 2015.

[51] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 936–944, 2017.

[52] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *arXiv*, 2018.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[54] M. Al-Zareer, T. Ebbs-Picken, A. Michalak, C. Escobar, C. M. D. Silva, T. Davis, I. Osio, and C. H. Amon, "Heat generation rates and anisotropic thermophysical properties of cylindrical lithium-ion battery cells with different terminal mounting configurations," *Applied Thermal Engineering*, vol. 223, p. 119990, 2023.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.

[56] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *arXiv*, 2017.